Supervised Classification Using Probabilistic Decision Graphs

Jens D. Nielsen *

Department of Computer Science, University of Castilla-La Mancha, Campus Universitario Parque Científico y Tecnológico s/n, 02071 Albacete (Spain)

Rafael Rumí and Antonio Salmerón

Department of Statistics and Applied Mathematics, University of Almería, La Cañada de San Urbano s/n, 04120 Almería (Spain)

Abstract

A new model for supervised classification based on probabilistic decision graphs is introduced. A probabilistic decision graph (PDG) is a graphical model that efficiently captures certain context specific independencies that are not easily represented by other graphical models traditionally used for classification, such as the Naïve Bayes (NB) or Classification Trees (CT). This means that the PDG model can capture some distributions using fewer parameters than classical models. Two approaches for constructing a PDG for classification are proposed. The first is to directly construct the model from a dataset of labelled data, while the second is to transform a previously obtained Bayesian classifier into a PDG model that can then be refined. These two approaches are compared with a wide range of classical approaches to the supervised classification problem on a number of both real world databases and artificially generated data.

Key words: Supervised Classification, Graphical Models, Probabilistic Decision Graphs.

* Corresponding author. Address: Instituto de Investigación en Informática de Albacete - I^{3}A, Campus Universitario, Parque Científico y Tecnológico s/n. 02071 Albacete, Spain.
Tlf.: (+34) 967 599 200 Ext. 2677, Fax: (+34) 967 599 343
Email addresses: dalgaard@dsi.uclm.es (Jens D. Nielsen), rrumi@ual.es (Rafael Rumí), antonio.salmeron@ual.es (Antonio Salmerón).

1 Introduction

Classification is an important task within data analysis. The problem of classification consists of determining the class to which an individual belongs given that some features about that individual are known. Much attention has been paid in the literature to the induction of *classifiers* from data. In this paper we are concerned with *supervised classification*, which means that the classifier is induced from a set of data containing information about individuals for which the class value is known.

In the last decades, probabilistic graphical models, and particularly Bayesian networks (BNs) (Castillo et al., 1997; Jensen, 2001; Pearl, 1988) have been successfully applied to the classification problem, giving rise to the so-called *Bayesian network classifiers*, which by Friedman et al. (1997) was shown to be competitive with classical models like classification trees (Breiman et al., 1984; Quinlan, 1986).

Probabilistic decision graphs (PDGs) constitute a class of probabilistic graphical models that naturally capture certain context specific independencies (Boutilier et al., 1996) that are not easily represented by other graphical models (Jaeger, 2004; Jaeger et al., 2006). This means that the PDG model can capture some distributions with fewer parameters than classical models, which in some situations leads to a model less prone to over-fitting.

In this paper we propose a new model for supervised classification based on the representational capabilities of PDGs. We introduce algorithms for inducing PDG-based classifiers directly from data, as well as for transforming a previously existing Bayesian network classifier into a PDG, under the restriction that the structure of the Bayesian network is a *forest augmented Bayesian network* (Lucas, 2002).

The rest of the paper is organised as follows. We establish the notation used throughout the paper in section 2. The classification problem and the most commonly used classifiers are described in section 3. The general PDG model as well as the PDG classification model are introduced in section 4. Two different approaches to the construction of classifiers based on PDGs are studied in section 5. The proposed methods are experimentally tested in section 6 and the paper ends with conclusions in section 7.

2 Basic Notation

We will denote random variables by uppercase letters, and by boldfaced uppercase letters we denote sets of random variables, e.g. $\mathbf{X} = \{X_0, X_1, \ldots, X_n\}$. By $R(X)$ we denote the set of possible states of variable X, and this extends naturally to sets of random variables $R(\mathbf{X}) = \times_{X_i \in \mathbf{X}} R(X_i)$. 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 $\mathbf{Y}$.

Let G be a directed graph over nodes $\mathbf{V}$, $X_i \in \mathbf{V}$. We will denote by $pa_G(X_i)$ the set of parents of node X_i in G, by $de^*_G(X_i)$ the set of children of X_i, and by $nd_G(X_i)$ the set of non-descendants of X_i in G.

3 Classification

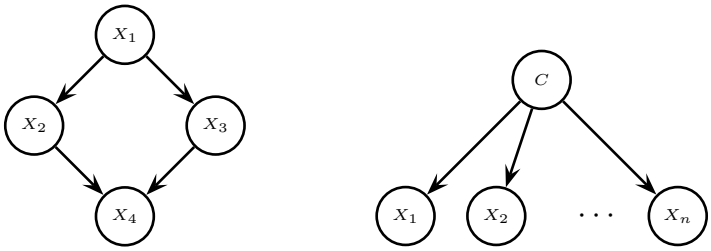
A classification problem can be described in terms of a set of *feature variables* $\mathbf{X} = \{X_1, \ldots, X_n\}$, that describe an individual, and a *class variable*, C, that indicates the class to which that individual belongs. A classification model, commonly called *classifier*, is a model oriented to predict the value of variable C given that the values of the features $X_1, \ldots, X_n$ are known. Throughout this paper, we will use the notation $\mathbf{C} = \{C\} \cup \mathbf{X}$. We will assume that all the variables in a classification model are discrete.

There are different kinds of classification models. A popular group of them are the so-called *Bayesian network classifiers*, which are particular types of Bayesian networks (Castillo et al., 1997; Jensen, 2001; Pearl, 1988). A Bayesian network (BN) model (see Fig. 1(a)) is a directed acyclic graph in which each node represents a random variable, and the existence of an arc between two nodes indicates that the corresponding random variables are statistically dependent. Every node has associated a probability distribution of the corresponding variable given its parents in the graph.

A key property of BNs is that the joint distribution over the variables in the network factorises according to the concept of d-separation as follows:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | pa(X_i)) \,. \tag{1}$$

This factorisation implies that the joint distribution of all the variables in the network can be specified with an important reduction of the number of free parameters.



(a) (b)

Figure 1. An example of a BN model (a), and the structure of the NB model classifier (b).

For example, the BN model with the structure given in Fig. 1(a) induces the following factorisation of the joint probability distribution:

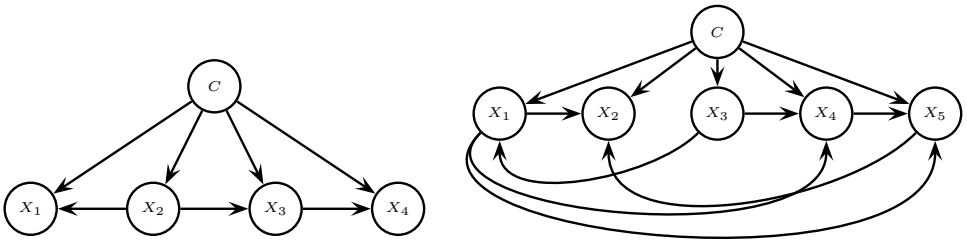
$$P(X_1, X_2, X_3, X_4) = P(X_1)P(X_2|X_1)P(X_3|X_1)P(X_4|X_2, X_3) \ .$$

A BN model can be used for classification purposes if its set of nodes corresponds to the variables $\mathbf{C} = \{C\} \cup \mathbf{X}$, as it can be used to compute the posterior distribution of the class variable given the features, so that an individual with observed features $x_1, \ldots, x_n$ will be assigned to class c^* such that

$$c^* = \arg\max_{c \in R(C)} P(C = c|\mathbf{X} = x_1, \ldots, x_n) \ . \tag{2}$$

The posterior distribution of the class variable given the features is proportional to $P(X_1, \ldots, X_n|C) \cdot P(C)$. Therefore, in the most general setting it would be necessary to specify a number of parameters exponential in the number of variables. However, if the distribution is represented as a Bayesian network, it is possible to take advantage of the factorisation encoded by the network. Moreover, it is common to use only some restricted classes of Bayesian networks when approaching classification problems, so that the number of free parameters to estimate does not grow exponentially with the number of variables.

