An Introduction to Bayesian Networks
Torun, 16th - 18th January, 2013

John M. Noble
Institute of Applied Mathematics and Mechanics
University of Warsaw,
ul. Banacha 2
02-097 Warszawa,
Poland

Contents

Introduction

The models that were later to be called Bayesian networks were introduced into artificial intelligence by J. Pearl in (1982) [41], a seminal article in the literature of that field. A Bayesian network provides a compact representation of a probability distribution that is too large to handle using traditional specifications and provides a systematic and localised method for incorporating probabilistic information about a situation. The description 'Bayesian networks' covers a large field of problems and techniques of data analysis and probabilistic reasoning, where data is collected on a large number of variables and the aim is to factorise the distribution, represent it graphically and exploit the graphical representation. Perhaps the earliest work that explicitly uses directed graphs to represent possible dependencies among random variables is that by S. Wright (1921) [65], developed by the same author in 1934 [66].

Bayesian networks only represent a small part of the wider field of graphical models; a Bayesian network is a probability distribution factorised along a directed acyclic graph (henceforth, written DAG). In many examples this is not the most efficient model for representing the independence structure and there is a wider field of graphical models and learning theory is not dealt with here. The reader is referred to Studený (2005) [61].

Situations where Bayesian networks provide the natural tools for analysis are, for example: computing the overall reliability of a system given the reliability of the individual components and how they interact, system security where Bayesian networks are used as a tool for assessing intrusion evidence and whether a network is under attack and forensic analysis. Further applications are, for example: finding the most likely message that was sent across a noisy channel, restoring a noisy image, mapping genes onto a chromosome. One of the leading applications of techniques from the area is to establishing genome pathways. Given DNA hybridization arrays, which simultaneously measure the expression levels for thousands of genes, a major challenge in computational biology is to uncover, from such measurements, gene/protein interactions and key biological features of cellular systems. This is discussed, for example, by Nir Friedman et. al. (2000) [26]. DAGs have also proved useful in a large number of situations where the graph is constructed along causal principles; parent variables are considered to be direct causes. Bayesian networks offer an alternative to 'naïve Bayes' models of supervised classification in machine learning, which exploits more of the structure.

Chapter 1

Preliminaries: Graph Theory and Probabilistic Independence

1.1 Graph Theory

Definition 1.1 (Graph, Simple Graph)**.** *A* graph $\mathcal{G} = (V, E)$ *consists of a finite set of nodes* V *and an edge set* E*, where each edge is contained in* $V \times V$*. The edge set therefore consists of ordered pairs of nodes.*

If there are d *nodes, then* $V = \{1, \ldots, d\}$ *will denote the indexing set. A graph* $\mathcal{G} = (V, E)$ *is said to be* simple *if* E *does not contain any edges of the form* (j, j) *(that is a loop from the node to itself) and any edge* $(j, k) \in E$ *that appears in* E *does so exactly once. That is, multiple edges are not permitted.*

For any two distinct nodes α *and* $\beta \in V$*, the ordered pair* $(\alpha, \beta) \in E$ *if and only if there is a directed edge from* α *to* β*. An* undirected *edge will be denoted* $\langle \alpha, \beta \rangle$*. In terms of directed edges,*

$$\langle \alpha, \beta \rangle \in E \Leftrightarrow (\alpha, \beta) \in E \qquad \textit{and} \qquad (\alpha, \beta) \in E.$$

For a simple graph that may contain both directed and undirected edges, the edge set E *may be decomposed as* $E = D \cup U$*, where* $D \cap U = \phi$*, the empty set. The sets* U *and* D *are defined by*

$$\langle \alpha, \beta \rangle \in U \Leftrightarrow (\alpha, \beta) \in E \quad \textit{and} \quad (\beta, \alpha) \in E.$$

$$(\alpha, \beta) \in D \Leftrightarrow (\alpha, \beta) \in E \quad \textit{and} \quad (\beta, \alpha) \notin E.$$

For the definitions of 'path', 'trail' and 'cycle', an undirected edge will be considered as a single edge.

All the graphs considered in this course will be simple graphs and the term 'graph' will be used to mean 'simple graph'. If $(i, j) \in D$, this is denoted by an arrow going *from* i *to* j. If $\langle i, j \rangle \in U$, this is denoted by an edge between the two variables i and j.

Definition 1.2 (Parent, Child, Directed and Undirected Neighbour, Family)**.** *Consider a graph* $\mathcal{G} = (V, E)$*, where* $V = \{1, \ldots, d\}$ *and let* $E = D \cup U$*, where* D *is the set of directed edges and* U *the set of undirected edges. Let* $j, k \in V$*. If* $(j, k) \in D$*, then* k *is referred to as a* child *of* j *and* j *as a* parent *of* k*.*

For any node $\alpha \subseteq V$, the set of parents *is defined as*

$$\Pi(\alpha) = \{\beta \in V \mid (\beta, \alpha) \in D\} \tag{1.1}$$

and the set of children *is defined as*

$$Ch(\alpha) = \{\beta \in V \mid (\alpha, \beta) \in D\}. \tag{1.2}$$

For any subset $A \subseteq V$, the set of parents of A is defined as

$$\Pi(A) = \cup_{\alpha \in A}\{\beta \in V \backslash A \mid (\beta, \alpha) \in D\}. \tag{1.3}$$

The set of directed neighbours *of a node α is defined as*

$$N_{(d)}(\alpha) = \Pi(\alpha) \cup Ch(\alpha)$$

and the set of undirected neighbours *of α as*

$$N_{(u)}(\alpha) = \{\beta \in V \mid \langle \alpha, \beta \rangle \in U\}. \tag{1.4}$$

For any subset $A \subseteq V$, the set of undirected neighbours *of A is defined as*

$$N_{(u)}(A) = \cup_{\alpha \in A}\{\beta \in V \backslash A \mid \langle \alpha, \beta \rangle \in U\}. \tag{1.5}$$

For a node α, the set of neighbours $N(\alpha)$ is defined as

$$N(\alpha) = N_{(u)}(\alpha) \cup N_{(d)}(\alpha).$$

The family *of a node β is the set containing the node β together with its parents and undirected neighbours. It is denoted:*

$$F(\beta) = \{\beta\} \cup \Pi(\beta) \cup N_{(u)}(\beta) = \{family\ of\ \beta\}.$$

When $\mathcal{G}$ is undirected, this reduces to $F(\beta) = \{\beta\} \cup N(\beta)$.

The notation Π_j will also be used to denote the parent set of variable j. Similarly with children, family and neighbour.

The notation $\alpha \sim \beta$ will be used to denote that $\alpha \in N(\beta)$; namely, that α and β are neighbours. Note that $\alpha \in N(\beta) \implies \beta \in N(\alpha)$.

In this text, a directed edge (j, k) is indicated by a pointed arrow from j to k; that is, from the parent to the child.

Definition 1.3 (Directed, Undirected Graph)**.** *If* all *edges of a graph are undirected, then the graph $\mathcal{G}$ is said to be* undirected. *If* all *edges are directed, then the graph is said to be* directed. *The* undirected *version of a graph $\mathcal{G}$, denoted by $\tilde{\mathcal{G}}$, is obtained by replacing the directed edges of $\mathcal{G}$ by undirected edges.*

Definition 1.4 (Trail). *Let $\mathcal{G} = (V, E)$ be a graph, where $E = D \cup U$; $D \cap U = \phi$, D denotes the directed edges and U the undirected edges. A* trail *τ between two nodes $\alpha \in V$ and $\beta \in V$ is a collection of nodes $\tau = (\tau_1, \ldots, \tau_m)$, where $\tau_i \in V$ for each $i = 1, \ldots, m$, $\tau_1 = \alpha$ and $\tau_m = \beta$ and such that for each $i = 1, \ldots, m-1$, $\tau_i \sim \tau_{i+1}$. That is, for each $i = 1, \ldots, m-1$, either $(\tau_i, \tau_{i+1}) \in D$ or $(\tau_{i+1}, \tau_i) \in D$ or $\langle \tau_i, \tau_{i+1} \rangle \in U$.*

Definition 1.5 (Sub-graph, Induced Sub-graph). *Let $A \subseteq V$ and $E_A \subseteq E \cap A \times A$. Then $\mathcal{F} = (A, E_A)$ is a* sub graph *of $\mathcal{G}$.*

If $A \subset V$ and $E_A = E \cap A \times A$, then $\mathcal{G}_A = (A, E_A)$ is the sub-graph induced by *A.*

Note that in general it is possible for a sub-graph to contain the same nodes, but fewer edges, but the sub-graph *induced* by the same node set will have the same edges.

Definition 1.6 (Connected Graph, Connected Component). *A graph is said to be* connected *if between any two nodes $j \in V$ and $k \in V$ there is a trail. A* connected component *of a graph $\mathcal{G} = (V, E)$ is an induced sub-graph $\mathcal{G}_A$ such that $\mathcal{G}_A$ is connected and such that if $A \neq V$, then for any two nodes $(\alpha, \beta) \in V \times V$ such that $\alpha \in A$ and $\beta \in V \backslash A$, there is no trail between α and β.*

Definition 1.7 (Path, Directed Path). *Let $\mathcal{G} = (V, E)$ denote a simple graph, where $E = D \cup U$. That is, $D \cap U = \phi$, D denotes the directed edges and U denotes the undirected edges. A* path *of length m from a node α to a node β is a sequence of distinct nodes $(\tau_0, \ldots, \tau_m)$ such that $\tau_0 = \alpha$ and $\tau_m = \beta$ such that $(\tau_{i-1}, \tau_i) \in E$ for each $i = 1, \ldots, m$. That is, for each $i = 1, \ldots, m$, either $(\tau_{i-1}, \tau_i) \in D$, or $\langle \tau_{i-1}, \tau_i \rangle \in U$.*

The path is a directed path *if $(\tau_{i-1}, \tau_i) \in D$ for each $i = 1, \ldots, m$. That is, there are no undirected edges along the directed path.*

It follows that a *trail* in $\mathcal{G}$ is a sequence of nodes that form a path in the undirected version $\tilde{\mathcal{G}}$.

Unlike a trail, a directed path $(\tau_0, \ldots, \tau_m)$ requires that the directed edge $(\tau_i, \tau_{i+1}) \in D$ for all $i = 0, \ldots, m-1$.

Definition 1.8 (Descendant, Ancestor). *Let $\mathcal{G} = (V, E)$ be a graph. A node α is a* descendant *of a node β if and only if there is a directed path from β to α. A node γ is an* ancestor *of a node α if and only if there is a directed path from γ to α.*

Let $E = U \cup D$, where U denotes the undirected edges and D denotes the directed edges. The set of descendants *$D(\alpha)$ of a node α is defined as*

$$D(\alpha) = \{\beta \in V \mid \exists \tau = (\tau_0, \ldots, \tau_k) : \tau_0 = \alpha,\ \tau_k = \beta,\ (\tau_j, \tau_{j+1}) \in D,\ j = 0, 1, \ldots, k\}. \tag{1.6}$$

That is, nodes β such that there is a directed path from α to β.

The set of ancestors *$A(\alpha)$ of a node α is defined as*

$$A(\alpha) = \{\beta \in V \mid \exists \tau = (\tau_0, \ldots, \tau_k) : \tau_0 = \beta,\ \tau_k = \alpha,\ (\tau_j, \tau_{j+1}) \in D,\ j = 0, 1, \ldots, k\}. \tag{1.7}$$

That is, nodes β such that there is a directed path from β to α.

In both cases, the paths are directed; they consist of directed edges only; they do not contain undirected edges.

Definition 1.9 (Cycle). *Let $\mathcal{G} = (V, E)$ be a graph. An m-cycle in $\mathcal{G}$ is a sequence of distinct nodes*

$$\tau_0, \ldots, \tau_{m-1}$$

such that $\tau_0, \ldots, \tau_{m-1}, \tau_0$ is a path (definition 1.7).

Definition 1.10 (Directed Acyclic Graph (DAG)). *A* graph *$\mathcal{G} = (V, E)$ is said to be a* directed acyclic graph *if each edge is* directed *(that is, $\mathcal{G}$ is a simple graph such that for each pair $(\alpha, \beta) \in V \times V$, $(\alpha, \beta) \in E \Rightarrow (\beta, \alpha) \notin E$) and for any node $\alpha \in V$ there does not exist any set of distinct nodes $\tau_1, \ldots, \tau_m$ such that $\alpha \neq \tau_i$ for all $i = 1, \ldots, m$ and $(\alpha, \tau_1, \ldots, \tau_m, \alpha)$ forms a directed path. That is, there are no m-cycles in $\mathcal{G}$ for any $m \geq 1$.*

Definition 1.11 (Tree, Leaf). *A* tree *is a graph $\mathcal{G} = (V, E)$ that is* connected *and such that for any node $\alpha \in V$, there is no trail between α and α and for any two nodes α and β in V with $\alpha \neq \beta$, there is a unique trail. A* leaf *of a tree is a node that is connected to exactly one other node.*

Definition 1.12 (Chord). *Let $\mathcal{G} = (V, E)$ be a graph. Let σ be an n cycle in $\mathcal{G}$. A* chord *of this cycle is a pair (α_i, α_j) of* non consecutive *nodes in σ such that $\alpha_i \sim \alpha_j$ in $\mathcal{G}$.*

Definition 1.13 (Triangulated). *An undirected graph is said to be* triangulated *if every cycle of length ≥ 4 has a chord.*

1.2 Probabilistic Preliminaries

The vector of d interacting random variables will be denoted by $\underline{X} = (X_1, \ldots, X_d)$. Let

$$\mathcal{X}_j = (x_j^{(1)}, \ldots, x_j^{(k_j)})$$

denote the state space for variable X_j and let $\mathcal{X} = \times_{j=1}^{d} \mathcal{X}_j$ denote the product space; the state space for $\underline{X}$. Let

$$\mathbb{P}_{X_1,\ldots,X_d} : \mathcal{X} \to [0, 1]$$

denote a probability function over $\mathcal{X}$. This may also be written as $\mathbb{P}_{\underline{X}}$.

Definition 1.14 (Independence). *Two random vectors $\underline{X}$ and $\underline{Y}$ are independent if their joint probability distribution factorises as*

$$\mathbb{P}_{\underline{X},\underline{Y}} = \mathbb{P}_{\underline{X}} \mathbb{P}_{\underline{Y}}.$$

$\underline{X}$ and $\underline{Y}$ are conditionally independent given a random vector $\underline{Z}$ if

$$\mathbb{P}_{\underline{X},\underline{Y},\underline{Z}} = \mathbb{P}_{\underline{X}|\underline{Z}} \mathbb{P}_{\underline{Y}|\underline{Z}} \mathbb{P}_{\underline{Z}}.$$

This is written $\underline{X} \perp \underline{Y} | \underline{Z}$.

The following characterisations of conditional independence follow trivially from the definition.

Theorem 1.15. *The following are all equivalent to $\underline{X} \perp \underline{Y}|\underline{Z}$: using $\mathcal{X}_X$, $\mathcal{X}_Y$ and $\mathcal{X}_Z$ to denote the state spaces of $\underline{X}$, $\underline{Y}$ and $\underline{Z}$ respectively,*

1. *For all $(\underline{x}, \underline{y}, \underline{z}) \in \mathcal{X}_Z \times \mathcal{X}_Y \times \mathcal{X}_Z$ such that $\mathbb{P}_{\underline{Y}|\underline{Z}}(\underline{y}|\underline{z}) > 0$ and $\mathbb{P}_{\underline{Z}}(\underline{z}) > 0$,*

$$\mathbb{P}_{\underline{X}|\underline{Y},\underline{Z}}(\underline{x}|\underline{y}, \underline{z}) = \mathbb{P}_{\underline{X}|\underline{Z}}(\underline{x}|\underline{z}).$$

2. *There exists a function $a : \mathcal{X}_X \times \mathcal{X}_Z \to [0, 1]$ such that for all $(\underline{x}, \underline{y}, \underline{z}) \in \mathcal{X}_X \times \mathcal{X}_Y \times \mathcal{X}_Z$ satisfying $\mathbb{P}_{\underline{Y},\underline{Z}}(\underline{y}, \underline{z}) > 0$,*

$$\mathbb{P}_{\underline{X}|\underline{Y},\underline{Z}}(\underline{x}|\underline{y}, \underline{z}) = a(\underline{x}, \underline{z})$$

3. *There exist functions $a : \mathcal{X}_X \times \mathcal{X}_Z \to \mathbf{R}_+$ and $b : \mathcal{X}_Y \times \mathcal{X}_Z \to \mathbf{R}_+$ such that for all $(\underline{x}, \underline{y}, \underline{z}) \in \mathcal{X}_X \times \mathcal{X}_Y \times \mathcal{X}_Z$ satisfying $\mathbb{P}_{\underline{Z}}(\underline{z}) > 0$,*

$$\mathbb{P}_{\underline{X},\underline{Y}|\underline{Z}}(\underline{x}, \underline{y}|\underline{z}) = a(\underline{x}, \underline{z})b(\underline{y}, \underline{z})$$

4. *For all $(\underline{x}, \underline{y}, \underline{z}) \in \mathcal{X}_X \times \mathcal{X}_Y \times \mathcal{X}_Z$ such that $\mathbb{P}_{\underline{Z}}(\underline{z}) > 0$,*

$$\mathbb{P}_{\underline{X},\underline{Y},\underline{Z}}(\underline{x}, \underline{y}, \underline{z}) = \frac{\mathbb{P}_{\underline{X},\underline{Z}}(\underline{x}, \underline{z})\mathbb{P}_{\underline{Y},\underline{Z}}(\underline{y}, \underline{z})}{\mathbb{P}_{\underline{Z}}(\underline{z})}.$$

5. *There exist functions $a : \mathcal{X}_X \times \mathcal{X}_Z \to \mathbf{R}_+$ and $b : \mathcal{X}_Y \times \mathcal{X}_Z \to \mathbf{R}_+$ such that*

$$\mathbb{P}_{\underline{X},\underline{Y},\underline{Z}}(\underline{x}, \underline{y}, \underline{z}) = a(\underline{x}, \underline{z})b(\underline{y}, \underline{z}).$$

Proof of theorem 1.15 The proof is trivial and is therefore omitted. □

Chapter 2

Conditional Independence and Graphical Models

Recall that for any collection of events $A_1, \ldots, A_n$,

$$\mathbb{P}(A_1 \cap \ldots \cap A_n) = \mathbb{P}(A_1)\mathbb{P}(A_2|A_1)\ldots\mathbb{P}(A_n|A_1 \cap \ldots \cap A_{n-1}).$$

Let $\underline{X} = (X_1, \ldots, X_d)$ and let $\mathcal{X}_j$ denote the state space of X_j for $j = 1, \ldots, d$. Let $\mathcal{X} = \times_{j=1}^{d} \mathcal{X}_j$ denote the state space of $\underline{X}$. Clearly, any probability distribution $\mathbb{P}_{X_1,\ldots,X_d}$ over $\mathcal{X}$ may be factorised as

$$\mathbb{P}_{X_1,\ldots,X_d} = \mathbb{P}_{X_{\sigma(1)}} \prod_{j=2}^{d} \mathbb{P}_{X_{\sigma(j)}|X_{\sigma(1)},\ldots,X_{\sigma(j-1)}}$$

for any permutation σ of $1, \ldots, d$. Let $\Pi_j^{(\sigma)} \subset \{X_{\sigma(1)}, \ldots, X_{\sigma(j-1)}\}$ satisfy

- $$X_{\sigma(j)} \perp \{X_{\sigma(1)}, \ldots, X_{\sigma(j-1)}\} \backslash \Pi_j^{(\sigma)} | \Pi_j^{(\sigma)}$$
- $$X_{\sigma(j)} \not\perp \{X_{\sigma(1)}, \ldots, X_{\sigma(j-1)}\} \backslash \Theta_j | \Theta_j$$

  for any strict subset $\Theta_j \subset \Pi_j^{(\sigma)}$.

Then, setting $\mathbb{P}_{X_{\sigma(j)}|\Pi_j^{(\sigma)}} = \mathbb{P}_{X_{\sigma(j)}}$ when $\Pi_{\sigma(j)} = \phi$ the empty set,

$$\mathbb{P}_{X_1,\ldots,X_d} = \prod_{j=1}^{d} \mathbb{P}_{X_{\sigma(j)}|\Pi_j^{(\sigma)}}.$$

Definition 2.1 (Factorisation, Bayesian Network)**.** *A* factorisation *of a probability distribution is a decomposition*

$$\mathbb{P}_{X_1,\ldots,X_d} = \prod_{j=1}^{d} \mathbb{P}_{X_{\sigma(j)}|\Xi_j^{(\sigma)}} \tag{2.1}$$

such that for each $j \in \{1,\dots,d\}$, $\Xi_j^{(\sigma)} \subseteq \{X_{\sigma(1)},\dots,X_{\sigma(j-1)}\}$. *A* Bayesian Network *is a factorisation of a probability distribution*

$$\mathbb{P}_{X_1,\dots,X_d} = \prod_{j=1}^{d} \mathbb{P}_{X_{\sigma(j)}|\Pi_j^{(\sigma)}} \tag{2.2}$$

such that

1. $\Pi_1^{(\sigma)} = \phi$ *(the empty set)*
2. $\Pi_j^{(\sigma)} \subseteq \{X_{\sigma(1)},\dots,X_{\sigma(j-1)}\}$
3. $X_{\sigma(j)} \perp \{X_{\sigma(1)},\dots,X_{\sigma(j-1)}\}\backslash\Pi_j^{(\sigma)} | \Pi_j^{(\sigma)}$
4. *For any strict subset* $\Theta_j \subset \Pi_j^{(\sigma)}$ *of* $\Pi_j^{(\sigma)}$,

$$X_{\sigma(j)} \not\perp \{X_{\sigma(1)},\dots,X_{\sigma(j-1)}\}\backslash\Theta_j | \Theta_j,$$

together with the directed acyclic graph with node set $V = \{X_1,\dots,X_d\}$ *and directed edges from each variable in* $\Pi_j^{(\sigma)}$ *to* $X_{\sigma(j)}$.

Unless otherwise stated, it will be assumed that the variables are labelled in such a way that $\sigma = I$, the identity.

For $\Pi_j = \{X_{l_{j,1}},\dots,X_{l_{j,m_j}}\}$, the state space of Π_j is $\mathcal{X}_{l_{j,1}} \times \dots \times \mathcal{X}_{l_{j,m_j}}$ and there are $q_j = \prod_{a=1}^{m_j} k_{l_{j,a}}$ configurations. These may be labelled $(\pi_j^{(l)})_{l=1}^{q_j}$. The parameters required for the probability distribution $\mathbb{P}_{X_1,\dots,X_d}$ are

$$\theta_{jil} = \mathbb{P}_{X_j|\Pi_j}(x_j^{(i)}|\pi_j^{(l)}) \quad j = 1,\dots,d \quad i = 1,\dots,k_j, \quad l = 1,\dots,q_j.$$

2.1 Directed Acyclic Graphs and Probability Distributions

The factorisation of equation (2.1) definition 2.1 may be represented by a *Directed Acyclic Graph*. For example, if the probability distribution over X, Y, Z, W satisfies

$$\mathbb{P}_{X,Y,Z,W} = \mathbb{P}_X \mathbb{P}_{Y|X} \mathbb{P}_{Z|X} \mathbb{P}_{W|Y,Z},$$

the factorisation may be represented by the graph in figure 2.1.
Now consider a random vector $\underline{X} = (X_1,\dots,X_d)$.

Definition 2.2 (Factorisation along a Directed Acyclic Graph). *A decomposition of a probability distribution over a set of variables* $V = \{X_1,\dots,X_d\}$ *which satisfies equation (2.1) with respect to an ordering* σ *is said to factorise according to a directed acyclic graph* (V, D) *if* V *is the node set of the graph and for each* $j = 1,\dots,d$, $\Xi_j^{(\sigma)}$ *is the parent set for node* $X_{\sigma(j)}$. *The factorisation corresponds to a Bayesian network if* $\Xi_j^{(\sigma)} = \Pi_j^{(\sigma)}$ *where* $X_{\sigma(j)} \not\perp \{X_{\sigma(1)},\dots,X_{\sigma(j-1)}\}\backslash\Theta_j | \Theta_j$ *for any strict subset* $\Theta_j \subset \Pi_j^{(\sigma)}$.

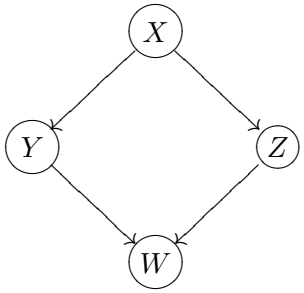

Figure 2.1: DAG representing the factorisation of a probability distribution

Figure 2.2: A Chain Connection

2.1.1 Connections in a Directed Acyclic Graph and Conditional Independence

Definition 2.3 (Instantiated)**.** *When the state of variable is known, the variable is said to be* instantiated.

Within a directed acyclic graph, there are three basic ways in which two nodes X_1, X_3 such that $(X_1, X_3) \notin D$ and $(X_1, X_3) \notin D$ can be connected via a third node. They are the *chain*, *fork* and *collider* connections respectively.

Chain Connections A chain connection between nodes X_1 and X_3 is a connection via a node X_2 such that the graph contains directed edges $X_1 \to X_2$ and $X_2 \to X_3$, but no edge between X_1 and X_3.

Consider a probability distribution over (X_1, X_2, X_3) factorised according to the graph in figure 2.2, as $\mathbb{P}_{X_1}\mathbb{P}_{X_2|X_1}\mathbb{P}_{X_3|X_2}$.
Clearly, $X_1 \not\perp X_3$ in general;

$$\mathbb{P}_{X_1,X_3}(x_1, x_3) = \mathbb{P}_{X_1}(x_1) \sum_{x_2 \in \mathcal{X}_2} \mathbb{P}_{X_2|X_1}(x_2|x_1)\mathbb{P}_{X_3|X_2}(x_3|x_2)$$

and this cannot be expressed in product form.

Conditioned on the instantiation $X_2 = x_2$,

$$\begin{aligned}\mathbb{P}_{X_1,X_3|X_2}(.,.|x_2) &= \frac{\mathbb{P}_{X_1,X_2,X_3}(.,x_2,.)}{\mathbb{P}_{X_2}(x_2)} = \frac{\mathbb{P}_{X_1}(.)\mathbb{P}_{X_2|X_1}(x_2|.)\mathbb{P}_{X_3|X_2}(.|x_2)}{\mathbb{P}_{X_2}(x_2)} \\ &= \left(\frac{\mathbb{P}_{X_1}(.)\mathbb{P}_{X_2|X_1}(x_2|.)}{\mathbb{P}_{X_2}(x_2)}\right)\left(\mathbb{P}_{X_3|X_2}(.|x_2)\right) = \left(\mathbb{P}_{X_1|X_2}(.|x_2)\right)\left(\mathbb{P}_{X_3|X_2}(.|x_2)\right)\end{aligned}$$

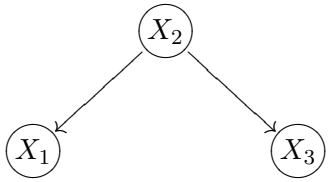

Figure 2.3: A Fork Connection

where Bayes rule has been used and so, following characterisation 3 of conditional independence from theorem 1.15, $X_1 \perp X_3 | X_2$.

Fork Connections A *fork* connection between two nodes X_1 and X_3 is a situation where there is no edge between X_1 and X_3, but there is a node X_2 such that the graph contains directed edges $X_2 \mapsto X_1$ and $X_2 \mapsto X_3$. It is illustrated in Figure 2.3.

A distribution over the variables (X_1, X_2, X_3) that factorises according to the DAG in figure 2.3 has factorisation

$$\mathbb{P}_{X_1,X_2,X_3} = \mathbb{P}_{X_2}\mathbb{P}_{X_1|X_2}\mathbb{P}_{X_3|X_2}.$$

It is clear that $X_1 \not\perp X_3$ in general;

$$\mathbb{P}_{X_1,X_3}(x_1, x_3) = \sum_{x_2 \in \mathcal{X}_2} \mathbb{P}_{X_2}(x_2)\mathbb{P}_{X_1|X_2}(x_1|x_2)\mathbb{P}_{X_3|X_2}(x_3|x_2)$$

and this cannot be expressed in product form. But conditioned on X_2,

$$\mathbb{P}_{X_1,X_3|X_2} = \frac{\mathbb{P}_{X_1,X_3,X_2}}{\mathbb{P}_{X_2}} = \frac{\mathbb{P}_{X_2}\mathbb{P}_{X_1|X_2}\mathbb{P}_{X_3|X_2}}{\mathbb{P}_{X_2}} = \mathbb{P}_{X_1|X_2}\mathbb{P}_{X_3|X_2}.$$

It follows that $X_1 \perp X_3 | X_2$ following characterisation 3) from the characterisations of conditional independence listed in the statement of theorem 1.15.

Collider Connections A *collider connection* between two nodes X_1 and X_3 is a connection such that the graph does not contain an edge between X_1 and X_3, but there is a node X_2 such that the graph contains directed edges $X_1 \mapsto X_2$ and $X_3 \mapsto X_2$. A collider connection is illustrated in figure 2.4.

The factorisation of the distribution $\mathbb{P}_{X_1,X_2,X_3}$ corresponding to the DAG for the collider is

$$\mathbb{P}_{X_1,X_2,X_3} = \mathbb{P}_{X_1}\mathbb{P}_{X_3}\mathbb{P}_{X_2|X_1,X_3}.$$

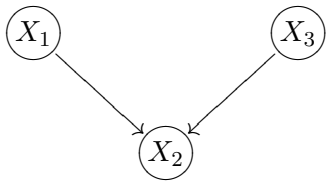

Figure 2.4: A Collider Connection

In general, $X_1 \not\perp X_3|X_2$. But for each $(x, z) \in \mathcal{X}_1 \times \mathcal{X}_3$,

$$\begin{aligned}\mathbb{P}_{X_1,X_3}(x,z) &= \sum_{y\in\mathcal{X}_2} \mathbb{P}_{X_1}(x)\mathbb{P}_{X_3}(z)\mathbb{P}_{X_2|X_1,X_3}(y|x,z) \\ &= \mathbb{P}_{X_1}(x)\mathbb{P}_{X_3}(z) \sum_{y\in\mathcal{X}_2} \mathbb{P}_{X_1|X_2,X_3}(y|x,z) \\ &= \mathbb{P}_{X_1}(x)\mathbb{P}_{X_3}(z).\end{aligned}$$

so that $X_1 \perp X_3$.

A Causal Interpretation So far, the discussion has considered sets of random variables where, based on the ordering of the variables, the parent set of a variable is a subset of those of a lower order. The representation of a probability distribution by factorising along a Directed Acyclic Graph may be particularly useful if there are cause to effect relations between the variables, the ancestors being the cause and the descendants the effect. For a causal model, the connections have the following interpretations:

Fork Connection: Common cause For the fork connection, illustrated by figure 2.2, X_2 may be a *cause* that influences both X_1 and X_3 which are *effects*. The variables are only related through X_2. The situation is illustrated by the following example, taken from a cartoon by Albert Engström; 'during a convivial discussion at the bar one evening, about the unhygienic nature of galoshes, one of the participants pipes up, "you have a very good point there. Every time I wake up wearing my galoshes, I have a sore head."

Let X_1 denote the state of the feet and X_3 the state of the head. These two variables are related; $X_1 \not\perp X_3$. But there is a common cause; X_2, which denotes the activities of the previous evening. Once it is known that he has spent a convivial evening drinking, the state of the feet gives no further information about the state of the head; $X_1 \perp X_3|X_2$.

Chain Connection This may similarly be understood as cause to effect. X_1 influences X_2, which in turn influences X_3, but there is no direct causal relationship between the values taken by X_1 and those taken by X_3. If X_2 is unknown, then $X_1 \not\perp X_3$, but once the state of X_2 is established, X_1 and X_3 give no further information about each other; $X_1 \perp X_3|X_2$.

Collider Connection For the collider connection, X_1 and X_3 are unrelated; $X_1 \perp X_3$. But they both influence X_2. For example, consider a burglar alarm (X_2) that is activated if a burglary takes place, but can also be activated if there is a minor earth tremor.

One day, somebody calls you while you are at work to say that your burglar alarm is activated. You get into the car to go home. But on the way home, you hear on the radio that there has been an earth tremor in the area. As a result, you return to work.

Once X_2 is instantiated, the information that there has been an earth tremor influences the likelihood that a burglary has taken place; $X_1 \not\perp X_3 | X_2$.

