Combining heterogeneous sources of data for the reverse-engineering of gene regulatory networks

A thesis submitted for the degree of

Doctor of Philosophy

Emma Steele

School of Information Systems, Computing and Mathematics

Brunel University

January 2010

Abstract

Gene Regulatory Networks (GRNs) represent how genes interact in various cellular processes by describing how the expression level, or activity, of genes can affect the expression of the other genes. Reverse-engineering GRN models can help biologists understand and gain insight into genetic conditions and diseases. Recently, the increasingly widespread use of DNA microarrays, a high-throughput technology that allows the expression of thousands of genes to be measured simultaneously in biological experiments, has led to many datasets of gene expression measurements becoming publicly available and a subsequent explosion of research in the reverse-engineering of GRN models. However, microarray technology has a number of limitations as a data source for the modelling of GRNs, due to concerns over its reliability and the reproducibility of experimental results.

The underlying theme of the research presented in this thesis is the incorporation of multiple sources and different types of data into techniques for reverse-engineering or learning GRNs from data. By drawing on many data sources, the resulting network models should be more robust, accurate and reliable than models that have been learnt using a single data source. This is achieved by focusing on two main strands of research. First, the thesis presents some of the earliest work in the incorporation of prior knowledge that has been generated from a large body of scientific papers, for Bayesian network based GRN models. Second, novel methods for the use of multiple microarray datasets to produce Bayesian network based GRN models are introduced. Empirical evaluations are used to show that the incorporation of literature-based prior knowledge and combining multiple microarray datasets can provide an improvement, when compared to the use of a single microarray dataset, for the reverse-engineering of Bayesian network based GRN models.

Acknowledgements

Firstly I would like to thank my supervisor Allan Tucker. Without his support, encouragement, guidance and understanding throughout my PhD, this thesis would have never been completed, and certainly not within the magic three years!

I would also thank my second supervisor Xiaohui Liu for all of his invaluable advice and wise words.

I would like to acknowledge my colleagues (past and present) in the Centre for Intelligent Data Analysis at Brunel for interesting discussions that have contributed towards my research, in particular Michael Hirsch, Veronica Vinciotti and Stephen Swift.

I would also like to thank members of the Biosemantics Association in the Netherlands for all their help and support during my PhD. They have been the perfect partners in collaboration! In particular Peter-Bram 't Hoen, Matthew Hestand (Leiden University) and Martijn Schuemie (Erasmus University Rotterdam), who have spent a lot of their time assisting me with this research.

To my office buddies, past and present, for keeping the IDA lab fun and a pleasure to work in: Kyri, Karl, Dan, Ana, Natalie, Jose, Jian, Yahya, Christine, Nota and others.

A big thankyou to my family for their encouragement throughout my PhD but also for not asking how the thesis was going!

Last but definitely not least, thankyou to my husband Andy for all his love and support, and especially for reading this thesis and allowing me to monopolise his brand-new computer without too much complaint!

And a last minute addition to these acknowledgements: my daughter Isla who arrived after this thesis was submitted, but before the viva. Thankyou for allowing me the time to finish this PhD!

Publications

The following publications have resulted from the research presented in this thesis:

1. Peeling, E., Tucker, A. & 't Hoen, P. (2007). Discovery of local regulatory structure from microarray gene expression data using Bayesian networks. In *Proceedings of the annual workshop on Intelligent Data Analysis in bioMedicine And Pharmacology (IDAMAP).*

2. Peeling, E. & Tucker, A. (2007). Consensus gene regulatory networks: combining multiple microarray gene expression datasets. In *AIP Conference Proceedings vol. 940, The 3rd International Symposium on Computational Life Sciences (COMPLIFE 2007).*

3. Steele, E. & Tucker, A. (2008). Consensus and meta-analysis regulatory networks for combining multiple microarray gene expression datasets. *Journal of Biomedical Informatics*, **41**, 914–926.

4. Steele, E., Tucker, A., 't Hoen, P., & Schuemie, M.J. (2009) Literature-based priors for gene regulatory networks. *Bioinformatics*, **25**, 1768-1774.

5. Steele, E. & Tucker, A. (2009) Selecting and weighting data for building Consensus Gene Regulatory Networks. In *LNCS*

5772, Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII, 190-201.

Please note that Peeling is the author's maiden name.

Item 1 results from preliminary work for this thesis. Items 2 and 3 result from the research described in Chapter 5. Item 4 results from the research presented in Chapter 4. Finally, item 5 is based on Chapter 6.

Contents

1 Introduction 1

1.1 Reverse-engineering gene regulatory network models 2

1.2 Thesis aims 5

1.3 Thesis contributions 6

1.4 Thesis outline 7

2 Gene Regulatory Networks 9

2.1 Regulation of gene expression 9

2.2 Microarray technology 12

2.2.1 The basics: data collection 13

2.2.2 Data processing 16

2.2.2.1 Image processing 16

2.2.2.2 Expression ratios 17

2.2.2.3 Normalisation 18

2.2.3 Analysis of microarray expression data 22

2.2.3.1 The gene expression matrix 22

2.2.3.2 Basic analysis techniques 23

2.2.3.3 Inferring regulatory relationships 26

2.2.4 Using multiple microarray gene expression datasets 30

2.3 Other types of data 34

2.3.1 Protein-protein interaction data 34

2.3.2 Transcription factor binding site data 35

2.3.3 Literature-based knowledge 35

2.3.4 Use of prior knowledge in inferring regulatory relationships 38

2.4 Summary 41

3 Modelling GRNs using Bayesian networks 43
3.1 Bayesian networks 43
3.1.1 Conditional independence 45
3.1.2 Inference 45
3.1.3 Explaining away 48
3.1.4 Parameter learning 49
3.1.5 Equivalence classes 50
3.1.6 Causality 50
3.2 Bayesian networks to model gene regulation 51
3.2.1 Simple regulatory structures 52
3.2.2 Modelling using microarray expression data 52
3.2.3 More complex regulatory interactions 54
3.3 Learning Bayesian network structures 56
3.3.1 Constraint-based learning 56
3.3.2 Score-based learning 57
3.3.3 Comparison of both approaches 58
3.3.4 Confidence levels for network edges 59
3.4 Evaluation of model performance 61
3.4.1 Network structure comparison to current knowledge 62
3.4.2 Prediction of gene expression values 65
3.5 Recent work in using BNs to model GRNs 65

4 Prior knowledge 70
4.1 Introduction 70
4.2 Related work 71
4.3 Informative Bayesian network priors 73
4.3.1 Calculating the prior probability of a network structure . . 73
4.3.2 Weighting the prior 76
4.4 Literature-based Bayesian network priors 77
4.4.1 Literature-based gene concept profiling 77
4.4.2 Edge prior probabilities from literature-based knowledge . 79
4.5 Evaluating the use of literature-based priors 81
4.5.1 Evaluation procedure 81

4.5.2 Application: yeast 83
4.5.2.1 Comparison to documented interactions 84
4.5.2.2 Expression value prediction 84
4.5.3 Application: *E. coli* 88
4.5.3.1 Comparison to documented interactions 88
4.5.3.2 Expression value prediction 89
4.5.4 Application: human 92
4.6 Discussion 96

5 Combining multiple microarray gene expression datasets 99
5.1 Introduction 99
5.2 Pre-learning aggregation 100
5.3 Post-learning aggregation 101
5.3.1 Consensus Bayesian networks 101
5.3.2 Bayesian network meta-analysis 103
5.3.3 Related work 105
5.4 Comparison of pre- and post-learning aggregation methods 106
5.4.1 Comparison procedure 107
5.4.2 Application: synthetic network 108
5.4.3 Application: *E. coli* SOS response network 112
5.4.4 Application: yeast heat stress network 116
5.5 Discussion 120

6 Incorporating network reliability 123
6.1 Introduction 123
6.2 Consensus Bootstrapped Bayesian Networks 125
6.2.1 Algorithm 125
6.2.2 Incorporating weights 128
6.2.3 Comparison to the original Consensus algorithm 133
6.3 Measuring network reliability 133
6.3.1 A prediction-based reliability measure for networks 134
6.3.2 Prediction-based network weighting 136
6.3.3 Prediction-based network selection 136
6.3.4 Related work 137

6.4 Network selection or network weighting: a comparison 138
6.4.1 Datasets and experiment design 138
6.4.2 Application: synthetic networks 141
6.4.2.1 Performance comparison of approaches 141
6.4.2.2 Selection of the consensus threshold 144
6.4.2.3 Method comparison for the network selection approach 147
6.4.3 Application: yeast heat-stress network 148
6.5 Discussion 152

7 Inducing heuristics for the Consensus approach 156
7.1 Introduction 156
7.2 Motivation 157
7.3 Rule learning 159
7.3.1 Inputs and Outputs 160
7.3.2 Algorithm basics 161
7.3.3 Relational rule learning 162
7.3.4 ACE 166
7.4 Experimental results 167
7.4.1 Rule learning inputs 167
7.4.2 Rule learning outputs 170
7.4.3 Discussion 173
7.5 Testing the heuristics on real data examples 175
7.6 Summary 176

8 Conclusions 178
8.1 Thesis contributions 178
8.1.1 Incorporation of prior knowledge 179
8.1.2 Combining multiple microarray datasets 180
8.1.3 Incorporating dataset selection or weighting based on reliability 181
8.2 Limitations 182
8.3 Further work 183
8.3.1 Extension of modelling techniques 183

8.3.2 Optimising the weighting influence of prior knowledge . . . 183
8.3.3 Applicability of Consensus Bayesian networks to diverse microarray studies 184
8.3.4 Combining prior knowledge and multiple microarray datasets 184
8.3.5 New expression data technology 184

A Additional tables and results 186
A.1 Chapter 6 additional tables and results 186
A.2 Chapter 7 additional tables and results 189
A.2.1 Training examples for learning 189
A.2.2 Learnt heuristic rules 192
A.2.2.1 Rules for 3 network inputs 192
A.2.2.2 Rules for 4 network inputs 195
A.2.2.3 Rules for 5 or 6 network inputs 197
A.2.3 Test case results 198
A.2.3.1 Yeast: heat-stress network 198
A.2.3.2 Yeast: cell-cycle network 198
A.2.3.3 *E. coli*: SOS response network 200

Glossary 203

References 208

List of Figures

1.1 A simple gene regulatory network model 3

2.1 The process of gene expression 10
2.2 Regulation of gene expression: a TF binds to a promoter region of DNA, activating the expression of nearby genes 11
2.3 A gene regulatory network. Part (a) shows the network with TFs (denoted by a p following the gene name), influencing target genes. Part (b) shows the network with interacting genes only 12
2.4 Complementary binding of DNA strands. The base components of DNA strands are referred to as C,A,G and T. Binding occurs between the C and G parts and the A and T parts. 13
2.5 The process of a DNA microarray experiment. Adapted from Khan *et al.* (2002) 15
2.6 Distribution of log ratio values for each array (a) pre- and (b) post-scale normalisation. In this case, scale normalisation adjusts the interquartile ranges of intensity log ratios so they are equal by array. 19
2.7 Ratio-Intensity plot 21
2.8 Expression profiles for a single gene, for a set of healthy samples and a set of diseased samples (each point on the x-axis refers to a different array — arrays for healthy samples are in blue and arrays for diseased samples are in red. The arrays are temporally related) 23
2.9 Regulatory relationships shown in expression profiles 28
2.10 A simple Bayesian network over 4 variables. (a) shows the DAG component whilst (b) shows the conditional probability distribution attached to the node Gene4 29

3.1 DAG component of a simple Bayesian network over 4 random discrete-valued gene variables. This example network is adapted from Murphy (2001b). 44
3.2 Conditional independence in Bayesian networks: this figure shows the DAG component of a Bayesian network. The nodes W, X and Y are conditionally independent given the value of Z. 46
3.3 PDAG and an alternative DAG representing the equivalence class and conditional independencies amongst nodes of the Gene2 BN shown in Fig 3.1 51
3.4 (a) shows a standard gene regulatory network, involving a TF parent node to target genes X_i. (b) is a modified version of the same network, but with an additional parent node representing a second regulating gene. 52
3.5 A more complex DAG structure representing a regulatory structure between a subset of yeast genes 53
3.6 A simple DBN 55
3.7 Cyclic behaviour represented in a DBN. This figure is adapted from Kim *et al.* (2003) 56
3.8 A bootstrapped network is actually a matrix of confidence level estimates for each possible edge in the network. (a) shows the network graph representation (b) shows the matrix representation. With larger variable sets, this is an easier way to view the network (c) shows the PDAG extracted from the bootstrapped network, when the confidences are thresholded at 0.45 60
3.9 An example of a ROC curve representing the evaluation of a bootstrapped network 63

4.1 (a) shows the edge prior probabilities for a set of nodes whilst (b) shows an example network structure S_1 76
4.2 For the fitted normal distribution of correlation values for yeast genes (parameters $\hat{\mu} = 0.002$ and $\hat{\sigma} = 0.02$), the p-value for each correlation value, based on the fitted normal distribution is plotted 80

4.3 Comparison of Area Under the ROC Curve (AUC) values for each yeast network generated with a different prior weight from 0 (no prior) to 1 (fully weighted prior). 85

4.4 Comparison of expression value prediction between the yeast networks learnt with prior weight 0 and 0.6 for the TF genes. 86

4.5 Yeast networks, learnt with prior weights 0 (expression data only) and 0.8 respectively. Edges are shown if they have confidence greater than 0.2. Solid edges represent regulatory relationships confirmed in the Yeastract database. Gray edges indicate edges that are not documented 87

4.6 Comparison of AUC values for each E. coli network generated with a different prior weight from 0 (no prior) to 1 (fully weighted prior). 89

4.7 Comparison of expression value prediction between the E. coli networks learnt with prior weight 0 and 1 for the TF genes. 90

4.8 *E. coli* networks, learnt with prior weights 0 (expression data only) and 0.8 respectively. Edges are shown if they have confidence greater than 0.2. Solid edges represent regulatory relationships confirmed in the RegulonDB database. Gray edges indicate edges that are not documented 91

4.9 Each network shows the links to and from the gene CCNH, a central node of the cell-cycle network. Nodes that are outlined in black (not gray) are confirmed targets of CCNH. As the prior weight increases, we can see that the number of confirmed targets also increases 94

4.10 Human network for prior weights from 0 (expression data only) and 0.4. Each edge has a confidence that greater than or equal to 0.2 (edges with lower confidence are not included). Blue edges have a high prior probability (> 0.7) in the literature correlation matrix. Red edges have a confidence greater than 0.2 in the expression data only network. Black edges have support in both the literature correlation matrix and the expression data only network 95

5.1 Synthetic regulatory networks. (a) True network. (b) Synthetic consensus network. Edges are shaded or marked according to robustness (i.e. their consensus threshold c) — bold edges obtain a high consensus threshold ($c \geq 0.75$). Bold and dashed edges have $0.50 \leq c < 0.75$, whereas the dashed (only) edges have $c \leq 0.25$. . . 109
5.2 AUC performance of learnt synthetic networks 110
5.3 *E. coli* SOS response transcriptional module: true network structure 113
5.4 AUC performance of learnt *E. coli* networks 114
5.5 *E. coli* consensus network generated from 2 datasets (0.1 bootstrap threshold) — 1.0 consensus threshold (all edges appear in both datasets) 116
5.6 Yeast heat-stress regulatory network: true network structure . . . 117
5.7 AUC performance of learnt yeast networks 119
5.8 Yeast consensus network generated from all datasets (0.1 bootstrap threshold) — 0.8 consensus threshold 120

6.1 Consensus Bootstrapped Bayesian networks. On the top line are the three input bootstrapped networks, displayed as matrices of edge input confidences between node pairs. The bottom line shows the Consensus Bootstrapped BNs, also displayed as matrices of edge consensus confidences, for three consensus thresholds (from $1/n$ to 1 at steps of $1/n$ where $n = 3$). On the right, the set C of ordered input confidences from the input networks for the edge from node 1 to 3 is shown and the table associates the consensus confidence for each consensus threshold. 129

6.2 Weighted Consensus networks: deciding the consensus confidence for a single edge. The input confidences from five input networks are ordered descendingly, paired with the network weights that range from 0.1 to 0.4. The consensus confidences associated with a range of consensus thresholds from 10% to 100% (0.1 to 1.0) are indicated. For example, the consensus confidence 0.24 is associated with the lowest consensus threshold of 10%, whilst the consensus confidence 0.1 is associated with the highest consensus threshold of 100%. 130
6.3 Correlation between network median predictive accuracy and AUC for a collection of synthetic datasets. The synthetic datasets were generated based on the same true network, using differential equations and with different levels of added noise (see Section 6.4.1 for more details). 135
6.4 Adding Gaussian noise to a clean dataset 139
6.5 Synthetic results: across all network sets, a comparison of the maximum AUC and the length of consensus threshold interval for which this is achieved 142
6.6 Synthetic networks: approaches performance comparison for a selection of individual network sets. The x-axis indicates the consensus threshold (between 0-100%) whilst the y axis indicates the network's AUC value. 145
6.7 Prediction-based selection. (a) shows boxplots of the distribution of x, the highest median prediction accuracy of the networks that have *not* been selected by n the optimum number of selected networks, whilst (b) shows boxplots of the distribution of the lowest median prediction accuracy of the networks that have been selected by n the optimum number of selected networks 149
6.8 Yeast heat stress network: comparison of approaches by AUC . . 150

7.1 Patterns in input network predictive accuracies for synthetic cases 1-6. The *y*-axis represent the network's median predictive accuracy. Black lines indicate the predictive accuracies for input networks. The red line shows a predictive accuracy of 0.33 158
7.2 Learning heuristics for network selection: example output rule . . 161
7.3 General to specific search for rules 163
7.4 Learning heuristics for network selection: example output rule, represented using FOPL 166

A.1 Test case: yeast heat-stress network AUC plot 199
A.2 Test case: yeast cell-cycle network AUC plot 200
A.3 Test case: *E. coli* SOS-response network AUC plot 201

List of Tables

3.1 Conditional probability tables for each node in the DAG shown in Figure 3.1. Note that G1=*Gene*1, G2=*Gene*2, G3=*Gene*3 and G4=*Gene*4 44
3.2 Example CPT for target gene TYE7 (from Figure 3.5), that is controlled by two TFs, RPN4 and HSF1 54
3.3 Comparison between the learnt network and the true network in terms of true and false positives and negatives 62

4.1 Yeast expression value prediction results. This table compares the expression data only (prior weight 0) network with each posterior network (prior weights 0.2-1), through the average TF prediction accuracies and the significance of the CMH-test 86
4.2 *E. coli* expression value prediction results. This table compares the expression data only (prior weight 0) network with each posterior network (prior weights 0.2-1), through the average TF prediction accuracies and the significance of the CMH-test 90

5.1 Summary of synthetic datasets 108
5.2 Summary of *E. coli* datasets 114
5.3 Summary of yeast datasets 118

6.1 Synthetic results: statistical significance performance comparison between approaches. Using a paired *t*-test, this table shows the *p*-value for whether Approach 1 significantly outperforms Approach 2. 144
6.2 Yeast heat stress datasets: prediction-based reliability measures . 151

7.1 Learning heuristics for network selection: inputs to the rule learning algorithm 161
7.2 Learning heuristics for network selection: FOPL inputs to the rule learning algorithm 165
7.3 Learning heuristics: background knowledge features 171
7.4 Learning heuristics: test case outcomes 175

A.1 Collections of input networks for synthetic datasets (generated by differential equations). This table provides details of the Gaussian variance added to each of the datasets for each set of input networks. The 'collective reliability' of a network set is measured by the median of the median predictive accuracies for the networks generated from each dataset. Note that in general, the Gaussian noise applied across the datasets increases as the collective reliability level of the network set decreases. 187
A.2 Synthetic networks: approaches performance comparison summary 188
A.3 Method comparison for prediction-based selection 189
A.4 Learning heuristics: training examples and class labels 191
A.5 Summary of yeast cell-cycle datasets 199

Chapter 1

Introduction

Over the past couple of decades the face of biological research has changed from time-consuming, relatively small-scale experiments (the 'white coat' era) to automated experiments that generate reams and reams of data. Biology-related datasets are now more diverse, originating from different sources (for example, experiments and textual sources) and are massively larger due to an increase in the speed of data capture (through automated experimental procedures and increased paper publication rates) and the size of data storage. Bioinformatics — which focuses on the analysis of biological data using computers — has grown out of the need to effectively analyse such data. Its emphasis is on developing robust and efficient data analysis techniques and algorithms to extract more knowledge from these large and complex sets of experimental data.

A current 'hot topic' in Bioinformatics research is the modelling of Gene Regulatory Networks (GRNs). GRNs describe how genes interact in various cellular processes. A GRN model indicates which genes affect the activity of other genes, for example gene A influences gene B and then in turn gene B may influence gene C. Constructing network models that represent how genes interact can provide insight into genetic diseases. For example, a number of human genetic disorders (e.g. muscular dystrophy) result from the absence or malfunction of certain genes, which can cause a disruption in the usual patterns of gene interactions in some cells. Therefore, building gene network models can help scientists understand and gain insight into genetic conditions, for example by highlighting particular genes of interest for further investigation.

The underlying theme of the research presented in this thesis is the incorporation of multiple sources and different types of data into techniques for reverse-engineering, or learning, GRN models. By drawing on many data sources, the resulting network models should be more robust, accurate and reliable than models that have been learnt using a single data source. In the remainder of this introductory chapter, the motivations, aims and contributions of this thesis are presented fully. An overview of GRNs and the limitations of data sources and current modelling techniques is presented in Section 1.1. Following this, Section 1.2 sets out the thesis aims and research questions. The thesis contributions are listed in Section 1.3. Finally, the structure of the remainder of the thesis is outlined in Section 1.4.

1.1 Reverse-engineering gene regulatory network models

The 'central dogma of molecular biology' relates to the process of gene expression, which is the key mechanism underlying GRNs. Genetic information for cellular organisms is stored in their genome, which contains segments of DNA that encode genes. Gene expression is a process whereby genes are 'activated' and translated to proteins, in order to perform processes and functions in a cell. Since proteins are involved in every cellular process from metabolism to muscle development, it is gene expression that allows all cellular processes to occur. Although the same genomic DNA is present (with some exceptions) in every cell in an organism, genes are expressed differently in different cell types, or under different environmental conditions and for different purposes. For example, a yeast cell in a sugar solution will turn on genes to make proteins that process the sugar to alcohol (Vohradsky, 2001), whilst in animals, genes that encode muscle proteins are expressed only in muscle cells and not in the cells of the brain (Twyman, 2003). It is these differences that make gene expression an important topic of research to biologists.

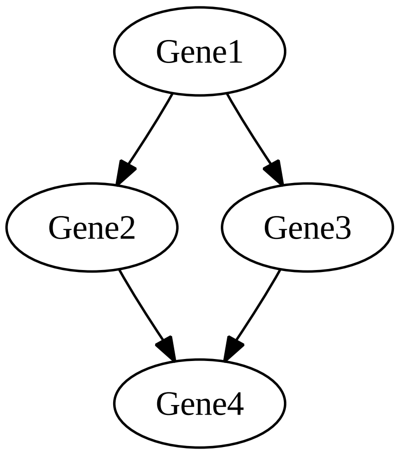


Figure 1.1: A simple gene regulatory network model

A gene becomes expressed when it is activated by a special type of protein called a transcription factor binding to a segment of nearby DNA. Since proteins themselves are created by the gene expression process, this means that the expression of one gene can affect, or regulate, the expression of other genes. A GRN can be formed when we consider how genes interact together in this way. A simple example of a GRN model is shown in Fig 1.1. Here, the expression of $Gene1$ influences the expression of $Gene2$ and $Gene3$ by the production of transcription factor proteins that activate their expression. In turn, the expression of $Gene2$ and $Gene3$ influence the expression of $Gene4$ in the same way.

DNA microarrays are an experimental technique that allow the expression of thousands of genes to be measured simultaneously. They were first used to measure global gene expression, across the yeast genome, in 1997 (DeRisi *et al.*, 1997). Since then, microarray technology has become increasingly popular and is used in almost every biological research group, which has led to many publicly available datasets of gene expression measurements for a range of organisms across various experimental conditions. Since expression measurements across a set of genes can provide an indication of regulatory relationships between genes, the increasing availability of microarray data has led to an explosion of research in the reverse-engineering of GRN models based on microarray-generated data.

Many data analysis techniques have been proposed to gain insight into gene regulatory relationships, based on microarray gene expression data. At the most basic level, clustering techniques allow groups of co-regulated genes to be discovered and this is used sometimes as a basis for learning a GRN model, since genes that exhibit similar or correlated expression patterns are likely to be regulated by a common set of genes. However, basic analysis techniques such as clustering are not able to reveal the more complex structure of the gene regulation process. GRNs can contain nonlinear relationships and combinatorial regulatory control (where a group of genes may interact together to regulate another set of genes). Thus a group of more complex analysis techniques for reverse-engineering GRN models have been applied to and/or developed for the task. This thesis focuses on one technique in particular, Bayesian networks (Pearl, 1991). Bayesian networks have become a popular and successful method for the reverse engineering of GRNs from expression data (Friedman *et al.*, 2000; Pe'er *et al.*, 2006; Segal *et al.*, 2003a) since they are able to represent the network qualitatively (with a network graph) and quantitatively (probability distributions quantify the strength of influences and dependencies between nodes/variables in the network graph) and thus are relatively easy to interpret by non-technical people.

Whilst the microarray provides the most available genome-wide data source on gene expression, it has a number of limitations as a data source for the modelling of GRNs. Microarray data is subject to both natural biological variations across samples (biological noise) and experimental noise, which may be introduced throughout the stages of the experiment (e.g. sample preparation) (Causton *et al.*, 2003). In addition, a key issue with microarray datasets is often referred to as the *curse of dimensionality* (Somorjai *et al.*, 2003). A single microarray dataset usually contains a large number of genes (commonly thousands) but the number of samples is much lower, which can make it very difficult to extract reliable regulatory interactions from a single dataset.

There are also concerns over the reproducibility of results across microarray platforms or laboratories (MAQC consortium, 2006; Tan *et al.*, 2003), which has led to questions over the reliability of microarray gene expression data. Microarray expression datasets often come from different microarray platforms (which measure gene expression using different methods). This can mean that the data can contain different biases and it can be difficult, or sometimes impossible, to compare the datasets since measurement units may vary. Additionally, studies come from different laboratories meaning that data is collected with different measurement biases under different atmospheric conditions.

1.2 Thesis aims

The previous section introduced GRNs and motivated the use of data analysis techniques for reverse-engineering GRN models. It also discussed the reliability issues that surround microarray gene expression data, which is the most available data source on gene expression levels and so is frequently used to build GRN models. The objective of the research presented in this thesis is to build GRN models that have enhanced performance, based on a richer and/or broader collection of data than a single microarray dataset. To achieve this, this thesis presents two main strands of research, which are described next.

First, it is proposed that the drawbacks of using only microarray data to reconstruct GRNs can be alleviated by incorporating other complementary data sources into the modelling process. This thesis investigates whether text-based knowledge from the body of scientific literature, when integrated into the reverse-engineering process as prior knowledge for Bayesian network models, can improve the resulting GRN model over the use of microarray data only. Previous research in the use of prior knowledge from text-based sources in GRN modelling has focused on using online biological databases. These databases rely on the addition of manual annotations and keeping reliable and up-to-date information within them is challenging as the publication rate of scientific papers continues to increase rapidly. The research presented in this thesis aims to improve on this with the use of advanced text-mining techniques to harness prior knowledge directly from the body of scientific literature and integrate this into the model learning process.

The second strand of research aims to take advantage of multiple publicly available microarray gene expression datasets that have been generated in similar biological studies. This thesis addresses the question of whether the use of multiple microarray datasets to reverse-engineer a GRN model can produce an improvement over the use of a single dataset. In addition, a comparison is made of two different approaches for utilising multiple datasets with Bayesian network models: pre- and post-learning aggregation. When learning from multiple datasets, there is a choice for *when* to aggregate knowledge within the datasets. In pre-learning aggregation, data is combined prior to learning, and a model is learnt from the combined dataset. In post-learning aggregation individual models are learnt from each dataset, and these are combined after learning. Whilst pre-learning aggregation is simpler, post-learning aggregation approaches have the advantage that they are more suitable for combining microarray expression datasets generated by different platforms or in different laboratories, since they do not necessarily require normalisation of the datasets, which can be complicated on cross-platform microarray datasets.

Further to this, this thesis also addresses whether taking into account the dataset quality when learning from multiple data sources, through dataset selection or weighting, can improve the final model. Different methods are considered for assessing the quality of datasets and networks. An empirical evaluation investigates whether weighting the influence of each dataset can improve the final GRN model, and compares weighting datasets against simply excluding the least reliable datasets.

1.3 Thesis contributions

The key contributions of this thesis are outlined below:

- **The incorporation of prior knowledge from the whole body of literature for Bayesian network based GRN models.**
  This thesis presents some of the first research in the incorporation of prior knowledge that has been generated from a large body of scientific papers, for Bayesian network based GRN models. The use of advanced text-mining techniques means information contained in a huge number of documents can be represented in a simple format. This thesis presents a method for including this information as a prior probability distribution over candidate Bayesian network GRN models. An empirical evaluation shows that the use of literature-based prior knowledge can improve both the number of true

regulatory interactions present and the predictive performance of the learnt network model, in comparison to a network that has been learnt solely from expression data.

- **Two novel post-learning aggregation approaches for generating Bayesian network based GRN models from multiple microarray gene expression datasets.**
  Post-learning aggregation is a new approach for using multiple microarray datasets to produce a Bayesian network based GRN model. Each of the methods is based on aggregating high-level features of Bayesian network models that have been generated from different microarray gene expression datasets. *Bayesian networks meta-analysis* is based on combining statistical confidences attached to network features whilst *Consensus Bayesian networks* identify consistent network features that exist across all datasets. An empirical evaluation demonstrates that both methods can produce GRN models that improve on models learnt from a single dataset or by the use of a pre-learning aggregation approach.

- **An investigation on the effect of dataset reliability on the resulting GRN models.**
  This thesis presents further development of the Consensus Bayesian networks approach, to incorporate weighting of the input microarray datasets based on their reliability or quality. Empirical evaluation results demonstrate that discarding unreliable datasets provides a more consistent improvement in GRN model performance than using weighting. Additionally, the thesis presents heuristics, induced using machine learning, for selecting which datasets to use based on their reliability measures.

1.4 Thesis outline

The remainder of the thesis is structured as follows:

- Chapter 2 forms the first part of the literature review in this thesis. The biological background (gene expression and regulation, GRNs and microarray technology) is presented, followed by a discussion on the reverse-engineering of GRN models. Further data types (additional to microarray gene expression data) are also introduced.

- Chapter 3 focuses on Bayesian networks and how they can be used to model GRNs. The second part of the literature review is included here in a thorough discussion of state-of-the-art in using Bayesian networks to reverse-engineer GRNs.

- Chapter 4 presents the first key contribution — the use of literature-based prior knowledge to improve Bayesian network based GRN models. This work has been submitted as a journal publication to *Bioinformatics* and is currently in the second cycle of the review process.

- Chapter 5 presents the second key contribution — the use of multiple microarray gene expression datasets to produce Bayesian network based GRN models, using two novel post-learning aggregation approaches. Some of the work in this chapter has been published in (Peeling & Tucker, 2007; Steele & Tucker, 2008).

- Chapter 6 presents the third key contribution — an investigation on the effect of dataset reliability on the resulting GRN models. In this chapter, the post-learning aggregation method of Consensus Bayesian networks is developed further to allow weighting of each dataset and an empirical evaluation compares dataset weighting with dataset selection. Part of this work has been submitted as a conference publication for IDA 2009.

- Chapter 7 extends the work in the previous chapter by presenting the use of machine learning to induce heuristics for selecting which datasets to use in Consensus Bayesian networks, based on their reliability measures.

- Finally, Chapter 8 presents the thesis conclusions, sets out the contributions and discusses potential avenues for further research.

Chapter 2

Gene Regulatory Networks

Gene Regulatory Networks (GRNs) describe how the expression of a gene — its activity level — can regulate the expression of other genes. Building network models that represent how genes interact in this way is a topic of particular current interest in Bioinformatics. This chapter forms the first part of the literature review in this thesis. GRNs are introduced together with the types of data that can be used to reverse-engineer GRN models. Section 2.1 explains gene expression and regulation, which are the underlying processes that form GRNs. DNA microarrays are the main tool currently available for measuring gene expression levels. These are introduced in section 2.2, together with a discussion of microarray data analysis and its drawbacks. In Section 2.3 we discuss other types of data that can provide information on gene expression and regulation. Finally, in Section 2.4 the main points of the chapter are summarised, with particular focus on the motivation for the contributions of the thesis.

2.1 Regulation of gene expression

Genetic information for cellular organisms is stored in their genome, which contains segments of DNA that encode genes (Causton *et al.*, 2003). Gene expression (see Fig. 2.1) is the process by which genes are transcribed, where the gene is copied into what is known as messenger RNA (mRNA), which is essentially a duplicate of the information carried by the gene on the DNA. After transcription, the mRNA is translated to form proteins, which are the main building blocks and functional molecules of a living cell. Proteins are involved in every cellular process from metabolism to muscle development. Therefore it is gene expression that allows all cellular processes to occur. The same genomic DNA is present (with some exceptions) in every cell in an organism. However, all cells are not the same since genes are expressed differently in different cell types, as described earlier in Section 1.1. It is these differences that make gene expression an important topic of research to biologists.

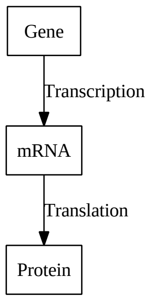


Figure 2.1: The process of gene expression

Regulatory interactions between genes concern the regulation of gene expression. Genes are encoded in DNA strands and each gene is surrounded by DNA sequences that control its expression. The expression of a gene is initiated when transcription factors (TFs) bind to these DNA segments (also known as promoter sites) and activate nearby genes, causing them to be expressed. TFs also control expression by repressing (deactivating) genes in the same way. Figure 2.2 shows a simplified version of the process of gene regulation. TFs are so-called as they are factors that regulate the process of transcription (the first step in gene expression). They themselves are usually proteins, which are produced during gene expression processes. This means that we can form a hierarchy of regulatory interactions, where genes and proteins can be linked (beginning with TFs that are present in the egg at the beginning of development) (Twyman, 2003).

At the most basic level, regulatory interactions occur where TFs *activate* (turn on) or *repress* (turn off) the expression of certain genes, or a subset of genes. More complex interactions involve feedback loops: genes can regulate themselves when a TF controls the expression of the same gene from which it was produced.

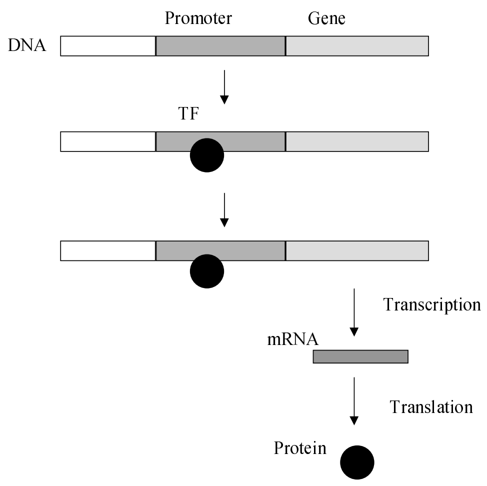


Figure 2.2: Regulation of gene expression: a TF binds to a promoter region of DNA, activating the expression of nearby genes

A Gene Regulatory Network (GRN) can be formed when we consider how genes interact together. Since TFs themselves are gene products (proteins), we can consider a gene to interact with other genes through the gene expression and regulation process. For example, gene Y may be activated by a protein that is produced when gene X is expressed. This can be represented in a network format by gene X influencing gene Y, $X \rightarrow Y$. TFs can control the expression of many genes (their targets), and in turn each gene may be regulated by multiple TFs, sometimes acting in combination or under different conditions. Thus, we can talk about regulatory networks where each gene may interact with both regulators (TFs) and targets. Figure 2.3 shows a GRN (for a subset of yeast genes) in two formats. Figure 2.3a shows the network with TFs influencing target genes. The TFs, which are proteins produced by an expressed gene, are denoted by a rectangle node and the gene name[1] followed by a p. Figure 2.3b shows the network with interacting genes only. Here, the TF nodes are removed and merged with the gene nodes. For example, since the TF $HSF1p$ regulates gene $RPN4$, there is an interaction between the genes $HSF1$ and $RPN4$, since a product of $HSF1$ (when it is expressed) regulates $RPN4$. There are also a number of feedback loops,

[1]A gene name is often three characters followed by a number

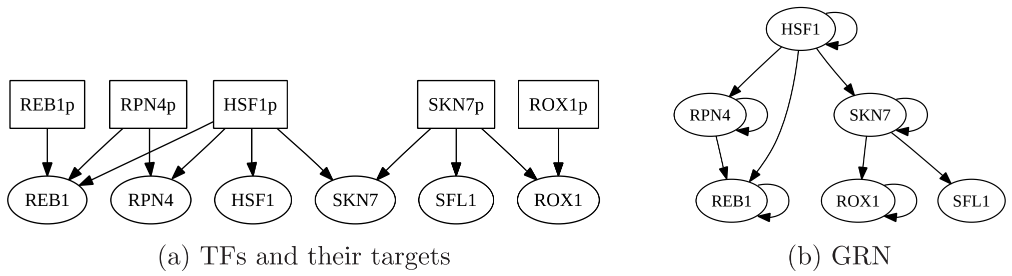


Figure 2.3: A gene regulatory network. Part (a) shows the network with TFs (denoted by a p following the gene name), influencing target genes. Part (b) shows the network with interacting genes only

where the protein produced by an expressed gene can also regulate that gene. For this reason, throughout this thesis regulator genes (those that activate or repress expression) are often referred to as TFs themselves, although technically the TF is the protein produced when the gene is expressed.

Understanding the regulation of gene expression is important to biologists for a number of reasons. For example, a number of human diseases are genetic disorders (e.g. muscular dystrophy) that can result from the absence or malfunction of TFs, which disrupts the regulation of gene expression in some cells (Twyman, 2003). Wet lab experiments to investigate gene regulation interactions can be expensive and time-consuming. Because of this, building models to understand and gain insight into gene regulation, since they can assist in highlighting TFs and genes of interest for further investigation, is an increasingly popular topic of research. These models are usually based on measurements of gene expression (see section 2.2) and other available types of data (see Section 2.3).

2.2 Microarray technology

DNA microarrays are an experimental technique that allow the expression of thousands of genes to be measured simultaneously. They were first used to measure global gene expression, across the yeast genome, in 1997 (DeRisi *et al.*, 1997). Since then, microarray technology has become increasingly popular and now provides a key source of experimental data for modelling gene regulatory interactions from expression levels. In this section we introduce the microarray and discuss a number of issues with the accuracy and reliability of microarray expression data.

2.2.1 The basics: data collection

A DNA microarray is a glass or polymer slide to which DNA molecules are attached. Such a slide may also be referred to as a DNA array, a DNA chip or a gene chip. The attached molecules are usually referred to as spots or features. A single microarray slide may contain tens of thousands of spots. Microarray slides can be produced quickly and efficiently in an automated fashion, for example by the use of inkjet printing.

A DNA microarray can measure a gene's expression level at a particular time by measuring the abundance of mRNA molecules in a cell in a sample taken at that time. Recall from Section 2.1, that when a gene is expressed, it is first transcribed into mRNA, known as a transcript. To understand how a DNA microarray measures the abundance of mRNA, it is first useful to understand that a gene's transcript (the mRNA produced during expression) is a complementary copy of the DNA segment that encodes that gene. Complementary strands of DNA or mRNA tend to hybridise or bind together to form a single, double-stranded molecule (see Fig. 2.4). This means that a gene and its transcript are likely to bind. Note that DNA/mRNA strands that are not fully complementary may also bind — in general, the greater the complementarity, the stronger the binding.

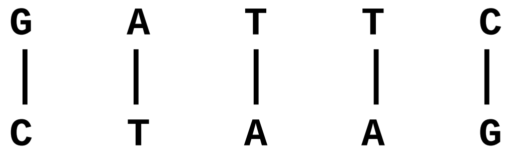


Figure 2.4: Complementary binding of DNA strands. The base components of DNA strands are referred to as C,A,G and T. Binding occurs between the C and G parts and the A and T parts.

In gene expression studies, each spot on the microarray usually encodes a single gene (in practise however, it can be difficult to identify every gene unambiguously due to similarities within gene families). There are different types of microarrays that measure gene expression levels in different ways (often these different types are referred to as different platforms). The most popular and commonly used type is the two-channel hybridised array, which compares the gene expression levels in cells in two different samples that are collected under different conditions. Figure 2.5 shows the experimental process for two channel arrays in diagrammatic form. Usually, one of the samples will be a control sample of cells and the other will measure the gene expression levels in the same cells or cell type under a certain condition. mRNA is extracted from both samples and is labelled differently according to which sample it is taken from. Typically, this is done by applying red dye to one sample and green dye to the other. Then both samples are washed over the slide. Since mRNA binds to its complementary DNA, mRNA molecules in the sample will hybridise to spots on the array that represent their complementary sections of DNA. This means that for genes that are expressed, mRNA will bind to those spots on the array. Note that binding may also take place between mRNA and genes that are not fully complementary.

To measure the abundance of mRNA that has hybridised to each spot on the array, an image of the array is produced using a laser to detect the amount of fluorescence emitted by the dye-labelled mRNA at each spot. For a two-channel array, the image would be scanned twice to measure the amount of fluorescence for each dye label. For example suppose that samples 1 and 2 are labelled with red and green dyes respectively. Then if mRNA from sample 1 is present, the spot will fluoresce red and if mRNA from sample 2 is present it will fluoresce green. If mRNA from both samples is present, the spot will appear yellow. If neither are present then the spot will not fluoresce and will appear black. In this way, relative expression levels for each gene on the array (represented by a spot) can be estimated using the array image. The further image and data processing is covered later in Section 2.2.2.

Other microarray platforms also use the same principle of complementary binding of DNA/mRNA. However the estimation of expression levels may differ. For example, in single channel arrays (e.g. Affymetrix GeneChip), an absolute expression value, rather than a relative value, is recorded for each spot. In this case, only one sample is collected for each array. For these type of experiments,

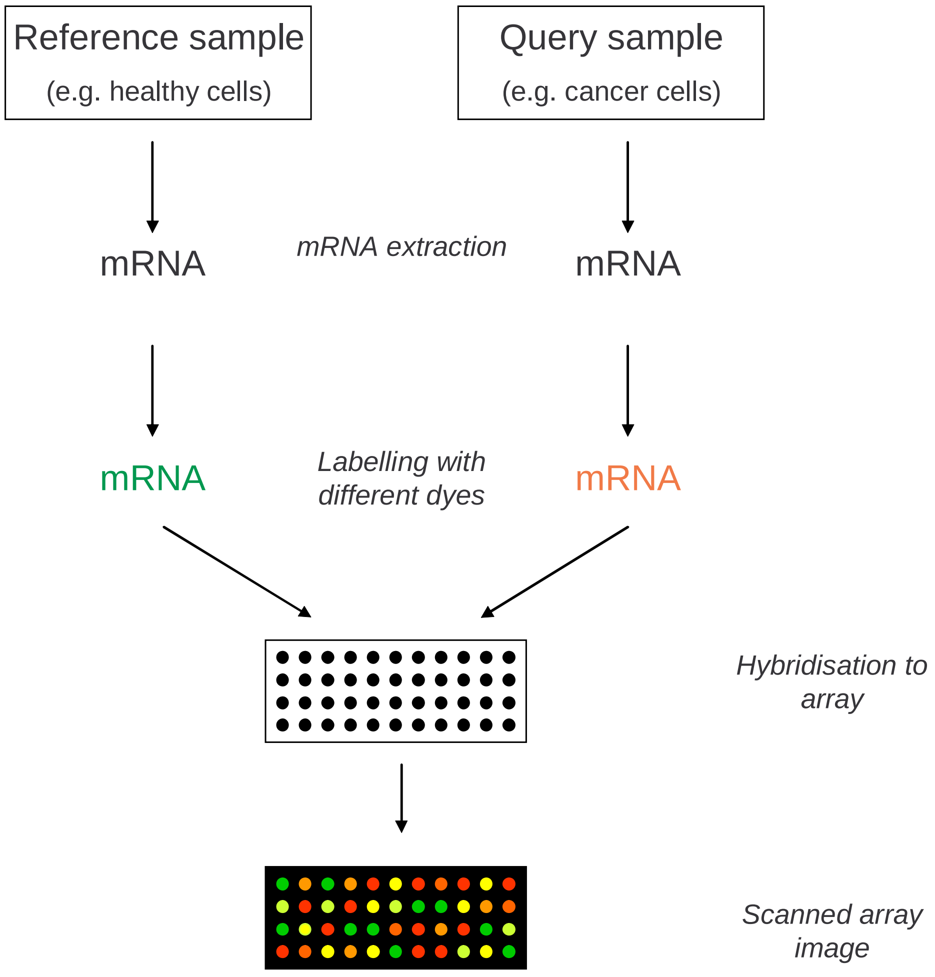


Figure 2.5: The process of a DNA microarray experiment. Adapted from Khan *et al.* (2002)

multiple experiments and arrays are necessary to compare expression levels in different samples under different conditions.

2.2.2 Data processing

Section 2.2.1 explained the first major part of a microarray-based experiment: data collection. In this step, biological samples are prepared and this is followed by the extraction and labelling of mRNA from the samples. Then, the extracted and labelled mRNA is washed over the array and hybridisation takes place. The final step in the data collection is the scanning of the array by a laser to excite fluorescence from the dye labels, producing an image of the microarray. In this section we continue by describing the second part of a microarray-based experiment — the data processing to produce expression level measurements for the genes represented on the array, which is a task with a number of parts. In the first step (see Section 2.2.2.1) the microarray image is processed to produce values that represent the fluorescence intensities of each spot. Then, an expression measurement must be inferred from the measured spot intensity (see Section 2.2.2.2). Finally (see Section 2.2.2.3), it is necessary to normalise these expression measurements within the array, and if it is a set of array experiments, across a set of arrays as well.

2.2.2.1 Image processing

The image processing step extracts fluorescence intensities for each spot on the array, based on a digital image of the scanned array. Most commercially available microarray scanners provide software to do this automatically. However, it is important to understand how the image processing is carried out and its impact on the resulting data, (Causton *et al.*, 2003).

The main task in microarray image processing (Yang *et al.*, 2001) is to identify the spots on the array. This can be difficult due to artefacts or contaminants on the slide (e.g. scratches, dye or dust). However, since the spots are printed onto the array slide in a regular arrangement, a process known as gridding assists in locating each spot. Usually the user is required to specify approximate locations of subgrids on the array which are used to place initial grids over the array image. The placement of the grids is then improved to best represent the spots, often using a centre-of-mass calculation for each spot. Grid placement is very important, since if it is aligned incorrectly, expression levels may be linked to the wrong genes. To avoid this, many arrays contain ‘landing light’ features. These are spots on the array that will always fluoresce strongly, so they are easy to identify and subsequently assist with grid placement. Once the grid is in place, the spot area and the background are determined. Typically, this is done by using a fixed region centred on the centre-of-mass, or by actually identifying the spot boundary. The latter method can provide a better estimation of the fluorescence intensity, but is computationally more difficult.

2.2.2.2 Expression ratios

Once spot locations are determined, expression levels need to be inferred based on the spot fluorescence intensities (Quackenbush, 2001). For each spot a set of summary statistics can be reported, including the mean, median and mode of the pixel intensity distribution as well as the total intensity of the spot. Usually, one of these statistics is used to represent the spot — a popular choice is the background-subtracted median (which is the median of the spot intensity with the median of the background intensity subtracted) or total intensity of the spot.

Recall that a two-channel array will have two samples applied to the slide, labelled with different dyes. In this case the ratio of the intensities for each dye label is calculated. Ratios are useful because they allow us to measure the relative change in a gene between two conditions. Typically in two-channel array experiments, one sample will consist of cells from a reference or control condition and the other sample will contain cells under the condition of interest (the query sample). If the intensity from the reference sample label for the jth gene is R_j, and the intensity from the query sample is Q_j, then the expression ratio M_j is:

$$M_j = \frac{Q_j}{R_j}$$

A ratio is a useful way to measure changes in expression, as those genes that do not have a change in their expression between the two conditions will have a ratio of 1. However, ratios can be problematic for the following reason. If a gene has a two-fold increase in expression in the query sample compared to the reference sample, the expression ratio will be 2. However, if a gene has a two-fold decrease in expression in the query sample compared to the reference sample, the expression ratio will be 0.5. The scale for expression increases is different to that for expression decreases. The most common way of dealing with this is to apply a logarithmic transformation (at base 2) to the ratio, as it creates an even scale for the ratios, as $\log_2(2) = 1$, $\log_2(1) = 0$ and $\log_2(0.5) = -1$. The expression measurements transformed in this way are often referred to as intensity log ratios or expression levels.

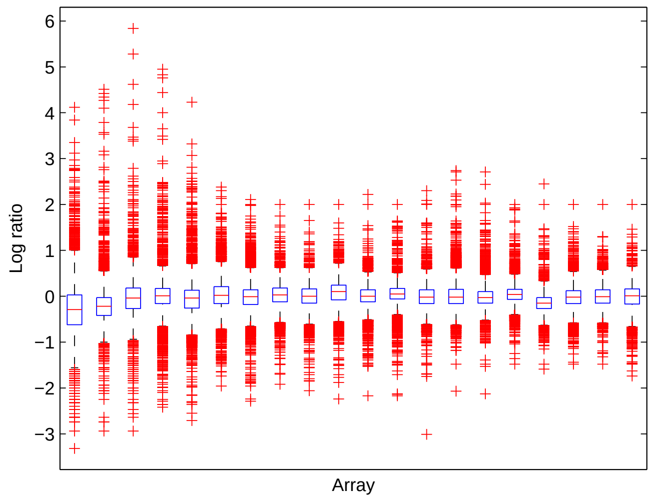
2.2.2.3 Normalisation

Normalisation is the transformation of expression levels (intensity log ratios) from microarray data to adjust for systematic variations (arising from variation in the technology rather than true biological variations) so that measurements from two different samples can be directly compared. There are a number of reasons why data from two arrays may not be comparable. For example, there may be differences in the way samples are labelled, the total amount of mRNA extracted, or changes in the photomultiplier tube settings of the scanner (Smyth & Speed, 2003). Consider Fig. 2.6a which shows unnormalised log ratio data box-plotted for a number of arrays. We see how the interquartile range of values can differ across the arrays. The aim of normalisation is to adjust this so that the log ratios are comparable across multiple arrays.

