Bayesian network classifiers which perform well with continuous attributes: Flexible classifiers.

Aritz Pérez
Intelligent Systems Group
Dept. de C.C.I.A.
Universidad del País Vasco
aritz@si.ehu.es

Abstract
When modelling a probability distribution with a Bayesian network, we are faced with the problem of how to handle continuous variables. Most previous works have solved the problem by discretizing them with the consequent loss of information. Another common alternative assumes that the data are generated by a Gaussian distribution (parametric approach), such as conditional Gaussian networks, with the consequent error in the estimation if the true density differs from it.

In order to break with the strong parametric assumption, this work introduces the conditional flexible network paradigm for supervised classification. This paradigm is a Bayesian network which estimates the true density of the continuous variables using kernels. Moreover, some of the most popular Bayesian multinomial network based classifier induction algorithms (naive Bayes, tree-augmented naive Bayes, $k$-dependence Bayesian classifier and Bayesian network-augmented naive Bayes) are adapted to the conditional flexible network paradigm. Besides their thresholded versions are introduced in order to avoid the compulsory addition of arcs between low correlated variables.

The conditional flexible network can be seen as a generalization of the conditional Gaussian network because it allows a more flexible and precise estimation of the true densities. From the point of view of modelling correlations between predictor variables, the classifiers presented in this work can be seen as the natural extension of the flexible naive Bayes classifier proposed by John and Langley (1995) breaking with the naive Bayes independence assumption allowing dependencies between variables.

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

Supervised classification 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) [42, 47], based on probabilistic graphical models (PGMs) [3, 35], 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 (in)dependence relations between triplets 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 type of variables are considered: the class variable or class \( C \), and the rest of the variables or predictors \( X = (X_1, \ldots, X_d, X_{d+1}, \ldots, X_{d'}) \). \( \{X_1, \ldots, X_d\} \) 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 \( x \) consists in choosing the class \( c^* \) with the highest a posteriori probability, \( c^* = \arg\max_c p(c|x) \). This entails the use of the winner-takes-all rule. The classification process can be done in the following way with the BNs:

\[
p(c|x) \propto \rho(c, x) = p(c) f(x|c) = p(c) \prod_{i=1}^{d} f(x_i|\text{pa}_i) \prod_{j=d+1}^{d'} p(x_j|\text{pa}_j) \quad (1)
\]

where \( \text{pa}_i \) is an instantiation of the predictors \( \text{Pa}_i \), 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 [8]. This kind of classifier is known as generative, and it forms the most common approach in the BN literature for classification [4, 13, 19, 32, 33, 38, 46, 54]. Generative classifiers use the joint probability function of the predictor variables and the class, \( p(c, x) \). They classify a new instance by using the Bayes rule in order to compute the posterior probability, \( p(c|x) \) of the class variable given the values of the predictors (see Equation 1). On the other hand, discriminative classifiers [24, 7, 51, 55] directly model the posterior probability of the class conditioned to the predictor variables \( p(c|x) \). This work presents a set of generative classifiers based on a new family of BNs that we call conditional flexible network.

It has to be highlighted that this work is centered on the difficulties of modelling continuous variables directly (without discretization), and their relations in the BN paradigm. The modelling of the mixed domains in BNs can be done as follows: to learn a density function conditioned to a discrete variable is equivalent to creating the partitions in the data base 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
In order to improve the readability and simplicity of the paper, from here on, we deal with continuous domains only (without discrete predictors).

![Figure 1: Different structure complexities of BN based classifiers.](image)

One of the simplest classifiers based on BNs is the naive Bayes (NB) [13, 33, 38]. NB assumes that the predictors are conditionally independent given the class. Figure 1(a) represents a NB structure. In spite of this strong assumption, its performance is surprisingly good, even in data bases which do not hold with the independence assumption [12].

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 Figures 1(b), 1(c) and 1(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 Bayesian network-augmented naive Bayes (BAN) structure. On one hand, NB structure forbids any arc between predictor variables. On the other hand, the BAN structure does not constrain the number of arcs allowed between variables. 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 ($k$DB) structures to $k$ plus the class variable. It must be noted that all the structures shown in Figure 1 are constrained to graphs with the class variable as the root, which is also the father of each predictor variable of the model (augmented NB models).

![Figure 2: Complete and incomplete kDB structures with $k=2$.](image)

We say that a structure is complete when all predictor variables are included and no more dependencies can be allowed between them. Otherwise, the struc-
ture is incomplete. Complete and incomplete $k$DB structures with $k = 2$ are shown in Figure 2.

The structures themselves represent domain knowledge and can be interpreted in terms of conditional (in)dependencies, constructing the associated independence graph. In addition, they represent a factorization of the joint distribution $\rho(c, x)$, which is based on the relations of conditional (in)dependencies that are inferred from the structure. Therefore, the structures can be understood as simplifications of the joint distribution, $\rho(c, x)$. Once this factorization represents a simplification, it requires fewer parameters than $\rho(c, x)$.

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 estimate directly 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.

The most extended 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 [3, 47] (BMN). This paradigm only handles discrete variables and if a continuous variable is present, it must be discretized with the consequent loss of information [63]. In spite of that, using discretization plus multinomial option, the true density is generally correctly estimated for classification purposes [4]. This approach could have some problems to model a graph with a complex structure and/or with variables discretized in many intervals. On the one hand, the number of parameters to be estimated can be very high. For example, in a BAN structure the number of parameters needed is $O(r \prod_{i=1}^{d} r_i)$ where $r$ is the cardinality of the class $C$, and $r_i$ is the cardinality of the variable $X_i$. On the other hand, the number of relevant cases used to compute each parameter can be very low and, therefore, the statistics obtained might not be robust [25]. A battery of BMN-based classifier induction algorithms of different structural complexities has been proposed in the literature: naive Bayes [13, 33, 38] (NB), tree-augmented naive Bayes [19] (TAN), $k$-dependence Bayesian classifier [54] ($k$DB), semi naive Bayes [32, 46], Bayesian network-augmented naive Bayes [4] (BAN) and general Bayesian network [4]. 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 extended proposal and it usually provides a reasonable approximation to many real-world densities.
This choice assumes that continuous variables conditioned to a value of their parents follow a conditional Gaussian density. The paradigm, based on BNs which makes this assumption, is known as *conditional Gaussian network* (CGN) [1, 21, 35, 36, 37, 44]. In this work, we reference the classifiers based on CGN as Gaussian classifiers. Although a Gaussian density may provide a reasonable approximation to many real world distributions, it is certainly not always the best approximation. This problem is illustrated in the example of Figure 3, where a mixture of two Gaussian is estimated by means of a single one. This suggests another direction in which we might profitably extend and improve the Gaussian (parametric) approach: by using more general approaches to density estimation (non-parametric), for example, kernel estimation [27]. If the true densities are not too far from the Gaussian the classifiers based on CGNs have a behavior, at least, comparable to the classifiers based on BMNs [44]. It must be noted that CGNs can model more complex graphs than BMNs because for CGNs a complete graph with the class and continuous predictors needs only $O(rd^2)$ parameters to be modelled. 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 train set).

It was previously noted that, in order to break with the strong parametric assumption, this work presents the *conditional flexible network* paradigm (CFN). CFN uses the nonparametric *kernel density estimation* in order to model the conditional density of a continuous variable given a specific value of its parents. The kernel estimation method can approximate more complex distributions than the Gaussian parametric approach. It should be clarified that kernels are not used by CFNs to model the borders between classes (class boundaries), as the support vector machines do.

This work is an introduction of the CFNs limited to the supervised classification using the Bayes rule. Consequently, we present the CFN 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 [13, 33, 38], TAN [19], kDB [54] and BAN [4]). The classifiers presented are ordered by their structural complexity, ranging from naive Bayes to complete graph. We call the classifiers based on CFN *flexible classifiers*.

It must be noted that the CGN and CFN can also handle directly discrete variables, with a unique constraint: a continuous predictor can not be parent of a discrete predictor. In this work, as we noted before, we deal only with
continuous predictors, without discretizing them.

Figure 4: Representation of a set of Bayesian network based classifiers taking into account the structural complexity and the density estimation skill used by the model.

Figure 4 represents a set of well known classifiers based on Bayesian networks considering their structural complexity and the flexibility\(^1\) of the density estimation that they use. The origin of the coordinates is a classifier modelled with Gaussian densities under the conditional independence assumption between predictors given the class variable. On the other hand, the classifier furthest from the origin is a BAN structure modelled using kernel density estimation. Kernel density estimation can be seen as a more flexible estimator compared to the multinomial distribution, in the same way that kernel density estimation is considered more flexible than the histograms [57].

