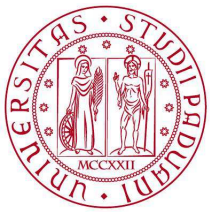


UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Sede Amministrativa: Università degli Studi di Padova

Dipartimento di Scienze Statistiche
SCUOLA DI DOTTORATO DI RICERCA IN SCIENZE STATISTICHE
CICLO XXIII

Measures of Variability for Graphical Models

Direttore della Scuola: Ch.mo Prof. Alessandra Salvan

Supervisore: Ch.mo Prof. Adriana Brogini

Co-supervisore: Ch.mo Prof. Korbinian Strimmer

Dottorando: Marco Scutari

31 Gennaio 2011

人生（ジンセイ）の貸（か）し借（か）りすませ

大晦日（おおみそか）

今敏

Contents

Notation

Random Variables

$\mathbf{X}$	random vector
$X_i \in \mathbf{X}$; $X_1, \ldots, X_p$	random variables
$\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbf{X}$	disjoint subsets of random variables
$F_X(x)$	cumulative distribution function
$\mathsf{E}(X_i)$	expected value
$\mathsf{VAR}(X_i)$	variance
$\mathsf{COV}(X_i, X_j)$	covariance
$\mathsf{COR}(X_i, X_j)$	correlation
$\Sigma = [\sigma_{ij}]$, $i, j = 1, \ldots, k$	covariance matrix
$\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_k]$	eigenvalues of the covariance matrix

Graphs

$\mathbf{V}$	node (vertex) set
$v \in \mathbf{V}$	node (vertex)
$\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbf{V}$	disjoint subsets of nodes
A	arc set
$a \in A$	(directed) arc
$G = (\mathbf{V}, A)$	directed acyclic graph (DAG)
E	edge set
$e \in E$	edge (undirected arc)
$U = (\mathbf{V}, E)$	undirected graph (UG)

Bayesian and Markov Networks

$\perp\!\!\!\perp_G$	graphical separation
$\perp\!\!\!\perp_P$	probabilistic independence
$\mathbf{\Pi}_{X_i}$	parents of node X_i
$\mathbf{S}_{AB}$	d-separating set for nodes A and B
$\mathbf{C}_1$, $\mathbf{C}_2$, ..., $\mathbf{C}_k$	cliques
$\psi_1, \psi_2, \ldots, \psi_k$	potentials associated with the cliques

Conditional Independence Tests and Network Scores

$\mathrm{MI}(X, Y)$	mutual information
$\mathrm{MI}(X, Y \mid \mathbf{Z})$	conditional mutual information
$\mathrm{X}^2(X, Y)$	Pearson's X^2
$\mathrm{X}^2(X, Y \mid \mathbf{Z})$	conditional Pearson's X^2
ρ_{XY}	Pearson's correlation coefficient
$\rho_{XY \mid \mathbf{Z}}$	partial correlation coefficient
$\mathrm{T}(X, Y)$	correlation's t test
$\mathrm{T}(X, Y \mid \mathbf{Z})$	partial correlation's t test
$\mathrm{Z}(X, Y)$	Fisher's Z test
$\mathrm{Z}(X, Y \mid \mathbf{Z})$	conditional Fisher's Z test
$\mathrm{BIC}(\mathbf{X} \mid G)$	Bayesian Information Criterion (BIC)

Multivariate Bernoulli and Trinomial

$\mathbf{B} = [B_1, B_2, \ldots, B_k]^T$	multivariate Bernoulli random vector
$B_1, B_2, \ldots, B_k$, $k \in \mathbb{N}$	Bernoulli random variables
$B_i \sim Ber(p_i)$	Bernoulli r.v. with probability of success p_i
$\mathbf{T} = [T_1, T_2, \ldots, T_k]^T$	multivariate Trinomial random vector
$T_1, T_2, \ldots, T_k$, $k \in \mathbb{N}$	Trinomial random variables
$T_i \sim Tri(p_{i1}, p_{i2}, p_{i3})$	Trinomial r.v. with parameters p_{i1}, p_{i2}, p_{i3}
$\mathbf{p}$	parameter collection of a multivariate Bernoulli or Trinomial random vector
$\tilde{\mathbf{p}}$	reduced parameter collection of a multivariate Bernoulli or Trinomial random vector

Probabilistic Models for Network Structures

$e_{ij} \sim E_{ij}$	Bernoulli r.v. associated with an edge
p_{ij}	probability that $X_i - X_j$ is present in the graph
$a_{ij} \sim A_{ij}$	Trinomial r.v. associated with an arc
$\overrightarrow{a_{ij}}$	arc $X_i \to X_j$, also meaning $\{X_i \to X_j\} \in A$
$\overleftarrow{a_{ij}}$	arc $X_i \leftarrow X_j$, also meaning $\{X_i \leftarrow X_j\} \in A$
$\mathring{a}_{ij}$	event $\{X_i \to X_j, X_i \leftarrow X_j\} \notin A$
$\overrightarrow{p_{ij}}$	probability that $\{X_i \to X_j\} \in A$
$\overleftarrow{p_{ij}}$	probability that $\{X_i \leftarrow X_j\} \in A$
$\mathring{p}_{ij}$	probability that $\{X_i \to X_j, X_i \leftarrow X_j\} \notin A$

Measures of Variability

$\mathrm{tr}(\Sigma)$	trace
$\det(\Sigma)$	determinant
$\|\|\|\Sigma\|\|\|_F$	Frobenius matrix norm
$\mathsf{VAR}_T(\Sigma)$	total variance
$\mathsf{VAR}_G(\Sigma)$	generalized variance
$\mathsf{VAR}_N(\Sigma)$	squared Frobenius matrix norm
$\overline{\mathsf{VAR}}_T(\Sigma)$	normalized total variance
$\overline{\mathsf{VAR}}_G(\Sigma)$	normalized generalized variance
$\overline{\mathsf{VAR}}_N(\Sigma)$	normalized squared Frobenius matrix norm
t_T	test statistic for the total variance
t_{G_1}, t_{G_2}	test statistics for the generalized variance
t_N	Nagao's test statistic
$\hat{\alpha}_T, \hat{\alpha}_{G_1}, \hat{\alpha}_{G_2}, \hat{\alpha}_N$	observed p-values
$\tilde{\alpha}_T, \tilde{\alpha}_{G_1}, \tilde{\alpha}_{G_2}, \tilde{\alpha}_N$	p-values with finite sample correction

Shrinkage estimators

$\hat{\Sigma}$	maximum likelihood estimator of Σ
$\tilde{\Sigma}$	shrinkage estimator of Σ
T	covariance matrix of the target distribution
λ	shrinkage coefficient
λ^*	Ledoit-Wolf estimator for the shrinakge coef.

Chapter 1

Introduction

1.1 Overview

In recent years, Bayesian networks have been successfully applied in several different disciplines, including medicine, biology and epidemiology (see for example Friedman et al. (2000) and Holmes and Jain (2008)). This has been made possible by the rapid evolution of structure learning algorithms, from constraint-based ones (such as PC (Spirtes et al., 2000), Grow-Shrink (Margaritis, 2003), IAMB (Tsamardinos et al., 2003) and its variants (Yaramakala and Margaritis, 2005)) to score-based (such as TABU search (Russell and Norvig, 2009), Greedy Equivalence Search (Chickering, 2002) and genetic algorithms (Larrañaga et al., 1997)) and hybrid ones (such as Max-Min Hill-Climbing (Tsamardinos et al., 2006)).

The main goal in the development of these algorithms has been the reduction of the number of either independence tests or score comparisons needed to learn the structure of the Bayesian network. Their correctness has been proved assuming either very large sample sizes in relation to the number of variables (in the case of Greedy Equivalence Search) or the absence of both false positives and false negatives (in the case of Grow-Shrink and IAMB). In most cases the characteristics of the learned networks have been studied using a small number of reference data sets (Elidan, 2001) as benchmarks, and differences from the true structure have been measured with purely descriptive measures such as Hamming distance (Jungnickel, 2008).

This approach to model validation is not possible for real world data sets, as the true structure of their probability distribution is not known. An alternative is provided by the use of either parametric or nonparametric bootstrap (Efron and Tibshirani, 1993). By applying a learning algorithm to a sufficiently large number of bootstrap samples

it is possible to obtain the empirical probability of any feature of the resulting network (Friedman et al., 1999a), such as the structure of the Markov Blanket of a particular node. The fundamental limit in the interpretation of the results is that the "reasonable" level of confidence for thresholding depends on the data and the learning algorithm.

In this thesis we extend the aforementioned bootstrap-based approach for the inference on the structure of a Bayesian network. The directed graph representing the network structure and its underlying undirected graph are modelled using a multivariate extension of the Bernoulli and Trinomial distributions (Krummenauer, 1998b); each component is associated with an arc. These assumptions allow the derivation of both exact and asymptotic measures of the variability of the network structure or any of its parts (Scutari, 2009). These measures are then applied to some common learning strategies used in literature using the implementation provided by the **bnlearn** R package (Scutari, 2010b).

The outline of the thesis is as follows.

Chapter 2 provides an introduction to the theory of Bayesian networks, including their definition, their fundamental properties and a brief overview of their relationship with Markov networks. Common algorithms for model estimation are then introduced.

Chapter 3 illustrates features of the **bnlearn** package using synthetic data sets.

Chapter 4 introduces the multivariate Bernoulli and Trinomial distributions and derives the probabilistic properties related to their first and second order moments.

Chapter 5 uses the results introduced in the previous chapter to study the behaviour of three univariate measures of multivariate variability from classic literature (total variance, generalized variance and the squared Frobenius matrix norm) when applied to the structure of Bayesian and Markov networks. Exact and asymptotic measures of variability are derived, and their properties are studied under the multivariate Bernoulli and Trinomial assumptions.

Chapter 6 examines the performance of some common learning strategies found in current literature using the measures of variability introduced in this thesis as well as the ones commonly found in literature. The simulations are performed using the **bnlearn** package and accepted reference data sets.

Chapter 7 provides the conclusion and discussion and explores possible directions for future research.

1.2 Main Contributions of the Thesis

The original contributions of this Ph.D. thesis, which includes both theoretical results and practical applications, are listed below.

- The derivation of the probabilistic properties of two multivariate discrete distributions, the multivariate Bernoulli and the multivariate Trinomial distribution (Krummenauer, 1998b), with particular attention to the first two moments and the related quantities (Scutari, 2009). These distributions have proved to be useful in defining a formal model for the structure of a graphical model and studying its behaviour when using bootstrap resampling (Friedman et al., 1999a). Furthermore, these two distributions have applications to the theory of experimental design in addition to being of general interest in probability and random graph theory. The investigation of their information-theoretic properties is also being explored in recent literature (Leonenko and Seleznjev, 2010).

- The study of three univariate measures of multivariate variability (total variance, generalized variance, and the Frobenius matrix norm) to measure the variability of the structure of a Bayesian network (Scutari, 2009). This is of interest in settings where the network structure itself (rather than the parameters of the probability distribution) is the quantity of interest, such as in causal graphical models. The application of the multivariate Bernoulli and Trinomial distributions to this problem allows the derivation of descriptive statistics with a clear interpretation (unlike quantities like Hamming distance) and hypothesis testing. The extension of this approach to other graphical models based on undirected graphs, such as Markov networks, is straightforward.

- Regularized estimation based on the work of Ledoit and Wolf (2003), Hausser and Strimmer (2009) and Schäfer and Strimmer (2005) is applied both to the learning of Bayesian networks and to the subsequent inference based on the multivariate Bernoulli.

- The implementation of an R package (R Development Core Team, 2010) called **bnlearn** for learning and performing inference on Bayesian networks (Scutari, 2010b). **bnlearn** provides a free implementation of several classic and modern learning algorithms and inference procedures (both exact and approximate). Formerly some of these learning algorithms lacked a free-software implementation, and this limited their usefulness in scientific research. Furthermore, each learn-

ing algorithm can be used with many different conditional independence tests or score functions, unlike most available software packages. Both discrete and continuous data are supported, and several reference data sets (Elidan, 2001; Edwards, 2000) are included in the package itself. A novel, parallel implementation of some structure learning algorithms is also introduced.

Thanks to its free-software licensing **bnlearn** has seen a wide adoption and has been constantly updated and improved since its first public release in September 2007. It has been used by the author in two other papers (Chavan et al., 2009; Nagarajan et al., 2010) and it will be prominently featured in a book on graphical models by Springer written by the author with Radharkishnan Nagarajan and Sujay Datta (Nagarajan et al., 2011). Several independent researchers have also started using **bnlearn** in their work, covering such different fields as bioinformatics (Frohlich et al., 2009; Lee, 2010; Lin et al., 2010), image analysis (Hasanat et al., 2010) and social studies (Ge et al., 2010).

Chapter 2

Bayesian Networks

In this chapter we will cover the definition and fundamental properties of Bayesian networks, along with a brief overview of their relationship with Markov networks. Then we will introduce some common algorithms for model estimation present in literature.

2.1 An Introduction to Bayesian Networks

Bayesian networks are a class of *graphical models*, which allow an intuitive representation of the probabilistic structure of multivariate data using graphs. They are composed by two parts:

- a set of random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_p\}$ describing the features of the data. The probability distribution of $\mathbf{X}$ is called the *global distribution* of the data, while the ones associated with each $X_i \in \mathbf{X}$ are called *local distributions*.
- a *directed acyclic graph* (DAG), denoted $G = (\mathbf{V}, A)$. Each node $v \in \mathbf{V}$ is associated with one variable X_i, and they are often referred to interchangeably. The directed arcs $a \in A$ that connect them represent direct stochastic dependencies; so if there is no arc connecting two nodes the corresponding variables are either marginally independent or conditionally independent given a subset of the remaining variables.

The correspondence between the graphical separation (denoted with $\perp\!\!\!\perp_G$) induced by the absence of a particular arc and probabilistic independence (denoted with $\perp\!\!\!\perp_P$) provides a direct and easily interpretable way to express how the features interact with each other. Formally it is called an *independency map* (Pearl, 1988) and is defined as follows.

Definition 2.1. *A graph G is a dependency map (or D-map) of the probabilistic dependence structure P of $\mathbf{X}$ if there is a one-to-one correspondence between the random variables in $\mathbf{X}$ and the nodes $\mathbf{V}$ of G, such that for all disjoint subsets $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ of $\mathbf{X}$ we have*

$$\mathbf{A} \perp\!\!\!\perp_P \mathbf{B} \,|\, \mathbf{C} \Longrightarrow \mathbf{A} \perp\!\!\!\perp_G \mathbf{B} \,|\, \mathbf{C}. \tag{2.1}$$

Similarly, G is an independency map (or I-map) of P if

$$\mathbf{A} \perp\!\!\!\perp_P \mathbf{B} \,|\, \mathbf{C} \Longleftarrow \mathbf{A} \perp\!\!\!\perp_G \mathbf{B} \,|\, \mathbf{C}. \tag{2.2}$$

G is said to be a perfect map of P if it is both a D-map and an I-map, that is

$$\mathbf{A} \perp\!\!\!\perp_P \mathbf{B} \,|\, \mathbf{C} \Longleftrightarrow \mathbf{A} \perp\!\!\!\perp_G \mathbf{B} \,|\, \mathbf{C}, \tag{2.3}$$

and in this case P is said to be isomorphic to G.

Graphical separation, called *d-separation* (as in *directed separation*), is also defined to provide a clear correspondence between the topology of the directed acyclic graph G and the dependence relationships it represents.

Definition 2.2. *If $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are three disjoint subsets of nodes in a directed acyclic graph G, then $\mathbf{C}$ is said to d-separate $\mathbf{A}$ from $\mathbf{B}$, denoted $\mathbf{A} \perp\!\!\!\perp_G \mathbf{B} \,|\, \mathbf{C}$, if along every path between a node in $\mathbf{A}$ and a node in $\mathbf{B}$ there is a node v satisfying one of the following two conditions:*

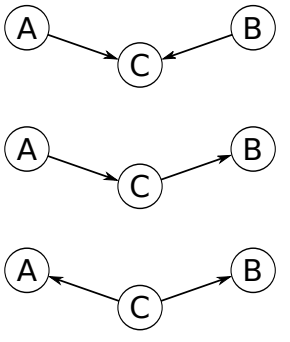
1. *v has converging arcs (i.e. there are two arcs pointing to v from the adjacent nodes in the path) and none of v or its descendants (i.e. the nodes that can be reached from v) are in $\mathbf{C}$.*

2. *v is in $\mathbf{C}$ and does not have converging arcs.*

These two definitions, even though they appear completely abstract in nature, form the core of all the properties which make Bayesian networks a very useful tool in modelling multivariate data.

A very important one is the decomposition of the global distribution into the local ones, which is an application of the *chain rule* of probability and is known as the *Markov property* of Bayesian networks. For discrete data the probability function P of $\mathbf{X}$ can be written as

$$\mathrm{P}\left(\mathbf{X}\right) = \prod_{i=1}^{p} \mathrm{P}\left(X_i \,|\, \mathbf{\Pi}_{X_i}\right) \tag{2.4}$$



$$\mathbf{A} \not\perp\!\!\!\perp_G \mathbf{B} \,|\, \mathbf{C}$$
$$\mathrm{P}(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \mathrm{P}(\mathbf{C} \,|\, \mathbf{A}, \mathbf{B})\, \mathrm{P}(\mathbf{A})\, \mathrm{P}(\mathbf{B})$$

$$\mathbf{A} \perp\!\!\!\perp_G \mathbf{B} \,|\, \mathbf{C} \Rightarrow \mathbf{A} \perp\!\!\!\perp_P \mathbf{B} \,|\, \mathbf{C}$$
$$\begin{aligned} \mathrm{P}(\mathbf{A}, \mathbf{B}, \mathbf{C}) &= \\ &= \mathrm{P}(\mathbf{B} \,|\, \mathbf{C})\, \mathrm{P}(\mathbf{C} \,|\, \mathbf{A})\, \mathrm{P}(\mathbf{A}) \\ &= \mathrm{P}(\mathbf{A} \,|\, \mathbf{C})\, \mathrm{P}(\mathbf{B} \,|\, \mathbf{C})\, \mathrm{P}(\mathbf{C}) \end{aligned}$$

Figure 2.1: Graphical separation, conditional independence and probability decomposition for the three *fundamental connections* (from top to bottom): *converging connection*, *serial connection* and *diverging connection*.

where $\mathbf{\Pi}_{X_i}$ is the set of the parents of X_i. A similar result holds for continuous data, whose global density function $f_{\mathbf{X}}$ can be written as

$$f\left(\mathbf{X}\right) = \prod_{i=1}^{p} f\left(X_i \,|\, \mathbf{\Pi}_{X_i}\right), \tag{2.5}$$

and for mixed data (which includes both discrete and continuous features). A simple application of this property is shown in Figure 2.1 using the *fundamental connections* (Jensen, 2001), the three possible configurations of three nodes and two arcs. In the first step, which is called a *convergent connection* or *v-structure*, C has two incoming arcs (thus violating the second condition of Definition 2.2) and is included in the set of nodes we are conditioning on (thus also violating the first condition). Therefore, we can conclude that C does not d-separate A and B and that, according to Definition 2.1, we cannot conclude that A is independent from B given C. We can also see that given the structure of the converging connection we have $\mathbf{\Pi}_A = \{\varnothing\}$, $\mathbf{\Pi}_B = \{\varnothing\}$ and $\mathbf{\Pi}_C = \{A, B\}$, so according to Equation 2.4

$$\mathrm{P}\left(\mathbf{X}\right) = \mathrm{P}\left(A\right) \mathrm{P}\left(B\right) \mathrm{P}\left(C \,|\, A, B\right). \tag{2.6}$$

Again we can see that A and B are not conditionally independent given C because C depends on their joint distribution. On the other hand, A and B are independent given C in the other two connections (which are called, respectively, *serial* and *diverging*) because both the conditions in Definition 2.2 are satisfied. In the serial connection we

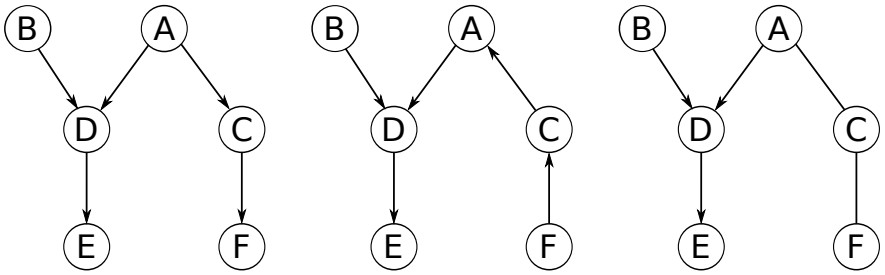


Figure 2.2: Two score equivalent Bayesian networks (on the left and in the middle) and the partially directed graph representing the equivalence class they belong to (on the right). Note that the arc D → E is a *compelled arc*, because its reverse E → D would introduce two additional v-structures in the graph.

have that $\mathbf{\Pi}_A = \{\varnothing\}$, $\mathbf{\Pi}_B = \{C\}$ and $\mathbf{\Pi}_C = \{A\}$, so

$$\mathrm{P}\left(\mathbf{X}\right) = \mathrm{P}\left(A\right)\mathrm{P}\left(C \mid A\right)\mathrm{P}\left(B \mid C\right). \tag{2.7}$$

It is easy to show that the diverging connection results in an equivalent decomposition, which can be obtained from the previous one by repeated applications of Bayes' theorem. Therefore these two networks are said to be *equivalent*, as they lead to the same conditional independence statements and identify the same probability distribution.

Because equivalence is symmetric, reflexive and transitive, this relation defines a set of *equivalence classes* over the possible graph structures of a Bayesian network. Generalizing from the example presented above, it can be shown that the only arcs whose direction is needed to identify an equivalence class are those belonging to a v-structure (Chickering, 1995); changing the direction of any other arc does not change the conditional independence statements encoded by the network, and results in another element of the same equivalence class.

Theorem 2.1. *Two Bayesian networks defined over the same set of variables are equivalent if and only if they have the same skeleton (i.e. the same underlying undirected graph) and the same v-structures.*

Equivalence classes are usually represented with *partially directed acyclic graphs* (PDAGs) in which only arcs belonging to v-structures and those which could introduce additional v-structures or cycles are directed. Such arcs are called *compelled* (Pearl, 1988), because their direction is defined by the equivalence class of the network even though they are not part of any v-structure. A simple example is shown in Figure 2.2.

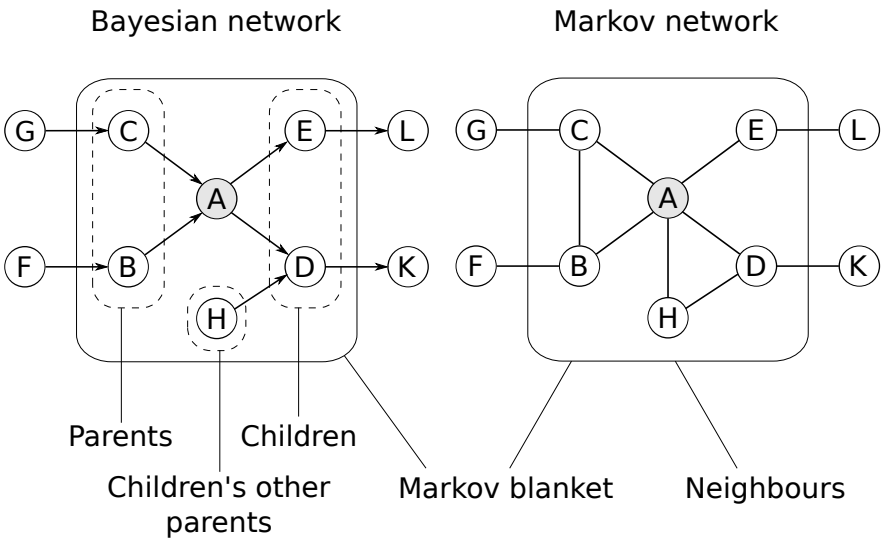


Figure 2.3: The Markov blanket of the node A in a Bayesian network (on the left) and in the corresponding Markov network given by its *moral graph* (on the right). The two graphs express the same dependence structure, so the Markov blanket of A is the same.

Another fundamental quantity of interest that is closely related to Definitions 2.1 and 2.2 is the *Markov blanket* of a node, the set of nodes that completely d-separates that node from the rest of the graph.

Definition 2.3. *The Markov blanket of a node $A \in \mathbf{V}$ is the minimal subset $\mathbf{S}$ of $\mathbf{V}$ such that*

$$A \perp\!\!\!\perp_P \mathbf{V} - \mathbf{S} - A \,|\, \mathbf{S}. \tag{2.8}$$

In any Bayesian network the Markov blanket of a node A is the set consisting of the parents of A, the children of A and all the other nodes sharing a child with A.

In other words, the Markov blanket of a variable A identifies which variables interact directly with A; once their values are known, the behaviour of A does not depend on any other variable in the network. Essentially it is an extension of the concept of neighbourhood (which would contain only the parents and the children of A) adapted to d-separation, which is why it also includes the nodes sharing a child with A. Like the neighbourhoods, Markov blankets are symmetric; if a variable A is in the Markov blanket of B, then B is in the Markov blanket in A. Intuitively this is a direct consequence of the above interpretation of the meaning of the Markov blanket; if A interacts directly with B, then B also interacts directly with A. We can also easily prove that this is the case from Definition 2.3.

Furthermore, Markov blankets provide an easy way to compare Bayesian networks with graphical models based on undirected graphs, which are known as *Markov networks* or *Markov random fields*. The directed acyclic graph of a Bayesian network can be transformed in the undirected graph of the corresponding Markov networks by:

1. connecting the non-adjacent nodes in each v-structure with an undirected arc, usually called an *edge*; this is equivalent to adding an edge between any pair of nodes belonging to the same Markov blanket.

2. ignoring the direction of the other arcs, effectively replacing them with edges.

This transformation is called *moralization* (because it "marries" non-adjacent parents sharing a common child) and the resulting graph is called a *moral graph* (Castillo et al., 1997).

2.2 Bayesian Network Learning Algorithms

Fitting graphical models is called *learning*, a term borrowed from expert systems and artificial intelligence theory, and in general requires a two-step process.

The first step consists in finding the graph structure that encodes the conditional independencies present in the data. Ideally it should coincide with the minimal I-map of the global distribution, or it should at least identify a distribution as close as possible to the correct one in the probability space. This step is called *network structure* or simply *structure learning* (Korb and Nicholson, 2004; Koller and Friedman, 2009), and is similar in approaches and terminology to model selection procedures for classical statistical models. Several algorithms have been presented in literature for this problem, thanks to the application of many results from probability, information and optimization theory. Despite the (sometimes confusing) variety of theoretical backgrounds and terminology they can all be traced to only three approaches: *constraint-based*, *score-based* and *hybrid*.

The second step is called *parameter learning* and, as the name suggests, deals with the estimation of the parameters of the global distribution. Assuming the graph structure is known from the previous step, this can be done efficiently by estimating the parameters of the local distributions.

Both structure and parameter learning are often performed using a combination of numerical algorithms and prior knowledge on the data. Even though significant progress has been made on the performance and scalability of learning algorithms, an

effective use of prior knowledge and relevant theoretical results can still speed up the learning process severalfold and improve the accuracy of the resulting model. Such a boost has been used in the past to overcome the limitations on computational power, leading to the development of the so-called *expert systems* (for real-world examples see the MUNIN (Andreassen et al., 1989), ALARM (Beinlich et al., 1989) and Hailfinder (Abramson et al., 1996) networks); it can still be used today to tackle larger and larger problems and obtain reliable results.

2.2.1 Constraint-based Algorithms

Constraint-based algorithms are based on the seminal work of Pearl on maps and its application to causal graphical models. His *Inductive Causation* (IC) algorithm (Verma and Pearl, 1991) provides a framework for learning the structure of Bayesian networks using conditional independence tests.

The details of the IC algorithm are illustrated in Algorithm 2.1. The first step identifies which pairs of variables are connected by an arc, regardless of its direction; clearly they cannot be independent given any other subset of variables, because they cannot be d-separated. This step can also be seen as a backwards selection procedure starting from the saturated model (the graph in which every pair of nodes is connected

Algorithm 2.1 Inductive Causation Algorithm

1. For each pair of variables A and B in $\mathbf{V}$ search for set $\mathbf{S}_{AB} \subset V$ such that A and B are independent given $\mathbf{S}_{AB}$ and $A, B \notin \mathbf{S}_{AB}$. If there is no such a set, place an undirected arc between A and B.
2. For each pair of non-adjacent variables A and B with a common neighbour C, check whether $C \in \mathbf{S}_{AB}$. If it is not, set the direction of the arcs $A - C$ and $C - B$ to $A \rightarrow C$ and $C \leftarrow B$.
3. Set the direction of arcs which are still undirected by applying recursively the following two rules:
   (a) if A is adjacent to B and there is a strictly directed path from A to B then set the direction of $A - B$ to $A \rightarrow B$.
   (b) if A and B are not adjacent but $A \rightarrow C$ and $C - B$, then change the latter to $C \rightarrow B$.
4. Return the resulting (partially) directed acyclic graph.

by an arc) and removing arcs whenever the removal can be justified on grounds of conditional independence.

The second step deals with the identification of the v-structures among all the pairs of non-adjacent nodes A and B with a common neighbour C. V-structures are the only fundamental connection in which the two non-adjacent nodes are not independent conditional on the third one; so if there is a subset of nodes that contains C and d-separates A and B the three nodes are part of a v-structure centered on C. This condition can be verified by performing a conditional independence test for A and B against every possible subset of their common neighbours that includes C.

At the end of the second step both the skeleton and the v-structures of the network are known, so the equivalence class the Bayesian network belongs to is uniquely identified. The third and last step of the IC algorithm identifies compelled arcs and orients them recursively to obtain the partially directed acyclic graph describing the equivalence class identified by the previous steps.

A major problem of the IC algorithm is that the first two steps cannot be applied in the form described in Algorithm 2.1 to any real world problem due to the exponential number of conditional independence relationship that would need to be examined. This problem has led to the development of many algorithms, such as:

- *PC*: the first practical application of the IC algorithm (Spirtes et al., 2000).
- *Grow-Shrink* (GS): based on the *Grow-Shrink Markov blanket* algorithm (Margaritis, 2003), a simple forward selection Markov blanket detection algorithm.
- *Incremental Association* (IAMB): based on the *Incremental Association Markov blanket* algorithm (Tsamardinos et al., 2003), a two-phase selection scheme (a forward selection followed by an attempt to remove false positives).
- *Fast Incremental Association* (Fast-IAMB): a variant of IAMB which uses speculative stepwise forward selection to reduce the number of conditional independence tests (Yaramakala and Margaritis, 2005).
- *Interleaved Incremental Association* (Inter-IAMB): another variant of IAMB which uses forward stepwise selection (Tsamardinos et al., 2003) to avoid false positives in the Markov blanket detection phase.

All these algorithms with the exception of PC first learn the Markov blanket of each node. This preliminary step greatly simplifies the identification of neighbours (which are a subset of the Markov blanket of the node) and of the v-structures (both the

Algorithm 2.2 Hill Climbing Algorithm

1. Choose a network structure G over $\mathbf{V}$, usually (but not necessarily) empty.
2. Compute the score of G, denoted as $Score_G = \text{Score}(G)$.
3. Set $maxscore = Score_G$.
4. Repeat the following steps as long as $maxscore$ increases:
   (a) For every possible arc addition, deletion or reversal not resulting in a cyclic network:
      i. compute the score of the modified network G^*, $Score_{G^*} = \text{Score}(G^*)$.
      ii. if $Score_{G^*} > Score_G$, set $G = G^*$ and $Score_G = Score_{G^*}$.
   (b) update $maxscore$ with the new value of $Score_G$.
5. Return G.

Markov blankets of A and B must include C). This in turn results in a significant reduction of the overall computational complexity of the learning algorithm, which is usually measured with the number of conditional independence tests. Further improvements are possible by leveraging the symmetry of Markov blankets introduced in Section 2.1; an example will be shown in Chapter 3.

2.2.2 Score-based Algorithms

Score-based learning algorithms represent the application of general optimization techniques to the problem of learning the structure of a Bayesian network. Each candidate network is assigned a *network score* reflecting its goodness of fit, which is then taken as an objective function to maximize.

The main limitation of this approach lies in the dimension of the space of directed acyclic graphs, which grows more than exponentially in the number of nodes (Harary and Palmer, 1973). This means that an exhaustive search is not feasible in all but the most trivial cases, and has led to an extensive use of heuristic optimization algorithms. Some examples are:

- *greedy search* algorithms such as *hill-climbing* with *random restarts* or *tabu search* (Bouckaert, 1995). These algorithms explore the search space starting from a network structure (usually without any arc) and adding, deleting or reversing one arc at a time until the score can no longer be improved (see Algorithm 2.2);

- *genetic* algorithms, which simulate natural evolution through the iterative selection of the "fittest" models and the hybridization of their characteristics (Larrañaga et al., 1997). In this case the search space is explored through the *crossover* (which combines the structure of two networks) and *mutation* (which introduces random alterations) stochastic operators;

- the *simulated annealing* algorithm (Bouckaert, 1995), which performs a stochastic local search by (always) accepting changes that increase the network score and, at the same time, allowing changes that decrease it (with a probability inversely proportional to the score decrease).

A broad overview of these approaches is present in Russell and Norvig (2009).

2.2.3 Hybrid Algorithms

Hybrid learning algorithms combine constraint and score-based algorithms to offset their weaknesses and produce reliable network structures in a wide variety of situations. The two best-known members of this family are the *Sparse Candidate* algorithm (SC) by Friedman et al. (1999b) and the *Max-Min Hill-Climbing* algorithm (MMHC) by Tsamardinos et al. (2006). The former is illustrated in Algorithm 2.3.

Both these algorithms are based on two steps, called *restrict* and *maximize*. In the first one the candidate set for the parents of each node X_i is reduced from the whole node set $\mathbf{V}$ to a smaller set $\mathbf{C}_i \subset \mathbf{V}$ of nodes whose behaviour has been shown to be related in some way to that of X_i. This in turn results in a smaller and (usually) more

Algorithm 2.3 Sparse Candidate Algorithm

1. Choose a network structure G over $\mathbf{V}$, usually (but not necessarily) empty.
2. Repeat the following steps until convergence:
   (a) **Restrict:** select a set $\mathbf{C}_i$ of candidate parents for each node $X_i \in \mathbf{V}$, which must include the parents of X_i in G.
   (b) **Maximize:** find the network structure G^* that maximizes $\text{Score}(G^*)$ among the networks in which the parents of each node X_i are included in the corresponding set $\mathbf{C}_i$.
   (c) set $G = G^*$.
3. return G.

regular search space for the second step, which seeks the network that maximizes a given score function among the ones that satisfy the constraints imposed by the $\mathbf{C}_i$ sets.

