The PDG-mixture model for clustering

M. Julia Flores, José A. Gámez, and Jens D. Nielsen

Computing Systems Dept. & SIMD Lab in I^{3}A
University of Castilla-La Mancha, Albacete, Spain.
{julia,jgamez,dalgaard}@dsi.uclm.es

Abstract. Within data mining, clustering can be considered the most important unsupervised learning problem which deals with finding a structure in a collection of unlabeled data. Generally, clustering refers to the process of organizing objects into groups whose members are *similar*. Among clustering approaches, those methods based on probabilistic models have been extensively developed, such as Naïve Bayes (NB) with a latent class (cluster identifier) found via an EM algorithm.
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 commonly used models. In this paper we propose to use a mixture of PDG models in cluster discovery, and an algorithm for automatic induction of the mixture and the models is introduced.
The proposed approach was experimentally evaluated on both synthetic and real-world databases, and the presentation of the results includes a comparison with related techniques. The comparison demonstrates competitive performance of the mixture of PDG models with respect to likelihood. Also, the mixture of PDG models have a tendency to use fewer models (clusters) to represent domains where other models use large amounts of clusters.

Key words: Probabilistic graphical models, clustering, data mining

1 Introduction

The increasing availability of data in our information society has led to the need for valid tools for its modeling and analysis. One core task in data mining is classification. Classification is the process of assigning labels to data instances using a function that takes a unlabeled data-instance as input and outputs a label. Unlike classification (aka supervised classification), which analyzes class-labeled data objects, clustering (aka unsupervised classification) analyzes data objects without consulting a known class label. In general, the class labels are not present in the training data simply because they are not known to begin with. Clustering can be used to generate such labels. The data instances are clustered or grouped based on the principle of maximizing the intraclass similarity and minimizing the interclass similarity. Each formed cluster can be viewed as a class of objects. Clustering can also facilitate taxonomy formation, that is, the

organisation of observations into a hierarchy of classes that group similar events together. Clustering also facilitates knowledge discovery through learning of new concepts that characterize common features or patterns, being used in many fields such as pattern recognition, image analysis and bioinformatics.

Among the different existing approaches, we will focus on the so called model-based methods [1]. Model-based clustering assumes that the data were generated by a specific model and tries to recover the original (generative) model from the data. The model that we recover from the data then defines clusters and can be used to assigns a label (or a set of possible labels) to new unlabeled data instances. EM and COBWEB belongs to this family [1].

Another possible classification on clustering methods uses as parameter the nature of the produced clusters and distinguishes *Hard*, *Soft*, *Hierarchical* and *Probabilistic*. These do not necessarily have to be disjoint sets, for example the Independency Tree [2] clustering is both hierarchical and probabilistic. The Probabilistic Decision Graph (PDG) [3] was originally proposed as an efficient representation of probabilistic transition systems. In this study, we consider the more generalized version of PDGs proposed by [4]. PDGs constitute a class of probabilistic graphical models that can represent some context specific independencies that can not efficiently be captured by other commonly used models.

The performance of the PDG model w.r.t. general probability estimation has previously been studied and results suggest that the model performs competitively when compared to state of the art models[5]. The PDG model has also been successfully applied to supervised classification problems [6] and to the problem of learning from incomplete data[7]. In this paper we extend the application area of PDGs to include also the clustering problem. The motivation for initiating this study was not only the previous successes of the PDG model to related problems such as classification and learning from incomplete data. But also the natural way in which a mixture of PDG models can take advantage of common sub-patterns in different clusters by reusing of parameters. As a result, a mixture of PDG models may provide a more compact model than conventional probabilistic clustering models. Furthermore, if context specific independencies exists within the same cluster, a PDG model may be able to capture this in a single model of this cluster, while other model that does not have this flexibility in representation may need to break the cluster into different clusters conditioning on the context.

2 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.

By $P(\mathbf{X})$ we will denote a joint probability distribution over $\mathbf{X}$, and by $P(\mathbf{Y}|\mathbf{Z})$ for disjoint $\mathbf{Y}$ and $\mathbf{Z}$ the conditional distribution of $\mathbf{Y}$ given $\mathbf{Z}$. To refer the probability of $\mathbf{X} = \mathbf{x}$ we use $P(\mathbf{X} = \mathbf{x})$ or simply $P(\mathbf{x})$. When computing a probability using a model M, we may indicate this by conditioning on the model $P(\mathbf{x}|M)$, however, we will use only $P(\mathbf{x})$ when M is clear from context.

