Structural-EM for Learning PDG Models from Incomplete Data

Jens D. Nielsen
Computer Science Dept.
University of Castilla-La Mancha
Albacete, Spain
dalgaard@dsi.uclm.es

Rafael Rumí **Antonio Salmerón**
Statistic and Applied Mathematics Dept.
University of Almería
Almería, Spain
{rrumi|antonio.salmeron}@ual.es

Abstract

Probabilistic Decision Graphs (PDGs) are a class of graphical models that can naturally encode some context specific independencies that cannot always be efficiently captured by other popular models, such as Bayesian Networks. Furthermore, inference can be carried out efficiently over a PDG, in time linear in the size of the model. The problem of learning PDGs from data has been studied in the literature, but only for the case of complete data. In this paper we propose an algorithm for learning PDGs in the presence of missing data. The proposed method is based on the EM algorithm for estimating the structure of the model as well as the parameters. We test our proposal on artificially generated data with different rates of missing cells, showing a reasonable performance.

1 Introduction

The Probabilistic Decision Graph (PDG) model was first introduced by Bozga and Maler (1999), and was originally proposed as an efficient representation of probabilistic transition systems. In this study, we consider the more generalised version of PDGs proposed by Jaeger (2004).

PDGs constitute a class of probabilistic graphical models that can represent some context specific independencies that can not efficiently be captured by Bayesian network (BN) models. Also, probabilistic inference can be carried out directly in the PDG structure and has a time complexity linear in the size of the PDG model. This makes learning of PDGs especially interesting, as we are learning directly the inference structure, which is in contrast to the usual scenario when learning general BN models.

The performance of the PDG model w.r.t. general probability estimation has previously been studied and results suggest that the model in general performs competitively when compared to BN or Naïve BN models (Nielsen and Jaeger, 2006). The PDG model has also been successfully applied to supervised classification problems (Nielsen et al., 2007).

In this paper we are concerned with the estimation of PDGs from data. The problem has been addressed by Jaeger et al. (2006), where an algorithm based on the optimisation of a score is proposed for learning from complete data. However, the task of learning PDGs in the presence of missing data has not yet been explored in the literature. The difficulty arises in the computation of the score for a model given the database with missing values. A similar problem is found in the case of learning BNs from incomplete databases. Friedman (1997) addressed this problem by proposing an algorithm for estimating the structure of a BN model based on the Expectation-Maximisation (EM) principle (Dempster et al., 1977; Lauritzen, 1995).

We propose an algorithm for learning PDGs inspired by the proposal of Friedman (1997), based on the EM principle. Both the structure and the parameters are re-adjusted in each iteration of the algorithm. That is, the adjustments made to the structure are guided by the expected increase in some score metric, while the adjustments made to the parameters are guided by the expected likelihood of a completed version of the incomplete data.

2 Background and Notation

We will denote random variables by uppercase letters, e.g. X, and sets with boldface uppercase letters, e.g. $\mathbf{X}$. When X_i is a discrete categorical random variable, we will by lowercase letter $x_{i,j}$ refer to the j'th state of X_i under some ordering. We will by $R(X_i)$ refer to the set of possible states of X_i, and by $R(\mathbf{X}) = \times_{X_i \in \mathbf{X}} R(X_i)$ when $\mathbf{X}$ is a set of variables. We will use r_i as a shorthand for $|R(X_i)|$. By lowercase bold letters we refer to joint states of sets of variables, e.g. $\mathbf{x} \in R(\mathbf{X})$. When $X_i \in \mathbf{X}$ and $\mathbf{x} \in R(\mathbf{X})$ we denote $\mathbf{x}[X_i]$ the projection of $\mathbf{x}$ onto coordinate X_i.

Let $G = \langle \mathbf{V}, \mathbf{E} \rangle$ be a directed graph structure with set of nodes $\mathbf{V} = \{V_1, \ldots, V_n\}$ and set of directed edges $\mathbf{E} \subset \mathbf{V} \times \mathbf{V}$. We will then by $ch_G(V_i)$ and $pa_G(V_i)$ refer the set of children of V_i and parents of V_i respectively in structure G, hence $ch_G(V_i) = \{V_j \in \mathbf{V} : (V_i, V_j) \in \mathbf{E}\}$ and $pa_G(V_i) = \{V_j \in \mathbf{V} : (V_j, V_i) \in \mathbf{E}\}$. A tree is a directed acyclic graph where one unique node $V_r \in \mathbf{V}$ is designated root and has no parents $pa_G(V_r) = \emptyset$ while all other nodes have exactly one parent. A forest structure is a set of such trees.

2.1 The Probabilistic Decision Graph Model

A PDG encodes a joint probability distribution over a set of categorical random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ by a factorisation defined by a structure over a set of local distributions.

Definition 2.1 (The PDG Structure). *Let F be a forest structure over $\mathbf{X} = \{X_1, \ldots, X_n\}$. A* PDG-structure *$G = \langle \mathbf{V}, \mathbf{E} \rangle$ for $\mathbf{X}$ w.r.t. F is a set of* rooted *acyclic directed graphs over nodes $\mathbf{V}$, such that:*

1. *Each node $\nu \in \mathbf{V}$ represents a unique $X_i \in \mathbf{X}$ and all $X_i \in \mathbf{X}$ are represented by at least one node $\nu \in \mathbf{V}$. We will by $\nu_{i,j}$ refer to the j'th node representing X_i under some ordering of the set of nodes representing X_i.*

2. *For each node $\nu_{i,j}$, each possible state $x_{i,h}$ of X_i and each successor $X_k \in ch_F(X_i)$ there exists exactly* one *edge $(\nu_{i,j}, \nu_{k,l}) \in \mathbf{E}$ with label $x_{i,h}$, where $\nu_{k,l}$ is some node representing X_k.*

Let $X_k \in ch_F(X_i)$. By $succ(\nu_{i,j}, X_k, x_{i,h})$ we refer to the unique node $\nu_{k,l}$ representing X_k that is reached from $\nu_{i,j}$ by following the edge with label $x_{i,h}$.