This is known as *explaining away*.

Definition 2.4 (S-Active Trail)**.** *Let $\mathcal{G} = (V, E)$ be a directed acyclic graph. Let $S \subset V$ and let $X, Y \in V \backslash S$. A trail τ between the two variables X and Y is said to be* S-active *if*

1. *Every collider node in τ is in S, or has a descendant (definition 1.8) in S.*
2. *Every other node is outside S.*

Definition 2.5 (Blocked Trail)**.** *A trail between X and Y that is not S-active is said to be* blocked by *S.*

The following definition is basic.

Definition 2.6 (d-separation)**.** *Let $\mathcal{G} = (V, E)$ be a directed acyclic graph, where $V = \{X_1, \ldots, X_d\}$ is a collection of random variables. Let $S \subset V$ such that all the variables in S are instantiated and all the variables in $V \backslash S$ are not instantiated. Two distinct variables X_i and X_j not in S are* d-separated *by S if all trails between X_i and X_j are blocked by S.*

Let C and D denote two sets *of variables. If every trail from any variable in C to any variable in D is blocked by S, then the sets C and D are said to be d-separated by S. This is written*

$$C \perp\!\!\!\perp D \,\|_{\mathcal{G}}\, S. \tag{2.3}$$

The set S blocks every path between C and D.

The terminology d-separation is short for *directed* separation. The insertion of the letter 'd' points out that this is not the standard use of the term 'separation' found in graph theory.

Definition 2.7 (d-connected)**.** *If two variables X and Y are not d-separated, they are said to be d-connected.*

It is clear that if all the parents of a variable X and all the children of X and all the variables sharing a child with X are instantiated, then X is d-separated from the rest of the network. This set of variables is known as the *Markov blanket* of the variable X.

Definition 2.8 (Markov Blanket)**.** *The* Markov blanket *of a variable X is the set consisting of the parents of X, the children of X and the variables sharing a child with X.*

For the DAG in figure 2.2, $X_1 \perp\!\!\!\perp X_3 \|_{\mathcal{G}} X_2$ while $X_1 \not\perp\!\!\!\perp X_3 \|_{\mathcal{G}} \phi$. For the DAG in figure 2.3, $X_1 \perp\!\!\!\perp X_3 \|_{\mathcal{G}} X_2$ while $X_1 \not\perp\!\!\!\perp X_3 \|_{\mathcal{G}} \phi$. For figure 2.4, $X_1 \perp\!\!\!\perp X_3 \|_{\mathcal{G}} \phi$ while $X_1 \not\perp\!\!\!\perp X_3 \|_{\mathcal{G}} X_2$. The notation $\not\perp\!\!\!\perp$ denotes d-connected. These statements correspond to the conditional independence statements derived.

2.1.2 Bayes Ball

The *Bayes ball* provides a convenient method for deciding whether or not two nodes are d-separated. Variables are d-connected if the Bayes ball can be passed between them employing the following rule. The uninstantiated (hidden) nodes are depicted as unshaded; instantiated nodes as shaded. In the case of collider connection, unshaded means that neither the node nor any of its descendants is instantiated, while shaded means that the node or one of its descendants is instantiated.

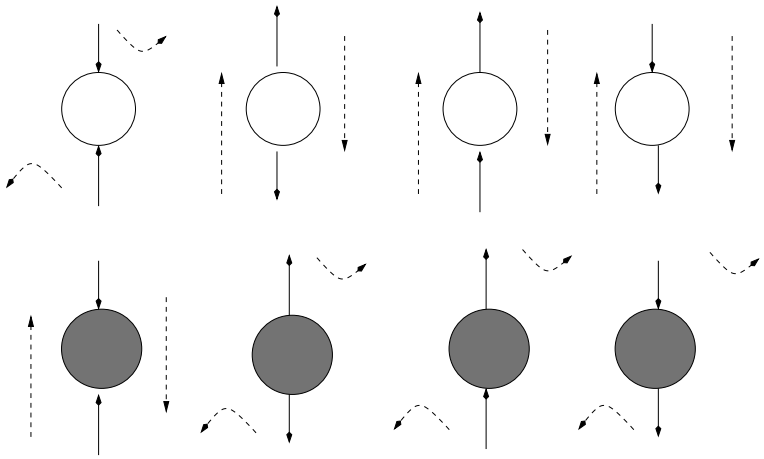

Figure 2.5: Bayes Ball

Consider the three types of connection in a DAG; chain, collider and fork.

- For the *chain* connection illustrated in figure 2.2, the Bayes ball algorithm indicates that *if* the node is instantiated, *then* the ball does not move from X_1 to X_3 through X_2. The communication in the trail is *blocked*. If the node is *not* instantiated, then communication is possible.

- For the *fork* connection illustrated in figure 2.3, the algorithm states that *if* node X_1 is instantiated, then again communication between X_2 and X_3 is *blocked*. If the node is *not* instantiated, then communication is possible.

- For the *collider* connection illustrated in figure 2.4, the Bayes ball algorithm states that the ball *does* move from X_2 to X_3 if node X_1 or any of its descendants is instantiated. Instantiation of X_1 or a descendant opens communication between the parents.

For a collider node X_1, instantiation of any of the descendants of X_1 *also* opens communication. If node X_1 is uninstantiated, *and* none of its descendants is instantiated, then there is no communication.

The Markov blanket satisfies the following property:

Proposition 2.9. *Let A be a variable in a DAG. If all the variables in the* Markov blanket *of A are instantiated, then A is d-separated from the remaining uninstantiated variables.*

Proof The *Markov blanket* is the set of parents of A, children of A and any variables sharing a child with A. Consider the 'Bayes Ball' algorithm, started at A. The ball cannot travel through an instantiated chain or fork connection, nor can it travel through a collider, where none of the descendants are instantiated. Otherwise, it can travel through a node along the graph.

Therefore if all variables in the Markov blanket are instantiated, Bayes ball cannot pass through any of the parents (by definition, the connection is necessarily chain or fork). It cannot pass through a child to any offspring of the child (the connection necessarily chain). If it passes through an instantiated child to another parent of the instantiated child, it cannot pass further: connection at the point of the instantiated parent of the instantiated child is either chain or fork. □

2.2 d-Separation and Conditional Independence

The following result shows that if a probability distribution factorises along a given DAG $\mathcal{G}$, then every d-separation statement for the DAG implies the corresponding conditional independence statement for the distribution.

Theorem 2.10 (d-Separation Implies Conditional Independence)**.** *Let $\mathcal{G} = (V, E)$ be a directed acyclic graph and let $\mathbb{P}$ be a probability distribution that factorises along $\mathcal{G}$. Then for any three disjoint subsets $A, B, S \subset V$, it holds that $A \perp B|S$ (A and B are independent given S) if $A \perp\!\!\!\perp B \| _{\mathcal{G}} A$ (A and B are d-separated by S).*

Proof of theorem 2.10 Let $V = \{X_1, \ldots, X_d\}$ denote the set of variables, let $A \subset V$, $B \subset V$ and $C \subset V$ be three disjoint sets of variables and let $\tilde{A}$, $\tilde{B}$, $\tilde{C}$, $\tilde{V}$ denote the indexing sets of the variables in A, B, C and V respectively. For any set $W \subseteq V$ of variables, let $\tilde{W}$ denote the set of indices. Suppose that $A \perp\!\!\!\perp B \| _{\mathcal{G}} S$. Let A, B and S denote also the random vectors $\underline{X}_A$, $\underline{X}_B$ and $\underline{X}_S$ of variables in A, B and S respectively and let $\mathcal{X}_A$, $\mathcal{X}_B$ and $\mathcal{X}_S$ denote their respective state spaces. It is required to show that for all $\underline{a} \in \mathcal{X}_A$, $\underline{b} \in \mathcal{X}_B$ and $\underline{s} \in \mathcal{X}_S$,

$$\mathbb{P}_{A,B|S}(\underline{a}, \underline{b}|\underline{s}) = \mathbb{P}_{A|S}(\underline{a}|\underline{s})\mathbb{P}_{B|S}(\underline{b}|\underline{s}).$$

Let $D = V \backslash (A \cup B \cup S)$. Let

$$E_1 = \{Y \in V | \text{there is an } S\text{-active trail from } A \text{ to } Y\}$$

$$E_2 = \{Y \in V | \text{there is an } S\text{-active trail from } B \text{ to } Y\}$$

$$D_1 = D \cap E_1 \cap E_2, \qquad D_2 = D \cap E_1 \cap E_2^c, \qquad D_3 = D \cap E_2 \cap E_1^c, \qquad D_4 = D \cap (D_1^c \cup D_2^c \cup D_3^c).$$

From characterisation 5 of theorem 1.15, it is required to show that there are two functions F and G such that

$$\mathbb{P}_{A,B,S}(\underline{a}, \underline{b}, \underline{s}) = F(\underline{a}, \underline{s})G(\underline{b}, \underline{s}).$$

Let $\mathbb{P}(X_j|\Pi_j)$ denote the conditional probability function of X_j given the parent variables Π_j. Then

$$\mathbb{P}(X_1,\ldots,X_d) = \prod_{j\in\tilde{A}} \mathbb{P}(X_j|\Pi_j) \prod_{j\in\tilde{B}} \mathbb{P}(X_j|\Pi_j) \prod_{j\in\tilde{C}} \mathbb{P}(X_j|\Pi_j) \times \prod_{j\in\tilde{D}_1} \mathbb{P}(X_j|\Pi_j) \prod_{j\in\tilde{D}_2} \mathbb{P}(X_j|\Pi_j) \prod_{j\in\tilde{D}_3} \mathbb{P}(X_j|\Pi_j) \prod_{j\in\tilde{D}_4} \mathbb{P}(X_j|\Pi_j).$$

Since all the nodes of D are uninstantiated and there is no S-active trail from A to B, it follows that any nodes in D_1 are *colliders* both with ancestors (definition 1.8) that are either in A or ancestors of A (there is a trail with only fork or chain connections to a variable in A) and similarly with B, together with all the descendants (definition 1.8) of these colliders. Neither these colliders nor their descendants are instantiated (that is, belong to S); neither do the nodes in D_1 have descendants that belong to either A or B, otherwise it follows from the definitions that there would be an active trail between A and B. Therefore, any descendant of a variable in D_1 is also in D_1. Marginalising over the variables in D_1 does not involve the parent variables of A, B or S, nor does it involve the variables in D_2 or D_3 or their ancestors since $\sum_{\mathcal{X}_j} \mathbb{P}(X_j|\Pi_j) = 1$.

There is no S-active trail from a variable in D_4 to any variable in A or B. It follows that parents of variables in D_4 are either in D_4 or in S; if the parent is not in S and there is an S-active trail between the parent and a variable in either A or B, then there is an S-active trail from the variable itself; the additional link is either an uninstantiated fork or uninstantiated chain connection.

Now, using ϕ to denote the empty set, let $S_2 = \{X \in S \mid \Pi(X) \cap D_2 \neq \phi\}$ (there is an S-active trail from A to a parent of $X \notin S$ but not from B to a parent of $X \notin S$), $S_3 = \{X \in S \mid \Pi(X) \cap D_3 \neq \phi\}$ (there is an S-active trail from B to a parent of $X \notin S$ but not from A to a parent of $X \notin S$) and $S_4 = S \cap S_2^c \cap S_3^c$ (nodes $X \in S$ such that there is no S-active trail either from A to a parent of $X \notin S$ or from B to a parent of $X \notin S$). Then $S_2 \cap S_3 = \phi$, the empty set, otherwise there would be a collider node in S that would result in an active trail from A to B.

It is also clear that $\Pi(S_4) \subseteq S \cup D_4$, where $\Pi(S_4)$ denotes the parent variables of the variables in S_4; that is, $\Pi(S_4) = \{Y|(Y,X) \in E, \quad X \in S_4\}$. The sets S_2, S_3, S_4 are disjoint. Using $\mathcal{X}_W$ to denote the state space of the random vector formed from the variables in a set W, it follows that

$$\begin{aligned}\mathbb{P}(A,B,S) \quad = \quad & \sum_{\mathcal{X}_{D_2}} \sum_{\mathcal{X}_{D_3}} \sum_{\mathcal{X}_{D_4}} \sum_{\mathcal{X}_{D_1}} \left(\prod_{j\in\tilde{D}_1} \mathbb{P}(X_j|\Pi_j) \prod_{j\in\tilde{D}_4} \mathbb{P}(X_j|\Pi_j) \prod_{j\in\tilde{S}_4} \mathbb{P}(X_j|\Pi_j) \right) \\ & \times \left(\prod_{j\in\tilde{A}} \mathbb{P}(X_j|\Pi_j) \prod_{j\in\tilde{S}_2} \mathbb{P}(X_j|\Pi_j) \prod_{j\in\tilde{D}_2} \mathbb{P}(X_j|\Pi_j) \right) \\ & \times \left(\prod_{j\in\tilde{B}} \mathbb{P}(X_j|\Pi_j) \prod_{j\in\tilde{S}_3} \mathbb{P}(X_j|\Pi_j) \prod_{j\in\tilde{D}_3} \mathbb{P}(X_j|\Pi_j) \right).\end{aligned}$$

The sums are taken from right to left, starting with $\sum_{\mathcal{X}_{D_1}}$. None of the variables in D_1 are in $A \cup B \cup S$ or the parent sets of variables in A, B, S_2, S_3, D_2 or D_3. The parents of variables in D_4 are either

in D_4 or S, the parents of variables in S_4 are either in S_4 or D_4. The parents of variables in D_3 in $D_3 \cup B$. The parents of variables in D_2 are in $D_2 \cup A$. It follows that $\mathbb{P}(A, B, S)$ has a factorisation

$$\mathbb{P}(A, B, S) = \psi_1(S)\psi_2(A, S)\psi_3(B, S)$$

where the constructions of ψ_1, ψ_2 and ψ_3 are clear from the context. This factorisation clearly satisfies the required criteria. It follows that d-separation implies conditional independence. □

Of course, the converse is not true in general; d-separation is a convenient way of locating some of the independence structure of a distribution. It does not, in general, locate the entire independence structure.

2.3 The Locally Directed Markov Property

This section introduces the *local directed Markov condition*, a necessary and sufficient condition so that a probability function $\mathbb{P}$ over a set of variables V can be factorised along a graph $\mathcal{G}$.

Definition 2.11 (Local Directed Markov Condition, Locally $\mathcal{G}$ - Markovian)**.** *Let $V = \{X_1, \ldots, X_d\}$ be a set of random variables. A probability function $\mathbb{P}$ over the random vector $\underline{X} = (X_1, \ldots, X_d)$ satisfies the* local directed Markov condition *with respect to a DAG $\mathcal{G} = (V, D)$ or, equivalently, is said to be* locally $\mathcal{G}$-Markovian *if and only if there is an ordering of the variables σ such that $\Pi_j^{(\sigma)} \in \{X_{\sigma(1)}, \ldots, X_{\sigma(j-1)}\}$ for each $j \in \{1, \ldots, d\}$ and such that $X_{\sigma(j)}$ is conditionally independent, given $\Pi_j^{(\sigma)}$ (the set of parents of $X_{\sigma(j)}$) of all the variables in the set $V \backslash (V_j^{(\sigma)} \cup \Pi_j^{(\sigma)})$, where $V_j^{(\sigma)}$ is the set of all* descendants *of $X_{\sigma(j)}$. That is,*

$$V_j^{(\sigma)} = \{Y \in V | \textit{there is a directed path from } X_{\sigma(j)} \textit{ to } Y\}. \tag{2.4}$$

The condition may be expressed as follows:

$$X_{\sigma(j)} \perp V \backslash (V_j^{(\sigma)} \cup \Pi_j^{(\sigma)} \cup \{X_{\sigma(j)}\}) | \Pi_j^{(\sigma)}. \tag{2.5}$$

Proposition 2.12. *Let $\mathbb{P}$ be a probability distribution over a set of variables $V = \{X_1, \ldots, X_d\}$. Then $\mathbb{P}$ satisfies the l.d.m.p. with respect to a graph $\mathcal{G} = (V, D)$ if and only if there is an ordering of the variables σ such that $\mathbb{P}$ factorises along $\mathcal{G}$.*

Proof Firstly, assume that $\mathbb{P}$ is locally $\mathcal{G}$-Markovian and assume that the variables are ordered in such a way that for each $j \in \{1, \ldots, d\}$, $X_j \perp V \backslash (V_j \cup \Pi_j) | \Pi_j$ where V_j is defined in equation (2.4). Let $\pi_j(x_1, \ldots, x_{j-1})$ denote the instantiation of Π_j when $\underline{X}$ is instantiated as $(x_1, \ldots, x_d)$. Then for all $j = 1, \ldots, d$ and any π_j such that $\mathbb{P}_{\Pi_j}(\pi_j) > 0$,

$$\mathbb{P}_{X_j | X_1, \ldots, X_{j-1}}(x_j | x_1, \ldots, x_{j-1}) = \mathbb{P}_{X_j | \Pi_j}(x_j | \pi_j)$$

with $\mathbb{P}_{X_j | X_1, \ldots X_{j-1}}(x_j | x_1, \ldots, x_{j-1}) = \mathbb{P}_{X_j}(x_j)$ if $\Pi_j = \phi$. It follows directly that

$$\mathbb{P}_{X_1,\ldots,X_d} = \prod_{j=1}^{d} \mathbb{P}_{X_j|\Pi_j}$$

and hence, by definition, that $\mathbb{P}$ factorises along $\mathcal{G}$.

Secondly, suppose that $\mathbb{P}$ factorises along a graph $\mathcal{G} = (V, D)$. Then it is clear (for example by using the Bayes ball algorithm) that

$$X_j \perp\!\!\!\perp V \backslash (V_j \cup \Pi_j) \| _{\mathcal{G}} \Pi_j$$

where V_j is the set of variables defined by equation 2.4. If Π_j is instantiated, then any trail from X_j to a variable in $V \backslash (V_j \cup \Pi_j \cup \{X_j\})$ has to pass through a node in Π_j, which will be either a chain or fork connection. It follows from theorem 2.10 that

$$X_j \perp V \backslash (V_j \cup \Pi_j \cup \{X_j\}) | \Pi_j,$$

from which it follows that $\mathbb{P}$ is locally $\mathcal{G}$-Markovian. □

Chapter 3

Markov models and Markov equivalence

3.1 I-maps and Markov equivalence

If a probability distribution factorises according to a directed acyclic graph, then any d-separation statement in the graph implies the corresponding conditional independence statement for the distribution. A distribution that has all the conditional independence statements corresponding to the entire set of d-separation statements for a DAG $\mathcal{G}$ is said to belong to the *Markov model* of $\mathcal{G}$.

Definition 3.1 (Markov Model). *Let $V = \{X_1, \ldots, X_d\}$ denote a set of variables and let $\mathcal{G} = (V, D)$ be a directed acyclic graph. Let $\mathcal{V}$ denote the entire set of subsets of V. Let $\mathbb{P}$ be a probability function for the random vector $\underline{X} = (X_1, \ldots, X_d)$. Let*

$$\mathcal{I}(\mathbb{P}) = \{(X, Y, S) \in V \times V \times \mathcal{V} | \quad X \neq Y, \quad X, Y \notin S, \qquad X \perp Y | S\}. \tag{3.1}$$

Note that $\phi \in \mathcal{V}$ and $X \perp Y | \phi$ means that $X \perp Y$.

The Markov Model *$\mathcal{M}_{\mathcal{G}}$ determined by a directed acyclic graph $\mathcal{G} = (V, D)$ is the set of triples $(X, Y, S) \in V \times V \times \mathcal{F}$ such that the d-separation statement $X \perp\!\!\!\perp Y \|_{\mathcal{G}} S$ holds in the DAG. That is,*

$$\mathcal{M}_{\mathcal{G}} = \{(X, Y, S) \in V \times V \times \mathcal{V} | X \perp\!\!\!\perp Y \|_{\mathcal{G}} S\}. \tag{3.2}$$

That is, the Markov model *is the set of conditional independence relations satisfied by all distributions that are locally $\mathcal{G}$-Markovian.*

A distribution $\mathbb{P}$ is said to belong to the Markov Model of $\mathcal{G}$, written $\mathbb{P} \in \mathcal{M}_{\mathcal{G}}$, if and only if $\mathcal{M}_{\mathcal{G}} \subseteq \mathcal{I}(\mathbb{P})$.

The collection of triples $\mathcal{M}_{\mathcal{G}}$ defined in equation (3.2) definition 3.1 represents the entire set of conditional independence statements that it is possible to infer from the DAG, but this collection does not necessarily represent the complete set of independence statements that hold for a collection of variables under a given probability distribution. When it does, it is known as a *perfect I-map*.

Definition 3.2 (Perfect I-Map, Faithful). *A DAG $\mathcal{G} = (V, D)$ over a set of variables V is known as a* perfect I-map *for a probability function $\mathbb{P}$ over V if for any three disjoint subsets of variables A, B*

and S,

$$A \perp B|S \Leftrightarrow A \perp\!\!\!\perp B \parallel_{\mathcal{G}} S,$$

If $\mathcal{G}$ *is a perfect* I*-map for* $\mathbb{P}$*, then* $\mathcal{G}$ *is said to be* faithful *to* $\mathbb{P}$.

Since it is the collection of independence statements that is in view, rather than the probability distribution $\mathbb{P}$ itself, the term *consistency* is used to mean exactly the same thing.

Definition 3.3 (Consistency)**.** *Let* $V = \{X_1, \ldots, X_d\}$ *be a collection of random variables and let* $\mathcal{I}(\mathbb{P})$ *denote the entire collection of conditional independence statements (defined by equation (3.1)) of the distribution* $\mathbb{P}$*. A Directed Acyclic Graph* $\mathcal{G} = (V, D)$ *is* consistent *with a set of conditional independence statements* $\mathcal{I}(\mathbb{P})$ *if and only if*

$$\mathcal{I}(\mathbb{P}) = \mathcal{M}_{\mathcal{G}}.$$

In other words, a DAG $\mathcal{G} = (V, D)$ is consistent with $\mathcal{I}(\mathbb{P})$ defined by equation (3.1) if and only if $\mathcal{G}$ is faithful to $\mathbb{P}$, if and only if $\mathcal{G}$ is a perfect I-map of $\mathbb{P}$.

A set of variables $(X_1, \ldots, X_d)$, may be ordered in $d!$ ways. Each permutation σ of $1, \ldots, d$ gives an ordering $(X_{\sigma(1)}, \ldots, X_{\sigma(d)})$ and for each permutation, the distribution may be factorised according to a Bayesian network, represented by the corresponding directed acyclic graph.

The input for the construction of the directed acyclic graph consists of a list of d conditional independence statements, one for each variable, all of the form $\{X_{\sigma(j)}\} \perp U_j^\sigma | \Pi_j^\sigma$, where U_j^σ is the list of σ-predecessors of $X_{\sigma(j)}$, without Π_j^σ.

For a given collection of variables $V = \{X_1, \ldots, X_d\}$, there may be several *different* DAGs, each representing the *same* independence structure. Two DAGs which represent exactly the same independence structure are said to be I-equivalent.

Definition 3.4 (I-sub-map, I-map, I-equivalence, Markov Equivalence)**.** *Let* $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *be two DAGs over the same variables. The DAG* $\mathcal{G}_1$ *is said to be an* I-sub-map *of* $\mathcal{G}_2$ *if any pair of variables d-separated by a set in* $\mathcal{G}_1$ *are also d-separated by the same set in* $\mathcal{G}_2$*. That is, the set of d-separation statements for* $\mathcal{G}_1$ *is a subset of the set of d-separation statements for* $\mathcal{G}_2$*. They are said to be* I-equivalent *if* $\mathcal{G}_1$ *is an* I*-sub-map of* $\mathcal{G}_2$ *and* $\mathcal{G}_2$ *is an* I*-sub-map of* $\mathcal{G}_1$.

I*-equivalence is also known as* Markov equivalence.

Example 3.5.

In the following example on three variables, all three factorisations give the same independence structure. Consider a probability distribution $\mathbb{P}_{X_1,X_2,X_3}$ with factorisation

$$\mathbb{P}_{X_1,X_2,X_3} = \mathbb{P}_{X_1}\mathbb{P}_{X_2|X_1}\mathbb{P}_{X_3|X_2}.$$

It follows that

$$\mathbb{P}_{X_1,X_2,X_3} = \mathbb{P}_{X_2}\mathbb{P}_{X_1|X_2}\mathbb{P}_{X_3|X_1,X_2} = \mathbb{P}_{X_2}\mathbb{P}_{X_1|X_2}\frac{\mathbb{P}_{X_1,X_2,X_3}}{\mathbb{P}_{X_1,X_2}} = \mathbb{P}_{X_2}\mathbb{P}_{X_1|X_2}\frac{\mathbb{P}_{X_1,X_2}\mathbb{P}_{X_3|X_2}}{\mathbb{P}_{X_1,X_2}} = \mathbb{P}_{X_2}\mathbb{P}_{X_1|X_2}\mathbb{P}_{X_3|X_2},$$

so that $X_1 \perp X_3|X_2$. Also,

$$\mathbb{P}_{X_1,X_2,X_3} = \mathbb{P}_{X_3}\mathbb{P}_{X_2|X_3}\mathbb{P}_{X_1|X_2,X_3} = \mathbb{P}_{X_3}\mathbb{P}_{X_2|X_3}\mathbb{P}_{X_1|X_2},$$

since $X_1 \perp X_3|X_2$. For the first and last of these, X_2 is a chain node, while in the second of these X_2 is a fork node. The conditional independence structure associated with chains and forks is the same. The three corresponding DAGs are given in figure 3.1.

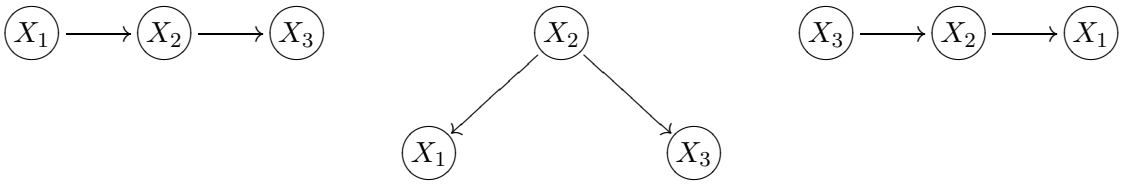

Figure 3.1: Three DAGs, each with the same independence structure

□

In general, the factorisations resulting from different orderings of the variables will not necessarily give I-equivalent maps. This is illustrated by the following example on four variables.

Example 3.6.

Consider a probability distribution over four variables, which may be factorised as

$$\mathbb{P}_{X_1,X_2,X_3,X_4} = \mathbb{P}_{X_1}\mathbb{P}_{X_2}\mathbb{P}_{X_3|X_1,X_2}\mathbb{P}_{X_4|X_3}.$$

The DAG associated with the factorisation is the one on the left in figure 3.2. Assume that the DAG on the left in figure 3.2 and $\mathbb{P}_{X_1,X_2,X_3,X_4}$ are faithful to each other. The factorisation obtained using the ordering (X_1, X_4, X_3, X_2) is:

$$\mathbb{P}_{X_1,X_2,X_3,X_4} = \mathbb{P}_{X_1}\mathbb{P}_{X_4|X_1}\mathbb{P}_{X_3|X_1,X_4}\mathbb{P}_{X_2|X_1,X_3,X_4} = \mathbb{P}_{X_1}\mathbb{P}_{X_4|X_1}\mathbb{P}_{X_3|X_1,X_4}\mathbb{P}_{X_2|X_1,X_3}.$$

This is the factorisation corresponding to a Bayesian network since $X_1 \not\perp X_4$, $X_3 \not\perp \{X_1, X_4\}\backslash\Theta|\Theta$ for any $\Theta \subseteq \{X_1, X_4\}$, and $X_2 \perp X_4|\{X_1, X_3\}$ but $X_2 \not\perp \{X_1, X_3, X_4\}\backslash\Theta|\Theta$ for any strict subset $\Theta \subset \{X_1, X_3\}$. The corresponding DAG (the graph on the right of figure 3.2) gives less information on conditional independence; $X_4 \not\perp X_1\|_{\mathcal{G}_2}X_3$, using $\mathcal{G}_2$ to denote the DAG on the right in figure 3.2. The two corresponding DAGs are shown in figure 3.2. The graph on the right is a strict I-sub-map of the graph on the left. □

This example illustrates that while d-separated variables are conditionally independent conditioned on the separating set, it does not hold that conditionally independent variables are necessarily d-separated.

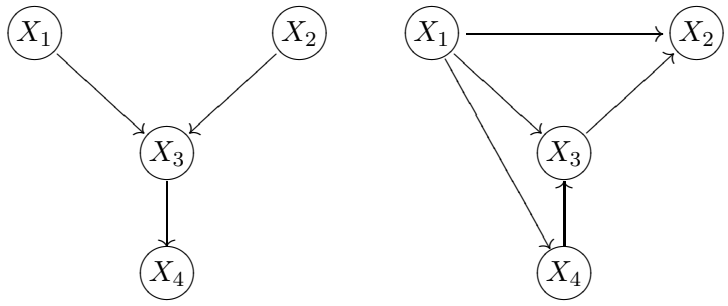

Figure 3.2: DAGs with different d-separation properties, arising from different factorisations of the same distribution

3.1.1 Probability Distributions Without a Faithful Graph

Given a probability distribution $\mathbb{P}_{\underline{X}}$ over a random vector $\underline{X} = (X_1, \ldots, X_d)$, is it always possible to find an ordering of the variables such that all the conditional independence statements of the distribution correspond to d-separation statements in the directed acyclic graph?

Any probability distribution satisfies the *decomposition* property. That is, for any probability distribution, $X \perp Y \cup Z | S$ implies that both $X \perp Y | S$ and $X \perp Z | S$ hold. This property also holds for d-separation in a DAG; for any DAG $\mathcal{G}$, $X \perp\!\!\!\perp Y \cup Z \| _{\mathcal{G}} S \Rightarrow X \perp\!\!\!\perp Y \| _{\mathcal{G}} S$ and $X \perp\!\!\!\perp Z \| _{\mathcal{G}} S$.

d-separation also satisfies the *composition* property, that if both $X \perp\!\!\!\perp Y \| _{\mathcal{G}} S$ and $X \perp\!\!\!\perp Z \| _{\mathcal{G}} S$ hold, then $X \perp\!\!\!\perp Y \cup Z \| _{\mathcal{G}} S$.

The following basic example illustrates a situation where *composition* does not hold for the probability distribution and therefore there is no faithful DAG.

Example 3.7 (Tossing Three Coins)**.**

Let Y_1, Y_2, Y_3 be three independent identically distributed binary variables, with probability function $\mathbb{P}(0) = \mathbb{P}(1) = \frac{1}{2}$. Let

$$X_1 = \begin{cases} 1 & Y_2 = Y_3 \\ 0 & Y_2 \neq Y_3 \end{cases}$$

$$X_2 = \begin{cases} 1 & Y_1 = Y_3 \\ 0 & Y_1 \neq Y_3 \end{cases}$$

$$X_3 = \begin{cases} 1 & Y_1 = Y_2 \\ 0 & Y_1 \neq Y_2 \end{cases}$$

Then X_1, X_2, X_3 provide the classic example of three random variables that are pairwise independent, but not jointly independent.

$$\mathbb{P}_{X_1,X_2,X_3}(1,1,1) = \mathbb{P}(Y_1 = Y_2 = Y_3) = \mathbb{P}_{Y_1,Y_2,Y_3}(1,1,1) + \mathbb{P}_{Y_1,Y_2,Y_3}(0,0,0) = \frac{1}{4}$$

$$\mathbb{P}_{X_1,X_2,X_3}(1,1,0) = \mathbb{P}_{X_1,X_2,X_3}(1,0,1) = \mathbb{P}_{X_1,X_2,X_3}(0,1,1) = \mathbb{P}(Y_2 = Y_3 = Y_1, Y_1 \neq Y_2) = 0$$

$$\begin{aligned}\mathbb{P}_{X_1,X_2,X_3}(1,0,0) &= \mathbb{P}_{X_1,X_2,X_3}(0,1,0) = \mathbb{P}_{X_1,X_2,X_3}(0,0,1) = \mathbb{P}(Y_2 = Y_3, Y_1 \neq Y_3, Y_1 \neq Y_2) \\ &= \mathbb{P}(Y_1 = 1, Y_2 = Y_3 = 0) + \mathbb{P}(Y_1 = 0, Y_2 = Y_3 = 1) = \frac{1}{4}\end{aligned}$$

$$\mathbb{P}_{X_1,X_2,X_3}(0,0,0) = 0$$

It follows that

$$\mathbb{P}_{X_1,X_2}(1,1) = \mathbb{P}_{X_1,X_2}(1,0) = \mathbb{P}_{X_1,X_2}(0,1) = \mathbb{P}_{X_1,X_2}(0,0) = \frac{1}{4}$$

so that $\mathbb{P}_{X_1}(1) = \mathbb{P}_{X_1}(0) = \frac{1}{2}$ and in all cases

$$\mathbb{P}_{X_1,X_2} = \mathbb{P}_{X_1}\mathbb{P}_{X_2}.$$

But

$$\frac{1}{4} = \mathbb{P}_{X_1,X_2,X_3}(1,1,1) \neq \mathbb{P}_{X_1}(1)\mathbb{P}_{X_2}(1)\mathbb{P}_{X_3}(1) = \frac{1}{8}.$$

Since $X_1 \perp X_2$ but $X_3 \not\perp \{X_1, X_2\}$, $X_3 \not\perp X_1|X_2$ and $X_3 \not\perp X_2|X_1$, it follows that the factorisation obtained for the distribution $\mathbb{P}_{X_1,X_2,X_3}$ is

$$\mathbb{P}_{X_1,X_2,X_3} = \mathbb{P}_{X_1}\mathbb{P}_{X_2}\mathbb{P}_{X_3|X_1,X_2}.$$

In the corresponding DAG, $X_1 \not\perp X_2 \| _{\mathcal{G}} \phi$, $X_1 \not\perp X_3 \| _{\mathcal{G}} \phi$ and $X_2 \not\perp X_3 \| _{\mathcal{G}} \phi$, even though the independence statements $X_1 \perp X_3$ and $X_2 \perp X_3$ hold.