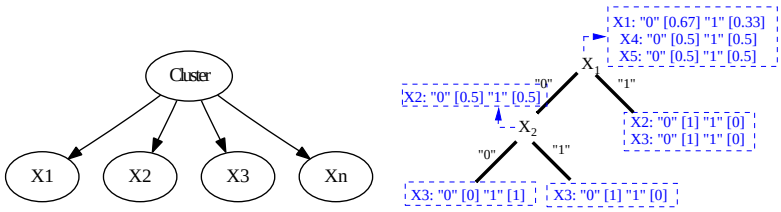
Let $G = \langle \mathbf{V}, \mathbf{E} \rangle$ be a directed graph structure with set of nodes $\mathbf{V} = \{V_1, \ldots, V_n\}$ and set of directed edges $\mathbf{E} \subset \mathbf{V} \times \mathbf{V}$. We will then by $ch_G(V_i)$ and $pa_G(V_i)$ refer the set of children of V_i and parents of V_i respectively in structure G, hence $ch_G(V_i) = \{V_j \in \mathbf{V} : (V_i, V_j) \in \mathbf{E}\}$ and $pa_G(V_i) = \{V_j \in \mathbf{V} : (V_j, V_i) \in \mathbf{E}\}$. When G is clear from context we drop the subscript. 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.

3 Techniques that perform probabilistic clustering

Expectation-Maximisation and Naïve Bayes. In statistical computing, an expectation-maximisation (EM) algorithm [8] is an algorithm for finding maximum likelihood estimates of parameters in probabilistic models, where the model depends on unobserved latent variables. EM is frequently used for data clustering in machine learning and computer vision. EM alternates between performing an expectation (E) step, which computes the expected sufficient statistics by including the latent variables as if they were observed, and a maximization (M) step, which computes the maximum using expected sufficient statistics of the parameters by maximizing the expected likelihood on the observed cases found in the E step. The parameters found on the M step are then used to begin another E step, and the process is repeated.

In a probabilistic clustering task, one often includes in the model a special latent variable C that is never observed. Each states of this C then corresponds to a cluster, and inferring cluster membership is then done by answering queries like $P(C = c_i|\mathbf{X} = \mathbf{x})$. The Naïve Bayes (NB) model for clustering takes this approach, and represents a joint probability distribution that incorporate one strong independence assumptions which often have no bearing in reality, hence are (deliberately) naïve: all the variables are independent given cluster membership. The NB model is a special instance of a Bayesian Network model [9]with the structure shown in Fig. 1.(a).

The independencies that are assumed by the NB model yields the factorisation of the joint probability distribution $P(\mathbf{X}, C)$ over the domain $\mathbf{X}$ of observed variables and cluster variable C: $P(\mathbf{X}, C) = P(C) \prod_{X=\mathbf{X}} P(X|C)$. For a given observation $\mathbf{y}$ of variables $\mathbf{Y} \subseteq \mathbf{X}$, the probability of $\mathbf{y}$ being a member of cluster c_i is $P(C = c_i|\mathbf{Y} = \mathbf{y}) = \frac{1}{P(\mathbf{Y}=\mathbf{y})} P(C = c_i) \prod_{Y \in \mathbf{Y}} P(Y = \mathbf{y}[Y]|C = c_i)$ where $P(\mathbf{Y} = \mathbf{y}) = \sum_{c \in R(C)} P(C = c) \prod_{Y \in \mathbf{Y}} P(Y = \mathbf{y}[Y]|C = c)$.



(a)NB structure (b) A simple Independency Tree

Fig. 1. Two examples of probabilistic structures for clustering.

For learning the parameters in the NB model one needs to reason from incomplete data as no data contains observations for C, and the standard approach is to use EM. For estimating the optimal number of clusters (states of C) a common approach is to use cross-validation.

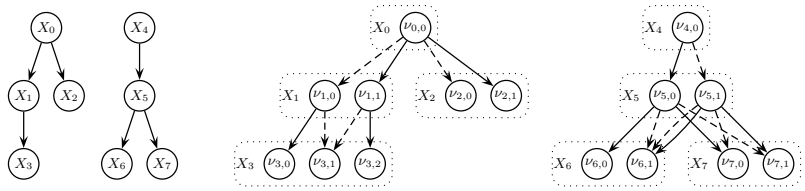
Independency Trees. In [2] the Independency Tree (IT) was presented as a model able to perform clustering and also as an approximate way for factorisation. In Fig. 1(b) we show an example of an IT model. In general, the IT model can be interpreted as an extended probability tree [10] which introduces a new and very important element: a list of probabilistic marginal potentials associated to every node.