Example 2.1. *A forest F over binary variables $\mathbf{X} = \{X_0, \ldots, X_7\}$ can be seen in Figure 1(a), and a PDG structure over $\mathbf{X}$ w.r.t. F in Figure 1(b). The labelling of nodes in the PDG-structure is indicated in subscripts and (redundant) by the dashed boxes, e.g., the nodes representing X_2 are $\{\nu_{2,0}, \nu_{2,1}\}$. Dashed edges correspond to edges labelled 0 and solid edges correspond to edges labelled 1, for instance $succ(\nu_{5,0}, X_6, 0) = \nu_{6,1}$.*

A PDG structure is instantiated by assigning to every node a local probability distribution over the variable that it represents. By a PDG model over discrete random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ we refer to a pair $\mathcal{G} = \langle G, \mathbf{\Theta} \rangle$ where G is a PDG structure over $\mathbf{X}$ and $\mathbf{\Theta}$ is an instantiation of G. We denote by $\mathbf{p}^{\nu_{i,j}}$ the local distribution assigned to node $\nu_{i,j}$, and by $p^{\nu_{i,j}}_{x_{i,h}}$ the probability for state $x_{i,h}$ in local distribution $\mathbf{p}^{\nu_{i,j}}$. The semantics of the local distribution $\mathbf{p}^{\nu_{i,j}}$ is defined by the path(s) leading to the node $\nu_{i,j}$ from the root, that is, how $\nu_{i,j}$ can be *reached.* Let G be a PDG structure over variables $\mathbf{X}$ w.r.t. forest F. A node $\nu_{i,j}$ in G is *reached* by $\mathbf{x} \in R(\mathbf{X})$ if

- $\nu_{i,j}$ is a root in G, or
- $X_i \in ch_F(X_k)$, $\nu_{k,l}$ is reached by $\mathbf{x}$ and $\nu_{i,j} = succ(\nu_{k,l}, X_i, \mathbf{x}[X_k])$.

By $reach_G(i, \mathbf{x})$ we denote the unique node representing X_i reached by $\mathbf{x}$ in PDG-structure G.

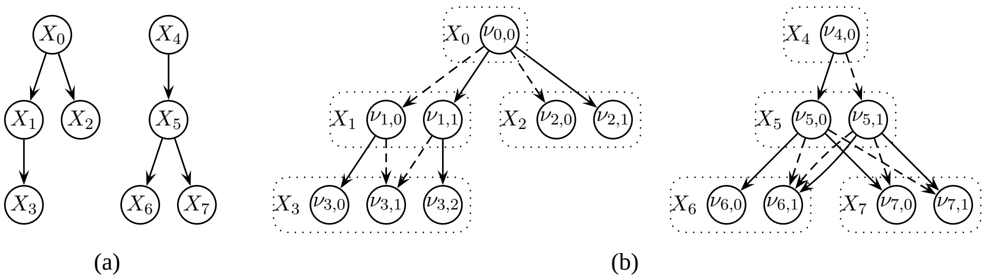
A PDG model $\mathcal{G} = \langle G, \mathbf{\Theta} \rangle$ over variables $\mathbf{X}$ represents a joint distribution $P^{\mathcal{G}}$ by the following factorisation:

$$P^{\mathcal{G}}(\mathbf{x}) = \prod_{X_i \in \mathbf{X}} p^{reach_G(i,\mathbf{x})}_{\mathbf{x}[X_i]}. \tag{1}$$

Example 2.2. *To instantiate the PDG structure in Fig. 1(b), we assign a local distribution to each node in the structure with the probabilistic interpretation given in Fig. 1(c). We can read some context specific independencies of this table, e.g. X_6 is independent of X_5 only in the context $X_4 = 0$.*

2.2 Selecting PDG models using complete data

For assessing models in the presence of observed data, we can use a penalised likelihood score function. Let $\mathcal{G}$ be a PDG model over variables $\mathbf{X} =$



X_0
X_1
X_2
X_3
X_4
X_5
X_6
X_7
(a)
X_0 $\nu_{0,0}$
X_1 $\nu_{1,0}$ $\nu_{1,1}$
X_2 $\nu_{2,0}$ $\nu_{2,1}$
X_3 $\nu_{3,0}$ $\nu_{3,1}$ $\nu_{3,2}$
X_4 $\nu_{4,0}$
X_5 $\nu_{5,0}$ $\nu_{5,1}$
X_6 $\nu_{6,0}$ $\nu_{6,1}$
X_7 $\nu_{7,0}$ $\nu_{7,1}$
(b)

