Supervised Classification Using Hybrid Probabilistic Decision Graphs

Antonio Fernández[1], Rafael Rumí[1], José del Sagrado[2], and Antonio Salmerón[1]

[1] Dept. of Mathematics, University of Almería,
Ctra. Sacramento s/n, 04120 Almería, Spain
[2] Dept. of Computer Science, University of Almería,
Ctra. Sacramento s/n, 04120 Almería, Spain
{afalvarez,rrumi,jsagrado,antonio.salmeron}@ual.es
http://elvira.ual.es/programo

Abstract. In this paper we analyse the use of probabilistic decision graphs in supervised classification problems. We enhance existing models with the ability of operating in hybrid domains, where discrete and continuous variables coexist. Our proposal is based in the use of mixtures of truncated basis functions. We first introduce a new type of probabilistic graphical model, namely probabilistic decision graphs with mixture of truncated basis functions distribution, and then present an initial experimental evaluation where our proposal is compared with state-of-the-art Bayesian classifiers, showing a promising behaviour.

Keywords: Supervised classification, Probabilistic decision graphs, Mixtures of truncated basis functions, Mixtures of polynomials, Mixtures of truncated exponentials.

1 Introduction

The Probabilistic Decision Graph (PDG) model was introduced in [2] as an efficient representation of probabilistic transition systems. In this study, we consider the more general version of PDGs proposed in [8].

PDGs are probabilistic graphical models that can represent some context specific independencies that are not efficiently captured by conventional graphical models as Bayesian Networks (BNs). In addition, probabilistic inference can be carried out directly over the PDG structure in a time linear in the size of the PDG model.

PDGs have mainly been studied as representations of joint distributions over *discrete* random variables, showing a competitive performance when compared to BNs and Latent class Naive BNs [15]. The discrete PDG model has also been successfully applied to supervised classification problems [16] and unsupervised clustering [4].

The need to handle discrete and continuous variables simultaneously has motivated the development of new probabilistic graphical models incorporating that feature, mainly hybrid Bayesian networks [3, 14, 9, 10, 17, 18]. Also, PDGs have been recently extended in order to allow the inclusion of *continuous* variables when the joint distribution of the model is a *conditional Gaussian* (CG) [6].

In this paper we extend the PDG classifiers introduced in [16] in order to incorporate continuous variables. We rely on the *Mixture of truncated basis functions* (MoTBFs) model [10]. Unlike CG models, MoTBFs do not rely on the normality assumption, and do not impose any restriction on the structure of the conditional distributions involved in the PDG, so that discrete variables can be conditioned on continuous ones and vice versa.

2 Notation and Preliminaries

We will use uppercase letters to denote random variables, and boldfaced uppercase letters to denote random vectors, e.g. $\mathbf{X} = \{X_0, X_1, \dots, X_N\}$. By $R(X)$ we denote the set of possible states of variable X, and similarly for random vectors, $R(\mathbf{X}) = \times_{X_i \in \mathbf{X}} R(X_i)$. By lowercase letters x (or $\mathbf{x}$) we denote some element of $R(X)$ (or $R(\mathbf{X})$). When $\mathbf{x} \in R(\mathbf{X})$ and $\mathbf{Y} \subseteq \mathbf{X}$, we denote by $\mathbf{x}[\mathbf{Y}]$ the projection of $\mathbf{x}$ onto coordinates $\mathbf{Y}$. Throughout this document we will consider a set $\mathbf{W}$ of discrete variables and a set $\mathbf{Z}$ of continuous variables, and we will use $\mathbf{X} = \mathbf{W} \cup \mathbf{Z}$.

In what concerns the structure of the PDG model, we will make use of the following notation. Let G be a directed graph over nodes $\mathbf{V}$. Let $\nu \in \mathbf{V}$, we then denote by $pa_G(\nu)$ the set of parents of node ν in G, by $ch_G(\nu)$ the set of children of ν in G, by $de_G(\nu)$ the set of descendants of ν in G, that is recursively defined as $de_G(\nu) = \{\nu' : \nu' \in ch_G(\nu) \vee [\nu' \in ch_G(\nu'') \wedge \nu'' \in de_G(\nu)]\}$, and we use as shorthand notation $de^*_G(\nu) = de_G(\nu) \cup \nu$. By $an_G(\nu)$ we understand the set of ancestors (or predecessors) of ν in G, that is recursively defined as $an_G(\nu) = \{\nu' : \nu' \in pa_G(\nu) \vee [\nu' \in pa_G(\nu'') \wedge \nu'' \in an_G(\nu)]\}$.

2.1 Discrete PDGs

The discrete PDG model was introduced in [8] as representation of joint distributions over discrete random variables. The structure is formally defined as follows:

Definition 1 (PDG Structure [8]). *Let F be a forest of directed tree structures over a set of discrete random variables $\mathbf{W}$. A* PDG structure $G = \langle \mathbf{V}, \mathbf{E} \rangle$ *for $\mathbf{W}$ w.r.t. F is a set of* rooted *DAGs, such that:*

1. *Each node $\nu \in \mathbf{V}$ is labelled with exactly one $W \in \mathbf{W}$. By $\mathbf{V}_W$, we will refer to the set of all nodes in a PDG structure labelled with the same variable W. For every variable W, $\mathbf{V}_W \neq \emptyset$, we will say that ν represents W when $\nu \in \mathbf{V}_W$.*
2. *For each node $\nu \in \mathbf{V}_W$, each possible state $w \in R(W)$ and each successor $Y \in ch_F(W)$ there exists exactly* one *edge labelled with w from ν to some node ν' representing Y. Let $Y \in ch_F(W)$, $\nu \in \mathbf{V}_W$ and $w \in R(W)$. By $succ(\nu, Y, w)$ we will then refer to the unique node $\nu' \in \mathbf{V}_Y$ that is reached from ν by an edge with label w.*

An example, taken from [6], of a PDG structure and its corresponding variable forest are depicted in Figs. 1(b) and (a) respectively. A PDG structure consists of two layers, one for variable and one for nodes. The variable layer conforms a directed forest over the variables F and the node layer is a one-root directed acyclic graph structure. Children, parents, descendants or ancestors of a variable in a PDG structure G, are located according to structure F. So, using Fig. 1(b) as an example, on the variable layer we have: $pa_G(W_1) = \{W_0\}$, $ch_G(W_0) = \{W_1, W_2\}$, $de_G(W_0) = \{W_1, W_2, W_3\}$ and $an_G(W_3) = \{W_1, W_0\}$. On the node layer, we have: $succ(\nu_0, W_1, 0) = \nu_1$, $succ(\nu_0, W_2, 1) = \nu_4$ and $succ(\nu_1, W_3, 0) = \nu_6$.

