Machine Learning, 59, 237–265, 2005
2005 Springer Science + Business Media, Inc. Manufactured in The Netherlands.

Latent Classification Models

HELGE LANGSETH helgel@math.ntnu.no
Department of Mathematical Sciences, Norwegian University of Science and Technology, N-7491 Trondheim, Norway

THOMAS D. NIELSEN tdn@cs.auc.dk
Department of Computer Science, Aalborg University, Fredrik Bajers Vej 7E, DK-9220 Aalborg Ø, Denmark

Editors: Pedro Larrañaga, Jose A. Lozano, Jose M. Peña and Iñaki Inza

Abstract. One of the simplest, and yet most consistently well-performing set of classifiers is the Naïve Bayes models. These models rely on two assumptions: (i) All the attributes used to describe an instance are conditionally independent given the class of that instance, and (ii) all attributes follow a specific parametric family of distributions. In this paper we propose a new set of models for classification in continuous domains, termed latent classification models. The latent classification model can roughly be seen as combining the Naïve Bayes model with a mixture of factor analyzers, thereby relaxing the assumptions of the Naïve Bayes classifier. In the proposed model the continuous attributes are described by a mixture of multivariate Gaussians, where the conditional dependencies among the attributes are encoded using latent variables. We present algorithms for learning both the parameters and the structure of a latent classification model, and we demonstrate empirically that the accuracy of the proposed model is significantly higher than the accuracy of other probabilistic classifiers.

Keywords: classification, probabilistic graphical models, Naïve Bayes, correlation

1. Introduction

Classification is the task of predicting the class of an instance from a set of attributes describing that instance, i.e., to apply a mapping from the attribute space into a predefined set of classes. When learning a classifier we seek to find such a mapping based on a database of labelled instances. Classifier learning, which has been an active research field over the last decades, can therefore be seen as a model selection process where the task is to find the single model, from some set of models, with the highest classification accuracy.

Recently, research on classifier learning has primarily focused on domains where all variables are either discrete or have been discretized during the preprocessing phase. However, several important classification tasks (such as the analysis of gene expressions, online monitoring of process systems, etc.) are continuous by nature, and by discretizing the continuous attributes we risk to loose information which is relevant for the classification task (Friedman et al., 2000).[1]

For both continuous and discrete domains, one of the simplest and yet most consistently well-performing set of classifiers is the Naïve Bayes models (Duda & Hart, 1973). Generally, in the Naïve Bayes models all attributes are assumed to be conditionally independent given the class variable. This assumption is clearly violated in many real world domains, and this shortcoming has inspired several extensions of the model. One such extension is the tree augmented Naïve Bayes model for continuous domains (Friedman, Goldszmidt, & Lee, 1998); in this model, the conditional dependencies are modelled using a tree structure among the attributes. Alternatively, instead of allowing a more general correlation structure in the model, another approach is to transform the data s.t. the transformed data abides to the independence assumption in the model. This approach has, e.g., been pursued by Bressan and Vitrià (2002), who consider applying a class conditional Independent Component Analysis (Hyvärinen, Karhunen, & Oja, 2001) and then using a Naïve Bayes model on the transformed data.

In this paper we take the former approach and propose a new class of models for continuous domains, termed *latent classification models* (LCMs). In the latent classification model, the conditional dependencies among the attributes are encoded using latent variables, which allows the model to be interpreted as a combination of a Naïve Bayes model and a mixture of factor analyzers. We propose an algorithm for learning both the parameters and the structure of an LCM, and experimental results demonstrate that the accuracy of the proposed model is significantly higher than the accuracy of other probabilistic classifiers.

2. Continuous Bayesian classifiers

In the context of classification, we shall use $\{X_1, \ldots, X_n\}$ to denote the attributes describing the instances to be classified; as we focus on continuous domains it follows that $sp(X_i) = \mathbb{R}$, for all $1 \leq i \leq n$. Furthermore, we shall use Y to denote the (discrete) class variable, where $sp\,(Y)$ is the set of possible classes (for notational convenience we also assume that $sp\,(Y) = \{1, 2, \ldots, |sp\,(Y)\,|\}$).

When doing classification in a probabilistic framework, a Bayes optimal classifier will classify a new instance $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ to class y^* according to:

$$y^* = \arg\min_{y \in sp(Y)} \sum_{y' \in sp(Y)} L(y, y')P(y' \mid \boldsymbol{x}),$$

where $L(\cdot, \cdot)$ is the loss-function (see, e.g., Ripley, 1996; Mitchell, 1997; McLachlan, 2004). An example of such a loss-function is the 0/1-loss, where $L(\cdot, \cdot)$ is defined s.t. $L(y, y') = 0$ if $y = y'$ and 1 otherwise. When learning a probabilistic classifier, the task is therefore to learn the probability distribution $P(Y = y \mid \boldsymbol{X} = \boldsymbol{x})$ from a set of N labelled training samples $\mathcal{D}_N = \{\boldsymbol{D}_1, \ldots, \boldsymbol{D}_N\}$, where $\boldsymbol{D}_i = (x_1^i, \ldots, x_n^i, y^i)$ is a configuration over the attributes together with a class label.

