International Journal of Approximate Reasoning 50 (2009) 341–362

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

Bayesian classifiers based on kernel density estimation: Flexible classifiers

Aritz Pérez *, Pedro Larrañaga, Iñaki Inza

Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, University of The Basque Country, Spain

A R T I C L E   I N F O

Article history:
Received 17 September 2007
Received in revised form 16 August 2008
Accepted 22 August 2008
Available online 13 September 2008

Keywords:
Bayesian network
Kernel density estimation
Supervised classification
Flexible naive Bayes

A B S T R A C T

When learning Bayesian network based classifiers continuous variables are usually handled by discretization, or assumed that they follow a Gaussian distribution. This work introduces the *kernel based Bayesian network* paradigm for supervised classification. This paradigm is a Bayesian network which estimates the true density of the continuous variables using kernels. Besides, tree-augmented naive Bayes, *k*-dependence Bayesian classifier and complete graph classifier are adapted to the novel *kernel based Bayesian network* paradigm. Moreover, the strong consistency properties of the presented classifiers are proved and an estimator of the mutual information based on kernels is presented. The classifiers presented in this work can be seen as the natural extension of the *flexible naive Bayes classifier* proposed by John and Langley [G.H. John, P. Langley, Estimating continuous distributions in Bayesian classifiers, in: Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence, 1995, pp. 338–345], breaking with its strong independence assumption.

Flexible tree-augmented naive Bayes seems to have superior behavior for supervised classification among the flexible classifiers. Besides, flexible classifiers presented have obtained competitive errors compared with the state-of-the-art classifiers.

1. Introduction

Supervised classification [6,26] is an outstanding task in data analysis and pattern recognition. It requires the construction of a classifier, that is, a function that assigns a class label to instances described by a set of variables. There are numerous classifier paradigms, among which *Bayesian networks* (BN) [60,65], based on *probabilistic graphical models* (PGMs) [10,50], are well known and very effective in domains with uncertainty. A Bayesian network is a directed acyclic graph of nodes representing variables, and arcs representing conditional independence relations between triplets of sets of variables. This kind of PGM assumes that each random variable follows a conditional probability distribution given a specific value of its parent variables. BNs are used to encode a factorization of the joint distribution among the domain variables, based on the conditional independencies represented by the directed graph structure. This fact, combined with the Bayes rule, can be used for classification.

In order to induce a classifier from data, two types of variables are considered: the class variable or *class* C, and the rest of the variables or *predictors* $\boldsymbol{X} = (X_1, \ldots, X_d, X_{d+1}, \ldots, X_{d'})$. $\{X_1, \ldots, X_d\}$ is the set of continuous predictors and $\{X_{d+1}, \ldots, X_{d'}\}$ is the set of discrete predictors. Assuming a symmetric loss function, the process of classifying an instance $\boldsymbol{x}$ consists of choosing the class c^* with the highest *a posteriori* probability, $c^* = \arg_c \max p(c|\boldsymbol{x})$. This entails the use of the winner-takes-all rule. The *a posteriori* distribution can be obtained in the following way with the BNs:

* Corresponding author. Tel.: +34 943018070; fax: +34 943015590.
E-mail addresses: aritz.perez@ehu.es (A. Pérez), pedro.larranaga@ehu.es (P. Larrañaga), inza@si.ehu.es (I. Inza).

0888-613X/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2008.08.008

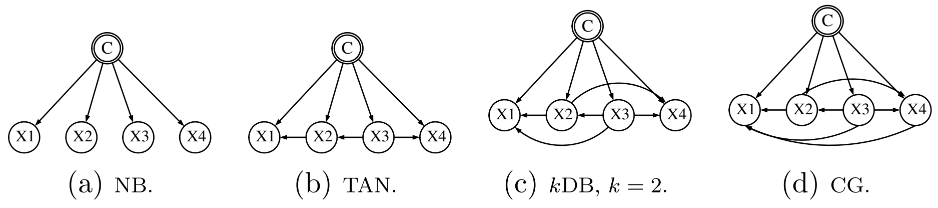


Fig. 1. Different structure complexities of BN based classifiers.

$$p(c|\boldsymbol{x}) \propto \rho(c,\boldsymbol{x}) = p(c)f(\boldsymbol{x}|c) = p(c)\prod_{i=1}^{d} f(x_i|\boldsymbol{pa}_i)\prod_{j=d+1}^{d'} p(x_j|\boldsymbol{pa}_j) \tag{1}$$

where $\boldsymbol{pa}_i$ is an instantiation of the predictors $\boldsymbol{pa}_i$, which is the set of parents of X_i in the graph, $p(\cdot)$ is a probability distribution, $f(\cdot)$ is a density function and $\rho(\cdot)$ a generalized probability function [17]. This kind of classifier is known as *generative*, and it forms the most common approach in the BN literature for classification [11,25,30,47,48,56,64,73]. Generative classifiers use the joint probability function of the predictor variables and the class, $\rho(c,\boldsymbol{x})$. They classify a new instance by using the Bayes rule in order to compute the posterior probability, $p(c|\boldsymbol{x})$ of the class variable given the values of the predictors (see Eq. (1)). On the other hand, *discriminative* classifiers [35,40,70,74] directly model the posterior probability of the class conditioned on the predictor variables $p(c|\boldsymbol{x})$. This work presents a set of generative classifiers based on a new family of BNs that we call *kernel based Bayesian network*.

It has to be highlighted that this work is centered on the difficulties of modeling continuous variables directly (without discretization), and their relations in BN paradigm. The modeling of the mixed domains in BNs can be done as follows: to learn a density function conditioned on a discrete variable is equivalent to creating the partitions in the database inducted by the different values of the discrete variable, and then learn the density functions associated with each value of the discrete variable uniquely from the cases of its corresponding partition. In order to improve the readability and simplicity of the paper, from here on, we deal with continuous domains only (without discrete predictors).

One of the simplest classifiers based on BNs is the *naive Bayes* (*NB*) [25,48,56]. NB assumes that the predictors are conditionally independent given the class. Fig. 1a represents an NB structure. In spite of this strong assumption, its performance is surprisingly good, even in databases which do not hold with the independence assumption [23].

The good performance of the NB classifier has motivated the investigation of classifiers based on BNs which relax this strong independence assumption. A set of examples of different structures which break the conditional independence assumption are shown in Fig. 1b–d. They are ordered by their structure complexity, which is related with the number of dependencies between variables that they capture. Thus, the complexity ranges from the simplest naive Bayes to the complete graph (CG) structure. On one hand, NB structure forbids any arc between predictor variables. On the other hand, the CG structure includes all the possible arcs between predictors. NB does not model any class conditional dependence between predictors, while CG models all the conditional dependencies. In tree-augmented naive Bayes (TAN) structures the maximum number of parents for a predictor is constrained to one plus the class variable and in k-dependency Bayesian classifier (kDB) structures to k plus the class variable. It must be noted that all the structures shown in Fig. 1 are constrained to graphs with the class variable as the root. The class variable is the father of all predictor variables included in the model (NB augmented classifiers).

The structures themselves represent domain knowledge and can be interpreted in terms of conditional independencies, constructing the associated independence graph. In addition, they represent a factorization of the joint distribution $\rho(c,\boldsymbol{x})$, which is based on the relations of conditional (in)dependencies that are inferred from the structure. In most cases the structures and their corresponding factorizations can allow a representation of the joint distribution with fewer parameters. For example, given a domain with r class labels and d discrete predictors X_i with r_i states, a complete graph structure requires $\Theta(r\prod_{i=1}^{d} r_i)$ parameters.[1] On the other hand, a naive Bayes structure requires $\Theta(\sum_{i=1}^{d} r \cdot r_i)$ parameters.

A classifier based on BNs is determined by the structure of the graph and the distributions and density functions which model the (in)dependence relations between the variables. In order to model a density function of a continuous variable, three approaches are generally considered:

(1) To discretize it and estimate the probability distribution of the discretized variable by means of a multinomial probability distribution.
(2) To directly estimate the density function in a parametric way using, for example, Gaussian densities.
(3) To directly estimate the density function in a non-parametric way using, for example, the kernel density functions [77].
(4) To directly estimate the density in a semi-parametric way using, for example, finite mixture models [28].

[1] The expression "the function α is $\Theta(\beta)$" indicates that, in the worst case, α is bounded tightly by the function β asymptotically [14].

The most widely used approach in the literature on supervised classification is the estimation using the multinomial distribution over the discretized variables. The BN which assumes that all variables follow a multinomial probability distribution is known as *Bayesian multinomial network* [10,65] (*BMN*) . This paradigm only handles discrete variables and if a continuous variable is present, it must be discretized with the consequent loss of information [82]. The loss of information is illustrated in Section 5.1.1 using six continuous artificial domains. In spite of that, using discretization plus multinomial option, the true density is generally estimated with enough accuracy for classification purposes [11]. This approach could have some problems when modeling a graph with a complex structure and/or with variables discretized in many intervals because the number of parameters to be estimated can be very high, e.g. $\Theta(r\prod_{i=1}^{d} r_i)$ for CG. Besides, as the number of parameters to be estimated increases, so does the risk of overfitting. Moreover, the number of relevant cases used to compute each parameter can be very low and, therefore, the statistics obtained might not be robust [38]. A battery of BMN-based classifier induction algorithms of different structural complexities has been proposed in the literature: naive Bayes (NB) [25,48,56], tree-augmented naive Bayes (TAN) [30], k-dependence Bayesian classifier (kDB) [73], semi naive Bayes [47,64], Bayesian network-augmented naive Bayes [11] and general Bayesian network [11]. From here on, the classifiers which are based on BMN paradigm will be called *multinomial classifiers*.

The second approach directly estimates the true density of the variables using a parametric density. The Gaussian function is the most popular proposal and it usually provides a reasonable approximation to many real-world densities [41]. This choice assumes that continuous variables conditioned on a value of their parents follow a conditional Gaussian density. The paradigm, based on BN which makes this assumption, is known as *conditional Gaussian network* (*CGN*) [7,32,50–52,62]. In this work, we reference the classifiers based on CGN as *Gaussian classifiers*. It must be noted that CGNs have fewer complexity difficulties to model complex graphs than BMNs have, e.g. a complete graph with the class and continuous predictors needs only $\Theta(rd^2)$ parameters to be modeled in the CGN paradigm. Besides, the estimation of the parameters is more reliable because they are learned from the partitions induced only by the class (by average n/r cases, where n is the number of cases in the training set). If the true densities are not too far from the Gaussian, the classifiers based on CGNs obtain classification error rates at least equal to the classifiers based on BMNs [62]. Although a Gaussian density may provide a reasonable approximation to many real-world distributions, it is certainly not always the best approximation, e.g., bimodal densities. The decrease of the classification performance due to the loss of information under the Gaussian assumption is illustrated in Section 5.1.1. This suggests another direction in which we might profitably extend and improve the Gaussian (parametric) approach: by using more general approaches to density estimation, e.g., kernel density estimation [41] (non-parametric).

In order to break with the strong parametric assumption, this work presents the kernel based Bayesian network (KBN) paradigm. The KBN paradigm uses the non-parametric *kernel density estimation* for modeling the conditional density of a continuous variable given a specific value of its parents, $f(\boldsymbol{x}|C=c)$. The kernel estimation method can approximate more complex distributions than the Gaussian parametric approach. Moreover, kernel density estimation can be seen as a more flexible[2] estimator compared to the multinomial distribution, in the same way that kernel density estimation is considered more flexible than the histograms [77].

The alternative for breaking with the strong parametric assumption is the semi-parametric approach. The semi-parametric estimation combines the advantages of both parametric and non-parametric estimators. This approach is not restricted to specific functional forms, and the size of the model only grows with the complexity of the problem being solved [4]. On the other hand, the process of learning the semi-parametric model using the data set is computationally intensive compared to the simple procedures needed for parametric or non-parametric learning procedures [4]. One of the most commonly used non-parametric approaches is the so-called mixture model [28,55] and, especially, Gaussian mixture model [5,55]. However, this work is focused in the non-parametric kernel based approach.

This work is an introduction of the KBN paradigm limited to the supervised classification using the Bayes rule. Consequently, we present the KBN paradigm and a battery of classifier induction algorithms supported by it, many of them adapted from the algorithms developed for the Bayesian multinomial network paradigm (NB [25,48,56], TAN [30] and kDB [73]). The classifiers presented are ordered by their structural complexity, ranging from naive Bayes to complete graph. We call the classifiers based on the KBN paradigm *flexible classifiers*. The origin of the term flexible comes from flexible naive Bayes classifier [41], i.e. the NB structure in the KBN framework.

It must be noted that the BMN, CGN and KBN paradigms handle discrete variables similarly. Thus, given a discrete domain (with discrete predictors only) and a structure, the three paradigms represent the same factorization of the joint distribution. The experimentation of this work has been performed in continuous domains for highlighting the differences between the BMN, CGN and KBN paradigms.

The paper is organized as follows. Section 2 introduces the non-parametric kernel based density estimation which will be used to learn the KBN based classifiers. Section 3 presents a group of novel estimators for the computation of the amount of mutual information based on kernels. Their corresponding expressions will be used by the classifier induction algorithms proposed. Section 4 introduces a set of classifier induction algorithms, based on KBN. They are ordered by their structural complexity: flexible naive Bayes, flexible tree-augmented Bayesian classifier, flexible k-dependence Bayesian classifier

[2] We use the *flexibility* term in a qualitative way, not as a quantitative measure. It represents the capability of modeling densities of different shapes with precision. For example, kernel density estimation is more flexible than the Gaussian approach because it can model clearly better densities with more than one mode (see Section 5.1.1).

and flexible complete graph classifier. Note that except for flexible naive Bayes [41], the rest of these classifiers are novel approaches. The computational and storage requirements of the proposed algorithms are also analyzed. Moreover, the desirable asymptotic property of the flexible classifiers (strong pointwise consistency [41]) is proved. In Section 5 the experimental results for the classifiers proposed are presented and analyzed. The experimental study includes results in ten kinds of artificial datasets and 21 datasets of the UCI repository [59]. The artificial domains illustrate the advantages of the classifiers based on KBN paradigm, which model the correlations between predictors, for handling continuous predictor variables. They also illustrate the loss of information due to the discretization process and the effect of the smoothing degree in the classification performance. The classification error in the 21 UCI repository datasets is estimated for the presented algorithms and for 10 benchmarks. The estimated errors are compared using the Friedman plus Nemenyi post hoc test [19]. Additionally, the effect of the smoothing degree in classification performance is studied in the selected datasets. The estimation of the bias plus variance decomposition of the expected error [46] is also performed in some of the selected UCI data sets. And finally, Section 6 summarizes the main conclusions of our paper and exposes the future work related with the KBN paradigm for classification.

