



AALBORG UNIVERSITY
DENMARK

Aalborg Universitet

Learning Bayesian Networks with Mixed Variables

Bøttcher, Susanne Gammelgaard

Publication date:
2005

Document Version
Publisher's PDF, also known as Version of record

[Link to publication from Aalborg University](#)

Citation for published version (APA):
Bøttcher, S. G. (2005). Learning Bayesian Networks with Mixed Variables. (Ph.D. Report Series; No. 11).

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- ? Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- ? You may not further distribute the material or use it for any profit-making activity or commercial gain
- ? You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.

**Learning Bayesian Networks
with Mixed Variables**

Susanne Gammelgaard Bøttcher

Ph.D. Thesis

2004

AALBORG UNIVERSITY
Department of Mathematical Sciences

Preface

This thesis is the result of my Ph.D. study at the Department of Mathematical Sciences, Aalborg University, Denmark. The work has mainly been founded by Aalborg University, but also in parts by the ESPRIT project P29105 (BaKE) and by Novo Nordisk A/S.

The thesis concerns learning Bayesian networks with both discrete and continuous variables and is based on the following four papers:

- I. Learning Conditional Gaussian Networks.
- II. `deal`: A Package for Learning Bayesian Networks.
- III. Prediction of the Insulin Sensitivity Index using Bayesian Networks.
- IV. Learning Dynamic Bayesian Networks with Mixed Variables.

Many of the results in Paper I are published in Böttcher (2001). Paper II is published in Böttcher and Dethlefsen (2003a). The developed software package, `deal`, is written in R (R Development Core Team 2003) and can be downloaded from the Comprehensive R Archive Network (CRAN) <http://cran.r-project.org/>. Paper II and Paper III are written together with Claus Dethlefsen, Aalborg University.

The individual papers are self-contained with an individual bibliography and figure, table and equation numbering. Parts and bits therefore appear in more than one paper. A basic understanding of the results in Paper 1 is though an advantage in reading the other papers. Those who are not familiar with Bayesian networks in general, might consult introductory books such as Jensen (1996) and Cowell, Dawid, Lauritzen and Spiegelhalter (1999).

I am much indebted to my supervisor Steffen L. Lauritzen for inspiring discussions and for many valuable comments. Also thanks to Søren Lundbye-Christensen who has always taken the time to help me, when I had some questions.

I owe a special thanks to Claus Dethlefsen for an inspiring collaboration and for sharing his expertise within statistical programming with me.

Part of the work is based on data provided by Torben Hansen, Novo Nordisk A/S. Thanks for letting us use these data.

From September 1999 to November 1999 I visited the Fields Institute in Toronto, Canada, as a participant in the research program "Causal Interpretation and Identification of Conditional Independence Structures". I wish to thank all the people there for a memorable stay.

Acknowledgements also go to my colleagues at the Department of Mathematical Sciences for providing a nice working environment and especially to E. Susanne Christensen, who has been my mentor and one of my best friends during the years I have worked at the department.

Special thanks go to my colleagues and very dear friends Malene, Kim and Claus. We always have a lot of fun together and we are also there for each other during the hard times.

Both my parents and my in-laws have been very supportive during this study and that has meant a lot to me, so a word of thanks also goes to them.

All my thanks go to my two sons, Johannes and Magnus, for reminding me of the things that are the most important in life.

Finally, I would like to thank my husband Peter for his invaluable support and endless encouragement during my Ph.D. study. Also thanks for caring for Johannes and Magnus during my sometimes long work hours. It was the thought of having more time to spend with the three of you that finally made me finish this thesis.

Aalborg, Denmark, March 2004

Susanne Gammelgaard Bøttcher

Summary

The main topic of this thesis is learning Bayesian networks with discrete and continuous variables.

A Bayesian network is a directed acyclic graph that encodes the joint probability distribution for a set of random variables. The nodes in the graph represent the random variables and missing arrows between the nodes, specify properties of conditional independence between the variables. It consists of two parts, a knowledge base and an inference engine for handling this knowledge. This thesis relies on already developed methods for inference and concentrate on constructing the knowledge base.

When constructing the knowledge base, there are two things to consider, namely learning the graphical structure and learning the parameters in the probability distributions. In this thesis, the focus is on learning Bayesian networks, where the joint probability distribution is conditional Gaussian. To learn the parameters, conjugate Bayesian analysis is used and parameter independence and complete data are assumed. To learn the graphical structure, network scores for the different structures under evaluation, are calculated and used to discriminate between the structures. To calculate these scores, the prior distribution for the parameters for each network under evaluation, must be specified. An automated procedure for doing this is developed. With this procedure, the parameter priors for all possible networks are deduced from marginal priors calculated from an imaginary database.

Bayes factors to be used when searching for structures with high network score, are also studied. To reduce the search complexity, classes of models are identified for which the Bayes factor for testing an arrow between the same two variables, is the same.

To be able to use the methods in practice, a software package called `deal`, written in `R`, is developed. The package includes procedures for defining priors, estimating parameters, calculating network scores, performing heuristic search as well as simulating data sets with a given dependency structure.

To illustrate the Bayesian learning procedure, a dataset from a study concerning the insulin sensitivity index, is analyzed. The insulin sensitivity index is an index that can be used in assessing the risk of developing type 2 diabetes. Interest is in developing a method to determine the index from measurements of glucose and insulin concentrations in plasma sampled subsequently after an glucose intake. As the dependency relations between the glucose and insulin measurements are complicated, it is proposed to use Bayesian networks. The conclusion is that the insulin sensitivity index for a non-diabetic glucose tolerant subject can be predicted from the glucose and insulin measurements, the gender and the body mass index, using Bayesian networks.

Finally, dynamic Bayesian networks with mixed variables are studied. A dynamic Bayesian network is just a simple extension of an ordinary Bayesian network and is applied in the modeling of time series. It is shown how the methods developed for learning Bayesian networks with mixed variables, can be extended to use for learning dynamic Bayesian networks with mixed variables. As the Markov order of a times series is not always known, it is also shown how to learn this order.

Summary in Danish – sammendrag

Denne afhandling omhandler konstruktion (indlæring) af bayesianske netværk med diskrete og kontinuerte variable.

Et bayesiansk netværk er en orienteret graf uden kredse, der beskriver den simultane sandsynlighedsfordeling for en mængde af stokastiske variable. Knuderne i grafen repræsenterer de stokastiske variable og manglende pile imellem knuderne repræsenterer betingede uafhængighedsantagelser. Et bayesiansk netværk består af to dele, en vidensbase og en inferensmaskine til at håndtere denne viden. Denne afhandling bruger allerede udviklede metoder til inferens og fokuserer på at konstruere vidensbasen.

Konstruktionen af vidensbasen kan deles op i to dele, nemlig selektion af den grafiske struktur og estimation af parametrene i sandsynlighedsfordelingerne. I denne afhandling fokuseres der på bayesianske netværk, hvor den simultane sandsynlighedsfordeling er betinget gaussisk. Til parameter estimation bruges konjugeret bayesiansk analyse og det antages, at parametrene er uafhængige og at data er fuldstændige.

Til selektion af den grafiske struktur beregnes et mål for hvor godt en given struktur beskriver data, i afhandlingen kaldet for en netværksscore. Netværksscoren beregnes for alle de strukturer, der tages i betragtning og bruges således til at diskriminere imellem de forskellige strukturer.

For at kunne beregne disse netværksscorer skal man kende apriori fordelingen for parametrene i alle de betragtede netværk. En automatisk procedure til at deducere disse apriori fordelinger fra marginale apriori fordelinger, beregnet fra en imaginær database, udvikles. Desuden studeres bayes faktorer, da disse bruges i forskellige søge strategier til søgning efter netværk med høj netværksscore. For at reducere søge kompleksiteten identificeres klasser af modeller, hvor bayes

faktoren til at teste en pil mellem de samme to variable, er den samme.

For at kunne bruge de udviklede metoder i praksis, er et software program, kaldet `deal`, udviklet. Pakken, som er skrevet til R, inkluderer procedurer til at definere apriori fordelinger, estimere parametre, beregne netværksscorer, søge efter netværk med høj netværksscore og simulerer datasæt med en given afhængighedsstruktur.

Til illustration af den bayesianske indlæringsprocedure analyseres et datasæt fra et studie, der omhandler insulin sensitivitets indekset. Insulin sensitivitets indekset er et indeks, der kan bruges til at vurdere risikoen for at udvikle type 2 diabetes. Formålet med studiet er at udvikle en metode, der kan bestemme dette indeks ud fra gentagne målinger af glukose og insulin koncentrationerne i plasma efter et glukose indtag. Da afhængighedsstrukturen mellem glukose og insulin målingerne er kompleks, bruges bayesianske netværk til at repræsentere disse afhængigheder. Konklusionen er at insulin sensitivitets indekset for ikke-diabetiske glukose tolerante individer, kan predikteres fra glukose og insulin målingerne, kønnet og body mass indekset, ved at bruge bayesianske netværk.

Til sidst i afhandlingen studeres dynamiske bayesianske netværk med blandede variable. Et dynamisk bayesiansk netværk er en simpel udvidelse af de sædvanlige bayesianske netværk og anvendes til modellering af tidsrække data. Det vises hvordan de metoder, der er udviklet til indlæring af de sædvanlige bayesianske netværk med blandede variable, kan udvides, så de kan anvendes til indlæring af dynamiske bayesianske netværk med blandede variable. Da Markov ordenen af en tidsrække ikke altid er kendt, vises det også, hvordan man kan indlære denne orden.

Contents

Preface	iii
Summary	v
Summary in Danish – sammendrag	vii
Introduction	1
Paper I. Learning Conditional Gaussian Networks	11
1 Introduction	13
2 Bayesian Networks	14
3 Bayesian Networks for Mixed Variables	15
4 Learning the Parameters in a CG Network	17
5 Learning the Structure of a CG Network	23
6 The Master Prior Procedure	26
7 Local Masters for Mixed Networks	31
8 Model Search	34
9 Example	41
Paper II. deal: A Package for Learning Bayesian Networks	57
1 Introduction	59
2 Bayesian Networks	61
3 Data Structure	62
4 Specification of a Bayesian Network	63
5 Parameter Learning	67
6 Learning the Structure	73
7 Hugin Interface	78
8 Example	79
9 Discussion and Future Work	82
10 Manual Pages for deal	84

Paper III. Prediction of the Insulin Sensitivity Index using Bayesian		
Networks		107
1	Introduction	109
2	Data	110
3	Bayesian Networks	111
4	Inference	113
5	Results	115
6	Discussion	124
 Paper IV. Learning Dynamic Bayesian Networks with Mixed Variables		127
1	Introduction	129
2	Dynamic Bayesian Networks	130
3	Dynamic Bayesian Networks for Mixed Variables	134
4	Examples of DBNs	137
5	Learning DBNs with Mixed Variables	140
6	Specifying Prior Distributions	149
7	Example	152

Introduction

The main focus of this Ph.D. thesis is to develop statistical methods for learning Bayesian networks with mixed variables. To be able to use these methods in practice, the software package `deal` is developed. Besides, the methods are extended to use for dynamic Bayesian networks.

Background

Bayesian networks was developed in the late 80's by Pearl (1988) and Lauritzen and Spiegelhalter (1988). For terminology and theoretical aspects, see Lauritzen (1996), Jensen (1996) and Cowell et al. (1999) among others.

A Bayesian network is a directed acyclic graph that encodes the joint probability distribution for a set of random variables. The nodes in the graph represent the random variables and missing arrows between the nodes, specify properties of conditional independence between the variables.

A Bayesian network consists of two parts, a knowledge base and an inference engine for handling this knowledge. Generally, inference is computationally heavy as it involves calculating huge joint distributions, especially if there are many variables in the network. Therefore efficient methods of implementing Bayes' theorem are being used. These implementations uses the fact that the the joint probability distribution of all the variables in a network, factorizes according to the structure of the graph. The distributions of interest can then be found by a series of local computations, involving only some of the variables at a time, see *e.g.* Cowell et al. (1999) for a thorough treatment of these methods. The methods are implemented in *e.g.* Hugin (<http://www.hugin.com>). Bayesian networks are therefore suitable for problems where the variables exhibit a complicated dependency structure. See Lauritzen (2003) for a recent overview over different applications.

In this thesis, we will rely on already developed methods for inference and concentrate on constructing the knowledge base. The work is documented through four papers, which will be described in the following.

Paper I. Learning Conditional Gaussian Networks

When constructing the knowledge base there are two things to consider, namely specifying the graphical structure and specifying the probability distributions. Paper I addresses these issues for Bayesian networks with mixed variables.

In this paper, the focus is on learning Bayesian networks, where the joint probability distribution is conditional Gaussian. For an introductory text on learning Bayesian networks, see Heckerman (1999). To learn the parameters in the local probability distributions, conjugate Bayesian analysis is used. As conjugate local priors, the Dirichlet distribution is applied for discrete variables and the Gaussian-inverse gamma distribution is applied for continuous variables, given a configuration of the discrete parents. We assume parameter independence and complete data. To learn the graphical structure, network scores for the different structures under evaluation, are calculated and these scores are used to discriminate between the structures. To calculate these scores, the prior distribution for the parameters, for each network under evaluation, must be specified. In Heckerman, Geiger and Chickering (1995) and Geiger and Heckerman (1994) an automated procedure for doing this in respectively the purely discrete case and the purely continuous case, is developed. Their work is based on principles of likelihood equivalence, parameter modularity, and parameter independence. It leads to a method where the parameter priors for all possible networks, are deduced from one joint prior distribution, in this thesis called a master prior distribution.

In Paper I, we build on their results and develop a method, which can be used on networks with mixed variables. If used on networks with only discrete variables or only continuous variables, it coincides with the methods developed in respectively Heckerman et al. (1995) and Geiger and Heckerman (1994).

If the number of random variables in a network is large, it is computationally infeasible to calculate the network score for all the possible structures. Therefore different methods for searching for structures with high network score, are being used, see *e.g.* Cooper and Herskovits (1992). Many of these methods use Bayes factors as a way of comparing the network scores for two different models. We therefore study Bayes factors for mixed networks. To reduce the search complexity, classes of models are identified for which the Bayes factor

for testing an arrow between the same two variables, is the same.

Finally, an analysis of a simple example illustrates the developed methods and is also used for showing how the strength of the prior parameter distribution affects the result of the analysis.

Paper II. `deal`: A Package for Learning Bayesian Networks

To be able to use the methods presented in Paper I in practice, we have developed a software package called `deal`, written in R (R Development Core Team 2003).

In particular, the package includes procedures for defining priors, estimating parameters, calculating network scores, performing heuristic search as well as simulating data sets with a given dependency structure. The package can be downloaded from the Comprehensive R Archive Network (CRAN) <http://cran.R-project.org/> and may be used under the terms of the GNU General Public License Version 2.

The package supports transfer of the learned network to Hugin (<http://www.hugin.com>). The Hugin graphical user interface (GUI) can then be used for further inference in this network. Besides, `deal` adds functionality to R, so that Bayesian networks can be used in conjunction with other statistical methods available in R for analyzing data. In particular, `deal` is part of the `gR` project, which is a newly initiated workgroup with the aim of developing procedures in R for supporting data analysis with graphical models, see <http://www.r-project.org/gR>.

Paper III. Prediction of the Insulin Sensitivity Index using Bayesian Networks

To illustrate the Bayesian learning procedure, we have in Paper III analyzed a dataset collected by Torben Hansen, Novo Nordisk A/S.

The insulin sensitivity index, S_I , is an index that can be used in assessing the risk of developing type 2 diabetes. The index is determined from an intravenous glucose tolerance test (IVGTT), where glucose and insulin concentrations in plasma are subsequently sampled after an intravenous glucose injection. However, an IVGTT is time consuming and expensive and therefore not suitable for large scale epidemiological studies. Therefore interest is in developing a method to assess S_I from measurements from an oral glucose tolerance test (OGTT). In an OGTT, glucose and insulin concentrations in plasma are, after an glucose

intake, sampled at a few time points.

In the present study, 187 non-diabetic glucose tolerant subjects underwent both an OGTT and an IVGTT. From the IVGTT, the S_I values are determined using Bergmans minimal model (Bergman, Ider, Bowden and Cobelli 1979) as done in Pacini and Bergman (1986). The aim of our analysis is to determine the S_I values from the measurements from the OGTT and investigate whether the S_I values from the oral study, are correlated to the S_I values determined from the intravenous study.

As the dependency relations between the glucose and insulin measurements are complicated, we propose to use Bayesian networks. We learn various Bayesian networks, relating measurements from the OGTT to the S_I values determined from the IVGTT. We conclude that the S_I values from the oral study, determined using Bayesian networks, are highly correlated to the S_I values from the intravenous study, determined using Bergmans minimal model.

Paper IV. Learning Dynamic Bayesian networks with Mixed Variables

A dynamic Bayesian network is an extension of an ordinary Bayesian network and is applied in the modeling of time series, see Dean and Kanazawa (1989). In Murphy (2002) a thorough treatment of these models for first order Markov time series, is presented and in Friedman, Murphy and Russell (1998), learning these networks in the case with only discrete variables, is described. In Paper IV, methods for learning dynamic Bayesian networks with mixed variables, are developed. These methods are just simple extensions of the methods described in Paper I for learning Bayesian networks with mixed variables. It is therefore also straight forward to use `deal` to learn dynamic Bayesian networks.

Contrary to previous work, we consider time series with Markov order higher than one and show how the Markov order can be learned.

To illustrate the developed methods, the Wölfer's sunspot numbers are analyzed.

References

- Anderson, T. W. (1971). *The Statistical Analysis of Time Series*, John Wiley and Sons, New York.
- Badsberg, J. H. (1995). *An Environment for Graphical Models*, PhD thesis, Aalborg University.
- Bergman, R. N., Ider, Y. Z., Bowden, C. R. and Cobelli, C. (1979). Quantitative estimation of insulin sensitivity, *American Journal of Physiology* **236**: E667–E677.
- Bernardo, J. M. and Smith, A. F. M. (1994). *Bayesian Theory*, John Wiley & Sons, Chichester.
- Bøttcher, S. G. (2001). Learning Bayesian Networks with Mixed Variables, *Artificial Intelligence and Statistics 2001*, Morgan Kaufmann, San Francisco, CA, USA, pp. 149–156.
- Bøttcher, S. G. (2003). Learning Bayesian networks with mixed variables, Aalborg University, <http://www.math.auc.dk/~alma>.
- Bøttcher, S. G. and Dethlefsen, C. (2003a). deal: A Package for Learning Bayesian Networks, *Journal of Statistical Software* **8**(20): 1–40.
- Bøttcher, S. G. and Dethlefsen, C. (2003b). Learning Bayesian networks with R, in K. Hornik, F. Leisch and A. Zeileis (eds), *Proceedings of the 3rd international workshop on distributed statistical computing*. ISSN 1609-395X.
- Bøttcher, S. G., Milsgaard, M. B. and Mortensen, R. S. (1995). *Monitoring by using dynamic linear models - illustrated by tumour markers for cancer*, Master's thesis, Aalborg University.
- Chickering, D. M. (1995). A transformational characterization of equivalent Bayesian-network structures, *Proceedings of Eleventh Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, CA, USA, pp. 87–98.
- Chickering, D. M. (1996). Learning Bayesian networks is NP-Complete, in D. Fisher and H. J. Lenz (eds), *Learning from Data: Artificial Intelligence and Statistics V*, Springer-Verlag, New York, pp. 121–130.

- Chickering, D. M. (2002). Optimal structure identification with greedy search, *Journal of Machine Learning Research* **3**: 507–554.
- Cooper, G. and Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data, *Machine Learning* **9**: 309–347.
- Cowell, R. G., Dawid, A. P., Lauritzen, S. L. and Spiegelhalter, D. J. (1999). *Probabilistic Networks and Expert Systems*, Springer-Verlag, Berlin-Heidelberg-New York.
- Dawid, A. P. (1982). The Well-Calibrated Bayesian, *Journal of the American Statistical Association* **77**(379): 605–610.
- Dawid, A. P. and Lauritzen, S. L. (1993). Hyper Markov laws in the statistical analysis of decomposable graphical models, *The Annals of Statistics* **21**(3): 1272–1317.
- Dean, T. and Kanazawa, K. (1989). A model for reasoning about persistence and causation, *Computational Intelligence* **5**: 142–150.
- DeGroot, M. H. (1970). *Optimal Statistical Decisions*, McGraw-Hill, New York.
- Drivsholm, T., Hansen, T., Urhammer, S. A., Palacios, R. T., Vølund, A., Borch-Johnsen, K. and Pedersen, O. B. (2003). Assessment of insulin sensitivity index and acute insulin reponse from an oral glucose tolerance test in subjects with normal glucose tolerance, Novo Nordisk A/S.
- Edwards, D. (1995). *Introduction to Graphical Modelling*, Springer-Verlag, New York.
- Friedman, N., Murphy, K. P. and Russell, S. (1998). Learning the Structure of Dynamic Probabilistic Networks, *Proceedings of Fourteenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, CA, USA, pp. 139–147.
- Frydenberg, M. (1990). Marginalization and collapsibility in graphical interaction models, *Annals of Statistics* **18**: 790–805.
- Furnival, G. M. and Wilson, R. W. (1974). Regression by Leaps and Bounds, *Technometrics* **16**(4): 499–511.
- Geiger, D. and Heckerman, D. (1994). Learning Gaussian Networks, *Proceedings of Tenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, CA, USA, pp. 235–243.

- Harrison, P. J. and Stevens, C. F. (1976). Bayesian forecasting, *Journal of Royal Statistics* **38**: 205–247.
- Haughton, D. M. A. (1988). On The Choice of a Model to fit Data From an Exponential Family, *The Annals of Statistics* **16**(1): 342–355.
- Heckerman, D. (1999). A Tutorial on Learning with Bayesian Networks, in M. Jordan (ed.), *Learning in Graphical Models*, MIT Press, Cambridge, MA.
- Heckerman, D., Geiger, D. and Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data, *Machine Learning* **20**: 197–243.
- Ihaka, R. and Gentleman, R. (1996). R: A language for data analysis and graphics, *Journal of Computational and Graphical Statistics* **5**: 299–314.
- Jensen, F. V. (1996). *An Introduction to Bayesian Networks*, UCL Press, London.
- Lauritzen, S. L. (1992). Propagation of probabilities, means and variances in mixed graphical association models, *Journal of the American Statistical Association* **87**(420): 1098–1108.
- Lauritzen, S. L. (1996). *Graphical Models*, Clarendon press, Oxford, New York.
- Lauritzen, S. L. (2003). Some modern applications of graphical models, in P. J. Green, N. L. Hjort and S. Richardson (eds), *Highly Structured Stochastic Systems*, Oxford University Press, Oxford, pp. 13–32.
- Lauritzen, S. L. and Jensen, F. (2001). Stable local computation with conditional Gaussian distributions, *Statistics and Computing* **11**: 191–203.
- Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems (with discussion), *J. Royal Statist. Soc. Series B*.
- Martin, B. C., Warram, J. H., Krolewski, A. S., Bergman, R. N., Soeldner, J. S. and Kahn, C. R. (1992). Role of glucose and insulin resistance in development of type 2 diabetes mellitus: results of a 25-year follow-up study, *The Lancet* **340**: 925–929.
- Morrison, D. F. (1976). *Multivariate Statistical Methods*, McGraw-Hill, USA.

- Murphy, K. P. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*, PhD thesis, University of California, Berkeley.
- Pacini, G. and Bergman, R. N. (1986). MINMOD: a computer program to calculate insulin sensitivity and pancreatic responsiveness from the frequently sampled intravenous glucose tolerance test, *Computer Methods and Programs in Biomedicine* **23**: 113–122.
- Pearl, J. (1988). *Probabilistic Reasoning in Intelligence Systems: Networks of Plausible Inference*, Morgan Kaufmann, Los Altos, CA.
- R Development Core Team (2003). *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-00-3.
- Robinson, R. W. (1977). Counting unlabeled acyclic digraphs, *Lecture Notes in Mathematics, 622: Combinatorial Mathematics V* pp. 239–273.
- Schaerf, M. C. (1964). Estimation of the covariance and autoregressive structure of a stationary time series, *Technical report*, Department of Statistics, Stanford University.
- Seber, G. A. F. (1984). *Multivariate Observations*, John Wiley and Sons, New York.
- Shachter, R. D. and Kenley, C. R. (1989). Gaussian influence diagrams, *Management Science* **35**: 527–550.
- Spiegelhalter, D. J. and Lauritzen, S. L. (1990). Sequential updating of conditional probabilities on directed graphical structures, *Networks* **20**: 579–605.
- Steck, H. and Jaakkola, T. S. (2002). On the Dirichlet Prior and Bayesian Regularization, *Conference on Advances in Neural Information Processing Systems*, Vol. 15, MIT Press, Cambridge, USA.
- Tong, H. (1996). *Non-Linear Time Series*, Clarendon Press, Oxford.
- Venables, W. N. and Ripley, B. D. (1997). *Modern Applied Statistics with S-PLUS*, second edn, Springer-Verlag, New York.
- West, M. and Harrison, J. (1989). *Bayesian Forecasting and Dynamic Models*, Springer-Verlag, New York.

Yule, G. U. (1927). On a method for investigating periodicities in disturbed series with special reference to Wölfer's sunspot numbers, *Philosophical Transactions of the Royal Society, Series A* **226**: 267–298.

Paper I

Learning Conditional Gaussian Networks

Susanne G. Bøttcher

Many of the results in this paper have been published in

*Proceedings of the Eight International Workshop on
Artificial Intelligence and Statistics (2001),
pp. 149-156.*

Learning Conditional Gaussian Networks

Susanne G. Bøttcher

Aalborg University, Denmark

Abstract.

This paper considers conditional Gaussian networks. The parameters in the network are learned by using conjugate Bayesian analysis. As conjugate local priors, we apply the Dirichlet distribution for discrete variables and the Gaussian-inverse gamma distribution for continuous variables, given a configuration of the discrete parents. We assume parameter independence and complete data. Further, to learn the structure of the network, the network score is deduced. We then develop a local master prior procedure, for deriving parameter priors in these networks. This procedure satisfies parameter independence, parameter modularity and likelihood equivalence. Bayes factors to be used in model search are introduced. Finally the methods derived are illustrated by a simple example.

1 Introduction

The aim of this paper is to present a method for learning the parameters and structure of a Bayesian network with discrete and continuous variables. In Heckerman et al. (1995) and Geiger and Heckerman (1994), this was done for respectively discrete networks and Gaussian networks.

We define the local probability distributions such that the joint distribution of the random variables is a conditional Gaussian (CG) distribution. Therefore we do not allow discrete variables to have continuous parents, so the network factorizes into a discrete part and a mixed part. The local conjugate parameter priors are for the discrete part of the network specified as Dirichlet distributions and for the mixed part of the network as Gaussian-inverse gamma distributions, for each configuration of discrete parents.

To learn the structure, D , of a network from data, d , we use the network score, $p(d, D)$, as a measure of how probable D is. To be able to calculate this score for all possible structures, we derive a method for finding the prior distribution of the parameters in the possible structures, from marginal priors calculated from an imaginary database. The method satisfies parameter independence, parameter modularity and likelihood equivalence. If used on networks with only

discrete or only continuous variables, it coincides with the methods developed in Heckerman et al. (1995) and Geiger and Heckerman (1994).

When many structures are possible, some kind of strategy to search for the structure with the highest score, has to be applied. In Cooper and Herskovits (1992), different search strategies are presented. Many of these strategies use Bayes factors for comparing the network scores of two different networks that differ by the direction of a single arrow or by the presence of a single arrow. We therefore deduce the Bayes factors for these two cases. To reduce the number of comparisons needed, we identify classes of structures for which the corresponding Bayes factor for testing an arrow between the same two variables in a network, is the same.

Finally a simple example is presented to illustrate some of the methods developed.

In this paper, we follow standard convention for drawing a Bayesian network and use shaded nodes to represent discrete variables and clear nodes to represent continuous variables.

The results in Section 2 to Section 7 are also published in Bøttcher (2001).

2 Bayesian Networks

A Bayesian network is a graphical model that encodes the joint probability distribution for a set of variables X . For terminology and theoretical aspects on graphical models, see Lauritzen (1996). In this paper we define it as consisting of

- A directed acyclic graph (DAG) $D = (V, E)$, where V is a finite set of vertices and E is a finite set of directed edges between the vertices. The DAG defines the structure of the Bayesian network.
- To each vertex $v \in V$ in the graph corresponds a random variable X_v , with state space \mathcal{X}_v . The set of variables associated with the graph D is then $X = (X_v)_{v \in V}$. Often we do not distinguish between a variable X_v and the corresponding vertex v .
- To each vertex v with parents $\text{pa}(v)$, there is attached a local probability distribution, $p(x_v | x_{\text{pa}(v)})$. The set of local probability distributions for all variables in the network is denoted \mathcal{P} .

- The possible lack of directed edges in D encodes conditional independencies between the random variables X through the factorization of the joint probability distribution,

$$p(x) = \prod_{v \in V} p(x_v | x_{\text{pa}(v)}).$$

A Bayesian network for a set of random variables X is thus the pair (D, \mathcal{P}) . In order to specify a Bayesian network for X , we must therefore specify a DAG D and a set \mathcal{P} of local probability distributions.

3 Bayesian Networks for Mixed Variables

In this paper we are interested in specifying networks for random variables X of which some are discrete and some are continuous. So we consider a DAG $D = (V, E)$ with vertices $V = \Delta \cup \Gamma$, where Δ and Γ are the sets of discrete and continuous vertices, respectively. The corresponding random variables X can then be denoted $X = (X_v)_{v \in V} = (I, Y) = ((I_\delta)_{\delta \in \Delta}, (Y_\gamma)_{\gamma \in \Gamma})$, *i.e.* we use I and Y for the sets of discrete and continuous variables, respectively. We denote the set of levels for each discrete variable $\delta \in \Delta$ as \mathcal{I}_δ .

In this paper we do not allow discrete variables to have continuous parents. This *e.g.* ensures availability of exact local computation methods, see Lauritzen (1992) and Lauritzen and Jensen (2001). The joint probability distribution then factorizes as follows:

$$p(x) = p(i, y) = \prod_{\delta \in \Delta} p(i_\delta | i_{\text{pa}(\delta)}) \prod_{\gamma \in \Gamma} p(y_\gamma | i_{\text{pa}(\gamma)}, y_{\text{pa}(\gamma)}),$$

where $i_{\text{pa}(\gamma)}$ and $y_{\text{pa}(\gamma)}$ denote observations of the discrete and continuous parents respectively, *i.e.* $i_{\text{pa}(\gamma)}$ is an abbreviation of $i_{\text{pa}(\gamma) \cap \Delta}$ etc.

We see that the joint probability distribution factorizes into a purely discrete part and a mixed part. First we look at the discrete part.

3.1 The Discrete Part of the Network

We assume that the local probability distributions are unrestricted discrete distributions with

$$p(i_\delta | i_{\text{pa}(\delta)}) \geq 0 \quad \forall \quad \delta \in \Delta.$$

A way to parameterize this is to let

$$\theta_{i_\delta|i_{\text{pa}(\delta)}} = p(i_\delta|i_{\text{pa}(\delta)}, \theta_{\delta|i_{\text{pa}(\delta)}}), \quad (1)$$

where $\theta_{\delta|i_{\text{pa}(\delta)}} = (\theta_{i_\delta|i_{\text{pa}(\delta)}})_{i_\delta \in \mathcal{I}_\delta}$.

Then $\sum_{i_\delta \in \mathcal{I}_\delta} \theta_{i_\delta|i_{\text{pa}(\delta)}} = 1$ and $0 \leq \theta_{i_\delta|i_{\text{pa}(\delta)}} \leq 1$. All parameters associated with a node δ is denoted θ_δ , *i.e.* $\theta_\delta = (\theta_{i_\delta|i_{\text{pa}(\delta)}})_{i_\delta \in \mathcal{I}_\delta}$.

Using this parameterization, the discrete part of the joint probability distribution is given by

$$p(i | (\theta_\delta)_{\delta \in \Delta}) = \prod_{\delta \in \Delta} p(i_\delta | i_{\text{pa}(\delta)}, \theta_{\delta|i_{\text{pa}(\delta)}}).$$

3.2 The Mixed Part of the Network

Now consider the mixed part. We assume that the local probability distributions are Gaussian linear regressions on the continuous parents, with parameters depending on the configuration of the discrete parents. Let the parameters in the distribution be given by $\theta_{\gamma|i_{\text{pa}(\gamma)}} = (m_{\gamma|i_{\text{pa}(\gamma)}}, \beta_{\gamma|i_{\text{pa}(\gamma)}}, \sigma_{\gamma|i_{\text{pa}(\gamma)}}^2)$. Then

$$(Y_\gamma | i_{\text{pa}(\gamma)}, y_{\text{pa}(\gamma)}, \theta_{\gamma|i_{\text{pa}(\gamma)}}) \sim \mathcal{N}(m_{\gamma|i_{\text{pa}(\gamma)}} + \beta_{\gamma|i_{\text{pa}(\gamma)}} y_{\text{pa}(\gamma)}, \sigma_{\gamma|i_{\text{pa}(\gamma)}}^2), \quad (2)$$

where $\beta_{\gamma|i_{\text{pa}(\gamma)}}$ are the regression coefficients, $m_{\gamma|i_{\text{pa}(\gamma)}}$ is the regression intercept, and $\sigma_{\gamma|i_{\text{pa}(\gamma)}}^2$ is the conditional variance. Thus for each configuration of the discrete parents of γ , the distribution of Y_γ is Gaussian with mean and variance given as in (2). There are three special cases of the above situation, namely when γ has no discrete parents, when it has no continuous parents and when it has no parents at all. If it has no discrete parents, (2) is just the Gaussian distribution,

$$(Y_\gamma | y_{\text{pa}(\gamma)}, \theta_\gamma) \sim \mathcal{N}(m_\gamma + \beta_\gamma y_{\text{pa}(\gamma)}, \sigma_\gamma^2),$$

and $\theta_\gamma = (m_\gamma, \beta_\gamma, \sigma_\gamma^2)$. When γ has no continuous parents, we have

$$(Y_\gamma | i_{\text{pa}(\gamma)}, \theta_{\gamma|i_{\text{pa}(\gamma)}}) \sim \mathcal{N}(m_{\gamma|i_{\text{pa}(\gamma)}}, \sigma_{\gamma|i_{\text{pa}(\gamma)}}^2),$$

with $\theta_{\gamma|i_{\text{pa}(\gamma)}} = (m_{\gamma|i_{\text{pa}(\gamma)}}, \sigma_{\gamma|i_{\text{pa}(\gamma)}}^2)$, *i.e.* for each γ , the mean depends solely on $i_{\text{pa}(\gamma)}$. Finally, when γ has no parents at all,

$$(Y_\gamma | \theta_\gamma) \sim \mathcal{N}(m_\gamma, \sigma_\gamma^2),$$

with $\theta_\gamma = (m_\gamma, \sigma_\gamma^2)$.

With $\theta_\gamma = (\theta_{\gamma|i_{\text{pa}(\gamma)}})_{i_{\text{pa}(\gamma)} \in \mathcal{I}_{\text{pa}(\gamma)}}$, the mixed part of the joint distribution can be written as

$$p(y|i, (\theta_\gamma)_{\gamma \in \Gamma}) = \prod_{\gamma \in \Gamma} p(y_\gamma | i_{\text{pa}(\gamma)}, y_{\text{pa}(\gamma)}, \theta_{\gamma|i_{\text{pa}(\gamma)}}).$$

3.3 The Joint Network

If we let $\theta = ((\theta_\delta)_{\delta \in \Delta}, (\theta_\gamma)_{\gamma \in \Gamma})$, the joint probability distribution for $X = (I, Y)$ is given by

$$p(x|\theta) = \prod_{\delta \in \Delta} p(i_\delta | i_{\text{pa}(\delta)}, \theta_{\delta|i_{\text{pa}(\delta)}}) \prod_{\gamma \in \Gamma} p(y_\gamma | i_{\text{pa}(\gamma)}, y_{\text{pa}(\gamma)}, \theta_{\gamma|i_{\text{pa}(\gamma)}}). \quad (3)$$

It can easily be shown by induction that when the local probability distributions are given as defined in (1) and (2), the joint probability distribution for X is a CG distribution with density of the form

$$p(x|\theta) = p(i, y|\theta) = p(i) |2\pi \Sigma_i|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(y - M_i)^T \Sigma_i^{-1} (y - M_i)\right\}.$$

For each i , M_i is the unconditional mean, that is unconditional on continuous variables and Σ_i is the covariance matrix for all the continuous variables in the network. In Shachter and Kenley (1989) formulas for calculating Σ_i from the local probability distributions can be found.

A Bayesian network, where the joint probability distribution is a CG distribution is in the following called a *CG network*.

4 Learning the Parameters in a CG Network

When constructing a Bayesian network there is, as mentioned earlier, two things to consider, namely specifying the DAG and specifying the local probability distributions. In this section we assume that the structure of the DAG is known and the distribution type is given as in the previous section and we consider the specification of the parameters in the distributions. For this we need the concept of conjugate Bayesian analysis.

4.1 Conjugate Bayesian Analysis

There are several ways of assessing the parameters in probability distributions. An expert could specify them, or they could be estimated from data. In our approach we encode our uncertainty about the parameter θ in a *prior* distribution $p(\theta)$, use data to update this distribution, *i.e.* learn the parameter and hereby, by using Bayes' theorem, obtain the *posterior* distribution $p(\theta|\text{data})$, see DeGroot (1970).

Consider a situation with one random variable X . Let θ be the parameter to be assessed, Θ the parameter space and d a random sample of size n from the probability distribution $p(x|\theta)$. We call d our database and $x^c \in d$ a case. Then, according to Bayes' theorem,

$$p(\theta|d) = \frac{p(d|\theta)p(\theta)}{p(d)}, \quad \theta \in \Theta, \quad (4)$$

where $p(d|\theta) = \prod_{x^c \in d} p(x^c|\theta)$ is the joint probability distribution of d , also called the likelihood of θ . Furthermore the denominator is given by

$$p(d) = \int_{\Theta} p(d|\theta)p(\theta)d\theta,$$

and for fixed d it may be considered as a normalizing constant. Therefore (4) can be expressed as

$$p(\theta|d) \propto p(d|\theta)p(\theta),$$

where the proportionality constant is determined by the relation $\int_{\Theta} p(\theta|d)d\theta = 1$.

When the prior distribution belongs to a given family of distributions and the posterior distribution, after sampling from a specific distribution, belongs to the same family of distributions, then this family is said to be closed under sampling and called a *conjugate family* of distributions. Further, if a parameter or the distribution of a parameter has a certain property which is preserved under sampling, then this property is said to be a *conjugate property*.

In a conjugate family of distributions it is generally straightforward to calculate the posterior distribution.

4.2 Some Simplifying Properties

In the previous section we showed how to update a prior distribution for a single parameter θ . In a Bayesian network with more than one variable, we also have to look at the relationship between the different parameters for the different variables in the network. In this paper we assume that the parameters associated with one variable is independent of the parameters associated with the other variables. This assumption was introduced by Spiegelhalter and Lauritzen (1990) and we denote it *global parameter independence*. In addition to this, we will assume that the parameters are independent for each configuration of the discrete parents, which we denote as *local parameter independence*. So if the parameters have the property of global parameter independence and local parameter independence, then

$$p(\theta) = \prod_{\delta \in \Delta} \prod_{i_{\text{pa}(\delta)} \in \mathcal{I}_{\text{pa}(\delta)}} p(\theta_{\delta} | i_{\text{pa}(\delta)}) \prod_{\gamma \in \Gamma} \prod_{i_{\text{pa}(\gamma)} \in \mathcal{I}_{\text{pa}(\gamma)}} p(\theta_{\gamma} | i_{\text{pa}(\gamma)}), \quad (5)$$

and we will refer to (5) simply as *parameter independence*.

