Approximate Bayesian Learning of Partition Directed Acyclic Graphs

KARL AMUNDSSON
Approximate Bayesian Learning of Partition Directed Acyclic Graphs

KARL AMUNDSSON

Master’s Thesis in Mathematical Statistics (30 ECTS credits)
Master Programme in Applied and Computational Mathematics (120 credits)
Royal Institute of Technology year 2016
Supervisor at KTH: Timo Koski
Examiner: Timo Koski

TRITA-MAT-E 2016:61
ISRN-KTH/MAT/E--16/61-SE

Royal Institute of Technology
SCI School of Engineering Sciences
KTH SCI
SE-100 44 Stockholm, Sweden
URL: www.kth.se/sci
Contents

1 Introduction .......................... 1
  1.1 Graph definitions .................. 1
  1.2 Probability definitions .......... 2
  1.3 Bayesian networks ................. 2

2 Bayesian learning of PDAGS ......... 8

3 Summation of a clustering algorithm developed by Dahl 10

4 The approximation of the algorithm developed by Dahl 12

5 Results .............................. 14
  5.1 Comparing the results for heart disease data .................. 14
    5.1.1 Description of the heart disease data .................. 14
    5.1.2 Results of the heart disease data .................. 14
  5.2 Comparing the results for simulated data .................. 17
    5.2.1 Description of the simulated data .................. 17
    5.2.2 Results of the simulated data .................. 19

6 Conclusion .......................... 21

A Theory ................................ 22

B Generating vertex labeled DAGs .... 24
Abstract

Partition directed acyclic graphs (PDAGs) is a model whereby the conditional probability tables (CPTs) are partitioned into parts with equal probability. In this way, the number of parameters that need to be learned can be significantly reduced so that some problems become more computationally feasible. PDAGs have been shown to be connected to labeled DAGs (LDAGs) and the connection is summarized here. Furthermore, a clustering algorithm is compared to an exact algorithm for determining a PDAG. To evaluate the algorithm, we use it on simulated data where the expected result is known.
Sammanfattning

Partitionerade riktade acykliska grafer (engelska: PDAGs) är en modell där tabeller över betingade sannolikheter partitioneras i delar med lika sannolikhet. Detta gör att antalet parametrar som ska bestämmas kan reduceras, vilket i sin tur gör problemet beräkningsmässigt enklare. Ett känt samband mellan PDAGs och betecknade riktade acykliska grafer (engelska: LDAGs) sammanfattas här. Sedan jämförs en klustringssalgoritm med en algoritm som exakt bestämmer en PDAG. Klustringsalgoritmens pålitlighet kollas genom att använda den på simulerad data där det förväntade resultatet är känt.
Chapter 1

Introduction

In [5], something called partition directed acyclic graphs (PDAG) was introduced, see Definition 1.3.5, which is an attempt to model features not detected by regular DAGs. An exact recursive algorithm for Bayesian learning of PDAGs was also developed. The problem is that the time complexity of this algorithm is such that only graphs where the vertices have at most 4 parents can be handled. We will develop an approximate algorithm that significantly improves the time complexity and is feasible for vertices with greater than 4 parents. We begin by explaining some theory and definitions that will be needed to state our problem.

1.1 Graph definitions

Definition 1.1.1 (Graph, [6]). A graph $G$ is a pair $(V,E)$ where $V = \{1, \ldots, d\}$ and $E \subseteq V^2$. The elements of $V$ are called vertices and the elements of $E$ are called edges. The graph is simple if $(v,v) \not\in E$ for all $v \in V$ and there is at most one edge between each pair of vertices. An edge $(k,l) \in E$ is directed if $(l,k) \not\in E$. If $(l,k) \in E$, the edge is undirected. The graph is (un)directed if each edge is (un)directed.

Remark. In this report, every graph will be assumed to be simple.

Definition 1.1.2 (Path, Cycle, [6]). Let $G = (V,E)$ be a graph. A path of length $m$ from $\alpha \in V$ to $\beta \in V$ is sequence of distinct vertices $(v_0, \ldots, v_m)$ such that $\alpha = v_0$, $\beta = v_m$ and $(v_i, v_{i+1}) \in E$ for all $i = 0, \ldots, m - 1$. An $m$-cycle is a sequence of vertices $(v_0, \ldots, v_{m-1}, v_0)$ such that $(v_0, \ldots, v_{m-1})$ is path of length $m - 1$ and $(v_{m-1}, v_0) \in E$.

Definition 1.1.3 (Connected). Let $G$ be a graph. Then $G$ is connected if there is a path between each pair of vertices in $G$.

Definition 1.1.4 (DAG, [6]). A Direct Acyclic Graph (DAG) is a directed graph with no $m$-cycles for any $m \geq 1$.

Definition 1.1.5 ([6]). Let $G = (V,E)$ be a directed graph and take any $v \in V$. The parents of $v$, denoted $\Pi_{v}$, is the set $\Pi_{v} = \{\beta \in V \mid (\beta, \alpha) \in E\}$.

Example 1.1.1. In Figure 1.1, the parents of the vertex $v$ are $A$ and $C$. 
1.2 Probability definitions

Let $X = (X_1, \ldots, X_d)$ be a vector of random variables. Then for the random variable $X_j$, $1 \leq j \leq d$, $\mathcal{X}_j$ will be used to denote the state space of $X_j$. The elements of $\mathcal{X}_j$ will be denoted using lower case letters $x_j$. Finally, for a subset $V \subseteq \{1, \ldots, d\}$, $p_{X_V} : \mathcal{X} \rightarrow [0, 1]$ is a probability function over the product space $\mathcal{X} = \times_{i \in V} \mathcal{X}_i$.

**Definition 1.2.1 (Independence, [6]).** Two random vectors $X_1 = (X_1, \ldots, X_n)$ and $Y = (Y_1, \ldots, Y_m)$ are independent, written $X \perp Y$, if the joint probability distribution $(X, Y) = (X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ factorizes as

$$p_{X,Y}(x,y) = p_X(x)p_Y(y) \quad \forall (x,y) \in \mathcal{X} \times \mathcal{Y},$$

where $\mathcal{Y}$ is the state space of $Y$. $X$ and $Y$ are conditionally independent given a random vector $Z$, written $X \perp Y | Z$, if we can write

$$p_{X|Y,Z}(x|y,z) = p_{X|Z}(x|z)$$

for all $(x,y,z) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ such that $p_{Y|Z}(y|z), p_Z(z) > 0$. Here, $\mathcal{Z}$ is the state space of $Z$.

**Remark.** We will for the most part write the function $p_X$ as $p(X)$. This means that the independence condition could be written as $p(X, Y) = p(X)p(Y)$.