$\mathbf{p}^{\nu_{0,0}} = P(X_0)$	$\mathbf{p}^{\nu_{3,0}} = P(X_3 \vert X_0 = 0, X_1 = 1)$	$\mathbf{p}^{\nu_{5,1}} = P(X_5 \vert X_4 = 0)$
$\mathbf{p}^{\nu_{1,0}} = P(X_1 \vert X_0 = 0)$	$\mathbf{p}^{\nu_{3,1}} = P(X_3 \vert X_1 = 0)$	$\mathbf{p}^{\nu_{6,0}} = P(X_6 \vert X_4 = 1, X_5 = 1)$
$\mathbf{p}^{\nu_{1,1}} = P(X_1 \vert X_0 = 1)$	$\mathbf{p}^{\nu_{3,2}} = P(X_3 \vert X_0 = 1, X_1 = 1)$	$\mathbf{p}^{\nu_{6,1}} = P(X_6 \vert X_4 = 0 \vee \{X_4 = 1, X_5 = 0\})$
$\mathbf{p}^{\nu_{2,0}} = P(X_2 \vert X_0 = 0)$	$\mathbf{p}^{\nu_{4,0}} = P(X_4)$	$\mathbf{p}^{\nu_{7,0}} = P(X_7 \vert X_4 = X_5)$
$\mathbf{p}^{\nu_{2,1}} = P(X_2 \vert X_0 = 1)$	$\mathbf{p}^{\nu_{5,0}} = P(X_5 \vert X_4 = 1)$	$\mathbf{p}^{\nu_{7,0}} = P(X_7 \vert X_4 \neq X_5)$

(c)

Figure 1: A forest F over binary variables $\mathbf{X} = \{X_0, \ldots, X_7\}$ is shown in (a), and a PDG-structure over $\mathbf{X}$ w.r.t. variable forest F is shown in (b). In the PDG-structure in (b), solid edges are labelled with value 1 and dashed edges are labelled with value 0. In (c), we have indicated the probabilistic interpretation of the parameters for each node in the PDG structure of (b).

$\{X_1, \ldots, X_n\}$ and let $\mathbf{D}$ be a set of N complete observations of $\mathbf{X}$, then we define a general score function as:

$$S_\lambda(\mathbf{D}, \mathcal{G}) = (1-\lambda) \cdot L(\mathbf{D}, \mathcal{G}) - \lambda \cdot \mathit{size}(\mathcal{G}) \,, \quad (2)$$

where $0 < \lambda < 1$, $\mathit{size}(\mathcal{G})$ is some measure of complexity of $\mathcal{G}$ and $L(\mathbf{D}, \mathcal{G})$ is the log-likelihood of $\mathbf{D}$ given $\mathcal{G}$. A typical definition of $\mathit{size}(G)$ is the number of free parameters in model $\mathcal{G}$. Using the following notation $\mathbf{D} = \{X_i^k : 1 \leq i \leq n, 1 \leq k \leq N\}$, the log-likelihood $L(\mathbf{D}, \mathcal{G})$ is:

$$\begin{aligned} L(\mathbf{D}, \mathcal{G}) &= \log \prod_{k=1}^{N} P^{\mathcal{G}}(X_1^k, \ldots, X_n^k) \\ &= \sum_{k=1}^{N} \log P^{\mathcal{G}}(X_1^k, \ldots, X_n^k) \\ &= \sum_{i=1}^{n} \sum_{h=1}^{r_i} \sum_{j=1}^{v_i} \#_{\mathbf{D}}(x_{i,h}, \nu_{i,j}) \log p_{x_{i,h}}^{\nu_{i,j}} \,, \quad (3) \end{aligned}$$

where v_i is the number of nodes representing X_i and $\#_{\mathbf{D}}(E)$ is the count of instances in $\mathbf{D}$ satisfying requirement E. For example, in Eq. (3) $\#_{\mathbf{D}}(x_{i,h}, \nu_{i,j})$ is the count of data items in $\mathbf{D}$ where variable X_i is observed in state $x_{i,h}$ and where the $\nu_{i,j}$ is reached.

3 Learning from Incomplete Data

Assume incomplete data, that is $\mathbf{D} = \mathbf{D}_O \cup \mathbf{D}_M$, where $\mathbf{D}_O$ is the elements of $\mathbf{D}$ containing a value and $\mathbf{D}_M = \mathbf{D} \setminus \mathbf{D}_O$. We can compute the expected log likelihood of $\mathbf{D}$ in model $\mathcal{G}$ (over variables $\mathbf{X}$), given some distribution P^* over $\mathbf{D}_M$ as:

$$\begin{aligned} &E[L(\mathbf{D}, \mathcal{G}) \vert \mathbf{D}_O, P^*] = \\ &\quad \sum_{i=1}^{n} \sum_{h=1}^{r_i} \sum_{j=1}^{v_i} E[\#_{\mathbf{D}}(x_{i,h}, \nu_{i,j})] \log p_{x_{i,h}}^{\nu_{i,j}} \,, \quad (4) \end{aligned}$$

where the second expectation also is with respect to $\mathbf{D}_O$ and P^*. In a structural EM algorithm like the one proposed by Friedman (1997), we optimise the expected likelihood instead of directly optimising the likelihood which in the presence of incomplete data no longer decomposes. Decomposability of the likelihood is important for model selection in which the search procedure in each step evaluates candidate models from a neighbourhood that is generated from a current model by local transformation. We define the expected score as a function Q:

$$\begin{aligned} &Q(\mathcal{G}, \mathbf{D} \vert \mathcal{G}^*) = \\ &\quad (1-\lambda) E[L(\mathbf{D}, \mathcal{G}) \vert \mathbf{D}_O, \mathcal{G}^*] - \lambda \mathit{size}(\mathcal{G}) \;. \quad (5) \end{aligned}$$

In Eq. (5) we use the current model $\mathcal{G}^*$ as the reference distribution P^*. The structural EM procedure can now be stated as in Algorithm 1.