An immediate approach for learning $P(Y = y \mid \boldsymbol{x})$ is to use a standard algorithm for learning Bayesian networks (BNs) over continuous domains, see, e.g., Geiger and Heckerman (1994) and Monti and Cooper (1999b). These learning algorithms basically give each BN (visited during structure search) a score based on how accurately it represents the database. Examples of such scoring functions include the Bayesian metric for Gaussian networks (Geiger & Heckerman, 1994) as well as score functions based on a penalized log-likelihood, e.g., the Minimum Description Length principle (Rissanen, 1978; Lam & Bacchus, 1994) and the Bayesian Information Criterion (Schwarz, 1978). However, such global scoring functions are not necessarily well-suited for learning a classifier (Greiner,

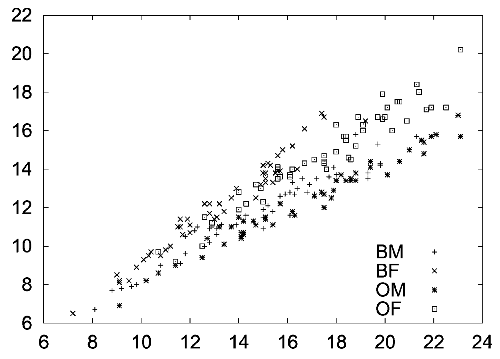


Figure 1. Plot of *Width of the frontal lip* vs. *Rear width* for each of the four classes in the `crabs` domain. The conditional correlation between the two attributes given the class is evident.

Grove, & Schuurmans, 1997). One way around this problem is to estimate a classifier's accuracy on unseen data using the wrapper approach, i.e., by applying cross-validation (over the *training data*) and using the estimated accuracy as the score of the classifier (Kohavi & John, 1997). The use of the wrapper approach, however, comes at the cost of higher computational complexity.

In order to relieve this potential complexity problem, one approach is to focus on a particular sub-class of BNs instead of BNs in general; these sub-classes are usually defined by the independence statements they encode. For example, the Naïve Bayes models (Duda & Hart, 1973) is a set of simple models that have shown to provide very good classification accuracy in many domains. In a Naïve Bayes model the attributes are assumed to be conditionally independent given the class, however, as also shown in the following example, this independence assumption is clearly violated in many domains.

Example 1. The crabs dataset origins from Campbell and Mahon (1974), where each crab is characterized by the following five attributes: *Width of the frontal lip*, *Rear width*, *Length along the midline*, *Maximum width of the carapace* and *Body depth*.

The goal of the classification task is to predict the correct sex and color of a crab. The possible classes are: Blue male (BM), Blue female (BF), Orange male (OM) and Orange female (OF). A plot of the *Width of the frontal lip* vs. *Rear width* for each of the four classes is shown in Figure 1. From the figure it is evident that there is a strong conditional correlation between the attributes given the class; similar results are also obtained when considering other pairs of attributes. One implication of these conditional correlations is that the Naïve Bayes classifier will perform poorly in this domain. This is also confirmed by our experimental study (Section 5), where the Naïve Bayes classifier achieved an accuracy of only 39.5%.

In general, existing techniques for handling conditional dependencies among the attributes can roughly be characterized as either (i) using a set of models that admits a more general correlation structure or (ii) relying on a preprocessing of the data. As an example of the former, Friedman, Goldszmidt, and Lee (1998) propose an extension of the

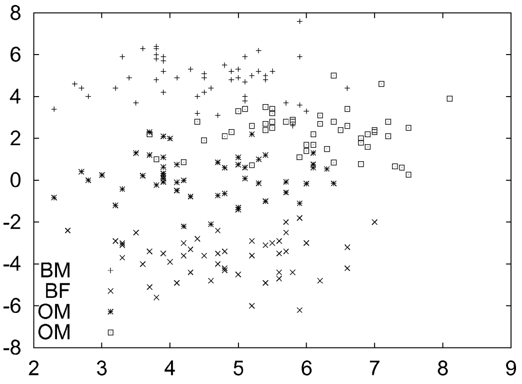


Figure 2. Plot of the expected values of the first two factors from a PCA of the crabs database. Note that the conditional correlation between attributes given the class is reduced significantly. In the spirit of Bressan and Vitrià (2002), one PCA model is fitted per class.

Tree Augmented Naïve Bayes (TAN) model (Friedman, Geiger, & Goldszmidt, 1997) to facilitate continuous domains. In the TAN framework, each attribute is allowed to have at most one parent besides the class variable. Alternatively, Gama (2000) propose the so-called linear Bayes model, where all continuous attributes are combined and associated with a multivariate Gaussian to encode the conditional dependencies; discrete attributes are treated as in the Naïve Bayes model.

Instead of working with a more general correlation structure in the model, another approach for handling the conditional dependencies is to preprocess the data before learning the classifier. For example, if the attributes are transformed into a vector of variables that are conditionally independent given the class variable, then a Naïve Bayes classifier can be learned based on the transformed data. One approach for doing this type of preprocessing is to employ a Principal Component Analysis (PCA) on the database (see, e.g., Kendall 1980); the factors in the PCA are by definition independent (assuming a multivariate Gaussian distribution over the attributes). Thus, if we produce a database where the attributes, describing a given object, are replaced by the expected values of the factors, then this database would describe the objects consistent with the Naïve Bayes classifier. Note that one PCA must be made for each class in order to obtain *conditional* independence.[2]

Example 2. Consider again the crabs domain described in Example 1. By employing a PCA (for each class) on the database, we can transform the attributes into a collection of mutually independent factors; Figure 2 shows a plot of the expected values of the first two factors. Observe that the conditional correlation between the attributes given the class is reduced significantly. In fact, the Naïve Bayes classifier achieved an accuracy of 94.0% when working on the transformed data (Section 5).

