Technical Report
EHU-KZAA-TR-3-2010

UNIVERSITY OF THE BASQUE COUNTRY
Department of Computer Science and Artificial Intelligence

Learning Bayesian network classifiers for multi-dimensional supervised classification problems by means of a multi-objective approach

Juan Diego Rodríguez and Jose Antonio Lozano

March 2010

San Sebastian, Spain
http://www.ccia-kzaa.ehu.es/

Learning Bayesian network classifiers for multi-dimensional supervised classification problems by means of a multi-objective approach

Juan D. Rodríguez and Jose A. Lozano

Intelligent Systems Group
Department of Computer Science and Artificial Intelligence
University of the Basque Country
Paseo Manuel de Lardizábal 1, 20080. San Sebastian - Donostia, Spain
juandiego.rodriguez@ehu.es, ja.lozano@ehu.es
http://www.sc.ehu.es/isg

Abstract

A classical supervised classification task tries to predict a single class variable based on a data set composed of a set of labelled examples. However, in many real domains more than one variable could be considered as a class variable, so a generalization of the single-class classification problem to the simultaneous prediction of a set of class variables should be developed. This problem is called multi-dimensional supervised classification.

In this paper, we deal with the problem of learning Bayesian network classifiers for multi-dimensional supervised classification problems. In order to do that, we have generalized the classical single-class Bayesian network classifier to the prediction of several class variables. In addition, we have defined new classification rules for probabilistic classifiers in multi-dimensional problems.

We present a learning approach following a multi-objective strategy wich considers the accuracy of each class variable separately as the functions to optimize. The solution of the learning approach is a Pareto set of non-dominated multi-dimensional Bayesian network classifiers and their accuracies for the different class variables, so a decision maker can easily choose by hand the classifier that best suits the particular problem and domain.

1 Introduction

Supervised classification (Bishop, 2006; Duda et al., 2001) is one of the most important tasks in pattern recognition field. A supervised classification problem involves the learning, or induction, of a classification model or classifier. This classifier is a function that assigns one or more values to a set of class variables, or labels, based on the values of a set of predictive variables or features. The classifier is usually learnt from a set of labeled cases (the training set) by means of a classifier induction algorithm.

Classical supervised classification focuses on the prediction of a single class variable, but many real domains consider more than one class variable, so it would be useful to extend it to the prediction of multiple class variables.

We have called this problem multi-dimensional supervised classification, and it must not be confused with other classification tasks such as multi-class (Tax and Duin, 2002): problems with a

single class variable that can take more than two values, multi-task (Caruana, 1997): an inductive transfer approach, where a main task is predicted helped by the prediction of some extra tasks, or multi-label classification (Tsoumakas and Katakis, 2007): where an instance can be classified with several different labels. Note, however, that this last problem can be seen as a multi-dimensional classification problem where each label or category is a binary class variable whose value is one when the instance is included in that category or zero otherwise. Other classification tasks in pattern recognition can be seen as a multi-dimensional classification problem, such as structured prediction (Bakir et al., 2007; Daumé and Marcu, 2005), where there are several class variables with a conditional structure among them, or hierarchical classification (Dumais and Chen, 2000; Cesa-Bianchi et al., 2006), where there is a hierarchical structure (two or more levels) among the class variables.

There are several possibilities to adapt single-class classifiers to multi-dimensional classification problems, but none of them exactly captures the problem characteristics. One approach consists of constructing a single class variable that models all possible combinations of classes. This class variable models the Cartesian product of all the class variables. The problem arises because this compound class variable can easily end up with an excessively high cardinality. This leads to computational problems because of the high number of parameters the model has to estimate. Furthermore, the model does not reflect the real structure of the classification problem. Another approach is to develop multiple classifiers, one for each class variable. However, this approach does not capture the real characteristics of the problem either, because it does not model the correlations between the different class variables and so, it does not take advantage of the information that they may provide. The previous approaches are clearly insufficient and suboptimal for the resolution of problems where class variables have high cardinalities or high degrees of correlation among them. The basic idea of multi-dimensional classification is that the use of the correlations between class variables may help in the classification task, so it is important to model as well as possible the correlations among class variables.

Recent works (van der Gaag and de Waal, 2006; de Waal and van der Gaag, 2007) present Bayesian network classifiers that use the correlations among class variables to model multi-dimensional classification problems. They propose learning and inference algorithms for multi-dimensional Bayesian network classifiers, but they restrict the structure of the Bayesian networks. For these restricted models, the authors have proved that the complexity of the learning task is polynomial in the number of variables involved.

In this paper, we break these structural constraints allowing the learning of any multi-dimensional Bayesian network structure. For this purpouse, we have developed a multi-objective optimization structural learning approach whose objective is to maximize the accuracy of each class variable separately. Moreover, we realize that in multi-dimensional classification there could be more than one classification rule, and so, we have developed specific classification rules for these problems. A very preliminary version of this work was presented in Rodríguez and Lozano (2008).

The rest of the paper is organized as follows. In Section 2 the multi-dimensional supervised classification problem is proposed. Section 3 provides the multi-dimensional Bayesian network classifiers proposed in this paper. The multi-objective learning of the multi-dimensional Bayesian classifiers is introduced in Section 4. Finally, the simulation study and the conclusions are presented in sections 5 and 6 respectively.

2 Multi-dimensional supervised classification

In this section we present, in detail, the nature of the multi-dimensional supervised classification problem and how to define and evaluate a multi-dimensional classifier.

2.1 Multi-dimensional supervised classification problems

We call multi-dimensional supervised classification to the generalization of the classical supervised classification task where more than one class variable should be simultaneously predicted.

An approach to multi-dimensional supervised classification consists of building a classifier from training data in order to predict the value of a class vector of m class variables or labels $\mathbf{C} = (C_1, \cdots, C_m)$ given the predictive attributes or features $\mathbf{X} = (X_1, \cdots, X_n)$ of an unseen unlabeled instance $\boldsymbol{x} = (x_1, \cdots, x_n)$. We suppose that $(\boldsymbol{X}, \boldsymbol{C})$ is a random vector with a joint feature-label probability distribution $p(\boldsymbol{x}, \boldsymbol{c})$.
A classifier ψ is a function that maps $\boldsymbol{X}$ into $\boldsymbol{C}$:

$$\begin{array}{rll} \psi: & \{1,...,r_1\} \times ... \times \{1,...,r_n\} & \rightarrow \{1,...,t_1\} \times ... \times \{1,...,t_m\} \\ & (x_1, \cdots, x_n) & \mapsto (c_1, \cdots, c_m) \end{array}$$

where r_i and t_j are the cardinalities of the feature X_i (for $i = 1, \ldots, n$) and the class variable C_j (for $j = 1, \ldots, m$) respectively.

A classifier is learnt from a training set $D = \{(\boldsymbol{x}^{(1)}, \boldsymbol{c}^{(1)}), ..., (\boldsymbol{x}^{(d)}, \boldsymbol{c}^{(d)})\}$ with a classifier induction algorithm $A(\cdot)$. Given the induction algorithm $A(\cdot)$, which is assumed to be a deterministic function of the training set, the classifier obtained from a training set D is denoted as $\psi = A(D)$. Therefore, a multi-dimensional classification problem can be defined as the induction, from a data set D, of a classification function ψ that given a feature vector $\boldsymbol{x}$, returns a class vector $\boldsymbol{c}$.

2.2 Multi-dimensional classification rules

In probabilistic classification, the induction algorithm learns a probability distribution $p(\boldsymbol{x}, \boldsymbol{c})$ or $p(\boldsymbol{c}|\boldsymbol{x})$ from the training data and classifies a new instance based on this. For that purpose, a classification rule must be defined. The multi-dimensional nature of the problem allows us to develop different classification rules that would make no sense in single-class classification because they take into account multiple class variables.

In single-class supervised classification, the most commonly used classification rule returns the most likely class value given the features. We call it *one-dimensional classification rule*:

$$\begin{array}{rll} \hat{c} = & argmax_c \{p(c|x_1, \cdots, x_n)\} & = argmax_c \left\{ \dfrac{p(c, x_1, \cdots, x_n)}{p(x_1, \cdots, x_n)} \right\} \\ = & argmax_c \{p(c, x_1, \cdots, x_n)\} & \end{array}$$

This classification rule can be easily generalized to the prediction of more than one class variable. The multi-dimensional classifier returns the most probable combination of class variables given the features. We call it *joint classification rule*:

$$\begin{array}{rll} (\hat{c}_1, \cdots, \hat{c}_m) = & argmax_{c_1, \cdots, c_m} & \{p(c_1, \cdots, c_m|x_1, \cdots, x_n)\} \\ = & argmax_{c_1, \cdots, c_m} & \left\{ \dfrac{p(c_1, \cdots, c_m, x_1, \cdots, x_n)}{p(x_1, \cdots, x_n)} \right\} \\ = & argmax_{c_1, \cdots, c_m} & \{p(c_1, \cdots, c_m, x_1, \cdots, x_n)\} \end{array}$$

However, we can go beyond and develop other classification rules in order to estimate the values of the class variables. We propose another classification rule that consists on marginalizing each class variable for the rest of class variables simultaneously. We call it *marginal classification*

rule and estimates the value of each class variable C_j (with $j = 1, \cdots, m$) as follows:

$$\begin{aligned}\hat{c}_j = &\quad argmax_{c_j} \quad \{p(c_j|x_1, \cdots, x_n)\} = argmax_{c_j} \left\{ \sum_{\mathbf{c}_{\neg j}} p(c_j, \mathbf{c}_{\neg j}|x_1, \cdots, x_n) \right\} \\ = &\quad argmax_{c_j} \quad \left\{ \sum_{\mathbf{c}_{\neg j}} \frac{p(c_j, \mathbf{c}_{\neg j}, x_1, \cdots, x_n)}{p(x_1, \cdots, x_n)} \right\} = argmax_{c_j} \left\{ \sum_{\mathbf{c}_{\neg j}} p(c_j, \mathbf{c}_{\neg j}, x_1, \cdots, x_n) \right\}\end{aligned}$$

where $\mathbf{c}_{\neg j} = \{c_1, \cdots, c_{j-1}, c_{j+1}, \cdots, c_m\}$.

The previous classification rules estimate the class variables values in one step: the joint classification rule estimates all the class variables simultaneously and the marginal classification rule estimates each class variable separately. In multi-dimensional classification we can propose classification rules that estimate the class variables in a procedure with more than one step.

We propose a classification rule based on the marginal values of the rest of the class variables. In the first step, each class variable is estimated using the marginal classification rule previously defined, and then, each class variable value is estimated again using the class values for the rest of the class variables estimated at the end of the first step as evidence. We call it *conditional classification rule*:

$$\begin{aligned}\widehat{c}_j = &\quad argmax_{c_j}\{p(c_j, \widehat{\mathbf{c}}^*_{\neg j}|x_1, \cdots, x_n)\} \quad = argmax_{c_j} \left\{ \frac{p(c_j, \widehat{\mathbf{c}}^*_{\neg j}, x_1, \cdots, x_n)}{p(x_1, \cdots, x_n)} \right\} \\ = &\quad argmax_{c_j}\{p(c_j, \widehat{\mathbf{c}}^*_{\neg j}, x_1, \cdots, x_n)\}\end{aligned}$$

where $\widehat{\mathbf{c}}^*_{\neg j}$ are the predicted values for $\{C_1, \cdots, C_{j-1}, C_{j+1}, \cdots, C_m\}$ using the marginal classification rule.

Moreover, we can extend the previous classification rule and continue estimating the class values taking into account the estimated values for the rest of the classes in the previous step. This procedure should continue until a stop criterion is reached, for example when the estimated class values do not change in two consecutive steps. We call it *iterative classification rule*:

$$\begin{aligned}\widehat{\mathbf{c}}^0_j = &\quad argmax_{c_j}\{p(c_j, \widehat{\mathbf{c}}^0_{\neg j}|x_1, \cdots, x_n)\} \quad = argmax_{c_j} \left\{ \frac{p(c_j, \widehat{\mathbf{c}}^0_{\neg j}, x_1, \cdots, x_n)}{p(x_1, \cdots, x_n)} \right\} = \\ = &\quad argmax_{c_j}\{p(c_j, \widehat{\mathbf{c}}^0_{\neg j}, x_1, \cdots, x_n) \\ \widehat{\mathbf{c}}^s_j = &\quad argmax_{c_j}\{p(c_j, \widehat{\mathbf{c}}^{(s-1)}_{\neg j}|x_1, \cdots, x_n)\} \quad = argmax_{c_j} \left\{ \frac{p(c_j, \widehat{\mathbf{c}}^{(s-1)}_{\neg j}, x_1, \cdots, x_n)}{p(x_1, \cdots, x_n)} \right\} = \\ = &\quad argmax_{c_j}\{p(c_j, \widehat{\mathbf{c}}^{(s-1)}_{\neg j}, x_1, \cdots, x_n)\end{aligned}$$

where $\widehat{\mathbf{c}}^0_j$ are the predicted values for $\{C_1, \cdots, C_{j-1}, C_{j+1}, \cdots, C_m\}$ using the marginal classification rule and $\widehat{\mathbf{c}}^{(s-1)}_{\neg j}$ are the predicted values for $\{C_1, \cdots, C_{j-1}, C_{j+1}, \cdots, C_m\}$ in the $s-1$ step using the conditional classification rule.

2.3 Multi-dimensional classification evaluation

Once a classifier is constructed it is needed to measure its associated error. The *prediction error* of a single-class classifier ψ is the probability of the wrong classification of unlabeled instances $\mathbf{x}$ and is denoted as $\epsilon(\psi)$:

$$\epsilon(\psi) = p(\psi(\mathbf{X}) \neq C) = E_{\mathbf{X}}[\delta(c, \psi(\mathbf{x}))] \tag{1}$$

where $\delta(x, y)$ is a loss function whose result is 1 if $x \neq y$ and 0 if $x = y$.

However, in multi-dimensional classification we can measure the correctness of an instance in different ways:

- *Joint evaluation*: This consists of evaluating the estimated values of all class variables simultaneously, that is, it only registers a success if all the classes are correctly predicted, and otherwise registers an error (see Equation 2). This rule generalizes the previous single-class evaluation measure to multi-dimensional classification.

$$\epsilon(\psi) = p(\psi(\mathbf{X}) \neq \mathbf{C}) = E_{\mathbf{X}}[\delta(\mathbf{c}, \psi(\mathbf{x}))] \tag{2}$$

- *Single evaluation*: This consists of checking separately if each class is correctly classified. For example, if we classify an instance $\boldsymbol{x}$ as $(\hat{c}_1 = 0.\hat{c}_2 = 1)$ and the real value is $(c_1 = 0.c_2 = 0)$, we count $\hat{c}_1$ as a success and $\hat{c}_2$ as an error. This approach provides one performance function for each class C_j (for $j = 1, \cdots, m$). The output of this evaluation is a vector $\boldsymbol{\epsilon}$ of size m with the performance function of the multi-dimensional classifier for each class variables (see Equation 3):

$$\epsilon_j(\psi) = p(\psi_j(\mathbf{X}) \neq C_j) = E_{\mathbf{X}}[\delta(c_j, \psi_i(\mathbf{x}))] \tag{3}$$

  where $\psi_j(\mathbf{X})$ is the estimated label of the multi-dimensional classifier for the j-th class variable.

Ideally, we would like to calculate exactly the error of a classifier, but in most real world problems the feature-label probability distribution $p(\mathbf{x}, \mathbf{c})$ is unknown. So, the prediction error of a classifier ψ is also unknown, it can not be exactly computed, and thus, must be estimated from data. There are several estimators of the prediction error, from the simple Resubstitution (Devroye and Wagner, 1979) and Hold-out (McLachlan, 1992) to the more complex Bootstrap (Efron and Tibshirani, 1993) and Bolstered (Braga-Neto et al., 2004). In this work we use one of the most popular error estimation techniques: k-fold cross-validation (k-cv) (Stone, 1974) in its repeated version. In k-cv the dataset is divided into k folds, a classifier is learnt using $k-1$ folds and an error value is calculated by testing the classifier in the remaining folds. Finally, the k-cv estimation of the error is the average value of the errors committed in each fold. The repeated r times k-cv consists of estimating the error as the average of r k-cv estimations with different random fold partitions. This method considerably reduces the variance of the error estimation (Rodríguez et al., 2010).

In multi-dimensional classification we could be interested in learning the most accurate classifier for all class variables simultaneously (measured with a joint evaluation). However, it makes sense to find the most accurate classifiers for each single class variable (measured with single evaluations). The learning approach that we present considers the single evaluation of each class variable as the functions to optimize in a multi-objective optimization problem. We will present this approach in Section 4.