By considering other orderings of the variables, the other possible factorisations are

$$\mathbb{P}_{X_1,X_2,X_3} = \mathbb{P}_{X_1}\mathbb{P}_{X_3}\mathbb{P}_{X_2|X_1,X_3} = \mathbb{P}_{X_2}\mathbb{P}_{X_3}\mathbb{P}_{X_1|X_2,X_3}$$

and in none of the cases do the d-separation statements of the DAG represent all the conditional independence statements of the distribution.

The type of situation described here, where the distribution does not satisfy a composition property, can be summarised as follows: it is the situation where X_1 tells you nothing about X_3 and X_2 tells you nothing about X_3, but X_1 and X_2 taken together tell you everything about X_3. This is the principle on which any good detective novel is based, as Edward Nelson puts it in his book 'Radically Elementary Probability Theory' [40].

Example 3.8 (Hidden Variables)**.**

If a set of variables $\{X_1, \ldots, X_d\}$ are related according to *causal* principles, there is usually a DAG that is faithful to the distribution. In many causal situations, though, the set of variables may be split into observable variables $\underline{Z}$ and unobservable variables, $\underline{U}$. Usually, the observable variables are descendants of the unobservable; observations are made on the observable and, from these observations, inferences made about the unobservable.

Even if there is a faithful DAG corresponding to the full set of variables $(\underline{U}, \underline{Z})$, there is often no faithful DAG corresponding to the observable variables $\underline{Z}$. For example, consider a probability distribution over the 5 variables $\{U_1, Z_1, Z_2, Z_3, Z_4\}$ which factorises according to

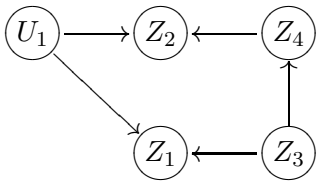

Figure 3.3: Faithful DAG, U_1 hidden

$$\mathbb{P}_{U_1,Z_1,Z_2,Z_3,Z_4,Z_5} = \mathbb{P}_{U_1}\mathbb{P}_{Z_3}\mathbb{P}_{Z_4|Z_3}\mathbb{P}_{Z_1|U_1,Z_3}\mathbb{P}_{Z_2|U_1,Z_4}$$

and suppose the corresponding graph given by figure 3.3 is faithful. In this example, there is no faithful DAG for the distribution over (Z_1, Z_2, Z_3, Z_4); the set of d-separation statements for any DAG along which the distribution can be factorised will be a strict subset of the set of conditional independence statements. Two examples of factorisations are:

$$\mathbb{P}_{Z_1,Z_2,Z_3,Z_4} = \mathbb{P}_{Z_1}\mathbb{P}_{Z_2|Z_1}\mathbb{P}_{Z_3|Z_1,Z_2}\mathbb{P}_{Z_4|Z_1,Z_2,Z_3}$$

$$\mathbb{P}_{Z_1,Z_3,Z_4,Z_2} = \mathbb{P}_{Z_1}\mathbb{P}_{Z_3|Z_1}\mathbb{P}_{Z_4|Z_3,Z_1}\mathbb{P}_{Z_2|Z_1,Z_4}$$

Either an edge $Z_2 \sim Z_3$ or an edge $Z_1 \sim Z_4$ will be present. No DAG will represent all the CI statements.

3.2 Characterisation of Markov Equivalence

When trying to fit a graphical model to data, all the DAGs in a Markov equivalence class will fit the data equally well and efficient algorithms for finding the structure will therefore only examine the different equivalence classes, rather than all the different possible DAGs.

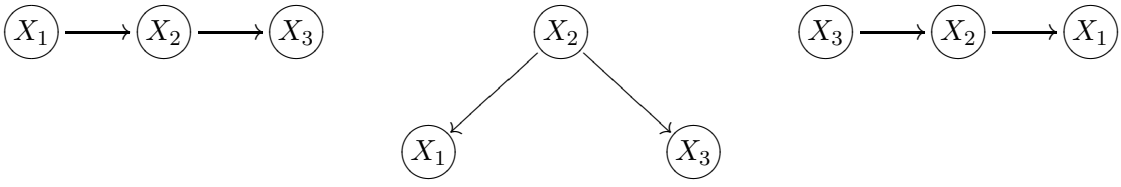

Figure 3.4: Three DAGs, each with the same independence structure

Figure 3.4 shows three directed acyclic graphs, each with the same d-separation statements; $X_1 \perp\!\!\!\perp X_3 \| _{\mathcal{G}} X_2$ and the DAG does not admit any other d-separation statements. The d-separation statements of the DAG in figure 3.5 are different from those for the DAGs in figure 3.4.

The key result in this section characterising Markov equivalence is theorem 3.11, which states that the two features of a directed acyclic graph which are necessary and sufficient for determining its Markov structure are its *immoralities* and its *skeleton*. These are defined below.

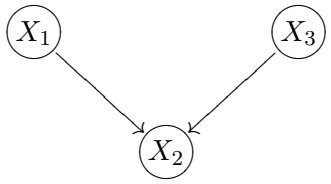

Figure 3.5: Not Markov Equivalent to the Graphs in Figure 3.4

Definition 3.9 (Immorality)**.** *Let $\mathcal{G} = (V, E)$ be a graph. Let $E = D \cup U$, where D contains directed edges, U contains undirected edges and $D \cap U = \phi$. An* immorality *in a graph is a triple (α, β, γ) such that $(\alpha, \beta) \in D$ and $(\gamma, \beta) \in D$, but $(\alpha, \gamma) \notin D$, $(\gamma, \alpha) \notin D$ and $\langle \alpha, \gamma \rangle \notin U$.*

Definition 3.10 (Skeleton)**.** *The* skeleton *of a graph $\mathcal{G} = (V, E)$ is the graph obtained by making the graph* undirected*. That is, the skeleton of $\mathcal{G}$ is the graph $\tilde{\mathcal{G}} = (V, \tilde{E})$ where $\langle \alpha, \beta \rangle \in \tilde{E} \Leftrightarrow (\alpha, \beta) \in D$ or $(\beta, \alpha) \in D$ or $\langle \alpha, \beta \rangle \in U$.*

The characterisation is given by the following theorem.

Theorem 3.11. *Two DAGs are Markov equivalent if and only if they have the same* skeleton *and the same* immoralities*.*

The key to establishing theorem 3.11 will be to consider the active trails (definition 2.4) in the graph. The following two definitions are also required.

Definition 3.12 (S-active node)**.** *Let $\mathcal{G} = (V, E)$ be a Directed Acyclic Graph and let $S \subset V$. Recall the definition of a trail (definition 1.4) and the definition of an active trail (definition 2.4). A node $\alpha \in V$ is said to be* S-active *if either $\alpha \in S$ or there is a directed path* from *the node α* to *a node $\beta \in S$.*

Definition 3.13 (Minimal S- active trail)**.** *Let $\mathcal{G} = (V, E)$ be a Directed Acyclic Graph and let $S \subset V$. An S-active trail τ in $\mathcal{G}$ between two nodes α and β is said to be a* minimal S- active trail *if it satisfies the following two properties:*

1. *if k is the number of nodes in the trail, the first node is α and the kth node is β, then there does not exist an S-active trail between α and β with fewer than k nodes and*
2. *there does not exist a* different *S-active trail ρ between α and β with exactly k nodes such that for all $1 < j < k$ either $\rho_j = \tau_j$ or ρ_j is a descendant of τ_j.*

The proof of theorem 3.11 follows directly from lemma 3.14.

Lemma 3.14. *Let $\mathcal{G}_1 = (V, D_1)$ and $\mathcal{G}_2 = (V, D_2)$ be two directed acyclic graphs with the same skeletons and the same immoralities. Then for all $S \subset V$, a trail is a minimal S-active trail in $\mathcal{G}_1$ if and only if it is a minimal S-active in $\mathcal{G}_2$.*

Proof of lemma 3.14 Recall the notation from definition 1.2; $\alpha \sim \beta$ denotes that two nodes $(\alpha, \beta) \in V \times V$ are neighbours. For a directed graph $\mathcal{G} = (V, D)$, that is either $(\alpha, \beta) \in D$ or $(\beta, \alpha) \in D$. Since $\mathcal{G}_1$

and $\mathcal{G}_2$ have the same skeletons, any trail τ in $\mathcal{G}_1$ is also a trail in $\mathcal{G}_2$. Let $S \subset V$. Assume that τ is a minimal S-active trail in $\mathcal{G}_1$. It is now proved, by induction on the number of collider nodes along the path, that τ is also an S-active trail in $\mathcal{G}_2$. By definition, a single node will be considered an S-active trail, for any $S \subset V$. The proof is in three parts: Let τ be a *minimal* S-active trail in $\mathcal{G}_1$. Then

1. If τ contains no colliders in $\mathcal{G}_1$, then it is S-active in $\mathcal{G}_2$.
2. If τ contains at least one collider connection centred at node τ_j, then τ is S-active in $\mathcal{G}_2$ if and only if τ_j is S-active in $\mathcal{G}_2$.
3. If τ contains at least one collider centred at node τ_j, then τ_j is an S-active node in $\mathcal{G}_2$.

Part 1 If τ is an S-active trail in $\mathcal{G}_1$ and does not contain any collider connections in $\mathcal{G}_1$, then none of the nodes on τ are in S. This can be seen by considering the Bayes ball algorithm, which characterises d-separation. It follows that the trail is S-active in $\mathcal{G}_2$ if and only if it does not contain a collider connection in $\mathcal{G}_2$.

Let τ be a minimal S-active trail in $\mathcal{G}_1$ with k nodes and no collider connections in $\mathcal{G}_1$. Suppose that a node τ_i is a collider node in $\mathcal{G}_2$, so that τ_{i-1} and τ_{i+1} are parents of τ_i in $\mathcal{G}_2$. Then, so that no new immoralities are introduced, it follows that $\tau_{i-1} \sim \tau_{i+1}$. Since τ_i is either a chain or a fork in $\mathcal{G}_1$, it follows that in $\mathcal{G}_1$, the connections between nodes $\tau_{i-2}, \tau_{i-1}, \tau_i, \tau_{i+1}, \tau_{i+2}$ take one of the forms shown in figure 3.6 when τ_i a chain node or those in figure 3.7 when τ_i a fork node.

It is clear from the figure that the trail of length $k-1$ in $\mathcal{G}_1$, obtained by removing τ_i and using the direct link from τ_{i-1} to τ_{i+1} is also an S-*active* trail in $\mathcal{G}_1$, contradicting the assumption that τ was a minimal S-active trail. Hence τ_i is a chain node or a fork node in $\mathcal{G}_2$.

It follows that there are no collider connections along the trail τ taken in $\mathcal{G}_2$ and hence, since it does not contain any nodes that are in S, it is an S- active trail in $\mathcal{G}_2$.

Part 2 Assume that any minimal S-active trail in $\mathcal{G}_1$ containing n collider connections is also S-active in $\mathcal{G}_2$. This is true for $n = 0$ by part 1. Let τ be a trail with k nodes that is minimal S-active in $\mathcal{G}_1$ and with $n+1$ collider connections in $\mathcal{G}_1$. Consider one of the collider connections centred at τ_j, with parents τ_{j-1} and τ_{j+1}. Let $\tilde{\tau}^{(0,j-1)} = (\tau_0, \tau_1, \ldots, \tau_{j-2}, \tau_{j-1})$ and let $\tilde{\tau}^{(j+1,k)} = (\tau_{j+1}, \ldots, \tau_k)$. Both $\tilde{\tau}^{(0,j-1)}$ and $\tilde{\tau}^{(j+1,k)}$ are minimal S-active in $\mathcal{G}_1$ and they both have a number of collider connections less than or equal to n. By the inductive hypothesis, they are therefore both S-active in $\mathcal{G}_2$.

Because the trail τ is minimal S-active in $\mathcal{G}_1$, it follows that $\tau_{j-1} \not\sim \tau_{j+1}$. This is because both τ_{j-1} and τ_{j+1} are S- active nodes in $\mathcal{G}_1$ (they have a common descendant in S to make the trail active), and neither is in S (neither is the centre of a collider along τ) it follows that if $\tau_{j-1} \sim \tau_{j+1}$, then the trail on $k-1$ nodes obtained by removing the node τ_j would be S-active in $\mathcal{G}_1$, for the following reason: any chain or fork $(\tau_{j-2}, \tau_{j-1}, \tau_{j+1})$ or $(\tau_{j-1}, \tau_{j+1}, \tau_{j+2})$ would be active because both τ_{j-1} and τ_{j+1} are uninstantiated. Any collider $(\tau_{j-2}, \tau_{j-1}, \tau_{j+1})$ or $(\tau_{j-1}, \tau_{j+1}, \tau_{j+2})$ would be active because both τ_{j-1} and τ_{j+1} have a descendant in S. It follows that $\tau_{j-1} \not\sim \tau_{j+1}$. This holds in both $\mathcal{G}_1$ and $\mathcal{G}_2$, since the skeletons are the same.
Since $\tilde{\tau}^{(1,i-1)}$ and $\tilde{\tau}^{(i+1,k)}$ are both active, and $\tau_{j-1} \to \tau_j \leftarrow \tau_{j+1}$ is a collider, the trail τ is active if and only if τ_j is an active node. That is, it is either in S or has a descendant in S.

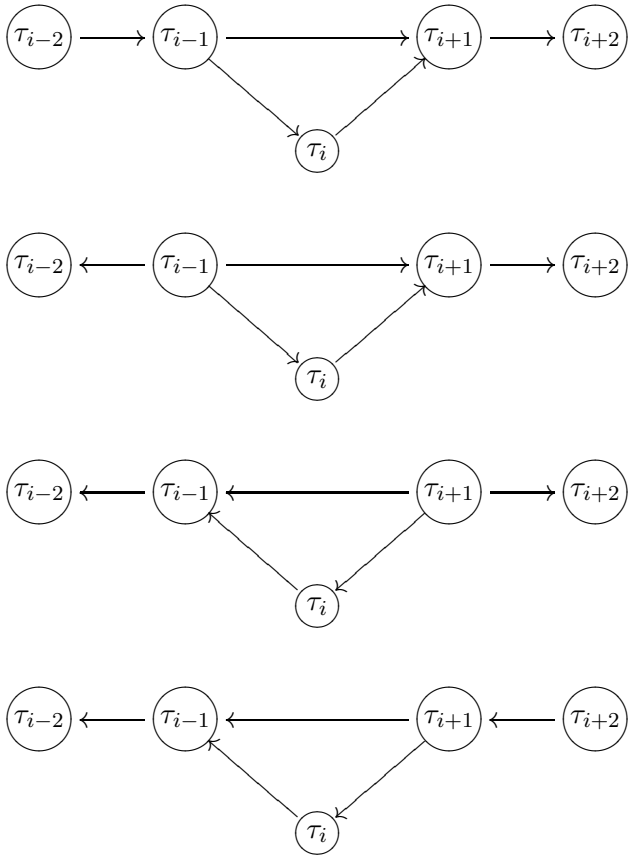

Figure 3.6: Possible connections between the nodes when τ_i is chain node

Part 3 Let τ be a minimal S-active trail in $\mathcal{G}_1$ and let $\tau_j \in \tau$ be a collider node in $\mathcal{G}_1$. Since the trail τ is a minimal S-active trail in $\mathcal{G}_1$, it follows either that $\tau_j \in S$ or τ_j, considered in $\mathcal{G}_1$, has a descendant in S. That is, considered in $\mathcal{G}_1$, there is a directed *path* from τ_j to a node $w \in S$. Let ρ denote the shortest such path. If $\tau_j \in S$, then the length of the path is 0 and τ_j is also an S-active node in $\mathcal{G}_2$.

Assume there is a directed edge from τ_j to $w \in S$ in $\mathcal{G}_1$. If there are links from τ_{j-1} to w or τ_{j+1} to w, then these links are $\tau_{j-1} \to w$ or $\tau_{j+1} \to w$ respectively, otherwise the DAG would have cycles. If both are present, then the trail τ violates the second assumption of the minimality requirement. This is seen by considering the trail formed by taking w instead of τ_j in τ. It follows that either $\tau_{j-1} \not\to w$ or $\tau_{j+1} \not\to w$ or neither of the edges are present. Without loss of generality, assume $\tau_{j-1} \not\to w$ (since the argument proceeds in the same way if $\tau_{j+1} \not\to w$). The diagram in figure 3.8 may be useful.

Since w is not a parent of τ_j in $\mathcal{G}_1$, it cannot be a parent of τ_j in $\mathcal{G}_2$, since both graphs have the same immoralities and (τ_{j-1}, τ_j, w) is not an immorality in $\mathcal{G}_1$.

Furthermore, $\tau_{j-1} \not\sim \tau_{j+1}$ (since they are both uninstantiated, and, in $\mathcal{G}_1$ both have a common descendant in S, so that if $\tau_{j-1} \sim \tau_{j+1}$ then the trail with τ_j removed would be active whether the connections at τ_{j-1} and τ_{j+1} are chain, fork or collider, contradicting the minimality assumption). Since both graphs have the same immoralities and $\tau_{j-1} \not\sim \tau_{j+1}$, it follows that $(\tau_{j-1}, \tau_j, \tau_{j+1})$ is an immorality in both $\mathcal{G}_1$ and $\mathcal{G}_2$ and hence that τ_{j-1} is a parent of τ_j in $\mathcal{G}_2$. Therefore, τ_j is a parent of

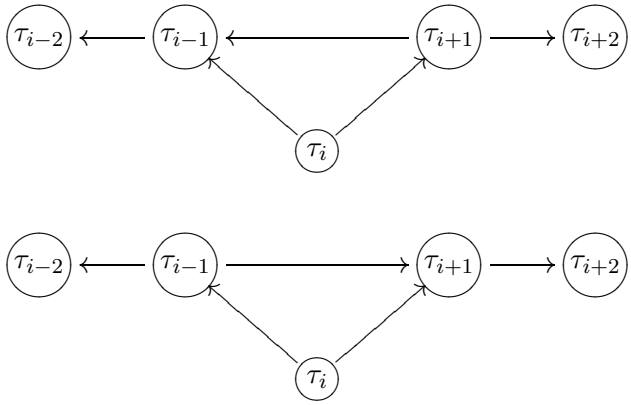

Figure 3.7: Possible connections between the nodes if τ_i is a fork node

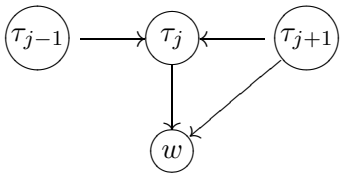

Figure 3.8: Illustration where τ_j is an uninstantiated collider node

w in $\mathcal{G}_2$ and is therefore w is an S- active node in $\mathcal{G}_2$.

Assume that for the shortest directed path ρ from τ_j to w in $\mathcal{G}_1$, the first l links have the same directed edges in $\mathcal{G}_2$. Now suppose that the shortest directed path is ρ, where $\tau_j = \rho_0, \ldots, \rho_{l+p} = w$ and consider the links $\rho_l \sim \rho_{l+1}$ and $\rho_{l+1} \sim \rho_{l+2}$. If $\rho_l \sim \rho_{l+2}$, then in $\mathcal{G}_1$, the directed edge $\rho_l \to \rho_{l+2}$ is present, otherwise there is a cycle. If the directed edge $\rho_l \to \rho_{l+2}$ is present in $\mathcal{G}_1$, then the path ρ is not minimal. Therefore, $\rho_l \not\to \rho_{l+2}$. This holds in both $\mathcal{G}_1$ and $\mathcal{G}_2$, because both graphs have the same skeletons. By a similar argument, $\rho_{l-1} \not\to \rho_{l+1}$. (there would be a cycle in $\mathcal{G}_1$ if (ρ_{l+1}, ρ_{l-1}) were present; ρ would not be minimal in $\mathcal{G}_1$ if (ρ_{l-1}, ρ_{l+1}) were present. Since the skeletons are the same, $\rho_{l-1} \not\to \rho_{l+1}$ in either $\mathcal{G}_1$ or $\mathcal{G}_2$). Since $\rho_l \not\to \rho_{l+2}$, it follows that ρ_l and ρ_{l+2} are not both parents of ρ_{l+1} in $\mathcal{G}_2$; otherwise $\mathcal{G}_2$ would contain an immorality not present in $\mathcal{G}_1$. Similarly, since $\rho_{l-1} \not\to \rho_{l+1}$, the edge $\rho_l \to \rho_{l+1}$ is present in $\mathcal{G}_2$, otherwise $\mathcal{G}_2$ would have either the immorality $(\rho_{l-1}, \rho_l, \rho_{l+1})$, since the edge (ρ_{l-1}, ρ_l) is present in $\mathcal{G}_2$ by assumption. It follows that the directed edges (ρ_l, ρ_{l+1}) and (ρ_{l+1}, ρ_{l+2}) are both present in $\mathcal{G}_2$. By induction, therefore, the whole directed path ρ is also present in $\mathcal{G}_2$ and hence τ_j is an S- active in both $\mathcal{G}_1$ and $\mathcal{G}_2$. □

Proof of Theorem 3.11 This follows directly: let $\mathcal{G}_1$ and $\mathcal{G}_2$ denote two DAGs with the same skeleton and the same immoralities. For any set S and any two nodes X_i and X_j, it follows from the lemma, together with the definition of d- separation, that

$$X_i \perp\!\!\!\perp X_j \| _{\mathcal{G}_1} S \Leftrightarrow X_i \perp\!\!\!\perp X_j \| _{\mathcal{G}_2} S; \tag{3.3}$$

if there is an S-active trail between the two variables in one of the graphs, then there is a minimal S-active trail in that graph and hence there is also a minimal S-active trail between the two variables in the other. If there is no S-active trail between the two variables in one of the graphs then there is no S active trail between the two variables in the other. By definition, two variables are d-separated by a set of variables S if and only if there is no S-active trail between the two variables. Two graphs are Markov equivalent, or I-equivalent (definition 3.4), if and only if equation (3.3) holds for all $(X_i, X_j, S) \in V \times V \times \mathcal{V}$. □

3.3 Conditional Independence Statements

Let (X, Y, W, Z) be disjoint sets of random variables, each with a finite state space. For a probability distribution over $V \supseteq X \cup Y \cup W \cup Z$, the following logical relations hold, discussed by Pearl (1988) [43]:

1. **decomposition** If $X \perp Y \cup W | Z$ then $X \perp Y | Z$ and $X \perp W | Z$.
2. **contraction** If $X \perp Y | Z$ and $X \perp W | Y \cup Z$ then $X \perp W \cup Y | Z$.
3. **weak union** If $X \perp Y \cup Z | W$ then $X \perp Y | Z \cup W$.
4. **intersection** If $X \perp Y | W \cup Z$ and $X \perp W | Y \cup Z$ then $X \perp W \cup Y | Z$.

A collection of sets $\mathcal{V}$ such that these statements are satisfied for any $X, Y, Z, W \in \mathcal{V}$ has come to be known as a *graphoid*. Any independence model is a graphoid and it is straightforward to establish these from the definition of conditional independence.

The corresponding d-separation statements hold in a DAG. Let (X, Y, W, Z) be four disjoint sets of nodes in a DAG $\mathcal{G} = (V, D)$, then the following statements may be verified:

1. **decomposition** If $X \perp\!\!\!\perp Y \cup W \|_{\mathcal{G}} Z$ then $X \perp\!\!\!\perp Y \|_{\mathcal{G}} Z$ and $X \perp\!\!\!\perp W \|_{\mathcal{G}} Z$.
2. **contraction** If $X \perp\!\!\!\perp Y \|_{\mathcal{G}} Z$ and $X \perp\!\!\!\perp W \|_{\mathcal{G}} Y \cup Z$ then $X \perp\!\!\!\perp W \cup Y \|_{\mathcal{G}} Z$.
3. **weak union** If $X \perp\!\!\!\perp Y \cup Z \|_{\mathcal{G}} W$ then $X \perp\!\!\!\perp Y \|_{\mathcal{G}} Z \cup W$.
4. **intersection** If $X \perp\!\!\!\perp Y \|_{\mathcal{G}} W \cup Z$ and $X \perp\!\!\!\perp W \|_{\mathcal{G}} Y \cup Z$ then $X \perp\!\!\!\perp W \cup Y \|_{\mathcal{G}} Z$.

In addition, a DAG also satisfies *composition*:

5 **composition** in a graph: If $X \perp\!\!\!\perp Y \|_{\mathcal{G}} Z$ and $X \perp\!\!\!\perp W \|_{\mathcal{G}} Z$, then $X \perp\!\!\!\perp Y \cup W \|_{\mathcal{G}} Z$.

A graphoid for which *composition* holds, namely

5 **composition** for a probability distribution:

$$\text{If} \quad X \perp Y | Z \quad \text{and} \quad X \perp W | Z \quad \text{then} \quad X \perp Y \cup W | Z \tag{3.4}$$

is known as a *compositional graphoid*

Composition does not necessarily hold for a probability distribution; example 3.7 does not satisfy a composition property. Composition is a common feature of graphical models; they all satisfy composition.

In addition, DAGS also satisfy the following properties:

1. Let $V = A \cup B \cup S$ where A, B and S are disjoint subsets and suppose that $A \perp B|S$. For any $\alpha \in A$ and $\gamma \in A \cup S$,

$$\alpha \perp \gamma|(A \cup S)\backslash\{\alpha, \gamma\} \Leftrightarrow \alpha \perp \gamma|(A \cup B \cup S)\backslash\{\alpha, \gamma\}.$$

2. Let $\mathcal{G} = (V, D)$ denote a directed acyclic graph. Let $X \subseteq V$, $Y \subseteq V$ and $Z \subseteq V$ denote sets of nodes and let $\alpha, \beta, \gamma, \delta \in V\backslash X \cup Y \cup Z$ denote individual nodes.

   (a) If $X \perp\!\!\!\perp Y \| _{\mathcal{G}} Z$ and $X \perp\!\!\!\perp Y \| _{\mathcal{G}} Z \cup \{\gamma\}$ then either $X \perp\!\!\!\perp \{\gamma\} \| _{\mathcal{G}} Z$ or $Y \perp\!\!\!\perp \{\gamma\} \| _{\mathcal{G}} Z$

   (b) If $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} \{\gamma, \delta\}$ and $\gamma \perp\!\!\!\perp \delta \| _{\mathcal{G}} \{\alpha, \beta\}$ then either $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} \{\gamma\}$ or $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} \{\delta\}$.

In fact, it is not possible to axiomatise conditional independence, as Studený proved. The proof may be found in [61].

3.4 Markov Equivalence, The Essential Graph and Chain Graphs

Consider the DAG in figure 3.9. Using the characterisation given by theorem 3.11, the DAG in figure 3.9 is equivalent to the DAGs in figure 3.10.

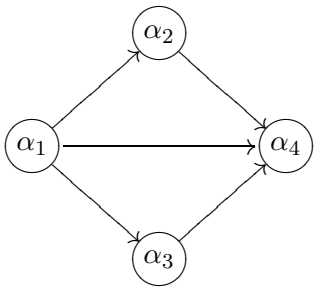

Figure 3.9: A DAG on four nodes

For the DAG in figure 3.9, all the DAGs with the same skeleton can be enumerated, and it is clear that those in figure 3.10 are the only two that satisfy the criteria. To find the DAGs equivalent to the one in figure 3.9, the immorality $(\alpha_2, \alpha_4, \alpha_3)$ has to be preserved and no new immoralities may be added. The directed edges (α_1, α_4), (α_2, α_4) and (α_3, α_4) are therefore *essential*, the directed edges (α_2, α_4) and (α_3, α_4) to form the immorality, the directed edge (α_1, α_4) because the connection $(\alpha_2, \alpha_1, \alpha_3)$ is either a fork or chain, forcing (α_1, α_4) to prevent a cycle. These three directed edges will be present in any equivalent DAG. The other three edges may be oriented in 2^3 different ways, but only 5 of these

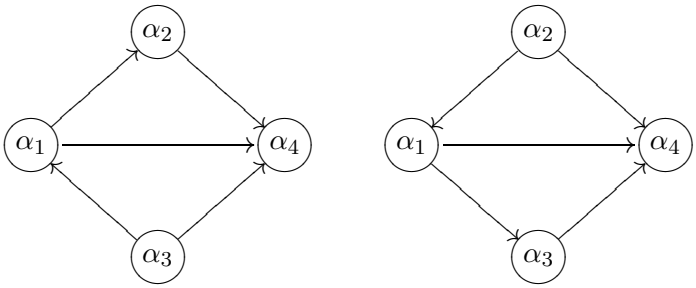

Figure 3.10: The equivalent DAGs

lead to DAGs (the other graphs contain cycles) and of these 5, only the three shown in figures 3.9 and 3.10 have the same immoralities.

A useful starting point for locating all the DAGs that are Markov equivalent to a given DAG is to locate the *essential graph*, given in the following definition.

Definition 3.15 (Essential Graph)**.** *Let $\mathcal{G}$ be a Directed Acyclic Graph. The* essential graph *$\mathcal{G}^*$ associated with $\mathcal{G}$ is the graph with the same skeleton as $\mathcal{G}$, but where an edge is directed in $\mathcal{G}^*$ if and only if it occurs as a directed edge with the same orientation in* every *DAG that is Markov equivalent to $\mathcal{G}$. The* directed *edges of $\mathcal{G}^*$ are the* essential *edges of $\mathcal{G}$.*

The edges that are directed in an essential graph are the *compelled* edges.

Definition 3.16 (Compelled Edge)**.** *Let $\mathcal{G} = (V, E)$ be a chain graph, where $E = D \cup U$. A directed edge $(\alpha, \beta) \in D$ is said to be* compelled *if it occurs in at least one of the configurations in figure 3.11.*

Lemma 3.17. *In an essential graph, the directed edges are the compelled edges; all other edges are undirected.*

Proof From the definition, the directed edges are those that necessarily have the same direction in every Markov equivalent DAG. The figure in the top right shows the immoralities; these are necessarily directed. The direction is forced in the figure in the top left; otherwise the graph contains an additional immorality. The direction is forced in the structure on the bottom left; otherwise there is a cycle. The direction is forced in the structure in the bottom right; otherwise both (γ_1, α) and (γ_2, α) are forced to prevent cycles appearing and $(\gamma_1, \alpha, \gamma_2)$ is an additional immorality.

To show that these are the only compelled edges: Consider two nodes α and β which are neighbours. Firstly, suppose that α and β do not have any other common neighbours. If α does not have a neighbour γ such that there is a directed edge (γ, α), then the direction (α, β) is not forced; no additional immorality or cycle is created by either direction.

Now suppose that α and β have at least one common neighbour. Suppose that there are no neighbours γ such that both (α, γ), (γ, β) in the directed edge set, then it is not necessary to force the direction (α, β) to prevent a cycle.