2. Kernel density estimation

The kernel based d-dimensional estimator [80] in its more general form is

$$f(\boldsymbol{x};\boldsymbol{H}) = n^{-1}\sum_{i=1}^{n} K_{\boldsymbol{H}}(\boldsymbol{x}-\boldsymbol{x}^{(i)}) \tag{2}$$

where $\boldsymbol{H}$ is a $d \times d$ bandwidth or smoothing matrix (BM), $\boldsymbol{x} = (x_1,\ldots,x_d)$ is a d-dimensional instantiation of $\boldsymbol{X}$, n is the number of cases from which the estimator is learned, i is the index of a case in the training set, and $K_{\boldsymbol{H}}(\cdot)$ is the kernel function used. Note that $\boldsymbol{X}$ is a multivariate variable and, thus, $f(\boldsymbol{x};\boldsymbol{H})$ is a multivariate function. The kernel based density estimate $f(\cdot;\boldsymbol{H})$ is determined by averaging n kernel densities $K_{\boldsymbol{H}}(\cdot)$ placed at each observation $\boldsymbol{x}^{(i)}$. The kernel function $K_H(\cdot)$ used is defined as:

$$K_{\boldsymbol{H}}(\boldsymbol{x}) = |\boldsymbol{H}|^{-1/2}K(\boldsymbol{H}^{-1/2}\boldsymbol{x}) \tag{3}$$

assuming that K is a d-dimensional density function. A kernel density estimator is characterized by means of

(1) The kernel density K selected.
(2) The bandwidth matrix $\boldsymbol{H}$.

$\boldsymbol{H}$ plays the role of scaling factor which determines the spread of the kernel at each coordinate direction. The kernel density estimate is constructed centering a scaled kernel at each observation. So, the kernel density estimator is a sum of bumps placed at the observations. The kernel function $K(\cdot)$ determines the shape of the bumps. The value of the kernel estimate at the point $\boldsymbol{x}$ is simply the average of the n kernels at that point. One can think of the kernel as spreading a "probability mass" of size $1/n$ associated with each data point in its neighbourhood. Combining contributions from each data point means that in regions where there are many observations it is expected that the true density has a relatively large value. The choice of the shape of the kernel function is not particularly important. However, the choice of the value for the bandwidth matrix is very important for density estimation [80]. Examples for the univariate and bivariate density estimation are shown in Fig. 2a and b. Our KBN paradigm uses the d-dimensional Gaussian density with identity covariance matrix, $\boldsymbol{\Sigma} = \boldsymbol{I}$, in order to estimate density functions:

$$K(\boldsymbol{x}) = (2\pi)^{-d/2}\exp(-1/2\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}) \sim N(\boldsymbol{0},\boldsymbol{I}) \tag{4}$$

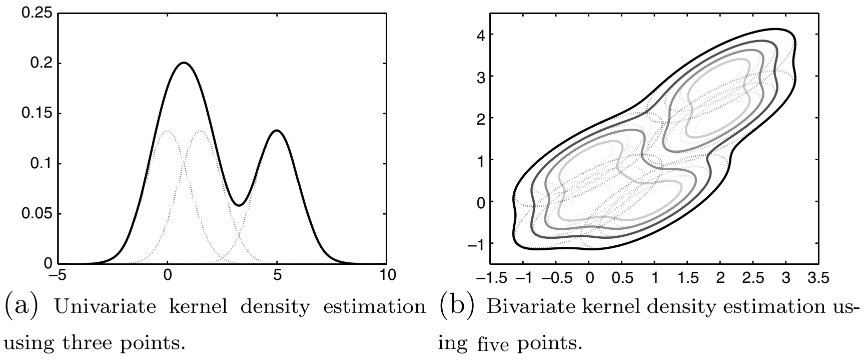


Fig. 2. Univariate and bivariate Gaussian kernel density estimators. Broken and solid lines represent the contribution of each kernel to the density estimation and the kernel based density estimate, respectively.

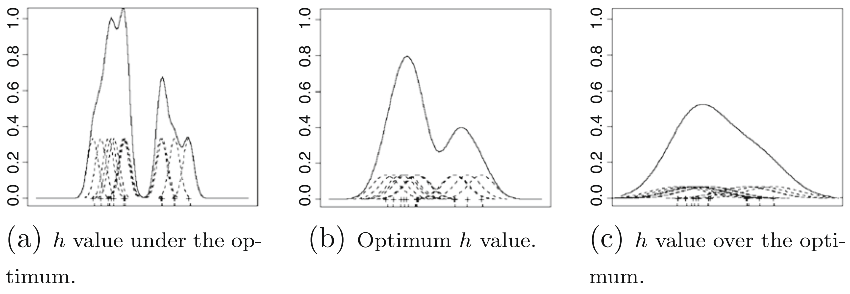


Fig. 3. The effect in the density estimation of different smoothing degrees, controlled by the parameter h.

It must be noted that one can use another kernel function as well, such as Epanechnikov, biweight, triweight, triangular, rectangular or uniform [77,80].

Thus, $K_{\boldsymbol{H}}(\boldsymbol{x} - \boldsymbol{x}^{(i)})$ is equivalent to $N(\boldsymbol{x}^{(i)}, \boldsymbol{H})$ density function. $\boldsymbol{H}$ is a square symmetric matrix $d \times d$, and it has, in its most general form, $\frac{d(d+1)}{2}$ different parameters. This number of parameters can be very high, even for low dimensional densities. This suggests constraining $\boldsymbol{H}$ to a less general form. For example, if a diagonal matrix is used, only d different parameters must be learned.

2.1. Selecting the bandwidth matrix $\boldsymbol{H}$

The selection of a good BM $\boldsymbol{H}$ is crucial for the density estimation, even more than the kernel generic function $K(\cdot)$ [18]. $\boldsymbol{H}$ establishes the degree of smoothing of the density function estimation.

In the univariate case, the smoothing degree depends on a unique parameter, h. Fig. 3 shows the effect of the parameter h in the estimation of the true density using the same twelve training cases. Intuitively, with h near to zero, a noisy estimation is obtained by the *undersmooth* effect (see Fig. 3a). As h increases the noise in the estimation is reduced and the univariate density begins to approximate the true density, until the optimum[3] is reached (see Fig. 3b). As h increases, and distances itself from the optimum, the estimation starts to lose details due to the *oversmooth* effect (see Fig. 3c). Finally, as h tends to ∞, the function becomes uniform.

The degree of smoothing is crucial for density estimation [80], e.g. for minimizing the mean squared error. On the other hand, the influence of the smoothing degree does not have such a crucial role in classification problems under 0–1 loss functions, since density estimation is only an intermediate step for classifying. For example, Gaussian approach obtains good results in many non-Gaussian domains [62]. Intuitively, smoothing degree controls the sensitivity of the classifier to changes in the training set. When the smoothing degree tends to zero, flexible complete graph classifier tends to the nearest neighbor classifier (maximum sensitivity). When the smoothing degree tends to infinity, it tends to the Euclidean distance classifier (minimum sensitivity) [68]. The effect of the smoothing degree in supervised classification has been empirically studied in Sections 5.1.2 and 5.2.3.

In order to guarantee an efficient parametrization of a great number of densities, our goal is to use a computationally inexpensive technique to compute the BM $\boldsymbol{H}$. The number of parameters to be estimated for the specification of a full bandwidth matrix for a d-dimensional density is $d(d+1)/2$. This number becomes unmanageable very quickly as d increases, which suggests that $\boldsymbol{H}$ should be restricted to a simpler form [78]. In this work we use *differential scaled* [78] approach because it depends on a unique smoothing parameter $h : \boldsymbol{H} = diag(h_1^2, \ldots, h_d^2) = h^2 \mathrm{diag}(s_1^2, \ldots, s_d^2)$, where h_i is the smoothing parameter of the variable X_i, s_i is a dispersion measure (e.g. sample standard deviation of X_i) and $diag(\cdot)$ represents a diagonal matrix. This parametrization allows different amounts of smoothing in each coordinate direction and it performs an implicit standardization of the variables.

The principal problem selecting optimum BM, in the majority of practical situations, is that the real density function is unknown. Thus, in general, it is impossible to optimize any appropriate loss criteria (a distance function between the real and estimated densities). In our work, we use the *normal rule* [77]. It selects the BM that minimizes the *mean integrated square error* of the estimated density, under the assumption that the variables follow a multidimensional Gaussian density with identity covariance matrix $\boldsymbol{I}$. Under this assumption we can compute the optimum h as follows [77]:

$$h = \left(\frac{4}{(m+2)n}\right)^{\frac{1}{m+4}} \tag{5}$$

where m is the number of continuous variables in the density function to be estimated in spite of the number of continuous variables in the domain, d. For example, in order to model the density $f(x_1, \ldots, x_l)$ we set $m = l$ in Eq. (5).

We have chosen the normal rule plus differential scaled [78] approach because it obtains density estimations that are good enough for classifying (see Sections 5.1.2 and 5.2.3). Furthermore, we know that the normal rule tends to oversmooth

[3] The optimum is defined in terms of a loss function. Mean squared error and classification error are usually used in density estimation and classification, respectively. The optimum term is defined as the parameter value which minimizes the selected loss function.

Table 1
Storage and computational requirements of a classifier with a kDB structure based on both KBN paradigm, with the differential scaled bandwidth matrix plus the normal rule approach, and CGN paradigm

CGN based		KBN based	
Training space	Testing time	Training space	Testing time
$\Theta(rd^2)$	$\Theta(rd(k+1))$	$\Theta(nd+r(d+1))$	$\Theta(nd(k+1))$

The domain has d predictor variables and r classes, the training set has n cases. The computational requirements have been obtained for classifying a new instance.

[80] the estimation and, thus, it could obtain more stable estimations with noisy data, i.e. estimations with less variance. At the same time, it avoids the increase of the high computational requirements related to the classifier learning process (see Table 1). It only has to compute a fixed number of operations independently from the dimension of the density and the number of examples used to estimate it. There are other approaches to decide the smoothing degree for obtaining a good classification performance. The wrapper approach [43,45] could be one of the most interesting procedures from the performance point of view (see Sections 5.1.2 and 5.2.3). But it involves intensive computations with flexible classifiers, especially compared with the normal rule.

3. Mutual information and conditioned mutual information estimators

In this section we propose a set of novel estimators based on kernels for the computation of the amount of mutual and conditioned mutual information [15]. These estimators will be used in the classifier induction algorithms presented in Section 4. The estimators are presented in their multivariate form and they inherit the flexibility properties of the kernel density estimation. Thus, we think that they are suitable to compute the amount of mutual information between any pair of continuous variables.

Some mutual information based measures are directly related to the expected error of BN based classifiers at different structural complexity levels [61]. Improving the conditioned mutual information, $I(X_i;X_j|C)$, is likely to decrease the Kullback Leibler divergence between the true and the estimated joint probability distribution. The algorithm TAN proposed by Friedman et al. [30], which guides the structural learning with the conditioned mutual information, finds a maximum likelihood TAN structure. Besides, the rank of variables obtained with the use of mutual information, $I(X_i;C)$, sorts the variables by the class entropy reduction, $H(C|X_i)$. The class entropy reduction is directly related with the discriminative power of a variable [61]. For example, if $I(X_i;C) > I(X_j;C)$ then $H(C|X_i) < H(C|X_j)$ and, therefore, the classifier which includes only the variable X_i instead of X_j tends to obtain lower classification errors.

The amount of mutual information between two multivariate variables (random vectors) $\boldsymbol{X}$ and $\boldsymbol{Y}$ is defined as [15]

$$I(\boldsymbol{X},\boldsymbol{Y}) = \int f(\boldsymbol{x},\boldsymbol{y}) \log \frac{f(\boldsymbol{x},\boldsymbol{y})}{f(\boldsymbol{x})f(\boldsymbol{y})} \mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{y} = E_{p(\boldsymbol{x},\boldsymbol{x})}\left(\log \frac{f(\boldsymbol{x},\boldsymbol{y})}{f(\boldsymbol{x})f(\boldsymbol{y})}\right) \tag{6}$$

Based on this definition we propose the following estimator, $\widehat{I}(\boldsymbol{X},\boldsymbol{Y})$, for the amount of mutual information between two continuous multivariate variables $\boldsymbol{X}$ and $\boldsymbol{Y}$, $I(\boldsymbol{X},\boldsymbol{Y})$, using kernel density estimation based on a training set of size n, $\{(\boldsymbol{x}^{(1)},\boldsymbol{y}^{(1)}),\ldots,(\boldsymbol{x}^{(n)},\boldsymbol{y}^{(n)})\}$:

$$\widehat{I}(\boldsymbol{X},\boldsymbol{Y}) = \frac{1}{n}\sum_{i=1}^{n} \log \frac{\hat{f}(\boldsymbol{x}^{(i)},\boldsymbol{y}^{(i)})}{\hat{f}(\boldsymbol{x}^{(i)})\hat{f}(\boldsymbol{y}^{(i)})} \tag{7}$$

where $\hat{f}(\cdot)$ is a kernel based density estimation (see Eq. (2)). Note that Eq. (7) estimates the expectance of the function $\log[f(\boldsymbol{x},\boldsymbol{y})/(f(\boldsymbol{x})f(\boldsymbol{y}))]$, when the instances of the training set are independent and identically distributed [6]. This estimation becomes exact in the limit, as $n \to \infty$ [6]. Another proposal based on kernels can be found in [57].

Alternatively, in order to compute the amount of mutual information between a multivariate continuous variable $\boldsymbol{X}$ and a multivariate discrete variable $\boldsymbol{C}$ we propose the following estimator:

$$\widehat{I}(\boldsymbol{X},\boldsymbol{C}) = \frac{1}{n}\sum_{i=1}^{n} \log \frac{\hat{\rho}(\boldsymbol{x}^{(i)},\boldsymbol{c}^{(i)})}{\hat{f}(\boldsymbol{x}^{(i)})\hat{p}(\boldsymbol{c}^{(i)})} = \frac{1}{n}\sum_{i=1}^{n} \log \frac{\hat{f}(\boldsymbol{x}^{(i)}|\boldsymbol{c}^{(i)})}{\hat{f}(\boldsymbol{x}^{(i)})} \tag{8}$$

where $\hat{\rho}(\boldsymbol{x}^{(i)},\boldsymbol{c}^{(i)}) = \hat{p}(\boldsymbol{c}^{(i)})\hat{f}(\boldsymbol{x}^{(i)}|\boldsymbol{c}^{(i)})$. $\hat{p}(\boldsymbol{c})$ is the estimated multinomial distribution of variable $\boldsymbol{C}$ and $\hat{f}(\boldsymbol{x}|\boldsymbol{c})$ is the kernel based density estimator of variable $\boldsymbol{X}$ conditioned on $\boldsymbol{C}=\boldsymbol{c}$ (computed with the partition of the cases which have the class value $\boldsymbol{C}=\boldsymbol{c}$).