The paper is organized as follows. Section 2 introduces the non-parametric kernel based density estimation which will be used to model the CFNs. In Section 3 a group of estimators for the amount of mutual information based on kernels is presented. Their corresponding expressions will be used by the classifier induction algorithms proposed. Section 4 introduces a set of classifier induction algorithms, based on CFNs, ordered by their structural complexity: naive Bayes, tree-augmented naive Bayes, \(k\)-dependence Bayesian classifier and Bayesian network-augmented naive Bayes. In addition, their thresholded versions are presented. The computational and storage requirements of the proposed algorithms are also analyzed. In Section 5 the experimental results for the classifiers proposed are presented and analyzed. The experimental study includes results in four artificial data sets and 18 data sets of the UCI repository [41]. The artificial domains illustrate the advantages of the classifiers based on CFNs which model the correlations between predictors. The classification error in the 18 UCI repository data sets is estimated for the presented algorithms and

\(^1\)For us the flexibility of a density estimator represents the capability of modelling densities.
for ten benchmarks. The estimation of the bias plus variance decomposition of the expected error [31] for the introduced algorithms is also performed in the selected 18 UCI repository data sets. And finally, Section 6 summarizes the main conclusions of our paper and exposes the future work related with CFNs.

2 Kernel density estimation

The kernel based $d$-dimensional estimator [61] in its more general form is

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

where $H$ is a $d \times d$ bandwidth or smooth matrix (BM), $x$ is a $d$-dimensional instance, $n$ is the number of cases from which the estimator is learned, $i$ is the index of a case in the train set, and $K_H(\cdot)$ is the kernel function used. The kernel based density estimate $f(\cdot; H)$ is determined by averaging $n$ kernel densities $K_H(\cdot)$ placed at each observation $x^{(i)}$. The kernel function $K_H(\cdot)$ used is defined as:

$$K_H(x) = |H|^{-1/2}K(H^{-1/2}x)$$ (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 $H$

$H$ 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 $x$ is simply the average of the $n$ kernel ordinates 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, as we will note later, the choice of value for the bandwidth is very important [61]. Examples for the univariate and bivariate density estimation are shown at Figures 5(a) and 5(b).

Our CFNs use the $d$-dimensional unitary Gaussian density with unitary covariance matrix, $\Sigma = I$, in order to estimate density functions:

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

It must be noted that one can use another kernel function as well, such as Epanechnikov, biweight, triweight, triangular, rectangular or uniform [57, 61].
Figure 5: 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.

Thus, $K_H(x - x^{(i)})$ is equivalent to $N(x^{(i)}, H)$ density function. $H$ is an 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 $H$ 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 $H$

Although our work is centered on supervised classification, this subsection avoids the BM selection problem from the density estimation point of view. In order to approach the Bayes error\(^2\), we try to minimize the estimation error of each density estimated. The selection of a good BM $H$ is crucial for the density estimation, even more than the kernel generic function $K(\cdot)$ used\(^9\). $H$ establishes the degree of smoothing of the density function estimation.

In the univariate case the smooth degree depends on a unique parameter\(^3\), $h$. Figure 2.1 shows the effect of the parameter $h$ in the estimation of the true density using the same twelve train cases. Intuitively, with $h$ near to zero, a noisy estimation is obtained by the undersmooth effect (see Figure 6(a)). As $h$ increases the noise in the estimation is reduced and the univariate density begins to approximate the true density, until the optimum is reached (see Figure 6(b)). As $h$ increases, and distances itself from the optimum, the estimation starts to lose details due to the oversmooth effect (see Figure 6(c)). Finally, as $h$ tends to $\infty$, the function becomes uniform.

\(^2\)Bayes error is the error associated to the Bayes classifier and it is the lower bound for the error to be reached by a classifier. The Bayes classifier can be seen as the classifier which knows the true density of the data and it selects the corresponding optimum class for each instance, based on this knowledge.

\(^3\)If $d = 1$, then $H^2 = H$. 

8
In order to guarantee an efficient parametrization of a great number of densities, our goal is to present a computationally inexpensive technique to compute the BM $H$. Thanks to its low computational cost, this technique allows to search in the space of possible classifier structures (even the most complex), or to compute the amount of mutual information between every pair of predictor variables.

In order to determine a complete BM, the number of parameters to be computed is $O(d^2)$. Therefore, the problem becomes intractable quickly as $d$ grows, so it suggests that it is necessary to constrain the BM $H$. Normally, in the literature, three options have been considered to approximate the optimal BM [58] and, all of them depend on a unique smooth parameter $h$:

1. $H = h^2I$. Setting the smoothing parameter to be constant for every variable implies that the amount of smoothing in each direction is the same. This is viable only if the scales of all variables are fairly constant, so this would be done only after each variable is standardized to a common scale. We call this option scaled [58].

2. $H = \text{diag}(h_1^2, ..., h_d^2) = h^2 \text{diag}(s_1^2, ..., s_d^2)$. $h_i$ is the smooth parameter of the variable $X_i$ and $s_i$ is a dispersion measure such as the sample standard deviation of $X_i$. This parametrization allows different amounts of smoothing in each coordinate direction. This approach is also the “practical” version of the scaled proposed because it does not need a previous standardization of the variables. We call this option differential scaled [58].

3. $H = h^2S$, where $S$ is an estimate of the covariance matrix of the predictors. The idea is to use a kernel with the same shape as the dispersion of the multivariate density. This transformation is often called sphering [58]. The estimator associated was introduced by Fukunaga [20].

The three options only need to compute a unique parameter $h$ and, thus, they are suitable for the CFN paradigm, since they can provide an estimator in an efficient way. In general, there is no better approach for all domains. Wand and
Jones [60] advise us not to use the spherering approach in general for densities estimation. The density can have a local shape which is very different to overall shape. On the other hand, Duong and Hazelton [16] use the spherering approach and they obtain good results in the majority of their estimations. As we will explain in Section 5, we have tested three options experimentally in different classification tasks by means of the estimated accuracy and the bias-variance decomposition. We also conclude that there is not a better approach in general. The three transformations obtain quite similar results in almost all domains, but, differential scaled and spherering seem to have superior behavior in a subset of the domains included in Section 5 (generally the most difficult ones).

The principal problem selecting optimum BM, in the majority of practical situations, is that the real density function is unknown and, thus, it is really difficult to optimize any appropriate loss criteria (a distance function between the real and estimated densities). Most of the approaches in the literature have tried to minimize square loss related criteria because it is one of the easiest (from the point of view of mathematics). In our work, we will use the normal rule [57], which selects the BM $H$ that minimizes the mean integrated squared error under the assumption that the variables follow a multidimensional Gaussian density with the identity covariance matrix $I$. Under this assumption we can compute the optimum $h^2$ as follows [57]:

$$h^2 = \left(\frac{4}{d+2}\right)^{\frac{1}{d+4}} n^{-\frac{1}{4}}$$

We have chosen the normal rule because of its computational efficiency. It only has to compute a fixed number of operations $O(1)$ independent from the dimension of the density and the number of examples used to estimate it. This allows an efficient parametrization of any CFN based classifier model. Furthermore, we know that the normal rule tends to oversmooth [61] the estimation and, thus, it could obtain more stable estimations with noisy data. We are interested in learning good classifiers and we think that this rule makes estimations with a good generalization power for classification. Taking into account the results of Section 5, it seems that the normal rule at least inducts classifiers as good as those obtained with the heuristic proposed by John and Langley [2, 27].

In order to estimate difficult densities, a variable bandwidth is desirable because it possesses the flexibility necessary for capturing complicated shapes of curves. The optimal smoothing degree usually varies across the real line: for example, data points in regions where the data are sparse will have flatter kernel associated with them [56, 57]. Thus, often a fixed bandwidth is inadequate, and it has locally adaptive variants, where the bandwidth is chosen separately for each fitting point: variable and adaptive bandwidth selection methods [61]. Adaptive bandwidth selection methods perform significantly better than any constant-bandwidth method in density estimation tasks [28]. Non-parametric discriminant analysis with variable kernel approach gives improved results for the long-tailed models, while giving results comparable to the fixed-kernel method for other models [57]. In order to model flexible classifiers, these adaptive-bandwidth methods have been refused because of their computational
cost and space requirements.

2.2 Asymptotic properties of kernel density estimation

It is known that the kernel density estimation obtains good approximations of the true density: the estimation converges to the true density when a sufficient number of cases is learned. Moreover, theoretically, \( \forall t, p(\lim_{n \to \infty} |\hat{f}_n(x) - f(x)| < t) = 1 \) [57], where \( \hat{f}_n(x) \) is a kernel based estimator using \( n \) train samples.