The idea of transforming the data has been pursued by, e.g., Hinton, Dayan, and Revow (1997) for the task of recognizing handwritten digits. More specifically, Hinton, Dayan, and Revow (1997) employs a (class conditional) mixture of PCAs as well as a mixture of

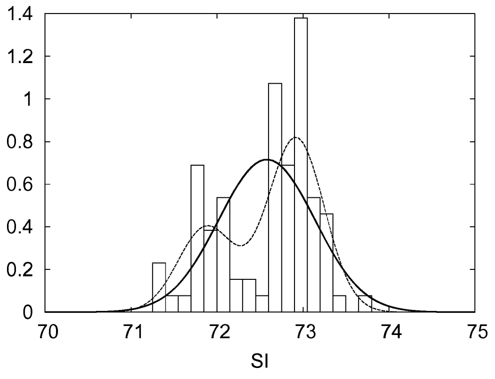


Figure 3. The *Silicon* contents in several samples from the "float-processed" class taken from the glass2 domain is shown together with the maximum likelihood estimate of Gaussian (solid line) and the mixture of Gaussians maximizing the BIC score (dashed).

factor analyzers (FAs) for transforming the data; one of the main differences between the two methods is that PCA tries to model as much of the variance (among the attributes) as possible with the least amount of components, whereas FA tries to explain as much of the correlation with as few factors as possible. In general, the transformation of the data relies on an a priori selection of the dimensionality of the transformed space. Minka (2000) describes a method for selecting the dimensionality for PCA by treating it as a task of density estimation, where the dimensionality is chosen according to a Bayesian scoring function.

Traditionally, the Naïve Bayes classifier makes the assumptions that the continuous attributes are generated by a specific parametric family of distributions (usually Gaussians). Even though assuming Gaussians may provide an adequate approximation in some domains, there are also domains where this assumption is not appropriate.

Example 3. Figure 3 shows a histogram of the *Silicon* contents in the samples from the class of "float-processed" glass; the data have been taken from the glass2 domain, which is available from the UCI repository (Blake & Merz, 1998). The maximum likelihood Gaussian approximation (thick line) is given together with a mixture of two Gaussians (dashed line). Although the fit of the Gaussian is rather poor, the mixture of Gaussians is able to make a reasonable approximation using only two components. The accuracy of this approximation also has an impact on classification accuracy: Whereas the Naïve Bayes classifier (assuming Gaussian distributions) performs rather poorly on this domain with 62.0% accuracy, the quality is improved to 78.0% when Mixtures of Gaussians are used. (See also Section 5.)

To avoid having to make the Gaussian assumption, one possibility would be to discretize the continuous attributes (see, e.g., Fayyad & Irani, 1993), however, this may also result in a loss of information which can affect classification accuracy (Friedman et al., 2000). Alternatively, John and Langley (1995) propose to substitute the Gaussian distribution with a kernel density estimation using a Gaussian kernel. This approach can be seen as a special case of having a mixture of Gaussians associated with each attribute, where the mixture components are given uniform weights and the number of mixture components corresponds to the number of cases in the database. Along the same lines, Monti and Cooper (1999a) propose the finite-mixture-augmented Naïve Bayes that combines the Naïve Bayes model and the finite mixture model. The mixture variable in this model ensures a more general probability distribution over the attributes (i.e., a mixture of Gaussians) and also accounts for some of the dependencies among the attributes. In the same spirit, Vilalta and Rish (2003) address the problem of having the classes distributed into multiple regions in the attribute space. In particular, they propose to decompose each class into a collection of clusters, where each cluster is considered a class on its own; thus, ensuring that a class will not be distributed in multiple regions. Finally, Friedman, Goldszmidt, and Lee (1998) suggest an approach to overcome the drawback of either discretizing a continuous attribute or having a pure Gaussian distribution associated with it; specifically, they propose to have a dual representation of an attribute by including both the continuous attribute as well as its discretized version.

3. Latent classification models

As described in Section 2, one approach for handling the conditional dependencies between the attributes is to make a transformation of the data before learning the classifier, e.g. by applying either a (mixture of) PCAs or FAs. For example, by applying a FA we make a dimensionality reduction of the covariance structure of the data. This reduction is performed by modelling the attributes, $\boldsymbol{X}$, using a q-dimensional vector of *factor variables* $\boldsymbol{Z}$ (with $q \leq n$) and by assuming the generative model:

$$\boldsymbol{X} = \boldsymbol{L}\boldsymbol{Z} + \boldsymbol{\epsilon},$$

where $\boldsymbol{L}$ is the regression matrix, $\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Theta})$ is an n-dimensional random variable with diagonal covariance matrix $\boldsymbol{\Theta}$. In this model, the factor variables model the dependencies among the attributes, and $\boldsymbol{\epsilon}$ can be interpreted as the sensor noise associated with the attributes.