The amount of mutual information between two multivariate continuous variables, $\boldsymbol{X}$ and $\boldsymbol{Y}$, conditioned on a multidimensional multinomial variable $\boldsymbol{C}$ is defined as

$$I(\boldsymbol{X},\boldsymbol{Y}|\boldsymbol{C}) = \sum_{\boldsymbol{c}=1}^{|\boldsymbol{C}|} p(\boldsymbol{c}) I_c(\boldsymbol{X},\boldsymbol{Y}) \tag{9}$$

where $I_{\boldsymbol{c}}(\boldsymbol{X},\boldsymbol{Y}) = E_{p(\boldsymbol{x},\boldsymbol{x}|\boldsymbol{C}=\boldsymbol{c})}\left(\log \frac{f(\boldsymbol{x},\boldsymbol{y}|\boldsymbol{c})}{f(\boldsymbol{x}|\boldsymbol{c})f(\boldsymbol{y}|\boldsymbol{c})}\right)$. Using Eqs. (7) and (9), we propose the following estimator for the amount of conditioned mutual information:

$$\widehat{I}(\boldsymbol{X},\boldsymbol{Y}|\boldsymbol{C}) = \sum_{\boldsymbol{c}=1}^{|\boldsymbol{C}|} \hat{p}(\boldsymbol{c}) \frac{1}{n_{\boldsymbol{c}}} \sum_{i=\boldsymbol{c}:1}^{\boldsymbol{c}:n_c} \log \frac{\hat{f}(\boldsymbol{x}^{(i)},\boldsymbol{y}^{(i)}|\boldsymbol{c})}{\hat{f}(\boldsymbol{x}^{(i)}|\boldsymbol{c})\hat{f}(\boldsymbol{y}^{(i)}|\boldsymbol{c})} \tag{10}$$

The super-index $\boldsymbol{c}:j$ refers to the jth case in the partition induced by the value $\boldsymbol{c}$, and $n_{\boldsymbol{c}}$ is the number of cases verifying that $\boldsymbol{C} = \boldsymbol{c}$.

All the estimators presented in Eqs. (7), (8) and (10) can be adjusted to any KBN based classifier because $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{C}$ can be of any arbitrary dimension.

4. Classifiers based on kernel based Bayesian networks: flexible classifiers

The densities to be estimated in the KBN paradigm with continuous predictors are of the form $f(\boldsymbol{x}|\boldsymbol{y},c)$, $\boldsymbol{X}$ and $\boldsymbol{Y}$ being continuous multivariate predictor variables and C the class variable, a univariate discrete variable. That is the essence of the KBN paradigm. The estimation of the density function $f(\boldsymbol{x}|\boldsymbol{y},c)$ can be determined in the following form using kernel density estimation:

$$f(\boldsymbol{x}|\boldsymbol{y},c) \simeq \hat{f}(\boldsymbol{x}|\boldsymbol{y},c) = \frac{\hat{f}(\boldsymbol{x},\boldsymbol{y}|c)}{\hat{f}(\boldsymbol{y}|c)}$$

As we have explained at the beginning of Section 1, in order to classify a new unlabeled instance, $\boldsymbol{x}$, the factorization of $\rho(c,\boldsymbol{x})$ is evaluated for every value of the class, $c \in \{1,\ldots,r\}$. Then the a posteriori probabilities for each class, $p(c|\boldsymbol{x})$, are obtained using the Bayes rule. Finally, classification is performed under winner-takes-all rule, i.e. choosing the class c^* with the maximum a posteriori probability, $c^* = arg_c max p(c|\boldsymbol{x})$. Note that in the classification of an unlabeled case all the instances in the training set are used for computing the factorization based on kernel density estimation (see Section 2).

Typically, for modeling domains with continuous and discrete variables CGN paradigm has the following structural constraint: a continuous predictor cannot be parent of a discrete predictor. This constraint could be avoided for the KBN paradigm (and also for the CGN) using the Bayes rule as follows:

$$p(\boldsymbol{x}_d|\boldsymbol{x}_c,c) = p(\boldsymbol{x}_d,c)p(\boldsymbol{x}_c|\boldsymbol{x}_d,c)/p(c)p(\boldsymbol{x}_c|c)$$

where $\boldsymbol{X}_d$ is a multivariate discrete variable (discrete random vector) and $\boldsymbol{X}_c$ a multivariate continuous one. However, as we have noted before, this work is centered on the difficulties of modeling continuous variables and their relations in the KBN paradigm.

The following sections present a group of flexible classifier induction algorithms, ordered by their structural complexity: flexible naive Bayes, flexible tree-augmented naive Bayes, flexible k-dependence Bayesian classifier and complete graph Bayesian classifier. Note that all the classifiers presented in this paper are extensions of the flexible naive Bayes classifier [41]. Thus, in the presented models, the class variable is the father of each predictor variable. Examples of their different structure complexities are shown in Fig. 1.

4.1. Flexible naive Bayes

The least complex structure is the naive Bayes structure (NB structure), which assumes that predictor variables are conditionally independent given the class. The *naive Bayes* classifier induction algorithm (*NB*) [25,48,56] learns the complete naive Bayes structure, which is known *a priori*. The accuracy obtained with this classifier is surprisingly high in some domains, even in datasets that do not obey its strong conditional independence assumption [23]. Thanks to the conditional independence assumption, the factorization of the joint probability is greatly simplified.

Originally, NB classifier was introduced for the BMN (multinomial NB or mNB) and it was adapted to the KBN paradigm (flexible NB or fNB) by John and Langley [41]. This classifier has been recently used in [8,53,54]. It must be noted that our version of the fNB uses the normal rule [77] for computing h instead of the heuristic proposed in [41].

4.2. Flexible tree-augmented naive Bayes

The tree-augmented naive Bayes structures (TAN structures) break with the strong independence assumption made by NB structures, allowing probabilistic dependencies among predictors. The TAN structures consist of graphs with arcs from the class variable to all the predictors, and with arcs between predictors, taking into account that the maximum number of parents of a variable is one plus class. Example of the TAN structure is shown in Fig. 1b.

This section introduces the novel adaptation of Friedman et al.'s [30] algorithm to the KBN paradigm. Friedman et al.'s algorithm [30] follows the general outline of Chow and Liu's procedure [13], but instead of using the mutual information between two variables, it uses conditional mutual information between predictors given the class variable to construct the maximal weighted spanning tree. In order to adapt this algorithm to continuous variables, we need to estimate the mutual information between every pair of continuous predictor variables conditioned by the class variable $I(X_i,X_j|C)$, computing the estimator proposed in Eq. (10).

The algorithm [30] starts from a complete NB structure and continues adding allowed edges between predictor until the complete TAN structure is formed. The edges are included in the order of decreasing estimation of the conditional mutual information, $I(X_i, X_j|C)$. The edges included between predictors create a tree structure. In order to decide the direction of the edges, we randomly select one predictor as the root of the tree and transform the edges between predictors into arcs. Originally, the TAN algorithm was introduced for BMN (multinomial TAN of mTAN) [30]. This paper adapts it to the novel KBN paradigm (flexible TAN or fTAN).

4.3. Flexible k-dependence Bayesian classifier

The k-dependence Bayesian classifier structure (kDB structure) extends TAN structures allowing a maximum of k predictor parents plus the class for each predictor variable (NB, TAN and CG structures are equivalent to kDB structures with $k = 0, k = 1$ and $k = n - 1$, respectively). An example of the kDB structure is shown in Fig. 1c.

The kDB structure allows each predictor X_i to have not more than k predictor variables as parents. There are two reasons to restrict the number of parents of a variable with algorithms based on BMNs. Firstly, the reduction of the search space. Secondly, the probability estimated for a multinomial variable becomes more unreliable as additional multinomial parents are added. That is because the size of the conditional probability tables increases exponentially with the number of parents [73], and fewer cases are used to compute the necessary statistics. The use of a KBN instead of a BMN avoids the problem of modeling a structure without the restriction in the number of parents, as the number of required parameters only grows linearly with the differentially scaled plus normal rule approach. In addition, to estimate the parameters, the entire dataset is used instead of learning from a dataset partition. Thus, the KBN paradigm allow the construction of classifiers with a high number of dependencies between continuous variables.

This section introduces the novel adaptation of Sahami's [73] algorithm, called k-dependence Bayesian classifier, to the KBN paradigm. This algorithm is a greedy approach which uses the class conditioned mutual information between each pair of predictor variables $I(X_i, X_j|C)$ and the mutual information between the class and each predictor $I(X_i, C)$ to lead the structure search process. In order to adapt this algorithm to the KBN paradigm, we compute the amounts of mutual information $I(X_i, C)$ and $I(X_i, X_j|C)$, using the estimators proposed in Eqs. (8) and (10), respectively.

Sahami's algorithm [73] starts from a structure with only the class variable. At each step, from the subset of non-included predictor variables, the variable $X_{\max}$ with the highest $I(X_i, C)$ is added. Next, the arcs are added from the predictors previously included in the structure until the variable $X_{\max}$. The arcs are included following the order of $I(X_{\max}, X_j|C)$, from the greatest value to the smallest. They are added as long as the maximum number of parents k is not surpassed. Originally, kDB algorithm was introduced for the BMN (multinomial kDB or mkDB) and we have adapted it to the KBN paradigm (flexible kDB or fkDB).

4.4. Flexible complete graph classifier: Parzen window classifier

All the complete graphs represent the same factorization of the joint distribution, a factorization without any simplification derived from the conditional (in)dependence statements. We can take any of the complete acyclic graphs for classification because they are Markov equivalent [12]. Therefore, the structure of the *flexible complete graph* classifier (fCG) can be fixed randomly provided that there are no cycles. An example of this structure is shown in Fig. 1d.

The CG structure can be seen as the particular case of kDB when $k \geqslant d - 1$, but avoiding the cost associated with its structural learning. Moreover, the fCG classifier is equivalent to the Parzen window classifier [31,63,71].

4.5. Storage and computational complexity

KBN based classifiers require more storage space and have more computational cost than CGN based classifiers for learning and classifying new instances. First, we introduce the requirements associated with the structural learning and then, the requirements associated with the parametric learning. At the end, the requirements related to each of the KBN based classifiers introduced are presented.

In order to obtain the structure to be modeled from data (structural learning), generally, the classifier induction algorithms proposed in this section need to compute the amount of mutual information between every pair of variables conditioned on the class (for all i and $j, I(X_i, X_j|C)$), and the mutual information between each variable and the class (for all $i, I(X_i, C)$).[4] The number of operations involved for computing $I(X_i; X_j|C)$, using the estimator based on kernels proposed in Eq. (10), is $\Theta\left(\sum_c n_c^2\right)$, where n_c is the number of instances in the partition of the training set induced by c. Thus, TAN and kDB algorithms need $\Theta\left(d^2 \sum_c n_c^2\right)$ computations, where d is the number of predictor variables included in the model. The computational cost for computing $I(X_i; C)$, given the estimator proposed in Eq. (7), is $\Theta(n^2)$. Therefore, kDB algorithm needs $\Theta(dn^2)$ computations additionally.

For modeling a classifier based on the KBN paradigm (using differentially scaled plus normal rule approach) from data given its structure (parametric learning), every instance in the training set and the variances of the predictors conditioned

[4] The structure for NB and CG classifiers are given *a priori* and, therefore, their computational cost is constant.

on each class c are stored. Therefore, a classifier with kDB structure stores $\Theta(nd + r(d+1))$ values (cases and parameters) to proceed with classification, regardless wether considering NB, TAN or CG complexity. In order to classify a new instance, the same KBN based classifier requires to compute $\Theta(nd(k+1))$ operations. Table 1 summarizes the storage and computational complexity for both CGN and KBN based classifiers given a kDB structure.

Therefore, the number of operations for learning each flexible classifier proposed is: $\Theta(nd)$ for fNB [41] and fCG, $\Theta\left(d^2\sum_c n_c^2\right)$ for fTAN, and $\Theta\left(d^2\sum_c n_c^2 + dn^2\right)$ for fkDB. Due to the high computational requirements for learning the classifier and for classifying new instances, it can be said that flexible classifiers are more suitable for low and medium sized datasets.

4.6. Asymptotic properties of flexible classifiers

In this section we discuss the asymptotic properties of the flexible classifiers for modeling the conditional distribution of the class given the predictors, $p(C|\boldsymbol{X})$. The Section is based on the definition of *strong pointwise consistency* and the theorems and lemmas presented by John and Langley in [41].

Definition 1 (Strong pointwise consistency [41]). If $f(\boldsymbol{x})$ is a probability density function and $\hat{f}_n(\boldsymbol{x})$ is a kernel estimate of $f(\boldsymbol{x})$ based on n independent and identically distributed instances, then $\hat{f}_n(\boldsymbol{x})$ is strongly pointwise consistent if $\hat{f}_n(\boldsymbol{x}) \to f(\boldsymbol{x})$ almost surely for all $\boldsymbol{x}$; i.e., for every $\epsilon > 0, p(\lim_{n\to\infty}|\hat{f}_n(\boldsymbol{x}) - f(\boldsymbol{x})| < \epsilon) = 1$.

The strongest asymptotic results we could hope for regarding flexible classifier would be that its estimate of $p(C|\boldsymbol{X})$ is strongly consistent.

John and Langley demonstrate, based on [9], the consistency of kernel density estimation for modeling continuous variables (*strong consistency for reals* theorem [41]). It must be noted that kernel density estimate using Gaussian kernel with the normal rule satisfies the constraints of the strong consistency for reals. Then, they demonstrate, based on [20], the consistency of the multinomial approach used by the flexible classifiers among others (*strong consistency for nominals* [41]). Furthermore, they demonstrate the consistency of the product of strongly consistent estimates (*consistency of products* [41]) and the related *consistency of the quotient* (see the proof of Theorem 3 in [41]). Following, based on these lemmas and theorems, we demonstrate the strong consistency of a flexible classifier when its structure captures the true conditional dependencies of the domain.

Lemma 2 (Consistency of flexible factors). *The estimates of flexible classifiers for each factor $f(x|\boldsymbol{pa}, c)$ is strongly consistent, where X is a continuous or discrete variable and Pa a set of continuous and/or discrete variables.*

Proof. Any factor $f(x|\boldsymbol{pa}, c)$, using the Bayes rule, can be decomposed into a quotient of products of probability density functions $f(a|b), f(b)$ or $f(a)$, where A is a set of continuous variables and B a set of discrete ones. The probability density functions $f(a|b), f(b)$ and $f(a)$ are strongly consistent. By the strong consistency of the product and the quotient [41] the flexible classifier's estimator of each factor $f(x|\boldsymbol{pa}, c)$ is strongly consistent. □

Lemma 3 (Consistency of flexible classifiers). *Let the true conditional distribution of the class given the attributes be $p(C|\boldsymbol{X}) = \frac{p(C)\prod_{i=1}^d f(x_i|\boldsymbol{Pa}_i, C)}{\prod_{i=1}^d f(x_i|\boldsymbol{Pa}_i)}$. A flexible classifier $\hat{p}(C|\boldsymbol{X}) = \frac{\hat{p}(C)\prod_{i=1}^d \hat{f}(x_i|\boldsymbol{Pa}_i, C)}{\prod_{i=1}^d \hat{f}(x_i|\boldsymbol{Pa}_i)}$ is a strongly consistent estimator of $p(C|\boldsymbol{X})$.*

Proof. By Lemma 2 flexible classifier's estimates of factors $P(C), f(x_i|\boldsymbol{Pa}_i)$ and $f(x_i|\boldsymbol{Pa}_i, C)$ are strongly consistent. Then, by the strong consistency of products and quotients, flexible classifier's estimate of $p(C|\boldsymbol{X})$ is strongly consistent. □

Therefore, by Lemma 3 a flexible classifier is a strongly consistent estimator when the conditional dependencies of the domain are captured by its structure. This fact is illustrated in Section 5.1.1 by means of six representative artificial domains.

5. Experimental results

This section presents the experimental results of the previously introduced flexible classifiers: from fNB to fCG.