First, in line 5 of Alg. 1 we basically need to find MAP parameters for $\mathcal{G}$. Exact methods are usually intractable, so normally some approximation

Algorithm 1 The structural EM procedure

```
1: procedure SEM(D)
2:     Let $\mathcal{G}^0 = \langle G^0, \mathbf{\Theta}^0 \rangle$ be the initial model.
3:     $n \leftarrow 0$
4:     repeat
5:         $\mathbf{\Theta}^{n+1} \leftarrow \underset{Legal\ \mathbf{\Theta}}{argmax}\ Q(\langle G^n, \mathbf{\Theta} \rangle, \mathbf{D} | \mathbf{D}_O, \mathcal{G}^n)$
6:         $G^{n+1} \leftarrow \underset{G \in \mathcal{N}(G^n)}{argmax}\ Q(\langle G, \cdot \rangle, \mathbf{D} | \mathbf{D}_O, \langle G^n, \mathbf{\Theta}^{n+1} \rangle)$
7:         $\mathcal{G}^{n+1} \leftarrow \langle G^{n+1}, \mathbf{\Theta}^{n+1} \rangle$
8:         $n \leftarrow n + 1$
9:     until $Q(\mathcal{G}^n, \mathbf{D} | \mathcal{G}^{n-1}) \leq Q(\mathcal{G}^{n-1}, \mathbf{D} | \mathbf{D}_O, \mathcal{G}^{n-1})$
10:    return $\mathcal{G}^{n-1}$
```

method is employed. Originally, Friedman (1997) proposed to use a standard EM approach in this step while Peña et al. (2000) propose as a more computationally efficient alternative to use the *branch and bound* procedure of Ramoni and Sebastiani (1997). However, the choice of approach in this step is not crucial to the following discussion in Sec. 4.

Second, in line 6 of Alg. 1, the function $\mathcal{N}(\cdot)$ is the neighbourhood generating function. We will define simple split and merge operations that implement structural modifications for generating neighbours from a current PDG model $\mathcal{G}$. We will show how to compute the expectations needed to evaluate the expected score of a neighbour.

4 SEM for PDG Models

In this section we will explain how Alg. 1 can be applied to PDG models. First, for constructing an initial model we need a forest structure over the variables in the domain. We accomplish this using the algorithm of Chow and Liu (1968) that induces a maximum weight spanning tree using mutual information as the edge weights. In Sec. 5 we explain how to compute mutual information from incomplete data.

Inducing the initial tree by finding a maximum weight spanning tree using mutual information as edge-weights, is different from previously proposed approaches for inducing variable forests/trees. In (Jaeger et al., 2006) a χ^2 test for conditional independence is used to assign marginally independent variables in different trees and conditionally independent variables in different sub-trees. In this study we use the above mentioned mutual information based method as it is less data-intensive compared to repeated χ^2 tests.

Assuming that we have the initial tree structure F over the variables, we initialise a PDG model as follows: for every variable $X_i \in \mathbf{X}$ with $pa_F(X_i) = X_k$, we create $v_i = r_k$ new nodes $\{\nu_{i,1}, \nu_{i,2}, \ldots, \nu_{i,v_i}\}$ representing X_i. We then connect every node $\nu_{k,j}$ representing X_k such that $succ(\nu_{k,j}, X_i, x_{k,z}) = \nu_{i,z}$. That is, for state $x_{k,z}$ of variable X_k the node $\nu_{i,z}$ is always reached. Constructing the initial PDG model in this way allows every variable to be modelled as marginally dependent on its parent and its set of children in F.

Finally, we use a random parametrisation $\mathbf{\Theta}^0$ of the initial structure G^0.

4.1 The Neighbourhood of a Model

In this subsection we explain how the neighbourhood $\mathcal{N}(G)$ of a PDG structure G is generated. We include operations that work on the PDG structure only, and leave the structure over the variables fixed. Operations that change the structure over the variables (e.g. operations that swap the position of two variables) are problematic as they potentially require the creation of a lot of new node connections. Offhand, it is not intuitive to us how to best do this in general, and therefore we choose to focus on the following two less dramatic structural changes that have both previously been used by Jaeger et al. (2006) for learning in the case of complete data[1].

Merging Nodes The merge operator takes a pair of nodes $\{\nu_{i,a}, \nu_{i,b}\}$ representing the same variable X_i. The nodes $\nu_{i,a}$ and $\nu_{i,b}$ are selected such that $succ(\nu_{i,a}, X_j, x_{i,h}) = succ(\nu_{i,b}, X_j, x_{i,h})$ for any state $x_{i,h} \in R(X_i)$ and child $X_j \in ch_F(X_i)$ in the variable forest F. The merge operation then simply consists in replacing nodes $\nu_{i,a}$ and $\nu_{i,b}$ with a new node $\nu_{i,c}$, where $\nu_{i,c}$ has as children exactly the children of $\nu_{i,a}$ (or $\nu_{i,b}$) and as parents inherits the union of parents of $\nu_{i,a}$ and parents of $\nu_{i,b}$.