A consequence of parameter independence is that, for each configuration of the discrete parents, we can update the parameters in the local distributions independently. This also means that if we have *local conjugacy*, *i.e.* the distributions of $\theta_{\delta} | i_{\text{pa}(\delta)}$ and $\theta_{\gamma} | i_{\text{pa}(\gamma)}$ belongs to a conjugate family, then because of parameter independence, we have *global conjugacy*, *i.e.* the joint distribution of θ belongs to a conjugate family.

Further, we will assume that the database d is complete, that is, in each case it contains at least one instance of every random variable in the network. With this we can show that parameter independence is a conjugate property.

Due to the factorization (3) and the assumption of complete data,

$$\begin{aligned} p(d|\theta) &= \prod_{c \in d} p(x^c | \theta) \\ &= \prod_{c \in d} \left(\prod_{\delta \in \Delta} p(i_{\delta}^c | i_{\text{pa}(\delta)}^c, \theta_{\delta} | i_{\text{pa}(\delta)}) \prod_{\gamma \in \Gamma} p(y_{\gamma}^c | y_{\text{pa}(\gamma)}^c, i_{\text{pa}(\gamma)}^c, \theta_{\gamma} | i_{\text{pa}(\gamma)}) \right), \end{aligned}$$

where i^c and y^c respectively denotes the discrete part and the continuous part of

a case x^c . Another way of writing the above equation is

$$\begin{aligned}
 p(d|\theta) &= \prod_{\delta \in \Delta} \prod_{i_{\text{pa}(\delta)} \in \mathcal{I}_{\text{pa}(\delta)}} \prod_{c: i_{\text{pa}(\delta)}^c = i_{\text{pa}(\delta)}} p(i_{\delta}^c | i_{\text{pa}(\delta)}, \theta_{\delta} | i_{\text{pa}(\delta)}) \\
 &\times \prod_{\gamma \in \Gamma} \prod_{i_{\text{pa}(\gamma)} \in \mathcal{I}_{\text{pa}(\gamma)}} \prod_{c: i_{\text{pa}(\gamma)}^c = i_{\text{pa}(\gamma)}} p(y_{\gamma}^c | y_{\text{pa}(\gamma)}^c, i_{\text{pa}(\gamma)}, \theta_{\gamma} | i_{\text{pa}(\gamma)}), \tag{6}
 \end{aligned}$$

where the product over cases is split up into a product over the configurations of the discrete parents and a product over those cases, where the configuration of the discrete parents is the same as the currently processed configuration. Notice however that some of the parent configurations might not be represented in the database, in which case the product over cases with this parent configuration just adds nothing to the overall product.

By combining (5) and (6) it is seen that

$$p(\theta|d) = \prod_{\delta \in \Delta} \prod_{i_{\text{pa}(\delta)} \in \mathcal{I}_{\text{pa}(\delta)}} p(\theta_{\delta} | i_{\text{pa}(\delta)} | d) \prod_{\gamma \in \Gamma} \prod_{i_{\text{pa}(\gamma)} \in \mathcal{I}_{\text{pa}(\gamma)}} p(\theta_{\gamma} | i_{\text{pa}(\gamma)} | d),$$

i.e. the parameters remain independent given data. We call this property *posterior parameter independence*. In other words, the properties of local and global independence are conjugate.

Notice that the posterior distribution, $p(\theta|d)$, can be found using *batch* learning or *sequential* learning. In batch learning, $p(\theta|d)$ is found by updating $p(\theta)$ with all cases in d at the same time, *i.e.* in a batch. In sequential learning, $p(\theta)$ is updated one case at a time, using the previous posterior distribution as the prior distribution for the next case to be considered. When the database d is complete, batch learning and sequential learning leads to the same posterior distribution and the final result is independent of the order in which the cases in d are processed. It is of course also possible to process some of the cases in a batch and the rest sequentially, which could be done if *e.g.* a new case is added to an already processed database, see Bernardo and Smith (1994).

4.3 Learning in the Discrete Case

We now consider batch learning of the parameters in the discrete part of the network. Recall that the local probability distributions are unrestricted discrete distributions defined as in (1). As pointed out in the previous section we can,

because of the assumption of parameter independence, find the posterior distribution of $\theta_{\delta|i_{\text{pa}(\delta)}}$ for each δ and each configuration of $\text{pa}(\delta)$ independently.

So given a specific configuration of $i_{\text{pa}(\delta)}$, we need to find $p(\theta_{\delta|i_{\text{pa}(\delta)}}|d)$. From Bayes' theorem, Equation (4), we have that

$$p(\theta_{\delta|i_{\text{pa}(\delta)}}|d) \propto \prod_{c: i_{\text{pa}(\delta)}^c = i_{\text{pa}(\delta)}} p(i_{\delta}^c|i_{\text{pa}(\delta)}, \theta_{\delta|i_{\text{pa}(\delta)}}) p(\theta_{\delta|i_{\text{pa}(\delta)}}). \quad (7)$$

A conjugate family for multinomial observations is the family of Dirichlet distributions. So let the prior distribution of $\theta_{\delta|i_{\text{pa}(\delta)}}$ be a Dirichlet distribution \mathcal{D} with hyperparameters $\alpha_{\delta|i_{\text{pa}(\delta)}} = (\alpha_{i_{\delta}|i_{\text{pa}(\delta)}})_{i_{\delta} \in \mathcal{I}_{\delta}}$, also written as

$$(\theta_{\delta|i_{\text{pa}(\delta)}}|\alpha_{\delta|i_{\text{pa}(\delta)}}) \sim \mathcal{D}(\alpha_{\delta|i_{\text{pa}(\delta)}}). \quad (8)$$

The probability function for this Dirichlet distribution is given by

$$p(\theta_{\delta|i_{\text{pa}(\delta)}}|\alpha_{\delta|i_{\text{pa}(\delta)}}) = \frac{\Gamma(\alpha_{+\delta|i_{\text{pa}(\delta)}})}{\prod_{i_{\delta} \in \mathcal{I}_{\delta}} \Gamma(\alpha_{i_{\delta}|i_{\text{pa}(\delta)}})} \prod_{i_{\delta} \in \mathcal{I}_{\delta}} (\theta_{i_{\delta}|i_{\text{pa}(\delta)}})^{\alpha_{i_{\delta}|i_{\text{pa}(\delta)}} - 1},$$

where $\alpha_{+\delta|i_{\text{pa}(\delta)}} = \sum_{i_{\delta} \in \mathcal{I}_{\delta}} \alpha_{i_{\delta}|i_{\text{pa}(\delta)}}$ and $\Gamma(\cdot)$ is the gamma function. Because of notational convenience, we do not in what follows write the hyperparameters explicitly in the conditioning.

It then follows from (7) and (8) that the posterior distribution is given as

$$(\theta_{\delta|i_{\text{pa}(\delta)}}|d) \sim \mathcal{D}(\alpha_{\delta|i_{\text{pa}(\delta)}} + n_{\delta|i_{\text{pa}(\delta)}}),$$

where the vector $n_{\delta|i_{\text{pa}(\delta)}} = (n_{i_{\delta}|i_{\text{pa}(\delta)}})_{i_{\delta} \in \mathcal{I}_{\delta}}$, also called the counts, denotes the number of observations in d where δ and $\text{pa}(\delta)$ have that specific configuration. Notice that, for at given parent configuration, the number of observations in a batch, $|b|$, is the same as $n_{+\delta|i_{\text{pa}(\delta)}}$, where $n_{+\delta|i_{\text{pa}(\delta)}} = \sum_{i_{\delta} \in \mathcal{I}_{\delta}} n_{i_{\delta}|i_{\text{pa}(\delta)}}$.

Because of parameter independence, the joint prior distribution of all the parameters for the discrete variables in the network, is given by the product of the local parameter priors.

The above learning procedure can also be used for sequential learning by applying the above formulas one case at a time, using the previous posterior distribution as the prior distribution for the next case to be processed.

4.4 Learning in the Mixed Case

In the mixed case we write the local probability distributions as

$$(Y_\gamma | i_{\text{pa}(\gamma)}, y_{\text{pa}(\gamma)}, \theta_\gamma | i_{\text{pa}(\gamma)}) \sim \mathcal{N}(z_{\text{pa}(\gamma)}(m_\gamma | i_{\text{pa}(\gamma)}, \beta_\gamma | i_{\text{pa}(\gamma)})^\top, \sigma_{\gamma | i_{\text{pa}(\gamma)}}^2),$$

where $z_{\text{pa}(\gamma)} = (1, y_{\text{pa}(\gamma)})$. This vector has dimension $k + 1$, where k is the number of continuous parents to γ .

As in the discrete case we can because of parameter independence update the parameters for each γ and each configuration of the discrete parents independently. By Bayes' theorem,

$$p(\theta_\gamma | i_{\text{pa}(\gamma)} | d) \propto \prod_{c: i_{\text{pa}(\gamma)}^c = i_{\text{pa}(\gamma)}} p(y_\gamma^c | y_{\text{pa}(\gamma)}^c, i_{\text{pa}(\gamma)}, \theta_\gamma | i_{\text{pa}(\gamma)}) p(\theta_\gamma | i_{\text{pa}(\gamma)}).$$

We now join all the observations y_γ^c for which $i_{\text{pa}(\gamma)}^c = i_{\text{pa}(\gamma)}$ in a vector y_γ^b , *i.e.* $y_\gamma^b = (y_\gamma^c)_{i_{\text{pa}(\gamma)}^c = i_{\text{pa}(\gamma)}}$. The same is done with the observations of the continuous parents of γ , *i.e.* $y_{\text{pa}(\gamma)}^b = (y_{\text{pa}(\gamma)}^c)_{i_{\text{pa}(\gamma)}^c = i_{\text{pa}(\gamma)}}$. As the observations in d are independent, $p(y_\gamma^b | y_{\text{pa}(\gamma)}^b, i_{\text{pa}(\gamma)}, \theta_\gamma | i_{\text{pa}(\gamma)})$ is the likelihood function for a multivariate normal distribution with mean vector $z_{\text{pa}(\gamma)}^b(m_\gamma | i_{\text{pa}(\gamma)}, \beta_\gamma | i_{\text{pa}(\gamma)})^\top$ and covariance matrix $\sigma_{\gamma | i_{\text{pa}(\gamma)}}^2 I$, where I is the identity matrix and $z_{\text{pa}(\gamma)}^b$ is defined through $y_{\text{pa}(\gamma)}^b$.

The posterior distribution of $\theta_\gamma | i_{\text{pa}(\gamma)}$ can now be written as

$$p(\theta_\gamma | i_{\text{pa}(\gamma)} | d) \propto p(y_\gamma^b | y_{\text{pa}(\gamma)}^b, i_{\text{pa}(\gamma)}, \theta_\gamma | i_{\text{pa}(\gamma)}) p(\theta_\gamma | i_{\text{pa}(\gamma)}).$$

A standard conjugate family for these observations is the family of Gaussian-inverse gamma distributions. Let the prior joint distribution of $(m_\gamma | i_{\text{pa}(\gamma)}, \beta_\gamma | i_{\text{pa}(\gamma)})$ and $\sigma_{\gamma | i_{\text{pa}(\gamma)}}^2$ be as follows.

$$\begin{aligned} (m_\gamma | i_{\text{pa}(\gamma)}, \beta_\gamma | i_{\text{pa}(\gamma)} | \sigma_{\gamma | i_{\text{pa}(\gamma)}}^2) &\sim \mathcal{N}_{k+1}(\mu_\gamma | i_{\text{pa}(\gamma)}, \sigma_{\gamma | i_{\text{pa}(\gamma)}}^2 \tau_{\gamma | i_{\text{pa}(\gamma)}}^{-1}) \\ (\sigma_{\gamma | i_{\text{pa}(\gamma)}}^2) &\sim \mathcal{IG}\left(\frac{\rho_\gamma | i_{\text{pa}(\gamma)}}{2}, \frac{\phi_\gamma | i_{\text{pa}(\gamma)}}{2}\right). \end{aligned}$$

The posterior distribution is then

$$\begin{aligned} (m_{\gamma|i_{\text{pa}(\gamma)}}, \beta_{\gamma|i_{\text{pa}(\gamma)}} | \sigma_{\gamma|i_{\text{pa}(\gamma)}}^2, d) &\sim \mathcal{N}_{k+1}(\mu'_{\gamma|i_{\text{pa}(\gamma)}}, \sigma_{\gamma|i_{\text{pa}(\gamma)}}^2 (\tau_{\gamma|i_{\text{pa}(\gamma)}}^{-1})') \\ (\sigma_{\gamma|i_{\text{pa}(\gamma)}}^2 | d) &\sim \mathcal{IG}\left(\frac{\rho'_{\gamma|i_{\text{pa}(\gamma)}}}{2}, \frac{\phi'_{\gamma|i_{\text{pa}(\gamma)}}}{2}\right), \end{aligned}$$

where

$$\begin{aligned} \tau'_{\gamma|i_{\text{pa}(\gamma)}} &= \tau_{\gamma|i_{\text{pa}(\gamma)}} + (z_{\text{pa}(\gamma)}^b)^{\text{T}} z_{\text{pa}(\gamma)}^b \\ \mu'_{\gamma|i_{\text{pa}(\gamma)}} &= (\tau'_{\gamma|i_{\text{pa}(\gamma)}})^{-1} (\tau_{\gamma|i_{\text{pa}(\gamma)}} \mu_{\gamma|i_{\text{pa}(\gamma)}} + (z_{\text{pa}(\gamma)}^b)^{\text{T}} y_{\gamma}^b) \\ \rho'_{\gamma|i_{\text{pa}(\gamma)}} &= \rho_{\gamma|i_{\text{pa}(\gamma)}} + |b| \\ \phi'_{\gamma|i_{\text{pa}(\gamma)}} &= \phi_{\gamma|i_{\text{pa}(\gamma)}} + (y_{\gamma}^b - z_{\text{pa}(\gamma)}^b \mu'_{\gamma|i_{\text{pa}(\gamma)}})^{\text{T}} y_{\gamma}^b \\ &\quad + (\mu_{\gamma|i_{\text{pa}(\gamma)}} - \mu'_{\gamma|i_{\text{pa}(\gamma)}})^{\text{T}} \tau_{\gamma|i_{\text{pa}(\gamma)}} \mu_{\gamma|i_{\text{pa}(\gamma)}}, \end{aligned}$$

where $|b|$ denotes the number of observations in b .

As for the discrete variables, we can with these formulas also use the sequential approach and update the parameters one case at a time.

Further, because of parameter independence, the joint prior distribution is given as the product of the local prior distributions for all parameters in the network.

5 Learning the Structure of a CG Network

In this section we consider how to learn the structure of a CG network.

5.1 The Network Score

There are basically two ways of determining which DAG should represent the conditional independencies between a set of random variables. First, if the relations between the variables are well understood by an expert, then he could specify the DAG, using a causal interpretation of the arrows. Second, we could learn the DAG from data. That is, we could find out how well a DAG D represents the conditional independencies, by measuring how probable D is, given that we have observed data d . Different approaches use different measures. An

often used measure is the posterior probability of the DAG, $p(D|d)$, which from Bayes' theorem is given by

$$p(D|d) \propto p(d|D)p(D),$$

where $p(d|D)$ is the likelihood of D and $p(D)$ is the prior probability. As the normalizing constant does not depend upon structure, another measure, which gives the relative probability, is

$$p(D, d) = p(d|D)p(D).$$

We refer to the above measures as *network scores*. So learning the DAG from data, we can in principle first calculate the network scores for all possible DAGs and then select the DAG with the highest network score. If many DAGs are possible, it is computationally infeasible to calculate the network score for all these DAGs. In this situation it is necessary to use some kind of search strategy to find the DAG with the highest score, see *e.g.* Cooper and Herskovits (1992).

In some cases it can be more accurate to average over the possible DAGs for prediction, instead of just selecting a single DAG. So if x is the quantity we are interested in, we can use the weighted average,

$$p(x|d) = \sum_{D \in DAG} p(x|d, D)p(D|d),$$

where DAG is the set of all DAGs and $p(D|d)$ is the weight.

Again, if many DAGs are possible, this sum is too heavy to compute, so instead, by using a search strategy, we can find a few DAGs with high score and average over these.

5.2 The Network Score for a CG Network

In order to calculate the network score for a specific DAG D , we need to know the prior probability and the likelihood of the DAG. For simplicity, we could for example choose to let all DAGs be equally likely, then

$$p(D|d) \propto p(d|D).$$

In a CG network, the likelihood of the DAG D is given by

$$\begin{aligned}
 p(d|D) &= \int_{\theta \in \Theta} p(d|\theta, D)p(\theta|D)d\theta \\
 &= \prod_{\delta \in \Delta} \prod_{i_{\text{pa}(\delta)} \in \mathcal{I}_{\text{pa}(\delta)}} \int \prod_{c: i_{\text{pa}(\delta)}^c = i_{\text{pa}(\delta)}} p(i_{\delta}^c | i_{\text{pa}(\delta)}, \theta_{\delta | i_{\text{pa}(\delta)}}, D) p(\theta_{\delta | i_{\text{pa}(\delta)}} | D) d\theta_{\delta | i_{\text{pa}(\delta)}} \\
 &\times \prod_{\gamma \in \Gamma} \prod_{i_{\text{pa}(\gamma)} \in \mathcal{I}_{\text{pa}(\gamma)}} \int \prod_{c: i_{\text{pa}(\gamma)}^c = i_{\text{pa}(\gamma)}} p(y_{\gamma}^c | y_{\text{pa}(\gamma)}^c, i_{\text{pa}(\gamma)}, \theta_{\gamma | i_{\text{pa}(\gamma)}}, D) p(\theta_{\gamma | i_{\text{pa}(\gamma)}} | D) d\theta_{\gamma | i_{\text{pa}(\gamma)}}.
 \end{aligned}$$

Again we see that we can consider the problem for the discrete part and the mixed part of the network separately.

The discrete part is from the formulas in Section 4.3 found to be

$$\prod_{\delta \in \Delta} \prod_{i_{\text{pa}(\delta)} \in \mathcal{I}_{\text{pa}(\delta)}} \frac{\Gamma(\alpha_{+\delta | i_{\text{pa}(\delta)}})}{\Gamma(\alpha_{+\delta | i_{\text{pa}(\delta)}} + n_{+\delta | i_{\text{pa}(\delta)}})} \prod_{i_{\delta} \in \mathcal{I}_{\delta}} \frac{\Gamma(\alpha_{i_{\delta} | i_{\text{pa}(\delta)}} + n_{i_{\delta} | i_{\text{pa}(\delta)}})}{\Gamma(\alpha_{i_{\delta} | i_{\text{pa}(\delta)}})}.$$

In the mixed part of the network, the local marginal likelihoods are non-central t distributions with $\rho_{\gamma | i_{\text{pa}(\gamma)}}$ degrees of freedom, location vector $z_{\text{pa}(\gamma)}^b \mu_{\gamma | i_{\text{pa}(\gamma)}}$ and scale parameter $s_{\gamma | i_{\text{pa}(\gamma)}} = \frac{\phi_{\gamma | i_{\text{pa}(\gamma)}}}{\rho_{\gamma | i_{\text{pa}(\gamma)}}} (I + (z_{\text{pa}(\gamma)}^b)^{\top} \tau_{\gamma | i_{\text{pa}(\gamma)}}^{-1} (z_{\text{pa}(\gamma)}^b)^{\top})^{-1}$. The index b is defined as in Section 4.4.

So the mixed part is given by

$$\prod_{\gamma \in \Gamma} \prod_{i_{\text{pa}(\gamma)} \in \mathcal{I}_{\text{pa}(\gamma)}} \frac{\Gamma((\rho_{\gamma | i_{\text{pa}(\gamma)}} + |b|)/2)}{\Gamma(\rho_{\gamma | i_{\text{pa}(\gamma)}}/2) [\det(\rho_{\gamma | i_{\text{pa}(\gamma)}} s_{\gamma | i_{\text{pa}(\gamma)}} \pi)]^{\frac{1}{2}}} \times \left[1 + \frac{1}{\rho_{\gamma | i_{\text{pa}(\gamma)}}} (y_{\gamma}^b - z_{\text{pa}(\gamma)}^b \mu_{\gamma | i_{\text{pa}(\gamma)}}) s_{\gamma | i_{\text{pa}(\gamma)}}^{-1} (y_{\gamma}^b - z_{\text{pa}(\gamma)}^b \mu_{\gamma | i_{\text{pa}(\gamma)}})^{\top} \right]^{\frac{-(\rho_{\gamma | i_{\text{pa}(\gamma)}} + |b|)}{2}}.$$

The network score for a CG network is thus the product of the prior probability for the DAG D , the term for the discrete part and the term for the mixed part. Notice that the network score has the property that it factorizes into a product over terms involving only one node and its parents. This property is called *decomposability*.

To evaluate which DAG or possible several DAGs that represent the conditional independencies in a Bayesian network well, we want to find the DAG or DAGs

with the highest network scores. To calculate these scores, we must specify the local probability distributions and the local prior distributions for the parameters for each network under evaluation. In the next section, a method for doing this is developed.

6 The Master Prior Procedure

The papers Heckerman et al. (1995) and Geiger and Heckerman (1994) develops a method for finding the prior distributions for the parameters in respectively the purely discrete case and the purely continuous case. The work is based on principles of likelihood equivalence, parameter modularity, and parameter independence. It leads to a method where the parameter priors for all possible networks are deduced from one joint prior distribution, in the following called a *master prior* distribution.

In this paper we will build on this idea, which can be used on networks with mixed variables. We will therefore in the following describe their method for the pure cases.

6.1 The Master Prior in the Discrete Case

In the purely discrete case, or the discrete part of a mixed network, the following is a well known classical result.

Let A be a subset of Δ and let $B = \Delta \setminus A$. Let the discrete variables i have the joint distribution

$$p(i|\Psi) = \Psi_i.$$

Notice here, that the set $\Psi = (\Psi_i)_{i \in \mathcal{I}}$ contains the parameters for the joint distribution, contrary to θ in Section 3, which contains the parameters for the conditional local distributions.

In the following we use the notation $z_{i_A} = \sum_{j:j_A=i_A} z_j$, where z is any parameter. Then the marginal distribution of i_A is given by

$$p(i_A|\Psi) = \Psi_{i_A},$$

and the conditional distribution of i_B given i_A is

$$p(i_B|i_A, \Psi) = \frac{\Psi_i}{\Psi_{i_A}} = \Psi_{i_B|i_A}.$$

Further if the joint prior distribution for the parameters Ψ is Dirichlet, that is

$$(\Psi) \sim \mathcal{D}(\alpha),$$

where $\alpha = (\alpha_i)_{i \in \mathcal{I}}$, then the marginal distribution of Ψ_A is Dirichlet, *i.e.*

$$(\Psi_A) \sim \mathcal{D}(\alpha_A),$$

with $\alpha_A = (\alpha_{i_A})_{i_A \in \mathcal{I}_A}$. The conditional distribution of $\Psi_{B|i_A}$ is

$$(\Psi_{B|i_A}) \sim \mathcal{D}(\alpha_{B|i_A}),$$

with $\alpha_{B|i_A} = (\alpha_{i_B|i_A})_{i_B \in \mathcal{I}_B}$ and $\alpha_{i_B|i_A} = \alpha_i$. Furthermore the parameters are independent, that is

$$p(\Psi) = \prod_{i_A \in \mathcal{I}_A} p(\Psi_{B|i_A}) p(\Psi_A). \quad (9)$$

From the above result we see, that for each possible parent/child relationship, we can find the marginal parameter prior $p(\Psi_{\delta \cup \text{pa}(\delta)})$. Further, from this marginal distribution we can, for each configuration of the parents, find the conditional local prior distribution $p(\Psi_{\delta|i_{\text{pa}(\delta)}})$. Notice that $\Psi_{\delta|i_{\text{pa}(\delta)}} = \theta_{\delta|i_{\text{pa}(\delta)}}$, where $\theta_{\delta|i_{\text{pa}(\delta)}}$ was specified for the conditional distributions in Section (3.1). Further, because of parameter independence, given by (9), we can find the joint parameter prior for any network as the product of the local priors involved.

To use this method, we must therefore specify the joint Dirichlet distribution, *i.e.* the master Dirichlet prior. This was first done in Heckerman et al. (1995) and here we follow their method. We start by specifying a prior Bayesian network (D, \mathcal{P}) . From this we calculate the joint distribution $p(i|\Psi) = \Psi_i$. To specify a master Dirichlet distribution, we must specify the parameters $\alpha = (\alpha_{i_\delta})_{i_\delta \in \mathcal{I}}$ and for this we use the following relation for the Dirichlet distribution,

$$p(i) = \mathbb{E}(\Psi_i) = \frac{\alpha_i}{n},$$

with $n = \sum_{i \in \mathcal{I}} \alpha_i$. Now we let the probabilities in the prior network be an estimate of $\mathbb{E}(\Psi_i)$, so we only need to determine n in order to calculate the parameters α_i . We determine n by using the notion of an imaginary database. We imagine that we have a database of cases, from which we from total ignorance have updated the distribution of Ψ . The sample size of this imaginary database is thus n . Therefore we refer to the estimate of n as the *imaginary sample size* and it expresses how much confidence we have in the dependency structure expressed in the prior network.

6.2 The Master Prior in the Gaussian Case

For the Gaussian case, the following result is used, see *e.g.* Dawid and Lauritzen (1993). Let A be a subset of Γ and let $B = \Gamma \setminus A$. If

$$(y|m, \Sigma) \sim \mathcal{N}(m, \Sigma),$$

then

$$(y_A|m, \Sigma) \sim \mathcal{N}(m_A, \Sigma_{AA})$$

and

$$(y_B|y_A, m_{B|A}, \beta_{B|A}, \Sigma_{B|A}) \sim \mathcal{N}(m_{B|A} + \beta_{B|A}y_A, \Sigma_{B|A}),$$

where

$$\Sigma = \begin{pmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{pmatrix}, \quad \Sigma_{B|A} = \Sigma_{BB} - \Sigma_{BA}\Sigma_{AA}^{-1}\Sigma_{AB},$$

$$m_{B|A} = m_B - \beta_{B|A}m_A \quad \text{and} \quad \beta_{B|A} = \Sigma_{BA}\Sigma_{AA}^{-1}.$$

Further, if

$$(m|\Sigma) \sim \mathcal{N}\left(\mu, \frac{1}{\nu}\Sigma\right) \quad \text{and} \quad (\Sigma) \sim \mathcal{IW}(\rho, \Phi),$$

where the scale matrix Φ is partitioned as Σ , then

- $(m_A|\Sigma_{AA}) \sim \mathcal{N}(\mu_A, \frac{1}{\nu}\Sigma_{AA})$
- $(\Sigma_{AA}) \sim \mathcal{IW}(\rho, \Phi_{AA})$
- $(\Sigma_{B|A}) \sim \mathcal{IW}(\rho + |A|, \Phi_{B|A})$
- $(m_{B|A}, \beta_{B|A}|\Sigma_{B|A}) \sim \mathcal{N}(\mu_{B|A}, \Sigma_{B|A} \otimes \tau_{B|A}^{-1})$
- $m_A, \Sigma_{AA} \perp\!\!\!\perp m_{B|A}, \beta_{B|A}, \Sigma_{B|A}$

where

$$\mu_{B|A} = (\mu_B - \Phi_{BA}\Phi_{AA}^{-1}\mu_A, \Phi_{BA}\Phi_{AA}^{-1})$$

and

$$\tau_{B|A}^{-1} = \begin{pmatrix} \frac{1}{\nu} + \mu_A^T \Phi_{AA}^{-1} \mu_A & -\mu_A^T \Phi_{AA}^{-1} \\ -\Phi_{AA}^{-1} \mu_A & \Phi_{AA}^{-1} \end{pmatrix},$$

and \otimes denotes the Kronecker product. Notice that the dimension of $\mu_{B|A}$ is given as $(|B|, |B| \times |A|)$.

As in the discrete case, this result shows us how to deduce the local probability distributions and the local prior distributions from the joint distributions.

Further, because of parameter independence, the joint parameter prior for any Gaussian network can be specified as the product of the local priors. Notice that the parameters found here for a node given its parents, coincides with the parameters specified in Section 3.2.

Before we show how to construct the master prior, we need the following result. The Gaussian-inverse Wishart prior is conjugate to observations from a Gaussian distribution (DeGroot 1970). So let the probability distribution and the prior distribution be given as above. Then, given the database $d = \{y^1, \dots, y^n\}$, the posterior distributions are

$$(m|\Sigma, d) \sim \mathcal{N}(\mu', \frac{1}{\nu'}\Sigma) \text{ and } (\Sigma|d) \sim \mathcal{IW}(\rho', \Phi'),$$

where

$$\begin{aligned} \nu' &= \nu + n, \\ \mu' &= \frac{\nu\mu + n\bar{y}}{\nu + n}, \\ \rho' &= \rho + n, \\ \Phi' &= \Phi + ssd + \frac{\nu n}{\nu + n}(\mu - \bar{y})(\mu - \bar{y})^T, \end{aligned} \tag{10}$$

with

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i \text{ and } ssd = \sum_{i=1}^n (y_i - \bar{y})(y_i - \bar{y})^T.$$

From these updating formulas we see that ν' and ρ' are updated with the number of cases in the database. Further μ' is a weighted average of the prior mean and the sample mean, each weighted by their sample sizes. Finally Φ is updated with the ssd , which expresses how much each observation differs from the sample mean, and an expression for how much the prior mean differs from the sample mean.

To specify the master prior, we need to specify the four parameters ν , μ , ρ and Φ . As for the discrete variables we start by specifying a prior Bayesian network, (D, \mathcal{P}) . From this, a prior joint probability distribution $p(y|m, \Sigma) = \mathcal{N}(m, \Sigma)$ can be deduced. Now imagine that the mean m and the variance Σ were calculated from an imaginary database, so that they actually are the sample mean and the sample variance. Further, assume that before this imaginary database was observed, we were totally ignorant about the parameters. The formulas in (10) can now be used to “update” the parameters on the basis of the imaginary

database. As we have not seen any cases before, ν and ρ are estimated by the size of the imaginary database. Further

$$\mu = m \quad \text{and} \quad \Phi = \text{ssd} = (\nu - 1)\Sigma.$$

In Geiger and Heckerman (1994), μ and Φ are found in a slightly different way. They use the fact that the marginal likelihood $p(y)$ is a multivariate non-central t distribution with ρ degrees of freedom, location vector μ and scale matrix $S = \frac{\nu+1}{\nu\rho}\Phi$. Now the mean and covariance matrix in the t distribution is given by

$$\mathbb{E}(y) = \mu \quad \text{and} \quad \text{Cov}(y) = \frac{\rho}{\rho - 2}S.$$

They then let the mean and covariance matrix from the prior network estimate the mean and covariance matrix in the t distribution, which implies that

$$\mu = m \quad \text{and} \quad \Phi = \frac{\nu(\rho - 2)}{\nu + 1}\Sigma.$$

Experimental results have not shown noticeable differences between the two approaches.

6.3 Properties of the Master Prior Procedure

The method for finding prior parameter distributions described in the previous section has some properties, which we will describe here. In this section we use Ψ as a parameter defined for a joint distribution, *i.e.* Ψ can be the parameter for the discrete variables or in the continuous case, $\Psi = (m, \Sigma)$.

Clearly a consequence of using the above method is that the parameters are independent. Further it can be seen, that if a node v has the same parents in two DAGs D and D^* , then

$$p(\Psi_{v|\text{pa}(v)}|D) = p(\Psi_{v|\text{pa}(v)}|D^*).$$

This property is referred to as *parameter modularity*. Now both the discrete and the Gaussian distribution has the property that if the joint probability distribution $p(x)$ can be factorized according to a DAG D , then it can also be factorized according to all other DAGs, which represents the same set of conditional independencies as D . A set of DAGs, D^e , which represents the same independence constraints is referred to as *independence equivalent* DAGs. So let D and D^* be independence equivalent DAGs, then

$$p(x|\Psi, D) = p(x|\Psi, D^*).$$

This means, that from observations alone we can not distinguish between different DAGs in an equivalence class. In the papers Heckerman et al. (1995) and Geiger and Heckerman (1994) it is for respectively the discrete and the Gaussian case shown, that when using the master prior procedure for the construction of parameter priors, the marginal likelihood for data is also the same for independence equivalent networks, *i.e.*

$$p(d|D) = p(d|D^*).$$

This equivalence is referred to as *likelihood equivalence*. Note that likelihood equivalence imply that if D and D^* are independence equivalent networks, then they have the same joint prior for the parameters, *i.e.*

$$p(\Psi|D) = p(\Psi|D^*).$$

7 Local Masters for Mixed Networks

In this section we will show how to specify prior distributions for the parameters in a CG network. In the mixed case, the marginal of a CG distribution is not always a CG distribution. In fact it is only a CG distribution if we marginalize over continuous variables or if we marginalize over a set B of discrete variable, where $(B \perp\!\!\!\perp \Gamma) \mid (\Delta \setminus B)$, see Frydenberg (1990). Consider the following example. We have a network of two variables, i and y , and the joint distribution is given by

$$p(i, y) = p(i)\mathcal{N}(m_i, \sigma_i^2).$$

Then the marginal distribution of y is given as a mixture of normal distributions

$$p(y) = \sum_{i \in \mathcal{I}} p(i)\mathcal{N}(m_i, \sigma_i^2),$$

so there is no simple way of using this directly for finding the local priors.

7.1 The Suggested Solution

The suggested solution is very similar to the solution for the pure cases. We start by specifying a prior Bayesian network (D, \mathcal{P}) and calculate the joint probability distribution

$$p(i, y|H) = p(i|\Psi)\mathcal{N}(m_i, \Sigma_i),$$

with $H = (\Psi, (m_i)_{i \in \mathcal{I}}, (\Sigma_i)_{i \in \mathcal{I}})$. So from the conditional parameters in the local distributions in the prior network, we calculate the parameters for the joint distribution. Then we translate this prior network into an imaginary database, with imaginary sample size n . From the probabilities in the discrete part of the network, we can, as in the pure discrete case, calculate α_i for all configurations of i . Now α_i represents how many times we have observed $I = i$ in the imaginary database. We can assume that each time we have observed the discrete variables I , we have observed the continuous variables Y and therefore set $\nu_i = \rho_i = \alpha_i$. Now for each configuration of i , we let m_i be the sample mean in the imaginary database, and Σ_i the sample variance. Further, as for the pure Gaussian case, we use $m_i = \mu_i$ and $\Phi_i = (\nu_i - 1)\Sigma_i$. However, for Φ_i to be positive, ν_i has to be larger than 1, for all configurations i and this has an impact on how small we can choose n to be, as $n = \sum_i \nu_i$. If the number of discrete variables is large, and/or the number of configurations of the discrete variables is large, then we might have to let n be larger than the value, that really reflects our confidence in the prior network. For these situations it might therefore be better to *e.g.* let $\Phi_i = \nu_i \Sigma_i$ as we then can choose the value of n any way we want. Or, we can just choose ν_i and ρ_i independently of n .

All the parameters needed to define the joint prior distributions for the parameters are now specified, so

$$\begin{aligned} p(\Psi) &= \mathcal{D}(\alpha), \\ p(M_i | \Sigma_i) &= \mathcal{N}(\mu_i, \frac{1}{\nu_i} \Sigma_i), \\ p(\Sigma_i) &= \mathcal{IW}(\rho_i, \Phi_i). \end{aligned}$$

But we can not use these distributions to derive priors for other networks, so instead we use the imaginary database to derive local master distributions.

Let, for each family $A = v \cup \text{pa}(v)$, the marginal CG distribution of X_a given H_A be given by

$$(X_A | H_A) \sim CG(\Psi_{i_{A \cap \Delta}}, m_{A \cap \Gamma | i_{A \cap \Delta}}, \Sigma_{A \cap \Gamma | i_{A \cap \Delta}}).$$

Then we suggest that the marginal prior distributions, also called the *local masters*, are found in the following way:

Let, for any variable z , $z_{i_{A \cap \Delta}} = \sum_{j:j_{A \cap \Delta}=i_{A \cap \Delta}} z_j$. Then

$$\begin{aligned} (\Psi_{A \cap \Delta}) &\sim \mathcal{D}(\alpha_{A \cap \Delta}), \\ (\Sigma_{A \cap \Gamma | i_{A \cap \Delta}}) &\sim \mathcal{IW}(\rho_{i_{A \cap \Delta}}, (\tilde{\Phi}_{A \cap \Gamma | i_{A \cap \Delta}}), \\ (m_{A \cap \Gamma | i_{A \cap \Delta}} | \Sigma_{A \cap \Gamma | i_{A \cap \Delta}}) &\sim \mathcal{N}(\bar{\mu}_{A \cap \Gamma | i_{A \cap \Delta}}, \frac{1}{\nu_{i_{A \cap \Delta}}} \Sigma_{A \cap \Gamma | i_{A \cap \Delta}}), \end{aligned}$$

where

$$\bar{\mu}_{i_{A \cap \Delta}} = \frac{(\sum_{j:j_{A \cap \Delta}=i_{A \cap \Delta}} \mu_j \nu_j)}{\nu_{A \cap \Delta}},$$

and

$$\tilde{\Phi}_{i_{A \cap \Delta}} = \Phi_{i_{A \cap \Delta}} + \sum_{j:j_{A \cap \Delta}=i_{A \cap \Delta}} \nu_j (\mu_j - \bar{\mu}_{i_{A \cap \Delta}}) (\mu_j - \bar{\mu}_{i_{A \cap \Delta}})^T.$$

The equations in the above result are well known from the analysis of variance theory, see *e.g.* Seber (1984). The marginal mean is found as a weighted average of the mean in every group, where a group here is given as a configuration of the discrete parents we marginalize over. The weights are the number of observations in each group. The marginal *ssd* is given as the within group variation plus the between group variation. Notice that with this method, it is possible to specify mixed networks, where the mean in the mixed part of the network depends on the discrete parents, but the variance does not.

From the local masters we can now, by conditioning as in the pure cases, derive the local priors needed to specify the prior parameter distribution for a CG network. So the only difference between the master procedure and the local master procedure is in the way the marginal distributions are found.

7.2 Properties of the Local Master Procedure

The local master procedure coincides with the master procedure in the pure cases. Further, the properties of the local master procedure in the mixed case, are the same as of the master prior procedure in the pure cases.

Parameter independence and parameter modularity follows immediately from the definition of the procedure. To show likelihood equivalence, we need the following result from Chickering (1995). Let D and D^* be two DAGs and let R_{D,D^*} be the set of edges by which D and D^* differ in directionality. Then, D and D^* are independence equivalent if and only if there exists a sequence of $|R_{D,D^*}|$ distinct arc reversals applied to D with the following properties:

- After each reversal, the resulting network structure is a DAG, *i.e.* it contains no directed cycles and it is independence equivalent to D^* .
- After all reversals, the resulting DAG is identical to D^* .
- If $w \rightarrow v$ is the next arc to be reversed in the current DAG, then w and v have the same parents in both DAGs, with the exception that w is also a parent of v in D .

Note that as we only reverse $|R_{D,D^*}|$ distinct arcs, we only reverse arcs in R_{D,D^*} . For mixed networks this means that we only reverse arcs between discrete variables or between continuous variables, as the only arcs that can differ in directionality are these. So we can use the above result for mixed networks.

From the above we see that we can show likelihood equivalence by showing that $p(d|D) = p(d|D^*)$ for two independence equivalent DAGs D and D^* that differ only by the direction of a single arc. As $p(x|H, D) = p(x|H, D^*)$ in CG networks, we can show likelihood equivalence by showing that $p(H|D) = p(H|D^*)$.

In the following let $v \rightarrow w$ in D and $w \rightarrow v$ in D^* . Further let ∇ be the set of common discrete and continuous parents for v and w . Of course, if v and w are discrete variables, then ∇ only contains discrete variables. The relation between $p(H|D)$ and $p(H|D^*)$ is given by:

$$\begin{aligned} \frac{p(H|D)}{p(H|D^*)} &= \frac{p(H_{v|w \cup \nabla}, D)p(H_{w|\nabla}, D)}{p(H_{w|v \cup \nabla}, D^*)p(H_{v|\nabla}, D^*)} \\ &= \frac{p(H_{v \cup w|\nabla}, D)}{p(H_{v \cup w|\nabla}, D^*)}. \end{aligned} \quad (11)$$

When using the local master procedure, the terms in (11) are equal. This is evident, as we find the conditional priors from distributions over families A , in this case $A = v \cup w \cup \nabla$, which is the same for both networks. Therefore likelihood equivalence follows.