3 Multi-dimensional Bayesian networks classifiers

3.1 Preliminaries and notation

In this paper we generalize the single-class oriented Bayesian network classifiers to domains with more than one class variable. Bayesian networks (Pearl, 1988) are powerful tools for knowledge representation and inference under uncertainty conditions. These formalisms have been extensively used as classifiers (Langley et al., 1992; Larrañaga et al., 2005) and have become a classical and well-known classification paradigm. In spite of the popularity of Bayesian network classifiers, few works have taken into account their generalization to multiple class variables (van der Gaag and de Waal, 2006; de Waal and van der Gaag, 2007).

A Bayesian network is a pair $B = (S, \mathbf{\Theta})$ where S is a directed acyclic graph (DAG) whose vertices correspond to random variables and whose arcs represent conditional (in)dependence relations among variables, and $\mathbf{\Theta}$ is a set of parameters [1]. We consider Bayesian networks over a finite set $\mathbf{V} = \{C_1, \cdots, C_m, X_1, \cdots, X_n\}$ where each variable C_j and X_i take a finite set of values. Θ is formed by parameters $\theta_{c_j|\mathbf{Pa}(c_j)}$ and $\theta_{x_i|\mathbf{Pa}(x_i)}$ for each value that C_j and X_i can takes and for each value assignment $\mathbf{Pa}(x_i)$ and $\mathbf{Pa}(c_j)$ to the sets $\mathbf{Pa}(X_i)$ and $\mathbf{Pa}(C_j)$ of parents of X_i and C_j respectively.

A Bayesian network classifier is usually represented as a Bayesian network with a particular structure: the class variables are on the top of the graph and are the parents of the predictive variables, i.e. there are no arcs from predictors to class variables.

The network B defines a joint probability distribution $p(c_1, \cdots, c_m, x_1, \cdots, x_n)$ given by:

$$p(c_1, \cdots, c_m, x_1, \cdots, x_n) = \prod_{j=1}^{m} \theta_{c_j|\mathbf{Pa}(c_j)} \prod_{i=1}^{n} \theta_{x_i|\mathbf{Pa}(x_i)}$$

The Bayesian network, $B = (S, \mathbf{\Theta})$, can be defined by an expert who is able to list the conditional independences between problem variables or, more frequently, it can be learnt from domain data. In general, the problem of learning a Bayesian network classifier from data can be seen as: given a training set $D = \{(\mathbf{x}^{(1)}, \mathbf{c}^{(1)}), \cdots, (\mathbf{x}^{(d)}, \mathbf{c}^{(d)})\}$ of d instances of $(\mathbf{X}, \mathbf{C})$, find a network B that best matches D (Friedman et al., 1997)

There are two main approaches for automatically learning Bayesian networks from data (Neapolitan, 2003): while the first one is based on a score + search process in the space of possible structures, the second tries to detect the conditional independences by means of statistical hypothesis tests. In this work, we use a score + search multi-objective process. Our scores are the k-fold cross-validation estimations of the accuracies of each class variable separately.

3.2 Structure of multi-dimensional class Bayesian networks classifiers

A multi-dimensional class Bayesian network classifier is a generalization of the classical one-class variable Bayesian classifiers for domains with multiple class variables (van der Gaag and de Waal, 2006). It models the relationships between the variables by directed acyclic graphs (DAG) over the class variables and over the feature variables separately, and then connects the two sets of variables by means of a bipartite directed graph. So, the DAG structure $S = (\mathbf{V}, \mathbf{A})$ has the set $\mathbf{V}$ of random variables partitioned into the sets $\mathbf{V}_C = \{C_1, \ldots, C_m\}$, $m > 1$, of class variables and the set $\mathbf{V}_F = \{X_1, \ldots, X_n\}$, $n \geq 1$, of feature variables. Moreover, the set of arcs $\mathbf{A}$ can be partitioned into three sets: $\mathbf{A}_{CF}$, $\mathbf{A}_C$ and $\mathbf{A}_F$ with the following properties:

[1] This definition takes into account that the objective is to define multi-dimensional Bayesian classifiers

- $\mathbf{A}_{CF} \subseteq \mathbf{V}_C \times \mathbf{V}_F$ is composed of the arcs between the class variables and the feature variables, so we can define the feature selection subgraph of S as $S_{CF} = (\mathbf{V}, \mathbf{A}_{CF})$. This subgraph represents the selection of features that seems relevant for classification given the class variables.
- $\mathbf{A}_C \subseteq \mathbf{V}_C \times \mathbf{V}_C$ is composed of the arcs between the class variables, so we can define the class subgraph of S induced by $\mathbf{V}_C$ as $S_C = (\mathbf{V}_C, \mathbf{A}_C)$.
- $\mathbf{A}_F \subseteq \mathbf{V}_F \times \mathbf{V}_F$ is composed of the arcs between the feature variables, so we can define the feature subgraph of S induced by $\mathbf{V}_F$ as $S_F = (\mathbf{V}_F, \mathbf{A}_F)$.

In Figure 1, we show a multi-dimensional class Bayesian network classifier with 3 class variables and 5 features and its partition into the three subgraphs.

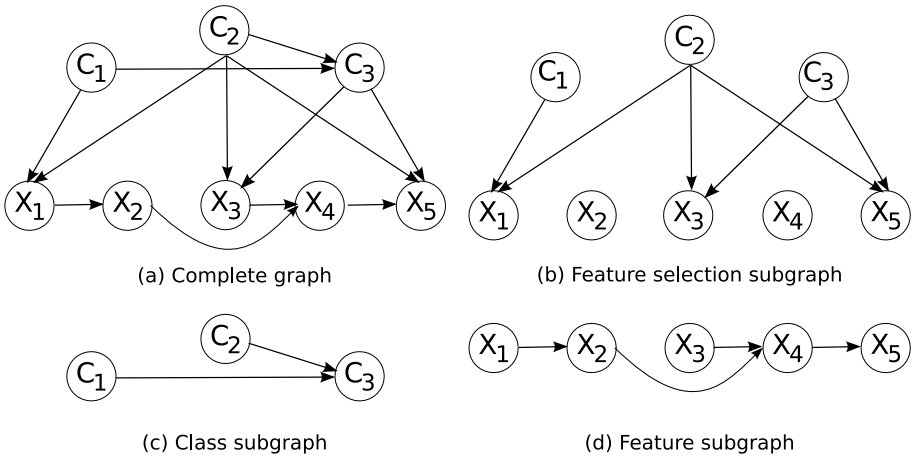

Figure 1: A multi-dimensional class Bayesian network classifier and its division

Depending on the structure of the class subgraph and the feature subgraph, van der Gaag and de Waal (2006) and de Waal and van der Gaag (2007) distinguish the following sub-families of multi-dimensional class Bayesian network classifiers [2]:

- *Multi-dimensional naive Bayes classifier (MDnB)*: the class subgraph and the feature subgraph are empty and the feature selection subgraph is complete.
- *Multi-dimensional tree-augmented classifier (MDTAN)*: both the class subgraph and the feature subgraph are directed trees.
- *Multi-dimensional polytree-augmented classifier (MDPoly)*: both the class subgraph and the feature subgraph are polytrees.

In addition to these structures, we have considered another structure for experimental purpouses:

- *Multi-dimensional J/K dependences Bayesian classifier (MDJ/K)*: This structure allows each class variable C_i to have, apart from the class variables, a maximum of J dependences with other class variables C_j, and each predictive variable X_i to have a maximum of K dependences with other predictive variables X_j.

[2] In van der Gaag and de Waal (2006) and de Waal and van der Gaag (2007) they use the term *fully* instead of *multi-dimensional* to name the classifiers.

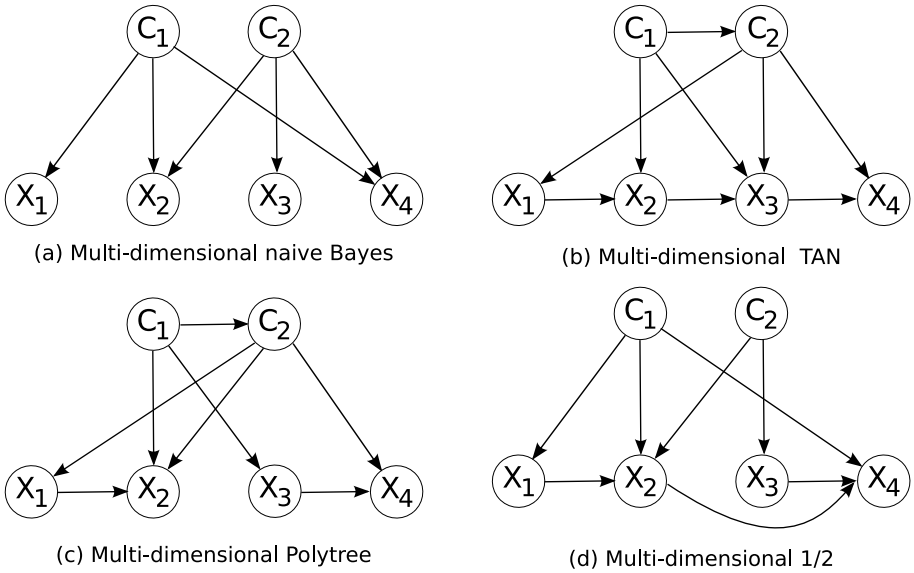

Figure 2: Multi-dimensional class Bayesian classifiers

In Figure 2, we show the different families of multi-dimensional classifiers.

In van der Gaag and de Waal (2006) and de Waal and van der Gaag (2007), the authors present a single-objective learning approach of MDnB and MDTAN classifiers [3] In this work we propose a multi-objective learning approach of multi-dimensional Bayesian classifiers. This approach also allows to learn classifiers without structural restrictions in either the class subgraph or the feature subgraph. In Section 4.3 we will introduce this structural learning approach in detail.

4 Learning multi-dimensional Bayesian network classifiers by means of multi-objective optimization

4.1 Multi-objective approach to multi-dimensional classification

As we have seen in Section 3.1, in one-class supervised classification we have to find a classifier that maximizes the accuracy of the class variable given an data set of instances $D = \{(\mathbf{x}^{(1)}, \mathbf{c}^{(1)}), \cdots, (\mathbf{x}^{(d)}, \mathbf{c}^{(d)})\}$. In multi-dimensional classification, the aim could be to find a classifier that maximizes the accuracy of all the class variables simultaneously (joint evaluation), or to find the classifier that maximizes the accuracy of each class variable (single evaluation). Our approach considers the learning of multi-dimensional Bayesian classifiers whose objective is to maximize the accuracy of each class variable. In order to carry out this approach, we suppose that there are classifiers whose accuracy can not been improved for one class variable without getting worse for any other class variable.

To carry out this approach to multi-dimensional classification by means of multi-objective optimization, we use a well-known multi-objective evolutionary algorithm (MOEA): the *multi-*

[3] In this article we have called them vG-MDnB and dW-MDTAN.

objective evolutionary algorithm based on decomposition (Zhang and Li, 2007; Li and Zhang, 2009) (MOEA/D). We have chosen this algorithm because of its success in experimental domains.

Other multi-objective approaches to supervised classification have been developed by the research community, but none of them takes into account multi-dimensional class prediction. All the approaches developed so far have focused on a single class variable and try to optimize different aspects such as: accuracy of the classifier and number of selected attributes, sensitivity on ROC curves, rule mining and partial classification, model accuracy versus model complexity, feature selection, accuracy on two different data sets or ensemble learning by means of integration of diverse classifiers (Freitas, 2004; Handl et al., 2007).

4.2 Multi-objective optimization

A multi-objective optimization problem (MOP) can be defined as an optimization problem with multiple objectives measured with different performance functions, usually in conflict with each other, and a set of restrictions. Hence, the optimization consists of finding such a solution which would give the values of all the objective functions acceptable to a decision maker (Osyczka, 1985), who have to chose the prefered optimal solution. The aim is to find good compromises (trade-offs) rather than a single solution (Coello et al., 2006).

Formally, a multi-objective optimization problem can be formulated as finding the vector $\mathbf{x}$ that satisfies l inequality restrictions $g_i(\mathbf{x}) \geq 0$ for $i = 1, 2, \cdots, l$ and k equality restrictions $h_i(\mathbf{x}) = 0$ for $i = 1, 2, \cdots, k$ and optimizes (maximizes or minimizes) the vector of objective functions:

$$\mathbf{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})]$$

An important concept in multi-objective optimization is the **Pareto dominance**: A vector $\mathbf{u} = (u_1, \cdots, u_m)$ is said to *dominate* $\mathbf{v} = (v_1, \cdots, v_m)$ (denoted by $\mathbf{u} \preceq \mathbf{v}$) if and only if $\mathbf{u}$ is partially less (on minimization) than $\mathbf{v}$. That is $\forall i \in \{1, \cdots, m\}, u_i \leq v_i \wedge \exists j \in \{1, \cdots, m\} : u_j < v_j$

A *Edgeworth-Pareto optimal solution* (Stadler, 1988) is a *nondominated* solution, that is, a solution that is impossible to improve in any objective function without a simultaneous worsening in some other objectives. The set of Pareto optimal solutions composes a Pareto optimal set and their images form a Pareto front. The expected solution of a multi-objective optimization problem is therefore, a Pareto front representing the values of the performance functions for each objective. The Pareto front usually contains more than one element because there exist different trade-off solutions to the problem. So, in practice, a human decision maker have to choose the most suitable solution.

MOEA/D decomposes the multi-objective optimization problem into a number of scalar optimization subproblems and then optimizes them simultaneously. We use a Tchebycheff approach (Miettinen, 1999) to decompose the problem into P subproblems by using a weigh vector $(\boldsymbol{\lambda^1}, \cdots, \boldsymbol{\lambda^P})$. It maintains a population of classifiers composed of the best solutions found so far for each subproblem. At each step of the algorithm, and for each subproblem i, it develops a new solution crossing current solutions in the neighbour of the i-th subproblem (in the proposed experimentation we use one-point cross-over for binary solutions and PMX cross-over for permutation solutions (Larrañaga et al., 1999)). A neighborhood of a weight vector $\boldsymbol{\lambda^j}$ is deïňĄned as a set of its T closest weight vectors in $(\boldsymbol{\lambda^j_1}, \cdots, \boldsymbol{\lambda^j_m})$. The new solution replaces a maximum of nr solutions of the neighbourhood that are improved by the new solution. Finally, at each iteration of the problem we maintain a external population with the best non-dominated solutions found so far.

The main input parameters for the algorithm are:

- P: The number of subproblems in which we decompose the multi-objective optimization problem.
- $(\boldsymbol{\lambda^1}, \cdots, \boldsymbol{\lambda^P})$: A uniform spread of weight vectors.
- T: The number of neighbors of each subproblem.
- nr: Number of replacements in the neighbourhood.
- A stop criterion.

The output is the set of best non-dominated solutions found during the search.

The algorithm works as follows:

- **Step 1: Initialization.**
  - **Step 1.1**: Initialize an external population EP.
  - **Step 1.2**: Compute Euclidean distances between the P weight vectors and find the T closest neighbors of each subproblem.
  - **Step 1.3**: Generate an initial population.
  - **Step 1.4**: Initilize $\boldsymbol{Z}$, a vector with the best solution for each subproblem.
- **Step 2: Update.** For $i = 1, \cdots, P$ do
  - **Step 2.1**: Reproduction. Selection of 2 solutions and generation of a new solution based on genetic operators.
  - **Step 2.2**: Improvement of the new solution if needed.
  - **Step 2.3**: Update of neighborhood solutions.
  - **Step 2.4**: Update of external population EP.
    - Remove from EP all the solutions dominated by the new solutions.
    - Add the new solution to EP if no solution in EP dominates it.
- **Step 3: Stoping Criteria**

4.3 Structural learning approach

This section describes in detail the multi-objective optimization approach to learn the multi-dimensional class Bayesian classifier structures proposed in this work.

In our approach, we use a multi-objective optimization problem to carry out the learning of the structure S of a multi-dimensional Bayesian network classifier B. The search space is composed of the multi-dimensional Bayesian classifiers with no structural restrictions in A_C, A_F and A_{CF} sets of arcs. The objective functions of the multi-objective optimization problem are the r repeated k-fold cross-validation error estimations ($k$cv) of each class variable separately (the *single evaluation* of each class):

$$\mathbf{kcv}(\boldsymbol{\psi}) = [kcv_1(\boldsymbol{\psi}), \cdots, kcv_m(\boldsymbol{\psi})]$$

where $\boldsymbol{\psi}$ is a multi-dimensional classifier.

To carry out the search procedure, each possible multi-dimensional Bayesian network classifier structure is codified by a vector formed by three parts (Figure 3):

- The first part is a permutation of the class variables V_C and represents an ancestral order over them. Its size is equal to the number of class variables.
- The second part is a binary vector that represents all the possible arcs of the feature selection subgraph A_{CF}. Its size is equal to the number of class variables times the number of predictive variables.
- The third part is a permutation of the features V_F and represents an ancestral order over them. Its size is equal to the number of predictive variables.