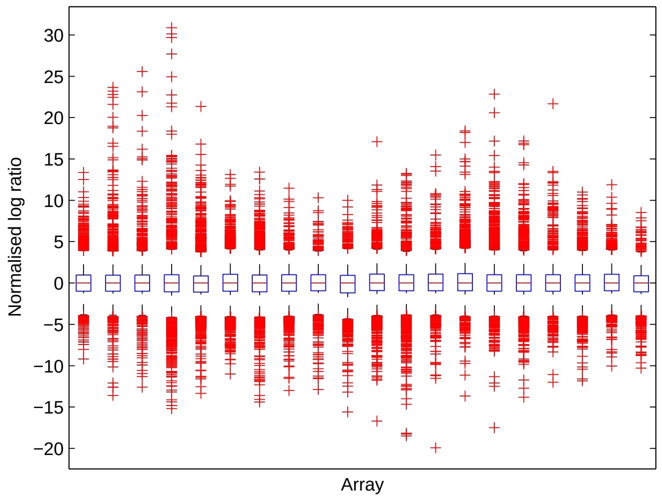
The most common and simple method for normalisation is to apply a normalisation factor L to the log ratio data as follows:

$$M'_j = M_j - L$$

where M_j is the log ratio of the jth gene, and M'_j is the normalised log ratio. This is a simple scaling procedure, sometimes referred to as scale normalisation, where the data range is adjusted by a constant factor across all spots. There are many methods for calculating the normalisation factor L. Often, it can be mean or median of the log ratios across all genes, or a subset of genes, on the array. Where a subset of genes is used, these are typically ‘housekeeping’ genes



(a) Pre-normalisation



(b) Post-normalisation

Figure 2.6: Distribution of log ratio values for each array (a) pre- and (b) post-scale normalisation. In this case, scale normalisation adjusts the interquartile ranges of intensity log ratios so they are equal by array.

— genes which are believed not to change in expression level. The basis of using such genes to control the normalisation factor is that their expression level will remain constant in all samples. Figure 2.6b shows the boxplot distribution of scale-normalised log ratio values (using a median normalisation factor) for the same arrays that are shown in Fig 2.6a. We can see that the median and quartile ranges are now comparable across the arrays.

Some other, more complex, methods for normalisation are based on the raw intensities (i.e. prior to the calculation of log ratios). For example, linear regression analysis is based on plotting the intensities for each sample in a 2 channel experiment in a scatterplot. In theory, in data without systematic variations the plot of each spot should cluster around a straight line whose gradient is 1. Therefore, to normalise, a best-fit line is calculated for the scatterplot using regression techniques, and this is adjusted to fit around a line with a gradient of 1. Lowess normalisation (where Lowess stands for locally weighted linear regression) (Cleveland, 1979) is a technique applied to adjust for dye bias, which occurs when the relationship between spot intensity and gene expression is not the same for all dyes — for example, for a given concentration of mRNA, the intensity of red and green dyes differs (Wickham, 2004). This type of bias can be seen when constructing a ratio-intensity (R-I) plot, which plots the log product (base 10) of the intensities against the log ratio (base 2) of the intensities Q_j and R_j for the query and reference samples (see Fig. 2.7). In unbiased data, the log ratios should be centred around 0. In Fig. 2.7 this does not occur and the plot shows that variation of the log ratio occurs as a function of the intensity (represented by the log product). In this particular example there is more variation as the intensity increases. Lowess normalisation involves applying a local weighted normalisation factor $y(x_k)$. If we set $x_j = \log_{10}(Q_j \times R_j)$ and $y_j = \log_2(Q_j/R_j)$, then the lowess factor $y(x_j)$ is the dependence of the $\log_2(Q_j/R_j)$ on the $\log_{10}(Q_j \times R_j)$. We can use this function, point by point, to correct the log ratio values M_j so that (Quackenbush, 2002):

$$M'_j = M_j - y(x_j)$$

Similarly, Loess normalisation (Workman *et al.*, 2002) is a nonlinear normalisation technique that can be used to adjust for a spatial bias of the two channels

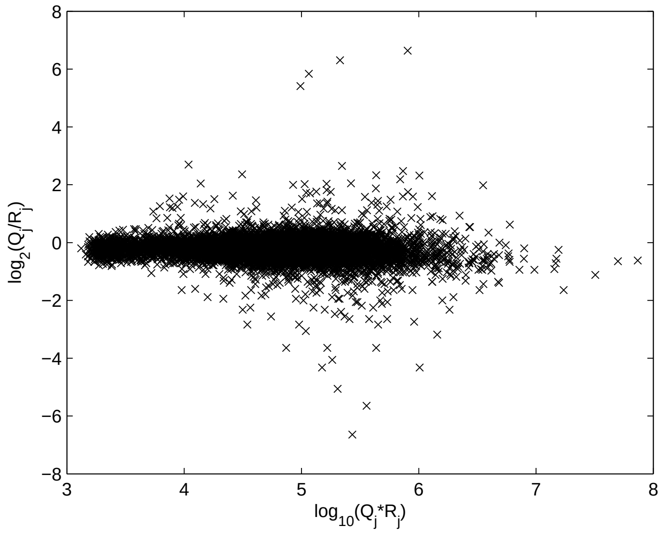


Figure 2.7: Ratio-Intensity plot

— where (for example) some areas of the array slide one of the dyes may be more concentrated than the other.

In the research presented in this thesis, microarray samples from different datasets are used. These datasets are publicly available and found on online expression databases, and often have had some type of appropriate normalisation performed for the arrays within the dataset. Typically only the log ratios are available, and not the original intensities, which restricts the type of normalisation that can be performed. However, since the datasets are from different sources further scale normalisation is usually essential in order to make the expression log ratios comparable across datasets. We use a scale normalisation technique — a simple adjustment of the log-ratios from a series of arrays so that each array has the same median absolute deviation (Park & Wang, 2006). Each log ratio is transformed using the following formula:

$$M'_{ij} = \frac{M_{ij} - \text{median}_i}{MAD_i}$$

where M_{ij} is the log ratio of the jth gene in the ith array (since we deal with multiple arrays), and M'_{ij} is the normalised log ratio. median_i is the median of log ratios across all genes in the ith array and the median absolute deviation

MAD_i is defined as the median of absolute deviations from the median: $MAD_i = \text{median}_i\{|M_{ij} - \text{median}_i\,|\}$.

2.2.3 Analysis of microarray expression data

After performing the microarray experiment to collect data, and processing the array slide to produce normalised expression levels for a set of genes, the next step is to use this expression data to discover interesting patterns and relationships amongst genes and between experimental conditions. This may include identifying groups of genes with similar patterns of expression or genes that have interesting expression patterns (e.g. they are active in certain samples but not in others). First, in Section 2.2.3.1 we describe the format of a microarray expression dataset: the gene expression matrix. Then, in Section 2.2.3.2 we provide an overview of some basic techniques in expression data analysis. Following this, in Section 2.2.3.3, we introduce a particular type of analysis — inferring gene regulatory relationships from expression data — which is the main focus of this thesis.

2.2.3.1 The gene expression matrix

Usually, to be able to perform useful analysis, we require expression levels for a set of genes over a set of different arrays, where each array represents a different experimental condition. This makes up the gene expression matrix, where each entry M_{ji} is the intensity log ratio, which we refer to as the expression level, for gene j in the ith array. The columns of the matrix represent different arrays which we also refer to as samples. Typically, each sample represents a different experimental condition. The rows of the matrix represent each gene's expression profile. This shows how the gene's expression changes across the experimental conditions. In particular, if the samples are temporally related this shows how a gene's expression changes over time under a particular environmental condition. Alternatively, some of the samples may come from healthy cells and some may come from diseased cells. In this case the samples are split into different classes (e.g. healthy and diseased) and we can view the difference between the gene expression profiles across the different classes. For example, Fig. 2.8 shows the

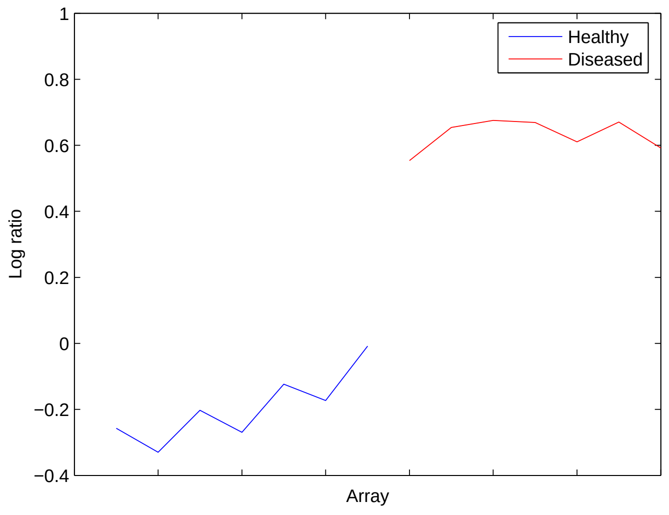


Figure 2.8: Expression profiles for a single gene, for a set of healthy samples and a set of diseased samples (each point on the x-axis refers to a different array — arrays for healthy samples are in blue and arrays for diseased samples are in red. The arrays are temporally related)

expression profiles for a single gene, for a set of healthy samples and a set of diseased samples. It can be seen that in diseased cells the expression of the gene increases significantly, indicating that it could be a gene of interest with respect to this particular disease.

2.2.3.2 Basic analysis techniques

Some types of information that biologists can discover from a gene expression matrix using simple analysis techniques includes the following:

- *Differentially expressed genes*
  These are genes which vary significantly in their expression level between two different states. It is usually these genes that are of interest in that particular experimental condition.

- *Co-expressed genes*
  These are genes that have similar or correlated expression patterns. Groups

of genes which have correlated expression profiles will often have similar biological functions and/or be regulated by a common set of transcription factors.

- *Classification of genes or samples*
  Where samples are labelled into classes, such as healthy and diseased, biologists may be particularly interested in which genes are expressed only in healthy samples and which are expressed in only diseased samples.

- *Temporal analysis*
  Where the arrays complete a time-series of experiments for a particular environmental condition or biological process, biologists are interested in how a gene's expression changes across the time-series.

Differentially expressed genes are those that vary significantly in expression between two states. In a 2 channel microarray experiment, this is equivalent to finding genes whose (normalised) log ratio is significantly different from 0. The simplest method for finding these genes is to calculate the mean $\hat{\mu}$ and standard deviation $\hat{\sigma}$ for the log ratio values of all genes in the array and calculate a confidence interval around $\hat{\mu}$. Genes with log ratio values outside this interval are then defined as differentially expressed. The Analysis of Variance (ANOVA) is an alternative statistical technique that can be used for identifying genes that are differentially expressed between different classes (Causton *et al.*, 2003). ANOVA tests for significant differences between means of two or more populations by comparing variances. The basis of the technique is that variances in measurements may be partitioned depending on the measurement source. In the case of a microarray experiment, we may consider partitioning the measurements by different arrays, the dyes used to label samples, or a different 'class' of sample, such as healthy or diseased cells.

Groups of genes that have correlated expression profiles are often referred to as 'co-expressed' genes. Co-expressed genes often have similar functions or a common set of regulator genes (Heyer *et al.*, 1999). Even if genes are negatively correlated, this can indicate a true relationship, as the expression of one gene may be activated at the same time as another gene's expression is repressed. Clustering techniques are a popular method in the microarray data analysis community, used on gene expression matrices to discover co-expressed genes (Eisen *et al.*, 1998).

Clustering can also help to reduce the dimensionality of data — the expression profiles for thousands of genes can be reduced to a few groups, where all genes in the group have similar behaviour. It can also help to detect outlier genes that have unusual behaviour. Clustering is a well-established data mining technique and two classic methods that are often used for gene expression data are k-means and hierarchical clustering. In hierarchical clustering, objects (in this case, objects are genes) that are close together are iteratively grouped to form a hierarchy of objects. In order to define whether genes are close together, a distance measure, such as Euclidean distance, must be used on their expression profiles. In k-means, the algorithm finds k clusters of genes (where k is pre-defined). The objects are partitioned into k groups and the 'centre' of each group is calculated in Euclidean space. The clusters are iteratively improved by constructing new clusters based on the centres (MacQueen, 1967). Whilst there are newer clustering techniques, there is no overwhelming evidence that these are more appropriate or obtain a better performance than the established techniques (Causton *et al.*, 2003).

Clustering is an unsupervised data-driven technique — so it groups genes based only on their gene expression profiles. In contrast, supervised data mining techniques use additional information to group genes and make decisions. Classification is a supervised method that is used to learn descriptions (of some form) for categories of object annotations. The learnt descriptions can then be used to make predictions on the annotation of new objects. For example, in microarray data analysis, classification can be used to identify expression patterns for healthy and diseased samples and then the learnt descriptions can later be used to assign such classes to unannotated samples based on gene expression data alone. The first classification analysis applied to gene expression data was by Golub *et al.* (1999). In this work, a class discovery procedure automatically discovered the distinction between acute myeloid leukaemia and acute lymphoblastic leukaemia based on expression profiles, and this was used to determine the class of new leukaemia cases. Like clustering, classification is a well-established field of data mining. Techniques range from the simple linear regression and k-nearest neighbour to neural networks and decision trees.

Time series experiments involve taking samples from cells at certain points of a biological process. The results can help biologists discover information about the order and time scale of expression events (Causton *et al.*, 2003). A classic time-series expression dataset is (Spellman *et al.*, 1998), in which the yeast cell-cycle is studied. The cell-cycle is a process that occurs in all eukaryotic organisms (which include animals, plants and fungi) where cells grow and replicate. In this time-series analysis, the focus was on identifying trends and periodicity in the data in order to find genes whose expression profiles correlate with known events during the cell-cycle. An issue with time-series analysis for microarray expression data is that datasets often only contain a small number of samples, sometimes taken at long time intervals, so complex techniques such as Fourier analysis (which is used in signal processing) are inappropriate. Instead simpler techniques are used, where expression profiles are 'timeshifted' and correlations and patterns are sought between them (Zou & Conzen, 2005).

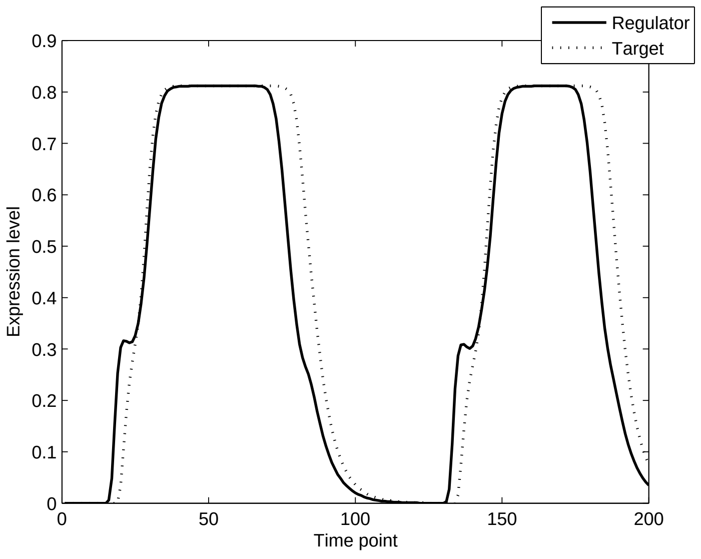
2.2.3.3 Inferring regulatory relationships

Analysing gene expression data is the main focus of this thesis. In particular, we wish to use gene expression data to infer regulatory relationships among sets of genes. Using data to learn a model of a gene regulatory network is also sometimes referred to as 'reverse-engineering'. As discussed in Section 2.1, expression measurements across a set of genes can provide an indication of regulatory relationships between genes. An example of a simple regulatory relationship would be where a regulator gene (transcription factor) induces the expression of a target gene. Figure 2.9a shows how such a regulatory relationship may be seen in the expression profiles of the regulator and target genes. Note that this is synthetic time-series expression data, generated to mimic a regulatory relationship, and contains a relatively high number of samples compared to an average gene expression dataset. In this case, the regulator gene activates the expression of the target gene, and we can see the target gene expression profile mirrors that of the regulator gene, but is time-shifted by a few time points. In Fig. 2.9b, the expression profiles of a regulator and a target gene from yeast are shown, taken from real gene expression data (Gasch *et al.*, 2000). In this case there is no consistent temporal relationship between samples (some samples in the series form short time-series of a few points, but most are standalone experimental conditions). However a correlation between the two expression profiles can be seen.

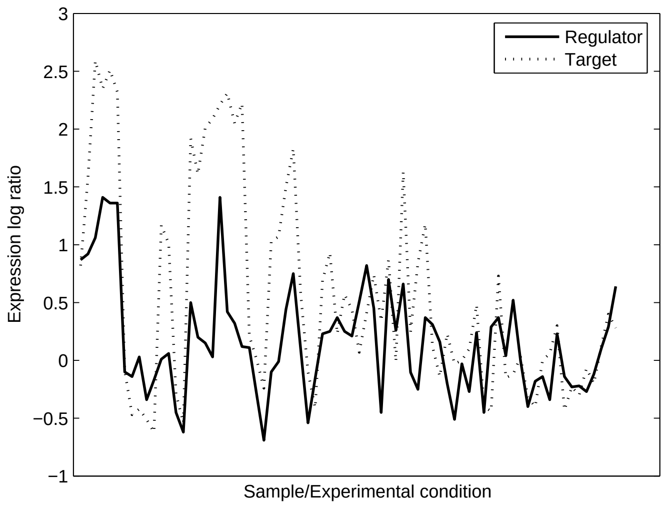
Basic microarray analysis techniques, as described in Section 2.2.3.2, can be used to gain some insight into the gene regulatory structure. As mentioned earlier, it is believed that co-expressed genes may share a common set of regulator genes. For example, clustering techniques allow groups of co-regulated genes to be discovered and this is used sometimes as a basis for gene network learning. Many techniques use clustering for discovering groups of co-regulated genes in the first step to building a gene regulatory network model (Bar-Joseph *et al.*, 2003; Segal *et al.*, 2003a).

However, basic analysis techniques such as clustering alone are not able to reveal the more complex structure of the gene regulation process — that is, how genes interact and their inter-dependencies. Clustering only extracts groups of correlated genes, whereas GRNs can contain nonlinear relationships and combinatorial regulatory control (where a group of regulator genes may interact together to regulate a set of target genes). Thus a group of more complex analysis techniques for reverse-engineering gene regulatory network models have been applied to and/or developed for the task. An overview of the two techniques that are the most prevalent in the literature is provided next; they are Boolean networks and Bayesian networks (Schlitt & Brazma, 2007). Note that there are many modelling techniques for GRNs that have been proposed (e.g. Differential equation modelling, petri nets, stochastic modelling and logical formalisms, to name but a few). However, here we are specifically referring to the reverse-engineering of GRNs from microarray expression data.

Boolean networks are the simplest network model method. They were first proposed by Kauffman (1969). A Boolean network consists of a directed network graph where the nodes are Boolean variables (a Boolean variable has only two possible states — true and false) and a set of Boolean functions, each one associated with a different node. When a Boolean network is describing a gene regulatory network each node corresponds to two possible states of gene expression — either on or off (Filkov, 2006). The state of a target gene can be predicted by the other genes that influence it, through a Boolean function — each node x_i is

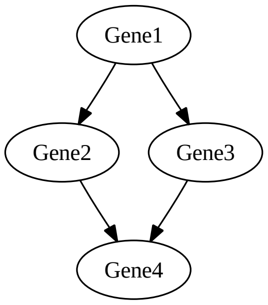


(a) Synthetic time-series



(b) Real data (not time-series)

Figure 2.9: Regulatory relationships shown in expression profiles



(a) DAG

Gene2	Gene3	p(Gene4=off)	p(Gene4=on)
off	off	0.9	0.1
on	off	0.2	0.8
off	on	0.2	0.8
on	on	0.1	0.9

(b) Conditional probabilities for Gene4

Figure 2.10: A simple Bayesian network over 4 variables. (a) shows the DAG component whilst (b) shows the conditional probability distribution attached to the node Gene4

assigned a Boolean function $\mathrm{f}_i(x_{i_1}, ...x_{i_n})$ where x_{i_k} is a node influencing x_i. Since it takes Boolean inputs and outputs a Boolean value, a Boolean function can be thought of as a logical rule. A limitation of Boolean networks is the simplicity in the way that target genes are predicted using Boolean functions — given a particular gene, many Boolean functions may fit the observed expression data equally well. To address this, Shmulevich *et al.* (2002) introduced Probabilistic Boolean networks, an extension of Boolean networks where each gene node may have a family of Boolean functions.

Bayesian Networks have become a popular method for the reverse engineering of GRNs (Friedman *et al.*, 2000; Pe'er *et al.*, 2006; Segal *et al.*, 2003a) from expression data since they are able to represent the network qualitatively (with a network graph) and quantitatively (probability distributions quantify the strength of influences and dependencies between gene nodes in the network graph). A Bayesian Network consists of two components — a Directed Acyclic Graph (DAG) consisting of edges between nodes that represent variables in the domain, and a set of conditional probability distributions associated with each node (see Figure 2.10 for a simple example). The directed edges between nodes indicate the existence of influences and dependencies, the strength of which are quantified by the conditional probabilities. Like Boolean networks, Bayesian networks are relatively easy to interpret by non-technical people due to the transparent nature of the graphical network representation.

Bayesian networks are able to model more complex behaviour than Boolean networks. However as Boolean networks are simpler, they are generally more efficient to learn from data. In Bayesian networks, the conditional probability distributions are able to represent both simple and more complex types of dependencies and can utilise discretised (multiple states) or continuous modelling for gene nodes. They naturally deal with uncertainty (which is important due to both the stochastic nature of gene expression (McAdams & Arkin, 1997) and the noisiness of gene expression data) through the use of conditional probability distributions. Additionally, it is a well-studied method and there are many established techniques for learning Bayesian networks from data. Bayesian networks are our tool of choice to learn gene regulatory network models from gene expression data and they are covered in detail in Chapter 3.

2.2.4 Using multiple microarray gene expression datasets

This thesis focuses on combining multiple sources of data in order to produce GRN models that can potentially be more robust and with greater confidence attached than those derived from a single dataset. In recent years, there has been a rapid increase in the amount of publicly available microarray data. For example, there are online databases where laboratories can contribute microarray expression datasets for different organisms, including yeast (SGD-project, 2008) and *E. coli* (GenExpDB, 2008). Usually, these datasets are accompanied by a peer-reviewed journal publication giving details of the experiment objectives and primary data use.

The opportunity to utilise multiple sources of available expression data for learning GRN models is further motivated by a number of recognised issues relating to the quality of microarray expression datasets. Firstly, microarray data is subject to experimental noise. Whilst differences in expression level measurements can be caused by natural biological variations between samples, they may also result from systematic experimental noise, which can be introduced at all stages of the experimental process, for example, from the sample preparation to the hybridisation step to the scanning of the array. In order to measure and avoid the effects of noise, it is considered good practise to include replicate samples in the experimental design (Causton *et al.*, 2003). Replicates are of two types. Technical replicates are used to assess experimental noise, whilst biological replicates allow biological variability to be measured. Technical replicates repeat some steps of the experimental process on the same sample. A common method for technical replicates is called 'dye-swapping'. This is where multiple extracts are collected, so that we have an extract from each sample, labelled with each dye, and hybridised to the microarray. Biological replicates are obtained by identical sample preparation from multiple biological specimens. Typically, analysis of replicate samples involves pooling or averaging the expression levels across the replicates (Lee *et al.*, 2000; Wernisch, 2002).

The issue of experimental noise is not helped by the fact that due to the relative expense of microarray experiments, sample sizes are often relatively small, in comparison to the number of measured genes. A single microarray dataset usually contains a large number of genes (commonly thousands) but the number of samples is much lower (in the tens or possibly hundreds). This problem is often referred to as the *curse of dimensionality* (Bellman, 1961). For this particular application, it can make it difficult to reliably infer regulatory interactions from a single expression dataset.

Additionally, there are concerns over the reproducibility of results across different microarray platforms (MAQC consortium, 2006). A number of studies have been conducted to compare different platforms; some studies claim that significant differences exist across platforms (Kuo *et al.*, 2002; Tan *et al.*, 2003) whilst other research indicates that different platform technologies produce comparable datasets (Woo *et al.*, 2004; Yuen *et al.*, 2002). As discussed earlier, there are different platforms for microarray technology. In two channel array platforms, two differentially labelled mRNA samples are hybridised together on an array. On one channel platforms, such as the commercially available Affymetrix GeneChip, a single sample is hybridised to each array. One channel microarrays give estimates of the absolute value of expression whereas two-channel technology can estimate only relative differences in expression between genes. Comparing datasets across different platforms is difficult for a number of reasons. Firstly, measurement units may vary. This is especially the case between relative and absolute measures of gene expression, and absolute measures must be transformed to relative measures for comparison purposes. Additionally, studies will come from different laboratories meaning that data is collected with different measurement biases under different atmospheric conditions, leading to different types and levels of experimental noise across datasets. Whilst normalisation techniques can be used, previous research has established that comparing between datasets using standard normalisation techniques is not straightforward due to the different platform and experimental biases involved (Jarvinen *et al.*, 2004; Yauk *et al.*, 2004). Whilst scale normalisation has the theoretical benefit of making arrays between datasets comparable, in practise, bias and artefacts may still remain in the data after normalisation.

The data quality issues described here, together with the increased availability of microarray gene expression data in the public domain, have motivated research in the analysis of many microarray datasets generated by multiple studies, rather than a single dataset, in order to make more robust conclusions from results. For example, meta-analysis methods have been proposed to combine analysis of microarray datasets (Hu *et al.*, 2006). Meta-analysis refers to a set of statistical methods, originating in medical statistics, for combining the results of several studies that address a set of related research hypotheses (Sutton *et al.*, 2000). It has mainly been used to combine measures for the statistical significance (e.g. p-values) of differentially expressed genes across a number of similar studies (Conlon *et al.*, 2006; Rhodes *et al.*, 2002), or to combine effect sizes, which measure the magnitude of the effect of a medical treatment (e.g. for cancer) (Choi *et al.*, 2003). There has also been other research published on combining multiple microarray datasets for other tasks, such as the functional classification of genes (Ng *et al.*, 2003). This research showed that combining multiple datasets can improve classification, even if the experimental focus or conditions varies over the datasets. Lee *et al.* (2004) have used multiple microarray datasets to produce co-expressed networks of genes, where gene pairs are linked if they are correlated across many datasets. They showed that co-expression across multiple datasets is correlated to functional relatedness. Discovering co-expression networks across multiple microarray datasets has since been studied further (Chen *et al.*, 2008; Choi *et al.*, 2005; Yan *et al.*, 2007).

Combining multiple microarray studies specifically for inferring robust regulatory relationships is an area of research that is in its infancy. Wang *et al.* (2006) were the first to address the issue with regards to reverse-engineering GRN models. They use a framework where individual models for each dataset are combined into an overall, consistent solution. Their method is based on linear programming where GRNs are represented using non-linear differential equations. More recently, Redestig *et al.* (2007) and Shi *et al.* (2007) have presented different methods for finding pairwise gene regulatory relationships from multiple expression time-series. Note that searching for pairwise relationships is a simplification of discovering a network model, since many combinatorial relationships will not be uncovered.

The difficulties in comparing directly amongst datasets produced on different microarray platforms and in different laboratory conditions motivates a key premise of this thesis, that combining microarray datasets at the model-level rather than the dataset-level is more appropriate. Later in this thesis (see chapter 5) we compare two frameworks for combining microarray datasets to infer GRNs: *pre-* and *post-learning aggregation*. Pre-learning approaches combine data at the dataset-level by using standard scale normalisation to transform each dataset and concatenating them to form a large set of observations. A model is then learnt from the concatenated dataset. In post-learning aggregation individual models are learnt from each dataset and the models are combined. This allows regulatory interactions to be inferred from each dataset, and then an aggregate model can be built based on the set of models. A post-learning aggregation framework can combine microarray datasets generated by different platforms, research groups and laboratories without requiring any special normalisation to be applied to the datasets. Post-learning aggregation is the approach taken by Wang *et al.* (2006) in previous research (see previous paragraph), but in this thesis Bayesian networks are used as the modelling technique, due to their natural suitability for and previous success in reverse-engineering GRNs. In Chapter 5 we show that our post-learning aggregation approach for Bayesian networks produces better results than pre-learning aggregation for reverse-engineering GRNs from multiple expression datasets.

2.3 Other types of data

Whilst the microarray provides the most available genome-wide data source on gene expression, Section 2.2 has highlighted its reliability and quality issues. Section 2.2.4 motivated the use of multiple microarray datasets to alleviate these issues. However, there are also other data sources that can provide further complementary knowledge on the regulation of gene expression. In this section we introduce these potential sources of data, discuss how they can be used for the reverse-engineering of GRN models and provide an overview of relevant work.

2.3.1 Protein-protein interaction data

Gene regulatory relationships and protein-protein interactions are closely linked. Recall that proteins are gene products that are created during the gene expression process, and that proteins initiate the process of gene expression, so they are of interest in discovering gene regulatory interactions. Physically interacting proteins are often co-expressed (Ge *et al.*, 2001). This means that co-expressed genes discovered using microarray expression data may not be explained by a gene regulatory interaction, but instead by interacting proteins (Nariai *et al.*, 2005). Therefore, for example, protein-protein interaction data can be useful in reducing spurious regulatory interactions inferred by expression data alone. However, it should be noted that co-expressed genes may also be physically interacting proteins: the two conditions are not mutually exclusive.

Just as microarray technology is a high-throughput method for measuring gene expression levels, there are high-throughput technologies available for detecting physical interactions between proteins. Two widely used experimental technologies for detecting protein-protein interactions are the yeast two-hybrid (Y2H) system (specifically for the yeast organism) and affinity purification followed by mass spectrometry (AP-MS). These systems can produce sets of interacting proteins, however like all high-throughput technologies, the resulting data is subject to noise. Different datasets of interacting proteins are often contradictory (Jansen *et al.*, 2003). To deal with this uncertainty, research has been carried out on estimating the reliability of a discovered protein-protein interaction, providing a confidence for a protein-protein pair based on multiple data sources (Bader *et al.*, 2004; Saito *et al.*, 2003).

2.3.2 Transcription factor binding site data

Transcription Factor Binding Site (TFBS) experimental data can also be very useful in predicting gene regulatory interactions. Recall that a gene becomes expressed when a transcription factor binds to a segment of DNA close to the gene. Thus, if we can identify the location of binding sites for a particular transcription factor, we can also identify potential target genes. Therefore, sometimes this type of data is referred to as location data. There are a number of different experimental techniques for discovering binding sites, some localised and some high-throughput technology (Elnitski *et al.*, 2006). A popular high-throughput technology is named ChIP-chip. ChIP-chip combines a method known as chromatin immunoprecipitation (the 'Ch-IP' part) with microarray technology (the 'chip' part). For a specific protein, or group of proteins, ChIP-chip can be used to identify their binding sites. In general, the experimental results can be reported as a set of p-values, where each p-value reflects the confidence in a regulatory relationship existing between a protein and a target gene (Lee *et al.*, 2002).

2.3.3 Literature-based knowledge

There is a wealth of knowledge on gene regulation locked in the scientific literature. When experimental studies are conducted, for example using microarrays or binding site experiments, the results are typically published in a peer-reviewed journal. Thus any confirmed regulatory relationships are likely to be documented in the body of literature in the field. There are a number of ways in which this information can be extracted from the literature.

There are a number of online databases that contain comprehensive knowledge on genes and their products (e.g. proteins). These databases can be considered the backbone of literature-based knowledge on genes — their information often feeds into databases that store more complex types of knowledge, such as relationships between genes. For example, GenBank is an annotated collection of all publicly available DNA sequences, hosted by the National Center for Biotechnology Information (NCBI) (McEntyre & Ostell, 2002-2005). Essentially, this means it contains information about all genes in all organisms. Individual researchers can upload new discovered sequences to the GenBank and this means there can be multiple entries for the same entity, and some information may be contradictory. The NCBI also maintains RefSeq, which is a curated database of DNA sequences — there is only one entry for each molecule.

There are also online databases that build a more complex picture of biological systems. For example, KEGG (Kyoto Encyclopedia of Genes and Genomes) (Kanehisa & Goto, 2000; Kanehisa *et al.*, 2008) is a well-used database of biological systems. It includes information on genes and proteins, their functions, and known interactions (referred to as pathways). To show how entities relate and interact, gene and proteins are stored in a graph format. All information is drawn from other databases such as GenBank, or manually entered based on published materials.

The Gene Ontology (GO) (The Gene Ontology Consortium, 2000) is another collection of databases that relies on individual researchers to contribute knowledge, in order to build descriptions of gene products. GO started with three organism databases (yeast, fly and mouse) but now includes many different organisms. A key concept of GO is the controlled use of vocabulary (ontologies), so that information in the database is consistent. Gene entities can be linked using different relationship types (e.g. *regulates*, *is-a*, *part-of*) so interactions between genes can also be identified.

However, these online databases rely on the manual addition of knowledge from researchers in the field, and as the publication rate increases and subsequently the volume of scientific papers becomes prohibitively large for such a practice, keeping up-to-date information within them becomes increasingly challenging. Additionally, the reliance of manual updates for such databases brings the risk of erroneous or contradictory information. This has led to the development of text mining techniques to automatically extract information about biological entities and their relationships directly from a collection of scientific papers.

Many text mining techniques have been developed specifically for use with biological literature, due to its complex nature (Krallinger & Valencia, 2005). At the most basic level, the PubMed and Medline databases store information relating to each paper published in the biological domain — including author, title and abstracts. Many text mining tools use PubMed or Medline as their knowledge base. The first step in biological text mining is to identify biological entities, such as genes and proteins. This is known as named-entity recognition (NER). Often there are common names for the same biological entity, so it is necessary to be able to identify each entity unambiguously. For example, genes have a number of different identification attributes, such as gene names, symbols and accession numbers, and additionally there are sometimes multiple gene names that actually refer to the same gene.

The next step is to identify relationships and interactions between biological entities (e.g. interactions between genes, and interactions between proteins) from the literature. A simple and efficient method for extracting relationships between entities is based on the co-occurrence of terms in a sentence or abstract (Jelier *et al.*, 2005). This technique, whilst simple, is also effective. It has been shown that the co-occurrence of gene names in a paper abstract frequently reflects a true relationship between the two genes (Jenssen *et al.*, 2001; Stapley & Benoit, 2000).

The Associative Concept Space (ACS) is a biological text mining tool that extends the simple co-occurrence technique (van der Eijk *et al.*, 2004). The ACS makes use of a thesaurus in order to avoid ambiguities by mapping synonyms to biological entities — each entity (e.g. a gene) is described using a 'concept profile'. The ACS is based on calculating a 'distance' between two concepts. A key point is that distances are calculated based not only on the co-occurrence of entities in the same document (a 'one-step' relation), but also on indirect, multi-step relations, where concepts are linked via a number of documents. The ACS improves on simple co-occurrence methods in a number of ways. It can reveal relations between genes based on their contexts, i.e. the other concepts with which they are mentioned, and due to this does not require the genes to be mentioned in the same article for a relationship to be discovered. This research has been extended further by the development of literature-based gene concept profiling and association scores (Jelier *et al.*, 2007; Schuemie *et al.*, 2007a) that describe the overlap in the contexts in which the genes are mentioned. It can discover relationships between biological entities based on a huge number of documents and represent the relationships using an association score matrix, where the association score between a entity-pair reflects the strength of the relationship between them.

2.3.4 Use of prior knowledge in inferring regulatory relationships

In this section 2.3, we have identified a number of data types that offer complementary knowledge to microarray expression data. Whilst expression data gives us information on expression levels, each of these other data types offers a different aspect of knowledge. Protein-protein interaction data provides information on which proteins interact. TFBS data provides a confidence on whether a gene pair have a regulatory relationship, based on binding sites. Literature mining can extract knowledge that has been published in scientific papers. The use of microarray expression data alone for inferring gene regulatory relationships has drawbacks due to its quality and reliability issues. Consequently, it has been recognised that integrating the use of these complementary data types into the inference of gene regulatory relationships can be beneficial. Typically, these complementary data sources are used as prior knowledge in some way. For example, the prior knowledge may be used to identify potential transcription factors, group genes into potential regulatory relationships, or place a probability distribution over candidate network models, prior to learning a model from microarray expression data. In the rest of this section we provide an overview of relevant work in combining prior knowledge with expression data.

The combination of protein-protein interaction data and microarray expression data has also been used to build protein-protein interaction networks and GRNs in parallel (Nariai *et al.*, 2005). In this research, a single model with three components is learnt from the data: a Bayesian network representing a GRN, a

Markov network[1] representing a protein-protein interaction network, and a structural connection between the two networks. The aim of this structural connection is to clearly distinguish regulatory relationships from protein-protein interactions, since genes that are co-expressed and thought to be part of a regulatory relationship may actually be interacting proteins. Segal *et al.* (2003b) demonstrate a similar approach to build a probabilistic model based on both data types. A probabilistic model is constructed for each data type and then a unified model is created based on the individual models.

Most work that combines TFBS location data with microarray expression data is tested on the yeast organism, since ChIP-chip experiments have been carried out to identify the binding sites for over a hundred of its transcription factors (Lee *et al.*, 2002). The most relevant work to this thesis on combining TFBS data with microarray data for inferring GRNs is by Bernard & Hartemink (2005), since it is based on the use of Bayesian networks. In this work a prior probability distribution over candidate Bayesian networks is constructed based on the yeast location data. They found that the use of a prior distribution increased rate of true regulatory relationships discovered. However, they also found that using the location data alone had a good accuracy in comparison with the combination of data types. Similarly, Xu *et al.* (2004) also use location data to build a prior distribution for a Bayesian model to identify regulatory interactions in yeast. This research extends the Module Networks work by Segal *et al.* (2003a) that builds regulatory modules based on expression data only. In other work, Segal *et al.* (2003c) demonstrate a similar approach to the research in which they combine protein-protein interaction and expression data, but use DNA sequence information (relating to binding site DNA sequences) instead of protein-protein interaction data. Gao *et al.* (2004) present MA-Networker, which uses expression and location data. The basis of MA-Networker is the use of a 'coupling factor', which is calculated for each transcription factor-target gene pairing based on both data types. Once again, this research is tested on the yeast organism. They find that 58% of the genes whose promoter region is bound by a transcription factor are true regulatory targets. Imoto *et al.* (2003) propose the use of TFBS data as

[1]A Markov network is similar to a Bayesian network, but only has undirected edges between nodes

prior knowledge to build a prior probability distribution over potential Bayesian models, but do not experimentally test their method using location data.

The incorporation of literature-based data for learning GRNs began with the use of online biological databases, such as KEGG. Imoto *et al.* (2003) introduce energy functions to incorporate prior knowledge sources into Bayesian GRN models and propose the incorporation of many types of different prior knowledge, including literature-based knowledge. Their experiments are conducted with prior knowledge extracted from regulatory interactions that are recorded in the Yeast Proteome Database (YPD). Later, Werhli & Husmeier (2007) extended the approach of Imoto *et al.* to multiple sources of prior knowledge and test their approach on combining protein-protein interactions and KEGG pathways with expression data.

More recent work has seen the integration of text mining techniques to extract knowledge, due to the limitations of online databases. Li *et al.* (2006) use co-occurrence text mining to build a prior regulatory network, and then microarray expression data is used to improve the network. Aerts *et al.* (2008) use text mining to identify DNA sequence information in scientific publications, as a means of identifying the location, organism and target gene information for regulatory relationships. Suwannaroj & Niranjan (2008) use clustering based on the combination of co-expression and text-mined literature gene relationships to build a network of co-regulated genes.

Later in this thesis we present some of the first research on the incorporation of prior knowledge from a large body of relevant literature (see Chapter 4) in combination with expression data for reverse-engineering GRNs using Bayesian networks. We decide to use literature-based data for a number of reasons. Publicly available and genome-wide experimental data, such as protein-protein interactions and TFBS location data can be noisy and difficult to obtain, since experiments are relatively expensive and can be lengthy to run. For example, location data is often only available for a small set of transcription factors. This is why previous research tends to focus on the yeast organism, where a large set of data is readily available. However, by using literature-extracted knowledge, we can harness a huge amount of information from numerous sources for any organism for which there are published papers. We use literature-based gene concept profiling (introduced earlier in Section 2.3.3) to generate a prior distribution over candidate Bayesian network models.

2.4 Summary

This chapter has formed the first part of the literature review, and provides motivation for the contributions of this thesis. In Section 2.1, the mechanisms of gene expression and regulation were introduced. The process of gene expression is often referred to as the ‘central dogma of molecular biology’, since it initiates all cellular processes in living organisms. In a process known as gene regulation, genes interact in order to activate and repress the expression of other genes. Understanding how genes operate in these networks — known as gene regulatory networks (GRNs) — is a major goal of computational biology. For example, understanding GRNs can help biologists gain insight into the causes of genetic conditions or cancer.

This thesis is focused on the reverse-engineering of GRNs, that is, inferring GRN models directly from data sources. In particular, in Section 2.2 we described DNA microarrays, a high-throughput technology for measuring the expression levels of thousands of genes simultaneously. Microarray data can help biologists discover regulatory relationships between genes through the analysis of expression level patterns. However there are a number of recognised issues with microarray expression data. It is subject to both biological variations across samples and experimental noise, which may be introduced throughout the stages of the experiment (e.g. sample preparation or hybridisation). In addition, a key issue with microarray datasets is often referred to as the *curse of dimensionality.* A single microarray dataset usually contains a large number of genes (commonly thousands) but the number of samples is much lower, which can make it very difficult to extract reliable regulatory interactions from a single dataset. Microarray technology is now widely used, and with the subsequent, rapid increase of publicly available microarray data comes the opportunity to produce regulatory network models based on multiple datasets. However, normalisation processes that are intended to allow direct comparison of multiple datasets do not always remove the inherent noise and biases of the microarray platform or laboratory process and environment from which the data is generated. This motivates the first contribution of this thesis: Consensus Bayesian networks and Bayesian networks meta-analysis, approaches for combining Bayesian network models of GRNs at the model-level, which has no requirement for normalising between datasets. This is first presented in Chapter 5 and extended further in Chapter 6.

The drawbacks of using only microarray data to reconstruct GRNs can also be alleviated by incorporating other complementary data sources into the modelling process. There are many other data sources that contribute to available knowledge on GRNs, such as TFBS location data, protein-protein interactions, and literature-based knowledge, which are all covered in Section 2.3. However, publicly available and genome-wide experimental data, such as protein-protein interactions and TFBS location data can be noisy and difficult to find, since experiments are relatively expensive and can be lengthy to run. A better alternative is the use of text mining to extract knowledge from the literature, which can harness a huge amount of information from numerous sources for any organism for which there are published papers. This motivates the second contribution of this thesis, presented in Chapter 4, some of the first research on the incorporation of prior knowledge from a large body of literature in combination with expression data for reverse-engineering GRNs using Bayesian networks.

Next, in Chapter 3, we describe Bayesian networks, our tool of choice for the reverse-engineering of GRNs, and present the remaining part of our literature review, focusing on the use of Bayesian networks with microarray gene expression data.

Chapter 3

Modelling GRNs using Bayesian networks

Bayesian Networks have become a popular method for computational modelling of GRNs from microarray expression data since they are able to represent networks qualitatively (using graphs), and quantitatively (using probability distributions) and thus are relatively easy to interpret by non-technical people. This chapter introduces Bayesian networks for modelling GRNs. The first section (3.1) provides an overview of the basics of Bayesian networks. Following this, in Section 3.2 we explain how Bayesian networks can be used to model GRNs. Section 3.3 describes methods for learning Bayesian networks and attaching confidence levels to network features. In Section 3.4 we discuss methods for evaluating the performance of learnt Bayesian network models. Finally, a review of literature on modelling GRNs using Bayesian networks is provided in Section 3.5.

3.1 Bayesian networks

Bayesian Networks (BNs) (Mitchell, 1997; Murphy, 2001b; Pearl, 1991) are graph-based models of probability distributions that capture properties of conditional independence between variables. A BN consists of two components. The first is a Directed Acyclic Graph (DAG) consisting of directed arcs between nodes that represent random variables in the domain. If there is an arc (also referred to as a

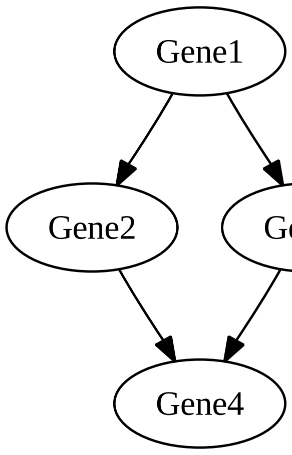


Figure 3.1: DAG component of a simple Bayesian network over 4 random discrete-valued gene variables. This example network is adapted from Murphy (2001b).

p(G1=on)	p(G2=off)
0.50	0.50

G1	p(G2=on)	p(G2=off)
off	0.50	0.50
on	0.80	0.20

G1	p(G3=on)	p(G3=off)
off	0.75	0.25
on	0.25	0.75

G2	G3	p(G4=off)	p(G4=on)
off	off	0.90	0.10
on	off	0.20	0.80
off	on	0.20	0.80
on	on	0.10	0.90

Table 3.1: Conditional probability tables for each node in the DAG shown in Figure 3.1. Note that G1=$Gene1$, G2=$Gene2$, G3=$Gene3$ and G4=$Gene4$

link or an edge) from node A to another node B, then A is said to be a *parent* of B, and B is a *child* or *descendant* of A. Informally, a directed link between nodes $A \rightarrow B$ indicates the existence of a direct influence from A on B. The second component is a set of Conditional Probability Distributions (CPDs) associated with each node. The strengths of the influences indicated by directed links are quantified by these conditional probabilities. The CPDs can be modelled by either a continuous distribution or with Conditional Probability Tables (CPTs) for discrete-valued variables.

For example, consider the BN consisting of the DAG in Figure 3.1, together with the CPTs in Table 3.1 which represent the CPDs for each of the four random variables, which are discrete-valued (binary). In this example, which is adapted from the Sprinkler network in (Murphy, 2001b) to instead show gene regulatory interactions, the variable $Gene4$ is influenced by $Gene2$ and $Gene3$. From the CPT for $Gene4$, we can see that if either $Gene2$ or $Gene3$ is expressed ('on'), then there is a strong probability that $Gene4$ is also on. The node $Gene1$ has a strong influence on $Gene2$ if $Gene1$ is on, but otherwise $Gene2$ is equally probable to be on or off. The node $Gene1$ has no parent nodes, so its CPT specifies the prior probability that it is expressed ('on') or not ('off').

3.1.1 Conditional independence

As mentioned in earlier in this chapter, BNs specify a set of conditional independencies amongst a set of variables. Thus, the concept of conditional independence between sets of variables is a key underlying principle of BNs. Suppose we have three random variables X, Y and Z. We say that X is conditionally independent of Y given Z if:

$$p(X|Y, Z) = p(X|Z)$$

Essentially, this means that once we know the value of Z then X and Y are independent. It can be shown that each node in a BN is conditionally independent of all its non-descendants given its parents (Pearl, 1991). For example, for the network shown in Figure 3.2 the nodes W, X and Y are conditionally independent given the value of their parent node Z. For the previous example network shown in Figure 3.1, $Gene4$ and $Gene1$ are independent when given the values of $Gene2$ and $Gene3$.

3.1.2 Inference

A BN specifies a Joint Probability Distribution (JPD) across the set of variables. Consider the network specified by the DAG in Figure 3.1. The joint probability

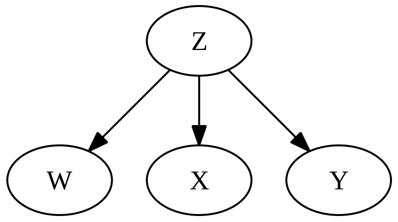


Figure 3.2: Conditional independence in Bayesian networks: this figure shows the DAG component of a Bayesian network. The nodes W, X and Y are conditionally independent given the value of Z.

of all nodes in this DAG can be specified using the chain rule of probability, as follows:

$$P(G1, G2, G3, G4) = P(G1)P(G2|G1)P(G3|G1, G2)P(G4|G1, G2, G3)$$

However, since $G2$ and $G3$ are conditionally independent given $G1$, and $G4$ is conditionally independent of $G1$ given $G2$ and $G3$, this can be rewritten as:

$$P(G1, G2, G3, G4) = P(G1)P(G2|G1)P(G3|G1)P(G4|G2, G3)$$

These probabilities are specified in the CPTs in Table 3.1. Thus, the conditional independence relationships allow us to represent the JPD more compactly.

BNs are so-called because they use Bayes rule to perform inference (Murphy, 2001b). Bayes rule provides a method for calculating the posterior probability $P(A|B)$, based on the values of $P(A)$ — the prior probability, $P(B|A)$ (the likelihood) and $P(B)$. This is useful in cases where $P(A|B)$ is the quantity of interest, but there is only observed data available to calculate $P(B|A)$. Bayes rule is as follows:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.1}$$

Using Bayes rule together with the JPD described by a BN, we can infer the probability distribution on the value of a target variable, given the observed values of other variables in the network. For the example network in Fig 3.1, if it is observed that the $Gene4$ is expressed, $G4 = on$, and we want to find the probability that it is $Gene3$ is expressed, $G3 = on$, then by Bayes rule:

$$\begin{align} P(G3 = on|G4 = on) &= \frac{P(G4 = on|G3 = on)P(G3 = on)}{P(G4 = on)} \tag{3.2} \\ &= \frac{\sum_{g1,g2} P(G1 = g1, G2 = g2, G3 = on, G4 = on)}{\sum_{g1,g2,g3} P(G1 = g1, G2 = g2, G3 = g3, G4 = on)} \tag{3.3} \\ &= \frac{0.4138}{0.6125} \tag{3.4} \\ &= 0.6755 \tag{3.5} \end{align}$$

Similarly, if we want to find the probability that $Gene2$ is expressed, $G2 = on$, given that $Gene4$ is expressed, $G4 = on$:

$$\begin{align} P(G2 = on|G4 = on) &= \frac{P(G4 = on|G2 = on)P(G2 = on)}{P(G4 = on)} \tag{3.6} \\ &= \frac{\sum_{g1,g3} P(G1 = g1, G3 = g3, G2 = on, G4 = on)}{\sum_{g1,g2,g3} P(G1 = g1, G2 = g2, G3 = g3, G4 = on)} \tag{3.7} \\ &= \frac{0.2938}{0.6125} \tag{3.8} \\ &= 0.4796 \tag{3.9} \end{align}$$

Thus we can find which parent node of $Gene4$ has a heavier influence. In this case, $P(G3 = on|G4 = on) > P(G2 = on|G4 = on)$ so it is more likely to be $Gene3$.