The simplest kind of Bayesian network for classification is the so-called *Naïve Bayes* (NB) model (Minsky, 1963). In the NB classifier, the feature variables are assumed to be independent given the class variable, which corresponds to the structure depicted in Fig. 1(b). It means that the posterior distribution of the class variable factorises as



(a) TAN (b) kdB

Figure 2. Sub-figure (a) shows an example of a TAN structure with 4 features, and (b) shows an example of a kdB structure ($k = 2$) with 5 features.

$$P(C|X_1, \ldots, X_n) \propto P(C) \prod_{i=1}^{n} P(X_i|C) \ , \tag{3}$$

and therefore, the number of free parameters is linear in the number of variables. The drawback of this model is that the independence assumption is made previously to the induction of the model from the data. Therefore, this assumption might be not supported by the data. However, this is usually compensated by the reduction on the number of free parameters to estimate, which also makes the NB classifier less prone to over-fitting than other more complex models (Domingos and Pazzani, 1997).

The *Tree Augmented Naïve Bayes* (TAN) model (Friedman et al., 1997) relaxes the independence assumption behind the NB model, by allowing some dependencies among the features. More precisely, the TAN model assumes that the feature variables are arranged in a directed tree structure, which means that each variable has one more parent besides the class variable, except the root of the directed tree, whose only parent is the class. An example of a TAN structure is shown in Fig. 2(a).

Both the TAN and NB structures are particular cases of the *Forest Augmented Bayesian Network* (FAN) model (Lucas, 2002). This model assumes that the feature variables form a *forest* of directed tree structures. An example of a FAN can be obtained from Fig. 2(a) by removing the arc between X_2 and X_3.

The more general Bayesian network classifier is the *k-dependence Bayesian network* (kdB) (Sahami, 1996). A kdB classifier is a Bayesian network which allows each feature to have a maximum of k feature variables as parents, apart from the class variable which is a parent of every feature. Fig. 2(b) shows an example of a kdB structure with $k = 2$.

Another important group of classifiers is based on the induction of a set of rules, arranged as a tree, that partition the sample space of the features into homogeneous groups in terms of the value of the class variable. The models within this group are usually called tree-structured classifiers or *classification*

tree (CT) models. A CT model is a hierarchical model, composed by terminal leaves and decision nodes. Each decision node represents a test about a feature variable, with a finite number of outcomes. Every outcome is connected to another decision node or terminal leaf. Leaf nodes have no further links, but they bear a value for the class variable (see Fig. 3).

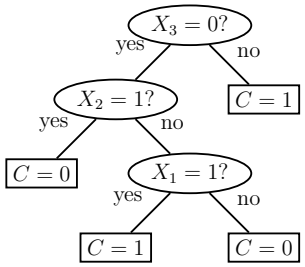


Figure 3. Example of a CT structure with 3 binary features and a binary class. Oval nodes are decision nodes, and square nodes are terminal leaves.

There exist several methods for inducing CTs from data. The models induced by the *CART* method (Breiman et al., 1984) are binary trees where the selection of the variables to include in the model is made according to *entropy measures*, and the tests are selected according to the *goodness of split*. The *C4.5* (Quinlan, 1993) and its predecessor *ID3* (Quinlan, 1986) allow more than two outcomes in the decision nodes.

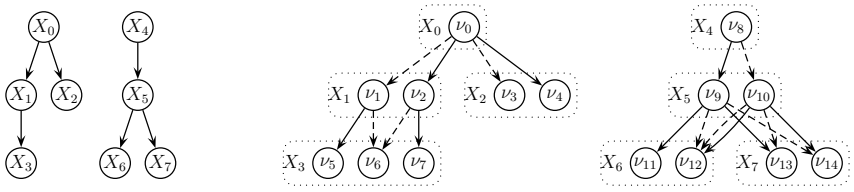
A different approach is followed in the *Dirichlet classification tree* (Abellán and Moral, 2003), in which the imprecise Dirichlet model is used to estimate the probabilities of the values of the class variable.

4 The Probabilistic Decision Graph model

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).

A PDG encodes a joint probability distribution over a set of discrete random variables $\mathbf{X}$ by representing each random variable $X_i \in \mathbf{X}$ by a set of nodes $\{\nu_0, \ldots, \nu_l\}$. Nodes are organised in a set of rooted DAG structures that are consistent with an underlying forest-structure over variables $\mathbf{X}$. The structure is formally defined as follows:

Definition 4.1 (The PDG Structure) *Let F be a variable forest over domain $\mathbf{X}$. A* PDG-structure $G = \langle \mathbf{V}, \mathbf{E} \rangle$ *for $\mathbf{X}$ w.r.t. F is a set of* rooted *DAGs, such that:*



(a) (b)

Figure 4. A variable 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).

(1) Each node $\nu \in \mathbf{V}$ is labelled with some $X_i \in \mathbf{X}$. By $\mathbf{V}_{X_i}$, we will refer to the set of all nodes in a PDG-structure labelled with the same variable X_i. For every variable X_i, $\mathbf{V}_{X_i} \neq \emptyset$.

(2) For each node $\nu_i \in \mathbf{V}_{X_i}$, each possible state $x_{i,h}$ of X_i and each successor $X_j \in ch_F(X_i)$ there exists exactly one *edge labelled with $x_{i,h}$ from ν_i to some node ν_j labelled with random variable X_j. Let $X_j \in ch_F(X_i)$ and $\nu_i \in \mathbf{V}_{X_i}$. By $succ(\nu_i, X_j, x_{i,h})$ we will then refer to the unique node $\nu_j \in \mathbf{V}_{X_j}$ that is reached from ν_i by an edge with label $x_{i,h}$.*

Example 4.1 *A variable forest F over binary variables $\mathbf{X} = \{X_0, \ldots, X_7\}$ can be seen in Figure 4(a), and a PDG structure over $\mathbf{X}$ w.r.t. F in Figure 4(b). The labelling of nodes ν in the PDG-structure is indicated by the dashed boxes, e.g., the nodes labelled with X_2 are visualised as the set $\mathbf{V}_{X_2} = \{\nu_3, \nu_4\}$. Dashed edges correspond to edges labelled 0 and solid edges correspond to edges labelled 1, for instance $succ(\nu_9, X_6, 0) = \nu_{12}$.*

A PDG-structure is instantiated by assigning to every node ν a local multinomial distribution over the variable that it represents. We will refer to such local distributions by $p^\nu = (p^\nu_1, \ldots p^\nu_{k_i}) \in \mathbb{R}^{k_i}$, where $k_i = |R(X_i)|$ is the number of distinct states of X_i. Then, by $p^\nu_{x_{i,h}}$ we refer to the h'th element of p^ν under some ordering of $R(X_i)$.