From the classification point of view, this property is likely to reach the Bayes error, for classifier structures which model the true correlations between variables, when the train set has the sufficient number of cases. In Subsection 5.1 the evolution of the error with the number of cases is studied in four kinds of artificial domains.

3 Mutual information and conditioned mutual information estimators

In this section we propose a set of estimators based on kernels for the amount of mutual information [6]. These estimators will be used in the classifier induction algorithms presented in Section 4. Some mutual information based measures are related directly to the expected error of BN based classifiers at different structural complexity levels [43]. 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.

The amount of mutual information between two multivariate variables \( X \) and \( Y \) is defined as [6]

\[
I(X, Y) = \int f(x, y) \log \frac{f(x, y)}{f(x)f(y)} \, dx \, dy = E(\log \frac{f(x, y)}{f(x)f(y)})
\] (6)

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

\[
\hat{I}(X, Y) = \frac{1}{n} \sum_{i=1}^{n} \log \frac{\hat{f}(x^{(i)}, y^{(i)})}{\hat{f}(x^{(i)})\hat{f}(y^{(i)})}
\] (7)

where \( \hat{f}(\cdot) \) is a kernel based density estimation (see Equation 2). A similar proposal can be found in [39].

On the other hand, in order to compute the amount of mutual information between a multivariate continuous variable \( X \) and a multivariate discrete
variable $C$ we propose the following estimator:

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

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

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

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

where $I_c(X, Y) = E(\log \frac{f(x,y|c)}{f(x|c)f(y|c)})$. Using Equations 7 and 9, we propose the following estimator for the amount of conditioned mutual information:

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

The super-index $c: j$ refers to the $j$-th case in the partition induced by the value $c$, and $n_c$ is the number of cases verifying that $C = c$.

All the estimators presented (Equations 7, 8 and 10) can be adjusted to any CFN based paradigm because $X$, $Y$ and $C$ can be of any arbitrary dimension.

### 4 Classifiers based on conditional flexible networks: flexible classifiers

As we said previously, CFN is a BN. In other words, it is a directed acyclic graph with arcs representing conditional (in)dependence relations, nodes representing variables, and a set of parameters which model the conditional (in)dependence relations of the graph. It has the same structural constraints as CGN: a discrete node can not have continuous parents in the graph. The main difference between CGN and CFN is the density estimation that they use to model the graph and create the factorization associated (parametric vs non-parametric approaches respectively).

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

$$f(x|y, c) \cong \hat{f}_c(x|y) = f_c(x|y) = \frac{\hat{f}_c(x,y)}{f_c(y)}$$

12
The following Subsections present a group of well-known flexible classifier induction algorithms, ordered by their structural complexity: flexible naive Bayes, flexible tree-augmented naive Bayes, flexible \(k\)-dependence Bayesian classifier and flexible Bayesian network-augmented naive Bayes. Examples of their different structure complexities are shown in Figure 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. It is not mandatory to include the entire set of predictor variables (complete NB structure). The space of possible NB structure is \(O(2^d)\). An example of the NB structure is shown at Figure 1(a).

The naive Bayes classifier induction algorithm (\(NB\)) [13, 33, 38] learns the complete naive Bayes structure, which is known \textit{a priori} (arcs from the class to each predictor). The accuracy obtained with this classifier is surprisingly high in some domains, even in data sets that do not obey its strong conditional independence assumption [12]. Originally, NB classifier was introduced for the BMN (multinomial NB or \(mNB\)) and we have adapted to the CFN paradigm (flexible NB or \(fNB\)).

Thanks to the conditional independence assumption, the factorization of the joint probability is greatly simplified. After adapting Equation 1 to NB structure particularities, the following factorization is obtained:

\[
p(c \mid x) \propto p(c) \prod_{i=1}^{d} f(x_i \mid c)
\]

where \(f(x_i \mid c)\) is estimated by means of a kernel based density. For example, the factorization of Figure 1(a) results in \(p(c \mid x) \propto p(c)f(x_1\mid c)f(x_2\mid c)f(x_3\mid c)f(x_4\mid c)\).

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 only to a subset of selected 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 Figure 1(b).

This subsection introduces the adaptations of Friedman et al.’s algorithm [19] to the CFN paradigm. This algorithm finds the tree structure that maximizes the likelihood given the data. Friedman et al.’s algorithm [19] follows the general outline of Chow and Liu’s procedure [5], 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 calculate 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 Equation 10.

The algorithm [19] starts from a complete NB structure and continues adding allowed arcs between predictor until the complete TAN structure is formed. The arcs are included in order of their conditional mutual information $I(X_i, X_j | C)$. Originally, TAN algorithm was introduced for the BMN (multinomial TAN or mTAN) and we have adapted it to the CFN paradigm (flexible TAN or fTAN).

The factorization of the TAN structures is more complex than in the case of NB structures because of the existence of conditional dependencies between predictors variables given the class. For example, the factorization of Figure 1(b) is: $p(c|x) \propto p(c)f(x_1|x_2,c)f(x_2|x_3,c)f(x_3|x_4,c)f(x_4|c)$. fTAN estimates each function $f(\cdot)$ using kernels.

4.3 Flexible $k$-dependence Bayesian classifier

The $k$-dependence Bayesian classifier structure ($k$DB structure) extends TAN structures allowing a maximum of $k$ predictor parents plus the class for each predictor variable (NB and TAN structures are equivalent to $k$DB structures with $k = 0$ and $k = 1$ respectively). The $k$DB structures can be regarded as a spectrum of allowable dependence in a given probabilistic graphical model with the NB structure at the most restrictive extreme and the complete BAN at the most general structure. Example of the $k$DB structure is shown in Figure 1(c).

The $k$DB 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 [54], and fewer cases are used to compute the necessary statistics. The use of a CFN instead of a BMN avoids the problem of modelling a structure without the restriction in the number of parents, as the number of required parameters only grows quadratically. In addition, to estimate the parameters, the entire data set is used instead of learning from a data set partition. Thus, CFNs allow the construction of classifiers with a high number of dependencies between variables.

This subsection introduces the adaptation of Sahami’s algorithm called the $k$-dependence Bayesian classifier [54] to the CFN 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 CFN paradigm, we compute the amounts of mutual information $I(X_i, C)$ and $I(X_i, X_j | C)$, using the estimators proposed in Equations 8 and 10 respectively.

Sahami’s algorithm [54] starts from a structure with only the class variable.
At each step, from the subset of non-included predictor variables, the variable $X_{\text{max}}$ with the highest $I(X_i, C)$ is added. Next, arcs from the variables included in the structure to variable $X_{\text{max}}$ are added while it is possible, as long as the maximum number of parents $k$ is not surpassed. The arcs are added following the order of $I(X_{\text{max}}, X_j|C)$ from the greatest value to the smallest. The algorithm continues until no more arcs are allowed and, thus, a complete structure is learned for a specific $k$ value. Originally, $k$DB algorithm was introduced for the BMN (multinomial $k$DB or m$k$DB) and we have adapted to the CFN paradigm (flexible $k$DB or f$k$DB).

The classification process with $k$DB structures and TAN structures is done in a similar way. For example, the factorization of Figure 1(c) is:

$$p(c|x) \propto p(c)f(x_1|x_2, x_3, c)f(x_2|x_3, c)f(x_3|x_4|x_2, x_3, c).$$

$\text{f$k$DB}$ estimates each $f(\cdot)$ using kernel based density.

### 4.4 Bayesian network-augmented naive Bayes: Parzen window classifier

A classifier with unrestricted structure, known as Bayesian network-augmented naive Bayes (BAN) [4], could represent the factorization of the joint distribution without any simplification (BAN complete structure). An example of this structure is shown in Figure 1(d). The associated factorization is:

$$p(c|x) \propto p(c)f(x_1|x_2, x_3, x_4, c)f(x_2|x_3, c)f(x_3|x_4|x_2, x_3, c) = \rho(x, c)$$

where each density $f(\cdot)$ is also estimated using kernels. Originally, BAN classifier was introduced for the BMN (multinomial BAN or mBAN) [4], it have been previously adapted to the CGN paradigm (Gaussian BAN or gBAN) in [44] and this work adapts it to the CFN (flexible BAN or fBAN).

The different BAN complete structures are equivalent because they represent the exact factorization of the joint density $\rho(x, c)$: Thus, the structure can be fixed a priori. Besides, for classification, they are equivalent to the Parzen window classifier [20, 45, 52]. Consequently, all the classifiers presented in this work can be seen as a constrained Parzen window classifier, in the same way in which they represent simplifications of the joint density $\rho(x, c)$.