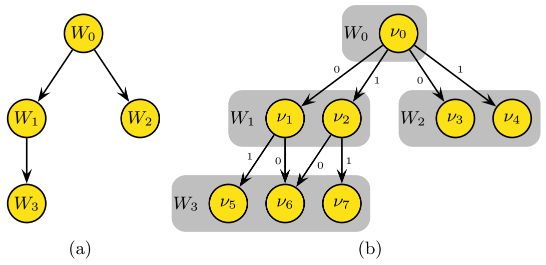


Fig. 1. (a) A forest structure F formed by a single tree over binary variables $\mathbf{W} = \{W_0, W_1, W_2, W_3\}$. (b) A PDG structure over $\mathbf{W}$ with underlying forest F.

A PDG structure G is *instantiated* by assigning a real function $f^\nu : R(W_i) \to \mathbb{R}_0^+$, with $\nu \in \mathbf{V}_{W_i}$ to every node ν in the structure. It represents the global real function $f_G : R(\mathbf{W}) \to \mathbb{R}_0^+$, recursively defined as follows:

$$f_G^\nu(\mathbf{w}) := f^\nu(\mathbf{w}[W]) \prod_{Y \in ch_F(W)} f_G^{succ(\nu, Y, \mathbf{w}[W])}(\mathbf{w}), \tag{1}$$

for all $\mathbf{w} \in R(\mathbf{W})$. f_G is then defined on $R(\mathbf{W})$ as:

$$f_G(\mathbf{w}) := \prod_{\nu : \nu \text{ is a root}} f_G^\nu(\mathbf{w}). \tag{2}$$

Equation (1) defines a factorisation with one factor f^ν for each $W \in \mathbf{W}$. The set of nodes in a PDG structure associated with a given element $\mathbf{w} \in R(\mathbf{W})$ is characterised by function *reach*.

Definition 2 (Reach). *A node ν representing variable W_i in G is* reached *by $\mathbf{w} \in R(\mathbf{W})$ if*

1. *ν is a root in G, or*
2. *$W_j = pa_F(W_i)$, node ν' representing variable W_j is* reached *by $\mathbf{w}$ and $\nu = succ(\nu', W_i, \mathbf{w}[W_j])$.*

By $reach_G(W_i, \mathbf{w})$ we denote the unique node representing W_i reached by $\mathbf{w}$ in PDG structure G.

As an example [6], consider again the PDG structure of Fig. 1(b) and let $\mathbf{w} = \{W_0 = 0, W_1 = 1, W_2 = 1, W_3 = 1\}$. Then $reach_G(W_0, \mathbf{w}) = \nu_0$, $reach_G(W_1, \mathbf{w}) = \nu_1$, $reach_G(W_2, \mathbf{w}) = \nu_3$ and $reach_G(W_3, \mathbf{w}) = \nu_5$. Function f_G in (2) can be reformulated as

$$f_G(\mathbf{w}) := \prod_{W_i \in \mathbf{W}} f^{reach_G(W_i, \mathbf{w})}(\mathbf{w}[W_i]) \,. \tag{3}$$

When all the local functions f^ν in an instantiated PDG structure G over $\mathbf{W}$ are probability distributions, f_G defines a joint multinomial probability distribution over $\mathbf{W}$ [8]. In fact, f^ν_G in (1) defines a multinomial distribution over variables $W \cup ch_F(W)$. We will refer to such instantiated PDG structures as PDG models.

Definition 3 (PDG model [8]). *A PDG model $\mathcal{G}$ is a pair $\mathcal{G} = \langle G, \theta \rangle$, where $G = \langle \mathbf{V}, \mathbf{E} \rangle$ is a valid PDG structure (Definition 1) over some set $\mathbf{W}$ of discrete random variables and $\theta = \{f^\nu : \nu \in \mathbf{V}\}$ is a set of real functions, each of which defines a discrete probability distribution.*

Example 1 (taken from [6]). Consider the PDG structure in Fig. 1. It encodes a factorisation of the joint distribution of $\mathbf{W} = \{W_0, W_1, W_2, W_3\}$, with $f^{\nu_0} = P(W_0)$, $f^{\nu_1} = P(W_1|W_0 = 0)$, $f^{\nu_2} = P(W_1|W_0 = 1)$, $f^{\nu_3} = P(W_2|W_0 = 0)$, $f^{\nu_4} = P(W_2|W_0 = 1)$, $f^{\nu_5} = P(W_3|W_0 = 0, W_1 = 1)$, $f^{\nu_6} = P(W_3|W_1 = 0, \{W_0 = 0 \vee W_0 = 1\})$, $f^{\nu_7} = P(W_3|W_0 = 1, W_1 = 1)$.

The PDG structure plus the set of conditional distributions given above constitute a PDG model over the set of variables $\mathbf{W} = \{W_0, W_1, W_2, W_3\}$. Assume that we want to evaluate the PDG model for a given configuration of $\mathbf{W}$, for instance, $(0, 1, 1, 1)$. According to (1), the returned value is

$$\begin{aligned} f_G(0,1,1,1) &= f^{\nu_0}(0) f^{\nu_1}(1) f^{\nu_3}(1) f^{\nu_5}(1) \\ &= P(W_0 = 0) P(W_1 = 1|W_0 = 0) P(W_2 = 1|W_0 = 0) \\ &\quad\; P(W_3 = 1|W_0 = 0, W_1 = 1). \end{aligned}$$

2.2 Conditional Gaussian PDGs

The first attempt to include continuous variables in PDGs came along with the so-called *conditional Gaussian PDGs* [6]. These models represent joint distributions of discrete and continuous variables simultaneously, conforming a conditional Gaussian (CG) distribution [12]. It means that the joint over the continuous variables is assumed to be a mixture of multivariate Gaussians, and the joint over the discrete variables is a multinomial.