Bayes rule can be used to infer the probability distribution for any variable, given the values of the remaining variables, as described above. However, if each node has 2 states, then the JPD has size $O(2^n)$, where n is the number of nodes. This only grows if a node has greater than 2 states. Therefore summing over the JPD takes exponential time. Thus, many alternative exact and approximate inference methods have been proposed to make the problem tractable. For example, a frequently used method for exact inference involves techniques known as variable elimination and the junction tree inference algorithm (Murphy, 2002).

Variable elimination makes use of the conditional independence assumptions defined in the network to simplify calculations (as described earlier in this section). Implicitly, variable elimination can then be used to create an undirected graph, equivalently representing the JPD, called the 'junction tree' on which inference is easier to perform. There are many methods for approximate inference (Murphy, 2001b), including those based on Monte-Carlo sampling. The research presented in later chapters of this thesis makes use of the Bayes Net Toolbox (Murphy, 2001a) to implement the junction-tree inference algorithm for the prediction of node values in BNs.

3.1.3 Explaining away

In cases where a node has multiple parents, there can be a situation where the influencing nodes compete to explain observed data. This is referred to as 'explaining away' (Murphy, 2001b). For example, consider the network in Figure 3.1, where $Gene4$ has two parent nodes, $Gene3$ and $Gene2$. In the case where $Gene4$ is observed, i.e. it has a value, $Gene2$ and $Gene3$ become conditionally dependent. Recall from the previous section, that if $Gene4$ is expressed ('on'), then the probability that $Gene2$ is true is 0.4796. Now, if we assume that $Gene4$ is expressed and also that $Gene3$ is expressed, we can calculate the probability of $Gene2$ being expressed as follows:

$$P(G2 = on|G4 = on, G3 = on) = \frac{P(G4 = on, G2 = on, G3 = on)}{P(G4 = on, G3 = on)} \tag{3.10}$$

$$= \frac{\sum_{g1} P(G1 = g1, G2 = on, G3 = on, G4 = on)}{\sum_{g1,g2,g3} P(G1 = g1, G2 = g2, G3 = on, G4 = on)} \tag{3.11}$$

$$= 0.2991 \tag{3.12}$$

This illustrates how the probability of $Gene2$ being 'on' (or true) decreases, when we know that $Gene3$ is also 'on'.

3.1.4 Parameter learning

Given a network structure and a set of observations on all nodes, we can learn the parameters (that is, the parameters of the CPDs attached to each node) of the network using maximum likelihood parameter estimation. The aim is to find the parameters of the CPDs for each node that best fit the data. Formally, this can be expressed as follows (Needham *et al.*, 2007). Given a BN that represents a probability distribution X and a set of observations (our dataset) D, then we wish to learn a set of parameters θ for X that maximise the likelihood that the data D comes from X. If $D = x_1, x_2, ..., x_N$ (a set of N training examples), and the likelihood is denoted $L(\theta)$ then this can be expressed as follows:

$$\arg\max_\theta L(\theta) = \arg\max_\theta P(D|\theta) = \arg\max_\theta \prod_{i=1}^{N} P(x_i|\theta)$$

The likelihood of the data given the model is product of the probabilities of each example (assuming that the data are independent). For discrete-valued variables, this can be done simply by calculating the frequencies of possible states in the data observations. However, it is often necessary to use a prior for the parameters — otherwise, combinations of variable states that are not contained in the data will receive a zero probability. The prior $P(\theta)$ is taken into account by maximising the posterior probability $P(\theta|D)$ through the use of Bayes rule $P(\theta|D) = \frac{P(\theta)P(D|\theta)}{P(D)}$. Since $P(D)$, the probability of the data (or also known as the marginal likelihood), is a constant, $P(\theta|D) \approx P(\theta)P(D|\theta)$. The research presented in later chapters of this thesis makes use of the Bayes Net Toolbox (Murphy, 2001a) to implement the maximum-likelihood parameter estimation (with priors) to learn the CPTs of learnt network structures.

There are also methods for parameter learning with only partially observed data (i.e. where some nodes have missing data), such as the well-known Estimation Maximisation (EM) process. A description is omitted as in this research we do not deal with learning from missing data.

3.1.5 Equivalence classes

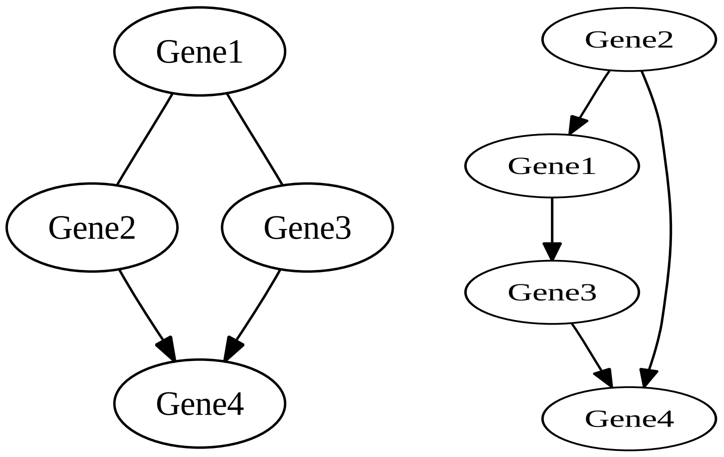
It is important to note that more than one DAG may represent the same set of conditional independencies. A set of such DAGs belong to the same equivalence class. Pearl & Verma (1991) showed that two DAGs are equivalent if and only if they have the same skeleton and the same v-structures. The *skeleton* is the underlying graph with undirected edges and a *v-structure* is an ordered triple of nodes X,Y and Z such that $X \rightarrow Y$ and $Y \leftarrow Z$, and X and Z are not adjacent (not directly connected). In other words, equivalent graphs agree on the same underlying undirected structure, but the direction of some arcs may vary. This means that the equivalence class of a set of conditional independencies can be represented using a partially directed acyclic graph (PDAG), where only some arcs are directed.

For example, for the network shown in Figure 3.1, the PDAG representing the BN equivalence class is shown in Figure 3.3a. The v-structure between the $Gene2$,$Gene3$ and $Gene4$ nodes is preserved. However, the arcs between $Gene1$, $Gene3$ and $Gene2$ become undirected. This PDAG represents the same conditional independencies as the DAGs shown in the original network (Figure 3.1), and an alternative DAG that is shown in Figure 3.3b. Although this DAG does not represent the same intuitive ordering of variables, the conditional independence relationships are identical.

It is possible to derive the PDAG representing the equivalence class for any DAG using an algorithm derived by Chickering (1995).

3.1.6 Causality

An advantage of BNs that is often cited is that they can indicate causal knowledge amongst a set of variables (Mitchell, 1997). However, it is not as simple as considering the directionality of the arcs in the DAG: an arc $A \rightarrow B$ does not necessarily imply that A causes B. This follows from the discussion on equivalence classes: two DAGs may represent the same independence relationships amongst variables, despite the directionality of some arcs varying. Instead, a common interpretation of causality in BNs has been related to the conditional independence relationships amongst variables. The key idea is that at least three



(a) PDAG (b) Alternative DAG

Figure 3.3: PDAG and an alternative DAG representing the equivalence class and conditional independencies amongst nodes of the Gene2 BN shown in Fig 3.1

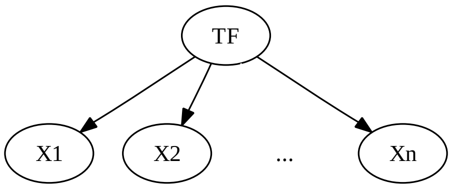
variables need to be measured, and one of these variables acts as a control for the relationship among the other two variables (Murphy, 2001b). In terms of conditional independence, this is expressed by stating that a variable is conditionally independent of other variables in the network, given the value of its parent nodes. For example, in the BN in Figure 3.1 $Gene4$ is conditionally independent of other variables in the network, when given the values of $Gene2$ and $Gene3$, implying that $Gene2$ and $Gene3$ cause $Gene4$. However, BNs and their relationship to causality is a complicated and frequently discussed topic. Pearl (2000) contains a thorough discussion on whether causality can be distinguished from correlation or conditional independence assumptions implied by a DAG.

3.2 Bayesian networks to model gene regulation

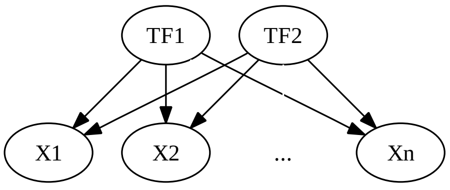
Bayesian Networks have become a popular method for computational modelling of GRNs from microarray expression data (Friedman *et al.*, 2000; Hartemink *et al.*, 2002; Pe'er *et al.*, 2006). In this section we describe how BNs can be used to model a GRN. Recall from section 3.1 that a BN describes a set of conditional independence relationships among a set of variables. In order to represent a GRN using a BN, we apply the notion of conditional independence to gene regulation.

3.2.1 Simple regulatory structures

It makes sense that genes which are regulatory in nature (TFs) will render the genes that they control independent. In other words, if a TF controls a set of genes, these target genes become conditionally independent given the regulator gene (the TF). Therefore, this type of simple regulatory structure involving a TF and its target genes can be easily represented using a BN. Suppose we have a set of target genes X_i and a regulatory gene TF. Then a network representing this can be formed with the TF variable as a parent node to the X_i nodes, as shown in Figure 3.4a. In this case we have that the X_i are conditionally independent on the TF node. The network structure also makes sense in terms of links between nodes i.e. the TF directly influences the values of the target genes. For control by multiple regulators, additional parent nodes representing TFs can be added, as shown in Figure 3.4b. In this case the target genes are conditionally independent, given the values of all controlling TFs.



(a) Single regulator



(b) Multiple regulators

Figure 3.4: (a) shows a standard gene regulatory network, involving a TF parent node to target genes X_i. (b) is a modified version of the same network, but with an additional parent node representing a second regulating gene.

Since regulator genes can also be regulated by other TFs, we can build up a more complex network structure using the principles described, that include longer chains of gene regulation. For example, Fig 3.5 shows a DAG structure for a set of yeast genes.

3.2.2 Modelling using microarray expression data

Microarray gene expression data is the most common type of data used to model GRNs using BNs. In this case each node actually represents the expression value

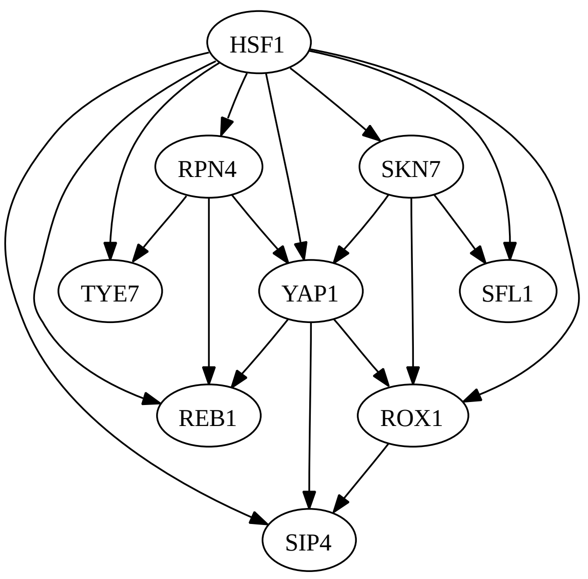


Figure 3.5: A more complex DAG structure representing a regulatory structure between a subset of yeast genes

for that gene. Note that the exact representation of the expression value may vary - i.e. whether it is a log ratio or absolute value, depending on the microarray platform and data preprocessing techniques. Recall from chapter 2, that gene expression values are a continuous data type. However, in the research presented in this thesis gene expression values are discretised into three states using an equal-frequency binning method. This means that when using inference to find the expression value of a gene, we are actually finding the discretised state of that gene expression value. Discretisation is a technique commonly applied to real gene expression data for its use to model GRNs. For example, research by high-profile researchers in this field has used discretisation to classify gene expression values into 2 or 3 states such as 'active', 'no change' or 'repressed' (Friedman *et al.*, 2000; Hartemink *et al.*, 2002). Discretisation is of particular benefit with real data that have a small number of samples and/or can be noisy. It is a simple method that still allows complex regulatory structures to be modelled whilst avoiding the need to deal with parameterised continuous distributions.

This means that the CPDs attached to each node will be CPTs since the gene expression nodes are discrete-valued. The CPTs show how regulators influence

RPN4	HSF1	p(TYE7=high)	p(TYE7=low)
low	low	0.50	0.50
high	low	0.40	0.60
low	high	0.25	0.75
high	high	0.10	0.90

Table 3.2: Example CPT for target gene TYE7 (from Figure 3.5), that is controlled by two TFs, RPN4 and HSF1

the target genes. TFs may interact in a number of different ways to effect regulation on target genes — for example, a pairing of TFs may work where one is an activator and another is a repressor. An example of a CPT for the gene TYE7, that has two regulators, (taken from the DAG shown in Figure 3.5) is shown in Table 3.2. According to this CPT, TYE7 is repressed (low) when its TFs HSF1 and RPN4 are activated (highly expressed). Furthermore, the TF HSF1 has a larger influence on the repression than RPN4, as we can see from the probabilities when RPN4 is low and HSF1 is high.

Additionally, when we think of a node to represent the gene expression value, we can apply the ideas of inference, explaining away and causality (described in Section 3.1) to GRNs. For example, inference can be used to find the probability distribution of expression values for a particular gene, based on the observed values of other genes, or to predict the expression value of a gene based on its probability distribution. In particular, this method can be used to validate GRN structures, by predicting the expression values of genes in the network over new independent datasets. This is discussed in more detail in section 3.4.

3.2.3 More complex regulatory interactions

The BN structures described in Section 3.2.1 only describe static relationships, where a relationship is found between the observations of gene expression values taken in the same sample. They do not model temporal interactions. For example, there may be a time delay between the activation of a TF and the resulting increase/decrease in expression by a target gene. In this case a relationship is found between the observations of expression values taken in samples collected

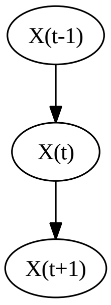
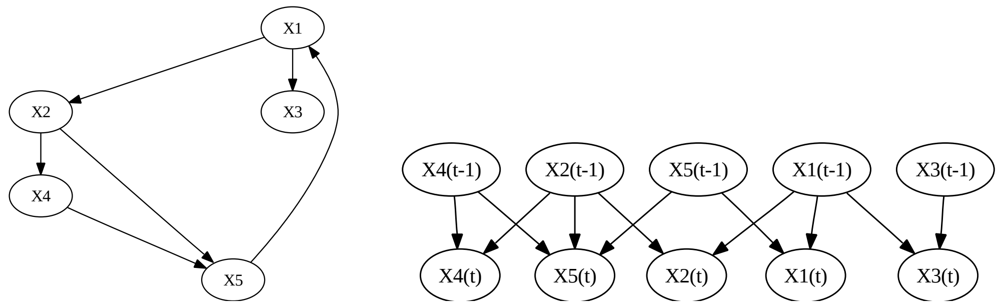


Figure 3.6: A simple DBN

at different time points during the same experiment. Such phenomena can be modelled using Dynamic Bayesian Networks (DBNs). When data samples come from a time-series of measurements, an observation at time t may carry some information on adjacent times (both before and after). This can be represented in a BN structure by using nodes for the values of variables at different time points, e.g. $t-1$ or $t+1$. As in static BNs, arcs between nodes indicate how variables influence one another; in general influence flows between time-consecutive nodes, a simple instance of which is shown in Figure 3.6.

Since they can drill down to individual time steps, DBNs can also model cyclic behaviour. This is particularly useful in GRNs, which often contain feedback loops where TFs regulate themselves. An example of this is shown in Figure 3.7. Part (a) shows a cyclic graph where a feedback loop exists between variables X_1,X_2 and X_5. This is not a valid representation for a DAG since it includes a cycle. However it can be represented in DAG if we include nodes for each X_i at different time steps (which represent different sample points). This is shown in part (b). Here, there are arcs included between the same nodes at different time slices. Additional arcs show the influences between different nodes at different time steps.

In order to apply DBNs to model temporal and cyclic behaviour in gene regulation, gene expression data is required that contains samples taken at different times during an experiment (preferably where the samples also are evenly spaced). Acquiring multiple sets of such data can be difficult. In the research presented in this thesis, we focus on static BNs to model GRNs for two reasons: to first build a solid foundation for combining datasets to model simple regulatory structures



(a) Cyclic graph (b) Potential representation as a DBN

Figure 3.7: Cyclic behaviour represented in a DBN. This figure is adapted from Kim *et al.* (2003)

and secondly due to the paucity of suitable temporal data. However Section 8.3 outlines how the research can be extended to model more complex regulatory structures.

3.3 Learning Bayesian network structures

There are two main approaches to building BN structures, based on a set of data observations. Constraint-based approaches compute conditional independencies between variables and use these as constraints to build a PDAG. The search-and-score approach looks at maximising a score based on how well the network fits the data. In this section we describe both approaches in more detail, and explain why we use the search-and-score method to learn BNs that represent GRNs. Following this, in Section 3.3.4 we describe a method for calculating the confidence level of learnt network features (such as an interaction between a pair of genes).

3.3.1 Constraint-based learning

Constraint-based approaches work by establishing conditional independencies between variables and forming a PDAG based on this. If two variables X and Y can be found to be conditionally independent given another set of variables S, then there is no link between X and Y in the network structure.

Examples of constraint-based approaches are IC (Inferred Causation) (Pearl, 2000) and the PC algorithm (Spirtes *et al.*, 2000). Both algorithms follow a similar process. First, an undirected graph structure (the skeleton) is found by searching for conditional independence relationships among sets of variables. For each pair of variables X and Y, a set S of variables is sought such that X and Y are conditionally independent given S. X and Y are only connected in the graph if no set of variables can be found (i.e. X and Y are not conditionally independent). The PC algorithm invokes efficiency savings here, as it only considers a reduced subset S of conditioning variables, which is those variables that are adjacent to X and Y. Once the undirected graph structure is found, four rules are used to direct arcs to form a PDAG (Pearl, 1991).

Conditional independence between variables can be established using a variety of methods, but the most commonly used involves partial correlation. Partial correlation measures the degree of association between two random variables, with the effect of a set of (control) variables removed. If the partial correlation between the variables X and Y, conditional on a set of variables S is zero, then X and Y are conditionally independent.

3.3.2 Score-based learning

The search-and-score approach has generally been more popular for learning BNs, in particular for inferring gene regulatory relationships. The approach performs a search through the space of possible networks and scores each structure. The aim is to identify the network with the maximum score. A variety of search strategies can be used, the simplest being a greedy hill-climb. In the research presented in this thesis, we use a simulated annealing approach in order to avoid local maxima. The search begins with an empty network. At each stage of the search, networks in the current neighbourhood are found by applying operators such as *add arc*, *remove arc* and *reverse arc* to the current network.

We use the *Bayesian Information Criterion* (BIC) (Schwartz, 1978) for scoring candidate networks. The BIC function is a combination of the model log-likelihood and a penalty term that favours less complex models — as such it is similar to the minimum description length:

$$BIC = \log\ P(\theta) + \log\ P(\theta|D) -\ 0.5\ k\ \log(n)$$

where θ represents the model, D is the data, n is the number of observations (sample size) and k is the number of parameters. $\log\ P(\theta)$ is the prior probability of the network model θ, $\log\ P(\theta|D)$ is the log-likelihood while the term $k\ \log(n)$ is a penalty term, which helps to prevent overfitting by biasing towards simpler, less complex models. The BIC is part of a family of Information Criterion scoring functions that take a common formulation but with different penalty terms (Stoica, 2004). For example, the Akaike Information Criterion (AIC) (Akaike, 1974) has a penalty term of $2k$ (twice the number of parameters), whereas the BIC has the penalty term $k\ log(n)$ that depends on the number of model parameters but also the number of samples. Since the BIC's penalty term takes the number of samples into account it is more appropriate for dealing with microarray datasets, which commonly contain only a small number of sample points.

3.3.3 Comparison of both approaches

Comparative evaluations of constraint-based and search-and-score approaches for learning BNs, specifically for the modelling of GRNs, have been previously carried out (Pournara, 2005; Werhli *et al.*, 2006). In particular, Werhli *et al.* find that for certain experimental types (interventional microarray studies, where the environment or cells are deliberately interfered with for the purposes of the experiment), the score-based approach outperforms the constraint-based approach. However, for simple observational microarray studies, there is insufficient evidence that either approach performs best.

Constraint-based approaches have the advantage of efficiency, even on large datasets. Using a constraint-based approach, a graph structure can be found quickly by simply checking conditional independence relationships based on partial correlations of different sets of variables. However, as shown in Werhli *et al.*, the score-based approach can produce better-performing networks on interventional study data. This type of data makes up a large proportion of the publicly available microarray studies. Score-based learning is also a more flexible method; it can produce networks that have the potential to present a more detailed model of interactions among genes, since all edges can be directed, whilst the constraint-based approach produces PDAGs only. In addition, since the score-based approach is not deterministic, confidence levels for each edge can be inferred (as described in the next subsection 3.3.4). In this thesis, the search-and-score approach, as described in Section 3.3.2, is used to learn BNs that represent GRNs, in conjunction with the BIC scoring mechanism that is implemented in the Bayes Net Toolbox (Murphy, 2001a).

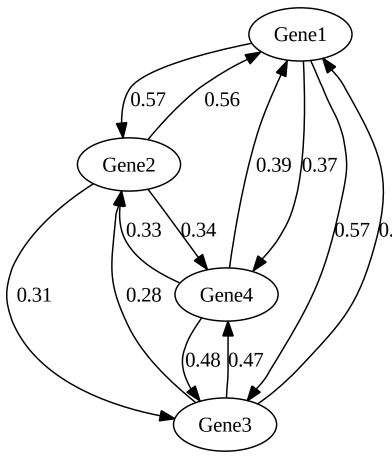
3.3.4 Confidence levels for network edges

When learning BNs using the search-and-score method, a different network may be learnt each time. This is the case where there is a random element to the search strategy, such as in simulated annealing. As we are interested in which genes have regulatory interactions, i.e. if an edge appears between two genes, it is not enough to just learn a network with a high score, since two networks with similar high scores may represent a different set of gene interactions.

Friedman *et al.* (1999) devised a method for computing a level of confidence for features within a BN. For example, a feature could be the existence of an arc between two nodes in the network. We make use of this method to generate more robust network structures and obtain confidence levels on whether two nodes are connected.

The method is based on a well-known statistical method, Efron's Bootstrap (Efron & Tibshirani, 1993). Given a dataset D containing N observations, we create a new dataset by re-sampling N times, with replacement from D. A BN is learnt from the re-sampled dataset. This process is repeated m times, so finally m BNs have been learnt. An estimate of the confidence level for each feature is computed by the proportion of networks that contain that feature.

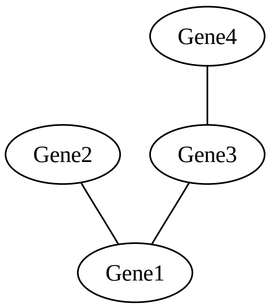
We define a feature as the existence of an edge between two nodes in the network. Thus, network structures learnt using this bootstrapping method are essentially confidence matrices, where the i,j th entry indicates the confidence level of the directed edge from node i to node j. We refer to such a matrix as the *bootstrapped network*. Figure 3.8 shows an example bootstrapped network in both a graph and matrix format.



(a) Graph representation

	$G1$	$G2$	$G3$	$G4$
$G1$	0	0.57	0.57	0.37
$G2$	0.56	0	0.31	0.34
$G3$	0.54	0.28	0	0.47
$G4$	0.39	0.33	0.48	0

(b) Matrix representation



(c) PDAG representing the bootstrapped network, at confidence threshold 0.45

Figure 3.8: A bootstrapped network is actually a matrix of confidence level estimates for each possible edge in the network. (a) shows the network graph representation (b) shows the matrix representation. With larger variable sets, this is an easier way to view the network (c) shows the PDAG extracted from the bootstrapped network, when the confidences are thresholded at 0.45

To create bootstrapped networks that take consideration of equivalence classes, BNs are learnt from each resampled dataset and then converted to PDAGs. Confidence estimates for each edge are then calculated on this set of PDAGs. It is important to note that the edge $i \rightarrow j$ may have a different confidence estimate to the edge $i \leftarrow j$. Where directed edges are present in a PDAG, they contribute only to the confidence estimate for the edge in that direction, whereas undirected edges contribute to the confidence estimate for an edge in both directions.

A PDAG that represents the bootstrapped network at a certain confidence level can be formed by thresholding. If an edge has a confidence level above the threshold, it is included in the PDAG (and if edges are found in both directions — e.g. from node $i \rightarrow j$ and $i \leftarrow j$, then the edge is undirected). Thus, if directional dependencies have enough support in the bootstrapping process they will be captured and represented in the final thresholded PDAG. An example of such a thresholded PDAG is shown in Figure 3.8. Note that this method of thresholding does present the possibility that the extracted PDAG may not be a PDAG — that is, the network structure could be cyclic. However, in our experiments, this did not occur. If it was the case, the network can be converted to acyclic by undirecting an edge in the cycle. The edge to be undirected can be selected by finding which one has the least support to be directed (that is, it has the smallest difference between the confidences in each direction).

3.4 Evaluation of model performance

After learning a BN, we wish to evaluate the performance of the model. This section relates specifically to the evaluation of BNs learnt to represent GRNs. We present two methods which are used to evaluate the research in this thesis. The first method makes a comparison of the network structure to documented knowledge. In terms of GRNs, this means comparing the regulatory interactions represented by the BN with interactions documented in the literature (which have been confirmed, usually through biological experiments). The second method of evaluation does not require documented knowledge, instead it uses BN inference to predict node (gene expression) values based on unseen and independent observations.

	Edge present in	
	learnt network	true network
True positive (TP)	✓	✓
False positive (FP)	✓	✗
True negative (TN)	✗	✗
False negative (FN)	✗	✓

Table 3.3: Comparison between the learnt network and the true network in terms of true and false positives and negatives

3.4.1 Network structure comparison to current knowledge

In the first evaluation method, gene interactions represented by the BN are compared to documented gene interactions found in the literature. A 'true' network is formed from documented interactions, where a directed edge $X \rightarrow Y$, between two genes X and Y, exists if a confirmed regulatory interaction between that pair of genes can be found documented in the literature. The learnt BN structure can be compared to the true network in terms of true and false positives and negatives - see Table 3.3. A true positive (TP) is an edge that is present in both the learnt and true networks. A false positive (FP) is an edge that is present in the learnt network but not in the true network. A false negative (FN) is an edge that is in the true network but not in the learnt network, whilst a true negative (TN) is an edge that is not in the true or learnt network. In terms of the directionality of edges in the learnt network, if the direction conflicts with that in the true network, then the edge is counted as a FP. If the learnt network contains an undirected edge that is directed in the true network we count this as a TP.

The true networks can be obtained from various sources according to the organism of interest. For example, *E. coli* regulatory interactions are documented in the online database RegulonDB (Salgado *et al.*, 2006) and yeast interactions (both confirmed and potential) are listed in the YEASTRACT database (Teixeira *et al.*, 2006). However the online databases from which our 'true' networks are extracted are limited to interactions that have been confirmed by biological studies. For example, RegulonDB contains regulatory information for only about

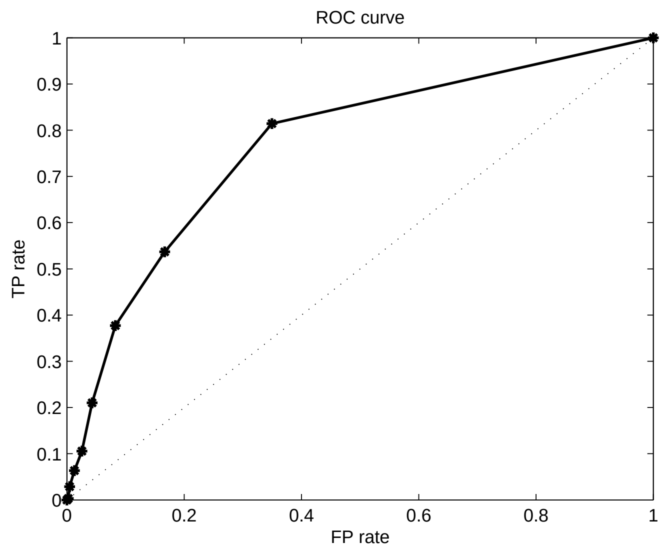


Figure 3.9: An example of a ROC curve representing the evaluation of a bootstrapped network

25% of the genes in the *E. coli* genome (Faith *et al.*, 2007). Therefore the proportion of FP interactions recorded in our learnt networks is likely to be higher than in reality.

In order to compare the numbers of TP and FP edges in different networks, we use Receiver Operator Characteristic (ROC) curves (Krzanowski & Hand, 2009). A ROC curve allows one to view graphically the performance of a classifier by plotting the TP rate (the proportion of true interactions that are identified) against the FP rate (proportion of incorrectly identified interactions):

$$TPrate = \frac{TP}{TP + FN} \qquad FPrate = \frac{FP}{FP + TN}$$

In a ROC space, a perfect network (i.e. identical to the true network) would have a TP rate of 1 and a FP rate of 0, which would sit at the top-left corner of the plot. For this particular application of modelling GRNs, whilst we do not want to 'miss' documented interactions (i.e. a high TP rate is desirable), a low FP rate can be more important as FPs are significantly more costly to biologists (since it may lead to unnecessary and expensive wet lab experiments).

For our experiments we plot a ROC curve for a bootstrapped network, where each point corresponds to the TP and FP rates of the PDAG extracted from the bootstrapped network at a particular confidence threshold. The Area Under the ROC Curve (AUC) is a global measure of the classifier performance, and is often used in classification problems. AUC is a value between 0 and 1. The AUC measures *discrimination*, that is, the ability of the model to correctly classify instances (in this case, an instance is whether an interaction between a pair of genes exists). The AUC also specifies the probability that when we draw one positive and one negative example at random, a higher value is assigned to the positive than to the negative example. This direct interpretation of the AUC originates from the use of the ROC in applications where instances can be assigned a value or score that can be used to rank instances from most to least likely positive. For example, in medical studies where patients are classified into diseased and healthy and assigned a score based on the severity of their disease (Hanley & McNeil, 1982).

In this thesis, the AUC is used to compare bootstrapped networks generated by different algorithms. In general, the closer the AUC is to 1 (and further away from 0.5) the better the overall performance of the network. Figure 3.9 shows a ROC curve (the solid line) representing the evaluation of a bootstrapped network. Each point on the curve represents the TP and FP rate of a PDAG extracted from the bootstrapped network at a different confidence threshold. In this case the thresholds are between 0 and 1 at intervals of 0.1. The points closest to (1,1) represent lower thresholds (0,0.1,0.2,...) , where more edges appear in the PDAG. The points closest to (0,0) represent higher thresholds (1, 0.9, 0.8,...), where fewer edges appear in the PDAG. Note that multiple points on the ROC curve are concentrated at (0,0), due to a lack of edges in the bootstrapped network with high confidences. The AUC is this case is 0.77. The dotted line indicates the TP=FP line, which has an AUC of 0.5. Any point on this line represents a PDAG which cannot discriminate between true and false edges. For this reason, a well performing network should have a curve above the TP=FP line and an AUC that is greater than 0.5.

3.4.2 Prediction of gene expression values

Another method of BN performance evaluation is the prediction of node values on an independent dataset. A BN that is better at predicting node values on unseen data can be said to be more robust as it is less likely to be overfitting on the training data. In the case of a BN that represents a GRN, prediction is of gene expression values on an independent microarray expression dataset. To predict gene expression values, we estimate the CPDs using the same expression dataset from which the structure was learnt, using maximum likelihood parameter estimation (assuming we have complete data). Using the parameterised structure, we can then predict the (discretised) expression value of each gene, based on the expression values of its influencing genes in the network, over samples from unseen independent datasets. The success of gene expression prediction can be measured using *prediction accuracy*, which is the proportion of samples where the predicted discrete states are correct.

3.5 Recent work in using BNs to model GRNs

Research on using BNs to model gene regulation first began in the late 1990s. Prior to BNs, most analyses performed on gene expression data to infer regulatory relationships were clustering techniques used to extract groups of co-regulated genes. However, clustering can only extract groups of correlated genes and not the regulatory network structure. BNs are able to discover more complex, non-linear relationships and transparently represent the nature of interactions (for example, *how* regulators act in combination) through their conditional probability distributions. However, note that clustering is still used sometimes as a basis for gene network learning using BNs (Segal *et al.*, 2003a). For example, Bar-Joseph *et al.* (2003) use a clustering basis for their computational framework to discover gene regulation modules. The remainder of this section documents the original papers and notable more recent work on using BNs to model GRNs. Related work on the incorporation of prior information and multiple gene expression datasets into GRN modelling, the main research areas covered by this thesis, is covered in later chapters 4, 5 and 6.

Friedman *et al.* (2000) published the first research on using BNs to learn gene interactions from yeast expression data. They use the fact that BNs capture properties of conditional independence amongst variables to model statistical dependencies amongst sets of genes, as described earlier in Section 3.2.1. Within the BN, the expression levels of genes are represented as nodes and directed arcs between nodes indicate interactions between genes. The Sparse Candidate algorithm is used to restrict the search space of possible networks for learning. This algorithm uses simple local statistics (e.g. correlation) to identify a relatively small number of candidate 'parent' genes. This is necessary because of the large number of genes usually included in a standard set of expression data. A search-and-score learning process is used, with the Bayesian score (which is similar to the BIC score, but does not include penalty terms). The algorithm is applied to a yeast gene expression dataset (Spellman *et al.*, 1998), which is a classic test dataset in bioinformatics research. Learnt networks are biologically validated in two ways: order relations and Markov relations. Markov relations concern whether genes are directly connected in the network, and order relations are concerned with the direction of the influences between genes. For example, if gene X is an ancestor of Y, then this provides an indication that X has a causal influence on Y. Analysis of the learnt networks uncovered many interesting relationships, many of which make sense biologically (the learnt BNs were not formally validated using TP and FP rates, but when the research was published less was known about regulatory interactions). Since this work, BNs have been used frequently in learning GRNs. A summary of the most significant and relevant research is presented next.

Hartemink *et al.* (2002) present a similar BN methodology for constructing gene networks, which was developed concurrently with the work by Friedman *et al.* Networks are structured similarly with variables representing gene expression levels; however they also include latent (hidden) variables representing gene protein levels, which are not included in the data and are therefore unobserved. They extend the BN framework by adding the ability to 'annotate' the network edges. For example, an edge $X \rightarrow Y$ can be annotated with '+' (positive — indicates if X is high, Y is biased to be high) or '$-$' (negative — if X is high, Y is biased to be low). These correspond to activator/repressor roles of regulator genes.

Imoto *et al.* (2002) use BNs and nonparametric regression with Gaussian noise models in order to capture linear and nonlinear relationships between genes. This method uses continuous modelling for the gene variables as opposed to discretisation, which can require a large number of parameters to be learnt and use suboptimal thresholds in the discretisation process, potentially leading to a loss of information. However, discretisation can offset noise in the data. Imoto *et al.* (2002) observe that their constructed networks confirm the results of Friedman *et al.*, though extra genes are found to mediate some of the relationships found by Friedman *et al.*

Segal *et al.* (2003a) published work on *Module Networks* — a method based on BNs for learning *regulatory modules* from gene expression data. A regulatory module is defined as a set of genes whose expression levels are controlled by a small set of regulator genes. Modules are interacting, so the learnt modules provide a global view of the regulatory network, as opposed to smaller, local-scale networks. The modules also provide further detail as to the conditions under which regulation occurs in the form of testable hypotheses — 'regulator X regulates module Y under conditions W' — referred to as a *regulatory* or *regulation program*. The algorithm is fairly involved and described in further detail in a technical paper (Segal *et al.*, 2005). Candidate regulator genes are drawn from a list of putative transcription factors. Regulatory programs are represented using regression trees. These allow the *if-then* contexts (as described earlier) for regulation to be used. Genes are initially assigned to modules through a clustering procedure. Modules and regulatory programs are iteratively improved using an expectation-maximisation procedure and scored using the Bayesian score. At each iteration, the regulatory program for each module is refined and genes are reassigned to the module which best fits its regulatory program.

More recently, Pe'er *et al.* (2006) presented an algorithm, MinReg, which scales to learning large BN-based GRN models efficiently from microarray data. This work uses biologically motivated restrictions on the network structure in order to reduce the search space during learning. For example, such restrictions include limiting the parent nodes to putative transcription factors. Additionally, a gene is only considered to be a possible regulator gene if it achieves high scores consistently for many target genes in the learning process. The algorithm performed well in both a statistical and biological validation process over synthetic and yeast microarray data. In particular, the algorithm was tested on microarray data for a mammalian organism (mouse), which is generally more complex than simpler organisms such as yeast, and discovered many key regulator genes. Min-Reg also outperformed the Module Networks algorithm when both were applied to the same set of yeast genes and microarray dataset.

Friedman *et al.* (1998) and Murphy & Mian (1999) first postulated the use of DBNs to model gene expression data. Friedman *et al.* discuss extending learning BN structures to DBNs and present gene networks as an applications, but do not actually use real microarray data. The technical report by Murphy & Mian was a review of DBNs and learning algorithms and did not actually contain any experiments using biological data.

Ong *et al.* (2002) were the first to apply DBNs to real time-series microarray data, using latent (hidden and unobserved) nodes to represent regulatory modules, a group of genes that are controlled together by one regulator. In the following year, Kim *et al.* (2003) applied DBNs to the classic yeast data set (Spellman *et al.*, 1998). They compared discrete and continuous models and appropriate learning algorithms for each type of model, though they make no conclusion over whether either method is superior.

Zou & Conzen (2005) present a DBN approach which the authors assert has increased accuracy and reduced computational time compared with other DBN methods. This paper deals with two key issues — the lack of a systematic method for establishing a biologically relevant transcriptional time lag and the excessive computational time needed for model selection (in a search-and-score learning approach). The results presented in the paper — the approach is applied to yeast time series expression data (Chou *et al.*, 1998) — show the method is significantly faster and identifies more known gene relationships than a standard DBN. In order to reduce the number of potential regulator genes (and thus the model search space), they use the biological fact that most regulator genes exhibit an earlier or simultaneous change in expression when compared to their targets (Yu *et al.*, 2003). They also estimate the time delay between the regulator and target from the data which also reduces the number of potential networks. In previous work the time-delay has usually been assumed as the sampling time unit — which may not always be constant in the data and may have no bearing at all on the transcriptional time lag.

In summary, the use of BNs to reverse-engineer GRNs is a well established area of research. BNs have been successfully applied to many microarray gene expression datasets by many researchers and have been proven to provide a natural and transparent representation for a GRN. For these reasons, BNs are chosen as the tool of choice for modelling GRNs in this thesis. However, whilst BNs are a well-suited modelling technique, the data quality issues associated with microarray expression data (as described in chapter 2) have not yet been fully addressed. The remaining chapters in this thesis address the incorporation of additional data sources and the use of multiple microarray datasets for modelling GRNs using BNs. There has been previous research in integrating prior knowledge into BN modelling (which is discussed fully in chapter 4) but the work presented in this thesis focuses on integrating a particular data source, text-based knowledge from literature, on a scale which has not been addressed previously. There has been little research in general in using multiple microarray datasets to reverse-engineer GRNs, and all research of which we are aware of has focused on the use of other modelling techniques. Chapters 5 and 6 present a novel approach for using multiple microarray datasets with BNs.

Chapter 4

Prior knowledge

4.1 Introduction

Whilst the microarray provides the most available genome-wide data source on gene expression, there are concerns over its reliability and the reproducibility of results across microarray platforms or laboratories (MAQC consortium, 2006; Tan *et al.*, 2003). However, the drawbacks of using only microarray data to reconstruct GRNs can be alleviated by incorporating other complementary data sources as prior knowledge in the modelling process. There are many other data sources that contribute to available knowledge on GRNs, such as transcription factor binding site location data, protein-protein interactions, and literature-based knowledge.

This chapter presents some of the first research on the incorporation of prior knowledge from the whole body of biological and medical-related literature into BN models of GRNs. In this work we use a comprehensive collection of prior knowledge (based on the body of related literature in the field) and show that this content helps to improve the modelling process. The ability to use such a large body of prior knowledge lies in the use of advanced biological text mining techniques, literature-based gene concept profiling and the Associative Concept Space (ACS) (Jelier *et al.*, 2007; Schuemie *et al.*, 2007a). Using these techniques, a measure of association between a pair of genes can be calculated based not only on the co-occurrence of entities in the same document (a 'one-step' relation), but also on indirect, multi-step relations, where concepts are linked via a number of documents. It allows the generation of an association matrix for gene-pairs, where each entry represents how related the genes are based on a database of scientific literature. The research presented in this chapter has been conducted in collaboration with the Biosemantics Association (`http://www.biosemantics.org`), who have been responsible for the development of these text mining techniques.

BNs provide a natural mechanism for incorporating prior knowledge through the use of a prior probability distribution on candidate network structures. Building on existing network edge decomposition techniques for building such a prior distribution, this chapter presents a methodology to translate literature-based gene association matrices into a prior probability distribution across network structures, which can then be integrated into the BN learning process. We evaluate its use by comparing BN models learnt with and without prior knowledge on three different gene sub-networks for yeast, *E. coli* and human organisms, and also investigate the effect of weighting the influence of the prior probability distribution. The experimental findings show that literature-based priors can improve both the number of true regulatory interactions present in the network and the accuracy of expression value prediction on genes, in comparison to a network learnt solely from microarray expression data.

The remainder of the chapter is organised as follows. An overview of previous research in relation to this work is provided in Section 4.2. Section 4.3 describes how prior knowledge can be incorporated in learning BNs. Section 4.4 explains literature-based gene concept profiling, and presents a method for translating the literature-based knowledge to prior probabilities for BN network structures. Section 4.5 details the experimental results on three real sub-networks. Finally, Section 4.6 summarises and discusses the findings.

4.2 Related work

Previous research on the integration of literature-based knowledge into the reverse-engineering of GRNs with BNs has utilised online databases such as KEGG as sources of prior knowledge. In the most notable comparable work, Imoto *et al.* (2003) use energy functions to incorporate prior knowledge sources into Bayesian

GRN models and propose the incorporation of many types of different prior knowledge, including literature-based knowledge extracted from regulatory interactions that are recorded in the Yeast Proteome Database (YPD). Later, Werhli & Husmeier (2007) extended the approach of Imoto *et al.* to multiple sources of prior knowledge and applied their approach on combining protein-protein interactions and KEGG pathways with expression data. The use of advanced text mining techniques provides an advantage over this research, since databases such as YPD and KEGG rely on the addition of manual annotations and as the volume of scientific publications becomes prohibitively large, keeping up-to-date information within them becomes increasingly challenging. In contrast, literature-based gene concept profiling can quickly harness the information contained in a huge number of documents into a simple, clear format.

Informative network structure priors have previously been used to incorporate prior knowledge into BN based GRN models. For example, Bernard & Hartemink (2005) use the technique to incorporate transcription factor binding site location data. However, it has not been applied with the type of literature-based information used in this research, and weighting the influence of the prior knowledge has not been addressed.

More recently, in research that was developed concurrently with the work that is presented in this chapter, Larsen *et al.* (2007) and Almasri *et al.* (2008) have used literature-based prior knowledge generated using a tool called PathwayAssist (Nikitin *et al.*, 2003; Novichkova *et al.*, 2003), a natural language processing tool that can identify potential gene interactions based on a collection of literature. In this tool, the natural language processing is based on parsing the meaning of sentences; entities must appear in the same sentence to be related, which is not a restriction of the concept profiling technology that is used in this chapter. The work in this chapter also extends beyond this by considering a weighting on the influence of the prior knowledge during learning. Additionally, we provide a more thorough evaluation of the learnt networks, through gene expression value prediction in addition to a comparison of documented interactions contained in the learnt networks.

4.3 Informative Bayesian network priors

As discussed in chapters 2 and 3, in this research BNs are used to model GRNs. BNs provide a natural mechanism for incorporating prior knowledge relating to the network structure through a prior probability distribution across candidate network structures. Recall the score-based search method for learning BNs (see Section 3.3). This approach performs a search through the space of possible networks, scoring each structure, to identify the network with the maximum score. We use the *Bayesian Information Criterion* (BIC) for scoring candidate networks. As described in section 3.3.2, the BIC function is calculated by:

$$BIC = \log\ P(S) + \log\ P(S|D) -\ 0.5\ k\ \log(n)$$

where $P(S)$ is the log prior probability of the network model S. Usually the prior probability of network structure $P(S)$ is chosen to be uninformative — that is, it is a uniform prior, where every structure is equally likely. Therefore it is usually not included in the score calculation. However, in this research we consider the use of an informative prior (i.e. which is not uniformly distributed over each possible network) based on knowledge contained in the scientific literature. This means that if a particular network structure is favoured in the literature, it will have a higher prior probability, and this will be considered in the scoring process. Section 4.3.1 explains how the prior probability for a network structure can be calculated using an edge decomposition method and Section 4.3.2 discusses how the influence of the prior can be varied by using weighting in the score.

4.3.1 Calculating the prior probability of a network structure

A prior probability distribution for candidate network structures assigns each possible structure a probability such that all probabilities sum to 1. However, enumerating all possible structures is infeasible in most cases. A more intuitive method to calculate the prior probability is by using an edge decomposition technique developed by Castelo & Siebes (2000). Most often, we will find that prior knowledge can be easily represented by edge probabilities (i.e. probabilities indicating whether an edge exists between each possible pair of nodes). For example, it is appropriate for including the literature-based knowledge, since this can be represented using measures of association between two genes (see Section 4.4.1). Bernard & Hartemink (2005) also use this edge-wise decomposition method to include location binding data (which provides a confidence on whether two genes have a regulatory relationship) into BNs. An overview of the approach is provided next.

Suppose that we wish to find the prior probability for a network structure S and probabilities for the existence of each edge in S are provided by an expert or based on some prior knowledge. Now, if a and b are two nodes in a network where B is the expert prior knowledge then

$$p(a \rightarrow b|B) + p(a \leftarrow b|B) + p(a...b|B) = 1$$

where $a \rightarrow b, a \leftarrow b$ indicate directed edges and $a...b$ indicates that an edge between a and b does not exist. In other words, the probabilities of the edge existing in either direction or not existing at all sum to 1.

The next step is to use these edge probabilities to form a probability for the whole network. If we make the assumption that the prior information on the existence of each edge are independent of one another (in other words, the probabilities for the edge between a and b is not related to the edges between a and c or c and d, and so on), then we can multiply the probabilities together to obtain the probability of the whole network S, such that

$$P(S|B) = \prod_{x_i \leftrightarrow x_j \in S, i \neq j} p(x_i \leftrightarrow x_j|B) \prod_{x_i...x_j \in S, i \neq j} p(x_i...x_j|B) \tag{4.1}$$

where x_i and x_j are nodes in S, $x_i \leftrightarrow x_j$ represents an edge between x_i and x_j in either direction and $x_i...x_j$ represents an edge that does not exist. In other words, the prior probability of the whole network S is formed by multiplying the probabilities for each edge in S to exist, and for each edge that is not in S, to not exist. We should note that in this case the independence assumption may be violated if the different sources of the prior information are not independent, which can lead to edge probabilities becoming dependent. For example, if one edge probability is high, another related edge may have a higher probability. This could lead to the introduction of surplus false positive edges into the learnt networks. However, as noted by Castelo & Siebes (2000), the assumption of independence ensures that the calculation of the prior probability distribution is tractable.

Then, taking logs:

$$\log P(S|B) = \sum_{x_i \leftrightarrow x_j \in S, i \neq j} \log p(x_i \leftrightarrow x_j|B) + \sum_{x_i ... x_j \in S, i \neq j} \log p(x_i ... x_j|B) \tag{4.2}$$

Thus, by the edge decomposition method, the log prior probability of a network S (as required for the BIC score of a network) can be calculated by summing the log prior probabilities that each edge the network S contains does exist and that each edge not present in the network S does not exist.

For example, consider candidate networks of four nodes, x_1,x_2,x_3 and x_4, where the prior probabilities for the existence of each edge are shown in the matrix in Figure 4.1a. In this matrix, the i, jth entry indicates the probability that an edge $x_i \rightarrow x_j$ exists. The probability that an edge between two nodes does not exist can be inferred from this matrix, i.e. $p(x_i...x_j) = 1 - p(x_i \rightarrow x_j) - p(x_j \rightarrow x_i)$. Now consider the example network structure S_1 shown in Figure 4.1b. The log prior probability for this structure can be found by applying equation 4.2 so that

$$\log P(S_1) = \sum_{x_i \leftrightarrow x_j \in S_1, i \neq j} \log p(x_i \leftrightarrow x_j) + \sum_{x_i ... x_j \in S_1, i \neq j} \log p(x_i ... x_j)$$

where:

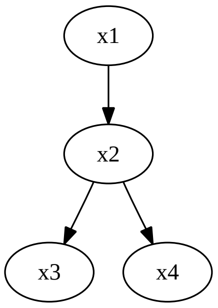
$$\sum_{x_i \leftrightarrow x_j \in S_1, i \neq j} \log p(x_i \leftrightarrow x_j) = log\ p(x_1 \rightarrow x_2) + log\ p(x_2 \rightarrow x_3) + log\ p(x_2 \rightarrow x_4)$$

and

$$\sum_{x_i ... x_j \in S_1, i \neq j} \log p(x_i ... x_j) = log\ p(x_1 ... x_3) + log\ p(x_1 ... x_4) + log\ p(x_3 ... x_4)$$

	x_1	x_2	x_3	x_4
x_1	-	0.35	0.5	0.1
x_2	0.2	-	0.8	0.75
x_3	0.2	0.1	-	0.2
x_4	0.25	0.15	0.15	-

(a) Edge prior probabilities



(b) Example network S_1

Figure 4.1: (a) shows the edge prior probabilities for a set of nodes whilst (b) shows an example network structure S_1

so, this means that

$$\log\ P(S_1) = log\ 0.35 + log\ 0.8 + log\ 0.75 + log\ 0.3 + log\ 0.65 + log\ 0.65 \quad (4.3)$$
$$= -3.6262 \quad (4.4)$$

and removing logs, we have that

$$P(S_1) = 0.0266$$

4.3.2 Weighting the prior

The influence of the prior can be varied by including a weight in the score calculation:

$$BIC = w\ log\ P(S) + log\ P(S|D) -\ 0.5\ k\ log(n)$$

where $w \in (0, 1]$ and is referred to as the *prior weight.* A weight of 0 corresponds to a uniform (uninformative prior) whilst a weight of 1 includes the full log prior probability in the score calculation.