8 Model Search

In the search for Bayesian networks with high network score, we can, in theory, calculate the network score for all possible DAGs and then choose the DAG or DAGs with the highest score.

In Robinson (1977), a recursive formula for the number of possible DAGs that contains n nodes, is found to be

$$f(n) = \sum_{i=1}^n (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} f(n-i),$$

where $\binom{n}{i}$ are the binomial coefficient. As we in mixed networks do not allow discrete nodes to have continuous parents, the number of possible mixed DAGs is given by

$$f(|\Delta|, |\Gamma|) = f(|\Delta|) \times f(|\Gamma|) \times 2^{|\Delta| \times |\Gamma|},$$

where $f(|\Delta|)$ and $f(|\Gamma|)$ are the numbers of DAGs for respectively the discrete and the continuous nodes, and $2^{|\Delta| \times |\Gamma|}$ denotes the number of different combinations of arrows from discrete to continuous nodes. If the number of random variables in the network is large, it is computationally infeasible to calculate the network score for all the possible DAGs. Therefore different methods for searching for DAGs with high network score have been tried, see *e.g.* Cooper and Herskovits (1992). In Section 8.3 we will describe one of these methods, namely greedy search with random restarts. This method, like many others, make use of Bayes factors as a way of comparing the network scores for two different DAGs. In the next section we will therefore consider Bayes factors for mixed networks.

8.1 Bayes Factors

A way to compare the network score for two different networks, D and D^* , is to calculate the *posterior odds*, given by

$$\frac{p(D|d)}{p(D^*|d)} = \frac{p(D, d)}{p(D^*, d)} = \frac{p(D)}{p(D^*)} \times \frac{p(d|D)}{p(d|D^*)},$$

where $p(D)/p(D^*)$ is the *prior odds* and $p(d|D)/p(d|D^*)$ is the *Bayes factor*.

The posterior odds is for numerical reasons often calculated using the logarithm,

$$\log \left(\frac{p(D|d)}{p(D^*|d)} \right) = \log(p(D|d)) - \log(p(D^*|d)).$$

For two models that differ only by a single arrow, the Bayes factor is, because of decomposability, especially simple. In this section, we will specify the Bayes

factor in the case where two DAGs differ by the direction of a single arrow and in the case where two DAGs differ by the presence of a single arrow.

First we look at the former case. As discrete nodes can not have continuous parents, we only look at reversing an arrow between two discrete variables or two continuous variables. In the following let $v \leftarrow w$ in D and $v \rightarrow w$ in D^* . Further let ∇_w be the parents of w in D and ∇_v the parents of v in D^* . As D and D^* only differ by the direction of the arrow between v and w , the parents of w in D^* are ∇_w and v and the parents of v in D are ∇_v and w . Notice that if v and w are discrete nodes, then the nodes in ∇_v and ∇_w can only be discrete, whereas if v and w are continuous nodes, they can be both discrete and continuous.

To simplify, we let the database consist of just one case, so $d = \{x\}$. As the likelihood terms are decomposable, the Bayes factor is given by

$$\begin{aligned} \frac{p(x|D)}{p(x|D^*)} &= \frac{p(v|\nabla_v, w, D)p(w|\nabla_w, D)}{p(w|\nabla_w, v, D^*)p(v|\nabla_v, D^*)} \\ &= \frac{\int p(x_v|x_{w \cup \nabla_v}, H_{v|w \cup \nabla_v}, D)p(H_{v|w \cup \nabla_v}|D)dH_{v|w \cup \nabla_v}}{\int p(x_w|x_{v \cup \nabla_w}, H_{w|v \cup \nabla_w}, D^*)p(H_{w|v \cup \nabla_w}|D^*)dH_{w|v \cup \nabla_w}} \\ &\times \frac{\int p(x_w|x_{\nabla_w}, H_{w|\nabla_w}, D)p(H_{w|\nabla_w}|D)dH_{w|\nabla_w}}{\int p(x_v|x_{\nabla_v}, H_{v|\nabla_v}, D^*)p(H_{v|\nabla_v}|D^*)dH_{v|\nabla_v}}. \end{aligned}$$

So to calculate the Bayes factor between D and D^* , we only need to consider the terms involving the conditional distributions of v and of w .

Notice that if $\nabla_v = \nabla_w$, then D and D^* are independence equivalent networks and the Bayes factor is equal to one.

Now let D and D^* be two different networks, that differ by a single arrow between the nodes v and w , with $v \leftarrow w$ in D and $v \nleftarrow w$ in D^* . Here v and w can be either both discrete variables, both continuous or v continuous and w discrete. Again, let ∇_v be the set of variables that are parents of v in D^* , so in D the parents of v are ∇_v and w . As the likelihood terms are decomposable, the Bayes factor is given by

$$\begin{aligned} \frac{p(x|D)}{p(x|D^*)} &= \frac{p(x_v|x_{w \cup \nabla_v}, D)}{p(x_v|x_{\nabla_v}, D^*)} \\ &= \frac{\int p(x_v|x_{w \cup \nabla_v}, H_{v|w \cup \nabla_v}, D)p(H_{v|w \cup \nabla_v}|D)dH_{v|w \cup \nabla_v}}{\int p(x_v|x_{\nabla_v}, H_{v|\nabla_v}, D^*)p(H_{v|\nabla_v}|D^*)dH_{v|\nabla_v}}. \end{aligned}$$

8.2 Equivalent Bayes Factors

To compare network scores for all networks which differ by only one arrow, is computationally inefficient. When using the local master procedure, we can reduce the number of comparisons needed.

Our goal is to identify classes of DAGs for which the corresponding Bayes factors for testing an arrow between the same two variables in the network, are the same. So let D_1 and D_1^* be two different networks that differ by a single arrow between the nodes v and w , with $v \leftarrow w$ in D_1 and $v \not\leftarrow w$ in D_1^* . Further, let ∇_{v_1} be the set of variables that are parents of v in both D_1 and D_1^* , *i.e.* in D_1 the parents of v are ∇_{v_1} and w and in D_1^* just ∇_{D_1} .

Further let D_2 and D_2^* be another two networks different from D_1 and D_1^* that differ by an arrow between v and w and let ∇_{v_2} be the set of variables that are parents of v in both D_2 and D_2^* . There are two situations to consider, namely when $v \leftarrow w$ in D_2 and when $v \rightarrow w$ in D_2 .

Consider first the former situation. The Bayes factor for testing D_1 against D_1^* was in the previous section found to be

$$\frac{p(x|D_1)}{p(x|D_1^*)} = \frac{\int p(x_v|x_{w \cup \nabla_{v_1}}, H_{v|w \cup \nabla_{v_1}}, D_1) p(H_{v|w \cup \nabla_{v_1}} | D_1) dH_{v|w \cup \nabla_{v_1}}}{\int p(x_v|x_{\nabla_{v_1}}, H_{v|\nabla_{v_1}}, D_1^*) p(H_{v|\nabla_{v_1}} | D_1^*) dH_{v|\nabla_{v_1}}}. \quad (12)$$

Likewise the Bayes factor for testing D_2 against D_2^* is

$$\frac{p(x|D_2)}{p(x|D_2^*)} = \frac{\int p(x_v|x_{w \cup \nabla_{v_2}}, H_{v|w \cup \nabla_{v_2}}, D_2) p(H_{v|w \cup \nabla_{v_2}} | D_2) dH_{v|w \cup \nabla_{v_2}}}{\int p(x_v|x_{\nabla_{v_2}}, H_{v|\nabla_{v_2}}, D_2^*) p(H_{v|\nabla_{v_2}} | D_2^*) dH_{v|\nabla_{v_2}}}.$$

As the local master procedure has the property of parameter modularity, then if $\nabla_{v_1} = \nabla_{v_2}$ it follows that

$$p(H_{v|w \cup \nabla_{v_1}} | D_1) = p(H_{v|w \cup \nabla_{v_2}} | D_2),$$

and

$$p(x_v|x_{w \cup \nabla_{v_1}}, H_{v|w \cup \nabla_{v_1}}, D_1) = p(x_v|x_{w \cup \nabla_{v_2}}, H_{v|w \cup \nabla_{v_2}}, D_2).$$

So the Bayes factor for testing the arrow from v to w is equivalent to testing this arrow in any other network, where v has the same parents as in D_1 , *i.e.* if $\nabla_{v_1} = \nabla_{v_2}$. This is illustrated in Figure 1.

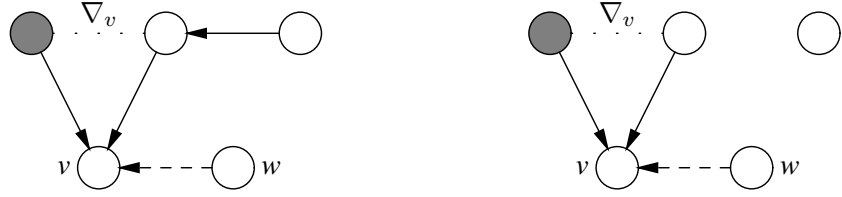


Figure 1: Equivalence due to parameter modularity.



Figure 2: Equivalence due to property of local master procedure.

Consider now the situation where $v \rightarrow w$ in D_2 . Let ∇_{w_2} be the set of variables, that are parents of w in both D_2 and D_2^* . The Bayes factor is given as

$$\begin{aligned} \frac{p(x|D_2)}{p(x|D_2^*)} &= \frac{p(x_w|x_{v \cup \nabla_{w_2}}, D_2)}{p(x_w|x_{\nabla_{w_2}}, D_2^*)} \\ &= \frac{\int p(x_w|x_{v \cup \nabla_{w_2}}, H_{w|v \cup \nabla_{w_2}}, D_2) p(H_{w|v \cup \nabla_{w_2}}|D_2) dH_{w|v \cup \nabla_{w_2}}}{\int p(x_w|x_{\nabla_{w_2}}, H_{w|\nabla_{w_2}}, D_2^*) p(H_{w|\nabla_{w_2}}|D_2^*) dH_{w|\nabla_{w_2}}}. \end{aligned}$$

Again we see that because of parameter modularity, this Bayes factor is the same as the Bayes factor given in (12), if $\nabla_{v_1} = \nabla_{w_2}$, *i.e.* if w in D_2 has the same parents as v does in D_1 , with the exception that v is a parent of w in D_2 . For an illustration, see Figure 2.

To show that these situations are the only ones where the Bayes factors always are the same, it is easy to find an example where $\nabla_{v_1} \neq \nabla_{w_2}$ and the Bayes factors are not same.

The above result is summarized in the following theorem.

Theorem 8.1

The Bayes factor for testing the arrow $v \leftarrow w$ in a DAG D_1 is equivalent to the Bayes factor for testing the same arrow in any other network D_2 if and only if the following two criteria are met:

- (1) $v \leftarrow w$ and v in D_2 has the same parents as in D_1 .

- (2) $v \rightarrow w$ and w in D_2 has the same parents as v does in D_1 , with the exception that v is a parent of w in D_2 .

Although using the two criteria reduces the number of comparisons, there will still, for large networks, be too many comparisons needed for finding the most likely DAG. Therefore it is still necessary to use some kind of search strategy.

8.3 Greedy search with random restarts

As mentioned earlier, many search strategies use Bayes factors as a way to compare the network score for two different networks. In the following we will describe one such strategy called *greedy search*.

Greedy search is initialized by choosing a network D from which to start the search. Let Δe be the posterior odds between two networks that differ by an arrow. Calculate then Δe for all DAGs D^* that differ from D by a single arrow e , either added, removed or reversed. Make the change e for which Δe is a minimum, that is where $p(D^*|d)$ is a maximum and continue the search from this new network. The search is terminated when there is no e with Δe smaller than 1. As shown in the previous section, the posterior odds is because of decomposability especially simple, as D and D^* only differ by one arrow. Further, it is possible to reduce the time complexity by using the equivalence criteria developed in Section 8.2.

As this search is local in the sense that it only evaluates local changes to the network, there is a chance that the found maximum is only a local maximum. A way to overcome this problem is to randomly perturb the structure of the start network D and restart the greedy search from this new network. This can be repeated a manageable number of times and between the networks found by the search strategy, the network with the highest score is chosen.

8.4 Priors on DAGs

In this section we will consider how to assign prior probabilities to the possible DAGs in a given problem. As shown in various papers, there are different ways of doing this. The Bayesian way would be to assess the prior belief in each DAG, but as the number of different DAGs grow, this is not manageable. Instead automated methods is being used.

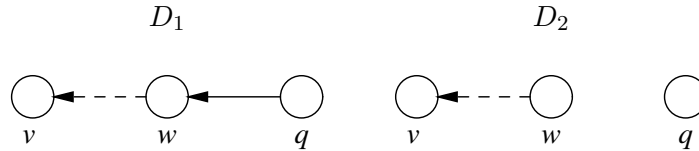


Figure 3: Models for which the Bayes factors are equivalent.

An often used approach is to assume that all DAGs are equally likely, thus letting the prior probability distribution over DAGs be uniform. This approach is mostly used only for simplicity and can be refined in various ways. For example, if we know that some of the DAGs are not possible, then we can assign probability zero to these and equal probabilities to the rest. Because of likelihood equivalence, DAGs within the same equivalence class will, with this approach, be assigned the same network score.

One argument against letting the prior over DAGs be uniform is that the number of different DAGs in an equivalence class varies between equivalence classes. This means that the conditional independencies represented in an equivalence class with many DAGs, a priori are more probable than those represented in an equivalence class with fewer DAGs. When using model averaging, this is a problem because it involves a sum over all the different DAGs. The conditional independencies represented by a large equivalence class, therefore influence the result more than those represented by a small equivalence class. A way to handle this problem is to either include only one DAG from each equivalence class or instead let all equivalence classes be equally likely and assign to each DAG a prior probability inversely proportional to the number of DAGs in the equivalence class it belongs to.

This last approach has, however, an affect on the posterior odds. Consider the following example, illustrated in Figure 3.

According to criteria one in Theorem 8.1, the Bayes factor for testing the presence of the arrow $v \leftarrow w$ in D_1 is equivalent to testing $v \leftarrow w$ in D_2 , *i.e.*

$$\frac{p(v|w, D_1)}{p(v|D_1^*)} = \frac{p(v|w, D_2)}{p(v|D_2^*)}.$$

If we assign equal priors to all DAGs, the posterior odds are the same as the Bayes factors and they will therefore also be equivalent in the above example. However, if we let all equivalence classes be equally likely and assign to each DAG a prior probability inversely proportional to the number of DAGs in the

equivalence class it belongs to, the posterior odds are no longer the same as the Bayes factors. In the above example, the number of DAGs in the equivalence classes for D_1 , D_1^* , D_2 and D_2^* are respectively 3, 2, 2 and 1. So the prior odds are not equivalent, *i.e.*

$$\frac{p(D_1)}{p(D_1^*)} = \frac{2}{3} \neq \frac{1}{2} = \frac{p(D_2)}{p(D_2^*)},$$

and therefore the posterior odds are not equivalent either. So this approach should not be used if we in a search strategy want to utilize that some of the Bayes factors are equivalent.

9 Example

In the following, some of the methods derived are illustrated by a simple example. This example was constructed by Morrison (1976) and also studied in Edwards (1995).

9.1 The Dataset

The dataset is from a hypothetical drug trial, where the weight losses of male and female rats under three different drug treatments have been measured after one and two weeks. Thus we have the discrete variables I_{sex} and I_{drug} with states

$$\begin{aligned} I_{sex} &= \{\text{male} = 1, \text{female} = 2\} \\ I_{drug} &= \{1, 2, 3\}, \end{aligned}$$

and the continuous variables Y_{w1} and Y_{w2} which respectively represents the weight losses after one and two weeks. For every drug, four rats of each sex have been treated, which gives a total of 24 observations. The observations are shown in Table 1.

9.2 Specifying the Prior Network

We start by specifying a prior Bayesian network (D, \mathcal{P}) . To simplify the specification of the joint parameter prior, we choose to let all the variables be inde-

<i>sex</i>	<i>drug</i>	<i>w1</i>	<i>w2</i>	<i>sex</i>	<i>drug</i>	<i>w1</i>	<i>w2</i>
1	1	5	6	2	1	7	10
1	1	7	6	2	1	8	10
1	1	9	9	2	1	6	6
1	1	5	4	2	1	9	7
1	2	9	12	2	2	7	6
1	2	7	7	2	2	10	13
1	2	7	6	2	2	6	9
1	2	6	8	2	2	8	7
1	3	14	11	2	3	14	9
1	3	21	15	2	3	14	8
1	3	12	10	2	3	16	12
1	3	17	12	2	3	10	5

Table 1: Observations of weight loss of male and female rats under three different drug treatments.

pendent, so the local probability distribution for each node only depends on the node itself, and we can specify them as follows.

For each discrete variable, we let each state be equally likely, so

$$p(i_{sex} = 1) = p(i_{sex} = 2) = \frac{1}{2}$$

and

$$p(i_{drug} = 1) = p(i_{drug} = 2) = p(i_{drug} = 3) = \frac{1}{3}.$$

This in fact is true by design.

For the continuous variables we use the sample mean and the sample variance as an initial estimate of the mean and the variance. Using this approach, the position and scale of the parameters are determined. We find that

$$p(y_{w1}) = \mathcal{N}(9.6, 17.1)$$

and

$$p(y_{w2}) = \mathcal{N}(8.7, 7.6).$$

So jointly

$$p(i, y) = p(i)\mathcal{N}(m_i, \Sigma_i),$$

with

$$p(i) = \frac{1}{6}, \quad m_i = \begin{pmatrix} 9.6 \\ 8.7 \end{pmatrix} \quad \text{and} \quad \Sigma_i = \begin{pmatrix} 17.1 & 0 \\ 0 & 7.6 \end{pmatrix},$$

for all possible configurations of i .

Be aware that in this way the dataset is used twice, namely both to initially specify the local probability distributions and later to find the posterior parameter distributions. This could result in parameter values that are overfitted to data.

9.3 Specifying Parameter Priors

In order to specify parameter priors for all possible networks, we use the local master procedure.

First we translate the prior network into an imaginary database. The parameters needed to represent this imaginary database are n , α_i , ν_i , ρ_i , μ_i and Φ_i .

Here we let $\Phi_i = (\nu_i - 1)\Sigma_i$, so ν_i must be larger than 1. This means in this example that n must be larger than 6. We choose $n = 12$ and find that

$$\alpha_i = \nu_i = \rho_i = p(i)n = \frac{1}{6}12 = 2.$$

Further

$$\mu_i = m_i = \begin{pmatrix} 9.6 \\ 8.7 \end{pmatrix} \quad \text{and} \quad \Phi_i = (\nu_i - 1)\Sigma_i = \begin{pmatrix} 17.0 & 0 \\ 0 & 7.6 \end{pmatrix},$$

for all configurations of i .

We can now specify parameter priors for all possible networks. As an illustration, consider the parameter prior for the network in Figure 4.

We need to find the local masters for the following four families

$$\begin{aligned} A_1 &= \{sex\}, \\ A_2 &= \{drug\}, \\ A_3 &= \{w1\}, \\ A_4 &= \{sex, w1, w2\}. \end{aligned}$$

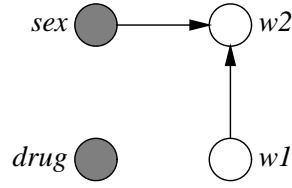


Figure 4: The DAG in the example for specification of local parameter priors.

As the variables in A_1 , A_2 and A_3 do not have any parents, the local masters for these families are also the local parameter priors. Thus the local parameter prior for I_{sex} is given by

$$\Psi_{sex} \sim \mathcal{D}(\alpha_{sex}),$$

with

$$\alpha_{i_{sex}=1} = \sum_{j:j_{sex}=1} \alpha_j = 6 \quad \text{and} \quad \alpha_{i_{sex}=2} = \sum_{j:j_{sex}=2} \alpha_j = 6.$$

Similarly the local parameter prior for I_{drug} is

$$\Psi_{drug} \sim \mathcal{D}(\alpha_{drug}),$$

with

$$\alpha_{i_{drug}=1} = \alpha_{i_{drug}=2} = \alpha_{i_{drug}=3} = 4.$$

For Y_{w1} we find the local parameter prior to be

$$\begin{aligned} \Sigma_{w1} &\sim \mathcal{IW}(\rho, \tilde{\Phi}_{w1}), \\ m_{w1} | \Sigma_{w1} &\sim \mathcal{N}(\bar{\mu}_{w1}, \frac{1}{\nu} \Sigma_{w1}), \end{aligned}$$

with

$$\rho = \sum_j \rho_j = 12 \quad \text{and} \quad \nu = \sum_j \nu_j = 12,$$

and

$$\begin{aligned} \bar{\mu} &= \frac{\sum_i \mu_i \nu_i}{\nu} = \begin{pmatrix} 9.6 \\ 8.7 \end{pmatrix}, \\ \tilde{\Phi} &= \sum_i \Phi_i + \sum_i \nu_i (\mu_i - \bar{\mu})(\mu_i - \bar{\mu})^T = \begin{pmatrix} 102.6 & 0 \\ 0 & 45.6 \end{pmatrix}, \end{aligned}$$

so

$$\bar{\mu}_{w1} = 9.6 \quad \text{and} \quad \tilde{\Phi}_{w1} = 102.6.$$

The local master for the family A_4 is given as

$$\begin{aligned} (\Sigma_{i_{sex}}) &\sim IW(\rho_{i_{sex}}, (\tilde{\Phi}_{i_{sex}})), \\ (m_{i_{sex}}) | (\Sigma_{i_{sex}}) &\sim \mathcal{N}(\bar{\mu}_{i_{sex}}, \frac{1}{\nu_{i_{sex}}} (\Sigma_{i_{sex}})), \end{aligned}$$

with

$$\rho_{i_{sex}=1} = \sum_{j:j_{sex}=1} \rho_j = 6 \quad \text{and} \quad \rho_{i_{sex}=2} = \sum_{j:j_{sex}=2} \rho_j = 6.$$

Likewise for $\nu_{i_{sex}}$. Further

$$\bar{\mu}_{i_{sex}=1} = \frac{\sum_{j:j_{sex}=1} \mu_j \nu_j}{\nu_{i_{sex}=1}} = \begin{pmatrix} 9.6 \\ 8.7 \end{pmatrix}$$

and

$$\begin{aligned} \tilde{\Phi}_{i_{sex}=1} &= \sum_{j:j_{sex}=1} \Phi_j + \sum_{j:j_{sex}=1} \nu_j (\mu_j - \bar{\mu}_{i_{sex}=1})(\mu_j - \bar{\mu}_{i_{sex}=1})^T \\ &= \begin{pmatrix} 51.3 & 0 \\ 0 & 22.8 \end{pmatrix} \end{aligned}$$

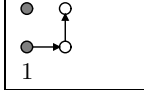
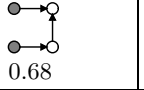
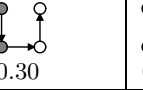
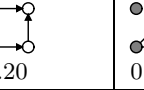
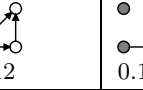

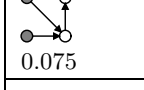
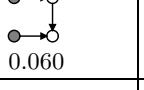
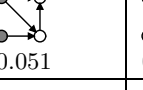
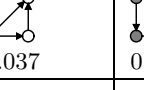
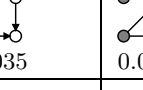

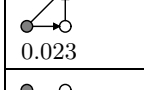
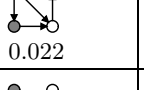
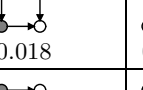
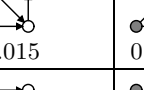
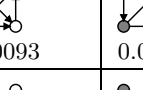
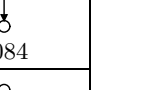
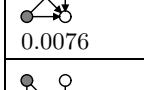
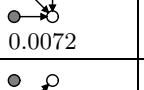
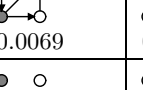
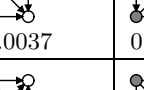
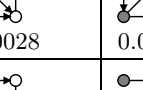
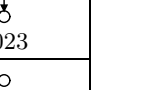
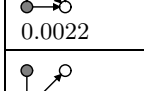
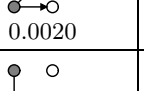
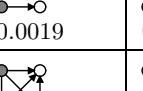
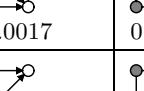
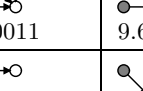
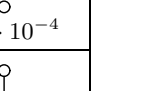
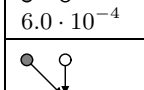
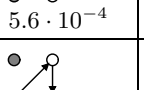
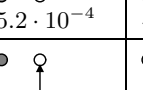
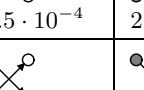
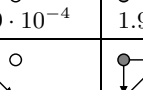
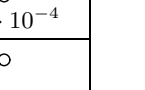
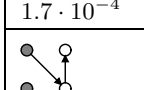
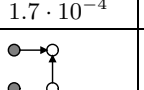
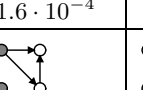
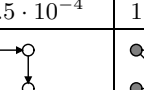
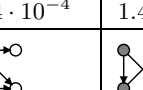
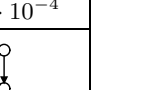
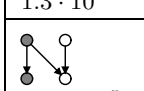
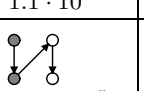
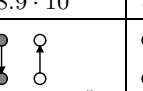
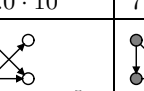
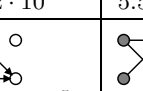
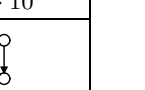
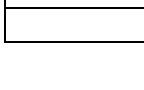
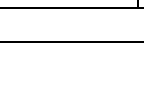
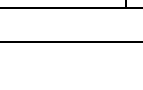
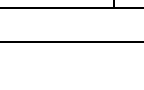
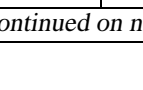
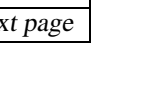
and the same for $i_{sex} = 2$.

The local parameter prior for Y_{w2} given Y_{w1} and I_{sex} can now be found by conditioning in this local master distribution.

We have now specified the parameters needed to calculate the likelihood of a DAG, $p(d|D)$. To calculate the network score of D , we also need to specify the prior probability of D . In this example we just choose to let all DAGs be equally likely and thus use the likelihood $p(d|D)$ as the network score.

9.4 Result

Using the formula on page 35, we find that for a network with two discrete and two continuous nodes, there are 144 possible DAGs. So in this example, there are no computational problems in calculating the network score for all these DAGs. Further, if we only calculate the score for DAGs that are not independence equivalent, the number of different DAGs are reduced to 88.

Prior network		● ○ ● ○	Imaginary sample size			12
 1	 0.68	 0.30	 0.20	 0.12	 0.12	
 0.075	 0.060	 0.051	 0.037	 0.035	 0.028	
 0.023	 0.022	 0.018	 0.015	 0.0093	 0.0084	
 0.0076	 0.0072	 0.0069	 0.0037	 0.0028	 0.0023	
 0.0022	 0.0020	 0.0019	 0.0017	 0.0011	 $9.6 \cdot 10^{-4}$	
 $6.0 \cdot 10^{-4}$	 $5.6 \cdot 10^{-4}$	 $5.2 \cdot 10^{-4}$	 $4.5 \cdot 10^{-4}$	 $2.9 \cdot 10^{-4}$	 $1.9 \cdot 10^{-4}$	
 $1.7 \cdot 10^{-4}$	 $1.7 \cdot 10^{-4}$	 $1.6 \cdot 10^{-4}$	 $1.5 \cdot 10^{-4}$	 $1.4 \cdot 10^{-4}$	 $1.4 \cdot 10^{-4}$	
 $1.3 \cdot 10^{-4}$	 $1.1 \cdot 10^{-4}$	 $8.9 \cdot 10^{-5}$	 $8.0 \cdot 10^{-5}$	 $7.2 \cdot 10^{-5}$	 $5.5 \cdot 10^{-5}$	
 $5.2 \cdot 10^{-5}$	 $5.0 \cdot 10^{-5}$	 $4.7 \cdot 10^{-5}$	 $4.5 \cdot 10^{-5}$	 $4.2 \cdot 10^{-5}$	 $4.2 \cdot 10^{-5}$	

continued on next page



































continued from previous page					
Prior network	● ○	● ○	Imaginary sample size		12
 $3.9 \cdot 10^{-5}$	 $3.8 \cdot 10^{-5}$	 $3.4 \cdot 10^{-5}$	 $3.2 \cdot 10^{-5}$	 $2.7 \cdot 10^{-5}$	 $2.4 \cdot 10^{-5}$
 $2.2 \cdot 10^{-5}$	 $2.0 \cdot 10^{-5}$	 $1.6 \cdot 10^{-5}$	 $1.3 \cdot 10^{-5}$	 $1.1 \cdot 10^{-5}$	 $1.0 \cdot 10^{-5}$
 $5.9 \cdot 10^{-6}$	 $4.9 \cdot 10^{-6}$	 $3.6 \cdot 10^{-6}$	 $3.0 \cdot 10^{-6}$	 $1.1 \cdot 10^{-6}$	 $9.0 \cdot 10^{-7}$
 $3.2 \cdot 10^{-7}$	 $3.0 \cdot 10^{-7}$	 $2.6 \cdot 10^{-7}$	 $2.5 \cdot 10^{-7}$	 $1.5 \cdot 10^{-7}$	 $1.3 \cdot 10^{-7}$
 $9.4 \cdot 10^{-8}$	 $8.9 \cdot 10^{-8}$	 $7.8 \cdot 10^{-8}$	 $7.4 \cdot 10^{-8}$	 $7.2 \cdot 10^{-8}$	 $5.9 \cdot 10^{-8}$
 $4.5 \cdot 10^{-8}$	 $3.8 \cdot 10^{-8}$	 $2.1 \cdot 10^{-8}$	 $1.8 \cdot 10^{-8}$		

Table 2: The DAGs in the reduced search space, listed in decreasing order of probability. The number below each DAG is the Bayes factor between the given DAG and the DAG with the highest network score.

In Table 2 the result of the learning procedure is given. The DAGs are listed in decreasing order of probability, and the number below each DAG is the posterior odds between the given DAG and the DAG with the highest network score. This number expresses the relative probability of a DAG, that is, relative to the DAG with the highest network score. As we have chosen a uniform prior over DAGs, the posterior odds is in this example equal to the Bayes factor.

Before analyzing the result, we can discard some of the networks in Table 2. By design, the discrete variables *sex* and *drug* are independent, so there should not be an arrow between *sex* and *drug*. Further, there is a time restriction between *w1* and *w2*, as *w1* is observed before *w2*. So if *w1* and *w2* are dependent, the arrow between *w1* and *w2* must go from *w1* to *w2*. Taking these restrictions into account, we only consider the 32 different DAGs listed in Table 3.

In the most probable DAG, we see that *w2* depends on *w1* and *w1* depends on

































Prior network		● ○	Imaginary sample size			12
 1	 0.68	 0.12	 0.075	 0.051	 0.023	
 0.0093	 0.0020	 0.0019	 0.0017	 $9.6 \cdot 10^{-4}$	 $4.5 \cdot 10^{-4}$	
 $1.6 \cdot 10^{-4}$	 $1.5 \cdot 10^{-4}$	 $1.4 \cdot 10^{-4}$	 $1.3 \cdot 10^{-4}$	 $1.1 \cdot 10^{-4}$	 $8.9 \cdot 10^{-5}$	
 $7.2 \cdot 10^{-5}$	 $3.4 \cdot 10^{-5}$	 $2.0 \cdot 10^{-5}$	 $1.6 \cdot 10^{-5}$	 $3.6 \cdot 10^{-6}$	 $3.0 \cdot 10^{-6}$	
 $3.2 \cdot 10^{-7}$	 $3.0 \cdot 10^{-7}$	 $2.6 \cdot 10^{-7}$	 $2.5 \cdot 10^{-7}$	 $1.5 \cdot 10^{-7}$	 $1.3 \cdot 10^{-7}$	
 $7.2 \cdot 10^{-8}$	 $5.9 \cdot 10^{-8}$					

Table 3: The DAGs in the reduced search space, listed in decreasing order of probability. The number below each DAG is the Bayes factor between the given DAG and the DAG with the highest network score.

drug. Further *w2* and *drug* are conditionally independent given *w1* and both *w1* and *w2* are independent on *sex*.

Almost the same dependency structure is seen in the second and third best DAG, except that here *w2* also depends on respectively *sex* and *drug*.

Generally we see that in the first 12 DAGs, *w1* depends on *drug*. The first DAG that does not show this dependency relation is only 0.00016 times as probable as the best DAG. Likewise we see that in the first 7 DAGs, *w2* depends on *w1* and the first DAG that does not contain this dependency relation is only 0.0020 as probable as the best DAG. Therefore we should not consider any model that does not include these dependencies.

It is not clear which independencies should be included in the model, except for those introduced when we reduced the search space. The second DAG is

for example 0.68 times as probable as the first DAG, and the third to the sixth DAG is between 0.12 and 0.023 as probable as the best DAG. This suggests that there is some unexplained variation not accounted for in the best DAG and it might therefore be more accurate to select *e.g.* the first six models and use model averaging.

In Edwards (1995) the dataset is analyzed using undirected graphical models. He uses the software MIM for maximum likelihood estimation and likelihood ratio test. The result is displayed in Figure 5 and we see that it is not in conflict with our result.

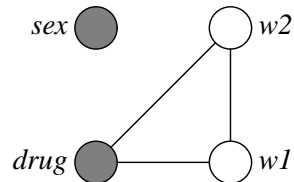


Figure 5: Previous result.

9.5 Sensitivity to Prior Information

In this section we will explore how the size of the imaginary database and the choice of the prior network influences the result. The findings agree with findings for a purely discrete case described in Steck and Jaakkola (2002).

Recall that the prior network ideally expresses which dependency structure we believe there is between the variables in the network and the size of the imaginary database expresses how much confidence we have in this dependency structure.

In the previous section we used the empty network as the prior network and set the size n of the imaginary database to 12. This is less than the number of real observations in the example, which is 24. We will therefore also learn the networks using a larger value of n and to see the difference clearly, we use $n = 2000$. The result is given in Table 4.

If we look at the three best networks from the previous result, we see that the relative probabilities for these networks in this result, are between 0.94 and 0.97. They are no longer the most probable networks, but they are still very probable. Actually all the networks are very probable and the relative probability of the least probable network is as much as 0.78.

































Prior network		● ○	Imaginary sample size		2000
 1	 0.99	 0.97	 0.96	 0.96	 0.95
 0.94	 0.93	 0.89	 0.89	 0.89	 0.89
 0.88	 0.88	 0.88	 0.87	 0.87	 0.87
 0.86	 0.86	 0.85	 0.85	 0.84	 0.83
 0.79	 0.79	 0.79	 0.79	 0.78	 0.78
 0.78	 0.78				

Table 4: The revised result with the prior network and the imaginary sample size specified as in the first line of this table.

The reason for this is that the prior network is the empty network, which represents that all the variables are independent. This model is therefore a submodel of all other models. When n is large, we have much confidence in these independencies, so all networks will a priori be very probable. As the real database only contains few observations, we have not enough information to differentiate between these networks and all the networks are therefore almost equally likely.

We will now explore what happens if we change the prior network. First we will learn the structure using the most probable structure from Table 3 as the prior network. The results with $n = 12$ and $n = 2000$ are given in respectively Table 5 and Table 6.

For $n = 12$ we see almost the same result as when using the empty network. The best networks are, not surprisingly, the same, only the order between them are a little different. To some extent, this also applies for $n = 2000$.

Prior network		Imaginary sample size	12		
 1	 0.59	 0.38	 0.34	 0.17	 0.10
 0.064	 0.056	 $2.7 \cdot 10^{-4}$	 $1.3 \cdot 10^{-4}$	 $1.2 \cdot 10^{-4}$	 $4.8 \cdot 10^{-5}$
 $4.5 \cdot 10^{-5}$	 $2.2 \cdot 10^{-5}$	 $1.9 \cdot 10^{-5}$	 $7.9 \cdot 10^{-6}$	 $7.5 \cdot 10^{-8}$	 $5.0 \cdot 10^{-8}$
 $4.4 \cdot 10^{-8}$	 $2.9 \cdot 10^{-8}$	 $2.9 \cdot 10^{-8}$	 $2.6 \cdot 10^{-8}$	 $1.9 \cdot 10^{-8}$	 $1.7 \cdot 10^{-8}$
 $2.1 \cdot 10^{-11}$	 $1.4 \cdot 10^{-11}$	 $9.9 \cdot 10^{-12}$	 $8.9 \cdot 10^{-12}$	 $6.5 \cdot 10^{-12}$	 $5.8 \cdot 10^{-12}$
 $3.6 \cdot 10^{-12}$	 $2.4 \cdot 10^{-12}$				

Table 5: The revised result with the prior network and the imaginary sample size specified as in the first line of this table.

Further we see that for both $n = 12$ and $n = 2000$, the 32 networks categorize as follows. The 8 networks with both arrows $drug \rightarrow w1$ and $w1 \rightarrow w2$ are the 8 most probable networks. In the succeeding 8 networks we have $drug \rightarrow w1$ and $w1 \leftrightarrow w2$, after that the 8 networks with $drug \leftrightarrow w1$ and $w1 \rightarrow w2$. In the last 8 networks we have $drug \leftrightarrow w1$ and $w1 \leftrightarrow w2$. Also we see that within each category, the networks are almost equally likely, mostly pronounced for $n = 2000$. These findings are what we expected. The arrows included in the prior network are all represented in the most probable networks and these networks are all almost equally likely, as the prior network is a submodel of these. Further there is a large difference in relative score between the different categories, which shows that networks which include the arrows $drug \rightarrow w1$ and $w1 \rightarrow w2$, are much more likely than those that do not. As this is valid for both $n = 12$ and $n = 2000$, it is not only due to the influence of the prior network, but also because the dataset supports these dependencies.

We will now explore what happens if we choose the prior network to be the least


































Prior network			Imaginary sample size		2000
 1	 1	 0.99	 0.98	 0.98	 0.98
 0.97	 0.96	 $6.5 \cdot 10^{-4}$	 $6.4 \cdot 10^{-4}$	 $6.4 \cdot 10^{-4}$	 $6.3 \cdot 10^{-4}$
 $1.9 \cdot 10^{-4}$	 $1.8 \cdot 10^{-4}$	 $1.8 \cdot 10^{-4}$	 $1.8 \cdot 10^{-4}$	 $3.5 \cdot 10^{-9}$	 $3.5 \cdot 10^{-9}$
 $3.5 \cdot 10^{-9}$	 $3.5 \cdot 10^{-9}$	 $3.4 \cdot 10^{-9}$	 $3.4 \cdot 10^{-9}$	 $3.4 \cdot 10^{-9}$	 $3.4 \cdot 10^{-9}$
 $2.2 \cdot 10^{-12}$	 $2.2 \cdot 10^{-12}$	 $2.2 \cdot 10^{-12}$	 $2.2 \cdot 10^{-12}$	 $6.4 \cdot 10^{-13}$	 $6.4 \cdot 10^{-13}$
 $6.4 \cdot 10^{-13}$	 $6.4 \cdot 10^{-13}$				

Table 6: The revised result with the prior network and the imaginary sample size specified as in the first line of this table.

probable network from Table 3. The results are for $n = 12$ and $n = 2000$ given in respectively Table 7 and Table 8.