Figure 3: Codification of an individual

To recover a Bayesian network classifier structure, each individual is decodified as follows:

- The first permutation part represents an ancestral order in V_C. Therefore, each class variable can be a parent of its successors in the ancestral order. In order to determine the set of arcs in A_C, we carry out a statistical test (see below).
- The binary part represents A_{CF}. A value 1 represents an arc between a class variable and a feature.
- The last permutation part represents an ancestral order in V_F. We let each predictive variable be a parent of its successors in the ancestral order. We carry out a statistical test (see below), in order to determine the set of arcs in A_F.

The independence test we use to determine if a dependence between two variables is strong enough to be part of the model is based on the mutual information between them: It is known (Kullback, 1959) that $2N\hat{I}(X_i, X_j)$, if X_i and X_j are independent, asymptotically follows a χ^2 distribution with $(r_i - 1)(r_j - 1)$ degrees of freedom where N is the number of cases, X_i and X_j are random variables and r_i and r_j are the cardinality of X_i and X_j respectively: $Lim_{N\to\infty} 2N\hat{I}(X_i, X_j) \rightsquigarrow \chi^2_{(r_i-1)(r_j-1)}$.

Based on this result, a null hypothesis test can be carried out in a multi-dimensional Bayesian network classifier to check the possible dependences in A_C. The null hypothesis H_0 is: "the random variables C_i and C_j are independent". So, if the quantity $2N\hat{I}(C_i, C_j)$ surpasses a threshold s_α for a given test size $\alpha = \int_{s_\alpha}^{\infty} \chi^2_{(t_i-1)(t_j-1)} ds$, where t_i is the cardinality of C_i, and t_j the cardinality of C_j, the null hypothesis is rejected and there is a dependence between C_i and C_j. Therefore the arc between C_i and C_j is included in the model. This test was used on single-class Bayesian network classifiers to check the dependences among the class variables and the features (Blanco, 2005).

In this work, we use the conditional mutual information between a feature X_i and a feature X_j given its parents $\mathbf{Pa}(X_j)$ to determine if the relation between two features X_i and X_j in A_F should be included in the model. We have generalized the previous result to the case of conditional mutual information as follows (Kullback, 1959):

$$Lim_{N\to\infty} 2N\hat{I}(X_i, X_j|\mathbf{Pa}(X_j)) \rightsquigarrow \chi^2_{(r_i-1)(r_j-1)(|\mathbf{Pa}(X_j)|)}$$

where r_i is the cardinality of X_i, r_j the cardinality of X_j and $|\mathbf{Pa}(X_j)|$ the cardinality of the parents of X_j.

Analogously to the hypothesis test described before, based on these results we can perform the following conditional independence test. The null hypothesis H_0 is: "the random variables X_i and X_j are conditionally independent given $\mathbf{Pa}(X_j)$". So, if the quantity $2N\hat{I}(X_i, X_j|\mathbf{Pa}(X_j))$ surpasses a threshold s_α for a given test size $\alpha = \int_{s_\alpha}^{\infty} \chi^2_{(r_i-1)(r_j-1)(|\mathbf{Pa}(X_j)|)} ds$, the null hypothesis is rejected and the random variables X_i and X_j are considered dependent given $\mathbf{Pa}(X_j)$. Therefore the arc is included in the model.

For example, given a problem with 2 class variables and 3 features, the following individual $(1, 2|1, 0.1, 0.1, 0|2, 1, 3)$, is decodified in the following way:

In order to build A_C, we use the first permutation part formed by $(1, 2, \cdots)$. It represents the ancestral order (C_1,C_2). If $2NI(C_1, C_2)$ surpasses the previously defined independence test, the arc is included in the model. We can suppose that in this case the arc is included in the model. The second part is a binary vector formed by $(\cdots, 1, 0.1, 0.1, 0.\cdots)$ and represent the dependences among the class variables and the features, that is, A_{CF}. There are three features and two class variables, so the first 3 positions of this part $(\cdots, 1, 0.1, \cdots)$ represents the dependences between the first class variable C_1 and the features and the following 3 positions $(\cdots, 0.1, 0.\cdots)$ the dependences of the second class variable C_2 with the features. A 1 in a position represents an arc. In this case there are dependences from C_1 to X_1 and X_3, and from C_2 to X_2. Finally, we have to build A_F. In order to do this, we use the last permutation part formed by $(\cdots, 2, 1, 3)$. This represents the ancestral order (X_2,X_1,X_3). The arcs that surpass the previously defined conditional independence test are included in the model. In this case we have to check $2NI(X_2, X_1|\mathbf{Pa}(X_1))$, $2NI(X_2, X_3|\mathbf{Pa}(X_3))$ and $2NI(X_1, X_3|\mathbf{Pa}(X_3))$. In this example, we can suppose that the only conditional dependence among variables that does not surpass the independence test is the one from X_2 to X_1 ($2NI(X_2, X_1|\mathbf{Pa}(X_1))$), and therefore, it is the only arc among features that is not included in the model. The obtained classifier structure is given in Figure 4.

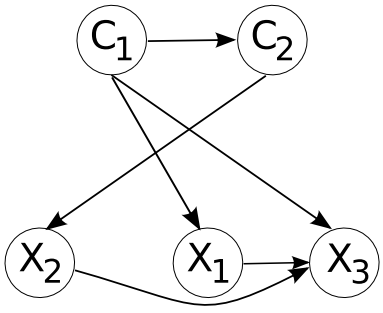

Figure 4: The classifier encoded in the individual $(1, 2|1, 0.1, 0.1, 0|2, 1, 3)$. We suppose that the only conditional dependence among variables that does not surpass the independence test is the one from X_2 to X_1

5 Experimental set-up

In this section, we present the proposed experimentation in order to evaluate our learning approach to multi-dimensional Bayesian network classifiers. We analyze the following aspects:

- The robustness and stability of the proposed learning approach in relation to the variability

sources: How does the Pareto set change in different MOEA/D runs or with different training sets?

- We compare the proposed classification rules.
- The result of our proposal is compared with both single-objective learning approaches and single-class approaches to multi-dimensional classification, in the context of Bayesian networks classifiers.

5.1 The experimental process

In order to carry out the experimentation required to evaluate our approach, we use some artificial multi-dimensional data sets and one real world data set.

The artificial data sets are sampled from multi-dimensional feature-label probability distributions $p(\mathbf{x}, \mathbf{c})$ represented as multi-dimensional Bayesian network classifiers. These Bayesian network classifiers have been created in two steps. First, the structure of the multi-dimensional Bayesian network classifier was created with the Java Bayes software (Cozman, 2000) and then we obtained the parameters by sampling a Dirichlet distribution with all parameters equal to one. We have chosen the following structures: *MDnB*, *MDTAN*, *MDPoly* and *MDJ/K*. These structures have been created with 8 predictive variables and taking 2 and 3 class variables into account. The cardinality of the predictive variables ranges from 2 to 4 and the cardinality of the classes from 2 to 3. In the case of MD$J/K$, we use $J = 1$ and $K = 2$ (Figure 2) for 2 class variables and $J = 2$ and $K = 2$ for 3 class variables. Once the multi-dimensional Bayesian network classifiers have been created, we sample 5 data sets from each of them. Specifically, we sample 40 artificial data sets (5 for each different structure (4) and number of class variables (2)) of 200 instances each.

Typical benchmark data repositories in supervised classification do not provide data sets with multiple class variables. However, there is a data set in the UCI Machine Learning Repository (Asuncion and Newman, 2007) where several attributes can be used as class variables: Automobile data set. This data set has 205 instances and 26 predictive variables. Some variables are continuous and others discrete and there are missing values in some instances. We pre-processed this data set by discretizing all the continuous variables to 2 nominal values (Fayyad and Irani, 1993) and deleting the instances and predictive variables with missing values (we deleted 2 predictive variables and 13 instances). This data set is used considering two class variables (price and symboling) and three class variables (highway-mpg, price and symboling).

We use the MOEA/D algorithm as the learning engine. The parameters used for the experiments with this algorithm were fixed as follows:

- Number of subproblems: $S = 100*$(individual size).
- Number of neighbours for each subproblem: $T = 20$.
- Stop criterion: 100.000 evaluations.
- Number of replacements in the neighbourhood: $nr = 2$.
- Objective functions: We use the 5 repeated 5-fold cross-validation error estimation of each class variable.

In order to study the stability of the proposed learning approach with regard to different runs of the learning algorithm, we run MOEA/D 5 times for each artifical data set.

So as to make all these experiments possible, we use different open source libraries in Java. For the classification utilities we use the ICLAB library (Calvo and Flores, 2009) and the weka library (Witten and Frank, 2000) and for the multi-objective optimization utilities we use the jMetal library (Durillo et al., 2006).

5.2 Pareto evaluation

The results of the multi-objective learning are presented in a Pareto front composed of non-dominated solutions. Each point of the Pareto front represents the accuracies for each class of a specific classifier. Therefore, in order to compare different runs of the MOEA/D algorithm with the same data set or the classifiers obtained with different data sets, we need to compare several Pareto fronts. The community working in the field of multi-objective optimization has devised several measures to evaluate different Pareto fronts. Basically a good Pareto front has its points uniformly spread around the real Pareto front and covers the complete real Pareto front. In this paper, we have adopted different measures to evaluate the obtained Pareto fronts and the classifiers on it:

- Size of the dominated space $S(A)$ (Zitzler and Thiele, 1999): It measures the amount of the objective space that is covered by a given non-dominated set of classifiers A (Figure 5 for an illustration in a two class problem). A high value of $S(A)$ indicates that the classifiers in A have good accuracy values for all class variables.

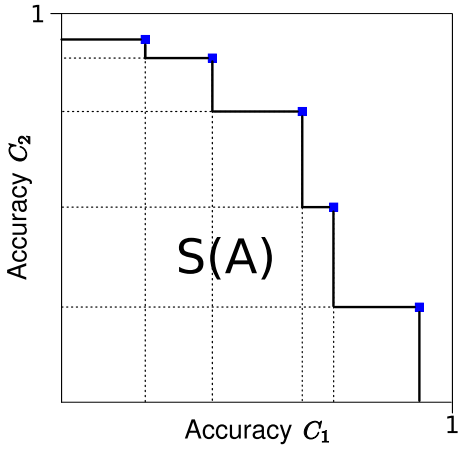

Figure 5: Calculation of $S(A)$.

- Non-uniformity of a Pareto front $D(A)$ (Lee et al., 2005): This quantity measures the non-uniformity of the distribution of a Pareto front, and it is given by the distribution of the Euclidean distance d_i between each pair of closest points along the Pareto front:

$$D(A) = \sqrt{\frac{\sum_i (d_i/dm - 1)^2}{\mid A \mid -1}}$$

  This quantity is the standard deviation of the distances normalized by the average distance dm. If $D(A) = 0$, the spacing in the Pareto front is uniform. Therefore, a lower value means a more uniform spread of the Pareto front.

In addition to the previous two measures and in order to better evaluate the obtained Pareto fronts from a classification point of view, we have included in our analysis some classifiers of the Pareto front we call *representative classifiers*. Therefore, we have selected the most extreme classifiers (the most accurate classifiers for a single class variable) and the most balanced classifier (the highest average accurate classifier) of each Pareto front (Figure 6).

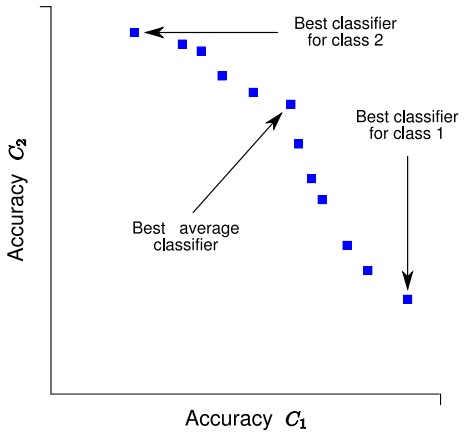

Figure 6: Representative classifiers of a Pareto front.

6 Results

The proposed approach returns a Pareto front composed of a set of non-dominated classifiers. The obtained Pareto fronts show different trade-off solutions to the multi-dimensional classification problem. Each point of the Pareto front represents the accuracies of a classifier for the different class variables.

In order to illustrate the achieved results, we have plotted some Pareto fronts produced in a single MOEA/D run. For each multi-dimensional structure (MDnB, MDTAN, MDPoly and MD1/2) with 2 class variables, we show the results for one data set sampled with that structure (figures 7, 8, 9 and 10). Each figure shows a Pareto front for each classification rule. We also show the results for the Automobile data set (Figure 11) with 2 class variables. [4]

This results section is organized at follows: First, we measure the robustness and stability of the proposed approach for different variability sources (Section 6.1). Then, the results for different classification rules are compared (Section 6.2). Finally, we compare the results of the proposed multi-objective learning approach with other Bayesian network classification approaches to multi-dimensional classification (Section 6.3).

[4]The complete results (for all the data sets and MOEA/D runs) can be consulted at *http://www.sc.ehu.es/ccwbayes/members/juandiego/MOPLearning/*

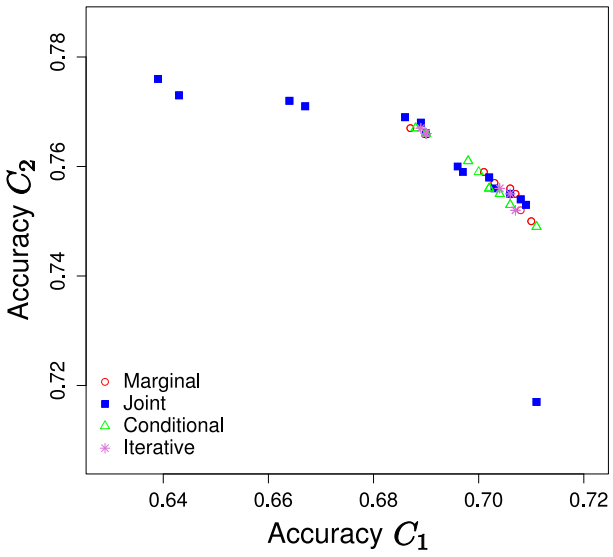

Figure 7: Accuracy values of classifiers in the Pareto fronts learnt in one data set sampled from a 2 class multi-dimensional classifier with a MDnB structure in a single MOEA/D run.

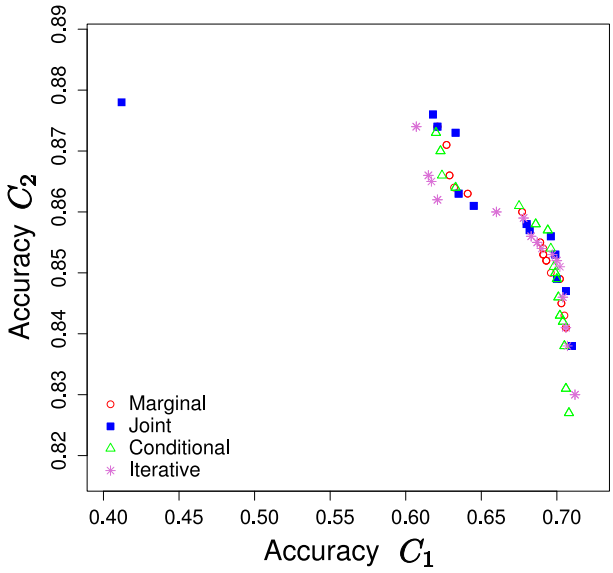

Figure 8: Accuracy values of classifiers in the Pareto fronts learnt in one data set sampled from a 2 class multi-dimensional classifier with a MDTAN structure in a single MOEA/D run.

6.1 Robustness and stability of the proposed learning approach

We start by analyzing the stability of our approach with regard to different executions of the learning algorithm. In order to do that, we measure the variance accross 5 different MOEA/D runs with the same training set. The results can be consulted in tables 1, 2, 3, 4, 5 and 6. Each value in the following tables represents the average and the standard deviation for 5 MOEA/D

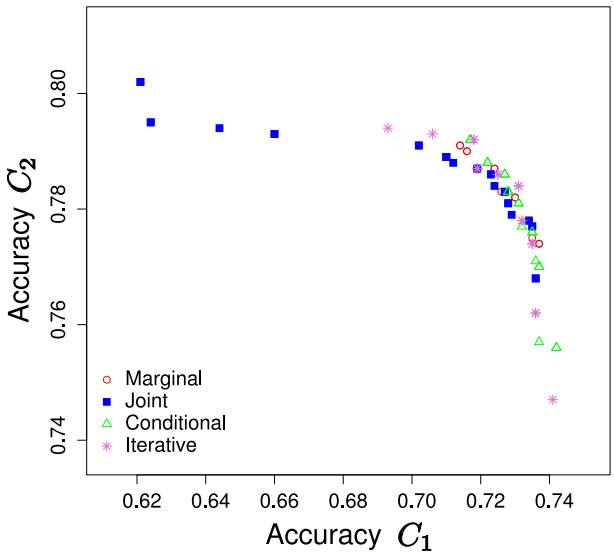

Figure 9: Accuracy values of classifiers in the Pareto fronts learnt in one data set sampled from a 2 class multi-dimensional classifier with a MDPoly structure in a single MOEA/D run.