Definition 4.2 (The PDG model) *A PDG model $\mathcal{G}$ is a pair $\mathcal{G} = \langle G, \theta \rangle$, where G is a valid PDG-structure (Def. 4.1) over some set $\mathbf{X}$ of discrete random variables and θ is a set of local distributions that fully instantiates G.*

Definition 4.3 (Reach) *Let G be a PDG structure over variables $\mathbf{X}$ w.r.t. forest F. A node ν in G labelled with X_i is* reached *by $\mathbf{x} \in R(\mathbf{X})$ if*

- *ν is a root in G, or*
- *$X_i \in ch_F(X_j)$, $\nu' \in \mathbf{V}_{X_j}$, ν' is reached by $\mathbf{x}$ and $\nu = succ(\nu', X_i, \mathbf{x}[X_j])$.*

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

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

$$P^{\mathcal{G}}(\mathbf{x}) = \prod_{X_i \in \mathbf{X}} p_{\mathbf{x}[X_i]}^{reach_G(i,\mathbf{x})}\,, \tag{4}$$

where $p^{reach_G(i,\mathbf{x})}$ refers to the parameters contained in the unique node from $\mathbf{V}_{X_i}$ that is reached by joint configuration $\mathbf{x}$ and $p_{\mathbf{x}[X_i]}^{reach_G(i,\mathbf{x})}$ then refers to the specific entry in that parameter vector for the value of X_i in $\mathbf{x}$. A set of nodes $\mathbf{V}_{X_i}$ in a PDG structure G over variables $\mathbf{X}$ partitions the state space $R(\mathbf{X})$ into a set of disjoint subsets, namely $(\nu \in \mathbf{V}_{X_i})\{\mathbf{x} \in R(\mathbf{X}) : reach_G(i,\mathbf{x}) = \nu\}$. We will denote by $\mathscr{A}_G(X_i)$ the partitioning of $R(\mathbf{X})$ defined by $\mathbf{V}_{X_i}$ in G. Then, the PDG structure G imposes the following conditional independence relations:

$$X_i \perp\!\!\!\perp nd_G(X_i) | \mathscr{A}_G(X_i)\,, \tag{5}$$

where $nd_G(X_i)$ denotes the non-descendants of X_i in structure G.

4.1 The PDG classifier

In this section we introduce the PDG classification model. First, we give the following formal definition of the model:

Definition 4.4 (The PDG Classifier) *A* PDG classifier $\mathcal{C}$ *is a PDG model that, in addition to the structural constraints of Def. 4.1, satisfies the following two structural constraints:*

(1) G defines a forest containing a single tree over the variables $\mathbf{C}$, and
(2) C is the root of this tree.

The PDG model was initially inspired by ROBDDs (Bryant, 1992) which is a modelling framework that allows efficient representation of boolean expressions. As we will see in the following example, the PDG model has inherited the ability to represent boolean expressions efficiently, at least to some extent.

Example 4.2 *Let $\mathbf{X}$ be a set of truth-valued feature variables, and let C be a truth valued class variable. Assume that the label ($c \in R(C)$) of an individual $\mathbf{x} \in R(\mathbf{X})$ is determined as:*

$$C = (((x_0 \oplus x_1) \oplus x_2) \oplus \cdots) \oplus x_n\,, \tag{6}$$

where $x_i = \mathbf{x}[X_i]$ and $\oplus$ is the logical exclusive-or operator. Assume that no other relations exists, that is all $X \in \mathbf{X}$ are independent given any subset $S \subset \mathbf{X}$. Then, using the terminology of Jaeger (2003) it can be realised that the concept defined by Eq. (6) is order-n polynomially separable*, where $n = |\mathbf{X}|$. We say that a concept is recognised by a classifier if for any individual $\mathbf{x} \in R(\mathbf{X})$ the correct class label c is assigned to $\mathbf{x}$ by that classifier. Jaeger (2003) proved that if a concept A is not* order-n polynomially separable *then*

there exists no order-m association classifier *($m < n$) that recognises A. The NB model is an* order-1 association classifier *and the TAN and FAN models are* order-2 association classifiers*. The concept is efficiently recognised by a PDG classifier. Consider $n = 4$ we have the concept*

$$C = ((X_1 \oplus X_2) \oplus X_3) \oplus X_4 \,. \tag{7}$$

The PDG classifier with the structure shown in Fig. 5(a) can recognise the concept of Eq. (7). The two parameter nodes representing X_4 contains zero-one distributions while the rest of the parameter nodes can contain any positive distribution without affecting the classifier.

The structure of Fig. 5(a) defines a model that contains 11 free parameters, and adding more feature variables to the exclusive-or function determining the label for the instance only yields an addition of 4 extra parameters to the model. As noted above, neither NB nor TAN classifiers can recognise this concept, and the kdB model would need $k = 4$ to recognise it.

Fig. 5(b) shows a PDG structure that efficiently represents the model where the class label of an individual $\mathbf{x} \in R(\mathbf{X})$ is determined by the parity of feature variables:

$$C = \begin{cases} \texttt{true} & \text{if } (\sum_i \delta_{\texttt{true}}(x_i)) \text{ is odd} \\ \texttt{false} & \text{otherwise} \end{cases}, \tag{8}$$

where $x_i = \mathbf{x}[X_i]$ and $\delta_{\texttt{true}}$ is the indicator function. The concept in (8) is captured by the PDG model with the structure shown in Fig. 5(b) with maximum likelihood estimates. This model defines 2n+1 free parameters where n is the number of feature variables, and again, this number grows linearly as we add more feature variables to the concept. Also, this model defines a concept that is not recognised by any NB, TAN nor FAN models (for n>2). A kdB model can recognise the concept for k=n, but will require exponentially many parameters to do so.

5 Learning the PDG Classifier

In this section we propose two different approaches to learning the PDG classifier. The first approach, presented in Section 5.1, is based on a transformation of a given FAN classifier into an equivalent PDG which can then subsequently refined. Previous comparative studies have demonstrated the strength of the PDG model as a secondary structure in probability estimation (Jaeger et al., 2004). In the study of Jaeger et al. (2004), a PDG is learned from a Junction Tree model and thereafter a series of merging operations is applied. These operations effectively remove redundant parameters and parameters with little or no data-support. It is shown that the PDG model can typically be much

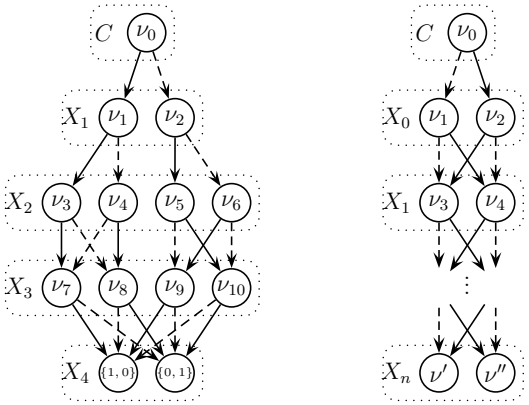


Figure 5. The 2 different structures of PDG classifiers discussed in Example 4.2. Solid edges correspond to the value `true` and dashed edges correspond to `false`. The structure shown in (a) can recognise the concept of Eq. (7) while the structure in (b) can recognise the concept of Eq. (8).

smaller than the JT model without degrading the precision of the represented distribution.

The second approach, presented in Section 5.2, concerns direct learning of PDG classifiers from labelled data.