First, in Section 5.1, we study the behavior of the flexible classifiers and its sensitivity to changes in the smoothing degree using artificial domains. Then, in Section 5.2, we present a set of results in 21 *UCI repository* continuous datasets [59]. These results include a comparison of the classifiers using the Friedman plus Nemenyi post hoc test [19] based on the estimated errors (Section 5.2.2), the study of the effect of the smoothing degree in the performance of flexible classifiers (Section 5.2.3), and the bias plus variance decomposition of the expected error [46] (Section 5.2.4).

5.1. Artificial datasets

This section, which includes two subsections, illustrates the behavior of the flexible classifiers using artificial domains. Section 5.1.1 studies the classification performance of the NB and TAN classifiers based on the BMN, CGN and KBN paradigms, by means of six ad hoc designed representative artificial datasets. The use of the flexible classifiers which model the correlation between predictors is justified, presenting their advantages. Section 5.1.2 studies the effect of the smoothing degree in the performance of the flexible classifiers by means of four artificial domains.

At each artificial domain a parameter λ is fixed in order to guarantee a Bayes error of $\epsilon_B = 0.1$. We define the error of a classifier M under the winner-takes-all rule [26] as:

$$\epsilon_M = \int f(\boldsymbol{x})(1 - p(c^*|\boldsymbol{x}))d\boldsymbol{x} \tag{11}$$

where $p(\cdot)$ and $f(\cdot)$ are the true probability distribution and density functions of the domain, respectively, and $c* = \arg_c \max[p_M(c|\boldsymbol{x})]$ under the winner-takes-all rule. The Bayes error ϵ_B is defined as the error of the Bayes classifier, which is obtained by the decision $c^* = arg_c max[p(c|\boldsymbol{x})]$ under the winner-takes-all rule.

5.1.1. Error estimation

This section illustrates the differences between NB and TAN classifiers based on the BMN (plus discretization [27]), CGN and KBN paradigms. For this purpose, we have designed four representative artificial domains with two continuous predictors, $\{X, Y\}$, and two equally probable classes, $c \in \{0, 1\}$ with $p(C = 0) = 0.5$. It must be noted that TAN, kDB (at $k > 0$ values), and complete graph structures are equivalent in the bivariate domains. Taking into account the dependencies between predictor variables (correlated or non-correlated) and four types of densities (Gaussian, Gaussian mixture, chi square and chi square mixture), we propose the following six domains:

- Domains with non-correlated Gaussian predictors (NCG).
- Domains with correlated Gaussian predictors (CG).
- Domains with non-correlated Gaussian mixture predictors (NCGM).
- Domains with correlated Gaussian mixture predictors (CGM).
- Domains with non-correlated chi square predictors (NCCh).
- Domains with correlated chi square mixture predictors (CChM).

We have used the mixture of two or three densities for each class (NCGM, CGM and CChM) so as to model non-Gaussian densities. Note that these densities are very different to a single Gaussian. The chi square domains (NCCh y CChM) are based on the following pseudo-chi square density:

$$f_{\text{chi}}(x; x_0, \kappa) = \begin{cases} \frac{1}{2^{\kappa/2}\Gamma(\kappa/2)} x^{(\kappa/2)-1} e^{-(x-x_0)/2} & x - x_0 > 0 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

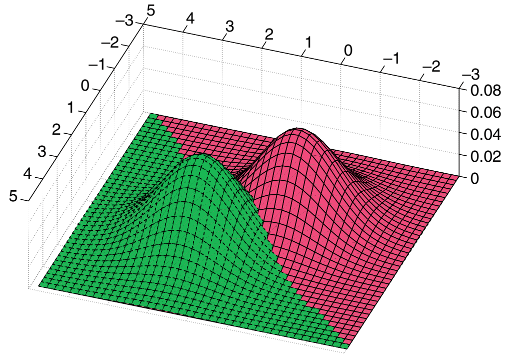
where κ specifies the number of degrees of freedom. In order to experiment with long tailed densities, we have set $\kappa = 4$. Besides, in contrast to the symmetry of the Gaussian density, pseudo-chi square has a skewness of $\sqrt{8/\kappa}$. The bivariate version of pseudo-chi square is defined as $f_{\text{chi}}(x, y; (x_0, y_0), \kappa) = f_{\text{chi}}(x; x_0, \kappa) \cdot f_{\text{chi}}(y; y_0, \kappa)$.

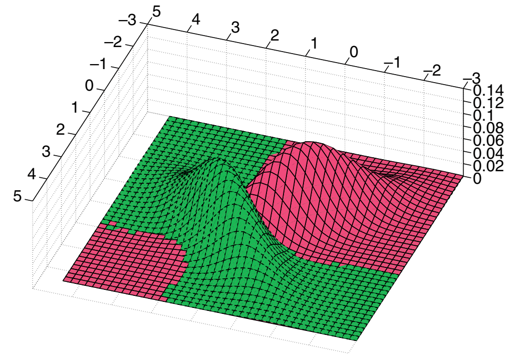
We have created by simulation the training and test sets for each of the mentioned domains. Each simulated dataset is defined by means of its associated generalized probability function $\rho(x, y, c) = p(c)f(x, y|c)$. In order to obtain a domain with dependent variables given the class, we use a function $f(x, y|c)$ which can not be decomposed into $f(x|c)f(y|c)$. The NCG, CG, NCGM, CGM, NCCh and CChM domains are defined by the following densities $f(x, y|c)$:

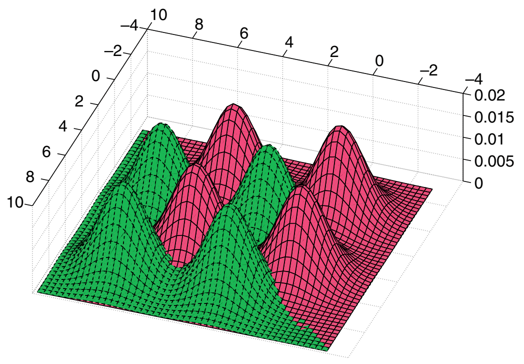
- The NCG dataset is generated from the density functions defined by $f(x, y|c) = f(x|c)f(y|c)$ where $f(x|C = 0) \equiv f(y|0) \sim N(\mu_0 = 0, \sigma_0^2 = 1)$ and $f(x|1) \equiv f(y|1) \sim N(\lambda, 1)$ with $\lambda = 1.8$ (see Fig. 4a).
- The CG dataset is defined by the density functions $f(x, y|0) \sim N(\boldsymbol{\mu_0}) = (0, 0), \boldsymbol{\Sigma_0} = [1, 0.75; 0.75, 1]$ and $f(x, y|1) \sim N((\lambda, \lambda), [1, -0.75; -0.75, 1])$ with $\lambda = 1.4$ (see Fig. 4b).
- The NCGM dataset is defined by the mixture functions $f(x|0) \equiv f(y|0) \sim 0.5N(0, 1) + 0.5N(\lambda, 1)$ and $f(x|1) \equiv f(y|1) \sim 0.5N(\lambda/2, 1) + 0.5N(3\lambda/2, 1)$ with $\lambda = 4.7$ (see Fig. 4c).
- The CGM dataset is defined by the mixture functions $f(x, y|0) \sim 1/3(N((0, 0), I = [1, 0; 0, 1]) + N((\lambda, \lambda), I) + N((0, 2\lambda), I))$ and $f(x, y|1) \sim 1/3(N((\lambda, 0), I) + N((0, \lambda), I) + N((\lambda, 2\lambda), I))$ with $\lambda = 3.4$ (see Fig. 4d).
- The NCCh dataset is generated from the density functions defined by $f(x, y|0) = f_{\text{chi}}(x, y; (x_0, y_0) = (0, 0), \kappa = 4)$ and $f(x, y|1) = f_{\text{chi}}(x, y; (\lambda, \lambda), 4)$ with $\lambda = 3.5$ (see Fig. 4a).
- The CChM dataset is defined by the mixture functions $f(x, y|0) = 1/3[f_{\text{chi}}(x, y; (0, 0), 4) + f_{\text{chi}}(x, y; (\lambda, \lambda), 4) + f_{\text{chi}}(x, y; (0, 2\lambda), 4)]$ and $f(x, y|1) = 1/3[f_{\text{chi}}(x, y; (0, \lambda), 4) + f_{\text{chi}}(x, y; (\lambda, 0), 4) + f_{\text{chi}}(x, y; (\lambda, 2\lambda), 4)]$ with $\lambda = 7.9$ (see Fig. 4d).

It must be noted that the NCCh and CChM domains are similar to the NCG and CGM (see Fig. 4), replacing Gaussian by pseudo-chi square densities.

In order to study the errors of different classifiers (mNB, mTAN, gNB, gTAN, fNB and fTAN) with different training set sizes, at each artificial domain proposed, we have sampled training sets with 10, 20, 40, 80, 160, 320, 640, 1280 and 2560 cases. The experiments have been repeated 50 times for estimating the classification errors for each training set size. All the classifiers learned have been tested on an independent dataset of 3000 cases. The variables have been discretized [27] for each training-test pair, learning the discretization policies from the training set. The evolution of the errors estimated, with each classifier learned, is presented in Fig. 5.


(a) Non-correlated, Gaussian domain.


(b) Correlated, Gaussian domain.


(c) Non-correlated, Gaussian mixture domain.

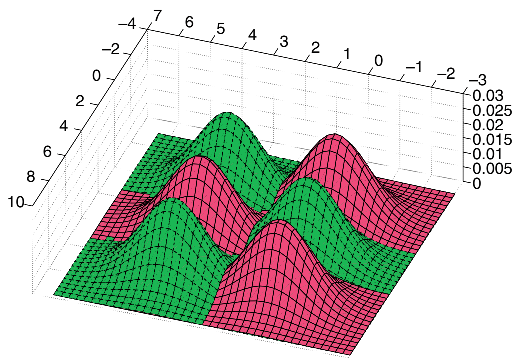
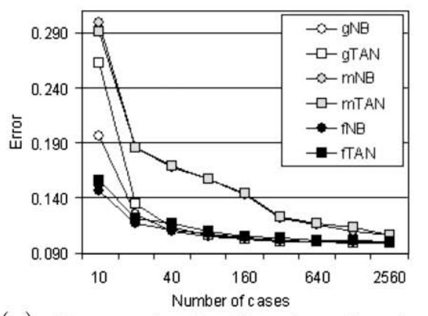

(d) Correlated, Gaussian mixture domain.

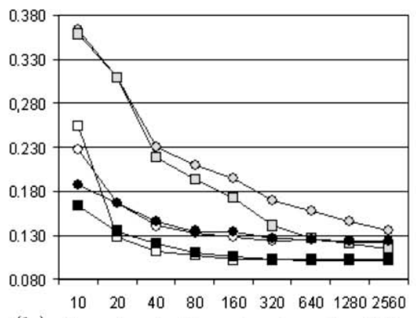
Fig. 4. Visualization of the artificial domains proposed. At each artificial domain $f(x, y, c = 0)$ is represented with broken lines and a lighter surface, and $f(x, y, c = 1)$ with continuous solid lines and a darker surface.

From the results shown in Fig. 5, the following conclusions can be obtained:

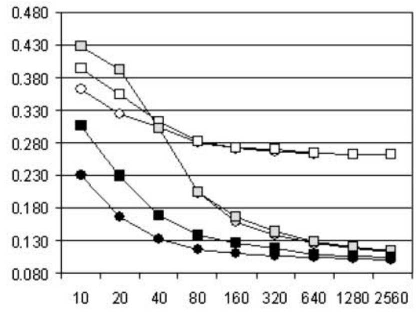
- In the NCG domain (see Fig. 4a) all the classifiers reach the Bayes error. Gaussian and flexible classifiers behave similarly (see Fig. 5a). They reach the Bayes error with training sizes of 160 or greater. This suggests that NCG domains can be modeled similarly for classifying using the kernel density estimation and the parametric Gaussian approach. The multinomial classifiers show a slower learning curve.
- In the CG domain (see Fig. 4b) the classifiers which do not model the correlation between predictors (gNB, mNB and fNB) seem to behave somewhat worse than gTAN and fTAN (see Fig. 5b). Gaussian and flexible TAN reach the Bayes error with training size of 320 or greater. By contrast, mTAN does not reach the Bayes error but behaves a little better than NB classifiers. These results suggest that CG domains could generally be better modeled by Gaussian and flexible classifiers which consider the correlation between predictor variables. Thus, gTAN and fTAN show a better behavior than gNB and fNB, respectively.
- In the NCGM domain (see Fig. 4c) the flexible classifiers behave notably better than Gaussian based classifiers (see Fig. 5c). Flexible and multinomial classifiers reach the Bayes error, but flexible classifiers converge to the Bayes error quicker. Flexible classifiers can clearly model the NCGM domains better than Gaussian classifiers.
- In the CGM domain (see Fig. 4d) flexible TAN classifier behaves clearly better than the rest of the classifiers (see Fig. 5d). This result suggests the importance of modeling the correlation between predictors using kernel based density estimation for the CGM domains. Multinomial TAN plus discretization converges to the same error value as NB models. Each variable is discretized independently, in an univariate way. Thus, the discretization algorithm [27] loses useful information about the class conditional dependencies between predictor variables. It must be noted that the most used discretization algorithms are univariate [24].
- The results in the NCCh domain are similar to those obtained in the NCGM. The flexible classifiers behave clearly better than Gaussian based classifiers (see Fig. 5e). Flexible and multinomial classifiers reach the Bayes error but multinomial classifiers converge to the Bayes error quicker.
- The results in the CChM domain are similar to those obtained in the CGM. In the CChM domain flexible TAN classifier behaves clearly better than the rest of the classifiers (see Fig. 5f). Again, these results suggest the importance of



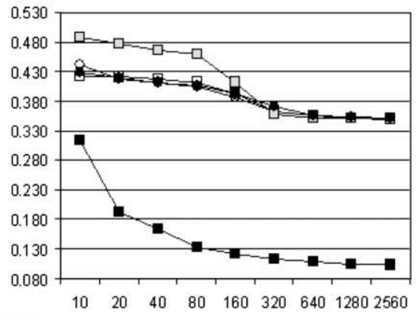
(a) Non-correlated, Gaussian domain, NCG.



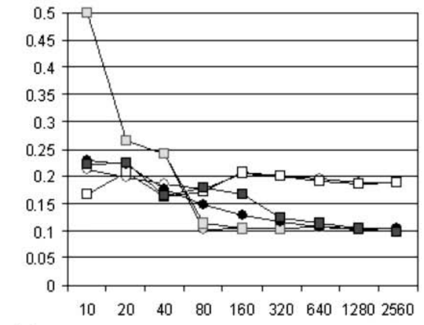
(b) Correlated, Gaussian domain, CG.



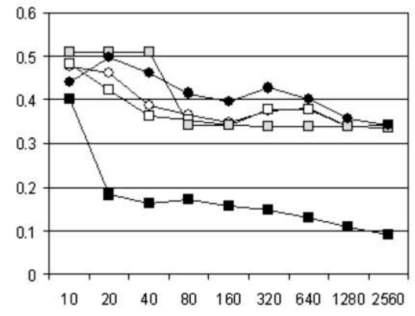
(c) Non-correlated, Gaussian mixture domain, NCGM.



(d) Correlated, Gaussian mixture domain, CGM.