Starting from this generative model, one can be tempted to try the following approach for learning a Naïve Bayes classifier: (i) Learn a factor analyzer from the data, (ii) for each data-point $\boldsymbol{D}_i = (\boldsymbol{x}^i, y^i)$ let the expectation $\mathbb{E}(\boldsymbol{Z} \mid \boldsymbol{X} = \boldsymbol{x}^i)$ represent observation $\boldsymbol{x}^i$ in the database, and (iii) learn a Naïve Bayes classifier using the transformed dataset generated in step (ii). This approach produces latent variables that are unconditionally independent ($Z_i \perp\!\!\!\perp Z_j$, $i \neq j$), but (typically) conditionally dependent ($Z_i \not\perp\!\!\!\perp Z_j \mid Y$). That is, the independence statements are not consistent with the Naïve Bayes model. Moreover, when we substitute the actual observations with the expected values of the factor variables in step (ii), we disregard information about the uncertainty in the generative model.

3.1. The linear generative model

In order to avoid the shortcomings of the method described above, we propose a new set of models, termed *latent classification models* (LCMs). Roughly, an LCM can be seen

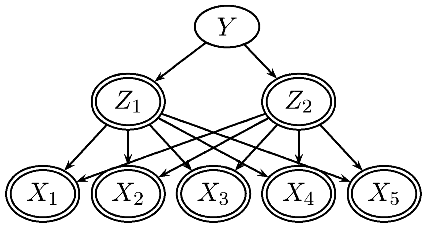


Figure 4. A graphical representation of an LCM with $n = 5$ attributes and $q = 2$ latent variables. Note that all latent variables as well as the attributes are continuous (and thus drawn with double line).

as combining an FA with the Naïve Bayes model. By relying on FA rather than PCA we have an explicit generative model (that is amenable to statistical analysis), which focuses on the correlation structure among the attributes instead of the variance in general. As we shall see below, the latter property also ensures a natural coupling to the NB model. More specifically, in the proposed model the factor variables appear as children of the class variable in the graphical representation of the model (see Figure 4). This also implies that an LCM is an instance of a conditional Gaussian network, where the variables can be partitioned into three disjoint subsets: $\{Y\}$ is the class variable, $\mathcal{X}$ is the set of attributes, and $\mathcal{Z}$ is a set of latent variables ($\mathcal{Z}$ has the same role as the factor variables in an FA). In an LCM the class variable appear as root, $\mathcal{X}$ constitute the leaves with only latent variables as parents, and the latent variables are all internal having the class variable as parent and the attributes as children. Note that not all of the independence assumptions underlying the factor analysis model carry over to the LCM. In particular, the latent variables in an LCM are conditionally independent given the class, but marginally dependent.

For the quantitative part of the LCM, we assume that:

- The class variable, Y, follows a multinomial distribution, that is, $P(Y = j) = p_j$, where $1 \leq j \leq |sp(Y)|$, $p_j \geq 0$ and $\sum_{j=1}^{|sp(Y)|} p_j = 1$.
- Conditionally on $Y = j$, the latent variables, follow a Gaussian distribution with $\mathbb{E}[\boldsymbol{Z} \mid Y = j] = \boldsymbol{\mu}_j$ and $\text{Cov}(\boldsymbol{Z} \mid Y = j) = \boldsymbol{\Gamma}_j$. Moreover, it follows from the model structure that $\boldsymbol{\Gamma}_j$ has to be diagonal (meaning that $Z_k \perp\!\!\!\perp Z_l \mid \{Y = j\}$ for all $k \neq l$ and for all $j = 1, \ldots, |\, sp(Y)\,|$). Note that as opposed to the generative model underlying factor analysis, we do not assume that $\mathbb{E}(\boldsymbol{Z} \mid Y) = \boldsymbol{0}$ and $\text{Cov}(\boldsymbol{Z} \mid Y) = \boldsymbol{I}$.
- Conditionally on $\boldsymbol{Z} = z$, the variables $\boldsymbol{X}$ follow a Gaussian distribution with $\mathbb{E}(\boldsymbol{X} \mid \boldsymbol{Z} = z) = \boldsymbol{L}z$ and $\text{Cov}(\boldsymbol{X} \mid \boldsymbol{Z} = z) = \boldsymbol{\Theta}$, where $\boldsymbol{\Theta}$ must be diagonal to comply with the model structure.

From the description above it follows that the model assumes a *linear* mapping $\boldsymbol{L} : \mathbb{R}^q \mapsto \mathbb{R}^n$ that takes a vector of (unobservable) variables and maps it to a vector of (observable) attributes; $\boldsymbol{\Theta}$ can be interpreted as the precision of the mapping from the latent variables (corresponding to the factor space) to the attribute space.[3] Moreover, analogously to standard factor analysis we also assume that $\boldsymbol{\Theta}$ is diagonal, meaning that $X_i \perp\!\!\!\perp X_j \mid \boldsymbol{Z}$, for all $i \neq j$.

It is important to emphasize that an LCM does not encode a *class-conditional* factor analysis in the spirit of, e.g., Bressan and Vitrià (2002). Instead the attributes are conditionally independent of the class variable given the factor variables ($\boldsymbol{X} \perp\!\!\!\perp Y \mid \boldsymbol{Z}$), and the factor variables are directly dependent on the class variable. Moreover, the same mapping, $\boldsymbol{L}$, from the latent space to the attribute space is used for all classes. Thus, the relation between the class variable and the attributes is conveyed by the latent variables, i.e., the latent variables summarizes the information from the attributes which is relevant for classification.