### 4.5 Thresholded versions: Learning incomplete structures

Previously presented classifiers have some flexibility limitations even using kernel density estimation. All of them generate classifiers with complete structures at different structural complexity degrees. Therefore, some arcs are necessarily added, and the addition of these arcs can produce less informative structures and less accurate classifiers. In order to avoid this problem some arcs are forbidden, taking into account the amounts of mutual information $I(X_i, C)$ and $I(X_i, X_j|C)$, and two thresholds $t_{\text{class}}$ and $t_{\text{cond}}$ respectively. Thresholded versions (fNB, fTAN, f$k$DB and fBAN) are similar to the previously presented algorithms (fNB, fTAN, f$k$DB and fBAN) but they forbid the less relevant arcs, in terms of mutual information. Our goal is not to present a new feature selection
technique, even some predictors could be discarded. They are discarded taking into account their irrelevancy for the class prediction (low $I(X_i, C)$ values) without using any redundancy measure. Our goal is to obtain more informative structures and, therefore, we have tried to avoid the compulsory addition of arcs between low-correlated variables in the algorithms.

The arcs from the class to any variable $X_i$ are forbidden if the predictor variable has less amount of mutual information with the class, $I(X_i, C)$, than a previously fixed threshold $t_{class}$. Therefore, each of the predictors $X_i$ included verifies that $I(X_i, C) > t_{class}$.

In the same way, the arcs between any two variables $X_i$ and $X_j$ are forbidden if they have an amount of conditional mutual information $I(X_i, X_j | C)$ less than $t_{cond}$. This decision avoids the compulsory addition of arcs between low conditional dependent variables given the class and, therefore, more informative structures should be obtained.

Generally, in order to obtain theoretical thresholds in an efficient way which guarantees the inclusion of the most relevant arcs between variables, the distribution of the mutual information based on kernels must be known. But the true distribution is unknown and therefore, in order to minimize the error of each CFN based classifier the thresholds, $t_{class}$ and $t_{cond}$, have been obtained experimentally using a stratified 10-fold cross-validation.

4.6 Storage and computational complexity

CFN 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 CFN based classifiers introduced are presented.

In order to obtain the structure to be modelled 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 to the class ($\forall i, j I(X_i, X_j | C)$), and the mutual information between each variable and the class ($\forall i I(X_i, C)$). In order to compute $\forall i, j I(X_i, X_j | C)$, using the estimator based on kernels proposed in Equation 10, the number of required operations is $O(n^2 d^2)$. On the other hand, the number of operations required to compute $\forall i I(X_i, C)$, using the estimator based on kernels proposed in Equation 7, is $O(n^2 d)$.

In order to model a classifier based on CFNs from data given its structure a priori (parametric learning), every instance in the train set and the covariances between each pair of correlated predictors (given by the structure) have to be stored. Therefore, the classifier needs to learn from data $O(nd + (k + 1)d)$ parameters to proceed with classification, where $n$ is the number of instances in the training set, $d$ is the number of predictors included, and $k$ is related with the given structure (kDB structures, with $k = 0$ for NB, $k = 1$ for TAN and $k = d-1$ for BAN). In order to classify a new instance, the same CFN based classifier requires to compute $O(nd+(k+1)d)$ operations. Table 1 summarizes the storage
Table 1: Storage and computational complexity of the parametric learning of a classifier with a $k$DB structure based on both CGN and CFN paradigms, in a domain with $d$ predictor variables.

<table>
<thead>
<tr>
<th>Operations</th>
<th>CGN based</th>
<th>CFN based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train on $n$ cases</td>
<td>$O(nd)$</td>
<td>$O(m(k + 1)d)$</td>
</tr>
<tr>
<td>Test on $m$ cases</td>
<td>$O(m(k + 1)d)$</td>
<td>$O(nd + (k + 1)d)$</td>
</tr>
</tbody>
</table>

and computational complexity for both CGN and CFN based classifiers given a $k$DB structure.

Therefore, in order to induce the classifier structure and learn its parameters, the algorithms presented have the following computational requirements: $fNB$ $O(nd)$, $tNB$ $O(u^2d)$, and $fTAN$, $tFTAN$, $tfDB$, $tfkDB$ and $tfBAN$ $O(u^2d^2)$ operations. Due to the computational complexity requirements of the algorithms proposed, and considering the classifier induction and class prediction stages, it can be said that they are more suitable for low and medium sized data sets.

5 Experimental results

This section presents the experimental results of the previously introduced CFN based classifier induction algorithms: from $fNB$ to $fBAN$. In order to gain transparency and simplicity, and to reduce space, results for the thresholded versions have been removed from the tables.

First, in Subsection 5.1, we motivate the use of classifiers based on CFNs, which model the correlation between predictors, presenting their advantages by means of some ad-hoc designed artificial data sets. Then, in Subsection 5.2, we present a set of results in eighteen UCI repository continuous data sets [41]. These results include the estimation of the error (Subsection 5.2.1), a comparison table which summarizes some hypothesis tests (Subsection 5.2.2) and the bias plus variance decomposition [31] of the expected error (Subsection 5.2.3), at each data set, for each previously introduced classifier induction algorithm.

It must be highlighted that spherical and differential scaled smoothing options obtain quite similar errors and bias plus variance decomposition of the expected error in the experimented domains. Both options show significant inferior errors than scaled. Due to this similarity in the estimated errors and to the extra computational requirements of the sphering, we have decided to perform the following experimentation using the differential scaled proposal.

5.1 Motivation: artificial data sets

This subsection presents the results of the classifiers in four artificial domains with two continuous predictors, $\{X, Y\}$, and two equally probable classes, $c \in \{0, 1\}$ with $p(C = 0) = 0.5$. Taking into account the dependencies between predictor variables (correlated or non-correlated) and the kind of densities (Gaus-
sian or non-Gaussian), we propose the following four kinds of domains:

- Domains with non-correlated Gaussian predictors (NCG).
- Domains with correlated Gaussian predictors (CG).
- Domains with non-correlated non-Gaussian predictors (NCNG).
- Domains with correlated non-Gaussian predictors (CNG).

We have created by simulation an example of each of the four kinds of domains mentioned. Each simulated data set is defined by means of its associated generalized function \( \rho(x, y, c) = p(c) f(x, y|c) \):

- NCG data set is generated from the density functions defined by \( f(x|C = 0) \equiv f(y|0) \sim N(\mu_0 = 0, \sigma_0^2 = 1) \) and \( f(x|1) \equiv f(y|1) \sim N(\lambda, 1) \) with \( \lambda = 1.812 \) (see Figure 7(a)).
- CG data set is defined by the density functions \( f(x, y|0) \sim N(\mu_0 = (0, 0), \Sigma_0 = [1, 0.75; 0.75, 1]) \) and \( f(x, y|1) \sim N((\lambda, \lambda), [1, -0.75; -0.75, 1]) \) with \( \lambda = 1.437 \) (see Figure 7(b)).
- NCNG data set 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 \( \lambda = 4.703 \) (see Figure 7(c)).
- CNG data set 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 \( \lambda = 3.387 \) (see Figure 7(d)).

At each artificial domain the parameter \( \lambda \) is selected 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 [14] as:

\[
\epsilon_M = \int f(x)(1 - p(c^*|x))dx
\]  

where \( p(\cdot) \) and \( f(\cdot) \) are the true probability distribution and density functions of the domain, respectively. Besides, \( c^* = \arg \max_c p_M(c|x) \) where \( p_M(\cdot) \) is the distribution learned by the classifier. Therefore, the Bayes error \( \epsilon_B \) is defined as the error of the Bayes classifier, which is obtained under the winner-takes-all rule, by the decision \( c^* = \arg \max_c p(c|x) \).

In order to study the errors of different classifiers (gNB, gTAN, fNB and fTAN\(^4\)) with different training set sizes, at each artificial domain proposed, we have sampled 8 different sets with sizes of 10, 20, 40, 80, 160, 320, 640, and 1280 respectively. All the classifiers learned have been tested on an independent data set of 2000 cases. In order to correctly estimate the errors for each train set size the experiment has been repeated 5 times. The evolution of the errors...
estimated, with each classifier learned, are presented in the graphics of Figure 9.

From the results represented in Figure 9 the following conclusions can be obtained:

- In the NCG domain (see Figure 7(a)) all the classifiers behave similarly (see Figure 8(a)). They reach the Bayes error with train sizes of 180 or greater. This suggests that NCG domains can be modelled similarly for classifying using the kernel density estimation with the normal rule, and using the parametric Gaussian approach.