1.3 Bayesian networks

Using the identity $p(X, Y) = p(X|Y)p(Y)$ repeatedly, we can write

$$p(X_1, \ldots, X_d) = p(X_{\sigma(1)}) \prod_{j=2}^d p(X_{\sigma(j)}|X_{\sigma(1)}, \ldots, X_{\sigma(j-1)})$$  \hspace{1cm} (1.1)$$

for each permutation $\sigma$ of $\{1, \ldots, d\}$. However, for many applications, we don’t need full dependence on $X_{\sigma(1)}, \ldots, X_{\sigma(j-1)}$. Instead, we would like to get the factorization

$$p(X_1, \ldots, X_d) = p(X_{\sigma(1)}) \prod_{j=2}^d p(X_{\sigma(j)}|\Pi_j^\sigma),$$

where $\Pi_j^\sigma \subseteq \{X_{\sigma(1)}, \ldots, X_{\sigma(j-1)}\}, 2 \leq j \leq d$. Furthermore, we want $\Pi_j^\sigma$ to be as small as possible in the sense that we cannot describe $p(X_{\sigma(j)}|X_{\sigma(1)}, \ldots, X_{\sigma(j-1)})$ with a strictly smaller subset of $\Pi_j^\sigma$. Using the definition of conditional independence, we get the following definition of a Bayesian network.

**Definition 1.3.1 (Bayesian network, [6]).** A Bayesian network is a factorization of the probability distribution

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

such that
1. $\Pi_1^\sigma = \emptyset$.
2. $\Pi_j^\sigma \subseteq \{X_{\sigma(1)}, \ldots, X_{\sigma(j-1)}\}$.
3. $X_{\sigma(j)} \perp \{X_{\sigma(1)}, \ldots, X_{\sigma(j-1)}\} \setminus \Pi_j^\sigma | \Pi_j^\sigma$.
4. For any strict subset $\Theta_j \subset \Pi_j^\sigma$,
   $$X_{\sigma(j)} \not\perp \{X_{\sigma(1)}, \ldots, X_{\sigma(j-1)}\} \setminus \Theta_j | \Theta_j.$$

Here, we have implicitly introduced the notation $p(X|\emptyset) = p(X)$, where $\emptyset$ is the empty set. If condition 4 is not satisfied, we just say that we have a factorization over the probability distribution.

**Remark.** Note here that it is the definition of conditional independence that makes it possible to write $p(X_{\sigma(j)}|X_{\sigma(1)}, \ldots, X_{\sigma(j-1)}) = p(X_{\sigma(j)}|\Pi_j)$.

A nice way to represent a Bayesian network is through a DAG as follows.

**Definition 1.3.2** (Factorization along a DAG). Suppose $V = \{1, \ldots, d\}$ are the vertices of a DAG $G$ and let $X_1, \ldots, X_d$ be random variables. Then we say that we have a factorization along a DAG if there is a permutation $\sigma$ of $\{1, \ldots, d\}$ such that conditions 1-3 of Definition 1.3.1 are satisfied and each $\Pi_j$ are the parents of vertex $j$. If additionally condition 4 is true, we have a factorization over a Bayesian network.

**Example 1.3.1.** The DAG in Figure 1.1 gives the factorization

$$p(X_v, X_A, X_C) = p(X_v|X_A, X_C)p(X_C|X_A)P(X_A).$$

Factorization along a DAG is thus a good way to describe problems where the dependency structure in equation 1.1 can be reduced. However, for many applications this is still too much so we need some more structure. One way to refine the factorization is to introduce local structure, in our case something called local context-specific independence (CSI).

**Definition 1.3.3** (Local Context-Specific Independence, [5]). Let $v \in V$ be a node and let $A$ and $C$ form a partition of its parents $\Pi_v$. For the definition of a partition, see appendix A. Additionally, for $B \subseteq V$, let $\mathcal{X}_B$ denote the outcome space of $X_B$. Then we say that $X_v$ is locally contextually independent of $X_A$ in the context $X_C = x_c$ if

$$p(x_v | x_A, x_C) = p(x_v | x_C),$$

holds for all $(x_v, x_A) \in \mathcal{X}_v \times \mathcal{X}_A$ whenever $p(x_A, x_C) > 0$. This is denoted by

$$X_v \perp X_A | x_C.$$

Note the similarity to Definition 1.2.1. The difference here is that we do not require it to be true for all $x_C \in \mathcal{X}_C$ but only for a specific $x_C$. Hence, the name context is appropriate.
Example 1.3.2. Consider Figure 1.1 and assume that $X_v, X_A$ and $X_C$ are binary random variables.

Then $X_v | X_A, X_C$ is also a binary random variable. Now, take $X_A$ and $X_C$ so that when $X_C = x_C = 0$, then $p(x_v | X_A = 0, x_C = 0) = p(x_v | X_A = 1, x_C = 0)$. Then $X_v \perp X_A | x_C$.

We noted that DAGs are a convenient way to represent Bayesian networks and these same DAGs can now be extended to carry the information of CSIs by labels.

Definition 1.3.4 (Labeled Directed Acyclic Graph, [5]). Let $G = (V, E)$ be a DAG for the random variables $\{X_1, \ldots, X_d\}$. For all $(u, v) \in E$, let $L_{(u,v)} = \Pi_v \{u\}$. A label on an edge $(u, v) \in E$ is defined as the set

$$L_{(u,v)} = \{x_{L_{(u,v)}} \in X_{L_{(u,v)}} : X_v \perp X_u | x_{L_{(u,v)}}\}.$$

An LDAG $G_L = (V, E, L_E)$ is a DAG to which the label set $L_E = \{L_{(u,v)} : L_{(u,v)} \neq \emptyset\}_{(u,v) \in E}$ has been imposed.

Example 1.3.3. In Figure 1.5, we show a compact way of collecting the labels in a DAG.

To see that these concepts actually have real world applications, consider the following example from [2].

Example 1.3.4 ([2]). A guard of a secured building expects three types of persons to approach the building’s entrance: workers in the building, approved visitors, and spies. As a person approaches the building, the guard can note gender and whether or not the person wears a badge. Spies are mostly men. Spies always wear badges in an attempt to fool the guard. Visitors do not wear badges because they do not have one. Female workers tend to wear badges more often than do male workers. The task of the guard is to identify the type of person approaching the building.

Now, letting $h$ denote the person approaching the building, $g$ the gender and $b$ whether the person wears a badge, we can represent the problem by the Bayesian network seen in Figure 1.2.

Assume now that we know that the person approaching is a spy or a visitor. Then note that $b \perp g | h = \text{spy}$ since a spy always wears a badge. Similarly, since a visitor
never wears a badge, \( b \perp g|h = \text{visitor} \). The Bayesian network does not include this information. Traditionally, this information was shown in something called Bayesian multinets, as in Figure 1.3.