In the Sparse Candidate algorithm these two steps are applied iteratively until there is no change in the network or no network improves the network score; the choice of the heuristics used to perform them is left to the implementation. In the Max-Min Hill-Climbing algorithm, on the other hand, *restrict* and *maximize* are performed only once; the *Max-Min Parents and Children* (MMPC) heuristic is used to learn the candidate sets $\mathbf{C}_i$ and a hill-climbing greedy search to find the optimal network.

2.2.4 Parameter Learning

Once the structure of the network has been learned from the data, the task of estimating and updating the parameters of the global distribution is greatly simplified by the application of the Markov property.

Local distributions in practice involve only a small number of variables; furthermore their dimension usually does not scale with the size of $\mathbf{X}$ (and is often assumed to be bounded by a constant when computing the computational complexity of algorithms), thus avoiding the so called *curse of dimensionality*. This means that each local distribution has a comparatively small number of parameters to estimate from the sample, and that estimates are more accurate due to the better ratio between the size of parameter space and the sample size.

The number of parameters needed to uniquely identify the global distribution, which is the sum of the number of parameters of the local ones, is also reduced because the conditional independence relationships encoded in the network structure fix large parts of the parameter space. For example, in graphical Gaussian models partial correlation coefficients involving (conditionally) independent variables are equal to zero by definition, and joint frequencies factorize into marginal ones in multinomial distributions.

However, parameter estimation is still problematic in many situations. For example it is increasingly common to have sample sizes much smaller than the number of variables included in the model; this is typical of microarray data, which have a few ten or hundred observations and thousands of genes. Such a situation, which is called "small n, large p", leads to estimates with high variability unless particular care is taken both in structure and parameter learning (Castelo and Roverato, 2006; Schäfer and Strimmer, 2005; Hastie et al., 2009).

Dense networks, which have a high number of arcs compared their nodes, represent another significant challenge. Exact inference quickly becomes unfeasible as the number of nodes increases, and even approximate procedures based on Monte Carlo simulations and bootstrap resampling require large computational resources (Koller and Friedman, 2009; Korb and Nicholson, 2004). Numerical problems stemming from floating point approximations (Goldberg, 1991) and approximate numerical algorithms (such as the ones used in matrix inversion and eigenvalue computation) should also be taken into account.

2.3 Pearl's Causality

In Section 2.1 Bayesian networks have been defined in terms of conditional independence statements and probabilistic properties, without any implication that arcs should represent cause-and-effect relationships. The existence of equivalence classes of networks undistinguishable from a probabilistic point of view provides a simple example of this fact.

However, from an intuitive point of view it can be argued that a "good" Bayesian network structure should represent the causal structure of the data it is describing. Such networks are usually fairly sparse (i.e. they have a number of arcs comparable to the number of nodes) and their interpretation is at the same time clear and meaningful, as explained by Pearl (2009) in his book on causality:

> It seems that if conditional independence judgements are byproducts of stored causal relationships, then tapping and representing those relationships directly would be a more natural and more reliable way of expressing what we know or believe about the world. This is indeed the philosophy behind causal Bayesian networks.

Learning such causal models, especially from observational data, presents significant challenges. In particular three additional assumptions are needed:

- each variable $X_i \in \mathbf{X}$ is conditionally independent of its non-effects, both direct and indirect, given its direct causes. This assumption is called the *causal Markov assumption*, and represents the causal interpretation of the Markov property introduced in Section 2.1;
- there must exist a network structure which is faithful to the dependence structure of $\mathbf{X}$;

- there must be no *latent variables* (unobserved factors influencing the variables in the network) acting as *confounding factors*. Such variables may induce spurious correlations between the observed variables, thus introducing bias in the causal network. Even though this is often listed as a separate assumption, it is really a corollary of the first two: the presence of unobserved variables breaks the faithfulness (because the network structure does not include them) and possibly the causal Markov property (if an arc is wrongly added between the observed variables due to the influence of the latent one).

These assumptions are difficult to verify in real world settings, as the set of the potential confounding factors is not usually known. The best we can do is to address this issue, along with selection bias, by implementing a carefully planned experimental design.

Furthermore, even when dealing with interventional data collected from a scientific experiment (where we can set the value of at least some variables and observe the resulting changes) there are usually multiple equivalent network structures that represent reasonable causal models. Many arcs may not have a definite direction, resulting in graph structures differing even in the topological ordering of the nodes. When the sample size is small there may also be several non-equivalent networks fitting the data equally well. So, in general, we are not able to identify a single, "best", causal network but rather a small set of likely causal networks that fit our knowledge of the data.

2.4 Bayesian and Markov Networks

Markov networks (also known as *Markov random fields*) are, together with Bayesian networks, the subjects of most past and current literature on graphical models.

These two classes of graphical models share many common traits. Markov networks are also defined as the minimal I-map of the dependence structure P of $\mathbf{X}$, the only difference being that in this case $U = (\mathbf{V}, E)$ is an *undirected graph*. All the arcs of U, which are usually called *edges* in this setting, are *undirected*; the relationship between the two nodes linked by an edge is symmetric, without the distinction between parents and children that characterizes Bayesian networks.

Graphical separation (which is called *undirected separation* or *u-separation* in Castillo et al. (1997)) is easily defined due to this symmetry.

Definition 2.4. *If* $\mathbf{A}$, $\mathbf{B}$ *and* $\mathbf{C}$ *are three disjoint subsets of nodes in an undirected graph* U, *then* $\mathbf{C}$ *is said to separate* $\mathbf{A}$ *from* $\mathbf{B}$, *denoted* $\mathbf{A} \perp\!\!\!\perp_G \mathbf{B} \,|\, \mathbf{C}$, *if every path between a node in* $\mathbf{A}$ *and a node in* $\mathbf{B}$ *contains at least one node in* $\mathbf{C}$.

On the other hand, the decomposition of the global distribution of $\mathbf{X}$ given by the *Markov property* is less straightforward because local distributions are not associated with single nodes, but with the *cliques* (maximal subsets of nodes in which each element is adjacent to all the others) $\mathbf{C}_1$, $\mathbf{C}_2$, ..., $\mathbf{C}_k$ present in the graph; so

$$\mathrm{P}(\mathbf{X}) = \prod_{i=1}^{k} \psi_i(\mathbf{C}_i) \qquad \text{for discrete data and} \tag{2.9}$$

$$f(\mathbf{X}) = \prod_{i=1}^{k} \psi_i(\mathbf{C}_i) \qquad \text{for continuous data.} \tag{2.10}$$

The functions $\psi_1, \psi_2, \ldots, \psi_k$ are called *Gibbs' potentials* (Pearl, 1988), *factor potentials* (Castillo et al., 1997) or simply *potentials*, and are non-negative functions representing the relative mass of probability of each clique. They are proper probability or density functions only when the graph is *decomposable* or *triangulated*, that is when it contains no induced cycles other than triangles. With any other type of graph inference becomes very hard, if possible at all, because $\psi_1, \psi_2, \ldots, \psi_k$ have no direct statistical interpretation. Decomposable graphs are also called *chordal* (Diestel, 2005) because any cycle of length at least four has a chord (a link between two nodes in a cycle that is not contained in the cycle itself). In this case the global distribution factorizes again according to the chain rule and can be written as

$$\mathrm{P}(\mathbf{X}) = \frac{\prod_{i=1}^{k} \mathrm{P}(\mathbf{C}_i)}{\prod_{i=1}^{k} \mathrm{P}(\mathbf{S}_i)} \qquad \text{for discrete data and} \tag{2.11}$$

$$f(\mathbf{X}) = \frac{\prod_{i=1}^{k} f(\mathbf{C}_i)}{\prod_{i=1}^{k} f(\mathbf{S}_i)} \qquad \text{for continuous data,} \tag{2.12}$$

where $\mathbf{S}_i$ are the nodes of $\mathbf{C}_i$ which are also part of any other clique up to $\mathbf{C}_{i-1}$ (Pearl, 1988).

The two characterizations of graphical separation and of the Markov properties presented above do not appear to be closely related, to the point that these two classes of graphical models seem to be very different in construction and interpretation. There are indeed dependency models that have an undirected perfect map but not a directed acyclic one, and vice versa (see Pearl (1988), pages 126 – 127 for a simple example of a dependency structure that cannot be represented as a Bayesian network). However, it can be shown (Pearl, 1988; Castillo et al., 1997) that every dependency structure that can be expressed by a decomposable graph can be modeled both by a Markov network and a Bayesian network. This is clearly the case for the small networks shown in Figure

2.3, as the undirected graph obtained from the Bayesian network by *moralization* is decomposable. It can also be shown that every dependency model expressible by an undirected graph is also expressible by a directed acyclic graph, with the addition of some auxiliary nodes. These two results indicate that there is a significant overlap between Markov and Bayesian networks, and that in many cases both can be used to the same effect.

Chapter 3

The bnlearn R Package

In this chapter we will provide an overview of the features implemented in the **bnlearn** R package with the help of `learning.test`, a simple data set shipped with **bnlearn**, and `hailfinder`, a reference network from the *Bayesian network repository* (Elidan, 2001). In particular, we will focus on the functions for learning the structure and the parameters of Bayesian networks, thus covering the most common learning strategies present in modern literature. Furthermore, the parallel implementations of some of the above methods will be introduced.

3.1 An Overview of bnlearn

bnlearn (Scutari, 2010b) is an R package (R Development Core Team, 2010) implementing several algorithms for learning the structure of Bayesian networks, as well as support for basic parametric and bootstrap inference, conditional probability queries and cross-validation (Koller and Friedman, 2009). It is licensed under the GNU Public License (GPL), version 2 or later. **bnlearn** focuses on three areas:

- the creation and manipulation of network structures. Both are achieved in a user-friendly and consistent way through the use of an ad-hoc class called `bn` and its methods.

- learning the structure of Bayesian networks and estimating its parameters. Formerly some of these learning algorithms lacked a free-software implementation, which limited their usefulness in scientific research. Furthermore, learning algorithms can be chosen separately from the statistical criterion they are based on (which is usually not possible in the reference implementation provided by the

algorithm's authors), so that the best one for the data at hand can be used.

- performing inferential procedures. Some common inference procedures, both exact and approximate, are implemented; among them are prediction, bootstrap, cross-validation and conditional probability queries.

Both discrete and continuous data are supported. Several functions can use the functionality provided by the **snow** package (Tierney et al., 2008) to improve their performance via distributed and parallel computing. All the network scores and conditional independence tests used in the learning algorithms are also available for independent use; together with the functions for graph manipulation they allow the user to design custom learning strategies.

Several reference data sets from literature, both small (Lauritzen and Spiegelhalter, 1988; Mardia et al., 1979) and large (Beinlich et al., 1989; Abramson et al., 1996; Elidan, 2001), are included in the package to simplify performance measurements and improve examples' reproducibility. R scripts to generate more observations are included for synthetic data sets, and it is also possible to generate more observations from the Bayesian networks learned from real-world data sets.

Advanced plotting options are provided by the **Rgraphviz** package (Gentry et al., 2010) for network structures and by the **lattice** package for diagnostic plots (Sarkar, 2008).

3.2 Manipulating Network Structures

bnlearn and its dependencies (the **utils** package, which is bundled with R) are all available from CRAN, as are the suggested packages **snow** and **graph** (Gentleman et al., 2010). The last suggested package, **Rgraphviz** (Gentry et al., 2010), can be installed from BioConductor and is loaded along with **bnlearn** if present.

```
> library(bnlearn)
Loading required package: Rgraphviz
Loading required package: graph
Loading required package: grid
Package Rgraphviz loaded successfully.
```

Once **bnlearn** is loaded, we can access its functions, documentation and data sets, such as `learning.test`.

```
> data(learning.test)
```

`learning.test` contains six discrete variables, stored as factors, each with 2 (for `F`) or 3 (for `A`, `B`, `C`, `D` and `E`) levels. Its dependence structure is particularly simple, as can be seen both from Figure 3.1 and from the decomposition of the global distribution,

$$\mathrm{P}\left(\mathbf{X}\right) = \mathrm{P}(A)\,\mathrm{P}(C)\,\mathrm{P}(F)\,\mathrm{P}(B \mid A)\,\mathrm{P}(D \mid A, C)\,\mathrm{P}(E \mid B, F). \tag{3.1}$$

The corresponding network structure can be created in three different ways. First, we can use a *model string*, which is expresses the dependence structure in terms of conditional dependence (note the similarity with the right hand of Equation 3.1):

```
> res = empty.graph(nodes = names(learning.test))
> modelstring(res) = "[A][C][F][B|A][D|A:C][E|B:F]"
```

or equivalently:

```
> res = model2network("[A][C][F][B|A][D|A:C][E|B:F]")
```

In the former case an empty network (i.e. a network with no arcs) is created with `empty.network` and then modified; in the latter the network is created with the right structure from the start.

We can also use either the arc set or the adjacency matrix of the graph to the same effect.

```
> res2 = empty.graph(nodes = names(learning.test))
> arcs = matrix(c("A", "B", "A",  "D", "C", "D", "B", "E",
+   "F",  "E"), ncol = 2, byrow = TRUE,
+   dimnames = list(NULL, c("from", "to")))
> arcs(res2) = arcs
> res3 = empty.graph(nodes = names(learning.test))
> amat = matrix(c(0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
+   0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
+   0, 0, 0), byrow = TRUE, ncol = 6)
> amat(res3) = amat
```

All these approaches result in the same network structure. Furthermore, unless the `check.cycles` argument is set to `FALSE` the network is guaranteed to be acyclic.

```
> all.equal(res, res2)
[1] TRUE
> all.equal(res, res3)
[1] TRUE
```

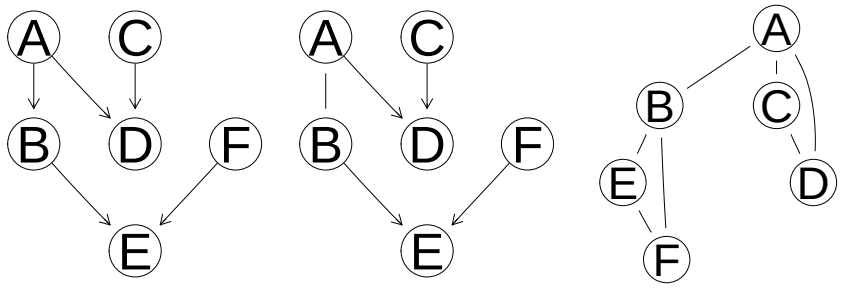


Figure 3.1: The network structure of `learning.test`, the PDAG representation of its equivalence class and it moral graph.

The networks created above are stored in objects of class `bn`, which are specifically designed to simplify and speed up common operations on Bayesian networks. The structure of the network is stored using its arc set representation, which is easy to manipulate and memory efficient; undirected arcs are stored as their two possible orientations (so an arc $\mathtt{A} - \mathtt{B}$ would be stored as $\{\mathtt{A} \rightarrow \mathtt{B}, \mathtt{B} \rightarrow \mathtt{A}\}$).

Once a network structure is stored in a `bn` object it can be manipulated and investigated using the methods defined for this class. The most basic is the `print` function, which produces the following output for the network created above.

```
> res

  Random/Generated Bayesian network

  model:
   [A][C][F][B|A][D|A:C][E|B:F]
  nodes:                                6
  arcs:                                 5
    undirected arcs:                    0
    directed arcs:                      5
  average markov blanket size:          2.33
  average neighbourhood size:           1.67
  average branching factor:             0.83

  generation algorithm:                 Empty

```

Further information on the network structure can be extracted from any `bn` object with the following functions (function names are reported in parenthesis):

- whether the network is acyclic (`acyclic`) or completely directed (`directed`) and if there is a path between two nodes (`path`);
- whether the network structure is equal to another one (`all.equal`) and, if not, a detailed description of the differences (`compare`);
- the moral graph (`moral`), the skeleton (`skeleton`), the equivalence class (`cpdag`) and the v-structures (`vstructs`) of the network;
- the labels of the nodes (`nodes`), of the root nodes (`root.nodes`), of the leaf nodes (`leaf.nodes`) and the topological ordering of the network (`node.ordering`);
- the parents (`parents`), children (`children`), Markov blanket (`mb`), neighbourhood (`nbr`), in-degree (`in.degree`), out-degree (`out.degree`) and degree (`degree`) of each node;
- the directed arcs (`directed.arcs`), the undirected arcs (`undirected.arcs`), the whole arc set (`arcs`) and it size (`narcs`);
- the adjacency matrix (`amat`) and the model string (`modelstring`) of the network;
- the number of parameters (`nparams`) associated with the network for a particular data set and the number of tests or network scores computed in the learning of the network (`ntests`).

The `arcs`, `amat` and `modelstring` functions can also be used in combination with `empty.graph` to create a `bn` object with a specific structure from scratch, as shown above. It is also possible to modify, drop or reverse single arcs with the `set.arc`, `drop.arc` or `reverse.arc` respectively.

Combining these functions it is possible to investigate many of the properties of Bayesian networks illustrated in Chapter 2. For example, it is easy to verify that Markov blankets are indeed symmetric (as stated in Section 2.1).

```
> "C" %in% mb(res, "A")
[1] TRUE
> "A" %in% mb(res, "C")
[1] TRUE
```

The symmetry of neighbourhoods can be verified in the same way.

```
> "D" %in% nbr(res, "A")
[1] TRUE
> "A" %in% nbr(res, "D")
[1] TRUE
```

We can also check that the Markov blanket of a given node (`A` in this example) is indeed composed by its children (`chld`), its parents (`par`) and its children's other parents (`o.par`).

```
> chld = children(res, "A")
> par = parents(res, "A")
> o.par = unlist(sapply(chld, parents, x = res))
> unique(c(chld, par, o.par[o.par != "A"]))
[1] "B" "D" "C"
> mb(res, "A")
[1] "B" "C" "D"
```

Finally, we can check that networks belonging to the same equivalence class have equivalent probability decompositions. For example, we can see that altering the direction of an arc ($\mathtt{A} - \mathtt{B}$) that is not compelled and is not part of a v-structure does not alter the probability of the data (computed as $\log \mathrm{P}(\mathbf{X})$).

```
> res = set.arc(res, "A", "B")
> score(res, learning.test, type = "loglik")
[1] -23832.13
> res = set.arc(res, "B", "A")
> score(res, learning.test, type = "loglik")
[1] -23832.13
```

3.3 Learning a Bayesian Network

3.3.1 Fundamental Assumptions of Structure Learning Algorithms

All structure learning algorithms operate under a set of common assumptions:

- there must be a one-to-one correspondence between the nodes of the graph and the random variables included in the model; this means in particular that there must not be multiple nodes which are functions of a single variable.

- there must be no unobserved (also called *latent* or *hidden*) variables that are parents of an observed node; otherwise only part of the dependency structure can be observed, and the model is likely to include spurious arcs. Specific algorithms have been developed for this particular case, typically based on Bayesian posterior distributions or the EM algorithm (Dempster et al., 1977); see for example Friedman (1997), Elidan and Friedman (2005) and Binder et al. (1997).

- all the relationships between the variables in the network must be conditional independencies, because they are by definition the only ones that can be expressed by graphical models.

- every combination of the possible values of the variables in $\mathbf{X}$ must represent a valid, observable (even if really unlikely) event. This assumption implies a strictly positive global distribution, which is needed to have uniquely determined Markov blankets and, therefore, a uniquely identifiable model. Constraint-based algorithms work even when this is not true, because the existence of a perfect map is also a sufficient condition for the uniqueness of the Markov blankets (Pearl, 1988).

- observations must be stochastically independent. If some form of temporal or spatial dependence is present it must be specifically accounted for in the definition of the network, as in *dynamic Bayesian networks* (Koller and Friedman, 2009).

3.3.2 Choosing the Global and Local Distributions

In principle there are many possible choices for both the global and the local distribution functions, depending on the nature of the data and the aims of the analysis. However, literature have focused mostly on two cases:

- *multinomial data* (the so-called *discrete case*): both the global and the local distributions are multinomial, and the latter are represented as *conditional probability tables* (CPTs). This is by far the most common assumption, and the corresponding Bayesian networks are usually referred to as *discrete Bayesian networks* (or simply as *Bayesian networks*).

- *multivariate normal data* (the so-called *continuous case*): the global distribution is multivariate normal, and the local distributions are normal random variables linked by linear constraints. Local distributions are in fact linear models in which the parents play the role of explanatory variables. These Bayesian networks are

label	conditional independence test
`mi`	Mutual Information
`mi-sh`	Mutual Information (shrinkage estimator)
`mc-mi`	Mutual Information (permutation test)
`x2`	Pearson's X^2
`mc-x2`	Pearson's X^2 (permutation test)

Table 3.1: Conditional independence tests for discrete data implemented in **bnlearn**.

called *Gaussian Bayesian networks* in Geiger and Heckerman (1994), Neapolitan (2003) and most recent literature on the subject.

Other general distributional assumptions require ad-hoc learning algorithms or present various limitations due to the difficulty of specifying the distribution functions in closed form. For example many models for mixed data, such as the one presented in Bøttcher and Dethlefsen (2003), do not allow a node associated with a continuous variable to be the parent of a node associated with a discrete variable.

The choice of a particular set of global and local distributions determines which conditional independence tests and which network scores can be used to learn the structure of the Bayesian network.

Conditional independence tests and network scores for discrete data are functions of the conditional probability tables implied by the graphical structure of the network through the observed frequencies $\{n_{ijk}, i = 1, \ldots, R, j = 1, \ldots, C, k = 1, \ldots, L\}$ for the random variables X and Y and all the configurations of the conditioning variables $\mathbf{Z}$. Two common conditional independence tests are:

- *mutual information*: an information-theoretic distance measure (Cover and Thomas, 2006) defined as

$$\mathrm{MI}(X, Y \mid \mathbf{Z}) = \sum_{i=1}^{R} \sum_{j=1}^{C} \sum_{k=1}^{L} \frac{n_{ijk}}{n} \log \frac{n_{ijk} n_{++k}}{n_{i+k} n_{+jk}}. \tag{3.2}$$

  It is proportional to the log-likelihood ratio test G^2 (they differ by a $2n$ factor, where n is the sample size) and it is related to the deviance of the tested models.

- *Pearson's* X^2: the classic Pearson's X^2 test (Bishop et al., 2007) for contingency

label	network score
`k2`	Cooper & Herskovits' K2
`bde`	Bayesian Dirichlet score equivalent (BDeu)
`aic`	Akaike Information Criterion
`bic`	Bayesian Information Criterion
`loglik`	Log-Likelihood

Table 3.2: Network scores for discrete data implemented in **bnlearn**.

tables,

$$\mathrm{X}^2(X, Y \mid \mathbf{Z}) = \sum_{i=1}^{R}\sum_{j=1}^{C}\sum_{k=1}^{L}\frac{(n_{ijk} - m_{ijk})^2}{m_{ijk}}, \quad \text{where} \quad m_{ijk} = \frac{n_{i+k}n_{+jk}}{n_{++k}}. \tag{3.3}$$

In both cases the null hypothesis of independence can be tested using either the asymptotic χ^2 distribution or the Monte Carlo permutation approach described in Edwards (2000). Other choices include Fisher's exact test (Bishop et al., 2007) and the shrinkage estimator for the mutual information defined by Hausser and Strimmer (2009).

Network scores commonly found in literature are:

- the *Bayesian Information Criterion* (BIC), a penalized likelihood score defined as

$$\mathrm{BIC}\left(\mathbf{X} \mid G\right) = \sum_{i=1}^{n}\log \mathrm{P}_{X_i}\left(X_i \mid \Pi_{X_i}\right) - \frac{d}{2}\log n \tag{3.4}$$

  where d is the number of parameters of the network. It is numerically equivalent to the information-theoretic *Minimum Description Length* measure by Rissanen (2007), even though it has a completely different derivation. BIC converges asymptotically to the posterior density of the network (Schwarz, 1978).

- the *Bayesian Dirichlet equivalent* (BDe) score, the posterior density associated with a uniform prior over both the space of the network structures and of the parameters (Heckerman et al., 1995).

Both these score functions are said to be *score equivalent*, because they assign the same score to networks belonging to the same equivalence class. They are also *decomposable* into the components associated with each node, which is a significant computational advantage when learning the structure of the network (only the relevant parts of the score need to be recomputed at each iteration of the learning algorithm).

label	conditional independence test
`mi-g`	Mutual Information
`mi-g-sh`	Mutual Information (shrinkage estimator)
`mc-mi-g`	Mutual Information (permutation test)
`cor`	Pearson's Correlation
`mc-cor`	Pearson's Correlation (permutation test)
`zf`	Fisher's Z Test
`mc-zf`	Fisher's Z Test (permutation test)

Table 3.3: Conditional independence tests for multivariate Gaussian data implemented in **bnlearn**.

In the continuous case conditional independence tests and network scores are functions of the partial correlation coefficients $\rho_{XY\,|\,\mathbf{Z}}$ of X and Y given $\mathbf{Z}$. Two common conditional independence tests are:

- the exact Student's t test (Hotelling, 1953) for Pearson's correlation coefficient, defined as
$$\mathrm{T}(X, Y \,|\, \mathbf{Z}) = \rho_{XY\,|\,\mathbf{Z}} \sqrt{\frac{n-2}{1-\rho^2_{XY\,|\,\mathbf{Z}}}}. \tag{3.5}$$

- *Fisher's Z* test (Fisher, 1921), a transformation of the linear correlation coefficient with an asymptotic normal distribution defined as
$$\mathrm{Z}(X, Y \,|\, \mathbf{Z}) = \frac{\sqrt{n - |\mathbf{Z}| - 3}}{2} \log \frac{1 + \rho_{XY\,|\,\mathbf{Z}}}{1 - \rho_{XY\,|\,\mathbf{Z}}}. \tag{3.6}$$

Both tests can also be performed using a Monte Carlo permutation approach such as the ones described in Legendre (2000). Other possible choices are the mutual information test defined in Kullback (1968), which is proportional to the corresponding log-likelihood ratio test, or the shrinkage estimators developed by Schäfer and Strimmer (2005).

Commonly used network scores are again the *Bayesian Information Criterion*, this time defined as
$$\mathrm{BIC}\,(\mathbf{X} \,|\, G) = \sum_{i=1}^{n} \log f_{X_i}\,(X_i \,|\, \Pi_{X_i}) - \frac{d}{2} \log n \tag{3.7}$$
and the *Bayesian Gaussian equivalent* (BGe) score, the Wishart posterior density of the network (Geiger and Heckerman, 1994).

label	network score
`bge`	Bayesian Gaussian score equivalent (BGe)
`loglik-g`	Log-Likelihood
`aic-g`	Akaike Information Criterion (Gaussian)
`bic-g`	Bayesian Information Criterion (Gaussian)

Table 3.4: Network scores for multivariate Gaussian data implemented in **bnlearn**.

3.3.3 Including Prior Information on the Data

Prior information on the data, such as the ones elicited from experts in the relevant fields, can be integrated in all structure learning algorithms by means of the `blacklist` and `whitelist` arguments. Both of these arguments accept a set of arcs which is guaranteed to be either present (for the former) or missing (for the latter) from the Bayesian network; any arc whitelisted and blacklisted at the same time is assumed to be whitelisted, and is thus removed from the blacklist.

This combination represents a flexible way to describe an arbitrary set of assumptions on the data, and is also able to deal with partially directed graphs:

- any arc whitelisted in both directions (i.e. both `A` $\rightarrow$ `B` and `B` $\rightarrow$ `A` are whitelisted) is present in the graph, but the choice of its direction is left to the learning algorithm. Therefore one of `A` $\rightarrow$ `B`, `B` $\rightarrow$ `A` and `A` $-$ `B` is guaranteed to be in the Bayesian network.

- any arc blacklisted in both directions, as well as the corresponding undirected arc, is never present in the graph. Therefore if both `A` $\rightarrow$ `B` and `B` $\rightarrow$ `A` are blacklisted, also `A` $-$ `B` is considered blacklisted.

- any arc whitelisted in one of its possible directions (i.e. `A` $\rightarrow$ `B` is whitelisted, but `B` $\rightarrow$ `A` is not) is guaranteed to be present in the graph in the specified direction. This effectively amounts to blacklisting both the corresponding undirected arc (`A` $-$ `B`) and its reverse (`B` $\rightarrow$ `A`).

- any arc blacklisted in one of its possible directions (i.e. `A` $\rightarrow$ `B` is blacklisted, but `B` $\rightarrow$ `A` is not) is never present in the graph. The same holds for `A` $-$ `B`, but not for `B` $\rightarrow$ `A`.

label	structure learning algorithm
`gs`	Grow-Shrink (GS)
`iamb`	Incremental Association (IAMB)
`fast.iamb`	Fast Incremental Association (Fast-IAMB)
`inter.iamb`	Interleaved Incremental Association (Inter-IAMB)
`hc`	Hill-Climbing (HC)
`tabu`	Tabu Search
`mmhc`	Max-Min Hill-Climbing (MMHC)
`rsmax2`	General 2-Phase Restricted Maximization (RSMAX2)

Table 3.5: Bayesian network structure learning algorithms implemented in **bnlearn** and the respective function names (also used as labels in the functions which take a structure learning algorithm as an argument).

3.3.4 Learning the Structure of the Network

Structure learning algorithms are defined as general learning strategies, and as such they do not require any assumption on the data. For example, the algorithms outlined in Algorithms 2.1, 2.2 and 2.3 only refer to a generic conditional independence test and a generic network score in the selection of the network. They are not even required to be in closed form, even though this is preferable for reasons of computational efficiency. Therefore, the choice of a structure learning algorithm is not influenced by the data it will be applied to or by the choice of a particular combination of global and local distributions.

bnlearn implements several structural learning algorithms, spanning all the three classes covered in Section 2.2; a list is provided in Table 3.5. All of them return an object of class `bn`, which can be used to learn the parameters of the network and to perform inference, and have several arguments to customize their behaviour. Commonly used ones include `test` (the conditional independence test used in constraint-based and hybrid algorithms), `score` (the network score used in score-based and hybrid algorithms) and `blacklist`/`whitelist`. For a detailed explanation of all the arguments of these functions we refer the reader to Scutari (2010a) and Scutari (2010b).

For example, the Grow-Shrink algorithm learns the following network from the `learning.test` data.

```
1 > bn.gs = gs(learning.test)
2 > bn.gs
3
4   Bayesian network learned via Constraint-based methods
```

```
5
6   model:
7     [partially directed graph]
8   nodes:                                   6
9   arcs:                                    5
10    undirected arcs:                       1
11    directed arcs:                         4
12  average markov blanket size:             2.33
13  average neighbourhood size:              1.67
14  average branching factor:                0.67
15
16  learning algorithm:                      Grow-Shrink
17  conditional independence test:
18                                 Mutual Information (discrete)
19  alpha threshold:                         0.05
20  tests used in the learning procedure:    43
21  optimized:                               TRUE
22
```

The algorithm is not able to determine the direction of all the arcs; not all of them are compelled or part of a v-structure. However, we can see that `bn.gs` is identical to the partially directed acyclic graph representing the equivalence class of true structure of the network (see Figure 3.1).

```
1 > all.equal(cpdag(res), bn.gs)
2 [1] TRUE
```

Constraint-based algorithms and score-based algorithms making use of score-equivalent network scores are not able to distinguish between equivalent networks, because they have the same probability decomposition and encode the same conditional independence statements. Therefore this is the best possible outcome we could have hoped for from a structure learning algorithm: it correctly identifies the equivalence class the true network belongs to.

Changing the conditional independence test (from the mutual information to Pearson's X^2) or the constraint-based learning algorithm (from Grow-Shrink to the Incremental Association) does not alter the result.

```
1 > all.equal(cpdag(res), gs(learning.test, test = "x2"))
2 [1] TRUE
```

```
> all.equal(cpdag(res), iamb(learning.test))
[1] TRUE
> all.equal(cpdag(res), iamb(learning.test, test = "x2"))
[1] TRUE
```

The true network structure of `learning.test` can still be completely learned if we have some prior information on the direction $\texttt{A} - \texttt{B}$, which is the only undirected arc in `bn.gs`.

```
> undirected.arcs(bn.gs)
     from to
[1,] "A"  "B"
[2,] "B"  "A"
```

In that case we can use the `whitelist` parameter to include this knowledge in the structure learning process. If we whitelist $\texttt{A} \rightarrow \texttt{B}$, the alternative orientation of that arc, $\texttt{B} \rightarrow \texttt{A}$, is automatically blacklisted because they can't both be present in the graph at the same time.

```
> whitelist = matrix(c("A", "B"), ncol = 2)
> bn.gs2 = gs(learning.test, whitelist = whitelist)
> all.equal(res, bn.gs2)
[1] TRUE
> whitelist(bn.gs2)
     from to
[1,] "A"  "B"
> blacklist(bn.gs2)
     from to
[1,] "B"  "A"
```

On the other hand, score-based algorithms always return completely directed networks, even when using score-equivalent network scores. Consider for example the network learned by the hill-climbing algorithm.

```
> bn.hc = hc(learning.test)
> bn.hc

  Bayesian network learned via Score-based methods

  model:
   [A][C][F][B|A][D|A:C][E|B:F]
```

```
  nodes:                                  6
  arcs:                                   5
    undirected arcs:                      0
    directed arcs:                        5
  average markov blanket size:            2.33
  average neighbourhood size:             1.67
  average branching factor:               0.83

  learning algorithm:                     Hill-Climbing
  score:
                                Bayesian Information Criterion
  penalization coefficient:               4.258597
  tests used in the learning procedure:   40
  optimized:                              TRUE

```

The network described by `bn.hc` is identical to the true structure of the network. However, if we start the hill-climbing search from a different network, such as the one containing only the `B` → `A` arc, we learn another network from the same equivalence class (the one containing `B` → `A` instead of `A` → `B`).

```
> start = empty.graph(names(learning.test))
> start = set.arc(start, "B", "A")
> bn.hc2 = hc(learning.test, start = start)
> modelstring(bn.hc2)
[1] "[B][C][F][A|B][E|B:F][D|A:C]"
> all.equal(cpdag(bn.hc), cpdag(bn.hc2))
[1] TRUE
```

The reason of this behaviour is in the definition of the score-based algorithms themselves: they operate in the space of the network structures, and therefore they can only learn elements of that space.