4.4 Literature-based Bayesian network priors

Section 4.3 introduced informative structure priors, which is the natural mechanism of BNs for incorporating prior knowledge into the network learning process. This involves including a log prior probability in the score for each network structure that is considered during the learning search. This prior probability can be calculated using an edge decomposition technique, which is particularly appropriate for the literature-based knowledge we use, since it is a gene-pair association score matrix. This section describes literature-based gene concept profiling in further detail and how it can be used to produce an association matrix (section 4.4.1). Note that literature-based gene concept profiling is a text mining approach previously developed by the Biosemantics Association, collaborators in the research in this chapter. Following this, Section 4.4.2 presents a methodology for translating gene-pair association scores into network edge probabilities. This is a new technique developed by the author of this thesis in collaboration with the Biosemantics Association.

4.4.1 Literature-based gene concept profiling

Information in the literature about biomedical concepts such as genes can be summarised using a technique known as concept profiling (Jelier *et al.*, 2007; Schuemie *et al.*, 2007a). This technique uses a thesaurus containing biomedical concepts. Biomedical concepts may be single word objects found in the biomedical literature, such as gene or organism names, or may be common multiple-word combinations. A thesaurus is required since one concept may have many homonyms (for example, a single gene often has multiple idenitifiers). For our experiments we use a combination of the UMLS Metathesaurus (McCray & Miller, 1998) and the Biosemantics Association's own gene thesaurus, which was created by combining information from several databases, including Entrez Gene and Uniprot.

The concept recognition software Peregrine (Schuemie *et al.*, 2007b), which also disambiguates homonyms, can detect occurrences of thesaurus-concepts in Medline articles published after 1980, resulting in a list of concepts per paper. From this concept profiles are constructed. A concept profile itself contains concepts: it is a vector of concepts with weights, where the weight describes the strength of the association between the concept and the concept to which the profile belongs, i.e. the concept profile for concept i is $\mathbf{w_i} = (w_{i1}, w_{i2}, ..., w_{iM})$ where M is the number of concepts in the thesaurus. The weights in a concept profile for concept i are derived from the set of documents associated with concept i, D_i, which is a subset of the total set of documents D. The weight w_{ik} is based on the uncertainty coefficient between the occurrence of the concept i and the occurrence of the other concept k; it expresses the relative amount of information gained about whether concept i occurs in each document $d \in D_i$ by knowing that concept k occurs in document d (Jelier *et al.*, 2008). Since each gene is a concept in our thesaurus, we can construct concept profiles for all genes.

Since concept profiles are weight vectors, we can calculate a measure of association between two concept profiles and since genes have concept profiles, we can calculate a measure of association between two genes. In this research Pearson's correlation coefficient is used to compute an association score for each pair of genes, which tells us how correlated, or similar, their concept profiles are. Therefore, two genes will have a high correlation if their profiles share the same set of concepts with high weights, and the same set of concepts with low weights. This then allows the generation of a correlation matrix between genes that is based on knowledge contained in the literature. Interestingly, we can even calculate the correlation between genes that have never been mentioned together in the literature, based on shared concepts in the respective concept profiles.

The resulting correlation matrix provides an indication of whether a set of genes are closely related. For example, if genes are mentioned in a similar literature context, then they will have a high correlation. However, we should note that this correlation score only provides information about relationships between genes in the broadest sense. For example, it does not indicate whether a certain regulatory relationship only holds under certain conditions. To solve such ambiguities is beyond the capabilities of current natural language processing techniques. By using homonym disambiguation in the concept recognition process, the most reliable correlation scores currently possible are generated, but still these scores should be taken as indications of probabilities, not as proven biological facts.

4.4.2 Edge prior probabilities from literature-based knowledge

The correlation matrix produced by literature-based gene concept profiling can then be used as the basis for calculating the candidate network prior probabilities using the edge-wise decomposition technique. The correlation value provides a measure of whether two genes are related according to the literature. This section presents a method to translate this correlation value into a probability that an edge exists between that gene pair, based on the idea that a higher correlation translates to a higher probability.

Note that with this type of literature-based prior knowledge there is only information available for whether an edge exists or not and no information on the edge directionality (since the correlation matrix is symmetric). For this reason we calculate a probability that an edge exists between two nodes, without concern for the edge direction. This means that networks that differ only by the direction of edges (such as networks in the same equivalence class) are considered as the same network structure for the purposes of calculating the prior probability. In this way, the literature-based knowledge is used to provide a starting point on relationships between genes whilst the expression data is then used to infer the directionality of relationships. For example, if the literature-based prior provides evidence for a relationship between genes A and B and the expression data infers that gene A activates gene B, then a network containing an edge $A \rightarrow B$ would score higher during the learning process than a network with the edge $B \rightarrow A$, on the basis of the information provided by the expression data.

To translate the correlation values to edge probabilities, we adopt an approach that is based on the confidence level (in statistics) of the correlation values. This is because even correlation values that are small (in absolute terms) may represent a significant association, if they are relatively higher than the majority of values. The method used generates a p-value that represents the confidence level for the correlation value of each gene-pair.

To do this, we first generate the distribution of correlation values for all possible gene pairs, and fit this to a normal distribution. In other words, the mean μ and standard deviation σ of a normal distribution are generated, based on the

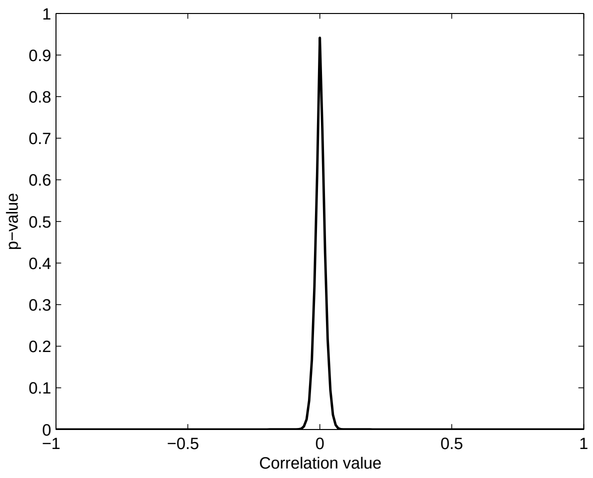


Figure 4.2: For the fitted normal distribution of correlation values for yeast genes (parameters $\hat{\mu} = 0.002$ and $\hat{\sigma} = 0.02$), the p-value for each correlation value, based on the fitted normal distribution is plotted

correlation values. For the organisms considered in the empirical evaluation later, we find that a normal distribution fits well, with the mean very close to zero (e.g. for yeast $\mu = 0.002$). Then, for each gene-pair correlation value of interest, its associated p-value can be calculated based on how far the correlation value deviates from the fitted normal distribution mean — so outlying correlations are more significant, and given a low p-value. The p-value is calculated by using a two-tailed one-sample Z-test, which allows us to test whether a value differs significantly from the mean of a normal distribution. For example, for the distribution of yeast correlation values, the p-values computed for each correlation value are plotted in Figure 4.2. We can see that, in absolute terms, some fairly small correlation values have low p-values, since they differ significantly from the mean. Finally, we define the probability for an edge existing between the gene pair as $P(\textit{edge exists}) = 1 - p$. This means that the probability the edge does not exist is $P(\textit{edge does not exist}) = p$.

Note that the method described above is equivalent to determining a two-tailed confidence interval around the mean containing the correlation value c. The probability $P(\textit{edge exists})$ can be calculated as the probability of a value

X falling within this interval. Since μ is close to zero, this approximates to $P(X \leq |c|)$.

4.5 Evaluating the use of literature-based priors

This section reports on the experiments performed to evaluate the use of the literature prior knowledge. To do this a comparison is carried out between networks learnt from microarray expression data only, with networks learnt from the same expression data with a literature-based prior. The influence of the prior is also varied using weighting. Initial experiments were carried out on gene sub-networks from two basic organisms — yeast and the bacteria *E. coli*, followed by a network of genes in a higher eukaryote (human).

4.5.1 Evaluation procedure

To evaluate the performance of networks learnt with a literature prior compared to those networks learnt from expression data alone, we compare the regulatory relationships found in the networks: transcription factors (TFs) and the target genes regulated by these TFs, which are the child genes of TFs in the networks. Microarray expression datasets are used from three different organisms: yeast, *E. coli* and human. For each microarray dataset, the expression values were discretised into three states using an equal frequency based method (i.e. for each gene, one third of values are categorised as 'low', one-third as 'normal' and one-third as 'high'). For each organism, a literature-based gene correlation matrix was constructed based on all abstracts contained in Medline.

For each organism, hierarchical clustering was used to identify groups of related genes in the literature correlation matrix, and a subgroup of genes was formed by combining the clusters that contained TFs. Each subgroup contained 200-300 genes, which allowed the inference of a larger-scale network whilst maintaining the efficiency of learning. The rationale behind this selection procedure was to increase the likelihood that the genes selected are related in the literature, meaning that the prior knowledge can have a significant effect on the network learnt. If genes were selected for which there is little prior knowledge, it is difficult to measure the effect of using prior knowledge. Selecting genes from clusters that include documented TFs also increases the occurrence of ‘true’ (documented) regulatory relationships within the subgroup of genes, since these will only occur with known TFs as parent genes.

Due to the large number of genes in the selected subgroup, a restriction was imposed on possible network structures — parent nodes must be documented TFs. As well as increasing the efficiency of the network structure search during learning, this meant that each edge in the learnt network had the potential to be documented as a regulatory relationship (thus removing the possibility of unavoidable false positives that would occur when a parent node is not a documented TF).

For each dataset, seven bootstrapped networks were learnt: the expression data network (with prior weight 0) and six posterior networks, learnt with the prior weight set at 0.2, 0.4, 0.5, 0.6, 0.8 and 1 respectively. As described in Section 3.3.4 generating a bootstrapped network is more robust than learning a single network structure. The following terminology is used throughout the remainder of the chapter. A network learnt from microarray expression data with a literature-based prior (i.e. prior weight > 0) is the *posterior network*. A network learnt from expression data only (i.e. a prior weight of 0) is referred to as the *expression data network*.

Previous research on learning GRNs has often evaluated learnt networks by comparing them to documented gene interactions that form a ‘true network’ (usually compiled from online databases of confirmed regulatory interactions) in terms of true and false positives. This type of evaluation is described in Section 3.4.1. However, note that this type of comparison should be treated with some caution for these experiments, since information distilled into a database containing regulatory interactions essentially comes from the literature, which is also where our prior information comes from. However, a comparison to documented interactions can still assist us in measuring the effect of using a prior to learn the network. Therefore, we do use a comparison of the AUC measures (which represent the degree of overlap between the learnt and ‘true’ networks) between the expression data and posterior networks.

Another method of network analysis is the prediction of gene expression values on an independent dataset (described in Section 3.4.2). If a gene node in a posterior network has increased accuracy in prediction on an independent dataset than the same gene node in an expression data network, we can say that the prior does add value to the learnt network. Due to the restriction on the network structure during learning, where only documented TFs can be parent nodes, we compare the prediction accuracies on only the TF nodes between the expression data and posterior networks.

In order to measure the statistical significance of the differences between TF prediction accuracies across networks with different prior weights, the bootstrap learning process is run several times for each dataset. Then, the Cochran-Mantel-Haenszel (CMH) test (McDonald, 2008) is applied, which can be used for repeated tests of independence. The objective of the test is to establish whether two variables are independent, conditional on a third variable that identifies the repeat tests. The null hypothesis is that the two variables are independent of each other within each repetition — that having one value of one variable does not mean it is more likely to have one value of the second variable, or in other words that there is no significant difference between the two variables. In this case, we wish to establish whether the prediction performances of two different networks are independent where each TF identifies the repeat tests. Therefore, in order to obtain a significant result, the null hypothesis should be rejected in favour of the alternate hypothesis, that there is a significant difference between the prediction performances of the TFs in each network.

4.5.2 Application: yeast

For yeast, we based network learning on cell-cycle expression data (Spellman *et al.*, 1998) for a group of 204 genes, selected from all genes in the dataset by the procedure described in Section 4.5.1. 22 of the 204 genes are identified as TFs in the Yeastract database (Teixeira *et al.*, 2006), which lists documented regulatory interactions in yeast.

4.5.2.1 Comparison to documented interactions

Figure 4.3 shows the AUC for the networks learnt with each prior weight. The expression data network (a prior weight of 0) obtains an AUC of 0.56. As the prior weight increases, so does the AUC, up to the prior weight of 0.8, where the AUC peaks at 0.65. Figure 4.5 compares the documented links (true positives) in the expression data only network with those in the network learnt with prior weight 0.8, which has the highest AUC. Although the detail of the networks cannot be seen as they are so large, it can be seen by eye that there are many more documented edges in the network learnt with prior weight 0.8. Taking into account edges with confidence greater than 0.2 only, the TP rate for the expression data only network is 0.076 and for the network learnt with prior weight 0.8, the TP rate is 0.26. FP rates are similar in both networks (around 0.005).

At a prior weight of 1 the AUC dips slightly to just under 0.65. (Note that the prior weight of 1 does not indicate that the whole network is based on literature prior knowledge, rather it indicates that the prior knowledge is fully weighting in the score during learning). This indicates that including the prior does add knowledge to the learnt network. However a balance between the literature and expression data is required — the full prior weight of 1 does not obtain the optimal network.

4.5.2.2 Expression value prediction

Expression values were predicted for all TFs in each network (different prior weights) on an unseen cell-cycle expression dataset (Pramila *et al.*, 2006). In general, TFs in the posterior networks obtain higher predictive accuracies than the same TFs in the expression data network (see Table 4.1). Using the CMH test across all TFs, the posterior networks attain significantly higher accuracies with $p \leq 0.002$. TFs in the posterior network learnt with a prior weight of 0.6 exhibit the most significant difference to the expression data network, with $p = 0.00001$. The prediction accuracies for each TF in the networks learnt with prior weights of 0 and 0.6 respectively are shown in Figure 4.4. The expression value prediction and comparison to documented interactions (Figure 4.3) show the same networks

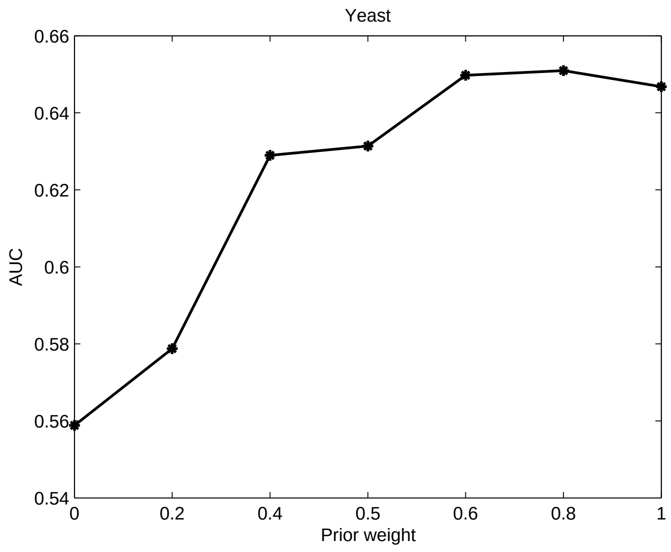


Figure 4.3: Comparison of Area Under the ROC Curve (AUC) values for each yeast network generated with a different prior weight from 0 (no prior) to 1 (fully weighted prior).

performing well — the network learnt with prior weight 0.6 also gains a high AUC value, although it is not the absolute maximum.

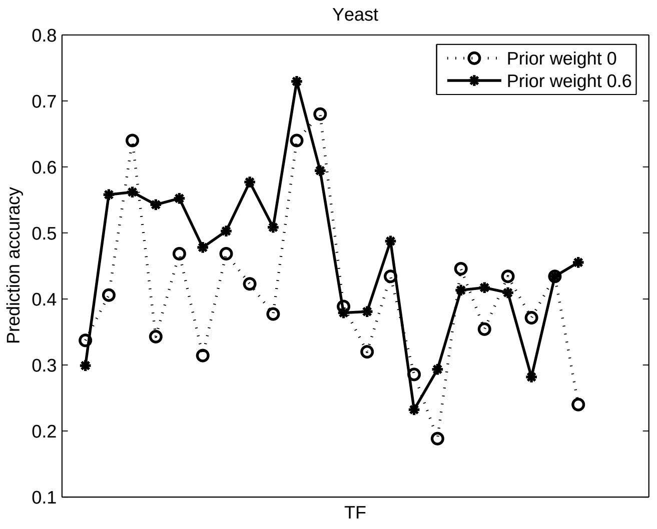
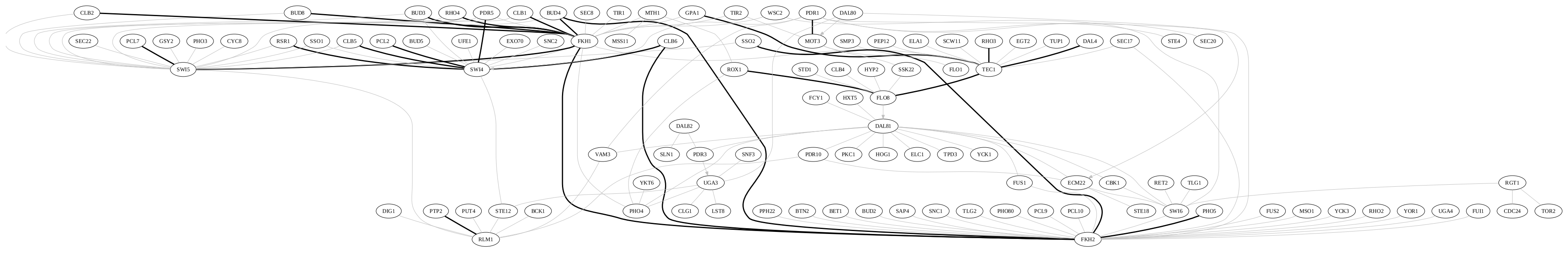


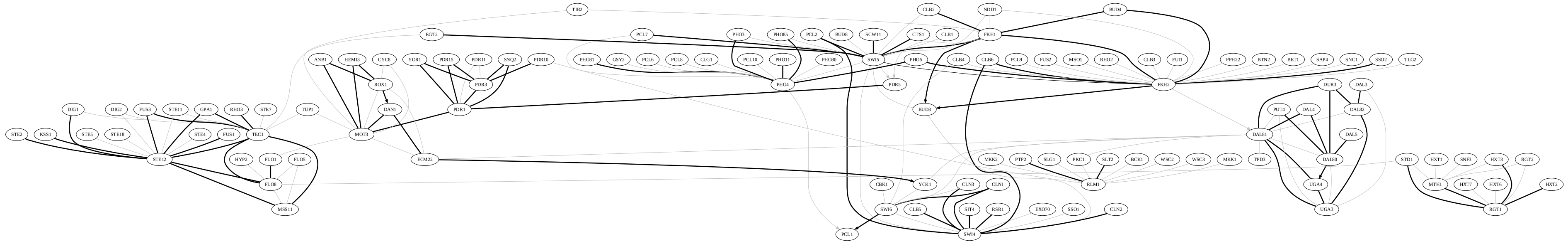
Figure 4.4: Comparison of expression value prediction between the yeast networks learnt with prior weight 0 and 0.6 for the TF genes.

Posterior network (prior weight)	Average TF prediction accuracy Expression only network	Posterior network	p-value from CMH-test
0.2	0.409	0.441	0.00280
0.4	0.409	0.463	0.00001
0.5	0.409	0.442	0.00210
0.6	0.409	0.459	0.00001
0.8	0.409	0.446	0.00004
1.0	0.409	0.456	0.00001

Table 4.1: Yeast expression value prediction results. This table compares the expression data only (prior weight 0) network with each posterior network (prior weights 0.2-1), through the average TF prediction accuracies and the significance of the CMH-test



(a) Prior weight 0



(b) Prior weight 0.8

Figure 4.5: Yeast networks, learnt with prior weights 0 (expression data only) and 0.8 respectively. Edges are shown if they have confidence greater than 0.2. Solid edges represent regulatory relationships confirmed in the Yeastract database. Gray edges indicate edges that are not documented

4.5.3 Application: *E. coli*

For the second set of experiments we considered network learning on *E. coli* expression data (Sangurdekar *et al.*, 2006). This dataset records transcriptional responses to more than 30 chemical and physiological perturbations. The selected group of 262 genes (again, selected from all genes in the dataset by the procedure described in Section 4.5.1) contains 17 TFs, according to the RegulonDB online database of *E. coli* regulatory interactions (Salgado *et al.*, 2006).

4.5.3.1 Comparison to documented interactions

Figure 4.6 shows the AUC for the networks learnt with each prior weight. The pattern exhibited is very similar to the yeast results. The expression data network (a prior weight of 0) obtains an AUC of 0.67. As the prior weight increases, so does the AUC, up to the prior weight of 0.8, where the AUC peaks at 0.81. At a prior weight of 1 the AUC dips slightly to just under 0.8. This shows that the use of the prior adds many edges that represent confirmed interactions but are not represented in the expression data. This may be because the expression data focuses on a particular type of experiment — chemical and physiological perturbations. The prior knowledge can help in adding regulatory relationships that are not exhibited in the microarray experiments.

Figure 4.8 compares the confirmed interactions in the expression data only network with those in the network learnt with prior weight 0.8, which has the highest AUC. As with the yeast networks, there are more documented edges in the network learnt with prior weight 0.8. Taking into account edges with confidence greater than 0.2 only, the TP rate for the expression data only network is 0.35 and for the network learnt with prior weight 0.8, the TP rate is 0.55. FP rates are similar in both networks (around 0.01). As in the yeast results, the optimum prior weight is not 1, but 0.8 — so a balance is required between the prior knowledge and expression data. It is worth noting that *E. coli* is a particularly well-studied organism and this may contribute towards such improvements in network performance in terms of AUC — there is more literature available, which is also more reliable and accurate, than for a less-studied organism.

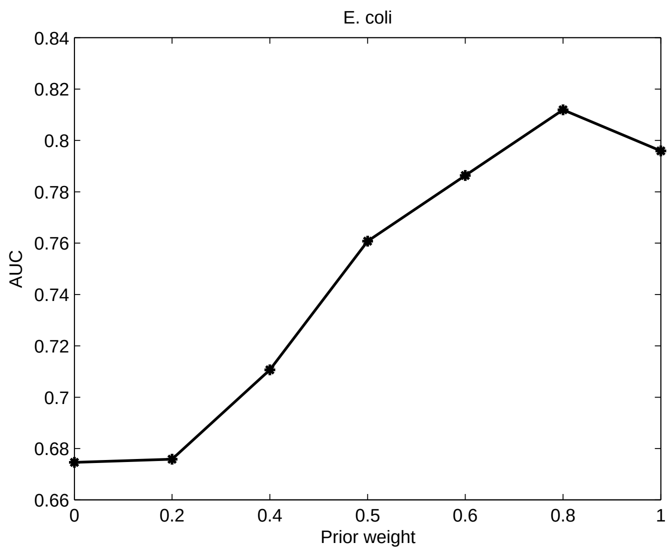


Figure 4.6: Comparison of AUC values for each E. coli network generated with a different prior weight from 0 (no prior) to 1 (fully weighted prior).

4.5.3.2 Expression value prediction

Expression values were predicted for all TFs in each network (different prior weights) on an unseen dataset (Faith *et al.*, 2007), which contains a wide range of different experiments with over 250 samples. However, this dataset does not contain data for all genes in the subset, so for each TF predictions are made using only the bootstrap samples where there are no target genes with missing data. For six TFs there are no bootstrap samples without missing target gene data so they are not included.

Table 4.2 details the average TF prediction accuracies for each posterior network in comparison to the expression data only network, and the corresponding p-value calculated using the CMH test. There is an increase in TF predictive accuracies in the posterior networks, and this is a significant increase ($p \leq 0.05$) for four of the posterior networks. In particular, the posterior network generated with prior weight 1 (full prior weighting) shows the most significant difference ($p = 0.00001$). Figure 4.7 plots the prediction accuracies for each TF in the networks learnt with prior weights of 0 and 1.

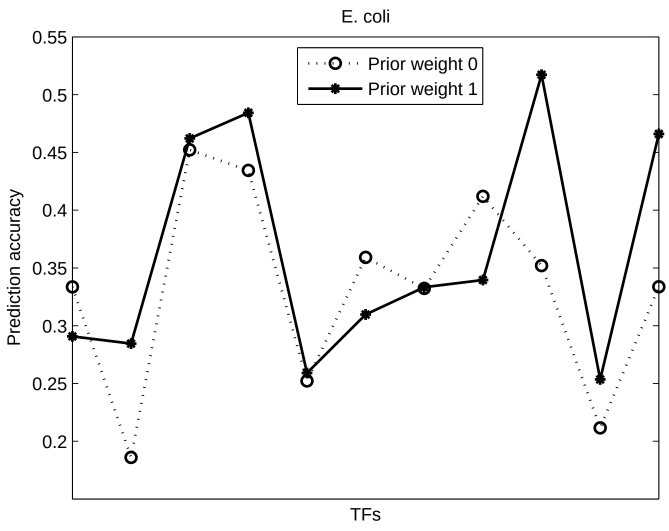
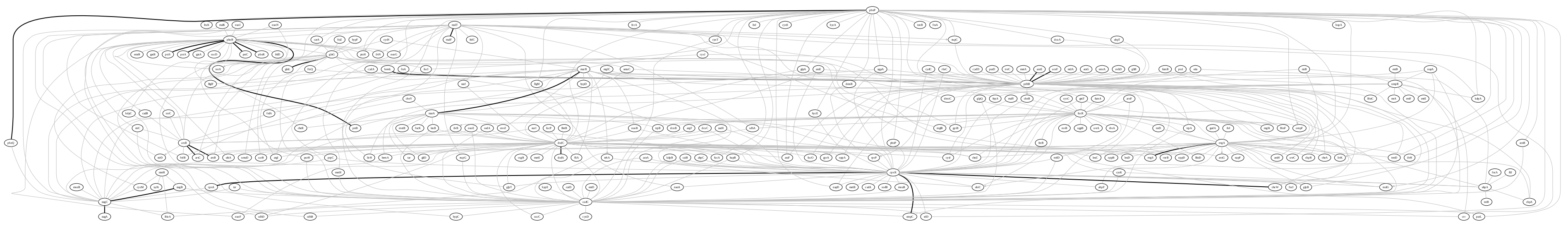


Figure 4.7: Comparison of expression value prediction between the E. coli networks learnt with prior weight 0 and 1 for the TF genes.

Posterior network (prior weight)	Average TF prediction accuracy Expression only network	Posterior network	p-value from CMH-test
0.2	0.332	0.349	0.03200
0.4	0.332	0.344	0.42500
0.5	0.332	0.355	0.31900
0.6	0.332	0.356	0.00600
0.8	0.332	0.358	0.03400
1.0	0.332	0.364	0.00001

Table 4.2: *E. coli* expression value prediction results. This table compares the expression data only (prior weight 0) network with each posterior network (prior weights 0.2-1), through the average TF prediction accuracies and the significance of the CMH-test


(a) Prior weight 0

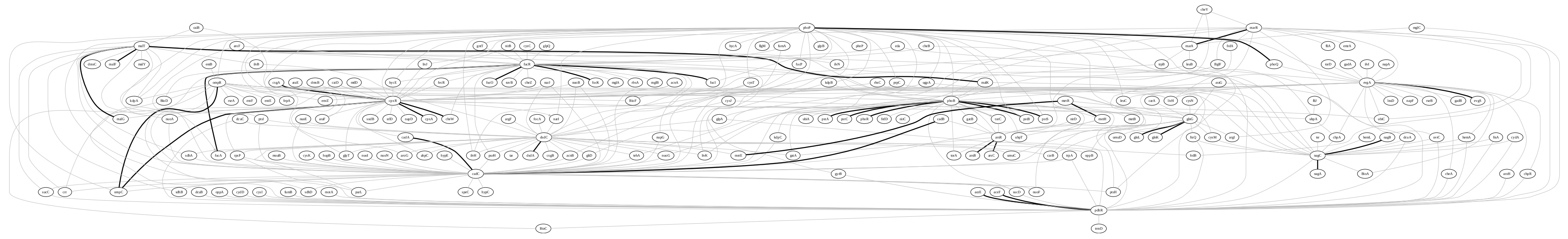

(b) Prior weight 0.8

Figure 4.8: *E. coli* networks, learnt with prior weights 0 (expression data only) and 0.8 respectively. Edges are shown if they have confidence greater than 0.2. Solid edges represent regulatory relationships confirmed in the RegulonDB database. Gray edges indicate edges that are not documented

4.5.4 Application: human

To demonstrate that literature-based prior knowledge approach can be useful for the modelling of GRNs in higher eukaryotes, a third dataset is used, which concerns muscle differentiation studied in cultures of primary human muscle precursor cells. A time-series (7 time points between 0 and 14 days of differentiation) of expression profiles of in vitro muscle differentiation has been generated for six human individuals: three healthy and three patients with Duchenne muscular dystrophy (DMD), which are known to display certain differentiation defects. More details can be found in (Sterrenburg *et al.*, 2006). The selected group of 258 genes consists of literature-based clusters, where each cluster contains at least one documented TF. For this biological system, there is no comprehensive database of regulatory relationships available, and no other suitable microarray expression datasets for evaluating prediction accuracy. Instead, experts in muscular dystrophy from the Biosemantics Association evaluated the biological interpretation of the learnt networks.

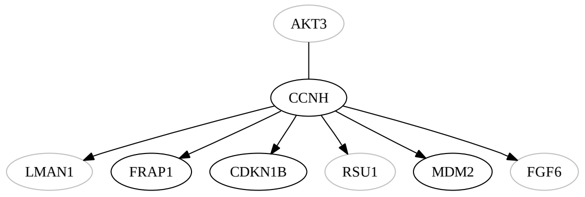
In particular, this evaluation involved examining the differences between the posterior networks generated using different prior weights. As might be expected, the effect of increasing the prior weight is the inclusion of more edges supported by the literature. Figure 4.10 shows the networks for prior weights 0 and 0.4 respectively. In these networks blue edges have a high prior probability (> 0.7) in the literature correlation matrix (indicating a high level of support in the literature prior knowledge). Red edges have a confidence greater than 0.2 in the expression data only network (indicating a high level of support in the microarray expression data). Black edges have support in both the literature correlation matrix and the expression data only network. We can see that there are few black edges in the expression data only network (prior weight 0), indicating few edges with strong support in the literature. However in the network learnt with prior weight 0.4, there is more of a mix of red and blue edges, indicating a mix of knowledge sources.

A further hypothesis is that spurious links, i.e. edges between genes that exhibit a pattern of co-regulation (e.g. correlated expression patterns) but have no regulatory relationship, will disappear from the network as the prior knowledge gains more weight in the learning process. And this does seem to be observed. In particular, in the networks with a prior weight of 0.4 or greater, we can identify subnetworks that are related to certain biological functions such as cell cycle control. The cell cycle network is relevant to the biological system under study since cell cycle arrest is one of the first steps in myoblast differentiation. In particular, compare the nodes of CCNH (for CCNH, also see Figure 4.9 for subnetwork detail for the prior weights 0 and 0.4) and NCOR2 across networks, which gain literature-supported edges in the networks with prior weight 0.4 and above. In the network with prior weight of 0.4, CCNH, which belongs to the family of cyclins that are involved in cell cycle control, forms a central node with several daughter genes (CDKN1B, CDK4, CDK7, CDK9, CCND3, FRAP1, MDM2). Fewer of these edges are present in the expression data only network (prior weight 0). There is also literature support for the regulatory relationship between CCNH and MDM2 and CDKN1B (Datta, 2002; Mandalb *et al.*, 1998).

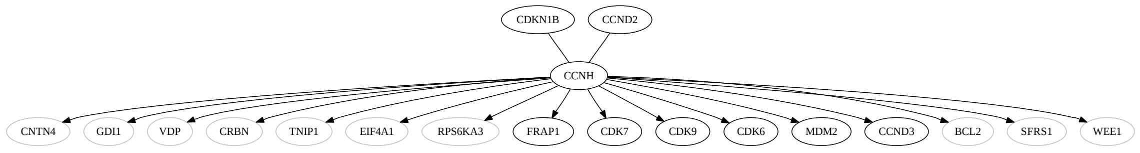
NCOR2, a nuclear co-receptor that inhibits muscle differentiation, is another central node in the network with prior weight 0.4. Consistent with literature (Bailey *et al.*, 1999), NCOR2 demonstrates reduced expression during differentiation, in particular in DMD myotubes. In the network with prior weight 0.4, there are links visible between NCOR2, SKIP and the histone deacetylases (HDAC3, PCAF) that are known to work together in the acetylation of the important muscle transcription factors MYOD1 and MEF2 (Gregoire *et al.*, 2007). As before, this presumably central role of NCOR2 and the histone deacetylation pathway is only evident upon incorporation of the literature prior.

Other possible gene relationships for which there is literature support are not present in the learnt networks. However, this is expected since the networks are learnt from expression data from a highly specific biological system in which not all possible relationships will be existent and also since not all literature-derived relationships are of regulatory nature.

The manual inspection of networks by experts led to an opinion that a literature prior weight of between 0.4 and 0.6 produced networks with the most relevant regulatory links. Higher prior weights appear to lead to the inclusion of too many edges due to literature associations that are not of regulatory nature. Note this is lower than the optimum prior weights found for the yeast and *E. coli*



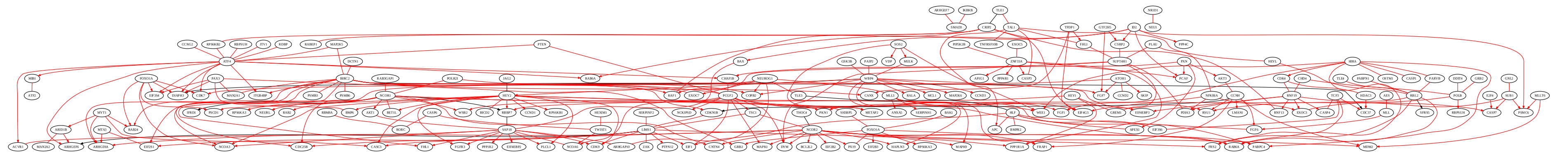
(a) Prior weight 0



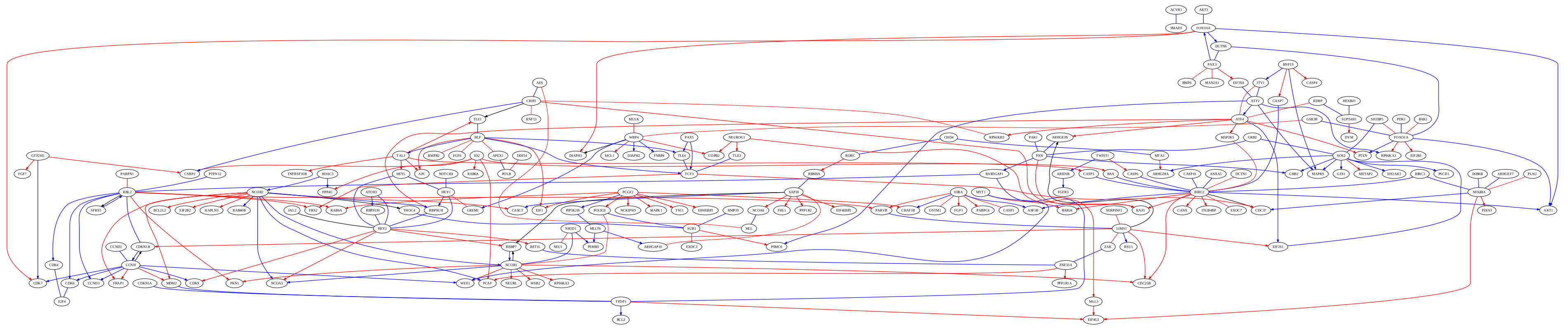
(b) Prior weight 0.4

Figure 4.9: Each network shows the links to and from the gene CCNH, a central node of the cell-cycle network. Nodes that are outlined in black (not gray) are confirmed targets of CCNH. As the prior weight increases, we can see that the number of confirmed targets also increases

networks, which were both set at between 0.6 - 0.8. This may be because there is less literature related to genes in the human organism, whereas yeast and *E. coli* are both well-studied organisms. Where there is less literature, it may be less reliable, so more weight needs to be assigned to the microarray expression data.



(a) Prior weight 0



(b) Prior weight 0.4

Figure 4.10: Human network for prior weights from 0 (expression data only) and 0.4. Each edge has a confidence that greater than or equal to 0.2 (edges with lower confidence are not included). Blue edges have a high prior probability (> 0.7) in the literature correlation matrix. Red edges have a confidence greater than 0.2 in the expression data only network. Black edges have support in both the literature correlation matrix and the expression data only network

4.6 Discussion

This chapter has focused on the use of prior knowledge to improve the performance of BNs learnt from microarray gene expression data. Whilst the microarray provides the most available genome-wide data source on gene expression, there are concerns over the reliability and comparability of different datasets. However, the use of prior knowledge sources, such as transcription factor binding site location data or literature-based information can help to alleviate these concerns and potentially improve the modelling process.

This chapter has presented some of the first research on the incorporation of prior knowledge from a large body of relevant literature for BN learning of gene networks. The use of literature-based gene concept profiling means information contained in a huge number of documents (for example from a database of papers such as Medline) can be represented using a gene correlation matrix. We make use of an informative prior probability distribution over BN structures, the natural mechanism for incorporating prior knowledge into BN learning, together with a method for computing the probability of a network structure using edge-wise decomposition. Building on these existing techniques, this chapter has contributed a method for translating literature-based gene-pair correlations to network edge prior probabilities and investigated the effects of weighting the influence of the prior knowledge during learning.

Comparable related work on integrating prior knowledge into BN learning for GRNs has focused less on the biological content of prior knowledge. Where literature-based knowledge sources were used, these were based on databases such as KEGG. The use of advanced text mining techniques provides a powerful advantage over this previous research, as it allows up-to-date information from a huge amount of literature to be used. In addition, in previous research where real datasets have been used for evaluation, typically this has been on a small scale using less than 40 genes in total. In this work, we have performed evaluation on networks of 200-250 genes, for three different organisms.

In the experiments presented in this chapter, posterior networks were learnt using different weights on the prior, ranging from 0.2 to 1, and a network from expression data only, which has an equivalent prior weight of 0. Two methods of network performance evaluation were used to compare the different networks. In the first, true and false positive rates are used to compare the learnt networks to a 'true' network of documented regulatory interactions. This type of comparison should be treated with some caution since information since documented regulatory are contained in the literature, which is also where our prior information comes from. However, it is still helpful in measuring the effect of using a prior to learn the network. The prediction of expression value on TF gene nodes is used as an alternative evaluation method (on yeast and *E. coli*). If the posterior networks exhibit increased accuracy in prediction on a independent dataset then we can say that the prior does add value to the learnt network.

When compared to documented relationships, the posterior networks were closer to the 'true' network than the expression data network. In yeast and *E. coli*, the posterior network with a prior weight of 0.8 gave AUCs of 0.65 and 0.81, improvements over the expression data network AUCs of 0.56 and 0.67 respectively. This shows that the prior can add many edges that represent confirmed interactions, but are not exhibited in the expression data. Microarray experiments are often focused on a particular subsystem of genes, so prior knowledge can assist in 'filling the gaps' for genes that are not the particular experimental focus of the expression dataset.

In general, the accuracy of expression value prediction for TF genes in the posterior networks was greater than for the same genes in the expression data network. Like the comparison to a true network, there is an optimal prior weight in terms of prediction accuracy, with the network learnt with prior weight 0.6 obtaining the largest improvement over the expression data network in yeast. This indicates that the posterior network structure is more robust for making predictions outside of the original dataset.

A lack of documented regulatory interactions for human meant that evaluation of the learnt networks was carried out with the help of experts in muscular dystrophy (which is the focus of this particular microarray dataset). The results provided evidence that incorporation of literature information does result in removal of edges that represent spurious correlations and also generates networks containing modules of functionally related genes.

However, careful weighting of the literature information appears to be needed. For yeast and *E. coli* an optimal AUC was achieved with a prior weight of 0.8. For the human network, a manual inspection by experts indicated that the optimal prior weight was around 0.4 - 0.6. It appears to be a careful balance — whilst the inclusion of the prior information helps to reduce spurious regulatory relationships, higher prior weights can lead to the inclusion of too many edges due to literature associations that are not of regulatory nature (e.g. proteins in the same multi-protein complex). The lower optimal prior weight for the human network also seems to indicate that the amount of literature has a bearing on the most appropriate prior weight — yeast and *E. coli* are well-studied organisms with a lot of related literature. Where there is less literature, for example with the human organism, more weight needs to be assigned to the microarray expression data.

Chapter 5

Combining multiple microarray gene expression datasets

5.1 Introduction

With the wider use of microarray expression profiling technology and subsequent rapid increase of publicly available microarray data comes the opportunity to produce GRN models based on multiple datasets. Drawing together a richer and/or broader collection of data has the potential to produce GRN models that are more robust, have greater confidence and place less reliance on a single dataset.

In this chapter we compare two frameworks for combining microarray datasets to model GRNs: *pre-* and *post-learning aggregation.* When learning from multiple datasets, there is a choice for *when* to aggregate knowledge within the datasets. In pre-learning aggregation, data is combined prior to learning, and a model is learnt from the combined dataset. In post-learning aggregation individual models are learnt from each dataset, and these are combined after learning. The resulting combined model represents prominent features which occur in all, or a subset of, the individual dataset models.

Combining expression datasets directly, as in pre-learning aggregation, can be difficult as experiments are often conducted on different microarray platforms, and in different laboratories, leading to inherent biases in the data that are not always removed through pre-processing such as normalisation. Additionally, comparison between different datasets can be difficult since measurement units may vary across platforms (see Section 2.2.4 for more details on bias and normalisation of microarray data). The key advantage of a post-learning aggregation framework is that it can combine microarray datasets generated by different platforms, research groups and laboratories without necessarily requiring any normalisation or transformation of the data.

This chapter contributes two novel approaches for post-learning aggregation, each based on aggregating high-level features of Bayesian network models that have been generated from different microarray expression datasets. *Bayesian networks meta-analysis* is based on combining confidence levels attached to network edges whilst *Consensus Bayesian networks* identify consistent network features across all datasets. Both approaches are applied to multiple datasets from synthetic and real (*E. coli* and yeast) networks and it is demonstrated that both methods can improve on a network generated from a single microarray dataset, or networks learnt using a pre-learning aggregation approach, where a combined dataset is formed by concatenating a collection of datasets and then applying standard scale normalisation.

The remainder of the chapter is organised as follows. In Sections 5.2 and 5.3 we describe pre- and post-learning aggregation methods in more detail. Section 5.4 details our experimental results on three real sub-networks. Finally, Section 5.5 summarises and discusses the findings.

5.2 Pre-learning aggregation

Pre-learning aggregation is the simplest approach for using multiple datasets to reverse-engineer a BN model. When applying the pre-learning aggregation approach, datasets are concatenated, if necessary after pre-processing steps such as normalisation, and a BN model is learnt from the combined dataset. For the purposes of the comparison between pre- and post-learning aggregation in this chapter, scale normalisation (as described in Section 2.2.2.3) is used on each microarray dataset to be combined. Scale normalisation has the benefit of making the arrays within a dataset comparable, and it also means that arrays between datasets are comparable. Thus, we can use scale normalisation to combine multiple microarray datasets into one, allowing the generation of a single BN from multiple studies.

5.3 Post-learning aggregation

In post-learning aggregation, individual models are learnt from each dataset, and these are combined after learning. As discussed in Chapter 3, in this research BNs are used to model GRNs from each microarray datasets. This section presents two novel approaches for post-learning aggregation with BNs. The first method is a *Consensus* approach that identifies the intersections — that is, common edges — amongst the network structures generated from different datasets. Only consistent features and dependencies appear in the final Consensus network, reducing the occurrence of spurious relationships. The second technique is based on meta-analysis, an established field of research for combining the statistical outcomes of medical studies. We use an inverse-variance weighting meta-analysis method to combine confidence levels that are attached to each network edge. The two methods are described next, followed by a review of related work on combining BN structures.

5.3.1 Consensus Bayesian networks

The Consensus approach (see Algorithm 1) is based on the identification of consistencies across a set of networks — edges that appear in all, or a certain proportion of the networks in the set are included in the Consensus network structure. For each input dataset, a bootstrapping approach is used to learn individual PDAG structures with confidence estimates attached to each edge. We use thresholding (as described in Section 3.3.4) to obtain a final PDAG from each bootstrapped network. Whilst bootstrapped BNs and thresholding have been used previously to learn more robust GRN models (Friedman *et al.*, 2000; Pe'er *et al.*, 2001), we use the thresholded bootstrapped networks as inputs to the Consensus algorithm in order to find consistencies across networks that have been generated from multiple datasets.

```
Consensus Bayesian networks
Input: Set of n individual networks (PDAG representation), consensus
       threshold (between 0 and 1)
Output: Consensus network

for each pair of nodes i,j do
    /* Build Consensus network structure                                  */
    if an edge $e_{ij}$ exists between nodes i and j in a proportion of individual
    networks $\geq$ consensus threshold then
        include edge $e_{ij}$ in the Consensus network
    end
    /* Assignment of edge direction                                       */
    if edge $e_{ij}$ exists in the Consensus network then
        if there is no conflict in the input edge directions then
            Consensus edge $e_{ij}$ is the same direction (whether directed or
            undirected)
        else
            if There is a majority direction (or undirection) then
                Assign edge $e_{ij}$ in the majority direction
            else
                Assign edge $e_{ij}$ as 'unknown' direction
            end
        end
    end
end
```

Algorithm 1: Consensus Bayesian networks

The Consensus algorithm is based on a simple idea: if an edge appears in multiple input networks, it is more likely to represent a true interaction. The algorithm proceeds as follows. Each pair of nodes is considered in turn and an edge between them is created in the Consensus network if such an edge exists in a proportion of the input PDAGs that exceeds the consensus threshold. Assigning the edge direction is a little more complex. If there is no conflict regarding that edge's direction in the input networks then its direction/undirection remains the same in the Consensus network. However, if there is conflict, this introduces some uncertainty regarding the edge direction. If there is a majority in the input networks regarding edge direction, then the edge is assigned the majority direction in the Consensus network. Thus, directed edges with enough support will appear in the Consensus network. If there is no majority then the edge is left as 'unknown direction'. Note that we make a distinction in the Consensus network between edges that are undirected and those that are 'unknown'. An edge that is undirected can be reversed, as in equivalent graphs. However uncertainty exists over the direction of an 'unknown' edge, or whether it can be reversed. We flag up 'unknown' edges in the graphs by using edges that are directed both ways, whereas undirected edges have no arrowheads.

5.3.2 Bayesian network meta-analysis

Meta-analysis refers to a set of statistical methods for combining the result of several studies that address a set of related research hypotheses. Meta-analysis originated in medical statistics (Sutton *et al.*, 2000) but recently has been used to identify highly expressed genes across multiple microarray datasets (Conlon *et al.*, 2006; Hu *et al.*, 2006). In medical statistics, meta-analysis is used to combine outcome measures such as incidence rates (e.g. the rate at which new cases of a disease occur in a population) from multiple medical studies.

We have developed an approach called Bayesian networks meta-analysis[1] (see Algorithm 2) that uses the fixed-effects meta-analysis method to combine the confidence levels for each edge over a set of bootstrapped networks, producing

[1]Bayesian network meta-analysis should not be confused with Bayesian meta-analysis, which involves using Bayesian models to perform the meta-analysis.

a single network that has an aggregated confidence level attached to each edge. The fixed-effects model assumes no heterogeneity between study results. Whilst this is obviously a naïve assumption, we find that it produces better results for this application than the more complicated random-effects model that accounts for study heterogeneity.

```
Bayesian Networks Meta-Analysis
Input: Set of $n$ individual networks with confidences attached to each
       edge (e.g. bootstrapped networks)
Output: Meta-analysis network with aggregated confidence levels
        attached to each edge

for each edge from node $i \rightarrow j$ (denoted $e_{ij}$) do
    let $T_{ij}(k)$ be the confidence level for edge $e_{ij}(k)$ in the $k$th network.
    Calculate the aggregated confidence level $\bar{T}$ for edge $e_{ij}$
    using $log(\bar{T}) = \frac{\sum_{k=1}^{n} w_k \log(T_{ij}(k))}{\sum_{k=1}^{n} w_k}$
    where $w_k = d_{ij}(k)$, the number of networks learnt during bootstrapping
    where an edge $i \rightarrow j$ exists
end
```

Algorithm 2: Bayesian Networks Meta-Analysis

The general fixed effect model for meta-analysis is the inverse variance-weighted method (DerSimonian & Laird, 1986). Each study outcome measure is given a weight that is inversely proportional to its variance. For n independent studies, let T_i be the observed outcome measure with variance v_i and weight w_i. Then, an estimate of an aggregate outcome measure, given all studies, is calculated as follows:

$$\bar{T} = \frac{\sum_{i=1}^{k} w_i T_i}{\sum_{i=1}^{k} w_i} \qquad \text{where} \qquad w_i = \frac{1}{v_i}$$

In BN meta-analysis, we define the study outcome measure as the confidence level estimates that are attached to each network edge. Thus, the fixed-effect meta-analysis model is applied to every network edge to obtain its combined confidence level estimate. We treat the confidence level as an incidence rate (i.e. the proportion of networks in which a particular network edge exists). For each input dataset, if the bootstrap approach is run m times resulting in m networks, then the confidence level, or incidence rate, for a particular edge e_{ij} that runs from node i to node j is $\frac{d_{ij}}{m}$ where d_{ij} is the number of networks where e_{ij} exists. Then, we define the outcome measure as the log incidence rate and its approximate variance (Sutton *et al.*, 2000) as:

$$\log(T_{ij}) = \log(d_{ij}/m), \qquad var(\log(T_{ij})) = \frac{1}{d_{ij}}$$

This means that the meta-analysis weight is defined as:

$$w_{ij} = \frac{1}{v_{ij}} = d_{ij}$$

This type of meta-analysis is essentially a weighted averaging technique where edges are weighted using their own confidence level. Thus, edges with high confidences are strongly weighted and more likely to have a high confidence level in the final Meta-analysis network.