![Bayesian network](image)

(a) Bayesian network when \( h \) is a spy or a visitor

![Bayesian network](image)

(b) Bayesian network when \( h \) is a worker

Figure 1.3: Representation as a Bayesian multinet

However, note that the same information can be represented with the label 

\[ \mathcal{L}_{(g,b)} = \{\text{spy, visitor}\} \]

so we can reduce the three graphs into a single one, shown in 1.4.

![Bayesian multinet](image)

Figure 1.4: Transformation of a Bayesian multinet into an LDAG

Another advantage of this representation is in a reduction of the number of free parameters. Observe that Figure 1.2 gives the probability distribution 

\[ p(h, b, g) = p(h)p(b|h, g)p(g|h). \]

Then to determine \( p(h, b, g) \) we need to determine its components. To determine \( p(h) \), we need 2 parameters, as \( h \) is a random variable taking three values and their sum is 1. Next, \( p(b|h, g) \) gives in total \( 3 \cdot 2 = 6 \) parameters as \( h \) can take 3 values and \( g \) two values. Finally, \( p(g|h) \) gives 3 parameters. In total, we thus have 11 parameters to learn. Now, the contextual independence implies that some of these distributions are equal and so the total number of parameters that need to be learned can be reduced. Specifically, we know that 

\[ p(b|g = \text{man}, h = \text{spy}) = p(b|g = \text{female}, h = \text{spy}) \]

and 

\[ p(b|g = \text{man}, h = \text{visitor}) = p(b|g = \text{female}, h = \text{visitor}). \]

Thus, we only need to learn 9 parameters. This might seem like a small gain, but in other applications the gain can be significant.
Definition 1.3.5 (Partition Directed Acyclic Graph, [5]). Let $G = (V, E)$ be a DAG for the random variables $\{X_1, \ldots, X_d\}$. For all $v \in V$, let $S_v$ be a partition of the set $X_{v,1} \cup \cdots \cup X_{v,k}$ into $k$ classes $s_{v,1}, \ldots, s_{v,k}$ such that the conditional probabilities $p(x_v|x_{v,1})$ are equal for all $x_{v,1} \in s_{v,c}, c = 1, \ldots, k$ and unrestricted otherwise. The collection of partitions $\mathcal{S} = \{S_v \mid v \in V\}$ and the corresponding probabilities for each class in each partition defines a Partition DAG (PDAG), denoted by $G_\mathcal{S}$.

Remark. When the context is clear, we will write $s_{v,c}$ simply as $s_c$ for $c = 1, \ldots, k$.

Example 1.3.5. Consider again Figure 1.1, but where we have added the label $L(A, v) = \{0\}$ to the edge $(A, v)$, see Figure 1.5.

Since by the definition of CSI, we have that
\[
p(X_v | X_A = 0, X_C = 0) = p(X_v | X_A = 1, X_C = 0) = p(X_v | X_C = 0),
\]
the LDAG gives rise to a conditional probability table (CPT) shown in Table 1.1.

| $X_{\Pi_v}$ | $p(X_v = 1 | X_{\Pi_v})$ |
|-------------|-------------------------|
| $(X_A, X_C) = (0, 0)$ | $p_1$ |
| $(X_A, X_C) = (1, 0)$ | $p_1$ |
| $(X_A, X_C) = (0, 1)$ | $p_2$ |
| $(X_A, X_C) = (1, 1)$ | $p_3$ |

Table 1.1: CPT obtained from the LDAG in Figure 1.5.

From the CPT, we see that we get a partition $S_v$ of $X_{\Pi_v}$ into $k = 3$ classes such that $p(x_v | x_{\Pi_v})$ are equal for all $x_{\Pi_v}$ in each class. Thus, this example illustrates the general fact that every LDAG correspond to a PDAG. However, the converse is only true under some conditions that we will now elaborate on. First we need the notion of Hamming distance and a class connection graph.

Definition 1.3.6 (Hamming distance). Let $F^n$ be the set of binary vectors of length $n$. Then for $x, y \in F^n$, the Hamming distance $d(x, y)$ is the number of positions at which the coefficients are different.

Example 1.3.6. $d(110, 010) = 1$, $d(1111, 0000) = 4$.

Definition 1.3.7 ([5]). A class connection graph $G_c = (N, E)$ for $v \in V$ is an undirected graph over the class elements $s_c = \{x_{\Pi_v}^{(i)}\}_{i=1}^q$ such that $q = |s_c|$ and

\[
N = \{1, \ldots, q\} \text{ and } \{l, r\} \in E \text{ if } d(x_{\Pi_v}^{(l)}, x_{\Pi_v}^{(r)}) = 1,
\]
where $d(\cdot, \cdot)$ is the Hamming distance between the two binary vectors.
Example 1.3.7. Using Table 1.1, we obtain the graph shown in Figure 1.6.

Definition 1.3.8 ([5]). A partition of a CPT is CSI-consistent if it can be constructed according to a collection of local CSIs.

Now, we get to the main result which gives a necessary and sufficient condition for when a PDAG is also an LDAG.

Theorem 1 ([5]). A partition $S_v = \{s_{v,1}, \ldots, s_{v,k}\}$ is CSI-consistent if and only if the corresponding class connection graphs $G_1, \ldots, G_k$ are connected.

Proof. First we show that CSI-consistent implies that the class connection graphs are connected. Take any class $s_c$ in the partition so that $p(x_v \mid x_{\Pi_v}^{(1)}) = p(x_v \mid x_{\Pi_v}^{(m)})$ for all $x_{\Pi_v}^{(1)}, x_{\Pi_v}^{(m)} \in s_c$. Take any two $x_{\Pi_v}^{(1)}, x_{\Pi_v}^{(m)}$ in $s_c$. If the equality $p(x_v \mid x_{\Pi_v}^{(1)}) = p(x_v \mid x_{\Pi_v}^{(m)})$ can be explained by one CSI, we can find a partition $A, C$ of $\Pi_v$ as in Definition 1.3.3 such that

$$p(x_v \mid x_{\Pi_v}^{(1)}) = p(x_v \mid x_A', x_C) = p(x_v \mid x_C) = p(x_v \mid x_A'', x_C) = p(x_v \mid x_{\Pi_v}^{(m)})$$

Since this equality is true for all $x_A \in X_A$, we can find, by the nature of $s_c$, a sequence $(x_{\Pi_v}^{(i)})_{i=1}^m$ in $s_c$ such that $d(x_{\Pi_v}^{(i)}, x_{\Pi_v}^{(i+1)}) = 1$. Hence, every pair corresponding to a local CSI are connected in $G_c$. If the equality could not be explained by one CSI, it could be explained by a collection of local CSIs. Hence, in this collection, we can find a local CSI giving us a sequence $(x_{\Pi_v}^{(i)})_{i=1}^m$ as above. Continuing in this way taking some element in this sequence and another suitable local CSI in the collection, we will eventually get to $x_{\Pi_v}^{(m)}$. Thus, there is a path to every pair in $s_c$ and so $G_c$ is connected.