Hybrid algorithms work in a similar way, because they combine constraint-based and score-based algorithms in a single learning procedure. For example, the Max-Min Hill-Climbing algorithm is able to learn the true structure of the network correctly. All arcs are directed, because the *restrict* phase uses the hill-climbing algorithm.

```
> bn.mmhc = mmhc(learning.test)
> all.equal(res, bn.mmhc)
[1] TRUE
```

As noted both in Friedman et al. (1999b) and Tsamardinos et al. (2006), there are many possible choices for the *restrict* and *maximize* phases and their parameters. The `rsmax2` functions allows the user to specify any constraint-based algorithm for the *restrict* phase (with the `restrict` argument) and any score-based algorithm for the *maximize* phase (with the `maximize` argument).

```
> bn.rsmax2 = rsmax2(learning.test, restrict = "iamb",
+   maximize = "tabu", test = "x2", score = "bde")
> all.equal(res, bn.rsmax2)
[1] TRUE
```

The conditional independence test and the network scores used by the above algorithms can be specified with the `test` and `score` arguments, like in the original functions; `whitelist` and `blacklist` are also supported. Additional arguments can be passed to the learning algorithms specified by `restrict` and `maximize` either in the same way as the original functions (for the former) or using the `maximize.args` argument (for the latter).

3.3.5 Learning the Parameters

Once the network structure is known we can estimate the parameters of the local distributions. Three possible approaches are:

- *maximum likelihood estimation*: this is the most common choice in literature. In discrete networks the conditional probabilities for each node are estimated with the respective relative frequencies; in continuous networks the regression coefficients are estimated with the usual maximum likelihood estimates.

- *Bayesian estimation*: the local distribution of each node is combined with its conjugate prior and its parameters are estimated from the resulting posterior distribution. In discrete networks the conditional probabilities are estimated with the *pseudo-counts* associated with the hyperparameters the posterior Dirichlet distribution (Geiger and Heckerman, 1994). In continuous networks the normal-Gamma and Laplace distributions have been used as prior distributions; the latter is often preferred because it results in sparser networks (Koller and Friedman, 2009).

- *regularized estimation*: the parameters of each local distribution are estimated penalizing values that would result in a non-smooth behaviour. These include

the shrinkage estimators from Hausser and Strimmer (2009) for discrete networks and from Schäfer and Strimmer (2005) for continuous networks, and the graphical lasso developed by Friedman et al. (2007).

Each of these choices has its own advantages and disadvantages. For example, maximum likelihood estimators are the simplest to compute. However, they require a large sample size to produce reliable estimates. This is particularly evident in discrete networks, where we can get sparse conditional probability tables (i.e. with many zero entries) even at moderate sample sizes. Bayesian and regularized estimators tend to perform better in that regard, because they have a smoothing effect on the distribution of the data. However, the choice of the weight of the prior distribution (for Bayesian estimators) and of the intensity of the regularization (for regularized estimators) present their own sets of challenges.

bnlearn implements the maximum likelihood estimators for both discrete and continuous networks, and the Bayesian posterior estimator for discrete networks. These are the three most common choices in literature. These estimators can be computed using the `bn.fit` function.

```
> fitted = bn.fit(bn.hc, data = learning.test)
> fitted$E

Conditional probability table:

, , F = a

   B
E           a         b         c
  a 0.8052498 0.2058824 0.1193738
  b 0.0973751 0.1797386 0.1144814
  c 0.0973751 0.6143791 0.7661448

, , F = b

   B
E           a         b         c
  a 0.4005080 0.3167939 0.2375954
  b 0.4902625 0.3664122 0.5066794
  c 0.1092295 0.3167939 0.2557252

```

Bayesian estimators can be chosen by specifying `method = "bayes"`; by default the parameters are estimated by maximum likelihood.

```
> fitted = bn.fit(bn.hc, learning.test, method = "bayes")
```

The *equivalent* or *imaginary sample size*, which expresses the confidence in the choice of the (flat) prior, can be specified with the `iss` argument. It can be thought of as the size of an imaginary sample supporting the prior distribution; so large values put more weight on the prior at the expense of the information contained in the data.

Note that the network structure stored in the object of class `bn` passed to `bn.fit` must be a directed acyclic graph; any undirected arc must be either dropped (with the `drop.arc` function) or replaced with a directed one (with the `set.arc` function).

3.4 Performing Inference on a Bayesian Network

Inference on Bayesian networks includes a huge variety of techniques, based on different approaches (frequentist, Bayesian, information-theoretic, etc.) and with different objectives (estimation, prediction, model validation, etc.). For this reason it is not possible to cover the applications of all these techniques in the space of this section. Instead we will limit ourselves to three common approximate inference techniques implemented in **bnlearn**: *bootstrap* (Efron and Tibshirani, 1993), *cross-validation* (Hastie et al., 2009) and *conditional probability queries* (Koller and Friedman, 2009).

To illustrate these techniques we will use the `hailfinder` data set included in **bnlearn**, which is generated from the reference network of the same name. Hailfinder is a Bayesian network designed by Abramson et al. (1996) to forecast severe summer hail in northeastern Colorado. It contains 20000 observations and 56 variables describing a large set of environmental characteristics of the region.

3.4.1 Bootstrap

In the setting of Bayesian networks bootstrap is used to asses the properties of the parameters of the network, as in Koller and Friedman (2009), or of its structure, as in Friedman et al. (1999a). In both cases the aspects being investigated are usually the expected value or the variance of some aspect of the Bayesian network.

For example, in Friedman et al. (1999a) the statistics of interest are the probabilities associated with particular structural features of the network, such as the composition of a Markov Blanket or the topological ordering of the nodes. They are computed as

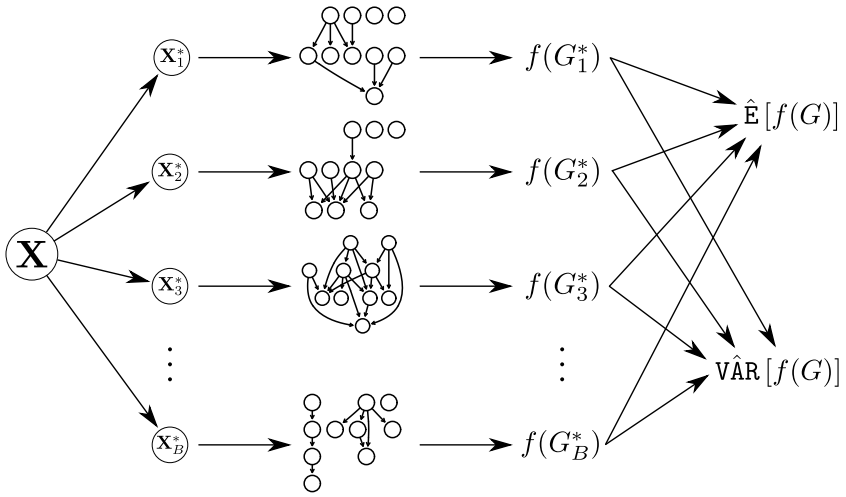


Figure 3.2: Nonparametric bootstrap estimate for a feature of a Bayesian network.

follows:

1. For $b = 1, 2, \ldots, R$

   (a) re-sample a new data set $\mathbf{X}^*_\mathbf{b}$ from the original data $\mathbf{X}$ using either parametric or nonparametric bootstrap.

   (b) learn a Bayesian network G_b from $\mathbf{X}^*_\mathbf{b}$.

2. Estimate the confidence in each feature f of interest as

$$\hat{\mathrm{P}}(f) = \frac{1}{R} \sum_{b=1}^{R} f(G_b). \tag{3.8}$$

In the case of structural features, f is either a simple indicator function (which is equal to 1 if the structural feature is present in the network G_b and 0 otherwise), or a counter (the absolute frequency of the structural feature in the graph).

We may be interested, for example, in the ability of the hill-climbing algorithm to learn a sparse network from the `hailfinder` data. Sparse networks have several good properties that make them useful in analyzing real world data: they are easier to interpret and most inference procedures are computationally tractable. The easiest way to assess whether a network is sparse is to count its arcs and then to compare that number to the number of nodes.

```
> sparse = bn.boot(hailfinder, algorithm = "hc",
+   R = 200, statistic = narcs)
> summary(unlist(sparse))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  63.00   64.00   65.00   64.69   65.00   67.00
> quantile(unlist(sparse), c(0.05, 0.95))
 5% 95%
 64  66
```

`hailfinder` has 56 nodes, so with 65 arcs it can be considered sparse. Furthermore, we can see that the bootstrap estimate has a very low variance because the boundaries of the 95% confidence interval are very close to the mean value. This is in part due to the large sample size of `hailfinder` (20000 observations) compared to the number of parameters (1768) of the network learned by the hill-climbing algorithm.

3.4.2 Cross-Validation

Cross-validation is probably the simplest and most widely used method in model validation to assess how the results of a statistical analysis will generalize to an independent data set. It has been applied to many classes of models, from regression to classification, to estimate the appropriate loss functions (such as the *classification error* or the *likelihood loss*).

Bayesian network learning algorithms are not explicitly targeted at classification problems; they seek to minimize the discrepancy between the estimated and the true dependence structure instead of the classification error. Furthermore, the very concept of a target variable (which is central in classification) is alien to Bayesian networks, which treat all the variables in the same way. Still there are some situations in which the classification error, estimated with the *prediction error*, may be of interest. For example, the Hailfinder network was designed to forecast severe summer hail in northeastern Colorado. In fact, some of the variables in the network (the ones with the names ending in `Fcst`) represent the weather conditions in different parts of the region, and the prediction of their values was the main goal of the original work by Abramson et al. (1996). If we take `CompPlFcst` (*Complete Plains Forecast*), we can see that even the Max-Min Hill-Climbing algorithm is not a very good classifier.

```
> bn.cv(hailfinder, 'mmhc', loss = "pred",
+   loss.args = list(target = "CompPlFcst"))

```

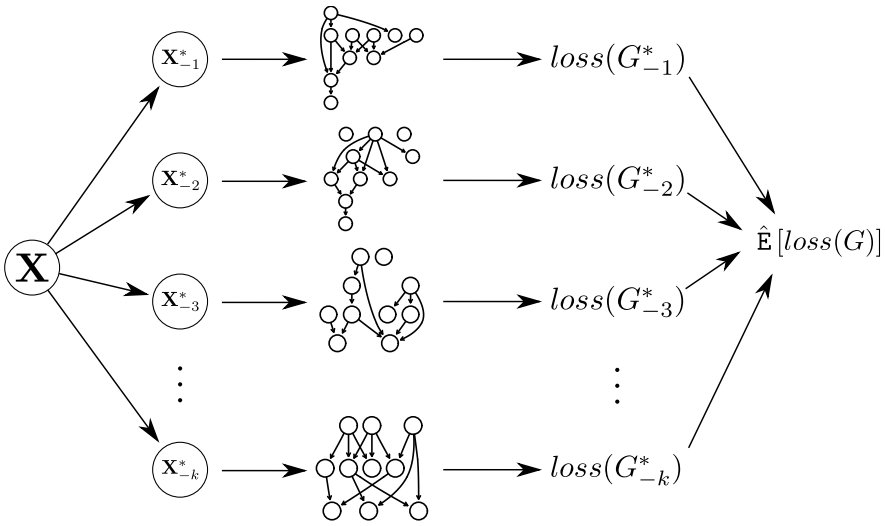


Figure 3.3: K-fold cross validation estimation of a loss function for a Bayesian network learning algorithm.

```
  k-fold cross-validation for Bayesian networks

  target learning algorithm:            Max-Min Hill-Climbing
  number of subsets:                    10
  loss function:                        Classification Error
  expected loss:                        0.5433
```

Hill-climbing and tabu search have comparable error rates (50.67% and 50.4%, respectively).

Cross-validation can also be used to evaluate a pre-determined network structure; in this case only the parameters are learned from the cross-validation samples. An example is the *naive Bayes classifier*, which is equivalent to a star-shaped Bayesian network with the training variable at the center and all the arcs pointing to the training variable (we refer the reader to Borgelt et al. (2009) for an introduction to this model). Naive Bayes classifiers are considered to be very good classifiers; indeed they apparently have a negligible classification error rate for this data set.

```
> naive = naive.bayes(training = "CompPlFcst",
+   data = hailfinder)
> bn.cv(hailfinder, naive, loss = "pred")

  k-fold cross-validation for Bayesian networks
```

```
6
7   target network structure:
8    [Naive Bayes Classifier]
9   number of subsets:                  10
10  loss function:                      Classification Error
11  training node:                      CompPlFcst
12  expected loss:                      0
```

3.4.3 Conditional Probability Queries

Conditional probability queries focus on the most common problem in the inference on the Bayesian networks: assessing the influence of the evidence e we have collected from some new data on the value y a variable of interest Y. The obvious measure for this quantity is the conditional probability

$$\mathrm{P}\left(Y = y \,|\, E = e\right) = \frac{\mathrm{P}\left(Y = y,\, E = e\right)}{\mathrm{P}\left(E = e\right)}, \tag{3.9}$$

which can be estimated in a number of ways (Korb and Nicholson, 2004; Koller and Friedman, 2009). The simplest one is called *logic* or *forward sampling*; it is a form of rejection sampling. A large number m of independent random observations are generated from the Bayesian network; then the proportion that satisfies the condition $E = e$ is used to estimate $\mathrm{P}\left(E = e\right)$ and the proportion that also satisfies $Y = y$ is used to estimate $\mathrm{P}\left(Y = y,\, E = e\right)$. The resulting estimator is

$$\hat{\mathrm{P}}\left(Y = y \,|\, E = e\right) = \frac{\sum_{i=1}^{m} \mathbb{I}\left(Y_i = y,\, E_i = e\right)}{\sum_{i=1}^{m} \mathbb{I}\left(E_i = e\right)}. \tag{3.10}$$

We can consider, for example, how the knowledge that the wind is blowing from east/north-east in the plains (`WindFieldPln == "E_NE"`) affects the the probability that the wind is blowing towards the west in the mountains (`WindFieldMt == "Westerly"`). To investigate it we generate 10^7 observations from the Bayesian network learned from `hailfinder` by the Max-Min Hill-Climbing algorithm.

```
> fitted = bn.fit(mmhc(hailfinder), hailfinder)
> cpquery(fitted, (WindFieldMt == "Westerly"),
+   (WindFieldPln == "E_NE"), n = 10^7)
[1] 0.4136172
> n = nrow(hailfinder)
```

```
> summary(hailfinder[, "WindFieldMt"]) / n
LVorOther  Westerly
  0.47615   0.52385
```

Obviously, the conditional probability is lower than the marginal one because the plains and the mountains are adjacent, so the wind can't change in direction by that much so suddenly.

Conditional distributions can be approximated by their empirical counterparts in a similar way. We can consider, for example, how the knowledge that there is a weather instability in the mountains (`InsInMt == "Strong"`) and that there is a marked cloud shading (`CldShadeConv == "Marked"`) influences the forecast for the plains.

```
> n = nrow(hailfinder)
> summary(hailfinder[, "CompPlFcst"]) / n
DecCapIncIns IncCapDecIns LittleChange
     0.22810      0.41205      0.35985
> cp = cpdist(fitted, nodes = "CompPlFcst",
+ (InsInMt == "Strong") & (CldShadeConv == "Marked"),
+   n = 10^7)
> n = nrow(cp)
> summary(cp[, CompPlFcst]) / n
DecCapIncIns IncCapDecIns LittleChange
   0.1888219    0.4812025    0.3299755
```

The three levels of `CompPlFcst` stand for *decreased instability* (`DecCapIncIns`), *increased instability* (`IncCapDecIns`) and *little change* (`LittleChange`). The conditional distribution shows an increased probability of the weather worsening (+6.9%) compared to the marginal one, which suggests that bad weather tends to spill from the mountains into the plains. This trend is confirmed by the decreased probability of `DecCapIncIns` (−3.9%) and `LittleChange` (−2.9%).

3.5 Parallel Structure Learning for Bayesian Networks

It is well known from literature that the problem of learning the structure of Bayesian networks is very hard to tackle. Its computational complexity, measured with the number of model comparisons (either through conditional independence tests or network scores), is exponential in the number of nodes in the worst case and polynomial in most real-world situations (Chickering, 1996). Furthermore, the computational complexity

of the conditional independence tests and the network scores themselves must be taken into account; in most cases it is linear in the sample size.

Therefore the performance boost provided by a *parallel implementation* is really welcome, especially when a large number of variables is involved in the analysis. **bnlearn** provides such an implementation for several structure learning algorithms using the functionality provided by the **snow** package. Learning algorithms are split into several parts which are then executed concurrently by several *slave processes*, started in the background and managed by **snow** as a single *cluster*. **snow** also manages the communications between the *master process* (i.e. the one controlled by the user) and the slaves, so both data and R commands are copied back and forth transparently.

3.5.1 Constraint-based Algorithms

Constraint-based algorithms display a *coarse-grained parallelism*, because they only need to synchronize their parts a couple of times. If we examine again Algorithm 2.1 we can see that:

1. the first step is *embarrassingly parallel*, as each d-separating set can be learned independently from the other ones. Another solution is to split this step in one part for each node, which will learn all the d-separating sets involving that particular node. The former approach can take advantage of a greater number of processors, while the latter has less overhead due to the smaller number of parts running in parallel;

2. the same holds for the second step. Once all the d-separating sets are known, it is embarrassingly parallel and can be split in the same way as the first step;

3. the third step is sequential, because each of its iterations requires the status of the previous one.

So the only two points in which the status of the various parts has to be collected is between the first and the second step and between the second and the third step.

Most modern constraint-based algorithms, which learn the Markov blankets of the nodes as an intermediate step, require an additional synchronization. For example, if we consider the Grow-Shrink algorithm as shown in Figure 3.4 we can see that:

1. each Markov blanket can be computed independently from the others;

2. each neighbourhood is a subset of the corresponding Markov blanket and therefore can be learned independently from the other ones. However, the consistency of

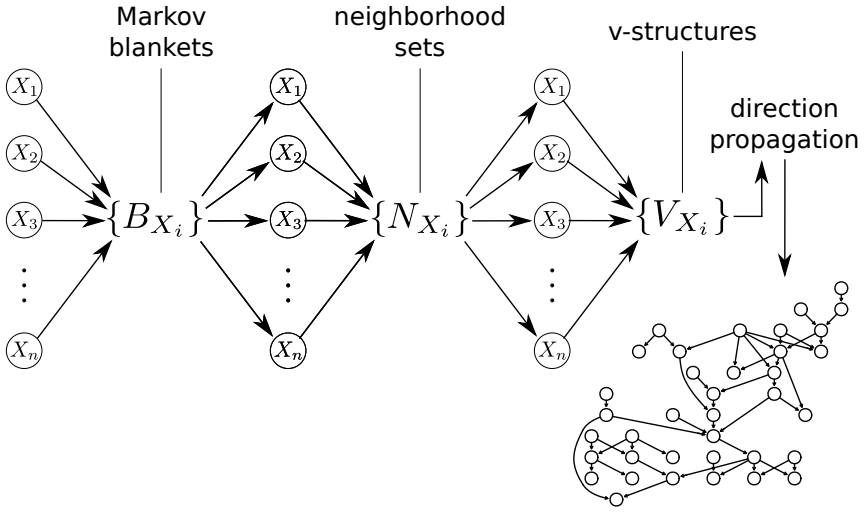


Figure 3.4: Parallel implementation of the Grow-Shrink algorithm present in **bnlearn**.

the Markov blankets must be checked before learning neighbourhood sets; due to errors in the conditional independence tests they may not be symmetric. A solution to this problem is to examine all pairs of nodes and remove them form each other's Markov blanket if they do not appear in both of them;

3. given the Markov blankets and the neighbourhood sets, the v-structures centered on a particular node (i.e. the one with the converging arcs) can again be computed in parallel. As in the previous step, the consistency of the neighbourhood sets must be checked and any departure from symmetry must be fixed before identifying the v-structures.

Furthermore, the final step of the Grow-Shrink algorithm (in which the directions of the compelled arcs are learned) also displays a *fine-grained parallelism* (i.e. the status of the slaves requires several synchronizations per second). The order in which arcs are considered in that step depends on the topology of the graph; undirected arcs whose orientations would result in the greatest number of cycles are considered first. That number can be computed in parallel for each arc, at the cost of introducing some overhead.

We will now examine the practical implications of parallelizing a constraint-based learning algorithm, starting with a simple cluster with two slave processes. For this task we will use again the `hailfinder` data, which is large enough to properly highlight the improvements resulting from parallel computing.

```
> library(snow)
> cl <- makeCluster(2, type = "MPI")
        2 slaves are spawned successfully. 0 failed.
> res = gs(hailfinder, cluster = cl)
> unlist(clusterEvalQ(cl, .test.counter))
[1] 2698 3765
> .test.counter
[1] 4
> stopCluster(cl)
```

We can see from the output of `clusterEvalQ` that the first slave process performed 2698 (41.71%) conditional tests, the second one 3765 (58.21%) and that only 4 tests were performed by the master process. The difference in the number of tests between the two slaves is due to the topology of the network; different nodes have Markov blankets and neighbourhood sets of different sizes, and require different numbers of tests.

Increasing the number of slave processes reduces the number of tests performed by each slave, further increasing the overall performance of the algorithm.

```
> cl <- makeCluster(3, type = "MPI")
        3 slaves are spawned successfully. 0 failed.
> res = gs(hailfinder, cluster = cl)
> unlist(clusterEvalQ(cl, .test.counter))
[1] 1667 2198 2598
> stopCluster(cl)
> cl <- makeCluster(4, type = "MPI")
        4 slaves are spawned successfully. 0 failed.
> res = gs(hailfinder, cluster = cl)
> unlist(clusterEvalQ(cl, .test.counter))
[1] 1116 1582 1860 1905
> stopCluster(cl)
```

The execution times of the Grow-Shrink algorithm for clusters of 2, 3, 4, 5 and 6 slaves are reported in Figure 3.5. It is clear from the figure that the gains in execution time follow the *law of diminishing returns* – i.e. adding more slave processes produces less and less improvements, up to the point where the increased overhead of the communications between the master and the slave processes starts actually degrading performance.

Another important consideration is whether the data set we are learning the network from is actually big enough (both in the number of observations and nodes) to make

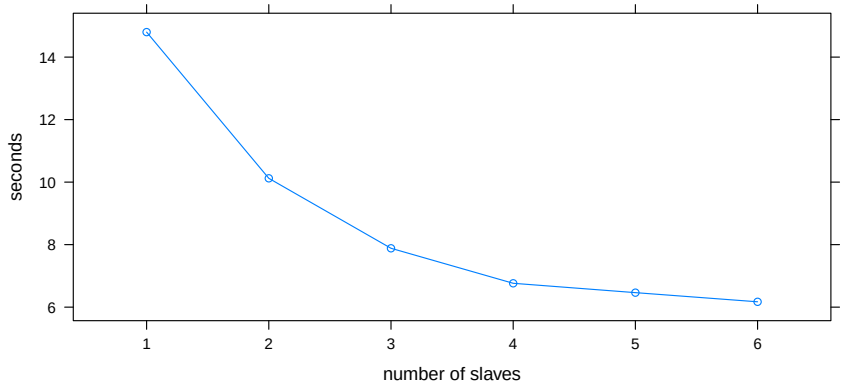


Figure 3.5: Performance of the Grow-Shrink algorithm for different numbers of slave processes, measured by its execution time (in seconds).

the use of the parallel implementation of a learning algorithm worthwhile. In fact, for `hailfinder` the traditional implementation of the Grow-Shrink algorithm is faster than the parallel one.

```
> system.time(gs(hailfinder))
   user  system elapsed
  4.000   0.004   4.004
```

There are three reasons for this disparity. First, the parallel implementation can not take advantage of the symmetry of the Markov blankets and the neighbourhood sets to reduce the number of tests. Both the Markov blanket and neighbourhood of each node are learned at the same time, so we do not know which nodes are part of the other ones. This more than doubles the number of conditional independence tests required by the algorithm.

```
> ntests(gs(hailfinder, optimized = TRUE))
[1] 2670
> ntests(gs(hailfinder, optimized = FALSE))
[1] 6467
```

Second, tests are almost never split in an optimal way among the slave processes. This can be seen quite clearly from the examples illustrated in this section: with 4 slaves the number of tests assigned to each of them range from 1116 (17.25% of the total) to 1905 (29.45% of the total). This variability introduces additional overhead in

the algorithm, because faster slaves (the ones that have fewer tests to perform) must wait for slower ones each time the status of the cluster has to be synchronized.

Third, passing data back and forth between the master and the slaves also takes some time. This is particularly relevant when the master and the slaves are separate processes (as opposed to separate threads within the same process), because in this case data have to be copied around multiple times instead of having a single copy of the data for all the slaves. The efficiency of such an operation depends on the operating system and the hardware the cluster is running on, so it must be evaluated on a case-by-case basis.

3.5.2 Score-based Algorithms

Score-based learning algorithms, being based on general-purpose optimization heuristics, benefit from several decades of research efforts aimed at taking advantage of the benefits offered by parallel computing (Rauber and Rünger, 2010).

Most score-based algorithm are *inherently sequential* in nature. Consider for example hill-climbing. In each iteration the state of the previous iteration is used as the starting point for the search of a new, better network structure. This is also true for tabu search and genetic algorithms, and makes the parallel implementation of these algorithms a challenging problem.

One possible solution is to provide a parallel implementation of the computations performed within a single iteration and to let the master process execute the iterations in a sequential way, synchronizing the status of the slaves each time. This would reduce a sequential problem to a fine-grained parallel one; it is known as the *move acceleration model* (if each slave computes part of the score of each candidate network) or the *parallel moves model* (if each slave manages some of the candidate networks). However, the resulting performance gain is likely to be outweighed by the overhead of the communications between the slaves and the master process.

Another solution, called the *parallel multistart model*, consists in starting several instances of the score-based algorithm from different starting networks. The use of significantly different starting points for the search improves the algorithm's ability to cover the search space and results in better and more robust solutions. For example, even if one of the instances gets stuck on a local maximum, another one may still find the global maximum and in this case the suboptimal solution is simply discarded.

We can easily implement the parallel multistart model using the functions provided in **bnlearn**. First, we need to generate the starting point for the search instances.

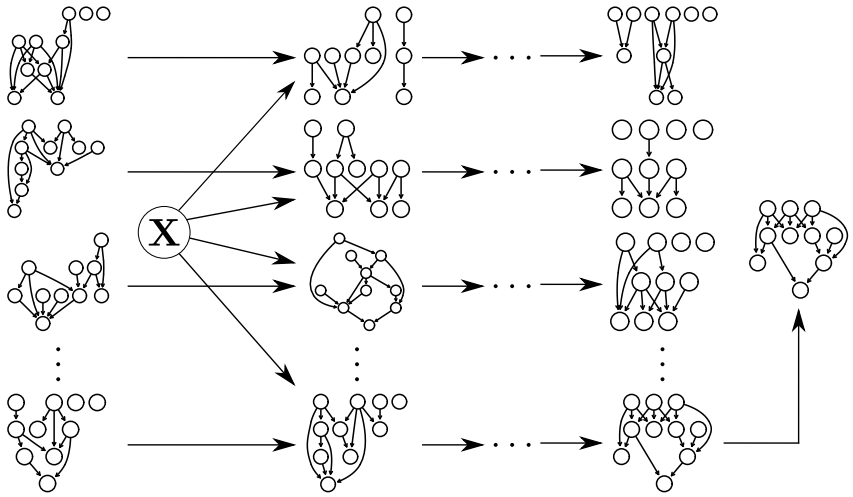


Figure 3.6: Parallel multistart implementation of a score-based learning algorithm.

Unless we have some prior knowledge about the structure of the network we can just generate them at random.

```
> r = random.graph(names(hailfinder), num = 4,
+   method = "melancon")
```

Then we can start the `snow` cluster and have each slave perform a hill-climbing search using one of the network we just generated.

```
> cl = makeCluster(4, type = "MPI")
> clusterEvalQ(cl, library(bnlearn))
> parallel.multistart = function(start) {
+   hc(hailfinder, start = start)
+ }
> l = parLapply(cl, r, parallel.multistart)
```

Once all slave processes have completed their searches, we can examine the scores of the network structures they returned.

```
> unlist(lapply(l, score, data = hailfinder))
[1] -992833.1 -993954.7 -990474.8 -1011764
```

The best one is the third (-990474.8); the other ones correspond to local maxima.

Tabu search can be easily modified in the same way, and with similar results.

```
> parallel.multistart = function(start) {
+   tabu(hailfinder, start = start)
+ }
> l = parLapply(cl, r, parallel.multistart)
> unlist(lapply(l, score, data = hailfinder))
[1] -990474.8 -997597.7 -991934 -993547.3
```

It is important to note that the execution time of the structure learning is not reduced by the parallel multistart, because each of the instances executed by the slaves processes takes (on average) as much time as the original score-based algorithm.

```
> r0 = random.graph(names(hailfinder),
+   method = "melancon")
> system.time(tabu(hailfinder, start = r0))
   user  system elapsed
414.130   0.000 414.137
> system.time(parLapply(cl, r, parallel.multistart))
   user  system elapsed
  0.020   0.010 432.221
```

3.5.3 Hybrid Algorithms

The advantages that parallel computing can bring to hybrid algorithms depend on the exact implementation of the *restrict* and *maximize* phases.

The *restrict* phase is usually implemented using the first two steps of a constraint-based algorithm or using another *local search algorithm*. Some examples of the latter are the simple ones proposed in Friedman et al. (1999b) for the Sparse Candidate algorithm or the ones, such as the ARACNE algorithm (Margolin et al., 2006), investigated in Meloni et al. (2009). Therefore all the considerations we made in Section 3.5.1 apply.

The *maximize* phase is usually implemented using a score-based learning algorithm. The computational cost of this phase is reduced by the constraints learned in the *restrict* phase, which enforce the sparseness of the network structure. This in turn guarantees a reasonable performance for most real-world data sets. All the considerations we made in Section 3.5.2 still apply; for example we can still implement the multistart model if we take care to select starting networks that satisfy the constraints.

Chapter 4

Multivariate Discrete Distributions in Structure Modelling

In this chapter we will introduce the multivariate Bernoulli and multivariate Trinomial distributions, which will provide the theoretical foundations for the variability measures defined in Chapter 5. We will also derive some properties related to their first and second order moments using results from probability and graph theory.

4.1 Modelling Graphical Structures

The network structure encodes a significant part of the information provided by a Bayesian network; it provides a simple, qualitative description of the relationships among the variables in $\mathbf{X}$. These relationships may even be the main interest of the analysis. This is often the case, for example, when learning Bayesian networks as causal graphical models under the assumptions detailed in Section 2.3.

For these reasons it is important to provide a full probabilistic model for the structure of a Bayesian network, so that we can define proper descriptive statistics and inference procedures. Of particular interest are measures of *confidence* and *variability*. The former provide a measure of the probability that a particular structural feature, such as a particular arc or the composition of a Markov blanket, may be the result of the real dependency structure of the data instead of being an artifact produced by the noisiness of the data. The latter measure how much a network structure is stable, and how each part contributes to the variability of the network as a whole.

Confidence measures have been developed by Friedman et al. (1999a) using bootstrap resampling, and later modified by Imoto et al. (2002) to estimate the marginal confidence in the presence of an arc (called *edge intensity*, and also known as *arc strength*) and its direction. Their approach can be summarized as follows:

1. For $b = 1, 2, \ldots, m$

   (a) sample a new data set $\mathbf{X}_b^*$ from the original data $\mathbf{X}$ using either parametric or nonparametric bootstrap.

   (b) learn a Bayesian network $G_b = (\mathbf{V}, A_b)$ from $\mathbf{X}_b^*$.

2. Estimate the confidence in each feature f of interest as

$$\hat{\mathrm{P}}(f) = \frac{1}{m} \sum_{b=1}^{m} f(G_b). \tag{4.1}$$

However, the empirical probabilities $\hat{\mathrm{P}}(f)$ are difficult to evaluate, because the distribution of $\mathcal{G}$ in the space of DAGs is unknown and because the confidence threshold value is an unknown function of both the data and the structure learning algorithm.

These limitations have severely limited the usefulness of the approach proposed by Friedman et al. (1999a), and have thus far prevented the development of effective measures of variability. In most cases Bayesian network structure learning algorithms are still studied using a small number of well-known reference data sets as benchmarks; a collection is maintained by Elidan (2001) in the *Bayesian Network Repository*. Differences from the true structure of these networks, which are known from literature, are measured with descriptive measures such as Hamming distance (Jungnickel, 2008) or the Structural Hamming Distance (Tsamardinos et al., 2006). As for Bayesian networks learned from real-world data, this approach to model validation is clearly not possible. Confidence is studied using pre-defined significance thresholds (a data-driven approach has been proposed by Nagarajan et al. (2010) and is currently being improved) and variability is usually not investigated at all.

Better solutions are possible once a probability distribution for the network structure is defined. We will first note that in the context of graphical models a network is uniquely identified by its arc set A (or by its edge set E in the case of undirected graphs), and that each arc or edge is uniquely identified by the nodes X_i and X_j, $i \neq j$ it is incident on.

Furthermore, an edge or an arc has only few possible states:

- an edge e_{ij} can be either present ($\{X_i - X_j\} \in E$) or missing from an undirected graph ($\{X_i - X_j\} \notin E$);
- in a directed graph and arc a_{ij} can be present in one of its two possible orientations ($\{X_i \to X_j\} \in A$ or $\{X_j \to X_i\} \in A$) or missing from the graph ($\{X_i \to X_j\} \notin A$ and $\{X_j \to X_i\} \notin A$).

This leads to the natural choice of a Bernoulli random variable for the first case,

$$e_{ij} \sim E_{ij} = \begin{cases} 1 & e_{ij} \in E \text{ with probability } p_{ij} \\ 0 & e_{ij} \notin E \text{ with probability } 1 - p_{ij} \end{cases}, \tag{4.2}$$

and to the choice of a Trinomial random variable for the second case,

$$a_{ij} \sim A_{ij} = \begin{cases} -1 & \overleftarrow{a_{ji}} \in A \text{ with probability } \overleftarrow{p_{ij}} \\ 0 & \overleftarrow{a_{ij}}, \overrightarrow{a_{ij}} \notin A \text{ with probability } \mathring{p}_{ij} \\ 1 & \overrightarrow{a_{ij}} \in A \text{ with probability } \overrightarrow{p_{ij}} \end{cases}, \tag{4.3}$$

where $\overrightarrow{a_{ij}}$ is $\{X_i \to X_j\}$ and $\overleftarrow{a_{ij}}$ is $\{X_j \to X_i\}$. Therefore a network structure can be modelled through its arc or edge set as follows:

- undirected graphs, such as Markov networks or the skeleton and the moral graph of Bayesian networks, can be modelled by a *multivariate Bernoulli random variable*.
- directed graphs, such as the directed acyclic graphs used in Bayesian networks, can be modelled by a *multivariate Trinomial random variable*.