When investigating the expressibility of LCMs, it follows that if the attributes (conditioned on the class variable) follow a Gaussian distribution, then there exists an LCM which can encode the joint distribution over $(Y, \boldsymbol{X})$. More formally we have (the proof is given in Appendix A):

Proposition 4. *Assume that Y follows a multinomial distribution, and that $\boldsymbol{X} \mid \{Y = j\} \sim \mathcal{N}(\boldsymbol{\alpha}_j, \boldsymbol{\Sigma}_j)$ for a given set of parameters $\{\boldsymbol{\alpha}_j, \boldsymbol{\Sigma}_j\}_{j=1}^{|sp(Y)|}$. Assume that rank$(\boldsymbol{\Sigma}_j) = n$ for at least one j. Then the joint distribution $(Y, \boldsymbol{X})$ can be represented by an LCM with $q \leq n \cdot \mid sp(Y) \mid$.*

Note that Proposition 4 only says that there is a $q \leq n \cdot \mid sp(Y) \mid$, which ensures that we can encode the joint distribution over $(Y, \boldsymbol{X})$ by means of an LCM. However, it does not state how to identify the minimal value of q for which this can be done. Finding a suitable value for q is considered part of the classifier learning, and will be addressed in Section 4.[4]

3.2. The non-linear generative model

In the (linear) model presented above, one of the main assumptions is that there exists a linear mapping, $\boldsymbol{L}$, from the factor space to the attribute space.[5] Consequently, conditionally on $Y = j$, the attributes are assumed to follow a multivariate Gaussian distribution with mean $\boldsymbol{L}\boldsymbol{\mu}_j$ and covariance matrix $\boldsymbol{L}\boldsymbol{\Gamma}_j\boldsymbol{L}^T + \boldsymbol{\Theta}$. However, even though this provides a sufficiently accurate approximation to the true probability distribution in many domains, there are also domains where this does not hold.

Example 5. Consider again the dataset glass2, which was examined in Example 3. By investigating the empirical probability distribution of the variables *Silicon* and *Potassium* for the "float-processed" class, we get the empirical density function depicted in Figure 5. From the contour lines it clearly follows that the process which has generated the data does not follow a multivariate Gaussian distribution. Hence, the proposed linear LCM does not provide a reasonable model for the domain; this was also confirmed by our experiments, where the model achieved an accuracy of 66.9% (see also Section 5).

In order to extend the expressibility of the linear LCMs (see Proposition 4) we propose a natural generalization, termed *non-linear LCMs*. Intuitively, to the extent that the linear LCM can be seen as a combination of an FA and a Naïve Bayes classifier, the non-linear LCM can be seen as combining a mixture of FAs with a Naïve Bayes classifier. More formally, for a mixture of FAs we have (i) a mixture variable M governing the mixture distribution, and (ii) a (generalized) FA for each mixture component $M = m$:

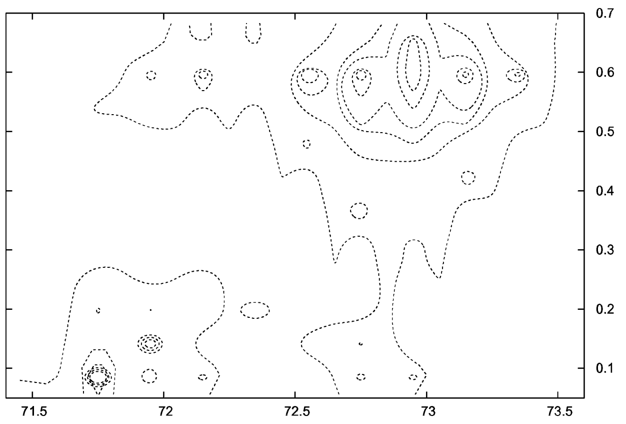


Figure 5. The figure depicts the empirical distribution function for the variables *Silicon* (x-axis) and *Potassium* (y-axis) conditioned on the class "float-processed" in the glass2 domain. From the plot it is evident that the process, which has generated the data, does not follow a Gaussian distribution.

$$\boldsymbol{X} \mid \{M = m\} = \boldsymbol{L}_m \boldsymbol{Z} + \boldsymbol{\eta}_m + \boldsymbol{\epsilon}_m.$$

Thus, for a mixture of FAs the regression matrix, $\boldsymbol{L}_m$, as well as the noise vector, $\boldsymbol{\epsilon}_m \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Theta}_m)$, varies with the mixture components.[6] Moreover, as opposed to standard factor analysis, the data mean is modelled explicitly through $\boldsymbol{\eta}_m$. Informally, by including $\boldsymbol{\eta}_m$ in the model we can focus on different parts of the attribute space when learning the different mixture components.