Splitting Nodes The splitting operator takes as input a single node $\nu_{i,j}$ with m parents where $2 \leq m$, and replaces $\nu_{i,j}$ with m new nodes all representing X_i. Each new node inherits all the children of $\nu_{i,j}$,

[1]Jaeger et al. (2006) use an additional third operator that redirects edges. We leave this operator out of the algorithm for simplicity. Furthermore, for a given tree structure over the variables, any PDG structure can be reached from any other PDG structure using only merge and split operations.

and exactly one unique parent of $\nu_{i,j}$.

4.2 Scoring a Neighbour Model

In this section we detail how to compute the score $Q(\langle G', \cdot\rangle, \mathbf{D}|\mathbf{D}_O, \langle G, \boldsymbol{\Theta}\rangle)$ of a neighbour $G' \in \mathcal{N}(G)$ generated by merging two nodes or splitting a node. In fact, we will not compute the full expected score, but only the terms that are different.

4.2.1 Scoring a Merge Operation

Assume PDG $\mathcal{G}'$ is constructed from PDG $\mathcal{G} = \langle G, \Theta\rangle$ by merging nodes $\nu_{i,a}, \nu_{i,b} \in \mathbf{V}_i$ in structure G. Let the node $\nu_{i,c}$ be the one replacing $\nu_{i,a}$ and $\nu_{i,b}$ in $\mathcal{G}'$, and assume that we have computed (and stored) all expected counts of the form $E[\#_{\mathbf{D}}(x_{i,h}, \nu_{i,j})|\mathbf{D}_O, \mathcal{G}]$. Then, computing the expected counts for model $\mathcal{G}'$ under distribution $P^{\mathcal{G}}$ reduces to computing expected counts for $\nu_{i,c}$, which can be done efficiently from expectations $\#_{\mathbf{D}}(x_{i,h}, \nu_{i,a})$ and $\#_{\mathbf{D}}(x_{i,h}, \nu_{i,b})$, that is :

$$E[\#_{\mathbf{D}}(x_{i,h}, \nu_{i,c})|\mathbf{D}_O, \mathcal{G}] = E[\#_{\mathbf{D}}(x_{i,h}, \nu_{i,a})|\mathbf{D}_O, \mathcal{G}] + E[\#_{\mathbf{D}}(x_{i,h}, \nu_{i,b})|\mathbf{D}_O, \mathcal{G}]\,.$$

Hence, computing the difference in expected score $\Delta Q_{merge}(\nu_{i,a}, \nu_{i,b}) = Q(\mathcal{G}', \mathbf{D}|\mathbf{D}_O, \mathcal{G}) - Q(\mathcal{G}, \mathbf{D}|\mathbf{D}_O, \mathcal{G})$ reduces to computing the difference between the terms of the expected score involving nodes $\nu_{i,a}$ and $\nu_{i,b}$ and the new node $\nu_{i,c}$:

$$\begin{aligned}\Delta Q_{merge}(\nu_{i,a}, \nu_{i,b}) &= Q(\mathcal{G}', \mathbf{D}|\mathcal{G}) - Q(\mathcal{G}, \mathbf{D}|\mathcal{G}) \\ &= (1-\lambda)\left(\sum_{h=1}^{r_i} E[L_h^{\nu_{i,c}} - L_h^{\nu_{i,a}} - L_h^{\nu_{i,b}}|\mathbf{D}_O, \mathcal{G}]\right) \\ &\quad + \lambda \cdot (r_i - 1) \quad (6)\end{aligned}$$

where $\nu_{i,c}$ is the node resulting from merging $\nu_{i,a}$ and $\nu_{i,b}$, and $L_h^{\nu_{i,j}}$ is the term in the log-likelihood corresponding to node $\nu_{i,j}$ and the hth state of X_i. The expectation in (6) obviously can be computed term by term, and we see that $E[L_h^{\nu_{i,c}}|\mathbf{D}_O, \mathcal{G}] = E[\#_{\mathbf{D}}(x_{i,h}, \nu_{i,c})|\mathbf{D}_O, \mathcal{G}] \log E[p^{\nu_{i,c}}_{x_{i,h}}|\mathbf{D}_O, \mathcal{G}]$, where the expectation $E[p^{\nu_{i,c}}_{x_{i,h}}|\mathbf{D}_O, \mathcal{G}]$ is computed as the fraction $\frac{E[\#_{\mathbf{D}}(x_{i,h}, \nu_{i,c})|\mathbf{D}_O, \mathcal{G}]}{E[\#_{\mathbf{D}}(\nu_{i,c})|\mathbf{D}_O, \mathcal{G}]}$. The count $\#_{\mathbf{D}}(\nu_{i,a})$ is just $\sum_{h=1}^{r_i} \#_{\mathbf{D}}(x_{i,h}, \nu_{i,a})$.

The expectations $E[\#_{\mathbf{D}}(x_{i,h}, \nu_{i,j})|\mathbf{D}_O, \mathcal{G}]$ for any state $x_{i,h} \in R(X_i)$ and node $\nu_{i,j} \in \mathbf{V}_i$ are exactly the expectations we would compute in the parametric EM step in line 5 of Alg. 1. Therefore, these counts have already been computed for structure $\mathcal{G}^n$ in line 5 of Alg. 1, and can easily be made available at no extra cost.