Formal definitions and properties of these distributions will be covered in detail in Section 4.2 and Section 4.3.

In addition to being the natural choice for a graphical structure, multivariate Bernoulli and Trinomial distributions are able to integrate smoothly with and extend the approaches presented above. The probabilities associated with each arc or edge correspond to the confidence from Friedman et al. (1999a) and the arc strength from Imoto et al. (2002), and can be estimated using bootstrap as in Equation 4.1. Distance and variability measures can be constructed from the first and second order moments of the multivariate distributions, providing statistically motivated descriptive measures

and hypothesis testing. Some examples of both will be introduced in Chapter 5 for variability measures.

The choice of bootstrap as an estimation technique is motivated by two considerations. First, the true structure of the network and its distribution are both unknown, and cannot be estimated from the real-world data in closed form. Therefore some kind of nonparametric approach, such as bootstrap, is required. Another option is Markov Chain Monte Carlo (MCMC) simulations (Friedman and Koller, 2003), but they are computationally expensive and convergence is problematic with large number of variables (Koller and Friedman, 2009). Second, bootstrap greatly simplifies the analysis of variability and its interpretation; network structures learned from the bootstrap samples are independent, which makes the estimation of second order moments straightforward (this is not true for networks obtained from MCMC simulations). Furthermore, as far as the the second order moments are concerned, the outcome of bootstrap estimation can be summarized in three cases according to the entropy (Cover and Thomas, 2006) of the set of the learned networks:

- *minimum entropy*: all the networks learned from the bootstrap samples have the same structure, that is

$$E_1 = E_2 = \ldots = E_m = E \qquad \text{or} \qquad A_1 = A_2 = \ldots = A_m = A. \tag{4.4}$$

  This is the best possible outcome of the resampling, because there is no variability in the estimated network structure.

- *intermediate entropy*: several network structures are observed with different frequencies m_b, $\sum m_b = m$. This is the case for almost all real-world data.

- *maximum entropy*: all possible network structures appear with the same frequency, that is

$$\mathrm{P}(U) = c \qquad \text{or} \qquad \mathrm{P}(G) = c \qquad \text{for every possible } U \text{ and } G. \tag{4.5}$$

  This is the worst possible outcome. In fact, this case corresponds to the non-informative prior distribution on the space of network structures used in computing the BDe and BGe network scores.

The values assumed by the first and second order moments and the related parameters of the multivariate Bernoulli and Trinomial distributions will be derived for these three cases in Section 4.4.

4.2 The Multivariate Bernoulli Distribution

Let $B_1, B_2, \ldots, B_k$, $k \in \mathbb{N}$ be Bernoulli random variables with marginal probability of success $p_1, p_2, \ldots, p_k$, that is $B_i \sim Ber(p_i)$, $i = 1, \ldots, k$. Then the distribution of the random vector $\mathbf{B} = [B_1, B_2, \ldots, B_k]^T$ over the joint probability space of $B_1, B_2, \ldots, B_k$ is a *multivariate Bernoulli random variable* (Krummenauer, 1998b), denoted as $Ber_k(\mathbf{p})$. Its probability function is uniquely identified by the parameter collection

$$\mathbf{p} = \{p_I : I \subseteq \{1, \ldots, k\}, I \neq \varnothing\}, \tag{4.6}$$

which represents the *dependence structure* among the marginal distributions in terms of simultaneous successes for every non-empty subset I of elements of the random vector.

However, several useful results depend only on the first and second order moments of $\mathbf{B}$,

$$\mathsf{E}(B_i) = p_i, \qquad \mathsf{VAR}(B_i) = p_i - p_i^2 \qquad \text{and} \qquad \mathsf{COV}(B_i, B_j) = p_{ij} - p_i p_j, \tag{4.7}$$

and the corresponding reduced parameter collection

$$\tilde{\mathbf{p}} = \{p_{ij} : i, j = 1, \ldots, k\}, \tag{4.8}$$

which is in fact used as an approximation of $\mathbf{p}$ in the generation random multivariate Bernoulli vectors in Krummenauer (1998a).

4.2.1 Uncorrelation and Independence

We will first consider a simple result that links covariance and independence of two univariate Bernoulli variables.

Theorem 4.1. *Let B_i and B_j be two Bernoulli random variables. Then B_i and B_j are independent if and only if their covariance is zero:*

$$B_i \perp\!\!\!\perp B_j \iff \mathsf{COV}(B_i, B_j) = 0 \tag{4.9}$$

Proof. If B_i and B_j are independent then by definition

$$\mathsf{COV}(B_i, B_j) = p_{ij} - p_i p_j = \mathrm{P}(B_i = 1, B_j = 1) - \mathrm{P}(B_i = 1)\,\mathrm{P}(B_j = 1) = 0, \tag{4.10}$$

as $\mathrm{P}(B_i = 1, B_j = 1) = \mathrm{P}(B_i = 1)\,\mathrm{P}(B_j = 1)$.

If, on the other hand, we have that $\mathsf{COV}(B_i, B_j) = 0$, then

$$p_{ij} - p_i p_j = 0 \Rightarrow p_{ij} = p_i p_j \Rightarrow B_i \perp\!\!\!\perp B_j \tag{4.11}$$

which completes the proof. □

This theorem can be extended to multivariate Bernoulli random variables as follows.

Theorem 4.2. *Let* $\mathbf{B} = [B_1, B_2, \ldots, B_k]^T$ *and* $\mathbf{C} = [C_1, C_2, \ldots, C_l]^T$, $k, l \in \mathbb{N}$ *be two multivariate Bernoulli random variables. Then* $\mathbf{B}$ *and* $\mathbf{C}$ *are independent if and only if*

$$\mathbf{B} \perp\!\!\!\perp \mathbf{C} \iff \mathsf{COV}(\mathbf{B}, \mathbf{C}) = \mathbf{O} \tag{4.12}$$

where $\mathbf{O}$ *is the zero matrix.*

Proof. If $\mathbf{B}$ is independent from $\mathbf{C}$, then by definition every pair (B_i, C_j), $i = 1, \ldots, k$, $j = 1, \ldots, l$ is independent. Therefore the covariance matrix of $\mathbf{B}$ and $\mathbf{C}$ is

$$\mathsf{COV}(B_i, C_j) = c_{ij} = 0 \Longrightarrow \mathsf{COV}(\mathbf{B}, \mathbf{C}) = [c_{ij}] = \mathbf{O}. \tag{4.13}$$

If conversely the covariance matrix $\mathsf{COV}(\mathbf{B}, \mathbf{C})$ is equal to the zero matrix, every pair (B_i, C_j) is independent as

$$c_{ij} = p_{ij} - p_i p_j = 0 \Longrightarrow p_{ij} = p_i p_j \tag{4.14}$$

This implies the independence of the random vectors $\mathbf{B}$ and $\mathbf{C}$, as their sigma-algebras

$$\sigma(\mathbf{B}) = \sigma(B_1) \times \ldots \times \sigma(B_k) \qquad \text{and} \qquad \sigma(\mathbf{C}) = \sigma(C_1) \times \ldots \times \sigma(C_l) \tag{4.15}$$

are functions of the sigma algebras induced by the two sets of independent random variables $B_1, B_2, \ldots, B_k$ and $C_1, C_2, \ldots, C_l$. □

The correspondence between uncorrelation and independence is identical to the analogous property of the multivariate Gaussian distribution (Ash, 2000), and is closely related to the strong normality defined for orthogonal second order random variables in Loève (1977). It can also be applied to disjoint subsets of components of a single multivariate Bernoulli variable, as they are also distributed as multivariate Bernoulli random variables.

Theorem 4.3. *Let* $\mathbf{B} = [B_1, B_2, \ldots, B_k]^T$ *be a multivariate Bernoulli random variable; then every random vector* $\mathbf{B}^* = [B_{i_1}, B_{i_2}, \ldots, B_{i_l}]^T$, $\{i_1, i_2, \ldots, i_l\} \subseteq \{1, 2, \ldots, k\}$ *is a multivariate Bernoulli random variable.*

Proof. The marginal components of $\mathbf{B}^*$ are Bernoulli random variables, because $\mathbf{B}$ is multivariate Bernoulli. The new dependency structure is defined as

$$\mathbf{p}^* = \{p_{I^*} : I^* \subseteq \{i_1, \ldots, i_l\} \subseteq \{1, \ldots, k\}, I^* \neq \varnothing\}, \tag{4.16}$$

and uniquely identifies the probability distribution of $\mathbf{B}^*$. □

Example 4.1. *Consider the trivariate Bernoulli random variable*

$$\mathbf{B} = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix} = \mathbf{B_1} + \mathbf{B_2} \quad \textit{where} \quad \mathbf{B_1} = \begin{bmatrix} 0 \\ B_2 \\ 0 \end{bmatrix} \quad \textit{and} \quad \mathbf{B_2} = \begin{bmatrix} B_1 \\ 0 \\ B_3 \end{bmatrix}. \tag{4.17}$$

Then the covariance matrix

$$\mathsf{COV}(\mathbf{B_1}, \mathbf{B_2}) = \mathsf{E}\left(\begin{bmatrix} 0 \\ B_2 \\ 0 \end{bmatrix} \begin{bmatrix} B_1 & 0 & B_3 \end{bmatrix}\right) - \mathsf{E}\left(\begin{bmatrix} 0 \\ B_2 \\ 0 \end{bmatrix}\right) \mathsf{E}\left(\begin{bmatrix} B_1 & 0 & B_3 \end{bmatrix}\right) \tag{4.18}$$

$$= \mathsf{E}\left(\begin{bmatrix} 0 & 0 & 0 \\ B_1B_2 & 0 & B_2B_3 \\ 0 & 0 & 0 \end{bmatrix}\right) - \begin{bmatrix} 0 \\ p_2 \\ 0 \end{bmatrix} \begin{bmatrix} p_1 & 0 & p_3 \end{bmatrix} \tag{4.19}$$

$$= \begin{bmatrix} 0 & 0 & 0 \\ p_{12} & 0 & p_{23} \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ p_1p_2 & 0 & p_2p_3 \\ 0 & 0 & 0 \end{bmatrix} = \tag{4.20}$$

$$= \begin{bmatrix} 0 & 0 & 0 \\ p_{12} - p_1p_2 & 0 & p_{23} - p_2p_3 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.21}$$

of the two components $\mathbf{B_1}$ *and* $\mathbf{B_2}$ *is equal to the zero matrix if and only if*

$$\begin{cases} p_{12} = p_1p_2 \\ p_{23} = p_2p_3 \end{cases} \Longrightarrow \{B_1 \perp\!\!\!\perp B_2, B_2 \perp\!\!\!\perp B_3\} \tag{4.22}$$

which in turn implies and is implied by $\mathbf{B_1} \perp\!\!\!\perp \mathbf{B_2}$.

4.2.2 Properties of the Covariance Matrix

The covariance matrix $\Sigma = [\sigma_{ij}]$, $i, j = 1, \ldots, k$ associated with a multivariate Bernoulli random vector has some interesting numerical properties. Due to the form of the central second order moments reported in Equation 4.7, the diagonal elements are bounded in the interval

$$\sigma_{ii} = p_i - p_i^2 \in \left[0, \frac{1}{4}\right]. \tag{4.23}$$

The maximum is attained for $p_i = \frac{1}{2}$, and the minimum for both $p_i = 0$ and $p_i = 1$. For the Cauchy-Schwarz theorem then

$$0 \leqslant \sigma_{ij}^2 \leqslant \sigma_{ii}\sigma_{jj} \leqslant \frac{1}{16} \Longrightarrow |\sigma_{ij}| \in \left[0, \frac{1}{4}\right]. \tag{4.24}$$

The eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_k$ of Σ are similarly bounded, as shown in the following theorem.

Theorem 4.4. *Let* $\mathbf{B} = [B_1, B_2, \ldots, B_k]^T$ *be a multivariate Bernoulli random variable, and let* $\Sigma = [\sigma_{ij}]$, $i, j = 1, \ldots, k$ *be its covariance matrix. Let* λ_i, $i = 1, \ldots, k$ *be the eigenvalues of* Σ*. Then*

$$0 \leqslant \sum_{i=1}^{k} \lambda_i \leqslant \frac{k}{4} \qquad \textit{and} \qquad 0 \leqslant \lambda_i \leqslant \frac{k}{4}. \tag{4.25}$$

Proof. Since Σ is a real, symmetric, non-negative definite matrix, the eigenvalues λ_i are non-negative real numbers (Seber, 2008); this proves the lower bound in both inequalities.

The upper bound in the first inequality holds because

$$\sum_{i=1}^{k} \lambda_i = \sum_{i=1}^{k} \sigma_{ii} \leqslant \max_{\{\sigma_{ii}\}} \sum_{i=1}^{k} \sigma_{ii} = \sum_{i=1}^{k} \max \sigma_{ii} = \frac{k}{4}, \tag{4.26}$$

as the sum of the eigenvalues is equal to the trace of Σ. This in turn implies

$$\lambda_i \leqslant \sum_{i=1}^{k} \lambda_i \leqslant \frac{k}{4}, \tag{4.27}$$

which completes the proof. □

These bounds define a convex set in $\mathbb{R}^k$, defined by the family

$$\mathcal{D} = \left\{ \Delta^{k-1}(c) : c \in \left[0, \frac{k}{4}\right] \right\} \tag{4.28}$$

where $\Delta^{k-1}(c)$ is the non-standard $k-1$ simplex

$$\Delta^{k-1}(c) = \left\{ (\lambda_1, \ldots, \lambda_k) \in \mathbb{R}^k : \sum_{i=1}^{k} \lambda_i = c, \lambda_i \geqslant 0 \right\}. \tag{4.29}$$

4.2.3 Sequences of Multivariate Bernoulli Variables

We will now consider a sequence of independent and identically distributed multivariate Bernoulli variables $\mathbf{B_1}, \mathbf{B_2}, \ldots, \mathbf{B_m} \sim Ber_k(\mathbf{p})$. The sum

$$\mathbf{S}_m = \sum_{i=1}^{m} \mathbf{B}_i \sim Bi_k(m, \mathbf{p}) \tag{4.30}$$

is distributed as a *multivariate Binomial random variable* (Krummenauer, 1998b), thus preserving one of the fundamental properties of the univariate Bernoulli distribution. A similar result holds for the *law of small numbers* (Billingsley, 1995), whose multivariate version states that a k-variate Binomial distribution $Bi_k(m, \mathbf{p})$ converges to a *multivariate Poisson distribution* $P_k(\mathbf{\Lambda})$:

$$\mathbf{S}_m \xrightarrow{d} P_k(\mathbf{\Lambda}) \qquad \text{as} \qquad m\mathbf{p} \to \mathbf{\Lambda}. \tag{4.31}$$

Both these distributions' probability functions, while tractable, are not very useful as a basis for closed-form inference procedures. An alternative is given by the asymptotic *multivariate Gaussian distribution* defined by the *multivariate central limit theorem* (Ash, 2000):

$$\frac{\mathbf{S}_m - m\mathsf{E}(\mathbf{B}_1)}{\sqrt{m}} \xrightarrow{d} N_k(\mathbf{0}, \Sigma). \tag{4.32}$$

The limiting distribution is guaranteed to exist for all possible values of $\mathbf{p}$, as the first two orders of moments are bounded and therefore are always finite.

4.3 The Multivariate Trinomial Distribution

The multivariate Trinomial distribution is the multivariate extension of the univariate Trinomial distribution, and its construction is similar to the multivariate Bernoulli.

They are both particular cases of the *multivariate Multinomial distribution* (Wishart, 1949; Johnson et al., 1997), and for this reason they share many common traits.

Let $T_1, T_2, \ldots, T_k$, $k \in \mathbb{N}$ be Trinomial random variables assuming values t_{i1}, t_{i2} and t_{i3}, $i = 1, \ldots, k$ and denoted as $T_i \sim Tri\left(p_{i(t_{i1})}, p_{i(t_{i2})}, p_{i(t_{i3})}\right)$, where

$$\mathrm{P}(T_i = t_{i1}) = p_{i(t_{i1})}, \qquad \mathrm{P}(T_i = t_{i2}) = p_{i(t_{i2})} \qquad \text{and} \qquad \mathrm{P}(T_i = t_{i3}) = p_{i(t_{i3})} \tag{4.33}$$

with $p_{i(t_{i1})} + p_{i(t_{i2})} + p_{i(t_{i3})} = 1$. Then the distribution of the random vector $\mathbf{T} = [T_1, T_2, \ldots, T_k]^T$ over the joint probability space of $T_1, T_2, \ldots, T_k$ is a *multivariate Trinomial random variable*, denoted as $Tri_k(\mathbf{p})$. The parameter collection $\mathbf{p}$ which uniquely identifies the distribution is

$$\mathbf{p} = \left\{ p_{I(T)} : I \subseteq \{1, \ldots, k\},\, T \in \bigtimes_{i=1}^{|I|} \{t_{i1}, t_{i2}, t_{i3}\},\, I \neq \varnothing \right\} \tag{4.34}$$

and the reduced parameter collection we will need to study the first and second order moments is

$$\tilde{\mathbf{p}} = \left\{ p_{ij(T)} : i, j = 1, \ldots, k,\, T \in \{t_{i1}, t_{i2}, t_{i3}\} \times \{t_{j1}, t_{j2}, t_{j3}\} \right\}. \tag{4.35}$$

In this thesis we will limit ourselves to Trinomial distributions T_i assuming values

$$t_{i1} = -1, \qquad t_{i2} = 0 \qquad \text{and} \qquad t_{i3} = 1 \tag{4.36}$$

for every $i = 1, \ldots, k$; we will denote them with t_1, t_2 and t_3 respectively. The corresponding parameters are

$$\mathrm{P}(T_i = -1) = p_{i(-1)}, \qquad \mathrm{P}(T_i = 0) = p_{i(0)} \qquad \text{and} \qquad \mathrm{P}(T_i = 1) = p_{i(1)}, \tag{4.37}$$

so we will write $T_i \sim Tri\left(p_{i(-1)}, p_{i(0)}, p_{i(1)}\right)$. The expected value and the variance of T_i are

$$\mathsf{E}(T_i) = p_{i(1)} - p_{i(-1)} \tag{4.38}$$

$$\mathsf{VAR}(T_i) = p_{i(1)} + p_{i(-1)} - \left[\, p_{i(1)} - p_{i(-1)} \,\right]^2 \tag{4.39}$$

and the covariance between two variables T_i and T_j is equal to

$$\begin{aligned}\mathsf{COV}(T_i, T_j) = &\left[p_{ij(1,1)} - p_{i(1)}p_{j(1)} \right] + \left[p_{ij(-1,-1)} - p_{i(-1)}p_{j(-1)} \right] - \\ &- \left[p_{ij(-1,1)} - p_{i(-1)}p_{j(1)} \right] - \left[p_{ij(1,-1)} - p_{i(1)}p_{j(-1)} \right]. \end{aligned} \tag{4.40}$$

4.3.1 Relationship with the Multivariate Bernoulli

The choice of -1, 0 and 1 as the values assumed by the elements T_i of $\mathbf{T}$ establishes a strong link between the Trinomial and the Bernoulli distribution.

Theorem 4.5. *Let T be a Trinomial random variable assuming values $t_1 = -1$, $t_2 = 0$, and $t_3 = 1$ with probabilities $p_{(-1)}$, $p_{(0)}$, and $p_{(1)}$, $p_{(-1)} + p_{(0)} + p_{(1)} = 1$. Then $|T| = B \sim Ber(p)$, where $p = p_{(-1)} + p_{(1)}$.*

Proof. The transformed random variable $|T|$ can assume only two values, $|t_2| = 0$ and $|t_1| = |t_3| = 1$, the former with probability $1 - p = p_{(0)}$ and the latter with probability $p = p_{(-1)} + p_{(1)}$. Therefore $|T| \sim Ber(p)$. □

This link can easily be extended to the multivariate case.

Theorem 4.6. *Let $\mathbf{T} = [T_1, T_2, \ldots, T_k]^T$ be a multivariate Trinomial random variable whose components have the same distribution as the univariate Trinomial described in Theorem 4.5. Then $|\mathbf{T}| = \mathbf{B} \sim Ber_k(\mathbf{p})$.*

Proof. As proved in Theorem 4.5, each element $|T_i|$ of the random vector $|\mathbf{T}|$ is a Bernoulli random variable. Furthermore the parameter collection of $\mathbf{T}$ assumes the form

$$\mathbf{p} = \left\{ p_{I(T)} : I \subseteq \{1, \ldots, k\},\, T \in \{-1, 0, 1\}^{|I|},\, I \neq \varnothing \right\} \tag{4.41}$$

and reduces to

$$\begin{aligned}\mathbf{p} &= \left\{ p_{I(T)} : I \subseteq \{1, \ldots, k\},\, T \in \{0, 1\}^{|I|},\, I \neq \varnothing \right\} \\ &= \{ p_I : I \subseteq \{1, \ldots, k\},\, I \neq \varnothing \} \end{aligned} \tag{4.42}$$

after the transformation. Therefore $|\mathbf{T}| \sim Ber_k(\mathbf{p})$ is a uniquely identified multivariate Bernoulli random variable according to the definition introduced at the beginning of Section 4.2. □

An important consequence of Theorem 4.6 is that the first and second order moments of the absolute value of a multivariate Trinomial simplify to those of the multivariate

Bernoulli.

$$\begin{aligned}
\mathsf{E}(|T_i|) &= p_{(-1)} + p_{(1)} = p_i && (4.43)\\
\mathsf{VAR}(|T_i|) &= p_{(-1)} + p_{(1)} - \left[\, p_{(-1)} + p_{(1)} \,\right]^2 = p_i - p_i^2 && (4.44)\\
\mathsf{COV}(|T_i|, |T_j|) &= \left[\, p_{ij(1,1)} + p_{i(1)}p_{j(1)} \,\right] + \left[\, p_{ij(-1,-1)} + p_{i(-1)}p_{j(-1)} \,\right] - \\
&\quad - \left[\, p_{ij(-1,1)} + p_{i(-1)}p_{j(1)} \,\right] - \left[\, p_{ij(1,-1)} + p_{i(1)}p_{j(-1)} \,\right] \\
&= p_{ij} - \left[\, p_{i(-1)} + p_{i(1)} \,\right] \left[\, p_{j(-1)} + p_{j(1)} \,\right] \\
&= p_{ij} - p_i p_j && (4.45)
\end{aligned}$$

Furthermore, the variance of each univariate Trinomial T_i can be decomposed in two parts: one is a function of the corresponding component $|T_i| = B_i$ of the transformed random vector, while the other depends only on the probabilities associated with -1 and 1.

$$\begin{aligned}
\mathsf{VAR}(T_i) &= p_{i(1)} + p_{i(-1)} - \left[\, p_{i(1)} - p_{i(-1)} \,\right]^2 \\
&= p_{i(1)} + p_{i(-1)} - \left[\, p_{i(1)} + p_{i(-1)} \,\right]^2 + 4p_{i(1)}p_{i(-1)} \\
&= p_i - p_i^2 + 4p_{i(1)}p_{i(-1)} \\
&= \mathsf{VAR}(B_i) + 4p_{i(1)}p_{i(-1)} && (4.46)
\end{aligned}$$

Covariance can be decomposed in a similar way.

$$\begin{aligned}
\mathsf{COV}(T_i, T_j) &= \left[\, p_{ij(1,1)} - p_{i(1)}p_{j(1)} \,\right] + \left[\, p_{ij(-1,-1)} - p_{i(-1)}p_{j(-1)} \,\right] - \\
&\quad - \left[\, p_{ij(-1,1)} - p_{i(-1)}p_{j(1)} \,\right] - \left[\, p_{ij(1,-1)} - p_{i(1)}p_{j(-1)} \,\right] \\
&= p_{ij} - p_i p_j - 2\left[\, p_{ij(-1,1)} - p_{i(-1)}p_{j(1)} \,\right] - 2\left[\, p_{ij(1,-1)} - p_{i(1)}p_{j(-1)} \,\right] \\
&= \mathsf{COV}(B_i, B_j) - \\
&\quad - 2\left[\, p_{ij(-1,1)} - p_{i(-1)}p_{j(1)} + p_{ij(1,-1)} - p_{i(1)}p_{j(-1)} \,\right]. && (4.47)
\end{aligned}$$

4.3.2 Properties of the Covariance Matrix

The covariance matrix Σ of a multivariate Trinomial random vector has again several interesting numerical properties, as was the case for the multivariate Bernoulli distribution. The form of Σ and some of its properties are similar to those derived in Section 4.2.2, with the notable exception of the correspondence between uncorrelation and stochastic independence proved in Section 4.2.1.

For example, the diagonal elements σ_{ii} of Σ are again bounded. This can be proved either by solving the constrained maximization problem

$$\max_{p_{i(1)}, p_{i(-1)}} \mathsf{VAR}(T_i) = p_{i(1)} + p_{i(-1)} - \left[p_{i(1)} - p_{i(-1)} \right]^2$$
$$\text{s.t.} \quad p_{i(1)} \geqslant 0, p_{i(-1)} \geqslant 0, p_{i(1)} + p_{i(-1)} \leqslant 1 \quad (4.48)$$

with the extended Lagrange multipliers method (Nocedal and Wright, 1999) or as a direct consequence of the following theorem by Moors and Muilwijk (1971).

Theorem 4.7. *If a discrete random variable X can take values only in the segment $[x_1, x_n]$ of the real axis, the maximum standard deviation of X equals $\frac{1}{2}(x_n - x_1)$. The maximum is reached if X takes the values x_1 and x_n with probabilities $\frac{1}{2}$ each.*

In both cases we obtain that the maximum variance is achieved for $p_{i(1)} = p_{i(-1)} = \frac{1}{2}$ and is equal to 1, which means that

$$\sigma_{ii} \in [0, 1] \qquad \text{and} \qquad |\sigma_{ij}| \in [0, 1]. \quad (4.49)$$

Furthermore, we can also prove that the eigenvalues $\lambda_1, \ldots, \lambda_k$ of Σ are bounded using the same arguments as in Theorem 4.4.

Theorem 4.8. *Let $\mathbf{T} = [T_1, T_2, \ldots, T_k]^T$ be a multivariate Trinomial random variable, and let $\Sigma = [\sigma_{ij}]$, $i, j = 1, \ldots, k$ be its covariance matrix. Let λ_i, $i = 1, \ldots, k$ be the eigenvalues of Σ. Then*

$$0 \leqslant \sum_{i=1}^{k} \lambda_i \leqslant k \qquad \textit{and} \qquad 0 \leqslant \lambda_i \leqslant k. \quad (4.50)$$

Proof. See the proof of Theorem 4.4. □

These bounds define a convex set in $\mathbb{R}^k$, defined by the family

$$\mathcal{D} = \left\{ \Delta^{k-1}(c) : c \in [0, k] \right\} \quad (4.51)$$

where $\Delta^{k-1}(c)$ is the non-standard $k-1$ simplex from Equation 4.29.

4.4 Bootstrap and Variability

We will now apply the results from Sections 4.2 and 4.3 to the analysis of the variability of the network structures described in Section 4.1. In particular we will consider the three possible outcomes of bootstrap resampling (*minimum*, *intermediate* and *maximum entropy*) and provide a characterization of the first and second order moments of the distributions associated with undirected and directed acyclic graphs.

4.4.1 Undirected Graphs

In the *minimum entropy case* all the edge sets $E_1, \ldots, E_m$ of the bootstrapped graphs are equal, which means that an edge e_{ij} is either present in all of them or is never present; therefore

$$\mathsf{E}(E_{ij}) = p_{ij} = \begin{cases} 1 & \text{if } e_{ij} \in E \\ 0 & \text{otherwise} \end{cases} \qquad \text{and} \qquad \Sigma = \mathbf{O}. \tag{4.52}$$

The *maximum entropy* case displays a completely different behaviour, as shown in the following theorem on the probability of one and two edges.

Theorem 4.9. *Let $U_1, \ldots, U_n$, $n = 2^m$, $m = \frac{1}{2}|\mathbf{V}|(|\mathbf{V}| - 1)$ be all possible undirected graphs with vertex set $\mathbf{V}$ and let*

$$\mathrm{P}(U_k) = \frac{1}{n} \qquad k = 1, \ldots, n. \tag{4.53}$$

Let e_{ij} and e_{kl}, $i \neq j$, $k \neq l$ be two distinct edges in $\mathbf{V} \times \mathbf{V}$. Then

$$\mathrm{P}(e_{ij}) = \frac{1}{2} \qquad \textit{and} \qquad \mathrm{P}(e_{ij}, e_{kl}) = \frac{1}{4}. \tag{4.54}$$

Proof. The number of possible structures of an undirected graph is given by the Cartesian product of the possible states of its m edges, resulting in

$$|\{0,1\} \times \ldots \times \{0,1\}| = |\{0,1\}^m| = 2^m \tag{4.55}$$

possible undirected graphs. Then e_{ij} is present in

$$|\{0,1\} \times \ldots \times \{1\} \times \ldots \times \{0,1\}| = \left|\{1\} \times \{0,1\}^{m-1}\right| = 2^{m-1} \tag{4.56}$$

graphs and e_{ij} and e_{kl} are simultaneously present in

$$|\{0,1\} \times \ldots \times \{1\} \times \{1\} \times \ldots \times \{0,1\}| = \left|\{1\}^2 \times \{0,1\}^{m-2}\right| = 2^{m-2} \tag{4.57}$$

graphs. Therefore

$$\mathrm{P}(e_{ij}) = \frac{2^{m-1}\,\mathrm{P}(U_k)}{2^m\,\mathrm{P}(U_k)} = \frac{1}{2} \qquad \text{and} \qquad \mathrm{P}(e_{ij}, e_{kl}) = \frac{2^{m-2}\,\mathrm{P}(U_k)}{2^m\,\mathrm{P}(U_k)} = \frac{1}{4}. \tag{4.58}$$

□

An immediate consequence of this theorem is that

$$p_{ij} = \frac{1}{2} \qquad \text{for every possible } e_{ij},\ i \neq j \text{ and} \qquad \Sigma = \frac{1}{4} I_m. \tag{4.59}$$

Note that each edge displays its maximum possible variability, and that the fact that all non-diagonal elements of Σ are equal to 0 proves that the edges are mutually independent according to Theorem 4.1.

The *intermediate entropy* case displays a middle ground behaviour between the *minimum* and *maximum entropy* cases. The probabilities associated with each edge and each pair of edges can be estimated from $E_1, \ldots, E_m$ with the respective empirical frequencies,

$$\hat{p}_{ij} = \frac{1}{m} \sum_{b=1}^{m} \mathbb{1}_{\{e \in E_b\}}(e_{ij}) \qquad \text{and} \qquad \hat{p}_{ij,kl} = \frac{1}{m} \sum_{b=1}^{m} \mathbb{1}_{\{e \in E_b\}}(e_{ij}) \mathbb{1}_{\{e \in E_b\}}(e_{kl}). \tag{4.60}$$

The expected value and the covariance matrix Σ do not have a definite form beyond the bounds derived in Section 4.2.2. For real-world data we have in general that most possible edges do not appear in any bootstrapped network because they represent conditional dependence relationships that are completely unsupported by the data. This means that $\mathsf{E}(e_{ij}) = 0$ and $\mathsf{VAR}(e_{ij}) = 0$ for many $e_{ij} \in \mathbf{V} \times \mathbf{V}$, so Σ is almost surely singular unless they are excluded from the analysis. Edges that appear with frequencies around $\frac{1}{2}$ have about the same marginal probability and variance as in the *maximum entropy* case, so their behaviour is very close to random noise. On the other hand, edges with probabilities near 0 or 1 are considered to have a good support (against or in favour, respectively). As $\hat{p}_{ij}$ approaches 0 or 1 e_{ij} approaches its *minimum entropy*.

The closeness of a multivariate Bernoulli distribution to the *minimum* and *maximum entropy* cases can be represented in an intuitive way by considering the eigenvalues

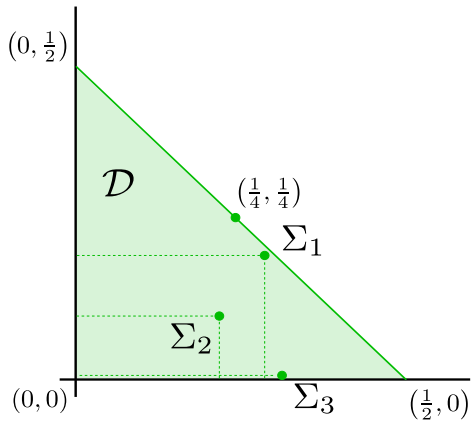


Figure 4.1: The covariance matrices Σ_1, Σ_2 and Σ_3 represented as functions of their eigenvalues in the non-standard simplex $\mathcal{D}$ (green). The points $(0, 0)$ and $(\frac{1}{4}, \frac{1}{4})$ correspond to the *minimum entropy* and *maximum entropy* cases.

$\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_k]^T$ of its covariance matrix Σ. Recall that the $\boldsymbol{\lambda}$ can assume values in the convex set $\mathcal{D}$ defined in Equation 4.29, which corresponds to the region of the first orthant delimited by the non-standard simplex $\Delta^{k-1}(\frac{k}{4})$. In the *minimum entropy* case we have that $\Sigma = \mathbf{O}$, so $\lambda_1 = \ldots = \lambda_k = 0$, and in the *maximum entropy case* $\Sigma = \frac{1}{4} I_k$, so $\lambda_1 = \ldots = \lambda_k = \frac{1}{4}$; both points lie on the boundary of $\mathcal{D}$, the first in the origin and the second in the middle of $\Delta^{k-1}(\frac{k}{4})$. Since the eigenvalues of Σ also lie in $\mathcal{D}$, the distance between $\boldsymbol{\lambda}$ and these two points provides an intuitive way of measuring the entropy of the set of bootstrapped network structures. A simple example comprising three multivariate Bernoulli distributions over a set of two edges is illustrated below.