For $n = 12$ we see almost the same result as with the other prior networks. For $n = 2000$ we see that the 8 most probable models actually are the 8 models that are possible with both the arrows $sex \rightarrow w1$ and $sex \rightarrow w2$. Further we see that all networks are almost equally likely and there is not, as would be expected, a large difference in score between networks with both arrows and the others. Actually for both $n = 12$ and $n = 2000$ the result is very similar to the result with the empty network as the prior networks. The reason for this is that the probability distribution of the prior network is estimated from data, *i.e.* we use the sample mean and sample variance as the mean and variance in the prior network. If data does not support a dependence between sex and respectively $w1$ and $w2$, then this prior network will be almost the same as the empty prior network and so will the result of the learning procedure. However, it can be seen that even small differences from the empty prior network have an impact


































Prior network			Imaginary sample size			12
 1	 0.25	 0.13	 0.094	 0.023	 0.012	
 0.0079	 0.0020	 0.0018	 $9.6 \cdot 10^{-4}$	 $7.4 \cdot 10^{-4}$	 $3.9 \cdot 10^{-4}$	
 $1.8 \cdot 10^{-4}$	 $1.7 \cdot 10^{-4}$	 $1.6 \cdot 10^{-4}$	 $1.4 \cdot 10^{-4}$	 $9.0 \cdot 10^{-5}$	 $3.9 \cdot 10^{-5}$	
 $3.7 \cdot 10^{-5}$	 $3.5 \cdot 10^{-5}$	 $2.0 \cdot 10^{-5}$	 $1.8 \cdot 10^{-5}$	 $1.2 \cdot 10^{-6}$	 $1.1 \cdot 10^{-6}$	
 $3.1 \cdot 10^{-7}$	 $2.9 \cdot 10^{-7}$	 $2.8 \cdot 10^{-7}$	 $2.6 \cdot 10^{-7}$	 $1.5 \cdot 10^{-7}$	 $1.4 \cdot 10^{-7}$	
 $6.2 \cdot 10^{-8}$	 $5.5 \cdot 10^{-8}$					

Table 7: The revised result with the prior network and the imaginary sample size specified as in the first line of this table.

when n is large, as the 8 most probable networks actually are the ones with both $sex \rightarrow w1$ and $sex \rightarrow w2$.


































Prior network			Imaginary sample size		2000
 1	 0.99	 0.94	 0.94	 0.91	 0.90
 0.86	 0.86	 0.67	 0.65	 0.61	 0.59
 0.59	 0.59	 0.54	 0.53	 0.38	 0.37
 0.35	 0.35	 0.34	 0.33	 0.32	 0.31
 0.25	 0.25	 0.22	 0.22	 0.22	 0.22
 0.20	 0.20				

Table 8: The revised result with the prior network and the imaginary sample size specified as in the first line of this table.

Acknowledgements

This research was supported by the ESPRIT project P29105 (BaKE). Also I would like to thank my supervisor Steffen L. Lauritzen for his dedicated help and support.

References

- Bernardo, J. M. and Smith, A. F. M. (1994). *Bayesian Theory*, John Wiley & Sons, Chichester.
- Bøttcher, S. G. (2001). Learning Bayesian Networks with Mixed Variables, *Artificial Intelligence and Statistics 2001*, Morgan Kaufmann, San Francisco, CA, USA, pp. 149–156.
- Chickering, D. M. (1995). A transformational characterization of equivalent Bayesian-network structures, *Proceedings of Eleventh Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, CA, USA, pp. 87–98.
- Cooper, G. and Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data, *Machine Learning* **9**: 309–347.
- Dawid, A. P. and Lauritzen, S. L. (1993). Hyper Markov laws in the statistical analysis of decomposable graphical models, *The Annals of Statistics* **21**(3): 1272–1317.
- DeGroot, M. H. (1970). *Optimal Statistical Decisions*, McGraw-Hill, New York.
- Edwards, D. (1995). *Introduction to Graphical Modelling*, Springer-Verlag, New York.
- Frydenberg, M. (1990). Marginalization and collapsibility in graphical interaction models, *Annals of Statistics* **18**: 790–805.
- Geiger, D. and Heckerman, D. (1994). Learning Gaussian Networks, *Proceedings of Tenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, CA, USA, pp. 235–243.
- Heckerman, D., Geiger, D. and Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data, *Machine Learning* **20**: 197–243.
- Lauritzen, S. L. (1992). Propagation of probabilities, means and variances in mixed graphical association models, *Journal of the American Statistical Association* **87**(420): 1098–1108.
- Lauritzen, S. L. (1996). *Graphical Models*, Clarendon press, Oxford, New York.
- Lauritzen, S. L. and Jensen, F. (2001). Stable local computation with conditional Gaussian distributions, *Statistics and Computing* **11**: 191–203.
- Morrison, D. F. (1976). *Multivariate Statistical Methods*, McGraw-Hill, USA.
- Robinson, R. W. (1977). Counting unlabeled acyclic digraphs, *Lecture Notes in Mathematics*, 622: *Combinatorial Mathematics V* pp. 239–273.

-
- Seber, G. A. F. (1984). *Multivariate Observations*, John Wiley and Sons, New York.
- Shachter, R. D. and Kenley, C. R. (1989). Gaussian influence diagrams, *Management Science* **35**: 527–550.
- Spiegelhalter, D. J. and Lauritzen, S. L. (1990). Sequential updating of conditional probabilities on directed graphical structures, *Networks* **20**: 579–605.
- Steck, H. and Jaakkola, T. S. (2002). On the Dirichlet Prior and Bayesian Regularization, *Conference on Advances in Neural Information Processing Systems*, Vol. 15, MIT Press, Cambridge, USA.

Paper II
deal: A Package for Learning
Bayesian Networks

Susanne G. Bøttcher and Claus Dethlefsen

Published in:
Journal of Statistical Software (2003),
8(20):1-40.

deal: A Package for Learning Bayesian Networks

Susanne G. Bøttcher

Aalborg University, Denmark

Claus Dethlefsen

Aalborg University, Denmark

Abstract.

`deal` is a software package for use with R. It includes several methods for analyzing data using Bayesian networks with variables of discrete and/or continuous types but restricted to conditionally Gaussian networks. Construction of priors for network parameters is supported and their parameters can be learned from data using conjugate updating. The network score is used as a metric to learn the structure of the network and forms the basis of a heuristic search strategy. `deal` has an interface to Hugin.

1 Introduction

A Bayesian network is a graphical model that encodes the joint probability distribution for a set of random variables. Bayesian networks are treated in *e.g.* Cowell et al. (1999) and have found application within many fields, see Lauritzen (2003) for a recent overview.

Here we consider Bayesian networks with mixed variables, *i.e.* the random variables in a network can be of both discrete and continuous types. A method for learning the parameters and structure of such Bayesian networks has recently been described by Bøttcher (2001). We have developed a package called `deal`, written in R (R Development Core Team 2003), which provides these methods for learning Bayesian networks. In particular, the package includes procedures for defining priors, estimating parameters, calculating network scores, performing heuristic search as well as simulating data sets with a given dependency structure. Figure 1 gives an overview of the functionality in `deal`. The package can be downloaded from the Comprehensive R Archive Network (CRAN) <http://cran.R-project.org/> and may be used under the terms of the

GNU General Public License Version 2.

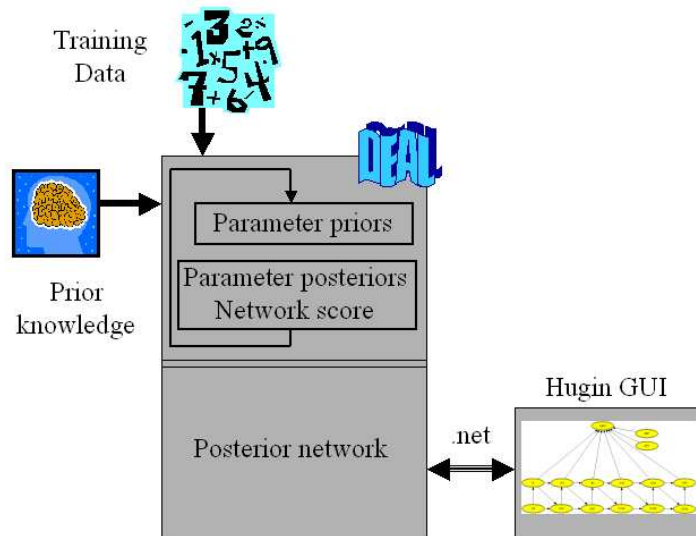


Figure 1: From prior knowledge and training data, a posterior network is produced by `deal`. The network may be transferred to Hugin for further inference.

In Section 2 we define Bayesian networks for mixed variables. To learn a Bayesian network, the user needs to supply a training data set and represent any prior knowledge available as a Bayesian network. Section 3 shows how to specify the training data set in `deal` and Section 4 discusses how to specify a Bayesian network in terms of a Directed Acyclic Graph (DAG) and the local probability distributions.

`deal` uses the prior Bayesian network to deduce prior distributions for all parameters in the model. Then, this is combined with the training data to yield posterior distributions of the parameters. The parameter learning procedure is treated in Section 5.

Section 6 describes how to learn the structure of the network. A network score is calculated and a search strategy is employed to find the network with the highest score. This network gives the best representation of data and we call it the *posterior network*.

Section 7 describes how to transfer the posterior network to Hugin (<http://www.hugin.com>). The Hugin graphical user interface (GUI) can then be

used for further inference in the posterior network.

In the appendix we provide manual pages for the main functions in `deal`.

2 Bayesian Networks

Let $D = (V, E)$ be a Directed Acyclic Graph (DAG), where V is a finite set of nodes and E is a finite set of directed edges (arrows) between the nodes. The DAG defines the structure of the Bayesian network.

To each node $v \in V$ in the graph corresponds a random variable X_v . The set of variables associated with the graph D is then $X = (X_v)_{v \in V}$. Often, we do not distinguish between a variable X_v and the corresponding node v . To each node v with parents $\text{pa}(v)$ a local probability distribution, $p(x_v | x_{\text{pa}(v)})$, is attached. The set of local probability distributions for all variables in the network is \mathcal{P} .

A Bayesian network for a set of random variables X is the pair (D, \mathcal{P}) .

The possible lack of directed edges in D encodes conditional independencies between the random variables X through the factorization of the joint probability distribution,

$$p(x) = \prod_{v \in V} p(x_v | x_{\text{pa}(v)}).$$

Here, we allow Bayesian networks with both discrete and continuous variables, as treated in Lauritzen (1992), so the set of nodes V is given by $V = \Delta \cup \Gamma$, where Δ and Γ are the sets of discrete and continuous nodes, respectively. The set of variables X can then be denoted $X = (X_v)_{v \in V} = (I, Y) = ((I_\delta)_{\delta \in \Delta}, (Y_\gamma)_{\gamma \in \Gamma})$, where I and Y are the sets of discrete and continuous variables, respectively. For a discrete variable, δ , we let \mathcal{I}_δ denote the set of levels.

To ensure *e.g.* availability of exact local computation methods, we do not allow discrete variables to have continuous parents. The joint probability distribution then factorizes into a discrete part and a mixed part, so

$$p(x) = p(i, y) = \prod_{\delta \in \Delta} p(i_\delta | i_{\text{pa}(\delta)}) \prod_{\gamma \in \Gamma} p(y_\gamma | i_{\text{pa}(\gamma)}, y_{\text{pa}(\gamma)}).$$

3 Data Structure

`deal` expects data as specified in a data frame which is a standard data structure in R. For example, standard ASCII data files with one column per variable and one line per observation can be read using `read.table()` which returns a data frame.

The rats example in Table 1 was constructed by Morrison (1976) and also studied in Edwards (1995). The data set is from a hypothetical drug trial, where the weight losses of male and female rats under three different drug treatments have been measured after one and two weeks.

Sex	Drug	W1	W2
M	D1	5	6
M	D1	7	6
M	D1	9	9
M	D1	5	4
M	D2	9	12
M	D2	7	7
M	D2	7	6
M	D2	6	8
M	D3	14	11
M	D3	21	15
M	D3	12	10
M	D3	17	12
F	D1	7	10
F	D1	8	10
F	D1	6	6
F	D1	9	7
F	D2	7	6
F	D2	10	13
F	D2	6	9
F	D2	8	7
F	D3	14	9
F	D3	14	8
F	D3	16	12
F	D3	10	5

Table 1: An example data file, *rats.dat*.

The data are loaded into a data frame `rats.df` by the following command

```
rats.df <- read.table("rats.dat",header=TRUE)
```

Before continuing, it is essential that the column variables have the correct types. Discrete variables should be specified as *factors* and continuous variables as *numeric*. To alter the type of a variable so that it is regarded as a discrete variable, use the `factor()` function (standard in R). In the rats example, `Sex` and

Drug are interpreted to be factors by `read.table`, and thus no changes are necessary.

We assume that we have observed complete data which means that no NA's are present in the data frame.

4 Specification of a Bayesian Network

As described in Section 2, a Bayesian network is specified by a DAG and a set of local probability distributions. In this section we will show how to specify these terms in `deal`.

4.1 The Network Class and Associated Methods

In `deal`, a Bayesian network is represented as an object of class `network`. The network object is a list of properties that are added or changed by the methods described in following sections.

A network is generated by the following command

```
rats <- network(rats.df)
```

and by default it is set to the empty network (the network without any arrows).

If the option `specifygraph` is set, a point and click graphical interface allows the user to insert and delete arrows until the requested DAG is obtained.

```
rats <- network(rats.df,specifygraph=TRUE)
```

A plot of the network (see Figure 2) is generated by

```
plot(rats)
```

Note that discrete nodes are grey and continuous nodes are white.

The primary property of a network is the list of nodes, in the example the list is: `nodes(rats)`. Each entry in the list is an object of class `node` representing a

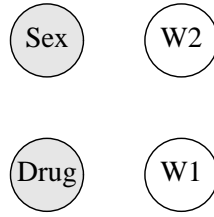


Figure 2: Graphical representation of the rats network.

node in the graph, which includes information associated with the node. Several methods for the network class operate by applying an appropriate method for one or more nodes in the list of nodes. The nodes appear in the node list in the same order as in the data frame used to create the network object.

It is possible to access the individual nodes in a network by referring either to their index (the column number in the data frame) or to their name:

```
rats.nd <- nodes(rats)# the list of nodes
rats.nd[[1]]          # the first node
rats.nd$Drug          # the node ``Drug``
```

A collection of networks is represented in an object of class `networkfamily`, which has associated `print()` and `plot()` functions.

4.2 Specification of the Probability Distributions

The joint distribution of the random variables in a network in `deal` is a conditional Gaussian (CG) distribution.

For discrete nodes, this means that the local probability distributions are unrestricted discrete distributions. We parameterize this as

$$\theta_{i_\delta|i_{pa(\delta)}} = p(i_\delta|i_{pa(\delta)}, \theta_{\delta|i_{pa(\delta)}}),$$

where $\theta_{\delta|i_{pa(\delta)}} = (\theta_{i_\delta|i_{pa(\delta)}})_{i_\delta \in \mathcal{I}_\delta}$. The parameters fulfill $\sum_{i_\delta \in \mathcal{I}_\delta} \theta_{i_\delta|i_{pa(\delta)}} = 1$ and $0 \leq \theta_{i_\delta|i_{pa(\delta)}} \leq 1$.

For continuous nodes, the local probability distributions are Gaussian linear regressions on the continuous parents with parameters depending on the configuration of the discrete parents. We parameterize this as

$$\theta_{\gamma|i_{pa(\gamma)}} = (m_{\gamma|i_{pa(\gamma)}}, \beta_{\gamma|i_{pa(\gamma)}}, \sigma_{\gamma|i_{pa(\gamma)}}^2),$$

so that

$$(Y_\gamma | i_{\text{pa}(\gamma)}, y_{\text{pa}(\gamma)}, \theta_{\gamma | i_{\text{pa}(\gamma)}}) \sim \mathcal{N}(m_{\gamma | i_{\text{pa}(\gamma)}} + y_{\text{pa}(\gamma)} \beta_{\gamma | i_{\text{pa}(\gamma)}}, \sigma_{\gamma | i_{\text{pa}(\gamma)}}^2).$$

A suggestion for the local probability distributions is generated and attached to each node as the property `prob`. The suggestion can then be edited afterwards.

For a discrete variable δ , the suggested local probability distribution $p(i_\delta | i_{\text{pa}(\delta)})$ is taken to be uniform over the levels for each parent configuration, *i.e.*

$$p(i_\delta | i_{\text{pa}(\delta)}) = 1/\mathcal{I}_\delta.$$

Define $z_{\text{pa}(\gamma)} = (1, y_{\text{pa}(\gamma)})$ and let $\eta_{\gamma | i_{\text{pa}(\gamma)}} = (m_{\gamma | i_{\text{pa}(\gamma)}}, \beta_{\gamma | i_{\text{pa}(\gamma)}})$, where $m_{\gamma | i_{\text{pa}(\gamma)}}$ is the intercept and $\beta_{\gamma | i_{\text{pa}(\gamma)}}$ is the vector of coefficients. For a continuous variable γ , the suggested local probability distribution

$$\mathcal{N}(z_{\text{pa}(\gamma)} \eta_{\gamma | i_{\text{pa}(\gamma)}}, \sigma_{\gamma | i_{\text{pa}(\gamma)}}^2),$$

is determined as a regression on the continuous parents for each configuration of the discrete parents.

The `prob` property for discrete nodes is a multi-way array with the node itself occupying the first dimension and the parents each occupying one dimension. For continuous nodes, $\sigma_{\gamma | i_{\text{pa}(\gamma)}}^2$ and $\eta_{\gamma | i_{\text{pa}(\gamma)}}$ are stored in a matrix with one row for each configuration of the discrete variables. The first column contains $\sigma_{\gamma | i_{\text{pa}(\gamma)}}^2$ and the remaining columns $\eta_{\gamma | i_{\text{pa}(\gamma)}}$.

It is possible to inspect the suggested local probability distributions by setting the option `inspectprob` as

```
rats <- network(rats.df, inspectprob=TRUE)
```

This gives a graphical way of inspecting the local probability distribution by clicking on the nodes. Then, it is possible to adjust the local distributions, *e.g.*

```
localprob(rats, "Sex") <- c(0.6, 0.4)
localprob(rats, "W1") <- c(10, 0)
localprob(rats, "W2") <- c(10, 0, 1) # if eg. W2|W1
```

4.3 The Joint Distribution

We now show how the joint probability distribution of a network can be calculated from the local probability distributions.

For the discrete part of the network, the joint probability distribution is found as

$$p(i) = \prod_{\delta \in \Delta} p(i_\delta | i_{\text{pa}(\delta)}).$$

For continuous variables, the joint distribution $\mathcal{N}(M_i, \Sigma_i)$ is determined for each configuration of the discrete variables by applying the following sequential algorithm, see Shachter and Kenley (1989).

The order is determined so that the joint distribution of the parents have already been determined for the current node. For notational convenience, we skip the index i and determine the joint distribution of node γ and all previously processed nodes, p . From the prior network, we have given $\eta_{\gamma|\text{pa}(\gamma)} = (m_{\gamma|\text{pa}(\gamma)}, \beta_{\gamma|\text{pa}(\gamma)})$ and $\sigma_{\gamma|\text{pa}(\gamma)}^2$. Previously evaluated are M_p and Σ_p . Now, the covariance is given by

$$\Sigma_{\gamma,p} = \Sigma_p \beta_{\gamma|p},$$

where $\beta_{\gamma|p}$ is a column vector of the regression coefficients given all previously evaluated nodes. All coefficients are zero, except the coefficients, $\beta_{\gamma|\text{pa}(\gamma)}$, corresponding to the parents of the node. The variance and mean are then given by

$$\begin{aligned} \Sigma_\gamma &= \sigma_{\gamma|\text{pa}(\gamma)}^2 + \Sigma_{\gamma,p} \beta_{\gamma|p} \\ M_\gamma &= m_{\gamma|\text{pa}(\gamma)} + \beta_{\gamma|p}^\top M_p. \end{aligned}$$

In deal, we can assess these quantities by

```
rats.j <- jointprior(rats)
```

and inspect the properties `jointmu`, containing M_i , `jointsigma`, containing Σ_i , and `jointalpha`. The discrete part, $p(i)$, is not returned directly, but is found by dividing `rats.j$jointalpha` with `sum(rats.j$jointalpha)`.

5 Parameter Learning

In the previous section we showed how to specify a Bayesian network, *i.e.* a DAG and the local probability distributions. In this section we will show how to estimate the parameters in the local probability distributions from data. The first sections present the theory behind the learning procedure and the last section shows how it is done in `deal`.

5.1 The Bayesian Approach

To estimate the parameters in the network, we use a Bayesian approach. We encode our uncertainty about parameters θ in a prior distribution $p(\theta)$, use data d to update this distribution, and hereby obtain the posterior distribution $p(\theta|d)$ by using Bayes' theorem,

$$p(\theta|d) = \frac{p(d|\theta)p(\theta)}{p(d)}, \quad \theta \in \Theta. \quad (1)$$

Here Θ is the parameter space, d is a random sample from the probability distribution $p(x|\theta)$ and $p(d|\theta)$ is the joint probability distribution of d , also called the likelihood of θ . We refer to this as *parameter learning* or just *learning*.

In `deal`, we assume that the parameters associated with one variable are independent of the parameters associated with the other variables and, in addition, that the parameters are independent for each configuration of the discrete parents, *i.e.*

$$p(\theta) = \prod_{\delta \in \Delta} \prod_{i_{\text{pa}(\delta)} \in \mathcal{I}_{\text{pa}(\delta)}} p(\theta_{\delta|i_{\text{pa}(\delta)}}) \prod_{\gamma \in \Gamma} \prod_{i_{\text{pa}(\gamma)} \in \mathcal{I}_{\text{pa}(\gamma)}} p(\theta_{\gamma|i_{\text{pa}(\gamma)}}). \quad (2)$$

We refer to (2) as *parameter independence*. Further, as we have assumed complete data, the parameters stay independent given data, see Bøttcher (2001). This means that we can learn the parameters of a node independently of the parameters of the other nodes, *i.e.* we update the *local parameter prior* $p(\theta_v|i_{\text{pa}(v)})$ for each node v and each configuration of the discrete parents.

As local prior parameter distributions, we use the Dirichlet distribution for the discrete variables and the Gaussian-inverse gamma distribution for the continuous variables. These distributions are conjugate to observations from the respective distributions and this ensures simple calculations of the posterior distributions.

In the next section we present an automated procedure for specifying the local parameter priors associated with any possible DAG. The procedure is called the *master prior procedure*. For the mixed case it is treated in Bøttcher (2001), for the purely discrete and the purely continuous cases it is treated in Heckerman et al. (1995) and Geiger and Heckerman (1994), respectively.

5.2 The Master Prior Procedure

In the following sections we will show how to deduce and update the local prior parameter distributions for discrete and continuous nodes, respectively. Here, we will summarize the steps in the master prior procedure.

The idea is that from a given Bayesian network, we can deduce parameter priors for any possible DAG. The user just has to specify the Bayesian network as he believes it to be. We call this network a *prior Bayesian network*.

1. Specify a prior Bayesian network, *i.e.* a prior DAG (Section 4.1) and prior local probability distributions (Section 4.2). Calculate the joint prior distribution (Section 4.3).
2. From this joint prior distribution, the marginal distribution of all parameters in the family consisting of the node and its parents can be determined. We call this the *master prior*.
3. The local parameter priors are now determined by conditioning in the master prior distribution.

This procedure ensures parameter independence. Further, it has the property that if a node has the same set of parents in two different networks, then the local parameter prior for this node will be the same in the two networks. Therefore, we only have to deduce the local parameter prior for a node given the same set of parents once. This property is called *parameter modularity*.

5.3 Discrete Nodes

We will now show how to find the local parameter priors for the discrete nodes. Recall that the local probability distributions are unrestricted discrete distributions defined as in Section 4.2.

Master Prior

Let $\Psi = (\Psi_i)_{i \in \mathcal{I}}$ be the parameters for the joint distribution of the discrete variables. The joint prior parameter distribution is assumed to be a Dirichlet distribution

$$p(\Psi) \sim \mathcal{D}(\alpha),$$

with hyperparameters $\alpha = (\alpha_i)_{i \in \mathcal{I}}$. To specify this Dirichlet distribution, we need to specify these hyperparameters.

Consider the following relation for the Dirichlet distribution,

$$p(i) = \mathbb{E}(\Psi_i) = \frac{\alpha_i}{N},$$

with $N = \sum_{i \in \mathcal{I}} \alpha_i$. Now we use the probabilities in the prior network as an estimate of $\mathbb{E}(\Psi_i)$, so we only need to determine N in order to calculate the parameters α_i .

We determine N by using the notion of an imaginary data base. We imagine that we have a data base of cases, from which we have updated the distribution of Ψ out of total ignorance. The *imaginary sample size* of this imaginary data base is thus N . It expresses how much confidence we have in the (in)dependencies expressed in the prior network, see Heckerman et al. (1995).

We use this joint distribution to deduce the master prior distribution of the family $A = \delta \cup \text{pa}(\delta)$. Let

$$\alpha_{i_A} = \sum_{j: j_A = i_A} \alpha_j,$$

and let $\alpha_A = (\alpha_{i_A})_{i_A \in \mathcal{I}_A}$. Then the marginal distribution of Ψ_A is Dirichlet, $p(\Psi_A) \sim \mathcal{D}(\alpha_A)$. This is the master prior in the discrete case.

Local Parameter Prior

From the master prior, we calculate the conditional distribution $\Psi_{\delta|i_{\text{pa}(\delta)}} = \theta_{\delta|i_{\text{pa}(\delta)}}$ which is the local parameter prior in the discrete case. Then,

$$\begin{aligned} \alpha_{i_{\delta|i_{\text{pa}(\delta)}}} &= \alpha_{i_A}, \\ \alpha_{\delta|i_{\text{pa}(\delta)}} &= (\alpha_{i_{\delta|i_{\text{pa}(\delta)}}})_{i_{\delta} \in \mathcal{I}_{\delta}}, \\ \theta_{\delta|i_{\text{pa}(\delta)}} | \alpha_{i_{\delta|i_{\text{pa}(\delta)}}} &\sim \mathcal{D}(\alpha_{\delta|i_{\text{pa}(\delta)}}). \end{aligned}$$

Local Parameter Posterior

Let $n_{\delta|i_{\text{pa}(\delta)}}$ be the number of cases observed with the particular parent configuration in the data base and let n be the total number of observations.

Then, the posterior parameters $\alpha'_{\delta|i_{\text{pa}(\delta)}}$ are given by

$$\alpha'_{\delta|i_{\text{pa}(\delta)}} = \alpha_{\delta|i_{\text{pa}(\delta)}} + n_{\delta|i_{\text{pa}(\delta)}}.$$

5.4 Continuous Nodes

We now show how to find the local parameter priors for the continuous nodes. Recall that the local probability distributions are normal distributions defined as in Section 4.2.

Master Prior

Bøttcher (2001) derived this procedure in the mixed case. For a configuration i of the discrete variables we let $\nu_i = \rho_i = \alpha_i$, where α_i was determined in Section 5.3. Also, $\Phi_i = (\nu_i - 1)\Sigma_i$.

The joint parameter priors are assumed to be distributed as

$$\begin{aligned} p(M_i|\Sigma_i) &= \mathcal{N}\left(\mu_i, \frac{1}{\nu_i}\Sigma_i\right), \\ p(\Sigma_i) &= \mathcal{IW}(\rho_i, \Phi_i), \end{aligned}$$

where \mathcal{IW} is the inverse Wishart distribution.

However, since the marginal distribution of a CG distribution is not necessarily a CG distribution, there is no simple way to derive priors for other networks. Instead we use the imaginary data base to derive local master priors.

Define the notation

$$\rho_{i_{A \cap \Delta}} = \sum_{j: j_{A \cap \Delta} = i_{A \cap \Delta}} \rho_j$$

and similarly for $\nu_{i_{A \cap \Delta}}$ and $\Phi_{i_{A \cap \Delta}}$. For the family $A = \gamma \cup \text{pa}(\gamma)$, the local

master prior is then found as

$$\begin{aligned}\Sigma_{A \cap \Gamma | i_{A \cap \Delta}} &\sim \mathcal{IW}(\rho_{i_{A \cap \Delta}}, \tilde{\Phi}_{A \cap \Gamma | i_{A \cap \Delta}}), \\ M_{A \cap \Gamma | i_{A \cap \Delta}} | \Sigma_{A \cap \Gamma | i_{A \cap \Delta}} &\sim \mathcal{N}\left(\bar{\mu}_{A \cap \Gamma | i_{A \cap \Delta}}, \frac{1}{\nu_{i_{A \cap \Delta}}} \Sigma_{A \cap \Gamma | i_{A \cap \Delta}}\right),\end{aligned}$$

where

$$\begin{aligned}\bar{\mu}_{i_{A \cap \Delta}} &= \frac{\sum_{j: j_{A \cap \Delta} = i_{A \cap \Delta}} \mu_j \nu_j}{\nu_{i_{A \cap \Delta}}}, \\ \tilde{\Phi}_{A \cap \Gamma | i_{A \cap \Delta}} &= \Phi_{i_{A \cap \Delta}} + \sum_{j: j_{A \cap \Delta} = i_{A \cap \Delta}} \nu_j (\mu_j - \bar{\mu}_{i_{A \cap \Delta}}) (\mu_j - \bar{\mu}_{i_{A \cap \Delta}})^\top.\end{aligned}$$

Local Parameter Prior

Using \mathcal{IG} for the inverse gamma distribution, the local prior parameters, given as

$$\begin{aligned}(m_{\gamma | i_{\text{pa}(\gamma)}}, \beta_{\gamma | i_{\text{pa}(\gamma)}} | \sigma_{\gamma | i_{\text{pa}(\gamma)}}^2) &\sim \mathcal{N}(\mu_{\gamma | i_{\text{pa}(\gamma)}}, \sigma_{\gamma | i_{\text{pa}(\gamma)}}^2 \tau_{\gamma | i_{\text{pa}(\gamma)}}^{-1}), \\ \sigma_{\gamma | i_{\text{pa}(\gamma)}}^2 &\sim \mathcal{IG}\left(\frac{\rho_{\gamma | i_{\text{pa}(\gamma)}}}{2}, \frac{\phi_{\gamma | i_{\text{pa}(\gamma)}}}{2}\right),\end{aligned}$$

are deduced from the local master prior by conditioning as follows. To simplify notation, we ignore all subscripts in the master prior and thus consider the configuration $i_{A \cap \Delta} = i_{\text{pa}(\gamma)}$ and assume that $A \cap \Gamma$ is ordered with γ as the first entry. Write $\text{pa}(\gamma)$ for the continuous parents $\{A \cap \Gamma\} \setminus \{\gamma\}$. Define the partitioning

$$\begin{aligned}\tilde{\Phi}_{A \cap \Gamma | i_{A \cap \Delta}} &= \begin{bmatrix} \tilde{\phi}_{\gamma} & \tilde{\Phi}_{\gamma, \text{pa}(\gamma)} \\ \tilde{\Phi}_{\text{pa}(\gamma), \gamma} & \tilde{\Phi}_{\text{pa}(\gamma)} \end{bmatrix}, \\ \bar{\mu}_{i_{A \cap \Delta}} &= \left(\bar{\mu}_{\gamma}, \bar{\mu}_{\text{pa}(\gamma)} \right).\end{aligned}$$

Then

$$\begin{aligned}\mu_{\gamma | i_{\text{pa}(\gamma)}} &= \left(\bar{\mu}_{\gamma} - \tilde{\Phi}_{\gamma, \text{pa}(\gamma)} \tilde{\Phi}_{\text{pa}(\gamma)}^{-1} \bar{\mu}_{\text{pa}(\gamma)}, \tilde{\Phi}_{\gamma, \text{pa}(\gamma)} \tilde{\Phi}_{\text{pa}(\gamma)}^{-1} \right), \\ \phi_{\gamma | i_{\text{pa}(\gamma)}} &= \tilde{\phi}_{\gamma} - \tilde{\Phi}_{\gamma, \text{pa}(\gamma)} \tilde{\Phi}_{\text{pa}(\gamma)}^{-1} \tilde{\Phi}_{\text{pa}(\gamma), \gamma}, \\ \rho_{\gamma | i_{\text{pa}(\gamma)}} &= \rho_{i_{A \cap \Delta}} + |\text{pa}(\gamma)|, \\ \tau_{\gamma | i_{\text{pa}(\gamma)}} &= \left(\begin{array}{cc} 1/\nu_{i_{A \cap \Delta}} + \bar{\mu}_{\text{pa}(\gamma)}^\top \tilde{\Phi}_{\text{pa}(\gamma)}^{-1} \bar{\mu}_{\text{pa}(\gamma)} & -\bar{\mu}_{\text{pa}(\gamma)}^\top \tilde{\Phi}_{\text{pa}(\gamma)}^{-1} \\ -\tilde{\Phi}_{\text{pa}(\gamma)}^{-1} \bar{\mu}_{\text{pa}(\gamma)} & \tilde{\Phi}_{\text{pa}(\gamma)}^{-1} \end{array} \right)^{-1}.\end{aligned}$$

Local Parameter Posterior

Define $z_{\text{pa}(\gamma)|i_{\text{pa}(\gamma)}}^b$ as the matrix with n rows and with a column of ones and columns of the observed continuous parents for a given configuration of the discrete parents. Let $y_{\gamma|i_{\text{pa}(\gamma)}}^b$ be the vector of observations of the node γ for a configuration of the discrete parents.

Then, the prior parameters $\tau_{\gamma|i_{\text{pa}(\gamma)}}$, $\mu_{\gamma|i_{\text{pa}(\gamma)}}$, $\rho_{\gamma|i_{\text{pa}(\gamma)}}$, $\phi_{\gamma|i_{\text{pa}(\gamma)}}$ are updated to posterior parameters (denoted with a prime) by the following relations

$$\begin{aligned}\tau'_{\gamma|i_{\text{pa}(\gamma)}} &= \tau_{\gamma|i_{\text{pa}(\gamma)}} + (z_{\text{pa}(\gamma)|i_{\text{pa}(\gamma)}}^b)^{\top} z_{\text{pa}(\gamma)|i_{\text{pa}(\gamma)}}^b, \\ \mu'_{\gamma|i_{\text{pa}(\gamma)}} &= (\tau'_{\gamma|i_{\text{pa}(\gamma)}})^{-1} \times \left(\tau_{\gamma|i_{\text{pa}(\gamma)}} \mu_{\gamma|i_{\text{pa}(\gamma)}} + (z_{\text{pa}(\gamma)|i_{\text{pa}(\gamma)}}^b)^{\top} y_{\gamma|i_{\text{pa}(\gamma)}}^b \right), \\ \rho'_{\gamma|i_{\text{pa}(\gamma)}} &= \rho_{\gamma|i_{\text{pa}(\gamma)}} + n, \\ \phi'_{\gamma|i_{\text{pa}(\gamma)}} &= \phi_{\gamma|i_{\text{pa}(\gamma)}} \\ &\quad + (y_{\gamma|i_{\text{pa}(\gamma)}}^b - z_{\text{pa}(\gamma)}^b \mu'_{\gamma|i_{\text{pa}(\gamma)}})^{\top} y_{\gamma|i_{\text{pa}(\gamma)}}^b \\ &\quad + (\mu_{\gamma|i_{\text{pa}(\gamma)}} - \mu'_{\gamma|i_{\text{pa}(\gamma)}})^{\top} \tau_{\gamma|i_{\text{pa}(\gamma)}} \mu_{\gamma|i_{\text{pa}(\gamma)}}.\end{aligned}$$

5.5 The Learning Procedure in deal

Assume that the training data are available in a data frame, `rats.df`, as described in Section 3. Also, assume that the user has specified a Bayesian network to be used as prior network, called `rats`, see Section 4.

The parameters of the joint distribution of the variables in the network are then determined by the function `jointprior()` with the size of the imaginary data base as optional argument. If the size is not specified, `deal` sets the size to a reasonably small value.

```
rats.prior <- jointprior(rats)
## auto set size of imaginary data base

rats.prior <- jointprior(rats,12)
## set size of imaginary data base to 12
```

The parameters in the object `rats.prior` may be assessed as

```
rats.prior$jointalpha
rats.prior$jointnu
rats.prior$jointrho
rats.prior$jointphi
```

The procedure `learn()` determines the master prior, local parameter priors and local parameter posteriors and may be called on all nodes or just a single node. The result is accessed using the `getnetwork()` extractor function.

```
rats <- getnetwork(learn(rats,rats.df,rats.prior))
## all nodes

rats <- getnetwork(learn(rats,rats.df,rats.prior,2))
## only node 2
```

In the result, each learned node has now attached two properties. These contain the parameters in the local prior distribution and the parameters in the local posterior distribution, respectively. For the node `Sex`, the properties are assessed as

```
localprior(nodes(rats)$Sex)
localposterior(nodes(rats)$Sex)
```

6 Learning the Structure

In this section we will show how to learn the structure of the DAG from data. The section is based on Bøttcher (2001), Heckerman et al. (1995) and Geiger and Heckerman (1994).

6.1 Network Score

As a measure of how well a DAG D represents the conditional independencies between the random variables, we use the relative probability

$$S(D) = p(D, d) = p(d|D)p(D),$$

and refer to it as a *network score*.

The network score factorizes into a discrete part and a mixed part as

$$S(D) = \prod_{\delta \in \Delta} S_{\delta}(D) \prod_{\gamma \in \Gamma} S_{\gamma}(D),$$

where $S_{\delta}(D)$ is the contribution from the discrete node δ and $S_{\gamma}(D)$ is the contribution from the continuous node γ .

For a discrete node, δ , the score contribution is given by

$$S_{\delta}(D) = \prod_{i_{\text{pa}(\delta)} \in \mathcal{I}_{\text{pa}(\delta)}} \frac{\Gamma(\alpha_{+\delta|i_{\text{pa}(\delta)}})}{\Gamma(\alpha_{+\delta|i_{\text{pa}(\delta)}} + n_{+\delta|i_{\text{pa}(\delta)}})} \prod_{i_{\delta} \in \mathcal{I}_{\delta}} \frac{\Gamma(\alpha_{i_{\delta}|i_{\text{pa}(\delta)}} + n_{i_{\delta}|i_{\text{pa}(\delta)}})}{\Gamma(\alpha_{i_{\delta}|i_{\text{pa}(\delta)}})},$$

where $\alpha_{+\delta|i_{\text{pa}(\delta)}} = \sum_{i_{\delta} \in \mathcal{I}_{\delta}} \alpha_{i_{\delta}|i_{\text{pa}(\delta)}}$ and $n_{+\delta|i_{\text{pa}(\delta)}} = \sum_{i_{\delta} \in \mathcal{I}_{\delta}} n_{i_{\delta}|i_{\text{pa}(\delta)}}$.

For a continuous node, γ ,

$$S_{\gamma}(D) = \prod_{i_{\text{pa}(\gamma)} \in \mathcal{I}_{\text{pa}(\gamma)}} \frac{\Gamma\left(\frac{\rho_{\gamma|i_{\text{pa}(\gamma)}} + n}{2}\right)}{\Gamma\left(\frac{\rho_{\gamma|i_{\text{pa}(\gamma)}}}{2}\right) \sqrt{\det(\rho_{\gamma|i_{\text{pa}(\gamma)}} s_{\gamma|i_{\text{pa}(\gamma)}} \pi)}} \times \left[1 + \frac{1}{\rho_{\gamma|i_{\text{pa}(\gamma)}}} a_{\gamma|i_{\text{pa}(\gamma)}} s_{\gamma|i_{\text{pa}(\gamma)}}^{-1} a_{\gamma|i_{\text{pa}(\gamma)}}^{\top} \right]^{-\frac{\rho_{\gamma|i_{\text{pa}(\gamma)}} + n}{2}},$$

where

$$s_{\gamma|i_{\text{pa}(\gamma)}} = \frac{\phi_{\gamma|i_{\text{pa}(\gamma)}}}{\rho_{\gamma|i_{\text{pa}(\gamma)}}} \left(I + z_{\text{pa}(\gamma)}^b \tau_{\gamma|i_{\text{pa}(\gamma)}}^{-1} (z_{\text{pa}(\gamma)}^b)^{\top} \right),$$

$$a_{\gamma|i_{\text{pa}(\gamma)}} = y_{\gamma|i_{\text{pa}(\gamma)}}^b - z_{\text{pa}(\gamma)|i_{\text{pa}(\gamma)}}^b \mu_{\gamma|i_{\text{pa}(\gamma)}}.$$

Note that the network score factorizes into a product over terms involving only one node and its parents. This property is called *decomposability*.