Formally, a CG-PDG is a PDG with forest F where variables can be discrete and continuous. Discrete variables are treated as in PDGs, and the continuous ones follow the next requirements:

- Every continuous variable $Z \in \mathbf{Z}$ is only allowed to have continuous children in F.
- A node ν representing a continuous variable $Z \in \mathbf{Z}$ has exactly one outgoing edge for each child of Z in F.
- A node ν representing $Z \in \mathbf{Z}$ with predecessors in F $\{Z_1, \ldots, Z_n\}$ defines the conditional density $f^\nu = \mathcal{N}(z; \alpha^\nu + \sum_{i=1}^n \beta_i^\nu z_i, \sigma^{2^\nu})$.

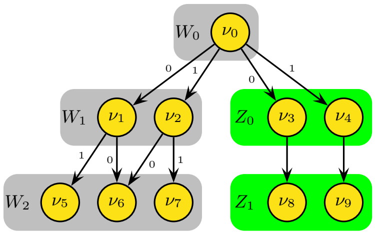


Fig. 2. An example of a structure of a CG-PDG, with discrete variables W_0, W_1, W_2 and continuous variables Z_0 and Z_1

Example 2. The structure depicted in Fig. 2 is compatible with a CG-PDG with discrete variables W_0, W_1, W_2 and continuous variables Z_0 and Z_1. The structure is instantiated as described in Table 1. Note that nodes corresponding to discrete variables contain probability tables, while nodes corresponding to continuous variables have densities instead. For instance, $f^{\nu_3} = \rho(Z_0|W_0 = 0)$ is a Gaussian density with fixed mean and variance. If a continuous variable has at least one continuous predecessor, then the density in each of its nodes is also Gaussian with fixed variance, but the mean is not constant, but rather a linear function of the continuous predecessors. That is the case of nodes ν_8 and ν_9.

Table 1. Instantiation of the structure in Fig. 2

$f^{\nu_0} = P(W_0)$	$f^{\nu_5} = P(W_2 \vert W_0 = 0, W_1 = 1)$
$f^{\nu_1} = P(W_1 \vert W_0 = 0)$	$f^{\nu_6} = P(W_2 \vert W_1 = 0)$
$f^{\nu_2} = P(W_1 \vert W_0 = 1)$	$f^{\nu_7} = P(W_2 \vert W_0 = 1, W_1 = 1)$
$f^{\nu_3} = \rho(Z_0 \vert W_0 = 0)$	$f^{\nu_8} = \rho(Z_1 \vert Z_0, W_0 = 0)$
$f^{\nu_4} = \rho(Z_0 \vert W_0 = 1)$	$f^{\nu_9} = \rho(Z_1 \vert Z_0, W_0 = 1)$

2.3 Mixtures of Truncated Basis Functions

The MoTBF framework is based on the abstract notion of real-valued *basis functions* $\psi(\cdot)$, which includes both polynomial and exponential functions as special cases. Let X be a continuous variable with domain $R(X) \subseteq \mathbb{R}$ and let $\psi_i : \mathbb{R} \rightarrow \mathbb{R}$, for $i = 0, \ldots, k$, define a collection of real basis functions. We say that a function $g_k : R(X) \mapsto \mathbb{R}_0^+$ is an MoTBF potential of level k wrt. $\Psi = \{\psi_0, \psi_1, \ldots, \psi_k\}$ if g_k can be written as [10]

$$g_k(x) = \sum_{i=0}^{k} a_i \, \psi_i \, (x) \, , \tag{4}$$

where a_i are real numbers. The potential is a density if $\int_{R(X)} g_k(x) \, dx = 1$.

Example 3. By letting the basis functions correspond to polynomial functions, $\psi_i(x) = x^i$ for $i = 0, 1, \ldots$, the MoTBF model reduces to an MOP model [18] for univariate distributions. Similarly, if we define the basis functions as $\psi_i(x) = \{1, \exp(-x), \exp(x), \exp(-2x), \exp(2x), \ldots\}$, the MoTBF model corresponds to an MTE model [14] with the exception that the parameters in the exponential functions are fixed.

In a conditional MoTBF density, the influence a set of continuous parent variables $\mathbf{Z}$ has on their child variable X is encoded only through the partitioning of the domain of $\mathbf{Z}$, denoted as $R(\boldsymbol{Z})$, into hyper-cubes, and not directly in the functional form of $g_k(x|\mathbf{z})$ inside each hyper-cube. More precisely, for a partitioning $\mathcal{P} = \{R(\boldsymbol{Z})^1, \ldots, R(\boldsymbol{Z})^m\}$ of $R(\boldsymbol{Z})$, the conditional MoTBF is defined for $\mathbf{z} \in R(\boldsymbol{Z})^j$, $1 \leq j \leq m$, as

$$g_k^{(j)}(x|\mathbf{z} \in R(\boldsymbol{Z})^j) = \sum_{i=0}^{k} a_i^{(j)} \, \psi_i^{(j)}(x). \tag{5}$$

Similarily, MoTBFs can be defined for discrete variables, in which case each potential value $g(x)$ represents the value $P(X = x)$ with $\sum_x g(x) = 1$. Conditional distributions of discrete variables given continuous and/or discrete variables can be defined analogously to (5). See [10, 11] for more details.

3 Hybrid PDGs Based on MoTBFs

CG-PDGs have two limitations. One is the normality assumption, that may not hold in applications with real data. The other one is the structural restriction that forbids discrete variables to have continuous parents in the structure. For instance, the structure in Fig. 3 is not valid for a CG-PDG since discrete variable W_1 has a continuous predecessor, Z_0.