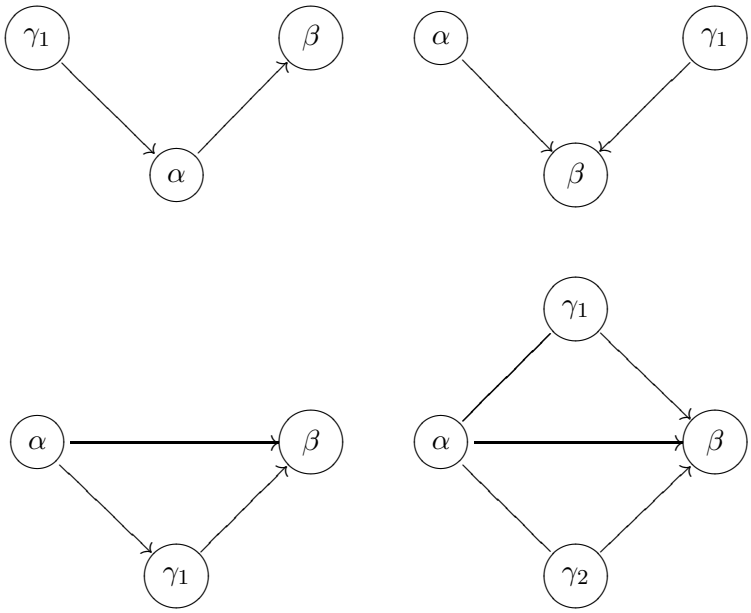

Figure 3.11: The directed edge (α, β) is compelled (definition 3.16)

Suppose, furthermore, that there is no pair of common neighbours γ_1 and γ_2, such that (γ_1, β) and (γ_2, β) are both in the directed edge set and $(\gamma_1, \alpha, \gamma_2)$ is not an immorality. Then it is not necessary to force the direction (α, β) to prevent either $(\gamma_1, \alpha, \gamma_2)$ becoming an immorality or else a cycle appearing. □

The essential graph contains both directed and undirected edges, and is an example of a *chain graph*. The following material gives the definition of a chain graph and deals mainly with the properties that will be used when considering the essential graph, with some extensions.

Definition 3.18 (Chain Graph)**.** *A* chain graph *is a graph $\mathcal{G} = (V, E)$, where the edge set contains both directed and undirected edges, $E = D \cup U$, where D is the set of directed edges and U the set of undirected edges. The node set V can be partitioned into n disjoint subsets $V = V_1 \cup \ldots \cup V_n$ where the sets $V_1, \ldots, V_n$ are the node sets of the connected components of (V, U), the graph obtained by removing all the directed edges.*

1. *$\mathcal{G}_{V_j}$ is an undirected graph for all $j = 1, \ldots, n$*
2. *For any $i \neq j$, and any $\alpha \in V_i$, $\beta \in V_j$, there is no cycle in $\mathcal{G} = (V, E)$ (definition 1.9) containing both α and β.*

The chain graph consists of components where the edges are undirected, which are connected by directed edges. The components with undirected edges are known as *chain components*, which are defined below.

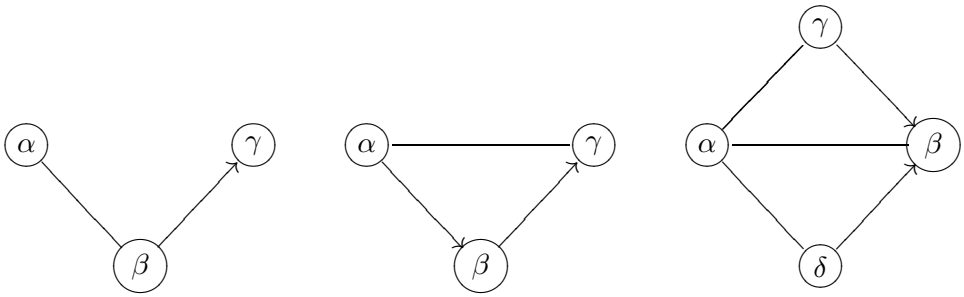

Figure 3.12: Forbidden subgraphs

Definition 3.19 (Chain Component)**.** *Let $\mathcal{G} = (V, E)$ be a chain graph, where $E = D \cup U$, D is the set of directed edges. Let $\widehat{\mathcal{G}} = (V, U)$ denote the graph obtained by* removing *all the* directed *edges from E. Each connected component of $\widehat{\mathcal{G}}$ is known as a* chain component.

The *chain components* (V_j, U_j), $j = 1, \ldots, n$ of $\mathcal{G}$ therefore satisfy the following conditions:

1. $V_j \subseteq V$ and U_j is the edge set obtained by retaining all *undirected* edges $\langle \alpha, \beta \rangle \in E$ such that $\alpha \in V_j$ and $\beta \in V_j$.

2. There is no undirected edge in E from any node in $V \backslash V_j$ to any node in V_j.

Theorem 3.20 states any essential graph is necessarily a chain graph and presents the additional features required to ensure that a chain graph is an essential graph corresponding to a directed acyclic graph. It gives a characterisation for essential graphs that is useful for some Monte Carlo algorithms for locating the graph structure.

Theorem 3.20. *Let $\mathcal{G} = (V, E)$ be a graph, where $E = D \cup U$. There exists a directed acyclic graph $\mathcal{G}^*$ for which $\mathcal{G}$ is the corresponding essential graph if and only if $\mathcal{G}$ satisfies the following conditions:*

1. *$\mathcal{G}$ is a chain graph,*

2. *Each chain component of $\mathcal{G}$ is triangulated,*

3. *The configurations shown in figure 3.12 do not occur in any induced sub-graph of a three variable set $\{\alpha, \beta, \gamma\} \subset V$ for the first two configurations or a four variable set $\{\alpha, \beta, \gamma, \delta\}$ for the third configuration.*

4. *Every directed edge $(\alpha_1, \alpha_2) \in D$ is compelled in $\mathcal{G}$.*

Proof Proof that an essential graph satisfies the conditions. To prove that it is a chain graph, the first part of the definition is easily satisfied and it is sufficient to show that there is no cycle in (V, E) containing $\alpha \in V_i$ and $\beta \in V_j$ for two distinct chain components V_i and V_j.

Recall that the edges of a cycle $\tau_0, \ldots, \tau_n$ are either directed (τ_i, τ_{i+1}) or undirected $\langle \tau_i, \tau_{i+1} \rangle$. Let (τ, γ) denote a directed edge in the cycle where $\gamma \in V_j$. Both connected components will have a node γ with this property. If there is an undirected edge $\langle \gamma, \gamma_1 \rangle$ in the cycle, then there is also an undirected edge $\langle \tau, \gamma_1 \rangle$ in the graph. If there is a directed edge (τ, γ_1) or (γ_1, τ) then the edge between γ and γ_1 is compelled contradicting the fact that it is undirected. If there is an undirected edge $\langle \tau, \gamma_1 \rangle$, then $\tau \in V_j$. Proceeding inductively, it is clear that if there is a cycle, then there is an undirected edge $\langle \tau_1, \tau_2 \rangle$ where $\tau_1 \in V_i$ and $\tau_2 \in V_j$ contradicting the fact that the two chain components are distinct. It follows that an essential graph is a chain graph.

Secondly, if there is a cycle of length ≥ 4 of undirected edges without a chord, then the DAG will have a directed cycle, otherwise additional immoralities will appear when the edges are directed, hence the chain components are triangulated.

Thirdly, the configuration stated cannot appear in an essential graph. The fourth requirement follows from the definition of an essential graph.

For the other direction: suppose a graph satisfies the four conditions stated. All the directed edges appear in configurations that are compelled and from the forbidden subgraphs, no undirected edges appear in compelled configurations where there should be a directed edge. It remains to show that the undirected edges may be oriented in a way that produces a directed acyclic graph.

For each chain component, orient the edges so that the chain component is a directed acyclic triangulated graph. This can be done. Then, since the first structure is forbidden, this operation does not produce additional immoralities in the whole graph. Furthermore, since there are no cycles containing two nodes α and β with $\alpha \in V_j$ and $\beta \in V_k$ for $j \neq k$, this operation does not produce directed cycles. The graph is therefore the essential graph of a DAG. □

3.5 The Moral Graph and the Independence Graph

Let $(\mathbb{P}, \mathcal{G})$ be a Bayesian network; that is, a probability distribution $\mathbb{P}$ over a random vector $\underline{X} = (X_1, \ldots, X_d)$, such that $\mathbb{P}$ factorises along a directed acyclic graph $\mathcal{G} = (V, D)$, and this is no longer true if any variable is eliminated from any of the parent sets.

Definition 3.21 (Moral Graph). *Let $\mathcal{G} = (V, D)$ be a directed acyclic graph. The moral graph $\mathcal{G}^{(m)} = (V, U)$ is the undirected graph such that for any $\alpha, \beta \in V$, $\langle \alpha, \beta \rangle \in U$ if and only if either $(\alpha, \beta) \in D$ or $(\beta, \alpha) \in D$ or $\{\alpha, \beta\} \in \Pi(\gamma)$ for some $\gamma \in V$. That is, the moral graph is the graph obtained by firstly for each node adding links between all the parent variables of the node and then undirecting all the directed edges.*

The moral graph satisfies the following property:

Theorem 3.22. *Let $\mathcal{G} = (V, D)$ be a directed acyclic graph and let $\mathcal{G}^{(m)} = (V, U)$ be its moral graph. There is an edge $\langle X, Y \rangle \in U$ if and only if $X \not\perp\!\!\!\perp Y \| _{\mathcal{G}} V \backslash \{X, Y\}$. That is, the moral graph has an edge if and only if X and Y are not d-separated by the remaining variables.*

Proof The proof of this is left as an exercise. □

The *independence graph* is defined as follows:

Definition 3.23 (Independence Graph). *Let* $\underline{X} = (X_1, \ldots, X_d)$ *be a random vector. The* independence graph $\mathcal{G} = (V, U)$ *is the undirected graph with vertex set* $V = \{X_1, \ldots, X_d\}$ *and where* $\langle X_i, X_j \rangle \in U$ *for* $i \neq j$ *if and only if* $X_i \not\perp X_j | V \backslash \{X_i, X_j\}$.

For undirected graphs, the definition of separation is as follows.

Definition 3.24 (Separator). *Let* $\mathcal{G} = (V, E)$ *be a graph. Let* $\alpha, \beta \in V$ *be two nodes. A subset* $S \in V$ *is called an* α, β *separator if every trail between* α *and* β *has at least one node in* S. *Let* $A \subset V$, $B \subset V$. *A set* $S \subset V$ *separates* A *and* B *if it is an* α, β *separator for eacn* $(\alpha, \beta) \in A \times B$. A *and* B *are said to be* separated *by* S. *The notation used in this text is* $A \perp\!\!\!\perp B \,[\!]\, S$.

The independence graph satisfies the following property:

Theorem 3.25. *Let* $\underline{X} = (X_1, \ldots, X_d)$ *and let* $\mathcal{G} = (V, U)$ *be the independence graph, with* $V = \{X_1, \ldots, X_d\}$. *Then for three disjoint sets* A, B *and* S *such that* $V = A \cup B \cup S$, *it holds that* $A \perp B | S$ *(*A *and* B *are conditionally independent given* S*) if and only if* $A \perp\!\!\!\perp B \,[\!]\, S$ *(*A *and* B *separated by* S*).*

Proof Firstly, assume that for three disjoint sets A, B and S such that $A \cup B \cup S$, $A \perp\!\!\!\perp B \,[\!]\, S$ in the independence graph. Then, for each $\alpha_1, \alpha_2 \in A$ and $\beta \in B$, set $C = V \backslash \{\alpha_1, \alpha_2, \beta\}$. From the definition of the independence graph,

$$\alpha_1 \perp \beta | C \cup \{\alpha_2\} \qquad \text{and} \qquad \alpha_2 \perp \beta | C \cup \{\alpha_1\}.$$

It follows from the **intersection** property that

$$\{\alpha_1, \alpha_2\} \perp \beta | C.$$

By successive applications of the **intersection** property to each variable, it follows that

$$A \perp \beta | V \backslash (A \cup \{\beta\}).$$

This holds for all $\beta \in B$. The *intersection* property gives:

$$A \perp \beta_1 | V \backslash (A \cup \{\beta_1\}) \qquad \text{and} \qquad A \perp \beta_2 | V \backslash (A \cup \{\beta_2\}) \Rightarrow A \perp \{\beta_1, \beta_2\} | V \backslash (A \cup \{\beta_1, \beta_2\}).$$

Successive applications of the intersection property to the variables in B give

$$A \perp B | S.$$

Now assume that $A \perp B | S$. Then, for each $\alpha \in A$ and $\beta \in B$, this may be rewritten as

$$\{\alpha\} \cup A \backslash \{\alpha\} \perp \{\beta\} \cup B \backslash \{\beta\} | S.$$

Using the **weak union** result, that $X \perp Y \cup Z|W \Rightarrow X \perp Y|Z \cup W$ it follows that

$$\alpha \perp B|S \cup A\backslash\{\alpha\}$$

and another application gives

$$\alpha \perp \beta|V\backslash\{\alpha,\beta\}.$$

□

Theorem 3.26. *Let $\mathbb{P}$ be a probability distribution that factorises along a DAG $\mathcal{G} = (V,D)$. Let $\mathcal{G}^{(m)} = (V,U^{(m)})$ denote its moral graph and let $\mathcal{G}^{(i)} = (V,U^{(i)})$ denote the independence graph of $\mathbb{P}$. Then $U^{(i)} \subseteq U^{(m)}$. Furthermore, if (V,D) is faithful to $\mathbb{P}$, then $U^{(i)} = U^{(m)}$.*

Proof From theorem 3.22, the moral graph has an edge $\langle X,Y\rangle$ if and only if $X \not\perp\!\!\!\perp Y\|_{\mathcal{G}}V\backslash\{X,Y\}$; there is no edge $\langle X,Y\rangle$ if and only if $X \perp\!\!\!\perp Y\|_{\mathcal{G}}V\backslash\{X,Y\}$. Since d separation implies conditional independence (theorem 2.10), it follows that the lack of an edge $\langle X,Y\rangle$ imples $X \perp Y|V\backslash\{X,Y\}$. From this, it follows directly that $U^{(i)} \subseteq U^{(m)}$.

For a faithful DAG, d-separation and conditional independence are equivalent, from which it follows that $U^{(i)} = U^{(m)}$ when $\mathbb{P}$ and $\mathcal{G} = (V,D)$ are faithful. □

If a distribution $\mathbb{P}$ does not have a faithful representation, then for any DAG $U^{(i)} \subset U^{(m)}$.

The following corollary is an obvious consequence of the preceeding.

Corollary 3.27. *Let $\mathcal{G} = (V,D)$ be a directed acyclic graph, along which a probability distribution $\mathbb{P}$ may be factorised and let $\mathcal{G}^{(m)}$ be the moral graph. Let $V = A \cup B \cup S$ where A, B and S are disjoint subsets. Then $A \perp\!\!\!\perp B\,[]\,S$ (A and B separated by S in $\mathcal{G}^{(m)}$) implies $A \perp B|S$ (A and B conditionally independent given S).*

Proof A clear consequence of the preceeding arguments. □

Notes The terminology *Markov model* corresponding to a Directed Acyclic Graph $\mathcal{G} = (V,E)$ was introduced into the literature and may be found in Andersson, Madigan, Perlman and Triggs [2]. The rules for determining *compelled* edges were formulated by Meek in [38]. A rigorous treatment covering chain graphs is [61] (Studený).

Chapter 4

The pioneering work of Arthur Cayley

Arthur Cayley F.R.S. (16 August 1821 - 26 January 1895) was a British mathematician, known for his work in pure mathematics. His contributions include the so-called Cayley-Hamilton theorem, that every square matrix satisfies its own characteristic polynomial, which he verified for matrices of order 2 and 3 (1858) [10]. He was the first to define the concept of a group in the modern way, as a set with a binary operation satisfying certain laws. From group theory, he is known for *Cayley's theorem*, which states that every group G is isomorphic to a subgroup of the symmetric group acting on G (1854) [9].

We do not discuss these aspects of his work; our attention is drawn to a short article by Arthur Cayley from 1853, where in an example that takes less than one page, he seems to develop several principles that later formed the basis of the subject of Bayesian networks. This article seems only to have been quoted five times since its publication.

We reproduce the article in its entirety.

> XXXVII. *Note on a Question in the Theory of Probabilities.*
>
> *By A. Cayley*.*
>
> The following question was suggested to me, either by some of Prof. Boole's memoirs on the subject of probabilities, or in conversation with him, I forget which; it seems to me a good instance of the class of questions to which it belongs.
>
> Given the probability α that a cause A will act, and the probability p that A acting the effect will happen; also the probability β that a cause B will act, and the probability q that B acting the effect will happen; required the total probability of the effect.
>
> As an instance of the precise case contemplated, take the following: say a day is called *windy* if there is at least w of wind, and a day is called *rainy* if there is at least r of rain, and a day is called *stormy* if there is at least W of wind, *or* if there is at least R of rain. The day may therefore be stormy because of there being at least W of wind, or because of there being at least R of rain, or on both accounts; but if there is less than W of wind *and* less than R of rain, the day will not be stormy. Then α is the probability that a day chosen at random will be windy, p the probability that a windy day chosen at random will

be stormy, β the probability that a day chosen at random will be rainy, q the probability that a rainy day chosen at random will be stormy. The quantities λ, μ introduced in the solution of the question mean in this particular instance, λ the probability that a windy day chosen at random will be stormy by reason of the quantity of wind, or in other words, that there will be at least W of wind, μ the probability that a rainy day chosen at random will be stormy by reason of the quantity of rain, or in other words, that there will be at least R of rain.

The sense of the terms being clearly understood, the problem presents of course no difficulty. Let λ be the probability that the cause A acting will act efficaciously; μ the probability that the cause B acting will act efficaciously; then

$$p = \lambda + (1 - \lambda)\mu\beta$$

$$q = \mu + (1 - \mu)\alpha\lambda,$$

which determine λ, μ; and the total probability ρ of the effect is given by

$$\rho = \lambda\alpha + \mu\beta - \lambda\mu\alpha\beta,$$

suppose, for instance, $\alpha = 1$, then

$$p = \lambda + (1 - \lambda)\mu\beta, \quad q = \mu + \lambda - \lambda\mu, \quad \rho = \lambda + \mu\beta - \lambda\mu\beta,$$

that is, $\rho = p$, for p is in this case the probability that (acting as a cause which is certain to act) the effect will happen, or what is the same thing, p is the probability that the effect will happen.

Machynlleth, August 16, 1853.

*Communicated by the Author.

In this short note, Cayley gives a prototype example of a causal network; rain and wind both have causal effects on the state of the day (stormy or not), which may be *inhibited*. He demonstrates the key principle of *modularity*, taking a problem with several variables and splitting it into its simpler component conditional probabilities, by considering the direct causal influences for each variable and considering the natural factorisation of the probability distribution in this problem into these conditional probabilities.

It should also be pointed out that Cayley was no stranger to graph theory; he proved *Cayley's tree formula*, that there are n^{n-2} distinct labelled trees of order n (1889) [13] and established links between graph theory and group theory, representing groups by graphs. The *Cayley graph* is named after him.

The variables here may be taken as

$$C = \begin{cases} 1 & \text{wind} \\ 0 & \text{no wind} \end{cases} \qquad D = \begin{cases} 1 & \text{rain} \\ 0 & \text{no rain} \end{cases}$$

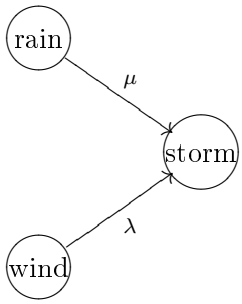

Figure 4.1: Rain and wind causing a storm

with

$$\alpha = p_C(1) \qquad \beta = p_D(1).$$

Let Y be the variable denoting whether there is a storm;

$$Y = \begin{cases} 1 & \text{storm} \\ 0 & \text{no storm} \end{cases}$$

Then, in Cayley's notation, if there is rain, it causes a storm with probability μ; if there is wind, it causes a storm with probability λ. The corresponding 'network', on three variables, is seen in figure 4.1. The subscripts μ and λ on the arrows indicate the probability that the cause, if active, will trigger the effect.

This is a *noisy 'or' gate*, which can be expressed as a logical 'or' gate by the addition of two variables, R and W. The variable R denotes *severe rain*, that is that the 'rain' variable reaches the threshold to trigger a storm. This happens if the quantity of rain is above a threshold. The W variable denotes *severe wind*; that is, that the 'wind' variable reaches the threshold to trigger a storm. This happens if the strength of wind is above a threshold. The variables, to form the logical or gate have conditional probability values given below; $p_{W|C}$ denotes the conditional probability function for the variable W given C and $p_{R|D}$ denotes the conditional probability function for the variable R given D.

$$p_{W|C} = \begin{array}{c|cc} C\backslash W & 1 & 0 \\ \hline 1 & \lambda & 1-\lambda \\ 0 & 0 & 1 \end{array} \qquad p_{R|D} = \begin{array}{c|cc} D\backslash R & 1 & 0 \\ \hline 1 & \mu & 1-\mu \\ 0 & 0 & 1 \end{array}$$

The network may now be expressed graphically according to figure 4.2. This DAG is a representation of the factorisation that Cayley is using;

$$p_{C,D,R,W,Y} = p_C p_D p_{R|C} p_{W|D} p_{Y|W,R}$$

where $p_{Y|W,R}$ denotes the conditional probability function for the variable Y, given W and R. For $Y = 1$, these values are given in the following table:

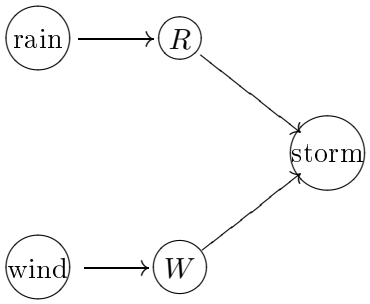

Figure 4.2: Rain and wind: logical 'or' gate

$$p_{Y|W,R}(1|.,.) = \begin{array}{c|cc} W\backslash R & 1 & 0 \\ \hline 1 & 1 & 1 \\ 0 & 1 & 0 \end{array}.$$

From the factorisation,

$$p_W(1) = \sum_x p_{W|C}(1|x)p_C(x) = \lambda\alpha, \quad p_R(1) = \mu\beta,$$

From Cayley, p is the probability that a windy day, chosen at random, will be stormy; $p = p_{Y|D}(1|1)$.

$$\begin{aligned} p &= p_{Y|D}(1|1) = \sum_{x_1} p_A(x_1) \sum_{x_2} p_{R|C}(x_2|x_1) \sum_{x_3} p_{Y|R,W}(1|x_2,x_3)p_{W|D}(x_3|1) \\ &= \beta\lambda\mu + \beta\mu(1-\lambda) + \beta(1-\mu)\lambda + (1-\beta)\lambda \\ &= \beta\mu - \beta\lambda\mu + \lambda = \lambda + (1-\lambda)\beta\mu. \end{aligned}$$

Similarly, q, the probability that a rainy day, chosen at random, will be stormy; $q = p_{Y|C}(1|1)$, is given by

$$q = \mu + (1-\mu)\alpha\lambda,$$

as computed by Cayley. Cayley is deriving the expression for the marginal probability of a stormy day, $\rho = p_Y(1)$;

$$\begin{aligned} p_Y(1) &= \sum_{x_1} p_C(x_1) \sum_{x_2} p_D(x_2) \sum_{x_3} p_{R|C}(x_3|x_1) \sum_{x_4} p_{W|D}(x_4|x_2)p_{Y|R,W}(1|x_3,x_4) \\ &= \sum_{x_3} p_R(x_3) \sum_{x_4} p_W(x_4)p_{Y|R,W}(1|x_3,x_4) \\ &= p_R(1)p_W(1) + p_R(1)p_W(0) + p_R(0)p_W(1) \\ &= \alpha\lambda + \beta\mu - \alpha\beta\lambda\mu. \end{aligned}$$

This simple construction from 1853 represents, to our knowledge, the first example of a causal network and the first construction of a noisy-or gate, with the concept of an inhibitor.

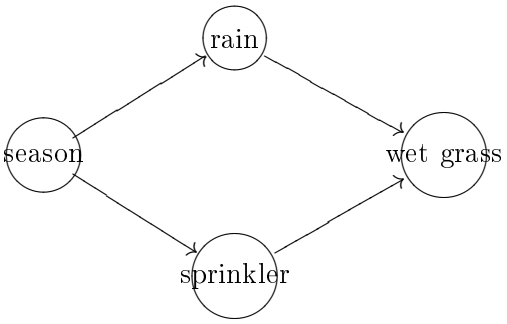

Figure 4.3: 'Sprinker and Wet Grass'

4.1 Arthur Cayley and Judea Pearl's intervention calculus

The rule of Bayes, for computing a conditional probability is

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A)\mathbb{P}(B|A)}{\mathbb{P}(B)}. \tag{4.1}$$

Starting with a well defined probability space $(\Omega, \mathcal{A}, \mathbb{P})$, the probability of an event $A \in \mathcal{A}$ has value $\mathbb{P}(A)$. An event $B \in \mathcal{A}$ is then *observed* and, using Bayes rule, one may compute the probability of the event A given the information that event B has been observed. The starting point is a probability space $(\Omega, \mathcal{A}, \mathbb{P})$, where $\mathcal{A}$, an algebra or σ-algebra, is the space of events over which $\mathbb{P}$ has been defined.

Bayes rule, equation (4.1) is no longer applicable if $B \notin \mathcal{A}$. The classic example is 'sprinkler and wet grass'. The variables are *season* (wet / dry), *rain* (yes / no), *sprinkler* (on / off), *wet grass* (yes / no). The corresponding DAG is given in figure 4.3.

If one *observes* that the sprinkler is on, one can infer that the season has been dry. If one knows that the state 'sprinkler on' has been forced, for reasons independent of the season or the amount of use (for example, if it is 'on' as a result of a regular maintenance procedure), then clearly no such inferences can be made; the additional information that regular maintenance work is taking place represents information above and beyond the simple observation that the sprinkler is on. The conditioning is no longer simply on event B.

Judea Pearl [50] proposed the following framework for computing conditional probabilities, when the conditioning is forced independently of other considerations in the network rather than observed. Pearl's framework requires two main assumptions:

1. There is a *causal* structure between the variables, which may be represented by a DAG, where the parent set represents the direct causes.

2. The intervention is made irrespective of the state of the system.

Pearl's framework is of particular value for describing a *controlled experiment* (as pointed out by D. Lindley [34]). Intervening to force the state of a variable, independently of other factors, amounts to a controlled experiment; individuals are randomly assigned to control group and treatment groups,

irrespective of other considerations. Links to other factors that may have been common causes for both the 'treatment' variable and chances of recovery have been removed.

Pearl's method can be summarised as follows: if an intervention is carried out whereby a variable X is forced to take a value x (written $X \leftarrow x$), irrespective of the state of the other variables, then the variable X is removed from the graph. That is, all edges to and from X are removed and the variable X is removed. For a variable Y, if $X \in \Pi_Y$ (the parent set of Y) in the original graph, the conditional probability $\mathbb{P}_{Y|\Pi_Y}$ with the instantiation $X = x$ in the parent set Π_Y is used.

In terms of the factorisation, suppose that

$$\mathbb{P}_{X_1,\ldots,X_d} = \prod_{j=1}^{d} \mathbb{P}_{X_j|\Pi_j}$$

is obtained from causal principles, where variables of lower order have causal effect on variables of higher order. Then the probability after an intervention $X_i \leftarrow x_i$ is defined as

$$\mathbb{P}_{X_1,\ldots X_d \| X_i} = \prod_{j \neq i} \mathbb{P}_{X_j|\Pi_j}$$

where the instantiation x_i is used for X_i when X_i appears in Π_j. This is known in the literature as 'do'-conditioning. There is the cryptic remark towards the end of Arthur Cayley's paper, which indicates that he may already had this framework in mind when considering causal probabilistic models. The phrase ' .... acting a cause which is certain to act' may be a clumsy way of expressing a brilliant insight into the intervention calculus, if by 'acting' he means intervening to force the state of the variable.

This reading may be somewhat strained; in Arthur Cayley's example, no human intervention is possible to force the states of the wind or rain variables. Since 'wind' and 'rain' are both ancestor variables, no links are removed from the DAG and in Pearl's framework, intervention conditioning is the same as the standard conditioning on an observation. The wording suggests, though, that he understood, from causal principles, that the two equations relating λ and μ to p and q remain valid if the conditioning on an ancestor variable is forced by intervention, rather than simply observed, one of the features of Pearl's intervention calculus.

4.2 Arthur Cayley: algebraic geometry and Bayesian networks

The emerging field of algebraic statistics advocates polynomial algebra as a tool in the statistical analysis of experiments and discrete data.

Recall the conditional independence axioms satisfied by conditional independence statements and those statements satisfied by d-separation statements for sets of variables in a DAG. A *factorisation* is equivalent to a set of conditional independence statements;

$$\{X_{\sigma(j)} \perp \Xi_j^\sigma | \Pi_j^\sigma \quad j = 1, \ldots, d\},$$

where $\Pi_j^\sigma \subset \{X_{\sigma(1)}, \ldots, X_{\sigma(j-1)}\}$ is the parent set of variable $X_{\sigma(j)}$ when ordering σ is employed and $\Xi_j^\sigma = \{X_{\sigma(1)}, \ldots, X_{\sigma(j-1)}\} \backslash \Pi_j^\sigma$.

Let $V = \{X_1, \ldots, X_d\}$ denote the variable set, let $\underline{X} = (X_1, \ldots, X_d)$ the random vector, let the state space for variable X_j be $\mathcal{X}_j = \{x_j^{(1)}, \ldots, x_j^{(k_j)}\}$ and the state space for $\underline{X}$ be $\mathcal{X} = \times_{j=1}^{d} \mathcal{X}_j$. Let $\mathcal{Y} = \times_{j=1}^{d}(1, \ldots, k_j)$ and $\mathbb{R}(\mathcal{Y})$ the ring of polynomial functions on $\mathbb{R}^{\mathcal{Y}}$.

A conditional independence statement $A \perp B|C$, where A, B and C are disjoint subsets of V, translates using proposition 8.1 from Sturmfels (2002) [62], into a set of homogeneous quadratic polynomials on $\mathbb{R}(\mathcal{Y})$, and these polynomials generate an *ideal*. Let $\mathcal{I}_{A\perp B|C}$ denote the ideal generated by the statement $A \perp B|C$. The ideal for a collection of independence statements, for example those corresponding to a factorisation, is defined as the sum of the ideals; let $\mathcal{M} = \{A_i \perp B_i|C_i \qquad i = 1, \ldots, m\}$, then

$$\mathcal{I}_{\mathcal{M}} = \mathcal{I}_{A_1\perp B_1|C_1} + \ldots + \mathcal{I}_{A_m\perp B_m|C_m}.$$

Cayley is using the expression of the conditional independence statements that define the factorisation in terms of polynomials to obtain the two polynomial equations

$$\begin{cases} p = \lambda + (1-\lambda)\mu\beta \\ q = \mu + (1-\mu)\lambda\alpha \end{cases} \tag{4.2}$$

and writes, '.... which determine λ and μ'. This amounts to finding roots of the two polynomials in λ, μ

$$\begin{cases} f_1(\lambda,\mu) = \lambda + (1-\lambda)\mu\beta - p \\ f_2(\lambda,\mu) = \mu + (1-\mu)\lambda\alpha - q \end{cases}$$

In terms of algebraic geometry, equation (4.2) defines the *affine variety*

$$V(f_1, f_2) = \left\{(\lambda,\mu) \in \mathbb{R}^2 | f_1(\lambda,\mu) = f_2(\lambda,\mu) = 0\right\}.$$

In his brief note, Cayley has pointed out the connections between Bayesian networks and algebraic geometry, a subject that he knew well. Cayley did much to clarify a large number of interrelated theorems in algebraic geometry and is known for the Cayley surface (1869) [11].

Chapter 5

Data Storage, Product Approximations, Chow Liu Trees

5.1 Introduction

Let $\underline{X} = (X_1, \ldots, X_d)$ denote a random vector with probability function $\mathbb{P}_{X_1,\ldots,X_d}$. Let

$$\mathcal{X}_j = (x_j^{(1)}, \ldots, x_j^{(k_j)})$$

denote the state space of X_j, $j = 1, \ldots, d$ and let

$$\mathcal{X} = \times_{j=1}^{d} \mathcal{X}_j.$$

The number of elements in the state space is $|\mathcal{X}| = \left(\prod_{j=1}^{d} k_j\right)$ and, without further assumptions on $\mathbb{P}$, $|\mathcal{X}| - 1$ elements are required to store the entire distribution.