Similarly to Consensus Bayesian networks, bootstrapping is used to generate the input individual networks that have confidences attached to each edge. However, in contrast to the Consensus method, Bayesian network meta analysis does not require thresholding of the input networks to obtain PDAGs, since it directly combines the confidences attached to each edge. However, the output meta-analysis network can be thresholded (using the same method that is described in Section 3.3.4 for bootstrapped networks) to obtain a PDAG — and this is what we do in order to evaluate our meta-analysis networks.

5.3.3 Related work

Previous research on combining BNs generally falls into two broad categories — qualitative and quantitative combination. Quantitative combination is based on aggregating the probability distributions in the networks (Pennock & Wellman, 1999). Qualitative combination has been referred to as *topological fusion* (Matzkevich & Abramson, 1992). This is based on combining the graphical structures of multiple networks using graph union (i.e. combining all edges in all networks). Since graph union can introduce cycles into the network structure, arc reversal is also used. This is where an arc $A \rightarrow B$ is reversed and then arcs are added between the parent nodes of A to B, and from the parent nodes of B to A. This maintains the underlying relationships between variables under the principle that it preserves the flow of information (Shachter, 1986). The final fused graph contains all arcs (some reversed) and nodes that are in the input DAGs.

In comparison to topological fusion, Consensus Bayesian networks focuses on graph intersection rather than union and also accounts for network equivalence classes which are not considered in topological fusion. At the consensus threshold $\frac{1}{n}$ (that corresponds to every edge from each of the n networks appearing in the combined structure), the Consensus approach is equivalent to graph union. However, the topological fusion network does not do as well as a $1/n$ Consensus network, as it is liable to the inclusion of misdirected edges. The key difference is that the Consensus method represents networks using equivalence classes — so if edges are reversible they are left undirected.

Specifically in microarray data analysis for learning GRNS, there is no research on combining BNs, however Wang *et al.* (2006) use a post-learning aggregation framework where gene networks are represented using non-linear differential equations to combine multiple microarray gene expression datasets.

5.4 Comparison of pre- and post-learning aggregation methods

In this section we report on the experiments performed to evaluate the use of the post-learning aggregation Consensus and Meta-analysis approaches on multiple microarray datasets and compare them to the use of pre-learning aggregation (combining the datasets and using scale normalisation prior to the generation of a network from the concatenated dataset) and the performance of the individual input networks, each generated from a single microarray dataset. Initial experiments were carried out on a set of four datasets for a synthetic network of 13 genes. We then progressed to two real applications: *E. coli* and yeast sub-networks.

5.4.1 Comparison procedure

For each application, every input dataset was scale-normalised and a network with confidences attached to each edge was learnt (using a bootstrapping approach with $m = 50$ iterations). The following aggregate networks were constructed:

- A single Meta-analysis network was constructed, where each edge has an attached confidence level.
- Multiple sets of Consensus networks were generated, where each set corresponds to a different bootstrap confidence threshold (0.1 to 0.9, at steps of 0.1) for the input networks generated from each dataset. This means that the input bootstrapped networks were all thresholded at the same value to form PDAGs, and these formed the input to the Consensus method. Each set of Consensus networks contains networks generated for each consensus threshold from 0 to 1, at steps of $1/n$ (where n is the number of datasets).
- A bootstrapped network was generated from a concatenated and scale-normalised dataset (referred to as the Normalisation Only network).

As described in Section 3.4.1, the learnt networks are evaluated by comparing them to documented gene interactions. These were obtained from various sources according to the application. Whilst the synthetic network was fully known, *E. coli* regulatory interactions are documented in the online database RegulonDB Salgado *et al.* (2006) and yeast interactions (both confirmed and potential) are listed in the YEASTRACT database Teixeira *et al.* (2006).

For each network type a ROC curve is plotted, where each point corresponds to a confidence level or a consensus threshold. An AUC value for the network is calculated based on this ROC curve. For the Meta-analysis and Normalisation Only networks each point of the ROC curve refers to the TP and FP rates of the PDAG extracted from the network at different bootstrap confidence thresholds (from 0 to 1 at steps of 0.1). For Consensus networks, each point of the

Dataset	Number of Observations
1	200
2	400
3	600
4	600

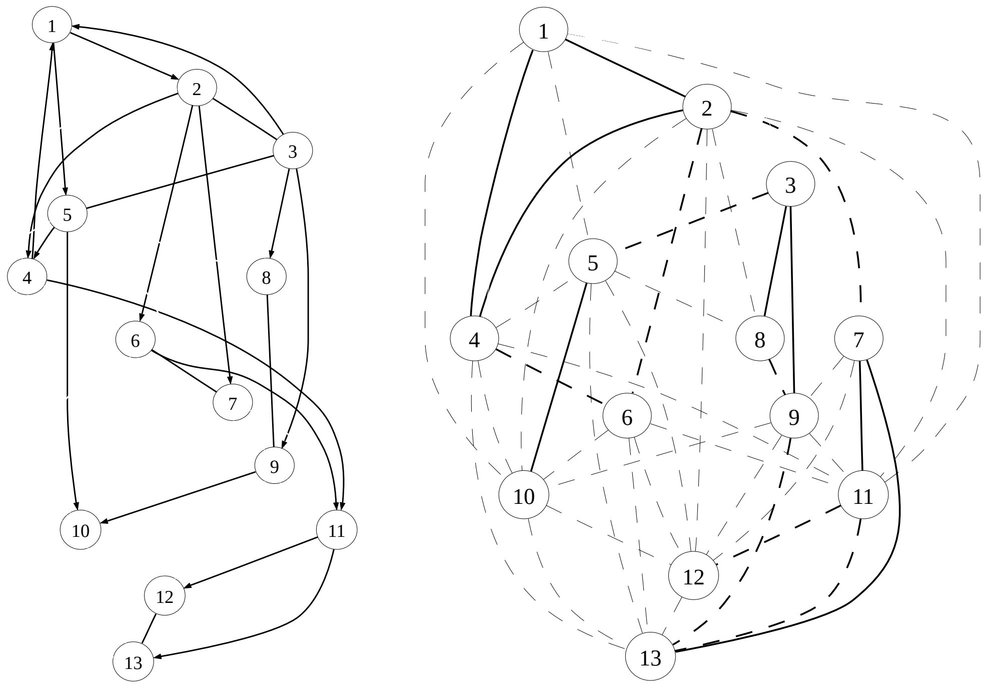
Table 5.1: Summary of synthetic datasets

ROC curve refers to the TP and FP rates of the Consensus network at different consensus thresholds (from 0 to 1, at steps of $1/n$). This means there are multiple ROC curves for the Consensus approach, each one constructed for a set of input networks obtained from a different bootstrap threshold. Since the Meta-analysis approach directly combines bootstrap confidences, and there is no initial thresholding step as for the Consensus approach — it has one ROC curve only.

In order to obtain statistical estimates on the significance of the results, we ran this process several (15) times for each dataset. Thus, mean TP and FP rates (in order to estimate a mean ROC curve) and AUC measurements were obtained for each method. Then a paired t-test was used to compare the relative performances of the different approaches and measure whether the differences between their mean AUCs are statistically significant.

5.4.2 Application: synthetic network

The synthetic regulatory network consists of 13 genes as shown in Figure 5.1a. Four time-series expression datasets were generated for the network using differential equations to mimic a transcriptional network. The change of the expression of each gene is determined by a function composed of three parts: activation by a single other gene, repression by a single other gene and decay. Each of these parts has one parameter, which is (uniformly) randomly selected from a predefined range. Each dataset generated varies because the parameters for activation, inhibition and decay are chosen randomly for each gene, the predefined range of these parameters may vary, the perturbations vary and other parameters of the simulation (such as the length of the time lag) may also vary. Each dataset had a varying number of samples ranging from 200-600, as detailed in Table 5.1.



(a) True network (b) Consensus network

Figure 5.1: Synthetic regulatory networks. (a) True network. (b) Synthetic consensus network. Edges are shaded or marked according to robustness (i.e. their consensus threshold c) — bold edges obtain a high consensus threshold ($c \geq 0.75$). Bold and dashed edges have $0.50 \leq c < 0.75$, whereas the dashed (only) edges have $c \leq 0.25$.

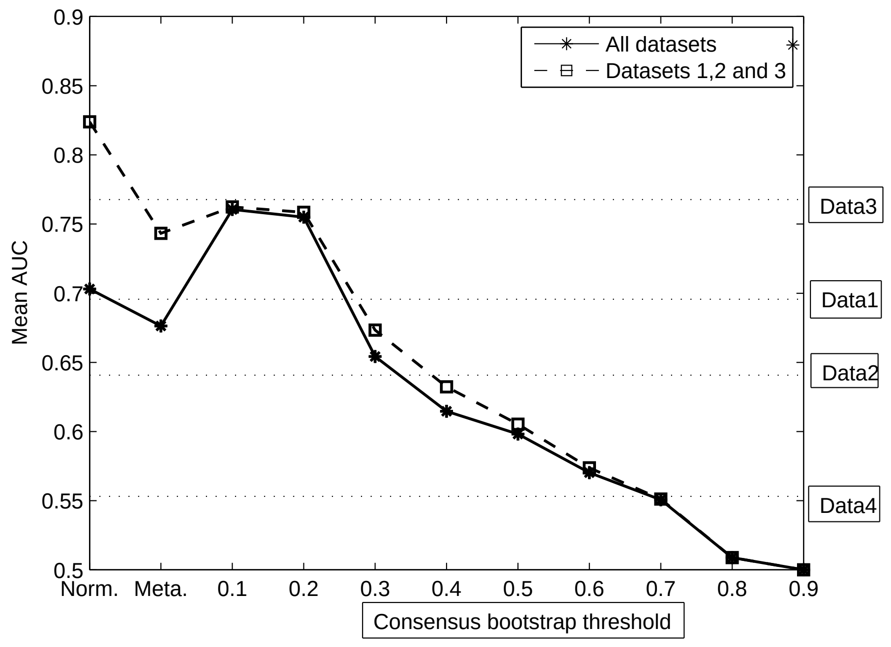


Figure 5.2: AUC performance of learnt synthetic networks

Figure 5.2 compares the difference in the mean AUC for each aggregation approach against each other and against the mean AUC of each individual dataset network (that are shown using horizontal lines). We also compare the combination of all datasets against the combination of a subset of the datasets (where the subset is chosen based on the performance of the networks). We refer to the networks generated by datasets 1-4 as Data1, Data2, Data3 and Data4 respectively, whilst the datasets themselves are referred to as dataset 1, dataset 2, dataset 3 and dataset 4.

Figure 5.2 shows that the Consensus approach performs best on the set of individual PDAGs extracted using a bootstrap threshold of 0.1. In this case the approach obtains a mean AUC of 0.76 (for a ROC curve that is obtained from set of Consensus networks, for Consensus thresholds from 0 to 1, at steps of 1/4 since there are $n = 4$ datasets). According to the paired t-test, this Consensus network set outperforms 3 of the 4 individual networks (Data1, Data2 and Data4), as well as the Normalisation Only and Meta-Analysis networks with statistical significance ($p < 0.01$). Meta-analysis, which obtains a mean AUC of

0.68, and Normalisation Only (obtaining a mean AUC of 0.70) only significantly outperform Data2 and Data4.

By selecting a consensus threshold we can obtain a single network structure from a set of Consensus networks. For example, a bootstrap threshold of 0.1 for the input networks, with a consensus threshold of 1.0 (where every edge in the Consensus network must appear in all input networks) provides the best TP and FP rates, which are 0.50 and 0.07 respectively (network not shown). In other words, it is able to identify half of the edges in the true network with a fairly low FP rate.

For the Consensus and Meta-analysis approaches, the robustness of an interaction can be identified using the confidence level or consensus threshold attached to its edge. The 'robustness' of an edge in a Consensus network indicates in how many datasets it is found. Thus we can view a set of Consensus networks as a single network with each edge having a consensus threshold, or as a set of networks, each generated at a different Consensus threshold. The 'robustness' attached to a Meta-analysis edge is slightly different, as it incorporates the original bootstrapped confidences. In this case it represents the strength of the edge's confidence over all the individual input networks. This is particularly useful for visualisation of the learnt networks. Figure 5.1b shows the learnt Consensus network (obtained from input networks thresholded at a confidence threshold of 0.1) with edges shaded according to their consensus threshold. It can be seen by eye there is a relationship between the more robust edges and the true network (shown in Figure 5.1a).

Figure 5.2 shows that Data1, Data2 and Data3 are much closer to the true network than Data4 (since they have higher AUC values). Data4 contains many FP edges. Upon closer inspection of dataset 4, we find that its randomly selected time lag length parameter is much larger than for the other datasets, perhaps explaining why the performance of Data4 is weaker. To eliminate the influence of dataset 4 we ran the Normalisation Only, Meta-analysis and Consensus approaches on datasets 1-3 only. Over the three datasets, Normalisation Only and Meta-analysis perform much better, their mean AUC increasing to 0.82 and 0.74 respectively. In fact, Normalisation Only outperforms all other networks with statistical significance $p < 0.01$, whereas the Consensus and Meta-analysis approaches are still unable to significantly outperform Data2. The difference between the performance of the Consensus and Meta-analysis approaches is no longer statistically significant. Since Data4 contains FP edges with high bootstrap confidences, Meta-analysis and Normalisation Only perform far more reliably when dataset 4 is removed from the input. By comparison, the Consensus approach is not so greatly affected by the removal of Data4 (see Figure 5.2). Since the Consensus approach identifies consistencies across the set of individual dataset networks, it is able to discard the false positives introduced by Data4.

5.4.3 Application: *E. coli* SOS response network

The second application considered is an example of a single transcriptional module in *E. coli* - an SOS repair system. The module consists of approximately 30 genes and one repressor TF, LexA. UV irradiation and other DNA damaging agents are known to trigger the induction of the stress-related SOS response, a coordinated increase in the level of expression in the set of genes, which is negatively regulated by LexA (Quillardet *et al.*, 2003). We selected a number of these genes (based on data availability) to form a sub-network (see Figure 5.3). Table 5.2 provides a summary of the four selected datasets, which are all focused on experiments related to SOS response. The datasets each originate from different research groups and microarray platforms including both 2-channel technology (cDNA microarrays) and single-channel arrays (Affymetrix olignucleotide microarrays). For the Affymetrix data, in order to create an equivalent to cDNA microarray log ratio values, we subtracted the average log expression level of a gene from one experiment from the log expression level for that gene in a given experiment, allowing comparisons of different genes to each other.

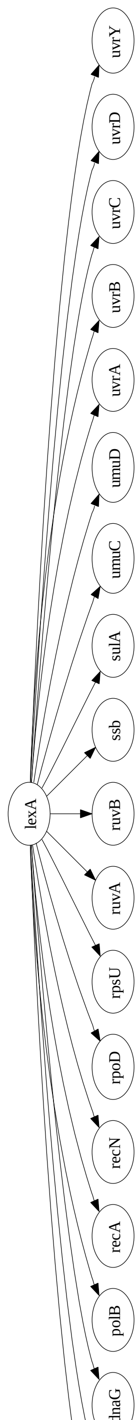


Figure 5.3: *E. coli* SOS response transcriptional module: true network structure

Dataset	Description	Platform	No. Observations
Courcelle *et al.* (2001)	UV irradiation	cDNA	15
Faith *et al.* (2007)	Various	Affymetrix	254
Khil *et al.* (2002)	DNA damage	cDNA	8
Sangurdekar *et al.* (2006)	Various inc. UV irradiation	cDNA	240

Table 5.2: Summary of *E. coli* datasets

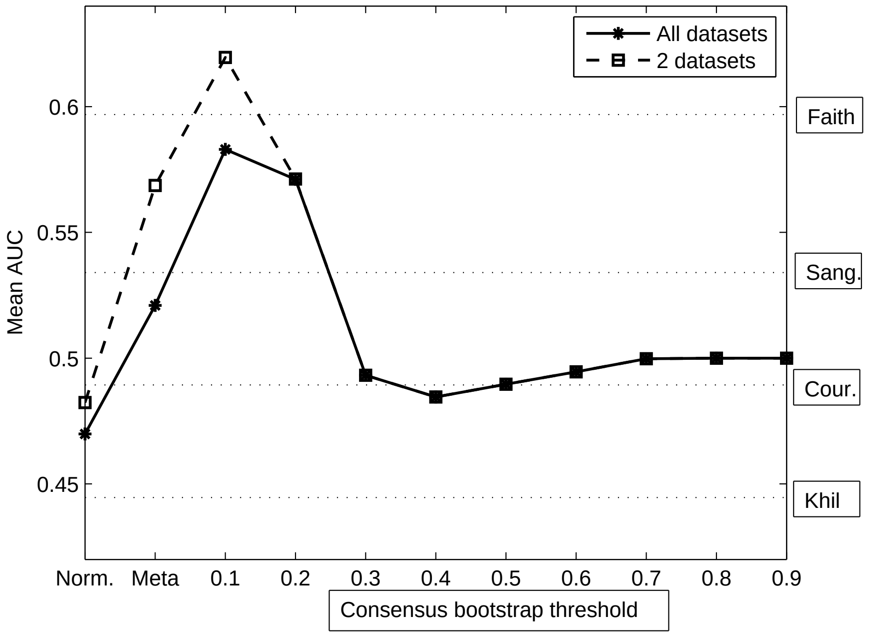


Figure 5.4: AUC performance of learnt *E. coli* networks

Figure 5.4 compares the difference in the mean AUC for each aggregation approach against each other and against the mean AUC of each individual input network (that are shown using horizontal lines). We also compare the combination of all datasets against the combination of a subset of the datasets (where the subset is chosen based on the performance of the input networks).

Figure 5.4 shows that the Consensus networks generated from sets of input networks thresholded at lower bootstrap confidences perform most successfully of the aggregation approaches (the best results are obtained with a bootstrap confidence threshold of 0.1). In this case the Consensus approach obtains a mean AUC of 0.58, outperforming three of the four individual input networks and the Normalisation Only and Meta-analysis approaches (with statistical significance $p < 0.01$). The low bootstrap threshold may be explained by the fact that there are very few edges with a high confidence level (e.g. over 0.5 or 0.6) and these only occur in the Faith and Sangurdekar networks, for which the datasets contain a larger number of observations. Meta-analysis obtains a mean AUC of 0.52 (significantly outperforming only one of the four individual networks), whilst the mean AUC for Normalisation Only is just 0.47 and it is significantly outperformed by two of the individual dataset networks.

We believe that the nature of the SOS module plays a part in the high number of FP edges and relatively low AUC, in comparison to the results on synthetic data. It is a sparse network — in fact a Naïve Bayes model — and so all variables are correlated, becoming independent conditional on the regulator LexA. This makes it more difficult to identify spurious interactions. Figure 5.5 shows a Consensus network (with a consensus threshold of 1.0 and generated from input PDAGs calculated at bootstrap confidence threshold of 0.1). Whilst interactions between LexA and six genes are found, there are many other discovered interactions — e.g. the UVR family are obviously related. In previous experiments on the Courcelle dataset we were able to identify the regulator LexA consistently from a group of candidate transcription factors (regulator genes) for each target gene using BNs (Peeling *et al.*, 2007). However, identifying the regulator when choosing from within a group of correlated genes is far more challenging. This of course also has a bearing on the calculation of FP edges between the learnt models and the 'true' network. In addition, it is likely that the 'true' network is in fact incomplete, which assists in explaining why the absolute performance of all input networks is much lower in comparison to the synthetic data experiments.

Similarly to the synthetic data, some datasets perform better than others. In this case, the networks generated from datasets with relatively small numbers of observations — Courcelle and Khil — perform more weakly, their networks obtaining AUCs of 0.49 and 0.44 respectively. We ran Normalisation Only, Meta-analysis and Consensus on the Faith and Sangurdekar networks only. This improved the results for the Consensus approach, increasing the mean AUC to

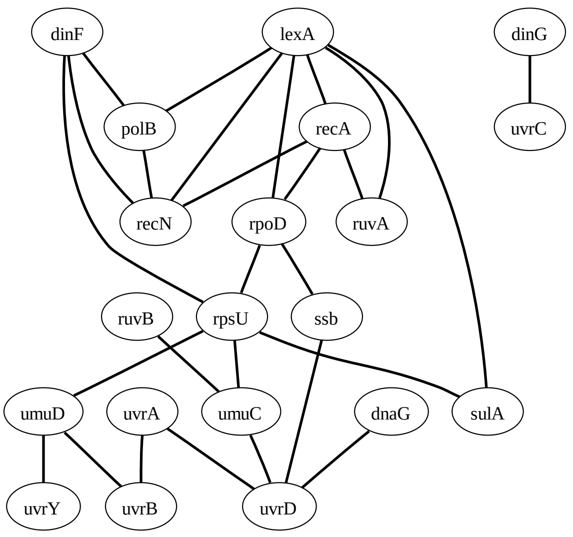


Figure 5.5: *E. coli* consensus network generated from 2 datasets (0.1 bootstrap threshold) — 1.0 consensus threshold (all edges appear in both datasets)

0.62. It outperforms both the Faith and Sangurdekar networks with $p = 0.025$. Meta-analysis also makes an improvement, the mean AUC increasing from 0.52 to 0.57, but is unable to outperform the Faith network.

On synthetic data (especially on the three 'best' datasets), the simple Normalisation Only approach produced one of the best performing networks. However on the *E. coli* data, the Normalisation Only approach does not obtain such successful results. In fact, the Normalisation Only networks are the worst performing networks, and do worse in terms of AUC than 3 of the individual dataset networks. However, this may be explained by the fact that the synthetic data are not generated to contain any experimental or platform biases whereas these are inherent in the real *E. coli* data.

5.4.4 Application: yeast heat stress network

We take the example of 9 transcription factors (TFs) related to heat-shock response from Wang *et al.* (2006) in order to evaluate the algorithm on a subnetwork of a manageable size and make a comparison between the two methods[1].

[1]In Wang *et al.* (2006) they use a network of 10 TFs. We remove the gene SOK2 due to the many missing values in some of the datasets.

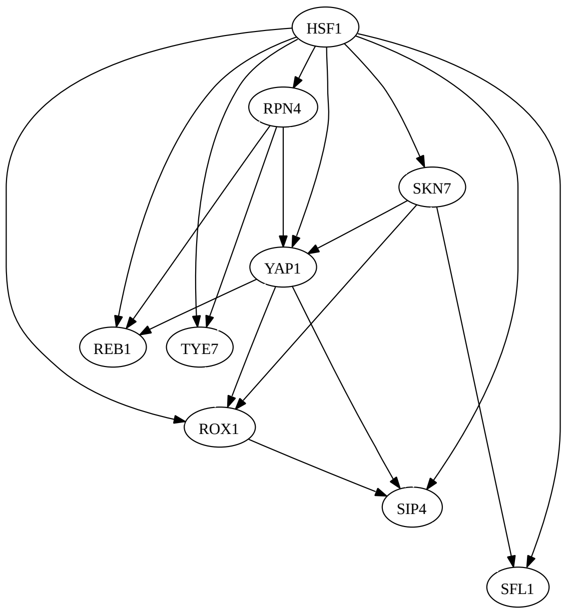


Figure 5.6: Yeast heat-stress regulatory network: true network structure

Two of the TFs selected (HSF1 and SKN7) are known to be directly involved in heat-shock response and are documented as regulating 4 TFs among the 9. The sub-network is shown in Figure 5.6. We use microarray datasets that are publicly available on the YeastBASE expression database. Most selected datasets are from studies that include heat-shock response experiments — see Table 5.3.

Figure 5.7 compares the difference in the mean AUC for each aggregation approach against each other and against the mean AUC of each individual dataset network (that are shown using horizontal lines). As before, we also compare the combination of all datasets against the combination of a subset of the datasets (where the subset is chosen based on the performance of the input networks).

Once again, the Consensus network set (generated from all input networks at a low bootstrap confidence threshold of 0.1) obtains the best results of the aggregating approaches, outperforming all individual input networks, obtaining a mean AUC of 0.53. Using the paired t-test, we find this network set outperforms 3 of the 5 individual input networks with statistical significance $p < 0.01$. The Meta-analysis and Normalisation Only networks obtain mean AUCs of only 0.46 and 0.47 respectively. They are significantly outperformed by the Consensus

Dataset	Description	Platform	No. Obs
Beissbarth *et al.* (2000)	Heat-shock response	cDNA	12
Eisen *et al.* (1998)	Cold-shock and heat-shock response	cDNA	14
Gasch *et al.* (2000)	Environmental changes inc heat-shock response	cDNA	173
Grigull *et al.* (2004)	Heat-shock response	cDNA	27
Spellman *et al.* (1998)	Cell-cycle	cDNA	73

Table 5.3: Summary of yeast datasets

network set and three of the five individual input networks.

Comparison of the AUC for each individual input network shows that three of the networks perform noticeably poorly. If we remove these networks from the input to the algorithms we find a marked improvement for all aggregation approaches (see Figure 5.7). The Consensus approach obtains the best results, with a mean AUC of 0.55 whilst the individual networks for the Gasch and Spellman datasets obtain mean AUCs of 0.53 — a statistically significant difference with $p = 0.10$. In this case, we find the best Consensus networks are generated when the input PDAGs have been obtained by thresholding the bootstrapped networks at relatively higher thresholds of 0.3 - 0.4. This is because the Gasch and Spellman networks have higher confidences attached to their edges than the networks generated from the other three datasets. The Meta-analysis and Normalisation Only approaches also show an improvement, so much so that there is no statistically significant difference in the AUC for the networks generated by them and the Consensus approach.

In Figure 5.7, we find that there is a significant dip in AUC at the 0.2 bootstrap threshold Consensus network. This is explained by that fact that there is a peak in edge confidences between 0.1 and 0.2 in the individual input networks (data not shown). Whilst a 0.2 thresholded individual PDAG includes the same edges as a 0.3 PDAG, a lower threshold means that more FP edges may be included, causing the AUC to decrease. Similarly, lowering the threshold from 0.2 to 0.1, more edges are included, but in this case they are TP edges, causing an increase

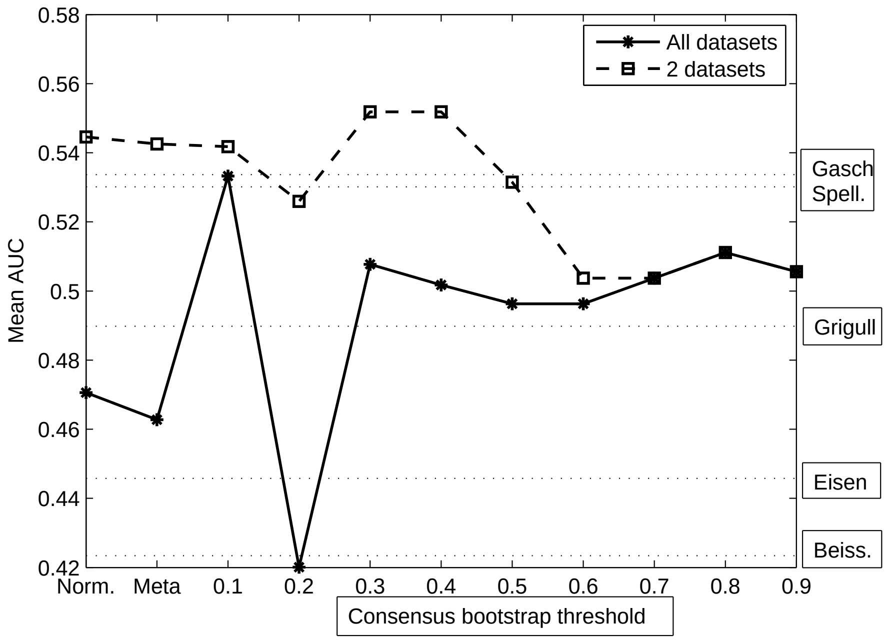


Figure 5.7: AUC performance of learnt yeast networks

in AUC.

In comparison to the work by Wang *et al.* (2006), both the Consensus and Meta-analysis networks are more successful based on our performance criteria. The Wang *et al.* network obtains a TP rate of 0.17 and a FP rate of 0.75. In comparison, our Consensus networks (from all networks with a bootstrap threshold of 0.1) obtain mean TP and FP rates of 0.58 and 0.54 respectively at a 0.8 consensus threshold and 0.16 and 0.09 at a 1.0 consensus threshold. Figure 5.8 shows such a Consensus network (0.8 consensus threshold) that contains 13 TP edges and 7 FP edges. This network shows which edges are more robust (i.e. found in more individual input networks). We should also point out that Wang *et al.* only use some of the time-series in the Gasch dataset to generate their aggregate network, whereas our Consensus network is generated from a broader set of studies.

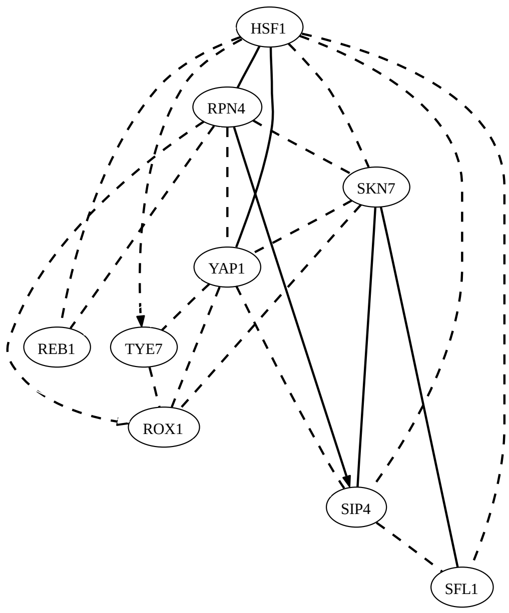


Figure 5.8: Yeast consensus network generated from all datasets (0.1 bootstrap threshold) — 0.8 consensus threshold

5.5 Discussion

The purpose of this chapter has been to investigate whether post-learning aggregation for generating GRNs from multiple microarray datasets (that is, learning models from each dataset and combining the models) can produce better results than concatenating the datasets after scale normalisation and then learning the model — a simple pre-learning aggregation method. We have presented two novel post-learning aggregation approaches for combining multiple microarray datasets to generate GRNs and compared them against a simple pre-learning aggregation approach, as well as the performance of the individual input networks that have been generated from each dataset.

Each of the new approaches is based on aggregating high-level features of BN models that have been generated from a set of individual microarray datasets. Thus, they possess the benefits of post-learning aggregation approaches, meaning they can be used to combine datasets generated by different platforms, research groups and laboratories and do not necessarily require normalisation of the datasets, which can be complicated on cross-platform microarray datasets.

Meta-analysis BNs combine confidence levels attached to network edges using an inverse-variance weighted method whilst Consensus BNs identify regulatory interactions that are found consistently across all datasets. Both methods produce networks with a measure of ‘robustness’ attached to each edge, which in a Consensus network indicates in how many datasets it is found. The ‘robustness’ attached to a Meta-analysis edge is slightly different, as it incorporates the original bootstrapped confidences. In this case it represents the strength of the edge’s confidence over all the individual input networks.

The pre- and post-learning aggregation approaches were compared with each other as well as against the performance of the individual input networks. On clean, unbiased synthetic data a simple pre-learning aggregation approach (the Normalisation Only network) performs very well — significantly outperforming both Consensus and Meta-analysis networks and the individual input networks. However, on real data that is biased and generally noisier, this did not hold. In fact, Normalisation Only often performed worse than many of the networks generated from a single dataset. On *E. coli* data, we found that Meta-analysis and Consensus networks both provided a significant improvement over Normalisation Only. In particular, the Consensus approach increased the AUC by over 0.1. On the yeast sub-network, the absolute increase in AUC was not as great, but was still statistically significant. Thus, on the basis of the experiments presented in this chapter, post-learning aggregation does provide an advantage over concatenating normalised datasets for learning from multiple *real* microarray datasets.

Whilst Consensus and Meta-analysis outperform Normalisation Only when learning from multiple microarray datasets, we also found that unless the worst-performing datasets were removed, the networks produced by post-learning aggregation approaches did not always outperform all the individual input networks. This leads to the question, is there a benefit to learning from multiple microarray datasets if the combined models do not outperform all individual dataset models? We believe so. When little is known about the datasets, post-aggregation learning can be used to identify the more robust and consistent interactions across datasets and filter out noisy and spurious relationships. The Consensus approach identifies consistencies amongst the collection of datasets and so it is least affected by poorly performing input networks. On the other hand, since Meta-analysis is a weighted-averaging technique, where edges with a high confidence level are given more influence, it can work well with only one well-performing dataset as the influence of lower confidence edges is weak. Conversely, however, its performance can be easily influenced by a single dataset that contains false positives and negatives with high confidence levels.

Since the Consensus approach is more robust to poorly performing input networks and is therefore more consistent in the resulting network performance, we conclude that it provides the greatest benefits for combining multiple microarray datasets. However, there is room for improvement in the method. For example, it would be desirable to reduce the number of parameters. When it is used in conjunction with bootstrapping to learn the input networks, the user is required to choose a bootstrap and a consensus threshold (although the final network can be viewed with edge 'robustness' rather than choosing a consensus threshold). In comparison, Meta-analysis is relatively simpler and 'parameter-free', since the bootstrap confidences are directly used to compute the aggregated network (however, if the user wishes to extract a PDAG, a threshold must be chosen).

Additionally, it was found that the datasets which generated the worst performing networks were generally those with a small number of samples (at least, in the case of real data). Including these datasets with a small number of samples can actually have a negative effect by shifting focus from a larger dataset. Therefore it may be advantageous to only accept datasets that are more reliable or of higher quality (e.g. those containing a larger number of samples), or at least to lessen the influence of lower-quality datasets. In the following chapter these issues are addressed with a generalised Consensus approach that requires fewer parameters, together with input dataset/network selection and weighting which allows different strengths of influence to be assigned to each input dataset/network.

Chapter 6

Incorporating network reliability

6.1 Introduction

The previous chapter presented Consensus Bayesian networks, an approach for combining network models generated from different microarray gene expression datasets. Although experimental results on real microarray datasets indicated that a Consensus model produced from a set of datasets is more accurate than a model generated from a single dataset, the method does have shortcomings. It is a parameter-heavy method, relying on the user to select confidence thresholds for bootstrapped input networks and the overall consensus threshold for the output network. Furthermore, the experimental results also indicated that in some cases, using only a subset of available datasets can produce a better Consensus model than when using all available datasets. This chapter addresses these limitations by presenting the following contributions:

- An improved Consensus approach with reduced parameters — Consensus Bootstrapped Bayesian Networks (CBBNs)
- A simple measure of network reliability
- The incorporation of network reliability measures into the Consensus method in order to take into account the reliability or quality of each input network

The Consensus approach is extended in two different ways. Firstly, a generalised Consensus approach is presented that can act directly on input networks with confidences attached to each edge. This means that the more robust bootstrapped network models (rather than PDAG models produced by thresholding a bootstrapped network) can be used as direct inputs to the Consensus algorithm. As well as removing a level of parameters (i.e. the selection of a confidence level at which to threshold the bootstrapped network) this should improve the reliability and performance of the output Consensus model by allowing all the information in the bootstrapped network to be used.

Second, two methods are introduced for incorporating network reliability measures into the Consensus approach, in order to place more influence on certain input networks. Experimental results presented in chapter 5 found that the reliability of networks varies across the datasets from which they are generated. For example, the performance of the Consensus network can be better when it is generated from networks for a subset of the datasets available, rather than the total set, especially if some datasets are particularly noisy or biased. In order to address the question of whether the use of reliability measures can improve the final Consensus model, this chapter considers how to assess the reliability of networks and compares different weighting- and selection-based methods for varying the influence of input networks with the Consensus method.

The remainder of the chapter is organised as follows. In Section 6.2 the extended Consensus approach is explained in more detail. Following this, in Section 6.3, methods for measuring the reliability of networks are considered. Further to this, two methods are introduced for including network reliability measures in the Consensus method. This section also includes a brief literature review on considering bias when learning or modelling from multiple datasets. A comparison of the different methods for incorporating network reliability is presented in Section 6.4. Experimental results on synthetic and real data are included and discussed. Finally, Section 6.5 discusses the overall conclusions from this chapter.

6.2 Consensus Bootstrapped Bayesian Networks

This section presents a generalised Consensus approach, Consensus Bootstrapped Bayesian networks (CBBNs), that can act directly on input networks with confidences attached to each edge. In the application we consider, these confidences are generated using a bootstrapping approach. The output Consensus network, at a given consensus threshold, is also a network with confidences attached to each edge. This means that the user can select a level of robustness (consensus threshold) for the Consensus network, and then the resulting output network provides a confidence for the existence of each edge. Since the generalised approach is based on networks with edge confidences, it also removes the problem of assigning uncertain edge directions from the original algorithm. Section 6.2.1 describes the algorithm in detail whilst the following Section 6.2.2 explains how weighting can be incorporated into the method to allow each input network to have a different influence on the final Consensus network. Section 6.2.3 compares the original Consensus approach with the new algorithms detailed here, highlighting the key differences.

6.2.1 Algorithm

The CBBNs approach takes as input a group of bootstrapped networks and a consensus threshold t, and as output it returns a Consensus network which is of the same form as the input networks — each edge has a confidence attached. In the input bootstrapped networks, the confidence attached to each edge is referred to as the *input confidence*. In the Consensus network we refer to the confidence level attached to each edge as the *consensus confidence*.

The algorithm proceeds as follows. For the input set of bootstrapped networks, the edge between each pair of nodes is considered in turn. For each possible edge a set C of input confidences is created, which contains the input confidences for that edge from each input network. Next, we order the input confidences in C from highest to lowest. Then a subset $C_{max} \subseteq C$ is created, where C_{max} contains the highest input confidences in C, and the size of this subset is based on the consensus threshold t and the number of input networks n such that $|C_{max}| = floor(nt)$. Recall that the consensus threshold t is between 0 and 1 (and can be thought of as a percentage[1]), so $|C_{max}| \leq n$. Then, we define the *consensus confidence*, the confidence for each edge in the final Consensus network at consensus threshold t, as the minimum input confidence in the subset C_{max}. In this way, the consensus confidence is taken from the set of input confidences. This makes the process analogous to the original approach. In the original approach, an edge exists in the Consensus network if it appears in a proportion of the input networks that exceeds the consensus threshold. In the new approach, an edge's consensus confidence is defined by the minimum input confidence in the input networks with the highest input confidences, where the proportion of these input networks exceeds the consensus threshold. By following the algorithm described for each possible edge in the network, we build a Consensus network where each edge has an attached consensus confidence. A Consensus network can be formed for each consensus threshold t. The algorithm is detailed step-by-step in Algorithm 3.

This means that for a set of n networks, a $1/n$ consensus threshold would lead to the consensus confidence for each edge to be the maximum input confidence for that edge across all input networks (i.e. the maximum input confidence in the set C). Conversely, under a 1.0 (100%) consensus threshold, the consensus confidence for each edge would be the minimum input confidence. In general, a low consensus threshold leads to higher consensus confidences in the Consensus network, whilst a high consensus threshold leads to lower consensus confidences in Consensus network. This is analogous to the original Consensus approach, where under a $1/n$ consensus threshold, an edge need only exist in one of the input networks to be included in the Consensus network, so there are likely to be more edges in the Consensus network than under a 1.0 consensus threshold, where an edge in the Consensus network must appear in every input network. In this respect, the benefit of the new CBBNs approach over the original approach is that a consensus confidence is applied to each edge in the Consensus network, which provides more information on the reliability of each edge in the Consensus network. For example, if a particular edge attains a high input confidence in

[1]Note that in this chapter, the consensus threshold may be referred to as a number between 0 and 1, or the equivalent percentage, which can be a more intuitive form of notation in this instance

```
***Consensus Bootstrapped Bayesian Networks***
**Input**: Set of $n$ bootstrapped networks, consensus threshold (between 0
        and 1)
**Output**: Consensus network

**for** *each pair of nodes i,j* **do**

    1. Create set $C = \left\{c_{ij_1}, ..., c_{ij_n}\right\}$ where $c_{ij_k}$ is the input confidence
       for the edge between nodes $i \rightarrow j$ in the $k$th input network

    2. Reorder and re-index the input confidences in $C$ from highest
       to lowest where $c_{ij_1}$ is the highest input confidence and $c_{ij_n}$ is
       the lowest

    3. Create subset $C_{max} \subseteq C$ such that $A = |C_{max}| = \text{floor}(nt)$ and
       $C_{max} = \left\{c_{ij_1}, ..., c_{ij_A}\right\}$

    4. Define the edge consensus confidence as $Con_{ij} = \min(C_{max})$

**end**
```

Algorithm 3: Consensus Bootstrapped Bayesian networks

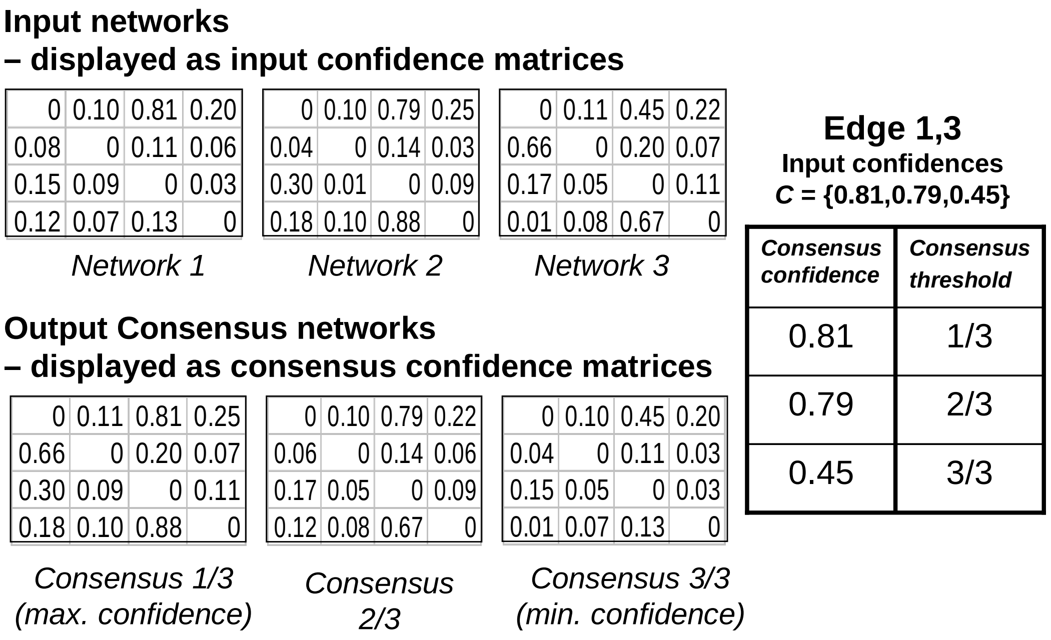
each input network, then it will still attain a high consensus confidence in the Consensus network at a low consensus threshold.

The CBBNs process on an example set of input networks is shown in Figure 6.1. On the top line are the three input bootstrapped networks, displayed as matrices of edge input confidences between node pairs. The bottom line shows three Consensus networks, each for a different consensus threshold ($t = 1/3$, $t = 2/3$ and $t = 1$). The Consensus networks are also displayed as matrices of the edge consensus confidences. We can see that in the Consensus network for the lowest consensus threshold $t = 1/3$, each edge has the maximum input confidence for that edge across all input networks. In the Consensus network for the middle consensus threshold $t = 2/3$, each edge has the second ordered input confidence for that edge across all input networks. Finally, in the Consensus network for the highest consensus threshold $t = 1$, each edge has the minimum input confidence for that edge across all input networks. On the right of the figure, the set C of ordered input confidences across the input networks for the edge from nodes $1 \rightarrow 3$ is shown. Each confidence is linked to its corresponding consensus threshold. For example, for the Consensus network at $t = 2/3$, the size of subset C_{max} is $|C_{max}| = nt = 3 \times (2/3) = 2$. Thus $C_{max} = \{0.81, 0.79\}$ and so the consensus confidence for edge $1 \rightarrow 3$ at $t = 2/3$ is $\min(C_{max}) = 0.79$.

6.2.2 Incorporating weights

In the CBBNs method described in the previous section each input network has an equal influence in the Consensus algorithm. When we incorporate weighting, each input network can be given a weight that is proportional to its influence on the final Consensus network. The calculation of input network weights is addressed in Section 6.3. For now, we assume that the weights for all input networks sum to 1. It is fairly simple to generalise the CBBNs algorithm to deal with unequal influences of each input network. The process is identical to the CBBNs approach described in Section 6.2.1, except each edge input confidence is paired with its network weight.

The algorithm proceeds as follows (also see Algorithm 4 for step-by-step details). Each edge is considered in turn. For each edge, a set C is created of input



Input networks
– displayed as input confidence matrices

Network 1

0	0.10	0.81	0.20
0.08	0	0.11	0.06
0.15	0.09	0	0.03
0.12	0.07	0.13	0

Network 2

0	0.10	0.79	0.25
0.04	0	0.14	0.03
0.30	0.01	0	0.09
0.18	0.10	0.88	0

Network 3

0	0.11	0.45	0.22
0.66	0	0.20	0.07
0.17	0.05	0	0.11
0.01	0.08	0.67	0

Output Consensus networks
– displayed as consensus confidence matrices

Consensus 1/3 (max. confidence)

0	0.11	0.81	0.25
0.66	0	0.20	0.07
0.30	0.09	0	0.11
0.18	0.10	0.88	0

Consensus 2/3

0	0.10	0.79	0.22
0.06	0	0.14	0.06
0.17	0.05	0	0.09
0.12	0.08	0.67	0

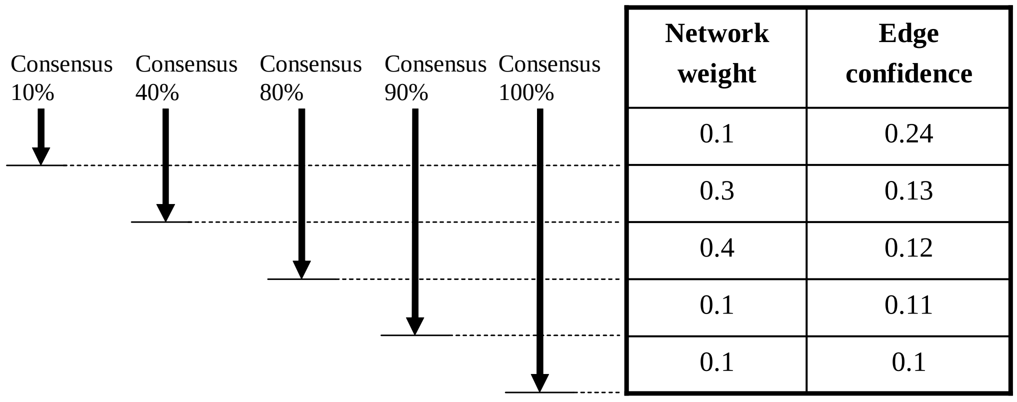
Consensus 3/3 (min. confidence)

0	0.10	0.45	0.20
0.04	0	0.11	0.03
0.15	0.05	0	0.03
0.01	0.07	0.13	0

Edge 1,3
Input confidences
***C* = {0.81,0.79,0.45}**

Consensus confidence	*Consensus threshold*
0.81	1/3
0.79	2/3
0.45	3/3

Figure 6.1: Consensus Bootstrapped Bayesian networks. On the top line are the three input bootstrapped networks, displayed as matrices of edge input confidences between node pairs. The bottom line shows the Consensus Bootstrapped BNs, also displayed as matrices of edge consensus confidences, for three consensus thresholds (from $1/n$ to 1 at steps of $1/n$ where $n = 3$). On the right, the set C of ordered input confidences from the input networks for the edge from node 1 to 3 is shown and the table associates the consensus confidence for each consensus threshold.