Example 4.2. *Consider three multivariate Bernoulli distributions* $\mathbf{B}_1$, $\mathbf{B}_2$, $\mathbf{B}_3$ *over two edges (denoted with* $e_1 \sim E_1$ *and* $e_2 \sim E_2$*) with covariance matrices*

$$\Sigma_1 = \frac{1}{25}\begin{bmatrix} 6 & 1 \\ 1 & 6 \end{bmatrix}, \qquad \Sigma_2 = \frac{1}{625}\begin{bmatrix} 66 & -21 \\ -21 & 126 \end{bmatrix}, \qquad \Sigma_3 = \frac{1}{625}\begin{bmatrix} 66 & 91 \\ 91 & 126 \end{bmatrix}. \tag{4.61}$$

and eigenvalues

$$\boldsymbol{\lambda}_1 = \begin{bmatrix} 0.28 \\ 0.20 \end{bmatrix}, \qquad \boldsymbol{\lambda}_2 = \begin{bmatrix} 0.2121 \\ 0.095 \end{bmatrix}, \qquad \boldsymbol{\lambda}_3 = \begin{bmatrix} 0.3069 \\ 0.0003 \end{bmatrix}. \tag{4.62}$$

Their position in $\mathcal{D}$ *is shown in Figure 4.1.* $\mathbf{B}_1$ *is the closest to* $\left(\frac{1}{4}, \frac{1}{4}\right)$*, the point corresponding to the maximum entropy case, while* $\mathbf{B}_2$ *and* $\mathbf{B}_3$ *are more distant because*

of the increasing correlation between e_1 and e_2 (which are independent in the maximum entropy case). The correlation coefficients for $\mathbf{B}_1$, $\mathbf{B}_2$ and $\mathbf{B}_3$ are

$$\mathsf{COR}_{\mathbf{B}_1}(E_1, E_2) = 0.1666, \quad \mathsf{COR}_{\mathbf{B}_2}(E_1, E_2) = -0.2303, \quad \mathsf{COR}_{\mathbf{B}_3}(E_1, E_2) = 0.9978 \tag{4.63}$$

and account for the increasing difference between the eigenvalues of each covariance matrix. In fact Σ_3 is nearly singular because of the very strong linear relationship between e_1 and e_2, and is therefore very close to one of the axes delimiting the first quadrant.

If we denote with

$$E_{00} = \{\varnothing\}, \qquad E_{01} = \{e_2\}, \qquad E_{10} = \{e_1\}, \qquad \text{and} \qquad E_{11} = \{e_1, e_2\} \tag{4.64}$$

all possible edge sets and with m_{00}, m_{01}, m_{10} and m_{11} the respective absolute frequencies in the bootstrapped networks, for $\mathbf{B}_1$ we have

$$m_{00} = 5, \qquad m_{01} = 5, \qquad m_{10} = 5 \qquad \text{and} \qquad m_{11} = 10. \tag{4.65}$$

This is indeed pretty close to a uniform distribution over the space of the graphs (which would require an absolute frequency of 6.25 for each possible edge set). The probability of both e_1 and e_2 is 0.6 and the variance is 0.24, which are again similar to the reference values for the maximum entropy case.

As for $\mathbf{B}_2$, we have

$$m_{00} = 0, \qquad m_{01} = 3, \qquad m_{10} = 7 \qquad \text{and} \qquad m_{11} = 15. \tag{4.66}$$

The distribution of the absolute frequencies presents significant differences from a uniform distribution. The probabilities of e_1 and e_2 are respectively 0.88 and 0.72, and their variances are 0.1056 and 0.2016. Considering also the correlation between e_1 and e_2, it is intuitively clear why Σ_2 is not as close as Σ_1 to $\left(\frac{1}{4}, \frac{1}{4}\right)$. This also true for $\mathbf{B}_3$, which has the same marginal distributions for e_1 and e_2 as $\mathbf{B}_2$ but with a much stronger correlation.

4.4.2 Directed Acyclic Graphs

The behaviour of the multivariate Trinomial distribution in the *minimum* and *intermediate entropy* cases is similar to the one of the multivariate Bernoulli in many

aspects, but presents profound differences in the *maximum entropy* case. The reason for these differences is that the structure of a Bayesian network is assumed to be acyclic. Therefore the state of each arc (i.e. whether is present in the graph and its direction) is influenced by the state of all other possible arcs even in the *maximum entropy* case, when otherwise they would be independent (this is trivial to prove by adapting Theorem 4.9). Furthermore, the acyclicity constraint cannot be written in closed form, which makes the derivation of exact results on the moments of the distribution particularly difficult.

To obtain some simple expressions for the expected value and the covariance matrix we will first prove a simple theorem on directed acyclic graphs, which essentially states that if we reverse the direction of every arc the resulting graph is still a directed acyclic graph.

Theorem 4.10. *Let $G = (\mathbf{V}, A)$ be a directed acyclic graph, and let $G^* = (\mathbf{V}, A^*)$ another directed graph such that*

$$\overrightarrow{a_{ij}} \in A^* \iff \overleftarrow{a_{ij}} \in A \qquad \text{and} \qquad \overleftarrow{a_{ij}} \in A^* \iff \overrightarrow{a_{ij}} \in A \tag{4.67}$$

for every $a_{ij} \in A$. Then G^* is also acyclic.

Proof. Let's assume by contradiction that G^* is cyclic; this implies that there are one or more nodes $v_i \in \mathbf{V}$ such that

$$v_i \xrightarrow{\overrightarrow{a_{ij}}} v_j \to \ldots \to v_k \xrightarrow{\overrightarrow{a_{ki}}} v_i \tag{4.68}$$

for some $v_j, v_k \in \mathbf{V}$. However, this would mean that in G we would have

$$v_i \xrightarrow{\overleftarrow{a_{ki}}} v_k \to \ldots \to v_j \xrightarrow{\overleftarrow{a_{ij}}} v_i \tag{4.69}$$

which is not possible since G is assumed to be acyclic. □

An immediate consequence of this theorem is that for every directed acyclic graph including the arc $\overrightarrow{a_{ij}}$ there is another directed acyclic graph including the arc $\overleftarrow{a_{ij}}$. Since in the *maximum entropy* case all directed acyclic graphs have the same probability, this implies that both directions of every arc have the same probability,

$$\overrightarrow{p_{ij}} = \overleftarrow{p_{ij}} \qquad \text{for every possible } a_{ij}, i \neq j. \tag{4.70}$$

Then the expected value of each marginal Trinomial distribution is equal to

$$\mathsf{E}(A_{ij}) = \overrightarrow{p_{ij}} - \overleftarrow{p_{ij}} = 0 \tag{4.71}$$

and its variance is equal to

$$\mathsf{VAR}(A_{ij}) = \overrightarrow{p_{ij}} + \overleftarrow{p_{ij}} - (\overrightarrow{p_{ij}} - \overleftarrow{p_{ij}})^2 = 2\overrightarrow{p_{ij}}. \tag{4.72}$$

The joint probabilities associated with each pair of arcs present similar symmetries. If we denote with $\mathring{a_{ij}}$ the event that arc a_{ij} is not present in the graph and consider that there is no explicit ordering among the arcs we have, with a slight abuse in notation,

$$\mathrm{P}(\overrightarrow{a_{ij}}, \overrightarrow{a_{kl}}) = \mathrm{P}(\overleftarrow{a_{ij}}, \overleftarrow{a_{kl}}), \qquad \mathrm{P}(\overrightarrow{a_{ij}}, \overleftarrow{a_{kl}}) = \mathrm{P}(\overleftarrow{a_{ij}}, \overrightarrow{a_{kl}}), \tag{4.73}$$

$$\mathrm{P}(\mathring{a_{ij}}, \overrightarrow{a_{kl}}) = \mathrm{P}(\overrightarrow{a_{ij}}, \mathring{a_{kl}}) = \mathrm{P}(\mathring{a_{ij}}, \overleftarrow{a_{kl}}) = \mathrm{P}(\overleftarrow{a_{ij}}, \mathring{a_{kl}}). \tag{4.74}$$

Then the expression for the covariance simplifies to

$$\mathsf{COV}(A_{ij}, A_{kl}) = 2\left[\mathrm{P}(\overrightarrow{a_{ij}}, \overrightarrow{a_{kl}}) - \mathrm{P}(\overrightarrow{a_{ij}}, \overleftarrow{a_{kl}})\right], \tag{4.75}$$

which can be interpreted as the difference in probability between a serial connection and a converging connection if the arcs are incident on a common node. Note that the sign of $\mathsf{COV}(A_{ij}, A_{kl})$ depends on the way the orientations of each arc are associated with 1 and -1; a simple way to obtain a consistent parameterization is to follow the topological ordering of the graph or the natural ordering of the variables (i.e. if $i \leqslant j$ then the arc incident on these nodes is taken to be A_{ij}, $\overrightarrow{a_{ij}}$ is associated with 1 and $\overleftarrow{a_{ij}}$ with -1).

These equalities drastically reduce the number of free parameters. The marginal Trinomial distribution of each arc now depends only on $\overrightarrow{p_{ij}}$, whose value can be derived from the following numerical approximation by Melançon et al. (2000).

Theorem 4.11. *The average number of arcs in a directed acyclic graph with n nodes is approximately $\frac{1}{4}n^2$.*

Theorem 4.12. *Let $G = (\mathbf{V}, A)$ be a directed acyclic graph with n nodes. Then for each possible arc $a_{ij} \in \mathbf{V} \times \mathbf{V}, i \neq j$ we have that in the maximum entropy case*

$$\overrightarrow{p_{ij}} = \overleftarrow{p_{ij}} \simeq \frac{1}{4} + \frac{1}{4(n-1)} \qquad \textit{and} \qquad \mathring{p}_{ij} \simeq \frac{1}{2} - \frac{1}{2(n-1)}. \tag{4.76}$$

Proof. Each possible arc can appear in the graph in only one direction at a time, so a directed acyclic graph can have at most $\binom{n}{2} = \frac{1}{2}n(n-1)$ arcs. Therefore

$$\overrightarrow{p_{ij}} + \overleftarrow{p_{ij}} \simeq \frac{\frac{1}{4}n^2}{\frac{1}{2}n(n-1)} = \frac{1}{2} + \frac{1}{2(n-1)}. \tag{4.77}$$

But in the *maximum entropy* case we also have that $\overrightarrow{p_{ij}} = \overleftarrow{p_{ij}}$, so

$$\overrightarrow{p_{ij}} = \overleftarrow{p_{ij}} \simeq \frac{1}{4} + \frac{1}{4(n-1)} \qquad \text{and} \qquad \mathring{p}_{ij} = 1 - 2\overrightarrow{p_{ij}} \simeq \frac{1}{2} - \frac{1}{2(n-1)}, \tag{4.78}$$

which completes the proof. □

The quality of this approximation for $\overrightarrow{p_{ij}}$ is examined in Figure 4.2 and Figure 4.3. In Figure 4.2 the values provided by Theorem 4.12 for directed acyclic graphs with 3, 4, 5, 6 and 7 nodes are compared to the corresponding true values. The latter have been computed by enumerating all possible directed acyclic graphs of that size (i.e. the whole population) and computing the relative frequency of each possible arc. In Figure 4.3 the values provided by Theorem 4.12 for graphs with 8 to 50 nodes are compared with the corresponding estimated values computed over a set of 10^9 directed acyclic graphs of the same size. The latter have been generated with uniform probability using the algorithm from Ide and Cozman (2002) as implemented in **bnlearn**.

We can clearly see that the approximate values are very close to the corresponding true (in Figure 4.2) or estimated (in Figure 4.3) ones for graphs with at least 6 nodes. This is not a significant limitation because for graphs with 3, 4 and 5 nodes the true values can be easily computed; they are provided in Appendix A along with the true values for other quantities of interest. Furthermore, it is evident both from Theorem 4.12 and from Figure 4.2 and Figure 4.3 that as the number of nodes diverges

$$\lim_{n\to\infty} \overrightarrow{p_{ij}} = \lim_{n\to\infty} \overleftarrow{p_{ij}} = \frac{1}{4} \qquad \text{and} \qquad \lim_{n\to\infty} \mathring{p}_{ij} = \frac{1}{2}. \tag{4.79}$$

If we take the absolute value of the Trinomial random variable A_{ij} associated with a_{ij}, according to Theorem 4.5 the resulting random variable is $Bi(p_{ij})$, $p_{ij} = \frac{1}{2}$, which is

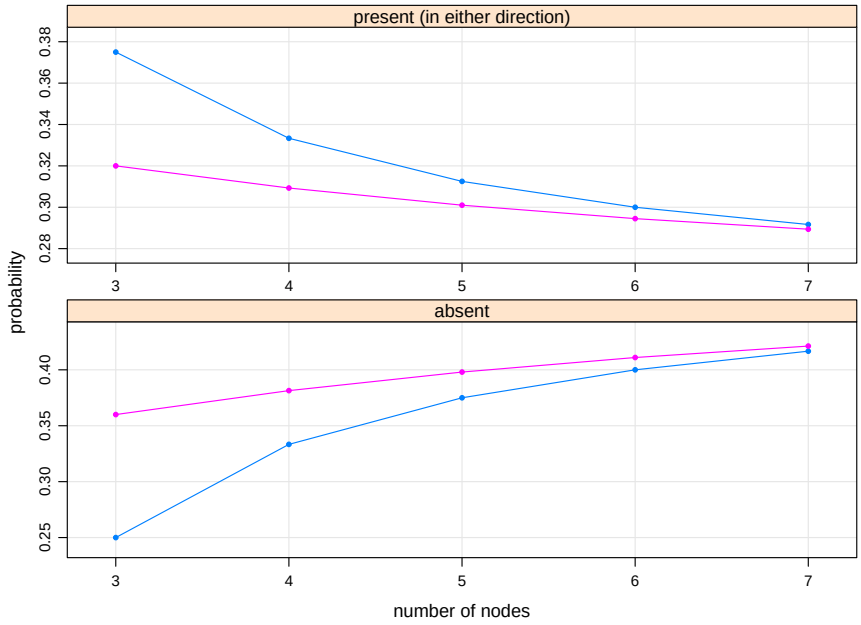


Figure 4.2: Exact (pink) and approximate (blue) probabilities of an arc being present or absent from a directed acyclic graph with 3, 4, 5, 6, and 7 nodes.

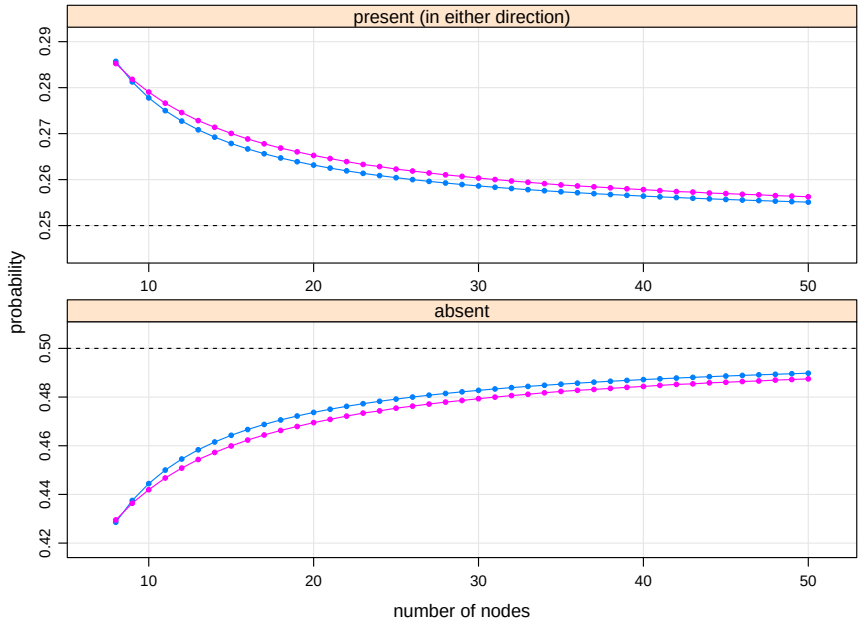


Figure 4.3: Estimated (pink) and approximate (blue) probabilities of an arc being present or absent from a directed acyclic graph with 8 to 50 nodes. The black dashed lines represent the respective limiting values.

the marginal distribution of an edge in an undirected graph in the *maximum entropy* case. The absolute value transformation can be interpreted as ignoring the direction of the arc; if we do that the marginal distribution of an arc is the same as the one of the corresponding edge for sufficiently large graphs.

No result similar to Theorem 4.11 has been proved for arbitrary pairs of arcs; therefore the structure of the covariance matrix Σ can be derived only in part. Variances can be approximated using again the probabilities provided by Theorem 4.12:

$$\mathsf{VAR}(A_{ij}) = 2\overrightarrow{p_{ij}} \simeq \frac{1}{2} + \frac{1}{2(n-1)} \to \frac{1}{2} \text{ as } n \to \infty. \tag{4.80}$$

Therefore, maximum variance (of each arc) and maximum entropy (of the network structure) are distinct, as opposed to what happens in undirected graphs. However, we can use the decomposition of the variance introduced in Equation 4.46 to motivate why this is still a "worst case" outcome of bootstrap resampling. We can see from Figure 4.5 that in the maximum entropy case the contribution of the presence of an arc (given by the transformation $|A_{ij}|$) and its direction (given by the $4\overrightarrow{p_{ij}}\overleftarrow{p_{ij}} = 4\overrightarrow{p_{ij}}^2$ term) to the variance are asymptotically equal. This is a consequence of the limits in Equation 4.79, which imply that an arc (modulo its direction) has the same probability to be present in or absent from the graph and that its directions also have the same probability. This represents the worst possible behaviour because we are not able to make any decision either about the presence of the arc or its direction.

As for the covariances, it is possible to obtain accurate bounds using *Hoeffding's identity* (Hoeffding, 1940; Fisher and Sen, 1994),

$$\mathsf{COV}(X, Y) = \iint_{\mathbb{R}^2} F_{X,Y}(x, y) - F_X(x)F_Y(y)dxdy, \tag{4.81}$$

and the decomposition of the joint distribution of dependent random variables provided by the *Farlie-Morgenstern-Gumbel* (FMG) family of distributions (Mari and Kotz, 2001), which has the form

$$F_{X,Y}(x, y) = F_X(x)F_Y(y)\left[1 + \varepsilon(1 - F_X(x))(1 - F_Y(y))\right], \qquad |\varepsilon| \leqslant 1. \tag{4.82}$$

In both cases $F_{X,Y}$, F_X and F_Y are the cumulative distribution functions of the joint and marginal distributions of X and Y.

Theorem 4.13. *Let $G = (V, A)$ be a directed acyclic graph, and let a_{ij}, $i \neq j$ and a_{kl},*

$k \neq l$ be two possible arcs in $\mathbf{V} \times \mathbf{V}$. Then in the maximum entropy case we have that

$$|\mathsf{COV}(A_{ij}, A_{kl})| \lessapprox 4 \left[\frac{3}{4} - \frac{1}{4(n-1)}\right]^2 \left[\frac{1}{4} + \frac{1}{4(n-1)}\right]^2 \tag{4.83}$$

and

$$|\mathsf{COR}(A_{ij}, A_{kl})| \lessapprox 2 \left[\frac{3}{4} - \frac{1}{4(n-1)}\right]^2 \left[\frac{1}{4} + \frac{1}{4(n-1)}\right]. \tag{4.84}$$

Proof. In the maximum entropy case both a_{ij} and a_{kl} have the same marginal distribution function,

$$F_A(a_{ij}) \simeq \begin{cases} 0 & \text{in } (-\infty, -1] \\ \dfrac{1}{4} + \dfrac{1}{4(n-1)} & \text{in } (-1, 0] \\ \dfrac{3}{4} - \dfrac{1}{4(n-1)} & \text{in } (0, 1] \\ 1 & \text{in } (1, +\infty) \end{cases}, \tag{4.85}$$

so their joint distribution can be written as a member of the Farlie-Morgenstern-Gumbel family of distribution as

$$F_{A_{ij},A_{kl}}(a_{ij}, a_{kl}) = F_A(a_{ij})F_A(a_{kl})[1 + \varepsilon(1 - F_A(a_{ij}))(1 - F_A(a_{kl}))]. \tag{4.86}$$

Then if we apply Hoeffding's identity from Equation 4.81 and replace the joint distribution function $F_{A_{ij},A_{kl}}(a_{ij}, a_{kl})$ with the right hand of Equation 4.86 we have that

$$\begin{aligned} &|\mathsf{COV}(A_{ij}, A_{kl})| = \\ &= \left| \sum_{\{-1,0,1\}} \sum_{\{-1,0,1\}} F_{A_{ij},A_{kl}}(a_{ij}, a_{kl}) - F_A(a_{ij})F_A(a_{kl}) \right| \\ &\leqslant \sum_{\{-1,0,1\}} \sum_{\{-1,0,1\}} \left| F_{A_{ij},A_{kl}}(a_{ij}, a_{kl}) - F_A(a_{ij})F_A(a_{kl}) \right| \\ &= \sum_{\{-1,0,1\}} \sum_{\{-1,0,1\}} |F_A(a_{ij})F_A(a_{kl})[1 + \varepsilon(1 - F_A(a_{ij}))(1 - F_A(a_{kl}))] - F_A(a_{ij})F_A(a_{kl})| \\ &= \sum_{\{-1,0\}} \sum_{\{-1,0\}} (1 - F_A(a_{ij}))(1 - F_A(a_{kl})). \end{aligned} \tag{4.87}$$

We can now compute the bounds for $|\mathsf{COV}(a_{ij}, a_{kl})|$ and $|\mathsf{COR}(a_{ij}, a_{kl})|$ using only the marginal distribution function F_A from Equation 4.85 and the variance from Equation 4.80, thus obtaining the expressions in Equation 4.83 and Equation 4.84. □

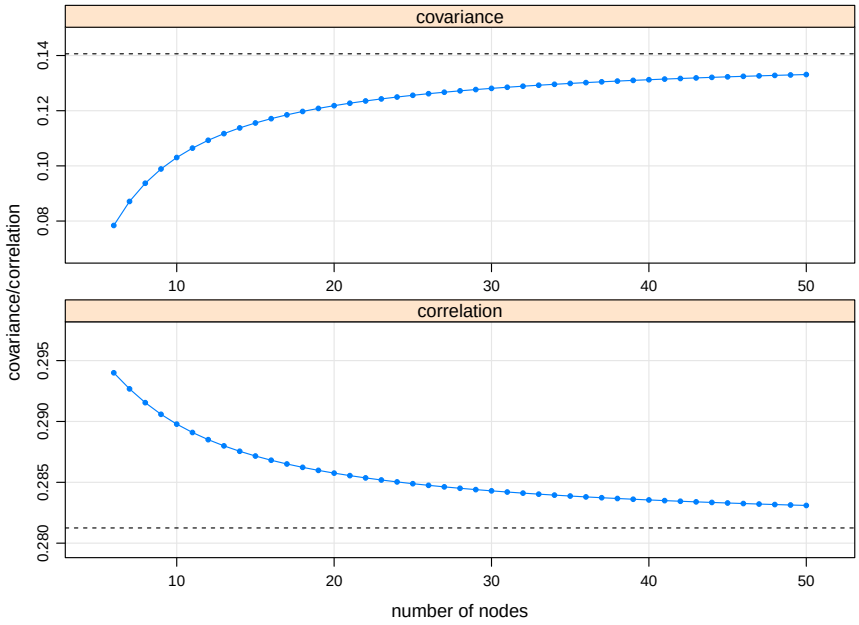


Figure 4.4: Bounds for the absolute value of the covariance and the correlation coefficient of two arcs in a directed acyclic graph with 6 to 50 nodes. The black dashed lines represent the respective limiting values.

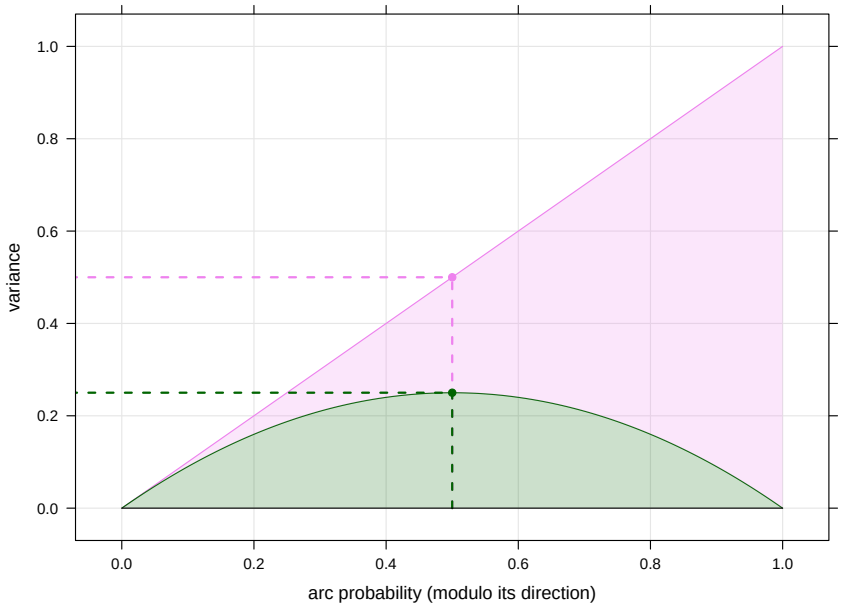


Figure 4.5: Decomposition of the asymptotic variance of an arc in the part that depends only on its presence (green) and the part that depends only on its direction (pink). The dots correspond to the respective values in the maximum entropy case.

The bounds obtained from this theorem appear quite precise in the light of the true values for the covariance and correlation coefficients (computed by enumerating all possible graphs of size 3 to 7 as it was done for $\overrightarrow{p_{ij}}$). Figure 4.4 shows the bounds for graphs with 6 to 50 nodes; for graphs with 3, 4 and 5 nodes the approximation of $\overrightarrow{p_{ij}}$ the bounds are based on is very loose, and the true values of covariance and correlation are known. Non-null covariances range from ± 0.08 (for graphs with 3 nodes) to ± 0.08410 (for graphs with 7 nodes), while non-null correlation coefficients vary from ± 0.125 (for graphs with 3 nodes) to ± 0.1423 (for graphs with 7 nodes). Both covariance and correlation appear to be strictly increasing in modulus as the number of nodes increases, and converge to the limiting values of the bounds (0.140625 and 0.28125, respectively) from below.

Some other interesting properties are apparent from true values of the covariance coefficients reported in Appendix A. They are reported below as conjectures because, while they describe a systematic behaviour that emerges from the graph sizes we have a complete enumeration for, we were not able to substantiate them with a formal proof.

Conjecture 4.1. *Arcs which are not incident on a common node are uncorrelated.*

This is a consequence of the fact that if we consider A_{ij} and A_{kl} with $i \neq j \neq k \neq l$, we have $\mathrm{P}(\overrightarrow{a_{ij}}, \overrightarrow{a_{kl}}) = \mathrm{P}(\overrightarrow{a_{ij}}, \overleftarrow{a_{kl}})$ and therefore $\mathsf{COV}(A_{ij}, A_{kl}) = 0$.

Conjecture 4.2. *The covariance matrix Σ is sparse.*

The proportion of arcs incident on a common node converges to zero as the number of nodes increases; therefore if we assume Conjecture 4.1 is true the proportion of elements of Σ that are equal to 0 has limit

$$1 \geqslant \lim_{n\to\infty} \frac{\binom{n}{2}\binom{n-2}{2}}{\binom{n}{2}\binom{n}{2} - \binom{n}{2}} \geqslant \lim_{n\to\infty} \frac{(n-2)(n-3)}{n(n-1)} = 1. \tag{4.88}$$

Furthermore, even arcs that are incident on a common node are not strongly correlated.

Conjecture 4.3. *Both covariance and correlation between two arcs incident on a common node are monotonically increasing in modulus.*

Conjecture 4.4. *The covariance between two arcs incident on a common node takes values in the interval* $[0.08,\ 0.140625]$ *in modulus, while the correlation takes values in* $[0.125, 0.28125]$ *in modulus.*

These intervals can be further reduced to $[0.08410, 0.140625]$ and $[0.1423, 0.28125]$ for graphs larger than 7 nodes due to Conjecture 4.3.

On the other hand in the *minimum entropy* case we have that

$$\mathsf{E}(A_{ij}) = \begin{cases} -1 & \text{if } \overleftarrow{a_{ij}} \in A \\ 0 & \text{if } \overleftarrow{a_{ij}}, \overrightarrow{a_{ij}} \notin A \\ 1 & \text{if } \overrightarrow{a_{ij}} \in A \end{cases} \qquad \text{and} \qquad \Sigma = \mathbf{O} \tag{4.89}$$

like in the *minimum entropy* case of undirected graphs. The *intermediate entropy* case again ranges from being very close to the *minimum entropy* case (when the graph structure displays little variability) to being very close to the *maximum entropy* case (when the graph structure displays substantial variability). The bounds on the eigenvalues of Σ derived in Theorem 4.8 allow again a graphical representation of the variability of the network structure, as it was done in Example 4.2 for the undirected graphs.

Chapter 5

Measuring the Variability of Network Structures

In this chapter we will introduce three univariate descriptive statistics of the variability of a network structure, which combine the results presented in the previous chapter and classic multivariate statistics. These statistics are easy to interpret and will be used to define some asymptotic and approximate hypothesis tests. Furthermore, we will provide some improvements in their estimation (and in the estimation of second order moments) using shrinkage regularization techniques introduced by Ledoit and Wolf (2003) and Schäfer and Strimmer (2005).

5.1 Descriptive Statistics for Undirected Graphs

Several functions have been proposed in literature as univariate measures of spread of a multivariate distribution, usually under the assumption of multivariate normality; for some examples see Muirhead (1982) and Bilodeau and Brenner (1999). Three of them in particular can be used as descriptive statistics for the multivariate Bernoulli distribution:

- the *generalized variance*,

$$\mathsf{VAR}_G(\Sigma) = \det(\Sigma). \tag{5.1}$$

- the *total variance*, called *total variation* in Mardia et al. (1979),

$$\mathsf{VAR}_T(\Sigma) = \mathrm{tr}(\Sigma). \tag{5.2}$$

- the squared *Frobenius matrix norm* of the difference between Σ and a predetermined matrix, such as

$$\mathsf{VAR}_N(\Sigma) = |||\Sigma - \frac{k}{4} I_k|||_F^2. \tag{5.3}$$

Both generalized variance and total variance associate high values of the statistic to unstable network structures, and are bounded due to the properties of the covariance matrix Σ of the multivariate Bernoulli. For total variance it is easy to show that

$$0 \leqslant \mathsf{VAR}_T(\Sigma) = \sum_{i=1}^{k} \sigma_{ii} = \sum_{i=1}^{k} \lambda_i \leqslant \frac{k}{4} \tag{5.4}$$

due to the bounds on the variances σ_{ii} from Equation 4.23 and the corresponding bounds on the eigenvalues λ_i from Theorem 4.4. Generalized variance is similarly bounded due to Hadamard's theorem on the determinant of a non-negative definite matrix (Seber, 2008):

$$0 \leqslant \mathsf{VAR}_G(\Sigma) = \prod_{i=1}^{k} \lambda_i \leqslant \prod_{i=1}^{k} \sigma_{ii} \leqslant \left(\frac{1}{4}\right)^k. \tag{5.5}$$

They reach the respective maxima in the *maximum entropy* case and are equal to zero only in the *minimum entropy* case. Generalized variance is also strictly convex (the maximum is reached only for $\Sigma = \frac{1}{4} I_k$), but it is equal to zero when Σ is rank deficient. For this reason it may be convenient to reduce Σ to a smaller, full rank matrix (say Σ^*) and consider $\mathsf{VAR}_G(\Sigma^*)$ instead of $\mathsf{VAR}_G(\Sigma)$; using the regularized estimator for Σ presented in Section 5.3 is also a viable option.

The squared Frobenius matrix norm from Equation 5.3, on the other hand, associates high values of the statistic to stable network structures. It can be rewritten in terms of the eigenvalues of Σ as

$$\mathsf{VAR}_N(\Sigma) = \sum_{i=1}^{k} \left(\lambda_i - \frac{k}{4}\right)^2. \tag{5.6}$$

It has a unique maximum (in the *minimum entropy* case), which can be computed as the solution of the constrained maximization problem in $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_k]^T$

$$\max_{\boldsymbol{\lambda} \in \mathcal{D}} f(\boldsymbol{\lambda}) = \sum_{i=1}^{k} \left(\lambda_i - \frac{k}{4}\right)^2 \qquad \text{subject to} \qquad \lambda_i \geqslant 0, \sum_{i=1}^{k} \lambda_i \leqslant \frac{k}{4} \tag{5.7}$$

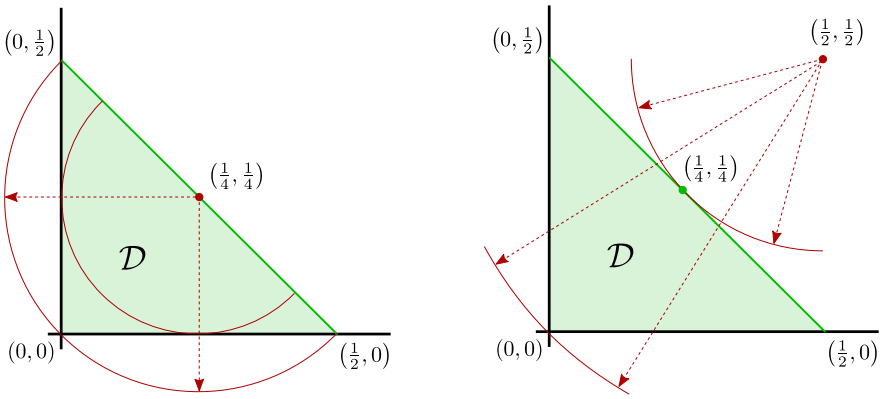


Figure 5.1: Squared Frobenius matrix norms computed using $\frac{1}{4}I_K$ (on the left) and $\frac{k}{4}I_k$ (on the right) in $\mathcal{D}$ for $k = 2$. The green area is the set $\mathcal{D}$ of the possible eigenvalues of Σ and the red lines are level curves.

using the extended Lagrange multipliers methods. Furthermore, it has a single minimum in $\boldsymbol{\lambda}^* = [\frac{1}{4}, \dots, \frac{1}{4}]$, which is the projection of $[\frac{k}{4}, \dots, \frac{k}{4}]$ onto the set $\mathcal{D}$ and coincides with the *maximum entropy* case.