The problem of storing the entire probability distribution is another expression of the 'curse of dimensionality'. The size of the problem is reduced if one instead stores lower dimensional marginals and approximates the distribution by an appropriate product of lower dimensional marginals.

The topic of storing a high dimensional discrete probability distribution in a digital medium appeared in the journal literature, probably for the first time, by J. Hartmanis (1959) [29] and P.M. Lewis II (1959) [33]. The Chow - Liu tree by Chow and Liu (1969) [17], approximately 10 years later, provides an influential and effective solution to the problem. Chow and Liu gave an algorithm for selecting second order factors for the product approximation so that among all such second order approximations, the constructed approximation has the minimum Kullback-Leibler distance to the actual distribution to be stored.

5.2 Product Approximations

5.2.1 Existence of Extensions with Given Marginals

For a set of random variables $\underline{X} = (X_1, \ldots, X_d)$, there are $\sum_{j=1}^{d-1} \binom{d}{j} = 2^d - 2$ lower dimensional marginal distributions. The *classical marginal problem* goes the other way round: given a subfamily $(\mathbb{P}_{W_i})_{i=1}^{s}$,

for $s < 2^d - 2$ and $W_i \subset V = \{X_1, \ldots, X_d\}$, the question is whether there exist a probability distribution $\mathbb{P}_V$ that satisfies the so-called *collective compatibility condition* given by equation (5.1).

$$\mathbb{P}_{W_i} = (\mathbb{P}_V)^{\downarrow W_i} \qquad \forall i = 1, \ldots, s \tag{5.1}$$

Here the notation $\downarrow A$ means the marginalisation down to a set of variables A. Some fundamental contributions to this problem are due to H.G. Kellerer [32] and others.

If the sets $(W_i)_{i=1}^s$ are disjoint, satisfying $\cup_i W_i = V$, then the problem has an obvious trivial solution:

$$\mathbb{P}(\underline{x}) = \prod_{i=1}^{s} \mathbb{P}_{W_i}(\underline{x}_{W_i})$$

where the product operation means first extending the probabilities $\mathbb{P}_{W_i}$ as functions, to functions $\tilde{\mathbb{P}}_{W_i}$ over the domain V where (using obvious notation) $\tilde{\mathbb{P}}_{W_i}(\underline{x}_{W_i}.\underline{x}_{V \setminus W_i}) = \mathbb{P}_{W_i}(\underline{x}_{W_i})$ for each $\underline{x}_{W_i} \in \mathcal{X}_{W_i}$ and then multiplying. With the appropriate projections of $\underline{x}$,

$$\mathbb{P}(\underline{x}) = \prod_{j=1}^{s} \mathbb{P}_{W_j}(\underline{x}_{W_j}).$$

If $(W_j)_{j=1}^s$ are not disjoint, then clearly the collection of probabilities $(\mathbb{P}_{W_j})_{j=1}^s$ should satisfy a *pairwise compatibility* condition:

$$\mathbb{P}_{C_{ij}} = \mathbb{P}_{W_i}^{\downarrow C_{ij}} = \mathbb{P}_{W_j}^{\downarrow C_{ij}} \qquad \forall i, j \in \{1, \ldots, s\}^2.$$

The following example due to Vorobev (1962) [64] shows that pairwise compatibility does not imply collective compatibility.

Example 5.1 (Vorobev's example)**.**

Let $V = \{1, 2, 3\}$, $W_1 = \{2, 3\}$, $W_2 = \{1, 3\}$, $W_3 = \{1, 2\}$. Suppose that the following three pairwise joint distributions are specified:

$\mathbb{P}_{W_1}(x_2, x_3) =$

$x_2 \backslash x_3$	0	1
0	$\frac{1}{2}$	0
1	0	$\frac{1}{2}$

$\mathbb{P}_{W_2}(x_3, x_1) =$

$x_1 \backslash x_3$	0	1
0	0	$\frac{1}{2}$
1	$\frac{1}{2}$	0

$\mathbb{P}_{W_3}(x_1, x_2) =$

$x_1 \backslash x_2$	0	1
0	$\frac{1}{2}$	0
1	0	$\frac{1}{2}$

These are pairwise compatible; $W_1 \cap W_2 = \{3\}$ and

$$\mathbb{P}_{W_1}^{\downarrow\{3\}}(x_3) = \mathbb{P}_{W_2}^{\downarrow\{3\}}(x_3) = \begin{array}{cc} 0 & 1 \\ \hline \frac{1}{2} & \frac{1}{2} \end{array}.$$

$W_1 \cap W_3 = \{2\}$ and

$$\mathbb{P}_{W_1}^{\downarrow\{2\}}(x_2) = \mathbb{P}_{W_3}^{\downarrow\{2\}}(x_2) = \begin{array}{cc} 0 & 1 \\ \hline \frac{1}{2} & \frac{1}{2} \end{array}.$$

$W_2 \cap W_3 = \{1\}$ and

$$\mathbb{P}_{W_2}^{\downarrow\{1\}}(x_1) = \mathbb{P}_{W_3}^{\downarrow\{1\}}(x_1) = \begin{array}{cc} 0 & 1 \\ \hline \frac{1}{2} & \frac{1}{2} \end{array}.$$

If a common extension $\mathbb{P}^*$ existed, it would follow that (for example)

$$\frac{1}{2} = \mathbb{P}_{W_1}(0,0) = \mathbb{P}^*(0,0,0) + \mathbb{P}^*(1,0,0) \leq \mathbb{P}_{W_2}(0,0) + \mathbb{P}_{W_3}(1,0) = 0,$$

which is a contradiction. The three marginals satisfy a pairwise compatibility condition, but not a collective compatibility condition. □

Without loss of generality, let $V = \cup_{j=1}^{s} W_j$. The condition to ensure that pairwise compatibility implies collective compatibility is known as the *acyclic condition.*

Definition 5.2. *Suppose that there is an ordering of the sets $W_1, \ldots, W_s$ such that for each j there is an $l < j$ such that*

$$B_j = W_j \cap \left(\cup_{k=1}^{j-1} W_k\right) \subseteq W_j \cap W_l \tag{5.2}$$

This property is known as the running intersection property. *A set of subsets of $W_1, \ldots, W_s$ having the running intersection property, given some ordering, is known as* acyclic.

Remark In example 5.1, if the ordering W_1, W_2, W_3 is chosen, then

$$W_3 \cap (W_1 \cup W_2) = \{1, 2\},$$

but $\{1, 2\}$ is not a subset of W_1 or W_2. It follows that equation (5.2) does not hold for this ordering. It is easy to check that there is no ordering that satisfies equation (5.2), hence acyclicity does not hold for example 5.1.

It is shown in Beeri et. al. (1983) [4] that acyclicity is *equivalent* to 'pairwise compatibility for all (i, j) implies collective compatibility'. Under acyclicity, it is straightforward to show that there is a unique product form extension,

$$\mathbb{P}^*(\underline{x}) = \frac{\prod_{l=1}^{s} \mathbb{P}(\underline{x}_{W_l})}{\prod_{h=1}^{s-1} \mathbb{P}(\underline{x}_{V_h})} \tag{5.3}$$

where each V_h is the intersection of two or more W_l. The uniqueness requires that $\mathbb{P}_{W_i}(\underline{x}_{W_i}) > 0$ for all $\underline{x}_{W_i}$ and all $i = 1, \ldots, s$.

5.2.2 Dependence Structures

Let $W_1, \ldots, W_s$ be sets of random variables, $V = \cup_{j=1}^{s} W_j$ and suppose that $W_1, \ldots, W_s$ satisfy the running intersection property of equation (5.2). With this ordering, set $B_1 = \phi$ and

$$B_j = W_j \cap \left(\cup_{k=1}^{j-1} W_k\right), \qquad j = 2, \ldots, k.$$

Let $A_j = W_j \backslash B_j$ so that $W_j = A_j \cup B_j$. It follows that $A_1, \ldots, A_s$ is a *partition* of V and that the sets $(B_j)_{j=1}^{s}$ satisfy

$$B_j \subset \cup_{i=1}^{j-1} A_i.$$

Definition 5.3 (Dependence Structure). *Let $(A_i)_{i=1}^k$ be a partition of a set V and let S be a sequence of pairs of subsets of V, $S = (A_i, B_i)_{i=1}^k$ satisfying*

$$B_1 = \phi, \qquad B_r \subset \cup_{i=1}^{r-1} A_i \subseteq V, \qquad r = 2, \ldots, k$$

Then S is a dependence structure.

Definition 5.4 (Product Approximation). *Let S be a dependence structure. Then the probability distribution defined by*

$$\mathbb{P}^{(S)}(\underline{x}) = \mathbb{P}_{A_1}(\underline{x}_{A_1}) \prod_{j=2}^{k} \mathbb{P}_{A_j|B_j}(\underline{x}_{A_j}|\underline{x}_{B_j})$$

is called the product approximation *of the probability distribution $\mathbb{P}$ determined by S.*

A *product approximation* is clearly a well defined *probability* distribution. Furthermore, it satisfied the following compatibility condition:

Lemma 5.5.

$$\mathbb{P}^{(S)\downarrow A_j \cup B_j}(\underline{x}_{A_j \cup B_j}) = \mathbb{P}_{A_j|B_j}(\underline{x}_{A_j}|\underline{x}_{B_j})\mathbb{P}(\underline{x}_{B_j}) \qquad \forall \underline{x} \in \mathcal{X}, \qquad j = 1, \ldots, s.$$

Proof By marginalising over $A_{j+1} \cup \ldots \cup A_s$,

$$\mathbb{P}^{(S)\downarrow A_1 \cup \ldots \cup A_j}(\underline{x}_{A_1 \cup \ldots \cup A_j}) = \mathbb{P}_{A_1}(\underline{x}_{A_1}) \prod_{k=2}^{j} \mathbb{P}_{A_k|B_k}(\underline{x}_{A_k}|\underline{x}_{B_k}) \qquad j = 1, \ldots, s$$

so that

$$\begin{aligned}
\mathbb{P}^{(S)\downarrow A_j \cup B_j}(\underline{x}_{A_j \cup B_j}) &= \mathbb{P}_{A_j|B_j}(\underline{x}_{A_j}|\underline{x}_{B_j}) \sum_{A_1 \cup \ldots \cup A_{j-1} \setminus B_j} \mathbb{P}_{A_1}(\underline{x}_{A_1}) \prod_{k=2}^{j-1} \mathbb{P}_{A_k|B_k}(\underline{x}_{A_k}|\underline{x}_{B_k}) \\
&= \mathbb{P}_{A_j|B_j}(\underline{x}_{A_j}|\underline{x}_{B_j}) \sum_{A_1 \cup \ldots \cup A_{j-1} \setminus B_j} \mathbb{P}^{(S)\downarrow A_1 \cup \ldots \cup A_{j-1}}(\underline{x}_{A_1 \cup \ldots \cup A_{j-1}}) \\
&= \mathbb{P}_{A_j|B_j}(\underline{x}_{A_j}|\underline{x}_{B_j})\mathbb{P}^{(S)\downarrow B_j}(\underline{x}_{B_j})
\end{aligned}$$

as required. □

Note that if $W_i = A_i \cup B_i$ for $i = 1, \ldots, s$ and $\mathbb{P}_{W_i}$ are given, then

$$\mathbb{P}^{(S)} = \frac{\prod_{i=1}^{s} \mathbb{P}_{W_i}}{\prod_{i=2}^{s} \mathbb{P}_{B_i}}$$

where $B_i = W_i \cap \cup_{j=1}^{i-1} W_j$ and the convention $\mathbb{P}_\phi \equiv 1$ is used. In this situation, clearly

$$\mathbb{P}^{(S)W_j} = \mathbb{P}_{W_j}.$$

It follows directly from this factorisation that

$$A_j \perp \cup_{k=1}^{j-1} A_k \setminus B_j |_{\mathbb{P}^{(S)}} B_j.$$

5.2.3 Classification

Many of the techniques of supervised learning, or classification, involve a Bayes rule and an approximate distribution. Variables are of two types, *symptom* variables $\underline{X}_O$ (O for observable) and *class* variables, or *diagnosis* variables, $\underline{X}_C$. A prior distribution $\mathbb{P}_C$ is placed over the class variables, evidence is obtained in the form of an instantiation $\underline{x}_O$ of $\underline{X}_O$ of the symptom variables and the posterior distribution over the class variables obtained using Bayes rule;

$$\mathbb{P}_{C|O} = \frac{\mathbb{P}_C \mathbb{P}_{O|C}}{\mathbb{P}_O} \propto \mathbb{P}_C \mathbb{P}_{O|C}.$$

In *supervised* classification, the probabilities $\mathbb{P}_{O|C}$ are *learned*, by observing the instantiations $\underline{x}_O$ in training examples where $\underline{x}_C$ is given. When classifying (where the class $\underline{x}_C$ is unknown, the class that maximises $\mathbb{P}_C \mathbb{P}_{O|C}$ is chosen, for a given set of symptoms $\underline{x}_O$.

Often in classification, the distribution $\mathbb{P}_{O|C}$ has too many states and instead a set of lower dimensional marginals is considered:

$$\mathcal{P}(\underline{x}_C) = \{\mathbb{P}_{A_j|B_j,C}(.|.,\underline{x}_C) \qquad j = 1, \ldots, s\}$$

where for each $\underline{x}_C$, $S_C := (A_j, B_j)_{j=1}^s$ is a dependence structure. The dependence structures may depend on $\underline{x}_C$. The class variable $\underline{x}_C$ is then chosen to maximise

$$\mathbb{P}_C \mathbb{P}_{A_1|C} \prod_{j=1}^{s} \mathbb{P}_{A_j|B_j,C}.$$

5.3 Reverse I-Projection and the Optimal Product Approximation

The *relative entropy*, or *information divergence*, or I-*divergence*, or *Kullback Leibler distance*, written $D_{KL}(\mathbb{P}\|\mathbb{P}^{(S)})$, is defined by

$$D_{KL}(\mathbb{P}\|\mathbb{P}^{(S)}) = \sum_{\underline{x}\in\mathcal{X}} \mathbb{P}(\underline{x}) \ln \frac{\mathbb{P}(\underline{x})}{\mathbb{P}^{(S)}(\underline{x})}. \tag{5.4}$$

The task of *Optimal Product Representation* of $\mathbb{P}$ is, for a given dependence structure S, to find a $\mathbb{P}_S^*$ such that $D(\mathbb{P}\|\mathbb{P}_S)$ is minimised. The solution $\mathbb{P}_S^*$ is called a *Reverse I-Projection* of $\mathbb{P}$ onto the set of all probability measures with S as a dependence structure.

Definition 5.6 (Shannon Entropy)**.** *Let $A \subseteq V$. The* Shannon entropy *of the set of variables A for a probability distribution $\mathbb{P}$ is defined as*

$$H_{\mathbb{P}}(A) := -\sum_{\underline{x}_A\in\mathcal{X}_A} \mathbb{P}_A(\underline{x}_A) \ln \mathbb{P}_A(\underline{x}_A)$$

where $\mathbb{P}_A = \mathbb{P}^{\downarrow A}$.

With $A = V$, it follows that, for a probability distribution $\mathbb{Q}$,

$$D_{KL}(\mathbb{P}\|\mathbb{Q}) = \sum_{\underline{x}\in\mathcal{X}} \mathbb{P}(\underline{x}) \ln \mathbb{P}(\underline{x}) - \sum_{\underline{x}\in\mathcal{X}} \mathbb{P}(\underline{x}) \ln \mathbb{Q}(\underline{x}) = -H(V) - \sum_{\underline{x}\in\mathcal{X}} \mathbb{P}(\underline{x}) \ln \mathbb{Q}(\underline{x}).$$

For a dependence structure $S = (A_i, B_i)_{i=1}^s$ and a probability distribution $\mathbb{Q}$ that factorises according to: $\mathbb{Q} = \prod_{i=1}^s \mathbb{Q}_{A_i|B_i}$, it is straightforward to compute that

$$\begin{aligned} D_{KL}(\mathbb{P}\|\mathbb{Q}) &= -H_{\mathbb{P}}(V) - \sum_{i=1}^{s} \sum_{\underline{x}_{A_i\cup B_i}} \mathbb{P}_{A_i\cup B_i}(\underline{x}_{A_i\cup B_i}) \ln \mathbb{Q}_{A_i|B_i}(\underline{x}_{A_i}|\underline{x}_{B_i}) \\ &= -H_{\mathbb{P}}(V) - \sum_{i=1}^{s} \sum_{\underline{x}_{B_i}} \sum_{\underline{x}_{A_i}} \mathbb{P}_{A_i|B_i}(\underline{x}_{A_i}|\underline{x}_{B_i}) \ln \mathbb{Q}_{A_i|B_i}(\underline{x}_{A_i}|\underline{x}_{B_i}). \end{aligned}$$

Now use *Gibb's inequality*; for any two probability distributions $\underline{f}$ and $\underline{g}$ over the same state space,

$$\sum_{j=1}^{L} f_j \ln f_j \geq \sum_{j=1}^{L} f_j \ln g_j. \tag{5.5}$$

This follows from the fact that

$$D_{KL}(\underline{f}\|\underline{g}) = \sum_{j=1}^{L} f_j \ln \frac{f_j}{g_j} \geq 0$$

with equality if and only if $\underline{f} = \underline{g}$. It follows that the *reverse I-projection of* $\mathbb{P}$ *onto a dependency structure* $S = (A_i, B_i)_{i=1}^s$ is

$$\mathbb{P}^{(S)} = \prod_{i=1}^{s} \mathbb{P}_{A_i|B_i}$$

and

$$D_{KL}(\mathbb{P}\|\mathbb{P}^{(S)}) = -H(V) + \sum_{i=1}^{k} \left(H(A_i \cup B_i) - H(B_i)\right).$$

Definition 5.7 (Mutual Information)**.** *The* mutual information $I(A, B)$ *between two disjoint sets of variables* A *and* B *is defined as*

$$I(A, B) = H(A) + H(B) - H(A \cup B).$$

This may be written as

$$I(A, B) = \sum \mathbb{P}_{A\cup B}(\underline{x}_{A\cup B}) \ln \frac{\mathbb{P}_{A\cup B}(\underline{x}_{A\cup B})}{\mathbb{P}_A(\underline{x}_A)\mathbb{P}_B(\underline{x}_B)} = D_{KL}(\mathbb{P}_{A\cup B}\|\mathbb{P}_A\mathbb{P}_B).$$

Note that $I(A, B) = 0 \Leftrightarrow \underline{X}_A \perp \underline{X}_B$.

If one is choosing a dependence structure $S = (A_i, B_i)_{i=1}^s$, from within a class $\mathcal{S}$ of dependence structures with the same storage properties, it follows that the dependence structure $S = (A_i, B_i)_{i=1}^s$ is chosen to maximise

$$Q(S) = -\sum_{i=1}^{k} H(A_i) + \sum_{i=1}^{k} I(A_i, B_i).$$

5.4 The Optimal Chow-Liu Product Approximation

For a Chow Liu tree, the dependence structure $(A_i, B_i)_{i=1}^k$ satisfies

$$|A_i \cup B_i| \leq 2 \qquad i = 1, \ldots, k.$$

Let $\mathcal{G} = (V, U)$ denote an undirected graph, where $V = \{1, \ldots, d\}$ is the indexing set for the nodes and U is the undirected edge set. An undirected graph $\mathcal{G}$ is *complete* if $U = \{\langle i, j\rangle : 1 \leq i < j \leq d\}$. The *degree of a node* i is defined as the number of distinct edges containing the node i.

A *subgraph* $\mathcal{H}$ of $\mathcal{G}$ is a graph (V_1, U_1) where $V_1 \subseteq V$ and $U_1 \subseteq U$. A subgraph V_1 is *induced* by $A \subset V$ if $V_1 = A$ and $U_1 = U \cap A \times A$. A subgraph $\mathcal{H}$ is a *spanning subgraph* of $\mathcal{G}$ if it is connected and $V_1 = V$.

An undirected tree $\mathcal{T}$ is a connected undirected graph that has no cycles. It follows that there is a unique path between any two nodes. A *spanning tree of a graph* is a spanning graph of $\mathcal{G}$ which is a tree.

A *labelled tree* is a tree on d nodes where each node is labelled by one of the integers $\{1, \ldots, d\}$. In the sequel, labelled trees will be referred to as trees.

A *weighted undirected graph* is

$$\mathcal{G} = ((V, U)|\mathbf{w})$$

where $\mathbf{w} : U \to \mathbb{R}_+$ (non negative real numbers). The weight of a tree is the sum of its edge weights. The weight to be used by the Chow-Liu algorithm will be defined via the mutual information

$$\mathbf{w}(i, j) := I(j, k) = H(j) + H(k) - H(j \cup k).$$

5.4.1 Chow Liu Tree with known $\mathbb{P}$

Definition 5.8 (Chow-Liu Dependence Structure)**.** *Let $(i_r)_{r=1}^d$ be an arbitrary permutation of $V = \{1, \ldots, d\}$. The singleton sets $A_r = \{i_r\}$ $r = 1, \ldots, d$ are a partition of V. Let σ be a sequence of pairs of singletons of V, $\sigma = (i_r, j_r)_{r=1}^d$, where*

$$j_1 = \phi, \qquad j_r \in \{i_1, \ldots, i_{r-1}\} \subseteq V \qquad r = 2, \ldots, d.$$

Then σ is a Chow-Liu dependence structure.

A Chow-Liu dependence structure will give a tree. Since the tree connects all the nodes, it is a *spanning tree*. Arrows are directed *from j_r to i_r*. If $j_r = \phi$, there is no arrow pointing to the node i_r. Any node in a *directed* tree with $j_r = \phi$ is called a root. By construction, i_1 is the only root. A tree with exactly one root is said to be *proper*.

Note that

$$i_r \perp_{\mathbb{P}(S)} \{i_1, \ldots, i_{r-1}\} \backslash \{j_r\} | j_r.$$

For σ thus defined,

$$Q(\sigma) = -\sum_{r=1}^{d} H(i_r) + \sum_{r=1}^{d} I(i_r, j_r).$$

The Chow-Liu dependence structure defines a product approximation of a known probability distribution $\mathbb{P}$ by

$$\mathbb{P}^{(\sigma)}(\underline{x}) = \mathbb{P}_{i_1}(x_{i_1}) \prod_{r=2}^{d} \mathbb{P}_{i_r|j_r}(x_{i_r}|x_{j_r}).$$

The following theorem is the first main result in [17].

Theorem 5.9. *Let $\mathbb{P}$ be a probability distribution over $\mathcal{X}$. Let $\mathcal{G} = ((V,U)|\mathbf{w})$ be a complete weighted graph with $\mathbf{w}$ given by*

$$\mathbf{w}(j,k) = I(j,k) \qquad \langle j,k\rangle \in U$$

where the $I(j,k)$s are computed using the $\mathbb{P}_{j,k}$s. Then the maximum weight spanning tree of $\mathcal{G}$ defines a Chow-Liu dependence structure σ, which maximises

$$Q(\sigma) = -\sum_{r=1}^{d} H(i_r) + \sum_{r=2}^{d} I(i_r, j_r).$$

Proof Firstly, $\sum_{r=1}^{d} H(i_r) = \sum_{i=1}^{d} H(i)$ so that the first term in $Q(\sigma)$ is independent of σ, hence the problem is equivalent to the maximisation of $\sum_{r=1}^{d} I(i_r, j_r)$. □

5.4.2 Chow-Liu Algorithm with Unknown $\mathbb{P}$

For $\mathbb{P}$ unknown, suppose there is an $n \times d$ data matrix $\mathbf{x}$, where $\mathbf{x} = \begin{pmatrix} \underline{x}_{(1)} \\ \vdots \\ \underline{x}_{(n)} \end{pmatrix}$. Each $\underline{x}_{(j)} \in \mathcal{X}$ for $j = 1, \ldots, n$.

Let $\mathcal{P}(\mathcal{X})$ denote the space of all probability distributions over $\mathcal{X}$; that is

$$\mathcal{P}(\mathcal{X}) = \{\mathbb{P}|\mathbb{P} = \{\mathbb{P}(\underline{x})\}_{\underline{x}\in\mathcal{X}}\}$$

Let $\mathcal{T}_d = (V, \sigma)$ be a spanning tree on V, where σ is a Chow-Liu dependence structure. Let $\mathbb{T}_d$ denote the set of all spanning trees, then

$$\mathcal{P}(\mathcal{X}, \mathbb{T}_d) = \{\mathbb{P}^{(\sigma)}\}$$

is the set of all tree dependent probability distributions on $\mathcal{X}$ and $\mathcal{P}(\mathcal{X}, \mathbb{T}_d) \subset \mathcal{P}(\mathcal{X})$. The empirical probability is defined as

$$\widehat{\mathbb{P}}_n(\underline{x}) = \frac{1}{n} \sum_{k=1}^{n} \mathbf{1}_{\underline{x}}(\underline{x}_{(k)}).$$

Lemma 5.10. *The maximum likelihood estimate $\widehat{\mathbb{P}}^{(ML)}$ is given by*

$$\widehat{\mathbb{P}}^{(ML)} = \arg \min_{\mathbb{P}\in\mathcal{P}(\mathcal{X},\mathbb{T}_d)} D_{KL}(\widehat{\mathbb{P}}_n \| \mathbb{P}).$$

Proof

$$D_{KL}(\widehat{\mathbb{P}}_n \| \mathbb{P}) = -\widehat{H}_n(V) - \sum_{\underline{x} \in \mathcal{X}} \widehat{\mathbb{P}}_n(\underline{x}) \ln \mathbb{P}(\underline{x}),$$

where $\widehat{H}_n(V) = -\sum_{\underline{x} \in \mathcal{X}} \widehat{\mathbb{P}}_n(\underline{x}) \ln \widehat{\mathbb{P}}_n(\underline{x})$. Note that this does not depend on the tree structure. For the other part,

$$\sum_{\underline{x} \in \mathcal{X}} \widehat{\mathbb{P}}_n(\underline{x}) \ln \mathbb{P}(\underline{x}) = \frac{1}{n} \sum_{j=1}^{n} \ln \mathbb{P}(\underline{x}_{(j)}),$$

which is the log likelihood function. Hence, the maximum likelihood estimate is equivalent to the *reverse I-projection* of $\widehat{\mathbb{P}}_n$ onto the set of tree dependent distributions $\mathcal{P}(\mathcal{X}, \mathbb{T}_d)$. □

5.4.3 The Log Likelihood Function

When $\sigma = (i_r, j_r)_{r=1}^{d}$ is a Chow-Liu dependence structure, the parameter set $\mathcal{P}$ is the set of two dimensional distributions given by

$$\mathcal{P} = \{\mathbb{P}_{i,j} | (i,j) \in V \times V, i \neq j\}.$$

The corresponding parametric probability is

$$\mathbb{P}^{(\sigma)}(\underline{x}) = \mathbb{P}_{i_1}(x_{i_1}) \prod_{r=2}^{d} \mathbb{P}_{i_r|j_r}(x_{i_r}|x_{j_r}) = \prod_{r=1}^{d} \mathbb{P}_{i_r}(x_{i_r}) \prod_{r=2}^{d} \frac{\mathbb{P}_{i_r,j_r}(x_{i_r}, x_{j_r})}{\mathbb{P}_{i_r}(x_{i_r}) \mathbb{P}_{j_r}(x_{j_r})} \qquad \underline{x} = (x_j)_{j=1}^{d} \in \mathcal{X}.$$

The likelihood function is therefore

$$L(\sigma, \mathcal{P}) = \prod_{j=1}^{n} \mathbb{P}^{(\sigma)}(\underline{x}_{(j)} | \mathcal{P})$$

and the log likelihood function, divided by n, is

$$\mathcal{L}(\sigma, \mathcal{P}) = \frac{1}{n} \sum_{j=1}^{n} \ln \mathbb{P}(\underline{x}_{(j)} | \sigma, \mathcal{P}).$$

which may be re-written as

$$\mathcal{L}(\sigma, \mathcal{P}) = \frac{1}{n} \sum_{x \in \mathcal{X}_{i_1}} N(x) \ln \mathbb{P}_{i_1}(x) + \frac{1}{n} \sum_{r=2}^{d} \sum_{(x,y) \in \mathcal{X}_{i_r} \times \mathcal{X}_{j_r}} N(x,y) \ln \mathbb{P}_{i_r,j_r}(x,y) - \frac{1}{n} \sum_{j=2}^{d} \sum_{x \in \mathcal{X}_{j_r}} N(x) \ln \mathbb{P}_{j_r}(x)$$

where $N(x)$ denotes number of appearances of the appropriate configuration x in the data matrix $\mathbf{x}$. This reduces to

$$\mathcal{L}(\sigma, \mathcal{P}) = \sum_{x \in \mathcal{X}_{i_1}} \widehat{\mathbb{P}}_{n;i_1}(x) \ln \mathbb{P}_{i_1}(x) + \sum_{r=2}^{d} \sum_{(x,y) \in \mathcal{X}_{i_r} \times \mathcal{X}_{j_r}} \widehat{\mathbb{P}}_{n;i_r,j_r}(x,y) \ln \mathbb{P}_{i_r,j_r}(x,y) - \sum_{j=2}^{d} \sum_{x \in \mathcal{X}_{j_r}} \widehat{\mathbb{P}}_{n;j_r}(x) \ln \mathbb{P}_{j_r}(x).$$

Using the notation $\mathbb{P}_{i_r|j_r} = \frac{\mathbb{P}_{i_r,j_r}}{\mathbb{P}_{j_r}}$, this may be written as

$$\mathcal{L}(\sigma, \mathcal{P}) = \sum_{x \in \mathcal{X}_{i_1}} \widehat{\mathbb{P}}_{n;i_1}(x) \ln \mathbb{P}_{i_1}(x) + \sum_{r=2}^{d} \sum_{y \in \mathcal{X}_{j_r}} \widehat{\mathbb{P}}_{n;j_r}(y) \sum_{x \in \mathcal{X}_{i_r}} \widehat{\mathbb{P}}_{n;i_r|j_r}(x,y) \ln \mathbb{P}_{i_r|j_r}(x|y). \tag{5.6}$$

The log likelihood $\mathcal{L}(\sigma, \mathcal{P})$ is to be maximised. For a fixed structure σ, it therefore follows from Gibb's inequality that the maximum likelihood estimates are:

$$\mathbb{P}^{(ML)}_{i_1} = \widehat{\mathbb{P}}_{n;i_1} \qquad \mathbb{P}^{(ML)}_{i_r|j_r} = \frac{\widehat{\mathbb{P}}_{n;i_r,j_r}}{\widehat{\mathbb{P}}_{n;j_r}} \qquad r = 2, \ldots, d.$$

from which

$$\begin{aligned}
\mathcal{L}(\sigma, \mathcal{P}^{(ML)}) &= \sum_{x \in \mathcal{X}_{i_1}} \widehat{\mathbb{P}}_{i_1}(x) \ln \widehat{\mathbb{P}}_{n;i_1}(x) + \sum_{r=2}^{d} \sum_{(x,y) \in \mathcal{X}_{i_r} \times \mathcal{X}_{j_r}} \widehat{\mathbb{P}}_{n;i_r,j_r}(x,y) \ln \frac{\widehat{\mathbb{P}}_{n;i_r,j_r}(x,y)}{\widehat{\mathbb{P}}_{n;j_r}(y)} \\
&= \sum_{r=1}^{d} \sum_{x \in \mathcal{X}_{i_r}} \widehat{\mathbb{P}}_{n;i_r}(x) \ln \widehat{\mathbb{P}}_{n;i_r}(x) + \sum_{r=2}^{d} \sum_{(x,y) \in \mathcal{X}_{i_r} \times \mathcal{X}_{j_r}} \widehat{\mathbb{P}}_{n;i_r,j_r}(x,y) \ln \frac{\widehat{\mathbb{P}}_{n;i_r,j_r}(x,y)}{\widehat{\mathbb{P}}_{n;i_r}(x)\widehat{\mathbb{P}}_{n;j_r}(y)} \\
&= \sum_{r=1}^{d} \sum_{x \in \mathcal{X}_{i_r}} \widehat{\mathbb{P}}_{n;i_r}(x) \ln \widehat{\mathbb{P}}_{n;i_r}(x) + \sum_{r=2}^{d} \widehat{I}(i_r, j_r)
\end{aligned}$$

where

$$\widehat{I}(i_r, j_r) = \sum_{(x,y) \in \mathcal{X}_{i_r} \times \mathcal{X}_{j_r}} \widehat{\mathbb{P}}_{n;i_r,j_r}(x,y) \ln \frac{\widehat{\mathbb{P}}_{n;i_r,j_r}(x,y)}{\widehat{\mathbb{P}}_{n;i_r}(x)\widehat{\mathbb{P}}_{n;j_r}(y)}$$

is the plug in estimate of the mutual information. Clearly, the first term in the expression for $\mathcal{L}(\sigma, \mathcal{P}^{(ML)})$ does not depend on σ and hence the maximum likelihood estimate $\sigma^{(ML)}$ is given by

$$\sigma^{(ML)} = \text{argmax}_\sigma \left\{ \sum_{r=2}^{d} \widehat{I}(i_r, j_r) \right\}.$$

The number of spanning trees on d nodes is d^{d-2}. This is Cayley's formula. An exhaustive search is not feasable in practise. Besides, as pointed out by Chow - Liu [17], a *greedy approach* finds the maximal spanning tree.