It can be shown that the network scores for two independence equivalent DAGs are equal. This property is called *likelihood equivalence* and it is a property of the master prior procedure.

In deal we use, for computational reasons, the logarithm of the network score. The log network score contribution of a node is evaluated whenever the node is learned and the log network score is updated. The results are inspected as

```
rats <- getnetwork(learn(rats,rats.df,rats.prior))
score(nodes(rats)$Sex)
score(rats) # log network score
```

6.2 Model Search

In principle, we could evaluate the network score for all possible DAGs and indeed this is provided in `deal`.

```
allrats <- networkfamily(rats.df,rats,rats.prior)
allrats <- nwfsort(getnetwork(allrats))
```

However, the number of possible DAGs grows more than exponentially with the number of nodes (see Table 2) and, in general, the problem of identifying the network with the highest score is NP-complete (see Chickering (1996)). If

# nodes	# networks
1	1
2	2–3
3	12–25
4	144–543
5	4800–29281
6	320000–3781503
7	$\approx 56 \cdot 10^6 - 10^9$
8	$\approx 10^{10} - 10^{11}$
9	$\approx 10^{13} - 10^{15}$
10	$\approx 10^{16} - 10^{18}$

Table 2: The (approximate) number of networks for a given number of nodes.

Since we do not allow arrows from continuous to discrete nodes, the number of networks for a given number of nodes is given as a lower and upper bound.

the number of random variables in a network is large, it is not computationally possible to calculate the network score for all the possible DAGs. For these situations a strategy for searching for DAGs with high score is needed. In `deal`, the search strategy *greedy search with random restarts*, see Heckerman et al. (1995), is implemented. As a way of comparing the network scores for two different DAGs, D and D^* , we use the posterior odds,

$$\frac{p(D|d)}{p(D^*|d)} = \frac{p(D, d)}{p(D^*, d)} = \frac{p(D)}{p(D^*)} \times \frac{p(d|D)}{p(d|D^*)},$$

where $p(D)/p(D^*)$ is the prior odds and $p(d|D)/p(d|D^*)$ is the Bayes factor. At the moment, the only option in `deal` for specifying prior distribution over

DAGs is to let all DAGs be equally likely, so the prior odds are always equal to one. Therefore, we use the Bayes factor for comparing two different DAGs.

In greedy search we compare models that differ only by a single arrow, either added, removed or reversed. In these cases, the Bayes factor is especially simple, because of decomposability of the network score.

Greedy search works as follows.

1. Select an initial DAG D_0 , from which to start the search.
2. Calculate Bayes factors between D_0 and all possible networks, which differ by only one arrow, that is
 - (a) One arrow is added to D_0 .
 - (b) One arrow in D_0 is deleted.
 - (c) One arrow in D_0 is turned.
3. Among all these networks, select the one that increases the Bayes factor the most.
4. If the Bayes factor is not increased, stop the search. Otherwise, let the chosen network be D_0 and repeat from 2.

In deal

```
rats.s <- getnetwork(autosearch(rats,rats.df,rats.prior))
```

returns all tried networks in a greedy search from the initial network `rats`, which may be constructed using `drawnetwork()`.

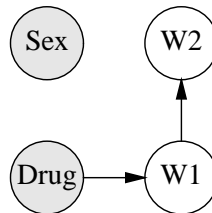


Figure 3: The network with the highest score in the rats example.

To manually assess the network score of a network (*e.g.* to use as initial network in a search), use

```
rats <- getnetwork(drawnetwork(rats,rats.df,rats.prior))
```

In the `drawnetwork()` procedure, it is possible to mark (ban) some of the arrows. In the search, `deal` then disregards any DAG which contains any of these arrows, and this reduces the search space.

The search algorithm may also be used with restarts which is implemented in the function `heuristic()`. The initial network is then perturbed according to the parameter `degree` and the search is performed starting with the perturbed network. The process is restarted the number of times specified by the option `restart`. A network family of all visited networks is returned.

```
rats.h <- getnetwork(heuristic(rats,rats.df,rats.prior,
                             restart=10,degree=5))
```

The perturbation of the initial network is done as follows

1. Randomly choose between one of three actions
 - (a) Insert an arrow.
 - (b) Delete an arrow.
 - (c) Turn an arrow.
2. After selection of the action, perform the action according to
 - Insert** Choose randomly between all possible insertions of one arrow.
 - Delete** Choose randomly between all possible deletions of one arrow.
 - Turn** Choose randomly between all possible turns of one arrow.
 If the action is not possible, return the unchanged network.

Perturbation is done automatically in `heuristic()` by calling the function `perturb()`. However, a random graph may also be generated by directly calling `perturb()`

```
rats.rn <- getnetwork(perturb(rats,rats.df,rats.prior,
                             degree=10))
```

6.3 Using Equivalence Relations to Speed up Model Search

In Bøttcher (2003) two types of equivalences are identified and it is shown that no other equivalences exist. Let D_1 and D_1^* be two different networks that differ by a single arrow between the nodes v and w , with $v \leftarrow w$ in D_1 and $v \nleftrightarrow w$ in D_1^* . Further, let D_2 and D_2^* be another two networks that differ by an arrow between v and w .

1. The Bayes factor for testing the arrow from v to w is equivalent to testing this arrow in any other network, where v has the same parents as in D_1 .
2. The Bayes factor for testing the arrow from v to w is equivalent to this arrow in the network, where w has the same parents in D_2 as v has in D_1 , with the exception that v is also a parent of w in D_2 .

We use the first equivalence in all functions that call the learning procedure, including `heuristic()`, `learn()`, `drawnetwork()`, `networkfamily()` and `perturb()`, by maintaining a so-called `trylist`. The `trylist` may be given as input to the functions and is returned in an updated version.

The `trylist` contains a list for each node in the network. The list for a node consists of the result after learning the node for all parent configurations that has previously been tried. When a node is learned after a change in its parent structure, we first look in the `trylist` to see if the node has been learned before with the same parent configuration (Equivalence 1). If the equivalence cannot be used, the node is learned and the result is inserted in the `trylist`. Utilization of Equivalence 2 is not yet implemented in `deal`.

The cost of looking in the `trylist` is smaller than learning a node. Note, however, that the `trylist` must be recalculated if the imaginary data base size is changed or if the data base is changed.

In `deal`, there is support for generating the complete `trylist`, that is, all nodes are learned with all possible parent configurations.

```
rats.tl <- maketrylist(rats,rats.df,rats.prior)
rats.h  <- getnetwork(heuristic(rats,rats.df,rats.prior,
                             trylist=rats.tl))
```

7 Hugin Interface

A network object may be written to a file in the Hugin `.net` language. Hugin (<http://www.hugin.com>) is commercial software for inference in Bayesian networks. Hugin has the ability to learn networks with only discrete variables, but cannot learn either purely continuous or mixed networks. `deal` may therefore be used for this purpose and the result can then be transferred to Hugin.

The procedure `savenet()` saves a network to a connection (for example a file).

For each node, we use point estimates of the parameters in the local probability distributions.

The `readnet()` procedure reads the network structure from a connection but does not, however, read the probability distributions. This is planned to be included in a future version of `deal`.

8 Example

In this section, `deal` is used to analyze a large data set which includes both discrete and continuous variables. The *ksl* data set, included in Badsberg (1995), is from a study measuring health and social characteristics of representative samples of Danish 70-year old people, taken in 1967 and 1984. In total, 1083 cases have been recorded and each case contains observations on nine different variables, see Table 3.

Node index	Variable	Explanation
1	Fev	Forced ejection volume – lung function
2	Kol	Cholesterol
3	Hyp	Hypertension (no/yes)
4	BMI	Body Mass Index
5	Smok	Smoking (no/yes)
6	Alc	Alcohol consumption (seldom/frequently)
7	Work	Working (yes/no)
8	Sex	Gender (male/female)
9	Year	Survey year (1967/1984)

Table 3: Variables in the *ksl* data set. The variables `Fev`, `Kol`, `BMI` are continuous variables and the rest are discrete variables.

The purpose of our analysis is to find dependency relations between the variables. One interest is to determine which variables influence the presence or absence of hypertension. From a medical viewpoint, it is possible that hypertension is influenced by some of the continuous variables `Fev`, `Kol` and `BMI`. However, in `deal` we do not allow continuous parents of discrete nodes, so we cannot describe such a relation. A way to overcome this problem is to treat `Hyp` as a continuous variable, even though this is obviously not most natural. This is done in the analysis below.

Further, the initial data analysis indicates a transformation of BMI into $\log(\text{BMI})$. With these adjustments, the data set is ready for analysis in `deal`.

First, `deal` is activated and the data are read into a data frame and prepared for analysis.

```
library(deal) ## invoke DEAL
data(ksl)      ## read data (included in DEAL)
```

The next step in the analysis is to specify a prior Bayesian network. We have no prior knowledge about specific dependency relations, so for simplicity we use the empty DAG as the prior DAG and let the probability distribution of the discrete variables be uniform. The assessment of the probability distribution for the continuous variables is based on data, as described in Section 4.2.

```
## specify prior network
ksl.nw <- network(ksl)

## make joint prior distribution
ksl.prior <- jointprior(ksl.nw)
```

We do not allow arrows into `Sex` and `Year`, as none of the other variables can influence these variables. So we create a ban list which is attached to the network. The ban list is a matrix with two columns. Each row contains the directed edge that is not allowed. The ban list could also have been created interactively using the function `drawnetwork()`.

```
## ban arrows towards Sex and Year
mybanlist <- matrix(c(5,5,6,6,7,7,9,
                    8,9,8,9,8,9,8),ncol=2)
banlist(ksl.nw) <- mybanlist
```

Finally, the parameters in the network are learned and structural learning is initiated using `autosearch()` and `heuristic()`. We use the prior DAG as starting point for the structural search.

```
## learn the initial network
ksl.nw <- getnetwork(learn(ksl.nw,ksl,ksl.prior))

## Do structural search
ksl.search <- autosearch(ksl.nw,ksl,ksl.prior,trace=TRUE)

## perturb 'thebest' and rerun search twice.
ksl.heuristic <- heuristic(getnetwork(ksl.search),
                           ksl,
                           ksl.prior,
                           restart=2,degree=10,
                           trace=TRUE,
                           trylist=gettrylist(ksl.search))

thebest2 <- getnetwork(ksl.heuristic)

savenet(thebest2, file("ksl.net"))
```

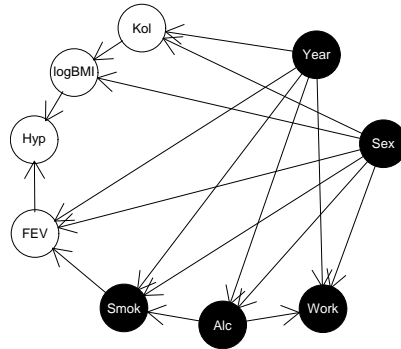


Figure 4: The network with the highest score, $\log(\text{score}) = -15957.91$.

The resulting network `thebest2` is shown in Figure 4 and it is the network with the highest network score among those networks that have been tried through the search.

In the result we see for the discrete variables that `Alc`, `Smok` and `Work` depend directly on `Sex` and `Year`. In addition, `Smok` and `Work` also depend on `Alc`.

These two arrows are, however, not causal arrows, as $\text{Smok} \leftarrow \text{Alc} \rightarrow \text{Work}$ in the given DAG represents the same probability distribution as the relations $\text{Smok} \leftarrow \text{Alc} \leftarrow \text{Work}$ and $\text{Smok} \rightarrow \text{Alc} \rightarrow \text{Work}$, *i.e.* the three DAGs are independence equivalent.

`Year` and `Sex` are independent on all variables, as specified in the ban list.

For the continuous variables all the arrows are causal arrows. We see that `Fev` depends directly on `Year`, `Sex` and `Smok`. So given these variables, `Fev` is conditionally independent on the rest of the variables. `Ko1` depends directly on `Year` and `Sex`, and `logBMI` depends directly on `Ko1` and `Sex`.

Given `logBMI` and `Fev`, the variable `Hyp` is conditionally independent on the rest of the variables. So according to this study, hypertension can be determined by the body mass index and the lung function forced ejection volume. However, as `Hyp` is not continuous by nature, other analyses should be performed with `Hyp` as a discrete variable, *e.g.* a logistic regression with `Hyp` as a response and the remaining as explanatory variables. Such an analysis indicates that, in addition, `Sex` and `Smok` may influence `Hyp`, but otherwise identifies `logBMI` as the main predictor.

9 Discussion and Future Work

`deal` is a tool box that adds functionality to `R` so that Bayesian networks may be used in conjunction with other statistical methods available in `R` for analyzing data. In particular, `deal` is part of the `gR` project, which is a newly initiated workgroup with the aim of developing procedures in `R` for supporting data analysis with graphical models, see <http://www.r-project.org/gR>.

In addition to methods for analyzing networks with either discrete or continuous variables, `deal` handles networks with mixed variables.

`deal` has some limitations and we plan to extend the package with the procedures described below. Also, it is the intention that the procedures in `deal` will eventually be adjusted to the other procedures developed under the `gR` project.

The methods in `deal` are only applicable on complete data sets and in the future, we would like to incorporate procedures for handling data with missing values and networks with latent variables.

The criteria for comparing the different network structures in deal, is the relative probability $S(D)$. We intend to also incorporate the Bayesian Information Criteria (BIC) and Akaike's Information Criteria (AIC) and let it be up to the user to decide which criteria to use.

Another possible extension of deal is to incorporate procedures for specifying mixed networks, where the variance in the mixed part of the network does not depend on the discrete parents, but the mean does.

Finally, we are working on an implementation of the greedy equivalence search (GES) algorithm, see Chickering (2002), which is an algorithm for search between equivalence classes. Asymptotically, for the size of the database tending to infinity, this algorithm guarantees that the search terminates with the network with the highest network score.

Acknowledgements

The work has been supported by Novo Nordisk A/S. We thank the reviewer for many helpful comments.

References

- Badsberg, J. H. (1995). *An Environment for Graphical Models*, PhD thesis, Aalborg University.
- Bøttcher, S. G. (2001). Learning Bayesian Networks with Mixed Variables, *Artificial Intelligence and Statistics 2001*, Morgan Kaufmann, San Francisco, CA, USA, pp. 149–156.
- Bøttcher, S. G. (2003). Learning Bayesian networks with mixed variables, Aalborg University, <http://www.math.auc.dk/~alma>.
- Chickering, D. M. (1996). Learning Bayesian networks is NP-Complete, in D. Fisher and H. J. Lenz (eds), *Learning from Data: Artificial Intelligence and Statistics V*, Springer-Verlag, New York, pp. 121–130.
- Chickering, D. M. (2002). Optimal structure identification with greedy search, *Journal of Machine Learning Research* **3**: 507–554.
- Cowell, R. G., Dawid, A. P., Lauritzen, S. L. and Spiegelhalter, D. J. (1999). *Probabilistic Networks and Expert Systems*, Springer-Verlag, Berlin-Heidelberg-New York.

- Edwards, D. (1995). *Introduction to Graphical Modelling*, Springer-Verlag, New York.
- Geiger, D. and Heckerman, D. (1994). Learning Gaussian Networks, *Proceedings of Tenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, CA, USA, pp. 235–243.
- Heckerman, D., Geiger, D. and Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data, *Machine Learning* **20**: 197–243.
- Lauritzen, S. L. (1992). Propagation of probabilities, means and variances in mixed graphical association models, *Journal of the American Statistical Association* **87**(420): 1098–1108.
- Lauritzen, S. L. (2003). Some modern applications of graphical models, in P. J. Green, N. L. Hjort and S. Richardson (eds), *Highly Structured Stochastic Systems*, Oxford University Press, Oxford, pp. 13–32.
- Morrison, D. F. (1976). *Multivariate Statistical Methods*, McGraw-Hill, USA.
- R Development Core Team (2003). *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-00-3.
- Shachter, R. D. and Kenley, C. R. (1989). Gaussian influence diagrams, *Management Science* **35**: 527–550.

10 Manual Pages for deal

autosearch

Greedy search

Description

From initial network, does local perturbations to increase network score.

Usage

```
autosearch(initnw, data, prior=jointprior(network(data)), maxiter=50,
           trylist= vector("list", size(initnw)), trace=TRUE,
           timetrace=TRUE, showban=FALSE, removecycles=FALSE)
```

```
heuristic(initnw, data, prior=jointprior(network(data)),
          maxiter=100, restart=10, degree=size(initnw),
```

```

    trylist= vector("list",size(initnw)),trace=TRUE,
    timetrace=TRUE,removecycles=FALSE)
gettable(x)

```

Arguments

<code>initnw</code>	an object of class <code>network</code> , from which the search is started.
<code>data</code>	a data frame used for learning the network, see <code>network</code> .
<code>prior</code>	a list containing parameter priors, generated by <code>jointprior</code> .
<code>maxiter</code>	an integer, which gives the maximum number of steps in the search algorithm.
<code>restart</code>	an integer, which gives the number of times to perturb <code>initnw</code> and rerun the search.
<code>degree</code>	an integer, which gives the degree of perturbation, see <code>perturb</code> .
<code>trylist</code>	a list used internally for reusing learning of nodes, see <code>maketrylist</code> .
<code>trace</code>	a logical. If <code>TRUE</code> , plots the accepted networks during search.
<code>timetrace</code>	a logical. If <code>TRUE</code> , prints some timing information on the screen.
<code>showban</code>	a logical passed to the plot method for network objects. If <code>FALSE</code> , the banned arrows are not shown in the plots (if <code>trace</code> is <code>TRUE</code>).
<code>removecycles</code>	a logical. If <code>TRUE</code> , all networks explored in the search is returned, except for networks containing a cycle. If <code>FALSE</code> , all networks are returned, including cyclic networks.
<code>x</code>	an output object from a search.

Details

In `autosearch`, a list of networks is in each step created with either one arrow added, one arrow deleted or one arrow turned (if a cycle is not generated). The network scores of all the proposal networks are calculated and the network with the highest score is chosen for the next step in the search. If no proposed network has a higher network score than the previous network, the search is terminated. The network with the highest network score is returned, along with a list containing all tried networks (depending on the value of `removecycles`).

`heuristic` restarts by perturbing `initnw` `degree` times and calling `autosearch` again. The number of restarts is given by the option `restart`.

Value

`autosearch` and `heuristic` returns a list with three elements, that may be accessed using the functions `getnetwork`, `gettable` and `gettrylist`. The elements are

<code>nw</code>	an object of class <code>network</code> , which gives the network with the highest score.
<code>table</code>	a table with all tried networks. If <code>removecycles</code> is <code>FALSE</code> , the networks may contain cycles. The table contains two columns: <code>model</code> with a string representation of the model and <code>score</code> with the corresponding log network score. The table can be translated to a <code>networkfamily</code> using <code>makenw</code> .
<code>trylist</code>	an updated list used internally for reusing learning of nodes, see <code>maketrylist</code> .

Author(s)

Susanne Gammelgaard Bøttcher (alma@math.auc.dk),
 Claus Dethlefsen (dethlef@math.auc.dk).

See Also

perturb

Examples

```
data(rats)
fit      <- network(rats)
fit.prior <- jointprior(fit,12)
fit      <- getnetwork(learn(fit,rats,fit.prior))
fit      <- getnetwork(insert(fit,2,1,rats,fit.prior))
fit      <- getnetwork(insert(fit,1,3,rats,fit.prior))
hisc     <- autosearch(fit,rats,fit.prior,trace=FALSE)
hisc     <- autosearch(fit,rats,fit.prior,trace=FALSE,
                      removecycles=TRUE) # slower
plot(getnetwork(hisc))

hisc2    <- heuristic(fit,rats,fit.prior,restart=10,trace=FALSE)
plot(getnetwork(hisc2))
print(modelstring(getnetwork(hisc2)))
plot(makenw(gettable(hisc2),fit))
```

drawnetwork

Graphical interface for editing networks

Description

drawnetwork allows the user to specify a Bayesian network through a point and click interface.

Usage

```
drawnetwork(nw,df,prior,trylist=vector("list",size(nw)),
            unitscale=20,cexscale=8,
            arrowlength=.25,nocalc=FALSE,
            yr=c(0,350),xr=yr,...)
```

Arguments

nw	an object of class network to be edited.
df	a data frame used for learning the network, see network.
prior	a list containing parameter priors, generated by jointprior.
trylist	a list used internally for reusing learning of nodes, see maketrylist.
cexscale	a numeric passed to the plot method for network objects. Measures the scaled size of text and symbols.
arrowlength	a numeric passed to the plot method for network objects. Measures the length of the edges of the arrowheads.

<code>nocalc</code>	a logical. If TRUE, no learning procedure is called, see eg. <code>rnetwork</code> .
<code>unitscale</code>	a numeric passed to the plot method for network objects. Scale parameter for chopping off arrow heads.
<code>xr</code>	a numeric vector with two components containing the range on x-axis.
<code>yr</code>	a numeric vector with two components containing the range on y-axis.
<code>...</code>	additional plot arguments, passed to the plot method for network objects.

Details

To insert an arrow from node 'A' to node 'B', first click node 'A' and then click node 'B'. When the graph is finished, click 'stop'.

To specify that an arrow must not be present, press 'ban' (a toggle) and draw the arrow. This is shown as a red dashed arrow. It is possible to ban both directions between nodes. The ban list is stored with the network in the property `banlist`. It is a matrix with two columns. Each row is the 'from' node index and the 'to' node index, where the indices are the column number in the data frame.

Note that the network score changes as the network is re-learned whenever a change is made (unless `nocalc` is TRUE).

Value

A list with two elements that may be accessed using `getnetwork` and `gettrylist`. The elements are

<code>nw</code>	an object of class <code>network</code> with the final network.
<code>trylist</code>	an updated list used internally for reusing learning of nodes, see <code>maketrylist</code> .

Author(s)

Susanne Gammelgaard Bøttcher <alma@math.auc.dk>,
Claus Dethlefsen <dethlef@math.auc.dk>.

See Also

`network`

Examples

```
data(rats)
rats.nw <- network(rats)
rats.prior <- jointprior(rats.nw,12)
rats.nw <- getnetwork(learn(rats.nw,rats,
  rats.prior))

## Don't run: newrat <- getnetwork(drawnetwork(rats.nw,rats,
  rats.prior))
```

 jointprior

Calculates the joint prior distribution

Description

Given a network with a `prob` property for each node, derives the joint probability distribution. Then the quantities needed in the local master procedure for finding the local parameter priors are deduced.

Usage

```
jointprior(nw,N=NA,phiprior="bottcher",timetrace=FALSE)
```

Arguments

<code>nw</code>	an object of class <code>network</code> . Each node must have a <code>prob</code> property to describe the local probability distribution. The <code>prob</code> property is created using <code>prob</code> method for network objects, which is called by the <code>network</code> function.
<code>N</code>	an integer, which gives the size of the imaginary data base. If this is too small, NA's may be created in the output, resulting in errors in <code>learn</code> . If no <code>N</code> is given, the procedure tries to set a value as low as possible.
<code>phiprior</code>	a string, which specifies how the prior for <code>phi</code> is calculated. Either of the priors <code>phiprior="bottcher"</code> and <code>phiprior="heckerman"</code> can be used.
<code>timetrace</code>	a logical. If <code>TRUE</code> , prints some timing information on the screen.

Details

For the discrete part of the network, the joint probability distribution is calculated by multiplying together the local probability distributions. Then, `jointalpha` is determined by multiplying each entry in the joint probability distribution by the size of the imaginary data base `N`.

For the mixed part of the network, for each configuration of the discrete variables, the joint Gaussian distribution of the continuous variables is constructed and represented by `jointmu` (one row for each configuration of the discrete parents) and `jointsigma` (a list of matrices – one for each configuration of the discrete parents). The configurations of the discrete parents are ordered according to `findex`. The algorithm for constructing the joint distribution of the continuous variables is described in Schachter and Kenley (1989).

Then, `jointalpha`, `jointnu`, `jointrho`, `mu` and `jointphi` are deduced. These quantities are later used for deriving local parameter priors.

For each configuration `i` of the discrete variables,

$$\nu_i = \rho_i = \alpha_i$$

and

$$\phi_i = (\nu_i - 1)\Sigma_i$$

if `phiprior="bottcher"`, see Bøttcher(2001) and

$$\phi_i = \nu_i(\rho_i - 2)\Sigma_i/(\nu_i + 1)$$

if `phiprior="heckerman"`, see Heckerman, Geiger and Chickering (1995).

Value

A list with the following elements,

jointalpha	a table used in the local master procedure for discrete variables.
jointnu	a table used in the local master procedure for continuous variables.
jointrho	a table used in the local master procedure for continuous variables.
jointmu	a numeric matrix used in the local master procedure for continuous variables.
jointsigma	a list of numeric matrices (not used in further calculations).
jointphi	a list of numeric matrices used in the local master procedure for continuous variables.

Author(s)

Susanne Gammelgaard Bøttcher (alma@math.auc.dk),
Claus Dethlefsen (dethlef@math.auc.dk).

References

- Bøttcher, S.G. (2001). Learning Bayesian Networks with Mixed Variables, *Artificial Intelligence and Statistics 2001*, Morgan Kaufmann, San Francisco, CA, USA, pp. 149-156.
- Heckerman, D., Geiger, D. and Chickering, D. (1995). Learning Bayesian networks: The combination of knowledge and statistical data, *Machine Learning* **20**: 197-243.
- Shachter, R.D. and Kenley, C.R. (1989). Gaussian influence diagrams, *Management Science* **35**:527-550.

See Also

network, prob

Examples

```
data(rats)
rats.nw <- network(rats)
rats.prior <- jointprior(rats.nw,12)

## Don't run: savenet(rats.nw,file("rats.net"))
## Don't run: rats.nw <- readnet(file("rats.net"))
## Don't run: rats.nw <- prob(rats.nw,rats)
## Don't run: rats.prior <- jointprior(rats.nw,12)
```

Description

Updates the distributions of the parameters in the network, based on a prior network and data. Also, the network score is calculated.

Usage

```
learn (nw, df, prior=jointprior(nw),
      nodelist=1:size(nw),
      trylist=vector("list",size(nw)),
      timetrace=FALSE)
```

Arguments

<code>nw</code>	an object of class <code>network</code> .
<code>df</code>	a data frame used for learning the network, see <code>network</code> .
<code>prior</code>	a list containing parameter priors, generated by <code>jointprior</code> .
<code>nodelist</code>	a numeric vector of indices of nodes to be learned.
<code>trylist</code>	a list used internally for reusing learning of nodes, see <code>maketrylist</code> .
<code>timetrace</code>	a logical. If <code>TRUE</code> , prints some timing information on the screen.

Details

The procedure `learn` determines the master prior, local parameter priors and local parameter posteriors, see Böttcher (2001). It may be called on all nodes (default) or just a single node.

From the joint prior distribution, the marginal distribution of all parameters in the family consisting of the node and its parents can be determined. This is the master prior, see `localmaster`.

The local parameter priors are now determined by conditioning in the master prior distribution, see `conditional`. The hyperparameters associated with the local parameter prior distribution is attached to each node in the property `condprior`.

Finally, the local parameter posterior distributions are calculated (see `post`) and attached to each node in the property `condposterior`.

A so-called trylist is maintained to speedup the learning process. The trylist consists of a list of matrices for each node. The matrix for a given node holds previously evaluated parent configurations and the corresponding log-likelihood contribution. If a node with a certain parent configuration needs to be learned, it is checked, whether the node has already been learned. The previously learned nodes are given as input in the trylist parameter and is updated in the learning procedure.

When one or more nodes in a network have been learned, the network score is updated and attached to the network in the property `score`.

The learning procedure is called from various functions using the principle, that networks should always be updated with their score. Thus, e.g. `drawnetwork` keeps the network updated when the graph is altered.

Value

A list with two elements that may be accessed using `getnetwork` and `gettrylist`. The elements are

`nw` an object of class `network`, with the `condposterior` properties updated for the nodes. Also, the property `score` is updated and contains the network score. The contribution to the network score for each node is contained in the property `loglik` for each node.

`trylist` an updated list used internally for reusing learning of nodes, see `maketrylist`.

Author(s)

Susanne Gammelgaard Bøttcher (alma@math.auc.dk),
Claus Dethlefsen (dethlef@math.auc.dk).

References

Bøttcher, S.G. (2001). Learning Bayesian Networks with Mixed Variables, *Artificial Intelligence and Statistics 2001*, Morgan Kaufmann, San Francisco, CA, USA, pp. 149-156.

See Also

`networkfamily`, `jointprior`, `maketrylist`, `network`

Examples

```
data(rats)
fit      <- network(rats)
fit.prior <- jointprior(fit,12)
fit.learn <- learn(fit,rats,fit.prior,timetrace=TRUE)
fit.nw    <- getnetwork(fit.learn)
fit.learn2<- learn(fit,rats,fit.prior,trylist=gettrylist(fit.learn),
                  timetrace=TRUE)
```

`maketrylist`

Creates the full trylist

Description

For faster learning, a trylist is maintained as a lookup table for a given parent configuration of a node.

Usage

```
maketrylist(inetnw,data,prior=jointprior(network(data)),
            timetrace=FALSE)
```

Arguments

<code>initnw</code>	an object of class <code>network</code> , from which the search is started.
<code>data</code>	a data frame used for learning the network, see <code>network</code> .
<code>prior</code>	a list containing parameter priors, generated by <code>jointprior</code> .
<code>timetrace</code>	a logical. If <code>TRUE</code> , prints some timing information on the screen.

Details

This procedure is included for illustrative purposes. For each node in the network, all possible parent configurations are created and learned. The result is called a trylist. To create the full trylist is very time-consuming, and a better choice is to maintain a trylist while searching and indeed this is automatically done. The trylist is given as output to all functions that call the learning procedure and can be given as an argument.

Value

A list with one element per node in the network. In the list, element i is a matrix with two columns: a string with the indices of the parent nodes, separated by ":", and a numeric with the log-likelihood contribution of the node given the parent configuration. Whenever learning is performed of a node given a parent configuration, the trylist is consulted to yield faster learning, especially useful when using `autosearch` or `heuristic`.

Author(s)

Susanne Gammelgaard Bøttcher (alma@math.auc.dk),
Claus Dethlefsen (dethlef@math.auc.dk).

See Also

`networkfamily`, `autosearch` `heuristic`

Examples

```
data(rats)
rats.nw <- network(rats)
rats.pr <- jointprior(rats.nw,12)
rats.nw <- getnetwork(learn(rats.nw,rats,rats.pr))
rats.tr <- maketrylist(rats.nw,rats,rats.pr)

rats.hi <- getnetwork(heuristic(rats.nw,rats,rats.pr,
  trylist=rats.tr))
```

network

Bayesian network data structure

Description

A Bayesian network is represented as an object of class `network`. Methods for printing and plotting are defined.

Usage

```
network(df, specifygraph=FALSE, inspectprob=FALSE,
        doprob=TRUE, yr=c(0, 350), xr=yr)
## S3 method for class 'network':
print(x, filename=NA, condposterior=FALSE,
      condprior=FALSE, ...)
## S3 method for class 'network':
plot(x, arrowlength=.25,
      notext=FALSE,
      sscale=7, showban=TRUE, yr=c(0, 350), xr=yr,
      unitscale=20, cexscale=8, ...)
```

Arguments

<code>df</code>	a data frame, where the columns define the variables. A continuous variable should have type <code>numeric</code> and discrete variables should have type <code>factor</code> .
<code>specifygraph</code>	a logical. If <code>TRUE</code> , provides a call to <code>drawnetwork</code> to interactively specify a directed acyclic graph and possibly a ban list (see below).
<code>inspectprob</code>	a logical. If <code>TRUE</code> , provides a plot of the graph and possibility to inspect the calculated probability distribution by clicking on the nodes.
<code>doprob</code>	a logical. If <code>TRUE</code> , do not calculate a probability distribution. Used for example in <code>rnetwork</code> .
<code>x</code>	an object of class <code>network</code> .
<code>filename</code>	a string or <code>NA</code> . If not <code>NA</code> , output is printed to a file.
<code>condprior</code>	a logical. If <code>TRUE</code> , the conditional prior is printed, see <code>conditional</code> .
<code>condposterior</code>	a logical. If <code>TRUE</code> , the conditional posterior is printed, see <code>learn</code> .
<code>sscale</code>	a numeric. The nodes are initially placed on a circle with radius <code>sscale</code> .
<code>unitscale</code>	a numeric. Scale parameter for chopping off arrow heads.
<code>cexscale</code>	a numeric. Scale parameter to set the size of the nodes.
<code>arrowlength</code>	a numeric containing the length of the arrow heads.
<code>xr</code>	a numeric vector with two components containing the range on x-axis.
<code>yr</code>	a numeric vector with two components containing the range on y-axis.
<code>notext</code>	a logical. If <code>TRUE</code> , no text is displayed in the nodes on the plot.
<code>showban</code>	a logical. If <code>TRUE</code> , banned arrows are shown in red.
<code>...</code>	additional plot arguments, passed to <code>plot.node</code> .

Value

The `network` creator function returns an object of class `network`, which is a list with the following elements (properties),

<code>nodes</code>	a list of objects of class <code>node</code> . If <code>doprob</code> is <code>TRUE</code> , the nodes are given the property <code>prob</code> which is the initial probability distribution used by <code>jointprior</code> .
<code>n</code>	an integer containing the number of nodes in the network.
<code>discrete</code>	a numeric vector of indices of discrete nodes.
<code>continuous</code>	a numeric vector of indices of continuous nodes.
<code>banlist</code>	a numeric matrix with two columns. Each row contains the indices <code>i -> j</code> of arrows that may not be allowed in the directed acyclic graph.
<code>score</code>	a numeric added by <code>learn</code> and is the log network score.
<code>relscore</code>	a numeric added by <code>nwfsort</code> and is the relative network score – compared with the best network in a network family.

Author(s)

Susanne Gammelgaard Bøttcher <alma@math.auc.dk>,
Claus Dethlefsen <dethlef@math.auc.dk>.

See Also

`networkfamily`, `node`, `rnetwork`, `learn`, `drawnetwork`, `jointprior`, `heuristic`, `nwequal`

Examples

```
A <- factor(rep(c("A1", "A2"), 50))
B <- factor(rep(rep(c("B1", "B2"), 25), 2))
thisnet <- network( data.frame(A,B) )

set.seed(109)
sex      <- gl(2,4,label=c("male", "female"))
age      <- gl(2,2,8)
yield    <- rnorm(length(sex))
weight   <- rnorm(length(sex))
mydata   <- data.frame(sex,age,yield,weight)
mynw     <- network(mydata)

# adjust prior probability distribution
localprob(mynw,"sex")   <- c(0.4,0.6)
localprob(mynw,"age")   <- c(0.6,0.4)
localprob(mynw,"yield") <- c(2,0)
localprob(mynw,"weight")<- c(1,0)

print(mynw)
plot(mynw)

prior <- jointprior(mynw)
```

```

mynw <- getnetwork(learn(mynw,mydata,prior))
thebest <- getnetwork(autosearch(mynw,mydata,prior))

print(mynw,condposterior=TRUE)

## Don't run: savenet(mynw,file("yield.net"))

```

networkfamily

Generates and learns all networks for a set of variables.

Description

Method for generating and learning all networks that are possible for a given set of variables. These may be plotted or printed. Also, functions for sorting according to the network score (see `nwfsort`) and for making a network family unique (see the `unique` method for `networkfamily` objects) are available.

Usage

```

networkfamily(data,nw=network(data), prior=jointprior(nw),
              trylist=vector("list",size(nw)), timetrace=TRUE)
## S3 method for class 'networkfamily':
print(x,...)
## S3 method for class 'networkfamily':
plot(x,layout=<<see below>>,
     cexscale=5,arrowlength=0.1,sscale=7,...)

```

Arguments

<code>nw</code>	an object of class <code>network</code> . This should be the empty network for the set of variables.
<code>data</code>	a data frame used for learning the network, see <code>network</code> .
<code>prior</code>	a list containing parameter priors, generated by <code>jointprior</code> .
<code>trylist</code>	a list used internally for reusing learning of nodes, see <code>maketrylist</code> .
<code>timetrace</code>	a logical. If <code>TRUE</code> , prints some timing information on the screen.
<code>x</code>	an object of class <code>networkfamily</code> .
<code>layout</code>	a numeric two dimensional vector with the number of plots in the rows and columns of each plotting page. Default set to <code>rep(min(1+floor(sqrt(length(x))),5),2)</code> .
<code>cexscale</code>	a numeric. A scaling parameter to set the size of the nodes.
<code>arrowlength</code>	a numeric, which gives the length of the arrow heads.
<code>sscale</code>	a numeric. The nodes are initially placed on a circle with radius <code>sscale</code> .
<code>...</code>	additional plot arguments passed to the plot method for network objects.

Details

`networkfamily` generates and learns all possible networks with the nodes given as in the initial network `nw`. This is done by successively trying to generate the networks with all possible arrows to/from each node (see `addarrows`). If there is a ban list present in `nw` (see `network`), then this is respected, as are the restrictions described in `insert`.

After generation of all possible networks, a test for cycles (see `cycletest`) is performed and only networks with directed acyclic graphs are returned.

Value

The function `networkfamily` returns a list with two components,

`nw` an object of class `networkfamily`.

`trylist` an updated list used internally for reusing learning of nodes, see `maketrylist`.

Note

Generating all possible networks can be *very* time consuming!

Author(s)

Susanne Gammelgaard Bøttcher <alma@math.auc.dk>,

Claus Dethlefsen <dethlef@math.auc.dk>.

See Also

`network`, `genlatex`, `heuristic`, `nwfsort`, `unique.networkfamily`, `elementin`, `addarrows`, `cycletest`

Examples

```
data(rats)
allrats <- getnetwork(networkfamily(rats))
plot(allrats)
print(allrats)
```

Description

Various extraction/replacement functions for networks

Usage

```

modelstring(x)
makenw(tb,template)
as.network(nwstring,template)
size(x)
banlist(x)
banlist(x) <- value
getnetwork(x)
gettrylist(x)

```

Arguments

<code>x</code>	an object of class <code>network</code> .
<code>tb</code>	a table output from <code>autosearch</code> or <code>heuristic</code> in the list property table. Can be translated into a <code>networkfamily</code> .
<code>template</code>	an object of class <code>network</code> with the same nodes as the networks described in the table <code>tb</code> .
<code>nwstring</code>	a string representing the network.
<code>value</code>	a numeric matrix with two columns. Each row contains the indices <code>i -> j</code> of arrows that may not be allowed in the directed acyclic graph.

Details

The string representation of a network is a minimal size representation to speed up calculations. The functions `modelstring`, `as.network` and `makenw` converts between the string representation and network objects.

`size` extracts the number of nodes in a network object.

`banlist` extracts the banlist from a network object.

`getnetwork` and `gettrylist` are accessor function that extracts a network object or trylist from the result from `autosearch`, `heuristic`, `learn`, `perturb`, `networkfamily`, `drawnetwork`.

node

Representation of nodes

Description

An important part of a network is the list of nodes. The nodes summarize the local properties of a node, given the parents of the node.

Usage

```

node (idx,parents,type="discrete",name=paste(idx),
      levels=2,levelnames=paste(1:levels),position=c(0,0))
## S3 method for class 'node':
print (x,filename=NA,condposterior=TRUE,condprior=TRUE,...)