(e) Non-correlated, chi square domain, NCCh.



(f) Correlated, chi square mixture domain, CChM

Fig. 5. The graphics represent the evolution, with respect to the number of cases, of the errors of different classifiers (mNB, mTAN, gNB, gTAN, fNB and fTAN) in each of the artificial domains proposed.

modeling the correlation between predictors with flexible classifiers. Multinomial TAN plus discretization converges to the same error value as NB models. The discretization algorithm [27] loses useful information once again, due to its univariate nature.

Taking into account the six continuous domains presented, at each structural complexity level, the flexible classifiers seem to be at least equally suitable as the Gaussian and multinomial classifiers (plus univariate supervised discretization [27]) for classifying in any continuous domain. Flexible classifiers which model the true dependencies between predictor variables reach the Bayes error in the artificial domains proposed.

In Gaussian domains, the flexible classifiers show a learning curve similar to the Gaussian classifiers. Multinomial classifiers have a slower learning curve than flexible classifiers at each type of the proposed domain (except in the NCCh domain). Flexible TAN, which models correlations between variables, obtains acceptable errors (less than 15%) with training set sizes of 20 in the NCG and CG domains, with 40 cases in the CGM and CGM domains, and with 160 cases in the NCCh and CChM domains. Besides, it reaches the Bayes error with less than 1280 training cases in most of the domains. This suggests that flexible classifiers which model the true dependencies between variables could perform well in multi-modal domains, even with small sample sizes.

5.1.2. Smoothing degree

This section illustrates the effect of the smoothing degree in the performance of the flexible classifiers. For this purpose, we have designed four representative univariate artificial domains.

- Gaussian domain (Ga).
- Gaussian mixture domain (GaM).
- Chi square domain (Ch).
- Chi square mixture domain (ChM).

It must be noted that all the presented flexible classifiers are equivalent in the univariate domains.

The training and test sets of each of the four univariate artificial domains have been created by simulation. Each simulated dataset is defined by means of its associated generalized probability function $\rho(x,c) = p(c)f(x|c)$. The domains have two equally probable classes, $c \in \{0,1\}$ with $p(C=0) = 0.5$. They are defined by the following densities $f(x|c)$:

- Ga domain: $f(x|C=0) \sim N(\mu = 0, \sigma^2 = 1)$ and $f(x|1) \sim N(\lambda, 1)$ with $\lambda = 2.5$.
- GaM domain: $f(x|0) \sim 1/2[N(0,1) + N(2\lambda, 1)]$ and $f(x|1) \sim 1/2[N(\lambda,1) + N(3\lambda,1)]$ with $\lambda = 3.0$.
- Ch domain: $f(x|0) = 1/2[f_{\text{chi}}(x; x_0 = 0, \kappa = 4)$ and $f(x|1) = f_{\text{chi}}(x;\lambda,4)$ with $\lambda = 5.8$.
- ChM domain: $f(x|0) = 1/2[f_{\text{chi}}(x;0,4) + f_{\text{chi}}(x;2\lambda,4)]$ and $f(x|1) = f_{\text{chi}}(x;\lambda,4) + f_{\text{chi}}(x;3\lambda,4)]$ with $\lambda = 6.8$.

In order to study the errors of flexible classifiers with different training set sizes, at each artificial domain proposed, we have sampled training sets with 10, 25, 100 and 250 cases. The experiments have been repeated 50 times to estimate the errors for each training set size. We have generated 50 training sets for each train size and 50 test sets with 3000 cases. The variables have been discretized independently for each training-test pair. The discretization policies and the classifiers have been learned using the training set.

At each experiment we compute the smoothing parameter h using the normal rule plus differential scaled (see Section 2.1). Note that the variance has been obtained using the maximum likelihood estimator. Then, the bandwidth matrix $\boldsymbol{H} = h^2$ is scaled using the following 18 coefficients, {0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 2.5, 5, 7.5, 10, 25, 75, 100}. For each scaled value of h^2 the classification error is estimated. Note that this scaling process can be understood as a wrapper optimization of the smoothing parameter, h.

The evolution of the estimated classification errors with respect to the smoothing degree are presented in Fig. 6.

The standard deviation, σ, is the most common measure of statistical dispersion, measuring how widely spread the values in a dataset are. The standard deviation is very sensitive to outliers. Note that the data obtained from long tailed densities

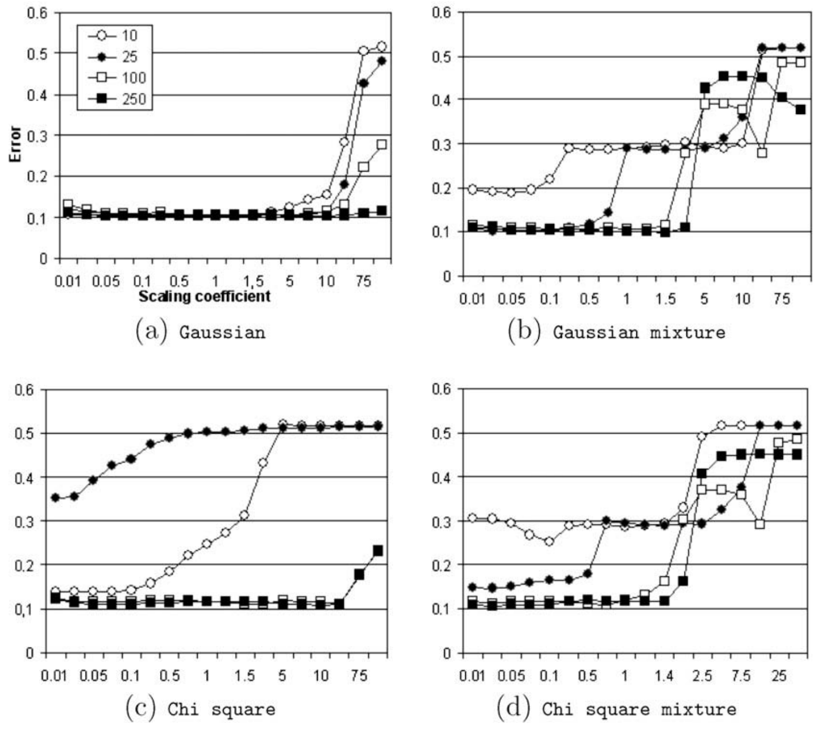


Fig. 6. The effect of the smoothing degree in the performance of the flexible classifiers.

generally presents outliers, e.g. chi square with $\kappa = 4$. This sensitivity is increased when the training set has few cases. It also increases when the density presents more than one mode, such as the GaM and ChM domains. Under these conditions the deviation overestimates the spread of the data. We say that standard deviation is overestimating the spread if the interval $[\mu - \sigma, \mu + \sigma]$ contains more than 90% of the cases.

The normal rule tends to oversmooth the estimations. Taking into account that the standard deviation tends to overestimate the spread of the data, the effect of the oversmoothing of the normal rule is increased. Next, the results presented in Fig. 5 are analyzed, taking into account the properties of the standard deviation and normal rule:

- In the Ga domain (see Fig. 6a) the standard deviation obtains a good measure of the spread of the data. In this case the normal rule obtains good estimations for classifying.
- In the GaM domain (see Fig. 6b) the deviation is overestimating the spread, especially with small training sets (10, 25). Given enough cases (more than 25), the normal rule obtains good estimations for classifying.
- In the Ch domain (see Fig. 6c) the deviation is overestimating the spread, especially with small training sets (10, 25). The normal rule obtains good estimations when the training set has enough cases (more than 25).
- In the ChM domain (see Fig. 6d) the deviation overestimates the spread of the data, especially with small training sets. Besides, the normal rule increases the oversmoothing effect. In spite of the oversmoothing, the normal rule obtains good approximations for classifying when the training set has enough cases (more than 25).

These results suggest that it could be useful to compute a spread measure more robust to outliers than the standard deviation, e.g. the interquartile range. The normal rule tends to obtain enough good density estimations for classifying if the deviation does not overestimate the spread of the training data. Besides, flexible classifiers seem to have a good classification behavior even when the optimum smoothing degree is not used. Moreover, they seem to be quite insensitive to the optimum smoothing parameter, especially when the training set has enough cases (more than 25 in the univariate case). Note that, for training sizes of 100 and 250, Fig. 6 presents error curves close to the Bayes error (0.1) with different scaling coefficient values: in Ga domain the error curves are close to the Bayes error for values in [0.01, 50] (see Fig. 6a), in GaM domain for values in [0.01, 2.5] (see Fig. 6b), in Ch domain for values in [0.01, 50] (see Fig. 6c) and in ChM domain for values in [0.01, 2.5] (see Fig. 6d).

On the other hand, Fig. 6 suggests that the performance of the flexible classifiers could be improved when the optimization of the smoothing degree is performed in a wrapper way, especially for small databases.

5.2. UCI datasets

This section is divided in three main parts. In Section 5.2.2 the classification error (see Eq. (11)) of the presented algorithms is estimated for each dataset. Moreover, the results of some other classifiers based on Bayesian networks (multinomial and Gaussian NB and TAN), k-nearest neighbour with $k = \{1, 3\}$, ID3 and C4.5 classification trees, quadratic discriminant analysis and multilayer perceptron are presented. Then, based on these results, flexible classifiers and the benchmarks included are compared across the UCI datasets by means of Friedman plus Nemenyi post hoc tests, as proposed in [19]. Additionally, in Section 5.2.3 the effect of the smoothing degree in classification performance is analyzed. Finally, in Section 5.2.4, and in order to study the nature of the error of the flexible classifiers, Kohavi and Wolpert's bias plus variance decomposition of the expected error [46] is performed in some of the UCI datasets included.

5.2.1. Main features and preprocessing

The results have been obtained in 21 *UCI repository* datasets [59], which only contain continuous predictor variables without missing values. Taking into account that BMN, CGN and KBN based classifiers handle discrete variables assuming that they are sampled from a multinomial distribution, we have decided to include only continuous domains to highlight the differences between each other. We have also decided to include only continuous domains because the most complex models, such as fCG, are prohibitive for mixed domains with many discrete variables. The main characteristics of the datasets included are summarized in Table 2.

In order to interpret the results we must take into account that most datasets of the UCI repository are already preprocessed [43]: in the datasets included, there are few irrelevant or redundant variables, and little noise [79]. It is more difficult to obtain statistically significant differences between the results of the algorithms in these types of datasets [79].

Some of the included datasets have a high-dimensional feature space (see Table 2). We have decided to reduce their dimensionality by a preprocess stage to avoid the computational problems related with the flexible classifiers presented (especially for computing the conditioned mutual information among each pair of predictors, $I(X_i; X_j|C)$).

The preprocess stage is performed for each training-test pair separately. In order to estimate the error (Sections 5.2.2 and 5.2.3) at each data set, 10 training-test pairs are generated using 10-fold cross validation procedure. In the bias plus variance decomposition (Section 5.2.4), 10 training-test pairs are generated ($N = 10$). Firstly, in order to reduce the computational requirements of the algorithms proposed (especially for fTAN and fkDB) we have performed a feature subset selection. If the number of predictors d is greater than 50, the dimensionality is reduced to 50 predictor variables. With this purpose, the mutual information with the class, $I(X_i; C)$, is computed for each variable, X_i, using the training set. Then, variables are sorted in descendant order of $I(X_i; C)$ and the first 50 variables are selected. So as to perform the experimentation with

Table 2
Main characteristics of the datasets: the number of different values of the class variable (r), the number of predictor variables (d), and the number of instances (n)

Label	r	d	n	Name of the dataset
`Balance`	3	4	625	Balance scale weight and distance
`Block`	5	10	5474	Page Block's classification
`Haberman`	2	3	307	Haberman's survival
`Image`	7	18	2310	Image segmentation
`Ionosphere`	2	34	351	Johns Hopkins University Ionosphere
`Iris`	3	4	150	Iris plant
`Letter`	26	16	20,000	Letter image recognition
`Liver`	2	6	345	Bupa liver disorders
`MFeatures`	10	649	2000	Multiple feature digit
`Musk`	2	166	6598	Musk Clean2
`Pendigits`	10	16	10,992	Pen-digit recognition of handwritten digits
`Pima`	2	8	768	Pima Indians diabetes
`Satellite`	6	36	6435	Landsat satellite
`Sonar`	2	60	208	Sonar, mines vs. rocks
`Spambase`	2	57	4601	Spam e-mail database attributes
`Thyroid`	3	5	215	Thyroid disease records
`Vehicle`	4	19	846	Vehicle Silhouetes
`Vowel`	11	10	990	Vowel recognition
`Waveform`	3	21	5000	Waveform data generation
`Wine`	3	13	179	Wine recognition
`Yeast`	10	9	1484	Yeast

the discrete classifiers, training and testing sets are discretized with the Fayyad and Irani entropy based method [27], using the discretization policy learned in the training set.

5.2.2. Classification error estimation

The classification error has been estimated for the following flexible classifiers: flexible NB (fNB), flexible TAN (fTAN), flexible kDB for $k = 2, d/2$ (f2DB, fd.5DB) and flexible complete graph (fCG).

The benchmark classifiers selected are: Gaussian NB and TAN [62] (gNB and gTAN), multinomial NB and TAN [30] (mNB and mTAN), k-nearest neighbour with $k = 1, 3$ [16] (k-NN) as lazy classifiers, ID3 [66] and C4.5 [67] as classification trees, and quadratic discriminant analysis (QDA) and multilayer perceptron (MP) [72] as discriminant functions. All of them, except gTAN [62] and mTAN [30], are implemented in Weka 3.4.3 [81].