5.1 Transforming a BN Classifier into a PDG

It is known that any clique tree obtained from a BN model can be represented by a PDG with a number of free parameters linear on the size of the clique tree (Jaeger, 2004, Theorem 5.1). A clique tree is the main structure for organising computations in many popular algorithms for exact inference in BN models, as can be seen, for instance, in (Jensen et al., 1990). As we will show later, if we consider BN models with FAN structure, then the equivalence in terms of number of free parameters is not only met for the clique tree, but also for the BN model. In the following we propose an algorithm for constructing a PDG model from a FAN model that represents the same distribution and contains the same amount of parameters.

The idea of the algorithm is to construct a PDG with variable forest given by the forest structure of the FAN. The root will be the class variable and the features are arranged in subtrees underneath the root. Let $\mathcal{B}$ be a FAN, and let $P^{\mathcal{B}}$ be the joint distribution represented by model $\mathcal{B}$. Each variable X will then be represented by $|R(pa_{\mathcal{B}}(X))|$ nodes such that there will be a unique node $\nu_{\mathbf{w}} \in \mathbf{V}_X$ for every $X \in \mathbf{X}$ and $\mathbf{w} \in R(pa_{\mathcal{B}}(X))$ for which $p^{\nu_{\mathbf{w}}} = P(X|pa_{\mathcal{B}}(X) = \mathbf{w})$. The nodes will then be connected in such a way that the path from the root node to the leaves defined by any full instance

$\mathbf{c} \in R(\mathbf{C})$ for each feature variable X reaches exactly the node that contains $P^{\mathcal{B}}(X|pa_{\mathcal{B}}(X) = \mathbf{c}[pa_{\mathcal{B}}(X)])$. The details can be found in function FANToPDG in Algorithm 1.

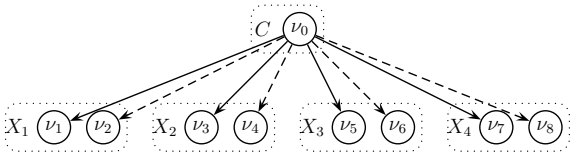
Algorithm 1 Function FANToPDG converts a BN classifier model with FAN structure to an equivalent PDG classifier model.

```
1: function FANToPDG(B)
2:     Create new node ν_root, and set p^{ν_root} = P^B(C).
3:     V_C = {ν_root}.
4:     for all X ∈ X do
5:         V_X ← ∅.
6:         for all w ∈ R(pa_B(X)) do
7:             Create new node ν_w, and set p^{ν^X_w} = P^B(X|pa_B(X) = w).
8:             V_X ← V_X ∪ {ν_w}.
9:     for all trees T over feature variables in B do
10:        Let L be a list of the variables in T.
11:        order L according to a depth-first traversal of T.
12:        while L ≠ ∅ do
13:            Let X be the next variable in L.
14:            for all ν_w ∈ V_X do
15:                Let c be the value of C in w.
16:                if pa_B(X) only contains C then
17:                    Set succ(ν_root, X, c) = ν^X_w.
18:                else
19:                    Let Y be the parent of X in B that is not C.
20:                    Let y be the value of Y in w.
21:                    for all ν_u ∈ V_Y where u[C] = c do
22:                        Set succ(ν_u, X, y) = ν_w.
23:            L ← L \ {X}.
24:    return a new PDG model with ν_root as root.
```

Before proving the correctness of Alg. 1, we give two examples of applying the FANToPDG function on specific FAN models.

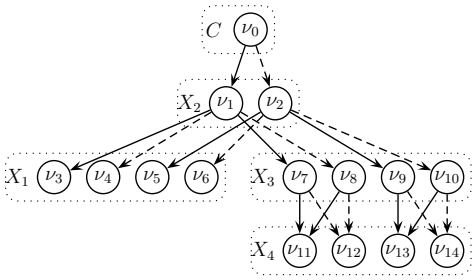
Example 5.1 (Constructing a PDG from a NB) *Consider the NB model $\mathcal{B}$ with four feature variables, that is $\mathbf{X} = \{X_1, \ldots, X_4\}$, and class C where all the variables are binary (see Fig. 1(b)). We can construct an equivalent PDG model using Algorithm 1. First, C is represented by a single node ν_0 containing the parameters $p^{\nu_0} = P^{\mathcal{B}}(C)$ inserted as root of the PDG structure. Next, every feature variable is connected underneath ν_0 represented each by two nodes connected under ν_0, yielding a PDG structure very similar to the NB structure as can be seen in Figure 6(a) which also shows the parameterisation of the PDG model.*

Example 5.2 (Constructing a PDG from a TAN) *Consider the TAN*



$p^{\nu_0} = P(C)$	$p^{\nu_1} = P(X_1 \mid C = 0)$
$p^{\nu_2} = P(X_1 \mid C = 1)$	$p^{\nu_3} = P(X_2 \mid C = 0)$
$p^{\nu_4} = P(X_2 \mid C = 1)$	$p^{\nu_5} = P(X_3 \mid C = 0)$
$p^{\nu_6} = P(X_3 \mid C = 1)$	$p^{\nu_7} = P(X_4 \mid C = 0)$
$p^{\nu_8} = P(X_4 \mid C = 1)$	

(a)



$p^{\nu_0} = P(C)$	$p^{\nu_1} = P(X_2 \mid C = 0)$
$p^{\nu_2} = P(X_2 \mid C = 1)$	$p^{\nu_3} = P(X_1 \mid X_2 = 0, C = 0)$
$p^{\nu_4} = P(X_1 \mid X_2 = 1, C = 0)$	$p^{\nu_5} = P(X_1 \mid X_2 = 0, C = 1)$
$p^{\nu_6} = P(X_1 \mid X_2 = 1, C = 1)$	$p^{\nu_7} = P(X_3 \mid X_2 = 0, C = 0)$
$p^{\nu_8} = P(X_3 \mid X_2 = 1, C = 0)$	$p^{\nu_9} = P(X_3 \mid X_2 = 0, C = 1)$
$p^{\nu_{10}} = P(X_3 \mid X_2 = 1, C = 1)$	$p^{\nu_{11}} = P(X_4 \mid X_3 = 0, C = 0)$
$p^{\nu_{12}} = P(X_4 \mid X_3 = 1, C = 0)$	$p^{\nu_{13}} = P(X_4 \mid X_3 = 0, C = 1)$
$p^{\nu_{14}} = P(X_4 \mid X_3 = 1, C = 1)$	

(b)

Figure 6. Examples of results of Algorithm 1. (a) shows the resulting PDG structure and parameters when translating the NB classifier with four features. (b) shows the resulting PDG structure and parameters when translating the TAN model of Figure 2(a).

model displayed in Figure 2(a) and assume all the variables are binary. The construction of an equivalent PDG using Algorithm 1 proceeds as follows: The class variable C is represented as a single node ν_0 containing the parameters $p^{\nu_0} = P^{\mathcal{B}}(C)$. Then, the root of the tree structure over the feature variables (X_2) is added under C represented by two nodes. At X_2, the variable tree structure branches and both X_1 and X_3 are connected as children of X_2 each represented by 4 nodes. Finally X_4 is connected as child of X_3, and connections are configured such that X_4 becomes dependent of C and X_3 only. The resulting structure and parameters can be seen in Fig. 6(b).