```

```
## S3 method for class 'node':
plot (x,cexscale=10,notext=FALSE,...)
nodes(nw)
value <- nodes(nw)
```

Arguments

<code>x</code>	an object of class <code>node</code> .
<code>parents</code>	a numeric vector with indices of the parents of the node.
<code>idx</code>	an integer, which gives the index of the node (the column number of the corresponding data frame).
<code>type</code>	a string, which gives the type of the node. Either "discrete" (for factors) or "continuous" (for numeric).
<code>name</code>	a string, which gives the name used when plotting and printing. Defaults to the column name in the data frame.
<code>levels</code>	an integer. If <code>type</code> is "discrete", this is the number of levels for the discrete variable.
<code>levelnames</code>	if <code>type</code> is "discrete", this is a vector of strings (same length as <code>levels</code>) with the names of the levels. If <code>type</code> is "continuous", the argument is ignored.
<code>position</code>	a numeric vector with coordinates where the node should appear in the plot. Usually set by <code>network</code> and <code>drawnetwork</code> .
<code>nw</code>	an object of class <code>network</code> .
<code>value</code>	a list of elements of class <code>node</code> .
<code>filename</code>	a string or NA. If not NA, output is printed to a file.
<code>condprior</code>	a logical. If TRUE, the conditional prior is printed, see <code>conditional</code> .
<code>condposterior</code>	a logical. If TRUE, the conditional posterior is printed, see <code>learn</code> .
<code>cexscale</code>	a numeric. Scale parameter to set the size of the nodes.
<code>notext</code>	a logical. If TRUE, no text is displayed in the nodes on the plot.
<code>...</code>	additional plot arguments.

Details

The operations on a node are typically done when operating on a `network`, so these functions are not to be called directly.

When a `network` is created with `network`, the nodes in the `nodelist` are created using the `node` procedure.

Local probability distributions are added as the property `prob` to each node using `prob.node`. If the node is continuous, this is a numeric vector with the conditional variance and the conditional regression coefficients arising from a regression on the continuous parents, using data. If the node has discrete parents, `prob` is a matrix with a row for each configuration of the discrete parents. If the node is discrete, `prob` is a multiway array which gives the conditional probability distribution for each configuration of the discrete parents. The generated `prob` can be replaced to match the prior information available.

`nodes` gives the list of nodes of a `network`. `localprob` gives the probability distribution for each node in the `network`.

Value

The node creator function returns an object of class `node`, which is a list with the following elements (properties),

<code>idx</code>	an integer. A unique index for this node. It MUST correspond to the column index of the variable in the data frame.
<code>name</code>	a string. The printed name of the node.
<code>type</code>	a string. Either "continuous" or "discrete".
<code>levels</code>	an integer. If the node is of type "discrete", this integer is the number of levels of the node.
<code>levelnames</code>	if <code>type</code> is "discrete", this is a vector of strings (same length as <code>levels</code>) with the names of the levels. If <code>type</code> is "continuous", the node does not have this property.
<code>parents</code>	a vector of indices of the parents to this node. It is best to manage this vector using the <code>insert</code> function.
<code>prob</code>	a numeric vector, matrix or multiway array, giving the initial probability distribution. If the node is discrete, <code>prob</code> is a multiway array. If the node is continuous, <code>prob</code> is a matrix with one row for each configuration of the discrete parents, reducing to a vector if the node has no discrete parents.
<code>condprior</code>	a list, generated by <code>conditional</code> giving the parameter priors deduced from <code>jointprior</code> using the master prior procedure (see <code>localmaster</code>).
<code>condposterior</code>	a list, which gives the parameter posteriors obtained from <code>learnnode</code> .
<code>loglik</code>	a numeric giving the log likelihood contribution for this node, calculated in <code>learnnode</code> .
<code>simprob</code>	a numeric vector, matrix or multiway array similar to <code>prob</code> , added by <code>makesimprob</code> and used by <code>rnetwork</code> .

Author(s)

Susanne Gammelgaard Bøttcher (alma@math.auc.dk),
 Claus Dethlefsen (dethlef@math.auc.dk).

`numbermixed`

The number of possible networks

Description

Calculates the number of different directed acyclic graphs for a set of discrete and continuous nodes.

Usage

```
numbermixed(nd, nc)
```

Arguments

`nd` an integer, which gives the number of discrete nodes.
`nc` an integer, which gives the number of continuous nodes.

Details

No arrows are allowed from continuous nodes to discrete nodes. Cycles are not allowed. The number of networks is given by Bøttcher (2003), using the result in Robinson (1977). When $nd+nc>15$, the procedure is quite slow.

Value

A numeric containing the number of directed acyclic graphs with the given node configuration.

Author(s)

Susanne Gammelgaard Bøttcher (alma@math.auc.dk),
Claus Dethlefsen (dethlef@math.auc.dk).

References

Bøttcher, S.G. (2003). Learning Conditional Gaussian Networks.
<http://www.math.auc.dk/~alma>. Aalborg University, 2003.
Robinson, R.W. (1977). Counting unlabeled acyclic digraphs, *Lecture Notes in Mathematics*, 622: *Combinatorial Mathematics V* pp. 239-273.

Examples

```
numbermixed(2,2)
## Don't run: numbermixed(5,10)
```

nwfsort

Sorts a list of networks

Description

According to the `score` property of the networks in a network family, the networks are sorted and the relative score, i.e. the score of a network relative to the highest score, is attached to each network as the `relscore` property.

Usage

```
nwfsort(nwf)
```

Arguments

`nwf` an object of class `networkfamily`.

Author(s)

Susanne Gammelgaard Bøttcher <alma@math.auc.dk>,
 Claus Dethlefsen <dethlef@math.auc.dk>.

perturb

Perturbs a network

Description

Randomly insert/delete/turn arrows to obtain another network.

Usage

```
perturb(nw,data,prior,degree=size(nw),trylist=vector("list",size(nw)),
        nocalc=FALSE,timetrace=TRUE)
```

Arguments

nw	an object of class <code>network</code> , from which arrows are added/removed/turned.
data	a data frame used for learning the network, see <code>network</code> .
prior	a list containing parameter priors, generated by <code>jointprior</code> .
degree	an integer, which gives the number of attempts to randomly insert/remove/turn an arrow.
trylist	a list used internally for reusing learning of nodes, see <code>maketrylist</code> .
nocalc	a logical. If <code>TRUE</code> no learning procedure is called, see eg. <code>rnetwork</code> .
timetrace	a logical. If <code>TRUE</code> , prints some timing information on the screen.

Details

Given the initial network, a new network is constructed by randomly choosing an action: remove, turn, add. After the action is chosen, we choose randomly among all possibilities of that action. If there are no possibilities, the unchanged network is returned.

Value

A list with two elements that may be accessed using `getnetwork` and `gettrylist`. The elements are

nw	an object of class <code>network</code> with the generated network.
trylist	an updated list used internally for reusing learning of nodes, see <code>maketrylist</code> .

Author(s)

Susanne Gammelgaard Bøttcher (alma@math.auc.dk),
Claus Dethlefsen (dethlef@math.auc.dk).

Examples

```
set.seed(200)
data(rats)
fit      <- network(rats)
fit.prior <- jointprior(fit)
fit      <- getnetwork(learn(fit,rats,fit.prior))
fit.new  <- getnetwork(perturb(fit,rats,fit.prior,degree=10))

data(ksl)
ksl.nw   <- network(ksl)
ksl.rand <- getnetwork(perturb(ksl.nw,nocalc=TRUE,degree=10))
plot(ksl.rand)
```

prob

Local probability distributions

Description

Methods for accessing or changing the local probability distributions and for accessing the local prior and posterior distributions

Usage

```
prob(x,df,...)

## S3 method for class 'node':
prob(x,df,nw,...)
## S3 method for class 'network':
prob(x,df,...)

localprob(nw)
value <- localprob(nw,name)

localprior(node)
localposterior(node)
```

Arguments

x an object of class node or network.

<code>df</code>	a data frame, where the columns define the variables. A continuous variable should have type <code>numeric</code> and discrete variables should have type <code>factor</code> .
<code>nw</code>	an object of class <code>network</code> .
<code>node</code>	an object of class <code>node</code> .
<code>name</code>	a string, which gives the node name.
<code>...</code>	additional arguments for specific methods.

Details

The `prob` methods add local probability distributions to each node. If the node is continuous, this is a numeric vector with the conditional variance and the conditional regression coefficients arising from a regression on the continuous parents, using data. If the node has discrete parents, `prob` is a matrix with a row for each configuration of the discrete parents. If the node is discrete, `prob` is a multiway array which gives the conditional probability distribution for each configuration of the discrete parents. The generated `prob` can be replaced to match the prior information available. `localprob` returns the probability distribution for each node in the network. In a learned network, the local prior and posterior can be accessed for each node using `localprior` and `localposterior`.

Author(s)

Susanne Gammelgaard Bøttcher (alma@math.auc.dk),
Claus Dethlefsen (dethlef@math.auc.dk).

<code>readnet</code>	<i>Reads/saves .net file</i>
----------------------	------------------------------

Description

Reads/saves a Bayesian network specification in the `.net` language (see <http://developer.hugin.com/documentation/net/>).

Usage

```
readnet(con=file("default.net"))
savenet(nw, con=file("default.net"))
```

Arguments

<code>con</code>	a connection.
<code>nw</code>	an object of class <code>network</code> .

Details

`readnet` reads only the structure of a network, i.e. the directed acyclic graph.
`savenet` exports the `prob` property for each node in the network object along with the network structure defined by the parents of each node.

Value

`readnet` creates an object of class `network` with the nodes specified as in the `.net` connection. The network has not been learned and the nodes do not have `prob` properties (see `prob.network`).
`savenet` writes the object to the connection.

Note

The call to `readnet(savenet(network))` is *not* the identity function as information is thrown away in both `savenet` and `readnet`.

Author(s)

Susanne Gammelgaard Bøttcher <alma@math.auc.dk>,
Claus Dethlefsen <dethlef@math.auc.dk>.

See Also

`network`

Examples

```
data(rats)
nw <- network(rats)
## Don't run: savenet(nw, file("default.net"))
## Don't run: nw2 <- readnet(file("default.net"))
## Don't run: nw2 <- prob(nw2, rats)
```

`score`

Network score

Description

Usage

```
score(x,...)

## S3 method for class 'node':
score (x,...)
## S3 method for class 'network':
score (x,...)
```

Arguments

`x` an object of class `node` or `network`.
`...` additional arguments for specific methods.

Value

For networks, the log network score is returned. For nodes, the contribution to the log network score is returned.

Author(s)

Susanne Gammelgaard Bøttcher <alma@math.auc.dk>,
Claus Dethlefsen <dethlef@math.auc.dk>.

`rnetwork`

Simulation of data sets with a given dependency structure

Description

Given a network with nodes having the `simprob` property, `rnetwork` simulates a data set.

Usage

```
rnetwork(nw, n=24, file=" ")
```

Arguments

`nw` an object of class `network`, where each node has the property `simprob` (see `makesimprob`).
`n` an integer, which gives the number of cases to simulate.
`file` a string. If non-empty, the data set is stored there.

Details

The variables are simulated one at a time in an order that ensures that the parents of the node have already been simulated. For discrete variables a multinomial distribution is used and for continuous variables, a Gaussian distribution is used, according to the `simplprob` property in each node.

Value

A data frame with one row per case. If a file name is given, a file is created with the data set.

Author(s)

Susanne Gammelgaard Bøttcher <alma@math.auc.dk>,
Claus Dethlefsen <dethlef@math.auc.dk>.

Examples

```
A <- factor(NA, levels=paste("A", 1:2, sep=""))
B <- factor(NA, levels=paste("B", 1:3, sep=""))
c1 <- NA
c2 <- NA
df <- data.frame(A,B,c1,c2)

nw <- network(df,doprob=FALSE) # doprob must be FALSE
nw <- makesimplprob(nw)      # create simplprob properties

set.seed(944)
sim <- rnetwork(nw,n=100)    # create simulated data frame
```

Paper III
Prediction of the Insulin Sensitivity
Index using Bayesian Networks

Susanne G. Bøttcher and Claus Dethlefsen

Prediction of the Insulin Sensitivity Index using Bayesian Networks

Susanne G. Bøttcher

Aalborg University, Denmark

Claus Dethlefsen

Aalborg University, Denmark

Abstract.

The insulin sensitivity index (S_I) can be used in assessing the risk of developing type 2 diabetes. An intravenous study is used to determine S_I using Bergmans minimal model. However, an intravenous study is time consuming and expensive and therefore not suitable for large scale epidemiological studies. In this paper we learn the parameters and structure of several Bayesian networks relating measurements from an oral glucose tolerance test to the insulin sensitivity index determined from an intravenous study on the same individuals. The networks can then be used in prediction of S_I from an oral glucose tolerance test instead of an intravenous study. The methodology is applied to a dataset with 187 patients. We find that the S_I values from this study are highly correlated to the S_I values determined from the intravenous study.

1 Introduction

Type 2 diabetes is a clinical syndrome that can result from several disorders that interfere with insulin secretion and/or the ability of the target tissues to respond to insulin. Martin, Warram, Krolewski, Bergman, Soeldner and Kahn (1992) found evidence in a 25 year follow-up study that insulin sensitivity index (S_I) can be used to predict the development of type 2 diabetes up to a decade before diagnosis. Assessment of S_I is by Bergmans minimal model, see Bergman et al. (1979), which is based on data from an intravenous glucose tolerance test (IVGTT). In the minimal model, the glucose and insulin kinetics are separately described by two sets of differential equations. The parameters in the model are traditionally estimated by a non-linear weighted least squares estimation technique, see for example Pacini and Bergman (1986). From these parameters, S_I can be determined.

However, an IVGTT is time consuming and expensive and therefore not suitable for large scale epidemiological studies. Interest is therefore in developing a

method to assess the insulin sensitivity index from an oral glucose tolerance test (OGTT).

In Drivsholm, Hansen, Urhammer, Palacios, Vølund, Borch-Johnsen and Pedersen (2003), multiple linear regression is used to derive predictive values of S_I from measurements from an OGTT. These are compared with the values of S_I obtained from an IVGTT and calculated using Bergmans minimal model. The results show that it is possible to predict estimates of S_I , which are highly correlated to IVGTT-derived S_I for subjects with normal glucose tolerance.

In this paper, we express the relation between the observed variables in a Bayesian network. We try different approaches of establishing a Bayesian network, which can be used to predict S_I from measurements from an OGTT. We learn the parameters and structure of a Bayesian network from a training data set, where all patients underwent both an IVGTT and an OGTT. Bergmans minimal model were used to determine S_I from the IVGTT. We then calculate the predictive value of S_I from the Bayesian network and compare it with the value of S_I obtained from the IVGTT.

Like the multiple linear regression approach, the Bayesian network approach gives predictions of S_I that are highly correlated to IVGTT-derived S_I for subjects with normal glucose tolerance. In addition, the complex dependency structure between the variables is modeled adequately. Further, using Bayesian networks makes it possible to incorporate any prior information available, *e.g.* the physiological understanding of the problem or results from previous studies.

2 Data

In this paper we consider 187 non-diabetic glucose tolerant subjects, with one parent having diabetes. All the subjects underwent a 75 gram frequently sampled OGTT. In such a test, the subject drinks 75 gram fluent glucose, after a 12 hour overnight fast. Venous blood samples are then drawn at 10, 5 and 0 minutes before the OGTT and after the start of the OGTT, at 10, 20, 30, 40, 50, 60, 75, 90, 105, 120, 140, 160, 180, 210 and 240 minutes. From these blood samples, the glucose and insulin concentrations are determined.

Within one week after the OGTT examination, all subjects underwent a tolbutamide modified frequently sampled IVGTT. In an IVGTT, glucose is injected directly into the venous. Blood samples are drawn at 10, 5 and 0 minutes before

the injection and frequently up until 180 minutes after the injection. At 20 minutes, a bolus of tolbutamide is injected to elicit secondary pancreatic beta cell response. In the time between the two examinations, the subjects were asked not to change their lifestyle. The insulin sensitivity index (S_I) was for each subject calculated from the observations in the IVGTT using Bergmans minimal model and estimated by a non-linear weighted least squares estimation technique, as described Pacini and Bergman (1986).

Other variables in the study are age, sex, weight, height, waist circumference, hip circumference, fat mass and information on physical activity. From the weight and height, the body mass index (BMI) can be calculated.

3 Bayesian Networks

We perform the analysis using Bayesian networks for discrete and continuous variables in which the joint distribution of all the variables are conditional Gaussian (CG), see Lauritzen (1992).

3.1 Bayesian Networks with Mixed Variables

Let $D = (V, E)$ be a Directed Acyclic Graph (DAG), where V is a finite set of nodes and E is a finite set of directed edges (arrows) between the nodes. The DAG defines the structure of the Bayesian network. To each node $v \in V$ in the graph corresponds a random variable X_v . The set of variables associated with the graph D is then $X = (X_v)_{v \in V}$. Often, we do not distinguish between a variable X_v and the corresponding node v . To each node v with parents $\text{pa}(v)$, a local probability distribution, $p(x_v | x_{\text{pa}(v)})$ is attached. The set of local probability distributions for all variables in the network is \mathcal{P} . A Bayesian network for a set of random variables X is then the pair (D, \mathcal{P}) .

The possible lack of directed edges in D encodes conditional independencies between the random variables X through the factorization of the joint probability distribution,

$$p(x) = \prod_{v \in V} p(x_v | x_{\text{pa}(v)}). \quad (1)$$

Here, we allow Bayesian networks with both discrete and continuous variables, as treated in Lauritzen (1992), so the set of nodes V is given by $V = \Delta \cup \Gamma$,

where Δ and Γ are the sets of discrete and continuous nodes, respectively. The set of variables X can then be denoted $X = (X_v)_{v \in V} = (I, Y) = ((I_\delta)_{\delta \in \Delta}, (Y_\gamma)_{\gamma \in \Gamma})$, where I and Y are the sets of discrete and continuous variables, respectively. For a discrete variable, δ , we let \mathcal{I}_δ denote the set of levels.

To ensure availability of exact local computation methods, we do not allow discrete variables to have continuous parents. The joint probability distribution then factorizes into a discrete part and a mixed part, so

$$p(x) = p(i, y) = \prod_{\delta \in \Delta} p(i_\delta | i_{\text{pa}(\delta)}) \prod_{\gamma \in \Gamma} p(y_\gamma | y_{\text{pa}(\gamma)}, i_{\text{pa}(\gamma)}).$$

A method for estimating the parameters and learning the dependency structure of a conditional Gaussian networks with mixed variables is presented in Bøttcher (2001) and implemented in the software package `deal`, see Bøttcher and Dethlefsen (2003).

3.2 Parameter and Structure Learning

To estimate the parameters in the network and to find the structure of the network, we use a Bayesian approach. So, considering the parameters, we encode our uncertainty about θ in a prior distribution $p(\theta)$, use data d to update this distribution, i.e. learn the parameters, and hereby obtain the posterior distribution $p(\theta|d)$ by using Bayes' theorem,

$$p(\theta|d) = \frac{p(d|\theta)p(\theta)}{p(d)}, \quad \theta \in \Theta. \quad (2)$$

Here, Θ is the parameter space, d is a random sample of size n from the probability distribution $p(x|\theta)$ and $p(d|\theta)$ is the joint probability distribution of d , also called the likelihood of θ . As prior parameter distributions we use the Dirichlet distribution for the discrete variables and the Gaussian inverse-Gamma distribution for the continuous variables. These distributions are conjugate to observations from the respective distributions and this ensures simple calculations of the posterior distributions.

Now, to learn the structure of the network, we calculate the posterior probability of the DAG, $p(D|d)$, which from Bayes' theorem is given by

$$p(D|d) = \frac{p(d|D)p(D)}{p(d)},$$

where $p(d|D)$ is the likelihood of D and $p(D)$ is the prior probability of D . As the normalizing constant $p(d)$ does not depend upon structure, another measure, which gives the relative probability, is

$$p(D, d) = p(d|D)p(D).$$

We use the above measure and refer to it as the *network score*. For simplicity, we choose to let $p(D)$ be the same for all DAGs, so we are only interested in calculating the likelihood $p(d|D)$. It is given as

$$p(d|D) = \int_{\theta \in \Theta} p(d|\theta, D)p(\theta|D)d\theta,$$

and we see that it, besides the likelihood of the parameters, also involves the prior distribution over the parameters, $p(\theta|D)$. This means that we for each possible DAG have to specify a prior distribution for the parameters. In the papers Heckerman et al. (1995) and Geiger and Heckerman (1994) an automated method for doing this in respectively the purely discrete and the purely Gaussian case is developed. In Bøttcher (2001) this method is extended to the mixed case. With this method, the parameter priors for all possible networks can be deduced from one joint parameter prior, called a master prior. To specify this master prior, we only have to specify a prior Bayesian network, *i.e.* a prior DAG and a prior probability distribution, together with a measure of how confident we are in the prior network. With a few assumptions, the network score is obtainable in closed form.

If many DAGs are possible, it is computational infeasible to calculate the network score for all DAGs. In this situation it is necessary to use some kind of search strategy to find the DAG with the highest score, see e.g. Cooper and Herskovits (1992). In this paper we use a search strategy called greedy search. In greedy search we compare DAGs that differ only by a single arrow, either added, removed or reversed. The change that increases the network score the most is selected and the search is continued from this new DAG.

4 Inference

Having established a Bayesian network for a set of random variables, this represents the knowledge we, at this stage, have about these variables. When information on some or all of the variables becomes available, we can use this

“knowledge base” to make inference about the unobserved variables in the network.

Inference in Bayesian networks is performed using Bayes’ theorem. Consider a network for a set of random variables X and assume that some of the variables, B , are observed and the rest, A , are not. We can then, by using Bayes’ theorem, calculate the conditional distribution of A given B as

$$p(A|B) \propto p(B|A)p(A).$$

Thus $p(A)$ is the prior distribution of A , *i.e.* the distribution of A before we observe B , $p(B|A)$ is the likelihood of A and $p(A|B)$ is the posterior distribution of A , *i.e.* the distribution of A , when we have observed B . Generally, finding these distributions is computationally heavy as it involves calculating huge joint distributions, especially if there are many variables in the network. Therefore efficient methods of implementing Bayes’ theorem are being used. These implementations use the fact that the joint probability distribution of all the variables in a network factorizes according to (1). The marginal or conditional distributions of interest can then be found by a series of local computations, involving only some of the variables at a time, see *e.g.* Cowell et al. (1999) for a thorough treatment of these methods.

So having observed some of the variables in a network, we can use this new evidence to calculate the posterior distribution of any unobserved variable X_v , given the evidence. Notice that we do not need to observe all the other variables before calculating the posterior distribution, as we can update the prior distribution of X_v with any information available. Of course, the more information we have, the better the posterior distribution is determined. However, not all information will have an impact on the posterior distribution of a variable X_v . Consider the following result. A node v is conditional independent on the rest of the nodes in the network, given the *Markov blanket* of v , $\text{bl}(v)$, *i.e.*

$$v \perp\!\!\!\perp V \setminus v | \text{bl}(v).$$

The Markov blanket of v is the set of v ’s parents, children and children’s parents, *i.e.*

$$\text{bl}(v) = \text{pa}(v) \cup \text{ch}(v) \cup \{w : \text{ch}(w) \cap \text{ch}(v) \neq \emptyset\},$$

where $\text{pa}(v)$ is the parents of v and $\text{ch}(v)$ is the children of v , see Cowell et al. (1999). So if all the variables in the Markov blanket are observed, we do not get further information about the distribution of X_v by observing the variables outside the Markov blanket. But if we have *not* observed all the variables in the Markov blanket, then observing some variable outside the Markov blanket, can influence the posterior distribution of X_v .

5 Results

We will now present the results obtained.

5.1 Preliminaries

In the present study, 187 subjects without known diabetes underwent both an OGTT and an IVGTT. In the OGTT, measurements were recorded of plasma glucose (G) and serum insulin levels (I) at time points 10, 5 and 0 before intake of 75 gram glucose and at 10, 20, 30, 40, 50, 60, 75, 90, 105, 120, 140, 160, 180, 210 and 240 minutes after the intake.

In this analysis, the observations to time 10, 5 and 0 before the glucose intake are, for both insulin and glucose, averaged and represented by the corresponding observation to time 0. Further, based on previous results, see Drivsholm et al. (2003), we use the logarithm of the insulin sensitivity index $\log S_I$ instead of S_I and we also include the sex of the patient and the body mass index (BMI) in the models. Sex is a binary variable, but we choose to treat it as a continuous variable. This has the effect that the variance is assumed equal for male and female observations, whereas the means can differ. If sex is treated as a discrete variable, the data is split into two groups with a parameter set for each group. and we have found that we do not have enough data to support this. Consider for example the simple case, where the only parent to $\log S_I$ is sex. If sex is treated as a continuous variable, the distribution of $\log S_I$ is given as

$$(\log S_I | \text{sex}) \sim \mathcal{N}(m + \beta \text{sex}, \sigma^2).$$

None of the parameters m , β and σ^2 depend on sex, but the mean is m if sex is 0 and $m + \beta$ if sex is 1. If sex is treated as a discrete variable, the distribution of $\log S_I$ is

$$(\log S_I | \text{sex}) \sim \mathcal{N}(m_{\text{sex}}, \sigma_{\text{sex}}^2),$$

i.e. both the mean and the variance depends on sex.

In the following we will try different ways of establishing a Bayesian network, which can be used to predict $\log S_I$ from measurements from an OGTT and from BMI and sex. So the networks we will consider in the following, only contain continuous variables. Notice that, when using the theory presented for mixed networks on networks with only continuous variables, it coincides with theory developed for purely continuous networks, see Bøttcher (2001). To learn

the parameters and structure of a Bayesian network, we use the software package `deal`, see Bøttcher and Dethlefsen (2003). The package is written for R, see Ihaka and Gentleman (1996).

To validate the models, we split the dataset into a subset with 140 subjects, used as training data, and a subset with 47 subjects, used as validation data. For each model, we use `deal` with the training data to learn the parameters and structure of the Bayesian network. The posterior parameter distribution of $\log S_I$ is used to derive point estimates of the parameters. For the Gaussian parameters, we use the mean of the posterior and for the gamma distributed parameter, we use the mode of the posterior. These point estimates are then transferred to Hugin (www.hugin.com). For each subject in both the training data and the validation data, the conditional distribution of $\log S_I$ is calculated given the observations from the OGTT using Hugin. In the following, we call this distribution the predictive distribution of $\log S_I$. Notice, however, that if a fully Bayesian approach had been used, the predictive distribution for one subject is

$$p(\log S_I|d) = \int_{\theta \in \Theta} p(\log S_I|d, \theta)p(\theta)d\theta,$$

where d denotes the subjects OGTT measurements and θ are the parameters. This distribution is a t distribution with degrees of freedom increasing with, among other numbers, the number of subjects in the training dataset. In this study we have 140 subjects and we find that the error using a Gaussian distribution instead, is very small.

The predictive distribution is then, for each subject, compared with the corresponding $\log S_I$ value determined from the IVGTT in the following way. For each subject we use the predictive distribution to calculate the 95%’s credibility intervals $\mu \pm 1.96 \cdot \sigma$, where 1.96 is the 97.5%’s quantile in the Gaussian distribution. So if a Bayesian network can predict the value of $\log S_I$, we expect that 95 % of the corresponding $\log S_I$ values found in the IVGTT study, will lie within this interval. If this is the case, we say that the predictive distribution of $\log S_I$ is *well calibrated*, see Dawid (1982).

Further, we perform an ordinary linear regression of the IVGTT obtained S_I on the predicted S_I and calculate the residual standard deviation, SD , and the correlation coefficient, R^2 , obtained from this regression. To show that there is no systematic bias in these regressions, we report the intercept and slope of these regressions lines.

5.2 The Different Models

In the following we will present different approaches for finding a Bayesian network, that can model the dependency relations between the variables in the problem. Further, we will present the results of a previous approach, where multiple linear regression is used and also the results of using the leaps and bounds algorithm for best subset selection.

Bayesian regression network

Previous results have shown that predictions of $\log S_I$ from a multiple regression on OGTT plasma glucose and serum insulin levels, BMI and sex, are highly correlated to the corresponding IVGTT-derived S_I estimates, see *e.g.* Drivsholm et al. (2003). We will therefore learn the parameters and the structure of a network, where $\log S_I$ can depend on these variables, and these variables are marginally independent, *i.e.* the only arrows that are allowed in the model, are arrows into $\log S_I$. This network represents a regression model, so we will refer to it as the Bayesian regression network. To learn this network, we need to specify a prior network, *i.e.* a prior DAG and a prior probability distribution. As prior DAG we use, for simplicity, the empty DAG, *i.e.* the one without any arrows. This DAG represent that all the variables are independent, so the local probability distribution for each node only depends on the node itself. To specify the prior probability distribution, we use the sample mean and the sample variance as an initial estimate of the mean and variance. As a measure of our confidence in this network, we use $N = 100$ for the size of the imaginary database. Figure 1 shows the result of the structural learning procedure. We see that $\log S_I$ depends on almost all of the insulin measurements, except for I10, and a few of the glucose measurements.

Bayesian network with empty prior network

In situations where not all the variables are observed, information is gained by modeling the possible correlations between the explanatory variables. So we will now learn a network, where these correlations are allowed. We only consider networks, where arrows between the glucose and insulin measurements point forward in time, where BMI and sex can not have any parents and where $\log S_I$ can not have any children. Again we use the empty DAG as prior DAG,

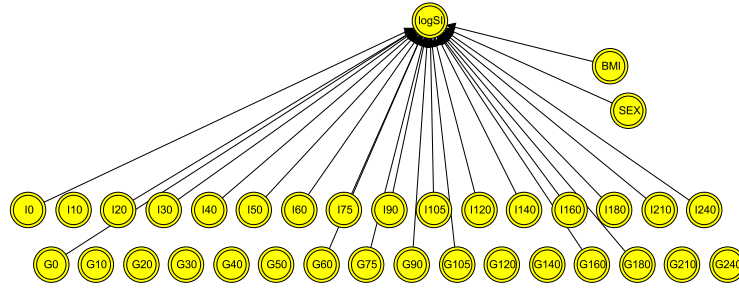


Figure 1: The result of the structural learning procedure for the Bayesian regression network.

the sample mean and sample variance to specify the prior probability distribution and $N = 100$ as a measure of our confidence in this network. The result of the structural learning procedure reveals a complicated dependency structure between the variables, see Figure 2.

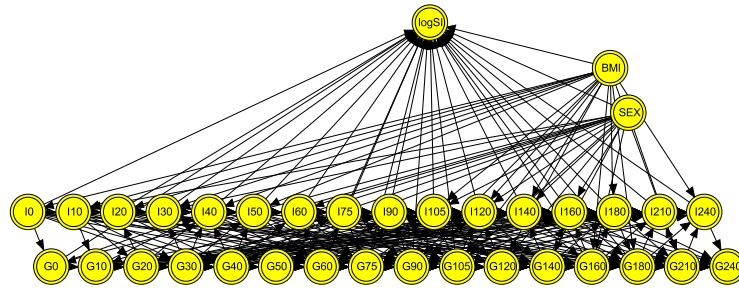


Figure 2: The Bayesian network with the empty network as prior.

The Markov blanket for $\log S_I$ in this network, is the same as the Bayesian regression network, see Figure 1. The reason for this is that $\log S_I$, in both networks, is not allowed to have any children and because we in both approaches have used the same prior network. So when all the variables in the Markov blanket are observed, as it is in our study, the prediction results are exactly the same as for the Bayesian regression network.

Bayesian network with physiological prior network

In the previous two networks, we have for simplicity used the empty DAG as prior DAG. We will now use a prior DAG, called the physiological network, where the knowledge we have about the physiological relations between the

variables is incorporated. In this network, insulin measurements and glucose measurements are assumed to be Markov processes. They are coupled so that the current glucose measurement depends on the previous insulin measurement and the current insulin measurement depends on the current glucose measurement, see Figure 3. This structure is consistent with the physiological model used in Bergmans minimal model to determine S_I from an IVGTT. In addition, we let the initial glucose and insulin measurements depend on BMI and sex.

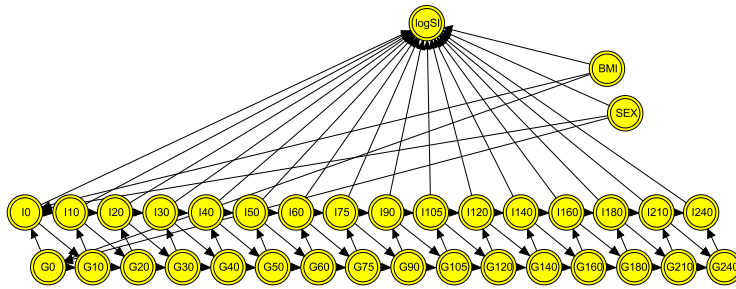


Figure 3: The physiological network.

Like before, we estimate the prior probability distribution from data. However, contrary to the empty network, the variables in the physiological network depends on other variables, so we perform a linear regression on the parents and use the sample mean and sample variance from these regressions as the mean and variance in the local prior probability distributions. Again we use $N = 100$ and we only consider networks where arrows between the glucose and insulin measurements point forward in time, where BMI and sex can not have any parents and where $\log S_I$ can not have any children. The result of the structural learning procedure is shown in Figure 4. As before, we see a complicated de-

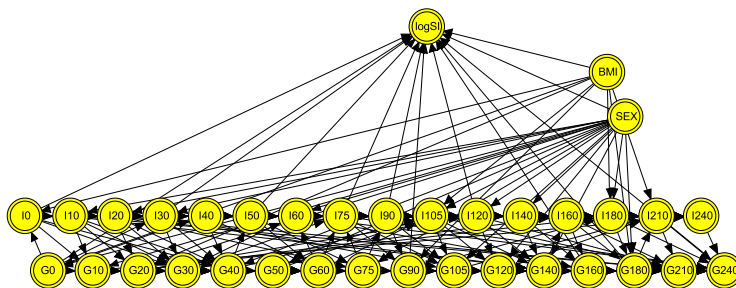


Figure 4: The Bayesian network with the physiological network as prior.

pendency structure between the variables. In Figure 5, the Markov blanket of $\log S_I$ is shown and we see that is quite different than with the empty prior,

shown in Figure 1. Only 6 of the insulin measurements and 5 of the glucose measurements are included in the present blanket.

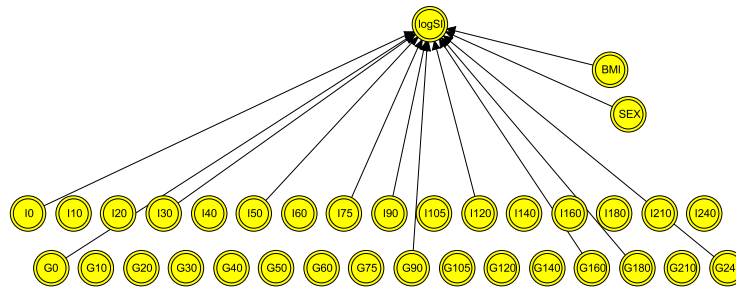


Figure 5: The Markov blanket for the Bayesian network with the physiological network as prior.

Results using multiple linear regression

In Drivsholm et al. (2003), multiple linear regression is used to derive predictive equations of $\log S_I$ using OGTT plasma glucose and serum insulin levels, BMI and gender. To limit the amount of blood samples drawn from the patients, they constrain the models to include glucose and insulin observations to the same time point. By a combination of backwards elimination and forward selection, they find the optimal model to be with sample time points 0, 30, 60, 105, 180, and 240. Notice, though, that they have found their model on the basis of a different training dataset than ours, as the partition of the dataset into training data and validation data is done randomly in both cases.

Results using the leaps and bound algorithm

Further, we have tried the leaps and bound algorithm by Furnival and Wilson (1974), using the Bayesian information criteria to find the best subset of the explanatory variables. With this approach, the optimal model is with I50, I90, G160, BMI and sex as explanatory variables. In theory, when the size of the database approaches infinity, using the Bayesian information criteria will result in the same subset of explanatory variables as when using the network score as selection criteria, see Haughton (1988).

5.3 Evaluation

To compare the different models, we first consider the network score. Notice that we can only compare network scores for networks that are learned using the same prior network.

To be able to evaluate all models using the network score, we have also calculated the log scores for the results found in the multiple linear regression approach and the leaps and bounds approach. This is done by formulating these results as Bayesian networks and calculating the log scores using respectively the empty network and the physiological network as prior network. Likewise, for the Bayesian regression network found by using the empty network as prior network, we have calculated the log score using the physiological network as prior network.

Model	Empty prior	Physiological prior
BR	-17878.30	-17848.33
BN	-16528.39	-14851.44
MLR	-17886.17	-17849.06
L&B	-17894.95	-17846.12

Table 1: Network scores for the different models.

The results are reported in Table 1. The Bayesian network model (BN) has the lowest log score, *i.e.* the highest network score, both when the empty network and the physiological network are used as prior network. This is obvious as the BN is selected using the network score as selection criteria and because the Bayesian regression (BR), the multiple linear regression (MLR) and the leaps and bounds (L&B) networks are included in the search space, when searching for the BN with the highest score. So unless we have only found a local maximum, instead of a global maximum, the score for the BN must be higher than the score for the other networks.

When comparing the scores found using the empty prior, we see that the network scores for the BR network, the MLR network and the L&B network are almost all the same. The network score for the BN is over a thousand times higher than for any of the other networks, indicating that the BN provides a much better fit to data. Recall, however, that the Markov blanket for the BR network and the BN are the same, so when all the variables in the Markov blanket are observed, the BR network and the BN will predict the same $\log S_I$ values. So the higher network score is not important when data are complete, but can have an impact

when data are incomplete.

When using the physiological network as prior network, we see almost the same result. The network score for the BR, MLR and L&B networks are almost all the same, whereas the network score for the BN is over 3000 times higher than for any of the other networks.

Model	Tr. data $R^2(SD)$	Val. data $R^2(SD)$	Tr. outside	Val. outside
BR with empty prior	0.76(0.31)	0.73(0.35)	1(1%)	1(2%)
BN with empty prior	0.76(0.31)	0.73(0.35)	1(1%)	1(2%)
BN with physiological prior	0.77(0.30)	0.73(0.36)	7(5%)	3(6%)
MLR	0.76(0.31)	0.66(0.40)	3(2%)	3(6%)
L&B	0.75(0.31)	0.73(0.36)	6(4%)	4(9%)

Table 2: The table lists the R^2 and SD values from the linear regressions of the IVGTT obtained $\log S_I$ on the predicted $\log S_I$ for both the training dataset and the validation dataset. Also listed are how many $\log S_I$ values that fall outside the credibility interval $\mu \pm 1.96 \cdot \sigma$.

In Table 2 the R^2 and SD values from the linear regression of the IVGTT obtained $\log S_I$ on the predicted $\log S_I$ are reported. The R^2 and SD values are for all five models acceptable and they are almost the same for all models, except for the multiple regression model, which on the validation dataset does not perform as well as the others. Table 3 shows the intercept and slope of the estimated regression lines and there are no evidence of any systematic bias. We therefore conclude that an OGTT can be used to determine the insulin sensitivity index.

Model	Tr. data (intercept, slope)	Val. data (intercept, slope)
BR empty prior	(-0.19, 1.09)	(-0.05, 1.01)
BN empty prior	(-0.03, 1.01)	(0.25, 0.87)
BN physiological prior	(-0.06, 1.03)	(0.14, 0.92)
MLR	(0, 1)	(0.06, 0.96)
L&B	(0, 1)	(0.11, 0.93)

Table 3: The intercept and slope of the regressions lines from the regressions of the IVGTT obtained S_I on the predicted S_I . Reported to show that there is no evidence of systematic bias.

In Table 2 we have also listed how many $\log S_I$ values that fall outside the credibility interval $\mu \pm 1.96 \cdot \sigma$. Approximately 5% of these predictions should

lie outside and 95 % inside the interval for the predictive distributions to be well calibrated. This is clearly fulfilled for the BN with the physiological network as prior network, so the predictive distribution for $\log S_I$ is, when using this network, well calibrated. With the MLR approach and the L&B approach it is almost fulfilled that 5% of the predictions lie outside the intervals. We will therefore conclude that the predictive distributions are also well calibrated in these cases. For the BR and the BN with the empty network as prior network, very few values lie outside the intervals, indicating that the variance is probably estimated to large. Figure 6 shows the predicted $\log S_I$ values and the intervals for the BN with the empty prior and for the BN with the physiological prior. We see that for the two models, the predicted $\log S_I$ values are almost the same, but the intervals are much wider for the BN with the empty prior, meaning that the variance in this model is larger.