4.2.2 Scoring a Split Operation

Let *inc*$(\nu_{i,j})$ be the set of edges incoming to $\nu_{i,j}$ in PDG structure $G = \langle \mathbf{V}, \mathbf{E}\rangle$, that is *inc*$(\nu_{i,j}) = \{(\nu_{k,z}, \nu_{i,j}) \in \mathbf{E}\}$. By $l^u_{\nu_{i,j}}$ we will denote the u'th element of *inc*$(\nu_{i,j})$ under some ordering. With $\nu^u_{i,j}$ we denote the node replacing $\nu_{i,j}$ for its uth incoming edge. Node $\nu_{i,j}$ is representing variable X_i and let the parent of X_i in the variable forest be X_k, hence by the definition of PDG structure, all parent nodes of $\nu_{i,j}$ represent variable X_k. The expected counts $E[\#_{\mathbf{D}}(\nu^u_{i,j}, x_{i,h})|\mathbf{D}_O, \mathcal{G}]$ for the node $\nu^u_{i,j}$ where $l^u_{\nu_{i,j}} = (\nu_{k,z}, \nu_{i,j})$ is labelled with state $x_{k,g}$ is then:

$$\begin{aligned}&E[\#_{\mathbf{D}}(\nu^u_{i,j}, x_{i,h})|\mathbf{D}_O, \mathcal{G}] = \\ &\qquad E[\#_{\mathbf{D}}(\nu_{k,z}, x_{k,g}, x_{i,h})|\mathbf{D}_O, \mathcal{G}]\,. \quad (7)\end{aligned}$$

The expectation in Eq. (7) can not be reconstructed from expected counts already computed for $\mathcal{G}$ in the structural parametric EM step of Alg. 1 (line 5) as was the case for the counts needed to evaluate a merge operation. However, anticipating that we will need such counts, we can store them during the computation of expectations in line 5 of Alg. 1. Assume that we have these expected counts available for structure G under the distribution defined by the PDG model $\mathcal{G} = \langle G, \Theta\rangle$. We can then compute the difference $\Delta Q_{split}(\nu_{i,j}) = Q(\mathcal{G}', \mathbf{D}|\mathcal{G}) - Q(\mathcal{G}, \mathbf{D}|\mathcal{G})$ for PDG model $\mathcal{G}'$ with structure G' generated by splitting node $\nu_{i,j}$ in structure G, as follows:

$$\begin{aligned}\Delta Q_{split}(\nu_{i,j}) &= Q(\mathcal{G}', \mathbf{D}|\mathcal{G}) - Q(\mathcal{G}, \mathbf{D}|\mathcal{G}) \\ &= (1-\lambda)\left[\sum_{h=1}^{r_i}\left(\sum_{u=1}^{m} E[L_h^{\nu^u_{i,j}}|\mathbf{D}_O, \mathcal{G}]\right) - \right. \\ &\left. E[L_h^{\nu_{i,j}}|\mathbf{D}_O, \mathcal{G}]\right] - \lambda(|\mathit{inc}(\nu_{i,j})| - 1)(r_i - 1)\,, \quad (8)\end{aligned}$$

where the log-likelihood terms $L^{\cdot}_{\cdot}$ are as described in Sec. 4.2.1. Further, it is clear that we can not split a root node as it is without parents.

4.3 Computing the Expectations

In order to compute the expected counts in sections 4.2.1 and 4.2.2, it is necessary to calculate probabilities of the form $P^{\mathcal{G}}(\{\nu \text{ is reached } \wedge X_i = x_i\}|\mathbf{Y} = \mathbf{y})$ for all $X_i \in \mathbf{X}$ and $\nu \in \mathbf{V}_i$, where $\mathcal{G}$ is a PDG over variables $\mathbf{X}$ and $\mathbf{y}$ is a joint observation of variables $\mathbf{Y} \subset \mathbf{X}$.

The computation of such probabilities can be done efficiently using the algorithm described by Jaeger (2004), which carries out the inference in time linear in the size of the PDG model. Broadly speaking, the desired probability is computed by first restricting the PDG $\mathcal{G}$ to $\mathbf{Y} = \mathbf{y}$. Then, for each node in the structure we compute parts of the product in (1) corresponding to incoming edges and outgoing edges. And, finally using these intermediate results stored in each node we can compute the desired conditional probabilities. We refer the reader to (Jaeger, 2004, Section 4) for details on PDG inference.

5 Estimating the Mutual Information with Missing Data

The mutual information between two random variables X and Y is defined as:

$$I(X,Y) = \sum_{i=1}^{|R(X)|} \sum_{j=1}^{|R(Y)|} p(x_i,y_j) \log \frac{p(x_i,y_j)}{p(x_i)p(y_j)} \quad . \tag{9}$$

As the joint distribution of X and Y is unknown, we need to estimate the mutual information from data. Assume we have a database $\mathbf{D}$ probably containing missing data. We require estimates for $\theta_{ij} = p(x_i,y_j)$, $\theta_{i.} = p(x_i)$ and $\theta_{.j} = p(y_j)$ for $i = 1,\ldots,|R(X)|$ and $j = 1,\ldots,|R(Y)|$. Actually, we only need to estimate θ_{ij}, since $\theta_{i.} = \sum_{j=1}^{|R(Y)|} \theta_{ij}$ and $\theta_{.j} = \sum_{i=1}^{|R(X)|} \theta_{ij}$.

Since $\mathbf{D}$ may contain missing data, we can use the EM algorithm to estimate the required parameters. The detailed procedure is given in Alg. 2.

Notice that steps 5 and 9 in algorithm 2 correspond, respectively, to the E and M steps of algorithm EM.