Conversely, assume that $G_c$ is connected. We want to show that the partition is CSI-consistent. Take any elements $x_{\Pi_v}^{(1)}, x_{\Pi_v}^{(m)} \in s_c$. Since $G_c$ connected, we can find a sequence $(x_{\Pi_v}^{(i)})_{i=2}^{m-1}$ such that $d(x_{\Pi_v}^{(i)}, x_{\Pi_v}^{(i+1)}) = 1$ for $i = 1, \ldots, m - 1$. But the situation $d(x_{\Pi_v}^{(i)}, x_{\Pi_v}^{(i+1)}) = 1$ can be represented by the label $L_{(u,v)}$ where $u$ is the position that differs. Hence, the partition is CSI-consistent. \qed
Chapter 2
Bayesian learning of PDAGS

Suppose we have a set of training data \( X = \{x_i\}_{i=1}^n \) consisting of \( n \) independent observational data \( x_i = (x_{i1}, \ldots, x_{id}) \) of the variables \( X_1, \ldots, X_d \). Here it is assumed that the data contains no missing values. Next, let

\[
\theta_{vc} = p(X_v = 1 \mid X_{\Pi v} \in s_{v,c}),
\]

where \( \{s_{v,1}, \ldots, s_{v,k_v}\} \) is a partition of \( S_v \). That is, \( \theta_{vc} \) is the probability of \( X_v = 1 \) given that \( X_{\Pi v} \) is assigned a value in class \( c \) of the partition \( S_v \). Now, given a PDAG \( G_S \), we let \( \theta_{G_S} \) denote collectively all unknown parameters in the CPTs

\[
\theta_{G_S} = \{\theta_{vc} \mid v \in \{1, \ldots, d\}, c \in \{1, \ldots, k_v\}\}.
\]

Finally, let \( n(v, c, 1) \) denote the total count of observations with \( x_v = 1 \) and \( x_{\Pi v} \in s_{v,c} \), i.e.,

\[
n(v, c, 1) = |\{x_i \in X \mid x_{iv} = 1 \text{ and } x_{(i)_{\Pi v}} \in s_c\}|
\]

Similarly, \( n(v, c, 0) \) denotes the total count of observations with \( x_{iv} = 0 \) and \( x_{(i)_{\Pi v}} \in s_{v,c} \).

The likelihood of a PDAG can then be written as

\[
p(X \mid \theta_{G_S}, G_S) = \prod_{v=1}^d \prod_{c=1}^{k_v} \theta_{vc}^{n(v, c, 1)} (1 - \theta_{vc})^{n(v, c, 0)}.
\]

In order to get a marginal likelihood that can be calculated analytically, we take a prior on \( \theta_{G_S} \) that factorizes over the DAG. In particular, we take the beta prior defined by

\[
p(\theta_{vc} \mid G_S) = \frac{\theta_{vc}^{\alpha_1 - 1} (1 - \theta_{vc})^{\alpha_0 - 1}}{\beta(\alpha_0, \alpha_1)},
\]

(2.1)

where \( \beta(\alpha_0, \alpha_1) = \frac{\Gamma(\alpha_0)\Gamma(\alpha_1)}{\Gamma(\alpha_0 + \alpha_1)} \) is the beta function expressed using the Gamma function \( \Gamma(x) \). \( \alpha_0 \) and \( \alpha_1 \) are the hyperparameters and they will in the experimentation be set to 1 as in [5]. Equation 2.1 now implies that

\[
p(\theta_{G_S} \mid G_S) = \prod_{v=1}^d \prod_{c=1}^{k_v} \frac{\theta_{vc}^{\alpha_1 - 1} (1 - \theta_{vc})^{\alpha_0 - 1}}{\beta(\alpha_0, \alpha_1)}.
\]

Since it is a density function, it also follows that

\[
\int_0^1 \frac{\theta_{vc}^{\alpha_1 - 1} (1 - \theta_{vc})^{\alpha_0 - 1}}{\beta(\alpha_0, \alpha_1)} \, d\theta_{vc} = 1 \implies \beta(\alpha_0, \alpha_1) = \int_0^1 \theta_{vc}^{\alpha_1 - 1} (1 - \theta_{vc})^{\alpha_0 - 1} \, d\theta_{vc}.
\]

(2.2)

8
Now, we get that

\[
p(X \mid G_S) = \int p(X, \theta_{G_S} \mid G_S)p(\theta_{G_S} \mid G_S) \, d\theta_{G_S} = \\
= \prod_{v=1}^{d} \prod_{c=1}^{k_v} \int_0^1 \theta_{vc}^{n(v,c,1)} (1 - \theta_{vc})^{n(v,c,0)} \frac{\theta_{vc}^{\alpha_1 - 1} (1 - \theta_{vc})^{\alpha_0 - 1}}{\beta(\alpha_0, \alpha_1)} \, d\theta_{vc} = \\
= \prod_{v=1}^{d} \prod_{c=1}^{k_v} \frac{1}{\beta(\alpha_0, \alpha_1)} \int_0^1 \theta_{vc}^{n(v,c,1)+\alpha_1 - 1} (1 - \theta_{vc})^{n(v,c,0)+\alpha_0 - 1} \, d\theta_{vc} = \\
= \prod_{v=1}^{d} \prod_{c=1}^{k_v} \frac{\beta(n(v, c, 0) + \alpha_0, n(v, c, 1) + \alpha_1)}{\beta(\alpha_0, \alpha_1)},
\]

where we in the last step used equation 2.2. Assuming a product prior on \( G_S \), we can consider each vertex separately, and write

\[
p(X \mid S_v) = \prod_{c=1}^{k_v} \frac{\beta(n(v, c, 0) + \alpha_0, n(v, c, 1) + \alpha_1)}{\beta(\alpha_0, \alpha_1)}.
\]

The aim is now to maximize \( \log p(G_S \mid X) \propto \log p(X \mid G_S) + \log p(G_S) \). For each term in the product, we assume that \( p(S_v) = |\mathcal{X}_{|v|} - 1 S(\mathcal{X}_{|v|}, k_v)^{-1} \), where \( S(\mathcal{X}_{|v|}, k_v)^{-1} \) is the Stirling number of the second kind. This means that we have a uniform distribution on the number of classes \( k_v \) in \( S_v \). Then we get

\[
p(G_S) = \prod_{v=1}^{d} |\mathcal{X}_{|v|} - 1 S(\mathcal{X}_{|v|}, k_v)^{-1}.
\]