Lemma 5.1 *Let $\mathcal{B}$ be a BN classifier model with FAN structure. Then the FANToPDG($\mathcal{B}$) function of Algorithm 1 returns a valid PDG classifier.*

Proof: First, observe that in lines 2 to 8 nodes representing every variable are created, satisfying condition (1) of Definition 4.1. Second, in the **for** loop at line 14 we connect a set of nodes $\mathbf{V}_X$ with a set $\mathbf{V}_Y$, where $Y \in pa_{\mathcal{B}}(X)$. Remember that there exists a node $\nu_{\mathbf{w}} \in \mathbf{V}_X$ for every $X \in \mathbf{X}$ and $\mathbf{w} \in R(pa_{\mathcal{B}}(X))$. When connecting a set of nodes $\mathbf{V}_X$ representing feature variable X there are two possible scenarios:

(1) The only parent of X in $\mathcal{B}$ is C, then in line 17 a unique outgoing edge for ν_{root} for every $c \in R(C)$ is created satisfying condition (2) of Definition 4.1.

(2) When X has feature variable Y as parent in $\mathcal{B}$, then in the loop of line 21, a unique outgoing edge for every $\nu_{\mathbf{u}} \in \mathbf{V}_Y$ and every $y \in R(Y)$ is created. To realise this, observe that by the two nested loops we effectively iterate over all values $\mathbf{u} \in R(pa_{\mathcal{B}}(Y))$ and thereby visit all $\nu_{\mathbf{u}} \in \mathbf{V}_Y$ that were previously created in lines 2 to 8. This will ensure that condition (2) of Definition 4.1 will be satisfied.

□

Theorem 5.1 *Let $\mathcal{B}$ be a BN classifier model with FAN structure. Then the FANToPDG($\mathcal{B}$) function of Algorithm 1 returns a PDG model $\mathcal{G}$ for which:*

(1) $\mathcal{G}$ has the same number of parameters as $\mathcal{B}$, and
(2) $P^{\mathcal{G}} = P^{\mathcal{B}}$.

Proof: A BN model $\mathcal{B}$ represents distribution $P^{\mathcal{B}}$ by the factorisation given in Eq. (1), while PDG model $\mathcal{G}$ represents distribution $P^{\mathcal{G}}$ by the factorisation given in Eq. (4). In order to prove the theorem, it is enough to show that when $\mathcal{B}$ has FAN structure and $\mathcal{G}$ is constructed from $\mathcal{B}$ by Algorithm 1, Equations (1) and (4) contain exactly the same factors. We will prove this by induction in the size of the set $\mathbf{C}$. Remember that the PDG factorisation consists of the nodes being reached in the structure.

As the base case, assume that $|\mathbf{C}| = 1$, that is, $\mathbf{C}$ only contains the class variable C. $\mathcal{G}$ would then consist of the single node ν_{root} with parameters $p^{\nu_{root}} = P^{\mathcal{B}}(C)$ and therefore it trivially holds that the distributions $P^{\mathcal{G}}$ and $P^{\mathcal{B}}$ have the same factors.

Next, assume that the theorem is true (that is, both distributions have exactly the same factors) for $|\mathbf{C}| = n$. If we add a new feature variable X to the FAN model $\mathcal{B}$, according to the definition of FAN we find that the only factor that contains the new variable is $P^{\mathcal{B}}(X|pa_{\mathcal{B}}(X) = \mathbf{w})$. Before adding the last variable X to the PDG structure as a child of (feature or class) variable Y, by assumption any configuration $\mathbf{v} \in R(\mathbf{C})$ will reach the unique node $\nu_{\mathbf{u}} \in \mathbf{V}_Y$ where $\mathbf{u} = \mathbf{v}[pa_{\mathcal{B}}(Y)]$. In line 21 of Algorithm 1 it is ensured that $succ(\nu_{\mathbf{u}}, X, y) = \nu_{\mathbf{w}}$ where $\mathbf{w} = \mathbf{v}[pa_{\mathcal{B}}(X)]$ and $y = \mathbf{v}[Y]$. And as $\nu_{\mathbf{w}}$ contains the values $P^{\mathcal{B}}(X|pa_{\mathcal{B}}(X) = \mathbf{w})$, the theorem is true for $|\mathbf{C}| = n + 1$. □

5.1.1 Refining by merging nodes

From Algorithm 1 we can construct a PDG representation of any FAN classifier. In a learning scenario we wish to take advantage of the full expressibility of the PDG language and not just obtain an equivalent representation.

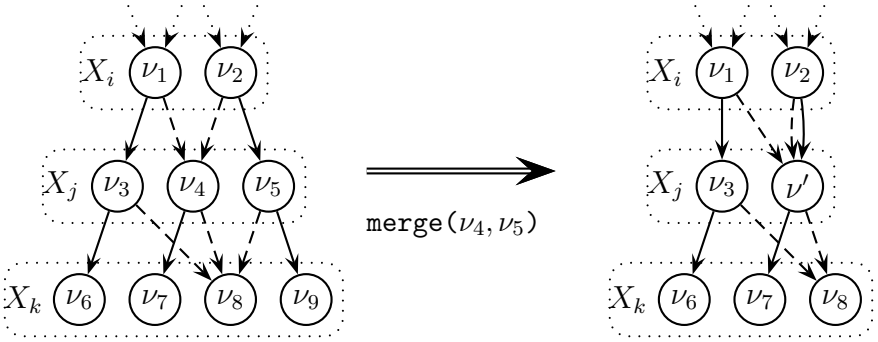


Figure 7. The structural changes resulting from the `merge` operation.

The `merge` operator is a binary operator that takes two nodes ν_1 and ν_2 representing the same variable in a PDG structure, and merges ν_2 into ν_1. This effectively reduces the number of parameters in the model, but will also introduce new independencies. The structural modification of merging node ν_2 into node ν_1 is performed by the following 2 steps:

(1) Move all links incoming to ν_2 towards ν_1.
(2) Remove any node that can not be reached from the root node by a directed path afterwards.

If the structure subjected to this transformation was a valid PDG structure, then the transformed one will also be a valid PDG structure. After removing all incoming links from ν_2, the structure is clearly not a PDG structure, as we have created one orphan node (ν_2) in addition to the original root node, and the structure is not a rooted DAG. However, the cleaning up done in the second step removes the newly created orphan node, and recursively any node that is orphaned from this removal.

An example of merging two nodes is shown in Figure 7, where on its left part, a section of a larger PDG structure is shown, while the right part displays the corresponding section after merging node ν_5 into ν_4, which effectively removes nodes ν_5 and ν_9 from the model.

Our criteria for choosing pairs of nodes for merging is based on the improvement in classification rate, that is, number of correctly classified individuals from our training data. To find the optimal pair $(\nu_1, \nu_2) \in \mathbf{V}_X \times \mathbf{V}_X$ for merging by exhaustive search over all such ordered pairs is inevitable a search that has polynomial time complexity in the size of $\mathbf{V}_X$. Instead, we employ a randomised merging strategy where random pairs are sampled from $\mathbf{V}_X$ and if the merge results in a gain in classification rate, it is implemented and otherwise it is not. This approach is very naïve indeed, but as initial experiments showed acceptable results compared to exhaustive search and superior execution time, we have not implemented more sophisticated methods.

Algorithm 2 This procedure learns a PDG classifier from a fully observed set of labelled data instances $\mathbf{D}$.