Given a leaf-node n in an IT structure, let $\mathbf{X}_n$ be the variables for which a marginal potential is associated with n, then all $\mathbf{X}_n$ are pair-wise marginally independent given the path to n. So, if the distribution for a given variable is shared by all leaves in a sub-tree, it can be stored in the root of that sub-tree for simplicity. For example, in Fig. 1(b) variables $X4$ and $X5$ are independent w.r.t. all the rest, and that is why their distributions appear in the root node.

Then, when one variable appears in a list for a node n it means that this distribution is common for all levels from here to a leaf, including intermediate nodes. On the other hand, there might be distributions that vary depending on the branch. For instance, in Fig. 1.(b) $X2$ distribution is different depending on the path (left or right) taken from the root, that is if $X1 = 0$ or $X1 = 1$.

The intuition underlying this model is based on the idea that inside each cluster the variables are independent. When we have a set of data, groups are defined by common values in certain variables, while the rest of the variables may vary independently. In an IT every cluster will be represented by a complete branch, with an associated factorisation. For the example, three clusters have been found, each one with a probability of $\frac{1}{3}$. If we look at the second branch/cluster it is characterised by $\{X1 = 0, X2 = 1\} + [X3 : 1.0/0.0, X4 : 0.5/0.5, X5 : 0.5/0.5]$[1].

[1] $X1$ to $X5$ are binary, Xi:p1/p2 indicates that $P(Xi = 0) = p1$ and $P(Xi = 1) = p2$.



(a) (b)

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

(c)

Fig. 2. 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).

3.1 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 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 1. A forest F over binary variables $\mathbf{X} = \{X_0, \ldots, X_7\}$ can be seen in Figure 2(a), and a PDG structure over $\mathbf{X}$ w.r.t. F in Figure 2(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 $\mathcal{G} = \langle G, \boldsymbol{\Theta} \rangle$ where G is a PDG structure over $\mathbf{X}$ and $\boldsymbol{\Theta}$ is an instantiation of G. We denote by $\mathbf{p}^{\nu_{i,j}}$ the local distribution assigned to node $\nu_{i,j}$, and by $p_{x_{i,h}}^{\nu_{i,j}}$ 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 $reach_G(X_i, \mathbf{x})$ we denote the unique node representing X_i reached by $\mathbf{x}$ in PDG-structure G.

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

$$P(\mathbf{X} = \mathbf{x}) = \prod_{X_i \in \mathbf{X}} p_{\mathbf{x}[X_i]}^{reach_G(X_i, \mathbf{x})}. \tag{1}$$

Example 2. To instantiate the PDG structure in Fig. 2(b), we assign a local distribution to each node in the structure with the probabilistic interpretation given in Fig. 2(c). We can read some context specific independencies of this table, e.g. X_6 is independent of X_5 only in the context $X_4 = 0$.

4 Mixtures of PDG models

In this section, we will describe our approach to probabilistic clustering using mixtures of PDG models.

In the previous Section 3.1 we introduced the PDG model for representing joint probability distribution over a finite set of discrete random variables. A typical approach to probabilistic clustering is to use a mixture of models. We propose a model that is a mixture of k PDG models by introducing a latent variable with one categorical state for each of the k PDGs. The marginal distribution of the latent variable defines the mixture of the k models. In Example 3 a specific mixture of 2 PDG models is presented.

Example 3. Consider 3 binary random variables X_0, X_1 and X_2. Let the distribution of X_2 be depending on X_0 and X_1, and furthermore, let the specific dependence be governed by an unobserved random variable C such that:

$$P(X_2|C=0) = \begin{cases} P_1(X_2) & \text{if } X_0 \text{ and } X_1 \text{ have even parity,} \\ P_2(X_2) & \text{otherwise.} \end{cases} \tag{2}$$

$$P(X_2|C=1) = \begin{cases} P_3(X_2) & \text{if } X_0 \wedge X_1 \text{ is true} \\ P_4(X_2) & \text{otherwise.} \end{cases} \tag{3}$$