There are several well known standard algorithms for finding the spanning tree of maximum weight, for example *Kruskal's algorithm* and *Prim's algorithm*. These algorithms are almost identical and find the maximum weight spanning tree in $O(d^2 \ln d)$ time.

Kruskal's algorithm Kruskal's Algorithm runs as follows:

1. The d variables yield $d(d-1)/2$ edges. The edges are indexed in decreasing order, according to their weights $b_1, b_2, b_3, \ldots, b_{d(d-1)/2}$.

2. The edges b_1 and b_2 are selected. Then the edge b_3 is added, *if it does not form a cycle.*

3. This is repeated, through $b_4, \ldots b_{d(d-1)/2}$, in that order, adding edges if they do not form a cycle and discarding them if they form a cycle.

This procedure returns a unique tree if the weights are different. If two weights are equal, one may impose an arbitrary ordering. From the $d(d-1)/2$ edges, exactly $d-1$ will be chosen.

Lemma 5.11. *Kruskal's algorithm returns the tree with the maximum weight.*

Proof The result may be proved by induction. It is clearly true for 2 nodes. Assume that it is true for d nodes and consider a collection of $d+1$ nodes, labelled $(X_1, X_2, \ldots, X_{d+1})$, where they are ordered so that for each $j = 1, \ldots, d+1$, the maximal tree from $(X_1, \ldots, X_j)$ gives the maximal tree from any selection of j nodes from the full set of $d+1$ nodes. Let $b_{(i,j)}$ denote the weight of edge (i,j) for $1 \leq i < j \leq d+1$. Edges will be considered to be undirected. Let $\mathcal{T}_j^{(d+1)}$ denote the maximal tree obtained by selecting j nodes from the $d+1$ and consider $\mathcal{T}_{d+1}^{(d+1)}$.

Let Z denote the leaf node in $\mathcal{T}_{d+1}^{(d+1)}$ such that among all leaf nodes in $\mathcal{T}_{d+1}^{(d+1)}$ the edge (Z,Y) in $\mathcal{T}_{d+1}^{(d+1)}$ has the smallest weight. Removing the node Z gives the maximal tree on d nodes from the set of $d+1$ nodes. This is seen as follows. Clearly, there is no tree with larger weight that can be formed with these d nodes, otherwise the tree on d nodes with larger weight, with the addition of the leaf (Z,Y) would be a tree on $d+1$ nodes with greater weight than $\mathcal{T}_{d+1}^{d+1}$. It follows that $Z = X_{d+1}$ and hence that X_{d+1} is a leaf node of $\mathcal{T}_{d+1}^{(d+1)}$.

By the inductive hypothesis, $\mathcal{T}_d^{(d+1)}$ may be obtained by applying Kruskal's algorithm to the weights $(b_{(i,j)})_{1 \leq i < j \leq d}$. Now consider an application of Kruskal's algorithm to the weights $(b_{(i,j)})_{1 \leq i < j \leq d+1}$ and note that for any (i,j) with $i < j$ such that the undirected edge (X_i, X_j) forms part of the tree $\mathcal{T}_d^{(d+1)}$, $b_{(i,d+1)} < b_{(i,j)}$ and $b_{(j,d+1)} < b_{(i,j)}$. Therefore, if the edges $(b_{(i,j)})_{1 \leq i < j \leq d+1}$ are listed according to their weight and the Kruskal algorithm applied, then all the edges used in $\mathcal{T}_d^{(d+1)}$ will appear further up the list than any edge $(b_{(k,d+1)})_{k=1}^d$ and therefore all the edges of $\mathcal{T}_d^{(d+1)}$ will be included by the algorithm before the edges $(b_{(k,d+1)})_{k=1}^d$ are considered. It follows that $\mathcal{T}_{d+1}^{(d+1)}$ is the graph obtained by applying Kruskal's algorithm to the nodes $(X_1, \ldots, X_{d+1})$. □

Corollary 5.12 (Prim's Algorithm). *The tree of maximal weight may be chosen by choosing any initial node C_i, adding a link $C_i - C_j$ where j is chosen such that $b_{ij} = \max_k b_{ik}$ and at each stage, adding the node C_a to the tree that maximises b_{ak} over nodes C_k already in the tree.*

Proof It is clear that, with the same ordering of the weight, Prim's algorithm returns the same tree as Kruskal's algorithm. □

5.4.4 The Chow-Liu Algorithm and Polytrees

A probability distribution $\mathbb{P}_{X_1,\ldots,X_d}$ factorises according to a *polytree* if there is an ordering of the variables σ such that

$$\mathbb{P}_{X_1,\ldots,X_d} = \prod_{j=1}^{d} \mathbb{P}_{X_{\sigma(j)}|\Pi_j^{(\sigma)}},$$

$$\Pi_j^{(\sigma)} \subseteq \{X_{\sigma(1)}, \ldots X_{\sigma(j-1)}\}$$

and where the directed graph, formed by placing directed edges from each variable in $\Pi_j^{(\sigma)}$ to $X_{\sigma(j)}$ for $j = 1, \ldots, d$ is a *tree*.

A distribution that factorises along a polytree satisfies the condition: if $\Pi_j^{(\sigma)} = \{Y_1^{(\sigma,j)}, \ldots, Y_m^{(\sigma,j)}\}$, then

$$\mathbb{P}_{\Pi_j^{(\sigma)}} = \prod_{k=1}^{m} \mathbb{P}_{Y_k^{(\sigma,j)}}.$$

To extend the Chow-Liu algorithm to polytrees, the *conditional* mutual information is required;

$$I(A,C|B) = \sum_{\underline{x}_{A\cup B\cup C}} \mathbb{P}_{A\cup B\cup C}(\underline{x}_{A\cup B\cup C}) \ln \frac{\mathbb{P}_{A\cup C|B}(\underline{x}_A, \underline{x}_C|\underline{x}_B)}{\mathbb{P}_{A|B}(\underline{x}_A|\underline{x}_B)\mathbb{P}_{C|B}(\underline{x}_C|\underline{x}_B)}.$$

The following lemma is required.

Lemma 5.13. *If $A \perp_{\mathbb{P}} B|C$, then*

$$\min(I(A,C), I(B,C)) \geq I(A,B).$$

Proof This follows from observing that if $A \perp_{\mathbb{P}} B|C$, then

$$I(A,B) + I(A,C|B) = I(A,C), \qquad I(A,B) + I(C,B|A) = I(B,C),$$

which follows from:

$$A \perp_{\mathbb{P}} B|C \Leftrightarrow \mathbb{P}_{A\cup C|B} = \frac{\mathbb{P}_{A\cup B\cup C}}{\mathbb{P}_B} = \frac{\mathbb{P}_{A\cup C}\mathbb{P}_{B\cup C}}{\mathbb{P}_B\mathbb{P}_C}$$

and hence

$$I(A,C|B) = \sum \mathbb{P}_{A\cup B\cup C} \ln \frac{\mathbb{P}_{A\cup C}\mathbb{P}_B}{\mathbb{P}_{A\cup B}\mathbb{P}_C} = I(A,C) - I(A,B).$$

The other is similar. □

Theorem 5.14. *Suppose that $\mathbb{P}$ factorises according to a polytree. Kruskal's algorithm will locate the skeleton of the polytree.*

Proof Let $A = \{i\}$, $B = \{j\}$ and $D = \{k\}$ be three distinct nodes. Assume that $i \perp_{\mathbb{P}} j|k$. This can happen in the following cases:

$$i \to k \to j, \qquad i \leftarrow k \leftarrow j, \qquad i \leftarrow k \to j,$$

where $m \to n$ indicates a directed path. In all cases, $I(i,k|j) > 0$ and $I(k,j|i) > 0$. It follows that

$$\min(I(i,k), I(j,k)) > I(i,j).$$

Kruskal's algorithm takes the edge of largest weight that does not form a cycle. The algorithm will therefore not choose the edge (i,j) if there is a node k between i and j in σ.

For $i \to k \leftarrow j$, $i \perp_{\mathbb{P}} j$ and hence the edge $i - j$ will not be chosen by Kruskal's algorithm. □

It is straightforward to find appropriate directions for the edges; if there are edges $i - j - k$, then the edges take directions $i \to j \leftarrow k$ if and only if $I(i,k) = 0$.

5.5 Asymptotic Consistency of the Maximum Likelihood Estimate

Let $\mathbb{W}(\mathcal{T}_d^{(ML)}(n)) = \sum_{r=2}^{d} \widehat{I}(i_r, j_r)$ denote the weight of the tree computed by the Chow-Liu algorithm. Suppose that there is a distribution $\mathbb{P}^0$ which is the true, but unknown distribution. Let

$$\mathbb{P}^{(\sigma)0} = \text{argmin}_{\mathbb{P}\in\mathcal{P}(\mathcal{X},\mathbb{T}_d)} D_{KL}(\mathbb{P}^0\|\mathbb{P}).$$

Then $\mathbb{P}^{(\sigma)0}$ is the reverse I-projection of $\mathbb{P}^0$ and the corresponding structure $\sigma^{(0)} = (i_r^0, j_r^0)_{j=2}^d$ is the Chow-Liu dependence structure. If $\mathbb{P} \in \mathcal{P}(\mathcal{X}, \mathbb{T}_d)$, then $\mathbb{P}^{(\sigma)0} = \mathbb{P}^0$.

Let $\mathcal{T}_d^0$ denote the tree structure corresponding to $\sigma^{(0)}$, then

$$\mathbb{W}(\mathcal{T}_d^0) = \sum_{r=2}^{d} I^0(i_r^0, j_r^0)$$

where $I^0(i_r^0, j_r^0)$ are the mutual informations computed with $\mathbb{P}^{0\downarrow(i_r^0, j_r^0)}$, which is the tree of maximal weight.

Let

$$\widehat{\mathbb{P}}_{ML;n}^{(\sigma_{ML})} = \text{argmin}_{\mathbb{P}\in\mathcal{P}(\mathcal{X},\mathbb{T}_d)} D_{KL}(\mathbb{P}_n\|\mathbb{P})$$

where $\mathbb{P}_n$ denotes the empirical distribution and σ_{ML} denotes the maximum likelihood Chow-Liu dependence structure. Let $\mathbb{W}(\mathcal{T}; n)$ denote the weight of tree $\mathcal{T}$ based on probability distribution $\mathbb{P}_n$ and, in particular, let $\mathbb{W}(\mathcal{T}_d^{(ML)}(n); n)$ denote the Chow Liu dependence tree weight based on σ_{ML} and $\widehat{\mathbb{P}}_n$. Then the following result holds:

Theorem 5.15.

$$\mathbb{W}(\mathcal{T}_d^{(ML)}(n); n) \overset{n\to+\infty}{\longrightarrow} \mathbb{W}(\mathcal{T}_d^0) \qquad \mathbb{P}^0 - a.s.$$

Proof This is a consequence of the strong law of large numbers; firstly, since $\mathcal{X}$ is finite, the strong law of large numbers gives that

$$\max_{x\in\mathcal{X}} |\mathbb{P}_n(x) - \mathbb{P}^0(x)| \overset{n\to+\infty}{\longrightarrow} 0 \qquad \mathbb{P}^0 - a.s.$$

from which a.s. convergence of all empirical marginal distributions follows and in particular

$$\widehat{I}_n(i,j) \overset{n\to+\infty}{\longrightarrow} I^0(i,j)$$

for all pairs (i, j). It follows that for each tree $\mathcal{T}_d$,

$$\mathbb{W}(\mathcal{T}_d; n) \overset{n\to+\infty}{\longrightarrow} \mathbb{W}(\mathcal{T}_d) \qquad \mathbb{P}^0 - a.s.$$

and hence, since $\mathbb{T}_d$ is a finite set,

$$\max_{\mathcal{T}_d\in\mathbb{T}_d} |\mathbb{W}(\mathcal{T}_d; n) - \mathbb{W}(\mathcal{T}_d)| \overset{n\to+\infty}{\longrightarrow} 0.$$

Note that, by construction, $\mathbb{W}(\mathcal{T}_d; n) \le \mathbb{W}(\mathcal{T}_d^{(ML)}(n); n)$ for all $\mathcal{T}_d$ and each n. Now let

$$\mathbb{T}_d^0 = \{\mathcal{T}_d \in \mathbb{T}_d | \mathbb{W}(\mathcal{T}_d) = \mathbb{W}(\mathcal{T}_d^0)\}.$$

Since $\mathbb{T}_d$ is finite, there is a positive constant δ such that

$$\delta = \min_{\mathcal{T}_d \in \mathbb{T}_d \setminus \mathbb{T}_d^0} |\mathbb{W}(\mathcal{T}_d^0) - \mathbb{W}(\mathcal{T}_d)| > 0.$$

Choose n large enough such that $\mathbb{P}^0$ a.s.

$$\max_{\mathcal{T}_d \in \mathbb{T}_d} |\mathbb{W}(\mathcal{T}_d; n) - \mathbb{W}(\mathcal{T}_d)| \leq \frac{\delta}{2}.$$

There is an n_δ such that this holds for all $n \geq n_\delta$ and such that there is a tree in $\mathbb{T}_d^0$, say $\mathcal{T}_d^0$ such that $\mathbb{W}(\mathcal{T}_d^{(ML)}(n)) = \mathbb{W}(\mathcal{T}_d^0)$ and such that

$$|\mathbb{W}(\mathcal{T}_d^0; n) - \mathbb{W}(\mathcal{T}_d^0)| \leq \frac{\delta}{2}.$$

In other words, for any $\epsilon > 0$ with $\frac{\delta}{2} > \epsilon$, it holds that for $n > n_\epsilon$,

$$|\mathbb{W}(\mathcal{T}_d^{(ML)}(n); n) - \mathbb{W}(\mathcal{T}_d^0)| < \epsilon \qquad \mathbb{P}^0 - a.s.$$

□

This result does not assert convergence of the sequence of trees $\mathcal{T}_d^{(ML)}(n)$ unless the set $\mathbb{T}_d^0$ contains exactly one element.

Chapter 6

Structure Learning Algorithms

A structure may be determined by causal reasoning, or formally constructed by engineering considerations, but this situation is not considered here; the problem considered is the task of locating a structure purely from data, learning a structure that encodes the key features of the dependence structure between the d variables of the data set, when presented with n complete or incomplete instantiations. This is the task of *structure learning*.

Based only on data, structure learning techniques will locate a suitable equivalence class, either by finding a DAG within the equivalence class or by finding the essential graph. Structure learning methods tend to fall generally into three categories: search and score techniques, where a score function is used and the algorithm attempts to find the structure that maximises the score function, *constraint based* methods, where conditional independence tests are carried out and independence relations thus established provide constraints that the d-separation statements of the output graph should satisfy and *hybrid* algorithms that use both constraint based and search and score techniques.

6.1 Search and score

Search and score algorithms operate by selecting various structures for examination and scoring them. The structure with the highest score from among those considered is selected. Several score functions have been considered in the literature. They all have the feature of giving higher scores to those where the best fitting distribution, given the graph structure, is closest to the empirical distribution, with a penalty for the number of parameters. The *likelihood function* for a graph structure D, given an $n \times d$ data matrix $\mathbf{x}$, where the rows, $(x_{i1}, \ldots, x_{id})$ for $i = 1, \ldots, n$ represent n instantiations of the random vector $\underline{X} = (X_1, \ldots, X_d)$, is given by the formula

$$L(D; \mathbf{x}) = \prod_{j=1}^{d} \prod_{l=1}^{q_j} \frac{\Gamma(\sum_{i=1}^{k_j} \alpha_{jil})}{\Gamma\left(n(\pi_j^l) + \sum_{i=1}^{k_j} \alpha_{jil}\right)} \prod_{i=1}^{k_j} \frac{\Gamma(n(x_j^i, \pi_j^l) + \alpha_{jil})}{\Gamma(\alpha_{jil})}, \tag{6.1}$$

where α_{jil} represents *virtual* information, the weight given to the (x_j^i, π_j^l) configuration before the data is received. Formula (6.1) is known as the *Cooper Herzkovitz likelihood* and was first derived in [19].

Using this as the basis of a score function has some practical difficulties; a large number of hyper-parameters $\underline{\alpha}$ have to be specified. The *log likelihood* is more practical

$$LL(D;\mathbf{x}) = \sum_{j=1}^{d}\sum_{i=1}^{k_j}\sum_{l=1}^{q_j} n(\pi_j^{(l)}, x_j^{(i)}) \ln \frac{n(\pi_j^{(l)}, x_j^{(i)})}{n(\pi_j^{(l)})}. \tag{6.2}$$

This cannot be used directly as a score function; it will favour graphs with many edges. A penalisation is usually introduced to favour graphs with fewer parameters. The most common score function considered is the *Bayesian Information Criterion*:

$$\mathrm{BIC}(D,\mathbf{x}) = LL(D;\mathbf{x}) - \frac{1}{2}(\ln n)|\underline{\theta}| \tag{6.3}$$

where D represents the DAG along which the factorisation is made and $|\underline{\theta}| = \sum_{j=1}^{d} q_j(k_j - 1)$ is the number of independent parameters required (recall that $\sum_{i=1}^{k_j}\theta_{jil} = 1$ for each (j,l)). The BIC was developed by Gideon E. Schwartz (1978) [57], who described Bayesian principles for using it. The negative of this score function is known as the *minimum description length* (MDL); $MDL = -BIC$. The MDL was introduced by Jorma Rissanen (1978)[52] independently and simultaneously.

Other score functions may be used, such as the *Akaike Information Criterion* (AIC), which is similar to the BIC, except that for the penalisation, $\frac{1}{2}\ln n$ is replaced by n. The AIC was introduced by Hirotsugu Akaike in 1974 [1].

The simplicity of the function in equation (6.2) as the basis of a score function is appealing, but the hyper-parameters of the Cooper Herzkovitz likelihood give the possibility to input prior information. The parameters $\underline{\alpha}$ represent the prior assessment of the probability values for a given network, but prior information can also be placed over the graph structures. If a prior distribution $p_{\mathcal{D}}$ is placed over the space of DAGs with d nodes, denoted $\mathcal{D}$, then the posterior distribution, given data $\mathbf{x}$, may be written as

$$p_{\mathcal{D}|\mathbf{X}}(D|\mathbf{x}) = \frac{L(D;\mathbf{x})p_{\mathcal{D}}(D)}{p_{\mathbf{X}}(\mathbf{x})} \propto L(D;\mathbf{x})p_{\mathcal{D}}(D),$$

where $L(D;\mathbf{x})$ is the Cooper Herskovitz likelihood of equation (6.1). The subject of constructing appropriate prior distributions is discussed for example by Castelo and Siebes (2000) [7] and by Flores, Nicholson, Brunskill, Korb and Mascaro (2011) [23]. This may be used to construct a score function based on equation (6.3), where the log posterior is used in place of $LL(D;\mathbf{x})$.

The approach of testing each possible structure is not computationally feasible. This is because the number of possible DAGs grows super exponentially in the number of nodes. Robinson [54] (1977) gave the following recursive function for computing the number $N(d)$ of DAGs with d nodes:

$$N(d) = \sum_{i=1}^{d}(-1)^{i+1}\binom{d}{i}2^{i(d-1)}N(d-i). \tag{6.4}$$

For $d = 5$ it is 29000 and for $d = 10$ it is approximately 4.2×10^{18}. Here $N(d)$ is a very large number, even for small values of d. It is not feasible to test all possible graphs, even for modest values of d.

6.1.1 Monte Carlo Methods

Search and score methods usually deal with this by running a Markov process through the search space. One example is the *Markov chain Monte Carlo model composition* algorithm introduced by Madigan and York [36] in 1995 and developed by Madigan, Andersson, Perlman and Volinsky [35] in 1997. These algorithms choose, as search space, the space of essential graphs.

An efficient Markov chain Monte Carlo algorithm for searching the model space was developed by Corander, Ekdahl and Koski (2008) [20], for families of models for which the marginal likelihood can be calculated analytically, either exactly or approximately, given any fixed structure. The model space is explored by a finite number of interacting parallel stochastic processes.

The use of 'Markov chain Monte Carlo' with reference to structure learning is misleading. The term McMC has a well defined standard mathematical meaning, which is to build up an empirical distribution that approximates the stationary distribution of the Markov chain. In structure learning for Bayesian networks, the stationary distribution of the Markov chain is irrelevant. The aim of producing a well constructed process is to visit the most promising candidate graphs within the search space. The search space is so large that very few graphs will be visited more than once and the overwhelming majority will not be visited at all. There is insufficient information to build up an empirical distribution over graph space. The decision is made purely by computing the score functions and choosing the graph visited which gives the largest score. The only consideration in constructing a process is to ensure that the process will visit structures that give a good representation of the data; properties of the search process, such as the Markov property, reversibility, convergence of the empirical distribution to a certain stationary distribution are all irrelevant.

6.1.2 Sparse Candidate Algorithm

The sparse candidate algorithm as developed by Friedman, Nachman and Pe'er (1999) [25]. The main idea of the technique is to identify a relatively small number of *candidate* parents for each variable. This is based on simple local statistics, such as correlation. Attention is then restricted to networks in which the parent set is a subset of the candidate parent set.

The algorithm proceeds as follows: let D_n denote the DAG chosen at iteration n, let $\Pi_i^{(n)}$ denote the parent set for variable X_i in D_n.

- For $i = 1, \ldots, d$, choose the candidate set $C_i^{(n)} = \{Y_1, \ldots, Y_k\}$ of candidate variables for Π_i, the parent set for variable X_i. The set $C_i^{(n)}$ is chosen as $\Pi_i^{(n-1)}$ together with children and parents of children of X_i in D_n, and all those variables $Y \notin MB(X_i)$ such that the score

$$\sum_{(x,y,z)\in\mathcal{X}_{X_i}\times\mathcal{X}_Y\times\mathcal{X}_{\Pi_i^{(n-1)}}} n_{X_i,Y,\Pi_i^{(n-1)}}(x,y,z)\ln\frac{n_{X_i,Y,\Pi_i^{(n-1)}}(x,y,z)n_{\Pi_i^{(n-1)}}(z)}{n_{Y,\Pi_i^{(n-1)}}(y,z)n_{X_i,\Pi_i^{(n-1)}}(x,z)}$$

  is sufficiently high. Here MB denotes *Markov blanket* (definition 2.8). Also, for a set W, $n_W(w)$ denotes the number of appearances of configuration w in the data matrix $\mathbf{x}$. If the test statistic is low, it supports $X_i \perp Y|\Pi_i^{(n-1)}$ and hence Y is not a candidate parent.

There are other ways of determining the candidate parents; anything in the current Markov blanket not d-separated from the variable by the Markov blanket should be included as a candidate parent.

- Find a high scoring network D_n where $\Pi_i^{D_n} \subset C_i^{(n)}$ for $i = 1, \ldots, d$.

This is the method successfully used for analysis genetic expression data by Friedman et. al. (2000) [26].

6.1.3 Optimal Reinsertion

The *optimal reinsertion* algorithm, introduced by Moore and Wong (2003) [39], is a search-and-score algorithm that works along the following lines: at each step a *target node* is chosen, all edges entering or leaving the target are deleted, and the optimal combination of in-edges and out-edges is found, the node is re-inserted with these edges. This involves searching through the legal candidate parent sets and, for each candidate parent set, the legal child sets. The optimal reinsertion may be combined with sparse candidate.

6.1.4 Greedy Search and Greedy Equivalence Search

The *Greedy Search*, introduced by Chickering (2002) [15], works along the following lines to produce a DAG, along which the probability distribution factorises, starting from the graph with no edges:

- *Forward phase* Let E_0 denote the graph with no edges. Let E_n denote the essential graph from stage n of the forward phase. Consider all possible DAGs within the Markov equivalence class, all possible DAGs obtained by adding exactly one edge to a DAG from this equivalence class and consider the set of essential graphs corresponding to this collection of DAGs. Let E_{n+1} denote the essential graph with the highest score if it has a higher score than E_n and continue to forward phase stage $n+1$. Otherwise, terminate the forward phase, with output E_n.

- *Backward phase* Let $\tilde{E}_0$ denote the output graph from the forward phase. Let $\tilde{E}_n$ denote the output graph from stage n of the backward phase. Consider all possible DAGs corresponding to the equivalence class $\tilde{E}_n$, all possible DAGs formed by an edge deletion from these DAGs and consider the set of essential graphs corresponding to this collection of DAGs. Let $\tilde{E}_{n+1}$ denote the essential graph with the highest score if it is higher than that for $\tilde{E}_n$ and continue to backward phase stage $n+1$. Otherwise terminate; $\tilde{E}_n$ is the output of the backward phase and of the greedy equivalence search algorithm.

After the forward and backward phase, this algorithm is guaranteed to return an optimal structure provided there exists a faithful DAG (definition 3.2). The faithfulness assumption may be relaxed; the algorithm returns a suitable structure provided the weaker *composition* condition hold; $X \perp Y|Z$ and $X \perp W|Z$ implies $X \perp Y \cup W|Z$. The compositional axiom is essential for the algorithm to return the correct graph.

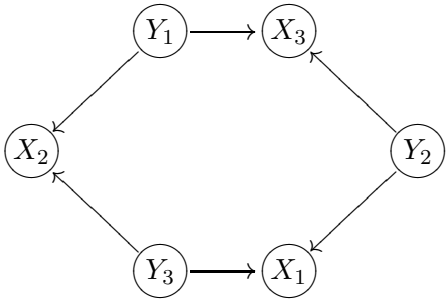

Figure 6.1: DAG for the natural factorisation; it is not faithful

6.2 Constraint Based Structure Learning

Constraint based structure learning methods carry out tests, for triples (X, Y, S) where X and Y are variables and S is a subset of variables, to decide whether or not $X \perp Y|S$. The results of these tests are the *constraints* and a graph is then chosen that satisfies as many of the constraints, thus established, as possible.

For larger numbers of variables, these are overwhelmingly faster than search and score algorithms; for each structure tested, a search and score algorithm has to compute a score function that involves the entire data set. Testing whether $X \perp Y|S$ only requires the variables X, Y and those in S. In most applications, locating the structure only requires examining conditioning sets S that have a maximum of three or four variables.

The main problem is that these algorithms usually *require* that there exists a graph faithful to the distribution and use the following result:

Theorem 6.1. *Let $\underline{X} = (X_1, \ldots, X_d)$ be a random vector, with probability distribution $\mathbb{P}$. A faithful DAG D contains an edge between two distinct variables X_i and X_j if and only if $X_i \not\perp X_j|S$ for any $S \subseteq V \backslash \{X_i, X_j\}$, where $V = \{X_1, \ldots, X_d\}$ is the variable set.*

Many constraint based algorithms assume that there exists a faithful DAG and remove an edge $X \sim Y$ as soon as they encounter a set S such that $X \perp Y|S$. In example 3.7, a structure learning algorithm based on this principle will perform disastrously badly and return the empty graph, falsely indicating that $Y_1, Y_2, Y_3, X_1, X_2, X_3$ are mutually independent.

Note that (X_1, X_2, X_3) do not satisfy *composition*; $X_1 \perp X_2$ and $X_1 \perp X_3$, but $X_1 \not\perp \{X_2, X_3\}$. The search-and-score *greedy equivalence search* algorithm will also fail with this example.

Some examples of constraint based methods, described in more detail below, are *Chow Liu tree*, *three phase dependency analysis*, the PC and MMPC algorithms, Fast algorithm and RAI algorithm. They all operate according to the principle of removing an edge $X \sim Y$ whenever a set S is located such that $X \perp Y|S$.

6.2.1 Three phase dependency analysis

The *three phase dependency analysis* algorithm (denoted TPDA) was introduced by Cheng, Greiner, Kelly, Bell and Liu (2002) [14], who write, 'this TPDA algorithm is correct (i.e., will produce the perfect model of the distribution) given a sufficient quantity of training data whenever the underlying model is monotone DAG faithful.' The algorithm requires the faithfulness assumption to hold and relies on theorem 6.1. The TPDA algorithm works in three phases; *draughting*, *thickening* and *thinning*, outlined in algorithm 1.

Algorithm 1 The Three Phase Dependency Analysis Algorithm

Stage 1: Draughting Locate the Chow - Liu tree
This stage is simply Kruskal's algorithm
Stage 2: Thickening Add edges
for $i = 1, \ldots, d-1$, $j = i+1, \ldots, d$ **do**
Let $C_{i,j}$ be the set of neighbours of X_i or X_j on a path between X_i and X_j
if $X_i \not\perp X_j | C$ for any subset $C \subseteq C_{ij}$ **then**
add an edge $X_i \sim X_j$
else
do not add an edge $X_i \sim X_j$
and let $S_{i,j}$ denote the set such that $X_i \perp X_j | S_{ij}$
end if
end for
Stage 3: Thinning Removing unnecessary edges
for $i = 1, \ldots, d-1$, $j = i+1, \ldots, d$ **do**
Let C_{ij} denote common neighbours of X_i and X_j
if $X_i \sim X_j$ and there is a set $C \subseteq C_{ij}$ such that $X_i \perp X_j | C$ **then**
remove the edge between X_i and X_j.
end if
end for
Stage 4: Directing edges For each vee structure $X_i \sim X_k \sim X_j$, (X_i, X_k, X_j) is an immorality if $X_k \notin S_{ij}$, otherwise it is not. Once the immoralities have been added, the additional compelled edges are obtained using Meek's rules.

This outlines the main steps of the algorithm; a precise description of the algorithm and proof that it returns a faithful DAG when it exists, is given in [14].

6.2.2 PC and MMPC algorithms

The PC algorithm was introduced in [60] by Spirtes Glymour and Scheines (1993) and was modified to produce the MMPC algorithm in [63] (2006). It is algorithm for locating the skeleton of a *faithful* DAG, which may be used to construct the essential graph if the separating sets, sets $S_{X,Y}$ such that $X \perp Y | S_{XY}$ are recorded. It works in three stages. Firstly, a forward stage starts with an empty graph

and proceeds according to algorithm 2

Algorithm 2 The MMPC Algorithm, Stage 1

for $j = 1, \ldots, d$ **do**
  Set $\mathcal{Z}_{j,0} = \phi$ (the empty set)
  for $i = 1, \ldots, d$ **do**
    if $i = j$ **then**
      Set $\mathcal{Z}_{j,i} = \mathcal{Z}_{j,i-1}$
    else
      if $X_i \perp X_j | \mathcal{Z}_{j,i-1}$ **then**
        Set $\mathcal{Z}_{j,i} = \mathcal{Z}_{j,i-1}$ and $S_{i,j} = \mathcal{Z}_{j,i-1}$ ($S_{i,j}$ the sepset)
      else
        Set $\mathcal{Z}_{j,i} = \mathcal{Z}_{j,i-1} \cup \{X_i\}$
      end if
    end if
  end for
  Set $\mathcal{Z}_j = \mathcal{Z}_{j,d}$
end for

Note that edge removal is on the principle of theorem 6.1. Assuming faithfulness and a perfect oracle (that is, tests always give the correct results), there are possibly too many edges after this stage. Secondly, a backward stage removes some of the edges; algorithm 3.