- In the CG domain (see Figure 7(b)) the classifiers which model the correlation between predictors, Gaussian and flexible TAN classifiers, seem to behave a little better than the classifiers which do not model the correlations, Gaussian and flexible NB (see Figure 8(b)). Gaussian and flexible TAN seem to reach the Bayes error with train sizes of 320 and 80, respectively. These results suggest that CG domains are better modelled.

\footnote{It must be noted that TAN, kDB at \( k > 0 \) values, and BAN complete structures are equivalent in the bivariate domains.}
by classifiers which consider the correlation between the predictor variables. Besides, the results suggest that flexible and Gaussian approaches can similarly model the CG domains.

- In the NCNG domains (see Figure 7(c)) flexible classifiers behave quite better than Gaussian based classifiers (see Figure 8(c)). They reach the Bayes error with train sizes of 320. It is clear that flexible classifiers can model better the NCNG domains.

- In the CNG domains (see Figure 7(d)) flexible TAN classifier behave quite better than the rest of the classifiers (see Figure 8(d)). These results suggest the need to model the correlation between predictors using kernel based density estimation for the CNG domains.
Therefore, it could be concluded that flexible classifiers which model the correlation between predictors are at least equally suitable than the Gaussian classifiers for modelling any continuous domain for the supervised classification task. Moreover, in CNG domains Flexible classifiers are likely to obtain quite better results.

The flexible classifiers which model correlations between variables reach the Bayes error at each artificial domain, even the most difficult ones, with train sets of size equal to or greater than 320. Besides, flexible classifiers obtain acceptable errors with train set sizes of 80 and in Gaussian domains even with fewer cases. This analysis suggests that flexible classifiers could model domains, even quite complex ones [61] (with 3 or 4 modes for each class conditioned density), with small sample sizes. Moreover, in order to model Gaussian domains the kernel density approach with normal rule converges to the Bayes error with fewer cases than the Gaussian approach, specially in the CG domain.

5.2 UCI data sets

This subsection is divided in three main parts. In Subsection 5.2.1 the classification error (see Equation 13) of the presented algorithms is estimated for each data set. In order to compare their results, quadratic discriminant analysis (QDA) has been included in the study as benchmark classifier. 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, and multilayer perceptron) are presented. In Subsection 5.2.2 the comparisons between every pair of flexible classifiers presented, based on the previously introduced error estimations, are summarized in a comparative table. Finally, in Subsection 5.2.3, and in order to study the nature of the error of the CFN-based algorithms, Kohavi and Wolpert’s bias plus variance decomposition of the expected error [31] is performed for the flexible classifiers presented and the benchmark QDA.

The results have been obtained in 18 UCI repository data sets [41], which only contain continuous predictor variables. In order to interpret the results we must take into account that most data sets of the UCI repository are already preprocessed [29]: in the data sets included, there are few irrelevant or redundant variables, and little noise [59]. Thus, it is more difficult to obtain statistically significant differences between the results of the algorithms in these types of data sets [59]. The main characteristics of the data sets included are summarized in Table 2.

Table 3 shows the results for the included benchmark algorithms: Gaussian NB and TAN [44] (gNB and gTAN), multinomial NB and TAN [19] (mNB and mTAN), k-nearest neighbour with $k = 1, 3$ [7] (k-NN) as lazy classifiers, ID3 [48] and C4.5 [49] as classification trees, and quadratic discriminant analysis (QDA) and multilayer perceptron [53] (MP) as discriminant functions. All of them, except gTAN [44], are implemented in Weka 3.4.3 statistical package [62]. The estimated classification errors summarized in Table 3 have been obtained, for each classifier at each data set, by a stratified 10-fold cross-validation process.
Table 2: Basic characteristics of the data sets: The number of different values of the class variable, the number of predictor variables, and the number of instances.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>num. class values</th>
<th>num. variables</th>
<th>num. instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balance</td>
<td>3</td>
<td>4</td>
<td>625</td>
</tr>
<tr>
<td>Block</td>
<td>5</td>
<td>10</td>
<td>5474</td>
</tr>
<tr>
<td>Bupa</td>
<td>2</td>
<td>6</td>
<td>246</td>
</tr>
<tr>
<td>Haberman</td>
<td>3</td>
<td>4</td>
<td>307</td>
</tr>
<tr>
<td>Hayes</td>
<td>3</td>
<td>4</td>
<td>160</td>
</tr>
<tr>
<td>Image</td>
<td>7</td>
<td>18</td>
<td>2310</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>2</td>
<td>34</td>
<td>351</td>
</tr>
<tr>
<td>Iris</td>
<td>3</td>
<td>4</td>
<td>150</td>
</tr>
<tr>
<td>Letter</td>
<td>26</td>
<td>16</td>
<td>20000</td>
</tr>
<tr>
<td>Liver</td>
<td>2</td>
<td>6</td>
<td>345</td>
</tr>
<tr>
<td>Pima</td>
<td>2</td>
<td>8</td>
<td>768</td>
</tr>
<tr>
<td>Satellite</td>
<td>6</td>
<td>36</td>
<td>6435</td>
</tr>
<tr>
<td>Sonar</td>
<td>2</td>
<td>60</td>
<td>208</td>
</tr>
<tr>
<td>Thyroid</td>
<td>3</td>
<td>5</td>
<td>215</td>
</tr>
<tr>
<td>Vehicle</td>
<td>4</td>
<td>19</td>
<td>846</td>
</tr>
<tr>
<td>Vowel</td>
<td>11</td>
<td>10</td>
<td>990</td>
</tr>
<tr>
<td>Waveform</td>
<td>3</td>
<td>21</td>
<td>50000</td>
</tr>
<tr>
<td>Wine</td>
<td>3</td>
<td>13</td>
<td>179</td>
</tr>
</tbody>
</table>

In order to learn the discrete classifiers presented in Table 3 (mNB, mTAN and ID3), data sets have been discretized with the Fayyad and Irani entropy based method [17].

QDA has been selected as the benchmark algorithm for the remainder of this section. Even its parametric assumption (multidimensional Gaussian distributed predictors) QDA obtains competitive results compared with the other state-of-the-art benchmarks included in Table 3:

- It has obtained the best score in more data sets than the others.
- It has the second best error average.

5.2.1 Classification error estimation

In order to estimate the classification error, for each flexible classifier and QDA benchmark, at each data set, a stratified 10-fold cross-validation process has been performed. The estimations of the error are summarized in Table 4.

Table 4 also summarizes four different analysis of the classification error estimated. The first analysis calculates for each classifier, the average estimated predictive accuracy across all data sets (Average row of the table). For example, QDA has obtained an average classification error of 16.7 across all domains (see Table 4).

The second analysis is a non-parametric paired Wilcoxon hypothesis test [15] for studying whether the QDA benchmark classifier, at each data set, has obtained statistically significant better or worse score values than each of the flexible classifiers included. This hypothesis test, with each data set, is based on the estimated predictive accuracies obtained with each fold of the stratified 10-fold cross-validation process. The test has been performed at $\alpha = 10\%$.
Table 3: The estimated errors obtained with a set of well known state-of-the-art algorithms. The best results, in each data set, are marked in grey.