Network weight	**Edge confidence**
0.1	0.24
0.3	0.13
0.4	0.12
0.1	0.11
0.1	0.1

Figure 6.2: Weighted Consensus networks: deciding the consensus confidence for a single edge. The input confidences from five input networks are ordered descendingly, paired with the network weights that range from 0.1 to 0.4. The consensus confidences associated with a range of consensus thresholds from 10% to 100% (0.1 to 1.0) are indicated. For example, the consensus confidence 0.24 is associated with the lowest consensus threshold of 10%, whilst the consensus confidence 0.1 is associated with the highest consensus threshold of 100%.

confidence-network weight pairs, where the kth pair contains the input confidence for that edge in the kth input network, and the weight w_k of the kth input network. The pairs in C are ordered descendingly by input confidence, as in the CBBNs approach. Then a subset $C_{max} \subseteq C$ of this ordered set is created, which contains the pairs with the highest input confidences. The size of the subset C_{max} depends on the network weights and the consensus threshold t: the sum of network weights of the pairs in subset C_{max} must equal or exceed the consensus threshold t. Then, the edge consensus confidence is the minimum input confidence in the subset C_{max}.

```
Weighted Consensus Networks
Input: Set of $n$ bootstrapped networks, each with an attached weight $w_i$
       indicating its influence such that $\sum_{i=1}^{n} w_i = 1$ and a consensus
       threshold (between 0 and 1)
Output: Consensus network

for each pair of nodes i,j do
    1. Create set $C = \{(c_{ij_1}, w_1), ..., (c_{ij_n}, w_n)\}$ where $c_{ij_k}$ is the edge
       confidence for the edge between nodes $i \to j$ and $w_k$ is the
       weight for the $k$th input network
    2. Reorder and re-index the confidence-weight pairs in $C$ from highest
       to lowest confidence where $c_{ij_1}$ is the highest edge confidence
       and $c_{ij_n}$ is the lowest
    3. Create subset $C_{max} \subseteq C$ such that $C_{max} = \{(c_{ij_1}, w_1), ..., (c_{ij_A}, w_A)\}$
       where $\sum_{k=1:A} w_k \geq t$
    4. Define the edge consensus confidence as $Con_{ij} = \min_{conf}(C_{max})$
end
```

Algorithm 4: Weighted Consensus networks

The subset selection process is illustrated in Figure 6.2 for a single edge example. In this case there are five input networks with weights ranging from 0.1 to

0.4. The network weight — edge input confidence pairs are ordered by input confidence so that $C = \{(0.1, 0.24), (0.3, 0.13), (0.4, 0.12), (0.1, 0.11), (0.1, 0.1)\}$. We build the subset C_{max} by considering the pairs in turn, from the highest input confidence pair to the lowest input confidence pair. For example, for the consensus threshold 0.8 (80%) we build C_{max} by continually adding pairs (in confidence order from highest to lowest) until the weights sum to or exceed 0.8. In this case, this means that $C_{max} = \{(0.1, 0.24), (0.3, 0.13), (0.4, 0.12)\}$ since the weights sum to 0.8. Then the consensus confidence for this edge is 0.12, the minimum confidence in the subset. Another way to view this is to say that for consensus thresholds from 0 to 0.1, C_{max} contains the pair (0.1,0.24) and the consensus confidence is 0.24. For consensus thresholds from 0.11 to 0.4, C_{max} contains the pairs (0.1,0.24) and (0.3,0.13) meaning that the consensus confidence is 0.13, and so on.

The Weighted Consensus networks approach updates the CBBNs approach so that input networks with higher weights have more influence over each edge's consensus confidence than input networks with lower weights. For example, for the single edge example shown in Figure 6.2, the input network with the highest input confidence of 0.24 only has a weight of 0.1, so it only influences the final Consensus network for a small interval of consensus thresholds (0-0.1 or 0-10% in this particular example). By contrast, the input network with weight 0.4 has an influence over a larger interval of consensus thresholds (41-80%). In the CBBNs approach, the input confidences from each network have an influence over equal intervals of the consensus threshold. For the same example, using the CBBNs approach, the input network with the highest confidence of 0.24 would influence the consensus confidence for a larger interval of consensus thresholds from 0-20%. However, by assigning a low weight to this network, its influence can be reduced. This may be appropriate if the user has less confidence in the reliability of this network. Section 6.3 discusses how the input networks weights can be calculated, based on the network's reliability.

6.2.3 Comparison to the original Consensus algorithm

Whilst the CBBNs algorithm draws on the same algorithmic basis as the original approach that is detailed in Chapter 5 — the idea that an interaction between nodes is more reliable if it is exhibited in more datasets or networks — the algorithms that are detailed in the previous sections (6.2.1 and 6.2.2) are essentially new methods that act on different inputs and produce a different type of output. Whilst the original method uses PDAGs to represent the input and output networks, the CBBNs and Weighted CBBNs approaches take as input bootstrapped networks (i.e. networks with confidences attached to each edge) and also output a single network that has confidences attached to each edge. This allows the use of more robust bootstrapped network models as input, which should help to improve the reliability of the output Consensus network.

Whilst it is possible to use bootstrapped networks with the original Consensus approach, as the algorithm requires PDAGs as the input network format it is necessary to threshold bootstrapped networks first in order to produce PDAGs. This leads to another layer of parameters, as the user must select a confidence threshold for each input data source. This highlights the key benefit of CBBNs and Weighted CBBNs — that the more robust bootstrapped networks can be used directly to produce a Consensus model.

Since the inputs and outputs of the original and updated algorithms are different, it is not straightforward (or appropriate) to make a direct performance comparison between them. Instead, we compare the CBBNs and Weighted CBBNs approach against the individual input networks and against a network generated from an aggregate dataset formed by combining the input datasets followed by scale normalisation (a pre-learning aggregation approach), as carried out in Chapter 5 for the original algorithm.

6.3 Measuring network reliability

This section considers how the measure of a network's quality or reliability can be calculated, and how this can be integrated into the CBBNs or Weighted Consensus algorithms described in the previous section. First, a method based on network predictive accuracy for measuring network reliability is presented in Section 6.3.1. Following this, in Section 6.3.2 we describe how prediction-based network reliability measures can be translated to weights for use with the Weighted Consensus networks approach. In Section 6.3.3 a further method is presented, which uses prediction-based network reliability measures to select and discard the input networks for use with the CBBNs algorithm. The final part (see Subsection 6.3.4) of this section considers related work on varying the influence of individual inputs when learning or modelling from multiple data sources.

6.3.1 A prediction-based reliability measure for networks

The robustness, or ability to generalise, of a network can be measured based on how well its node values can be predicted over independent datasets. A network that can predict node values with high accuracy on other independent datasets can be said to be more robust and reliable. This is a fairly intuitive measure of robustness as in particular, it shows that the network does not overfit the dataset from which it was generated, as it can perform well on other datasets.

The procedure for node value prediction over an unseen, independent dataset is described in detail in Section 3.4.2. A brief reminder is provided here. The conditional probability distributions for each node are estimated using the same dataset from which the network structure was learnt. Then the parameterised network model is used to predict the value of each node, based on the values of its influencing nodes in the network, over samples from independent datasets. The success of prediction on each node can be measured using *prediction accuracy*, which is the proportion of samples where the predicted value is correct.

To calculate prediction-based reliability measures for a set of networks, the following method is used. For each network prediction accuracy is calculated for each node in the network, over a random sample of observations equally distributed across the other datasets from which the other networks were generated. Then a median prediction accuracy is calculated for each network, derived across all nodes (the use of the median is to account for outliers, nodes that perform unusually badly or well). In this chapter, the median predictive accuracy is used as the measure of network reliability. Note that in the experiments described later in this chapter, since the input networks are generated from a bootstrap set of DAGs, the prediction accuracy for each node and network is averaged across the bootstrap set of networks.

In general a positive correlation can be found between the median predictive accuracy of a network (where the median is taken across all nodes in the network) on independent data samples and the AUC of the network, when compared against the true (documented) network. For example, Figure 6.3 shows the correlation between median predictive accuracy and AUC for a collection of synthetic datasets generated based on the same network structure. The Pearson Correlation Coefficient for the data in this plot is $\rho = 0.59$ ($p = 0.00002$). This relationship provides further motivation for the use of predictive accuracy as a measure of network quality or reliability.

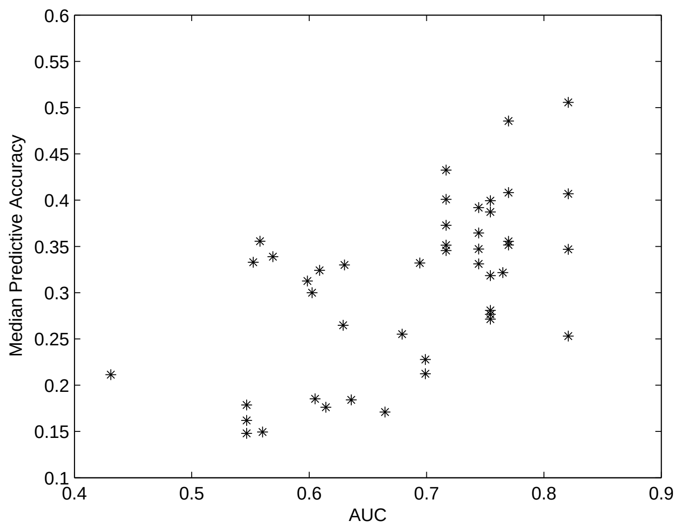


Figure 6.3: Correlation between network median predictive accuracy and AUC for a collection of synthetic datasets. The synthetic datasets were generated based on the same true network, using differential equations and with different levels of added noise (see Section 6.4.1 for more details).

6.3.2 Prediction-based network weighting

The prediction-based reliability measures for a set of networks can be translated to network weights, compatible for use with the Weighted Consensus networks approach that is described in Section 6.2.2. Simply, each network weight is calculated by its median prediction accuracy as a percentage of the total sum of median prediction accuracies across the set of all networks:

$$w_i = \frac{a_i}{\sum_{k=1}^{N} a_k}$$

where N is the number of networks, w_i is the weight for network i and a_i is the median predictive accuracy for network i. This method of calculation ensures that all weights sum to 1, as required by the Weighted Consensus networks algorithm. Then each weight represents the proportional influence for that network in the Consensus algorithm.

6.3.3 Prediction-based network selection

Experimental results presented in the previous chapter indicated that in some cases, using only a subset of available input datasets/networks can produce a better performing Consensus model than when using all available datasets. Therefore, as an alternative, the prediction-based reliability measures for a set of networks can also be used to select a subset of input networks for the CBBNs approach (as described in Section 6.2.1, where each network has an equal influence on the Consensus process). In this case, instead of weighting the influence of individual networks, the least reliable networks are simply discarded from the input.

Using the prediction-based reliability measure, the input networks can be ranked from most reliable to least reliable. There are then a number of ways in which a line can be drawn between the most and least reliable networks, and the least reliable networks discarded from the input:

- *Networks-above-x selection.*
  In this case a threshold x for the median predictive accuracy is selected. All

networks with median predictive accuracy $\geq x$ are selected, and networks with median predictive accuracy $< x$ are discarded.

- *Best-n-networks selection.*
  In this case, n is a number or proportion of the available networks. This is a simple method where the top n networks (i.e. the n networks with the highest median prediction accuracy) are selected as input, and the remaining networks discarded.

One issue with these network selection methods is that they introduce another parameter into the Consensus process. Later in this chapter, the methods of network weighting and network selection are compared to see which is more effective at improving the final Consensus model. A further goal of the comparison is to consider how the parameters for each network selection approach (i.e. the median predictive accuracy threshold x and the number or proportion of input networks n) can be chosen in order to optimise the final Consensus network performance.

6.3.4 Related work

Whilst using weights to vary the influence of each dataset when learning or modelling from multiple data sources is obviously not new, most previous research has focused on combining classifiers, whereas we are concerned with learning and combining network structures. In particular, the idea of combining models has some similarities to ensemble learning, such as boosting (Schapire, 2003), which aims to combine several weak classifiers into one strong classifier. In general the weak classifiers are combined according to some weighting that is related to their accuracy. Similarly, Bayesian Model Averaging (BMA) (Hoeting *et al.*, 1999) is a technique that calculates a (weighted) average over the posterior distributions of a set of potential models. There has been research in using BMA with BNs for classification and prediction. However, BMA is not model combination. Instead it is designed to address uncertainty in model selection given a particular dataset. Both types of technique are designed to improve classification or prediction for a model across a single dataset. By contrast, we are concerned with combining models generated from multiple datasets, which have their own biases and levels of noise. This means that we may have a high quality dataset requiring a high weight, and a low quality dataset that requires a low weight. Ensemble learning and BMA have not been designed to deal with this type of input.

6.4 Network selection or network weighting: a comparison

This section reports on the experiments performed to evaluate the use of the CBBNs approach on multiple microarray datasets. The standard CBBNs approach is compared with the prediction-based weighted and prediction-based selection Consensus approaches. It is also compared against the performance of the individual input networks and the 'Normalisation Only' approach — a network generated from a combined dataset that is formed by concatenating and normalising the individual datasets — as was carried out in the previous chapter for the original Consensus algorithm. An additional objective of these experiments is to evaluate which is the best method for prediction-based selection in conjunction with the CBBNs approach: networks-above-x selection or best-n-networks selection. In order to do this a number of Consensus networks are generated for prediction-based selection, based on the use of different subsets of input networks.

Section 6.4.1 details the datasets used and the experimental design. Sections 6.4.2 - 6.4.3 present results on synthetic and real microarray datasets.

6.4.1 Datasets and experiment design

In order to examine the relationship between the input network quality and the performance of the resulting Consensus network, experiments were first carried out on synthetic microarray datasets to provide a controlled setting, before moving on to evaluation on a real data application. In this section, the different types of datasets that are used are described, and the experiments performed and evaluation is explained.

The synthetic datasets are based on a synthetic network, which is generated based on a regulatory network structure of 13 genes. This is the same network that is used for the experiments detailed in the previous chapter (see Section 5.4.2). In

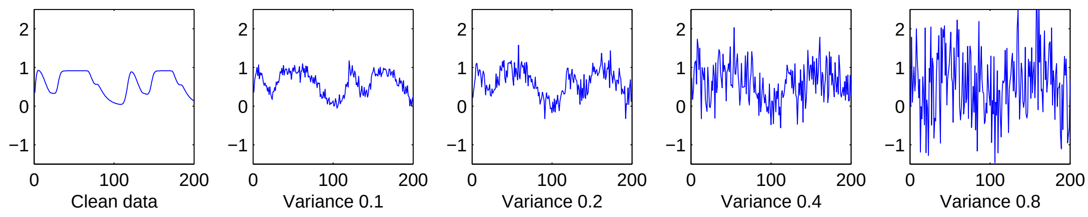


Figure 6.4: Adding Gaussian noise to a clean dataset

order to investigate how network quality affects the Consensus approach a number of further datasets were generated by adding Gaussian noise, with various variances, to each dataset. The effects of the various noise levels on one gene in dataset 1 is shown in Figure 6.4. This means we have clean datasets, generated directly from the differential equations and datasets with added noise. Each dataset was discretised using an equal frequency method with three bins. Input bootstrapped networks were then generated from each dataset. Each Consensus approach was run on different collections of input networks, which are detailed in Appendix A, Table A.1. Each set of input networks contains 4 networks, where each network is generated from a version of one of the original four datasets (no set contains more than one version of each dataset). An additional 4 collections of datasets were also generated, where Gaussian noise of different variances (randomly selected between 0.1 and 0.8) was added to each gene. These are denoted as 'variable noise by gene'. The sets of networks are ordered in Table A.1 by their *collective level of reliability*, which is measured as the median of median predictive accuracies for the input networks generated from each dataset in the set. In general, the Gaussian noise applied across the datasets increases as the collective level of reliability of the network set decreases.

For the real data application, we use the same five yeast microarray expression datasets used in the previous chapter (see Section 5.4.4). This is a sub-network of 9 regulatory genes that are related to heat-shock response. The microarray datasets are publicly available on the YeastBASE expression database — see Table 5.3 for more details. The learnt networks are evaluated by comparing them to documented gene interactions, obtained from the online YEASTRACT database (Teixeira *et al.*, 2006).

As described earlier, there are 18 collections of datasets for the synthetic networks and one set of real yeast gene expression datasets. For each collection of datasets — which is referred to as a case — a bootstrapped network is learnt for each individual dataset. For each case, the bootstrapped networks are then used as inputs to the Consensus approach. The following network types are generated for each case:

- Standard CBBNs method — generated from all input networks — referred to as the CBBN network
- Prediction-based weighted network, generated from all input networks — referred to as the weighted network
- Prediction-based selection networks, generated from subsets of all input networks, where the networks selected are those with the highest median predictive accuracies — referred to as the selection networks
- A bootstrapped network generated from a concatenated and scale-normalised dataset (referred to as the Normalisation Only network), in order to facilitate comparison to pre-learning aggregation

For the prediction-based selection method, a number of different networks are generated. For example, since there are four input networks in total, two subsets would be used to generate Consensus networks. The first subset would contain the three networks with highest median prediction accuracies, and the second subset would contain the two networks with the highest median prediction accuracies. This is because one objective of these experiments is to establish the best method for prediction-based selection. By examining the Consensus network performance results for prediction-based selection, in conjunction with the median prediction accuracies of the selected input networks, we can establish whether networks-above-x selection or best-n-networks selection works best, and comment on how the parameters x and n can be chosen effectively.

The performance of the Consensus networks is evaluated by comparing them against the true network in terms of TP and FP interactions. As described in Section 3.4.1, this comparison can be represented by a single AUC value. A higher AUC value (assuming values are above 0.5) indicates a better performing network. Note that a different network is generated for each consensus threshold from 0 to 1 (0-100%). This means that the AUC value may vary across consensus thresholds for each different approach. Therefore in the comparison, for each approach we consider the maximum AUC value achieved and the corresponding consensus threshold.

6.4.2 Application: synthetic networks

The AUC performance comparison plot (top) in Figure 6.5 shows the maximum AUC performance for each approach by network set (referred to as sets 1-18). In addition, the AUC of the best performing input network is recorded on the plot. Recall that a Consensus network of each type is generated for each consensus threshold from 0-1 (0-100%). This means that the maximum AUC performance (as shown in the plot) relates to a specific consensus threshold or interval of consensus thresholds. The consensus threshold interval comparison plot (bottom) in Figure 6.5 shows the length of the consensus threshold interval for which each approach achieves its maximum AUC value. In Appendix A, Table A.2 shows the same information in tabular form. For each approach and set of networks, the maximum AUC value and the consensus threshold or interval of thresholds for which this is achieved are listed. Additionally, this table lists the number of input datasets for which the best performance is achieved with the prediction-based selection approach.

6.4.2.1 Performance comparison of approaches

The AUC performance comparison plot (top) in Figure 6.5 (with full details given in Table A.2) shows the following:

- The CBBN approach outperforms, or equals, the performance of the best performing input network in 12 of the 18 cases.
- The CBBN approach outperforms, or equals, the performance of the Normalisation only network in 16 of the 18 cases.

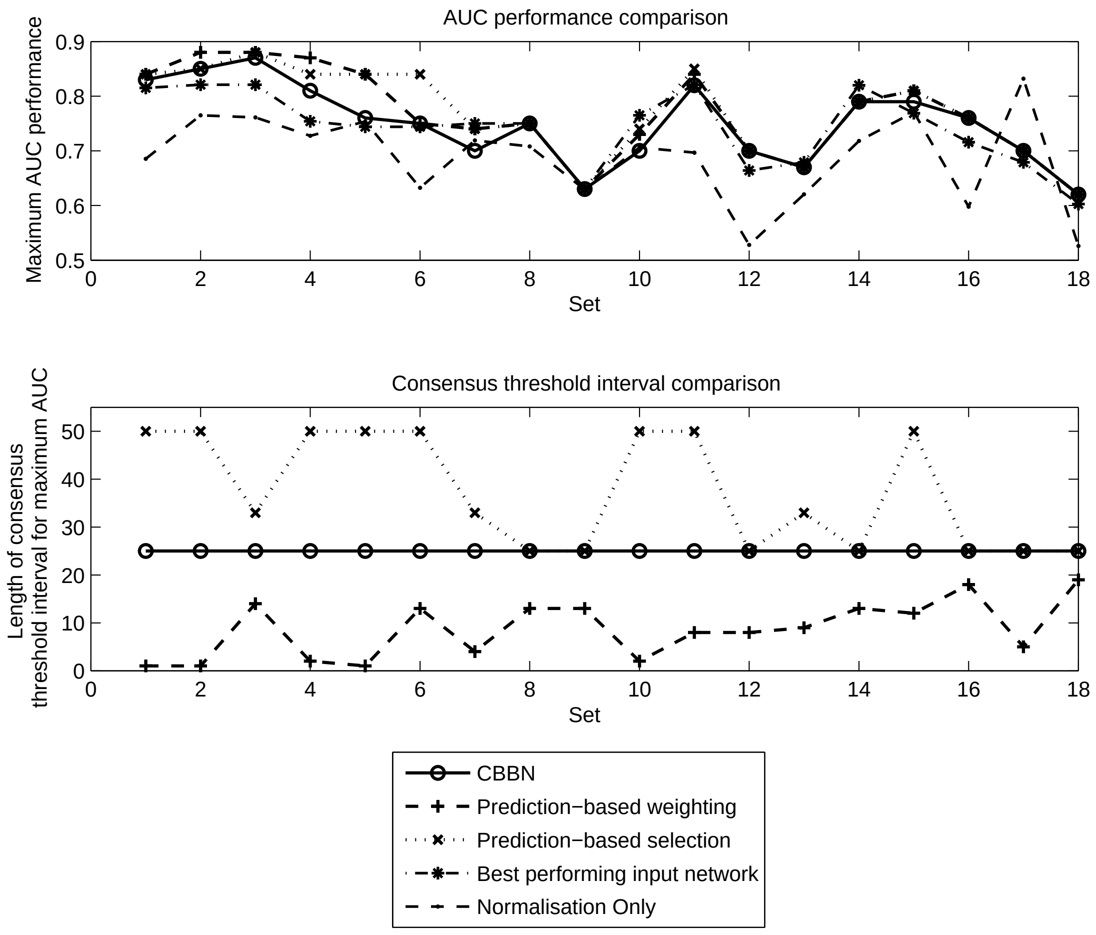


Figure 6.5: Synthetic results: across all network sets, a comparison of the maximum AUC and the length of consensus threshold interval for which this is achieved

- The prediction-based weighting and selection approaches outperform or equal the CBBN approach in all 18 cases. In 16 of these cases, both the weighting and selection also both outperform or equal the performance of the best performing input network. In 17 of these cases, both the weighting and selection also both outperform or equal the performance of the Normalisation only network.
- The prediction-based weighting approach outperforms the selection approach in 2 cases, and the prediction-based selection approach outperforms the weighting approach in 3 cases. In all other cases both approaches have an equal maximum AUC.

Based on these results, we can see that the two new approaches, prediction-based weighting and selection, always improve on or at least equal the performance of the standard CBBN. In particular, the new approaches are able to outperform the best performing input network and the Normalisation only network in more cases than the standard CBBN. However, there is less of a difference in performance between the weighting and selection methods — in the majority of cases they achieve equal maximum AUCs. In order to obtain a p-value indicating the statistical significance of these findings, a paired t-test was used, the results of which are shown in Table 6.1. All Consensus-based approaches outperform the Normalisation only network with $p \leq 0.003$. The weighting and selection approaches also both outperform the best input network with statistical significance (p=0.016 and 0.006 respectively), whilst the CBBN approach only obtains a p-value of 0.27 for outperforming the best input network. The weighting and selection approaches also outperform the CBBN method with statistical significance (p=0.008 and 0.009 respectively), but there is no significant difference between the weighting and selection approaches (p=0.63).

The AUC performance comparison plot (top) in Figure 6.5 also shows a relationship to the collective level of reliability of the network sets. Recall that the network sets are ordered in terms of their collective reliability (i.e. overall, the networks in set 1 have a higher prediction accuracy than those in set 18). It can be seen in this plot that in general the maximum AUC decreases as the collective level of reliability decreases, which is what we would expect. However, we also

Approach 1	Approach 2	Paired t-test p-value
CBBN	Normalisation only	0.003
Weighting	Normalisation only	0.0003
Selection	Normalisation only	0.0004
CBBN	Best input network	0.27
Weighting	Best input network	0.016
Selection	Best input network	0.006
Weighting	CBBN	0.008
Selection	CBBN	0.009
Weighting	Selection	0.63

Table 6.1: Synthetic results: statistical significance performance comparison between approaches. Using a paired t-test, this table shows the p-value for whether Approach 1 significantly outperforms Approach 2.

notice that although the weighting and selection approaches equal or outperform the CBBN and best input network in the majority of cases, the largest increases in performance are found with the most reliable collections of datasets (e.g. sets 1-6). This implies that whilst the weighting and selection Consensus approaches are beneficial for combining datasets with any level of noise, combining better quality data can produce even greater increases in performance.

6.4.2.2 Selection of the consensus threshold

The comparison of approaches discussed in the previous section (6.4.2.1) is based on the maximum AUC achieved by each method, where maximum AUC corresponds to a specific interval of consensus thresholds. Figure 6.5 shows that this interval varies by network set. Therefore, an issue that is raised for all approaches is, how do we predict the exact consensus threshold where the maximum AUC value is to be found? In order to consider this in more detail, Fig 6.6 compares the AUC performance for the different approaches, across all consensus thresholds from 0-1 (0-100%), for a selection of the network sets. In these plots we can see how the performance of each method can vary considerably across the range

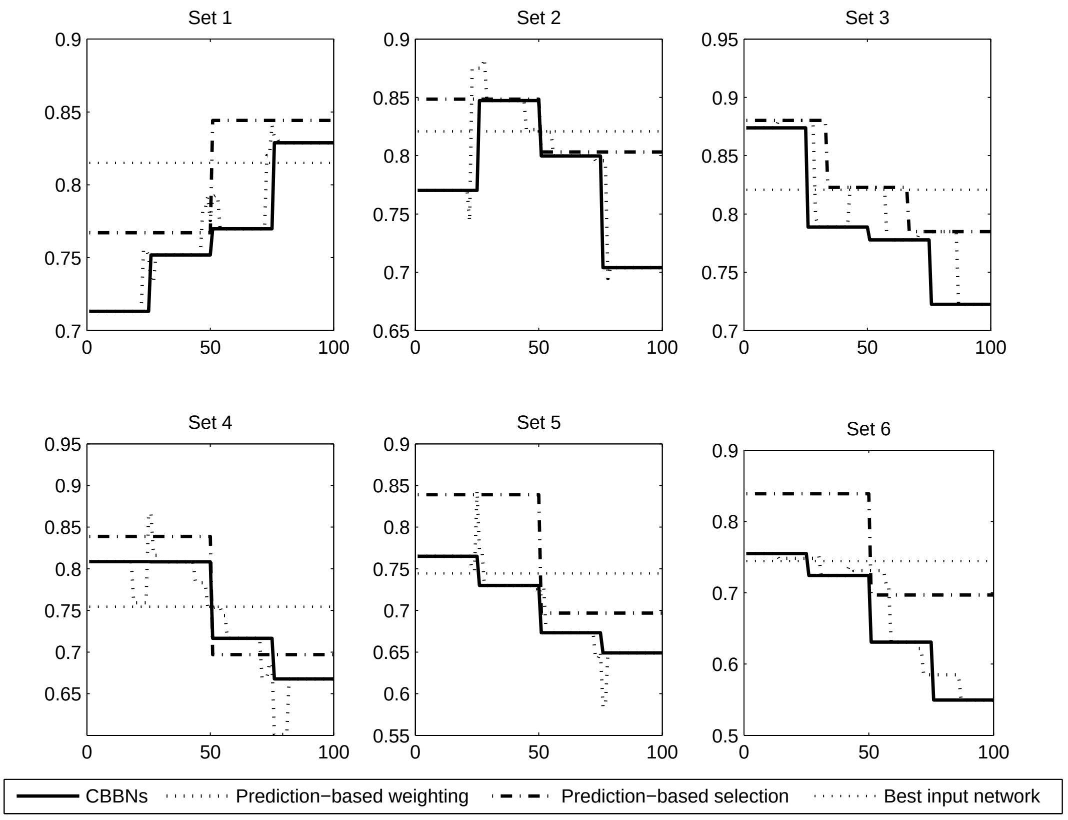


Figure 6.6: Synthetic networks: approaches performance comparison for a selection of individual network sets. The x-axis indicates the consensus threshold (between 0-100%) whilst the y axis indicates the network's AUC value.

of consensus thresholds.

First, we can see that the performance of the prediction-based weighting approach varies considerably by consensus threshold. The thresholds where the AUC changes correspond to 'weighting boundaries' — thresholds at which an additional network is able to influence the final Consensus network. In general, these weighting boundaries correspond to the areas around the standard equal weighted boundaries — at around the 25%,50% and 75% thresholds. This is because in these cases the networks weights are relatively close to being equal weights. However, even in these cases, the use of weights can cause a significant improvement in AUC. For example, in sets 4 and 5, there is a significant peak at around the 25% threshold for the AUC value of the weighted Consensus network. In particular, in set 5 at around the 25% threshold, the AUC value rises from around 0.77 (CBBN) to 0.85 (prediction-based weighting). However, there are also intervals of consensus threshold where the weighting can cause a significant decrease in AUC. For example, in set 4 at around the 25% consensus threshold, there is a significant decrease in the AUC value just prior to its significant increase.

Therefore, selection of the optimum consensus threshold for prediction-based weighting is not a trivial task. In comparison, for the CBBN or prediction-based selection approaches, we know that AUC value will only change at specific consensus thresholds (e.g. 25%,50% and 75% where there are four input networks). There are many more thresholds that can change the weighted Consensus network, due to different combinations of the weights. Although in some cases (e.g. sets 2,4 or 5), the weighted network equals or outperforms the selection networks, using network selection provides a key advantage — consistency. Where the AUC improves prediction-based selection, it does so for a much larger interval of consensus threshold. For example, in set 5, the prediction-based weighted Consensus network shows the highest AUC value for a specific consensus threshold, whereas the prediction-based selection Consensus network shows the same high AUC value for the 0-50% thresholds. This is also highlighted in the consensus threshold interval comparison plot (bottom) in Figure 6.5, which shows that the prediction-based selection network always achieves the maximum AUC for a larger interval than the prediction-based weighting network.

For prediction-based selection, the results indicate that there could be a relationship between the reliability of the individual networks and the optimum consensus threshold. For example in set 1 (see Figure 6.6), which comprises networks generated from clean data, the optimum consensus thresholds are between 50 and 100%. In set 2, which contains networks generated from data with a small amount of added noise, the consensus thresholds are between 25 and 50% provide the best AUC values for all approaches. In sets 3-6, which comprise noisier datasets, the optimum consensus thresholds are low — 25% or under. This finding could demonstrate that the consensus threshold is dependent on the coherency of the input network set. A set of high predictive accuracies imply that a set of input networks that are coherent — that is they fit well across all the individual datasets. A set of input networks with low predictive accuracies may not be so coherent and the low accuracies may indicate that the networks are quite different. In this case, when a low consensus threshold is chosen it means that edges appearing in only a few networks have a better chance of appearing in the final Consensus network, which can be beneficial when the networks are quite varied. Conversely, in the case where the set of input networks is highly coherent, high consensus edges are more likely to be robust and reliable. This is because the networks are more likely to be similar, so low consensus edges are more likely to be due to noise. However, the relatively small number of cases and limited scope of the synthetic network used means no firm conclusions can be made here. Section 6.5 proposes how this can be investigated further, using machine learning for the discovery of patterns in a larger range of network sets.

6.4.2.3 Method comparison for the network selection approach

Prediction-based selection involves improving the final Consensus network by the selection of a subset of the input networks, based on their reliability (as measured using prediction accuracy). In Section 6.3.3, two methods were proposed for selecting a subset of input networks — networks-above-x (selecting networks with prediction accuracy greater than x) and best-n-networks (selecting the best n networks, when they are ordered by their median prediction accuracy). Figure 6.7 provides boxplots that show the distribution of x and n for n = 2,3 and 4. Further details are provided in Appendix A. For each network set Table A.3 shows the optimum number of selected input networks (n) and the highest median prediction accuracy of the networks that have not been selected (this is the threshold x). The lowest median predictive accuracy of the selected networks is also shown.

Firstly, the boxplots and table clearly show that there is no single value for x, i.e. it is not possible to choose a prediction accuracy threshold x for network selection that is appropriate to all cases, since the distributions of highest median predictive accuracies overlap for n = 2,3 and 4. (Table A.3 shows that the highest median predictive accuracy of the unselected networks varies from 0.15 to 0.44.)

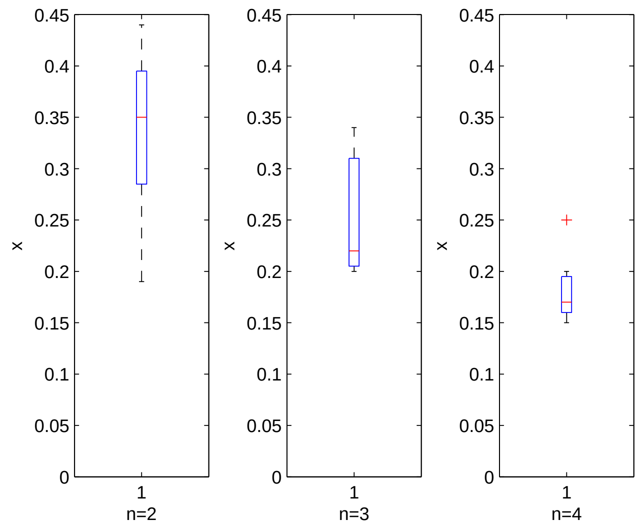
Given the variability of the dataset noise in each set, this is not surprising — a set of very noisy networks (e.g. set 18) will all have much lower predictive accuracies than a set of more reliable networks (e.g. set 1). Therefore it is not appropriate to select a single threshold x, as it could lead to the selection of no input networks.

However, further inspection of Figure 6.7a reveals that there is a relationship between n and x. Where fewer networks are selected, the selected networks have a higher threshold for x, the highest median predictive accuracy of the networks not selected. The same pattern is also seen if we compare the lowest median predictive accuracy of the selected networks with n (see Figure 6.7b). Together, these patterns imply that network sets with higher predictive accuracies (i.e. higher reliability) require a lower number of input networks, whilst network sets with lower predictive accuracies (i.e. lower reliability) require more input networks.

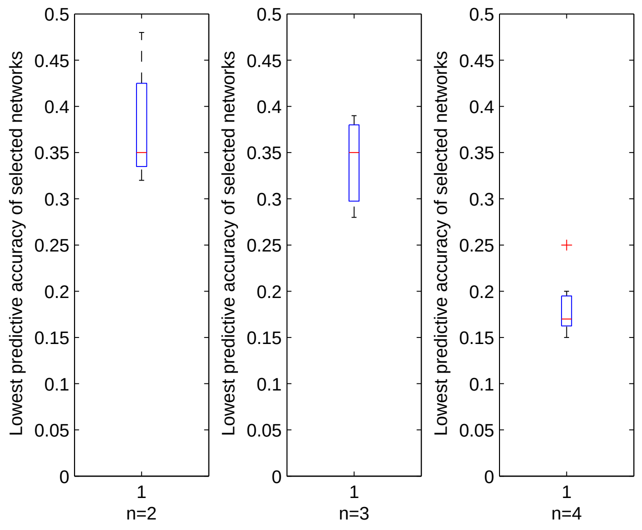
This is the type of pattern that we would expect — when we have higher quality data, we do not need so much of it, and conversely when we have lower-quality data, we need more of it in order to build the best performing model. However, as concluded previously for the selection of the consensus threshold, the number of cases and limited scope of the network used in these experiments means no firm conclusions can be drawn here. In particular, the boxplots in Fig 6.7 show that a single set of thresholds on the network predictive accuracies cannot be derived for the selection of n, the number of input networks, since the distributions overlap. However, deriving some general rules may be possible based on a larger and broader range of datasets. Therefore, Section 6.5 proposes the use of machine learning to discover patterns in a larger range of network sets, in order to guide the choice of n.

6.4.3 Application: yeast heat-stress network

In this section, the new Consensus approaches are applied to a set of real yeast microarray datasets, that have been generated by different heat-stress microarray studies. Figure 6.8 shows the AUC for each Consensus approach, across all consensus thresholds from 0 to 1, as well as the Normalisation only network and the best performing individual input network AUC (which is the network generated



(a) Highest median predictive accuracy of unselected networks



(b) Lowest median predictive accuracy of selected networks

Figure 6.7: Prediction-based selection. (a) shows boxplots of the distribution of x, the highest median prediction accuracy of the networks that have *not* been selected by n the optimum number of selected networks, whilst (b) shows boxplots of the distribution of the lowest median prediction accuracy of the networks that have been selected by n the optimum number of selected networks

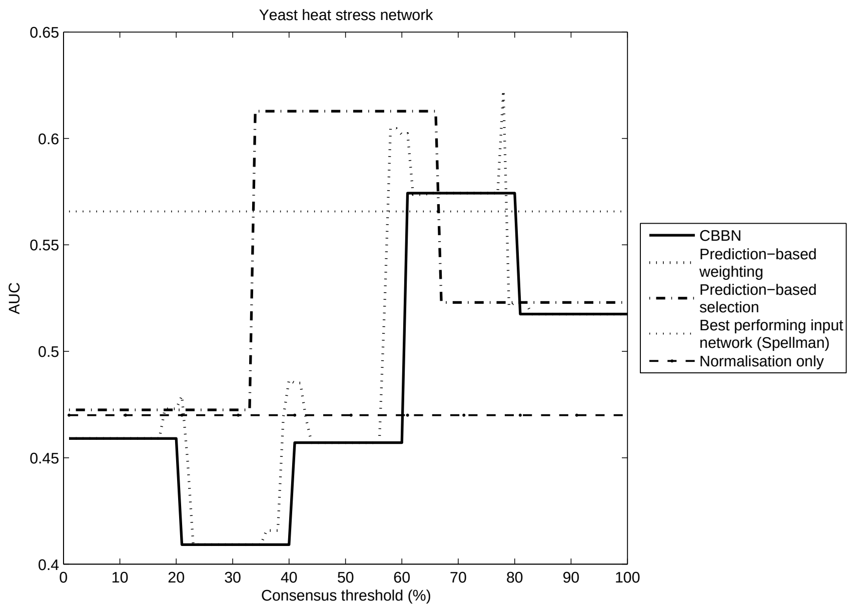


Figure 6.8: Yeast heat stress network: comparison of approaches by AUC

from the Spellman dataset). The prediction-based selection approach uses the best three networks in order to achieve the maximum AUC.

Figure 6.8 shows that:

- The best performing input network (Spellman) achieves an AUC of 0.566
- The Normalisation only network achieves an AUC of 0.471
- The CBBN achieves a maximum AUC of 0.574 between 61-80% consensus thresholds
- The prediction-based weighting network achieves a maximum AUC of 0.623 at the 78% consensus threshold
- The prediction-based selection network achieves a maximum AUC of 0.613 between 34-67% consensus thresholds

The prediction-based weighting network achieves the highest AUC, making an improvement of almost 0.05 in AUC over the CBBN approach. However, this is

Dataset	Median predictive accuracy	Prediction-based weight	Individual network AUC
Beissbarth	0.25	0.17	0.43
Eisen	0.26	0.18	0.43
Gasch	0.31	0.22	0.50
Grigull	0.30	0.21	0.49
Spellman	0.31	0.22	0.57

Table 6.2: Yeast heat stress datasets: prediction-based reliability measures

only at a single consensus threshold of 78%. Figure 6.8 shows that it is around the thresholds relating to equal weights (20%,40%,60%,80%) where the prediction-based weighting network attains the largest increases in AUC. This is because the prediction-based weights vary only slightly from a uniform distribution (see Table 6.2).

The prediction-based selection network also achieves a significant improvement in the maximum AUC when compared to the CBBN or input networks. The best result is achieved when the best three input networks are selected (where 'best' is defined by the highest median prediction accuracies, giving the Gasch, Grigull and Spellman networks, see Table 6.2). This network attains a maximum AUC of 0.613 between the consensus thresholds of 34% and 67%. This shows that the removal of the two noisiest datasets can significantly improve the performance of the final Consensus network. Although the prediction-based weighting Consensus network attains a slightly higher maximum AUC of 0.623 at a 78% consensus threshold, this is at a single consensus threshold. As noted in the results for the synthetic network, prediction-based selection improves the network performance more consistently across the consensus thresholds. Additionally, as input networks are removed in the selection approach, this reduces the number of different consensus threshold intervals that need to be considered, making the selection of the 'best' consensus threshold potentially much easier.

6.5 Discussion

This chapter has furthered the Consensus approach for combining multiple microarray datasets, resulting in three main contributions. First, the introduction of an improved Consensus approach, Consensus Bootstrapped Bayesian Networks (CBBNs), which allows the more robust bootstrapped network models (rather than PDAG models produced by thresholding a bootstrapped network) to be used as direct inputs to the Consensus algorithm. Second, this chapter has presented an examination of how network reliability can affect the final Consensus network. A measure of reliability for sets of networks, based on the prediction of node values has been introduced. Finally, methods for incorporating the use of network reliability measures into the Consensus algorithm have been proposed. Prediction-based network weighting allows different weights to be assigned to each network in order to vary the influence of each input network, whilst prediction-based network selection uses only a subset of the available input networks, based on their reliability, in order to produce a Consensus model. This chapter has also presented a comparison of these different Consensus approaches, using synthetic and real microarray datasets.

Experiments were performed on 18 collections of synthetic microarray expression datasets and one group of real yeast microarray datasets. The experiments on the groups of synthetic datasets have allowed the Consensus approaches to be evaluated over many different cases, where the noise levels in each dataset are varied. For example, in some cases all datasets were very noisy, whereas in other cases each dataset was of high quality, and in other cases noise levels were mixed across the datasets.

Next, the conclusions of the comparison are summarised with respect to the original aims:

- *In general, the new CBBNs approach outperforms the pre-learning aggregation 'Normalisation only' network and all individual input networks.*
  In the synthetic data based experiments, the CBBN approach outperformed all input networks in 66% of cases, and the Normalisation only network in 89% of cases. The CBBN approach outperformed all input networks and the Normalisation only network in the real yeast data case.

- *Prediction-based weighting improves the performance of the Consensus network, i.e. in general, the prediction-based weighted Consensus network outperforms the CBBNs approach.*
  In all cases presented in the comparison, prediction-based weighting was able to equal or improve the performance of the Consensus network, in comparison to the standard CBBNs approach. A paired *t*-test indicated that there was a significant improvement in performance.

- *Selecting a subset of input networks creates a performance improvement in the Consensus network, i.e. in general, prediction-based selection outperforms the standard CBBNs approach.*
  In all cases presented in the comparison, prediction-based selection was able to equal or improve the performance of the Consensus network, in comparison to the standard CBBNs approach. A paired *t*-test indicated that there was a significant improvement in performance.

- *In general, prediction-based weighting or prediction-based selection consistently produce the best performing Consensus network, but there is no significant difference in performance between weighting and selection.*
  Despite this, prediction-based selection provides an important benefit — consistency of performance across a larger interval of consensus thresholds. Often, the weighting approach would attain a maximum AUC at a specific consensus threshold, whereas prediction-based selection could attain the same maximum AUC for a larger range of consensus thresholds. This is potentially advantageous when trying to select the optimum consensus threshold.

This establishes that the prediction-based selection approach provides improvement in terms of network performance, and in addition the performance of the final Consensus network is more consistent over consensus thresholds. However, one original aim remains unanswered with respect to the selection approach: how is the number of input networks to be chosen? The results from synthetic data showed that there is a relationship between the reliability of the available input networks and the optimum number of these networks to be selected; essentially, fewer input networks are required with high-quality data, and conversely with lower-quality data, more input datasets are required in order to build the best performing model.

The final original objective refers to the selection of the optimal consensus threshold. The AUC performance of the final Consensus network, whichever approach is used, can vary considerably by consensus threshold. However, for prediction-based selection in particular, the experimental results indicated that there could be a relationship between the reliability of the individual input networks and the optimum consensus threshold. More reliable sets of input networks attained their maximum AUC values with higher consensus thresholds than noisier, less reliable sets of input networks.

However, the experiments presented in this chapter have been carried out on a limited set of data. For example, the synthetic network sets are all based on the same, small-scale network, and noise has been added in an artificial manner. Each case contained four available input datasets/networks. In order to draw firmer conclusions, a larger set of different datasets, based on different networks would be required.

Finally, to close this chapter, the author proposes the use of machine learning to discover patterns between network reliability and the optimum number of input networks or consensus threshold, based on a larger and broader range of network set cases. Whilst it can be straightforward to spot patterns in network quality by eye or simple analysis for a small group of cases, for a larger set of cases this is infeasible. This motivates the use of machine learning to automatically discover patterns and relationships of interest, based on a set of example cases. Further to this, machine learning techniques could be used to induce heuristics, or rules of thumb, for the selection of input networks or the consensus threshold. Exploratory research to investigate this proposal is presented in Chapter 7. Rule learning provides the most natural representation for the type of heuristics we wish to learn, since if-then rules are transparent and human-readable. Most importantly, this means that the learnt heuristics can be easily applied by the user to the problem of network and consensus threshold selection. Chapter 7 presents the rule-learning methodology and preliminary results for inferring heuristics based on the synthetic case results presented in this section, together with further network cases.

Chapter 7

Inducing heuristics for the Consensus approach

7.1 Introduction

Chapter 6 investigated the concept of network reliability and how this can impact the Consensus network model. A key conclusion was that prediction-based network selection can be used to improve the Consensus network performance in many cases. This is where only a subset of the available input networks are selected in order to produce the best performing Consensus model. However, an important problem remains concerning how to select the optimal number of input networks.

Additionally, one important parameter needs to be chosen by the user: the consensus threshold. The consensus threshold allows the user to select the level of 'robustness' of the final Consensus network. For example, a low threshold of 25% will allow network edges with limited support in the final model, whereas a threshold of 100% requires each edge to appear in each input network. However, how do we choose the most appropriate consensus threshold based on the input datasets/networks available, or the consensus threshold that produces the best performing model?

This chapter presents preliminary research in addressing these issues by using machine learning to induce heuristics, 'rules of thumb', in the form of classification rules, for selecting input networks and choosing different consensus thresholds. The remainder of this chapter is organised as follows. Section 7.2 describes the problem of constructing heuristics in more detail and motivates the use of machine learning. In Section 7.3, rule learning, the chosen machine learning method is described. Learnt heuristics are presented in Section 7.4, followed by the application of the learnt heuristics to examples from real microarray expression datasets in Section 7.5. Finally, the chapter concludes with a summary and discussion in Section 7.6.

7.2 Motivation

Heuristics, or 'rules of thumb', can assist the user in selecting the input networks and consensus threshold for the Consensus approach. Such heuristics must be based on information that can be extracted from the input networks themselves. For example, the input network AUCs cannot be used, as this is based on knowledge of the true network, which is usually unavailable. Instead, the heuristics should be based on measures of input network quality. This research uses the network predictive accuracy (measured on the other input datasets) as a measure of network quality, as explained in Chapter 6 (see Section 6.3).

When constructing heuristics for network and consensus threshold selection, it makes sense to examine the patterns in quality amongst the input networks — which is, in this case, measured using the network predictive accuracies. For example, consider Figure 7.1, which shows the input network predictive accuracies for input networks for synthetic cases 1-6 (from Chapter 6, see Section 6.4). In these plots the y-axis represent the network's median predictive accuracy. Note that we are only concerned with comparison on the y-axis and the x-axis is not used. The black lines indicate the predictive accuracies for input networks, and the red line shows a predictive accuracy of 0.33 (which represents a 'random' accuracy threshold, since the expression values are discretised into three bins). Recall that each of these sets is based on a total of four input networks, and for sets 1,2,3 and 6, the optimal number of input networks is 2. For sets 4 and 5, the optimal number of input networks is 4 (this can be seen in Fig 6.6). Some patterns can be seen immediately by eye. For example, in general, the cases for which two input networks is optimal have higher predictive accuracies than those for which four input networks is optimal. Therefore, based on this information, a heuristic for selecting two input networks may be ‘if at least two networks have predictive accuracy above 0.33 then two input networks is the optimal selection’.

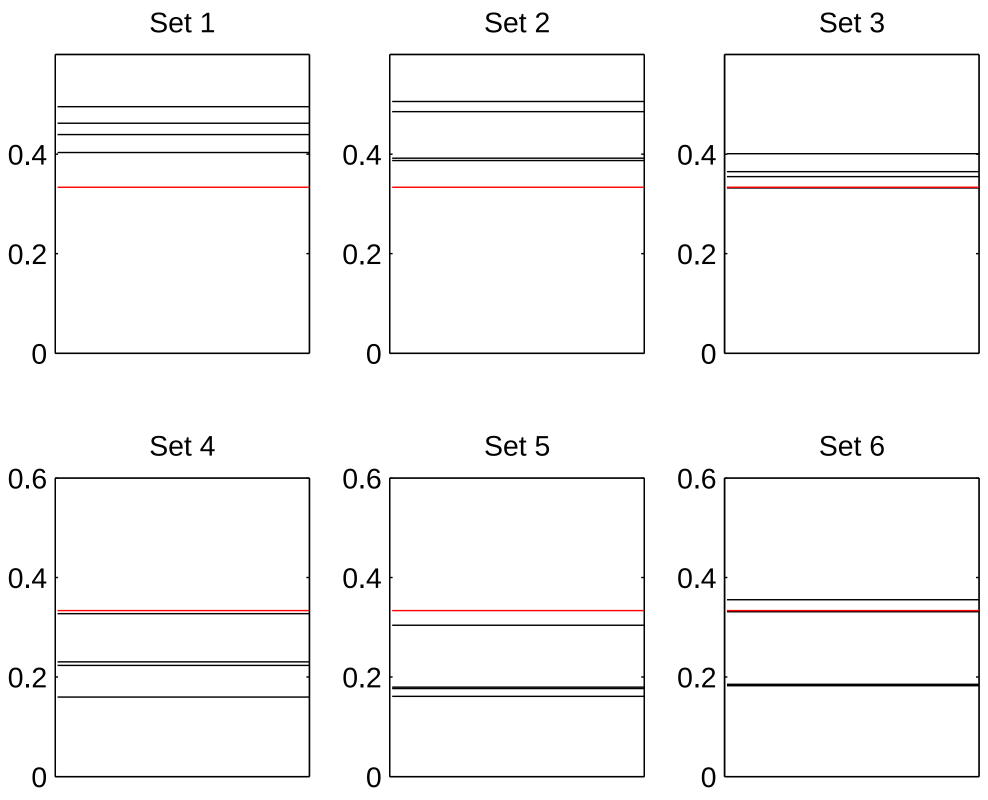


Figure 7.1: Patterns in input network predictive accuracies for synthetic cases 1-6. The y-axis represent the network’s median predictive accuracy. Black lines indicate the predictive accuracies for input networks. The red line shows a predictive accuracy of 0.33

In order to construct reliable heuristics, a large set of different cases is required. Whilst it can be straightforward to spot patterns in network quality by eye for a small group of cases, for a larger set of cases this is infeasible. This motivates the use of machine learning to automatically induce the heuristics, based on a set of example cases.