Table 3
The estimated errors obtained with a set of well known state-of-the-art algorithms

Dataset	k-NN		Classification trees		Discriminant func.		CGN		BMN	
	1-NN	3-NN	ID3	C4.5	QDA	MP	gNB	gTAN	mNB	mTAN
`Balance`	14.9 ± 3.8	14.7 ± 3.8	29.8 ± 4.6	23.0 ± 4.7	**8.3 ±2.8**	9.0 ± 1.9	9.3 ± 1.0	11.5 ± 1.8	27.7 ± 4.8	28.2 ± 4.3
`Block`	3.8 ± 0.8	3.8 ± 0.8	3.9 ± 0.5	**3.1 ±0.5**	6.2 ± 0.7	3.8 ± 0.7	10.1 ± 2.4	7.5 ± 0.8	6.6 ± 1.4	4.4 ± 0.5
`Haberman`	33.3 ± 3.8	32.0 ± 3.8	27.8 ± 3.8	28.4 ± 7.7	24.8 ± 4.3	26.5 ± 4.3	25.8 ± 4.9	**24.5 ±4.6**	27.8 ± 3.8	27.8 ± 3.8
`Image`	**3.1 ±1.1**	4.0 ± 1.2	7.5 ± 2.1	4.0 ± 1.5	19.6 ± 22.1	4.0 ± 1.3	21.1 ± 2.0	20.5 ± 8.5	11.3 ± 2.0	5.9 ± 1.7
`Ionosphere`	13.1 ± 4.5	14.8 ± 3.3	10.0 ± 6.2	11.7 ± 5.0	13.4 ± 4.3	11.4 ± 6.4	18.0 ± 6.3	**7.7 ±3.6**	10.5 ± 4.0	8.6 ± 4.1
`Iris`	4.7 ± 6.7	5.3 ± 7.2	6.0 ± 8.1	7.3 ± 8.1	**2.7 ±4.4**	3.3 ± 6.1	4.0 ± 6.8	3.3 ± 6.1	8.0 ± 7.8	7.3 ± 8.1
`Letter`	**3.9 ±0.5**	4.4 ± 0.5	20.9 ± 0.6	12.1 ± 0.6	11.5 ± 0.7	17.4 ± 1.2	35.9 ± 1.4	28.3 ± 1.1	26.0 ± 1.2	14.5 ± 0.4
`Liver`	39.1 ± 4.6	37.9 ± 6.2	41.4 ± 2.9	33.3 ± 6.7	39.7 ± 5.7	**32.2 ±7.1**	43.7 ± 7.8	39.7 ± 5.6	41.4 ± 2.9	41.4 ± 2.9
`MFeatures`	20.1 ± 1.7	18.1 ± 1.4	31.7 ± 2.0	23.6 ± 3.9	18.2 ± 2.3	**16.9 ±2.5**	23.3 ± 2.7	19.7 ± 1.6	22.6 ± 2.5	20.7 ± 3.0
`Musk`	4.7 ± 0.9	3.9 ± 0.7	6.6 ± 1.3	4.1 ± 0.7	13.7 ± 1.2	**2.6 ±0.6**	20.1 ± 1.8	23.8 ± 1.4	8.8 ± 1.1	6.0 ± 1.2
`Pendigit`	**0.7 ±0.3**	**0.7 ±0.2**	13.0 ± 1.3	3.7 ± 0.4	2.2 ± 0.5	6.2 ± 0.5	15.0 ± 1.1	7.3 ± 1.2	12.9 ± 0.9	4.5 ± 0.5
`Pima`	29.3 ± 4.0	27.2 ± 4.8	27.0 ± 5.4	24.9 ± 4.8	25.6 ± 3.7	25.0 ± 3.6	25.0 ± 3.3	24.6 ± 3.6	23.7 ± 3.7	**23.0 ±3.3**
`Satellite`	9.6 ± 0.9	**9.4 ±1.1**	18.9 ± 1.4	13.1 ± 1.4	14.4 ± 0.8	10.4 ± 1.0	20.5 ± 0.9	14.5 ± 1.6	17.9 ± 1.1	11.6 ± 1.4
`Sonar`	**13.0 ±4.8**	16.4 ± 6.9	23.6 ± 7.3	26.0 ± 9.0	23.5 ± 6.7	13.9 ± 6.8	31.7 ± 7.0	31.7 ± 9.7	22.5 ± 9.2	22.5 ± 11.2
`Spambase`	8.9 ± 1.4	9.8 ± 1.4	9.5 ± 1.3	**7.1 ±1.5**	19.3 ± 6.9	7.9 ± 1.3	18.0 ± 1.5	17.2 ± 1.5	10.3 ± 0.9	7.2 ± 0.8
`Thyroid`	3.3 ± 3.7	5.6 ± 5.4	8.4 ± 4.6	6.9 ± 4.6	**3.2 ±4.6**	3.3 ± 3.0	4.2 ± 4.3	**3.2 ±4.6**	5.0 ± 5.2	5.0 ± 5.5
`Vehicle`	30.4 ± 3.4	29.2 ± 2.5	32.7 ± 3.1	30.2 ± 4.5	40.2 ± 3.1	40.5 ± 4.7	54.3 ± 4.7	23.3 ± 4.0	30.6 ± 3.1	**20.0 ±5.3**
`Vowel`	**0.9 ±0.8**	3.1 ± 1.8	22.7 ± 4.6	20.4 ± 4.0	11.3 ± 3.0	20.5 ± 4.7	32.6 ± 2.9	22.7 ± 3.6	35.5 ± 4.4	28.1 ± 4.0
`Waveform`	22.5 ± 2.2	19.5 ± 1.5	30.3 ± 2.3	22.8 ± 2.0	**15.1 ±1.8**	15.5 ± 1.7	19.0 ± 0.7	17.6 ± 2.1	18.8 ± 0.8	18.0 ± 2.6
`Wine`	5.7 ± 4.4	4.0 ± 4.5	6.2 ± 6.9	6.1 ± 5.2	**0.6 ±1.7**	2.8 ± 4.5	2.8 ± 3.8	**0.6 ±1.8**	1.7 ± 2.6	4.5 ± 4.9
`Yeast`	48.1 ± 3.8	47.2 ± 3.0	43.5 ± 4.4	44.1 ± 3.7	42.9 ± 3.8	**40.6 ±3.0**	42.3 ± 5.0	42.9 ± 4.8	43.7 ± 4.0	43.2 ± 4.1
Average	14.9	**14.8**	20.1	16.9	17.0	14.9	22.7	18.7	19.7	16.8

The best results, in each dataset, are marked in bold.

Table 4
The estimated errors and their corresponding standard deviations obtained with the presented flexible classifiers: flexible naive Bayes (fNB), flexible tree-augmented naive Bayes (fTAN), flexible k-dependence Bayesian classifier with $k = 2, d/2$ (f2DB, fd.5DB) and flexible complete graph classifier (fCG)

Dataset	fNB	fTAN	f2DB	fd.5DB	fCG
Balance	8.3 ± 1.2	10.7 ± 1.9	12.8 ± 2.6	12.8 ± 9.9	**1.2 ±0.7**
Block	6.1 ± 0.6	**5.2 ±0.7**	5.8 ± 0.9	6.3 ± 0.7	6.2 ± 0.7
Haberman	25.5 ± 5.9	**24.8 ±4.3**	26.4 ± 4.7	**24.8 ±4.3**	26.4 ± 4.7
Image	18.7 ± 1.4	**15.7 ±1.6**	16.5 ± 1.4	18.4 ± 1.6	17.1 ± 1.7
Ionosphere	9.1 ± 4.6	**7.1 ±2.6**	**7.1 ±3.2**	10.0 ± 4.5	10.5 ± 5.0
Iris	4.0 ± 6.8	4.7 ± 6.7	4.7 ± 6.7	4.7 ± 6.7	**3.3 ±6.1**
Letter	28.7 ± 1.2	15.7 ± 0.5	11.4 ± 0.7	7.7 ± 0.8	**7.1 ±0.8**
Liver	**33.6 ±7.9**	37.4 ± 8.4	37.4 ± 8.3	38.2 ± 6.7	40.8 ± 6.6
MFourier	22.5 ± 3.4	**18.5 ±2.2**	23.1 ± 2.6	18.6 ± 2.3	19.0 ± 2.3
Musk	**13.4 ±1.7**	10.7 ± 1.6	18.3 ± 1.8	41.5 ± 1.7	40.0 ± 1.5
Pendigit	14.1 ± 1.0	4.3 ± 0.4	**3.1 ±0.5**	**3.1 ±0.4**	**3.1 ±0.3**
Pima	24.3 ± 4.3	**23.1 ±4.3**	25.4 ± 3.3	24.9 ± 4.7	25.8 ± 3.8
Satellite	18.3 ± 0.9	**12.0 ±0.9**	12.2 ± 0.7	12.6 ± 0.6	12.2 ± 0.8
Sonar	26.9 ± 8.1	19.7 ± 5.5	24.5 ± 7.1	**15.4 ±5.5**	16.8 ± 7.7
Spambase	22.8 ± 1.4	**20.3 ±1.6**	25.1 ± 2.6	21.6 ± 1.9	20.9 ± 1.8
Thyroid	**3.2 ±4.6**	4.2 ± 5.6	4.2 ± 5.6	4.2 ± 5.6	4.2 ± 5.6
Vehicle	29.7 ± 3.4	**31.1 ±3.6**	33.6 ± 2.9	34.6 ± 3.8	35.6 ± 3.4
Vowel	26.0 ± 4.1	10.6 ± 3.9	5.1 ± 2.7	**1.9 ±1.0**	**1.9 ±1.5**
Waveform	19.3 ± 0.7	**18.4 ±1.7**	19.6 ± 1.8	22.3 ± 1.9	20.5 ± 2.0
Wine	2.3 ± 3.7	**1.1 ±2.3**	**1.1 ±2.3**	1.7 ± 2.6	1.7 ± 2.5
Yeast	**40.8 ±3.9**	41.0 ± 3.9	42.2 ± 3.1	42.5 ± 3.6	43.9 ± 3.4
Average	18.9	**16.0**	17.1	17.5	17.5

The best results, in each dataset, are marked in bold.

In order to estimate the classification error, for each flexible classifier and benchmark, at each dataset, a stratified 10-fold cross-validation process has been performed. Stratified cross-validation obtains estimations with less variance [21,42,81]. The validation was performed in a paired way: the classifiers are trained and tested in the same train-test pairs. The estimated classification errors of the benchmarks and the flexible classifiers are summarized in Tables 3 and 4, respectively.

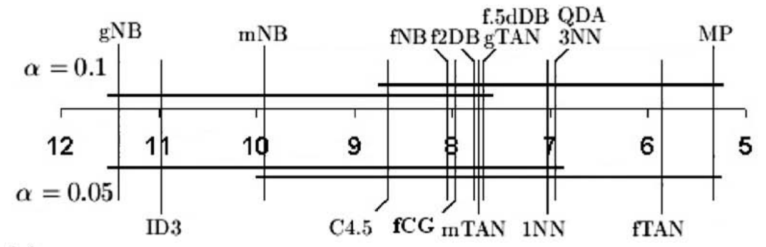
Using estimated errors in Tables 3 and 4, we compare the behavior of the flexible classifiers by means of the Friedman plus Nemenyi post hoc test, as proposed by Demšar in [19]. The Friedman test is a non-parametric equivalent of the repeated-measures ANOVA. It is used for comparing more than two algorithms over multiple datasets at the same time, based on average ranks. The null hypothesis being tested is that all classifiers obtain the same error. If the null hypothesis is rejected, it can be concluded that there are statistically significant differences between the classifiers, and then, the Nemenyi post hoc test is performed for comparing all classifiers with each other. It is equivalent to the Tukey test for ANOVA. The results for the Nemenyi post hoc test are shown in the critical difference diagrams [19]. These plots show the mean ranks of each model across all the domains in a numbered line. If there are not statistically significant differences between two classifiers, they are connected in the diagram by a straight line. For example, in Fig. 7c at $\alpha = 0.1$ level fTAN is significantly better than fCG. On the other hand, at $\alpha = 0.05$ the differences are not significant. A set of classifiers conform a cluster when they are connected in the diagram with the same line. Two clusters are significantly different (disjoint) when they are not connected.

Each test has been done at two significance levels $\alpha = \{0.1, 0.05\}$ as Demšar suggests in [19]. The studies proposed are:

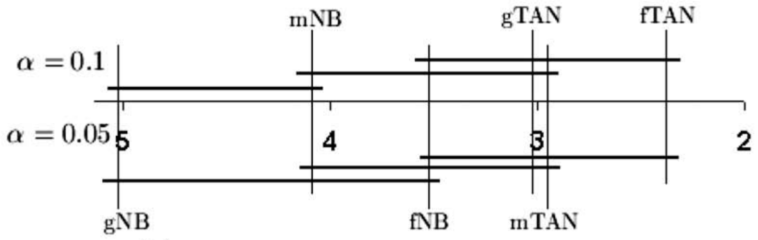
- Comparing the benchmarks and the flexible classifiers.
- Comparing the families of multinomial, Gaussian and flexible classifiers by means of the mNB, mTAN, gNB, gTAN, fNB and fTAN.
- Comparing the family of the flexible classifiers proposed.

The Friedman test for all the classifiers (benchmarks and flexible classifiers) rejects the null hypothesis at $\alpha = 0.05$. Thus, their errors can not be considered equivalent. The results for the Nemenyi post hoc test and the average ranks are shown in the critical difference diagrams [19] in Fig. 7a. The analysis is quite similar at $\alpha = \{0.1, 0.05\}$. No disjoint cluster of algorithms is obtained, but the good average rank obtained by fTAN classifier and MP, and the competitive results of the flexible classifiers should be highlighted.

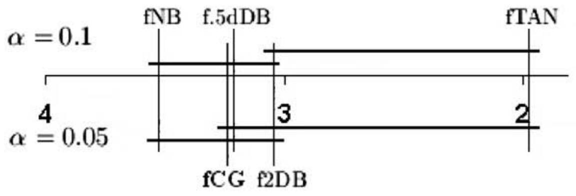
The comparison between the three BN based paradigms (the CMN, CGN and KBN paradigms) using mNB, mTAN, gNB, gTAN, fNB and fTAN classifiers seems to show two types of behaviors: NB and TAN classifiers. The Friedman test rejects the null hypothesis at $\alpha = \{0.1, 0.05\}$. The results for the Nemenyi test and the average ranks are shown in Fig. 7b. The analyses are similar at $\alpha = \{0.1, 0.05\}$. The different behavior of NB and TAN classifier families, especially at $\alpha = 0.1$, should be noted. Flexible classifiers, at each complexity level, seem to perform better than Gaussian and multinomial classifiers in terms of the average rank. However, the differences are not statistically significant using the Nemenyi test at $\alpha = 0.1$. It should be noted that using the Wilcoxon test, which has a larger discriminative power than Nemenyi test [19], the fTAN shows lower error levels than the mNB, mTAN, gNB, gTAN and fNB classifiers at $\alpha = 0.05$.



(a) Flexible classifiers and benchmarks. In order to simplify the diagram, we have decided to include only the clusters associated to the best (MP) and the worst (gNB) classifiers in terms of average ranks.



(b) gNB, gTAN, mNB, mTAN, fNB and fTAN.



(c) Flexible classifiers.

Fig. 7. Critical difference diagrams [19] representing Nemenyi post hoc test and average ranks for different sets of classifiers. The test has been performed at $\alpha = \{0.1, 0.05\}$ significance levels as Demšar suggests [19].

The last comparison has been performed between the flexible classifiers proposed. The Friedman test finds statistically significant differences in the overall behavior at $\alpha = \{0.1, 0.05\}$. The results of the Nemenyi test and the average ranks are shown in Fig. 7c. The fNB classifier seems to show the worst behavior, and fTAN the best one. The fTAN classifier obtains significant better errors than fNB at $\alpha = 0.05$ (see Fig. 7c). Moreover, fTAN obtains the highest number of best results across all datasets (see Table 4). The *curse of dimensionality* of the kernel density estimation [26,75,77] could explain the decrease of the performance of flexible classifiers while the structural complexity increases, such as from fTAN to fCG (see Table 4 and Fig. 7c).

5.2.3. Smoothing degree

This section illustrates the effect of the smoothing degree in the flexible classifiers by means of 21 UCI databases (see Table 2). The error estimation is performed in the same way as the experimentation of Section 5.2.2. In order to study the effect of the smoothing degree we have scaled the bandwidth matrix, which has been obtained with the normal rule plus differential scaled approach, using the following 15 coefficients $\{1.6^{-7}, \ldots, 1.6^{7}\}$. Note that this scaling process can be understood as a wrapper optimization of the smoothing parameter, h.

Fig. 8 summarizes the main results of this section. For simplicity, as fTAN and fNB are representative of the flexible classifiers presented, we only include the results for them.