The strategy is now to maximize \( p(S_v \mid X) = \frac{p(S_v)p(X \mid S_v)}{p(X)} \propto p(S_v)p(X \mid S_v) \) for each vertex \( v \in V \) as described in Chapter 3.
Chapter 3

Summation of a clustering algorithm developed by Dahl

**Definition 3.0.1.** A clustering of \( n \) objects can be represented by a set partition \( \pi = \{S_1, \ldots, S_q\} \) of a set \( S_0 = \{1, \ldots, n\} \) having the following properties:

1. \( S_i \neq \emptyset \) for \( i = 1, \ldots, q \).
2. \( S_i \cap S_j = \emptyset \) for all \( i, j = 1, \ldots, q \) such that \( i \neq j \).
3. \( \bigcup_{j=1}^{q} S_j = S_0 \).

In our case, it will be that \( n \) is the number of parent configurations for a vertex \( v \) and elements in the same partition \( S_i \) correspond to parent configurations with the same probability. A possible approach to finding the optimal partition is then to go through each possible partition and that is what is done in [5] using a recursive algorithm. But since the number of possible partitions are given by the Bell number \( B(n) \), which grows very fast, this is only possible for small values of \( n \). In fact, using only 4 parent nodes, the recursive algorithm in [5] took about a minute. The algorithm developed in [1] only requires \( \frac{n(n+1)}{2} \) density evaluations and for many kinds of densities it gives the optimal partition.

We now summarize the concepts by Dahl [1]. We want to find a probability model for \( y = \{y_i \mid i \in S_0\} \) that are parameterized by a set partition \( \pi \). Assuming that items in different partition sets are independent, we can write \( p(y|\pi) = \prod_{j=1}^{q} f(y_{S_j}) \), where \( y_{S_j} = (y_i)_{i \in S_j} \). \( f(y_{S}) \) is the component likelihood.

Similarly, we assume the following form on the prior \( p(\pi) \)

\[
p(\pi) \propto \prod_{j=1}^{q} h(S_j), \quad h(S_j) \geq 0.
\]

\( h(S) \) is known as the component cohesion. This means that we can write the posterior \( p(\pi \mid y) \) as

\[
p(\pi \mid y) \propto p(y \mid \pi)p(\pi) \propto \prod_{j=1}^{q} f(y_{S_j})h(S_j).
\]

The algorithm for finding the MAP clustering requires the following two conditions.
1. Here, we say that $S_i$ and $S_j$ overlap if $[\min(S_i), \max(S_i)] \cap [\min(S_j), \max(S_j)] \neq \emptyset$. If components $S_i$ and $S_j$ overlap, then there exist two other components $S_i^*$ and $S_j^*$, representing a reallocation of the items of $S_i$ and $S_j$ in which the number of items is preserved respectively, such that:

$$f(y_{S_i})f(y_{S_j}) \leq f(y_{S_i^*})f(y_{S_j^*}).$$

It should be noted that in [1], the data is sorted out so that $y_i \leq y_{i+1}$, $i = 1, \ldots, n - 1$. However, since we have a different likelihood, we will sort the data in another way. This will be explained in Chapter 4.

2. The cohesion $h(S)$ depends, at most, only on the number of items contained in $S$.

**Definition 3.0.2 ([1]).** A partition $\pi_{MAP}^k$ of $\{1, \ldots, k\}$ is said to be an *incomplete model partition* for $p(\pi | y) \propto \prod_{j=1}^q f(y_{S_j})h(S_j)$, the posterior for the partition of $\{1, \ldots, n\}$, if:

$$p(\pi = \pi^k \cup \{S_q\} | y) \leq p(\pi = \pi_{MAP}^k) \cup \{S_q\} | y)$$

for all partitions $\pi^k$ of $\{1, \ldots, k\}$, where $S_q = \{k+1, \ldots, n\}$.

**Theorem 2 ([1]).** If conditions 1 and 2 hold, then $\pi_{MAP}^k$ can be found among the following $k$ candidates:

$$\{\{1, \ldots, k\}\}$$

$$\pi_{MAP}^1 \cup \{\{2, \ldots, k\}\}$$

$$\vdots$$

$$\pi_{MAP}^{k-2} \cup \{\{k-1, k\}\}$$

$$\pi_{MAP}^{k-1} \cup \{\{k\}\}$$

$$\pi_{MAP}^k \cup \{\{k, k+1, \ldots, n\}\}$$

**Proof.** First note that $\{k, k+1, \ldots, n\}$ is not a valid component by Definition 3.0.2. This means that $k$ must be in a component of size $m$, where $1 \leq k \leq m$. Also, by condition 1 and 2 in Chapter 3, the optimal partition is non-overlapping. Hence, the component containing $k$ must be $\{k - m + 1, \ldots, k\}$. By Definition 3.0.2, the remaining $1, 2, \ldots, k - m$ must be partitioned as $\pi_{MAP}^{k-m}$. Thus, $\pi_{MAP}^k = \pi_{MAP}^{k-m} \cup \{k - m + 1, \ldots, k\}$.

By noting that $\pi_{MAP}^n = \pi_{MAP}$, the optimal partition can now be found by starting at $\pi_{MAP}^1 = \{\{1\}\}$ and then determine $\pi_{MAP}^k$, $2 \leq k \leq n$, from the list above.
Chapter 4

The approximation of the algorithm developed by Dahl

In our case, the sufficient statistics for each vertex will be \( y_v = \{(n_i(v,0), n_i(v,1)) | i \in S_0 \} \), where \( S_0 = \{1, \ldots, |X_{\Pi_v}| \} \). \( n_i(v,k) \) is the number of observations corresponding to \( i \) with \( x_v = k, k = 0, 1 \). This is a problem, because the algorithm by Dahl [1] requires univariate statistics. However, using the bijective Cantor pairing function \( f : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \) defined by \( f(n_0, n_1) = \frac{1}{2}(n_0 + n_1)(n_0 + n_1 + 1) + n_1 \), we can transform the sufficient statistic into a univariate one. Unfortunately, the data is still inherently two-dimensional since when we calculate the likelihood, the univariate data needs to be transformed into its original form. In fact, for our data, there are partitions that do not satisfy Condition 1.