Concept learning (Mitchell, 1997) is a machine learning approach that can be used to induce a general description of a target class or category of objects, based on a set of example cases, which are labelled as positive or negative instances of that particular category. For example, for the problem of learning heuristics for the selection of input networks or consensus thresholds, the target classes would be the optimal number of input networks, or the optimal interval of consensus thresholds. There are many different techniques that can be applied as concept learners. The various techniques are usually differentiated by their method for representing the learnt concept description. For example, decision trees represent the learnt description as a tree-structure model. Bayesian networks can also be used a concept learning technique, where one node in the network represents the target concept. Our chosen concept learning technique is rule learning, where the learnt description is represented as a set of if-then rules. Rule learning provides the most natural representation for the type of heuristics we wish to learn, since if-then rules are transparent and human-readable. Most importantly, this means that the learnt heuristics can be easily applied by the user to the problem of network and consensus threshold selection. The following section describes rule learning in more detail.

7.3 Rule learning

This section introduces the rule-based concept learning approach. Before describing the algorithm for inducing rules in more detail, in Section 7.3.1 we begin with a description of the algorithm inputs and outputs, relating them to an example that is based on the type of heuristic we wish to induce. Following this, in Section 7.3.2, the basic rule learning algorithm is described. Section 7.3.3 motivates the use of relational rule learning. Finally Section 7.3.4 introduces ACE, a data mining system that implements relational rule learning, which is used to learn the rule-based heuristics in this chapter.

7.3.1 Inputs and Outputs

In order to learn a set of rules that describe a target concept class or category, the following inputs to the algorithm are required:

- A target class
- A set of training examples, labelled as positive and negative instances of the target class
- A dataset of observations, or attributes, of the training examples

Table 7.1 shows how these inputs relate to the problem of interest — learning heuristics for the selection of input networks or consensus thresholds — using a small, 'toy' example. In this example, the target class is '*two-networks*' — representing cases where the optimal network selection is the two input networks with the highest predictive accuracies. First, in order to induce a set of rules that describe this class, a set of training examples is required. In Chapter 6, Section 6.4 presented a number of sets of synthetic networks and the results of applying the Consensus approach to them. These cases can be used as training examples, where each set is labelled as 'positive' if two networks form the optimal input network selection, and 'negative' otherwise. In Table 7.1 the first six sets (synthetic cases 1-6) are shown. Finally, as well as the positive/negative label, further attributes or observations are required for each training example. These attributes will be used to form the body of the if-then rule. For this problem, the attributes are the measures of quality for each input network — i.e. the predictive accuracy of each input network. Note that the input networks are ordered by their predictive accuracies.

The output of the algorithm is a set of if-then rules that satisfy the positively-labelled training examples, but do not satisfy the negatively-labelled examples. An if-then rule takes the format *IF* body *THEN* head, where the body is a list of attribute conditions to be satisfied, and the head is the target concept. For the problem described in Table 7.1, an example rule is shown in Figure 7.2. This rule satisfies all the positive examples and none of the negative examples. It says if the predictive accuracy of Network 1 exceeds 0.34, then the optimum number

Target concept: *two-networks*

Training examples	Class label	Attributes — predictive accuracies Network 1	Network 2	Network 3	Network 4
Set 1	+	0.50	0.46	0.44	0.40
Set 2	+	0.51	0.48	0.39	0.39
Set 3	+	0.40	0.36	0.35	0.33
Set 4	-	0.33	0.23	0.22	0.16
Set 5	-	0.30	0.18	0.18	0.18
Set 6	+	0.36	0.33	0.19	0.18

Table 7.1: Learning heuristics for network selection: inputs to the rule learning algorithm

of inputs networks is 2. Since the network accuracies are ordered by size, this is equivalent to saying that if at least one network has predictive accuracy greater than 0.34, then the optimum number of inputs networks is 2.

IF Network 1 $\leq$ 0.34 THEN *two-networks*

Figure 7.2: Learning heuristics for network selection: example output rule

7.3.2 Algorithm basics

Classic rule learning algorithms such as CN2 and FOIL (Clark & Niblett, 1989; Quinlan & Cameron-Jones, 1993) use two key concepts as the basis to their algorithms — the sequential covering loop and the general-to-specific search. The idea of sequential covering is to learn a set of rules, rule-by-rule. At each step, a rule is learnt that satisfies, or covers, as many positive training examples as possible, and no negative examples. After each rule is learnt, the positive training examples that are satisfied by the rule are removed from the set of examples. Then, if positive training examples remain in the set, the learning process continues and a new rule is learnt that again covers as many positive training examples as possible, and no negative examples. This iterative process continues until all positive training examples have been satisfied — in other words, no more positive training examples remain in the set.

Sequential covering is a method for iteratively learning a set of rules, but how exactly is each rule in the set constructed? The most popular approach for learning an individual rule is the general-to-specific search. In this approach, a search through the space of possible if-then rules is performed. Each rule encountered during the search is scored based on how many positive and negative training examples it covers. The search space is structured using the general-to-specific ordering. Figure 7.3 shows an example search space for the learning heuristics application. At the beginning of the search, very general rules, that cover many examples (both positive and negative) are considered. Usually, the search begins with the most general rule possible, which is the rule with the empty body. The rule classifies every example as the target concept, so it covers all positive examples, but also all negative examples. As the search progresses, the rule is specialised by adding attribute conditions to the body. The addition of attribute conditions to the rule body should reduce the number of examples that it covers. The idea is to reduce the negative examples covered by the rule, whilst maintaining a high number of covered positive examples. The search will stop when a pre-defined stopping condition is met. This may be when no negative examples are covered by the rule, or when the rule achieves a particular score that is based on the number of positive and negative training examples it covers.

7.3.3 Relational rule learning

In this research, we use a particular rule learning approach, known as relational rule learning, or Inductive Logic Programming (ILP) (Dzeroski & Lavrac, 2001; Lavrac & Dzeroski, 1994). ILP implements the same rule-learning algorithmic basis as described in Section 7.3.2, but uses first-order predicate logic (FOPL) to represent if-then rules. This means the rules are able to be more expressive than standard if-then rules, as they can contain variables and also make use of more complex data structures such as lists. The latter is particularly useful for the learning heuristics application, as it means the network predictive accuracies can be stored in an unordered list. This is beneficial as it means it is easier to deal

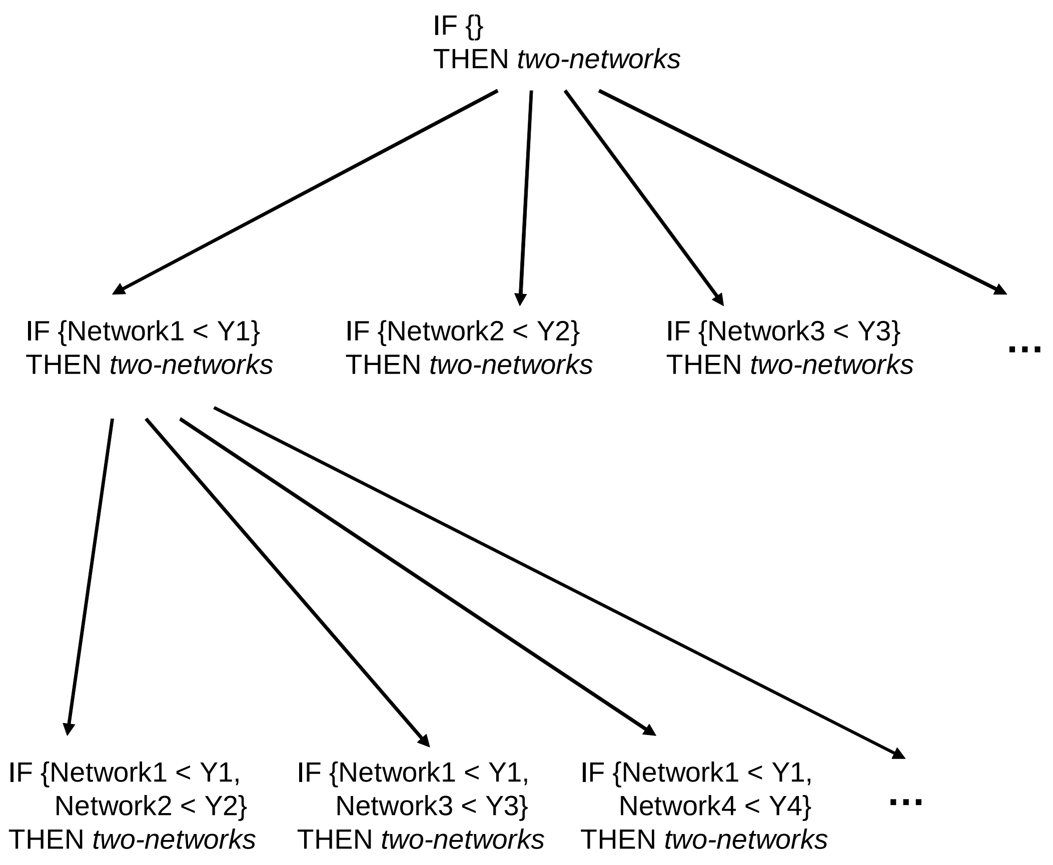


Figure 7.3: General to specific search for rules

with different numbers of input networks in each example. ILP also makes use of 'background knowledge' as input, in addition to the training examples data in order to construct rules. This is the features, or attribute conditions, that are used in the body of the learnt if-then rules, and since they are represented using FOPL they can be complex yet represented in a simple way.

In order to show how FOPL can represent the training examples, background knowledge (attributes) and output rules, we extend the 'toy' example for the learning heuristics application, which was used earlier in this section. Table 7.2 shows a set of training examples, with associated example data and background knowledge. The example data — the predictive accuracies of each network — are represented in list format, where networks are unordered — allowing the representation of cases with different numbers of input networks. For example, Set 23 has three input networks, and Set 34 has six input networks. At the bottom of the table, three background knowledge features are listed. The first is `n_networks_above(`*`N,PredAccBound`*`)`. This returns 'true' for a training example when *`N`* and *`PredAccBound`* are substituted for a number of networks and a predictive accuracy respectively, and there are *`N`* input networks with predictive accuracy above or equal to *`PredAccBound`*. For example, `n_networks_above(`*`4,0.4`*`)`, where *`N`* is 4 and *`PredAccBound`* is 0.4, satisfies the training examples Sets 1,2 and 34. Similarly, `n_networks_below(`*`N,PredAccBound`*`)` returns 'true' for a training example when there are *`N`* input networks with predictive accuracy below or equal to *`PredAccBound`*. Finally the background knowledge feature `predacc_cluster(`*`PredAccBound1,PredAccBound2`*`)` returns 'true' for a training example when there is a group of network predictive accuracies that are very close to each other (within 0.02), bounded by *`PredAccBound1`* below and *`PredAccBound2`* above. Note that it would be possible to represent these features in a standard propositional (non-relational) rule learning algorithm. However, FOPL provides an easier and more natural representation.

As described earlier, an if-then rule takes the format *IF* body *THEN* head, where the body is a list of conditions to be satisfied, and the head is the target concept. In FOPL, this is represented as follows:

head :- body

Target concept: *two-networks*

Training examples	Class label	Example data Predictive accuracies
Set 1	+	[0.50,0.46,0.44,0.40]
Set 2	+	[0.51,0.46,0.44,0.40]
Set 3	+	[0.40,0.36,0.35,0.33]
Set 4	-	[0.33,0.23,0.22,0.16]
Set 5	-	[0.30,0.18,0.18,0.18]
Set 6	+	[0.36,0.33,0.19,0.18]
Set 19	+	[0.30,0.32,0.30,0.28]
Set 23	-	[0.27,0.32,0.33]
Set 34	-	[0.52,0.53,0.47,0.49,0.50,0.50]
Background knowledge		
`n_networks_above(`*N,PredAccBound*`)`		
`n_networks_below(`*N,PredAccBound*`)`		
`predacc_cluster(`*PredAccBound1,PredAccBound2*`)`		

Table 7.2: Learning heuristics for network selection: FOPL inputs to the rule learning algorithm

An example rule is shown in Figure 7.4. This rule says that two input networks is optimal if there are at least 4 networks with predictive accuracy above 0.26, and at least 4 networks with predictive accuracy below 0.32, and there is a cluster of accuracies bounded below at 0.28. This rule satisfies one positive training example — Set 19 — and no negative examples.

```
two-networks  :-  n_networks_above(4,0.26),
                  n_networks_below(4,0.32),
                  predacc_cluster(0.28,OpenBound).
```

Figure 7.4: Learning heuristics for network selection: example output rule, represented using FOPL

7.3.4 ACE

The particular implementation of ILP that is used in this research is the ICL (inductive concept learning) component of the ACE data mining system (Blockeel *et al.*, 2002; De Raedt *et al.*, 2001). ACE uses the sequential covering loop and general-to-specific search to induce rules. It also has a number of other advantages. It uses a beam search when constructing individual rules. This means it considers multiple paths simultaneously in the search space, and therefore it is less likely to learn sub-optimal rules. It has mechanisms for dealing with noise, in particular by allowing induced rules to cover a number of negative training examples. It also has a number of different scoring functions that can be used to assess the accuracy of rules. Some scoring functions favour more general rules that cover a large number of positive examples, whilst still covering some negative examples, whilst other scoring functions favour specificity, by requiring that the number of covered negative examples is very small (which can be at the expense of the number of positive examples covered). ACE represents the training examples, attributes or background knowledge, and output rules using FOPL. The application of ACE to learn heuristics for the selection of input networks and consensus thresholds is presented next in Section 7.4.

7.4 Experimental results

This section reports on the rule learning carried out to produce heuristics for (a) the optimal number of input networks for the prediction-based selection approach, and (b) the optimal consensus threshold. First, in Section 7.4.1 the algorithm inputs — the target classes/concepts, training examples and background knowledge are described. The parameter settings in ACE are also detailed. Following this, Section 7.4.2 details the learnt rules for 2 network inputs and associated consensus thresholds are described. (Further results are provided in Appendix A. In Section 7.4.3 the results are discussed in more detail.

7.4.1 Rule learning inputs

For the set of training examples, the results from Chapter 6 (section 6.4) are used. This provides 18 training examples, each based on a synthetic true network and a set of synthetic microarray datasets, as described in Section 6.4.1. Additionally, further training examples were generated using SynTReN (Van den Bulcke *et al.*, 2006), an application for generating synthetic regulatory networks and microarray expression datasets. This enables learning to be carried out on a wider range of different collections of microarray datasets, and using synthetic networks means the results can still be evaluated against a 'gold standard' true network.

Given a global true network, SynTReN is able to generate synthetic microarray datasets based on the entire network, or a subnetwork. Subnetworks are generated using 'cluster-addition' or 'neighbour-addition' methods. The SynTReN application can be downloaded with sample global networks for yeast and *E. coli*, which have been built from databases of documented interactions. This means that extracted subnetworks are more likely to present true patterns of gene regulation, rather than an arbitrary network structure. In SynTReN, gene interactions are modelled using Michaelis-Menten and Hill kinetics. The user can also define the amount of biological and experimental noise that is present in the data.

A number of different cases (36) were generated using SynTReN. Each case was based on a different randomly-generated subnetwork from the yeast or *E. coli* global networks and contained between three and six microarray datasets. Each subnetwork was limited to a maximum of 30 genes for efficiency reasons. Each dataset had a varying number of samples — from 10 to 100, and random values set (between 1% and 50%) for biological and experimental noise.

This means that for each training example, the maximum possible number of input networks is between 3 and 6. Combining the original synthetic network examples with those generated using SynTReN provides a total of 54 training examples. Each example, with its data attributes (i.e. the predictive accuracy of each input network), is listed in Table A.4 in Appendix A.

The target classes for rule learning relate to the original objectives: to infer heuristics for the the optimal number of input networks, and the consensus threshold. First, we focus on the number of input networks. A class label is assigned to each training example, based on the optimal number of input networks for that case. If a prediction-weighted or individual input network has a better performance, these are ignored for these experiments, as the purpose is to find the optimal number of input networks for the prediction-based network selection Consensus approach. The assigned labels can be seen in Table A.4. Where the performance of the Consensus network on a different number of inputs is almost equal (where the AUCs are closer than 0.01), multiple labels may be assigned. For example, in Set 2, 2 or 4 network inputs are equally successful, so both labels are assigned. In total, out of all training examples 16 are labelled as `two-networks`, 17 are labelled as `three-networks`, 17 are labelled as `four-networks`, 8 are labelled as `five-networks` and 4 are labelled as `six-networks`. Rule learning is performed for each possible class label, where for each class the positive training examples are those examples assigned with that label, and the negative training examples are all the examples without that label.

The second objective of learning heuristics relates to finding the optimal consensus threshold. The consensus threshold intervals are dependent on the total number of input networks. For example, if there are four input networks, there are four possible consensus thresholds: 1-25%,26-50%,51-75% and 76-100%. Therefore, a set of heuristics is learnt based on each network selection class label: `two-networks`,`three-networks`,etc. The classes for the optimal consensus threshold were divided as follows:

2 datasets:	`low` : 1-50%	`high` 51-100%	
3 datasets:	`low` : 1-33%	`mid` : 34-66%	`high` : 67-100%
4 datasets:	`thres1` : 1-25%	`thres2` : 26-50%	`thres3` : 51-75%
	`thres4` : 76-100%		

Note that due to the low number of training examples labelled as `five-networks` or `six-networks`, learning the heuristics for the optimal consensus threshold based on five or six network inputs was not carried out. Each training example was assigned a class label for the optimal consensus threshold (see Table A.4). When learning the consensus threshold, for each class label combination the positive training examples are those examples assigned with both class labels, and the negative training examples are all those example labelled with the network selection class label, but without the consensus threshold class label. For example, when learning a heuristic for the low consensus threshold for two-networks examples, positive training examples are those labelled `low` and `two-networks`, and negative training examples are those labelled `high` and `two-networks`. Note that when learning for the consensus threshold, the data associated with each example includes only the predictive accuracies for the input networks used, and not those for all available networks. For example, if there were 4 possible input networks, but learning was carried out for the consensus threshold on 2 input networks, only the predictive accuracies for those 2 networks can be used in a learnt rule for the consensus threshold.

As well as class-labelled training examples, it is also necessary to provide background knowledge to the learning algorithm. The background knowledge is used as features for the body of the rules. Table 7.3 provides a list of the background knowledge used for learning, together with a natural language description for each feature. The first few features are fairly simple, relating to the total number of possible input networks, or the number of input networks with predictive accuracy above or below a certain bound. However, `predacc_cluster(`***`PredAccBound1, PredAccBound1`***`)` and `predacc_gap(`***`LowerBound1, LowerBound2, UpperBound1, UpperBound2`***`)` are slightly more complex. `predacc_cluster` satisfies training examples when there is a group of network predictive accuracies that are very close to each other (within 0.02), bounded by ***PredAccBound1*** below and ***PredAccBound2*** above. `predacc_gap` satisfies training examples where there is a large gap (greater than 0.07) between ordered predictive accuracies. For example, this would be satisfied by sets 2,4,5 and 6 in Figure 7.1. The accuracies at the lower end of the 'gap' are bounded between ***LowerBound1*** and ***LowerBound2*** and accuracies at the upper end of the 'gap' are bounded between ***UpperBound1*** and ***UpperBound2***.

The ACE data mining system (see Section 7.3.4) is used to learn rules based on the labelled training examples and background knowledge. ACE has a number of user-defined parameters. In the learning experiments carried out, the default parameters were used, except for the following. The minimum positive coverage was increased to 2, in order to avoid very specific rules that satisfy only one positive training example each. There are two possible rule scoring mechanisms that can be used to direct the search. These are the weighted relative accuracy (WRA), which favours more specific rules that satisfy few negative examples, and the standard m-estimate, which favours more general rules that cover larger numbers of positive examples, but this also risks covering more negative examples. Although we favour more general rules, we did use WRA where the m-estimate produced rules that covered a large number of negative examples (> 10).

7.4.2 Rule learning outputs

Next, the heuristics learnt are presented and commented on. In this section the rules for 2 network inputs are detailed. Learnt rules for 3-6 network inputs are covered in Appendix A. The following rules were learnt for `two-networks`, using the weighted relative accuracy scoring mechanism in ACE:

```
two-networks  :-  total_networks_below(4),
                  n_networks_above(2,0.33),
                  n_networks_below(3,0.49).
```

Covers 13/16 positive examples and 5/39 negative examples

Feature	Description
`total_networks_above(`*N*`)`	The total number of possible input networks is greater than or equal to *N*
`total_networks_below(`*N*`)`	The total number of possible input networks is greater than or equal to *N*
`total_networks(`*N*`)`	The total number of possible input networks is *N*
`n_networks_above(`*N,PredAccBound*`)`	There are *N* input networks with predictive accuracy greater than or equal to *PredAccBound*
`n_networks_below(`*N,PredAccBound*`)`	There are *N* input networks with predictive accuracy less than or equal to *PredAccBound*
`predacc_cluster(`*PredAccBound1, PredAccBound2*`)`	There is a group of network predictive accuracies that are within 0.02, bounded by *PredAccBound1* below and *PredAccBound2* above
`predacc_gap(`*LowerBound1 ,LowerBound2, UpperBound1,UpperBound2*`)`	There is a gap of over 0.07 in the network predictive accuracies such that accuracies at the lower end are bounded between *LowerBound1* and *LowerBound2* and accuracies at the upper end are bounded between *UpperBound1* and *UpperBound2*

Table 7.3: Learning heuristics: background knowledge features

This rule means that:

IF	the total number of possible input networks is 4 or below
AND	there are at least 2 networks with predictive accuracy above or equal to 0.33
AND	there are at least 3 networks with predictive accuracy below or equal to 0.49
THEN	two networks is the optimal input

```
two-networks  :-  total_networks_below(5),
                  n_networks_above(4,0.26),
                  n_networks_below(4,0.32),
                  predacc_cluster(0.28,OpenBound).
```

This rule means that:

IF	the total number of possible input networks is 5 or below
AND	there are at least 4 networks with predictive accuracy above or equal to 0.26
AND	there are at least 4 networks with predictive accuracy below or equal to 0.32
AND	there is a cluster of accuracies, bounded below at 0.28
THEN	two networks is the optimal input

Covers 2/3 positive examples and 0/39 negative examples

The first rule is fairly general, covering almost all the positive examples, although it also covers some negative examples. This rule only satisfies examples with 4 or fewer possible input networks, which is the majority of training examples. Although the second rule does not cover any negative examples, it is very specific as it only covers 2 positive examples.

Next, rules were learnt to establish which consensus threshold to apply, `low` or `high`, in order to obtain the optimal network performance:

```
low  :-  n_networks_below(1,0.43).
```

Covers 11/12 positive examples and 1/4 negative examples

This rule means that:

IF	One network (of the two with highest predictive accuracies) has predictive accuracy below or equal to 0.43
THEN	the optimal consensus threshold is low (1-50%)

```
high   :-  n_networks_above(2,0.46).
```

Covers 3/4 positive examples and 1/12 negative examples

This rule means that:

IF	Both networks have predictive accuracy above or equal to 0.46
THEN	the optimal consensus threshold is high (51-100%)

In general these rules specify that cases where there are higher predictive accuracies require a high consensus threshold, and cases with lower predictive accuracies require a low consensus threshold. The boundary for ‘lower’ and ‘higher’ predictive accuracies is found to be between 0.43 and 0.46.

7.4.3 Discussion

This section has presented the learning of heuristics, in the form of if-then rules for the selection of input networks and the consensus threshold for the prediction-based selection Consensus approach. The training examples were based on the synthetic cases generated for the experiments performed in chapter 6 and new networks and datasets generated using SynTReN. Each example was assigned class labels, based on the best-performing prediction-based selection Consensus network and consensus threshold. The background knowledge was based on simple features such as the total number of possible input networks, and the number of input networks with predictive accuracy above/below a certain bound. More complex background knowledge features representing ‘clusters’ and ‘gaps’ in the input network predictive accuracies were also used.

First, rules were learnt for network selection, that is, to find the optimal number of input networks. The target classes were `two-networks`, `three-networks`, `four-networks`, `five-networks` and `six-networks`. Note that when selecting two networks for example, we assume that these two networks are those with the highest predictive accuracies. The heuristics learnt are mainly based on the absolute values of the predictive accuracies. For example, a common rule structure is based on the requirement that a certain number of networks must have predictive accuracies between given bounds. Another feature of many rules is that they specify the total number of possible input networks in the example. This implies a dependency of the optimal number of input networks on the total number of available input networks, which might be expected. However, it also means that the learnt heuristics are slightly biased towards the examples with four possible input networks, as these make up the majority of the training examples. There were also not enough examples with a total number of input networks greater than 4, in order to construct reliable heuristics for `five-networks` or `six-networks`.

Second, rules were learnt for the selection of the optimal consensus threshold. Since the number of possible consensus threshold intervals is dependent on the number of input networks, a different set of rules was learnt for each number of input networks. A pattern is evident across the heuristics learnt for consensus threshold selection: cases where there are more networks with higher predictive accuracies require a higher consensus threshold, and cases with more networks that have lower predictive accuracies require a lower consensus threshold. The criteria for 'high' and 'low' predictive accuracies depends upon the number of consensus threshold intervals, and is stated in the learnt rules. This finding demonstrates that the consensus threshold is dependent on the coherency of the input network set. High predictive accuracies imply that a set of input networks are coherent — that is they fit across all the individual datasets from which they were generated. A set of input networks with low predictive accuracies may not be so coherent and the low accuracies may indicate that the networks are quite different. In this case, when a low consensus threshold is chosen for it means that edges appearing in only a few networks have a better chance of appearing in the final Consensus network, which can be beneficial when the networks are quite varied. Conversely, in the case where the set of input networks is highly coherent, high consensus edges are more likely to be robust and reliable. This is because the networks are more likely to be similar, so low consensus edges are more likely to be due to noise.

		Heuristics outcome		Actual outcome	
Test case	Predictive accuracies	Input networks	Consensus threshold	Input networks	Consensus threshold
Yeast heat-stress	[0.25,0.26,0.31,0.30,0.32]	2	low	3	mid
Yeast cell-cycle	[0.35,0.36,0.34,0.32]	4	thres2	4	thres2
E. coli SOS	[0.43,0.56,0.41,0.61]	-	high	2	high

Table 7.4: Learning heuristics: test case outcomes

7.5 Testing the heuristics on real data examples

The learnt heuristics presented in Section 7.4 are based on a set of synthetic training examples. This section discusses the application of the learnt rules to real data examples. In machine learning, testing learnt concept descriptions on unseen, real, examples is a key element of the learning process. In particular, it evaluates whether the learnt model overfits the training examples. Three real data examples are used to evaluate the heuristics. In Appendix A, the three test cases are described and commented on in detail. In this section we provide a discussion and summary of the test case results.

A summary of the findings is shown in Table 7.4. The input network selection is correctly predicted in 1/3 cases and the consensus threshold is correctly predicted in 2/3 cases. This is a small set of test cases, but does indicate that heuristics can assist in selecting input networks and consensus thresholds. In particular, the last two test cases confirm the general pattern of higher network predictive accuracies implying higher consensus thresholds. However, the real data evaluation has also highlighted two important issues:

- *Limited coverage of synthetic data training examples*
  For example, in the *E. coli* test case, no rules for network selection fit the example. In order to generate some reliable and general heuristics, a considerable number of training examples are required, covering a wide

range of networks and dataset noise levels. In these experiments, due to running time constraints, we only have 54 examples, which has proved to be not enough to cover one test case in this evaluation.

- *'Overlapping' heuristics*
  In the yeast cell-cycle test case, a number of heuristics for network selection were applicable. In this situation, there may need to be a process for resolving exactly how many datasets to use. For example, to use the smallest indicated number of input networks for simplicity reasons.

7.6 Summary

This chapter has presented the use of machine learning to induce rule-based heuristics, which can assist the user in selecting the optimal number of networks and the consensus threshold for input to the Consensus algorithm. Using established relational rule learning techniques, such heuristics were successfully induced based on synthetic training examples.

The learnt rules revealed interesting patterns in the input network predictive accuracies (a measure of network reliability or coherence), that could be used in deciding the number of input networks or consensus threshold. For example, a common rule structure was based on the requirement that a certain number of networks must have predictive accuracies between given bounds. A particular pattern of this type was evident across the heuristics learnt for consensus threshold selection: cases where there are more networks with higher predictive accuracies require a higher consensus threshold, and cases with more networks that have lower predictive accuracies require a lower consensus threshold. This finding implies an interesting consequence: the consensus threshold is dependent on the coherence, or similarity, within the set of input networks. A feature of many rules for input network selection is the specification of the total number of available input networks. Whilst some dependency of the optimal number of input networks on the total number of available input networks might be expected, it may also indicate some overfitting of the training examples.

The research presented in this chapter has only been the initial step in generating useable heuristics for selecting parameters for the Consensus approach. Whilst the test data evaluation showed that the learnt heuristics can be easily and usefully applied to real data, they were only able to make successful predictions in the yeast cell-cycle network case. The evaluation established that the set of training examples was not broad enough to cover some of the test cases. For example, in these experiments the number of available input networks only varied between three and six. In order to construct a set of heuristics that can be applied to almost any situation, the learning process would need many more cases, that vary across the base network, the number of available datasets and the microarray data noise levels. This would allow global patterns to be uncovered.

Chapter 8

Conclusions

This chapter draws together the conclusions reached based on the research presented in this thesis. First, the main contributions are summarised. This is followed by a discussion of the limitations of the research presented. Finally, potential avenues for further research are presented, based on both addressing the research limitations and extending the applicability of the work.

8.1 Thesis contributions

Gene Regulatory Networks (GRNs) describe how the expression level, or activity, of genes affect the expression of the other genes. Microarray technology allows the expression of thousands of genes to be measured simultaneously and is the major source of data for reverse-engineering GRN models based on gene expression levels. Microarrays are widely used, which has led to many publicly available datasets of gene expression measurements and subsequently an explosion of research in the reverse-engineering of GRN models based on microarray-generated data. However, the technology has a number of limitations as a data source for the modelling of GRNs, due to concerns over reliability and the reproducibility of experimental results. This research presented in this thesis has focused on the use of additional data sources - either complementary prior knowledge, or multiple microarray studies - to alleviate these limitations in using a single microarray study for the reverse engineering of GRNs. The following subsections summarise the contributions made with respect to reverse-engineering Bayesian network based models of GRNs.

8.1.1 Incorporation of prior knowledge

The drawbacks of using only microarray data to reconstruct GRNs can be alleviated by incorporating other complementary data sources as prior knowledge in the modelling process. There are many other data sources that contribute to available knowledge on GRNs, for example experimental-based data such as transcription factor binding site location knowledge or protein-protein interaction data, as well as text-based knowledge, which includes information locked in scientific papers.

This thesis has presented some of the first research in the incorporation of prior knowledge that has been generated from a large body of scientific papers, for Bayesian network based GRN models. The use of advanced text-mining techniques means information contained in a huge number of documents (for example from a database of papers such as Medline) can be represented in a simple gene-pair association matrix format. Chapter 4 presented a method for integrating this information into learning Bayesian network based GRN models by translating it into a prior probability distribution over candidate network structures.

An empirical evaluation, using data and networks for three different organisms, showed that the use of literature-based prior knowledge can improve both the number of true regulatory interactions present and the predictive performance of the learnt model, in comparison to a network learnt solely from expression data. In particular, varying the influence of the prior knowledge on the learning process was considered through the use of weighting. The experimental results indicated that careful weighting of the prior knowledge does appear to be needed. A low weighting on the prior knowledge can mean many spurious gene interactions are included in networks. Conversely, a high weighting may result in the inclusion of network edges that do not reflect regulatory relationships, as there is less emphasis on the expression data. Furthermore, the most suitable weighting value may be related to the amount of reliable prior knowledge available. Where there is less literature, for example with the human organism, the best results were obtained when less weight was assigned to the prior knowledge, in comparison to the yeast and *E. coli* organisms which required higher prior weights for the best results. These are well-studied organisms for which there is a large amount of literature.

8.1.2 Combining multiple microarray datasets

The rapid increase of publicly available microarray data provides the opportunity to produce GRN models based on multiple microarray datasets. Such models are potentially more robust with greater confidence, and place less reliance on a single dataset. Chapter 5 introduced the concepts of *pre-* and *post-learning aggregation*. In pre-learning approaches, such as using simple scale normalisation prior to the concatenation of datasets, a model is learnt from a combined dataset, whilst in post-learning aggregation individual models are learnt from each dataset and the models are combined. The resulting combined model represents prominent features which occur in all, or a subset of, the individual dataset models. A key advantage of the post-learning aggregation framework is that it can combine microarray datasets generated by different platforms, research groups and laboratories without requiring normalisation. This thesis has presented two novel post-learning aggregation methods for Bayesian network based GRN modelling. *Bayesian networks meta-analysis* is based on combining statistical confidences that are attached to network edges whilst *Consensus Bayesian networks* identifies consistent network features across all datasets.

Chapter 5 compared pre- and post-learning aggregation through an empirical evaluation on synthetic and real collections of microarray datasets. This demonstrated that both post-learning aggregation methods can produce GRN models that improve on models learnt from a single dataset or a combined dataset (i.e. a pre-learning aggregation approach). Overall, Consensus Bayesian networks was the better-performing aggregation approach, as it identifies consistencies amongst the collection of input networks and so it is least affected by those networks that perform poorly. However, there is room for improvement in the method - in particular, it is a parameter-heavy method when used in conjunction with bootstrapped input networks. An additional finding was that in some cases, using only a subset of available datasets produced a better performing Consensus model than when using all available datasets.

8.1.3 Incorporating dataset selection or weighting based on reliability

In order to further investigate the effect of input dataset quality when combining multiple microarray datasets, Chapter 6 presented further development of the Consensus Bayesian networks approach. An improved Consensus approach was presented - Consensus Bootstrapped Bayesian Networks (CBBNs), which allows the more robust bootstrapped network models to be used as direct inputs to the Consensus algorithm, meaning a reduction in the number of parameters. Additional improvements to the technique also allows the incorporation of weighting of the input networks based on their reliability or quality. A measure of reliability for sets of networks, based on the prediction of node values was introduced, together with methods for incorporating the use of network reliability measures into the Consensus algorithm. Prediction-based network weighting allows different weights to be assigned to each network in order to vary the influence of each input network, whilst prediction-based network selection uses only a subset of the available input networks, based on their reliability, in order to produce a Consensus model.

An empirical evaluation compared prediction-based network selection (i.e. using individual input networks generated by only a subset of available datasets) with prediction-based weighting. This demonstrated that network selection provides a more consistent improvement in GRN model performance than using network weighting.

However, an important problem remained concerning how to select the optimal number of input networks and the consensus threshold. Whilst relationships between the reliability of input networks and the optimum consensus threshold were indicated in the analysis of the experimental results, the small-scale of the evaluation meant no firm conclusions could be established here. As a first step in addressing this issue, Chapter 7 presents preliminary research exploring the use of machine learning to induce heuristics, ‘rules of thumb’, in the form of classification rules, for selecting the input networks and consensus threshold. This alleviates the issue of choosing parameters for the user.

8.2 Limitations

Empirical evaluation has shown that the incorporation of literature-based prior knowledge or combining multiple microarray datasets using Consensus Bayesian networks can provide a positive improvement over the use of a single microarray datasets for the reverse-engineering of Bayesian network based GRN models. However, there are a number of limitations to the research presented in this thesis, which are discussed in this section.

First, the techniques presented have only been designed for and tested with static Bayesian networks. In static Bayesian networks, temporal interactions are not considered. As discussed in section 3.2.3, modelling temporal interactions in gene regulation is important, as it allows cyclic behaviour to be represented. Additionally, time-delayed gene interactions cannot be presented in static Bayesian networks. However, developing the techniques for static models has provided a solid foundation from which to extend the techniques for dynamic behaviour. This is discussed next in section 8.3 as possible future work.

A second point is that all empirical evaluations in this thesis have used discretised microarray gene expression datasets. Discretisation is a technique commonly applied to gene expression data when reverse-engineering GRN models and is of particular benefit with real datasets that have a small number of samples and/or can be noisy. It is a simple method that still allows complex regulatory structures to be modelled whilst avoiding the need to deal with complex parameterised continuous distributions. However, it means that conclusions cannot be drawn about the broader use of the techniques, when applied to continuous gene expression data.

Furthermore, in the evaluation of Consensus Bayesian networks, the groups of multiple microarray datasets used were relevant to the network under consideration (for example, *E. coli* datasets from DNA damage experiments were used for the SOS response module). The use of different types of experimental studies with the Consensus process may lead to problems concerning the semantics of the input networks. In other words, the edges in different input networks may have subtly different meanings based upon the different experiments from which they have been generated. Additionally, the networks considered were small in size. Therefore, the applicability of the techniques to a broader range of microarray studies and larger global-size networks has not been considered.

8.3 Further work

The following subsections outline potential avenues for future research, which are based on addressing the limitations discussed in the previous section, and/or extending the applicability of the techniques presented in this thesis.

8.3.1 Extension of modelling techniques

Additional further work could involve extending the modelling techniques in a number of ways. As discussed in section 8.2, the techniques presented in this thesis have only been used with static Bayesian networks, although modelling temporal behaviour is important for GRNs. In particular, this should improve the directionality of learnt interactions and allow cyclic behaviour to be introduced to the models. Temporal information can be incorporated through time nodes and dynamic BNs (which were briefly discussed in chapter 3). The methodology for the incorporation of prior knowledge should be directly applicable to dynamic BNs. However, the Consensus Bayesian networks algorithms may need to be adjusted to deal with temporal nodes.

8.3.2 Optimising the weighting influence of prior knowledge

As discussed in section 8.1.1, when incorporating prior knowledge into Bayesian network based GRN models the most suitable weighting value may be related to the amount of reliable prior knowledge available. It would be useful to investigate this further in order to develop a heuristic for setting the value of the prior weight.

8.3.3 Applicability of Consensus Bayesian networks to diverse microarray studies

As discussed in section 8.2, Consensus Bayesian networks has only been evaluated with small networks on sets of similar microarray studies. Further research could investigate whether more diverse datasets could be combined as effectively, and for larger global networks. Some improvements to the technique may be required, for example by using additional nodes to represent the experiment or study type.

8.3.4 Combining prior knowledge and multiple microarray datasets

In this thesis, the incorporation of prior knowledge and the combination of multiple microarray datasets for the reverse-engineering of GRN models have been considered separately. A next step is to consider integrating both multiple microarray datasets and prior knowledge in the same learning process. There are a number of ways in which this could be carried out. For example, for use with Consensus Bayesian networks, prior knowledge could be integrated separately into each individual input network using the methods described in Chapter 4. An alternative method would be to integrate the prior knowledge into the combined network produced by Consensus Bayesian networks. Since Consensus Bayesian networks is based on combining networks, the technique has the potential to integrate many other heterogeneous types of data - provided that network models can be built from these datasets. Therefore, the incorporation of other data sources or expert knowledge such as transcription factor binding sites, protein-protein interaction data and textual information extracted from scientific literature, could be incorporated into the combined GRN model in this way.

8.3.5 New expression data technology

High-throughput technology for gene expression is constantly evolving. Microarray technology was developed in the 1990s and first used for a genome-wide expression study for yeast in 1997. Now, a decade later, new technologies are emerging for measuring the expression of huge numbers of genes. Most notable are SAGE (Serial Analysis of Gene Expression), LongSAGE and MPSS (Massively Parallel Signature Sequencing). These technologies represent the 'digital' age for gene expression profiling and can provide more reliable and less noisy data; rather than an estimate of expression that is available with microarrays, these new technologies can give an absolute count of the mRNA produced by the expression of a gene.

These technologies are new and expensive and are not yet being used as widely as microarrays. As a result, few datasets exist in the public domain. However, it would be relatively easy to adapt current microarray analysis techniques to work on these new types of expression data. Therefore, in future the new techniques presented in this thesis could be adapted for use on these new types of expression data. For example, Consensus Bayesian networks could be used to combine models from both new types of expression data and microarray expression data.

Appendix A

Additional tables and results

This appendix contains additional tables and results relating to Chapters 6 and 7.

A.1 Chapter 6 additional tables and results

Table A.1 details the different collections of synthetic input networks for the experiments in Chapter 6. Table A.2 provides full details of the performance comparison of different approaches on the synthetic networks. Finally, Table A.3 gives full details for the method comparison results on prediction-based selection.

Network set ID	Gaussian noise variance added to each dataset 1	2	3	4	Collective reliability level
1	clean	clean	clean	clean	0.45
2	0.1	0.1	0.1	0.1	0.44
3	0.1	0.1	0.1	0.8	0.40
4	0.2	0.2	0.1	0.1	0.38
5	0.2	0.2	0.2	0.2	0.36
6	0.2	0.2	0.8	0.2	0.35
7	0.2	0.8	0.1	0.1	0.31
8	0.8	0.4	0.1	0.1	0.31
9	variable noise by gene				0.31
10	variable noise by gene				0.31
11	0.1	0.1	0.8	0.4	0.30
12	variable noise by gene				0.28
13	variable noise by gene				0.27
14	0.4	0.1	0.1	0.4	0.26
15	0.1	0.2	0.4	0.8	0.26
16	0.2	0.4	0.8	0.4	0.23
17	0.4	0.4	0.4	0.4	0.23
18	0.8	0.8	0.8	0.8	0.18

Table A.1: Collections of input networks for synthetic datasets (generated by differential equations). This table provides details of the Gaussian variance added to each of the datasets for each set of input networks. The 'collective reliability' of a network set is measured by the median of the median predictive accuracies for the networks generated from each dataset. Note that in general, the Gaussian noise applied across the datasets increases as the collective reliability level of the network set decreases.

Network	CBBN		Prediction-based weighting		Prediction-based selection			Best input network	Normalisation only
set ID	Maximum AUC	Consensus threshold	Maximum AUC	Consensus threshold	No. datasets	Maximum AUC	Consensus threshold	Maximum AUC	Maximum AUC
1	0.83	76-100%	0.84	75%	2	0.84	51-100%	0.82	0.69
2	0.85	26-50%	0.88	28%	2	0.85	1-50%	0.82	0.77
3	0.87	1-25%	0.88	14-28%	3	0.88	1-33%	0.82	0.76
4	0.81	1-25%	0.87	25,26%	2	0.84	1-50%	0.75	0.73
5	0.76	1-25%	0.84	25%	2	0.84	1-50%	0.74	0.75
6	0.75	1-25%	0.75	1-13%	2	0.84	1-50%	0.74	0.63
7	0.70	1-25%	0.74	48-52%	3	0.74	34-66%	0.75	0.72
8	0.75	26-50%	0.75	30-43%	4	0.75	26-50%	0.75	0.71
9	0.63	51-75%	0.63	57-70%	4	0.63	51-75%	0.63	0.63
10	0.70	26-50%	0.73	27,28%	2	0.74	1-50%	0.77	0.71
11	0.82	1-25%	0.84	23-31%	2	0.85	1-50%	0.82	0.70
12	0.70	26-50%	0.70	32-40%	4	0.70	26-50%	0.66	0.53
13	0.67	1-25%	0.67	15-24%	3	0.67	1-33%	0.68	0.62
14	0.79	51-75%	0.79	56-68%	4	0.79	51-75%	0.82	0.72
15	0.79	1-25%	0.81	18-31%	2	0.81	1-50%	0.77	0.77
16	0.76	1-25%	0.76	1-18%	4	0.76	1-25%	0.72	0.60
17	0.70	26-50%	0.70	35-40%	4	0.70	26-50%	0.70	0.83
18	0.62	1-25%	0.62	1-19%	4	0.62	1-25%	0.60	0.53

Table A.2: Synthetic networks: approaches performance comparison summary

Network set ID	Optimum no. of of input networks (n)	Highest median predictive accuracy of *unselected* networks (x)	Lowest median predictive accuracy of *selected* networks
1	2	0.44	0.46
2	2	0.40	0.48
3	3	0.20	0.39
4	2	0.39	0.39
5	2	0.36	0.36
6	2	0.34	0.34
7	3	0.22	0.28
8	4	0.20	0.20
9	4	0.18	0.18
10	2	0.32	0.32
11	2	0.25	0.34
12	4	0.16	0.17
13	3	0.34	0.35
14	4	0.25	0.25
15	2	0.19	0.33
16	4	0.17	0.17
17	4	0.15	0.15
18	4	0.16	0.16

Table A.3: Method comparison for prediction-based selection

A.2 Chapter 7 additional tables and results

A.2.1 Training examples for learning

Table A.4 lists the training examples and attributes for the learning heuristics application presented in Chapter 7.

Training examples	Example data Predictive accuracies	Class labels # input networks	consensus threshold
Set 1	[0.50,0.46,0.44,0.40]	two-networks	high
Set 2	[0.51,0.46,0.44,0.40]	two-networks	low
		four-networks	
Set 3	[0.40,0.36,0.35,0.33]	two-networks	low
Set 4	[0.33,0.23,0.22,0.16]	four-networks	thres1
Set 5	[0.30,0.18,0.18,0.18]	four-networks	thres1
Set 6	[0.36,0.33,0.19,0.18]	two-networks	low
		three-networks	mid
		four-networks	thres1
Set 7	[0.35,0.23,0.23,0.18]	four-networks	thres1
Set 8	[0.41,0.41,0.40,0.19]	three-networks	low
		four-networks	thres1
Set 9	[0.35,0.35,0.15,0.25]	two-networks	low
Set 10	[0.30,0.21,0.32,0.35]	four-networks	thres2
Set 11	[0.35,0.21,0.28,0.35]	three-networks	mid
Set 12	[0.43,0.39,0.28,0.36]	two-networks	low
Set 13	[0.37,0.35,0.16,0.34]	two-networks	low
Set 14	[0.36,0.25,0.27,0.25]	four-networks	thres1
Set 15	[0.33,0.32,0.18,0.29]	two-networks	low
Set 16	[0.33,0.26,0.17,0.30]	four-networks	thres2
Set 17	[0.32,0.26,0.15,0.30]	three-networks	low
		four-networks	thres1
Set 18	[0.33,0.31,0.18,0.30]	four-networks	thres3
Set 19	[0.30,0.32,0.30,0.28]	two-networks	low
Set 20	[0.38,0.41,0.37,0.35,0.41]	four-networks	thres2
Set 21	[0.38,0.37,0.35,0.41]	three-networks	high
Set 22	[0.35,0.35,0.35,0.37]	three-networks	high
Set 23	[0.35,0.35,0.37]	two-networks	high
Set 24	[0.30,0.28,0.25,0.26,0.26]	two-networks	low
		five-networks	-
Set 25	[0.27,0.32,0.33]	three-networks	low
Set 26	[0.38,0.36,0.33,0.34]	three-networks	low
Set 27	[0.36,0.33,0.34]	two-networks	low

Training examples	Example data Predictive accuracies	Class labels # input networks	consensus threshold
Set 28	[0.44,0.43,0.41,0.43,0.44]	three-networks	low
Set 29	[0.44,0.41,0.43,0.44]	two-networks	low
Set 30	[0.30,0.28,0.34,0.32,0.29,0.28]	six-networks	-
Set 31	[0.30,0.34,0.32,0.29,0.28]	five-networks	-
Set 32	[0.45,0.49,0.30,0.49]	two-networks	high
Set 33	[0.45,0.49,0.49]	two-networks	high
Set 34	[0.49,0.49,0.49]	three-networks	high
Set 35	[0.57,0.69,0.55,0.66,0.69]	five-networks	-
Set 36	[0.57,0.69,0.66,0.69]	four-networks	thres4
Set 37	[0.82,0.84,0.84,0.85,0.82,0.82]	five-networks	-
Set 38	[0.82,0.84,0.85,0.82,0.82]	four-networks	thres3
Set 39	[0.30,0.29,0.32]	three-networks	low
Set 40	[0.30,0.25,0.24,0.30,0.25]	five-networks	-
Set 41	[0.30,0.25,0.24,0.30]	four-networks	thres1
Set 42	[0.30,0.25,0.24]	three-networks	low
Set 43	[0.50,0.50,0.43]	three-networks	mid
Set 44	[0.52,0.53,0.47,0.49,0.50,0.50]	six-networks	-
Set 45	[0.53,0.47,0.49,0.50,0.50]	five-networks	-
Set 46	[0.53,0.49,0.50,0.50]	three-networks	mid
Set 47	[0.49,0.50,0.50]	three-networks	low
Set 48	[0.27,0.26,0.26,0.30]	four-networks	thres1
Set 49	[0.26,0.26,0.30]	three-networks	low
Set 50	[0.33,0.34,0.29,0.32,0.34,0.31]	six-networks	-
Set 51	[0.33,0.29,0.32,0.34,0.31]	five-networks	-
Set 52	[0.33,0.32,0.34,0.31]	four-networks	thres2
Set 53	[0.32,0.34,0.31]	three-networks	mid
Set 54	[0.30,0.28,0.27,0.25,0.26,0.26]	six-networks	-

Table A.4: Learning heuristics: training examples and class labels

A.2.2 Learnt heuristic rules

In this section, learnt heuristic rules for 3-6 network inputs are presented.

A.2.2.1 Rules for 3 network inputs

The following rules were learnt for `three-networks`, using the standard *m*-estimate scoring mechanism in ACE:

```
three-networks  :-  total_networks_below(3),
                    n_networks_below(1,0.31).
```

Covers 5/17 positive examples and 0/38 negative examples

This rule means that:

IF	the total number of possible input networks is 3 or below
AND	there is at least 1 network with predictive accuracy below or equal to 0.31
THEN	three networks is the optimal input

```
three-networks  :-  total_networks(4),
                    n_networks_above(4,0.18),
                    n_networks_below(4,0.43),
                    predacc_cluster(OpenBound,0.35).
```

Covers 5/12 positive examples and 1/38 negative examples

This rule means that:

IF	the total number of possible input networks is 4
AND	there are at least 4 networks with predictive accuracy above or equal to 0.18
AND	there are at least 4 networks with predictive accuracy below or equal to 0.43
AND	there is a cluster of accuracies, bounded above at 0.35
THEN	three networks is the optimal input

```
three-networks  :-  total_networks_below(5),
                    n_networks_above(2,0.44),
                    n_networks_below(2,0.50),
                    predacc_cluster(OpenBound,0.43).
```

Covers 5/7 positive examples and 3/38 negative examples

This rule means that:

IF	the total number of possible input networks is 5 or below
AND	there are at least 2 networks with predictive accuracy above or equal to 0.44
AND	there are at least 2 networks with predictive accuracy below or equal to 0.50
AND	there is a cluster of accuracies, bounded above at 0.43
THEN	three networks is the optimal input

These three rules effectively divide the positive training examples into categories by the number of possible input networks. The first rule covers cases with 3 networks, the second only covers cases with 4 networks. The third rule covers cases with 5 or fewer input networks. This means they are fairly specific rules. Similarly to the rules for `two-networks` they are mainly based on bounding the predictive accuracy of the input networks.