Based on the mixture of FAs, we define a non-linear LCM as a conditional Gaussian network, where the variables can be partitioned into four disjoint subsets: $\{Y\}$ is the class variable, $\{M\}$ is the mixture variable, $\mathcal{X}$ is the set of attributes and $\mathcal{Z}$ is a set of latent variables. The structure of a non-linear LCM is identical to the structure of a linear LCM, except that we also have the mixture variable M as an internal node having Y as parent and with all variables in $\mathcal{X}$ as children (see Figure 6). For the quantitative part of the non-linear LCM we assume that:

- The mixture variable, M, follows a multinomial distribution, i.e., $P(M = m|Y = j) = p_{m,j}$, where $1 \leq m \leq |sp(M)\ |$, $p_{m,j} \geq 0$ and $\sum_{m=1}^{|sp(M)|} p_{m,j} = 1$, for all $1 \leq j \leq |sp(Y)\ |$.
- Conditionally on $\{\boldsymbol{Z} = \boldsymbol{z}, M = m\}$, the variables $\boldsymbol{X}$ follow a Gaussian distribution with moments $\mathbb{E}(\boldsymbol{X} \mid \boldsymbol{z}, m) = \boldsymbol{L}_m \boldsymbol{z} + \boldsymbol{\eta}_m$ and $\mathrm{Cov}(\boldsymbol{X} \mid \boldsymbol{z}, m) = \boldsymbol{\Theta}_m$, where $\boldsymbol{\Theta}_m$ is assumed to be diagonal for all $1 \leq m \leq |\ sp(M)\ |$.

Analogously to the linear LCM, the class variable, Y, follows a multinomial distribution and $\mathbf{Z}$ follows a conditional Gaussian distribution with $\mathbb{E}[\boldsymbol{Z} \mid Y = j] = \boldsymbol{\mu}_j$ and $\mathrm{Cov}(\boldsymbol{Z} \mid Y = j) = \boldsymbol{\Gamma}_j$. Observe that a non-linear LCM is a proper generalization of a linear LCM in the sense that with $|\ sp(M)\ | = 1$ the non-linear LCM reduces to a linear model. Thus, in the

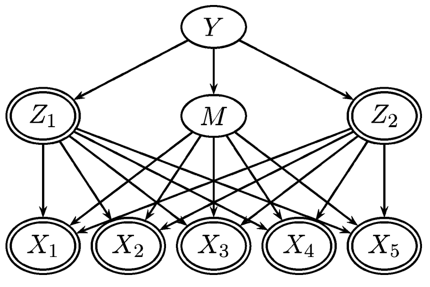


Figure 6. A graphical representation of a non-linear LCM with $n = 5$ attributes and $q = 2$ latent variables. Note that if $| sp(M) | = 1$, then the model simply corresponds to a linear LCM.

remainder of this paper, when simply referring to an LCM we mean the general non-linear model which include the linear model as a special case.

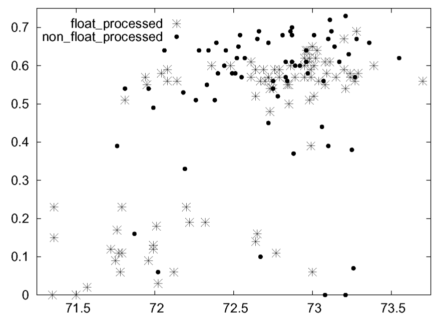
Another way to consider the non-linear LCM is provided by Ghahramani and Hinton (1996) in the related context of learning mixtures of FAs. In particular, Ghahramani and Hinton (1996) note that a mixture of FAs concurrently performs clustering (the Gaussian mixture model) and, within each cluster, local dimensionality reduction (factor analysis). This combination has the attractive property that it facilitates (among other things) dimensionality reduction to be performed within each cluster, thereby taking into account that different variables may be correlated within different clusters. Analogously, we can interpret the non-linear LCM as concurrently performing clustering and, within each cluster, local classification.[7]

Finally, as a generalization of Proposition 4 we have the following proposition which formalizes the expressibility of the model (the proof is given in Appendix A):

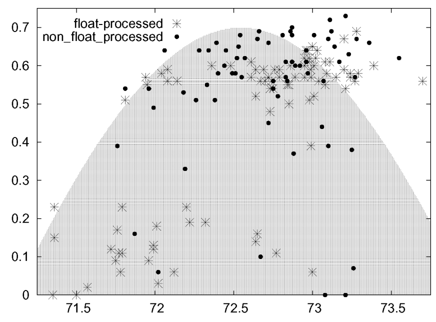
Proposition 6. *Assume that Y follows a multinomial distribution, and that* $\boldsymbol{X} \mid \{Y = j\} \sim \mathcal{P}(\boldsymbol{\Psi}_j)$, *where* $\mathcal{P}(\cdot)$ *is a distribution function over* $\mathbb{R}^n$ *and* $\{\boldsymbol{\Psi}_j\}_{j=1}^{|sp(Y)|}$ *are the parameters. Then the joint distribution for* $(Y, \boldsymbol{X})$ *can be approximated arbitrarily well by an LCM model.*

Example 7. Once again, consider the glass2 domain, which was also examined in Examples 3 and 5. As in Example 5, we restrict our attention to the variables *Silicon* and *Potassium*, for which the data is shown in Figure 7(a). If we learn a Naïve Bayes classifier from this two-variable dataset, the decision boundary will be as in Figure 7(b). The shaded area corresponds to pairs of attribute values that will be classified as "float-processed" by this classifier. In Figure 7(c) and (d) the decision boundary of the linear LCM classifier is shown with $q = 1$ and $q = 4$ respectively. Finally, Figure 7(e) and (f) give the same results for the non-linear LCM models with ($q = 4$, $| sp(M)| = 2$) and ($q = 2$, $| sp(M)| = 40$) respectively.