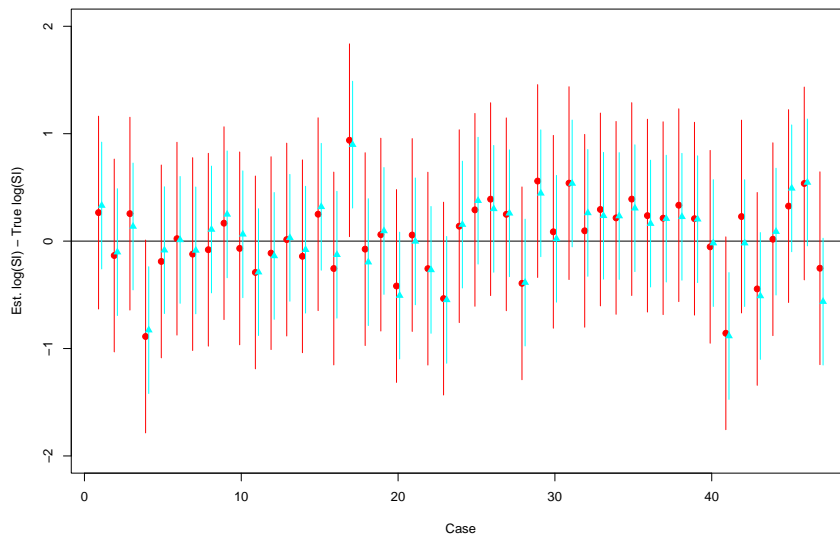


Figure 6: The predicted $\log S_I$ values and the credibility intervals for the Bayesian network with empty prior (dark and disks) versus the Bayesian network with physiological prior (light and triangles).

So to summarize, all the models give adequate predictions of the $\log S_I$ values. Evaluating the models using the different validation approaches all together, the BN with the physiological prior model gives a more precise predictive distribution of $\log S_I$ compared to the other models. We therefore suggest that this model should be used to derive the predictive values of $\log S_I$.

6 Discussion

We have established a promising way of determining the insulin sensitivity index from an oral glucose tolerance test rather than from an intravenous glucose tolerance test. All approaches give adequate predictions of S_I . The Bayesian network with the physiological prior estimates the most precise predictive distribution of S_I , so we claim that this is the best model. There are also other advantages by using a Bayesian network instead of an ordinary regression model. In a Bayesian network, we can use any prior knowledge available from *e.g.* previous studies or from the physiological understanding of the problem. Further, we can calculate the predictive distribution of $\log S_I$ in situations, where some of the observations are missing. This can be used when a single or a few observations are missing for a specific subject. It can also be used when certain time points are not observed at all, which could be the case if a dataset from another study, using fewer time points, is analyzed.

Acknowledgements

This work has been supported by Novo Nordisk A/S. The collection of data were done by Torben Hansen, Novo Nordisk A/S.

References

- Bergman, R. N., Ider, Y. Z., Bowden, C. R. and Cobelli, C. (1979). Quantitative estimation of insulin sensitivity, *American Journal of Physiology* **236**: E667–E677.
- Bøttcher, S. G. (2001). Learning Bayesian Networks with Mixed Variables, *Artificial Intelligence and Statistics 2001*, Morgan Kaufmann, San Francisco, CA, USA, pp. 149–156.
- Bøttcher, S. G. and Dethlefsen, C. (2003). `deal`: A Package for Learning Bayesian Networks, *Journal of Statistical Software* **8**(20): 1–40.
- Cooper, G. and Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data, *Machine Learning* **9**: 309–347.

- Cowell, R. G., Dawid, A. P., Lauritzen, S. L. and Spiegelhalter, D. J. (1999). *Probabilistic Networks and Expert Systems*, Springer-Verlag, Berlin-Heidelberg-New York.
- Dawid, A. P. (1982). The Well-Calibrated Bayesian, *Journal of the American Statistical Association* **77**(379): 605–610.
- Drivsholm, T., Hansen, T., Urhammer, S. A., Palacios, R. T., Vølund, A., Borch-Johnsen, K. and Pedersen, O. B. (2003). Assessment of insulin sensitivity index and acute insulin response from an oral glucose tolerance test in subjects with normal glucose tolerance, Novo Nordisk A/S.
- Furnival, G. M. and Wilson, R. W. (1974). Regression by Leaps and Bounds, *Technometrics* **16**(4): 499–511.
- Geiger, D. and Heckerman, D. (1994). Learning Gaussian Networks, *Proceedings of Tenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, CA, USA, pp. 235–243.
- Houghton, D. M. A. (1988). On The Choice of a Model to fit Data From an Exponential Family, *The Annals of Statistics* **16**(1): 342–355.
- Heckerman, D., Geiger, D. and Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data, *Machine Learning* **20**: 197–243.
- Ihaka, R. and Gentleman, R. (1996). R: A language for data analysis and graphics, *Journal of Computational and Graphical Statistics* **5**: 299–314.
- Lauritzen, S. L. (1992). Propagation of probabilities, means and variances in mixed graphical association models, *Journal of the American Statistical Association* **87**(420): 1098–1108.
- Martin, B. C., Warram, J. H., Krolewski, A. S., Bergman, R. N., Soeldner, J. S. and Kahn, C. R. (1992). Role of glucose and insulin resistance in development of type 2 diabetes mellitus: results of a 25-year follow-up study, *The Lancet* **340**: 925–929.
- Pacini, G. and Bergman, R. N. (1986). MINMOD: a computer program to calculate insulin sensitivity and pancreatic responsivity from the frequently sampled intravenous glucose tolerance test, *Computer Methods and Programs in Biomedicine* **23**: 113–122.

Paper IV
Learning Dynamic Bayesian Networks
with Mixed Variables

Susanne G. Böttcher

Learning Dynamic Bayesian Networks with Mixed Variables

Susanne G. Bøttcher

Aalborg University, Denmark

Abstract.

This paper considers dynamic Bayesian networks for discrete and continuous variables. We only treat the case, where the distribution of the variables is conditional Gaussian. We show how to learn the parameters and structure of a dynamic Bayesian network and also how the Markov order can be learned. An automated procedure for specifying prior distributions for the parameters in a dynamic Bayesian network is presented. It is a simple extension of the procedure for the ordinary Bayesian networks. Finally the Wölfer's sunspot numbers are analyzed.

1 Introduction

In this paper we consider dynamic Bayesian networks (DBNs) for discrete and continuous variables. A DBN is an extension of an ordinary Bayesian network and is applied in the modeling of time series.

DBNs for first order Markov time series are described in Dean and Kanazawa (1989). In Murphy (2002), a thorough treatment of these models is presented and in Friedman et al. (1998) learning these networks in the case with only discrete variables is described.

Here we consider DBNs with both discrete and continuous variables. In these networks we also allow some of the variables to be static, *i.e.* some of the variables do not change over time. We only treat the case where the distribution of the variables is conditional Gaussian (CG) and show how to learn the parameters and structure of the DBN when data is complete. Further we present an automated method for specifying prior parameter distributions for the parameters in a DBN. These methods are simple extensions of the ones used for ordinary Bayesian networks with mixed variables, described in Bøttcher (2001).

We consider time series, where the Markov order can be higher than one and show how the Markov order can be learned.

In Section 2, DBNs with static and time varying variables are defined. Section 3

presents these DBNs for the mixed case and Section 4 gives some examples of some well known models that can be represented as DBNs. Section 5 shows how to learn the parameters and structure of a DBN with mixed variables. Further, it shows how the Markov order can be learned. Section 6 presents a method for specifying prior distributions of the parameters in the DBN. In Section 7 Wölfer’s sunspot numbers are analyzed using a DBN.

2 Dynamic Bayesian Networks

A *Bayesian network* is a graphical model that encodes the joint probability distribution for a set of variables. For terminology and theoretical aspects on graphical models, see Lauritzen (1996). We define it as a *directed acyclic graph* (DAG) $D = (V, E)$, where V is a finite set of nodes and E is a finite set of directed edges between the nodes. The DAG defines the structure of the Bayesian network. To each node $v \in V$ in the graph corresponds a random variable X_v . The set of variables associated with the graph D is then $X = (X_v)_{v \in V}$.

To each vertex v with parents $\text{pa}(v)$, there is attached a local probability distribution, $p(x_v | x_{\text{pa}(v)})$. The possible lack of directed edges in D encodes conditional independencies between the random variables X through the factorization of the joint probability distribution,

$$p(x) = \prod_{v \in V} p(x_v | x_{\text{pa}(v)}).$$

In a Bayesian network, the set of random variables X is fixed. To model a multivariate time series we need a framework, where we allow the set of random variables to vary with time. For this we use dynamic Bayesian networks, defined as below. This definition is consistent with the exposition in Murphy (2002), but here we also allow for static variables and Markov orders higher than one.

Let X^t be a set of time varying random variables, that is X^t can take on the values X^0, X^1, \dots, X^T . We index the time varying variables by the non-negative integers to indicate that the observations are taken at discrete time points. The corresponding nodes in the graph are denoted V_t , so $X^t = (X_v^t)_{v \in V_t}$ for each time point t . Note however that V_t is “the same” for all time points t , so formally $V_t = \{(v, t), v \in V\}$. Further, let X^s be a set of static random variables, *i.e.* variables that do not change over time. The nodes corresponding

to X^s are denoted V_s . The set of variables associated with a DBN is then $X = ((X^t)_{t=0}^T, X^s)$ and the set of nodes is $V = ((V_t)_{t=0}^T, V_s)$.

We refer to the time varying variables at one time point as a *time slice* or just a slice. We let the static variables X^s belong to the time slice at time $t = 0$ and refer to this as the initial time slice. So the initial time slice includes the variables X^0 and X^s and, for $t = 1, \dots, T$, the time slice at time t includes the variables X^t .

We will mostly consider the variables in the initial time slice jointly, so to ease later notation we define $X^{\tilde{0}} = (X^0, X^s)$ and $V_{\tilde{0}} = (V_0, V_s)$.

The joint probability distribution of the variables in a dynamic Bayesian network can be very complex, as the number of variables grows over time. Therefore we assume that the time series we are dealing with, is *mth order Markov*, i.e.

$$p(x^t | x^{t-1}, \dots, x^0) = p(x^t | x^{t-1}, \dots, x^{t-m}),$$

for all time points $t = m, \dots, T$.

Further, we assume that the time series has *stationary dynamics*, so

$$p(x^t | x^{t-1}, \dots, x^{t-m}) = p(x^m | x^{m-1}, \dots, x^0),$$

for all $t = m, \dots, T$. Stationary dynamics refers to the fact that the conditional distributions are time independent, while the marginal distributions may be time dependent.

We will first introduce DBNs for time series that are first order Markov. With the above assumptions, a DBN for a first order Markov time series can be defined to be the pair $(B_{\tilde{0}}, B_{\rightarrow})$, where $B_{\tilde{0}}$ is a Bayesian network defining the probability distribution of $X^{\tilde{0}}$ as

$$p(x^{\tilde{0}}) = \prod_{v \in V_{\tilde{0}}} p(x_v^{\tilde{0}} | x_{\text{pa}(v)}^{\tilde{0}}),$$

and B_{\rightarrow} is a 2-slice temporal Bayesian network defining the conditional distribution of X^t as

$$p(x^t | x^{t-1}, x^s) = \prod_{v \in V_t} p(x_v^t | x_{\text{pa}(v)}^t, x_{\text{pa}(v)}^{t-1}, x_{\text{pa}(v)}^s).$$

The joint probability distribution for a DBN with $T + 1$ time points is given as

$$p(x^0, \dots, x^T, x^s) = p(x^{\tilde{0}}) \prod_{t=1}^T p(x^t | x^{t-1}, x^s).$$

As we assumed that the time series has stationary dynamics, the DBN is completely specified through $B_{\bar{0}}$ and B_{\rightarrow} .

For the dependency relations between the time slices we assume that arrows point forward in time, so the variables in time slice t can have parents in the time slices to time t and $t - 1$. Further, they can have parents from X^s . Due to stationary dynamics, the dependency relations between the time slices are the same for all time points. This also means that if a time varying variable X_v^t has a static variable X_w^s as a parent, then X_w^s is also a parent of X_v^1, \dots, X_v^T . The variables in the initial time slice can have parents from the initial time slice and therefore also from X^s , as X^s is included in the initial time slice.

Within a time slice, there are no restrictions of the dependency relations between the variables, as long as the structure is a DAG. Due to stationary dynamics, the dependency relations within a time slice are the same for the time slices to time $t = 1, \dots, T$. They are however not necessarily the same as for the time varying variables in the initial time slice.

So the structure of the DBN repeats itself over time, except for $B_{\bar{0}}$, where the time series is initialized.

Figure 1 shows an example of the structure of a first order Markov DBN, $(B_{\bar{0}}, B_{\rightarrow})$, with two time varying variables Y^t and Z^t and one static variable X^s . Because of the first order Markov property, the structure is completely specified through the first two time points and the structure of the DBN can therefore be represented by the DAG in Figure 2.

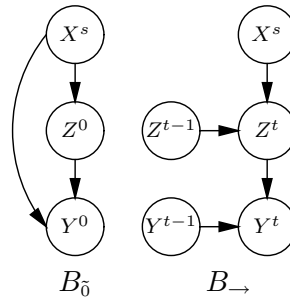


Figure 1: Example of a first order Markov DBN $(B_{\bar{0}}, B_{\rightarrow})$.

For time series with higher Markov order properties, we need to extend the definition.

Consider an m th order Markov time series. The joint probability distribution

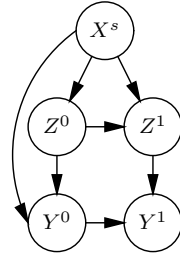


Figure 2: A first order Markov DBN $(B_{\bar{0}}, B_{\rightarrow})$ represented by the first two time points.

for $T + 1$ time points can be written as

$$\begin{aligned} p(x^0, \dots, x^T, x^s) &= p(x^{\bar{0}}, x^1, \dots, x^{m-1}) \prod_{t=m}^T p(x^t | x^{t-1}, \dots, x^{t-m}, x^s) \\ &= p(x^{\bar{0}}) p(x^1 | x^{\bar{0}}) \dots p(x^{m-1} | x^{m-2}, \dots, x^{\bar{0}}) \\ &\times \prod_{t=m}^T p(x^t | x^{t-1}, \dots, x^{t-m}, x^s). \end{aligned}$$

Following the definition for first order Markov time series, we let B_{\rightarrow} be a $m + 1$ -slice temporal Bayesian network defining the conditional distribution of X^t ,

$$p(x^t | x^{t-1}, \dots, x^{t-m}, x^s) = \prod_{v \in V_t} p(x_v^t | x_{\text{pa}(v)}^t, \dots, x_{\text{pa}(v)}^{t-m}, x_{\text{pa}(v)}^s),$$

for $t = m, \dots, T$.

The variables in time slice t can have parents in the time slices to times $t, \dots, t - m$ and they can have parents from X^s . Again, due to stationary dynamics, the dependency relations between and within the time slices are the same for all time points $t = m, \dots, T$. Further, if a time varying variable X_v^t has a static variable X_w^s as a parent, then X_w^s is also a parent of X_v^m, \dots, X_v^T .

The question is now how to initialize the time series. The probability distribution $p(x^{\bar{0}}, x^1, \dots, x^{m-1})$ can be written as

$$p(x^{\bar{0}}, x^1, \dots, x^{m-1}) = p(x^{\bar{0}}) p(x^1 | x^{\bar{0}}) \dots p(x^{m-1} | x^{m-2}, \dots, x^{\bar{0}}). \quad (1)$$

As arrows point forward in time, this factorization defines the possible dependency relations between the variables $X^{\bar{0}}, \dots, X^{m-1}$. As before we let $B_{\bar{0}}$ be a

Bayesian network defining the probability distribution of $X^{\tilde{0}}$ as

$$p(x^{\tilde{0}}) = \prod_{v \in V_{\tilde{0}}} p(x_v^{\tilde{0}} | x_{\text{pa}(v)}^{\tilde{0}}).$$

Now we also define Bayesian networks for the rest of the conditional distributions in (1). We let B_1 be a 2-slice Bayesian network defining the conditional distribution of X^1 given $X^{\tilde{0}}$ as

$$p(x^1 | x^{\tilde{0}}) = \prod_{v \in V_1} p(x_v^1 | x_{\text{pa}(v)}^1, x_{\text{pa}(v)}^{\tilde{0}}),$$

and likewise for B_2, \dots, B_{m-1} , where B_{m-1} is an m -slice Bayesian network defining the conditional distribution of X^{m-1} given $X^{m-2}, \dots, X^{\tilde{0}}$ as

$$p(x^{m-1} | x^{m-2}, \dots, x^{\tilde{0}}) = \prod_{v \in V_{m-1}} p(x_v^{m-1} | x_{\text{pa}(v)}^{m-1}, \dots, x_{\text{pa}(v)}^{\tilde{0}}).$$

So the variables in the time slice to time $t = 1$ can have parents from the time slice to time $t = 1$ and $t = 0$. The variables in time slice $m - 1$ can have parents from the time slices to time $t = 0, \dots, m - 1$. The dependency relations between the time slices to time $t = 0, \dots, m - 1$ are obviously not the same and the dependency relations within these time slices are not necessarily the same.

The tuple $(B_{\tilde{0}}, B_1, \dots, B_{m-1}, B_{\rightarrow})$ is thus a DBN for an m th order Markov time series, where the different Bayesian networks in the tuple defines the corresponding probability distributions as above. Notice that we could also just have specified the networks $B_{\tilde{0}}, B_1, \dots, B_{m-1}$ as one large network, with the necessary restrictions on the arrows.

3 Dynamic Bayesian Networks for Mixed Variables

In this section we consider DBNs with *mixed variables*, *i.e.* the variables in the network can be of discrete and continuous type. We let $V = \Delta \cup \Gamma$, where Δ and Γ are the sets of discrete and continuous variables, respectively. The corresponding random variables X can then be denoted $X = (X_v)_{v \in V} = (I, Y) = ((I_{\delta})_{\delta \in \Delta}, (Y_{\gamma})_{\gamma \in \Gamma})$. Again, we index the sets of nodes and the random variables with t for time varying variables, s for static variables and $\tilde{0}$ for the variables in the initial time slice.

To ensure availability of exact local computation methods, we do not allow continuous parents of discrete nodes, so the probability distributions factorize into a discrete part and a mixed part as presented below. To simplify notation, we present the theory for first order Markov time series and comment on how to extend it to higher order Markov assumptions by following the definitions introduced in the previous section.

We consider $B_{\bar{0}}$ and B_{\rightarrow} separately, and the joint probability distribution is obtained as specified in the previous section.

For $B_{\bar{0}}$ we have that

$$\begin{aligned} p(x^{\bar{0}}) &= \prod_{v \in V_{\bar{0}}} p(x_v^{\bar{0}} | x_{\text{pa}(v)}^{\bar{0}}) \\ &= \prod_{\delta \in \Delta_{\bar{0}}} p(i_{\delta}^{\bar{0}} | i_{\text{pa}(\delta)}^{\bar{0}}) \prod_{\gamma \in \Gamma_{\bar{0}}} p(y_{\gamma}^{\bar{0}} | i_{\text{pa}(\gamma)}^{\bar{0}}, y_{\text{pa}(\gamma)}^{\bar{0}}) \end{aligned} \quad (2)$$

and for B_{\rightarrow}

$$\begin{aligned} p(x^t | x^{t-1}, x^s) &= \prod_{v \in V_t} p(x_v^t | x_{\text{pa}(v)}^t, x_{\text{pa}(v)}^{t-1}, x_{\text{pa}(v)}^s) \\ &= \prod_{\delta \in \Delta_t} p(i_{\delta}^t | i_{\text{pa}(\delta)}^t, i_{\text{pa}(\delta)}^{t-1}, i_{\text{pa}(\delta)}^s) \\ &\quad \times \prod_{\gamma \in \Gamma_t} p(y_{\gamma}^t | i_{\text{pa}(\gamma)}^t, i_{\text{pa}(\gamma)}^{t-1}, i_{\text{pa}(\gamma)}^s, y_{\text{pa}(\gamma)}^t, y_{\text{pa}(\gamma)}^{t-1}, y_{\text{pa}(\gamma)}^s). \end{aligned} \quad (3)$$

To account for higher order Markov assumptions, we would just have to specify the probability distributions for the intervening networks accordingly.

To simplify notation for B_{\rightarrow} , we use the following notation, where the possible parent configurations are not explicitly defined. They must be specified in the given context and according to (3).

$$\begin{aligned} p(x^t | x^{t-1}, x^s) &= \prod_{v \in V_t} p(x_v^t | x_{\text{pa}(v)}^{\rightarrow}) \\ &= \prod_{\delta \in \Delta_t} p(i_{\delta}^t | i_{\text{pa}(\delta)}^{\rightarrow}) \prod_{\gamma \in \Gamma_t} p(y_{\gamma}^t | i_{\text{pa}(\gamma)}^{\rightarrow}, y_{\text{pa}(\gamma)}^{\rightarrow}). \end{aligned}$$

So for example, $i_{\text{pa}(\delta)}^{\rightarrow}$ contains the variables $i_{\text{pa}(\delta)}^t$, $i_{\text{pa}(\delta)}^{t-1}$ and $i_{\text{pa}(\delta)}^s$.

In this paper we only consider networks, where the joint distribution of the variables is conditional Gaussian. The local probability distributions are therefore

defined as in the following two sections. In these sections, we do not distinguish between the variables in $B_{\bar{0}}$ and B_{\rightarrow} , as the distribution of these variables is of the same type. The possible parent set differ however between variables in $B_{\bar{0}}$ and variables in B_{\rightarrow} . In the following we therefore just denote the parents of a variable x_v by $x_{\text{pa}(v)}$ and $x_{\text{pa}(v)}$ must be specified according to (2) or (3).

3.1 Distribution for Discrete Variables

When the joint distribution is conditional Gaussian, the local probability distributions for the discrete variables are just unrestricted discrete distributions with

$$p(i_\delta | i_{\text{pa}(\delta)}) \geq 0 \quad \forall \quad \delta \in \Delta.$$

We parameterize this as

$$\theta_{i_\delta | i_{\text{pa}(\delta)}} = p(i_\delta | i_{\text{pa}(\delta)}, \theta_{\delta | i_{\text{pa}(\delta)}}),$$

where $\theta_{\delta | i_{\text{pa}(\delta)}} = (\theta_{i_\delta | i_{\text{pa}(\delta)}})_{i_\delta \in \mathcal{I}_\delta}$.

Furthermore $\sum_{i_\delta \in \mathcal{I}_\delta} \theta_{i_\delta | i_{\text{pa}(\delta)}} = 1$ and $0 \leq \theta_{i_\delta | i_{\text{pa}(\delta)}} \leq 1$. All parameters associated with a node δ is denoted by θ_δ , so $\theta_\delta = (\theta_{i_\delta | i_{\text{pa}(\delta)}})_{i_\delta \in \mathcal{I}_\delta}$.

3.2 Distribution for Continuous Variables

For the continuous variables, the local probability distributions are Gaussian linear regressions with parameters depending on the configuration of the discrete parents. So let the parameters be given by $\theta_{\gamma | i_{\text{pa}(\gamma)}} = (m_{\gamma | i_{\text{pa}(\gamma)}}, \beta_{\gamma | i_{\text{pa}(\gamma)}}, \sigma_{\gamma | i_{\text{pa}(\gamma)}}^2)$. Then

$$(Y_\gamma | y_{\text{pa}(\gamma)}, i_{\text{pa}(\gamma)}, \theta_{\gamma | i_{\text{pa}(\gamma)}}) \sim \mathcal{N}(m_{\gamma | i_{\text{pa}(\gamma)}} + \beta_{\gamma | i_{\text{pa}(\gamma)}} y_{\text{pa}(\gamma)}, \sigma_{\gamma | i_{\text{pa}(\gamma)}}^2), \quad (4)$$

where $\beta_{\gamma | i_{\text{pa}(\gamma)}}$ are the regression coefficients, $m_{\gamma | i_{\text{pa}(\gamma)}}$ is the regression intercept, and $\sigma_{\gamma | i_{\text{pa}(\gamma)}}^2$ is the conditional variance. Thus for each configuration of the discrete parents of γ the distribution of Y_γ is Gaussian with mean and variance given as in (4). The parameters associated with a node γ is then $\theta_\gamma = (\theta_{\gamma | i_{\text{pa}(\gamma)}})_{i_{\text{pa}(\gamma)} \in \mathcal{I}_{\text{pa}(\gamma)}}$.

3.3 The Parameterized Distributions

With the above distributional assumptions, we can specify the parameterized DBN as follows.

Let $\theta^{\tilde{0}} = ((\theta_{\delta}^{\tilde{0}})_{\delta \in \Delta_{\tilde{0}}}, (\theta_{\gamma}^{\tilde{0}})_{\gamma \in \Gamma_{\tilde{0}}})$ and $\theta^{\rightarrow} = ((\theta_{\delta}^{\rightarrow})_{\delta \in \Delta_t}, (\theta_{\gamma}^{\rightarrow})_{\gamma \in \Gamma_t})$. Further, let $\theta = (\theta^{\tilde{0}}, \theta^{\rightarrow})$. Then $B_{\tilde{0}}$ is given as

$$p(x^{\tilde{0}} | \theta^{\tilde{0}}) = \prod_{\delta \in \Delta_{\tilde{0}}} p(i_{\delta}^{\tilde{0}} | i_{\text{pa}(\delta)}^{\tilde{0}}, \theta_{\delta | i_{\text{pa}(\delta)}}^{\tilde{0}}) \prod_{\gamma \in \Gamma_{\tilde{0}}} p(y_{\gamma}^{\tilde{0}} | i_{\text{pa}(\gamma)}^{\tilde{0}}, y_{\text{pa}(\gamma)}^{\tilde{0}}, \theta_{\gamma | i_{\text{pa}(\gamma)}}^{\tilde{0}}),$$

and B_{\rightarrow} as

$$\begin{aligned} p(x^t | x^{t-1}, x^s, \theta^{\rightarrow}) &= \prod_{\delta \in \Delta_t} p(i_{\delta}^t | i_{\text{pa}(\delta)}^{\rightarrow}, \theta_{\delta | i_{\text{pa}(\delta)}}^{\rightarrow}) \\ &\times \prod_{\gamma \in \Gamma_t} p(y_{\gamma}^t | y_{\text{pa}(\gamma)}^{\rightarrow}, i_{\text{pa}(\gamma)}^{\rightarrow}, \theta_{\gamma | i_{\text{pa}(\gamma)}}^{\rightarrow}). \end{aligned}$$

The joint distribution for $T + 1$ time points is given as

$$p(x^0, \dots, x^T, x^s, \theta) = p(x^{\tilde{0}} | \theta^{\tilde{0}}) \prod_{t=1}^T p(x^t | x^{t-1}, x^s, \theta^{\rightarrow}).$$

Notice that, due to stationarity, θ^{\rightarrow} is the parameter in the conditional distribution of x^t for every time point $t = 1, \dots, T$.

4 Examples of DBNs

We will now give some examples of some well known models that can be represented as DBNs. In the figures, shaded nodes represent discrete variables and clear nodes represent continuous variables.

4.1 Hidden Markov Models

A Hidden Markov Model (HMM) is a stochastic automaton, where each state generates an observation. Figure 3 shows a HMM, where the hidden states are

first order Markov.

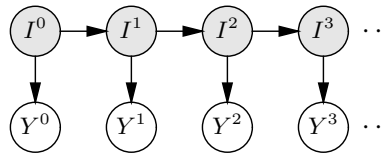


Figure 3: A Hidden Markov Model.

The hidden states, *i.e.* the discrete hidden variables, are denoted by I and the observations by Y . We have represented the observed variables as continuous, but they can also all be discrete. In this HMM, I^{t+1} is conditionally independent of I^{t-1} , given I^t . Further, Y^t is conditionally independent of the rest of the variables in the network, given I^t . A model like this is used in situations, where the observations do not follow the same model all the time, but can follow different models at different times. This gives for example the possibility to account for outliers.

When a HMM is represented as a DBN, we assume that the time series has stationary dynamics. So, together with the first order Markov property, we can specify the joint probability distribution for the variables in this network by just specifying the initial prior probabilities $p(i^0)$, the transition probabilities $p(i^t|i^{t-1})$ and the conditional Gaussian distributions $p(y^t|i^t)$ (or, if the observed variables are discrete, the conditional multinomial distributions $p(j^t|i^t)$).

There are many variants of this basic HMM, *e.g.* Buried Markov Model, Mixed-memory Markov Model and Hierarchical HMM, see Murphy (2002) for a presentation of these models represented as DBNs and their application within speech recognition.

4.2 Kalman Filter Models

A Kalman Filter Model (KFM), introduced by Harrison and Stevens (1976) as a state space model, models the dynamic behavior of a time series. In such a model, the continuous observations Y are indirect measurements of a latent Markov process Z .

In Figure 4, a KFM is shown. The structure is the same as for the HMM, since the two models assume the same set of conditional independencies. The probability distributions to be specified is the Gaussian distribution $p(z^0)$, the Gaus-

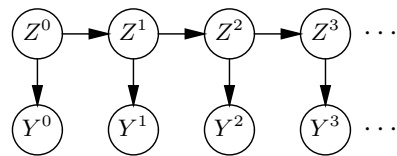


Figure 4: A Kalman Filter Model.

sian linear regression $p(z^t|z^{t-1})$ and the Gaussian linear regression $p(y^t|z^t)$. For a comprehensive treatment of KFMs and their applications, see West and Harrison (1989).

4.3 Multiprocess Kalman Filter Models

Multiprocess Kalman Filter Models (MKFMs), also known as switching state space Markov models, are an extension of the KFMs, see Harrison and Stevens (1976), where the aim is to discriminate between different KFMs.

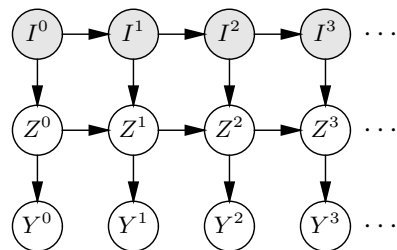


Figure 5: A Multiprocess Kalman Filter Model.

Figure 5 shows a MKFM. Again we see that the continuous observations Y are indirect measurements of a latent continuous Markov process Z , *i.e.* this part of the network represents a KFM. In addition, the process Z depends on the hidden states I , which in our example are first order Markov. Like the HMM, this model can be used in situations, where the observations do not follow the same model all the time, but can follow different models at different times, but here the models are KFMs. Applications include modeling piece-wise linear time series, which for example can be used for monitoring purposes, see *e.g.* Bøttcher, Milsgaard and Mortensen (1995).

Notice that because of the first order Markov property assumed for HMMs, KFMs and MKFMs, these models could have been represented by using only the first two time points, as the structure repeats over time.

4.4 Vector Autoregressive processes

Another classical time series model is the Vector Autoregressive process (VAR) of Markov order p . This model is equivalent to a DBN of Markov order p , in which all the variables are continuous and observed. So the local probability distributions in this model are Gaussian linear regressions on the continuous parents.

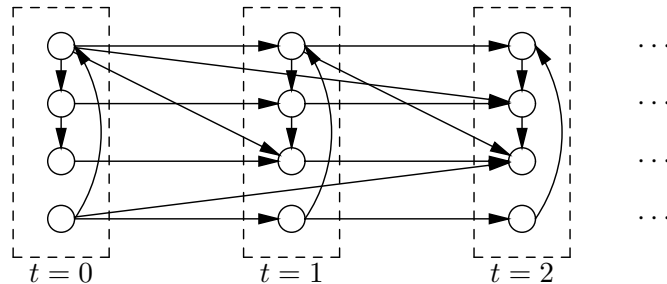


Figure 6: A Vector Autoregressive process.

In Figure 6, an example of a VAR process of order 2 is given. Because of the second order Markov property, this model can be represented by the first three time points.

In the next section, we will develop a method for learning the parameters and structure of a DBN. In this paper we assume that data are complete, so we can not learn networks with hidden variables. Therefore, the HMM, the KFM and the MKFM can only be learned with these methods, if a training dataset with complete data is available.

5 Learning DBNs with Mixed Variables

Learning first order Markov DBNs in the purely discrete case with no static variables is described in Friedman et al. (1998). Here we will consider learning DBNs with mixed variables for the case with both time varying and static variables. Further, we will also illustrate how to learn DBNs with higher Markov order and how to learn this order.

As noted in Murphy (2002), learning DBNs is, because of the way DBNs are defined, just a simple extension of learning BNs. This also applies for DBNs

with mixed variables, so we will use the theory for learning Bayesian networks with mixed variables, described in Bøttcher (2001).

5.1 Parameter Learning

To learn the parameters for a given DAG, we use a Bayesian approach. We specify a prior distribution of a parameter θ , use a random sample d from the probability distribution $p(x|\theta)$ and obtain the posterior distribution by using Bayes' theorem

$$p(\theta|d) \propto p(d|\theta)p(\theta).$$

The proportionality constant is determined by the relation $\int_{\Theta} p(\theta|d)d\theta = 1$, where Θ is the parameter space.

To obtain closed formed expressions, we use conjugate distributions of the parameters.

We assume that the parameters associated with $B_{\bar{0}}$ and B_{\rightarrow} are independent. Further, for the parameters in respectively $B_{\bar{0}}$ and B_{\rightarrow} , we assume that the parameters associated with one variable is independent of the parameters associated with the other variables and that the parameters are independent for each configuration of the discrete parents, *i.e.*

$$\begin{aligned} p(\theta) &= p(\theta^{\bar{0}})p(\theta^{\rightarrow}) \\ &= \prod_{\delta \in \Delta_{\bar{0}}} \prod_{i_{\text{pa}(\delta)} \in \mathcal{I}_{\text{pa}(\delta)}} p(\theta_{\delta}^{\bar{0}} | i_{\text{pa}(\delta)}) \prod_{\gamma \in \Gamma_{\bar{0}}} \prod_{i_{\text{pa}(\gamma)} \in \mathcal{I}_{\text{pa}(\gamma)}} p(\theta_{\gamma}^{\bar{0}} | i_{\text{pa}(\gamma)}) \quad (5) \\ &\times \prod_{\delta \in \Delta_{\rightarrow}} \prod_{i_{\text{pa}(\delta)} \in \mathcal{I}_{\text{pa}(\delta)}} p(\theta_{\delta}^{\rightarrow} | i_{\text{pa}(\delta)}) \prod_{\gamma \in \Gamma_{\rightarrow}} \prod_{i_{\text{pa}(\gamma)} \in \mathcal{I}_{\text{pa}(\gamma)}} p(\theta_{\gamma}^{\rightarrow} | i_{\text{pa}(\gamma)}). \end{aligned}$$

We refer to this as *parameter independence*. Notice though that it is slightly different than parameter independence for ordinary Bayesian networks, as we here assume that the parameters in B_{\rightarrow} are the same for each time point $t = 1, \dots, T$.

In the case with higher order Markov properties, parameter independence is also valid for the parameters in the networks B_1, \dots, B_{m-1} .

We also assume *complete data*, *i.e.* each case $^c x$ in a dataset d contains one instance of every random variable in the network. With this we can show posterior parameter independence. The likelihood $p(d|\theta)$ can be written as follows.

$$\begin{aligned}
p(d|\theta) &= \prod_{c \in d} p(c x^0, \dots, c x^T, c x^s | \theta) \\
&= \prod_{c \in d} \left(p(c x^0 | \theta^{\bar{0}}) \prod_{t=1}^T p(c x^t | c x^{t-1}, c x^s, \theta^{\rightarrow}) \right).
\end{aligned}$$

As the time series has stationary dynamics, we see that for each observations of the variables in B_0 , there are T observations of the variables in B_{\rightarrow} .

To simplify the expressions, we consider the likelihood terms for $B_{\bar{0}}$ and B_{\rightarrow} separately. For $B_{\bar{0}}$ we have that

$$\prod_{c \in d} p(c x^{\bar{0}} | \theta^{\bar{0}}) = \prod_{c \in d} \prod_{\delta \in \Delta_{\bar{0}}} p(c_i^{\bar{0}} | c_{\text{pa}(\delta)}^{\bar{0}}, \theta_{\delta | i_{\text{pa}(\delta)}}^{\bar{0}}) \prod_{\gamma \in \Gamma_{\bar{0}}} p(c_y^{\bar{0}} | c_{\text{pa}(\gamma)}^{\bar{0}}, c_{\text{pa}(\gamma)}^{\bar{0}}, \theta_{\gamma | i_{\text{pa}(\gamma)}}^{\bar{0}}),$$

where c_i and c_y respectively denotes the discrete part and the continuous part of a case $c x$. Our goal is to show posterior parameter independence, so we must show that the likelihood, like the parameters, factorizes into a product over nodes and a product over the configuration of the discrete parents of a node. Therefore we write this part of the likelihood as

$$\begin{aligned}
\prod_{c \in d} p(c x^{\bar{0}} | \theta^{\bar{0}}) &= \prod_{\delta \in \Delta} \prod_{i_{\text{pa}(\delta)} \in \mathcal{I}_{\text{pa}(\delta)}} \prod_{c: c_{\text{pa}(\delta)}^{\bar{0}} = i_{\text{pa}(\delta)}^{\bar{0}}} p(c_i^{\bar{0}} | i_{\text{pa}(\delta)}^{\bar{0}}, \theta_{\delta | i_{\text{pa}(\delta)}}^{\bar{0}}) \\
&\quad \times \prod_{\gamma \in \Gamma} \prod_{i_{\text{pa}(\gamma)} \in \mathcal{I}_{\text{pa}(\gamma)}} \prod_{c: c_{\text{pa}(\gamma)}^{\bar{0}} = i_{\text{pa}(\gamma)}^{\bar{0}}} p(c_y^{\bar{0}} | c_{\text{pa}(\gamma)}^{\bar{0}}, i_{\text{pa}(\gamma)}^{\bar{0}}, \theta_{\gamma | i_{\text{pa}(\gamma)}}^{\bar{0}}). \tag{6}
\end{aligned}$$

We see that the product over cases is split up into a product over the configurations of the discrete parents and a product over those cases, where the configuration of the discrete parents is the same as the currently processed configuration. Notice however that some of the parent configurations might not be represented in the database, in which case the product over cases with this parent configuration just adds nothing to the overall product.

In the case with m th order Markov properties, the likelihood terms for all the networks B_1, \dots, B_{m-1} , can be written as in (6).

The likelihood part from B_{\rightarrow} is given as,

$$\begin{aligned}
& \prod_{c \in d} \prod_{t=1}^T p(c_x^t | c_x^{t-1}, c_x^s, \theta^{\rightarrow}) \\
&= \prod_{c \in d} \prod_{t=1}^T \left(\prod_{\delta \in \Delta_t} p(c_\delta^t | i_{\text{pa}(\delta)}^{\rightarrow}, \theta_{\delta | i_{\text{pa}(\delta)}^{\rightarrow}}^{\rightarrow}) \prod_{\gamma \in \Gamma_t} p(c_\gamma^t | y_{\text{pa}(\gamma)}^{\rightarrow}, i_{\text{pa}(\gamma)}^{\rightarrow}, \theta_{\gamma | i_{\text{pa}(\gamma)}^{\rightarrow}}^{\rightarrow}) \right) \\
&= \prod_{\delta \in \Delta_t} \prod_{i_{\text{pa}(\delta)}^{\rightarrow} \in \mathcal{I}_{\text{pa}(\delta)}^{\rightarrow}} \prod_{t=1}^T \prod_{c: c_{\text{pa}(\delta)}^{\rightarrow} = i_{\text{pa}(\delta)}^{\rightarrow}} p(c_\delta^t | i_{\text{pa}(\delta)}^{\rightarrow}, \theta_{\delta | i_{\text{pa}(\delta)}^{\rightarrow}}^{\rightarrow}) \\
&\times \prod_{\gamma \in \Gamma_t} \prod_{i_{\text{pa}(\gamma)}^{\rightarrow} \in \mathcal{I}_{\text{pa}(\gamma)}^{\rightarrow}} \prod_{t=1}^T \prod_{c: c_{\text{pa}(\gamma)}^{\rightarrow} = i_{\text{pa}(\gamma)}^{\rightarrow}} p(c_\gamma^t | y_{\text{pa}(\gamma)}^{\rightarrow}, i_{\text{pa}(\gamma)}^{\rightarrow}, \theta_{\gamma | i_{\text{pa}(\gamma)}^{\rightarrow}}^{\rightarrow})
\end{aligned} \tag{7}$$

The product over cases is split up as before. Further, this is also a product over time points, so for each time point t , we take the product over cases with a specific configuration of the discrete parents.