However, the following heuristic reasoning might imply that for a certain sorting of the data, we get a good approximation. Suppose we know the number of clusters \( k \) and the probabilities \( p_1, \ldots, p_n \), some of which are equal, corresponding to each parent configuration. Then, by the law of large numbers,

\[
\hat{p}_i := \frac{n_i(v,1)}{n_i(v,0) + n_i(v,1)} \xrightarrow{P} E(X_v | X_{\Pi_v} = \pi_i) = p(X_v = 1 | X_{\Pi_v} = \pi_i) = p_i
\]

as \( n_i(v,0) + n_i(v,1) \to \infty \). Here, \( \pi_i \) is the parent configuration corresponding to \( i \). This means that if we sort the data so that \( \hat{p}_1 \leq \cdots \leq \hat{p}_n \) and if \( n_i(v,0) + n_i(v,1) \) is sufficiently large for each \( i \), the optimal partition will be

\[
\{\pi_1, \pi_2, \ldots, \pi_{i-1}\}, \{\pi_i, \ldots, \pi_{i_2-1}\}, \ldots, \{\pi_{i_k-1}, \ldots, \pi_n\}
\]

That is, the optimal partition is non-overlapping and so the same algorithm can be used. The algorithm is as follows:
**Data:** \( X = \{ x_i \}_{i=1}^n, x_i = (x_{i1}, \ldots, x_{id}) \), \( \Pi_v = (p_1, \ldots, p_m), m \leq d - 1 \)

**Result:** Optimal partition \( S_v \) of \( \mathcal{X}_v \)

\( y_v \leftarrow X \) Create the univariate sufficient statistics;

Sort \( y_v \):

\( N \leftarrow 2^m \) The number of parent configurations;

MAPs[0] ← [] Optimal incomplete model partitions for \( k = 1, \ldots, N \);

**for** \( k = 1 : (N + 1) \) **do**

\( S_q \leftarrow [[k + 1, \ldots, N]]; \)

**for** \( l = 1 : (k + 1) \) **do**

\( \text{candidate} \leftarrow [[l, l + 1, \ldots, k]]; \)

\( S_v \leftarrow \text{MAPs}[l - 1] + \text{candidate} + S_q; \)

Calculate posterior \( p(y_v | S_v); \)

If \( p(S_v | y_v) \) greater than previous maximal posterior, set \( S_{opt} \leftarrow S_v; \)

end

MAPs[k] ← \( S_{opt} - S_q; \)

end
Chapter 5

Results

5.1 Comparing the results for heart disease data

5.1.1 Description of the heart disease data

In [5], an exact method for finding the optimal partition for heart disease data is applied. The data([7]) consists of 1841 observations on 6 binary variables as shown in Table 5.1.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Outcomes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>Smoking</td>
</tr>
<tr>
<td>$X_2$</td>
<td>Strenuous mental work</td>
</tr>
<tr>
<td>$X_3$</td>
<td>Strenuous physical work</td>
</tr>
<tr>
<td>$X_4$</td>
<td>Systolic blood pressure</td>
</tr>
<tr>
<td>$X_5$</td>
<td>Ratio of $\beta$ and $\alpha$ lipoproteins</td>
</tr>
<tr>
<td>$X_6$</td>
<td>Family anamnesis of CHD</td>
</tr>
</tbody>
</table>

Table 5.1: Description of the variables in coronary heart disease (CHD) data.

In determining the optimal DAG, each vertex was allowed to have up to four parents and some partial ordering was imposed to rule out certain vertices from having certain parents. This implied that vertices 1, 2, 3, 4, 5, 6 had each 0, 1, 2, 4, 4, 0 possible parents, respectively. The DAG that was found can be seen in Figure 5.1. A more in depth analysis was then made for vertex 4, and for comparison, we will do the same analysis using the algorithm developed by Dahl.

5.1.2 Results of the heart disease data

For each of the 5 possible parent candidates of vertex 4, the log marginal likelihoods of the MAP partitions relative to trivial partition were calculated. The results are summarized in 5.2.
Table 5.2: The log marginal likelihood relative to the trivial partition for the MAP partition of $X_\Pi$ for different candidate sets of parents using an exact and an approximate method.

<table>
<thead>
<tr>
<th>Parents</th>
<th>Log marginal likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>2356</td>
<td>+0.0</td>
</tr>
<tr>
<td>1356</td>
<td>+13.5</td>
</tr>
<tr>
<td>1256</td>
<td>+13.4</td>
</tr>
<tr>
<td>1236</td>
<td>+0.0</td>
</tr>
<tr>
<td>1235</td>
<td>+10.5</td>
</tr>
</tbody>
</table>

(a) Exact method developed in [5]

<table>
<thead>
<tr>
<th>Parents</th>
<th>Log marginal likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>2356</td>
<td>+17.0</td>
</tr>
<tr>
<td>1356</td>
<td>+13.5</td>
</tr>
<tr>
<td>1256</td>
<td>+28.3</td>
</tr>
<tr>
<td>1236</td>
<td>+19.6</td>
</tr>
<tr>
<td>1235</td>
<td>+47.6</td>
</tr>
</tbody>
</table>

(b) Approximate method developed by Dahl [1]

It seems strange that an approximate method gives better results. The reason for that is that the values for $X_2$ are not the same in our data, see Table 5.3.

Table 5.3: Illustration of the data difference in the $X_2$ variable.

<table>
<thead>
<tr>
<th>Parent values</th>
<th>#0</th>
<th>#1</th>
<th>average $X_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0011</td>
<td>16</td>
<td>9</td>
<td>0.3600</td>
</tr>
<tr>
<td>1000</td>
<td>185</td>
<td>79</td>
<td>0.2992</td>
</tr>
<tr>
<td>1100</td>
<td>90</td>
<td>42</td>
<td>0.3182</td>
</tr>
<tr>
<td>class 1 total</td>
<td>291</td>
<td>130</td>
<td>0.3088</td>
</tr>
<tr>
<td>class 2 total</td>
<td>763</td>
<td>657</td>
<td>0.4627</td>
</tr>
</tbody>
</table>

(a) MAP partition of parent candidates 1256 as determined in [5].

<table>
<thead>
<tr>
<th>Parent values</th>
<th>#0</th>
<th>#1</th>
<th>average $X_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0011</td>
<td>16</td>
<td>9</td>
<td>0.3600</td>
</tr>
<tr>
<td>1000</td>
<td>252</td>
<td>79</td>
<td>0.2387</td>
</tr>
<tr>
<td>1100</td>
<td>23</td>
<td>42</td>
<td>0.6462</td>
</tr>
<tr>
<td>class 1 total</td>
<td>291</td>
<td>130</td>
<td>0.3088</td>
</tr>
<tr>
<td>class 2 total</td>
<td>763</td>
<td>657</td>
<td>0.4627</td>
</tr>
</tbody>
</table>

(b) MAP partition of parent candidates 1256 as determined in [5] using the data from [7].