In the worst case (see Fig. 8b), the differences between the error obtained with the normal rule plus differential scaled approach and the best smoothing degree are 8.6% and 9.0% for fNB and fTAN, respectively. This result suggests that, when the optimization of the smoothing degree is performed in a wrapper way, the performance of the flexible classifiers can be considerably improved in some domains (see Fig. 8b). In spite of this, on average, the difference with respect to the best smoothing degree is only 0.7% for both fNB and fTAN (see Fig. 8a). Besides, on average, the flexible classifiers obtain good results with the scaling coefficient in $[1.6^{-6}, \ldots, 1.6^{1}]$. Therefore, it can be concluded that the normal rule plus differential scaled approach usually obtains good approximations for classifying. Coupled with this, we think that using a more robust spread measure than standard deviation the flexible classifiers could obtain better results in some problematic domains, such as `Image`.

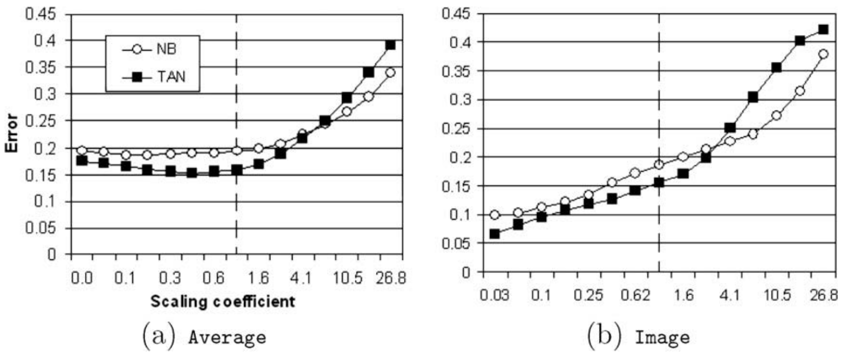


Fig. 8. The graphs show the classification errors obtained by fNB and fTAN classifiers. They illustrate the effect of the smoothing degree in the performance of the flexible classifiers. (a) Shows the average estimated errors obtained using 21 datasets (see Fig. 2); (b) shows the estimated errors obtained in Image dataset. The vertical dashed line indicates the normal rule plus differential scaled smoothing option.

5.2.4. Bias plus variance decomposition of the expected error

In this section, we perform the bias plus variance decomposition in order to study, in each dataset, the behavior of the expected error (expected misclassification rate [46]) of the flexible classifiers presented. We define the expected error of zero-one-loss function as:

$$E_{\mathrm{M}} = \int f(\boldsymbol{x}) \sum_{c=1}^{r} p(c|\boldsymbol{x})(1 - p_{\mathrm{M}}(c|\boldsymbol{x}))\mathrm{d}\boldsymbol{x} \tag{13}$$

where $p_{\mathrm{M}}(\cdot)$ is the estimation modeled by a classifier and $p(\cdot)$ is the true distribution of the domain. The bias-variance decomposition of the expected error can be useful to explain the behaviors of different algorithms [79]. The concept of bias-variance decomposition was introduced to machine learning for mean squared error by German et al. [33]. Later versions for zero-one-loss functions were given by Friedman [29], Kohavi and Wolpert [46], Domingos [22] and James [39]. The bias-variance decomposition of the expected error proposed in [46] is as follows:

$$E_{\mathrm{M}} = \int f(\boldsymbol{x})(\sigma^2_{\boldsymbol{x}} + \mathrm{bias}^2_{\boldsymbol{x}} + \mathrm{var}_{\boldsymbol{x}})\mathrm{d}\boldsymbol{x} \tag{14}$$

where $\boldsymbol{x}$ is an instance of the test set, $\sigma^2_{\boldsymbol{x}}$ is the "intrinsic" target noise, $\mathrm{bias}^2_{\boldsymbol{x}}$ is the square bias and $\mathrm{var}_{\boldsymbol{x}}$ is the variance associated with instance $\boldsymbol{x}$. Each term in Eq. (14) is defined as:

$$\sigma^2_{\boldsymbol{x}} = \frac{1}{2}\left(1 - \sum_{c=1}^{r} p(c|\boldsymbol{x})^2\right) \quad \mathrm{bias}^2_{\boldsymbol{x}} = \frac{1}{2}\sum_{c=1}^{r}(p(c|\boldsymbol{x}) - p_{\mathrm{M}}(c|\boldsymbol{x}))^2$$
$$\mathrm{var}_{\boldsymbol{x}} = \frac{1}{2}\left(1 - \sum_{c=1}^{r} p_{\mathrm{M}}(c|\boldsymbol{x})^2\right) \tag{15}$$

The averaged squared bias (or bias term of the decomposition) is defined as $\mathrm{bias}^2 = \int f(\boldsymbol{x})\mathrm{bias}^2_{\boldsymbol{x}} d\boldsymbol{x}$, and the averaged variance (or variance term) is defined as $\mathrm{var} = \int f(\boldsymbol{x})\mathrm{var}_{\boldsymbol{x}}\mathrm{d}\boldsymbol{x}$. The target noise is related with the expected error of the Bayes classifier. Therefore, it is independent of the learning algorithm and it is characteristic of each domain. In practice, if there are two instances in the test set with the same configuration for the predictors and a different value for the class, the estimated "intrinsic" noise is positive, otherwise it is zero [46]. Thus, it is considered zero given the datasets selected, and therefore, the expected error of the datasets included will decompose into bias plus variance. The bias component can be seen as the error due to the incorrect fitness of the hypothesis density function (modeled by the classifier) to the target density function (the real density underlying the data). By contrast, the variance component measures the variability of the hypothesis function, which is independent of the target density function. It can be seen as the expected error of the learning algorithm due to its sensitivity to changes in the training set. From these concepts, we can hypothesize that bias and variance terms become lower and higher, respectively, as the number of parameters needed to model the classifier grows (as classifier complexity increases).

The decompositions have been performed following Kohavi and Wolpert's proposal [46] with parameters $N = 10$ and $m = n/3$, where N is the number of training sets, m is their size and n is the size of the original dataset (see Table 2). We have set $N = 10$ because the bias estimation is precise enough for this value (see Fig. 1 of [46]), and because of its acceptable computational cost. We have set $m = n/3$ to ensure a minimum training set size which could avoid overfitting problems. Kohavi and Wolpert [46] choose a set of databases with at least 500 instances in order to ensure accurate estimates of the error. In order to obtain more robust conclusions, we have fulfilled this constraint and the following datasets have been selected for the experimentation: `Block`, `Image`, `Letter`, `MFeatures`, `Musk`, `Pima`, `Satellite`, `Vehicle`, `Vowel`, `Waveform` and `Yeast` (see Table 2).

Following the experimentation proposed by Kohavi and Wolpert [46], the probability distributions $p(c|\boldsymbol{x})$ and $p_{\mathrm{M}}(c|\boldsymbol{x})$ are determined as follows:

$$p(c|\boldsymbol{x}) = \begin{cases} 1 & (c, \boldsymbol{x}) \in T \\ 0 & \text{otherwise} \end{cases} \quad p_{\mathrm{M}}(c|\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} p_{\mathrm{M}}(c|x, d_i)$$

where T is the test set which is common for all training sets, and $p_{\mathrm{M}}(c|x, d_i)$ is the probability distribution learned from d_i training set. It must be noted that the estimation of $p(c|\boldsymbol{x})$ assumes that the "intrinsic" noise is zero.

Table 5 shows the results of the decomposition obtained for each classifier in the selected datasets –at least 500 instances. It also includes an additional row which contains the averages for each classifier across all the datasets selected. For example fTAN obtains a bias^2 = 11.6 plus *var* = 18.1 decomposition for `Balance`, and an *average* decomposition across all the datasets of bias^2 = 13.3 plus *var* = 9.4. Taking into account the results of Table 5, the following four behaviors of the decompositions should be highlighted (see Fig. 9):

- The behavior of the decomposition with `Vehicle` dataset (and in a similar way for `Waveform` and `Satellite`) is as follows (see Fig. 9a):
  - Bias notably decreases in the jump from fNB to fTAN. On the other hand, fTAN, f2DB, f*d*.5DB and fCG obtain quite similar bias component values.
  - Variance slightly increases with the increase of the complexity.

Table 5
Bias plus variance decomposition of the expected misclassification error rate

Dataset	fNB	fTAN	f2DB	f*d*.5DB	fCG
`Balance`	13.3 + 20.3	11.6 + 18.1	10.8 + 16.3	10.8 + 16.3	9.8 + 14.7
`Block`	3.5 + 1.3	3.2 + 1.3	3.4 + 1.3	3.6 + 1.2	3.7 + 1.2
`Image`	15.8 + 3.1	14.1 + 2.4	14.3 + 3.0	14.9 + 3.7	14.6 + 3.5
`Letter`	20.5 + 13.4	12.1 + 10.2	8.4 + 9.6	5.7 + 8.5	5.6 + 8.3
`MFeatures`	15.6 + 6.5	11.7 + 6.9	11.5 + 7.5	10.9 + 6.9	11.3 + 6.5
`Musk`	12.3 + 1.4	9.8 + 3.6	14.8 + 6.6	28.8 + 6.9	27.6 + 6.5
`Pendigits`	11.7 + 3.0	4.0 + 1.6	2.7 + 1.1	2.5 + 1.0	2.7 + 1.0
`Pima`	16.6 + 11.5	16.4 + 12.6	16.4 + 12.9	16.6 + 12.3	17.3 + 12.0
`Satellite`	17.3 + 1.3	9.1 + 3.1	8.4 + 4.7	8.3 + 4.8	8.1 + 4.6
`Spambase`	19.1 + 2.8	19.0 + 2.8	19.0 + 2.8	19.0 + 2.8	19.0 + 2.8
`Vehicle`	29.3 + 13.5	20.6 + 14.6	20.3 + 15.5	21.0 + 17.0	21.2 + 16.8
`Vowel`	21.1 + 24.7	11.1 + 17.5	8.9 + 14.7	7.2 + 12.1	6.9 + 11.7
`Waveform`	17.3 + 3.7	12.5 + 9.5	12.4 + 10.4	13.1 + 11.8	13.7 + 10.3
`Yeast`	31.1 + 27.2	30.7 + 27.5	31.2 + 27.4	31.2 + 27.1	31.9 + 27.2
Average	17.5 + 9.4	13.3 + 9.4	13.0 + 9.6	13.8 + 9.4	13.8 + 9.1

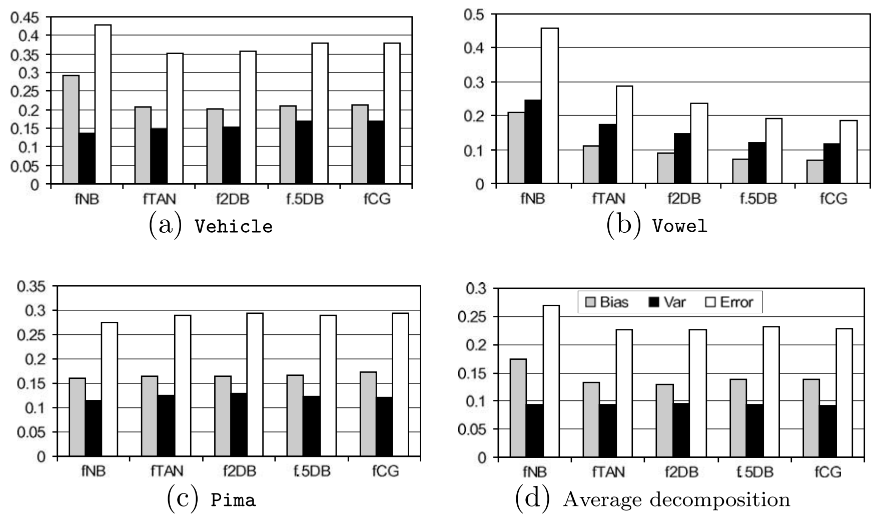


Fig. 9. The graphics represent the evolution of the bias plus variance decomposition of the expected error for different flexible classifiers ordered by their structural complexity. The graphics include the decompositions for three representative datasets (`Vehicle` (a), `Vowel` (b) and `Pima` (c)) and for the average over the datasets selected (d).

- The behavior of the decomposition with `Vowel` dataset (and in a similar way for `Balance`, `Letter and Pendigit`) is as follows (see Fig. 9b):
  - Bias decreases with the increase of the complexity.
  - Variance also decreases with the increase of the complexity.
- The behavior of the decomposition with `Pima` dataset (and in a similar way for `Block`, `Image`, `Spambase` and `Yeast`) is as follows (see Fig. 9c):
  - Bias and variance terms remains almost constant for all the flexible classifiers.

- Taking into account `Balance`, `Block`, `Image`, `Letter`, `MFeatures`, `Musk`, `Pendigits`, `Pima`, `Satellite`, `Spambase`, `Vehicle`, `Vowel`, `Waveform` and `Yeast` datasets, on average, the behavior of the decomposition is as follows (see Fig. 9d):
  - Bias notably decreases in the jump from fNB to fTAN. On the other hand, fTAN, f2DB, f*d*.5DB and fCG obtain quite similar bias components of the error. Therefore, it can be concluded that for the selected datasets fTAN model is precise enough for estimating the underlying density of the data.
  - Variance remains almost constant with the increase of the complexity, and thus, all the models seem to have a similar sensibility to changes in the training set.

In Musk dataset the bias component of the fNB is clearly lower than f*d*.5DB and fCG. The mutual information between predictors conditioned on the class reveals that some predictor variables are independent with respect to each other. Thus, the compulsory addition of arcs could be the reason for the increment of the bias term.

In order to understand the behavior of the variance term, the following two observations should be taken into account:

- The number of parameters (see Section 4.5) among the flexible classifiers presented is quite similar, and it is almost independent from their structural complexity (see Table 1 with $k = 0$ for fNB and $k = d - 1$ for fCG). Thus, from the parametric learning point of view, the sensibility to the changes in the training set (variance term) should be quite similar among flexible classifiers.
- We have observed in our experiments that given a factorization of the probability function $\rho(c, \boldsymbol{x})$, the probability mass tends to be more spread as fewer dependencies between variables are considered. For example, *fNB* tends to model a probability function more spread than *fCG*. Therefore, the classifiers, which model more dependencies between variables, tend to model an *a posteriori* distribution $p(c|\boldsymbol{x})$ with values nearest to 0 or 1 (*degenerated distributions*). If an instance $\boldsymbol{x}^{(i)}$ has a distribution more degenerated than another instance $\boldsymbol{x}^{(j)}$, then its variance should be lower, $\mathrm{var}_{\boldsymbol{x}^{(i)}} < \mathrm{var}_{\boldsymbol{x}^{(j)}}$ (see Eq. (15)).

The first observation explains the similar variance values for fNB, fTAN, f*k*DB and fCG classifiers in `Block`, `Image`, `Vehicle`, `MFeatures`, `Pima`, `Satimage` and `Spambase`. The first and the second observations could explain the slight decrease of the variance term as the structural complexity increases in `Vowel`, `Letter` and `Pendigits`.

Generally, among flexible classifiers, those which model correlations between predictor variables seem to be more suitable for modeling the underlying true densities of the variables, keeping at the same time the error due to the variance almost constant. Therefore, in absence of prior knowledge about the true density underlying the data, it may be advisable to use flexible classifiers which model correlations between predictors.