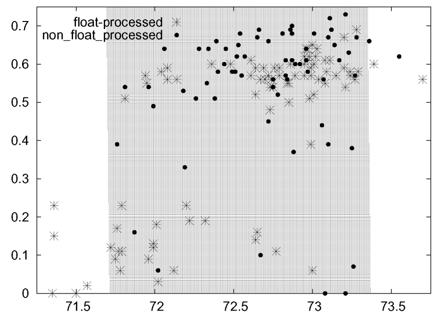
The plots clearly show how the expressive power differ between the model classes. The Naïve Bayes classifier is quite restricted, as is the LCM model with $q = 1$ in Figure 7(c). Figure 7(d) shows the effect of setting $q = 4$, meaning that the decision rule is fitted in $\mathbb{R}^4$ before it is projected down into $\mathbb{R}^2$. Introducing non-linearity to the LCM



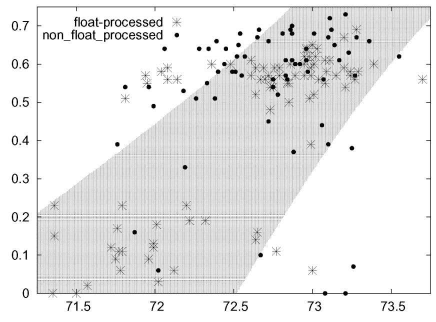
(a) Scatter-plot of the data.



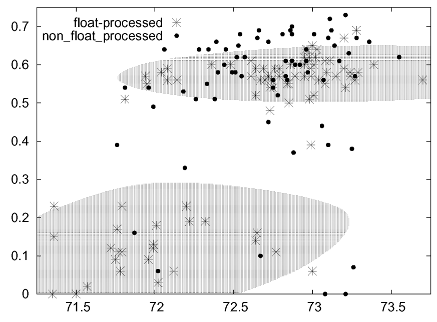
(b) Decision rule, NB



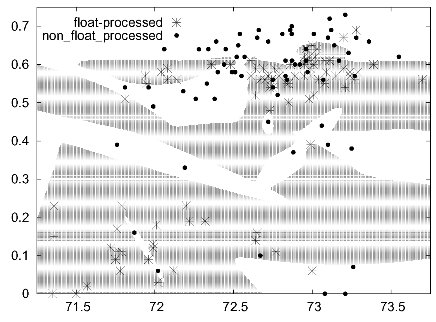
(c) Linear LCM, $q = 1$



(d) Linear LCM, $q = 4$



(e) Nonlinear LCM, $q = 4$, $|sp(M)| = 2$



(f) Nonlinear LCM, $q = 2$, $|sp(M)| = 40$

Figure 7. Decision boundaries for different classifiers working with a subset of the variables in the glass2 domain.

model enables the classifier to make more complex decision rules; Figure 7(f) indicates the "unlimited" expressibility of the non-linear LCM models. In particular, if the number of mixture components is the same as the number of instances in the data set, then the non-linear LCM is strongly related to a k-NN with $k = 1$.

4. Learning LCM models

In this section we describe a method for learning latent classification models from data. The construction of the algorithm is characterized by the score function for evaluating the quality of a model, and the search strategy for investigating the space of LCMs.

In the proposed algorithm we score a model based on its accuracy, which is estimated using the wrapper approach of Kohavi and John (1997). That is, the score is given as the average accuracy found by applying cross-validation over the training data (see also Section 2).

Considering the search strategy, we first note that the space of LCMs is defined by (i) the number of latent variables, (ii) the number of mixture components, (iii) the edge-set and (iv) the parametrization of the probability distributions. Thus, we can characterize a general learning algorithm by: (i) A systematic approach for selecting values for q and $|\ sp(M)\ |$, and, given such a pair of values, (ii) algorithms for learning the edge-set and the parameters in the model. More formally, a general LCM learning algorithm can be formulated as follows:

```
ALGORITHM 1.  Learn an LCM classifier with the *wrapper approach.*
*Input:*  A database $\mathcal{D}_N$.
*Output:* An LCM classifier.

 1. **For** possible values of $q$ and $|sp(M)|$:
    a) Partition the database into $W$ wrapper folds $\mathcal{W}_1, \ldots, \mathcal{W}_W$.
    b) **For** $w = 1, \ldots, W$:
       *i)* Learn a classifier from the dataset $\mathcal{D}_N \setminus \mathcal{W}_w$ (both edge-set
          and parameters).
       *ii)* Calculate the accuracy on the remaining training-set $\mathcal{W}_w$.
    c) Score the parameter-pair $(q, |sp(M)|)$ by the average accuracy
       obtained over the wrapper folds.
 2. Select the optimal values of $q$ and $|sp(M)|$.
 3. **Return** classifier learned with these parameters.
```

Algorithm 1. Learn an LCM classifier with the *wrapper approach.*

4.1. Learning the structure

Generally, structural learning of an LCM is comprised of three subtasks: Finding the number of latent variables, finding the number of mixture components, and determining the edge-set of the model. As described above, the selection of appropriate values for q and $|\ sp(M)\ |$ is basically performed using the wrapper approach (we discuss a possible search strategy in Section 5).[8] Thus, in what follows we focus on learning the edge-set between the latent variables, $\mathbf{Z}$, and the attributes, $\mathbf{X}$ (the other parts of the model are fixed a priori).[9] Furthermore, we insist that any edge between a latent variable Z_j and an attribute X_i must be directed from Z_j to X_i.