What is interesting is that the total sum of each class coincides, so that the densities coincide in this particular partition. Also, using our data, the statement “if $(X_1, X_5, X_6) = (1, 0, 0)$, then the probability $X_4 = 1$ is much lower than otherwise”, cannot be motivated.

In any case, this means that we only can do a comparison for the parent candidate 1356. Here, the results coincide.

Next, the optimal parent set with only 3 nodes using the parent candidate 156 was found in [5]. Again, the same MAP partition was found, here seen in Table 5.4.

Table 5.4: MAP partition of parent candidates 156. #0 and #1 denotes the total count of $X_4 = 0$ and $X_4 = 1$, respectively, for each parent value.

<table>
<thead>
<tr>
<th>Parent values</th>
<th>#0</th>
<th>#1</th>
<th>average $X_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>275</td>
<td>121</td>
<td>0.3056</td>
</tr>
<tr>
<td>class 1 total</td>
<td>275</td>
<td>121</td>
<td>0.3056</td>
</tr>
<tr>
<td>class 2 total</td>
<td>779</td>
<td>666</td>
<td>0.4609</td>
</tr>
</tbody>
</table>
Figure 5.1: DAG structure determined in [5].

Now, the approximate algorithm is sufficiently fast to go through all DAGs with 6 vertices. For a discussion on how to generate all DAGs, see Appendix B. The result can be seen in Figure 5.2 and Table 5.5.

Figure 5.2: DAG structure determined using the approximate algorithm.

<table>
<thead>
<tr>
<th>Class</th>
<th>Vert (Par)</th>
<th>1</th>
<th>2 (1345)</th>
<th>3 (1)</th>
<th>4 (15)</th>
<th>5 (13)</th>
<th>6 (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td></td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0100</td>
<td></td>
<td>00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0101</td>
<td></td>
<td>00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1101</td>
<td></td>
<td>00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0110</td>
<td></td>
<td>01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1100</td>
<td></td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1110</td>
<td></td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0111</td>
<td></td>
<td>01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1111</td>
<td></td>
<td>01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1010</td>
<td></td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0001</td>
<td></td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0000</td>
<td></td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1011</td>
<td></td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1001</td>
<td></td>
<td>01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0010</td>
<td></td>
<td>01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0011</td>
<td></td>
<td>01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.5: MAP partition for the parent configurations of each vertex in Figure 5.2.

Comparing Figure 5.1 and Figure 5.2, we find some similarities. Both graphs find the same links between vertices 1, 4, 5 and 1, 3, 5. The links between 2, 3, 5 are the
same except for direction.

Now, for the vertices 1, 2, 3, 4 and 6 the class connection graphs are connected which by Theorem 1 means that their corresponding partition is CSI-consistent. That is, they can be explained by some labels. Note however that for vertex 5, we have $s_{5,2} = \{00, 11\}$ which does not give a connected graph. Concentrating on the rest of the vertices and using that parent values with a Hamming distance of 1 give rise to a label, we obtain Figure 5.3.

Figure 5.3: LDAG structure determined using the approximate algorithm.

It might be interesting to compare these labels to the labels found in [4], where they developed an algorithm specifically for finding LDAGs, see Figure 5.4. It should be noted that in contrast to our uniform prior they used a prior that penalized the use of many free parameters in order to reduce overfitting.

As can be seen, the labels are the same for vertex 4. Not taking direction into account, every link in Figure 5.4 is in Figure 5.2.

5.2 Comparing the results for simulated data

5.2.1 Description of the simulated data

Here, a CSI-model introduced in [4] was simulated, consisting of 10 nodes in order to look at the learning method, see Figure 5.5.
Figure 5.5: The simulated CSI-model with labels.

Disregarding the labels, the probability distribution of $p(X_1, \ldots, X_{10})$ can be written as

$$p(X_1, \ldots, X_{10} = p(X_1)p(X_2|X_1) \cdot \ldots \cdot p(X_{10}|X_6, X_7, X_8, X_9).$$

The data is then created by first uniformly drawing each conditional probability distribution in the equation above and then accounting for the labels by putting some distributions equal to each other, as in the determination of the partition below.

In [5], the focus is on variable $X_5$ which has three parents $X_1, X_4$ and $X_8$ with labels such that if $X_1 = 0$, then $X_5$ becomes conditionally independent of the other two parents. For simplicity, we write $p(X_5|X_1 = x_1, X_4 = x_4, X_8 = x_8)$ as $p(X_5|x_1, x_4, x_8)$ in that order. So for example, by $p(X_5|0, 1, 0)$ we mean $p(X_5|X_1 = 0, X_4 = 1, X_8 = 0)$. By going through each label for $X_5$, we find

$$\mathcal{L}_{(4,5)} = \{(0, 0), (0, 1)\} \implies p(X_5|0, 0, 0) = p(X_5|0, 1, 0) \text{ and } p(X_5|0, 0, 1) = p(X_5|0, 1, 1)$$

$$\mathcal{L}_{(8,5)} = \{(0, 0), (0, 1)\} \implies p(X_5|0, 0, 0) = p(X_5|0, 0, 1) \text{ and } p(X_5|0, 1, 0) = p(X_5|0, 1, 1).$$

From here, it is easy to see that

$$p(X_5|0, 0, 0) = p(X_5|0, 1, 0) = p(X_5|0, 1, 1) = p(X_5|0, 0, 1).$$

That is, our labels imply that the 8 parent values should be partitioned into the following 5-partition:

$$\{(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1)\}, \{(1, 0, 0)\}, \{(1, 0, 1)\}, \{(1, 1, 0)\}, \{(1, 1, 1)\}.$$
5.2.2 Results of the simulated data

We ran the simulation four times and the results are shown in Figure 5.6. On the whole, the results are good with the exception of the third row. The result can be explained by the fact that the conditional distributions by randomness can be sufficiently close so that $10^5$ is too few elements for accurately finding the expected partition. This can also be seen by the adjusted Rand index; when the number off data points increases, the metric tends to, but does not reach, 1 with up to $10^5$ data points.
Figure 5.6: Comparison of the expected partition against the learned partition for two different metrics: zero-one-loss (left column) and adjusted Rand index (right column). The $x$-axis describe the size of the simulated data and $y$ the average of the metric over 100-simulations.
Conclusion

In this report, we have applied the algorithm developed by Dahl in order to approximate an exact method for learning PDAGs. We found that the algorithm made similar results as the exact method at a significantly shorter time.
Appendix A

Theory

Definition A.0.1 (Clusterings, Partitions). A clustering, or partition, of a data set \( X \) with \( n \) elements is a set \( C = \{C_1, \ldots, C_k\} \) where \( X = \bigcup_{i=1}^{k} C_i \) and \( C_i \cap C_j = \emptyset \) for all \( i, j = 1, \ldots, k \) with \( i \neq j \). We also require that each \( C_i \) is non-empty.