The use of $\frac{k}{4}I_k$ instead of $\frac{1}{4}I_k$ (the covariance matrix arising from the *maximum entropy* case) is motivated by the need of keeping the interpretation of $\mathsf{VAR}_N(\Sigma)$ as clear as possible. The squared Frobenius matrix norm of the difference between Σ and $\frac{1}{4}I_k$ has a varying number of global maxima depending on the number k of edges considered in the analysis. They are the solutions of the constrained maximization problem

$$\max_{\boldsymbol{\lambda} \in \mathcal{D}} f(\boldsymbol{\lambda}) = \sum_{i=1}^{k} \left(\lambda_i - \frac{1}{4}\right)^2 \qquad \text{subject to} \qquad \lambda_i \geqslant 0, \sum_{i=1}^{k} \lambda_i \leqslant \frac{k}{4}. \tag{5.8}$$

This configuration of global maxima is not a significant problem for the results based on the asymptotic distribution of the multivariate Bernoulli distribution, but prevents any direct interpretation of quantities based on this difference in matrix norm as descriptive statistics. On the other hand, the difference in squared Frobenius matrix norm

$$\mathsf{VAR}_N(\Sigma) = |||\Sigma - \frac{k}{4}I_k|||_F^2 = \sum_{i=1}^{k} \left(\lambda_i - \frac{k}{4}\right)^2 \tag{5.9}$$

has both a unique global minimum (because it is a convex function),

$$\min_{\mathcal{D}} \mathsf{VAR}_N(\Sigma) = \mathsf{VAR}_N\left(\frac{1}{4}I_k\right) = \sum_{i=1}^{k}\left(\frac{1}{4} - \frac{k}{4}\right)^2 = \frac{k(k-1)^2}{16}, \tag{5.10}$$

and a unique global maximum,

$$\max_{\mathcal{D}} \mathsf{VAR}_N(\Sigma) = \mathsf{VAR}_N(\mathbf{O}) = \sum_{i=1}^{k}\left(\frac{k}{4}\right)^2 = \frac{k^3}{16}, \tag{5.11}$$

which correspond to the *minimum entropy* ($\boldsymbol{\lambda} = [0, \ldots, 0]$) and the *maximum entropy* ($\boldsymbol{\lambda} = [\frac{1}{4}, \ldots, \frac{1}{4}]$) covariance matrices respectively (see Figure 5.1).

All the descriptive statistics introduced in this section can be normalized as follows:

$$\begin{aligned}
\overline{\mathsf{VAR}}_T(\Sigma) &= \frac{\mathsf{VAR}_T(\Sigma)}{\max_\Sigma \mathsf{VAR}_T(\Sigma)} = \frac{4\mathsf{VAR}_T(\Sigma)}{k}, \\
\overline{\mathsf{VAR}}_G(\Sigma) &= \frac{\mathsf{VAR}_G(\Sigma)}{\max_\Sigma \mathsf{VAR}_G(\Sigma)} = 4^k\mathsf{VAR}_G(\Sigma), \\
\overline{\mathsf{VAR}}_N(\Sigma) &= \frac{\max_\Sigma \mathsf{VAR}_N(\Sigma) - \mathsf{VAR}_N(\Sigma)}{\max_\Sigma \mathsf{VAR}_N(\Sigma) - \min_\Sigma \mathsf{VAR}_N(\Sigma)} = \frac{k^3 - 16\mathsf{VAR}_N(\Sigma)}{k(2k-1)}.
\end{aligned}$$

All of these normalized statistics vary in the $[0, 1]$ interval and associate high values to networks whose structure display a high variability across the bootstrap samples. Equivalently, we can define

$$\begin{aligned}
\overline{\overline{\mathsf{VAR}}}_T(\Sigma) &= 1 - \overline{\mathsf{VAR}}_T(\Sigma), \\
\overline{\overline{\mathsf{VAR}}}_G(\Sigma) &= 1 - \overline{\mathsf{VAR}}_G(\Sigma), \\
\overline{\overline{\mathsf{VAR}}}_N(\Sigma) &= 1 - \overline{\mathsf{VAR}}_N(\Sigma)
\end{aligned}$$

which associate high values to networks with little variability, and can be used as measures of the difference in behaviour from the *maximum entropy* case.

5.2 Hypothesis Tests for Undirected Graphs

5.2.1 Asymptotic Inference

The limiting distribution of the descriptive statistics defined in the previous section can be derived by replacing the covariance matrix Σ with its (unbiased) maximum likelihood estimator $\hat{\Sigma}$ and by considering the multivariate Gaussian distribution from

Equation 4.32. One of the simplest hypotheses we may be interested in is

$$H_0 : \Sigma = \frac{1}{4} I_k \qquad\qquad H_1 : \Sigma \neq \frac{1}{4} I_k, \tag{5.12}$$

which relates the sample covariance matrix with the one from the *maximum entropy* case. Such a test answers the question whether the network structure learned from the data is random noise or it encodes real, systematic dependence relationships. Testing such an hypothesis may be useful, for example, to check whether the sample is large enough to reliably learn the network structure. Furthermore, the null distribution is extremely regular; as shown in Section 4.4.1, the multivariate Bernoulli distribution in the *maximum entropy* case has independent marginals distributed as $Ber(\frac{1}{2})$. This makes the use of the asymptotic multivariate Gaussian distribution feasible, and convergence as fast as it can be in a high-dimensional setting.

For total variance we have from Muirhead (1982) that

$$t_T = 4m \operatorname{tr}(\hat{\Sigma}) \dot{\sim} \chi^2_{mk}, \tag{5.13}$$

and since the maximum value of $\operatorname{tr}(\Sigma)$ is achieved in the *maximum entropy* case, the hypothesis in Equation 5.12 assumes the form

$$H_0 : \operatorname{tr}(\Sigma) = \frac{k}{4} \qquad\qquad H_1 : \operatorname{tr}(\Sigma) < \frac{k}{4}. \tag{5.14}$$

Then the observed significance value is

$$\hat{\alpha}_T = \mathrm{P}(t_T \leqslant t_T^{oss}), \tag{5.15}$$

and can be improved with the finite sample correction

$$\tilde{\alpha}_T = \mathrm{P}\left(t_T \leqslant t_T^{oss} \,|\, t_T \in [0, mk]\right) = \frac{\mathrm{P}(t_T \leqslant t_T^{oss})}{\mathrm{P}(t_T \leqslant mk)}, \tag{5.16}$$

which accounts for the bounds on $\mathsf{VAR}_T(\Sigma)$ from Equation 5.4.

As for generalized variance, there are several possible asymptotic and approximate test statistics with different distributions:

- the Gaussian test statistic defined in Anderson (2003),

$$t_{G_1} = \sqrt{m} \left(\frac{\det(\hat{\Sigma})}{\det(\frac{1}{4} I_k)} - 1 \right) \dot{\sim} N(0, 2k); \tag{5.17}$$

- the Gamma test statistic defined in Steyn (1978),

$$t_{G_2} = \frac{mk}{2}\sqrt[k]{\frac{\det(\hat{\Sigma})}{\det(\frac{1}{4}I_k)}} \dot{\sim} Ga\left(\frac{k(m+1-k)}{2}, 1\right); \tag{5.18}$$

- the saddlepoint approximation defined in Butler et al. (1992).

As before, the hypothesis in Equation 5.12 assumes the form

$$H_0 : \det(\Sigma) = \det\left(\frac{1}{4}I_k\right) \qquad H_1 : \det(\Sigma) < \det\left(\frac{1}{4}I_k\right). \tag{5.19}$$

The observed significance values for the Gaussian and Gamma distributions are

$$\hat{\alpha}_{G_1} = \mathrm{P}(t_{G_1} \leqslant t_{G_1}^{oss}) \qquad \text{and} \qquad \hat{\alpha}_{G_2} = \mathrm{P}(t_{G_2} \leqslant t_{G_2}^{oss}), \tag{5.20}$$

and the respective finite sample corrections for the bounds on $\mathsf{VAR}_N(\Sigma)$ are

$$\tilde{\alpha}_{G_1} = \mathrm{P}\left(t_{G_1} \leqslant t_{G_1}^{oss} \,|\, t_{G_1} \in \left[-\sqrt{m}, 0\right]\right) = \frac{\mathrm{P}(t_{G_1} \leqslant t_{G_1}^{oss}) - \mathrm{P}(t_{G_1} \leqslant -\sqrt{m})}{\mathrm{P}(t_{G_1} \leqslant 0) - \mathrm{P}(t_{G_1} \leqslant -\sqrt{m})}, \tag{5.21}$$

$$\tilde{\alpha}_{G_2} = \mathrm{P}\left(t_{G_2} \leqslant t_{G_2}^{oss} \,|\, t_{G_2} \in \left[0, \frac{mk}{2}\right]\right) = \frac{\mathrm{P}(t_{G_2} \leqslant t_{G_2}^{oss})}{\mathrm{P}(t_{G_2} \leqslant \frac{mk}{2})}. \tag{5.22}$$

The test statistic associated with the squared Frobenius matrix norm is the test for the equality of two covariance matrices defined in Nagao (1973),

$$t_N = \frac{m}{2}\,\mathrm{tr}\left(\left[\hat{\Sigma}\left(\frac{1}{4}I_k\right)^{-1} - I_k\right]^2\right) = \frac{m}{2}\,\mathrm{tr}\left(\left[4\hat{\Sigma} - I_k\right]^2\right) \dot{\sim} \chi^2_{\frac{1}{2}k(k+1)}, \tag{5.23}$$

because

$$\begin{aligned}\mathrm{tr}\left(\left[4\hat{\Sigma} - I_k\right]^2\right) &= 16\,\mathrm{tr}\left(\left[U\Lambda U^H - \frac{1}{4}I_k\right]\left[U\Lambda U^H - \frac{1}{4}I_k\right]\right) = \\ &= 16\,\mathrm{tr}\left(U\left[\Lambda - \frac{1}{4}I_k\right]U^H U\left[\Lambda - \frac{1}{4}I_k\right]U^H\right) = 16\,\mathrm{tr}\left(\left[\Lambda - \frac{1}{4}I_k\right]^2\right) = \\ &= 16\sum_{i=1}^{k}\left(\lambda_i - \frac{1}{4}\right)^2 = 16|||\hat{\Sigma} - \frac{1}{4}I_k|||_F^2,\end{aligned} \tag{5.24}$$

where $U\Lambda U^H$ is the spectral decomposition of $\hat{\Sigma}$. Note that the matrix $\frac{k}{4}I_k$ used in $\mathsf{VAR}_N(\Sigma)$ is not a valid covariance matrix for a multivariate Bernoulli distribution (the

$t_T(\Sigma)$					
	10	20	50	100	200
Σ_1	0.491137	0.457610	0.405404	0.354943	0.291243
	0.906041	**0.863836**	**0.781414**	**0.691495**	**0.571734**
Σ_2	0.094193	0.026330	0.000852	0.000003	0.000000
	0.173766	**0.049704**	**0.001644**	**0.000007**	**0.000000**
Σ_3	0.094193	0.026330	0.000852	0.000003	0.000000
	0.173766	**0.049704**	**0.001644**	**0.000007**	**0.000000**
$t_{G_2}(\Sigma)$					
Σ_1	0.603944	0.524258	0.423183	0.341131	0.250054
	0.905218	**0.847522**	**0.735799**	**0.616696**	**0.465129**
Σ_2	0.121488	0.023514	0.000278	0.000000	0.000000
	0.182091	**0.0380138**	**0.000484**	**0.000000**	**0.000000**
Σ_3	0.000000	0.000000	0.000000	0.000000	0.000000
	0.000000	**0.000000**	**0.000000**	**0.000000**	**0.000000**
$t_N(\Sigma)$					
Σ_1	0.965205	0.909123	0.714937	0.436839	0.142271
	0.964547	**0.909108**	**0.714937**	**0.436839**	**0.142271**
Σ_2	0.564938	0.253762	0.017090	0.000142	0.000000
	0.556708	**0.253636**	**0.017090**	**0.000142**	**0.000000**
Σ_3	0.154551	0.014796	0.000008	0.000000	0.000000
	0.138557	**0.014628**	**0.000008**	**0.000000**	**0.000000**

Table 5.1: Asymptotic significance values of t_T, t_{G_2} and t_N for Σ_1, Σ_2 and Σ_3; the ones computed with finite sample corrections are reported in bold.

diagonal elements of Σ are bounded in $[0, \frac{1}{4}]$); therefore it can not be used in the definition of a statistical test. $\frac{1}{4}I_k$ on the other hand is a valid covariance matrix, and allows the interpretation of t_N as a test on the distance of $\hat{\Sigma}$ from the *maximum entropy* case.

The significance value for t_N is

$$\hat{\alpha}_N = \mathrm{P}(t_N \geqslant t_N^{oss}) \tag{5.25}$$

as the hypothesis in Equation 5.12 becomes

$$H_0 : |||\Sigma - \frac{1}{4}I_k|||_F^2 = 0 \qquad\qquad H_1 : |||\Sigma - \frac{1}{4}I_k|||_F^2 > 0. \tag{5.26}$$

Unlike the previous statistics, Nagao's test is not significantly affected by the bounds on the squared Frobenius matrix norm, to the point that the finite sample sample

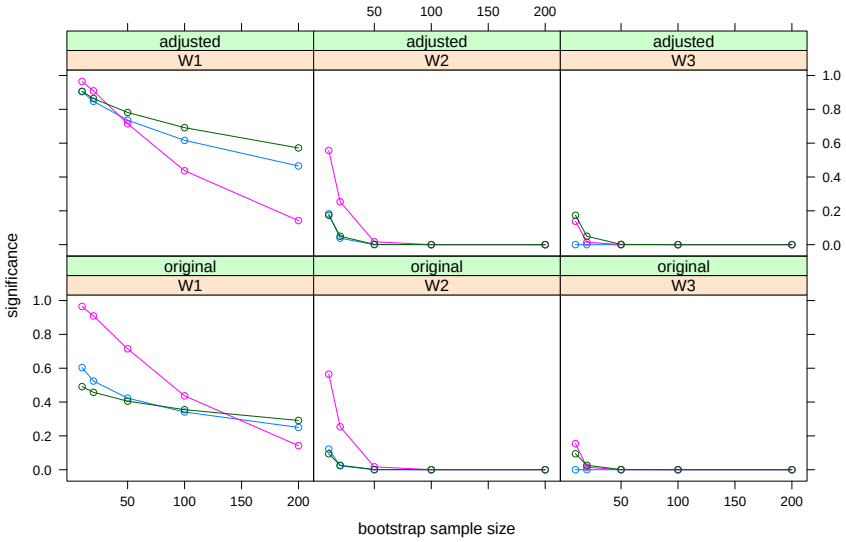


Figure 5.2: Asymptotic significance values of t_T (green), t_{G_2} (blue) and t_N (violet) for Σ_1, Σ_2 and Σ_3 from Table 5.1.

correction

$$\tilde{\alpha}_N = \mathrm{P}\left(t_N \geqslant t_N^{oss} \,|\, t_{G_1} \in [0, t_N^{max}]\right) = \frac{\mathrm{P}(t_N \geqslant t_N^{oss}) - \mathrm{P}(t_N > t_N^{max})}{\mathrm{P}(t_N \leqslant t_N^{max})} \tag{5.27}$$

is not appreciably better than the raw significance value.

Example 5.1. *Consider again the multivariate Bernoulli distributions* $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{W}_3$ *and their covariance matrices* Σ_1, Σ_2, Σ_3 *from Example 4.2. The respective asymptotic significance values for the statistics* t_T, t_{G_1} *and* t_N *are reported in Table 5.1.*

5.2.2 Monte Carlo Inference and Parametric Bootstrap

Another approach to compute the significance values of the statistics $\mathsf{VAR}_T(\Sigma)$, $\mathsf{VAR}_G(\Sigma)$ and $\mathsf{VAR}_N(\Sigma)$ for the set of hypothesis in Equation 5.12 is to apply parametric bootstrap again.

The null multivariate Bernoulli distribution has a diagonal covariance matrix, so its components are uncorrelated. According to Theorem 4.1 they are also independent, so the null distribution is completely specified by its marginals. Therefore, it is possible (and indeed quite easy) to generate observations from the null distribution and use them to estimate the significance value of the statistics $\overline{\mathsf{VAR}}_T(\Sigma)$, $\overline{\mathsf{VAR}}_G(\Sigma)$ and $\overline{\mathsf{VAR}}_N(\Sigma)$

$\overline{\overline{\mathsf{VAR}}}_T(\Sigma)$					
	10	20	50	100	200
Σ_1	0.569655	0.457109	0.129242	0.017416	0.000334
Σ_2	0.016834	0.000205	0	0	0
Σ_3	0.016834	0.000205	0	0	0
$\overline{\overline{\mathsf{VAR}}}_G(\Sigma)$					
Σ_1	0.784102	0.512839	0.14788	0.013678	0.000094
Σ_2	0.063548	0.000761	0	0	0
Σ_3	0.005909	0.000008	0	0	0
$\overline{\overline{\mathsf{VAR}}}_N(\Sigma)$					
Σ_1	0.743797	0.568819	0.239397	0.096544	0.019633
Σ_2	0.196996	0.037772	0.001018	0.000005	0
Σ_3	0.018292	0.000355	0	0	0

Table 5.2: Bootstrap significance values from parametric bootstrap for Σ_1, Σ_2 and Σ_3.

defined in Section 5.1:

1. Compute the value of test statistic T on the original covariance matrix Σ.
2. For $r = 1, 2, \ldots, R$:
   (a) generate m vectors of k random samples from a $Ber(\frac{1}{2})$ distribution;
   (b) compute their covariance matrix Σ_r^*;
   (c) compute T_r^* from Σ_r^*.
3. Compute the Monte Carlo significance value as

$$\hat{\alpha}_R = \frac{1}{R} \sum_{r=1}^{R} \mathbb{1}_{\{x \geqslant T\}}(T_r^*). \tag{5.28}$$

This approach has two important advantages over the parametric tests defined in Section 5.2.1:

- the test statistic is evaluated against the null distribution instead of its asymptotic approximation, thus removing any distortion caused by lack of convergence (which can be quite slow and problematic for large numbers of edges).
- each simulation r has a lower computational cost than the equivalent application of the structure learning algorithm to a bootstrap sample. Therefore Monte

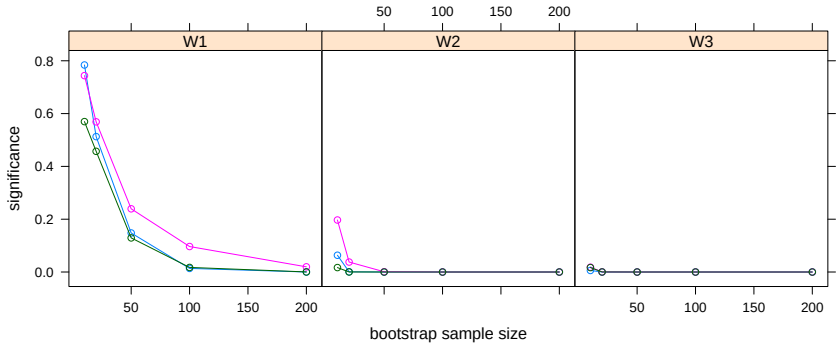


Figure 5.3: Monte Carlo significance values for the total variance (green), the generalized variance (blue) and the squared Frobenius matrix norm (violet) from Table 5.2.

Carlo tests can achieve a good precision with a smaller number of bootstrapped networks, as there is no need to approach the asymptotic null distribution. This in turn allows its application to larger problems while still maintaining a reasonable performance.

Example 5.2. *Consider the multivariate Bernoulli distributions* $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{W}_3$ *from Example 4.2 and Example 5.1 one last time. The corresponding significance values for the three normalized statistics* $\overline{\overline{\mathsf{VAR}}}_T(\Sigma)$, $\overline{\overline{\mathsf{VAR}}}_G(\Sigma)$ *and* $\overline{\overline{\mathsf{VAR}}}_N(\Sigma)$ *are reported in Table 5.2 for various sizes of the bootstrap samples* ($m = 10, 20, 50, 100, 200$). *Each one have been computed from* $R = 10^6$ *covariance matrices generated from the null distribution.*

5.3 Regularized Estimators and Statistics for Undirected Graphs

The analysis of the covariance matrix of a network structure is a high-dimensional problem in all but the most trivial cases. If a graph has n nodes, there are $\frac{1}{2}n(n-1)$ possible edges; so the number of edges we may be interested in grows quadratically in the number of nodes, which is itself large in many practical applications of graphical models. In this setting, the maximum likelihood estimator $\hat{\Sigma}$ of the covariance matrix is known to be inefficient and displays a considerable instability for most reasonable, finite sample sizes. All the optimality result concerning $\hat{\Sigma}$ are asymptotic in nature, and therefore they is no guarantee that they hold for finite samples.

These issues can be explained as a consequence of the inadmissibility of the maximum

likelihood estimator for the mean of multivariate distributions discovered by Stein (1956) and investigated by James and Stein (1961). A solution is provided in the form of a *regularized estimator* $\tilde{\Sigma}$, which includes some bias in order to increase the overall performance of the estimator. This is achieved by defining $\tilde{\Sigma}$ as a linear combination of the maximum likelihood estimator $\hat{\Sigma}$ and a *target distribution* with covariance matrix T, which is usually chosen to be much more regular:

$$\tilde{\Sigma} = \lambda T + (1 - \lambda)\hat{\Sigma}, \qquad \lambda \in [0, 1]. \tag{5.29}$$

Such an estimator is called a *shrinkage estimator*, because $\hat{\Sigma}$ is shrunk towards T in the space of the covariance matrices; λ is likewise called the *shrinkage coefficient*.

A closed-form estimator for λ has been derived by Ledoit and Wolf (2003) as the value that minimizes the mean squared error of $\tilde{\Sigma}$; it has the form

$$\lambda^* = \frac{\sum_{i=1}^{k}\sum_{j=1}^{k} \mathsf{VAR}(\hat{\sigma}_{ij}) - \mathsf{COV}(\hat{\sigma}_{ij}, t_{ij}) - \mathsf{Bias}(\hat{\sigma}_{ij})(t_{ij} - \hat{\sigma}_{ij})}{\sum_{i=1}^{k}\sum_{j=1}^{k}(t_{ij} - \hat{\sigma}_{ij})^2} \tag{5.30}$$

where $T = [t_{ij}]$, $i, j = 1, \ldots, k$. Since the estimators for the variances and the covariances are unbiased, this expression reduces to

$$\lambda^* = \frac{\sum_{i=1}^{k}\sum_{j=1}^{k} \mathsf{VAR}(\hat{\sigma}_{ij}) - \mathsf{COV}(\hat{\sigma}_{ij}, t_{ij})}{\sum_{i=1}^{k}\sum_{j=1}^{k}(t_{ij} - \hat{\sigma}_{ij})^2}. \tag{5.31}$$

The resulting estimator $\tilde{\Sigma}$ it is always positive definite (as long as T is) and dominates $\hat{\Sigma}$ in terms of mean square error. Furthermore, the good properties of λ^* do not require any distributional assumption beyond the existence of the quantities in the right hand of Equation 5.30.

Nonetheless, the form of the covariance matrix T must be chosen with some care according to the nature of the problem at hand. Two sensible choices for a multivariate Bernoulli distribution are the *maximum entropy* covariance matrix $\frac{1}{4}I_k$ and the diagonal matrix used in Schäfer and Strimmer (2005). The latter is called the "diagonal, unequal variance" target, and is defined as the diagonal matrix built from $\hat{\Sigma}$ by setting all non-diagonal elements to zero. We will denote it as $\mathrm{diag}(\hat{\Sigma})$. Both these choices are positive definite and have a very simple structure. Furthermore, they are valid covariance matrices for a multivariate Bernoulli, and always result in valid shrunk covariance matrices due to the convexity of the region $\mathcal{D}$ introduced in Section 4.2.2 (see Figure 5.4).

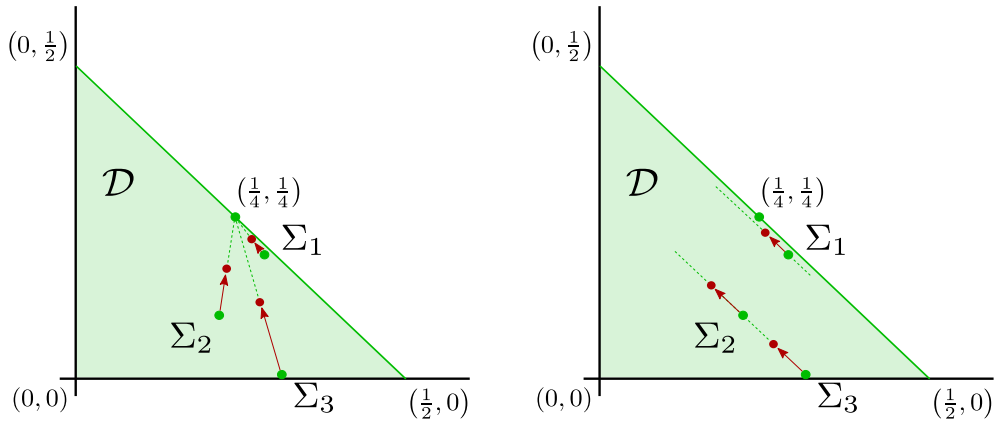


Figure 5.4: Changes in the eigenvalues of the covariance matrices Σ_1, Σ_2 and Σ_3 after shrinking towards the *maximum entropy* covariance matrix (on the left) and towards the "diagonal, unequal variance" target (on the right).

In the first case, the shrunk covariance matrix $\tilde{\Sigma} = [\tilde{\sigma}_{ij}]$ assumes the form

$$\tilde{\sigma}_{ij} = \begin{cases} \frac{1}{4}\lambda^* + (1-\lambda^*)(\hat{p}_i - \hat{p}_i^2) & \text{if } i = j \\ (1-\lambda^*)(\hat{p}_{ij} - \hat{p}_i\hat{p}_j) & \text{if } i \neq j \end{cases}. \tag{5.32}$$

If we denote the eigenvalues of $\hat{\Sigma}$ with s_i and those of $\tilde{\Sigma}$ with $\tilde{s}_i$ (to avoid confusion with the shrinkage coefficient), we can rewrite the descriptive statistics introduced in Section 5.1 as:

$$\mathsf{VAR}_T(\tilde{\Sigma}) = \operatorname{tr}\left(\lambda^*\frac{1}{4}I_k + (1-\lambda^*)\hat{\Sigma}\right) = \sum_{i=1}^{k}\left[\frac{1}{4}\lambda^* + (1-\lambda^*)s_i\right] = \sum_{i=1}^{k}\tilde{s}_i, \tag{5.33}$$

$$\mathsf{VAR}_G(\tilde{\Sigma}) = \det\left(\lambda^*\frac{1}{4}I_k + (1-\lambda^*)\hat{\Sigma}\right) = \prod_{i=1}^{k}\left[\frac{1}{4}\lambda^* + (1-\lambda^*)s_i\right] = \prod_{i=1}^{k}\tilde{s}_i, \tag{5.34}$$

$$\mathsf{VAR}_N(\tilde{\Sigma}) = |||(1-\lambda^*)\hat{\Sigma} + \frac{k-\lambda^*}{4}I_k|||_F^2 = (1-\lambda^*)^2\sum_{i=1}^{k}\left(s_i - \frac{1}{4}\cdot\frac{k-\lambda^*}{1-\lambda^*}\right)^2, \tag{5.35}$$

where $\tilde{s}_i = \frac{1}{4}\lambda^* + (1-\lambda^*)s_i$. Hypothesis tests can be similarly rewritten. The shrinkage coefficient can be estimated as

$$\lambda^* = \frac{\sum_{i=1}^{k}\sum_{j\neq i}\mathsf{VAR}(\hat{p}_{ij} - \hat{p}_i\hat{p}_j) + \sum_{i=1}^{k}\mathsf{VAR}(\hat{p}_i - \hat{p}_i^2)}{\sum_{i=1}^{k}\sum_{j\neq i}(\hat{p}_{ij} - \hat{p}_i\hat{p}_j)^2 + \sum_{i=1}^{k}(\hat{p}_i - \hat{p}_i^2 - \frac{1}{4})^2}, \tag{5.36}$$

and results in a closed-form expression which is easily computable from the reduced parameter collection $\tilde{\mathbf{p}}$. The required algebraic computations are straightforward but tedious, and are therefore detailed in Appendix B.

If we choose the "diagonal, unequal variance" target instead, the shrunk covariance matrix assumes the form

$$\tilde{\sigma}_{ij} = \begin{cases} \hat{p}_i - \hat{p}_i^2 & \text{if } i = j \\ (1 - \lambda^*)(\hat{p}_{ij} - \hat{p}_i\hat{p}_j) & \text{if } i \neq j \end{cases}, \tag{5.37}$$

and the estimator for the shrinkage coefficient reduces to

$$\lambda^* = \frac{\sum_{i=1}^{k}\sum_{j\neq i}\mathsf{VAR}(\hat{p}_{ij} - \hat{p}_i\hat{p}_j) + 2\sum_{i=1}^{k}\mathsf{VAR}(\hat{p}_i - \hat{p}_i^2)}{\sum_{i=1}^{k}\sum_{j\neq i}(\hat{p}_{ij} - \hat{p}_i\hat{p}_j)^2}. \tag{5.38}$$

The derivation and the closed-form expression of λ^* are again reported in Appendix B. The effects of the shrinkage on $\mathsf{VAR}_T(\tilde{\Sigma})$, $\mathsf{VAR}_G(\tilde{\Sigma})$ and $\mathsf{VAR}_N(\tilde{\Sigma})$ are not as easily expressed as for the previous target, with the notable exception of the total variance:

$$\mathsf{VAR}_T(\tilde{\Sigma}) = \mathsf{VAR}_T(\hat{\Sigma}), \tag{5.39}$$

$$\mathsf{VAR}_G(\tilde{\Sigma}) = \det\left(\lambda^* \operatorname{diag}(\hat{\Sigma}) + (1 - \lambda^*)\hat{\Sigma}\right), \tag{5.40}$$

$$\mathsf{VAR}_N(\tilde{\Sigma}) = |||(1 - \lambda^*)\hat{\Sigma} + \lambda^* \operatorname{diag}(\hat{\Sigma}) + \frac{k}{4}I_k|||_F^2. \tag{5.41}$$

5.4 Measures of Variability for Directed Acyclic Graphs

Most of the results presented in the sections above can be derived in a similar way for directed acyclic graphs. However, the behaviour of the multivariate Trinomial distribution in the *maximum entropy* case and the properties of Bayesian network structure learning algorithms make the interpretation of both descriptive statistics and hypothesis tests particularly difficult.

Both total variance and generalized variance are again bounded due to the inequalities in Equation 4.49; if we are considering k arcs then

$$0 \leqslant \mathsf{VAR}_T(\Sigma) \leqslant k \qquad \text{and} \qquad 0 \leqslant \mathsf{VAR}_G(\Sigma) \leqslant 1. \tag{5.42}$$

Therefore, we can define their normalized transforms as

$$\overline{\mathsf{VAR}}_T(\Sigma) = \frac{1}{k}\mathsf{VAR}_T(\Sigma) \qquad \text{and} \qquad \overline{\mathsf{VAR}}_G(\Sigma) = \mathsf{VAR}_G(\Sigma). \tag{5.43}$$

They reach the respective maxima for $\Sigma = I_k$, that is, when all marginal variances are at their maximum and arcs are uncorrelated. Generalized variance is again strictly convex (I_k is the only global maximum), while total variance reaches its maximum for any covariance matrix with diagonal elements equal to 1 regardless of the correlation structure between the arcs. Neither $\mathsf{VAR}_T(\Sigma)$ nor $\mathsf{VAR}_G(\Sigma)$ reach the respective maxima in the *maximum entropy* case, because the covariance matrix is very different from I_k (see Section 4.4.2); in that case we have that $\mathsf{VAR}_T(\Sigma) \simeq \frac{k}{2}$ and $\mathsf{VAR}_G(\Sigma) \lessapprox \frac{1}{2^k}$ due to Hadamard's theorem.

This makes the interpretation of both $\mathsf{VAR}_T(\Sigma)$ and $\mathsf{VAR}_G(\Sigma)$ problematic, because the *minimum entropy* case coincides with their lower bound but the *maximum entropy* case is not related to their upper bound. Even though it is possible to define a transformation of these statistics such that interpretation is again straightforward, it is not clear how such a transformation should be chosen (in terms of smoothness, continuity, etc.). Furthermore, the fact that arcs do not display their maximum (marginal) variability in the *maximum entropy* case raises the question of whether we should be more interested in a *maximum variance* case instead.

It is important to note that $\mathsf{VAR}_T(\Sigma)$ and $\mathsf{VAR}_G(\Sigma)$ can still be interpreted as spread measures in the usual way, since the *minimum entropy* case is also a *minimum variance* case (i.e. $\Sigma = \mathbf{O}$). However, this fact is not useful in the context of this thesis. The reason is that in the *maximum variance* case all arcs appear in all the bootstrapped networks, so $\mathsf{VAR}_T(\Sigma)$ and $\mathsf{VAR}_G(\Sigma)$ provide a good measure of the sparseness of the network structure but not of its variability. In other words, they associate high values of the statistic to dense networks, which are not necessarily the ones displaying the highest variability.

The squared Frobenius matrix norm is again defined using the covariance matrix associated with the *maximum entropy* case, denoted here as Σ_{maxent}, as follows:

$$\mathsf{VAR}_N(\Sigma) = |||\Sigma - k\Sigma_{maxent}|||_F^2. \tag{5.44}$$

The normalized transform is guaranteed to exist because the space of the eigenvalues

of Σ is closed and bounded (see Theorem 4.8); it is defined again as

$$\overline{\mathsf{VAR}}_N(\Sigma) = \frac{\max_\Sigma \mathsf{VAR}_N(\Sigma) - \mathsf{VAR}_N(\Sigma)}{\max_\Sigma \mathsf{VAR}_N(\Sigma) - \min_\Sigma \mathsf{VAR}_N(\Sigma)} \tag{5.45}$$

and associates high levels of the statistic to unstable network structures. The biggest issue of both $\mathsf{VAR}_N(\Sigma)$ and $\overline{\mathsf{VAR}}_N(\Sigma)$ is that their estimation is affected by two kinds of errors:

1. the approximations derived in Theorem 4.12, which are used in the computation of the diagonal elements of Σ_{maxent};
2. the approximate Monte Carlo estimation of the non-null covariances, which is subject to the natural instability of parameter estimation in high-dimensional settings.

Both these approximations are required due to the lack of a complete characterization Σ_{maxent}, and have the potential to degrade the quality of inferential procedures. Note that $\mathsf{VAR}_T(\Sigma)$ and $\mathsf{VAR}_G(\Sigma)$ and their normalized transforms are unaffected by this problem because they do not depend on the form of Σ_{maxent}.

Another important limitation in the interpretation of the quantities introduced in this section follows from the fact that most Bayesian network structure learning algorithms used in modern literature are score equivalent. The frequent inability of such algorithms to distinguish between the two possible directions of an arc introduces additional variability and possibly bias in the analysis. Consider for example the networks learned from the `learning.test` data set in Section 3.3.4. The direction of the arc between the nodes `A` and `B` was set to `A` $\rightarrow$ `B` by the Hill-Climbing algorithm due to its inability to deal with partially directed graphs. This choice is completely implementation-dependent, and has no statistical or probabilistic reason; `A` $\rightarrow$ `B` is preferred over `B` $\rightarrow$ `A` because of the order of the variables in the data frame the network was learned from. Even when resampling from the original data set, that order is usually preserved, often leading to completely wrong conclusions about the relative probabilities of the two directions. Such potential for bias is present in all classes of structure learning algorithms illustrated in Section 2.2, and the errors it causes are likely to propagate to neighbouring arcs due to the acyclicity constraint.