Our proposal to sidestep the above-mentioned restrictions is to adopt the MoTBF framework (see Sect. 2.3) within the PDG model. The formal definition is as follows.

Definition 4 (MoTBF-PDG). *We define an* MoTBF-PDG *as a PDG with forest F where variables can be discrete and continuous. Discrete variables are treated as in PDGs, and the continuous ones follow the next requirements:*

– *Continuous variables are allowed to have discrete and continuous successors and predecessors.*

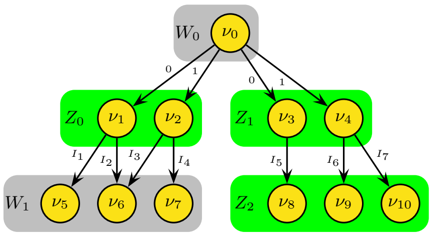


Fig. 3. An example of a PDG structure that is not compatible with a CG-PDG, since discrete variable W_1 has a continuous predecessor, Z_0. Labels $I_1, \dots, I_7$ indicate intervals of the domain of the corresponding continuous variable.

- *A node ν representing $Z \in \mathbf{Z}$ can have* one or more *outgoing edges for each Z_i child of Z in F. Each edge represents a interval the domain of Z, and all of them constitute a* partition *of it.*
- *A node ν representing $Z \in \mathbf{Z}$ with continuous predecessors $\{Z_1, \dots, Z_n\}$ defines an MoTBF density $f^\nu(z)$* conditional *on the branch that leads from the root to ν.*

Example 4. Consider the PDG structure in Fig. 3. Assume W_0 and W_1 are binary variables and Z_0, Z_1 and Z_2 are continuous variables with domain $[0,1]$. An instantiation of the PDG structure is given in Table 2, where the labels $I_1, \dots, I_7$ are, respectively, intervals $[0, 0.5)$, $[0.5, 1]$, $[0.5, 1]$, $[0, 0.5)$, $[0, 1]$, $[0, 0.3)$, $[0.3, 1]$.

Table 2. An instantiation of the PDG structure in Fig. 3. Potentials denoted as ρ correspond to conditional MoTBF densities as in (5).

$f^{\nu_0} = P(W_0)$	$f^{\nu_5} = P(W_1 \mid W_0 = 0, Z_0 \in [0, 0.5))$
$f^{\nu_1} = \rho(Z_0 \mid W_0 = 0)$	$f^{\nu_6} = P(W_1 \mid Z_0 \in [0.5, 1])$
$f^{\nu_2} = \rho(Z_0 \mid W_0 = 1)$	$f^{\nu_7} = P(W_1 \mid W_0 = 1, Z_0 \in [0, 0.5))$
$f^{\nu_3} = \rho(Z_1 \mid W_0 = 0)$	$f^{\nu_8} = \rho(Z_2 \mid W_0 = 0)$
$f^{\nu_4} = \rho(Z_1 \mid W_0 = 1)$	$f^{\nu_9} = \rho(Z_2 \mid W_0 = 1, Z_1 \in [0, 0.3))$
	$f^{\nu_{10}} = \rho(Z_2 \mid W_0 = 1, Z_1 \in [0.3, 1])$

The next proposition shows that an MoTBF-PDG actually represents a joint distribution of class MoTBF over the variables it contains.

Proposition 1. *Let G be a MoTBF-PDG over variables $\mathbf{X} = \mathbf{W} \cup \mathbf{Z}$. Function*

$$f_G(\mathbf{x}) = \prod_{X \in \mathbf{X}} f^{reach_G(X, \mathbf{x})}(\mathbf{x}[X])$$

represents an MoTBF distribution over $\mathbf{X}$.

Proof. According to (3), $f^{reach_G(X,\mathbf{x})}$ is the function stored in the unique parameter node ν of variable X that is reached by the path in G determined by configuration $\mathbf{X} = \mathbf{x}$. Let us index the variables in $\mathbf{X}$ as $X_1, \ldots, X_n$ and denote by $\nu_{\mathbf{x}[X_i]}$ the unique parameter node of variable X_i that is reached by the path in G determined by configuration $\mathbf{X} = \mathbf{x}$. Then,

$$\begin{aligned} f_G(\mathbf{x}) &= \prod_{i=1}^{n} f^{reach_G(X_i,\mathbf{x})}(\mathbf{x}[X_i]) = \prod_{i=1}^{n} f^{\nu_{\mathbf{x}[X_i]}}(\mathbf{x}[X_i]) \\ &= \prod_{i=1}^{n} f^{\nu_{\mathbf{x}[X_i]}}(\mathbf{x}[X_i]|\mathbf{x}[X_1, \ldots, X_{i-1}]) \end{aligned} \tag{6}$$

where, according to Definition 4, $f^{\nu_{\mathbf{x}[X_i]}}$ is a conditional probability function of X_i given the configuration in the branch upwards the root. Furthermore, also according to Definition 4, $f^{\nu_{\mathbf{x}[X_i]}}$ is of class MoTBF. As the product of MoTBF functions is known to be an MoTBF as well (see [10]) we can conclude, by applying the chain rule, that the factorisation in (6) is a joint MoTBF distribution over $X_1, \ldots, X_n$, i.e. over $\mathbf{X}$. □

In the next section we will study the problem of supervised classification and how MoTBF-PDGs can be used in that context.

4 PDG Classifiers

A classification problem can be described in terms of a set of *feature variables* $\mathbf{X} = \{X_1, \ldots, X_n\}$, that describes an individual, and a *class variable*, C, that indicates the class to which that individual belongs. A *classifier* is a model oriented to predict the value of variable C given that the values of the features $\mathbf{X}$ are known. If the joint probability distribution of C and $\mathbf{X}$ is known, it can be used to solve the classification problem by assigning to any individual with observed features $x_1, \ldots, x_n$ the class c^* such that

$$c^* = \arg\max_{c \in R(C)} P(C = c|\mathbf{X} = x_1, \ldots, x_n). \tag{7}$$

By *supervised classification* we understand the problem of learning a classifier from a set of labeled examples, i.e., from a database with variables $X_1, \ldots, X_n, C$ where the value of C is known in all the records in the database.