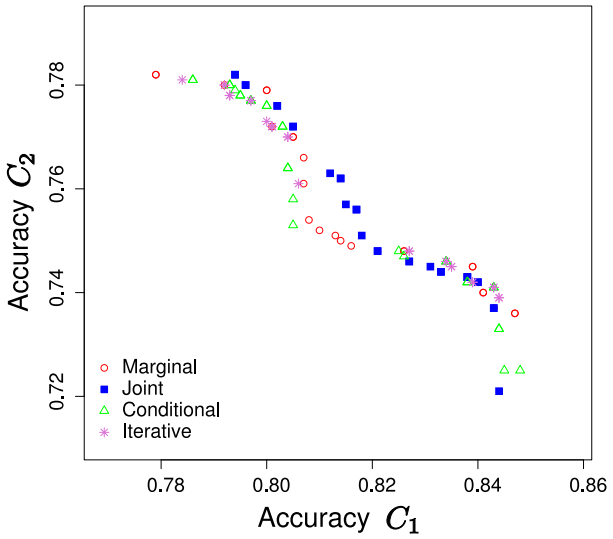

Figure 10: Accuracy values of classifiers in the Pareto fronts learnt in one data set sampled from a 2 class multi-dimensional classifier with a MD1/2 structure in a single MOEA/D run.

runs of the corresponding measure for each classification rule and each multi-dimensional structure.

Tables 1 and 2 show the mean and the standard deviation of the $S(A)$ value of the Pareto fronts obtained from the artificial data sets with 2 and 3 class variables respectively.

The size of dominated space of the obtained Pareto fronts is almost the same for different MOEA/D runs. The standard deviation of $S(A)$ is in the order of magnitude of 10^{-3} for almost all the data sets, a very low variance value because $S(A)$ is a value bound between 0 and 1.

Table 1: Mean and standard deviation for 5 MOEA/D runs of the $S(A)$ value of Pareto fronts obtained from 2 class artificial data sets.

$S(A)$		Data set 1	Data set 2	Data set 3	Data set 4	Data set 5
Joint	MDnB	0.660 ± 2.8E-3	0.585 ± 2.3E-3	0.546 ± 8.7E-3	0.577 ± 2.5E-3	0.642 ± 1.4E-3
	MDTAN	0.607 ± 5.8E-3	0.630 ± 1.1E-2	0.630 ± 4.5E-3	0.607 ± 6.1E-3	0.637 ± 6.2E-4
	MDPoly	0.615 ± 1.7E-3	0.574 ± 6.0E-3	0.649 ± 4.5E-3	0.660 ± 2.2E-3	0.591 ± 8.3E-3
	MD1/2	0.659 ± 7.6E-3	0.630 ± 3.8E-3	0.662 ± 1.7E-3	0.663 ± 7.3E-3	0.688 ± 8.8E-3
Mar.	MDnB	0.606 ± 2.7E-3	0.628 ± 8.6E-3	0.629 ± 4.8E-3	0.607 ± 2.9E-3	0.638 ± 9.5E-4
	MDTAN	0.657 ± 1.5E-3	0.583 ± 1.2E-3	0.544 ± 7.3E-3	0.576 ± 1.3E-3	0.641 ± 1.2E-3
	MDPoly	0.613 ± 1.2E-3	0.573 ± 3.6E-3	0.649 ± 1.9E-3	0.660 ± 2.6E-3	0.590 ± 4.3E-3
	MD1/2	0.658 ± 3.5E-3	0.627 ± 1.3E-3	0.660 ± 2.7E-3	0.660 ± 9.0E-3	0.687 ± 9.0E-3
Cond.	MDnB	0.658 ± 3.2E-3	0.585 ± 1.4E-3	0.542 ± 6.9E-3	0.577 ± 1.1E-3	0.641 ± 1.8E-3
	MDTAN	0.606 ± 3.0E-3	0.626 ± 4.6E-3	0.630 ± 1.7E-3	0.606 ± 2.5E-3	0.637 ± 1.2E-3
	MDPoly	0.613 ± 9.9E-4	0.572 ± 3.0E-3	0.646 ± 9.1E-4	0.659 ± 1.2E-3	0.589 ± 3.2E-3
	MD1/2	0.655 ± 1.1E-3	0.628 ± 1.9E-3	0.660 ± 1.0E-3	0.660 ± 7.2E-3	0.686 ± 6.5E-3
Iter.	MDnB	0.656 ± 4.9E-3	0.584 ± 8.9E-4	0.542 ± 6.9E-3	0.577 ± 1.3E-3	0.640 ± 1.1E-3
	MDTAN	0.605 ± 3.3E-3	0.626 ± 3.9E-3	0.629 ± 5.7E-4	0.606 ± 2.2E-3	0.637 ± 5.5E-4
	MDPoly	0.613 ± 2.1E-3	0.571 ± 1.3E-3	0.647 ± 1.8E-3	0.659 ± 1.8E-3	0.587 ± 5.6E-3
	MD1/2	0.655 ± 1.3E-3	0.627 ± 1.2E-3	0.660 ± 1.0E-3	0.659 ± 7.3E-3	0.684 ± 6.1E-3

Table 2: Mean and standard deviation for 5 MOEA/D runs of the $S(A)$ value of Pareto fronts obtained from 3 class artificial data sets.

$S(A)$		Data set 1	Data set 2	Data set 3	Data set 4	Data set 5
Joint	MDnB	0.705 ± 5.6E-3	0.735 ± 5.3E-3	0.739 ± 5.4E-3	0.771 ± 3.3E-3	0.858 ± 2.5E-3
	MDTAN	0.663 ± 1.1E-2	0.695 ± 8.9E-3	0.671 ± 3.9E-3	0.646 ± 6.1E-3	0.677 ± 6.0E-3
	MDPoly	0.581 ± 6.4E-3	0.589 ± 1.0E-2	0.525 ± 6.4E-3	0.482 ± 1.5E-2	0.591 ± 9.3E-3
	MD2/2	0.676 ± 8.5E-3	0.727 ± 1.8E-2	0.715 ± 4.1E-2	0.616 ± 9.1E-3	0.615 ± 3.8E-3
Mar.	MDnB	0.707 ± 5.4E-3	0.726 ± 1.1E-2	0.741 ± 4.2E-3	0.772 ± 4.4E-3	0.859 ± 2.7E-3
	MDTAN	0.663 ± 2.6E-3	0.695 ± 1.3E-3	0.673 ± 1.9E-3	0.638 ± 1.5E-2	0.679 ± 1.6E-2
	MDPoly	0.572 ± 1.0E-2	0.594 ± 5.3E-3	0.520 ± 4.4E-3	0.488 ± 1.3E-2	0.576 ± 1.4E-2
	MD2/2	0.678 ± 1.2E-2	0.725 ± 1.8E-2	0.706 ± 4.7E-2	0.620 ± 9.3E-3	0.612 ± 6.1E-3
Cond.	MDnB	0.701 ± 8.3E-3	0.728 ± 1.1E-2	0.736 ± 5.5E-3	0.772 ± 4.3E-3	0.860 ± 2.6E-3
	MDTAN	0.666 ± 3.2E-3	0.697 ± 4.5E-3	0.673 ± 2.8E-3	0.641 ± 1.4E-2	0.688 ± 1.4E-2
	MDPoly	0.576 ± 7.7E-3	0.596 ± 7.6E-3	0.520 ± 3.3E-3	0.482 ± 1.7E-2	0.586 ± 7.5E-3
	MD2/2	0.665 ± 3.0E-2	0.731 ± 1.2E-2	0.731 ± 4.0E-3	0.621 ± 7.0E-3	0.612 ± 4.9E-3
Iter.	MDnB	0.704 ± 5.9E-3	0.735 ± 1.0E-2	0.741 ± 5.7E-3	0.772 ± 1.8E-3	0.856 ± 3.2E-3
	MDTAN	0.664 ± 3.6E-3	0.689 ± 5.1E-3	0.672 ± 2.4E-3	0.632 ± 1.3E-2	0.687 ± 1.6E-2
	MDPoly	0.574 ± 6.7E-3	0.593 ± 1.0E-2	0.521 ± 5.8E-3	0.484 ± 9.9E-3	0.587 ± 5.0E-3
	MD2/2	0.681 ± 3.6E-3	0.725 ± 1.8E-2	0.731 ± 3.9E-3	0.612 ± 8.0E-3	0.612 ± 8.1E-3

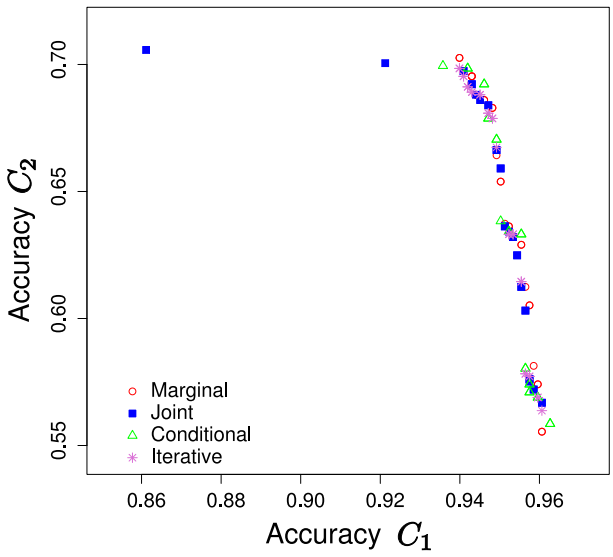

Figure 11: Accuracy values of the Pareto fronts learnt in the Automobile data set with 2 class variables in a single MOEA/D run.

Moreover, all the measured values are between 0.48 and 0.86. The variance of the size of the dominated space seems to be slightly lower in 2 class variable data sets than in 3 class variable data sets. However, the size of the dominated space of the proposed approach is very stable for different learning algorithm runs.

Tables 3 and 4 show the mean and the standard deviation of the $D(A)$ value of the Pareto fronts obtained from artificial data sets with 2 and 3 class variables respectively.

Contrary to the case of $S(A)$, we observe significative changes in the non-uniformity of the obtained Pareto fronts for different MOEA/D runs. The standard deviation of $D(A)$ is in the order of magnitude of 10^{-1} for almost all the data sets. Another interesting aspect is that the variance of the non-uniformity of the obtained Pareto fronts is very similar for different number of class variables. So, in this case it seems that the number of class variables does not influence in the uniformity of the Pareto front.

Finally, we analyze the representative classifiers of the obtained Pareto fronts. Tables 5 and 6 show the accuracy and its standard deviation of the representative classifiers of each Pareto front obtained from artificial data sets with 2 and 3 class variables respectively.

The accuracy of the representative classifiers of the obtained Pareto fronts is very similar for different MOEA/D runs. The standar deviation is in the order of magnitude of 10^{-3} of the accuracy for almost all the data sets, a very low variance value because all the measured values are between 0.56 and 0.95. The variance seems to be slightly lower in 2 class variable data sets than in 3 class variable data sets, but in the same order of magnitude.

If we focus on the most extreme classifiers (the most accurate for a particular class variable), we show that the variance of the acccuracy for that particular class variable is lower than the variance of the accuracy of the rest class variables, a good result if our objective is that class variable. Finally, the variance of the accuracy of the most balanced classifiers (the highest average

Table 3: Mean and standard deviation for 5 MOEA/D runs of the $D(A)$ value of Pareto fronts obtained from 2 class artificial data sets.

$D(A)$		Data set 1	Data set 2	Data set 3	Data set 4	Data set 5
Joint	MDnB	0.718±7.4E-2	1.022±6.3E-1	0.662±1.6E-1	0.643±8.8E-2	0.513±2.3E-1
	MDTAN	1.081±1.5E-1	1.357±2.5E-1	0.859±2.1E-1	2.050±4.7E-1	1.467±3.5E-1
	MDPoly	0.929±6.1E-1	0.483±1.7E-1	0.613±3.1E-1	0.657±1.9E-1	0.711±1.0E-1
	MD1/2	0.918±7.2E-1	0.678±1.4E-1	0.572±2.1E-1	1.117±5.0E-1	1.005±4.6E-1
Mar.	MDnB	0.628±8.3E-2	0.581±1.5E-1	0.665±1.5E-1	0.668±1.5E-1	0.591±1.7E-1
	MDTAN	1.158±1.7E-1	1.491±1.7E-1	0.627±2.4E-1	1.975±2.2E-1	1.378±2.5E-1
	MDPoly	0.566±1.4E-1	0.592±2.6E-1	0.496±1.1E-1	0.516±9.0E-2	0.655±1.6E-1
	MD1/2	0.480±1.3E-1	0.604±2.1E-1	0.539±1.5E-1	0.948±6.1E-1	0.603±2.5E-1
Cond.	MDnB	0.662±9.2E-2	0.575±1.5E-1	0.573±1.0E-1	0.458±1.3E-1	0.431±1.3E-1
	MDTAN	1.366±2.5E-1	1.478±2.6E-1	0.718±2.7E-1	1.947±2.7E-1	1.363±1.3E-1
	MDPoly	0.495±2.1E-1	0.493±1.9E-1	0.553±1.1E-1	0.369±9.9E-2	0.665±2.4E-1
	MD1/2	0.641±1.4E-1	0.532±1.4E-1	0.453±4.6E-2	0.765±3.3E-1	0.960±3.7E-1
Iter.	MDnB	0.697±1.2E-1	0.601±1.7E-1	0.561±1.7E-1	0.406±1.9E-1	0.531±2.0E-1
	MDTAN	1.194±1.7E-1	1.459±1.3E-1	0.759±6.3E-1	2.125±2.7E-1	1.425±2.6E-1
	MDPoly	0.582±1.9E-1	0.666±2.3E-1	0.479±1.2E-1	0.601±9.4E-2	0.620±1.6E-1
	MD1/2	0.640±1.3E-1	0.694±2.1E-1	0.670±7.3E-2	0.605±2.4E-1	0.921±4.3E-1

Table 4: Mean and standard deviation for 5 MOEA/D runs of the $D(A)$ value of Pareto fronts obtained from 3 class artificial data sets.

$D(A)$		Data set 1	Data set 2	Data set 3	Data set 4	Data set 5
Joint	MDnB	0.852±7.7E-2	0.667±7.9E-2	0.725±9.5E-2	0.845±1.2E-1	1.448±2.6E-1
	MDTAN	1.245±4.0E-1	0.970±3.4E-1	1.570±3.4E-1	0.851±1.9E-1	0.880±5.0E-2
	MDPoly	0.849±7.8E-2	0.795±6.2E-2	0.902±1.4E-1	0.777±1.0E-1	0.975±3.5E-2
	MD2/2	0.926±7.2E-2	0.892±1.0E-1	0.887±1.1E-1	0.698±3.7E-2	0.855±1.1E-1
Mar.	MDnB	0.961±1.4E-1	0.670±2.6E-2	0.655±5.6E-2	0.730±7.9E-2	1.595±3.7E-1
	MDTAN	0.971±4.4E-1	0.943±3.3E-1	1.552±4.5E-1	0.869±1.8E-1	0.947±6.1E-2
	MDPoly	0.914±5.6E-2	0.742±9.1E-2	0.959±8.9E-2	0.723±4.0E-2	1.019±7.2E-2
	MD2/2	0.922±7.5E-2	0.991±9.4E-2	0.854±2.0E-1	0.774±6.0E-2	0.894±1.2E-1
Cond.	MDnB	0.883±4.7E-2	0.635±3.9E-2	0.793±1.1E-1	0.705±4.8E-2	1.509±2.8E-1
	MDTAN	1.217±3.9E-1	0.959±3.9E-1	1.842±5.1E-1	0.875±3.2E-1	0.927±1.0E-1
	MDPoly	0.900±9.4E-2	0.736±8.2E-2	0.918±7.2E-2	0.864±1.2E-1	1.050±5.5E-2
	MD2/2	0.907±3.5E-2	1.042±1.4E-1	0.915±7.9E-2	0.729±1.2E-1	0.838±1.0E-1
Iter.	MDnB	0.945±1.6E-1	0.656±3.8E-2	0.635±4.6E-2	0.787±1.3E-1	1.561±1.7E-1
	MDTAN	1.018±3.9E-1	1.065±3.0E-1	1.896±4.5E-1	0.794±1.1E-1	0.915±5.0E-2
	MDPoly	0.938±5.1E-2	0.725±6.9E-2	0.911±1.0E-1	0.697±5.9E-2	1.087±7.1E-2
	MD2/2	0.912±4.5E-2	0.778±1.0E-1	0.836±1.2E-1	0.691±8.2E-2	0.802±3.1E-2

Table 5: Mean and standard deviation for 5 learning algorithm runs of the accuracy of the representative classifiers of Pareto fronts obtained from 2 class artificial data sets.