The PDGs in Fig. 3(a) and (b) encodes the relations of Eq. (2) and (3) respectively when solid edges ecode value 0 and dashed encode value 1. For the

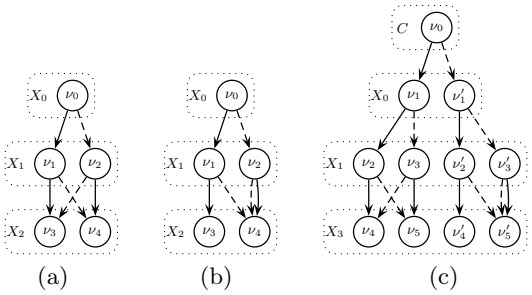


Fig. 3. In all structures, solid edges represent value 0 while dashed edges represent value 1. (a) A PDG encoding the relation in Eq. (2). (b) A PDG encoding the relation in Eq. (3). (c) A PDG that represents a mixture of the two PDG models of (a) and (b) through the latent variable C.

numerical part of the models we have in Fig. 3(a): $\mathbf{p}^{\nu_3} = P_1(X_2)$ and $\mathbf{p}^{\nu_4} = P_2(X_2)$, while in (b): $\mathbf{p}^{\nu_3} = P_3(X_2)$ and $\mathbf{p}^{\nu_4} = P_4(X_2)$. Finally, by introducing the latent variable C in Fig. 3(c) we mix the two models to obtain a PDG model representing the full domain, and prior distribution of C is specified in $\mathbf{p}^{\nu_0}$.

In Ex. 3 we introduced an example of a mixture over two specific PDG models. Please note that the logical expressions as those used here (Eq. (2) and Eq. (3)) demonstrates some of the representation power of PDG models. While most other models has structures that grow exponentially by the number of variables included in such logical expressions, PDGs usually grow only linearly. For the toy-example presented here the difference obviously diminishes.

4.1 Learning mixtures of PDG models

In order to induce a PDG mixture model from data, we will have to establish a strategy for learning both k, a strategy for learning the variable structure to be shared between all component models, and both parameters and local structure of each of the k models. Lastly, the marginal distribution $P(C)$ also needs to be estimated.

Learning PDG models from complete data was addressed in [11], and for the case of incomplete data in [7]. We will combine these two approaches in a new algorithm that learns PDG mixture models.

Learning a common variable structure. The first step of our approach will induce a good structure over the variables to be shared between all k mixture component models. Here we use the approach presented in [11]. A statistical test of independence is used to decide the best organisation of variables. Initially, marginally dependent variables are grouped together. Then, incrementally the a tree is build for each group by inducing PDG models including more and more variables, placing variables that are conditionally independent in different subtrees, where the condition used in the test of independence is defined by a

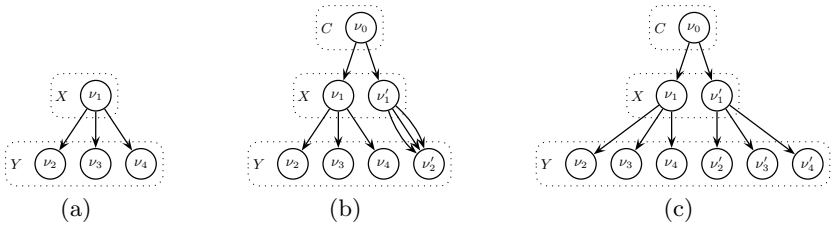


Fig. 4. (a) An inital model learned without latent variable. (b) The initialisation of the mixture by adding latent variable C to the model and extending the model with a new component by method 1. (c) Equivalent to (b) but method 2 is used for extending with a new component.

partition of the state space induced by the current PDG structure. The reader is referred to [11] for details.

Introducing the mixture. Once having learned an initial structure over the variables (as described above), we continue by adding a latent variable C to the model with $k = 1$ states, $R(C) = \{c_0\}$. One outgoing edge with label c_0 is connected to each of the roots of the PDG structure induced in the first step. We then extend C by adding one new state (incrementing k by one) and optimise structure and parameters by the structural-EM. If the likelihood of a separate hold-out dataset was increased by incrementing k, we loop and increment k once more.

Incrementing k. We consider two different strategies for introducing a new latent state.

1. We can extend the model with a new parameter node for each non-latent variable in the model. These new nodes are connected linearly without further bifurcations, and the edge labelled c_{k+1} is connected to the new root.
2. We can extend the model with a copy of the subtree(s) corresponding to an existing latent state c_i, $1 < i \leq k$. We choose the latent state with highest prior probability.