Next, rules were learnt to establish which consensus threshold to apply, `low`, `mid` or `high`, in order to obtain the optimal network performance with three input networks:

```
low    :-  n_networks_below(3,0.33).
```

Covers 5/9 positive examples and 0/8 negative examples

This rule means that:

IF	all 3 networks have predictive accuracy below or equal to 0.33
THEN	the optimal consensus threshold is low (1-33%)

```
mid    :-  n_networks_below(1,0.31),
           n_networks_above(1,0.34).
```

Covers 3/5 positive examples and 0/12 negative examples

This rule means that:

IF	there is at least 1 network with predictive accuracy below or equal to 0.31
AND	there is at least 1 network with predictive accuracy above or equal to 0.34
THEN	the optimal consensus threshold is mid (34-66%)

```
high   :-  n_networks_above(3,0.35),
           n_networks_below(2,0.49).
```

Covers 3/3 positive examples and 2/14 negative examples

This rule means that:

IF	all 3 networks have predictive accuracy above or equal to 0.35
AND	there are at least 2 networks with predictive accuracy below or equal to 0.49
THEN	the optimal consensus threshold is high (67-100%)

Similarly to the consensus threshold rules for two-networks, these rules specify that cases where there are higher predictive accuracies require a higher consensus threshold, and cases with lower predictive accuracies require a lower consensus threshold. The boundary for 'lower' and 'higher' predictive accuracies seems to be found to be between around 0.34. For example in the rule for the `low` threshold, all three networks should have predictive accuracy below or equal to 0.33. For the `mid` threshold there should be at least one network with accuracy above 0.34, but also one network with predictive accuracy below 0.31. Finally, the rule for the `high` threshold specifies that all 3 input networks should have predictive accuracy above 0.35.

A.2.2.2 Rules for 4 network inputs

The following rules were learnt for `four-networks`, using the standard *m*-estimate scoring mechanism in ACE:

```
four-networks  :-  predacc_gap(LB1,LB2,UB1,0.27),
                   n_networks_below(3,0.31).
```

Covers 8/17 positive examples and 0/38 negative examples
This rule means that:

IF	there is a 'gap' in the predictive accuracies, where the accuracies at the upper end of the gap are bound above by 0.27
AND	there is at least 1 network with predictive accuracy below 0.31
THEN	four networks is the optimal input

```
four-networks  :-  total_networks(4),
                   n_networks_above(4,0.18),
                   n_networks_below(1,0.32),
                   predacc_cluster(0.44,OpenBound).
```

Covers 5/9 positive examples and 3/38 negative examples
This rule means that:

IF	the total number of possible input networks is 4
AND	there are at least 4 networks with predictive accuracy above 0.18
AND	there is at least 1 network with predictive accuracy below 0.32
AND	there is a cluster of accuracies, bounded below at 0.44
THEN	four networks is the optimal input

The first rule covers examples with very low predictive accuracies — there is a gap, where the accuracies at the higher end of the gap are below 0.27, and at least three networks have predictive accuracy below 0.31. The second rule covers cases where the accuracies are higher — for example, there is a cluster of accuracies around 0.44.

Next, rules were learnt to establish which consensus threshold to apply, `thres1` or `thres2`, in order to obtain the optimal network performance. Rules are not learnt for the other thresholds, due to a lack of positive training examples in these classes — there are only 2 examples for `thres3`, and 1 example for `thres4`.

```
thres1  :-  n_networks_above(2,0.19),
            n_networks_below(2,0.27).
```

Covers 7/8 positive examples and 1/9 negative examples
This rule means that:

IF	there are at least 2 networks with predictive accuracy above or equal to 0.19
AND	there are at least 2 networks with predictive accuracy below or equal to 0.27
THEN	the optimal consensus threshold is thres1 (1-25%)

```
thres2  :-  n_networks_below(2,0.39),
            n_networks_above(1,0.34).
```

Covers 5/5 positive examples and 3/12 negative examples
This rule means that:

IF	there are at least 2 networks with predictive accuracy below or equal to 0.39
AND	there is at least 1 networks with predictive accuracy above or equal to 0.34
THEN	the optimal consensus threshold is thres2 (26-50%)

Again, these rules specify that cases where there are higher predictive accuracies require a higher consensus threshold, and cases with lower predictive accuracies require a lower consensus threshold. For a threshold between 1-25%, there should be at least two networks with predictive accuracy above 0.19 and two networks with predictive accuracy under 0.27. For a threshold between 26-50%, there should be at least one network with predictive accuracy above 0.34 and one network with predictive accuracy under 0.39.

A.2.2.3 Rules for 5 or 6 network inputs

Finally, ACE was used to construct rules for `five-networks` and `six-networks`, using the WRA scoring mechanism in ACE:

```
five-networks  :-  total_networks(5),
                   n_networks_above(4,0.46).
```

Covers 3/8 positive examples and 0/47 negative examples

This rule means that:

IF	there are 5 possible input networks
AND	there are at least 4 networks with predictive accuracy above or equal to 0.46
THEN	five networks is the optimal input

```
five-networks  :-  total_networks(5),
                   n_networks_above(1,0.29).
```

Covers 3/5 positive examples and 0/47 negative examples

This rule means that:

IF	there are 5 possible input networks
AND	there is at least 1 network with predictive accuracy below or equal to 0.29
THEN	five networks is the optimal input

```
six-networks  :-  total_networks(6).
```

Covers 4/4 positive examples and 2/51 negative examples

This rule means that:

IF	the total number of possible input networks is 6
THEN	six networks is the optimal input

Due to a lack of training examples for `five-networks` and `six-networks`, these rules are not very useful. They mainly rely on the number of possible input networks since there are so few examples with five or six possible input networks. Although the rules for `five-networks` do include other conditions as well, they are very specific rules and likely to overfit the training data.

A.2.3 Test case results

In this section the full details of applying the learnt heuristics real data test cases are provided.

A.2.3.1 Yeast: heat-stress network

The first real data example used is the yeast heat-stress network used in chapters 5 and 6. This is a sub-network of 9 regulatory genes that are related to heat-shock response. The five microarray datasets are publicly available on the YeastBASE expression database — see Table 5.3 for more details. The learnt networks are evaluated by comparing them to documented gene interactions, obtained from the online YEASTRACT database (Teixeira *et al.*, 2006). The heuristics presented in Section 7.4.3 were applied to this case in order to find the optimal network selection and consensus threshold. A comparison of the optimal network selection and consensus threshold prediction using the heuristics, compared to the actual outcome, is shown in Table 7.4. Figure A.1 shows the AUC performance of the prediction-based selection networks, each based on a different number of selected in input networks.

The learnt heuristics are not able to correctly predict the number of input networks as 3. This is because the example matches the second rule for `two-networks`, which allows for five available input networks.

A.2.3.2 Yeast: cell-cycle network

The second real data case again concerns yeast microarray expression datasets, but this time focuses on the cell-cycle network. The subnetwork consists of 19 genes that are involved in the cell-cycle process (Spellman *et al.*, 1998). Four publicly available microarray datasets were used, each generated by experiments focusing on the yeast cell-cycle — more details can be found in Table A.5. The heuristics presented in Section 7.4.3 were applied to each case in order to find

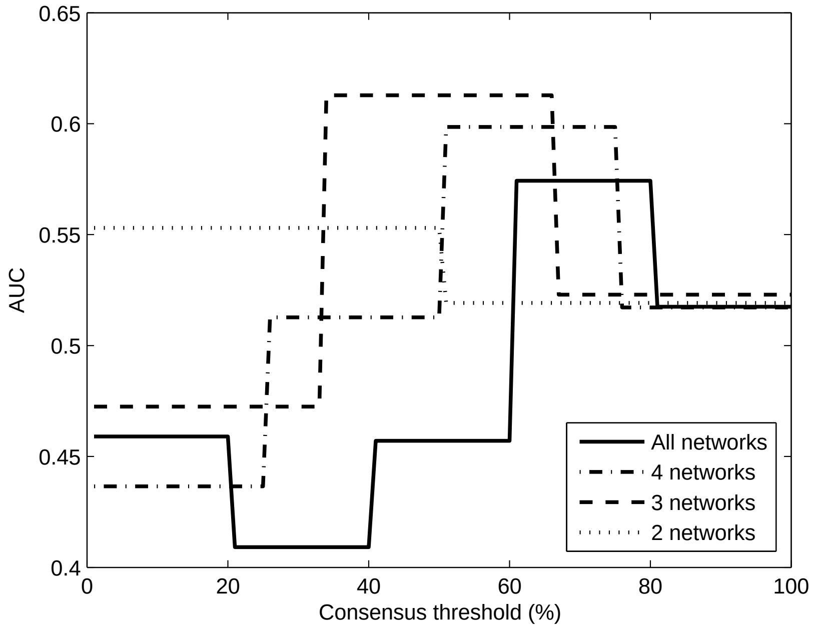


Figure A.1: Test case: yeast heat-stress network AUC plot

the optimal network selection and consensus threshold. A comparison of the optimal network selection and consensus threshold prediction using the heuristics, compared to the actual outcome, is shown in Table 7.4. Figure A.2 shows the AUC performance of the prediction-based selection networks, each based on a different number of selected in input networks.

The learnt heuristics are able to correctly predict the number of input networks as 4 and the consensus threshold as `thres2` (between 26-50%). This case fits the second rule for `four-networks`, which requires all four network predictive

Dataset	Description	Number of Observations
Pramila *et al.* (2006)	Cell-cycle	50
Pramila *et al.* (2002)	Cell-cycle	13
Spellman *et al.* (1998)	Cell-cycle	77
Zhu *et al.* (2000)	Cell-cycle	13

Table A.5: Summary of yeast cell-cycle datasets

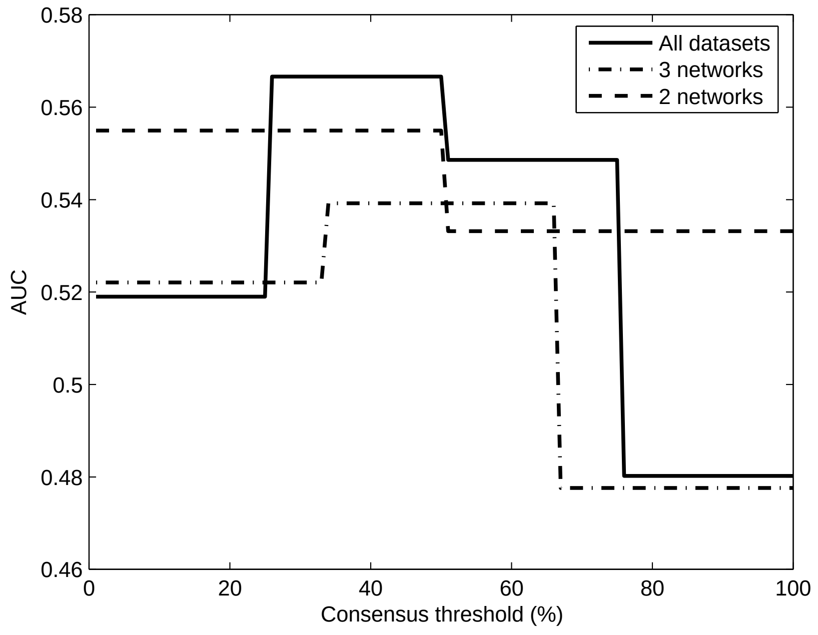


Figure A.2: Test case: yeast cell-cycle network AUC plot

accuracies to be above 0.18, and one network predictive accuracy to be above 0.32. It also meets the criteria for `thres2`, where one network predictive accuracy must above 0.34, and two must be below 0.39. However, we also note that this case also meets the criteria for `two-networks` and `three-networks`. This is not unexpected if we recall that some training examples had multiple labels, if two or more Consensus networks (different number of network inputs) all performed very well. In this test case, we can see that the Consensus networks generated from both two and four networks outperform all input networks and their AUCs are within 0.015. Additionally, the consensus threshold predicted for two network inputs is `low`, which is optimal in this case.

A.2.3.3 *E. coli*: SOS response network

The final real data example is the *E. coli* SOS-response network, used in chapter 5. This is a subnetwork of 19 target genes and one transcriptional repressor, LexA (see Figure 5.3 in Chapter 5). Table 5.2 provides a summary of the four

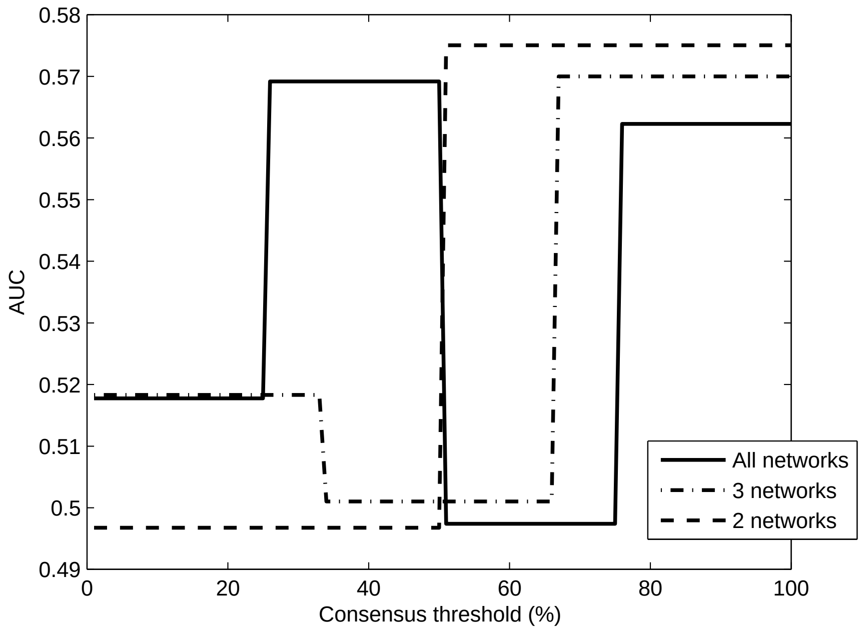


Figure A.3: Test case: *E. coli* SOS-response network AUC plot

available microarray expression datasets, which are all focused on experiments related to SOS response. The heuristics presented in Section 7.4.3 were applied to the case in order to find the optimal network selection and consensus threshold. A comparison of the optimal network selection and consensus threshold prediction using the heuristics, compared to the actual outcome, is shown in Table 7.4. Figure A.3 shows the AUC performance of the prediction-based selection networks, each based on a different number of selected in input networks.

The learnt heuristics are not able to predict the number of input datasets at all — the example does not fit any of the rules for `two-networks`, `three-networks` or `four-networks`. This is because this case has four networks that all have reasonably high predictive accuracies — and there are few synthetic cases in the training examples similar to this. However, the relatively high predictive accuracies do imply that a higher consensus threshold is more appropriate. Indeed, the actual optimal number of input networks is 2, and the heuristics are able to predict the correct consensus threshold, since the example matches the second rule for the `high` threshold with two input networks, which is indicated for higher predictive accuracies, as we have in this example. A high consensus threshold is also optimal for a three-network selection Consensus network, and close to optimal for four input networks.

Glossary

Area Under the ROC Curve (AUC)

A global measure of the classifier performance, and is often used in classification problems. In this thesis, AUC is used as a measure of comparison from learnt networks models to the true network. AUC is a value between 0 and 1

Associative Concept Space (ACS)

A biological text mining tool, used in this thesis, that extends the simple co-occurrence technique

Bayesian network (BN)

A method for representing the dependencies among a set of variables. A BN has two components - a qualitative representation of the network: a directed acyclic graph (DAG) and conditional probability distributions that are associated with each variable, which quantify the nature of each dependency

Bayesian Network Meta-Analysis

A post-learning aggregation method for combining BNs, based on combining confidence levels for network edges from the input BNs

Bootstrapping

The process of estimating a quantity by sampling from an approximating distribution. In this thesis, bootstrapping is used to construct BNs from resampled microarray datasets

Concept

In the context of the research in this thesis, a biomedical concept may be single word objects found in the biomedical literature, such as gene or organism names, or may be common multiple-word combinations

Concept profile

For use with the ACS, a concept profile is vector of biological concepts with weights, where the weight describes the strength of the association between the concept and the concept to which the profile belongs

Confidence level

In this thesis, confidence levels are assigned to network edges based on a bootstrapping procedure, where an estimate of the confidence level for each edge is computed by the proportion of networks that contain that edge

Consensus Bayesian Networks (CBNs)

A post-learning aggregation method for combining BNs. A CBN contains consistent network edges across input BNs

Consensus Bootstrapped Bayesian networks (CBBNs)

A post-learning aggregation method, analogous to CBNs, but for combining BN models produced using a bootstrap approach

GenBank

An annotated collection of all publicly available DNA sequences, hosted by the National Center for Biotechnology Information (NCBI)

Gene expression

A process by which genes are copied and translated to proteins. Since proteins are involved in every cellular process, it is gene expression that allows all cellular processes to occur

Gene Ontology (GO)

An initiative with the aim of standardising the representation of gene and gene product attributes across species and databases, by providing a controlled vocabulary of terms for describing gene product characteristics

Gene Regulatory Network (GRN)

A network graph that describes how genes interact in terms of regulation and expression

Intensity log-ratio

A common measure of gene expression

Kyoto Encyclopedia of Genes and Genomes (KEGG)

An online database of biological systems

Literature-based gene concept profiling

A biological text-mining technique, developed by the Biosemantics Association, for calculating the distances between genes and other biological concepts. It makes use of the ACS

Meta-analysis

A set of statistical methods, originating in medical statistics, for combining the results of several studies that address a set of related research hypotheses

Microarray

A high-throughput technology for measuring expression levels for thousands of genes simultaneously

mRNA

The copy (transcript) of a gene produced during the gene expression process. Microarray technology measures expression levels by detecting the abundance of mRNA present in a sample

Network model reliability

The reliability, or ability to generalise, of a network model can be measured based on how well its variable values can be predicted over independent datasets. A network that can predict values with high accuracy on other independent datasets can be said to be more reliable

Normalisation

In the context of microarray data, normalisation is the transformation of expression levels to adjust for systematic variations (arising from variation in the technology rather than true biological variations) so that measurements from two different microarray samples can be directly compared

Post-learning aggregation

When reverse-engineering from multiple data sources, reverse-engineering is performed on each data source separately. The resulting models are then aggregated

Posterior probability distribution

This distribution represents the knowledge or belief about an uncertain quantity, after observation of the data

Pre-learning aggregation

When reverse-engineering from multiple data sources, the data is aggregated prior to the learning/reverse-engineering process

Prediction accuracy

In parameter estimation with BNs, this is the proportion of samples where the predicted discrete states are correct

Prior knowledge

Pre-existing knowledge. In the context of research presented in this thesis, it is information known prior to the data collected from a microarray experiment

Prior probability distribution

This distribution represents the knowledge or belief about an uncertain quantity, prior to observation of the data

Protein-protein interaction data

is generated from technology for detecting physical interactions between proteins. Physically interacting proteins and gene expression are closely linked

PubMed/Medline

A digital archive of biomedical and life sciences journal literature

Reverse-engineering

Discovery of an underlying model or process based on data or observations

Text-mining

A family of techniques for the automatic extraction of information and knowledge from text-based sources. Relevant to the research presented in this thesis, many methods have been developed specifically for the mining of specialist biological literature

Transcription factor

A gene that initiates the regulation of itself or other genes. Strictly speaking, the transcription factor is the protein that is produced when the gene becomes expressed

Transcription Factor Binding Site (TFBS) location data

is generated from technology for discovering the binding sites of transcription factors. Since a gene becomes expressed when a transcription factor binds to a segment of DNA close to it, then if the location of binding sites for a particular transcription factor can be identified, then potential target genes can be found

Weighted Consensus Networks (WCNs)

A post-learning aggregation method, analogous to CBBNs, but which ultilises weights that represent the quality/reliability of each input bootstrapped BN model

References

Aerts, S., Haeussler, M., van Vooren, S., Griffith, O., Hulpiau, P., Jones, S., Montgomery, S., Bergman, C. & The Open Regulatory Annotation Consortium (2008). Text-mining assisted regulatory annotation. *Genome Biology*, **9**, R31. 40

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, **19**, 716–723. 58

Almasri, E., Larsen, P., Chen, G. & Dai, Y. (2008). Incorporating literature knowledge in Bayesian network for inferring gene networks with gene expression data. In *Bioinformatics Research and Applications (LNCS 4983)*, 184–195. 72

Bader, J., Chaudhuri, A., Rothberg, J. & Chant, J. (2004). Gaining confidence in high-throughput protein interaction networks. *Nature Biotechnology*, **22**, 78–85. 35

Bailey, P., Downes, M., Lau, P., Harris, J., Chen, S.L., Hamamori, Y., Sartorelli, V. & Muscat, G.E.O. (1999). The nuclear receptor corepressor N-CoR regulates differentiation: N-CoR directly interacts with MyoD. *Molecular Endocrinology*, **13**, 1155–1168. 93

Bar-Joseph, Z., Gerber, G.K., Lee, T.I., Rinaldi, N.J., Yoo, J.Y., Robert, F., Gordon, D.B., Fraenkel, E., Jaakkola, T.S., Young, R.A. & Gifford, D.K. (2003). Computational discovery of gene modules and regulatory networks. *Nature Biotechnology*, **21**, 1337–1342. 27, 65

Beissbarth, T., Fellenberg, K., Brors, B., Arribas-Prat, R., Boer, J., Hauser, N., Scheideler, M., Hoheisel, J., Schutz, G., Poustka, A. & Vingron, M. (2000). Processing and quality control of DNA array hybridization data. *Bioinformatics*, **16**.

Bellman, R. (1961). *Adaptive Control Processes: A Guided Tour*. Princeton University Press. 31

Bernard, A. & Hartemink, A. (2005). Informative structure priors: Joint learning of dynamic regulatory networks from multiple types of data. In *Proceedings of the Pacific Symposium on Biocomputing*, vol. 10, 459–470. 39, 72, 74

Biosemantics Association (2009). `http://www.biosemantics.org`. 71

Blockeel, H., Dehaspe, L., Demoen, B., Janssens, G., Ramon, J. & Vandecasteele, H. (2002). Improving the efficiency of inductive logic programming through the use of query packs. *Journal of Artificial Intelligence Research*, **16**, 135–166. 166

Castelo, R. & Siebes, A. (2000). Priors on network structures: Biasing the search for Bayesian networks. *International Journal of Approximate Reasoning*, **24**, 39–57. 73, 75

Causton, H.C., Quackenbush, J. & Brazma, A. (2003). *Microarray Gene Expression Data Analysis: A Beginners Guide*. Blackwell Publishing. 4, 9, 16, 24, 25, 26, 31

Chen, L., Wang, S. & Chen, R.S. (2008). A method to detect gene co-expression clusters from multiple microarrays. *Progress in Biochemistry and Biophysics*, **35**, 914–920. 32

Chickering, D. (1995). A transformational characterization of equivalent Bayesian network structures. In *Proceedings of Uncertainty in Artificial Intelligence 11*. 50

Choi, J., Yu, U., Kim, S. & Yoo, O. (2003). Combining multiple microarray studies and modeling interstudy variation. *Bioinformatics*, **19**, 84–90. 32

Choi, J., Yu, U., Yoo, O. & Kim, S. (2005). Differential coexpression analysis using microarray data and its application to human cancer. *Bioinformatics*, **21**, 4348–4355. 32

Chou, R., Campbell, M., Winzeler, E., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T., Gabrielian, A., Landsman, D., Lockhart, D. & Davis, R. (1998). A genome wide transcriptional analysis of the mitotic cell cycle. *Mol. Cell.*, **2**, 65–73. 68

Clark, P. & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, **3**, 261–283. 161

Cleveland, W. (1979). Robust locally weighted regression and smoothing scatter-plots. *Journal of the American Statistical Association*, **74**, 829–836. 20

Conlon, E., Song, J. & Liu, J. (2006). Bayesian models for pooling microarray studies with multiple sources of replications. *BMC Bioinformatics*, **7**. 32, 103

Courcelle, J., Khodursky, A., Peter, B., Brown, P. & Hanawalt, P. (2001). Comparative gene expression profiles following UV exposure in wild-type and SOS-deficient Escherichia coli. *Genetics*, **158**, 41–64.

Datta, N. (2002). Modulation of MDM2/p53 and cyclin-activating kinase during the megakaryocyte differentiation of human erythroleukemia cells. *Experimental Hematology*, **30**, 158–165. 93

De Raedt, L., Blockeel, H., Dehaspe, L. & Laer, W.V. (2001). Three companions for data mining in first order logic. In S. Dzeroski & N. Lavrac, eds., *Relational Data Mining*, 105–139, Springer-Verlag. 166

DeRisi, J., Iyer, V. & Brown, P. (1997). Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, **278**, 680–686. 3, 12

DerSimonian, R. & Laird, N. (1986). Meta-analysis in clinical trials. *Controlled Clinical Trails*, **7**, 177–188. 104

Dzeroski, S. & Lavrac, N. (2001). *Relational Data Mining*. Springer, Berlin. 162

Efron, B. & Tibshirani, R. (1993). *An Introduction to the Bootstrap*. 59

Eisen, M., Spellman, P., Brown, P. & Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *PNAS*, **95**, 14863–8. 25

Elnitski, L., Jin, V., Farnham, P. & Jones, S. (2006). Locating mammalian transcription factor binding sites: A survey of computational and experimental techniques. *Genome Research*, **16**, 1455–1464. 35

Faith, J., Hayete, B., Thaden, J., Mogno, I., Wierzbowski, J., Cottarel, G., Kasif, S., Collins, J. & Gardner, T. (2007). Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS Biology*, **5**. 63, 89

Filkov, V. (2006). Identifying gene regulatory networks from gene expression data. In S. Aluru, ed., *Handbook of Computational Molecular Biology*, Chapman and Hall. 27

Friedman, N., Murphy, K. & Russell, S. (1998). Learning the structure of dynamic probabilistic networks. In *Uncertainty in Artificial Intelligence*, vol. 14. 68

Friedman, N., Goldszmidt, M. & Wyner, A. (1999). Data analysis with Bayesian networks: A bootstrap approach. In *Proceedings of 15th Annual Conference on Uncertainty in Artificial Intelligence*. 59

Friedman, N., Linial, M., Nachman, I. & Pe'er, D. (2000). Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, **7**, 601–620. 4, 29, 51, 53, 65, 66, 67, 101

Gao, F., Foat, B. & Bussemaker, H. (2004). Defining transcriptional networks through integrative modeling of mRNA expression and transcription factor binding data. *BMC Bioinformatics*, **5**. 39

Gasch, A., Spellman, P., Kao, C., Carmel-Harel, O., Eisen, M., Storz, G., Botstein, D. & Brown, P. (2000). Genomic expression program in the response of yeast cells to environmental changes. *Mol. Cell*, **11**, 4241–4257. 26

Ge, H., Liu, Z., Church, G. & Vidal, M. (2001). Correlation between transcriptome and interactome mapping data from S. cerevisiae. *Nature Genetics*, **29**, 482–486. 34

GenExpDB (2008). University of Oklahoma *E. coli* Community's Gene Expression Database. `http://chase.ou.edu/oubcf/`. 30

Golub, T., Slonin, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J. & Cagliguiri, M. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, **286**, 531–537. 25

Gregoire, S., Xiao, L., Nie, J., Zhang, X., Xu, M., Li, J., Wong, J., Seto, E. & Yang, X.J. (2007). Histone deacetylase 3 interacts with and deacetylates myocyte enhancer factor 2. *Molecular and Cellular Biology*, **27**, 1280–1295. 93

Grigull, J., Mnaimneh, S., Pootoolal, J., Robinson, M. & Hughes, T. (2004). Genome-wide analysis of mRNA stability using transcription inhibitors and microarrays reveals post-transcriptional control of ribosome biogenesis factors. *Mol. Cell*, **24**, 5534–47.

Hanley, J. & McNeil, B. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, **143**, 29–36. 64

Hartemink, A., Gifford, D., Jaakkola, T. & Young, R. (2002). Bayesian methods for elucidating genetic regulatory networks. *IEEE Intelligent Systems*, **17**, 37–43. 51, 53, 66

Heyer, L., Kruglyak, S. & Yooseph, S. (1999). Exploring expression data: Identification and analysis of coexpressed genes. *Genome Research*, **9**, 1106–1115. 24

Hoeting, J., Madigan, D., Raftery, A. & Volinsky, C. (1999). Bayesian model averaging: a tutorial. *Statistical Science*, **14**, 382–417. 137

Hu, P., Greenwood, C. & Beyene, J. (2006). Statistical methods for meta-analysis of microarray data: A comparative study. *Information Systems Frontiers*, **8**, 8–20. 32, 103

Imoto, S., Goto, T. & Miyano, S. (2002). Estimation of genetic networks and functional structures between genes by using Bayesian networks and non-parametric regression. In *Pacific Symposium on Biocomputing*, vol. 7, 175–186. 67

Imoto, S., Higuchi, T., Goto, T., Kuhara, S. & Miyano, S. (2003). Combining microarrays and biological knowledge for estimating gene networks via Bayesian networks. In *Proceedings of the IEEE Computer Science Bioinformatics Conference (CSB'03)*, 104–113. 39, 40, 71, 72

Jansen, R., Yu, H., Greenbaum, D., Kluger, Y., Krogan, N., Chung, S., Emili, A., Snyder, M., Greenblatt, J. & Gerstein, M. (2003). A Bayesian networks approach for predicting protein-protein interactions from genomic data. *Science*, **302**, 449–453. 34

Jarvinen, A., Hautaniemi, S., Edgren, H., Auvinen, P., Saarela, J., Kallionoemi, O. & Monni, O. (2004). Are data from different gene expression microarrays comparable? *Genomics*, **83**, 1164–1168. 32

Jelier, R., Jenster, G., Dorssers, L.C.J., van der Eijk, C.C., van Mulligen, E.M., Mons, B. & Kors, J.A. (2005). Co-occurrence based meta-analysis of scientific texts: retrieving biological relationships between genes. *Bioinformatics*, **21**, 2049–2058. 37

Jelier, R., Jenster, G., Dorssers, L., Wouters, B., Hendriksen, P., Mons, B., Delwel, R. & Kors, J. (2007). Text-derived concept profiles support assessment of DNA microarray data for acute myeloid leukemia and for androgen receptor stimulation. *BMC Bioinformatics*, **8**. 38, 70, 77

Jelier, R., Schuemie, M., Roes, P., van Mulligen, E. & Kors, J. (2008). Literature-based concept profiles for gene annotation: The issue of weighting. *International Journal of Medical Informatics*, **77**, 354–362. 78

Jenssen, T.K., Lgreid, A., Komorowski, J. & Hovig, E. (2001). A literature network of human genes for high-throughput analysis of gene expression. *Nature Genetics*, **28**, 21–28. 37

Kanehisa, M. & Goto, S. (2000). KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, **28**, 27–30. 36

Kanehisa, M., Araki, M., Goto, S., Hattori, M., Hirakawa, M., Itoh, M., Katayama, T., Kawashima, S., Okuda, S., Tokimatsu, T. & Yamanishi, Y. (2008). KEGG for linking genomes to life and the environment. *Nucleic Acids Research*, **36**, D480–D484. 36

Kauffman, S. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, **22**, 437–467. 27

Khan, J., Saal, L., Bittner, M., Jiang, Y., Gooden, G., Glatfelter, A. & Meltzer, P. (2002). Expression profiling in cancer using cdna microarrays. *Methods in Molecular Medicine*, **68**, 205–222. x, 15

Khil, P. & Camerini-Otero, R. (2002). Over 1000 genes are involved in the DNA damage response of Escherichia coli. *Molecular Microbiology*, **44**, 89–105.

Kim, S., Imoto, S. & Miyano, S. (2003). Inferring gene networks from time series microarray data using dynamic Bayesian networks. *Briefings in Bioinformatics*, **4**, 228–235. xi, 56, 68

Krallinger, M. & Valencia, A. (2005). Text-mining and information-retrieval services for molecular biology. *Genome Biology*, **6**. 37

Krzanowski, W. & Hand, D. (2009). *ROC Curves for Continuous Data*. Chapman and Hall. 63

Kuo, W., Jenssen, T., A.J.Butte, L.Ohno-Machado & Kohane, I. (2002). Analysis of matched mRNA measurements from two different microarray technologies. *Bioinformatics*, **18**, 405–412. 31

Larsen, P., Almasri, E., Chen, G. & Dai, Y. (2007). A statistical method to incorporate biological knowledge for generating testable novel gene regulatory interactions from microarray experiments. *BMC Bioinformatics*, **8**. 72

Lavrac, N. & Dzeroski, S. (1994). *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York. 162

Lee, H., Hsu, A., Sajdak, J., Qin, J. & Pavlidis, P. (2004). Coexpression analysis of human genes across many microarray data sets. *Genome Research*, **14**, 1085–1094. 32

Lee, M.L.T., Kuo, F.C., Whitmore, G.A. & Sklar, J. (2000). Importance of replication in microarray gene expression studies: Statistical methods and evidence from repetitive cDNA hybridizations. *PNAS*, **97**, 9834–9839. 31

Lee, T., Rinaldi, N., Robert, F., Odom, D., Bar-Joseph, Z., Gerber, G., Hannett, N., Harbison, C., Thompson, C., Simon, I., Zeitlinger, J., Jennings, E., Murray, H., Gordon, D., Ren, B., Wyrick, J., Tagne, J., Volkert, T., Fraenkel, E., Gifford, D. & Young, R. (2002). Transcriptional regulatory networks in Saccharomyces cerevisiae. *Science*, **298**, 799–804. 35, 39

Li, S., Wu, L. & Zhang, Z. (2006). Constructing biological networks through combined literature mining and microarray analysis: a LMMA approach. *Bioinformatics*, **22**, 2143–50. 40

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, 281–297, University of California Press. 25

Mandalb, M., Bandyopadhyayb, D., Goepfert, T.M. & Kumara, R. (1998). Interferon-induces expression of cyclin-dependent kinase-inhibitors p21WAF1 and p27Kip1 that prevent activation of cyclin-dependent kinase by CDK-activating kinase (CAK). *Oncogene*, **16**, 217–225. 93

MAQC consortium (2006). The microarray quality control (MAQC) project shows inter- and intraplatform reproducibility of gene expression measurements. *Nature Biotechnology*, **24**, 1151–1161. 4, 31, 70

Matzkevich, I. & Abramson, B. (1992). The topological fusion of Bayes nets. In *Proceedings of Uncertainty in Artificial Intelligence 8*, 191–198, Morgan Kauffman. 105

McAdams, H. & Arkin, A. (1997). Stochastic mechanisms in gene expression. *PNAS*, **94**, 814–819. 30

McCray, A. & Miller, A. (1998). Making the conceptual connections: the Unified Medical Language System (UMLS) after a decade of research and development. *J. Am. Med. Inf. Ass.*, **4**, 484–500. 77

McDonald, J. (2008). *Handbook of Biological Statistics*. Sparky House Publications, Baltimore, Maryland, USA. 83

McEntyre, J. & Ostell, J. (2002-2005). *The NCBI Handbook*. 36

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill. 43, 50, 159

Murphy, K. (2001a). The Bayes Net Toolbox for Matlab. *Computing Science and Statistics*, **33**. 48, 49, 59

Murphy, K. (2001b). An introduction to graphical models. `http://www.cs.ubc.ca/~murphyk/Papers/intro_gm.pdf`. xi, 43, 44, 45, 46, 48, 51

Murphy, K. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis. 47

Murphy, K. & Mian, S. (1999). Modelling gene expression data using dynamic Bayesian networks. Tech. rep., Computer Science Division, University of California. 68

Nariai, N., Tamada, Y., Imoto, S. & Miyano, S. (2005). Estimating gene regulatory networks and proteinprotein interactions of Saccharomyces cerevisiae from multiple genome-wide data. *Bioinformatics*, **21**, 206–212. 34, 38

Needham, C.J., Bradford, J.R., Bulpitt, A.J. & Westhead, D.R. (2007). A primer on learning in Bayesian networks for computational biology. *PLoS Computational Biology*, **3**, e129. 49

Ng, S., Tan, S. & Sundararajan, V. (2003). On combining multiple microarray studies for improved functional classification by whole-dataset feature selection. *Genome Informatics*, **14**, 44–53. 32

Nikitin, A., Egorov, S., Daraselia, N. & Mazo, I. (2003). Pathway studio: the analysis and navigation of molecular networks. *Bioinformatics*, **19**, 2155–2157. 72

Novichkova, S., Egorov, S. & Daraselia, N. (2003). Medscan, a natural language processing engine for MEDLINE abstracts. *Bioinformatics*, **23**, 1699–1706. 72

Ong, I., Glasner, J. & Page, D. (2002). Modelling regulatory pathways in E. coli from time series expression profiles. *Bioinformatics*, **18**, 241–248. 68

Park, D. & Wang, X. (2006). Toward a general framework for microarray data comparison. In *Proceedings of the 6th IEEE International Conference on Computer and Information Technology (CIT'06)*. 21

Pearl, J. (1991). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kauffman, San Francisco, CA, USA. 4, 43, 45, 57

Pearl, J. (2000). *Causality: Models, Reasoning, and Inference*. Cambridge University Press, New York, NY, USA. 51, 57

Pearl, J. & Verma, T. (1991). A theory of inferred causation. In *Proceedings of Knowledge Representation and Reasoning 2*, 441–452, Morgan Kauffman, New York, USA. 50

Peeling, E. & Tucker, A. (2007). Consensus gene regulatory networks: combining multiple microarray gene expression datasets. In *AIP Conference Proceedings vol. 940, The 3rd International Symposium on Computational Life Sciences (COMPLIFE 2007)*. 8

Peeling, E., Tucker, A. & 't Hoen, P. (2007). Discovery of local regulatory structure from microarray gene expression data using Bayesian networks. In *Proceedings of the Annual Workshop on Intelligent Data Analysis in bioMedicine And Pharmacology (IDAMAP)*. 115

Pe'er, D., Regev, A., Elidan, G. & Friedman, N. (2001). Inferring subnetworks from perturbed expression profiles. In *Proceedings of the Ninth International Conference on Intelligent Systems for Molecular Biology (ISMB 2001)*. 101

Pe'er, D., Tanay, A. & Regev, A. (2006). MinReg: A scalable algorithm for learning parsimonious networks in yeast and mammals. *Journal of Machine Learning Research*, **7**, 167–189. 4, 29, 51, 67

Pennock, D. & Wellman, D. (1999). Graphical representations of consensus belief. In *Proceedings of Uncertainty in Artifical Intelligence 15*, 531–538, Morgan Kauffman. 105

Pournara, I. (2005). *Reconstructing gene networks by passive and active Bayesian learning*. Ph.D. thesis. 58

Pramila, T., Miles, S., GuhaThakurta, D., Jemiolo, D. & Breeden, L. (2002). Conserved homeodomain proteins interact with MADS box protein Mcm1 to restrict ECB-dependent transcription to the M/G1 phase of the cell cycle. *Genes and Development*, **16**, 3034–3045.

Pramila, T., Wu, W., Miles, S., Noble, W.S. & Breeden, L.L. (2006). The forkhead transcription factor Hcm1 regulates chromosome segregation genes and fills the S-phase gap in the transcriptional circuitry of the cell cycle. *Genes and Development*, **20**, 2266–2278. 84

Quackenbush, J. (2001). Computational analysis of microarray data. *Nature Reviews Genetics*, **2**, 418–427. 17

Quackenbush, J. (2002). Microarray data normalization and transformation. *Nature Genetics*, **32**. 20

Quillardet, P., Rouffaud, M. & Bouige, P. (2003). DNA array analysis of gene expression in response to UV irradiation in Escherichia coli. *Research in Microbiology*, **154**, 559–572. 112

Quinlan, J. & Cameron-Jones, R. (1993). Foil: A midterm report. In *P. Brazdil, editor, Proceedings of the 6th European Conference on Machine Learning, volume 667 of Lecture Notes in Artificial Intelligence*, 3–20, Springer-Verlag. 161

Redestig, H., Weicht, D., Selbig, J. & Hannah, M.A. (2007). Transcription factor target prediction using multiple short expression time series from arabidopsis thaliana. *BMC Bioinformatics*, **8**. 33

Rhodes, D., Barrette, T., Rubin, M., Ghosh, D. & Chinnaiyan, A. (2002). Meta-analysis of microarrays: Inter-study validation of gene expression profiles reveals pathway dysregulation in prostate cancer. *Cancer Research*, **62**, 4427–4433. 32

Saito, R., Suzuki, H. & Hayashizaki, Y. (2003). Construction of reliable protein-protein interaction networks with a new interaction generality measure. *Bioinformatics*, **19**, 756–763. 35

Salgado, H., Gama-Castro, S., Peralta-Gil, M., Diaz-Peredo, E., Sanchez-Solano, F., Santos-Zavaleta, A., Martinez-Flores, I., Jimenez-Jacinto, V., Bonavides-Martinez, C., Segura-Salazar, J., Martinez-Antonio, A. & Collado-Vides, J. (2006). Regulondb (version

5.0): Escherichia coli K-12 transcriptional regulatory network, operon organization, and growth conditions. *Nucleic Acids Research*, **34**. 62, 88, 107

Sangurdekar, D., Srienc, F. & Khodursky, A. (2006). A classification based framework for quantitative description of large-scale microarray data. *Genome Biology*, **7**, R32. 88

Schapire, R.E. (2003). The boosting approach to machine learning: An overview. In *D. D. Denison, M. H. Hansen, C. Holmes, B. Mallick, B. Yu, (Eds), Nonlinear Estimation and Classification*, Springer. 137

Schlitt, T. & Brazma, A. (2007). Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, **8**. 27

Schuemie, M., Chichester, C., Lisacek, F., Coute, Y., Roes, P., Sanchez, J., Kors, J. & Mons, B. (2007a). Assignment of protein function and discovery of novel nucleolar proteins based on automatic analysis of MEDLINE. *Proteomics*, **7**, 921–931. 38, 70, 77

Schuemie, M., Jelier, R. & Kors, J. (2007b). Peregrine: Lightweight gene name normalization by dictionary lookup. In *Proceedings of the Biocreative 2 workshop*. 77

Schwartz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, **6**, 461–464. 57

Segal, E., Shapira, M., Pe'er, D., Botstein, D., Koller, D. & Friedman, N. (2003a). Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nature Genetics*, **34**, 168–176. 4, 27, 29, 39, 65, 67

Segal, E., Wang, H. & Koller, D. (2003b). Discovering molecular pathways from protein interaction and gene expression data. *Bioinformatics*, **19**, 264–272. 39

Segal, E., Yelensky, R. & Koller, D. (2003c). Genome-wide discovery of transcriptional modules from dna sequence and gene expression. *Bioinformatics*, **19**, 273–282. 39

Segal, E., Pe'er, D., Regev, A., Koller, D. & Friedman, N. (2005). Learning module networks. *Journal of Machine Learning Research*, **6**, 557–588. 67

SGD-project (2008). Saccharomyces genome database. `http://www.yeastgenome.org/`. 30

Shachter, R. (1986). Evaluating influence diagrams. *Operations Research*, **34**, 871–882. 106

Shi, Y., Mitchell, T. & Bar-Joseph, Z. (2007). Inferring pairwise regulatory relationships from multiple time series datasets. *Bioinformatics*, **23**, 755–763. 33

Shmulevich, I., Dougherty, E., Kim, S. & Zhang, W. (2002). Probabilistic boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, **18**, 261–274. 29

Smyth, G. & Speed, T. (2003). Normalization of cDNA microarray data. *Methods*, **31**, 265–273. 18

Somorjai, R., Dolenko, B. & Baumgartner, R. (2003). Class prediction and discovery using gene microarray and proteomics mass spectroscopy data: curses, caveats, cautions. *Bioinformatics*, **19**, 1484–1491. 4

Spellman, P., Sherlock, G., Zhang, M., Iyer, V., Anders, K., Eisen, M., Brown, P., Botstein, D. & Futcher, B. (1998). Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Mol. Cell*, **9**, 3273–3297. 26, 66, 68, 83, 198

Spirtes, P., Glymour, C. & Scheines, R. (2000). *Causation, Prediction, and Search (2nd ed.)*. MIT Press, New York, N.Y, USA. 57

Stapley, B. & Benoit, G. (2000). Biobibliometrics: information retrieval and visualization from co-occurrences of gene names in medline abstracts. In *Proceedings of the Pacific Symposium on Biocomputing*. 37

Steele, E. & Tucker, A. (2008). Consensus and meta-analysis regulatory networks for combining multiple microarray gene expression datasets. *Journal of Biomedical Informatics*, **41**, 914–926. 8

Sterrenburg, E., van der Wees, C., White, S., Turk, R., de Menezes, R., van Ommen, G., den Dunnen, J. & t Hoen, P. (2006). Gene expression profiling highlights defective myogenesis in DMD patients and a possible role for bone morphogenetic protein 4. *Neurobiology of Disease*, **23**, 228–236. 92

Stoica, P. (2004). On information criteria and the generalized likelihood ratio test of model order selection. *IEEE Signal Processing Letters*, **11**. 58

Sutton, A., Abrams, K., Jones, D., Sheldon, T. & Song, F. (2000). *Methods for Meta-Analysis in Medical Research*. Wiley, Chichester, UK. 32, 103, 105

Suwannaroj, S. & Niranjan, M. (2008). Enhancing automatic construction of gene subnetworks by integrating multiple sources of information. *Journal of Signal Processing Systems*, **50**, 331–340. 40

Tan, P.K., Downey, T.J., Jr, E.L.S., Xu, P., Fu, D., Dimitrov, D.S., Lempicki, R.A., Raaka, .M. & Cam, M.C. (2003). Evaluation of gene expression measurements from commercial microarray platforms. *Nucleic Acids Research*, **31**, 5676–5684. 4, 31, 70

Teixeira, M., Monteiro, P., Jain, P., Tenreiro, S., Fernandes, A., Mira, N., Alenquer, M., Freitas, A., Oliveira, A. & S-Correia, I. (2006). The YEASTRACT database: a tool for the analysis of transcription regulatory associations in Saccharomyces cerevisiae. *Nucleic Acids Research*, **34**, D446–D451. 62, 83, 107, 139, 198

The Gene Ontology Consortium (2000). Gene ontology: tool for the unification of biology. *Nature Genetics*, **25**, 25–29. 36

Twyman, R. (2003). Gene expression. *Published online by the Wellcome Trust*. 2, 10, 12

Van den Bulcke, T., Leemput, K.V., Naudts, B., van Remortel, P., Ma, H., Verschoren, A., Moor, B.D. & Marchal, K. (2006). Syntren: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC Bioinformatics*, **7**. 167

van der Eijk, C.C., van Mulligen, E.M., Kors, J.A., Mons, B. & van den Berg, J. (2004). Constructing an associative concept space for literature-based discovery. *Journal of the American Society for Information Science and Technology*, **55**, 436–444. 37

Vohradsky, J. (2001). Neural model of the genetic network. *The Journal of Biological Chemistry*, **276**, 36168–36173. 2

Wang, Y., Joshi, T., Zhang, X., Xu, D. & Chen, L. (2006). Inferring gene regulatory networks from multiple microarray datasets. *Bioinformatics*, **22**, 2413–2420. 33, 106, 116, 119

Werhli, A. & Husmeier, D. (2007). Reconstructing gene regulatory networks with Bayesian networks by combining expression data with multiple sources of prior knowledge. *Statistical applications in genetics and molecular biology*, **6**. 40, 72

Werhli, A., Grzegorczyk, M. & Husmeier, D. (2006). Comparative evaluation of reverse engineering gene regulatory networks with relevance networks, graphical gaussian models and Bayesian networks. *Bioinformatics*, **22**, 2523–2531. 58

Wernisch, L. (2002). Can replication save noisy microarray data? *Comparative and Functional Genomics*, **3**, 372–374. 31

Wickham, H. (2004). Normalisation and visualisation of microarrays. Masters thesis, University of Auckland, New Zealand. 20

Woo, Y., Affourtit, J., Daigle, S., Viale, A., Johnson, K., Naggert, J. & Churchill, G. (2004). A comparison of cDNA, oligonucleotide, and affymetrix genechip gene expression microarray platforms. *Journal of Biomolecular Techniques*, **15**, 276–284. 31

Workman, C., Jensen, L., Jarmer, H., Berka, R., Gautier, L., Nielser, H., Saxild, H., Nielsen, C., Brunak, S. & Knudsen, S. (2002). A new non-linear normalization method for reducing variability in dna microarray experiments. *Genome Biology*, **3**. 20

Xu, X., Wang, L. & Ding, D. (2004). Learning module networks from genome-wide location and expression data. *FEBS Letters*, **578**, 297–304. 39

Yan, X., Mehan, M., Huang, Y., Waterman, M., Yu, P. & Zhou, X. (2007). A graph-based approach to systematically reconstruct human transcriptional regulatory modules. *Bioinformatics*, **23**, i577–i586. 32

Yang, Y., Buckley, M. & Speed, T. (2001). Analysis of cDNA microarray images. *Briefings in Bioinformatics*, **2**, 341–349. 16

Yauk, C., Nerndt, M., Williams, A. & Douglas, G. (2004). Comprehensive comparison of six microarray technologies. *Nucleic Acids Research*, **32**. 32

Yu, H., Luscombe, N., Qian, J. & Gerstein, M. (2003). Genomic analysis of gene expression relationships in transcriptional regulatory networks. *Trends Genetic*, **19**, 422–427. 68

Yuen, T., Wurmbach, E., Pfeffer, R., Ebersole, B. & Sealfon, S. (2002). Accuracy and calibration of commercial oligonucleotide and custom cDNA microarrays. *Nucleic Acids Research*, **30**. 31

Zhu, G., Spellman, P., Volpe, T., Brown, P., Botstein, D., Davis, T. & Futcher, B. (2000). Two yeast forkhead genes regulate the cell cycle and pseudohyphal growth. *Nature*, **406**, 90–94.

Zou, M. & Conzen, S. (2005). A new dynamic Bayesian network approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics*, **21**, 71–79. 26, 68