		Best Classifier for C_1		Best Classifier for C_2		Best average Classifier	
		C_1	C_2	C_1	C_2	C_1	C_2
Joint	MDnB	0.729 ± 3.3E-3	0.794 ± 1.7E-2	0.690 ± 1.1E-2	0.827 ± 2.6E-3	0.711 ± 9.2E-3	0.818 ± 6.2E-3
	MDTAN	0.716 ± 4.0E-3	0.820 ± 8.0E-3	0.605 ± 1.4E-2	0.872 ± 5.1E-3	0.678 ± 3.7E-2	0.846 ± 1.5E-2
	MDPoly	0.752 ± 2.5E-3	0.793 ± 8.0E-3	0.709 ± 2.3E-2	0.822 ± 4.6E-3	0.740 ± 6.9E-3	0.809 ± 7.4E-3
	MD1/2	0.836 ± 3.7E-3	0.745 ± 2.2E-2	0.792 ± 2.2E-2	0.791 ± 4.7E-3	0.824 ± 7.2E-3	0.772 ± 1.1E-2
Mar.	MDnB	0.727 ± 3.0E-3	0.802 ± 5.4E-3	0.695 ± 6.2E-3	0.825 ± 1.2E-3	0.708 ± 7.5E-3	0.820 ± 5.0E-3
	MDTAN	0.716 ± 3.1E-3	0.820 ± 8.0E-3	0.605 ± 1.5E-2	0.871 ± 3.0E-3	0.665 ± 3.7E-2	0.853 ± 1.4E-2
	MDPoly	0.751 ± 1.5E-3	0.794 ± 6.4E-3	0.718 ± 8.5E-3	0.822 ± 3.2E-3	0.736 ± 9.9E-3	0.811 ± 6.6E-3
	MD1/2	0.835 ± 3.4E-3	0.747 ± 1.9E-2	0.804 ± 8.6E-3	0.789 ± 4.3E-3	0.820 ± 1.2E-2	0.776 ± 1.2E-2
Cond.	MDnB	0.727 ± 3.1E-3	0.801 ± 4.1E-3	0.697 ± 3.8E-3	0.826 ± 1.3E-3	0.714 ± 7.5E-3	0.816 ± 6.7E-3
	MDTAN	0.716 ± 2.3E-3	0.820 ± 9.6E-3	0.609 ± 1.1E-2	0.870 ± 2.7E-3	0.697 ± 1.6E-2	0.844 ± 1.5E-2
	MDPoly	0.751 ± 1.3E-3	0.796 ± 5.6E-3	0.718 ± 9.9E-3	0.821 ± 3.0E-3	0.740 ± 9.5E-3	0.809 ± 7.9E-3
	MD1/2	0.834 ± 3.0E-3	0.751 ± 1.2E-2	0.803 ± 8.4E-3	0.789 ± 3.0E-3	0.819 ± 1.1E-2	0.778 ± 9.6E-3
Iter.	MDnB	0.727 ± 2.8E-3	0.803 ± 4.2E-3	0.698 ± 8.1E-3	0.825 ± 2.0E-3	0.711 ± 1.1E-2	0.816 ± 5.4E-3
	MDTAN	0.716 ± 1.9E-3	0.822 ± 6.7E-3	0.608 ± 8.8E-3	0.870 ± 2.2E-3	0.671 ± 4.0E-2	0.848 ± 1.4E-2
	MDPoly	0.751 ± 1.6E-3	0.795 ± 6.1E-3	0.720 ± 6.7E-3	0.820 ± 2.1E-3	0.737 ± 8.4E-3	0.811 ± 5.9E-3
	MD1/2	0.834 ± 2.6E-3	0.755 ± 1.1E-2	0.801 ± 8.6E-3	0.788 ± 2.3E-3	0.820 ± 9.6E-3	0.777 ± 9.1E-3

accurate classifiers) is similar for all class variables.

We also study the stability of the proposed approach for different training sets. To that end, we show in tables 7 and 8 the standar deviation for different training sets of the $S(A)$ and $D(A)$ values, and the accuracy of the representative classifiers of the obtained Pareto fronts with artificial data sets with 2 and 3 class variables respectively. Each value for each classification rule and each multi-dimensional structure, represents the average of the standard deviation for 5 different data sets and 5 MOEA/D runs for each data set.

We notice that the variance over different data sets is higher than the variance over different learning algorithm runs. That is because the variance for different training sets includes the variability for different learning algorithm runs. The size of the dominated space of the obtained Pareto fronts is almost the same for different training sets. The standard deviation of the $S(A)$ value is in the order of magnitude of 10^{-2} for almost all the data sets.

The non-uniformity of the Pareto fronts varies for different training sets. The standard deviation of the $D(A)$ value is in the order of magnitude of 10^{-1} for almost all the data sets and furthermore, it is very similar for different number of class variables. So, it seems that the number of class variables does not influence the uniformity of the Pareto front.

Finally, we focus on the variance of the accuracy of the representative classifiers of the obtained Pareto fronts. The standard deviation is in the order of magnitude of 10^{-2} for almost all the data sets, a low variance value, and it seems to be slightly lower in 2 class variable data sets than in 3 class variable data sets (but in the same order of magnitude).

We conclude the study of the robustness of the proposed approach with the following conclusion: Although the proposed approach seems to return different classifiers for different training sets and MOEA/D runs, they cover a similar part of the accuracy space and maintain similar accuracy values for the reference classifiers. So, our approach seems to be robust for both variability

Table 6: Mean and standard deviation for 5 learning algorithm runs of the accuracy of the representative classifiers of Pareto fronts obtained from 3 class artificial data sets.

		Best Classifier for C_1			Best Classifier for C_2		
		C_1	C_2	C_3	C_1	C_2	C_3
Joint	MDnB	0.807 ± 2.8E-3	0.924 ± 2.5E-3	0.857 ± 7.4E-3	0.733 ± 2.2E-2	0.950 ± 7.8E-4	0.895 ± 9.6E-3
	MDTAN	0.814 ± 3.6E-3	0.775 ± 1.8E-2	0.929 ± 6.2E-3	0.791 ± 3.8E-3	0.846 ± 2.7E-3	0.926 ± 3.5E-3
	MDPoly	0.694 ± 7.5E-3	0.721 ± 1.7E-2	0.751 ± 3.8E-2	0.564 ± 1.9E-2	0.825 ± 5.1E-3	0.760 ± 1.5E-2
	MD2/2	0.830 ± 3.9E-3	0.764 ± 1.1E-2	0.724 ± 3.0E-2	0.720 ± 3.1E-2	0.830 ± 4.7E-3	0.740 ± 5.4E-2
Mar.	MDnB	0.805 ± 5.3E-3	0.924 ± 3.0E-3	0.860 ± 6.4E-3	0.756 ± 2.1E-2	0.950 ± 3.6E-4	0.889 ± 9.2E-3
	MDTAN	0.814 ± 3.8E-3	0.772 ± 2.0E-2	0.931 ± 5.8E-3	0.794 ± 4.5E-3	0.846 ± 1.8E-3	0.925 ± 4.0E-3
	MDPoly	0.692 ± 7.0E-3	0.715 ± 2.0E-2	0.751 ± 3.1E-2	0.560 ± 1.9E-2	0.823 ± 4.7E-3	0.765 ± 1.5E-2
	MD2/2	0.830 ± 3.8E-3	0.768 ± 1.2E-2	0.727 ± 2.9E-2	0.724 ± 3.1E-2	0.830 ± 6.1E-3	0.741 ± 5.7E-2
Cond.	MDnB	0.806 ± 4.0E-3	0.925 ± 2.4E-3	0.858 ± 4.8E-3	0.733 ± 2.1E-2	0.950 ± 5.0E-4	0.894 ± 8.7E-3
	MDTAN	0.813 ± 2.9E-3	0.778 ± 1.6E-2	0.929 ± 6.6E-3	0.792 ± 2.7E-3	0.846 ± 1.5E-3	0.925 ± 3.0E-3
	MDPoly	0.690 ± 6.0E-3	0.721 ± 1.7E-2	0.757 ± 1.5E-2	0.563 ± 2.1E-2	0.824 ± 4.8E-3	0.749 ± 3.0E-2
	MD2/2	0.831 ± 3.4E-3	0.768 ± 8.9E-3	0.737 ± 1.5E-2	0.731 ± 3.4E-2	0.828 ± 3.4E-3	0.732 ± 5.4E-2
Iter.	MDnB	0.807 ± 3.6E-3	0.925 ± 2.0E-3	0.858 ± 6.4E-3	0.742 ± 2.6E-2	0.950 ± 1.1E-3	0.891 ± 1.3E-2
	MDTAN	0.813 ± 3.0E-3	0.781 ± 1.6E-2	0.932 ± 6.7E-3	0.792 ± 5.0E-3	0.847 ± 2.0E-3	0.925 ± 2.8E-3
	MDPoly	0.690 ± 4.1E-3	0.727 ± 9.5E-3	0.763 ± 1.0E-2	0.567 ± 2.6E-2	0.822 ± 4.4E-3	0.761 ± 2.2E-2
	MD2/2	0.829 ± 2.5E-3	0.769 ± 1.2E-2	0.739 ± 1.4E-2	0.730 ± 3.0E-2	0.827 ± 3.3E-3	0.761 ± 3.6E-2
		Best Classifier for C_3			Best average Classifier		
		C_1	C_2	C_3	C_1	C_2	C_3
Joint	MDnB	0.738 ± 1.1E-2	0.933 ± 4.8E-3	0.924 ± 4.0E-3	0.767 ± 2.8E-2	0.935 ± 1.0E-2	0.893 ± 1.4E-2
	MDTAN	0.724 ± 3.2E-2	0.806 ± 7.8E-3	0.958 ± 1.1E-3	0.800 ± 3.3E-3	0.814 ± 1.5E-2	0.940 ± 6.2E-3
	MDPoly	0.609 ± 2.3E-2	0.725 ± 1.3E-2	0.856 ± 3.1E-3	0.623 ± 3.2E-2	0.765 ± 2.9E-2	0.812 ± 3.8E-2
	MD2/2	0.736 ± 3.1E-2	0.763 ± 1.7E-2	0.839 ± 4.8E-3	0.790 ± 1.8E-2	0.787 ± 1.9E-2	0.786 ± 3.5E-2
Mar.	MDnB	0.739 ± 6.7E-3	0.932 ± 5.4E-3	0.925 ± 2.9E-3	0.777 ± 1.4E-2	0.935 ± 7.7E-3	0.895 ± 1.8E-2
	MDTAN	0.758 ± 3.1E-2	0.804 ± 1.0E-2	0.958 ± 1.2E-3	0.800 ± 6.1E-3	0.816 ± 1.5E-2	0.938 ± 9.5E-3
	MDPoly	0.610 ± 2.5E-2	0.731 ± 2.2E-2	0.856 ± 3.9E-3	0.670 ± 4.1E-2	0.785 ± 2.8E-2	0.805 ± 3.7E-2
	MD2/2	0.736 ± 3.1E-2	0.764 ± 2.3E-2	0.838 ± 6.2E-3	0.780 ± 2.6E-2	0.797 ± 2.0E-2	0.794 ± 1.9E-2
Cond.	MDnB	0.736 ± 7.0E-3	0.933 ± 4.5E-3	0.924 ± 3.0E-3	0.774 ± 1.9E-2	0.934 ± 9.9E-3	0.890 ± 1.9E-2
	MDTAN	0.747 ± 2.7E-2	0.805 ± 8.9E-3	0.958 ± 1.3E-3	0.797 ± 7.2E-3	0.814 ± 2.0E-2	0.944 ± 7.8E-3
	MDPoly	0.603 ± 2.9E-2	0.738 ± 2.4E-2	0.856 ± 2.6E-3	0.627 ± 2.9E-2	0.781 ± 2.5E-2	0.801 ± 2.8E-2
	MD2/2	0.741 ± 2.6E-2	0.765 ± 1.9E-2	0.839 ± 4.7E-3	0.791 ± 2.2E-2	0.789 ± 1.5E-2	0.795 ± 2.4E-2
Iter.	MDnB	0.737 ± 4.0E-3	0.935 ± 2.5E-3	0.924 ± 2.6E-3	0.764 ± 2.8E-2	0.940 ± 6.7E-3	0.900 ± 1.3E-2
	MDTAN	0.739 ± 5.6E-3	0.810 ± 4.4E-3	0.958 ± 6.5E-4	0.798 ± 4.2E-3	0.810 ± 2.4E-2	0.941 ± 7.6E-3
	MDPoly	0.614 ± 1.7E-2	0.734 ± 2.2E-2	0.855 ± 3.0E-3	0.625 ± 3.1E-2	0.775 ± 2.6E-2	0.800 ± 4.7E-2
	MD2/2	0.746 ± 2.1E-2	0.769 ± 1.1E-2	0.837 ± 5.3E-3	0.781 ± 2.3E-2	0.793 ± 1.4E-2	0.796 ± 2.5E-2

Table 7: Standard deviation for different training sets of the $S(A)$ and the $D(A)$ values, and the accuracy of the representative classifiers of Pareto fronts obtained from 2 class artificial data sets.

		$S(A)$	$D(A)$	Accuracy					
				Best C_1 Class.		Best C_2 Class.		Best Av. Class.	
				C_1	C_2	C_1	C_2	C_1	C_2
Joint	MDnB	3.9E-2	1.27E-01	3.0E-2	2.1E-2	2.2E-2	2.6E-2	2.3E-2	2.6E-2
	MDTAN	1.2E-2	3.17E-01	9.0E-3	1.9E-2	4.0E-2	5.3E-3	2.2E-2	1.1E-2
	MDPoly	2.9E-2	1.13E-01	2.6E-2	2.6E-2	2.4E-2	2.7E-2	2.6E-2	2.8E-2
	MD1/2	1.3E-2	1.86E-01	7.8E-3	2.2E-2	2.1E-2	2.0E-2	9.5E-3	1.5E-2
Mar.	MDnB	3.9E-2	3.27E-02	3.0E-2	2.3E-2	2.1E-2	2.6E-2	2.3E-2	2.7E-2
	MDTAN	1.2E-2	3.47E-01	8.7E-3	1.8E-2	4.2E-2	5.6E-3	3.2E-2	9.1E-3
	MDPoly	3.0E-2	4.72E-02	2.6E-2	3.0E-2	2.8E-2	2.8E-2	2.6E-2	2.9E-2
	MD1/2	1.3E-2	1.25E-01	7.3E-3	2.0E-2	1.5E-2	2.0E-2	9.0E-3	1.7E-2
Cond.	MDnB	3.9E-2	7.62E-02	3.0E-2	2.3E-2	2.1E-2	2.7E-2	2.5E-2	2.7E-2
	MDTAN	1.2E-2	2.71E-01	8.6E-3	1.8E-2	4.5E-2	6.0E-3	1.1E-2	9.0E-3
	MDPoly	2.9E-2	7.54E-02	2.6E-2	3.0E-2	3.2E-2	2.8E-2	2.7E-2	2.8E-2
	MD1/2	1.3E-2	1.54E-01	7.7E-3	1.5E-2	1.6E-2	2.0E-2	1.1E-2	2.2E-2
Iter.	MDnB	3.8E-2	7.28E-02	3.0E-2	2.3E-2	2.2E-2	2.6E-2	2.3E-2	2.6E-2
	MDTAN	1.2E-2	3.33E-01	9.1E-3	1.7E-2	4.7E-2	5.0E-3	2.2E-2	1.0E-2
	MDPoly	3.0E-2	4.73E-02	2.6E-2	2.9E-2	2.6E-2	2.8E-2	2.7E-2	2.7E-2
	MD1/2	1.3E-2	8.60E-02	8.4E-3	2.2E-2	1.3E-2	2.0E-2	9.6E-3	2.0E-2

sources, MOEA/D runs and training sets, in terms of the accuracy of classifiers returned.

6.2 Comparison of the proposed classification rules

One way to compare the different classification rules is by visually inspecting the obtained Pareto fronts. For each classification rule we plotted the Pareto fronts produced in a single MOEA/D run (figures 7, 8, 9 and 10). However, these Pareto fronts are obtained for a single data set and we want to compare the classification rules for all the data sets. We use statistical tests in order to know when there are statistical differences between the classification rules in relation with the $S(A)$ value, the $D(A)$ value and the accuracy of the representative classifiers. Specifically, we have used a Friedmand test (Demsar, 2006) with a Shaffer's static post-hoc test with $\alpha = 0.1$ (García and Herrera, 2008). Following the suggestion of García and Herrera (2008), we use the Shaffer's static instead of Nemenji post-hoc test because it is more powerful. The test results can be represented by means of critical difference diagrams (Demsar, 2006), which show the mean ranks of each classification rule across all the domains in a numbered line. If there is no statistically significant difference between two methods, they are connected in the diagram by a straight line.

The results of the experimentation can be consulted in tables 9, 10, 11 and 12. Each value represents the average and the standard deviation of the $S(A)$ and the $D(A)$ values for 5 different training sets and 5 MOEA/D runs for each classification rule and each multi-dimensional structure. The best values for each multi-dimensional structure are in bold.