Input: Data $\mathbf{D}$ containing full observations of feature variables $\mathbf{X}$ labelled with class membership.

function LearnPDGC($\mathbf{D}$)
  Divide $\mathbf{D}$ into $\mathbf{D}_{train}$ and $\mathbf{D}_{hold\text{-}out}$
  Create new node ν representing C.
  $p^{\nu} \leftarrow \hat{P}_{\mathbf{D}_{train}}(C)$
  Instantiate PDG model $\mathcal{G}$ with ν as a root.
  while $\mathbf{X} \neq \emptyset$ **do**
    $\langle X_i, X_j \rangle \leftarrow \underset{X_i \in \mathbf{X}, X_j \in \mathcal{G}}{\text{argmax}}\ \texttt{CA}(X_i, X_j, \mathcal{G}, \mathbf{D}_{train}, \mathbf{D}_{hold\text{-}out})$.
    Add X_i as a child of X_j in $\mathcal{G}$.
    Merge nodes bottom up from new leafs $\mathbf{V}_{X_i}$.
  return $\mathcal{G}$.

5.2 Direct Learning of PDG Classifiers from Data

In this section we propose an algorithm for learning PDG classifiers directly from labelled data, with no need to refer to a previously existing BN classifier. The algorithm builds the PDG structure incrementally by adding variables from $\mathbf{X}$ to the variable tree structure with root C guided by classification accuracy on a hold-out set. We use a merging procedure that collapses two nodes into a single node if doing so increases classification accuracy measured on the hold-out data.

In Algorithm 2, the notation $\hat{P}_{\mathbf{D}}(X)$ refers to the maximum likelihood estimate of the marginal distribution of X, obtained from data $\mathbf{D}$. The function $\texttt{CA}(X_i, X_j, \mathcal{G}, \mathbf{D}_{train}, \mathbf{D}_{hold\text{-}out})$ in Algorithm 2 calculates the classification accuracy of the PDG classifier constructed from $\mathcal{G}$ by adding feature variable X_i as a fully expanded child of variable X_j with parameters estimated from $\mathbf{D}_{train}$ in $\mathcal{G}$, measured on data-set $\mathbf{D}_{hold\text{-}out}$. A variable X_i is added as a fully expanded child of X_j in $\mathcal{G}$ by adding for every node ν_j representing X_j and every value $x_j \in R(X_j)$ a new node ν_i representing X_i such that $succ(\nu_j, X_i, x_j) = \nu_i$. Adding a variable as a fully expanded child potentially results in many nodes and consequently many independent parameters for estimation. Parameters for node ν_i are computed as maximum likelihood estimates of the marginal probability $\hat{P}_{\mathbf{D}_{train}[\nu_i]}(X_i)$, where $\mathbf{D}_{train}[\nu_i] = \{d \in \mathbf{D}_{train} | reach_{\mathcal{G}}(i, d) = \nu_i\}$. To be sure we have a minimum amount of data for estimating we will collapse newly created nodes that are being reached by less than a minimum number of data instances. That is, assume we add X_i as a fully expanded child, resulting in nodes $\mathbf{V}_{X_i}$. We can then collapse two nodes ν_k and ν_l that both are reached by too few data cases into a single node ν_{k+l} that will then be reached by $\mathbf{D}_{train}[\nu_k] + \mathbf{D}_{train}[\nu_l]$. Such collapses are continued until all nodes are reached

by the minimum number of data instances, which for our experiments have been set to $5 \cdot |R(X_i)|$ for nodes representing variable X_i.

In Line 9 of Algorithm 2, nodes are merged bottom up from the newly created leaves using the merging procedure described in Section 5.1.1.

In Example 4.2, we illustrated some concepts that can be recognised efficiently by a PDG classification model. However, learning these classification models from data is inherently difficult as the concepts from Example 4.2 are both examples of a concept where no proper subset S of the set of features $\mathbf{X}$ reveals anything about the class label C of the individual, while the full set of features $\mathbf{X}$ determines C. Inducing such concepts from data generally requires that we consider all features $\mathbf{X}$ together, which is intractable in practice. Indeed Algorithm 2 does not guarantee that an efficient and accurate PDG model will be recovered even when such a model exists for the given domain.

6 Experimental Evaluation

In this section we present the results of an experimental comparison of our proposed PDG based classifier with a set of commonly used classifiers.

6.1 Experimental Setup

We have tested our proposed algorithms (Alg. 1 and Alg. 2) against well known and commonly used algorithms for learning classification models including NB, TAN, kdB and Classification Tree (CT) models, introduced in Section 3. For the kdB learning algorithm, we used $k = 4$. We have used the implementation of the models and their learning algorithms available in the Elvira system (The Elvira Consortium, 2002). The methods included in the comparison are the following:

NB: As the structure of the NB model is fixed, its learning reduces to the learning the parameters which is done by computing maximum likelihood estimates.

TAN: The algorithm that we have used for learning TAN models is the one proposed by Friedman et al. (1997). This algorithm uses the well-known algorithm by Chow and Liu (1968) to induce an optimal tree structure over the feature variables.

kdB: We use the so-called kdB-algorithm proposed by Sahami (1996) for learning kdB models. We configured the algorithm to assign at most 4 parents to each feature variable.

CT: For CT models we have used three different algorithms for learning the model. The classic ID3 algorithm (Quinlan, 1986), its successor the C4.5 algorithm (Quinlan, 1993) and lastly the more recent Dirichlet classification tree algorithm (Abellán and Moral, 2003).

6.2 An Initial Experiment

As an initial experiment, we have generated a set of labelled data-instances from the following concept over 9 binary feature variables $X_i : 0 \leq i \leq 8$ and binary class variable C:

$$C = (X_0 \vee X_1 \vee X_2) \wedge (X_3 \vee X_4 \vee X_5) \wedge (X_6 \vee X_7 \vee X_8)\,. \tag{9}$$

The concept of Eq. (9) consists of 3 disjunctions over 3 variables each, and the three disjunctions are then connected in a conjunction. We will refer to the concept in Eq. (9) as the *discon-3-3*. The feature variables of *discon-3-3* are assumed to be marginally independent and have a uniform prior distribution. We have designed this concept especially for exposing the expressive power of the PDG model over the traditional models (NB, TAN, kdB and CT). Following the terminology of Jaeger (2003) this concept is *order-3 polynomial separable* and by (Jaeger, 2003, Theorem 2.6) it is not recognisable by classification models of order lower than 3, that is, models that does not include associations of order 3 or higher. NB and TAN models are examples of association-1 and association-2 classifiers respectively. kdB models with $k = 4$ are examples of association-4 classifiers and may therefore be able to recognise the concept. CT models can recognise this concept and the same is true for PDG models.

Figure 8 shows an example of a PDG classifier that recognizes the *discon-3-3* concept, in the figure we have represented edges labelled with `true` as solid and edges labelled with `false` as dashed. By setting parameters $p^{\nu_{32}} = [1, 0]$ and $p^{\nu_{33}} = [0, 1]$ and all other parameters as uniform or any other positive distribution. The posterior probability of C given some complete configuration $\mathbf{x} \in R(\mathbf{X})$ will be a zero-one distribution modelling the boolean function of Eq. (9).