After extending the latent variable with a new variable state, splitting and merging is performed to refine the model. In Ex. 4 we give examples of the two methods listed above.

Example 4. Fig. 4(a) shows an initial PDG structure over two random variables X and Y. In Fig. 4(b) the latent variable C is added and a new latent state is initialise by method 1, that is, using a single new node for each variable. In Fig. 4(c) we show the structure resulting from initialising the new latent state by method 2, that is, using a copy of one of the existing subtrees.

When incrementing k, the new marginal probability $p^{\nu_0}_{c_{k+1}} = P(C = c_{k+1})$ is initialised to $\frac{1}{k+1}$, and the existing probabilities of the old k states $p^{\nu}_i : 1 < i \leq$

$k$are scaled accordingly by a factor $\frac{k}{k+1}$. For the new parameter nodes created for each of the variables in the domain, the initialisation depends on the method we used for creating them (the two described above). When using method 1, the new parameter node for variable X is initialised by either using the relative frequency (empirical marginal distribution) $\hat{P}(X)$. When using method 2, the parameters are copied from the relevant sub-tree. We then draw a data instance d at random from the data set used for training, and use this as the "centre" of the new cluster, hence, we want to increase the probability assigned to d given the new cluster, $P(\mathbf{X} = d|C = c_{k+1})$. This achieved by tuning parameter $\mathbf{p}^{\nu}$ where ν represents X and $reach(X, \{d, c_{k+1}\}) = \nu$ as follows:

$$p^{\nu}_{x_i} \leftarrow \begin{cases} \frac{1.1+0.1p^{\nu}_{x_i}}{1.1} & \text{if } d[X] = x_i, \text{ and} \\ \frac{0.1p^{\nu}_{x_i}}{1.1} & \text{otherwise.} \end{cases} \tag{4}$$

We finally arrive at framework presented as Algorithm 1. Please note that in line 3 we optimise the BIC score of the model using the score+search method presented in [11]. In this method the score of a model is optimised by iteratively splitting and merging parameternodes in a given structure. In lines 4 and 8 we use the structural-EM approach of [7] optimising the expected BIC score. Basically, this approach uses the same operators (split and merge of parameternodes) but uses expected score instead of actual score as the actual score is not tractable to compute in the precense of missing values. Finally, in lines 4 and 7 we need to choose either method 1 or 2 for extending the latent variable, yielding two different versions of the algorithm.

Algorithm 1

1: **procedure** LearnMixtureOfPDGs($\mathbf{D}$)
2: Divide $\mathbf{D}$ into $\mathbf{D}_v$ being random sample of 10% of $\mathbf{D}$ and $\mathbf{D}_t = \mathbf{D} \setminus \mathbf{D}_v$.
3: Learn PDG $\mathcal{G}$ from $\mathbf{D}_t$.
4: Initialise mixture $\mathcal{G}^0$ from $\mathcal{G}$, and optimise $\mathcal{G}^0$ by structural-EM and $\mathbf{D}_t$.
5: $k \leftarrow 0$.
6: **repeat**
7: $\mathcal{G}^{k+1} \leftarrow \{\mathcal{G}^k$ extended with 1 latent state$\}$.
8: Optimise $\mathcal{G}^{k+1}$ by structural-EM and $\mathbf{D}_t$.
9: $k \leftarrow k + 1$.
10: **until** $P(\mathbf{D}_v|\mathcal{G}^k) < P(\mathbf{D}_v|\mathcal{G}^{k-1})$
11: **return** $\mathcal{G}^{k-1}$.

5 Empirical evaluation

In this section we perform a comparative analysis based on experimentation on 10 datasets. The *Exclusive* dataset is a dataset that is artificially generated from a boolean formula containing 5 boolean variables. Three of the variables are dependent such that one of them always assumes the value true while the other two assumes false. The last two variables are independent. The *TicTacToe* dataset is taken from the USI repository[12], and encodes the complete set of possible board configurations at the end of tic-tac-toe games. The *Greenhouse*

Table 1. Datasets used in the empirical evaluation.