The value E_{ij} computed in line 7 of Alg. 2 is the expected number of records in $\mathbf{D}$ where X takes its i-th value and Y takes its j-th value. It is computed by exploring all the records $\mathbf{d} \in \mathbf{D}$

Algorithm 2 EM for estimating the mutual information

```
1: procedure EM_MutualInformation($\mathbf{D}$)
2:   Let $\mathbf{\Theta}^0 = \{\theta_{ij},\ i = 1,\ldots,|R(X)|,\ j = 1,\ldots,|R(Y)|\}$ be a random parametrisation of $p(x,y)$.
3:   $n \leftarrow 0$.
4:   repeat
5:     for all $i = 1,\ldots,|R(X)|$ do
6:       for all $j = 1,\ldots,|R(Y)|$ do
7:         $E_{ij} \leftarrow E\left[\#_{\mathbf{D}}(X = x_i, Y = y_j)|\mathbf{\Theta}^n\right]$
8:     $\mathbf{\Theta}^{n+1} \leftarrow \emptyset$
9:     for all $i = 1,\ldots,|R(X)|$ do
10:      for all $j = 1,\ldots,|R(Y)|$ do
11:        $\theta_{ij}^{n+1} \leftarrow \frac{E_{ij}}{\sum_{k=1}^{|R(X)|}\sum_{l=1}^{|R(Y)|} E_{kl}}$
12:        $\mathbf{\Theta}^{n+1} \leftarrow \mathbf{\Theta}^{n+1} \cup \{\theta_{ij}^{n+1}\}$
13:    $n \leftarrow n + 1$.
14:  until $L(\mathbf{D}|\mathbf{\Theta}^n) \leq L(\mathbf{D}|\mathbf{\Theta}^{n-1})$.
15:  Estimate $I(X,Y)$ as:
       $$\hat{I}(X,Y) = \sum_{i=1}^{|R(X)|} \sum_{j=1}^{|R(Y)|} \theta_{ij}^n \log \frac{\theta_{ij}^n}{\theta_{i.}^n \theta_{.j}^n} \quad .$$
16:  return $\hat{I}(X,Y)$.
```

and calculating, for each record, the probability $P\{X = x_i, Y = y_j|\mathbf{d}, \mathbf{\Theta}^n\}$. That is, we compute:

$$E\left[\#_{\mathbf{D}}(X = x_i, Y = y_j)|\mathbf{\Theta}^n\right] = \sum_{\mathbf{d}\in\mathbf{D}} P\{X = x_i, Y = y_j|\mathbf{d}, \mathbf{\Theta}^n\} \quad . \tag{10}$$

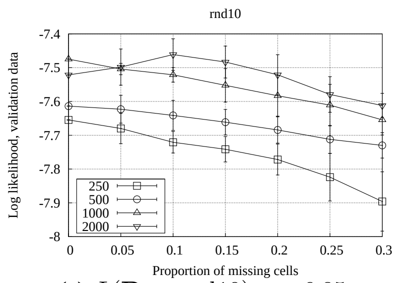
The probability in Eq. (10) will be equal to 0 if the record has a value different to (x_i, y_j) and equal to 1 if the record is exactly equal to (x_i, y_j). If some of the cells in the record is missing, the probability is computed using the current estimates $\mathbf{\Theta}^n$.

6 Experiments

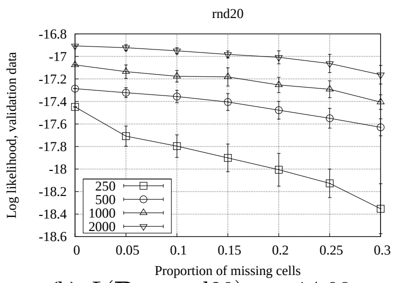
In order to test Alg. 1 we have performed experiments over different synthetic databases sampled from PDG models with 10, 20 and 40 variables generated at random[2]. We will refer to these models as rnd10, rnd20 and rnd40 respectively. From each PDG model, we constructed four databases containing 250, 500, 1000 and 2000 complete samples.

For each database, we have considered different rates of missing values, ranging from 5% to 30%. For each rate of missing values we generated

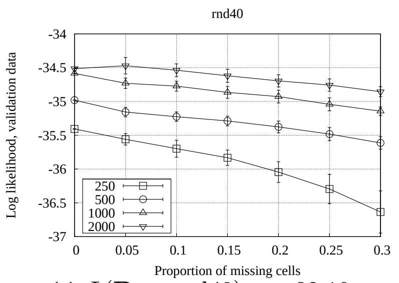
[2]The PDG models were generated at random with the restriction that they consist of a single variable tree.



(a) $L(\mathbf{D}_{\text{val}}, \text{rnd10}) = -6.85$



(b) $L(\mathbf{D}_{\text{val}}, \text{rnd20}) = -14.98$



(c) $L(\mathbf{D}_{\text{val}}, \text{rnd40}) = -32.10$

Figure 2: Mean and standard deviation of log-likelihood of a validation set of 10000 complete samples computed in the models leant from datasets sampled from model rnd10 (a), rnd20 (b) and rnd40 (c). The log-likelihood of the generating models are indicated beneath the plots.

50 databases from the original (complete) database by randomly erasing the value in a fraction of the cells according to the rate of missing values. The learning algorithm was then executed on each of the 50 databases measuring the quality of the learnt model as the log-likelihood of a separate validation database containing 10000 complete samples.