We have generated a database from the *discon-3-3* concept by enumerating all possible combinations of $R(\mathbf{X})$ and the corresponding class-label c which gives us a database of 512 labelled instances, 343 (or ≈67%) of which are positive examples. In Table 1 we have listed the mean classification rate (CR) from a 5-fold cross-validation and the mean size (S) of the models induced. Sizes refer to number of free parameters for the BN based classifiers as well as for the PDG based classifier, while for CT models the size refer to the number of leaf nodes in the tree. Each row corresponds to a specific model, PDG1 refers to

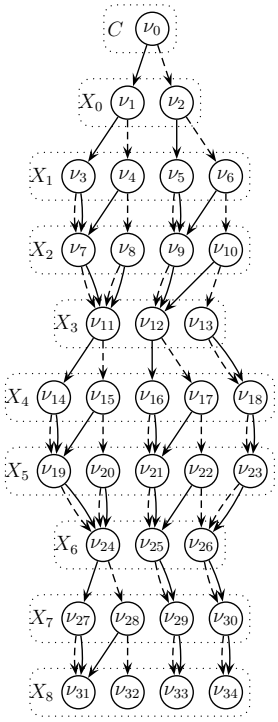


Figure 8. A PDG-classifier that recognises the concept of Eq. (9). Solid edges are are labelled with `true` and dashed edges are labelled `false`.

	CR	S
DIR	0.745	62.4
ID3	0.753	66.0
C4.5	0.753	66.0
TAN	0.698	35.0
NB	0.751	19.0
kdB	0.759	191.0
PDG1	0.722	31.0
PDG2	**0.825**	43.6

Table 1
Results of learning various kinds of classification models on a database sampled from the *discon-3-3* concept.

the model learnt from first inducing an equivalent TAN by Algorithm 1 and then merging nodes as described in Section 5.1.1, while PDG2 refers to the PDG classification model learnt directly from data by Algorithm 2.

From Table 1 we see first that no algorithm is able to learn a classifier that recognises the *discon-3-3* concept as they all have CR < 1. Next, it should be noticed that the direct learning of PDG classifiers (PDG2) is the most successful approach in this constructed example. Even the kdB classifier that uses 191 parameters on average compared to the PDG2 classifiers 43.6 parameters has a lower CR.

Name	Size	$\|\mathbf{X}\|$	$\|R(C)\|$	Baseline
australian [1]	690	14	2	0.555
car	1727	6	4	0.701
chess	3195	36	2	0.522
crx	653	15	2	0.547
ecoli	336	7	8	0.426
glass	214	9	6	0.355
heart [1]	270	13	2	0.556
image	2310	19	7	0.143
iris	150	4	3	0.333
monks-1	431	6	2	0.501
monks-2	600	6	2	0.657
monks-3	431	6	2	0.529
mushroom	5643	22	2	0.618
new-thyroid	215	5	3	0.698
nursery	12959	8	5	0.333
pima	768	8	2	0.651
postop	86	8	3	0.721
soybean-large	561	35	15	0.164
vehicle	846	18	4	0.258
voting-records	434	16	2	0.615
waveform	5000	21	3	0.339
wine	178	13	3	0.399
sat [1]	6435	36	6	0.238

Table 2
Data sets used in our experiments. The number of instances is listed in the Size column, $|\mathbf{X}|$ indicates the number of feature variables for the database, $|R(C)|$ refers to the number of different labels while Baseline gives the frequency of the most frequent class label. All databases are publicly available from the UCI repository.

In the following section we will investigate the performance of our proposals on a larger set of commonly used benchmark-data.

6.3 Main Experiments

In our main set of experiments we have used a sample of 23 data sets commonly used in benchmarking classifiers publicly available from the UCI repository (Newman et al., 1998). The datasets have been processed by removing the individuals with missing values and by discretising all continuous features using the k-means algorithm implemented in the Elvira System (The Elvira Consortium, 2002). A description of these dataset can be seen in Table 2.

6.3.1 Results

The results are listed in Tables 3 and 4. As in the initial experiment of Section 6.2 we have listed the mean classification rate (CR) from a 5-fold cross-validation and the mean size (S) of the models induced. As before, sizes refer to number of free parameters for the BN based classifiers as well as for the PDG based classifier, while for CT models the size refer to the number of leaf nodes in the tree. Each row corresponds to a specific model, PDG1 refers to the model learnt from first inducing an equivalent TAN by Algorithm 1 and then merging nodes as described in Section 5.1.1, while PDG2 refers to the

[1] Included in the UCI repository under the StatLog project.

PDG classification model learnt directly from data by Algorithm 2.

In order to determine whether or not there are significant differences in terms of classification accuracy among the tested classifiers, we have carried out a Friedman rank sum test (Demšar, 2006) using the classification rates displayed in Tables 3 and 4. According to the result of the test, there are no significant differences among the tested classifiers for the considered databases (p-value of 0.4656).

Regarding the convenience of transforming a BN into a PDG, we can say that a statistical comparison between methods TAN and PDG1 shows no significant differences between both of them (p-value of 0.9888 in a two-sided t-test). However, we find a slight edge in favour of PDG1, since out of the 23 used databases, PDG1 provides better accuracy than TAN in 11 databases, for only 9 with better performance of the TAN, besides 3 draws.

The same can be concluded if we examine PDG1 versus NB, when the first one reaches higher accuracy in 14 databases while NB is more successful in 9. Also, the comparison between NB and PDG2 is favourable to PDG2 in 12 cases for only 10 to NB.

Therefore, the experimental results show a competitive behaviour of the PDG classifiers in relation to the other tested models. Moreover, there is also a slight edge in favour of the PDG classifiers compared to their more close competitors, namely the NB and the TAN.

With respect to the comparison with tree-structured classifiers, PDGs have the added value that they are not just a blind model for classification, but actually a representation of the joint distribution of the variables contained in the model. Therefore, it can be efficiently used for other purposes as, for instance, probabilistic reasoning.

7 Conclusion

In this paper we have introduced a new model for supervised classification based on probabilistic decision graphs. The resulting classifiers are closely related to the so-called Bayesian network classifiers. Moreover, we have shown that any BN classifier with FAN structure has an equivalent PDG classifier with the same number of free parameters.

The experimental analysis carried out supports the hypothesis that the proposed models are competitive with the state-of-the-art BN classifiers and with classification trees.

	australian		car		chess		crx		ecoli	
	CR	S	CR	S	CR	S	CR	S	CR	S
DIR	0.828	226.4	0.746	579.2	0.929	78.8	0.818	216.0	0.472	193.6
ID3	0.787	592.8	0.750	964.8	**0.939**	88.4	0.761	590.4	0.537	2260.8
C4.5	0.778	563.2	0.752	966.4	0.929	95.6	0.760	582.8	0.520	2139.2
TAN	**0.859**	347.8	0.748	183.8	0.768	147.0	0.839	429.8	0.520	999.0
NB	0.846	81.0	0.727	63.0	0.593	75.0	**0.848**	111.0	**0.564**	231.0
kdB	0.842	27099.0	**0.774**	3023.0	0.840	1291.8	0.826	141063.0	0.523	64999.0
PDG1	0.835	289.8	0.745	150.8	0.799	125.4	0.844	378.0	0.550	882.2
PDG2	0.845	163.0	0.750	117.6	0.686	228.6	0.832	191.2	0.541	129.4