Id	Name	# Vars	size train	size test
1	Exclusive	5	48	24
2	TicTacToe	9	641	317
3A	Greenhouse-A	8	830	410
3B	Greenhouse-B	17	981	484
3C	Greenhouse-C	33	883	435
3D	Greenhouse-D	6	1026	506
3E	Greenhouse-E	6	981	484
4A	Sheep-A	24	2068	1019
4B	Sheep-B	23	2068	1019
5	PDG-mixture	3	1000	500
6	NB	5	1000	500
7	IT	6	1000	500

datasets belong to data obtained when analysing an important economical factor in the south-east of Spain, greenhousing production at Almería. Some of these datasets were studied in [13] by using Bayesian networks. The *Sheep* datasets are historical data of sheep and has previously been used to analyse genetic merit for milk production[14]. The *PDG-mixture* data is a dataset artificially generated from a mixture of 3 PDG models. The *NB* dataset has been sampled from a NB model with 3 latent states and 5 observable variables. Finally, the *IT* dataset was sampled from a IT model defining 3 clusters over 6 random variables. A brief description of the datasets can be found in table 1.

For learning IT models from the databases in Tab. 1, we have used the method presented in [2]. NB models was learnt using the Weka[15] system, using default settings of EM. When establishing the number of clusters, Weka uses cross-validation. The mixture of PDG models was learned using the algorithm discussed in Section 4.1.

In Tab. 2 we have listed log-likelihood (LL) for the learnt models measured over the test data which is a special separate dataset only brought in after the learning process to assess the quality of the learnt model. In Tab. 2 we also list the number of clusters (C) identified by the models and the size (S) of the models measured in the number of independent parameters defined by the respective models.

6 Discussion

We have compared the four algorithms over the ten datasets by using non-parametric Wilcoxon paired Signed-Ranks Test (α=0.05). From this statistical study we are in a position to say that the four models perform equally well in terms of log-likelihood but that some significant differences appear in the other two studied parameters (size and clusters). Clearly, IT produces a greater number of clusters than NB and both PDGs, but no significant difference is obtained when comparing pairwise NB, mixt-PDG-1 and mixt-PDG-2. However, the average number of clusters identified are 6.92 for NB, 5.33 for mixt-PDG-1 and 4.67 for mixt-PDG-2, which we find quite remarkable. With respect to size, there is no surprise, and the simplicity of NB makes it statistically superior in

Table 2. Results of Independency Tree (IT) learning, Naïve Bayes (NB) learning and the mixture of PDG models, the mixt-PDG-1 and mixt-PDG-2 columns refers to variation 1 and 2 of Alg. 1, respectively. The table contain log-likelihood (LL) of the learnt model measured over a separate test dataset, the number of clusters identified by the model (C), and the size of the model (S) measured in the number of independent parameters the model contains.

	IT			NB			mixt-PDG-1			mixt-PDG-2		
Id	LL	C	S	LL	C	S	LL	C	S	LL	C	S
1	-2.5481	3	8	-2.7920	3	17	-2.6942	4	16	-2.6383	4	17
2	-9.3111	5	78	-9.4090	2	37	-9.2862	7	308	-9.4228	3	396
3A	-6.3433	18	392	-6.3760	6	221	-6.3487	4	326	-6.2743	3	501
3B	-7.5005	7	214	-7.3797	4	151	-7.4285	5	332	-7.4957	3	604
3C	-17.0196	16	1270	-16.7899	10	1059	-16.8512	10	1345	-16.8505	11	6905
3D	-5.0027	6	96	-5.0174	5	124	-5.0191	3	166	-5.0609	3	262
3E	-5.6823	8	152	-5.6948	5	134	-5.7372	4	224	-5.7250	4	378
4A	-19.7172	63	3526	-19.7393	13	948	-19.3950	10	2163	-18.8837	6	7367
4B	-18.8637	53	2835	-18.4460	24	1679	-19.1973	6	1736	-18.8946	8	7852
5	-2.7918	7	22	-2.8033	7	48	-2.7976	3	32	-2.7943	3	40
6	-2.3400	4	14	-2.3440	2	11	-2.7526	4	31	-2.7528	5	46
7	-3.2364	3	12	-3.2768	2	13	-3.2541	4	32	-3.2592	3	29

this parameter with respect to PDG models, and almost statistically superior (p-value = 0.0639) with respect to IT (average number of parameters is 370 (NB) vs 718 (IT)). There exist also statistically difference between PDG-2 and the other three models, indicating that PDG-2 is the model needing more parameters. Finally, no statistical difference appear between PDG-1 and IT, although PDG-1 needs 23% fewer parameters than IT.