As score function we used the S_λ function of equation (2) with λ adjusted according to the size of the database to give a tradeoff between size and likelihood equivalent to the one imposed by the BIC score[3]. Finally, in order to speed up the algorithm, we put a limit of 10 iterations in each parametric EM[4] and 100 iterations in structural EM (the loop of Alg. 1).

6.1 Results

In Fig. 2(a-c) we show plots of mean and standard deviations of the log-likelihood of models learnt in the experiments described above.

First, the plots of Fig. 2 in general show the expected behaviour as mean likelihood generally decreases as a result of increasing the proportion of missing cells in the training data, while standard deviation increases. We note, also as expected, that the experiments on the larger data sets reach higher likelihood on the validation data and also show a more stable performance with less increase in standard deviation as the rate of missing values is increased.

[3]Setting $\lambda = \left(\frac{2N}{\log(N)} + 1\right)^{-1}$ where N is the number of observations yields BIC tradeoff.

[4]We run a 100 iterations parametric EM to optimise the parameters of the final model.

Second, in the experiment using 2000 samples from the rnd10 model (Fig. 2(a)) we observe an increase in likelihood up until a rate of 10% missing values. This behaviour may be caused by the algorithm over-fitting to the complete (rate 0% missing values) training data, while the introduction of some missing values helps the algorithm learn a less specific model with better ability to generalise. However, we only observe this behaviour for that specific data set which is somewhat unexpected assuming our explanation is valid.

Lastly, the initial tree structure is created using the classical algorithm of Chow and Liu (1968) as explained in Sect. 4. This initial model is itself a very commonly used model in probability estimation due to its simple restricted syntax and consequently efficient learning and inference. We therefore compare the quality of our final model to this initial model. From each experiment with missing data (72 total) we measured the likelihood of the validation data in the initial model as well as in the final PDG model. Using a Wilcoxon signed rank test for paired samples with significance level 0.05, we found significantly lower likelihood of the PDG model in 2 cases, no significant difference in 5 cases while in 65 cases we found significant better likelihood of the PDG model. The PDG performed significantly worse when using 500 samples of the rnd10 model with 25% and 30% missing cells, and no significant difference could be established for the experiments using the 250 samples of rnd10 with 30% missing, the 500 samples of rnd10 with 20% missing, the 250 samples of rnd20 with 30% missing, the 500 samples of rnd20 with 30% missing and the 500 samples of rnd40 with 30% missing.

7 Concluding Remarks

In this paper we have proposed an algorithm for learning PDG models in the presence of missing data. Our proposal was inspired by previous work on learning BN models from incomplete data by Friedman (1997). We have tested our proposal on synthetic data sampled from randomly constructed PDG models. The experiments show that the algorithm performs well and behaves well even when the rate of missing cells are increased. Statistical tests shows significant improvement in quality over the initial Markov tree models in 65 out of the 72 experiments with incomplete data.

The algorithm introduced here can be extended in various ways. For instance, the use of other scores could be considered. Also, a Bayesian approach could be followed as in (Friedman, 1998).

We have only focused on the scenario where data is missing completely at random (MCAR). MCAR is the most general setting, and when data is truly MCAR, one could employ simpler and more efficient approaches to learning, such as available-case-analysis. We plan to investigate simpler and less general approaches in the future. Future studies also include the extension of the current algorithm to handle scenarios where unobserved variables are known to influence the observed data. Finally, a more exhaustive comparative analysis including other inference efficient graphical models (such as Naïve Bayes models) will be the focus of the next stage of this study.

Acknowledgements

This work has been supported by the Spanish Ministry of Education and Science, through projects TIN2007-67418-C03-01,02.

References

Bozga, M. and Maler, O. (1999). On the representation of probabilities over structured domains. In *Proceedings of the 11th International Conference on Computer Aided Verification*, pages 261–273. Springer.

Chow, C. K. and Liu, C. N. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467.

Dempster, A. P., Laird, N. M., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Friedman, N. (1997). Learning belief networks in the presence of missing values and hidden variables. In *Proceedings of the Fourteenth International Conference on Machine Learning*.

Friedman, N. (1998). The Bayesian structural EM algorithm. In *Proceedings of the UAI'98 Conference*.

Jaeger, M. (2004). Probabilistic decision graphs - combining verification and AI techniques for probabilistic inference. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12:19–42.

Jaeger, M., Nielsen, J. D., and Silander, T. (2006). Learning probabilistic decision graphs. *International Journal of Approximate Reasoning*, 42(1-2):84–100.

Lauritzen, S. L. (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201.

Nielsen, J. D. and Jaeger, M. (2006). An empirical study of efficiency and accuracy of probabilistic graphical models. In *Proceedings of the Third European Workshop on Probabilistic Graphical Models*, pages 215–222.

Nielsen, J. D., Rumí, R., and Salmerón, A. (2007). El clasificador grafo de decisión probabilístico. Presented at: XXX Congreso Nacional de Estadística e Investigación Operativa. `http://www.ual.es/~dalgaard/publications/seio07.pdf`.

Peña, J. M., Lozano, J. A., and Larrañaga, P. (2000). An improved Bayesian structural EM algorithm for learning Bayesian networks for clustering. *Pattern Recognition Letters*, 21:779–786.

Ramoni, M. and Sebastiani, P. (1997). Learning Bayesian networks from incomplete databases. Technical Report KMI-TR-43, Knowledge Media Institute, The Open University.