We need to be able two compare different partitions and to do that, we need a measure of similarity. In this report, we will use two such metrics. The first metric is the so called the zero-one-loss metric and is defined simply as

\[
d(C, C') = \begin{cases} 
1 & \text{if } C = C' \\
0 & \text{otherwise,}
\end{cases}
\]

where \( C = \{C_1, \ldots, C_k\} \) and \( C' = \{C'_1, \ldots, C'_l\} \) are two clusterings (=partitions) of the same set \( X \) with \( n \) elements. This metric is a bit prohibitive in that it doesn’t take into account that there are partitions that are not equal but still very similar. The second metric we will use take this into account and is known as the adjusted Rand index. In order to define it, let \( n_{ij} = |C_i \cap C'_j| \) be the number of elements in common in \( C_i \) and \( C'_j \). Then consider Table A.1. Here, \( a_i \) is the sum of row \( i \) and \( b_j \) the sum of column \( j \). The adjusted Rand index is then calculated as

\[
d(C, C') = \frac{\sum_{i=1}^{k} \sum_{j=1}^{l} \binom{n_{ij}}{2} - \left[ \sum_{i=1}^{k} \binom{a_i}{2} \sum_{j=1}^{l} \binom{b_j}{2} \right] / \binom{n}{2}}{\frac{1}{2} \left[ \sum_{i=1}^{k} \binom{a_i}{2} + \sum_{j=1}^{l} \binom{b_j}{2} \right] - \left[ \sum_{i=1}^{k} \binom{a_i}{2} \sum_{j=1}^{l} \binom{b_j}{2} \right] / \binom{n}{2}}
\]

It can be shown, see [8], that this is of the form

\[
\frac{\text{Index} - \text{ExpectedIndex}}{\text{MaxIndex} - \text{ExpectedIndex}},
\]

Table A.1: Contingency table used to calculated the adjusted Rand index.

<table>
<thead>
<tr>
<th>cluster</th>
<th>( C'_1 )</th>
<th>( C'_2 )</th>
<th>( \ldots )</th>
<th>( C'_l )</th>
<th>Sums</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_1 )</td>
<td>( n_{11} )</td>
<td>( n_{12} )</td>
<td>( \ldots )</td>
<td>( n_{1l} )</td>
<td>( a_1 )</td>
</tr>
<tr>
<td>( C_2 )</td>
<td>( n_{21} )</td>
<td>( n_{22} )</td>
<td>( \ldots )</td>
<td>( n_{2l} )</td>
<td>( a_2 )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( C_k )</td>
<td>( n_{k1} )</td>
<td>( n_{k2} )</td>
<td>( \ldots )</td>
<td>( n_{kl} )</td>
<td>( n_k )</td>
</tr>
<tr>
<td>Sums</td>
<td>( b_1 )</td>
<td>( b_2 )</td>
<td>( \ldots )</td>
<td>( n_l )</td>
<td>( n )</td>
</tr>
</tbody>
</table>
but we omit the details. What it means is that the metric is negative when the index is less than the expected index, the expected value of the metric is 0 for a random partition and 1 if they are equal.
Appendix B

Generating vertex labeled DAGs

Definition B.0.1 (Incidence matrix, [3]). Let $G = (V, E)$ be a graph with vertex set $V = \{v_1, \ldots, v_n\}$. The incidence matrix of $G$ is the $|V| \times |V|$ matrix $B$ defined by

$$b_{ij} = \begin{cases} 1 & \text{if there is an edge between } v_i \text{ and } v_j \\ 0 & \text{otherwise.} \end{cases}$$

Here, $b_{ij}$ is the element at row $i$ and column $j$.

Note that depending on how the vertices are assigned to $v_1, \ldots, v_n$, we get different incidence matrices.

Example B.0.1. Using the notation $(v_1, v_2, v_3, v_4, v_5, v_6) = (1, 2, 3, 4, 5, 6)$ in Figure 5.1 we get the incidence matrix

$$
\begin{pmatrix}
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 0 & 0 & 1 & 1 & 1 \\
2 & 0 & 0 & 1 & 0 & 1 \\
3 & 0 & 0 & 0 & 1 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 1 & 0 \\
6 & 0 & 0 & 0 & 1 & 1 \\
\end{pmatrix}
$$

On the other hand, with the notation $(v_1, v_2, v_3, v_4, v_5, v_6) = (4, 5, 6, 3, 2, 1)$ we get the incidence matrix

$$
\begin{pmatrix}
4 & 5 & 6 & 3 & 2 & 1 \\
4 & 0 & 0 & 0 & 0 & 0 \\
5 & 1 & 0 & 0 & 0 & 0 \\
6 & 1 & 1 & 0 & 0 & 0 \\
3 & 1 & 1 & 0 & 0 & 0 \\
2 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 \\
\end{pmatrix}
$$

The interesting thing here is that the second example is lower triangular. In fact, we will prove that for every DAG we can assign $v_1, \ldots, v_n$ such that the incidence matrix is lower triangular.
Theorem 3. Let $G = (V, E)$ be a (finite) graph, where $V = \{v_1, \ldots, v_n\}$. Then there exist a vertex labeling, i.e. an assignment of $v_1, \ldots, v_n$, of $G$ such that the incidence matrix is lower triangular with zeros on the diagonal if and only if $G$ is a DAG.

Proof. First assume that $G$ is a DAG. As $G$ is finite, there must exist a vertex $v_1$ with no outgoing edge since otherwise the graph would not be acyclic. Now, let $G'$ be the graph that is obtained by removing $v_1$ and all edges connecting to it from $G$. Then obviously $G'$ is also a DAG. This means that we can find a vertex $v_2$ without any outgoing edges in $G'$. We see that we can continue in this way until all vertices $v_1, v_2, \ldots, v_n$ have been considered.

Observe now that $v_2$ can at most be connected to $v_1$, $v_3$ can at most be connected to $v_1$ and $v_2$ and so on. This means that the incidence matrix is lower triangular with zeros on the diagonal.

The other direction is similarly handled and omitted for brevity. \hfill \Box

Without taking into account the vertex-labeling, this theorem allows us to enumerate all DAGs with $n$ vertices by enumerating all lower triangular $n^2$ matrices with zeros on the diagonal and whose other elements are either 0 or 1. In order to take the labeling into account, we just order the vertices in $n!$ ways. In total, this gives us $n!2^{(n-1)(n-2)/2}$ DAGs, some of which may be equal. Obviously, this method of generating DAGs is only appropriate if the number of vertices is sufficiently small.
Bibliography