This issue raises a second, fundamental question in the analysis of the structures of directed acyclic graphs, namely whether transforming them to undirected graphs (thus ignoring the direction of the arcs) may substantially increase the reliability of the results. This can be achieved by considering either the skeleton of the graph (which

amounts to applying an absolute value transform, as detailed in Section 4.3.1 and Section 4.4.2) or the corresponding moral graph. The latter preserves all the information present in the original network structure and completely solves the problem of score equivalence, because Bayesian networks encoding the same probability distribution have the same moral graph. Furthermore, even though the link between the multivariate Trinomial and the multivariate Bernoulli is not as strong as for the skeleton, inference is still much easier because the limiting cases are completely characterized. On the other hand, using the skeleton allows a direct application of the decompositions presented in Equation 4.46 and Equation 4.47 for the variance and the covariance. The link provided by the application of the absolute value transformation may allow the derivation of further results, which may still result in a meaningful analysis despite the loss of part of the information present in the original network.

Chapter 6

Comparing Different Learning Strategies

In this chapter we will study how the use of different classes of conditional independence tests affects the performance of Bayesian network structure learning algorithms. To that end, we will consider the permutation and shrinkage tests implemented in **bnlearn** as used in the Grow-Shrink and Max-Min Hill-Climbing algorithms. The Bayesian networks learned with these tests will be studied first with the techniques usually found in literature, and then with the variability measures introduced in this thesis.

6.1 Conditional Independence Tests and Network Structures

In literature there are several studies on the performance of Bayesian network structure learning algorithms; one of the most extensive performed in recent years is presented in Tsamardinos et al. (2006). The focus of these studies is almost always the heuristics learning algorithms are based on, i.e. the maximization algorithms used in score-based algorithms or the techniques for learning the neighbourhood of each node in constraint-based algorithms. The influence of other components of the overall learning strategy, such as the conditional independence tests (and the associated type I error threshold) or the network scores (and the associated parameters, such as the equivalent sample size), is usually not investigated.

However, limiting such studies to the performance of the heuristics poses serious

doubts on their conclusions. First of all, the decisions made by the heuristics are based on the values of the statistical criteria they use to extract information from the data. Therefore, it is important to choose a conditional independence test or a network score presenting a good behaviour for the data at hand and to tune it appropriately. Furthermore, the task the Bayesian network will be used for should also be taken in consideration; for example, a simple structure (such as the naive Bayes classifier presented in Section 3.4.2) may be very good for prediction but useless for assessing the dependence structure of the data and the causal effects among the variables.

For this reason, we will now investigate the behaviour of permutation conditional independence tests and tests based on shrinkage estimators. These two classes of tests are usually not considered in literature, where the asymptotic χ^2 tests based on Pearson's X^2 and the mutual information are the *de facto* standard. In particular, we will study the permutation Pearson's X^2 test and the permutation mutual information test described in Edwards (2000), and the shrinkage test based on the estimator for the mutual information presented in Hausser and Strimmer (2009). Five performance indicators will be taken into consideration:

- the posterior density of the network (i.e. the BDe score) for the data it was learned from, as a measure of goodness of fit. The equivalent sample size will be set to 10 as suggested in Koller and Friedman (2009).
- the BIC score of the network for the data it was learned from, again as a measure of goodness of fit.
- the posterior density of the network for a new data set, as a measure of how well the network generalizes to new data.
- the BIC score of the network for a new data set, again as a measure of how well the network generalizes to new data.
- the Structural Hamming Distance (SHD) between the learned and the true structure of the network, as a measure of the quality of the learned dependence structure (and of the corresponding set of causal effects under the assumptions stated in Section 2.3).

These indicators will be estimated for each test using the **bnlearn** R package as follows:

1. a sample is generated from the true probability distribution of the ALARM network from Beinlich et al. (1989).

2. a network structure is learned with the Max-Min Hill-Climbing algorithm using the BDe score and one of the conditional independence tests under investigation. Two thresholds are considered for the type I error: 0.05 and 0.01. Since the results are very similar for both values, they are reported only for 0.05 for brevity.

3. a second network structure is learned from the same data with the asymptotic, parametric test based either on Pearson's X^2 or on the maximum likelihood estimator for the mutual information, depending on which test was used in the previous step.

4. the previous two steps are repeated using the BIC score instead of BDe.

5. the relevant performance indicators are computed for each pair of network structures, and the differences are standardized to express the relative difference over the values obtained with the asymptotic tests. In particular, BDe will be only considered for networks learned in step 2 and BIC for networks learned in step 4.

These steps will be repeated 50 times for each sample size. The data set needed to assess how well the network generalizes to new data is generated again from the true probability structure of the ALARM network and contains 20000 observations. The parameters of the network, which are the elements of the conditional probability tables associated with the nodes of the networks, are estimated using the corresponding empirical frequencies.

6.1.1 Permutation Tests

Nonparametric conditional independence tests, and permutation tests in particular, provide a feasible alternative to the corresponding parametric tests in a wide range of situations. Their main advantage is that they do not require a large sample size or particular distributional assumptions to perform well, because they operate conditioning on the available data (Pesarin and Salmaso, 2010). Therefore, they perform better than the parametric tests usually found in literature, because they are not limited by the rate of convergence to the respective asymptotic distributions. However, the computer time required by the generation of the permutations of the data and by the repeated evaluation of the test statistic have prevented their widespread use in many settings in which high-dimensional problems are the norm.

The effects of these properties of the permutation Pearson's X^2 and the permutation mutual information tests are shown in Figure 6.1 and Figure 6.2. First, we can clearly

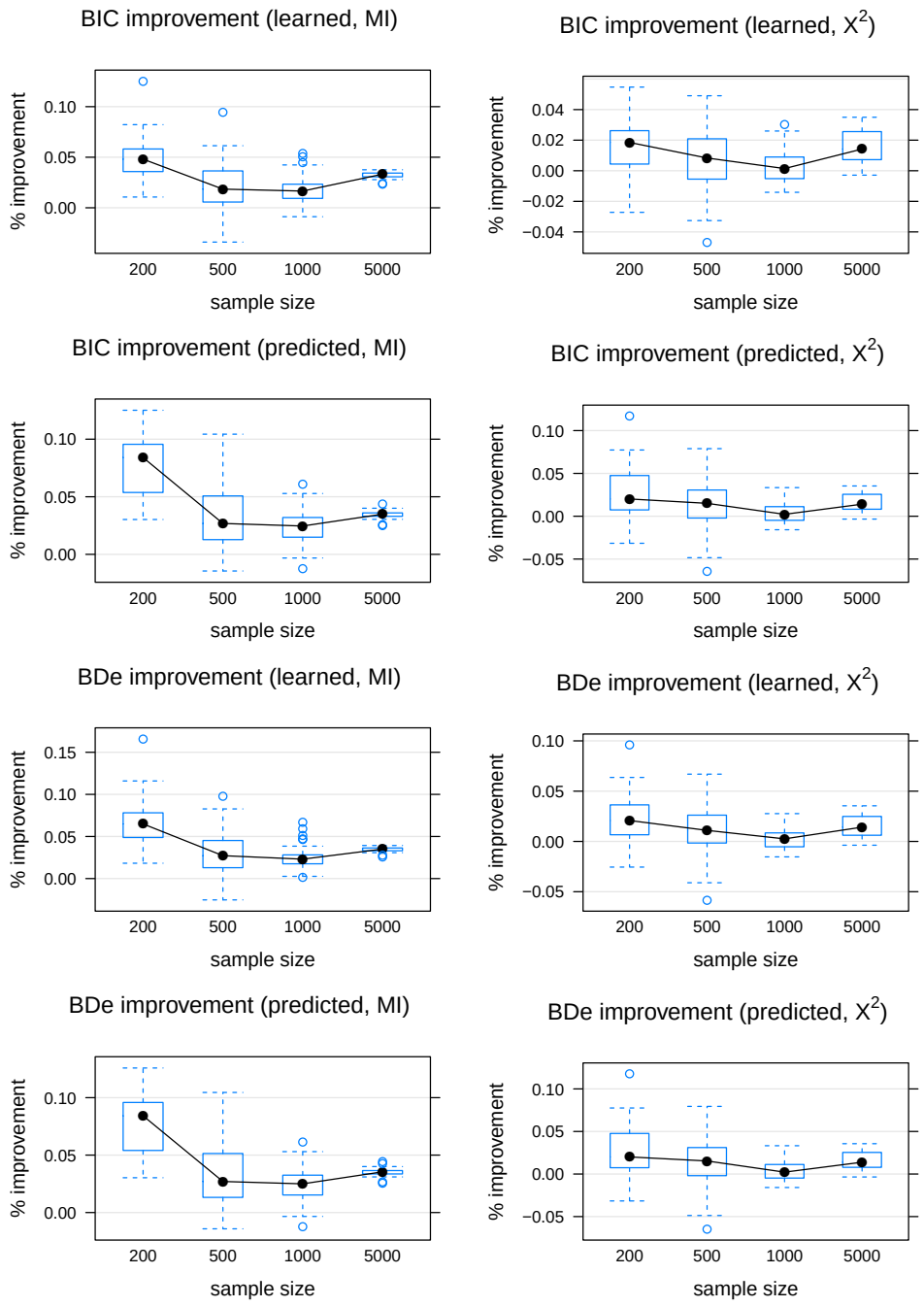


Figure 6.1: Improvements in Bayesian network structure learning when using permutation mutual information (on the left) and Pearson's X^2 (on the right) tests. The black dot in each boxplot represents the median.

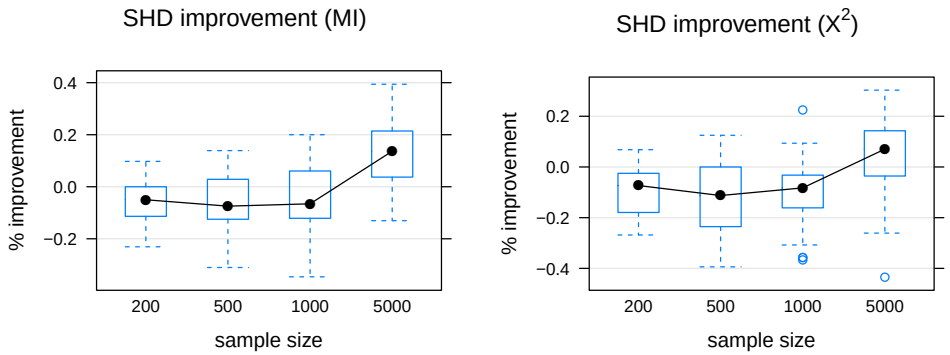


Figure 6.2: Differences in the Structural Hamming Distance when using permutation mutual information (on the left) and Pearson's X^2 (on the right) tests. The black dot in each boxplot represents the median.

see from the boxplots in Figure 6.1 that the use of permutation tests results in network structures with higher scores for all the considered sample sizes (200, 500, 1000 and 5000). This is also true when considering the new data set, meaning that the network structures learned with these tests are better for predicting the behaviour of new samples. As expected, the improvements in the BIC and BDe scores are particularly significant for low sample sizes; the probability structure of the ALARM network has 509 parameters, which means that the ratios between the number of observations and the number of parameters are 0.3929, 0.9823, 1.9646 and 9.8231 respectively.

It is also interesting to note that, even though the performance of parametric tests improves with the sample size, both permutation tests appear to improve at a faster rate. In fact, in all plots in Figure 6.1 the relative improvement for samples of size 5000 is greater than the corresponding improvement for samples of size 2000, regardless of the score we are considering or the data set it is computed from.

On the other hand, the network structures learned with the permutation tests considered in this section are often not as close to the true network structure as the ones learned with the corresponding parametric tests. This is can be clearly seen from the boxplots in Figure 6.2, which show that in the majority of simulations the relative difference between the SHD values is negative (i.e. the SHD associated with the parametric test is smaller than the SHD associated with the permutation test). Permutation tests outperform parametric tests only for samples of size 5000.

The comparatively poor performance of permutation tests in terms of SHD can be attributed to the conditioning on the observed sample that characterizes them. Most

of the samples considered in this analysis are too small to provide an adequate representation of the true probability structure of the ALARM network, as evidenced by the ratios between their sample sizes and the number of parameters. Therefore, the network structures learned with permutation tests from these samples are able to capture only part of the true dependence structure. The arcs that are most likely to be missed, however, are those that represent the weakest dependence relationships; otherwise the networks would not be able to fit new data so well.

In conclusion, permutation tests result in better network structures than the corresponding parametric tests, both in terms of goodness of fit and in how well the networks are able to generalize to new data. However, if the focus of the analysis is the structure of the network itself (such as when the Bayesian network is considered as a causal model) parametric tests may be preferable for small samples.

6.1.2 Tests Based on Shrinkage Estimators

The test based on the shrinkage estimator for the mutual information has a completely different behaviour than the permutation tests covered above.

As expected from a test based on a regularized estimator, the networks learned using shrinkage tests do not fit the data as well as the networks learned with the corresponding maximum likelihood tests. This can be clearly seen from the boxplots in Figure 6.3. The relative differences in the BIC and BDe scores are almost never positive for either the data the networks have been learned from or the new data, in particular for samples of size 10 and 20. Such small samples are most likely to result in sparse contingency tables, and therefore in high values of the shrinkage coefficient, as soon as a few conditioning variables are included in the tests. Larger samples are less affected by the regularization of the shrinkage estimator, because the shrinkage coefficient converges to zero as the number of observations diverges (Ledoit and Wolf, 2003). This means that for larger samples (i.e. 100, 150 and 200) the behaviour of the shrinkage test for the mutual information approaches the one of the classic mutual information test, as can be seen from the increasingly small difference between the two in terms of BIC and BDe scores.

An important side effect of the regularization performed by the shrinkage estimator is the reduction of the structural distance from the true network structure for small samples. We can see from Figure 6.4 that the shrinkage test outperforms the test based on the maximum likelihood estimator; there is a systematic improvement for sample sizes 10, 20 and 50 (i.e. SHD is smaller for the shrinkage test). Again as

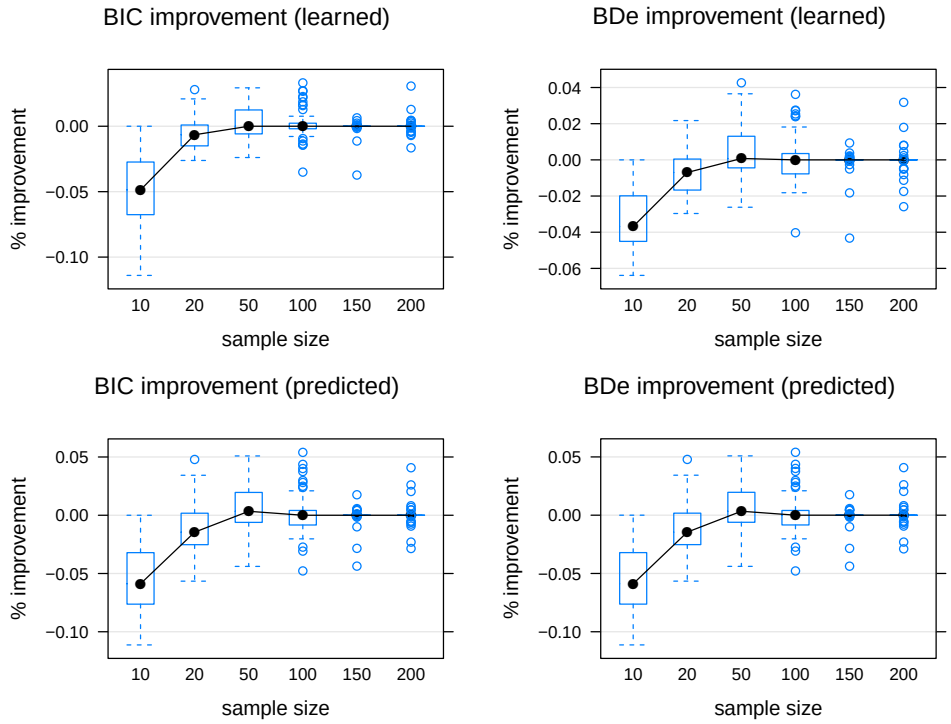


Figure 6.3: Improvements in Bayesian network structure learning when using the shrinkage estimator for the mutual information. The black dot in each boxplot represents the median.

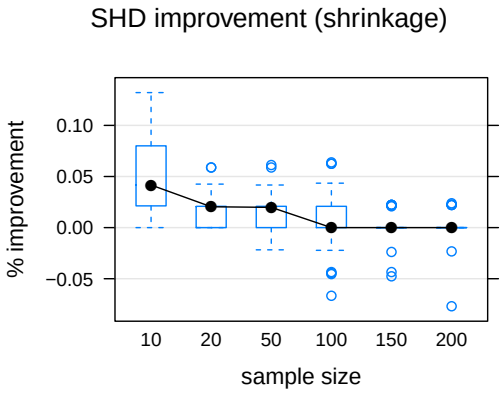


Figure 6.4: Differences in the Structural Hamming Distance when using the shrinkage estimator for the mutual information. The black dot in each boxplot represents the median.

the sample size increases the behaviour of the shrinkage test approaches the one of the corresponding maximum likelihood test. These simulations confirm the results produced with shrinkage tests for many "small n, large p" problems, such as in Schäfer and Strimmer (2005) and Krämer et al. (2009), which have led to a widespread use of shrinkage tests in biology and genetics.

6.2 Learning Strategies and Structure Variability

We will now look again at the results presented in the previous section and use the measures of variability from Chapter 5 to further support some of the considerations made therein. Due to the limitations in the analysis of directed acyclic graphs covered in Section 5.4, we will use the skeleton of the Bayesian networks learned from the ALARM data in combination with the descriptive statistics and the tests for the variability of undirected graphs as implemented in **bnlearn**.

6.2.1 Descriptive Statistics

Limiting ourselves to exploratory data analysis, we can use the descriptive statistics introduced in Section 5.1 to study how the variability of the network structure varies with the sample size. In particular, we can assess the influence of each component of the overall learning strategy, including but not limited to the structure learning algorithm, the network score, the conditional independence test and the respective parameters.

We will take into consideration the total variance and the squared Frobenius matrix norm in their normalized forms, $\overline{\mathsf{VAR}}_T(\Sigma)$ and $\overline{\mathsf{VAR}}_N(\Sigma)$. We will not consider the normalized generalized variance, $\overline{\mathsf{VAR}}_G(\Sigma)$, due to the arbitrariness implied by the choice of an algorithm to reduce Σ to a full rank matrix. The covariance matrix Σ will be estimated using the maximum likelihood estimator $\hat{\Sigma}$. Furthermore, only the results for the total variance will be discussed in detail due to the similarity between the behaviour of $\overline{\mathsf{VAR}}_T(\Sigma)$ and $\overline{\mathsf{VAR}}_N(\Sigma)$.

The values of $\overline{\mathsf{VAR}}_T(\Sigma)$ for the learning strategy used as a reference in the previous section (Max-Min Hill-Climbing, maximum likelihood estimator for the mutual information with $\alpha = 0.05$, BIC score, 20 runs for each sample size) are reported in Figure 6.5. As expected, the variability of the network structure decreases as the sample size diverges. This can be attributed, at least in part, to the consistency of the BIC score proved by Gámez et al. (2010). We can also see that the network structure is more stable for very small samples (10 and 20 observations) than for medium-sized samples

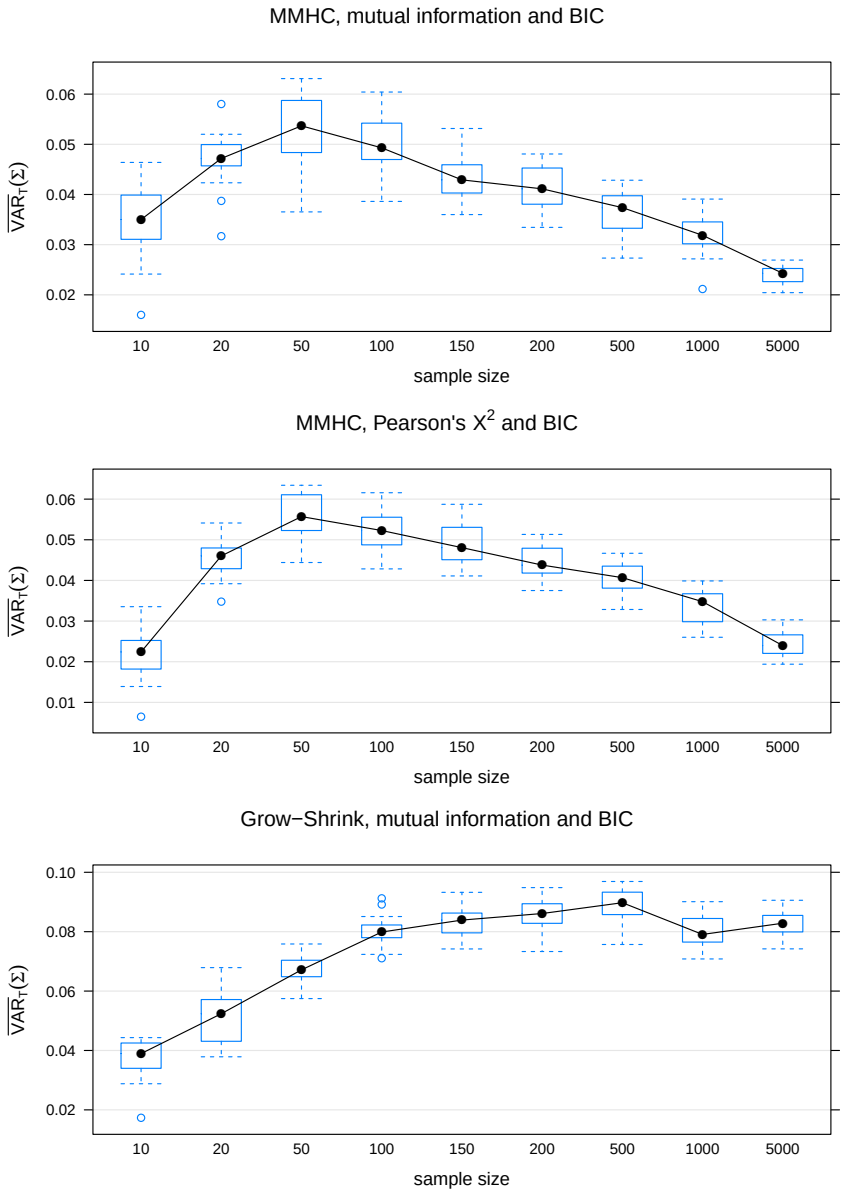


Figure 6.5: Normalized total variance for different sample sizes and three different learning strategies. The black dot in each boxplot represents the median.

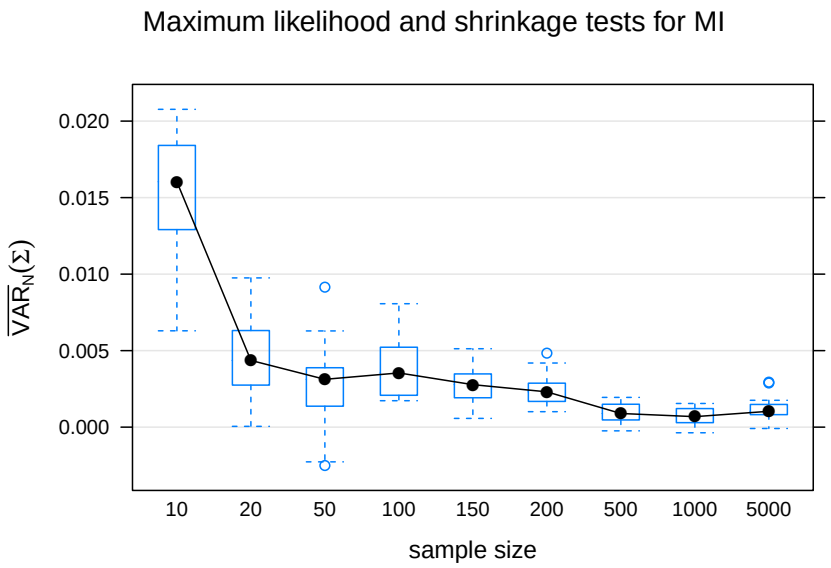


Figure 6.6: Difference in normalized total variance between the maximum likelihood and the shrinkage estimator for the mutual information. The black dot in each boxplot represents the median.

(50 to 200 observations); the highest variability is observed for sample size 50. The networks learned from these samples are almost all empty (i.e. they do not have any arc), and are therefore very stable but not very informative.

The last plot in Figure 6.5 shows the normalized total variance for the Grow-Shrink algorithm, which serves as a useful term of comparison. We can see that the variability of the network structure does not appear to decrease as the sample size increases, or at least it does so slowly that the trend is not discernible from the plot. This is clearly an undesirable behaviour, and is consistent with the claims that constraint-based algorithms are relatively unstable compared to score-based and hybrid algorithms (Spirtes et al., 2000). However, the normalized total variance never exceeds 0.10, which seems to imply that the performance of Grow-Shrink is still acceptable.

The difference in normalized total variance between the tests based on the maximum likelihood and the shrinkage estimator for the mutual information are shown in Figure 6.6. We can see that, as in Section 6.1.2, the shrinkage test provides a better performance than the classic parametric test for all the sample sizes considered in the analysis – i.e. the normalized total variance is always less than the normalized total variance computed from the sample samples using the test based on the maximum like-

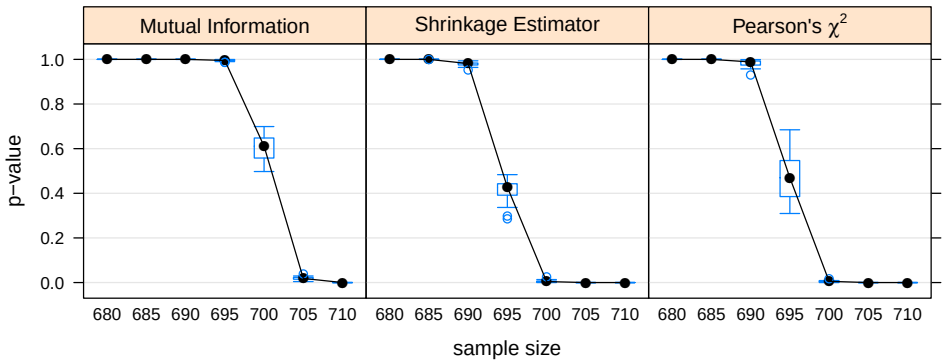


Figure 6.7: Significance values for three different conditional independence tests (maximum likelihood and shrinkage estimators of mutual information and Pearson's χ^2) used with the same structure learning algorithm (Grow-Shrink). The black dot in each boxplot represents the median.

lihood estimator. The difference between the two tests vanishes again as the sample size grows, for the same reasons stated above.

6.2.2 Testing Against the Maximum Entropy Distribution

Consider now the tests introduced in Section 5.2. In particular, we will use them to determine the minimum sample size required by each strategy to detect reliably at least part of the dependence structure of the data.

First we will compare the performance of the Grow-Shrink algorithm for three different conditional independence tests: the asymptotic test based on the maximum likelihood estimator for the mutual information, the corresponding shrinkage test and Pearson's X^2 test. The thresholds $\alpha = 0.01$ and $\alpha = 0.05$ for type I error will be used for each conditional independence test, and network variability will be assessed with the Monte Carlo test for the squared Frobenius matrix norm. As in the previous section, the results of the simulations are very similar; therefore, we will discuss only the results for $\alpha = 0.05$ for brevity.

The learning algorithm has been applied to samples of several sizes between 10 and 5000 (20 times for each size) from the ALARM network; only the relevant ones, 680, 685, 690, 695, 700, 705 and 710, are shown in Figure 6.7. All the tests considered in the analysis start producing relatively stable network structures – i.e. the null hypothesis corresponding to the maximum entropy case is rejected – at sample sizes 695 and 700.

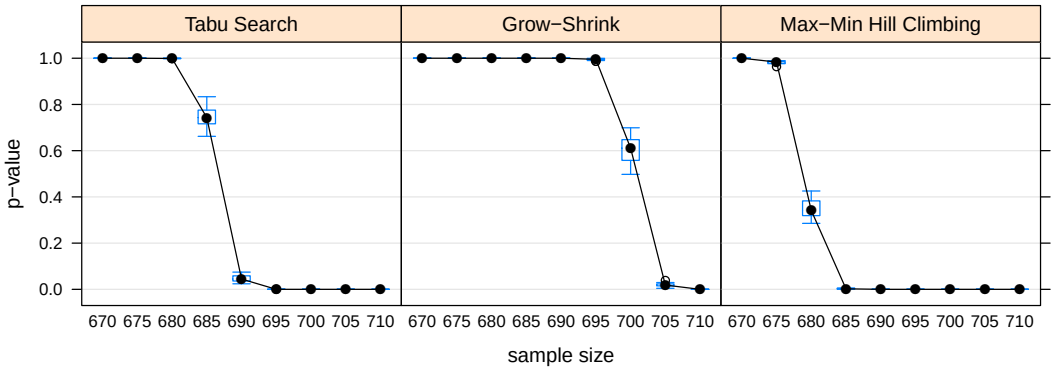


Figure 6.8: Significance values for three different structure learning algorithms (Grow-Shrink, tabu search and Max-Min Hill-Climbing) using the same conditional independence tests and network scores. The black dot in each boxplot represents the median.

All combinations of structure learning algorithms, conditional independence tests and values of α are not able to capture anything but noise from smaller samples. Pearson's X^2 test appears to perform slightly better than mutual information, as documented in Agresti (2002) when dealing with sparse contingency tables. This is also true for the shrinkage test. However, the difference among the three sets of significance values is very small.

It is also interesting to compare three different learning algorithms using the same conditional independence tests and network scores:

- tabu search (a score-based algorithm), combined with a BIC score;
- Grow-Shrink (a constraint-based algorithm), combined with the asymptotic χ^2 test for the mutual information and $\alpha = 0.05$;
- Max-Min Hill-Climbing (a hybrid algorithm), combined with a BIC score and the asymptotic mutual information test.

Again only the significance values for the relevant sample sizes are reported in Figure 6.8; in this case the differences between the learning strategies are more pronounced. The network structures learned with the Max-Min Hill-Climbing algorithm, which is one of the top performers up to date for large networks, display less variability than the ones learned with either tabu search or Grow-Shrink at the same sample size. In particular the difference between Max-Min Hill-Climbing and Grow-Shrink confirms

again the results presented in Tsamardinos et al. (2006) and the relative instability of constraint-based algorithms at small sample sizes. Furthermore, if we compare the significance values in Figure 6.8 with the ones in Figure 6.7 we can see that a choosing a good structure learning algorithm may be more important than choosing a good statistical test or network score for particularly small samples. From this we can argue that the contribution of the heuristic the former is based on to the stability of the network is determinant, as it offsets the errors made by the latter.

Chapter 7

Conclusions

In this thesis we proposed some new methods for the analysis of the structure of Bayesian and Markov networks, extending the bootstrap-based approach introduced by Friedman et al. (1999a). These methods focus on the multivariate variability of the network structure and are based on the univariate measures of multivariate variability present in classic multivariate statistics (Mardia et al., 1979; Muirhead, 1982; Bilodeau and Brenner, 1999).

Both descriptive statistics and hypothesis tests have been proposed and their properties studied under suitable probabilistic assumptions on the arcs (or edges) of the network (Scutari, 2009). Directed acyclic graphs (such as those representing Bayesian networks) and undirected graphs (such as Markov networks or the skeleton of Bayesian networks) have been modelled using the multivariate extensions of the Bernoulli and Trinomial distributions (Krummenauer, 1998b); each component have been associated with an arc or an edge. These assumptions represent the natural probabilistic model for these network structures; they do not impose any assumption that is not already implicit in the nature of the bootstrapped network structures themselves. Furthermore, inference on these probabilistic models requires only few simple, closed-form parameter estimators which can be computed as in Friedman et al. (1999a).

The use of these probabilistic models allowed the derivation of several bounds and exact results concerning the first two order of moments of both directed and undirected network structures. This in turn allowed the derivation of bounds for the descriptive statistics, giving them a clear interpretation as variability measures. The null distribution of hypothesis tests concerning the stability of the network structure have been similarly derived, and both Monte Carlo and asymptotic approaches have been studied.

To complement these theoretical results and to examine the behaviour of the statistics

introduced in this thesis we implemented the **bnlearn** R package (Scutari, 2010a,b), which is available from CRAN under the GPL license. **bnlearn** provides a free-software implementation of several structure and parameter learning algorithms, including some that were not publicly available under liberal licenses. In addition, several alternatives for network scores and conditional independence tests not commonly used in literature are also provided. **bnlearn** also explores the applications of parallel computing to Bayesian networks, which will be covered by the author in a book by Springer (Nagarajan et al., 2011) along with the other features of the package.

7.1 Open Problems

Further research on the analysis of the structure of Bayesian and Markov networks will focus on the improvement of the statistics proposed in this thesis and the derivation of further results on the multivariate Trinomial distribution, which lacks a complete characterization of the second order moments in the *maximum entropy case*.

The latter topic is of particular interest. A complete characterization of both limiting cases of the multivariate Trinomial distribution in terms of entropy may remove most of the limitations in the analysis of the structure of Bayesian networks. In particular, transforming the network structure to an undirected graph (the skeleton or the moral graph) would no longer be required. It is also important to note that such a characterization may also lead to interesting results in the theory of random graphs.

As for the statistics proposed in this thesis, two topics merit further investigation. First, it may be that the total variance, the generalized variance and the squared Frobenius matrix norm are not the best measures for the variability of a network structure. Their use in multivariate statistics is motivated largely by their properties for the multivariate normal distribution; therefore it may be that some other statistic, such as the ones developed for contingency tables or the analysis of variance, are more effective while still having a clear interpretation. Second, the development of tests for more complex hypotheses may help in answering some of the problems studied in Friedman et al. (1999a). Testing the stability of the network using the *maximum entropy* case as the null hypothesis is only a starting point for the development of tests concerning the distance between network structures and the dependence between arcs, which may result from the presence of a causal pathway.