Probabilistic graphical models, and more precisely Bayesian networks, have been used as classifiers, as they provide compact representations of joint probability distributions. Usually, the structure of the network is restricted in such a way that the class variable is set as root and the feature variables are connected to the class [5]. Similarly, PDGs have been used as classifiers by imposing certain structural restriction, ensuring that all the features are connected to the class. The formal definition is as follows.

Definition 5 (PDG Classifier [16]). *A* PDG classifier $\mathcal{C}$ *is a PDG model that, in addition to the structural constraints of Definition 1, satisfies the following two structural constraints:*

1. *G defines a forest containing a single tree over the variables $\mathbf{C} = \{C\} \cup \mathbf{X}$,*
2. *C is the root of this tree.*

In a PDG classifier, the forest is restricted to contain a single tree in order to guarantee that all the feature variables are connected to C by a path in G. By forcing C to be placed at the root, typical Bayesian network classifiers structures can be easily replicated, as for instance, the Naïve Bayes (NB) model.

Definition 6 (MoTBF-PDG Classifier). *An* MoTBF-PDG classifier $\mathcal{C}$ *is an MoTBF-PDG that satisfies the structural constraints in Definition 5.*

In a classification problem with class variable C and features $X_1, \ldots, X_n$ (discrete or continuous), if we denote by $\mathbf{C}$ the set $\{C, X_1, \ldots, X_n\}$, an MoTBF-PDG classifier G would be used to represent the joint distribution

$$f_G(\mathbf{c}) = f_G(c, x_1, \ldots, x_n).$$

According to (7), we need to determine the value

$$c^* = \arg\max_{c \in R(C)} f_G(c|x_1, \ldots, x_n) = \arg\max_{c \in R(C)} \frac{f_G(c, x_1, \ldots, x_n)}{\sum_{c \in R(C)} f_G(c, x_1, \ldots, x_n)}.$$

As $\sum_{c \in R(C)} f_G(c, x_1, \ldots, x_n)$ does not depend on c, solving the classification problem is equivalent to finding the value

$$c^* = \arg\max_{c \in R(C)} f_G(c, x_1, \ldots, x_n).$$

Hence, in order to classify an item with observed features $\mathbf{x} = (x_1, \ldots, x_n)$, we just have to compute, for each $c \in R(C)$, the value $f_G(c, x_1, \ldots, x_n)$, which amounts to evaluate the conditional MoTBF functions in the parameter nodes reached by $(c, x_1, \ldots, x_n)$ as described in (6).

4.1 Learning MoTBF-PDG Classifiers from Data

Learning an MoTBF-PDG classifier from data consists of determining the structure of the PDG and estimating the conditional MoTBF distributions in the parameter nodes. Assuming a fixed PDG structure (see Definition 1) the MoTBF probability function corresponding to each parameter node ν is estimated by first determining the elements in the data sample that reach ν, and then learning a univariate MoTBF using those data points following the method described in [11]. We refer the reader to that reference for further details on the estimation procedure for univariate MoTBF densities.

For determining the structure, we have considered three basic approaches:

1. Fix a *Naïve Bayes-like* structure, so that all the features are directly connected to the class variable. We denote this approach as `NB`. An example of a PDG with `NB` structure is depicted in Fig. 4.

2. Rank the feature variables according to their mutual information with the class variable and connect the variables conforming a chain rooted by the class variable followed by the features in a sequence according to their rank. The mutual information is estimated from data, discretising the continuous variables. We will refer to this approach by the term `ranked`. An example of a structure obtained in this way is found in the left panel of Fig. 5.
3. Rank the feature variables according to their mutual information with the class variable, and include the feature variables in the PDG one by one according to the rank (the class variable is always included on top of the structure). Unlike in the `ranked` approach, here each variable is inserted below any previously inserted variable. Among all possible insertion points, the one resulting in a better classification rate (CR) is chosen. The CR is computed in a validation set randomly drawn form the training database. In this paper, we have used an 80% of the training database for learning and a 20% for validation. We denote this approach by `rankedCR`. An example of an structure compatible with this approach is displayed in Fig. 5 (right).

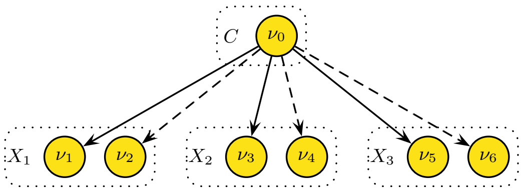
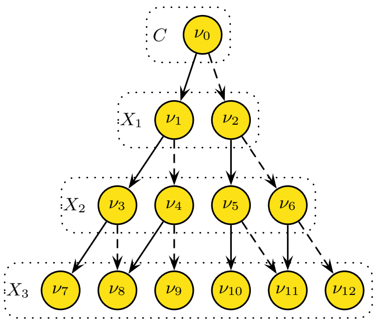


Fig. 4. A PDG structure compatible with the `NB` approach

During the process of inserting new variables in a PDG structure, it is necessary to decide the number of nodes to store at each variable, and the connections among them. In general, the maximum number of nodes at each variable depends on the number of nodes at its parent variable, and of the number of outgoing edge of each node at the parent.

Example 5. Consider the PDG at the right of Fig. 5. The maximum number of nodes at X_3 is 4, as its parent, X_2 has 2 nodes and each one has 2 outgoing arcs. However, in the PDG at the left of Fig. 5, the maximum number of nodes for X_3 is 8, as in this case X_2 has 4 nodes with 2 outgoing arcs each one. Note that, even though the maximum allowed is 8, in this example X_3 actually has 6 nodes, which indicates the presence of context specific independencies.