Algorithm 3 The MMPC Algorithm, Stage 2

for $j = 1, \ldots, d$ **do**
  Set $\mathcal{Y}_{j,0} = \mathcal{Z}_j$.
  for $k = 1, \ldots, d$, **do**
    if $X_k \in \mathcal{Y}_{j,k-1}$ and $\exists S \subseteq \mathcal{Y}_{j,k-1} \backslash \{X_k\}$ such that $X_j \perp X_k | S$ **then**
      Set $S_{j,k} = S$ and set $\mathcal{Y}_{j,k} = \mathcal{Z}_{j,k-1} \backslash \{X_k\}$.
    else
      Set $\mathcal{Y}_{j,k} = \mathcal{Y}_{j,k-1}$.
    end if
  end for
  Set $\mathcal{Z}_j = \mathcal{Y}_{j,d}$
end for

The algorithm may return false positives. Suppose a probability distribution may be represented by the DAG in figure 6.2. Working from T, the node C may enter the output, and remain in the output.
This is because C is dependent on T, conditioned on all subsets of parents and children of T; namely, ϕ (the empty set) and $\{A\}$. Note that the *collider* connection TAB, is opened when A is instantiated so that, when A is instantiated and B is uninstantiated, T is d-connected with C. For ϕ (the empty

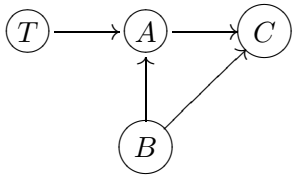

Figure 6.2: Min Max Parent Child: A False Positive

set), TAC is a chain connection, where A is uninstantiated, so that T is d-connected to C.

T and C are d-separated if and only if A and B are simultaneously instantiated; that is, $T \perp C|\{A, B\}$. But if B is independent from T given the empty set, so it will be removed from $\mathcal{Z}$. Therefore, the link TC will not be removed.

A third stage is implemented to remove the false positives; algorithm 4.

Algorithm 4 The MMPC Algorithm, Stage 3

for $j = 1, \ldots, d$ **do**
  Set $\mathcal{Y}_{j,0} = \mathcal{Z}_j$.
  for $k = 1, \ldots, d$ **do**
    if $X_k \in \mathcal{Y}_{j,k-1}$ but $X_j \notin \mathcal{Z}_k$ **then**
      Set $\mathcal{Y}_{j,k} = \mathcal{Y}_{j,k-1} \backslash \{X_k\}$
    else
      Set $\mathcal{Y}_{j,k} = \mathcal{Y}_{j,k-1}$
    end if
  end for
  Set $\mathcal{Z}_j = \mathcal{Y}_{j,d}$
end for

This returns the complete parent / child set for X_i. The *sep-sets* for the variables removed in the third stage have already been established. The sep-set for a pair of variables X, Y is the set $S_{X,Y}$ such that $X \perp Y|S_{X,Y}$, which caused the algorithm to remove the edge $X \sim Y$.

Establishing the Essential Graph Having recorded the sep-sets, it is now straightforward to construct the essential graph. For each vee structure (X, Z, Y) (that is a structure such that $\{X, Y\} \subset \mathcal{Z}^{(Z)}$, but $X \notin \mathcal{Z}^{(Y)}$), check whether or not $Z \in S_{XY}$. If $Z \in S_{XY}$, then (X, Z, Y) is not an immorality; the edges $X - Z - Y$ remain undirected at this stage. If $Z \notin S_{XY}$, then (X, Z, Y) is an immorality.

Finally the additional directed edges in the essential graph are known as *compelled edges* and may be established using a straightforward set of rules known as *Meek's rules* from Meek [38]. These are that once the immoralities have been established, if the edge $\alpha - \beta$ appears in any of the structures in figure 6.3, it is directed as $\alpha \mapsto \beta$. Immoralities are determined by the results of the conditional independence tests. In the first of these structures, if the edge $\gamma \to \alpha$ is present and (γ, α, β) is not

an immorality, then the direction $\alpha \to \beta$ is force. In the second, if $\alpha \to \gamma$ and $\gamma \to \beta$ are present, then $\alpha \to \beta$ is forced to prevent a cycle. In the third, if the immorality $(\gamma_1, \beta, \gamma_2)$ is present, but $(\gamma_1, \alpha, \gamma_2)$ is not an immorality, then $\alpha \to \beta$ is forced to prevent cycles.

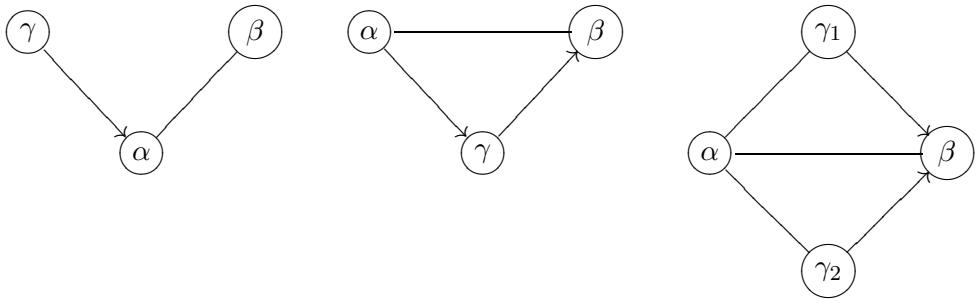

Figure 6.3: Meeks rules: the undirected edge $\alpha - \beta$ is given the direction $\alpha \mapsto \beta$ if it appears in one of these configurations.

6.2.3 Constraint based algorithms and hypothesis testing

One serious problem with constraint based algorithms is that even if the data is generated by a probability distribution with a faithful graphical representation, the power of the χ^2 tests used to establish independence decreases rapidly as the size of the conditioning sets increases, according to the inverse of the size of the conditioning set. With a test $H_0 : X \perp Y|S$ versus $H_1 : X \not\perp Y|S$, independence is *accepted* and hence the edge $X - Y$ *removed* from the skeleton whenever the null hypothesis of independence is not rejected. This, of course, violates an elementary statistical principle that is taught whenever the subject of hypothesis testing is introduced, even at the most basic level; a null hypothesis is never accepted. This can have rather serious consequences; an edge $X - Y$ can be removed simply because the test is unsatisfactory due to a large conditioning set, even if this results in a graph where X and Y are d-separated and it has been established through previous tests, where the null hypothesis has been rejected and the alternative hypothesis accepted, that $X \not\perp Y$. Other algorithms have been developed with this in mind, to keep the size of the conditioning sets as small as possible.

6.2.4 The FAST algorithm

The FAST algorithm takes this into account. It starts with the complete undirected graph. Suppose the variable set is $X_1, \ldots, X_d$, then the algorithm is described in algorithm 5.

The algorithm is run either until all variable pairs have fewer than k common neighbours, or until some pre-determined level of k is reached (usually 3 or 4), beyond which the conditional independence tests are considered unreliable.

Algorithm 5 The FAST algorithm

output the essential graph

initialisation complete undirected graph with d nodes

for $k \geq 0$ **do**

  for $1 \leq i \leq d-1$, $i+1 \leq j \leq d$ **do**

    if there is an edge $X_i \sim X_j$ **then**

      test for $X_i \perp X_j | C$ for all sets C such that $|C| = k$. If there is a set C such that the conditional independence statement holds, remove the edge $X_i \sim X_j$ and set $S_{ij} = C$

    end if

  end for

end for

Termination: the algorithm terminates either when all pairs X_i, X_j have fewer than k common neighbours, or else a pre-assigned value of k is reached.

Edge direction For vee structures $X_i - Z - X_j$, if $Z \notin S_{ij}$, direct the edges $X_i \rightarrow Z \leftarrow X_j$. If $Z \in S_{ij}$, then leave the edges unaltered

Compelled edges After locating the immoralities, apply Meek's rules to find all the directed edges of the essential graph

As with the MMPC algorithm a vee-structure $X - Z - Y$ in the resulting graph is declared an immorality if $Z \notin S_{X,Y}$ (the sep-set for X, Y) and is not an immorality otherwise. Meek's rules are then used to locate the other compelled edges.

6.2.5 Recursive Autonomy Identification

The *Recursive Autonomy Identification algorithm*, introduced by Yehezkel and Lerner in [68], proceeds under the assumption that there is a faithful graphical model and removes an edge $X - Y$ whenever there is a test result $X \perp Y | S$ for some S. Like the Fast algorithm, it starts with the smallest conditioning sets first and works upwards. It employs more of the structure than Fast's algorithm, which reduces the number of tests that have to be performed.

The additional structure used is that of the *essential graph*. At each stage of the algorithm, the immoralities produced by edge deletions in the current round are inserted. The essential graph is a *chain graph*. This means that the variable set V may be split into p disjoint subsets; $V = V_1 \cup \ldots \cup V_p$, where the graph, restricted to V_j, is undirected for each j and, for $i \neq j$, $\alpha \in V_i$ and $\beta \in V_j$, there is no cycle containing both α and β. (With a cycle, the direction of the edges is observed. $(\alpha_0, \ldots, \alpha_m)$ is a cycle if $\alpha_0 = \alpha_m$, $\alpha_i \neq \alpha_j$ for all other pairs (i, j) and for each $i, i+1$ there is either an undirected edge $\alpha_i - \alpha_{i+1}$ or a directed edge $\alpha_i \rightarrow \alpha_{i+1}$). The essential graph also satisfies the condition that the chain components are triangulated.

The first step of the RAI algorithm is given by algorithm 6.

This produces a graph that is an essential graph. After this initialisation (stage 0), the algorithm proceeds recursively. Algorithm 7 gives the nth stage of the RAI algorithm. For a graph $\mathcal{G} = (V, E)$,

Algorithm 6 The RAI algorithm, step 0

Initialization Starting with a variable set $V = \{X_1, \ldots, X_d\}$, the initial graph is the complete graph, with undirected edges between each pair of variables

for $1 \leq i < j \leq d$ **do**

Test whether or not $X_i \perp X_j$. If true, the edge $X_i \sim X_j$ is removed.

Record $S_{i,j} = \phi$, the empty set ($S_{i,j}$ is the *separator* or *sepset*).

for each vee structure $X_i - Z - X_j$ where there is no edge $X_i - X_j$ **do**

direct edges so that the triple (X_i, Z, X_j) is an immorality $X \rightarrow Z \leftarrow Y$.

end for

end for

Apply Meek's rules, to direct the additional compelled edges

the notation $\mathcal{G}_D$ denotes the sub-graph $(D, E \cap D \times D)$.

Algorithm 7 The RAI algorithm, step n for $n \geq 1$

Starting with a chain component that has no descendants and proceeding backwards, consider in turn each chain component $\mathcal{G}_C$ and the sub-graph $\mathcal{G}_D$ formed by taking the chain component $\mathcal{G}_C = (C, U_C)$ together with the chain components that have parent variables of $\mathcal{G}_C$ and all all the directed edges connecting these chain components.

for each $Y \in C$ and each neighbour X of Y (consider first the parents in different connected components, and then the undirected neighbours in the component $\mathcal{G}_C$) **do**

check whether there is a set $S_{XY} \subset D$ of size n such that $X \perp Y | S$.

if true **then**

remove the edge between X and Y and record S_{XY}.

For each new vee structure (X, Z, Y), declare it to be an immorality if and only if $Z \notin S_{XY}$.

Find the additional compelled edges generated by the immorality.

end if

Remove the chain component $\mathcal{G}_C$

end for

proceed recursively until the whole graph has been considered.

The RAI proceeds either until the size of the largest neighbour set in the undirected graph is equal to n (the size of the conditioning set in the current round) or until the algorithm has performed a pre-assigned number of rounds, governed by the reliability of the conditional independence tests. The output is the resulting essential graph.

This additional use of the structure by the RAI algorithm can reduce the number of tests required in comparison with the Fast algorithm.

Example 6.2 (Example for Recursive Autonomy Identification)**.**

Suppose that the DAG in figure 6.4 is faithful to the distribution $\mathbb{P}_{X_1,X_2,X_3,X_4,X_5,X_6,X_7}$.

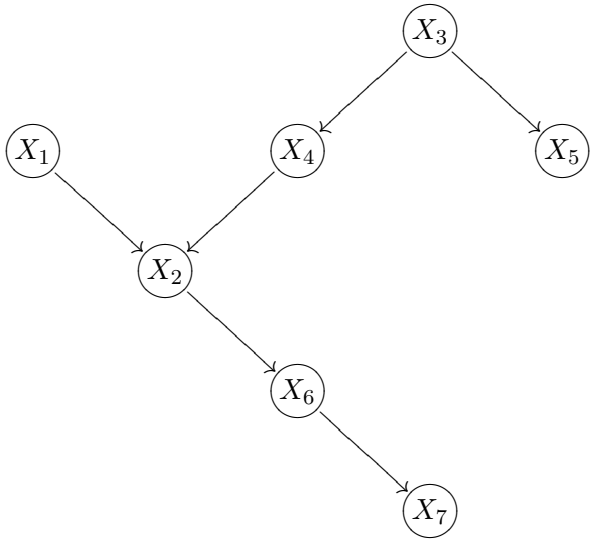

Figure 6.4: Example to illustrate RAI algorithm

Suppose also a 'perfect oracle'; independence tests give the correct results. After the first round, $X_1 \not\perp X_2$, $X_1 \not\perp X_6$, $X_1 \not\perp X_7$, but $X_1 \perp \{X_3, X_4, X_5\}$. $X_2 \not\perp X_j$ for any j, $X_3 \not\perp X_j$ for $j = 4,5,6,7$, $X_4 \not\perp X_j$ for $j = 5,6,7$, $X_5 \not\perp X_j$ for $j = 6,7$ and $X_6 \not\perp X_7$.

After the CI tests with conditioning sets size 0 have been carried out, the immoralities are determined;

$$(X_1, X_2, X_3), (X_1, X_6, X_3), (X_1, X_7, X_3), (X_1, X_2, X_4), (X_1, X_6, X_4)$$

$$(X_1, X_7, X_4), (X_1, X_2, X_5), (X_1, X_6, X_5), (X_1, X_7, X_5).$$

The edges $X_3 - X_4$, $X_3 - X_5$ and $X_4 - X_5$, $X_2 - X_6$, $X_2 - X_7$ $X_6 - X_7$ remain undirected.

Removing the undirected edges, the chain components are $A_1 = \{X_1\}$, $D = \{X_2, X_6, X_7\}$ and $A_2 = \{X_3, X_4, X_5\}$. D stands for *descendant*, A for *ancestor*.

Within D, $X_2 \perp X_7 | X_6$ and this is the only CI statement with a conditioning set size 1. The edge $X_2 - X_7$ is therefore removed and $X_2 - X_6 - X_7$ is not an immorality, since $X_6 \in S_{2,7}$ (the sep set).

Within A_2, $X_4 \perp X_5 | X_3$, hence $X_4 - X_5$ is removed and $X_4 - X_3 - X_5$ is not an immorality since $X_3 \in S_{4,5}$.

Now consider the directed edges from A_1 and A_2 to D. $\{X_3, X_4, X_5\} \perp \{X_6, X_7\} | X_2$, leading to removal of the 6 corresponding directed edges. $\{X_3, X_5\} \perp \{X_2\} | \{X_4\}$. Finally, Meeks rules may be used to direct $X_2 \to X_6$ and $X_6 \to X_7$ giving the essential graph in figure 6.5.

At this point, D is now removed. Since A_1 and A_2 have no ancestors, they are considered separately and the algorithm is finished. If A_1 and A_2 were descendants of other chain components, a chain component with no descendants would be chosen and the algorithm continues until all chain components have been considered.

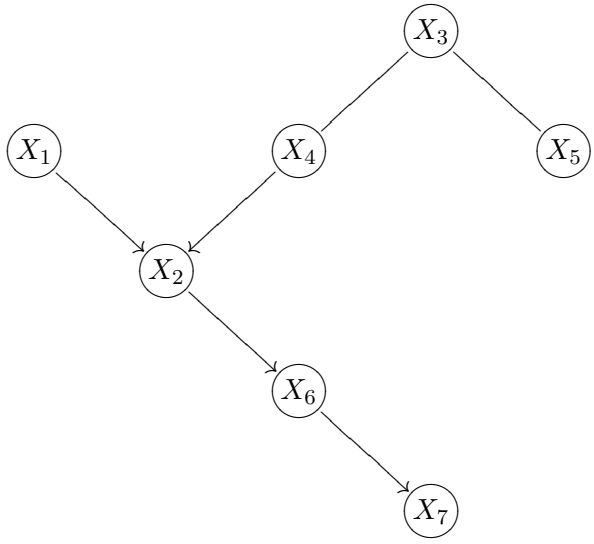

Figure 6.5: Example for RAI algorithm: essential graph

The algorithm is then repeated with conditioning sets of size 2, and so on, until the termination condition is satisfied.

6.2.6 Resolving contradictions

The constraint based algorithms discussed so far, the TPDA, MMPC, Fast and RAI, all rely on the results of conditional independence tests. There are two difficulties that can arise: firstly, the independence structure may not satisfy a faithfulness condition. Secondly, even if it does, with a finite data set, there is not a perfect oracle; the results of the CI (conditional independence) tests are not entirely reliable. A. Fast discusses the issues raised by the second of these difficulties at length in [22] (2010). For example, the MMPC, Fast and RAI algorithms may produce contradictory immoralities; $X \to Y \leftarrow Z$ and $Y \to Z \leftarrow W$.

Tsamardinos, Brown and Aliferis (2006) [63] propose the following solution: after running the MMPC algorithm, they do not try to direct the edges and simply take the skeleton. To produce a DAG, they take the skeleton from MMPC as the candidate edge set and run the MMHC (Maximum Minimum Hill Climbing) algorithm, a search and score algorithm. Starting with an empty graph, at each stage the best add / delete / reverse edge is chosen, where only edges from the skeleton obtained by the MMPC algorithm are considered, according to which operation gives the highest scoring graph. The results in that article indicate that, for the largest test network attempted, the search and score edge orientation phase takes 10 times as long as the constraint based MMPC skeleton finding phase.

Fast takes a different approach, largely on philosophical grounds. Search-and-score presents one criterion for deciding on the best graph, the one that produces the highest score. Constraint based presents an entirely different criterion. The result of each independence test is a 'yes' or a 'no' and

the best fitting graph is the graph that satisfies as many of these constraints as possible. To deal with contradictory immoralities, Fast develops the 'EDGE-OPT' algorithm, which finds the graph that satisfies the largest number of constraints.

The results of conditional independence tests should be consistent, in the sense that they should satisfy *decomposition*, *contraction*, *weak union* and *intersection*. If different CI tests produce results that contradict these relations results, then Bromberg and Margaritis (2009) [5] propose a system of *argumentation*, based on the power of the tests for deciding which results to accept into the set of constraints. These systems of argumentation seem to be in their initial stages; an assumption in Bromberg and Margaritis is that the random variables corresponding to different test statistics are independent.

6.2.7 The Xie - Geng Algorithm

The Xie - Geng algorithm, by Xie and Geng (2008) [67] is a constraint based algorithm that takes a different approach. It is not suitable for sets of *discrete* variables, because the conditioning sets for the CI tests are too large, but works well with multivariate Gaussian variables that satisfy a faithfulness assumption.

The algorithm first constructs the *independence graph*, an undirected graph, which contains an undirected edge $X - Y$ for each pair of variables such that $X \not\perp Y | V \backslash \{X, Y\}$ (X not independent of Y conditioned on all the other variables in the network). In the case of Gaussian variables, this simply reduces to testing whether or not the conditional covariance between the two variables is significant. The authors suggest considering the inverse of the statistical covariance matrix and adding an edge corresponding to each entry that is significantly different from zero. This can be unstable if the number of data is small compared with the number of variables or if the statistical covariance matrix is singular (or close to singular) for some other reason. A generalised inverse is unreliable for this procedure. The conditional covariance for the two variables conditioned on the other variables can always be computed, even when the sample covariance matrix is close to singular.

The *independence graph*, containing an edge if and only if $X \not\perp Y | V \backslash \{X, Y\}$, has the property that graphical separation implies independence. (Note, this is separation in the usual sense for undirected graphs and not d-separation).

Recall that in a Bayesian network, a variable X is d-separated by its Markov blanket $MB(X)$ (definition 2.8) from all the other variables in the network and hence independent of the rest of the network, conditionally on $MB(X)$. Therefore, in the moral graph of a Bayesian network (the undirected graph obtained by linking all the parents of a variable for each variable with undirected edges and then un-directing all the edges), a variable is graphically separated from all the other variables in the network by its Markov blanket.

The edge set of the independence graph is therefore a subset of the moral graph of any DAG corresponding to a factorisation of the distribution; the independence graph and moral graph are equal if the DAG is faithful. The Xie - Geng algorithm takes the view that the independence graph is the moral graph of a faithful DAG. It starts by recursively decomposing the graph, until it has decomposed it into cliques. It then uses the 'faithfulness' principle when dealing with the cliques to

locate immoralities, delete unnecessary edges in the cliques and direct the vee-structures corresponding to immoralities. For a clique C and set $S \subseteq C \backslash \{X, Y\}$, if $X \perp Y | S$, then the edge $X - Y$ is removed and for each $W \in C \backslash (S \cup \{X, Y\})$, the directions $X \rightarrow W \leftarrow Y$ are given to remaining vee structures $X - W - Y$.

It then re-assembles the pieces according to the following principle: an edge $X - Y$ belonging to a structure is removed at re-assembly if it has already been removed from one of the two structures being merged; for any variable W that is not in the sep-set $S_{X,Y}$ (the set that caused $X - Y$ to be removed) where edges $X - W$ and $Y - W$ are present, these edges are directed to form an immorality $X \rightarrow W \leftarrow Y$.

Example 6.3.

Suppose that a probability distribution $\mathbb{P}_{A,B,C,D,E,F,G,H}$ and the DAG in figure 6.6 are faithful.

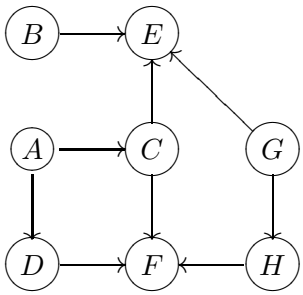

Figure 6.6: A faithful DAG to illustrate the Xie-Geng algorithm

The first step of the algorithm is to construct the independence graph, given in figure 6.7. This is constructed by starting with the empty graph and adding an undirected edge $X \sim Y$ if and only if $X \not\perp Y | V \backslash \{X, Y\}$. If the DAG in figure 6.6 is *faithful*, the independence graph is the moral graph.

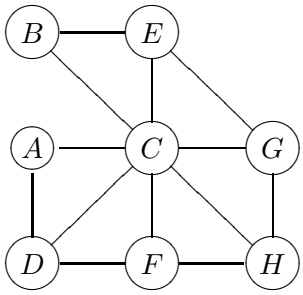

Figure 6.7: A faithful DAG to illustrate the Xie-Geng algorithm

The graph is now decomposed recursively; for example, take $\{C, G\}$, then this decomposes the graph

into $\{A, C, D, F\}$ and $\{B, E, C, F, G, H\}$. The independence graphs for these two sets of variables are illustrated in figure 6.8.

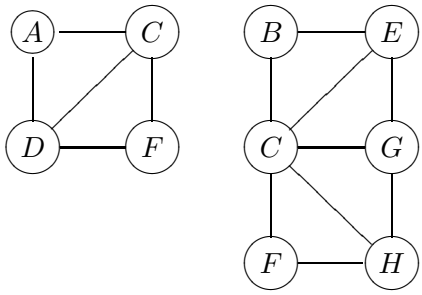

Figure 6.8: First stage of decomposition for the Xie-Geng algorithm

Now consider the piece on the left hand side of figure 6.8. The set $\{C, D\}$ may be used as the separation set, and the independence graphs of the two pieces $\{A, C, D\}$ and $\{C, D, F\}$ are shown in figure 6.9.

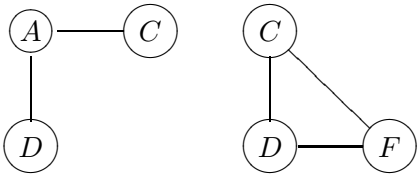

Figure 6.9: Further stage of decomposition for the Xie-Geng algorithm

The edge $C - D$ does not appear in the first graph, since $C \perp D|A$. This is clear from the DAG in figure 6.6, which is faithful to the distribution. It therefore follows that in the reconstruction stage, the edge $C - D$ will not be present and that $C - F - D$ will be an immorality. □

6.3 Hybrid Algorithms

Hybrid algorithms combine constraints with search and score. One example is the MMHC algorithm of Tsamardinos, Brown and Aliferis (2006) [63], which starts with the constraint based MMPC stage to locate the skeleton and then carries out a search and score based MMHC stage, using the skeleton obtained from MMPC as the candidate edge set.

6.3.1 L1-Regularisation

One method, introduced by Schmidt, Niculescu-Mizil and Murphy (2007) [56], places constraints on the model and then uses an L^1 score function, described below, as the basis of a search and score within the constrained space.

The method can be employed with Gaussian or binary variables. The binary case is outlined here.

In this algorithm, there is no restriction on the number of parents that a variable may have, but there is a constraint on the way in which the parents influence the variable. The state space of variable X_j is $\{-1, 1\}$ for each j and the conditional probabilities are modelled so that the *logit* function is linear:

$$\ln\left(\frac{p_{X_j|\Pi_j}(1|\underline{\pi}_j)}{1-p_{X_j|\Pi_j}(1|\underline{\pi}_j)}\right) = \left(\theta_{j,0} + \sum_{k=1}^{p_j} \theta_{j,k}\pi_{j,k}\right) \tag{6.5}$$

where $\underline{\pi}_j = (\pi_{j1}, \ldots, \pi_{jp_j})$, the configuration of Π_j, is a sequence of ± 1 corresponding to the states of the parent variables. The parent variables are only permitted to influence $\ln \frac{p}{1-p}$ linearly; no interactions are permitted. This permits a large number of parents, since the number of parameters is linear, rather than exponential, in the number of parents.

The algorithm works in two stages: like the MMPC, it first produces candidate parent children sets for each variable. Having constrained the search space, it then uses a search and score algorithm to determine the candidate parent / children sets. Having determined the parent / children sets, it runs the hill climbing part of the MMHC algorithm of Tsamardinos, Brown and Aliferis to obtain the structure, keeping the conditional probabilities of the form in equation (6.5). For a vector $\underline{x} = (x_1, \ldots, x_d)$ of 1's and -1's, let Π_j denote all the variables without j. That is, all variables permitted as possible parents for j at this stage. Let $\underline{\tilde{x}}^{(j)}$ denote the vector $\underline{x}$ without x_j. Let

$$LL(j, \underline{\theta}_j, \underline{x}) = \log p_{X_j|\Pi_j}(x_j|\underline{\tilde{x}}^{(j)})$$

denote the log likelihood function and, for $\mathbf{x}$ the data matrix with rows $\underline{x}_{(1)}, \ldots, \underline{x}_{(n)}$, let

$$LL(j, \underline{\theta}_j, \mathbf{x}) = \sum_{k=1}^{n} LL(j, \underline{\theta}_j, \underline{x}_{(k)}).$$

The parameters $\underline{\theta}_j$ are chosen to maximise the $L1$ regularisation score function,

$$L_1R(\underline{\theta}_j, \mathbf{x}) = LL(j, \underline{\theta}_j, \mathbf{x}) - \lambda\|\underline{\theta}_j\|_1,$$

where $\|\underline{\theta}_j\|_1 = \sum_{k=1}^{d-1} |\theta_{jk}|$ and λ is chosen appropriately. The sum is over the parameters corresponding to dependence on parent variables; the parameter $\theta_{j,0}$ is not included. The article [56] has some discussion about the appropriate choice of λ.

The L^1 regularisation, if λ is appropriately chosen, has the effect of choosing vectors $\underline{\theta}_j$ with a substantial number of zero components. Because of this property, it tends to favour a lower number of parameters in the model. For this reason, L^1 regularisation is a technique that is developing increasing importance.

Gibbs sampling A related approach to the problem of structure learning is found in Bulashevska and Eils (2005) [6]. The structure learning algorithm is intended for analysis of gene expression data, to locate gene regulatory interactions. As with Schmidt, Niculescu-Mizil and Murphy (2007) [56], the

parents influence the offspring independently of each other and the algorithm forms 'noisy OR' and 'noisy AND' gates. The parent sets are chosen using Gibbs sampling.

Chapter 7

Evaluation of Structure Learning and 'Causal Discovery'

7.1 Faithfulness and 'real world' data

The Recursive Autonomy Identification algorithm was analysed by B. Barros (2012) [3], applying it both to data simulated from test networks and to a financial data set. When applied to simulated data, simulated from the ALARM network, the algorithm performed very well; the performance was consistent with the results described by Yehezkel and Lerner [68]. For a data set generated by a probability distribution for which there exists a faithful DAG, the results verified that the algorithm is efficient and produces a graph that corresponds well to the distribution that generated the data, with low computational overheads. The feature of the algorithm of making all required tests with smaller conditioning sets before moving on to larger increases accuracy over methods that do not do this. The additional use made of the structure, identifying the chain components of the essential graph at each stage, ensures that fewer statistical calls (references to the data set) are required.

Some features were noted in the performance of the algorithm. In earlier stages, some contradictory directions appeared. That is, pairs of immoralities $X \to Y \leftarrow Z$, $Y \to Z \leftarrow W$, in situations where the edge $Y \sim Z$ would be deleted in subsequent rounds of the algorithm following tests with larger conditioning sets. The direction chosen for the edge during that round was dictated by which immorality appeared first. If the test $X \perp Z | S_{X,Z}$, yielding a sep-set $S_{X,Z}$ was carried out first, then the edge would take the direction $Y \leftarrow Z$. After carrying out the CI tests and determining the directions, Meek's orientation rules were applied to determine the structures for the next round of the algorithm.

The algorithm worked very well; with 10000 observations, it produced a graph that had the correct skeleton and only 4 edges with incorrect orientation.

The test of performance of an algorithm, MMHC, RAI, TPDA, or any other algorithm, is based on the ability of the algorithm to recover a probability distribution used to simulate data. There are several standard networks, including the ALARM network, that are used. Data is simulated from the network and the algorithm applied to the simulated data. Freedman and Humphreys (2000) p 33,34 [24] are somewhat scathing in their assessment of this procedure for verifying the utility of an

algorithm, of using simulated data from a distribution known to have good properties. They write,

> The ALARM network is supposed to represent causal relations between variables relevant to hospital emergency rooms, and Spirtes Glymour Scheines (1993) [58] p 11 claim to have discovered almost all the adjacencies and edge directions 'from sample data'. However, these 'sample data' are simulated; the hospitals and patients exist only in the computer program. The assumptions made by SGS (1993) [58] are all satisfied by fiat, having been programmed into the computer: the question of whether they are satisfied in the real world is not addressed. After all, computer programs operate on numbers, not on blood pressures or pulmonary ventilation levels (two of the many evocative labels on nodes in the ALARM network).

Freedman and Humphreys continue by stating,

> These kinds of simulations tell us very little about the extent to which modelling assumptions hold true for substantive applications.

The constraint based algorithms TPDA, MMPC, Fast and RAI all depend crucially on the modelling assumption that there is a DAG that is faithful to the set of conditional dependence / independence statements that can be established. We pinpoint two difficulties that can arise in the 'real world'; interaction effects without main effects and hidden common causes.

7.2 Interaction effects without main effects

Example 3.7 gives an example of a situation where these constraint based algorithms will miss key associations between the variables. Any situation where factors taken individually give no information, but where there are two-factor, or higher order factor interaction without main effects, will not be detected. If applied to genetic data, for example, the algorithm will not be able to detect situations where a single gene by itself has no apparent effect, but where the genome pathway may be opened by two genes acting together.

This situation will not lead to internal inconsistencies in the functioning of the algorithms; associations of this type will simply be missed and the output will be a DAG that does not show these associations, but it may not lead to reversed edges (situations where the algorithm has to choose between two contradictory directions for an edge).

Hidden variables In a 'real world' situation, there may well be hidden variables which are not measured and the experimenter may be unaware of their existence. This can lead to reversed edges, as the following example illustrates. Suppose that X, Y, Z, W are variables that are recorded, while H is a hidden variable, a common cause of X and Y, whose presence is not suspected by the researcher. Suppose that the causal relations between H, X, Y, Z, W are given by figure 7.1.