Appendix A

Moments of the Multivariate Trinomial Distribution

In this appendix we will list the first two moments of the Mutivariate Trinomial distribution used to model the behaviour of the arcs in directed acyclic graphs. All the quantities presented below have been computed by a complete enumeration of the directed acyclic graphs of a given size (3, 4, 5, 6 and 7), and are therefore exact values because they are estimated from the whole population. The number of graphs have been computed using the recursive relation from Harary and Palmer (1973), and the graphs themselves have been generated using the Markov Chain Monte Carlo algorithm from Ide and Cozman (2002).

For each graph size is reported:

- the marginal distribution of each arc, including its expected value and its variance.
- the covariance between each possible pair of arcs, both those not incident on a common node (a_{ij}, a_{kl}, $i \neq j \neq k \neq l$) and those incident on a common node.
- the joint probability table for each possible pair of arcs, both those not incident on a common node and those incident on a common node.

A.1 Number of directed acyclic graphs of given size

Graph size	3	4	5	6	7
Number of graphs	25	543	29281	3781503	1138779265

A.2 Moments for the 3-dimensional distribution

$$A_{ij} = \begin{cases} -1 & \text{with probability } 0.32 \\ 0 & \text{with probability } 0.36 \\ 1 & \text{with probability } 0.32 \end{cases} \qquad \begin{aligned} &\mathsf{E}(A_{ij}) = 0 \\ &\mathsf{VAR}(A_{ij}) = 0.64 \\ &\mathsf{COV}(A_{ij}, A_{kl}) = 0.08 \end{aligned}$$

	$\mathring{a_{ij}}$	$\overrightarrow{a_{ij}}$	$\overleftarrow{a_{ij}}$
$\mathring{a_{ik}}$	0.120000	0.120000	0.120000
$\overrightarrow{a_{ik}}$	0.120000	0.120000	0.080000
$\overleftarrow{a_{ik}}$	0.120000	0.080000	0.120000

arcs incident on a common node

A.3 Moments for the 4-dimensional distribution

$$A_{ij} = \begin{cases} -1 & \text{with probability } 0.309392 \\ 0 & \text{with probability } 0.381215 \\ 1 & \text{with probability } 0.309392 \end{cases} \qquad \begin{aligned} &\mathsf{E}(A_{ij}) = 0 \\ &\mathsf{VAR}(A_{ij}) = 0.618784 \end{aligned}$$

$$|\mathsf{COV}(A_{ij}, A_{kl})| = \begin{cases} 0 & \text{if } i \neq j \neq k \neq l \\ 0.081031 & \text{otherwise} \end{cases}$$

	$\mathring{a_{ij}}$	$\overrightarrow{a_{ij}}$	$\overleftarrow{a_{ij}}$
$\mathring{a_{kl}}$	0.145488	0.117863	0.117863
$\overrightarrow{a_{kl}}$	0.117863	0.095764	0.095764
$\overleftarrow{a_{kl}}$	0.117863	0.095764	0.095764

arcs not incident on a common node

	$\mathring{a_{ij}}$	$\overrightarrow{a_{ij}}$	$\overleftarrow{a_{ij}}$
$\mathring{a_{ik}}$	0.138121	0.121546	0.121546
$\overrightarrow{a_{ik}}$	0.121546	0.114180	0.073664
$\overleftarrow{a_{ik}}$	0.121546	0.073664	0.114180

arcs incident on a common node

A.4 Moments for the 5-dimensional distribution

$$A_{ij} = \begin{cases} -1 & \text{with probability } 0.301082 \\ 0 & \text{with probability } 0.397834 \\ 1 & \text{with probability } 0.301082 \end{cases}$$

$$\mathsf{E}(A_{ij}) = 0$$
$$\mathsf{VAR}(A_{ij}) = 0.602165$$

$$|\mathsf{COV}(A_{ij}, A_{kl})| = \begin{cases} 0 & \text{if } i \neq j \neq k \neq l \\ 0.081691 & \text{otherwise} \end{cases}$$

	a°_{ij}	$\overrightarrow{a_{ij}}$	$\overleftarrow{a_{ij}}$
a°_{kl}	0.152761	0.122536	0.122536
$\overrightarrow{a_{kl}}$	0.122536	0.068850	0.068850
$\overleftarrow{a_{kl}}$	0.122536	0.068850	0.068850

arcs not incident on a common node

	a°_{ij}	$\overrightarrow{a_{ij}}$	$\overleftarrow{a_{ij}}$
a°_{ik}	0.152761	0.122536	0.122536
$\overrightarrow{a_{ik}}$	0.122536	0.109695	0.068851
$\overleftarrow{a_{ik}}$	0.122536	0.068851	0.109695

arcs incident on a common node

A.5 Moments for the 6-dimensional distribution

$$A_{ij} = \begin{cases} -1 & \text{with probability } 0.294562 \\ 0 & \text{with probability } 0.410875 \\ 1 & \text{with probability } 0.294562 \end{cases}$$

$$\mathsf{E}(A_{ij}) = 0$$
$$\mathsf{VAR}(A_{ij}) = 0.589124$$

$$|\mathsf{COV}(A_{ij}, A_{kl})| = \begin{cases} 0 & \text{if } i \neq j \neq k \neq l \\ 0.082121 & \text{otherwise} \end{cases}$$

	a°_{ij}	$\overrightarrow{a_{ij}}$	$\overleftarrow{a_{ij}}$
a°_{kl}	0.169041	0.120917	0.120917
$\overrightarrow{a_{kl}}$	0.120917	0.086822	0.086822
$\overleftarrow{a_{kl}}$	0.120917	0.086822	0.086822

arcs not incident on a common node

	a°_{ij}	$\overrightarrow{a_{ij}}$	$\overleftarrow{a_{ij}}$
a°_{ik}	0.164510	0.123182	0.123182
$\overrightarrow{a_{ik}}$	0.123182	0.106220	0.065159
$\overleftarrow{a_{ik}}$	0.123182	0.065159	0.106220

arcs incident on a common node

A.6 Moments for the 7-dimensional distribution

$$A_{ij} = \begin{cases} -1 & \text{with probability } 0.289390 \\ 0 & \text{with probability } 0.421220 \\ 1 & \text{with probability } 0.289390 \end{cases}$$

$$\mathsf{E}(A_{ij}) = 0$$
$$\mathsf{VAR}(A_{ij}) = 0.578780$$

$$|\mathsf{COV}(A_{ij}, A_{kl})| = \begin{cases} 0 & \text{if } i \neq j \neq k \neq l \\ 0.82410 & \text{otherwise} \end{cases}$$

	$\mathring{a_{ij}}$	$\overrightarrow{a_{ij}}$	$\overleftarrow{a_{ij}}$
$\mathring{a_{kl}}$	0.177620	0.122800	0.122800
$\overrightarrow{a_{kl}}$	0.122800	0.083795	0.083795
$\overleftarrow{a_{kl}}$	0.122800	0.083795	0.083795

arcs not incident on a common node

	$\mathring{a_{ij}}$	$\overrightarrow{a_{ij}}$	$\overleftarrow{a_{ij}}$
$\mathring{a_{ik}}$	0.173986	0.123617	0.123617
$\overrightarrow{a_{ik}}$	0.123617	0.103489	0.062284
$\overleftarrow{a_{ik}}$	0.123617	0.062284	0.103489

arcs incident on a common node

Appendix B

Ledoit-Wolf Estimators for the Shrinkage Coefficient

In Chapter 5 we defined the values of the shrinkage coefficient λ that minimize the mean squared error of $\tilde{\Sigma}$ as

$$\lambda^* = \frac{\sum_{i=1}^{k} \sum_{j \neq i} \mathsf{VAR}(\hat{p}_{ij} - \hat{p}_i \hat{p}_j) + \sum_{i=1}^{k} \mathsf{VAR}(\hat{p}_i - \hat{p}_i^2)}{\sum_{i=1}^{k} \sum_{j \neq i} (\hat{p}_{ij} - \hat{p}_i \hat{p}_j)^2 + \sum_{i=1}^{k} (\hat{p}_i - \hat{p}_i^2 - \frac{1}{4})^2}, \tag{B.1}$$

for the target $T = \frac{1}{4} I_k$, and as

$$\lambda^* = \frac{\sum_{i=1}^{k} \sum_{j \neq i} \mathsf{VAR}(\hat{p}_{ij} - \hat{p}_i \hat{p}_j) + 2 \sum_{i=1}^{k} \mathsf{VAR}(\hat{p}_i - \hat{p}_i^2)}{\sum_{i=1}^{k} \sum_{j \neq i} (\hat{p}_{ij} - \hat{p}_i \hat{p}_j)^2}. \tag{B.2}$$

for $T = \mathrm{diag}(\hat{\Sigma})$. Recall from Section 4.4.1 that for two edges e_i and e_j, $i \leqslant k$, $j \leqslant k$, the probabilities p_i, p_j and p_{ij} are estimated from the m bootstrapped networks as

$$\hat{p}_i = \frac{1}{m} \sum_{b=1}^{m} \mathbb{1}_{\{e \in E_b\}}(e_i), \tag{B.3}$$

$$\hat{p}_j = \frac{1}{m} \sum_{b=1}^{m} \mathbb{1}_{\{e \in E_b\}}(e_j), \tag{B.4}$$

$$\hat{p}_{ij} = \frac{1}{m} \sum_{b=1}^{m} \mathbb{1}_{\{e \in E_b\}}(e_i) \mathbb{1}_{\{e \in E_b\}}(e_j). \tag{B.5}$$

In both Equation B.1 and Equation B.2 we need to derive a closed form expressions for $\mathsf{VAR}(\hat{p}_i - \hat{p}_i^2)$ and $\mathsf{VAR}(\hat{p}_{ij} - \hat{p}_i\hat{p}_j)$. The former can be rewritten as

$$\begin{aligned}\mathsf{VAR}(\hat{p}_i - \hat{p}_i^2) &= \mathsf{VAR}(\hat{p}_i) + \mathsf{VAR}(\hat{p}_i^2) - 2\mathsf{COV}(\hat{p}_i, \hat{p}_i^2) = \\ &= \left[\mathsf{E}(\hat{p}_i^2) - \mathsf{E}(\hat{p}_i)^2\right] + \left[\mathsf{E}(\hat{p}_i^4) - \mathsf{E}(\hat{p}_i^2)^2\right] - 2\left[\mathsf{E}(\hat{p}_i^3) - \mathsf{E}(\hat{p}_i)\mathsf{E}(\hat{p}_i^2)\right]. \quad \text{(B.6)}\end{aligned}$$

If we let $Y_i = \sum_{b=1}^{m} \mathbb{1}_{\{e \in E_b\}}(e_i) \sim Bin(m, p_i)$, we can compute all the expectations in the right hand of the previous expression using the first four moments of the Binomial random variable:

$$\mathsf{E}(\hat{p}_i) = \mathsf{E}\left(\frac{1}{m}Y_i\right) = p_i, \quad \text{(B.7)}$$

$$\mathsf{E}(\hat{p}_i^2) = \mathsf{E}\left(\left[\frac{1}{m}Y_i\right]^2\right) = \frac{1}{m^2}\left[mp_i + m(m-1)p_i^2\right], \quad \text{(B.8)}$$

$$\mathsf{E}(\hat{p}_i^3) = \mathsf{E}\left(\left[\frac{1}{m}Y_i\right]^3\right) = \frac{1}{m^3}\left[mp_i + 3m(m-1)p_i^2 + m(m-1)(m-2)p_i^3\right], \quad \text{(B.9)}$$

$$\begin{aligned}\mathsf{E}(\hat{p}_i^4) = \mathsf{E}\left(\left[\frac{1}{m}Y_i\right]^4\right) = \frac{1}{m^4}\Big[&mp_i + 7m(m-1)p_i^2 + 6m(m-1)(m-2)p_i^3 + \\ &+ m(m-1)(m-2)(m-3)p_i^4\Big]. \quad \text{(B.10)}\end{aligned}$$

As for $\mathsf{VAR}(\hat{p}_{ij} - \hat{p}_i\hat{p}_j)$, it can be rewritten as

$$\begin{aligned}&\mathsf{VAR}(\hat{p}_{ij} - \hat{p}_i\hat{p}_j) = \mathsf{VAR}(\hat{p}_{ij}) + \mathsf{VAR}(\hat{p}_i\hat{p}_j) - 2\mathsf{COV}(\hat{p}_{ij}, \hat{p}_i\hat{p}_j) = \\ &= \left[\mathsf{E}(\hat{p}_{ij}^2) - \mathsf{E}(\hat{p}_{ij})^2\right] + \left[\mathsf{E}(\hat{p}_i^2\hat{p}_j^2) - \mathsf{E}(\hat{p}_i\hat{p}_j)^2\right] - 2\left[\mathsf{E}(\hat{p}_{ij}\hat{p}_i\hat{p}_j) - \mathsf{E}(\hat{p}_{ij})\mathsf{E}(\hat{p}_i\hat{p}_j)\right]. \quad \text{(B.11)}\end{aligned}$$

As before, if we let $Y_{ij} = \sum_{b=1}^{m} \mathbb{1}_{\{e \in E_b\}}(e_i)\mathbb{1}_{\{e \in E_b\}}(e_j) \sim Bin(m, p_{ij})$ we have that

$$\mathsf{E}(\hat{p}_{ij}) = \mathsf{E}\left(\frac{1}{m}Y_{ij}\right) = p_{ij}, \quad \text{(B.12)}$$

$$\mathsf{E}(\hat{p}_{ij}^2) = \mathsf{E}\left(\left[\frac{1}{m}Y_{ij}\right]^2\right) = \frac{1}{m^2}\left[mp_{ij} + m(m-1)p_{ij}^2\right]. \quad \text{(B.13)}$$

The remaining expectations are computed against the joint distribution of the random variables Y_{ij}, $Y_i \sim Bin(m, p_i)$ and $Y_j \sim Bin(m, p_j)$. They can be rewritten as follows:

$$\begin{aligned}
\mathsf{E}(\hat{p}_i\hat{p}_j) &= \mathsf{E}\left(\left[\frac{1}{m}\sum_{b=1}^{m}\mathbb{1}_{\{e\in E_b\}}(e_i)\right]\left[\frac{1}{m}\sum_{c=1}^{m}\mathbb{1}_{\{e\in E_c\}}(e_j)\right]\right) \\
&= \frac{1}{m^2}\mathsf{E}\left(\sum_{b=1}^{m}\sum_{c=1}^{m}\mathbb{1}_{\{e\in E_b\}}(e_i)\mathbb{1}_{\{e\in E_c\}}(e_j)\right) \\
&= \frac{1}{m^2}\mathsf{E}\left(\sum_{b=1}^{m}\mathbb{1}_{\{e\in E_b\}}(e_i)\mathbb{1}_{\{e\in E_b\}}(e_j)\right) + \\
&\quad + \frac{1}{m^2}\mathsf{E}\left(\sum_{b=1}^{m}\sum_{c\neq b}\mathbb{1}_{\{e\in E_b\}}(e_i)\mathbb{1}_{\{e\in E_c\}}(e_j)\right) \\
&= \frac{1}{m^2}\left[mp_{ij} + (m^2p_ip_j - mp_{ij})\right] = p_ip_j \qquad \text{(B.14)} \\
\mathsf{E}(\hat{p}_i^2\hat{p}_j^2) &= \frac{1}{m^2}\left[mp_ip_j + m(m-1)p_i^2p_j^2\right] \qquad \text{(B.15)} \\
\mathsf{E}(\hat{p}_{ij}\hat{p}_i\hat{p}_j) &= \mathsf{E}\left(\left[\frac{1}{m}\sum_{b=1}^{m}\mathbb{1}_{\{e\in E_b\}}(e_i)\mathbb{1}_{\{e\in E_b\}}(e_j)\right]\cdot\right. \\
&\quad \left.\cdot\left[\frac{1}{m}\sum_{c=1}^{m}\mathbb{1}_{\{e\in E_c\}}(e_i)\right]\left[\frac{1}{m}\sum_{d=1}^{m}\mathbb{1}_{\{e\in E_d\}}(e_j)\right]\right) \\
&= \frac{1}{m^3}\mathsf{E}\left(\sum_{b=1}^{m}\sum_{c=1}^{m}\sum_{d=1}^{m}\mathbb{1}_{\{e\in E_b\}}(e_i)\mathbb{1}_{\{e\in E_b\}}(e_j)\mathbb{1}_{\{e\in E_c\}}(e_i)\mathbb{1}_{\{e\in E_d\}}(e_j)\right) \\
&= \frac{1}{m^3}\mathsf{E}\left(\sum_{b=1}^{m}\mathbb{1}_{\{e\in E_b\}}(e_i)\mathbb{1}_{\{e\in E_b\}}(e_j)\right) + \\
&\quad + \frac{1}{m^3}\mathsf{E}\left(\sum_{b=1}^{m}\sum_{c\neq b}\mathbb{1}_{\{e\in E_b\}}(e_i)\mathbb{1}_{\{e\in E_b\}}(e_j)\mathbb{1}_{\{e\in E_c\}}(e_i)\right) + \\
&\quad + \frac{1}{m^3}\mathsf{E}\left(\sum_{b=1}^{m}\sum_{d\neq b}\mathbb{1}_{\{e\in E_b\}}(e_i)\mathbb{1}_{\{e\in E_b\}}(e_j)\mathbb{1}_{\{e\in E_d\}}(e_j)\right) + \\
&\quad + \frac{1}{m^3}\mathsf{E}\left(\sum_{b=1}^{m}\sum_{c\neq b}\sum_{d\neq b}\mathbb{1}_{\{e\in E_b\}}(e_i)\mathbb{1}_{\{e\in E_b\}}(e_j)\mathbb{1}_{\{e\in E_c\}}(e_i)\mathbb{1}_{\{e\in E_d\}}(e_j)\right) \\
&= \frac{1}{m^3}\left[mp_{ij} + mp_{ij}(mp_i - 1) + mp_{ij}(mp_j - 1) + mp_{ij}(mp_i - 1)(mp_j - 1)\right] \\
&= p_{ij}p_ip_j \qquad \text{(B.16)}
\end{aligned}$$

The estimates for λ^* can then be computed by plugging in the maximum likelihood estimates $\hat{p}_i$, $\hat{p}_j$, $\hat{p}_{ij}$ in the formulas derived above.

Bibliography

Abramson, B., Brown, J., Edwards, W., Murphy, A., and Winkler, R. L. (1996). Hailfinder: A Bayesian System for Forecasting Severe Weather. *International Journal of Forecasting*, 12(1):57–71.

Agresti, A. (2002). *Categorical Data Analysis*. Wiley, 2nd edition.

Akaike, H. (1974). A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control*, 19(6):716 – 723.

Anderson, T. W. (2003). *An Introduction to Multivariate Statistical Analysis*. Wiley, 3rd edition.

Andreassen, S., Jensen, F., Andersen, S., Falck, B., Kjærulff, U., Woldbye, M., Sørensen, A., Rosenfalck, A., and Jensen, F. (1989). MUNIN – An Expert EMG Assistant. In Desmedt, J. E., editor, *Computer-Aided Electromyography and Expert Systems*. Elsevier.

Ash, R. B. (2000). *Probability and Measure Theory*. Academic Press, 2nd edition.

Beinlich, I., Suermondt, H. J., Chavez, R. M., and Cooper, G. F. (1989). The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks. In *Proceedings of the 2nd European Conference on Artificial Intelligence in Medicine*, pages 247–256. Springer-Verlag.

Billingsley, P. (1995). *Probability and Measure*. Wiley, 3rd edition.

Bilodeau, M. and Brenner, D. (1999). *Theory of Multivariate Statistics*. Springer-Verlag.

Binder, J., Koller, D., Russell, S., and Kanazawa, K. (1997). Adaptive Probabilistic Networks with Hidden Variables. *Machine Learning*, 29(2–3):213–244.

Bishop, Y. M. M., Fienberg, S. E., and Holland, P. W. (2007). *Discrete Multivariate Analysis: Theory and Practice*. Springer.

Borgelt, C., Steinbrecher, M., and Krus, R. (2009). *Graphical Models: Representations for Learning, Reasoning and Data Mining*. Wiley, 2nd edition.

Bøttcher, S. G. and Dethlefsen, C. (2003). deal: A Package for Learning Bayesian Networks. *Journal of Statistical Software*, 8(20):1–40.

Bouckaert, R. R. (1995). *Bayesian Belief Networks: from Construction to Inference*. PhD thesis, Utrecht University, The Netherlands.

Butler, R. W., Huzurbazar, S., and Booth, J. G. (1992). Saddlepoint Approximations for the Generalized Variance and Wilks' Statistic. *Biometrika*, 79(1):157–169.

Castelo, R. and Roverato, A. (2006). A Robust Procedure For Gaussian Graphical Model Search From Microarray Data With p Larger Than n. *Journal of Machine Learning Research*, 7:2621–2650.

Castillo, E., Gutiérrez, J. M., and Hadi, A. S. (1997). *Expert Systems and Probabilistic Network Models*. Springer.

Chavan, S. S., Bauer, M. A., Scutari, M., and Nagarajan, R. (2009). NATbox: a Network Analysis Toolbox in R. *BMC Bioinformatics*, 10(Suppl 11):S14. Supplement contains the Proceedings of the 6th Annual MCBIOS Conference (Transformational Bioinformatics: Delivering Value from Genomes).

Chickering, D. M. (1995). A Transformational Characterization of Equivalent Bayesian Network Structures. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI95)*, pages 87–98.

Chickering, D. M. (1996). Learning Bayesian Networks is NP-Complete. In Fisher, D. and Lenz, H. J., editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag.

Chickering, D. M. (2002). Optimal Structure Identification with Greedy Search. *Journal of Machine Learning Resesearch*, 3:507–554.

Cooper, G. F. and Herskovits, E. (1992). A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning*, 9(4):309–347.

Cover, T. A. and Thomas, J. A. (2006). *Elements of Information Theory*. Wiley.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society Series B*, 39:1–39.

Diestel, R. (2005). *Graph Theory*. Springer, 3rd edition.

Edwards, D. I. (2000). *Introduction to Graphical Modelling*. Springer, 2nd edition.

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

Elidan, G. (2001). Bayesian Network Repository. Main web site hosted at http://www.cs.huji.ac.il/site/labs/compbio/Repository/.

Elidan, G. and Friedman, N. (2005). Learning Hidden Variable Networks: The Information Bottleneck Approach. *Journal of Machine Learning Research*, 6:81–127.

Fisher, N. I. and Sen, P. K. (1994). *The Collected Works of Wassily Hoeffding*. Springer-Verlag.

Fisher, R. A. (1921). On the Probable Error of a Coefficient of Correlation Deduced from a Small Sample. *Metron*, 1:1–32.

Friedman, J., Hastie, T., and Tibshirani, R. (2007). Sparse Inverse Covariance Estimation With the Graphical Lasso. *Biostatistics*, 9:432–441.

Friedman, N. (1997). Learning Belief Networks in the Presence of Missing Values and Hidden Variables. In *Proceedings of the 14th International Conference on Machine Learning (ICML97)*, pages 125–133.

Friedman, N., Goldszmidt, M., and Wyner, A. (1999a). Data Analysis with Bayesian Networks: A Bootstrap Approach. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 206 – 215. Morgan Kaufmann.

Friedman, N. and Koller, D. (2003). Being Bayesian about Bayesian Network Structure: A Bayesian Approach to Structure Discovery in Bayesian Networks. *Machine Learning*, 50(1–2):95–126.

Friedman, N., Linial, M., and Nachman, I. (2000). Using Bayesian Networks to Analyze Expression Data. *Journal of Computational Biology*, 7:601–620.

Friedman, N., Pe'er, D., and Nachman, I. (1999b). Learning Bayesian Network Structure from Massive Datasets: The "Sparse Candidate" Algorithm. In *Proceedings of 15th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 206–221. Morgan Kaufmann.

Frohlich, H., Sahin, O., Arlt, D., Bender, C., and Beißbarth, T. (2009). Deterministic Effects Propagation Networks for Reconstructing Protein Signaling Networks from Multiple Interventions. *BMC Bioinformatics*, 10(1):322.

Gámez, J. A., Mateo, J., and Puerta, J. (2010). Learning Bayesian Networks by Hill Climbing: Efficient Methods Based on Progressive Restriction of the Neighborhood. *Data Mining and Knowledge Discovery*, pages 1–43.

Ge, Y., Li, C., and Yin, Q. (2010). Study on Factors of Floating Womens Income in Jiangsu Province Based on Bayesian Networks. In Zeng, Z. and Wang, J., editors, *Advances in Neural Network Research and Applications*, volume 67 of *Lecture Notes in Electrical Engineering*, pages 819–827. Springer.

Geiger, D. and Heckerman, D. (1994). Learning Gaussian Networks. Technical report, Microsoft Research, Redmond, Washington. Available as Technical Report MSR-TR-94-10.

Gentleman, R., Whalen, E., Huber, W., and Falcon, S. (2010). *graph: A Package to Handle Graph Data Structures*. R package version 1.26.0.

Gentry, J., Long, L., Gentleman, R., Falcon, S., Hahne, F., and Sarkar, D. (2010). *Rgraphviz: Provides Plotting Capabilities for R Graph Objects*. R package version 1.26.0.

Goldberg, D. (1991). What Every Computer Scientist Should Know About Floating Point Arithmetic. *ACM Computing Surveys*, 23(1):5–48.

Harary, F. and Palmer, E. M. (1973). *Graphical Enumeration*. Academic Press.

Hasanat, M. A., Ramachandram, D., and Mandava, R. (2010). Bayesian Belief Network Learning Algorithms for Modeling Contextual Relationships in Natural Imagery: a Comparative Study. *Artificial Intelligence Review*, pages 1–18.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition.

Hausser, J. and Strimmer, K. (2009). Entropy Inference and the James-Stein Estimator, with Application to Nonlinear Gene Association Networks. *Journal of Machine Learning Resesearch*, 10:1469–1484.

Heckerman, D., Geiger, D., and Chickering, D. M. (1995). Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning*, 20(3):197–243. Available as Technical Report MSR-TR-94-09.

Hoeffding, W. (1940). Masstabinvariante Korrelationstheorie. *Schriften des Mathematischen Instituts und des Instituts für Angewandte Mathematik der Universität Berlin*, 5(3):179–223.

Holmes, D. E. and Jain, L. C., editors (2008). *Innovations in Bayesian Networks: Theory and Applications*. Springer-Verlag.

Hotelling, H. (1953). New Light on the Correlation Coefficient and Its Transforms. *Journal of the Royal Statistical Society. Series B (Methodological)*, 15(2):193–232.

Ide, J. S. and Cozman, F. G. (2002). Random Generation of Bayesian Networks. In *Proceedings of the 16th Brazilian Symposium on Artificial Intelligence*, pages 366–375. Springer-Verlag.

Imoto, S., Kim, S. Y., Shimodaira, H., Aburatani, S., Tashiro, K., Kuhara, S., and Miyano, S. (2002). Bootstrap Analysis of Gene Networks Based on Bayesian Networks and Nonparametric Regression. *Genome Informatics*, 13:369–370.

James, W. and Stein, C. (1961). Estimation with Quadratic Loss. In Neyman, J., editor, *Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probability*, pages 361–379.

Jensen, F. V. (2001). *Bayesian Networks and Decision Graphs*. Springer.

Johnson, N. L., Kotz, S., and Balakrishnan, N. (1997). *Discrete Multivariate Distributions*. Wiley.

Jungnickel, D. (2008). *Graphs, Networks and Algorithms*. Springer-Verlag, 3rd edition.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Korb, K. and Nicholson, A. (2004). *Bayesian Artificial Intelligence*. Chapman and Hall.

Krämer, N., Schäfer, J., and Boulesteix, A. (2009). Regularized Estimation of Large-Scale Gene Association Networks Using Graphical Gaussian Models. *BMC Bioinformatics*, 10(1):384.

Krummenauer, F. (1998a). Efficient Simulation of Multivariate Binomial and Poisson Distributions. *Biometrical Journal*, 40(7):823–832.

Krummenauer, F. (1998b). Limit Theorems for Multivariate Discrete Distributions. *Metrika*, 47(1):47 – 69.

Kullback, S. (1968). *Information Theory and Statistics*. Dover Publications.

Larrañaga, P., Sierra, B., Gallego, M. J., Michelena, M. J., and Picaza, J. M. (1997). Learning Bayesian Networks by Genetic Algorithms: A Case Study in the Prediction of Survival in Malignant Skin Melanoma. In *Proceedings of the 6th Conference on Artificial Intelligence in Medicine in Europe (AIME'97)*, pages 261–272. Springer.

Lauritzen, S. L. and Spiegelhalter, D. (1988). Local Computation with Probabilities on Graphical Structures and their Application to Expert Systems (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 50(2):157–224.

Ledoit, O. and Wolf, M. (2003). Improved Estimation of the Covariance Matrix of Stock Returns with an Application to Portfolio Selection. *Journal of Empirical Finance*, 10:603–621.

Lee, J. K. (2010). *Statistical Bioinformatics: a Guide for Life and Biomedical Science Researchers*. Wiley.

Legendre, P. (2000). Comparison of Permutation Methods for the Partial Correlation and Partial Mantel Tests. *Journal of Statistical Computation and Simulation*, 67:37–73.

Leonenko, N. and Seleznjev, O. (2010). Statistical Inference for the ϵ-Entropy and the Quadratic Rényi Entropy. *Journal of Multivariate Analysis*, 101(9):1981–1994.

Lin, K., Husmeier, D., Dondelinger, F., Mayer, C. D., Liu, H., Prichard, L., Salmond, G. P. C., Toth, I. K., and Birch, P. R. J. (2010). Reverse Engineering Gene Regulatory Networks Related to Quorum Sensing in the Plant Pathogen Pectobacterium Atrosepticum. In Fenyö, D., editor, *Computational Biology*, pages 253–281. Humana Press.

Loève, M. (1977). *Probability Theory*. Springer-Verlag, 4th edition.

Mardia, K. V., Kent, J. T., and Bibby, J. M. (1979). *Multivariate Analysis*. Academic Press.

Margaritis, D. (2003). *Learning Bayesian Network Model Structure from Data*. PhD thesis, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA. Available as Technical Report CMU-CS-03-153.

Margolin, A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., Favera, R., and Califano, A. (2006). ARACNE: An Algorithm for the Reconstruction of Gene Regulatory Networks in a Mammalian Cellular Context. *BMC Bioinformatics*, 7(Suppl 1):S7.

Mari, D. D. and Kotz, S. (2001). *Correlation and Dependence*. Imperial College Press.

Melançon, G., Dutour, I., and Bousquet-Mélou, M. (2000). Random Generation of DAGs for Graph Drawing. Technical Report INS-R0005, Centre for Mathematics and Computer Sciences, Amsterdam.

Meloni, A., Ripoli, A., Positano, V., and Landini, L. (2009). Improved Learning of Bayesian Networks in Biomedicine. In *Proceedings of the 9th International Conference on Intelligent Systems Design and Applications*, pages 624–628. IEEE Computer Society.

Moors, J. J. A. and Muilwijk, J. (1971). An Inequality for the Variance of a Discrete Random Variable. *Sankhy: The Indian Journal of Statistics, Series B*, 33(3/4):385–388.

Muirhead, R. J. (1982). *Aspects of Multivariate Statistical Theory*. Wiley.

Nagao, H. (1973). On Some Test Criteria for Covariance Matrix. *The Annals of Statistics*, 1(4):700–709.

Nagarajan, R., Datta, S., and Scutari, M. (2011). *Graphical Models in R*. Springer. In preparation.

Nagarajan, R., Datta, S., Scutari, M., Beggs, M. L., Nolen, G. T., and Peterson, C. A. (2010). Functional Relationships Between Genes Associated with Differentiation Potential of Aged Myogenic Progenitors. *Frontiers in Physiology*, 1(21):1–8.

Neapolitan, R. E. (2003). *Learning Bayesian Networks*. Prentice Hall.

Nocedal, J. and Wright, S. J. (1999). *Numerical Optimization*. Springer-Verlag.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Pearl, J. (2009). *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd edition.

Pesarin, F. and Salmaso, L. (2010). *Permutation Tests for Complex Data: Theory, Applications and Software*. Wiley.

R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rauber, T. and Rünger, G. (2010). *Parallel Programming For Multicore and Cluster Systems*. Springer-Verlag.

Rissanen, J. (2007). *Information and Complexity in Statistical Models*. Springer.

Russell, S. J. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition.

Sarkar, D. (2008). *Lattice: Multivariate Data Visualization with R*. Springer.

Schäfer, J. and Strimmer, K. (2005). A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics. *Statistical Applications in Genetics and Molecular Biology*, 4:32.

Schwarz, G. E. (1978). Estimating the Dimension of a Model. *Annals of Statistics*, 6(2):461 – 464.

Scutari, M. (2009). Structure Variability in Bayesian Networks. Working Paper 13-2009, Department of Statistical Sciences, University of Padova. Deposited on arXiv in the Statistics - Methodology archive, available from http://arxiv.org/abs/0909.1685.

Scutari, M. (2010a). *bnlearn: Bayesian Network Structure Learning*. R package version 2.3.

Scutari, M. (2010b). Learning Bayesian Networks with the bnlearn R Package. *Journal of Statistical Software*, 35(3):1–22.

Seber, G. A. F. (2008). *A Matrix Handbook for Stasticians*. Wiley.

Spirtes, P., Glymour, C., and Scheines, R. (2000). *Causation, Prediction, and Search*. MIT Press.

Stein, C. (1956). Inadmissibility of the Usual Estimator for the Mean of a Multivariate Distribution. In Neyman, J., editor, *Proceedings of the 3rd Berkeley Symposium on Mathematical Statistics and Probability*, pages 197–206.

Steyn, H. S. (1978). On Approximations for the Central and Noncentral Distribution of the Generalized Variance. *Journal of the American Statistical Association*, 73(363):670–675.

Tierney, L., Rossini, A. J., Li, N., and Sevcikova, H. (2008). *snow: Simple Network of Workstations*. R package version 0.3-3.

Tsamardinos, I., Aliferis, C. F., and Statnikov, A. (2003). Algorithms for Large Scale Markov Blanket Discovery. In *Proceedings of the 16th International Florida Artificial Intelligence Research Society Conference*, pages 376–381. AAAI Press.

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm. *Machine Learning*, 65(1):31–78.

Verma, T. S. and Pearl, J. (1991). Equivalence and Synthesis of Causal Models. *Uncertainty in Artificial Intelligence*, 6:255–268.

Wishart, J. (1949). Cumulants of Multivariate Multinomial Distributions. *Biometrika*, 36:47–58.

Yaramakala, S. and Margaritis, D. (2005). Speculative Markov Blanket Discovery for Optimal Feature Selection. In *Proceedings of the 5th IEEE International Conference on Data Mining*, pages 809–812. IEEE Computer Society.