6. Conclusions and future work

This work presents the novel *kernel based Bayesian network* (KBN) paradigm for supervised classification. A kernel based Bayesian network is a Bayesian network which estimates the density of the continuous variables using kernel based estimators (Eq. (4)). Following the nomenclature introduced by John and Langley [41], we use the term *flexible classifiers* to name the family of classifiers based on KBN.

The KBN paradigm can be considered as an alternative to Bayesian multinomial networks because they can directly handle continuous variables without needing to discretize them. In addition, it can also be considered as an extension of the conditional Gaussian network from the point of view of the flexibility of their approaches for estimating densities, because kernel based Bayesian network uses a non-parametric kernel based density estimation instead of a parametric Gaussian one. In this paper we have demonstrated the strong consistency property of a flexible classifier for estimating $p(c|\boldsymbol{x})$ when its structure captures the true conditional dependencies between variables. Moreover, the kernel based estimation confers better flexibility properties than Gaussian approach in order to fit non-Gaussian densities. In summary, the factorizations codified by a KBN are better approximations to the true joint density and, therefore, the flexible classifiers obtain error rates closer to the Bayes error. This idea is illustrated in Section 5.1 by means of a set of four artificial domains. In the UCI datasets included, flexible classifiers seems to classify better than their Gaussian approaches and at least equal to their multinomial versions. Besides, they obtain competitive results compared to the state-of-the-art algorithms selected.

Among the flexible classifiers proposed, the novel *flexible tree-augmented naive Bayes* obtains the best ranking average and estimated errors. Besides, taking into account the bias plus variance decomposition of the expected error, it seems to model the true densities much better than the flexible naive Bayes, and thus, the error due to the bias component is lower. On the other hand, flexible classifiers which model the correlation between predictors obtain quite similar bias components of the error. Moreover, the error due to the variance remains almost constant among flexible classifiers and, therefore, all the models seem to have a similar sensibility to changes in the training set. In summary, among the flexible classifiers presented, in absence of prior knowledge about the true density underlaying the data, it is generally advisable to use the novel flexible tree-augmented naive Bayes for classifying.

One of the most important disadvantages of the kernel density estimation is related with the computational cost that is required to evaluate an instance. Our main future work line consists of relaxing the strong computational cost required. We have considered to approximate the kernel based estimation by means of semi-parametric approaches, such as, mixture of truncated exponentials [58,69], spline approximation [36,37] or Gaussian mixture models [55]. The Gaussian mixture models can be learned by means of the following approaches: EM algorithm [3,55], projection pursuit algorithm [1,2], iterative pairwise replacement algorithm [76] or M-Kernel merging procedure [83]. These semi-parametric approaches considerably reduce the computational cost, from a cost related with the number of training cases to a fixed one. These approaches will allow us to design and implement other search techniques in the space of possible structures: wrapper search [44] guided by the accuracy, or by using metaheuristics such as genetic algorithms [34] or estimation of distribution algorithms [49].

Acknowledgements

First, thanks to the helpful comments of the anonymous reviewers which have improved the quality of the paper. This work has been possible thanks to the Ph.D. grant *Beca para la Formación de Investigadores 2003-07* of the Basque Government. This work has been also supported by the Etortek, Saiotek and Research Groups 2007-2012 (IT-242-07) programs (Basque Goverment), TIN2005-03824 and Consolider Ingenio 2010-CSD2007-00018 projects (Spanish Ministry of Education and Science) and COMBIOMED network in computational biomedicine (Carlos III Health Institute).

References

[1] M. Aladjem, Projection pursuit fitting Gaussian mixture models, in: Proceedings of Joint IAPR, vol. 2396 of Lecture Notes in Computer Science, 2002, pp. 396–404.
[2] M. Aladjem, Projection pursuit mixture density estimation, IEEE Transactions on Signal Processing 53 (11) (2005) 4376–4383.
[3] J. Bilmes, A gentle tutorial on the EM algorithm and its application to parameter estimation for gaussian mixture models, Technical Report ICSI-TR-97-021, University of Berkeley, 1997.
[4] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.
[5] C.M. Bishop, Latent variable models, Learning in Graphical Models (1999) 371–403.
[6] C.M. Bishop, Pattern Recognition and Machine Learning, Information Science and Statistics, Springer, 2006.
[7] S.G. Bottcher, Learning Bayesian Networks with Mixed Variables, Ph.D. Thesis, Aalborg University, 2004.
[8] R. Bouckaert, Naive Bayes classifiers that perform well with continuous variables, In: Proceedings of the 17th Australian Conference on Artificial Intelligence, 2004, pp. 1089–1094.
[9] G. Casella, R.L. Berger, Statistical Inference, Wadsworth and Brooks, 1990.
[10] E. Castillo, J.M. Gutierrez, A.S. Hadi, Expert Systems and Probabilistic Network Models, Springer-Verlag, 1997.
[11] J. Cheng, R. Greiner, Comparing Bayesian network classifiers, in: Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, 1999, pp. 101–107.
[12] D.M. Chickering, Learning equivalence classes of Bayesian-network structures, Journal of Machine Learning Research 2 (2002) 445–498.
[13] C. Chow, C. Liu, Approximating discrete probability distributions with dependence trees, IEEE Transactions on Information Theory 14 (1968) 462–467.
[14] T.H. Cormen, L.E. Charles, R.L. Ronald, S. Clifford, Introductions to Algorithms, MIT Press, 2003.
[15] T.M. Cover, J.A. Thomas, Elements of Information Theory, John Wiley and Sons, 1991.
[16] T.T. Cover, P.E. Hart, Nearest neighbour pattern classification, IEEE Transactions on Information Theory 13 (1967) 21–27.
[17] M. DeGroot, Optimal Statistical Decisions, McGraw-Hill, New York, 1970.
[18] A. Delaigle, I. Gijbels, Comparison of data-driven bandwidth selection procedures in deconvolution kernel density estimation, Computational Statistics and Data Analysis 39 (2002) 1–20.
[19] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[20] L. Devroye, The equivalence in 11 of weak strong and complete convergence of kernel density estimates, Annals of Statistics 11 (1983) 896–904.
[21] N.A. Diamantidis, D. Karlis, E.A. Giakoumakis, Unsupervised stratification of cross-validation for accuracy estimation, Artificial Intelligence 116 (2000) 1–16.
[22] P. Domingos, A unified bias-variance decomposition and its applications, in: Proceedings of the 17th International Conference on Machine Learning, Morgan Kaufman, 2000, pp. 231–238.
[23] P. Domingos, M. Pazzani, On the optimality of the simple Bayesian classifier under zero-one loss, Machine Learning 29 (1997) 103–130.
[24] J. Dougherty, R. Kohavi, M. Sahami, Supervised and unsupervised discretization of continuous features, in: Proceedings of the 12th International Conference on Machine Learning, 1995, pp. 194–202.
[25] R. Duda, P. Hart, Pattern Classification and Scene Analysis, John Wiley and Sons, 1973.
[26] R. Duda, P. Hart, D. Stork, Pattern Classification, John Wiley and Sons, 2000.
[27] U. Fayyad, K. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, in: Proceedings of the 13th International Conference on Artificial Intelligence, 1993, pp. 1022–1027.
[28] M.A.T. Figueiredo, J.M.N. Leitão, A.K. Jain, On fitting mixture models, in: Energy Minimization Methods in Computer Vision and Pattern Recognition, vol. 1654 of Lecture Notes in Computer Science, 1999, pp. 732–749.
[29] J.H. Friedman, On bias, variance, 0/1 – loss, and the curse-of-dimensionality, Data Mining and Knowledge Discovery 1 (1997) 55–77.
[30] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Machine Learning 29 (1997) 131–163.
[31] K. Fukunaga, Statistical Pattern Recognition, Academic Press Inc., 1972.
[32] D. Geiger, D. Heckerman, Learning Gaussian networks. Technical Report, Microsoft Research, Advanced Technology Division, 1994.
[33] S. German, E. Bienenstock, R. Doursat, Neural networks and the bias-variance dilemma, Neural Computation 4 (1992) 1–58.

[34] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989.
[35] R. Greiner, W. Zhou, X. Su, B. Shen, Structural extension to logistic regression: discriminative parameter learning of belief net classifiers, Machine Learning 59 (3) (2005) 97–322.
[36] Y. Gurwicz, B. Lerner, Rapid spline-based kernel density estimation for Bayesian networks, in: Proceedings of the 17th International Conference on Pattern Recognition, vol. 3, 2004, pp. 700–703.
[37] Y. Gurwicz, B. Lerner, Bayesian network classification using spline-approximated kernel density estimation, Pattern Recognition Letters 26 (2005) 1761–1771.
[38] D.J. Hand, K. Yu, Idiot's Bayes – not so stupid after all?, International Statistical Review 69 (3) (2001) 385–398
[39] G.M. James, Variance and bias for general loss functions, Machine Learning 51 (2003) 115–135.
[40] T. Jebara, Machine Learning: Discriminative and Generative, Kluwer Academic Publishers, 2004.
[41] G.H. John, P. Langley, Estimating continuous distributions in Bayesian classifiers, in: Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence, 1995, pp. 338–345.
[42] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: International Joint Conference on Artificial Intelligence, vol. 14, 1995, pp. 1137–1145.
[43] R. Kohavi, Wrappers for Performance Enhancement and Oblivious Decision Graphs, Ph.D. Thesis, Stanford University, Computer Science Department, 1995.
[44] R. Kohavi, B. Becker, D. Sommerfield, Improving simple Bayes. Technical report, Data Mining and Visualization Group, Silicon Graphics, 1997.
[45] R. Kohavi, G. John, Wrappers for feature subset selection, Artificial Intelligence 97 (1–2) (1997) 273–324.
[46] R. Kohavi, D.H. Wolpert, Bias plus variance decomposition for zero-one loss functions, in: Lorenza Saitta (Ed.), International Conference on Machine Learning, Morgan Kaufman, 1996, pp. 275–283.
[47] I. Kononenko, Semi-naive Bayesian classifiers, in: Proceedings of the Sixth European Working Session on Learning, 1991, pp. 206–219.
[48] P. Langley, W. Iba, K. Thompson, An analysis of Bayesian classifiers, in: Proceedings of the 10th National Conference on Artificial Intelligence, 1992, pp. 223–228.
[49] P. Larrañaga, J.A. Lozano, Estimation of Distribution Algorithms, A New Tool for Evolutionary Computation, Kluwer Academic Publishers, 2002.
[50] S.L. Lauritzen, Graphical Models, Oxford University Press, 1996.
[51] S.L. Lauritzen, N. Wermuth. Mixed interaction models, Technical report r 84-8, Institute for Electronic Systems, Aalborg University, 1984.
[52] S.L. Lauritzen, N. Wermuth, Graphical models for associations between variables, some of which are qualitative and some quantitative, Annals of Statistics 17 (1989).
[53] B. Lerner, Bayesian fluorescence in situ hybridisation signal classification, Artificial Intelligence in Medicine 30 (3) (2004) 301–316.
[54] B. Lerner, N.D. Lawrence, A comparison of state-of-the-art classification techniques with application to cytogenetics, Neural Computing and Applications 10 (1) (2001) 39–47.
[55] G.J. McLachlan, D. Peel, Finite Mixture Models, Probability and Mathematical Statistics, John Wiley and Sons, 2000.
[56] M. Minsky, Steps toward artificial intelligence, Transactions on Institute of Radio Engineers 49 (1961) 8–30.
[57] Y. Moon, B. Rajagopalan, U. Lall, Estimation of mutual information using kernel density estimators, Physical Review 52 (3) (1995) 2318–2321.
[58] S. Moral, R. Rumí, A. Salmerón, Estimating mixtures of truncated exponential from data, in: First European Whorkshop on Probabilistic Graphical Models, 2002, pp. 156–167.
[59] P.M. Murphy, D.W. Aha, UCI repository of machine learning databases, Technical Report, University of California at Irvine, 1995, <http://www.ics.uci.edu/~mlearn>.
[60] R. Neapolitan, Learning Bayesian Networks, Prentice Hall, 2003.
[61] A. Pérez, P. Larrañaga, I. Inza, Information theory and classification error in probabilistic classifiers, in: Proceedings of the Ninth International Conference on Discovery Science, Lecture Notes in Artificial Intelligence, vol. 4265, 2006, pp. 347–351.
[62] A. Pérez, P. Larrañaga, I. Inza, Supervised classification with conditional Gaussian networks: increasing the structure complexity from naive Bayes, International Journal of Approximate Reasoning 43 (2006) 1–25.
[63] E. Parzen, On estimation of a probability density function and mode, Annals of Mathematical Statistics 33 (3) (1962) 1065–1076.
[64] M. Pazzani, Searching for dependencies in Bayesian classifiers, in: Learning from Data: Artificial Intelligence and Statistics V, 1997, pp. 239–248.
[65] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufman Publishers, 1988.
[66] J.R. Quinlan, Induction of decision trees, Machine Learning 1 (1986) 81–106.
[67] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman, 1993.
[68] S. Raudys, On the effectiveness of parzen window classifier, Informatica 2 (3) (1991) 434–453.
[69] V. Romero, R. Rumí, A. Salmeron, Learning hybrid Bayesian networks using mixture of truncate exponentials, International Journal of Approximate Reasoning 42 (2006) 54–68.
[70] T. Roos, H. Wettig, P. Grünwald, P. Myllymäki, H. Tirri, On discriminative Bayesian network classifiers and logistic regression, Machine Learning 59 (3) (2005) 267–296.
[71] F. Rosenblatt, Remarks on some nonparametric estimates of a density function, Annals of Mathematical Statistics 27 (1956) 832–837.
[72] F. Rosenblatt, Principles of Neurodynamics, Spartan Books, 1959.
[73] M. Sahami, Learning limited dependence Bayesian classifiers, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996, pp. 335–338.
[74] G. Santafé, J.A. Lozano, P. Larrañaga, Discriminative learning of Bayesian network classifiers via the TM algorithm, in: Proceedings of the Eighth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 2005, pp. 148–160.
[75] D.W. Scott, Multivariate Density Estimation: Theory, Practice and Visualization, John Wiley and Sons, 1992.
[76] D.W. Scott, W.F. Szewczyk, From kernels to mixtures, Technometrics 43 (3) (2001) 323–335.
[77] B.W. Silverman, Density Estimation for Statistics and Data Analysis, Chapman and Hall, London, 1986.
[78] J.S. Simonoff, Smoothing Methods in Statistics, Springer, 1996.
[79] P. van der Putten, M. van Someren, A bias-variance analysis of a real world learning problem: the CoIL challenge 2000, Machine Learning 57 (2004) 177–195.
[80] M.P. Wand, M.C. Jones, Kernel Smoothing, Monographs on Statistics and Applied Probability, Chapman and Hall, 1995.
[81] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufman, 2005.
[82] Y. Yang, G.I. Webb, Discretization for naive-Bayes learning: Managing discretization bias and variance, Technical Report 2003-131, School of Computer Science and Software Engineering, Monash University, 2003.
[83] A. Zhou, Z. Cai, L. Wei, M-kernel merging: towards density estimation over data streams, in: Proceedings of the Database Systems for Advanced Applications, 2003, pp. 285–292.