The number of outgoing arcs of a node corresponding to a discrete variable is equal to the number of possible values of the variable, which means that there are at least 2 outgoing arcs in such case. For a node corresponding to a continuous variable, the number of outgoing arcs may vary from 1 to any positive integer. In practice, it is necessary to establish a maximum number of arcs when learning



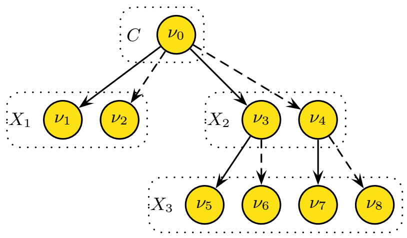


Fig. 5. Examples of PDG structures corresponding to the `ranked` (left) and `rankedCR` (right) approaches. It is assumed that the ranking of variables with respect to their mutual information with the class variable is X_1, X_2, X_3.

the PDG from data. Each outgoing arc corresponds to a subset of the domain of the continuous variable, so that the collection of the subsets associated with all the edges conforms a partition of its domain.

Example 6. Consider the PDG in Fig. 3 and its instantiation in Table 2. Node ν_1, corresponding to continuous variable Z_0 has 2 outgoing arcs. One of them corresponds to interval $[0, 0.5)$ and the other to interval $[0.5, 1]$.

The arcs emerging from a node not necessarily lead to different nodes, i.e., more than one arc may converge to the same node. Furthermore, arcs emerging from different nodes may converge to the same one. In this paper, we have considered a fixed number of intervals for every continuous variable, which is given as an argument to the classifier learning algorithm. The borders of the intervals are obtained following an equal frequency binning process.

Note that each node in a PDG represents a portion of the training data, namely those items that correspond to configurations that reach the parameter node. Therefore, as the PDG is expanded during its construction, the amount of data available for estimating the MoTBFs distributions in each node goes down. In order to avoid estimating the MoTBF functions from tiny samples, whenever a new variable is inserted during the process of constructing the PDG, we generate all the nodes for that variable and carry out a *collapse* operation [16], so that nodes that are not reached by a given minimum number of training items are collapsed. More precisely, the collapse operation involves the following steps:

1. Establish a *collapse threshold* $r_c > 0$.
2. Let $\nu_1, \ldots, \nu_k$ be the nodes for the current variable.
3. Let $size(\nu_i)$, $i = 1, \ldots, n$ denote the number of parameters estimated from data for node ν_i. This is the number of possible values of the variable minus 1, if the variable is discrete, and the parameters of the MoTBF density plus the number of intervals minus 1 if it is continuous.

4. Pairs of nodes (ν_i, ν_j) that are reached by a number of training items lower than $r_c \times size(\nu_i)$ and $r_c \times size(\nu_j)$ respectively, are chosen in increasing order of the previously mentioned thresholds and collapsed into a single new node, for which the corresponding probability function is re-estimated using the union of both training samples. When collapsing two nodes, the incoming and outgoing arcs are re-arranged appropriately.

The process is repeated until no nodes below the threshold remain, or until there is only one, in which case it is coupled with the smallest (in terms of $r_c \times size(\nu)$) parameter node for the same variable, and collapsed into a single one.

Once the MoTBF-PDG classifier has been constructed by any of the three approaches mentioned above, we carry out an operation aimed at reducing the chances of overfitting for the learnt model. The operation is called *merge*. It traverses the PDG structure bottom-up. For each variable, we explore a fraction of its parameter nodes, determined by a rate $r_m \in [0, 1]$ that we call *merge rate*. Then, the sampled parameter nodes are collapsed into a single one, similarly to the case of the collapse operation, but in this case *only if the classification rate*, computed from the validation set, *is increased* respect to the current model.

5 Experimental Evaluation

We have carried out an initial experimental evaluation aimed at evaluating the performance of MoTBF classifiers over a set of benchmark databases taken from the UCI (http://archive.ics.uci.edu/ml) and KEEL [1] repositories. A description of the datasets used in the experiments is given in Table 3.

In the experiments, we have induced MoTBF-PDG classifiers for each dataset using the three approaches explained in Sect. 4.1, i.e. `NB`, `ranked` and `rankedCR`, and several combinations of the parameters described there. More precisely, we have tested 2 and 3 intervals for the domain of the continuous parent variables, collapse thresholds of $r_c = 3, 5$ and 7, merge rates of $r_m = 0.25$ and 0.5. For the densities in the parameter nodes, we have used mixtures of polynomials (MOPs), which are one of the possible types of MoTBFs together with MTEs [10]. The polynomials have been learnt using the procedure described in [11] with limits on the degree of the polynomials equal to 4, 6, 8 and 10. We also tested discrete PDG classifiers as well as four Bayesian classifiers available in software Weka, namely Gaussian Naïve Bayes (called NB Simple in Weka), kernel NB, discrete NB and discrete TAN. We tested each algorithm measuring the classification rate (CR) using 5-fold cross validation. Our implementation of PDGs has been done using the `R` statistical package.

We found that the best combination of parameter values for the MoTBF-PDG classifiers was 3 intervals for the domain of the continuous parent variables, collapse threshold $r_c = 3$, merge rate $r_m = 0.25$, and maximum degree for polynomials equal to 6. We chose this combination of parameters by trying each possible combination of them an counting how many times each one was the winner in terms of classification rate of the constructed model. The results of

Table 3. Description of the datasets used in the experiments

	instances	#features	#continuous	#categorical	#classStates
appendicitis	106	7	7	0	2
banknote	1372	4	4	0	2
fourclass	862	2	2	0	2
haberman	306	3	3	0	2
iris	150	4	4	0	3
liver	345	6	6	0	2
newthyroid	215	5	5	0	3
phoneme	5404	5	5	0	2
pima	768	8	8	0	2
seeds	209	7	7	0	3
teaching	151	5	3	2	3
vertebral	309	6	6	0	2
wine	178	13	13	0	3

the experiments in terms of CR attained by each classifier are shown in Table 4. The sizes of the obtained models, measured as the number of parameters they contain, are displayed in Table 5. The values shown for MoTBF-PDG classifiers correspond to the configuration of parameters described above.