Posterior parameter independence now follows from (5), (6) and (7),

$$\begin{aligned}
p(\theta | d) &= p(\theta^{\bar{0}} | d) p(\theta^{\rightarrow} | d) \\
&= \prod_{\delta \in \Delta_{\bar{0}}} \prod_{i_{\text{pa}(\delta)} \in \mathcal{I}_{\text{pa}(\delta)}} p(\theta_{\delta | i_{\text{pa}(\delta)}}^{\bar{0}} | d) \prod_{\gamma \in \Gamma_{\bar{0}}} \prod_{i_{\text{pa}(\gamma)} \in \mathcal{I}_{\text{pa}(\gamma)}} p(\theta_{\gamma | i_{\text{pa}(\gamma)}}^{\bar{0}} | d) \\
&\times \prod_{\delta \in \Delta_t} \prod_{i_{\text{pa}(\delta)} \in \mathcal{I}_{\text{pa}(\delta)}} p(\theta_{\delta | i_{\text{pa}(\delta)}}^{\rightarrow} | d) \prod_{\gamma \in \Gamma_t} \prod_{i_{\text{pa}(\gamma)} \in \mathcal{I}_{\text{pa}(\gamma)}} p(\theta_{\gamma | i_{\text{pa}(\gamma)}}^{\rightarrow} | d).
\end{aligned}$$

So due to parameter independence and complete data, the parameters stay independent given data. This means that we can learn the parameters in the local distributions independently and also that the parameters in $B_{\bar{0}}$ and B_{\rightarrow} can be learned independently. Again, if the time series is m th order Markov, posterior parameter independence also follows and we can learn the parameters in $B_{\bar{0}}, \dots, B_{m-1}$ and B_{\rightarrow} independently.

Consider for example in $B_{\bar{0}}$ a parameter for a discrete node δ , with a specific configuration of the discrete parents, $i_{\text{pa}(\delta)}$. The posterior distribution of $\theta_{\delta | i_{\text{pa}(\delta)}}^{\bar{0}}$

is by Bayes' theorem found as

$$p(\theta_{\delta|i_{\text{pa}(\delta)}}^{\tilde{0}}|d) \propto \prod_{c: c_{\text{pa}(\delta)}^{\tilde{0}} = i_{\text{pa}(\delta)}^{\tilde{0}}} p(c_{\delta}^{\tilde{0}}|i_{\text{pa}(\delta)}^{\tilde{0}}, \theta_{\delta|i_{\text{pa}(\delta)}}^{\tilde{0}}) p(\theta_{\delta|i_{\text{pa}(\delta)}}^{\tilde{0}}).$$

Thus $\theta_{\delta|i_{\text{pa}(\delta)}}^{\tilde{0}}$ is updated with the cases in the database, where the configuration of the parents of δ is $i_{\text{pa}(\delta)}^{\tilde{0}}$.

Likewise with a parameter $\theta_{\delta|i_{\text{pa}(\delta)}}^{\rightarrow}$ in B_{\rightarrow} ,

$$p(\theta_{\delta|i_{\text{pa}(\delta)}}^{\rightarrow}|d) \propto \prod_{t=1}^T \prod_{c: c_{\text{pa}(\delta)}^{\rightarrow} = i_{\text{pa}(\delta)}^{\rightarrow}} p(c_{\delta}^t|i_{\text{pa}(\delta)}^{\rightarrow}, \theta_{\delta|i_{\text{pa}(\delta)}}^{\rightarrow}) p(\theta_{\delta|i_{\text{pa}(\delta)}}^{\rightarrow}).$$

Here $\theta_{\delta|i_{\text{pa}(\delta)}}^{\rightarrow}$ is, for each time point t , updated with the cases in the database for which the configuration of the parents of δ is $i_{\text{pa}(\delta)}^{\rightarrow}$.

In the next sections we will introduce the conjugate distributions of the parameters and show how these are learned. The only difference in how the parameters in $B_{\tilde{0}}$ and B_{\rightarrow} are learned, is the set of cases used to learn them. So in the following we do not differentiate between the parameters in $B_{\tilde{0}}$ and B_{\rightarrow} .

5.2 Learning the Discrete Variables

As described in DeGroot (1970), a conjugate family for multinomial observations is the family of Dirichlet distributions. Let the prior distribution of $\theta_{\delta|i_{\text{pa}(\delta)}}$ be a Dirichlet distribution, \mathcal{D} , with hyperparameters $\alpha_{\delta|i_{\text{pa}(\delta)}} = (\alpha_{i_{\delta}|i_{\text{pa}(\delta)}})_{i_{\delta} \in \mathcal{I}_{\delta}}$, also written as

$$(\theta_{\delta|i_{\text{pa}(\delta)}}|\alpha_{\delta|i_{\text{pa}(\delta)}}) \sim \mathcal{D}(\alpha_{\delta|i_{\text{pa}(\delta)}}).$$

The posterior distribution is then given as

$$(\theta_{\delta|i_{\text{pa}(\delta)}}|d) \sim \mathcal{D}(\alpha_{\delta|i_{\text{pa}(\delta)}} + n_{\delta|i_{\text{pa}(\delta)}}),$$

where the vector $n_{\delta|i_{\text{pa}(\delta)}} = (n_{i_{\delta}|i_{\text{pa}(\delta)}})_{i_{\delta} \in \mathcal{I}_{\delta}}$, also called the counts, denotes the number of observations in d where δ and $\text{pa}(\delta)$ have that specific configuration.

Again $\alpha_{\delta|i_{\text{pa}(\delta)}}$ and $n_{\delta|i_{\text{pa}(\delta)}}$ can be indexed by $\tilde{0}$ and \rightarrow , according to $B_{\tilde{0}}$ and B_{\rightarrow} . So for $B_{\tilde{0}}$ we have that $n_{i_{\delta}|i_{\text{pa}(\delta)}}^{\tilde{0}}$ is the number of cases in d with a given configuration of δ and $\text{pa}(\delta)$. Likewise for B_{\rightarrow} , where $n_{i_{\delta}|i_{\text{pa}(\delta)}}^{\rightarrow}$ is the number of cases in d and for every time point $t = 1, \dots, T$, with this configuration of δ and $\text{pa}(\delta)$.

5.3 Learning the Continuous Variables

For the continuous variables we can write the local probability distributions as

$$(Y_\gamma | y_{\text{pa}(\gamma)}, i_{\text{pa}(\gamma)}, \theta_\gamma | i_{\text{pa}(\gamma)}) \sim \mathcal{N}(z_{\text{pa}(\gamma)}(m_{\gamma|i_{\text{pa}(\gamma)}}, \beta_{\gamma|i_{\text{pa}(\gamma)}})^T, \sigma_{\gamma|i_{\text{pa}(\gamma)}}^2),$$

where $z_{\text{pa}(\gamma)} = (1, y_{\text{pa}(\gamma)})$. A standard conjugate family for these observations is the family of Gaussian-inverse gamma distributions. Let the prior joint distribution of $(m_{\gamma|i_{\text{pa}(\gamma)}}, \beta_{\gamma|i_{\text{pa}(\gamma)}})$ and $\sigma_{\gamma|i_{\text{pa}(\gamma)}}^2$ be as follows.

$$\begin{aligned} (m_{\gamma|i_{\text{pa}(\gamma)}}, \beta_{\gamma|i_{\text{pa}(\gamma)}} | \sigma_{\gamma|i_{\text{pa}(\gamma)}}^2) &\sim \mathcal{N}_{k+1}(\mu_{\gamma|i_{\text{pa}(\gamma)}}, \sigma_{\gamma|i_{\text{pa}(\gamma)}}^2 \tau_{\gamma|i_{\text{pa}(\gamma)}}^{-1}), \\ (\sigma_{\gamma|i_{\text{pa}(\gamma)}}^2) &\sim \mathcal{IG}\left(\frac{\rho_{\gamma|i_{\text{pa}(\gamma)}}}{2}, \frac{\phi_{\gamma|i_{\text{pa}(\gamma)}}}{2}\right). \end{aligned}$$

If $\theta_{\gamma|i_{\text{pa}(\gamma)}}$ is a parameter in $B_{\bar{0}}$, the posterior distribution is found by

$$p(\theta_{\gamma|i_{\text{pa}(\gamma)}}^{\bar{0}} | d) \propto \prod_{c: c_{\text{pa}(\gamma)}^{\bar{0}} = i_{\text{pa}(\gamma)}^{\bar{0}}} p(c_{\gamma}^{\bar{0}} | y_{\text{pa}(\gamma)}^{\bar{0}}, i_{\text{pa}(\gamma)}^{\bar{0}}, \theta_{\gamma|i_{\text{pa}(\gamma)}}^{\bar{0}}) p(\theta_{\gamma|i_{\text{pa}(\gamma)}}^{\bar{0}}).$$

We now join all the observations $c_{\gamma}^{\bar{0}}$ for which $c_{\text{pa}(\gamma)}^{\bar{0}} = i_{\text{pa}(\gamma)}^{\bar{0}}$ in a vector $b_{\gamma}^{\bar{0}}$, i.e.

$$b_{\gamma}^{\bar{0}} = (c_{\gamma}^{\bar{0}})_{c_{\text{pa}(\gamma)}^{\bar{0}} = i_{\text{pa}(\gamma)}^{\bar{0}}}.$$

The same is done with the observations of the continuous parents of γ , i.e.

$$b_{\text{pa}(\gamma)}^{\bar{0}} = (c_{\text{pa}(\gamma)}^{\bar{0}})_{c_{\text{pa}(\gamma)}^{\bar{0}} = i_{\text{pa}(\gamma)}^{\bar{0}}}.$$

The posterior distribution of $\theta_{\gamma|i_{\text{pa}(\gamma)}}^{\bar{0}}$ can now be written as

$$p(\theta_{\gamma|i_{\text{pa}(\gamma)}}^{\bar{0}} | d) \propto p(b_{\gamma}^{\bar{0}} | b_{\text{pa}(\gamma)}^{\bar{0}}, i_{\text{pa}(\gamma)}^{\bar{0}}, \theta_{\gamma|i_{\text{pa}(\gamma)}}^{\bar{0}}) p(\theta_{\gamma|i_{\text{pa}(\gamma)}}^{\bar{0}}).$$

As the distribution, $p(c_{\gamma}^{\bar{0}} | y_{\text{pa}(\gamma)}^{\bar{0}}, i_{\text{pa}(\gamma)}^{\bar{0}}, \theta_{\gamma|i_{\text{pa}(\gamma)}}^{\bar{0}})$, is a Gaussian distribution, then $p(b_{\gamma}^{\bar{0}} | b_{\text{pa}(\gamma)}^{\bar{0}}, i_{\text{pa}(\gamma)}^{\bar{0}}, \theta_{\gamma|i_{\text{pa}(\gamma)}}^{\bar{0}})$ is a multivariate Gaussian distribution. The covariance matrix is diagonal as all the cases in the database are independent. This way we consider all the cases in a *batch*.

The same formulation applies for parameters in B_{\rightarrow} . Notice that the observations included in b_{γ}^{\rightarrow} and $b_{\text{pa}(\gamma)}^{\rightarrow}$ are taken for each time point $t = 1, \dots, T$.

The posterior distribution is found to be

$$\begin{aligned} (m_{\gamma|i_{\text{pa}(\gamma)}}, \beta_{\gamma|i_{\text{pa}(\gamma)}} | \sigma_{\gamma|i_{\text{pa}(\gamma)}}^2, d) &\sim \mathcal{N}_{k+1}(\mu'_{\gamma|i_{\text{pa}(\gamma)}}, \sigma_{\gamma|i_{\text{pa}(\gamma)}}^2 (\tau_{\gamma|i_{\text{pa}(\gamma)}}^{-1})') \\ (\sigma_{\gamma|i_{\text{pa}(\gamma)}}^2 | d) &\sim \mathcal{IG}\left(\frac{\rho'_{\gamma|i_{\text{pa}(\gamma)}}}{2}, \frac{\phi'_{\gamma|i_{\text{pa}(\gamma)}}}{2}\right), \end{aligned}$$

where

$$\begin{aligned} \tau'_{\gamma|i_{\text{pa}(\gamma)}} &= \tau_{\gamma|i_{\text{pa}(\gamma)}} + (z_{\text{pa}(\gamma)}^b)^{\text{T}} z_{\text{pa}(\gamma)}^b \\ \mu'_{\gamma|i_{\text{pa}(\gamma)}} &= (\tau'_{\gamma|i_{\text{pa}(\gamma)}})^{-1} (\tau_{\gamma|i_{\text{pa}(\gamma)}} \mu_{\gamma|i_{\text{pa}(\gamma)}} + (z_{\text{pa}(\gamma)}^b)^{\text{T}} y_{\gamma}^b) \\ \rho'_{\gamma|i_{\text{pa}(\gamma)}} &= \rho_{\gamma|i_{\text{pa}(\gamma)}} + |b| \\ \phi'_{\gamma|i_{\text{pa}(\gamma)}} &= \phi_{\gamma|i_{\text{pa}(\gamma)}} + (y_{\gamma}^b - z_{\text{pa}(\gamma)}^b \mu'_{\gamma|i_{\text{pa}(\gamma)}})^{\text{T}} y_{\gamma}^b \\ &\quad + (\mu_{\gamma|i_{\text{pa}(\gamma)}} - \mu'_{\gamma|i_{\text{pa}(\gamma)}})^{\text{T}} \tau_{\gamma|i_{\text{pa}(\gamma)}} \mu_{\gamma|i_{\text{pa}(\gamma)}}, \end{aligned}$$

where $|b|$ denotes the number of observations in y_{γ}^b .

5.4 Structure Learning

To learn the structure of a DBN, we again use a Bayesian approach and calculate the posterior probability of a DAG D given data d ,

$$p(D|d) \propto p(d|D)p(D), \quad (8)$$

where $p(d|D)$ is the marginal likelihood of D and $p(D)$ is the prior probability of D .

In this paper we choose, for simplicity, to let all DAGs be equally likely a priori and therefore we use the measure

$$p(D|d) \propto p(d|D).$$

We refer to the above measure as a *network score*. We can, in principle, calculate the network score for all possible DAGs and then select the one with the highest score (or, if using model averaging, select a few with high score). In most situations however, there are too many different DAGs to evaluate and some kind of search strategy must be employed, see *e.g.* Cooper and Herskovits (1992).

The marginal likelihood $p(d|D)$ is given as follows.

$$\begin{aligned}
p(d|D) &= \int_{\theta \in \Theta} p(d|\theta, D)p(\theta|D)d\theta \\
&= \prod_{\delta \in \Delta_{\bar{0}}} \prod_{i_{\text{pa}(\delta)} \in \mathcal{I}_{\text{pa}(\delta)}} \int \prod_{c: c_{\text{pa}(\delta)}^{\bar{0}} = i_{\text{pa}(\delta)}^{\bar{0}}} p(c_{\bar{0}}^{\delta} | i_{\text{pa}(\delta)}^{\bar{0}}, \theta_{\delta}^{\bar{0}} | i_{\text{pa}(\delta)}^{\bar{0}}, D) p(\theta_{\delta}^{\bar{0}} | i_{\text{pa}(\delta)}^{\bar{0}} | D) d\theta_{\delta}^{\bar{0}} | i_{\text{pa}(\delta)}^{\bar{0}} \times \\
&\quad \prod_{\gamma \in \Gamma_{\bar{0}}} \prod_{i_{\text{pa}(\gamma)} \in \mathcal{I}_{\text{pa}(\gamma)}} \int \prod_{c: c_{\text{pa}(\gamma)}^{\bar{0}} = i_{\text{pa}(\gamma)}^{\bar{0}}} p(c_{\bar{0}}^{\gamma} | i_{\text{pa}(\gamma)}^{\bar{0}}, \theta_{\gamma}^{\bar{0}} | i_{\text{pa}(\gamma)}^{\bar{0}}, D) p(\theta_{\gamma}^{\bar{0}} | i_{\text{pa}(\gamma)}^{\bar{0}} | D) d\theta_{\gamma}^{\bar{0}} | i_{\text{pa}(\gamma)}^{\bar{0}} \times \\
&\quad \prod_{\delta \in \Delta_t} \prod_{i_{\text{pa}(\delta)} \in \mathcal{I}_{\text{pa}(\delta)}} \int \prod_{t=1}^T \prod_{c: c_{\text{pa}(\delta)}^{\rightarrow} = i_{\text{pa}(\delta)}^{\rightarrow}} p(c_{\rightarrow}^{\delta} | i_{\text{pa}(\delta)}^{\rightarrow}, \theta_{\delta}^{\rightarrow} | i_{\text{pa}(\delta)}^{\rightarrow}, D) p(\theta_{\delta}^{\rightarrow} | i_{\text{pa}(\delta)}^{\rightarrow} | D) d\theta_{\delta}^{\rightarrow} | i_{\text{pa}(\delta)}^{\rightarrow} \times \\
&\quad \prod_{\gamma \in \Gamma_t} \prod_{i_{\text{pa}(\gamma)} \in \mathcal{I}_{\text{pa}(\gamma)}} \int \prod_{t=1}^T \prod_{c: c_{\text{pa}(\gamma)}^{\rightarrow} = i_{\text{pa}(\gamma)}^{\rightarrow}} p(c_{\rightarrow}^{\gamma} | i_{\text{pa}(\gamma)}^{\rightarrow}, \theta_{\gamma}^{\rightarrow} | i_{\text{pa}(\gamma)}^{\rightarrow}, D) p(\theta_{\gamma}^{\rightarrow} | i_{\text{pa}(\gamma)}^{\rightarrow} | D) d\theta_{\gamma}^{\rightarrow} | i_{\text{pa}(\gamma)}^{\rightarrow}
\end{aligned}$$

We see that the marginal likelihood $p(d|D)$ factorizes into a product over terms involving only one node and its parents, called local marginal likelihoods, so the network score is *decomposable*. This also means that the likelihood factorizes into terms related to $B_{\bar{0}}$ and terms related to B_{\rightarrow} . For m th order Markov time series, the likelihood factorizes in a similar manner into terms related to $B_{\bar{0}}, \dots, B_{m-1}$ and B_{\rightarrow} .

Because of the way we specified the possible parent sets of variables in $B_{\bar{0}}$ and in B_{\rightarrow} , we can find the best DAG (the one with the highest network score) by finding the best DAG for $B_{\bar{0}}$ and the best DAG for B_{\rightarrow} . So we can learn the structure of $B_{\bar{0}}$ and B_{\rightarrow} independently and we can learn them just as we learn ordinary Bayesian networks with mixed variables as described in Bøttcher (2001). This also applies for m th order Markov time series in which we can learn the structure of $B_{\bar{0}}, \dots, B_{m-1}$ and B_{\rightarrow} independently.

In the following we do not distinguish between variables in $B_{\bar{0}}$ and B_{\rightarrow} , as the terms presented apply for both $B_{\bar{0}}$ and B_{\rightarrow} .

The network score contribution from the discrete variables in a network is given by

$$\prod_{\delta \in \Delta} \prod_{i_{\text{pa}(\delta)} \in \mathcal{I}_{\text{pa}(\delta)}} \frac{\Gamma(\alpha_{+\delta} | i_{\text{pa}(\delta)})}{\Gamma(\alpha_{+\delta} | i_{\text{pa}(\delta)} + n_{+\delta} | i_{\text{pa}(\delta)})} \prod_{i_{\delta} \in \mathcal{I}_{\delta}} \frac{\Gamma(\alpha_{i_{\delta}} | i_{\text{pa}(\delta)} + n_{i_{\delta}} | i_{\text{pa}(\delta)})}{\Gamma(\alpha_{i_{\delta}} | i_{\text{pa}(\delta)})}. \quad (9)$$

For the continuous variables, the local marginal likelihoods are non-central t

distributions with $\rho_{\gamma|i_{\text{pa}(\gamma)}}$ degrees of freedom, location vector $z_{\text{pa}(\gamma)}^b \mu_{\gamma|i_{\text{pa}(\gamma)}}$ and scale parameter $s_{\gamma|i_{\text{pa}(\gamma)}} = \frac{\phi_{\gamma|i_{\text{pa}(\gamma)}}}{\rho_{\gamma|i_{\text{pa}(\gamma)}}} (I + (z_{\text{pa}(\gamma)}^b) \tau_{\gamma|i_{\text{pa}(\gamma)}}^{-1} (z_{\text{pa}(\gamma)}^b)^{\text{T}})$. The index b is defined as in Section 5.3.

The network score contribution from the continuous variables is given by

$$\prod_{\gamma \in \Gamma} \prod_{i_{\text{pa}(\gamma)} \in \mathcal{I}_{\text{pa}(\gamma)}} \frac{\Gamma((\rho_{\gamma|i_{\text{pa}(\gamma)}} + |b|)/2)}{\Gamma(\rho_{\gamma|i_{\text{pa}(\gamma)}}/2) [\det(\rho_{\gamma|i_{\text{pa}(\gamma)}} s_{\gamma|i_{\text{pa}(\gamma)}} \pi)]^{\frac{1}{2}}} \times \left[1 + \frac{1}{\rho_{\gamma|i_{\text{pa}(\gamma)}}} (y_{\gamma}^b - z_{\text{pa}(\gamma)}^b \mu_{\gamma|i_{\text{pa}(\gamma)}}) s_{\gamma|i_{\text{pa}(\gamma)}}^{-1} (y_{\gamma}^b - z_{\text{pa}(\gamma)}^b \mu_{\gamma|i_{\text{pa}(\gamma)}})^{\text{T}} \right]^{\frac{-(\rho_{\gamma|i_{\text{pa}(\gamma)}} + |b|)}{2}}. \quad (10)$$

The network score is thus the product of (9) and (10).

So if the time series is first order Markov, we can find the best DAG by finding the best DAG for $B_{\bar{0}}$ and the best DAG for B_{\rightarrow} . If it is m th order Markov, we find the best DAGs for $B_{\bar{0}}, \dots, B_{m-1}$ and B_{\rightarrow} .

5.5 Learning the Markov Order

If the Markov order of the time series is unknown, we can learn it by choosing a ‘‘prior’’ order and learn the DBN with this order. The learned order can then be read from the best DAG for B_{\rightarrow} , by determining which time slices X^t has parents from. The slice furthest back in time will give the order.

It is important that the prior order is chosen high enough to ensure that no order higher than this is better in describing the time series. How high this prior order in practice should be chosen, depends on any prior information available on the time series, but also of how large a dataset the network is learned from. The higher we choose the order, the more complex the possible DAGs are, with more parameters to estimate and fewer cases to learn them from.

To increase the stability of the search procedure, it could therefore be better to start by learning a DBN with a low Markov order. If the best DAG for B_{\rightarrow} include dependencies up to the chosen order, a network with a higher order should be tried and this should be repeated until no dependencies of higher order reveal themselves. However, with this procedure there is a chance that the

best Markov order will not be learned. If *e.g.* a prior order of three is chosen and the learned network only reveals second order Markov properties, we would with this procedure conclude that the time series is second order Markov, even though the best order could be higher than three. An example of this is shown in Section 7.

Situations can arise, where the Markov order in the initial DAGs is higher than in B_{\rightarrow} . For example, if we have assumed that the time series is third order Markov, we need to learn the structure of $B_{\bar{0}}, B_1, B_2$ and B_{\rightarrow} . Consider now a situation where B_{\rightarrow} is learned to be first order Markov, *i.e.* X^t has only parents in X^t and X^{t-1} , while B_2 is learned to be second order Markov, *i.e.* to have time varying parents from $B_{\bar{0}}$. This is not necessarily a problem, but it should be noted that if we had assumed the first order Markov property, then there would have been more cases to learn the parameters in B_{\rightarrow} by. In such situations, the importance of specifying the initialization of the time series correctly, must be compared to the loss of precision in the distribution of the parameters in B_{\rightarrow} .

6 Specifying Prior Distributions

To learn the structure of the DAG we need to specify prior parameter distributions for all possible DAGs under evaluation. An automated procedure for doing this has been developed for ordinary Bayesian networks. We call it the *master prior procedure*. The procedure is for the purely discrete case treated in Heckerman et al. (1995), for the purely continuous case in Geiger and Heckerman (1994) and for the mixed case in Bøttcher (2001).

We will here give an outline of the procedure and show how it can be used for specifying prior parameter distributions for DBNs.

6.1 The Master Prior Procedure

The idea in the master prior procedure is that from a given Bayesian network, we can deduce parameter priors for any possible DAG. The user just has to specify a *prior Bayesian network*, which is the Bayesian network as he believes it to be. Also, he has to specify an *imaginary sample size*, N , which is a measure of how much confidence he has in the prior network. The procedure works as follows.

1. Specify an imaginary sample size.
2. Specify a prior Bayesian network, *i.e.* a prior DAG and prior local probability distributions. Calculate the joint prior distribution.
3. From the joint prior distribution and the imaginary sample size, the marginal distribution of all parameters in the family consisting of a node and its parents can be determined. We call this a *master prior*.
4. The local parameter priors are now determined by conditioning in these master prior distributions.

This procedure ensures parameter independence. Further, it has the property that if a node has the same set of parents in two different networks, then the local parameter prior for this node will be the same in the two networks. Therefore, we only have to deduce the local parameter prior for a node, given the same set of parents, once. This property is called *parameter modularity*. Finally, the procedure ensures *likelihood equivalence*, that is, if two DAGs represent the same set of conditional independencies, the network score for these two DAGs will be the same.

As an example, we will show how to deduce parameter priors for the discrete nodes.

Let $\Psi = (\Psi_i)_{i \in \mathcal{I}}$ be the parameters for the joint distribution of the discrete variables. The joint prior parameter distribution is assumed to be a Dirichlet distribution

$$p(\Psi) \sim \mathcal{D}(\alpha),$$

with hyperparameters $\alpha = (\alpha_i)_{i \in \mathcal{I}}$. To specify this Dirichlet distribution, we need to specify these hyperparameters. Consider the following relation for the Dirichlet distribution,

$$p(i) = \mathbb{E}(\Psi_i) = \frac{\alpha_i}{N},$$

with $N = \sum_{i \in \mathcal{I}} \alpha_i$. Now we let the probabilities in the prior network be an estimate of $\mathbb{E}(\Psi_i)$, so we only need to determine N in order to calculate the parameters α_i .

We determine N by using the notion of an imaginary data base. We imagine that we have a database of cases, from which we have updated the distribution of Ψ out of total ignorance. The *imaginary sample size* of this imaginary data base is thus N . It expresses how much confidence we have in the dependency structure expressed in the prior network, see Heckerman et al. (1995).

We use this joint distribution to deduce the master prior distribution of the family $A = \delta \cup \text{pa}(\delta)$. Let

$$\alpha_{i_A} = \sum_{j:j_A=i_A} \alpha_j,$$

and let $\alpha_A = (\alpha_{i_A})_{i_A \in \mathcal{I}_A}$. Then the marginal distribution of Ψ_A is Dirichlet, $p(\Psi_A) \sim \mathcal{D}(\alpha_A)$. This is the master prior in the discrete case. Notice that the parameters in the master prior can also be found as

$$\alpha_{i_A} = Np(i_A),$$

where $p(i_A) = \sum_{j:j_A=i_A} p(i)$.

The local parameter priors can now be found by conditioning in these master prior distributions. The conditional distribution of $\Psi_{\delta|i_{\text{pa}(\delta)}}$ is

$$p(\Psi_{\delta|i_{\text{pa}(\delta)}}) \sim \mathcal{D}(\alpha_{\delta|i_{\text{pa}(\delta)}}),$$

with $\alpha_{i_{\delta|i_{\text{pa}(\delta)}}} = \alpha_{i_A}$.

6.2 The Master Prior Procedure for DBNs

For DBNs, the parameter priors can also be found by using the above procedure. Consider a DBN for a first order Markov time series (the procedure is directly extendible to time series with higher order Markov properties). As the DAG from time $t = 1$ and forward repeats itself, the structure of the overall DAG is completely specified by the structure of the first two time slices. So we can specify all the parameter priors we need from a prior network consisting of the variables X^0 and X^1 . Notice that the parameter priors for B_{\rightarrow} are the same as the parameter priors for the parameters in X^1 , as this is the first time point in the time series.

We will also allow for different imaginary sample sizes for the parameters in $B_{\bar{0}}$ and the parameters in B_{\rightarrow} . One reason for this is that the parameters in B_{\rightarrow} are updated with more cases than the parameters in $B_{\bar{0}}$ and therefore might need a stronger prior distribution.

The procedure works almost as the procedure for ordinary Bayesian networks, the only difference being the different imaginary sample sizes.

1. Specify an imaginary sample size, $N^{\bar{0}}$, for $B_{\bar{0}}$, and an imaginary sample size, N^{\rightarrow} , for B_{\rightarrow} .

2. Specify a prior Bayesian network for the first two time slices. Calculate the joint prior distribution.
3. From the joint prior distribution and the imaginary sample size, the master prior for all parameters in a family can be determined. For families including only variables from $X^{\bar{0}}$, the imaginary sample size for $B_{\bar{0}}$ is used and for the other families, the imaginary sample size for B_{\rightarrow} is used.
4. The local parameter priors are now determined by conditioning in the appropriate master prior distribution.

It is obvious that parameter independence and parameter modularity still applies as these properties are not influenced by the use of different imaginary sample sizes. Neither is likelihood equivalence, as variables in $X^{\bar{0}}$ can not have parents from X^1 . This means that parameter priors for two DAGs that represent the same set of conditional independencies, are calculated using the same imaginary sample sizes. So likelihood equivalence also still applies.

As a simple example of the master prior procedure for DBNs, consider a time series for a single discrete variable I^0, \dots, I^T . Assume that the time series is first order Markov. The parameter priors for the DAG in Figure 7 are deduced as follows

$$\alpha_{i^0}^0 = N^0 p(i^0),$$

$$\alpha_{i^t | i^{t-1}}^{\rightarrow} = N^{\rightarrow} p(i^t, i^{t-1}).$$

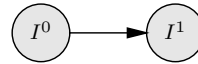


Figure 7: DAG for first order Markov time series.

7 Example

In this section, we will analyze the Wölfer's sunspot numbers using a dynamic Bayesian network. The Wölfer's sunspot numbers are annual measures of sunspot activity, collected from 1700 to 1988. In statistical terms, the sunspot numbers

is a univariate continuous time series Y^0, \dots, Y^{288} . The dataset we use is from Tong (1996).

The sunspot numbers are shown in Figure 8.

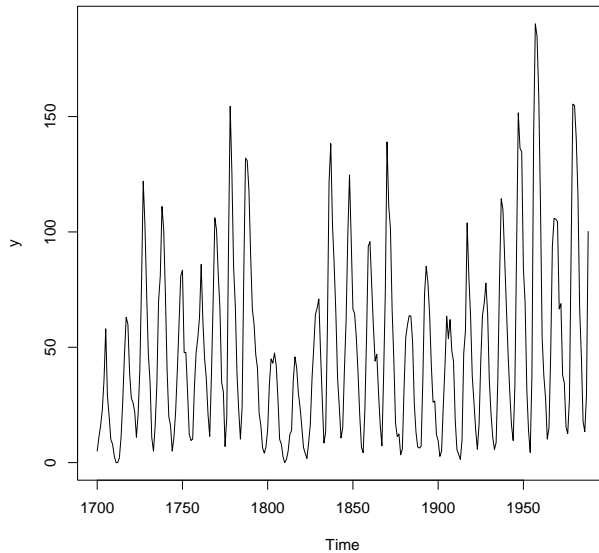


Figure 8: Wölfers' sunspot numbers.

Many statistical investigations of these numbers have been made. Anderson (1971) gives a short review of some of these studies. For example, for annual measures of sunspot activity from 1749 to 1924, Yule (1927) proposed the autoregressive process as a statistical model. He calculated the $AR(p)$ for $p = 2$ and $p = 5$ and found that an $AR(2)$ was sufficient, *i.e.* he estimated the sequence to be second order Markov. Another example is found in Schaerf (1964). She fits an autoregressive model with lags 1, 2, and 9.

Here we will use a DBN as the statistical model and learn the Markov order by structural learning of the DBN. The software package `deal`, see Böttcher and Dethlefsen (2003), is used for the analysis.

Our aim is to learn the Markov order, so we are only interested in learning the structure of B_{\rightarrow} . The structure of the initial networks is not of interest and

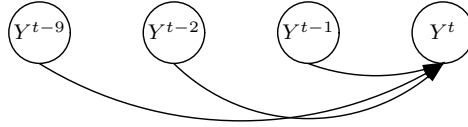


Figure 9: The learned network, B_{\rightarrow} , when an Markov order of 30 is assumed. The variables that do not influence Y^t , have been omitted.

are actually not likely to be determined by learning from the sunspot numbers. These numbers are namely represented by *one* time series, meaning that for the initial networks there are only one observation of each variable.

As the prior network we use the empty network, *i.e.* the one without any arrows. In order to get the right location and scale of the parameters, we estimate the prior probability distribution for the empty network from data, *i.e.* we use the sample mean and the sample variance as the mean and variance in the prior probability distribution.

As the number of observations in the sunspot series is relatively large, we can choose a rather high Markov order for the DBN. Anderson (1971) concludes that the order is not higher than 18. But to be absolutely sure that we capture the best order, we choose an order of 30. The result of the structural learning of B_{\rightarrow} is shown in Figure 9. The variables that do not influence Y^t , have been omitted in the figure. From the result we see that the sunspot numbers can be described by a Markov process of order 9 with lags 1, 2 and 9, *i.e.*

$$Y^t = m + \beta_1 Y^{t-1} + \beta_2 Y^{t-2} + \beta_9 Y^{t-9} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2),$$

with parameter estimates $m = 5.06$, $\beta_1 = 1.21$, $\beta_2 = 0.51$, $\beta_9 = 0.21$ and $\sigma^2 = 267.5$.

The result is in accordance with some of the previous studies, *e.g.* Schaerf (1964) as mentioned earlier. Other studies determine that an second order Markov process is sufficient, *e.g.* Yule (1927). But as mentioned, he only examines an order as high as 5.

We have also tried to learn B_{\rightarrow} using lower Markov order properties. If we *e.g.* use a Markov order of 3, we reach the conclusion that the sunspot numbers are 2. order Markov, with lags 1 and 2. This result is shown in Figure 10. Similarly, if we learn B_{\rightarrow} using the order 2, \dots , 7 or 8, we still reach the conclusion that the sunspot numbers are second order Markov, with lags 1 and 2. This is therefore an example of the importance of choosing the prior Markov order high enough.

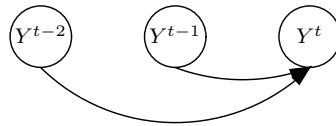


Figure 10: The learned network, B_{\rightarrow} , when the 3. order Markov property is assumed. The variable Y^{t-3} have been omitted as it does not influence Y^t .

As can be seen from Figure 8, the sunspot numbers are periodical with a period of between 10 and 11 years. To determine the period more precisely, we calculate the spectrum,

$$f(\omega) = \sigma^2 \left(1 - \sum_t \beta_t e^{-it\omega}\right)^{-2},$$

see Venables and Ripley (1997), using the parameter estimates obtained from deal.

The spectrum is shown in Figure 11.

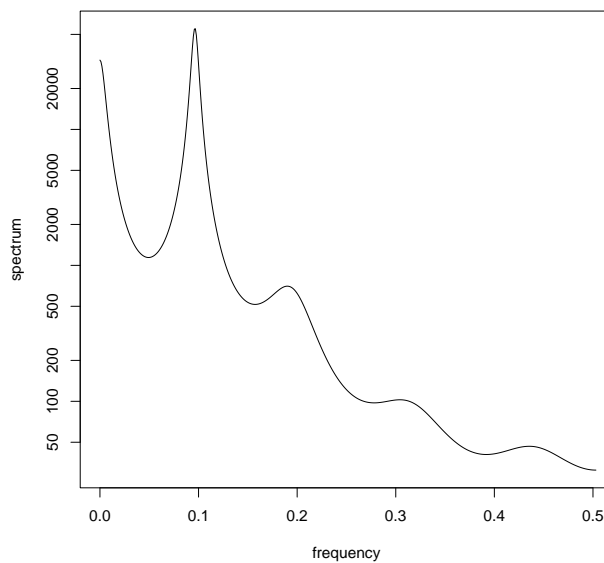


Figure 11: Spectrum of Wölfer's sunspot numbers.

There is a peak at frequency 0.096, which corresponds to a period of $1/0.096 = 10.40$ years. This result is also in accordance with previous studies.

Acknowledgements

This research was supported by Novo Nordisk A/S. Also I would like to thank Claus Dethlefsen for useful discussions and his help with the implementation of the example. Finally, I thank my supervisor Steffen L. Lauritzen for many valuable comments.

References

- Anderson, T. W. (1971). *The Statistical Analysis of Time Series*, John Wiley and Sons, New York.
- Bøttcher, S. G. (2001). Learning Bayesian Networks with Mixed Variables, *Artificial Intelligence and Statistics 2001*, Morgan Kaufmann, San Francisco, CA, USA, pp. 149–156.
- Bøttcher, S. G. and Dethlefsen, C. (2003). deal: A Package for Learning Bayesian Networks, *Journal of Statistical Software* **8**(20): 1–40.
- Bøttcher, S. G., Milsgaard, M. B. and Mortensen, R. S. (1995). *Monitoring by using dynamic linear models - illustrated by tumour markers for cancer*, Master's thesis, Aalborg University.
- Cooper, G. and Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data, *Machine Learning* **9**: 309–347.
- Dean, T. and Kanazawa, K. (1989). A model for reasoning about persistence and causation, *Computational Intelligence* **5**: 142–150.
- DeGroot, M. H. (1970). *Optimal Statistical Decisions*, McGraw-Hill, New York.
- Friedman, N., Murphy, K. P. and Russell, S. (1998). Learning the Structure of Dynamic Probabilistic Networks, *Proceedings of Fourteenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, CA, USA, pp. 139–147.

- Geiger, D. and Heckerman, D. (1994). Learning Gaussian Networks, *Proceedings of Tenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, CA, USA, pp. 235–243.
- Harrison, P. J. and Stevens, C. F. (1976). Bayesian forecasting, *Journal of Royal Statistics* **38**: 205–247.
- Heckerman, D., Geiger, D. and Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data, *Machine Learning* **20**: 197–243.
- Lauritzen, S. L. (1996). *Graphical Models*, Clarendon press, Oxford, New York.
- Murphy, K. P. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*, PhD thesis, University of California, Berkeley.
- Schaerf, M. C. (1964). Estimation of the covariance and autoregressive structure of a stationary time series, *Technical report*, Department of Statistics, Stanford University.
- Tong, H. (1996). *Non-Linear Time Series*, Clarendon Press, Oxford.
- Venables, W. N. and Ripley, B. D. (1997). *Modern Applied Statistics with S-PLUS*, second edn, Springer-Verlag, New York.
- West, M. and Harrison, J. (1989). *Bayesian Forecasting and Dynamic Models*, Springer-Verlag, New York.
- Yule, G. U. (1927). On a method for investigating periodicities in disturbed series with special reference to Wölfer's sunspot numbers, *Philosophical Transactions of the Royal Society, Series A* **226**: 267–298.