Having done this general comparison, we continue with the analysis of the behaviour of the algorithms in some particularly interesting cases. Thus, when investigating the number of clusters identified by the different models, one interesting behaviour is evident. Both the IT and NB models uses many more clusters to model the *Sheep* domains than does the mixture of PDG models. This is interesting with respect to the practical use of the model for clustering, where usually a smaller number of clusters is preferable as it may be easier to assign meaningful semantics to each cluster. The two databases *Sheep-A* and *Sheep-B* differs only in that *Sheep-A* includes a variable that represents the breeding value of the sheep, while *Sheep-B* excludes this variable. Following domain experts (the shepherds), this variable can naturally be used as a classification of the given sheep into 4 different classes (the 4 possible values of this variable). For *Sheep-B* we see that the IT model identifies 53 clusters, NB identifies 24 while our mixt-PDG-1 and mixt-PDG-2 approaches identify only 6 and 8 clusters respectively, though with a somewhat lower score in likelihood.

Finally, investigating the datasets 5, 6 and 7, sampled from a mixture of PDGs, a NB and an IT respectively, we find one surprise: IT scores higher likelihood than both PDGs and NBs on all three datasets. We expected each model to provide the closest representation of data sampled from that exact model type. However, when investigating the number of clusters identified we see that for dataset 5 only PDG approaches identifies the correct number of clusters. For dataset 6 non of the methods identifies the correct number of clusters, and finally for dataset 7 both IT and mixt-PDG-2.

7 Conclusion

In this paper we have shown how the PDG model can be used in data clustering by extending the model with a latent variable, yielding a mixture of PDG models. We have shown how to induce such models from data using EM and score based model-selection. Using two variations over the framework for induction of PDG mixtures, we have shown our proposal to be competitive with IT and NB model approaches, the latter being a standard technique in probabilistic clustering. On average, the PDG based approach identifies fewer numbers of clusters than both IT and NB, though non of the differences are statistically significant.

References

1. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. 2nd ed. Morgan Kaufmann (2006)
2. Flores, M.J., Gámez, J.A., Moral, S.: The independency tree model: A new approach for clustering and factorisation. In: Proc. of the 3rd PGM workshop. (2006) 83–90
3. Bozga, M., Maler, O.: On the representation of probabilities over structured domains. In: Proc. of the 11th Int. Conf. on Computer Aided Verification, Springer (1999) 261–273
4. Jaeger, M.: Probabilistic decision graphs - combining verification and AI techniques for probabilistic inference. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **12** (2004) 19–42
5. Nielsen, J.D., Jaeger, M.: An empirical study of efficiency and accuracy of probabilistic graphical models. In: Proc. of the 3rd PGM workshop. (2006) 215–222
6. Nielsen, J.D., Rumí, R., Salmerón, A.: Supervised classification using probabilistic decision graphs. Computational Statistics & Data Analysis **53**(4) (2009) 1299 – 1311
7. Nielsen, J.D., Rumí, R., Salmerón, A.: Structural-EM for learning PDG models from incomplete data. In: Proc. of the 4th PGM workshop. (2008) 217–224
8. Dempster, A.P., Laird, N.M., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B **39**(1) (1977) 1–38
9. Jensen, F.V.: Bayesian Networks and Decision Graphs. Springer (2001)
10. Salmerón, A., Cano, A., Moral, S.: Importance sampling in bayesian networks using probability trees. Computational Statistics & Data Analysis **34** (2000) 387–413
11. Jaeger, M., Nielsen, J.D., Silander, T.: Learning probabilistic decision graphs. International Journal of Approximate Reasoning **42**(1-2) (2006) 84–100
12. Newman, D., Hettich, S., Blake, C., Merz, C.: UCI repository of machine learning databases: `http://www.ics.uci.edu/~mlearn/MLRepository.html` (1998)
13. Céspedes, A., Rumí, R., Salmerón, A., Soler, F.: Analysis of the agricultural sector in the west-area of Almería by using bayesian networks (in spanish). In: Proc. of the 27th Spanish Conf. on Statistics & Operations Research. (2003) 3438–3455
14. Flores, M.J., Gámez, J.A., Mateo, J.L.: Mining the ESROM: A study of breeding value classification in manchego sheep by means of attribute selection and construction. Computers and Electronics in Agriculture **60**(2) (2008) 167–177
15. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. 2nd edn. Morgan Kaufmann, San Francisco (2005)