<table>
<thead>
<tr>
<th># Data Set</th>
<th>k-NN</th>
<th>BMN</th>
<th>CGN</th>
<th>Classification trees</th>
<th>Discriminant func.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-NN</td>
<td>3-NN</td>
<td>mNB</td>
<td>mTAN</td>
<td>gNB</td>
</tr>
<tr>
<td>1 Balance</td>
<td>15.2 ± 3.3</td>
<td>15.2 ± 3.3</td>
<td>29.3 ± 3.9</td>
<td>28.7 ± 3.5</td>
<td>9.3 ± 1.4</td>
</tr>
<tr>
<td>2 Block</td>
<td>3.8 ± 1.4</td>
<td>3.3 ± 0.8</td>
<td>4.2 ± 0.9</td>
<td>2.8 ± 0.6</td>
<td>4.2 ± 1.1</td>
</tr>
<tr>
<td>3 Bupa</td>
<td>37.1 ± 6.0</td>
<td>38.3 ± 5.6</td>
<td>36.8 ± 10.0</td>
<td>36.8 ± 10.0</td>
<td>44.1 ± 6.9</td>
</tr>
<tr>
<td>4 Haberman</td>
<td>32.3 ± 6.7</td>
<td>29.7 ± 4.6</td>
<td>27.1 ± 3.1</td>
<td>27.1 ± 3.1</td>
<td>25.8 ± 4.1</td>
</tr>
<tr>
<td>5 Hayes</td>
<td>26.3 ± 7.8</td>
<td>53.8 ± 9.8</td>
<td>40.0 ± 3.1</td>
<td>40.0 ± 3.1</td>
<td>32.5 ± 6.7</td>
</tr>
<tr>
<td>6 Image</td>
<td>2.3 ± 1.0</td>
<td>3.3 ± 0.9</td>
<td>9.8 ± 1.8</td>
<td>4.9 ± 1.4</td>
<td>21.3 ± 1.7</td>
</tr>
<tr>
<td>7 Ionosphere</td>
<td>13.7 ± 4.4</td>
<td>13.4 ± 4.7</td>
<td>9.4 ± 3.6</td>
<td>6.5 ± 3.3</td>
<td>18.8 ± 6.2</td>
</tr>
<tr>
<td>8 Iris</td>
<td>4.7 ± 5.2</td>
<td>4.7 ± 5.2</td>
<td>6.0 ± 5.5</td>
<td>5.3 ± 5.0</td>
<td>4.7 ± 4.3</td>
</tr>
<tr>
<td>9 Letter</td>
<td>15.1 ± 2.3</td>
<td>19.4 ± 1.7</td>
<td>32.1 ± 3.0</td>
<td>26.6 ± 2.8</td>
<td>37.1 ± 3.9</td>
</tr>
<tr>
<td>10 Liver</td>
<td>37.1 ± 6.0</td>
<td>38.3 ± 5.6</td>
<td>36.8 ± 10.0</td>
<td>36.8 ± 10.0</td>
<td>43.5 ± 6.9</td>
</tr>
<tr>
<td>11 Pima</td>
<td>29.8 ± 4.4</td>
<td>27.3 ± 4.9</td>
<td>22.1 ± 3.3</td>
<td>21.1 ± 3.6</td>
<td>24.7 ± 3.8</td>
</tr>
<tr>
<td>12 Satellite</td>
<td>8.4 ± 2.3</td>
<td>8.4 ± 1.2</td>
<td>14.6 ± 2.3</td>
<td>10.4 ± 1.4</td>
<td>16.4 ± 2.9</td>
</tr>
<tr>
<td>13 Sonar</td>
<td>13.4 ± 6.6</td>
<td>14.0 ± 6.7</td>
<td>14.4 ± 5.1</td>
<td>14.4 ± 5.1</td>
<td>31.3 ± 5.8</td>
</tr>
<tr>
<td>14 Thyroid</td>
<td>2.8 ± 2.3</td>
<td>6.5 ± 5.2</td>
<td>3.7 ± 3.5</td>
<td>3.3 ± 3.0</td>
<td>3.7 ± 2.8</td>
</tr>
<tr>
<td>15 Vehicle</td>
<td>30.1 ± 4.2</td>
<td>28.5 ± 4.0</td>
<td>37.3 ± 3.9</td>
<td>25.8 ± 4.6</td>
<td>56.7 ± 4.1</td>
</tr>
<tr>
<td>16 Vowel</td>
<td>0.9 ± 1.0</td>
<td>3.1 ± 1.9</td>
<td>30.1 ± 5.6</td>
<td>24.5 ± 4.7</td>
<td>32.1 ± 4.1</td>
</tr>
<tr>
<td>17 Waveform</td>
<td>22.7 ± 2.9</td>
<td>19.6 ± 2.6</td>
<td>18.3 ± 2.7</td>
<td>15.4 ± 3.0</td>
<td>18.5 ± 1.6</td>
</tr>
<tr>
<td>18 Wine</td>
<td>5.0 ± 3.9</td>
<td>5.0 ± 3.9</td>
<td>1.1 ± 2.2</td>
<td>1.7 ± 2.6</td>
<td>2.8 ± 2.8</td>
</tr>
<tr>
<td>Average</td>
<td>16.7</td>
<td>18.4</td>
<td>20.8</td>
<td>18.5</td>
<td>23.7</td>
</tr>
</tbody>
</table>
Table 4: Summary of the estimated errors. The first row of the table contains quadratic naive discriminant analysis and the flexible classifier induction algorithms: flexible naive Bayes (INB), flexible tree-augmented naive Bayes (FTAN), flexible $k$-dependence Bayesian classifier with $k = 2, d/2$ (f2DB, fd.5DB) and flexible Bayesian network-augmented naive Bayes (fBAN).

<table>
<thead>
<tr>
<th># Data Set</th>
<th>QDA</th>
<th>INB</th>
<th>FTAN</th>
<th>f2DB</th>
<th>fd.5DB</th>
<th>fBAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Balance</td>
<td>8.3 ± 2.8</td>
<td>8.5 ± 1.4</td>
<td>x10.2 ± 3.1</td>
<td>x12.2 ± 2.4</td>
<td>x12.2 ± 2.4</td>
<td>x10.4 ± 1.6</td>
</tr>
<tr>
<td>2 Block</td>
<td>4.2 ± 1.3</td>
<td>3.7 ± 1.1</td>
<td>√3.4 ± 0.9</td>
<td>√3.1 ± 0.8</td>
<td>√3.6 ± 1.1</td>
<td>√3.3 ± 0.9</td>
</tr>
<tr>
<td>3 Bupa</td>
<td>39.2 ± 6.8</td>
<td>33.9 ± 6.9</td>
<td>34.5 ± 5.3</td>
<td>40.0 ± 8.9</td>
<td>38.9 ± 9.6</td>
<td>39.8 ± 8.1</td>
</tr>
<tr>
<td>4 Haberman</td>
<td>25.1 ± 4.9</td>
<td>26.2 ± 7.3</td>
<td>25.5 ± 7.9</td>
<td>25.8 ± 7.9</td>
<td>25.8 ± 7.9</td>
<td>25.8 ± 7.9</td>
</tr>
<tr>
<td>5 Hayes</td>
<td>38.1 ± 10.6</td>
<td>37.5 ± 12.4</td>
<td>√24.4 ± 9.6</td>
<td>√21.9 ± 8.4</td>
<td>√21.9 ± 8.4</td>
<td>√23.1 ± 7.9</td>
</tr>
<tr>
<td>6 Image</td>
<td>10.1 ± 1.8</td>
<td>x15.8 ± 1.3</td>
<td>x13.9 ± 1.3</td>
<td>x14.5 ± 1.6</td>
<td>x14.4 ± 1.2</td>
<td>x14.1 ± 1.5</td>
</tr>
<tr>
<td>7 Ionosphere</td>
<td>12.3 ± 4.4</td>
<td>9.7 ± 3.9</td>
<td>√7.1 ± 4.7</td>
<td>√7.4 ± 3.3</td>
<td>10.5 ± 5.6</td>
<td>10.8 ± 4.9</td>
</tr>
<tr>
<td>8 Iris</td>
<td>2.7 ± 3.3</td>
<td>4.0 ± 4.4</td>
<td>4.0 ± 4.4</td>
<td>4.0 ± 4.4</td>
<td>4.0 ± 4.4</td>
<td>4.0 ± 4.4</td>
</tr>
<tr>
<td>9 Letter</td>
<td>14.5 ± 2.5</td>
<td>31.5 ± 2.9</td>
<td>21.4 ± 1.7</td>
<td>19.4 ± 2.1</td>
<td>15.6 ± 2.3</td>
<td>14.6 ± 1.9</td>
</tr>
<tr>
<td>10 Liver</td>
<td>39.1 ± 7.2</td>
<td>√34.2 ± 7.9</td>
<td>√33.9 ± 7.9</td>
<td>37.6 ± 7.9</td>
<td>37.9 ± 9.5</td>
<td>40.3 ± 8.3</td>
</tr>
<tr>
<td>11 Pima</td>
<td>25.8 ± 2.2</td>
<td>24.1 ± 5.4</td>
<td>√22.8 ± 4.6</td>
<td>26.8 ± 3.5</td>
<td>25.6 ± 3.2</td>
<td>26.4 ± 4.4</td>
</tr>
<tr>
<td>12 Satellite</td>
<td>12.3 ± 2.1</td>
<td>x15.0 ± 3.4</td>
<td>√10.1 ± 1.7</td>
<td>x13.2 ± 1.6</td>
<td>11.8 ± 2.0</td>
<td>12.0 ± 1.9</td>
</tr>
<tr>
<td>13 Sonar</td>
<td>20.6 ± 8.5</td>
<td>24.0 ± 4.6</td>
<td>20.2 ± 9.2</td>
<td>17.8 ± 8.8</td>
<td>14.4 ± 8.1</td>
<td>15.9 ± 9.1</td>
</tr>
<tr>
<td>14 Thyroid</td>
<td>3.7 ± 1.9</td>
<td>3.7 ± 2.8</td>
<td>4.2 ± 2.5</td>
<td>4.2 ± 2.5</td>
<td>4.2 ± 2.5</td>
<td>3.7 ± 2.8</td>
</tr>
<tr>
<td>15 Vehicle</td>
<td>15.0 ± 2.9</td>
<td>x41.4 ± 3.5</td>
<td>x30.3 ± 3.7</td>
<td>x32.3 ± 3.8</td>
<td>x35.6 ± 3.3</td>
<td>x34.8 ± 5.5</td>
</tr>
<tr>
<td>16 Vowel</td>
<td>12.4 ± 3.9</td>
<td>x25.5 ± 2.8</td>
<td>√9.7 ± 1.9</td>
<td>√6.2 ± 1.9</td>
<td>√3.0 ± 1.7</td>
<td>√2.4 ± 1.1</td>
</tr>
<tr>
<td>17 Waveform</td>
<td>15.8 ± 2.0</td>
<td>x18.9 ± 1.4</td>
<td>x18.8 ± 1.9</td>
<td>x20.3 ± 2.2</td>
<td>x22.3 ± 2.3</td>
<td>x18.9 ± 1.6</td>
</tr>
<tr>
<td>18 Wine</td>
<td>0.6 ± 1.7</td>
<td>1.7 ± 2.6</td>
<td>1.1 ± 2.3</td>
<td>1.1 ± 2.2</td>
<td>0.0 ± 0.0</td>
<td>0.6 ± 1.7</td>
</tr>
<tr>
<td>Average</td>
<td>16.7</td>
<td>18.8</td>
<td>16.4</td>
<td>17.1</td>
<td>16.8</td>
<td>16.7</td>
</tr>
<tr>
<td>Rank</td>
<td>3.06</td>
<td>3.94</td>
<td>2.72</td>
<td>3.72</td>
<td>3.22</td>
<td>3.17</td>
</tr>
</tbody>
</table>