Table 4. Classification rates attained by the tested classifiers

	MoTBF-PDG			Discrete-PDG			Weka			
Database	`NB`	`ranked`	`rankedCR`	`NB`	`ranked`	`rankedCR`	Discrete-NB	Kernel-NB	Gaussian-NB	Discrete-TAN
appendicitis	0.8403	0.8307	0.8597	0.7935	0.8121	0.8117	0.8022	0.8771	0.868	0.8489
banknote	0.8586	0.9752	0.9738	0.8571	0.9425	0.9388	0.6348	0.9227	0.8397	0.9344
fourclass	0.7553	0.8365	0.8202	0.725	0.7982	0.7982	0.6439	0.8677	0.7506	0.8411
haberman	0.7482	0.7384	0.7287	0.706	0.7125	0.7222	0.7353	0.7418	0.7483	0.7255
iris	0.94	0.94	0.92	0.9333	0.92	0.9333	0.7867	0.9667	0.96	0.9267
liver	0.5768	0.5739	0.6232	0.658	0.5884	0.6	0.5797	0.658	0.5623	0.5768
newthyroid	0.907	0.893	0.8791	0.9023	0.8698	0.8884	0.707	0.9581	0.9674	0.9442
phoneme	0.779	0.7613	0.7846	0.7435	0.8116	0.795	0.7065	0.7841	0.7606	0.805
pima	0.7474	0.7045	0.7383	0.7318	0.7358	0.7267	0.651	0.7422	0.7591	0.7448
seeds	0.8949	0.8854	0.8854	0.8854	0.8707	0.8806	0.8801	0.8994	0.9138	0.8994
teaching	0.411	0.4239	0.4566	0.5101	0.5034	0.5166	0.4297	0.5428	0.5295	0.4503
vertebral	0.7411	0.738	0.8412	0.7572	0.7604	0.7605	0.6764	0.7668	0.7766	0.8057
wine	0.9611	0.8873	0.9611	0.9552	0.8817	0.9497	0.944	0.9775	0.966	0.9662

5.1 Discussion

In order to determine the significance of the results in Table 4, we run Friedman's test with maxT statistic [7], reporting significant differences among the tested classifiers (p-value below 0.05) in terms of accuracy (classification rate). Then we carried out a *post hoc* analysis following Wilcoxon-Nemenyi-McDonald-Thompson's procedure for pairwise comparisons. The result of the *post hoc* analysis is shown in Fig. 6, where a box plot for the differences in classification rate between every pair of algorithms is displayed. Green boxes are used to highlight the cases where statistically significant differences were found, which are, from left to right, discrete TAN vs. discrete NB, Gaussian NB vs. discrete NB, kernel NB vs. discrete NB, MoTBF-PDG (rankedCR) vs. discrete NB, kernel NB vs. discrete PDG (NB) and kernel NB vs. discrete PDG (ranked).

Table 5. Sizes of the learnt classifiers computed as the number of parameters they contain

	MoTBF-PDG			Discrete-PDG			Weka			
Database	`NB`	`ranked`	`rankedCR`	`NB`	`ranked`	`rankedCR`	Discrete-NB	Kernel-NB	Gaussian-NB	Discrete-TAN
appendicitis	38.6	120	35	29	77.4	35.4	127	743	27	139
banknote	36	307.8	216.6	17	83.4	40.2	73	5489	19	79
fourclass	9	22	17.4	9	15.8	15.4	37	1725	9	39
haberman	19.6	37.4	21	13	42.6	14.6	55	919	13	59
iris	45.6	85.8	51.6	26	32.8	26	110	602	26	119
liver	56.2	275	117.6	25	165.8	63.4	109	2071	25	119
newthyroid	59.4	169.6	71.8	32	82	36.4	137	1077	32	149
phoneme	53.2	988.6	105.8	21	309.8	49	91	27021	21	99
pima	66.8	686.6	165.2	33	460.2	114.6	145	6145	29	159
seeds	80.6	192.4	100.6	44	90.8	47.6	191	1465	44	209
teaching	19.6	90.8	42.4	26	58.2	28.2	87	459	16	151
vertebral	50.6	242.2	151.6	25	128.6	56.6	109	1855	25	119
wine	146	381.4	189.2	80	238.4	95.2	353	2316	80	389

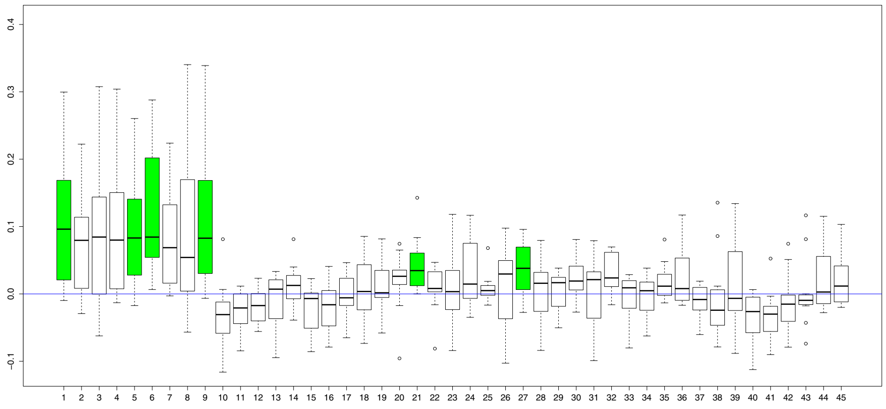


Fig. 6. Results of the pairwise comparison between the tested methods. Boxes in green indicate statistically significant differences

MoTBF-PDG classifiers are in general heavier in terms of parameters, as shown in Table 5, except kernel NB which always returns the largest model. However, it must be pointed out that we have not considered variable selection during the construction of any of the tested classifiers.