If the RAI algorithm is applied to the variables X, Y, Z, W, whose associations are described by the d-connection statements of the DAG in figure 7.1, then $X \perp Z|W$, giving $X \rightarrow Y \leftarrow Z$ and

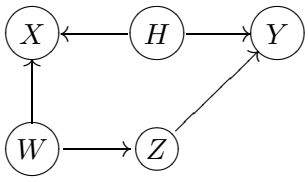

Figure 7.1: H is hidden and does not appear in the data matrix

$Y \perp W|Z$, giving the immorality $Y \rightarrow X \leftarrow W$. Even if there is a perfect oracle (sufficient data to give correct results for each CI test so that the results are consistent with the probability distribution over (X, Y, Z, W)), the edge between X and Y is a *reversed edge*, $X \leftrightarrow Y$. This notation means that, from the CI tests, one test gives a direction $X \rightarrow Y$; the other gives a direction $X \leftarrow Y$ and the algorithm will choose the direction depending on the order in which the tests are carried out.

In the RAI algorithm, the direction that an edge takes in the output graph, under such circumstances is determined by the order of the variables; if the test results $X \perp Z|W$ appears first, the output graph will contain $X \rightarrow Y$ and thus the graph will contain the false d-separation statement $W \perp Y|\{X, Z\}$, while if the result $W \perp Y|Z$ appears first, the output graph will contain the edge $Y \rightarrow X$ and the false d-separation statement $X \perp Z|\{W, Y\}$. The two possibilities are given in figure 7.2.

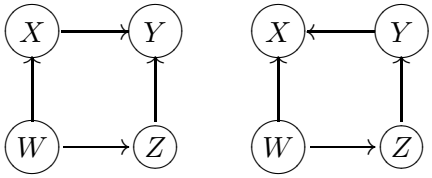

Figure 7.2: Possible outputs applying constraint based algorithm to variables (X, Y, Z, W) from figure 7.1

Contradictory results from CI tests Suppose that the test results give $X \not\perp Y$. One would therefore hope that X and Y are d-connected in the output graph. The decision to remove an edge $X \sim Y$ as soon as a test result 'do not reject $X \perp Y|S$' appears can lead to a situation where the edge $X \sim Y$ is removed, even if it leads to a graph where $X \perp Y \| _{\mathcal{G}} \phi$ (X and Y are d-separated by the empty set). The constraint based algorithms discussed here do not have a mechanism to ensure that the final graph contains those d-connection statements that have been established through rejecting independence.

The scope of structure learning Algorithms can detect associations, at the level of 'descriptive statistics', without reference to the process that generates the data and the nature of randomness. At

the level of descriptive statistics, the scope of constraint based algorithms is viewed along the following lines: from the $n \times d$ data matrix, an empirical distribution can be established (or, at least, if d is very large, empirical probability distributions of the marginalisation to subsets of the variables can be established). Any test result that produces $X \not\perp Y|S$ corresponds to a d-connection statement that is to be retained in the output graph; any test result where $X \not\perp Y|S$ is not rejected does not have to be retained in the output graph. The output graph attempts to have as few edges as possible, while retaining all the d-connection statements that were established through rejecting independence.

For large numbers of variables, there are clear difficulties that make serious inferential statistics impossible. The assumption is that the $n \times d$ data matrix represents n independent instantiations of a d-random vector $\underline{X}$. This assumption, together with an assumption that n is sufficiently large for a central limit theorem effect to hold is required for the test statistics to be approximately χ^2. Even if the nominal significance level α chosen for rejecting a null hypothesis can be considered as a measure of a probability in any serious way the number of tests required is large that the overall significance level could be close to 1. In terms of descriptive statistics, the output graph can be informative, but it is difficult to reach inferential conclusions from the output of these algorithms.

Application of Fast and RAI to financial data After testing the Fast and RAI algorithms on the training example of the ALARM network, where it performed well, the work of Barros [3] proceeded to run these algorithms on a financial data set, composed of the closing values of 18 stock market indices (Amsterdam stock index, Austrian traded index, Brussels stock index, etc ...) from 1st January 2005 to 1st January 2011, approximately 1000 instantiations of 18 variables.

The aim of the thesis was to detect *changes* in associations between the variables, to learn a structure, detect when the structure was no longer appropriate and update.

In the financial data set, the raw RAI algorithm gave no independence statements after the first round; for each pair of variables (X, Y), the result was 'reject independence'. Therefore, any pair of variables should be d-connected in the output graph. Yet the output graph, following application of the raw RAI algorithm, gave pairs of d-separated variables, which indicates that conditional independence was falsely accepted due to weak tests.

In order to deal with the situation where 'accept independence' from tests with large conditioning sets contradicted d-connection statements with lower order conditioning sets, Barros adopted a more conservative approach than the argumentation of Bromberg and Margaritis [5] and modified the algorithm so that it did not accept an independence statement that resulted in a d-separation in the output graph contradicting a dependence statement that has already been established. This modification worked well.

The output still gave a large number of 'reversed edges'. While the ALARM network gave one or two, the financial data set gave approximately 28 reversed edges, indicating situations that appeared in the DAG in figure 7.1, with possible output graphs corresponding to figure 7.2.

The presence of a *substantial* number of 'common cause' hidden variables would explain this.

This was a randomly chosen 'real world' data set and probably not appropriate for an algorithm based on a 'faithfulness' assumption. The variables here do not satisfy one of the motivating features of

the faithfulness assumption, that the variables stand in *causal* relation to each other; their association is more likely to be a result of hidden common causes, such as government policies, or global financial considerations that influence the various stock markets.

The same difficulties seemed to arise in other applications. The RAI algorithm was applied to the genetic data found in Friedman et. al. [23]. Tentative results seem to give substantially different output depending on the input order of the variables, suggesting hidden common causes.

Conclusion Constraint based algorithms offer a fast approach, which is convenient with data matrices when d, the number of variables, is very large. They can be many times faster than search and score algorithms. Unfortunately, these algorithms tend to assume 'faithfulness' and work on the principle of removing an edge whenever a conditional independence test gives the result 'do not reject $X \perp Y|S$'. This leads to several difficulties. Firstly, since tests with larger conditioning sets are weaker, it can lead to situations where deletion of an edge can contradict earlier d-connection statements. This difficulty is present even if there is a faithful DAG corresponding to the independence structure. Secondly, two-factor, or higher order interactions are not detected if there are no 'main effects'. Thirdly, hidden variables can lead to contradictory edges, resulting in d-separation statements not present in the probability distribution. If there is no faithful DAG that describes the underlying independence structure, this can manifest itself in other ways.

Modifications to remove the first of these difficulties have been considered, for example by Bromberg and Margaritis [5] using argumentation and the more conservative approach of Barros [3] retaining all dependence statements that have been established through rejecting independence.

The second and third of these difficulties have not been fully addressed by constraint based algorithms.

7.3 The 'Causal Discovery' Controversy

The discussion about structure learning has described various methods to locate structures that represent the independence relations within a data set. All these methods, search and score, constraint based, hybrid, yield results that fall under the heading of *descriptive statistics*. The search and score methods simply examine some of the available structures and choose the structure with the highest score of those examined. On the 'classical' side, there is no measure of confidence for the structure chosen; on the 'Bayesian' side, even if a prior distribution is placed over the structure space and the posterior used as the basis of a score function, there is no posterior assessment of the probability for the structure to lie in a certain subspace of the set of possible structures; only a small number of structures are visited and the structure chosen is the one visited that gives the largest score. With constraint based methods, even if the hypothesis that the data matrix represents n instantiations of i.i.d. random vectors held, the number of tests is so large that even with a small nominal significance level for each test, the overall significance level approaches 1.

The output structure can give useful information at the level of descriptive statistics, but little or no formal inference can be made. This is generally the case in multivariate statistics, where methods

are often more successful as descriptive than inferential tools.

Assume, though, that statistical associations have been established. Substantial parts of the literature suggest claims that a rigorous engine for inferring *causation* from association has been established. For example, Spirtes, Glymour and Scheines (1993) [58] claim to have algorithms for discovering causal relations based only on empirical data. The underlying assumption seems to be that, for a large class of problems, when immoralities are learned from data and Meek's rules then applied, cause to effect can be inferred for the directed edges of the essential graph. Schmidt, Niculesu-Mizil and Murphy (2007) [56] write, explaining why they are constructing techniques to produce *directed* graphs,

> '... undirected models cannot be used to model causality in the sense of Pearl [47], which is useful in many domains such as molecular biology, where interventions can be performed.'

The thrust of the quote is that directed edges whose direction can be interpreted as cause to effect, can be learned from data. But placing a causal interpretation on a directed arrow in a p-DAG that has been learned purely by applying a structure learning algorithm to data can be misleading.

In a situation where interventions can be performed, a *causal* directed graph can be obtained from the undirected graph through further controlled experiments. Consider the situation on three variables (X, Y, Z) where $X \perp Z|Y$, but $X \not\perp Y$, $X \not\perp Z$, $Y \not\perp Z$, $Y \not\perp X|Z$ and $Y \not\perp Z|X$. There are three DAGs along which the distribution $p_{X,Y,Z}$ may be factorised, given in figure 7.3.

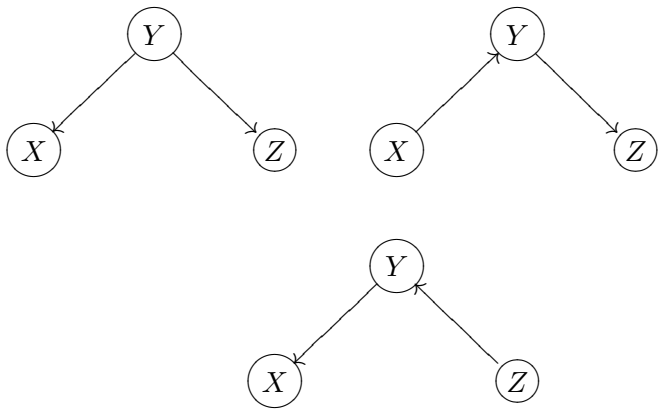

Figure 7.3: Three Markov equivalent DAGs

Suppose that an intervention may be carried out on the variable Y, forcing its state. This has the effect of removing arrows from parents of Y to Y. If the state $Y \leftarrow y$ is forced, this gives the graphs in figure 7.4.

If all the states of Y can be explored, in a controlled experiment, by randomly assigning levels of the 'treatment' variable Y, the causal structure can be determined from the Markov structure, but not otherwise.

As Freedman and Humphreys point out (1999) [24], commenting on automated causal learning, 'these claims are premature at best and the examples used in [58] to illustrate the algorithms are

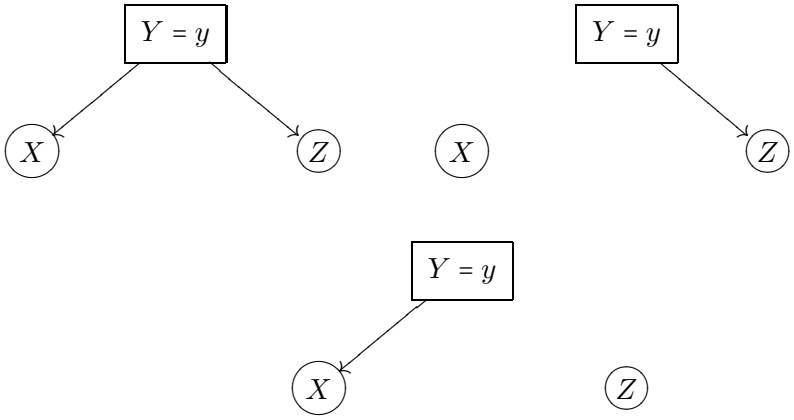

Figure 7.4: Intervention $Y \leftarrow y$ in figure 7.3

indicative of failure rather than success.' They point out that 'the gap between association and causation has yet to be bridged.'

7.4 Faithfulness and the great leap of faith

One of the leading assumptions behind 'causal discovery' is the assumption that distributions of interest satisfy the faithfulness assumption, that there is a DAG $\mathcal{G}$ with variable set $V = (U, O)$ where U denotes the unobserved variables and O the observed variables and a probability distribution $\mathbb{P}$ over (U, O) such that $\mathbb{P}$ factorises along $\mathcal{G}$ and $\mathcal{G}$ gives a faithful graphical representation of the independence structure.

This is described as follows;

> ' .... the faithfulness condition can be thought of as the assumption that conditional independence relations are due to causal structure rather than to accidents of parameter values.' Spirtes et. al. (2000) [59]

Example 3.7 gives an instance of a situation where the probability distribution does not have faithful graphical representation. For the variables $(Y_1, Y_2, Y_3, X_1, X_2, X_3)$, the DAG that best represents the associations between the variables is given by figure 6.1. In this graph, $X_1 = 1$ if $Y_2 = Y_3$ and 0 otherwise. $X_1 \perp Y_2$ and $X_1 \perp Y_3$, but $X_1 \not\perp \{Y_2, Y_3\}$. In this situation the influence of Y_2 and Y_3 on X_1 is not seen if the variables are considered separately, but the interaction effect is decisive.

Another statement of the same principle is found in Meek (1995) [38]

> In cases where $\mathcal{P}(\mathcal{G})$ (the set of distributions that factorise along a graph $\mathcal{G}$) can be parametrised by a family of distributions with a parameter of finite dimensions, the set of unfaithful distributions typically has Lebesgue measure zero. (Spirtes et. al. (2000) [59] pp 42 - 2)

This assumption, that the set of observable variables O may be extended to a set $V = (U, O)$ where U represents unobserved common causes, or confounders, and that there will exist a DAG over V that is faithful to the probability distribution over V, is re-stated in Robins, Scheines, Spirtes and Wasserman (2003) [53]. There is strong interest in classes of faithful distributions in the literature; the work of Zhang and Spirtes [69] requires that the class of distributions under consideration satisfy a stronger assumption than faithfulness in order to obtain uniform consistency in causal inference for a certain class of problems; [53] illustrates non-existence of uniform consistency when only faithfulness is assumed, because of the possibility of non-faithful distributions in the closure of the set of distributions under consideration.

Consider again example 3.7 and suppose that $O = (X_1, X_2, X_3)$, the values for (X_1, X_2, X_3) are observable and $U = (Y_1, Y_2, Y_3)$, the results of (Y_1, Y_2, Y_3) are hidden. Clearly, the set of distributions over 6 binary variables that factorises over the DAG in figure 6.1 can be described by a finite parameter space; 15 parameters are required to describe the entire set of distributions; the parameter space is $[0, 1]^{15}$. Furthermore, it is clear that the parameters to describe the distribution over $(Y_1, Y_2, Y_3, X_1, X_2, X_3)$ in example 3.7 correspond to exactly one point in the parameter space, which has Lebesgue measure zero. Nevertheless, examples where knowledge of two causes is required to explain the effect and where knowledge only of a single cause tells you nothing about an effect arise all the time in practise, in the real world.

Furthermore, the parametrisation of any distribution that has an independence structure has Lebesgue measure zero in the parameter space of all distributions over the variables in question. Meek's argument can equally well be used to argue against searching for any independence structure at all.

Faithfulness appears a convenient hypothesis to produce beautiful mathematics (and the relation between DAGs and probability distributions under this assumption has produced a very elegant and attractive mathematical theory), but it is difficult to see that it necessarily applies to real world situations; the real world does not respect the fact that the set of parameters that describe the situation have Lebesgue measure zero in a mathematical parameter space. Divergence between 'real world' behaviour and the assumption that it should fit into a convenient mathematical framework has been termed 'The Mind Projection Fallacy' by E.T. Jaynes (2003) [31].

7.5 Inferring non-causation and causation

Robins, Scheines, Spirtes and Wasserman (2003) [53] describe situations where *non*-causation can be inferred. A situation where such an inference can be made is given by figure 7.1 representing the *causal* associations between variables, where H is hidden and X, Y, W are observable. In this example, X is not a cause of Y, neither is Y a cause of X. This can be inferred from the CI tests; from the results $X \perp Z|W$ and $Y \perp W|Z$, it is possible to infer that the relation between X and Y is not cause to effect in either direction and that a common cause H would explain the test results.

The discovery of an immorality, though, does not necessarily imply causation. Suppose H_1 and H_2 are hidden and X, Z, Y are observable in figure 7.5. The distribution over (X, Z, Y) factorises

according to figure 7.6.

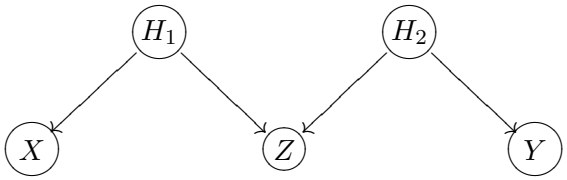

Figure 7.5: H_1 and H_2 hidden

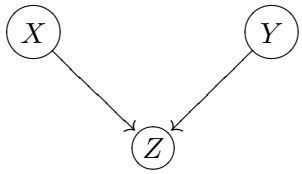

Figure 7.6: DAG for (X, Y, Z) from figure 7.5

If one were using immoralities as a guide to causation, one would conclude that X and Y were common causes of Z. As Freedman and Humphreys point out in [24], commenting on Spirtes Glymour Scheines (1993) [58] on a DAG produced from a sociological data set,

> The graph says, for instance, that race and religion cause region of residence.

In the context, this is non-sensical and raises a timely note of caution when inferring causality.

7.6 Summarising causal discovery

Freedman and Humphreys go on to summarise the attempts to automate 'causal discovery' with the example of smoking and lung cancer,

> The epidemiologists discovered an important truth - smoking is bad for you. The epidemiologists made this discovery by looking at the data and using their brains, two skills that are not readily automated. .... The examples in SGS (1993) [58] count against the automation principle, not for it.'

The conclusion drawn by the authors of this article is that the output produced by structure learning algorithms provides invaluable information. It can give good information about associations and can certainly point towards the possibility of causal relations, but they do not even begin to automate the process of learning causality; it is still necessary for researchers to use their brains to design experiments,

examine the data and use their brains again, taking into account circumstances and contexts additional to the raw data, to reach conclusions. As the example from SGS (1993) [58], extended by Freedman and Humphreys [24] shows, causation cannot be deduced from the presence of an immorality and, indeed, cannot be inferred from the output of structure learning algorithms alone.

7.7 Conclusion

The subject of Bayesian networks arguably has its origins in the short article of A. Cayley (1853) [8]. Cayley presents a *causal* network and exploits the factorisation of a joint probability distribution into its component parts based on causal principles. Cayley introduces the idea of the 'or' gate and points towards the use of techniques of algebraic geometry in Bayesian networks.

Graphical models provide an interaction between probability and graph theory, where separation statements in a graph imply independence statements in a probability distribution, so that certain independence statements can be established simply from d-separation statements in the corresponding DAG. Human intelligence finds in graphs a very natural and lucid (or graphic!) way of expressing connections between variables.

Bayesian networks are a versatile tool of artificial intelligence, as any artificial intelligence in real life must be able to reason probabilistically, in order to cope with uncertainty. They have a wide range of applications; for example, reliability theory, system security and in bioinformatics, where Bayesian network structure learning techniques are used to locate genome pathways. A potentially important field of applications is expert systems in medical diagnostics. Bayesian networks are useful in machine learning, i.e. supervised classification when the 'naïve Bayes' assumption is dropped.

If there are causal relations between variables, then the DAG provides a natural tool for representing a natural factorisation of the probability distribution, which has been exploited by Pearl to develop the so-called *intervention* calculus. This provides a natural language for describing controlled experiments.

One topic where further development is necessary is that of structure learning, learning the independence structure from data. The discussion of constraint based versus search and score algorithms indicates that, while constraint based algorithms have a substantial advantage in terms of computational speed, some of the assumptions on which these algorithms are based mean that there are certain important classes of association that the algorithms simply cannot detect. An important task is to relax these assumptions without increasing the computational complexity too drastically.

The problem of learning from data a graph that expresses the independence structure continues to be a major challenge. The number of possible graphs increases super-exponentially in the number of nodes, so that it is not possible to examine all structures available. There are two main philosophies behind structure learning techniques; search-and-score and constraint based. There are also hybrid algorithms, which use ideas from both. Search-and-score tend to employ a stochastic process to run through the space of structures, scoring each structure and then choosing the structure visited that has the highest score. Constraint based algorithms set up constraints through tests for conditional independence. If conditional independence is rejected, the corresponding d-connection statement should be present in the output graph; the output tries to give the graph with fewest edges that respects all

the connection statements that have been established.

While constraint based algorithms tend to be overwhelmingly faster and less computationally demanding, they tend to have restrictive assumptions. Those that rely on faithfulness tend to perform rather poorly when the underlying distribution does not satisfy this assumption. This happens, for example, if there are hidden common causes. The algorithms also fail to detect situations where there are interaction effects without main effects; X_1 by itself gives no information about X_3 and X_2 by itself gives no information about X_3, but $\{X_1, X_2\}$ together give full information about X_3.

The main challenge in structure learning is to develop algorithms that have the computational efficiency of the constraint based algorithms, while relaxing assumptions such as faithfulness, or composition, for the underlying distribution.

Closely connected with the assumption of faithfulness is the idea of causation. Constraint based algorithms assuming faithfuless will work well in situations where there are cause - to - effect relations between the variables. Learning causality from associations, though, seems premature without a substantial number of additional assumptions on the nature of the data available.

Literature Cited

[1] Akaike, H. [1974] *A new look at the statistical model identification* IEEE Transactions on Automatic Control vol 19 no 6 pp 716–723.

[2] Andersson, S.A.; Madigan, D.; Perlman, M.D.; Triggs, C.M. [1997] *A graphical characterisation of lattice conditional independence models* Annals of Mathematics and Artificial Intelligence vol. 21 pp. 27 - 50

[3] Barros, B. [2012] *Incremental Learning Algorithms for Financial Data Modelling* Master's Thesis, Linköping University, Department of Mathematics LiTH-MAT-INT-A–2012/01–SE

[4] Beeri, C.; Fagin, R.; Maier, D.; Yannakakis, M. [1983] *On the desirability of acyclic database schemes* J. Assoc. Comput. Mach. **30** pp 479 - 513.

[5] Bromberg, F.; Margaritis, D. [2009] *Improving the reliability of causal discovery from small data sets using argumentation* Journal of Machine Learning Research vol. 10 pp. 301 - 340

[6] Bulashevska, S.; Eils, R. [2005] *Inferring genetic regulatory logic from expression data* Bioinformatics vol 21 no 11 pp 2706 - 2713

[7] Castelo, R.; Siebes, A. [2000] *Priors on network structures. Biasing the search for Bayesian networks* International Journal of Approximate Reasoning vol 24 pp 39 - 57

[8] Cayley, A. [1853] *Note on a Question in the Theory of Probabilities* The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science vol. VI. - fourth series July - December, 1853, Taylor and Francis. p. 259

[9] Cayley, A. [1854] *On the theory of groups as depending on the symbolic equation* $\theta^n = 1$ Phil. Mag. vol. 7 no. 4 pp 40 - 47

[10] Cayley, A. [1858] *A Memoir on the Theory of Matrices* Phil. Trans. of the Royal Soc. of London, vol 148 p. 24

[11] Cayley, A. [1869] *A Memoir on Cubic Surfaces* Philosophical Transactions of the Royal Society of London (The Royal Society) vol 159 pp 231–326

[12] Cayley, A. [1878] *Desiderata and suggestions: No. 2. The Theory of groups: graphical representation* Amer. J. Math. vol. 1 no. 2 174–176

[13] Cayley, A. [1889] *A Theorem on Trees* Quarterly Journal of Mathematics vol 23 pp 276-378

[14] Cheng, J.; Greiner, R.; Kelly, J.; Bell, D. A.; Liu, W. [2002] *Learning Bayesian networks from data: An information-theory based approach* Artificial Intelligence vol 137 pp 43 - 90.

[15] Chickering, D. M. [2002] *Optimal structure identification with greedy search* Journal of Machine Learning Research, 507–554.

[16] Chickering, D.M.; Heckerman, D.; Meek, C. [2004] *Large Sample Learning of Bayesian Networks is NP - Hard* Journal of Machine Learning Research vol. 5 pp. 1287 - 1330

[17] Chow, C.K.; Liu, C.N. [1968] *Approximating Discrete Probability Distributions with Dependence Trees* IEEE Transactions on Information Theory, vol. IT - 14 no. 3

[18] Cooper, G.F. [1990] *The Computational Complexity of Probabilistic Inference using Bayesian Belief Networks* Artificial Intelligence vol. 42 pp. 393 - 405

[19] Cooper, G.F.; E. Herskovitz, E. [1992]*A Bayesian Method for the Induction of Probabilistic Networks from Data* Machine Learning vol. 9 pp. 309 - 347

[20] Corander, J.; Ekdahl, M.; Koski, T. [2008]*Parallel interacting McMC for learning topologies of graphical models* Data mining and knowledge discovery vol 17 no. 3 pp 431-456 Springer

[21] R.G. Cowell, A.P. David, S.L. Lauritzen and D.J. Spiegelhalter [1999] *Probabilistic Networks and Expert Systems* Springer, New York

[22] Fast, A. [2010] *Learning the structure of Bayesian networks with constraint satisfaction* Ph.D. thesis, Graduate School of the University of Massachusetts Amherst, Department of Computer Science

[23] Flores, M.J.; Nicholson, A.E.; Brunskill, A.; Korb, K.B., Mascaro, S. [2011] *Incorporating expert knowledge when learning Bayesian network structure: A medical case study* Artificial Intelligence in Medicine vol 53 pp 181 - 204

[24] D. Freedman and P. Humphreys [1999]*Are there Algorithms that Discover Causal Structure?* Synthese vol. 121 pp. 29 - 54

[25] Friedman, N.; Nachman, I.; Pe'er, D. [1999] *Learning Bayesian network structure from massive datasets: the 'sparse candidate' algorithm* Proc. Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI '99) pp 196 - 205

[26] Friedman, N.; Linial, M.; Nachman, I.; Pe'er, D. [2000] *Using Bayesian Networks to Analyse Expression Data* Journal of Computational Biology **7** no 3/4 pp 601 - 620

[27] Friedman, N. [2004] *Inferring Cellular Networks Using Probabilistic Graphical Models* Science Vol 303 no 5659 pp 799-805 DOI: 10.1126/science.1094068

[28] D. Geiger, T. Verma and J. Pearl [1990] *Identifying Independence in Bayesian Networks* Networks vol. 20 pp. 507 - 534.

[29] Hartmanis, J. [1959]*Application of some Basic Inequalities for Entropy* Information and Control vol. 2 pp 199 - 213

[30] D. Heckerman, D. Geiger and D.M. Chickering [1995] *Learning Bayesian Networks: The Combination of Knowledge and Statistical Data* Machine Learning vol. 20 pp. 197 - 243

[31] Jaynes, E.T. [2003] *Probability Theory. The Logic of Science* Cambridge University Press

[32] Kellerer, H.G. [1991]*Indecomposable marginal problems* Advances in probability distributions with given marginals: beyond the copulas, Springer Verlag, Berlin, pp 139 - 149

[33] Lewis II, P.M. [1959]*Approximating Probability Distributions to Reduce Storage Requirements* Information and Control vol. 2 pp 214 - 225

[34] Lindley, D. [2002] *Seeing and Doing: The Concept of Causation* International Statistical Review / Revue International de Statistique vol. 70 pp 191 - 197

[35] Madigan, D.; Andersson, S.A.; Perlman, M.D.; Volinsky, C.P. [1996] *Bayesian Model Averaging and Model Selection for Markov Equivalence Classes of Acyclic Digraphs* Communications In Statistics: Theory and Methods vol. 25, no. 11 pp. 2493-2519

[36] Madigan, D.; York, J. [1995] *Bayesian Graphical Models for Discrete Data* International Statistical Review vol. 63 pp. 215 - 232

[37] Malvestuto, F.M. [1988] *Existence of extensions and product extensions for discrete probability distributions* Discrete mathematics, vol. 69 pp 61 - 77

[38] Meek, C. [1995] *Causal inference and causal explanation with background knowledge* Proceedings of the Eleventh Conference on Uncertainty in Artificial Intellegence pp 403 - 410

[39] Moore, A.; Wong, W-K. [2003] *Optimal Reinsertion: A new search operator for accelerated and more accurate Bayesian network structure learning* Proceedings of the Twentieth International Conference on Machine Learning (ICML - 2003), Washington DC

[40] Nelson, E. [1987] *Radically Elementary Probability Theory* Princeton University Press

[41] J. Pearl [1982] *Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach* AAAI - 82 Proceedings pp. 133 - 136

[42] J. Pearl [1987] *Evidential Reasoning Using Stochastic Simulation of Causal Models* Artifical Intelligence, vol. 32, pp. 245-257.

[43] Pearl, J. [1988] *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* Morgan Kaufmann, San Mateo, CA.

[44] Pearl, J. [1990] *Probabilistic Reasoning in Intelligent Systems* 2nd revised printing, Morgan and Kaufman Publishers Inc., San Fransisco

[45] Pearl, J. [1995] *Causal Diagrams for Empirical Research* Biometrika vol. 82 pp. 669 - 710

[46] Pearl, J. [1995] *Causal Inference from Indirect Experiments* Artificial Intelligence in Medicine vol. 7 pp. 561 - 582

[47] Pearl, J. [2000] *Causality* Cambridge University Press

[48] Pearl, J.; Geiger, D.; and Verma, T. [1989] *Conditional Independence and its Representations* Kybernetica vol. 25 no. 2 pp. 33 - 44

[49] Pearl, J; and Verma, T. [1987] *The Logic of Representing Dependencies by Directed Acyclic Graphs* Proceedings of the AAAI, Seattle, Washington pp. 374 - 379

[50] Pearl, J. [1995] *Causal diagrams for empirical research* Biometrika vol 82 pp 669–710

[51] Rebane, G.; Pearl, J. [1987] *The recovery of causal poly-trees from statistical data* AAAI-87 Workshop on Uncertainty in AI, Seattle, Washington

[52] Rissanen, J. [1978] *Modelling By Shortest Data Description* Automatica, vol 14 pp 465-471

[53] Robins, J.M.; Scheines, R.; Spirtes, P.; Wasserman, L. [2003] *Uniform consistency in causal inference* Biometrika vol 90 no 3 pp 491- 515

[54] Robinson, R.W. [1977] *Counting Unlabelled Acyclic Digraphs* Springer Lecture Notes in Mathematics: Combinatorial Mathematics V, C.H.C. Little (ed.) pp. 28 - 43.

[55] Sadeghi, K.; Lauritzen, S, [2012] *Markov Properties for Mixed Graphs* submitted to Bernoulli, available on arxiv
`http://arxiv.org/pdf/1109.5909v2.pdf`

[56] Schmidt, M.; Niculescu-Mizil, A.; Murphy, K. [2007] *Learning graphical model structure using l1-regularization paths* Proceedings of the National Conference on Artificial Intelligence vol 22 no 2 pp 12- 78

[57] Schwartz, G.E. [1978] *Estimating the dimension of a model* Annals of Statistics vol 6 no 2 pp 461–464.

[58] Spirtes, P.; Glymour, C.; Scheines, R. [1993] *Causation, Prediction and Search* Lecture Notes in Statistics no. 81 Springer-Verlag New York

[59] Spirtes, P.; Glymour, C.; Scheines, R. [1997] *Reply to Humphreys and Freedman's Review of Causation, Prediction, and Search* British Journal for the Philosophy of Science vol 48 pp 555 - 568.

[60] Spirtes, P.; Glymour. C.; Scheines, R. [2000] *Causation, Prediction and Search* second edition, The MIT press.

[61] Studený, M. [2005] *Probabilistic Conditional Independence Structures* Springer Verlag.

[62] Sturmfels, B. [2002] *Solving Systems of Polynomial Equations* In: CBMS Lectures Series, American Mathematical Society.

[63] Tsamardinos, I.; Brown, L.E.; Aliferis, C.F. [2006] *The Max - Min Hill - Climbing Bayesian Network Structure Learning Algorithm* Machine Learning vol. 65 pp. 31 - 78

[64] Vorobev, N. N. [1962] *Consistent families of measures and their extensions* Theory of Probability and its Applications vol. 7 pp 147 - 162

[65] S. Wright [1921] *Correlation and Causation* Journal of Agricultural Research vol. 20 pp. 557 - 585

[66] Wright, S. [1934] *The method of path coefficients* Ann. Math. Statist. vol 5 pp 161 - 215.

[67] Xie, X.; Geng, Z. [2008] *A recursive method for structural learning of directed acyclic graphs* Journal of machine learning research vol. 9 pp. 459 - 483

[68] Yehezkel, R.; Lerner, B. [2009] *Bayesian network structure learning by recursive autonomy identification* Journal of Machine Learning Research vol. 10 pp 1527 - 1570

[69] Zhang, J; Spirtes, P. [2002] *Strong faithfulness and uniform consistency in causal inference* Proceedings of the nineteenth conference on uncertainty in artificial intelligence pp 632–639, Morgan Kaufmann Publishers Inc.