significance level. When QDA obtains a significative better or worse error it is represented in Table 4 by “✓” or “✗” symbols, respectively. For example, in the HAYES data set, flexible NB has obtained a predictive accuracy significantly better at α = 10% than QDA, which has obtained the worst score. Besides, for each data set, the algorithm with the best average score is shaded with grey as the best.

The third analysis is a hypothesis test in order to check if an algorithm is better or worse than the QDA overall data sets. A Wilcoxon paired test at α = 10% has been done, using the error estimated at each data set with the stratified 10-fold cross-validation process, as proposed in [10]. When QDA obtains a significative better or worse error it is represented in the average row of the Table 4 by “✓” or “✗” symbols, respectively.

The fourth analysis summarized in Table 4 ranks all the classifiers at each data set by means of their mean scores. The Rank row shows, for each classifier, the rank average across all the data sets. For example, the average rank of flexible NB is 3.94 across all domains. Taking into account the results presented in Table 4, the following conclu-
sions can be obtained:

- The flexible classifiers presented obtain quite competitive result compared with QDA benchmark. QDA does not obtain statistically significant better errors than any other flexible classifier, at $\alpha = 10\%$ in a Wilcoxon paired test over all data sets, as recently proposed in [10].

- fTAN obtain the best results:
  - fTAN achieves the best average error across all data sets, even better than the QDA.
  - It obtains seven significantly better errors against QDA across all the data sets, the greater number among the flexible classifiers.
  - It obtains the highest number of best errors at different data sets.
  - It obtains the best ranking average across all data sets, even better than the QDA.

- fNB achieves the poorest results:
  - It has the poorest mean among flexible classifiers presented.
  - It obtains the worst ranking average across all data sets.
  - It obtains seven significantly worse errors against the QDA, the greater number among the flexible classifiers, and only two significantly better errors, the lowest number among the flexible classifiers.

- Also, taking into account the results presented in Table 3, the fNB and fTAN algorithms seem to perform better than mNB and mTAN algorithms and, specially, than gNB and gTAN:
  - Flexible classifiers obtain best average errors than their multinomial and Gaussian based versions.
  - fNB shows better errors than gNB at $\alpha = 0.5\%$ and fTAN better than gTAN at $\alpha = 7.2\%$ in a Wilcoxon paired test across all the domains, as proposed in [10].

Besides, it must be noted that the thresholded classifier induction algorithms seem to obtain classifiers that perform at least as good as the original algorithms, at each structural complexity level. This fact motivates the use of other structural searching algorithms which could induct incomplete structures.

5.2.2 Comparative tables of the error
The last analysis of the errors estimated consists of a comparison between every pair of classifier induction algorithms, at each data set, using a Wilcoxon paired hypothesis test at $\alpha = 5\%$, based on the results of the stratified 10-fold cross-validation process. Table 5 contains a summary of this study.
Table 5: Comparative table of the estimated error of flexible classifiers: summary of the times that each algorithm has won and lost with respect to another one.

<table>
<thead>
<tr>
<th>Winners \ Losers</th>
<th>fNB</th>
<th>fTAN</th>
<th>f2DB</th>
<th>fd.5DB</th>
<th>fBAN</th>
<th>Won</th>
</tr>
</thead>
<tbody>
<tr>
<td>fNB</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>fTAN</td>
<td>4</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>27</td>
</tr>
<tr>
<td>f2DB</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>fd.5DB</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>fBAN</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Lost</td>
<td>17</td>
<td>5</td>
<td>15</td>
<td>16</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Won/Lost</td>
<td>0.70</td>
<td>5.40</td>
<td>0.73</td>
<td>0.56</td>
<td>0.57</td>
<td></td>
</tr>
</tbody>
</table>

We say that an algorithm has *won* once if it obtains better results in a data set than another algorithm at $\alpha = 5\%$ significance level in the Wilcoxon paired test. On the other hand, an algorithm has *lost* once when it obtains a worse result under the same conditions. Table 5 contains, for each algorithm, the number of times that it has won and lost against each other algorithm. The *lost* row and the *won column* show the total number of times that each algorithm has lost or won against the others. The *won/lost row* shows, for each algorithm, the ratio between the total of times it won and the total of times it lost. For example, bearing in mind the predictive accuracy, fTAN has won four times and lost three times against fNB. The total number of times that fTAN has won and lost are 27 and five respectively, and the won/lost ratio is 5.40 (see Table 5).

Looking at the results presented in Table 5 the following conclusions can be extracted:

- **fTAN classifier shows the most accurate behavior:**
  - fTAN obtains the best won/lost ratio.
  - It has won more times and lost fewer against any other flexible classifier.

- **f2DB, fd.5DB and fBAN perform quite similarly:**
  - They obtain quite similar won/lost ratios.
  - They have won and lost almost the same times against fNB and fTAN classifiers. Moreover, they have won and lost in the same data bases against fNB and fTAN.

- **fNB classifier seems to obtain the poorest results:**
  - Even fNB has a won/lost ratio better than fd.5DB and fBAN, it has won three times and lost five times against them.
5.2.3 Bias plus variance decomposition of the expected error

In this section, we perform the bias plus variance decomposition in order to study, in each data set, the behavior of the expected error (expected misclassification rate [31]) of the flexible classifiers presented. We define the expected error of zero-one loss function as:

\[ E_M = \int f(x) \sum_{c=1}^{r} p(c|x)(1 - p_M(c|x)) \, dx \]  

(14)

The bias-variance decomposition of the expected error can be useful to explain the behaviors of different algorithms [59]. The concept of bias-variance decomposition was introduced to machine learning for mean squared error by German et al. [22]. Later versions for zero-one-loss functions were given by Friedman [18], Kohavi and Wolpert [31], Domingos [11] and James [26].

The decompositions have been performed following Kohavi and Wolpert’s proposal [31] with parameters \( N = 10 \) and \( m = 1/3|BD| \), where \( N \) is the number of training sets, \( m \) is its size and \( |BD| \) is the size of the data set. We have set \( N = 10 \) because the bias estimation is precise enough for this value (see Figure 1 of [31]), and due to its acceptable computational cost. We have set \( m = 1/3|BD| \) to ensure a minimum training set size which could avoid overfitting problems. Kohavi and Wolpert [31] 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 data sets have been selected for the experimentation: Balance, Block, Image, Letter, Pima, Satellite, Vehicle, Vowel and Waveform (see Table 2).