Even though defining a variable selection strategy for learning MoTBF-PDG classifiers is a matter of future research, we carried out a simple experiment in order to have a glimpse on the impact of variable selection both on size and accuracy of the learnt PDGs. The experiment consisted of inducing a classification tree and then learning the PDG but using only the variables actually included in the tree. Table 6 shows a comparison of the results with and without variable selection. The results suggest that variable selection can have a remarkable impact on the obtained models. Note that the model sizes dramatically decrease while the CR is not seriously deteriorated and even improved in some cases.

Table 6. Results of the preliminary experiment with variable selection. Columns Old CR, Old size, New CR and New size contain the results of MoTBF-PDG classifier with `rankedCR` strategy and the same configuration of parameters as in the main experiment, where 'old' indicates including all the variables and 'new' only with the same variables as in the induced classification tree. #Vars indicates the ratio of included variables.

Database	Old CR	New CR	Old size	New size	#Vars
appendicitis	0.8597	0.8403	35	13	2/7
haberman	0.7287	0.7318	21	18.6	2/3
iris	0.92	0.94	51.6	28.4	2/4
pima	0.7383	0.7435	165.2	107.4	6/8
seeds	0.8854	0.9139	100.6	53.6	4/7
wine	0.9611	0.9216	189.2	40.8	3/13

6 Concluding Remarks

In this paper we have introduced a new hybrid probabilistic graphical model, called MoTBF-PDG, resulting from the combination of PDGs with the MoTBF framework. We have done that within the context of supervised classification, motivated by the fact that PDGs had already been successfully employed as classifiers. The initial experimental evaluation suggests that the new classifiers are potentially competitive with existing ones. The analysis also suggests that variable selection is necessary in order to obtain compact models less prone to overfitting. A more extensive experimentation, including larger datasets and more algorithms as the one presented in [13] is necessary for determining the impact of the different parameters involved in the learning process.

Finally, MoTBF-PDGs are not necessarily models restricted to the framework of supervised classification. They could, for instance, be used for regression problems, where the variable to predict is continuous. As a general model for representing a joint probability distribution, the necessary operations for carrying out probabilistic inference still remain to be developed for MoTBF-PDGs.

Acknowledgments. This work has been supported by the Spanish Ministry of Economy and Competitiveness, grant TIN2010-20900-C04-02, by Junta de Andalucía grants P11-TIC-7821, P12-TIC-2541, and by ERDF (FEDER) funds.

References

[1] Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F.: KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. J. Mult.-Valued Log. Soft Comput. 17, 255–287 (2011)

[2] Bozga, M., Maler, O.: On the Representation of Probabilities over Structured Domains. In: Halbwachs, N., Peled, D.A. (eds.) CAV 1999. LNCS, vol. 1633, pp. 261–273. Springer, Heidelberg (1999)

[3] Cobb, B.R., Shenoy, P.P.: Inference in Hybrid Bayesian Networks with Mixtures of Truncated Exponentials. Int. J. Approximate Reasoning 41, 257–286 (2006)

[4] Flores, M.J., Gámez, J.A., Nielsen, J.D.: The PDG-mixture Model for Clustering. In: 11th Int. Conf. on Data Warehousing & Knowledge Discovery, pp. 378–389 (2009)

[5] Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian Network Classifiers. Machine Learning 29, 131–163 (1997)

[6] Gámez, J.A., Nielsen, J.D., Salmerón, A.: Modelling and Inference with Conditional Gaussian Probabilistic Decision Graphs. Int. J. Approximate Reasoning 53, 929–945 (2012)

[7] Horthorn, T., Hornik, K., van de Wiel, M.A., Zeileis, A.: Implementing a Class of Permutation Tests: The coin Package. J. Stat. Soft. 28, 1–23 (2008)

[8] Jaeger, M.: Probabilistic Decision Graphs - Combining Verification and AI Techniques for Probabilistic Inference. Int. J. Uncertainty Fuzziness Knowledge Based Syst. 12, 19–42 (2004)

[9] Langseth, H., Nielsen, T.D., Rumí, R., Salmerón, A.: Parameter Estimation and Model Selection for Mixtures of Truncated Exponentials. Int. J. Approximate Reasoning 51, 485–498 (2010)

[10] Langseth, H., Nielsen, T.D., Rumí, R., Salmerón, A.: Mixtures of Truncated Basis Functions. Int. J. Approximate Reasoning 53, 212–227 (2012)

[11] Langseth, H., Nielsen, T.D., Pérez-Bernabé, I., Salmerón, A.: Learning mixtures of truncated basis functions from data. Int. J. Approximate Reasoning 55, 940–956 (2014)

[12] Lauritzen, S.L., Wermuth, N.: Graphical Models For Associations Between Variables, Some of Which Are Qualitative and Some Quantitative. The Annals of Statistics 17, 31–57 (1989)

[13] López-Cruz, P.L., Bielza, C., Larrañaga, P.: Learning mixtures of polynomials of multidimensional probability densities from data using B-spline interpolation. Int. J. Approximate Reasoning 55, 989–1010 (2014)

[14] Moral, S., Rumí, R., Salmerón, A.: Mixtures of Truncated Exponentials in Hybrid Bayesian Networks. In: Benferhat, S., Besnard, P. (eds.) ECSQARU 2001. LNCS (LNAI), vol. 2143, pp. 135–143. Springer, Heidelberg (2001)

[15] Nielsen, J.D., Jaeger, M.: An Empirical Study of Efficiency and Accuracy of Probabilistic Graphical Models. In: Third European Workshop on Probabilistic Graphical Models, pp. 215–222 (2006)

[16] Nielsen, J.D., Rumí, R., Salmerón, A.: Supervised Classification Using Probabilistic Decision Graphs. Comput. Stat. Data Anal. 53, 1299–1311 (2009)

[17] Romero, V., Rumí, R., Salmerón, A.: Learning Hybrid Bayesian Networks Using Mixtures of Truncated Exponentials. Int. J. Approximate Reasoning 42, 54–68 (2006)

[18] Shenoy, P., West, J.: Inference in Hybrid Bayesian Networks Using Mixtures of Polynomials. Int. J. Approximate Reasoning 52, 641–657 (2011)