	glass		heart		image		iris		monks-1	
	CR	S	CR	S	CR	S	CR	S	CR	S
DIR	0.401	342.0	0.752	148.0	0.855	757.4	0.880	29.4	0.611	49.6
ID3	0.463	1282.8	0.741	238.0	**0.900**	5702.2	0.867	103.8	0.611	49.6
C4.5	0.382	1201.2	0.741	212.0	0.881	5674.2	0.867	103.8	0.611	49.6
TAN	0.467	989.0	0.793	222.2	0.852	2414.0	0.920	194.0	0.724	59.0
NB	0.356	221.0	**0.837**	71.0	0.713	510.0	*0.927*	50.0	0.586	23.0
kdB	0.284	78749.0	0.778	19095.0	0.858	249374.0	0.900	1874.0	0.593	611.0
PDG1	**0.472**	893.8	0.785	196.2	0.852	2108.4	*0.927*	192.4	0.724	59.0
PDG2	0.260	101.0	0.804	115.8	0.741	895.6	*0.927*	46.8	**0.824**	34.2

	monks-2		monks-3		mushroom		new-thyroid		nursery	
	CR	S	CR	S	CR	S	CR	S	CR	S
DIR	*0.777*	515.2	*1.000*	25.6	0.989	30.8	0.791	36.6	0.953	849.0
ID3	0.757	536.4	*1.000*	25.6	**0.991**	35.2	0.837	147.0	**0.976**	3980.0
C4.5	*0.777*	514.8	*1.000*	26.4	0.990	31.6	0.828	144.6	0.975	3972.0
TAN	0.632	48.6	0.979	59.0	0.977	711.0	0.814	254.0	0.931	314.0
NB	0.597	23.0	0.940	23.0	0.946	153.0	**0.926**	62.0	0.903	99.0
kdB	0.706	539.0	0.972	596.6	0.990	108218.2	0.809	9374.0	0.970	7949.0
PDG1	0.635	39.4	*1.000*	57.2	0.988	683.2	0.800	234.8	0.933	238.2
PDG2	0.662	48.8	0.963	35.8	0.955	210.8	0.828	59.6	0.928	370.4

	pima		postop		soybean-large		vehicle		voting-records	
	CR	S	CR	S	CR	S	CR	S	CR	S
DIR	0.702	659.6	**0.735**	3.0	0.859	630.0	0.683	746.4	0.945	51.6
ID3	0.642	1142.8	0.587	178.2	0.826	2109.0	0.645	2416.8	0.943	78.0
C4.5	0.620	1131.6	0.620	165.0	0.859	1668.0	0.654	2570.4	0.945	82.8
TAN	0.763	289.0	0.713	116.0	0.877	2789.0	**0.717**	1379.0	0.945	185.0
NB	*0.770*	65.0	0.701	47.0	0.877	944.0	0.609	291.0	0.901	65.0
kdB	0.723	21249.0	0.646	1986.2	0.849	137663.0	0.681	142499.0	**0.956**	4049.0
PDG1	0.763	255.4	0.678	111.8	**0.882**	2732.6	0.695	1268.6	0.931	160.2
PDG2	0.744	149.8	0.700	42.8	0.785	685.0	0.663	618.2	0.917	115.8

Table 3
Results of learning models from 20 of the datasets listed in Table 2 (see Table 4 for the remaining 3 datasets). For each dataset the mean classification rate from 5-fold cross validation (CR) is listed along with the mean size (S) of the models induced over the 5 folds.

	waveform		wine		sat	
	CR	S	CR	S	CR	S
DIR	0.741	1375.8	0.662	91.8	0.835	3481.2
ID3	0.696	6379.8	0.774	123.0	0.807	12169.2
C4.5	0.690	6408.6	0.701	120.6	0.791	13705.2
TAN	0.813	1214.0	0.904	734.0	0.855	4229.0
NB	0.804	254.0	**0.955**	158.0	0.801	869.0
kdB	0.739	129374.0	0.673	69374.0	**0.876**	483749.0
PDG1	0.819	883.6	0.888	734.0	0.854	2891.4
PDG2	**0.826**	1154.8	0.933	157.2	0.850	3388.2

Table 4
Results of learning models from the 3 last datasets listed in Table 2. For each dataset the mean classification rate from 5-fold cross validation (CR) is listed along with the mean size (S) of the models induced over the 5 folds.

As future work, we plan to study the problem of selecting features. All the models considered in this work include all the available features, but the selection of an appropriate feature set can significantly improve the accuracy of the final models.

Acknowledgements

This work has been supported by the Spanish Ministry of Education and Science, through projects TIN2004-06204-C03-01 and TIN2007-67418-C03-01,02.

References

Abellán, J., Moral, S., 2003. Building classification trees using the total uncertainty criterion. International Journal of Intelligent Systems 18, 1215–1225.

Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D., 1996. Context-specific independence in Bayesian networks. In: Horvitz, E., Jensen, F. (Eds.), Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence. Morgan & Kaufmann, pp. 115–123.

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

Breiman, L., Friedman, J., Stone, C. J., Olshen, R., 1984. Classification and regression trees. Chapman & Hall.

Bryant, R. E., 1992. Symbolic boolean manipulation with ordered binary decision diagrams. ACM Computing Surveys 24 (3), 293–318.

Castillo, E., Gutiérrez, J. M., Hadi, A. S., 1997. Expert systems and probabilistic network models. Springer-Verlag.

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

Demšar, J., 2006. Statistical comparison of classifiers over multiple datasets. Journal of Machine Learning Research 7, 1–30.

Domingos, P., Pazzani, M. J., 1997. On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning 29 (2-3), 103–130.

Friedman, N., Geiger, D., Goldszmidt, M., 1997. Bayesian network classifiers. Machine Learning 29, 131–163.

Jaeger, M., 2003. Probabilistic classifiers and the concepts they recognize. In: Proceedings of the Twentieth International Conference on Machine Learning. AAAI Press, pp. 266–273.

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., Silander, T., 2004. Learning probabilistic decision graphs. In: Proceedings of 2nd European Workshop on Probabilistic Graphical Models. pp. 113–120.

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

Jensen, F. V., 2001. Bayesian networks and decision graphs. Springer.

Jensen, F. V., Lauritzen, S. L., Olesen, K. G., 1990. Bayesian updating in causal probabilistic netwroks by local computation. Computational Statistics Quarterly 4, 269–282.

Lucas, P. J., 2002. Restricted Bayesian network structure learning. In: Gámez, J., Salmerón, A. (Eds.), Proceedings of the 1st European Workshop on Probabilistic Graphical Models (PGM'02). pp. 117–126.

Minsky, M., 1963. Steps toward artificial intelligence. In: Feigenbaum, E. A., Feldman, J. (Eds.), Computers and Thoughts. McGraw-Hill, pp. 406–450.

Newman, D., Hettich, S., Blake, C., Merz, C., 1998. UCI repository of machine learning databases: `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Pearl, J., 1988. Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Publishers.

Quinlan, J. R., 1986. Induction of decision trees. Machine Learning 1, 81–106.

Quinlan, J. R., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers.

Sahami, M., 1996. Learning limited dependence Bayesian classifiers. In: KDD-96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. pp. 335–338.

The Elvira Consortium, 2002. Elvira: An environment for probabilistic graphical models. In: Gámez, J., Salmerón, A. (Eds.), Proceedings of the First European Workshop on Probabilistic Graphical Models. pp. 222–230.