The bias-variance decomposition proposed in [31] is as follows:

\[ E_M = \int f(x)(\sigma^2_x + \text{bias}^2_x + \text{var}_x) \, dx \]  

(15)

where \( x \) is an instance of the test set, \( \sigma^2_x \) is the “intrinsic” target noise, \( \text{bias}^2_x \) is the square bias and \( \text{var}_x \) is the variance associated with instance \( x \). \( \text{bias}^2 = \int f(x)\text{bias}^2_x \, dx \) and \( \text{var} = \int f(x)\text{var}_x \, dx \) are the averaged squared bias and variance (or bias and variance terms of the decomposition respectively). 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 [31]. Thus, it is considered zero given the data sets selected, and therefore, the expected error of the data sets 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). On the other hand, 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
Table 6: Bias plus variance decomposition of the expected misclassification error rate.

<table>
<thead>
<tr>
<th># Data Set</th>
<th>QDA</th>
<th>fNB</th>
<th>fTAN</th>
<th>f2DB</th>
<th>fL.5DB</th>
<th>fBAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Balance</td>
<td>6.3+8.6</td>
<td>12.2+2.4</td>
<td>10.6+17.8</td>
<td>9.8+16.4</td>
<td>9.8+16.4</td>
<td>9.1+15.1</td>
</tr>
<tr>
<td>2 Block</td>
<td>4.2+2.1</td>
<td>4.5+1.6</td>
<td>4.1+1.2</td>
<td>3.9+1.3</td>
<td>3.9+1.2</td>
<td>4.0+1.2</td>
</tr>
<tr>
<td>6 Image</td>
<td>8.5+1.3</td>
<td>16.0+3.2</td>
<td>13.3+2.0</td>
<td>13.4+2.0</td>
<td>13.0+2.3</td>
<td>13.0+2.1</td>
</tr>
<tr>
<td>9 Letter</td>
<td>14.3+16.3</td>
<td>21.7+18.7</td>
<td>16.0+16.1</td>
<td>14.1+15.4</td>
<td>13.1+14.5</td>
<td>13.3+14.6</td>
</tr>
<tr>
<td>11 Pima</td>
<td>18.8+10.9</td>
<td>16.6+11.2</td>
<td>17.0+12.6</td>
<td>17.2+13.5</td>
<td>17.4+12.8</td>
<td>18.1+12.8</td>
</tr>
<tr>
<td>12 Satellite</td>
<td>9.7+4.5</td>
<td>13.7+1.2</td>
<td>9.7+1.9</td>
<td>10.0+3.3</td>
<td>9.3+2.9</td>
<td>9.9+2.7</td>
</tr>
<tr>
<td>15 Vehicle</td>
<td>8.8+7.8</td>
<td>33.6+11.9</td>
<td>21.1+14.4</td>
<td>21.0+15.1</td>
<td>22.9+16.0</td>
<td>22.9+15.9</td>
</tr>
<tr>
<td>16 Vowel</td>
<td>8.8+12.6</td>
<td>19.1+24.4</td>
<td>11.6+18.6</td>
<td>7.6+14.7</td>
<td>5.5+11.4</td>
<td>5.1+11.1</td>
</tr>
<tr>
<td>17 Waveform</td>
<td>11.7+9.1</td>
<td>13.7+1.2</td>
<td>9.7+1.9</td>
<td>10.0+3.3</td>
<td>9.3+2.9</td>
<td>9.4+3.1</td>
</tr>
<tr>
<td>Average</td>
<td>10.1+9.6</td>
<td>16.8+10.4</td>
<td>12.6+9.6</td>
<td>11.9+9.4</td>
<td>11.6+8.9</td>
<td>11.6+8.7</td>
</tr>
</tbody>
</table>

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).

Table 6 shows the results of the decomposition obtained for each classifier in Balance, Block, Image, Letter, Pima, Satellite, Vehicle, Vowel and Waveform data sets. It also includes an additional row which contains the averages for each classifier across all the data sets selected. For example fTAN obtains a $bias^2 = 10.6$ plus $var = 17.8$ decomposition for Balance, and an average decomposition across all the data sets of $bias^2 = 12.6$ plus $var = 9.6$. Taking into account the results of Table 6, the following decompositions should be highlighted:

- The behavior of the decomposition with Vehicle data set (and in a similar way for Satellite) is as follows (see Figure 9(a)):
  - Bias notably decreases in the jump from fNB to fTAN. On the other hand, fTAN, f2DB, fL.5DB and fBAN obtain quite similar bias component values.
  - Variance slightly increases with the increase of the complexity.

- The behavior of the decomposition with Vowel data set (and in a similar way for Letter and Balance) is as follows (see Figure 9(b)):
  - Bias decreases with the increase of the complexity.
  - Variance also decreases with the increase of the complexity. This is an abnormal behavior because the increment in the number of parameters decreases the error due to the sensitivity to the changes in the train set.

- Taking into account Balance, Block, Image, Letter, Pima, Satellite, Vehicle, Vowel and Waveform data sets, on average, the behavior of the
decomposition is as follows (see Figure 9(c)):

- Bias notably decreases in the jump from fNB to fTAN. On the other hand, fTAN, f2DB, fd.5DB and fBAN obtain quite similar bias components of the error. Therefore, it can be concluded that for the selected data sets that fTAN model is precise enough to estimate the underlying density of the data.

- Variance slightly decreases with the increase of the complexity, and thus, all the models seem to have a similar sensibility to changes in the train set. It could be because the number of parameters needed to model fNB and fBAN structures is very similar within the flexible classifiers (see Table 1 with $k = 0, d–1$ for fNB and fBAN structures).

![Figure 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 two representative data sets (Vehicle (a) and Vowel (b)) and for the average over data sets selected (c). The diamonds and squares represent the bias and variance components of the expected errors respectively.](image)

Generally, among flexible classifiers, those which model correlations between predictor variables seem to be more suitable for modelling 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 underlaying the data, it may be advisable to use flexible classifiers which model correlations between predictors.

6 Conclusions and future work

This work presents the conditional flexible network (CFN) paradigm for the supervised classification. A conditional flexible network is a Bayesian network which estimates the density of the continuous variables using kernel based estimators (Equation 4). CFN based classifiers (flexible classifiers), being based on a kind of Bayesian network, use the Bayes rule to classify new instances.

CFN paradigm can be considered as an alternative to Bayesian multinomial networks because they can directly handle continuous variables without needing to discretize them. Besides, it can be also considered as an extension of the conditional Gaussian network from the point of view of the flexibility of their approaches for estimating densities, because conditional flexible network uses a non-parametric kernel based density estimation instead of a parametric Gaussian one. The kernel based estimation confers better flexibility properties than Gaussian approach in order to fit non-Gaussian densities. The factorizations codified by a CFN 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 Subsection 5.1 by means of a set of four artificial domains. Flexible classifiers seems to classify better than their Gaussian approaches, and at least, equal to their multinomial versions, in the UCI data sets included. Besides, they obtain competitive results comparing to the state-of-the-art algorithms selected.

Moreover, we present the thresholded versions of the classifier induction algorithms previously adapted, which implicitly forbid some arcs to induct incomplete structures, by means of thresholds applied to the amounts of mutual information \( I(X_i, C) \) and \( I(X_i, X_j|C) \). The good results of the thresholded versions motivates the design of new structural searching algorithms for inducting incomplete structures. Flexible tree-augmented naive Bayes obtains the best estimated errors among the flexible classifiers proposed. 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 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 slightly decreases with the increase of the complexity, and thus, all the models seem to have a similar sensibility to changes in the train set. In summary, we think that, in absence of prior knowledge about the true density underlaying the data, it is generally advisable to use flexible tree-augmented naive Bayes for the supervised classification task, among the flexible classifiers presented.

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 in relaxing the strong computational cost required. On one hand, we are considering to make a case condensed, following the M-Kernel merging procedure proposed in [64] for the data stream, in order to reduce the number of train cases learned. On the other hand, we could approximate the kernel based estimation by means of a mixture of truncated exponentials following the procedure proposed in [40, 50], and use the approximation obtained to classify new instances. Both approaches reduce considerably the computational cost, from a cost related with the number of train cases to a fixed one. Both approaches will allow us to design and implement other search techniques in the space of possible structures: wrapper search [30] guided by the accuracy, or by using metaheuristics such as genetic algorithms [23] or estimation distribution algorithms [34].

7 Acknowledgments

This work has been possible thanks to the PhD grant Beca para la Formación de Investigadores 2003-06 of the Basque Government, to the projects SAIOTEK and ETORTEK of the Basque Government, and to the project TIN2005-03824 of the Spanish Ministry of Education and Science.

References