For almost all the domains in the experimentation, multi-dimensional classifiers that use joint

Table 8: Standard deviation for different training sets of the $S(A)$ and the $D(A)$ values, and the accuracy of the representative classifiers of Pareto fronts obtained from 3 class artificial data sets.

		$S(A)$	$D(A)$	Accuracy											
				Best C_1 Class.			Best C_2 Class.			Best C_3 Class.			Best Av. Class.		
				C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3
Joint	MDnB	4.2E-2	2.16E-01	3.8E-2	1.6E-2	2.4E-2	4.6E-2	1.6E-2	1.7E-2	3.8E-2	1.6E-2	2.4E-2	4.3E-2	1.6E-2	1.9E-2
	MDTAN	1.3E-2	2.43E-01	1.9E-2	4.4E-2	1.6E-2	1.5E-2	2.8E-2	1.6E-2	1.9E-2	4.4E-2	1.6E-2	1.5E-2	3.1E-2	1.9E-2
	MDPoly	4.0E-2	6.31E-02	2.8E-2	5.3E-2	7.3E-3	5.0E-2	2.7E-2	3.7E-2	2.8E-2	5.3E-2	7.3E-3	3.0E-2	2.6E-2	2.4E-2
	MD2/2	4.4E-2	6.15E-02	5.0E-2	1.7E-2	3.7E-2	4.2E-2	1.5E-2	3.1E-2	5.0E-2	1.7E-2	3.7E-2	5.3E-2	1.8E-2	1.9E-2
Mar.	MDnB	4.3E-2	2.85E-01	3.6E-2	1.5E-2	2.6E-2	3.8E-2	1.5E-2	1.7E-2	3.6E-2	1.5E-2	2.6E-2	4.2E-2	1.9E-2	2.4E-2
	MDTAN	1.5E-2	1.98E-01	1.8E-2	4.4E-2	1.6E-2	1.7E-2	2.8E-2	1.6E-2	1.8E-2	4.4E-2	1.6E-2	1.9E-2	3.1E-2	2.2E-2
	MDPoly	3.7E-2	1.11E-01	2.9E-2	4.8E-2	1.4E-2	5.0E-2	2.5E-2	3.0E-2	2.9E-2	4.8E-2	1.4E-2	4.0E-2	2.0E-2	1.6E-2
	MD2/2	4.2E-2	5.87E-02	5.1E-2	1.6E-2	3.7E-2	3.9E-2	1.5E-2	2.8E-2	5.1E-2	1.6E-2	3.7E-2	5.1E-2	1.6E-2	2.6E-2
Cond.	MDnB	4.5E-2	2.42E-01	3.9E-2	1.5E-2	2.6E-2	4.0E-2	1.5E-2	1.6E-2	3.9E-2	1.5E-2	2.6E-2	4.3E-2	1.4E-2	2.4E-2
	MDTAN	1.6E-2	2.93E-01	1.8E-2	4.1E-2	1.7E-2	1.7E-2	2.8E-2	1.8E-2	1.8E-2	4.1E-2	1.7E-2	1.9E-2	2.9E-2	1.3E-2
	MDPoly	4.1E-2	7.50E-02	3.0E-2	5.1E-2	2.3E-2	4.7E-2	2.7E-2	2.3E-2	3.0E-2	5.1E-2	2.3E-2	1.7E-2	2.7E-2	1.0E-2
	MD2/2	4.7E-2	8.22E-02	5.0E-2	1.5E-2	4.0E-2	4.1E-2	1.4E-2	3.6E-2	5.0E-2	1.5E-2	4.0E-2	5.2E-2	1.8E-2	2.0E-2
Iter.	MDnB	4.2E-2	2.69E-01	3.7E-2	1.6E-2	2.3E-2	4.0E-2	1.5E-2	1.6E-2	3.7E-2	1.6E-2	2.3E-2	4.0E-2	1.3E-2	1.3E-2
	MDTAN	1.6E-2	3.04E-01	1.8E-2	4.4E-2	1.7E-2	1.5E-2	2.8E-2	1.7E-2	1.8E-2	4.4E-2	1.7E-2	1.5E-2	2.4E-2	1.7E-2
	MDPoly	3.9E-2	1.29E-01	3.0E-2	4.9E-2	1.9E-2	4.2E-2	2.7E-2	2.9E-2	3.0E-2	4.9E-2	1.9E-2	3.4E-2	3.5E-2	2.0E-2
	MD2/2	4.8E-2	5.62E-02	5.0E-2	1.7E-2	4.1E-2	4.4E-2	1.3E-2	2.1E-2	5.0E-2	1.7E-2	4.1E-2	5.4E-2	1.7E-2	1.9E-2

Table 9: Mean and standard deviation for different training sets and MOEA/D runs of the $S(A)$ and the $D(A)$ values of Pareto fronts obtained from 2 class artificial data sets.

		$S(A)$	$D(A)$
MDnB	Joint	**0.60173**±3.89E-2	0.71153±1.27E-1
	Mar.	0.60010±3.91E-2	0.62645±3.27E-2
	Cond.	0.60069±3.93E-2	**0.54014**±7.62E-2
	Iter.	0.60012±3.85E-2	0.55923±7.28E-2
MDTAN	Joint	**0.62221**±1.21E-2	1.36296±3.17E-1
	Mar.	0.62162±1.21E-2	**1.32568**±3.47E-1
	Cond.	0.62097±1.21E-2	1.37436±2.71E-1
	Iter.	0.62057±1.20E-2	1.39239±3.33E-1
MDPoly	Joint	**0.61786**±2.94E-2	0.67856±1.13E-1
	Mar.	0.61680±2.97E-2	0.56500±4.72E-2
	Cond.	0.61587±2.93E-2	**0.51505**±7.54E-2
	Iter.	0.61539±3.00E-2	0.58965±4.73E-2
MD1/2	Joint	**0.66037**±1.28E-2	0.85792±1.86E-1
	Mar.	0.65548±1.28E-2	**0.62295**±1.25E-1
	Cond.	0.65796±1.31E-2	0.67006±1.54E-1
	Iter.	0.65720±1.29E-2	0.70611±8.60E-2

Table 10: Mean and standard deviation for different training sets and MOEA/D runs of the $S(A)$ and the $D(A)$ values of Pareto fronts obtained from 3 class artificial data sets.

		$S(A)$	$D(A)$
MDnB	Joint	**0.76190**±4.22E-2	0.90754±2.16E-01
	Mar.	0.76090±4.35E-2	0.92197±2.85E-01
	Cond.	0.75945±4.52E-2	**0.90517**±2.42E-01
	Iter.	0.76153±4.22E-2	0.91664±2.69E-01
MDTAN	Joint	0.67019±1.27E-2	1.10314±2.43E-01
	Mar.	0.66977±1.52E-2	**1.05668**±1.98E-01
	Cond.	**0.67297**±1.57E-2	1.16390±2.93E-01
	Iter.	0.66883±1.65E-2	1.13755±3.04E-01
MDPoly	Joint	**0.55373**±4.00E-2	**0.85971**±6.31E-02
	Mar.	0.55011±3.69E-2	0.87143±1.11E-01
	Cond.	0.55232±4.07E-2	0.89339±7.50E-02
	Iter.	0.55201±3.95E-2	0.87141±1.29E-01
MD2/2	Joint	0.66975±4.35E-2	0.85140±6.15E-02
	Mar.	0.63151±4.18E-2	0.89847±5.87E-02
	Cond.	**0.67213**±4.73E-2	0.88617±8.22E-02
	Iter.	0.67204±4.82E-2	**0.80380**±5.62E-02

classification rule lead to higher $S(A)$ values (tables 9 and 10), indicating that better solutions (higher accuracies) are obtained using this classification rule. This result is corroborated with the statistical tests (figures 12 and 13).

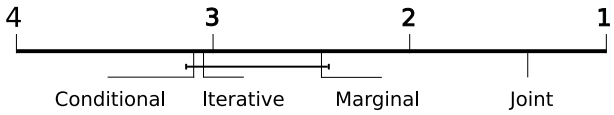

Figure 12: $S(A)$ ranking for the different classification rules on the 2 class artificial domains.

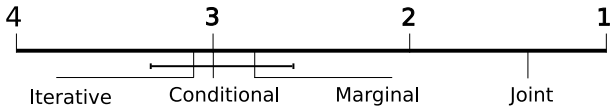

Figure 13: $S(A)$ ranking for the different classification rules on the 3 class artificial domains.

If we analyze the non-uniformity of the Pareto fronts produced with different classification rules (tables 9 and 10), we observe differences between artificial data sets with 2 and 3 class variables. In the 2 class variable case, the Pareto fronts produced with the joint classification rule have the worst $D(A)$ values in almost all cases, and the values for the rest of the classification rules are very similar between them. We have applied the previous described statistical test to the obtained results (Figure 14), and we show that there are significative differences between classifiers that use joint classification rule and classifiers that use conditional or iterative classification rules.

On the other hand, the $D(A)$ values in the 3 class variables case are very similar, and we do not find statistical differences in the use of any classification rule (Figure 15).

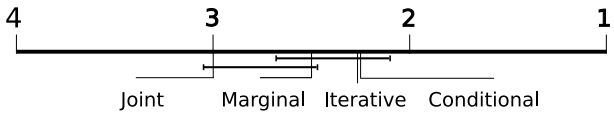

Figure 14: $D(A)$ ranking for the different classification rules on artificial domains with 2 class variables.

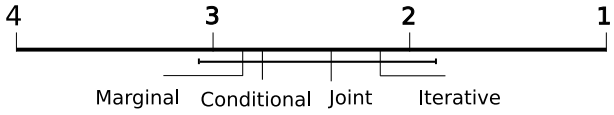

Figure 15: $D(A)$ ranking for the different classification rules on artificial domains with 3 class variables.

As in the previous section, we analyze the mean and the standard deviation of the accuracies of the representative classifiers produced with the different classification rules (tables 11 and 12). The tables show the results for the artificial data sets with 2 and 3 class variables respectively. In

all cases, the Pareto fronts of classifiers that use joint classification rule contain the most extreme solutions (the most accurate for a particular class variable) for the given data sets. But it seems that they do not find the most balanced solutions (the highest average accurate classifiers).

If we apply the previously described statistical test (figures 16 and 17) we show that the Pareto fronts with classifiers that use joint clasification rule contain more extreme solutions than Pareto fronts with classifiers that use other classification rules with statistical differences. However, there are no statistical differences to state which classification rule leads to the most balanced solutions.

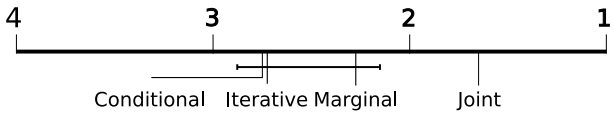

Figure 16: Accuracies ranking of the best extreme solutions for the different classification rules on artificial domains.

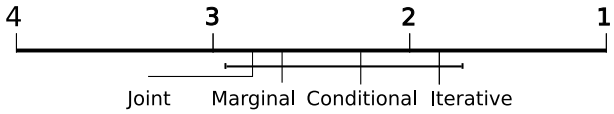

Figure 17: Accuracies ranking of the best balanced solutions for the different classification rules on artificial domains.

We finish the study of the comparison of the different classification rules with the following conclusion: We found no differences in the $D(A)$ value for different classification rules in the 3 classes data sets, but in the 2 classes data sets the Pareto fronts produced with a joint classification rule are less uniformly distibuted than classifiers produced with conditional or iterative classification rules. However, it seems that the Pareto fronts produced with a joint classification rule reach better $S(A)$ values. Moreover, we observe that classifiers produced with a joint classification rule are more extreme classifiers (the most accurate for each class variable) than classifiers produced with other classification rules. On the other hand, we can not state which classification rule produces more balanced classifiers (the highest average accurate classifiers).

6.3 Multi-objective learning approach versus other approaches to multi-dimensional classification

We compare the multi-objective learning approach proposed in this paper with the single-objective learning approaches to multi-dimensional classification proposed by de Waal and van der Gaag (2007) and van der Gaag and de Waal (2006) (vG-MDnB and dW-MDTAN, see section 3.2) [5]. We also compare our approach with single-class oriented Bayesian classifiers. For that purpose, we have chosen a naive Bayes classifier (nB) (Langley et al., 1992; Minsky, 1961) and a tree-augmented Bayesian classifier (TAN) (Friedman et al., 1997). For each data set, we train m single-class classifiers (m nB and m TAN), one for each class variable. Note that each of the points in the following figures that represents single-class classifiers represent, in fact, m classifiers: one for each class variable. Figure 18 summarizes the experimental process we propose to compare our approach versus other approaches to multi-dimensional classification.

Table 11: Mean and standard deviation of the accuracy for different training sets and MOEA/D runs of the representative classifiers of Pareto fronts obtained from 2 class artificial data sets.

		Best Classifier for C_1		Best Classifier for C_2		Best average Classifier	
		C_1	C_2	C_1	C_2	C_1	C_2
MDnB	Joint	**0.7279**±2.97E-2	0.7944±2.09E-2	0.6898±2.24E-2	**0.8266**±2.62E-2	0.7107±2.34E-2	0.8184±2.62E-2
	Mar.	0.7268±2.99E-2	0.8024±2.30E-2	0.6948±2.06E-2	0.8254±2.64E-2	0.7084±2.26E-2	**0.8200**±2.71E-2
	Cond.	0.7274±3.02E-2	0.8009±2.26E-2	0.6975±2.12E-2	0.8256±2.67E-2	**0.7141**±2.51E-2	0.8156±2.66E-2
	Iter.	0.7273±3.02E-2	**0.8030**±2.28E-2	**0.6983**±2.24E-2	0.8250±2.57E-2	0.7112±2.29E-2	0.8165±2.56E-2
MDTAN	Joint	0.7162±9.01E-3	0.8198±1.94E-2	0.6055±4.02E-2	**0.8720**±5.28E-3	0.6779±2.16E-2	0.8457±1.10E-2
	Mar.	**0.7163**±8.74E-3	0.8200±1.80E-2	0.6046±4.21E-2	0.8708±5.55E-3	0.6650±3.22E-2	**0.8526**±9.07E-3
	Cond.	0.7158±8.61E-3	0.8200±1.77E-2	**0.6089**±4.51E-2	0.8704±5.95E-3	**0.6972**±1.05E-2	0.8436±8.99E-3
	Iter.	0.7158±9.12E-3	**0.8223**±1.71E-2	0.6077±4.67E-2	0.8698±4.96E-3	0.6708±2.19E-2	0.8484±1.02E-2
MDPoly	Joint	**0.7519**±2.61E-2	0.7932±2.59E-2	0.7090±2.39E-2	**0.8224**±2.72E-2	**0.7405**±2.60E-2	0.8093±2.84E-2
	Mar.	0.7511±2.59E-2	0.7940±2.99E-2	**0.7181**±2.81E-2	0.8217±2.76E-2	0.7364±2.56E-2	**0.8113**±2.90E-2
	Cond.	0.7509±2.59E-2	**0.7961**±3.05E-2	0.7176±3.17E-2	0.8206±2.80E-2	0.7402±2.74E-2	0.8093±2.76E-2
	Iter.	0.7507±2.65E-2	0.7946±2.90E-2	0.7204±2.60E-2	0.8201±2.83E-2	0.7373±2.68E-2	0.8110±2.75E-2
MD1/2	Joint	**0.8356**±7.84E-3	0.7455±2.25E-2	0.7920±2.07E-2	**0.7913**±2.01E-2	**0.8236**±9.49E-3	0.7720±1.53E-2
	Mar.	0.8347±7.34E-3	0.7470±1.98E-2	**0.8043**±1.49E-2	0.7893±2.05E-2	0.8196±9.01E-3	0.7762±1.68E-2
	Cond.	0.8343±7.74E-3	0.7510±1.54E-2	0.8030±1.61E-2	0.7894±1.98E-2	0.8194±1.07E-2	**0.7778**±2.19E-2
	Iter.	0.8342±8.43E-3	**0.7548**±2.16E-2	0.8015±1.27E-2	0.7885±2.01E-2	0.8204±9.60E-3	0.7773±1.98E-2

Table 12: Mean and standard deviation of the accuracy for different training sets and MOEA/D runs of the representative classifiers of Pareto fronts obtained from 3 class artificial data sets.

		Best Classifier for C_1			Best Classifier for C_2		
		C_1	C_2	C_3	C_1	C_2	C_3
MDnB	Joint	**0.80736**±3.8E-2	0.92416±1.6E-2	0.85736±2.4E-2	0.73328±4.6E-2	**0.95032**±1.6E-2	**0.89480**±1.7E-2
	Mar.	0.80536±2.4E-3	0.92416±1.5E-2	**0.85968**±2.6E-2	**0.75600**±3.8E-2	0.94984±1.5E-2	0.88904±1.7E-2
	Cond.	0.80632±3.9E-2	0.92480±1.5E-2	0.85824±2.6E-2	0.73296±4.0E-2	0.95008±1.5E-2	0.89448±1.6E-2
	Iter.	0.80688±3.7E-2	**0.92488**±1.6E-2	0.85792±2.3E-2	0.74224±4.0E-2	0.94960±1.5E-2	0.89064±1.6E-2
MDTAN	Joint	**0.81426**±1.9E-2	0.77489±4.4E-2	0.92863±1.6E-2	0.79136±1.5E-2	0.84570±2.8E-2	**0.92560**±1.6E-2
	Mar.	0.81410±1.8E-2	0.77193±4.4E-2	0.93063±1.6E-2	**0.79392**±1.7E-2	0.84635±2.8E-2	0.92536±1.6E-2
	Cond.	0.81354±1.3E-3	0.77818±4.1E-2	0.92935±1.7E-2	0.79223±1.7E-2	0.84571±2.8E-2	0.92544±1.8E-2
	Iter.	0.81266±1.8E-2	**0.78082**±4.4E-2	**0.93231**±1.7E-2	0.79176±1.5E-2	**0.84658**±2.8E-2	0.92529±1.7E-2
MDPoly	Joint	**0.69384**±2.8E-2	**0.72096**±5.3E-2	0.75112±7.3E-3	0.56400±5.0E-2	**0.82488**±2.7E-2	0.75976±3.7E-2
	Mar.	0.69160±2.9E-2	0.71552±4.8E-2	0.75056±1.4E-2	0.55976±5.0E-2	0.82288±2.5E-2	**0.76480**±3.0E-2
	Cond.	0.69032±3.0E-2	0.72152±5.1E-2	**0.75744**±2.3E-2	0.56320±4.7E-2	0.82384±2.7E-2	0.74928±2.3E-2
	Iter.	0.69016±3.0E-2	0.72728±4.9E-2	0.76328±1.9E-2	**0.56736**±4.2E-2	0.82184±2.7E-2	0.76136±2.9E-2
MD2/2	Joint	0.83024±5.0E-2	0.76424±1.7E-2	0.72432±3.7E-2	0.72032±4.2E-2	0.83016±1.5E-2	0.74048±3.1E-2
	Mar.	0.83000±5.1E-2	0.76832±1.6E-2	0.72672±3.7E-2	0.72360±3.9E-2	**0.83024**±1.5E-2	0.74072±2.8E-2
	Cond.	**0.83080**±5.0E-2	0.76768±1.5E-2	0.73736±4.0E-2	**0.73104**±4.1E-2	0.82832±1.4E-2	0.73232±3.6E-2
	Iter.	0.82904±5.0E-2	**0.76888**±1.7E-2	**0.73912**±4.1E-2	0.73040±4.4E-2	0.82712±1.3E-2	**0.76144**±2.1E-2
		Best Classifier for C_3			Best Average Classifier		
		C_1	C_2	C_3	C_1	C_2	C_3
MDnB	Joint	0.73792±5.7E-2	0.93280±1.7E-2	0.92424±1.5E-2	0.76696±4.3E-2	0.93504±1.6E-2	0.89304±1.9E-2
	Mar.	**0.73880**±5.4E-2	0.93192±1.7E-2	**0.92496**±1.4E-2	**0.77720**±4.2E-2	0.93544±1.9E-2	0.89512±2.4E-2
	Cond.	0.73600±5.8E-2	0.93288±1.7E-2	0.92416±1.5E-2	0.77424±4.3E-2	0.93416±1.4E-2	0.89040±2.4E-2
	Iter.	0.73712±5.6E-2	**0.93488**±1.6E-2	0.92424±1.5E-2	0.76400±4.0E-2	**0.93976**±1.3E-2	**0.90032**±1.3E-2
MDTAN	Joint	0.72422±5.1E-2	0.80558±3.4E-2	0.95787±1.3E-2	0.80022±1.5E-2	0.81430±3.1E-2	0.94047±1.9E-2
	Mar.	**0.75818**±7.8E-2	0.80380±2.8E-2	0.95779±1.2E-2	**0.80031**±1.9E-2	**0.81565**±3.1E-2	0.93775±2.2E-2
	Cond.	0.74685±6.8E-2	0.80521±3.0E-2	**0.95819**±1.3E-2	0.79685±1.9E-2	0.81426±2.9E-2	**0.94367**±1.3E-2
	Iter.	0.73949±8.2E-2	**0.80975**±2.8E-2	0.95763±1.3E-2	0.79847±1.5E-2	0.80989±2.4E-2	0.94063±1.7E-2
MDPoly	Joint	0.60864±3.5E-2	0.72528±2.6E-2	0.85552±2.2E-2	0.62264±3.0E-2	0.76552±2.6E-2	**0.81208**±2.4E-2
	Mar.	0.60952±4.3E-2	0.73064±2.9E-2	**0.85616**±2.2E-2	0.60960±4.0E-2	**0.78528**±2.0E-2	0.80544±1.6E-2
	Cond.	0.60344±3.7E-2	**0.73792**±2.3E-2	0.85560±2.2E-2	**0.62680**±1.7E-2	0.78056±2.7E-2	0.80088±1.0E-2
	Iter.	**0.61360**±3.6E-2	0.73416±2.5E-2	0.85512±2.3E-2	0.62464±3.4E-2	0.77552±3.5E-2	0.80040±2.0E-2
MD2/2	Joint	0.73648±4.3E-2	0.76256±1.3E-2	0.83864±1.5E-2	0.79032±5.3E-2	0.78712±1.8E-2	0.78584±1.9E-2
	Mar.	0.73608±4.1E-2	0.76416±1.6E-2	0.83784±1.4E-2	0.77952±5.1E-2	**0.79664**±1.6E-2	0.79400±2.6E-2
	Cond.	0.74128±4.8E-2	0.76464±1.9E-2	**0.83896**±1.5E-2	**0.79152**±5.2E-2	0.78936±1.8E-2	0.79472±2.0E-2
	Iter.	**0.74560**±4.7E-2	**0.76944**±2.1E-2	0.83728±1.5E-2	0.78152±5.4E-2	0.79312±1.7E-2	**0.79648**±1.9E-2

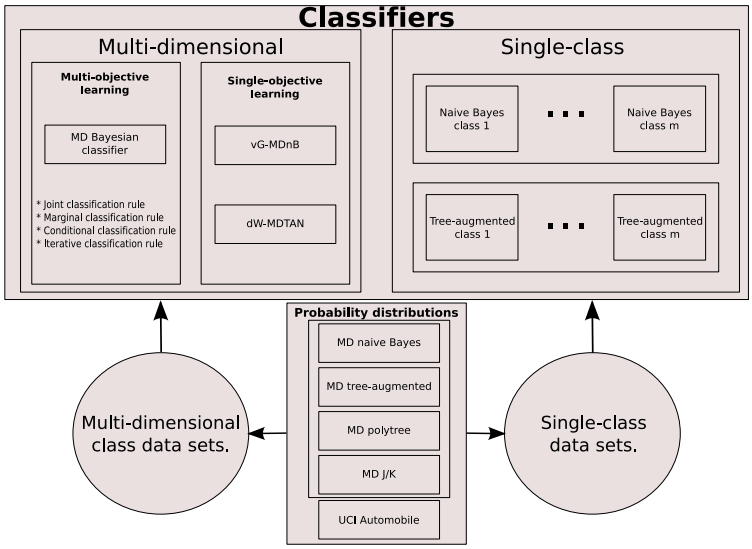

Figure 18: The proposed experimentation to compare the proposed learning approach versus other approaches to multi-dimensional classification

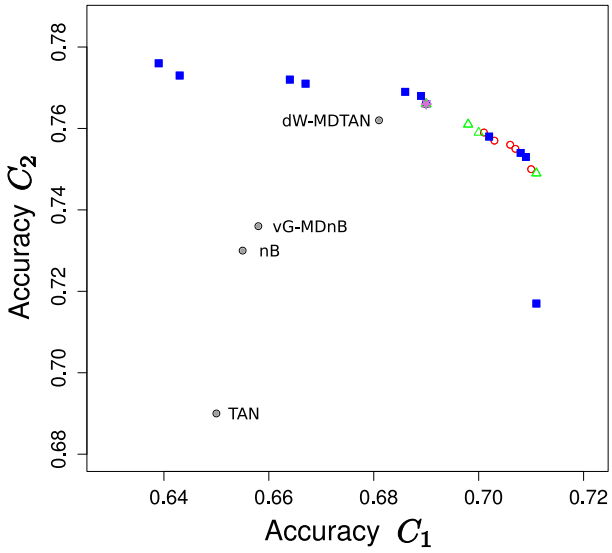

Figure 19: Accuracy values of the Pareto fronts learnt with a multi-objective learning approach vs. other Bayesian approaches in a MDnB structure data set.

In order to illustrate the obtained results, we have plotted the obtained Pareto front of the proposed approach versus vG-MDnB, dW-MDTAN and m nB and m TAN classifiers in one artificial data set from each of the proposed structures (figures 19, 20, 21 and 22) and in the Automobile data set (Figure 23). The Pareto fronts are created departing from the ones shown in figures 7, 8, 9, 10 and 11 with 2 class variables (we maintain the labels of the different classification

[5]Remember that vG-MDnB and dW-MDTAN obtain only one classifier and not a Pareto set

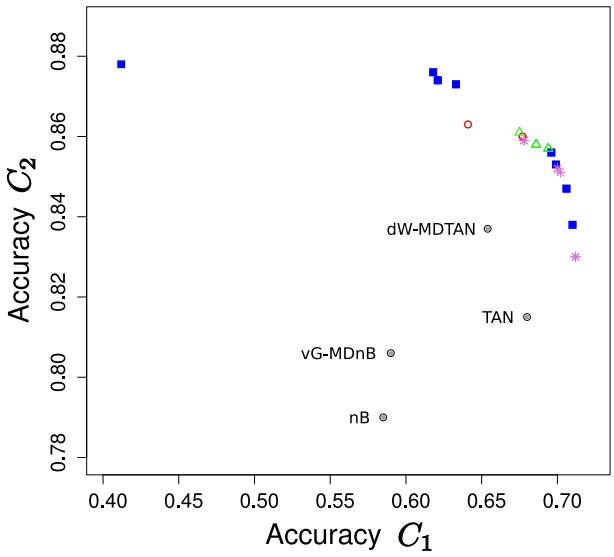

Figure 20: Accuracy values of the Pareto fronts learnt with a multi-objective learning approach vs. other Bayesian approaches in a MDTAN structure data set.

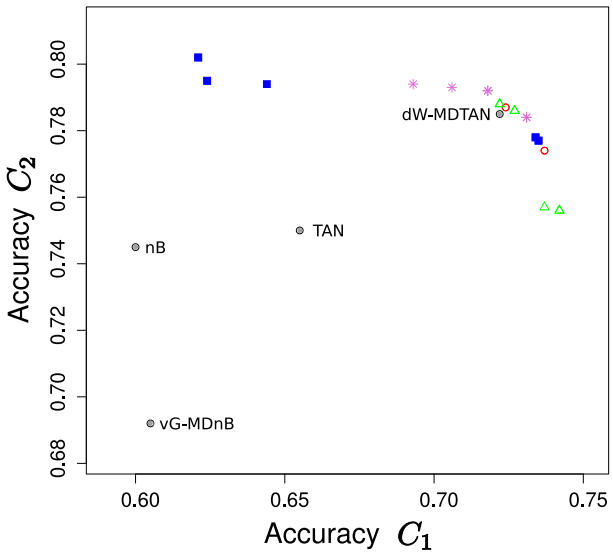

Figure 21: Accuracy values of the Pareto fronts learnt with a multi-objective learning approach vs. other Bayesian approaches in a MDPoly structure data set.

rules). Note that the Pareto fronts are composed mixing the classifiers obtained with different classification rules and removing the dominated classifiers.

The figures clearly show that the Pareto fronts obtained with the multi-dimensional learning approach dominates the solutions obtained with vG-MDnB, dW-MDTAN, m nB and m TAN for all the data sets. The proposed approach offers more accurate classifiers for all class variables than vG-MDnB, dW-MDTAN, m nB and m TAN classifiers. The dW-MDTAN classifier is the closest to the obtained Pareto fronts. For example in the Pareto front plotted in Figure 21, the solution

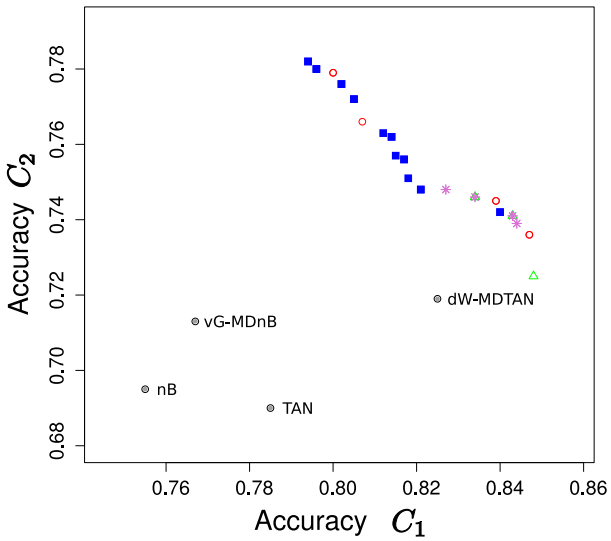

Figure 22: Accuracy values of the Pareto fronts learnt with a multi-objective learning approach vs. other Bayesian approaches in a MD1/2 structure data set.

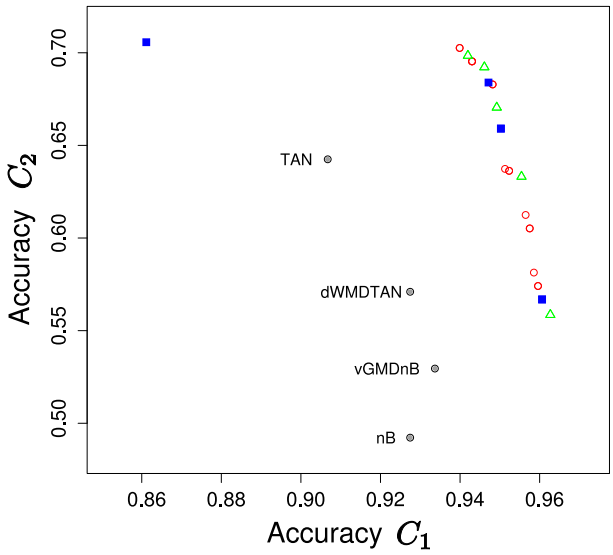

Figure 23: Multi-objective approach vs. other Bayesian approaches in the Automobile data set

of a dW-MDTAN is very close to the Pareto front, but it is still dominated by the classifiers in it.

Tables 13 and 14 show the accuracies of the representative classifiers in the obtained Pareto fronts versus the accuracies of vG-MDnB, dW-MDTAN, m nB and m TAN classifiers in the 2 class artificial data sets. The classifiers are evaluated using a 5 repeated 5-cv error estimator. Each value represents the average of the accuracy of the classifiers obtained for each data set structure and the standard deviation for the different traininig sets and learning algorithm runs for the multi-objective learnt classifiers and the standard deviation for the different training sets and $k$cv runs. This value is the best for the different classification rule. The best values for each

Table 13: Mean and standard deviation of the accuracy for different training sets and MOEA/D runs of the most accurate classifiers of the proposed approach versus of vG-MDnB, dW-MDTAN, m nB and m TAN classifiers in 2 class artificial data sets.

	MDnB		MDTAN	
	C_1	C_2	C_1	C_2
Best C_1 class.	**0.727**±3.0E-2	0.800±2.2E-2	**0.716**±8.9E-3	0.821±1.8E-2
Best C_2 class.	0.695±2.2E-2	**0.826**±2.6E-2	0.607±4.4E-2	**0.871**±5.4E-3
Best Av. class.	0.711±2.3E-2	0.818±2.6E-2	0.678±2.2E-2	0.848±9.8E-3
vG-MDnB	0.660±7.0E-3	0.772±4.3E-2	0.548±2.1E-2	0.796±1.5E-2
dW-MDTAN	0.675±7.7E-3	0.792±3.6E-2	0.699±2.9E-2	0.844±1.3E-2
m nB	0.667±1.9E-2	0.757±3.2E-2	0.539±3.9E-2	0.804±1.7E-2
m TAN	0.657±9.2E-3	0.740±6.0E-2	0.689±1.3E-2	0.857±2.2E-2
	MDPoly		MD1/2	
	C_1	C_2	C_1	C_2
Best C_1 class.	**0.751**±2.6E-2	0.794±2.9E-2	**0.835**±7.8E-3	0.750±2.0E-2
Best C_2 class.	0.716±2.7E-2	**0.821**±2.8E-2	0.800±1.6E-2	**0.790**±2.0E-2
Best Av. class.	0.739±2.6E-2	0.810±2.8E-2	0.821±9.7E-3	0.776±1.8E-2
vG-MDnB	0.643±2.6E-2	0.741±3.8E-2	0.770±9.3E-3	0.711±1.3E-2
dW-MDTAN	0.716±2.3E-2	0.762±4.9E-2	0.819±7.4E-3	0.720±8.0E-3
m nB	0.662±3.6E-2	0.778±2.0E-2	0.760±8.0E-3	0.707±2.1E-2
m TAN	0.683±1.4E-2	0.779±2.9E-2	0.792±8.4E-3	0.705±1.8E-2

multi-dimensional structure are in bold.

The proposed approach leads to most accurate classifiers in almost all cases. The extreme classifiers of the obtained Pareto fronts have the best accuracy values for all the class variables in all the data sets. The obtained most balanced classifiers dominate the classifiers learnt with other learning approaches (they have better accuracy for all class variables) in all cases except in the 2 classes MDTAN structure data sets. In those data sets, 2 TAN classifiers seem to improve the performance of the multi-objective learnt classifiers.

Finally, we test the proposed approach in the Automobile data set to check if it improves the performance of other approaches to multi-dimensional classification in real world domains. Tables 15 and 16 show the accuracies of the representative classifiers in the obtained Pareto fronts versus the accuracies of vG-MDnB, dW-MDTAN, m nB and m TAN classifiers in the Automobile data set. Each value represents the average of the accuracy of the obtained classifiers and the standard deviation for the different learning algorithm runs. The best values for each multi-dimensional structure are in bold.

In the tested real world data set, the classifiers learnt with the proposed approach improve the accuracy performance of vG-MDnB, dW-MDTAN, m nB and m TAN for all classes. The extreme classifiers of the obtained Pareto fronts have the best accuracy values for each of the class variables, and the most balanced classifiers dominate the classifiers learnt with other learning approaches (they have better accuracy for all class variables). The proposed approach substantially improves the performance of the other approaches to multi-dimensional classification in real world domains.

Table 14: Mean and standard deviation of the accuracy for different training sets and MOEA/D runs of the most accurate classifiers of the proposed approach versus of vG-MDnB, dW-MDTAN, m nB and m TAN classifiers in 3 class artificial data sets.

	MDnB			MDTAN		
	C_1	C_2	C_3	C_1	C_2	C_3
Best C_1 class.	**0.806**±2.9E-2	0.925±1.5E-2	0.858±2.5E-2	**0.814**±1.4E-2	0.776±4.3E-2	0.930±1.7E-2
Best C_2 class.	0.741±4.1E-2	**0.950**±1.5E-2	0.892±1.6E-2	0.792±1.6E-2	**0.846**±2.8E-2	0.925±1.7E-2
Best C_3 class.	0.737±5.6E-2	0.933±1.7E-2	**0.924**±1.4E-2	0.742±7.0E-2	0.806±3.0E-2	**0.958**±1.2E-2
Best Av. class.	0.771±4.2E-2	0.936±1.5E-2	0.895±2.0E-2	0.799±1.7E-2	0.814±2.9E-2	0.941±1.8E-2
vG-MDnB	0.732±6.2E-2	0.925±1.2E-2	0.858±1.4E-2	0.614±2.5E-1	0.607±2.4E-1	0.728±2.9E-1
dW-MDTAN	0.761±3.9E-2	0.922±1.3E-2	0.854±1.9E-2	0.646±2.6E-1	0.622±2.5E-1	0.736±2.9E-1
m **nB**	0.738±7.0E-2	0.920±1.2E-2	0.874±2.5E-2	0.632±2.5E-1	0.625±2.5E-1	0.734±2.9E-1
m **TAN**	0.740±8.8E-2	0.908±2.2E-2	0.880±2.4E-2	0.575±2.3E-1	0.606±2.4E-1	0.724±2.9E-1
	MDPoly			MD2/2		
	C_1	C_2	C_3	C_1	C_2	C_3
Best C_1 class.	**0.691**±2.9E-2	0.721±5.0E-2	0.756±1.6E-2	**0.830**±5.1E-2	0.767±1.6E-2	0.732±3.9E-2
Best C_2 class.	0.564±4.7E-2	**0.823**±2.7E-2	0.759±3.0E-2	0.726±4.2E-2	**0.829**±1.4E-2	0.744±2.9E-2
Best C_3 class.	0.609±3.8E-2	0.732±2.6E-2	**0.856**±2.2E-2	0.740±4.5E-2	0.765±1.8E-2	**0.838**±1.5E-2
Best Av. class.	0.621±3.0E-2	0.777±2.7E-2	0.805±1.7E-2	0.786±5.3E-2	0.792±1.7E-2	0.793±2.1E-2
vG-MDnB	0.560±3.7E-2	0.733±3.9E-2	0.753±2.2E-2	0.679±2.1E-2	0.739±2.6E-2	0.711±3.4E-2
dW-MDTAN	0.604±4.2E-2	0.760±3.0E-2	0.793±2.2E-2	0.736±3.4E-2	0.758±1.4E-2	0.740±2.7E-2
nB	0.556±2.6E-2	0.752±3.4E-2	0.758±3.8E-2	0.702±3.4E-2	0.756±2.9E-2	0.754±2.1E-2
TAN	0.608±7.0E-2	0.724±3.3E-2	0.778±3.8E-2	0.758±6.2E-2	0.742±5.8E-2	0.744±2.3E-2

Table 15: Mean and standard deviation of the accuracy for different learning algorithm runs of the most accurate classifiers of the proposed approach versus the mean and standard deviation of the accuracy for different k-cv evaluations of vG-MDnB, dW-MDTAN, m nB and m TAN classifiers in Automobile data set with 2 class variables.

	C_1	C_2
Best C_1 Class.	**0.959**±1.41E-03	0.616±2.50E-02
Best C_2 Class.	0.908±1.87E-02	**0.701**±5.64E-03
Best Av Class.	0.928±4.89E-03	0.699±6.05E-03
vG-MDnB	0.934 ±8.91E-02	0.529±7.18E-02
dW-MDTAN	0.927±7.99E-02	0.571±6.65E-02
m **nB**	0.927±8.72E-02	0.492±8.13E-02
m **TAN**	0.907±8.18E-02	0.642±3.74E-02

Table 16: Mean and standard deviation of the accuracy for different training sets and learning algorithm runs of the most accurate classifiers of the proposed approach versus the mean and standard deviation of the accuracy for different k-cv evaluations of vG-MDnB, dW-MDTAN, m nB and m TAN classifiers in Automobile data set with 3 class variables.

	C_1	C_2	C_3
Best C_1 Class.	**0.990**±3.32E-4	0.927±2.74E-3	0.595±3.73E-3
Best C_2 Class.	0.962±8.79E-3	**0.960**±6.63E-4	0.574±9.12E-3
Best C_3 Class.	0.950±3.02E-2	0.925±7.30E-3	**0.683**±6.47E-3
Best Av Class.	0.979±3.32E-4	0.940±1.66E-3	0.674±2.16E-3
vG-MDnB	0.962±3.48E-3	0.910±1.93E-2	0.591±4.89E-3
dW-MDTAN	0.889±4.39E-2	0.910±7.69E-3	0.655±6.32E-2
m **nB**	0.943±2.14E-3	0.930±1.23E-2	0.640±2.60E-3
m **TAN**	0.964±3.10E-3	0.900±8.40E-3	0.610±1.21E-3

7 Conclusions and future work

We call multi-dimensional supervised classification to the generalization of the single-class supervised classification problem to the simultaneous prediction of a set of class variables. In this paper, we face the problem of learning Bayesian network classifiers for multi-dimensional supervised classification problems. To that end, we present a multi-objective learning approach to multi-dimensional Bayesian classifiers. This approach returns a Pareto set of non-dominated multi-dimensional Bayesian classifiers learnt from the same data set. In addition, we have defined new classification rules for probabilistic classifiers in multi-dimensional class problems.

Our approach considers the learning of multi-dimensional Bayesian classifiers whose objective is to maximize the accuracy of each class variable, and it uses a multi-objective optimization algorithm (MOEA/D) to learn a set of non-dominated classifiers. We have used the 5 repeated 5-fold cross-validation error estimation (5cv) of each class variable as objective functions for the multi-objective optimization problem.

The robustness of the proposed learning approach is measured for different variability sources: The variance with regard to different learning algorithm runs and the variance with regard to changes in the training set. The proposed learning approach seems to be very robust for both variability sources. The classifiers in the Pareto fronts obtained with our approach are not evenly distributed for different training sets and MOEA/D runs, however they cover a similar part of the accuracy space and maintain very similar accuracy values.

We note that the multi-dimensional characteristic of the problem allows us to develop different classification rules for multi-dimensional classifiers that would make no sense in single-class classification because they take into account multiple class variables. We have presented four different classification rules: The joint classification rule returns the most probable combination of class variables given the features. By constrast, the marginal classification rule marginalizes each class variable for the rest of the class variables given the features, and returns the most probable value. The conditional classification rule begins using the marginal clasification rule and then uses the estimated class values as evidence in order to estimate again the class variables values. Finally, the iterative classification rule continues estimating the class variable values in different steps since the estimated class variables values do not change in two consecutive steps.

We show that the classifiers produced with the joint classification rule are more accurate for each class variable, but there are no differences in the average accuracy for all class variables simultaneously. The Pareto fronts produced with a joint classification dominate a higher part of the accuracy space. Their classifiers are similarly distributed across the Pareto front to the rest of the classification rules in the 3 classes data sets, but in the 2 classes data sets the Pareto fronts produced with a joint classification rule are less uniformly distibuted than classifiers produced with conditional or iterative classification rules.

Finally, we compare the proposed approach with other Bayesian classification approaches to multi-dimensional classification (vG-MDnB, dW-MDTAN, m nB and m TAN classifiers). The results show clearly that the classifiers obtained with the proposed approach improve the accuracy of the compared classifiers for each class variable and for all class variables simultaneously. Furthermore, it offers a very interesting graphic representation of the behaviour of several different multi-dimensional class Bayesian classifiers learnt from the same data set. So, a decision maker can easily choose the appropriate one from the Pareto front.

There are some paths where it is worth extending the learning approach presented in this paper. Firstly, we would like to extend the multi-objective learning to unrestricted Bayesian networks following the work done in Acid et al. (2005). Finally, it would be interesting to develop techniques to help a decision maker in the choosing of an appropriate classifier from the Pareto front.

8 Acknowledgements

This work has been partially supported by the Saiotek and Research Groups 2007-2012 (IT-242-07) programs (Basque Government), TIN2008-06815-C02-01 and Consolider Ingenio 2010 - CSD2007-00018 projects (Spanish Ministry of Science and Innovation) and COMBIOMED network in computational biomedicine (Carlos III Health Institute). We also would like to thank Iñaki Inza for his support.

References

Acid, S., de Campos, L. M., Castellano, J. G., 2005. Learning bayesian network classifiers: Searching in a space of partially directed acyclic graphs. Machine Learning 59 (3), 213–235.

Asuncion, A., Newman, D. J., 2007. UCI machine learning repository.
URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`

Bakir, G. H., Hofmann, T., Schölkopf, B., Smola, A. J., Taskar, B., Vishwanathan, S. V. N., 2007. Predicting Structured Data (Neural Information Processing). The MIT Press.

Bishop, C. M., 2006. Pattern Recognition and Machine Learning. Springer.

Blanco, R., 2005. Learning Bayesian network from data with factorisation and classification purposes. Applications in biomedicine. Ph.D. thesis, University of the Basque Country.

Braga-Neto, U., Hashimoto, R., Dougherty, E., Nguyen, D., Carroll, R., Jan 2004. Is cross-validation better than resubstitution for ranking genes? Bioinformatics 20 (2), 253–258.

Calvo, B., Flores, J. L., 2009. ICLAB intelligent computing laboratory. a library of data mining and optimization algorithms.
URL `http://iclab.sourceforge.net/`

Caruana, R., 1997. Multi-task learning. Machine Learning 28, 41–75.

Cesa-Bianchi, N., Gentile, C., Zaniboni, L., 2006. Incremental algorithms for hierarchical classification. Journal of Machine Learning Research 7, 31–54.

Coello, C., Lamont, G., Van Veldhuizen, D., 2006. Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation). Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Cozman, F., 2000. Java bayes: Bayesian networks in java.
URL `http://www.cs.cmu.edu/ javabayes/Home/index.html`

Daumé, H., Marcu, D., 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In: International Conference in Machine Learning (ICML). pp. 169–176.

de Waal, P., van der Gaag, L., 2007. Inference and learning in multi-dimensional Bayesian network classifiers. Lecture Notes in Artificial Intelligence 4724, 501–511.

Demsar, J., 2006. Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30.

Devroye, L., Wagner, T., 1979. Distribution-free perfomance bounds with the resubstitution error estimate. IEEE Transactions on Information Theory 25 (2), 208–210.

Duda, R., Hart, P., Stork, D., 2001. Pattern Classification. Wiley Interscience.

Dumais, S., Chen, H., 2000. Hierarchical classification of web content. In: SIGIR '00: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, New York, NY, USA, pp. 256–263.

Durillo, J., Nebro, A., Luna, F., Dorronsoro, B., Alba, E., December 2006. jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics. Tech. Rep. ITI-2006-10, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, E.T.S.I. Informática, Campus de Teatinos.

Efron, B., Tibshirani, R., 1993. An introduction to the bootstrap. Vol. 57 of Monographs on statistics and applied probability. Chapman and Hall.

Fayyad, U. M., Irani, K. B., 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In: International Joint Conference on Artificial Intelligence (IJCAI). pp. 1022–1029.

Freitas, A. A., 2004. A critical review of multi-objective optimization in data mining: a position paper. SIGKDD Exploration Newsletter 6 (2), 77–86.

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

García, S., Herrera, F., December 2008. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. Journal of Machine Learning Research 9, 2677–2694.

Handl, J., Kell, D., Knowles, J., 2007. Multiobjective optimization in bioinformatics and computational biology. IEEE/ACM Transactions on Computational Biology and Bioinformatics 4 (2), 279–292.

Kullback, S., 1959. Information theory and statistics. John Wiley and Sons., New York.

Langley, P., Iba, W., Thompson., K., 1992. An analysis of Bayesian classifiers. In: Proceedings of the 10th National Conference on Artificial Intelligence. pp. 223–228.

Larrañaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., Dizdarevic, S., 1999. Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artificial Intelligence Review 13 (2), 129–170.

Larrañaga, P., Lozano, J. A., Peña, J. M., Inza, I., 2005. Editorial. Machine Learning 59 (3), 211–212.

Lee, S., von Allmen, P., Fink, W., Petropoulos, A., Terrile, R., 2005. Comparison of multi-objective genetic algorithms in optimizing Q-Law-Thrust orbit transfers. In: Genetic and Evolutionary Computation Conference (GECCO) late-breaking papers.

Li, H., Zhang, Q., April 2009. Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II. IEEE Transactions on Evolutionary Computation 13 (2), 229–242.

McLachlan, G., 1992. Discriminant Analysis and Statistical Pattern Recognition. Wiley.

Miettinen, K., 1999. Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston, MA.

Minsky, M., 1961. Steps toward artificial intelligence. Proceedings of the Institute of Radio Engineers 49 (1), 8–30.

Neapolitan, R., 2003. Learning Bayesian Networks. Prentice Hall.

Osyczka, A., 1985. Multicriteria Optimization for Engineering Design. Academic Press.

Pearl, J., 1988. Probabilistic Reasoning in Intelligence Systems. Springer series in Statistics. Morgan-Kaufman.

Rodríguez, J. D., Lozano, J. A., 2008. Multi-objective learning of multi-dimensional Bayesian classifiers. In: Proceedings of the Eighth International Conference on Hybrid Intelligent Systems, HIS 2008. pp. 501–506.

Rodríguez, J. D., Perez, A., Lozano, J. A., 2010. Sensitivity analysis of k-fold cross validation in prediction error estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (3), 569–575.

Stadler, W., 1988. Multicriteria Optimization in Engineering and in the Sciences. Springer.

Stone, M., 1974. Cross-validatory choice and assessment of statistical predictions. Journal of the Royal Statistical Society Series B 36.

Tax, D. M. J., Duin, R. P. W., 2002. Using two-class classifiers for multiclass classification. International Conference on Pattern Recognition 2, 20124.

Tsoumakas, G., Katakis, I., 2007. Multi-label classification: An overview. International Journal of Data Warehousing and Mining 3 (3), 1–13.

van der Gaag, L., de Waal, P., 2006. Muti-dimensional Bayesian network classifiers. In: Proceedings of the Third European Workshop in Probabilistic Graphical Models. pp. 107–114.

Witten, I., Frank, E., 2000. Data mining: practical machine learning tools and techniques with Java implementations. The Morgan Kaufmann series in Data Management Systems.

Zhang, Q., Li, H., 2007. MOEA/D: A multi-objective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation 11 (6), 712–7314, Winner of the IEEE Transactions on Evolutionary Computation Outstanding Paper Award.

Zitzler, E., Thiele, L., Nov 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. Evolutionary Computation, IEEE Transactions on 3 (4), 257–271.