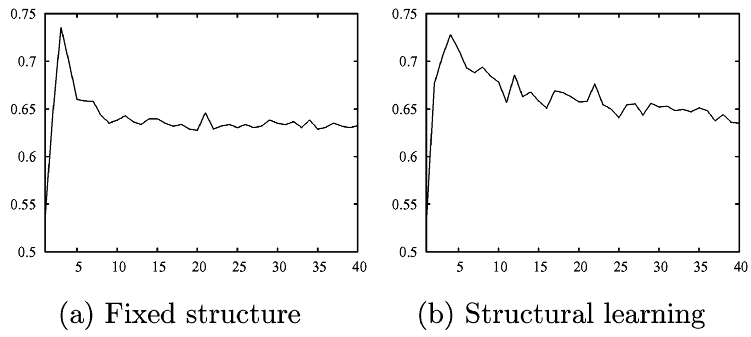


Figure 8. These plots show accuracy of the classifiers (y-axis) vs. the value of q (x-axis). The results are from the glass2 domain, which consists of 9 attributes and has 2 classes. The results are averaged over 10 runs. Part (a) shows the accuracy of our classifier with full structure ($\text{pa}(X_i) = \mathcal{Z}$ for $i = 1, \ldots, n$). Part (b) gives the same plot for a classifier where the link structure (edges from Z_j to X_i) are learned from data.

Structural learning of this edge-set is primarily motivated by the fact, that even though the number of parameters is only proportional to the number of arcs in the model, we still risk the problem of overfitting when increasing the number of latent variables. This problem is also illustrated in the following example.

Example 8. Consider again the glass2 dataset, taken from the UCI repository. Figure 8(a) shows the accuracy of a linear LCM vs. the number of latent variables included in the model; when increasing q with 1 we introduce $\mid sp(M) \mid (n + 2 \mid sp(Y) \mid) = 13$ additional parameters in the model. It is evident that by introducing too many parameters we risk reducing the classification accuracy of the model. On the other hand, Figure 8(b) shows the accuracy of the learned model when structural learning is performed (using the algorithm described below). Observe that the amount of overfitting is less apparent when structural learning is introduced, although it is still present.

Structural learning for Bayesian networks containing both continuous and discrete variables has previously been examined by several authors, see, e.g., Geiger and Heckerman (1994), Monti and Cooper (1999b), and Bøttcher (2001). Geiger and Heckerman (1994) give a decomposable score function for Gaussian networks that can be extended to certain subsets of, but unfortunately not all, hybrid networks. More precisely, Geiger and Heckerman (1994) require that if a discrete variable Δ is a parent of a continuous variable which appear in a connected subgraph (consisting of only continuous variables), then Δ should be a parent of *all* variables in that subgraph. Unfortunately, by considering the simple LCM ($n = q = 1$) depicted in Figure 9(a), it is apparent that the LCM structure do not fulfill this requirement.[10] However, because of the strong syntactical constraints on the model structure of an LCM, we are only interested in the parent set of $\boldsymbol{X}$, i.e., the edges from $\boldsymbol{Z}$ to $\boldsymbol{X}$. Thus, we do not really require the score of structure (a), but rather the *score difference* when structures (a) and (b) are compared. Moreover, since we have complete data (or, as in our case, data "completed" by the SEM algorithm (Friedman, 1998)) we can change the parent sets of the other variables in the model and still compute

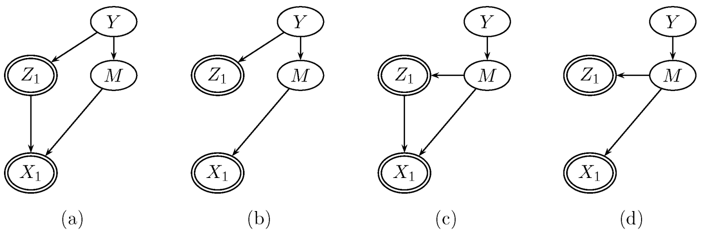


Figure 9. The structure in part (a) cannot be evaluated directly; Z_1 and X_1 are continuous, whereas Y and M are discrete. However, as we are only interested in the presence ofthe link from Z_1 to X_1 in structure (a), we do not need the score of this structure, but rather the score difference between structures (a) and (b). Due to the factoring of the score function, this is identical to the score difference for the structures(c) and (d). Since these structures can be scored, we can draw conclusions about whether structure (a) or (b) scores highest.

the score difference correctly (by simply exploiting that the score function decomposes). In particular, both of the structures in parts (c) and (d) of Figure 9 can be evaluated using the score function of Geiger and Heckerman (1994), and the score difference between these structures is the same as the score difference between structures (a) and (b). Thus, we can use the framework of Geiger and Heckerman (1994) to learn the edge-set from the latent variables $\boldsymbol{Z}$ to the attributes $\boldsymbol{X}$.

4.2. *Learning the parameters*

When learning the parameters in an LCM, we consider two types of models, corresponding to the situations where the parameters are either *tied* or *untied*. The difference being that when the parameters are tied, we constrain the covariance matrices $\boldsymbol{\Theta}_m$ s.t. $\boldsymbol{\Theta}_m = \boldsymbol{\Theta}_{m'}$, for all $1 \le m, m' \le \mid sp(M) \mid$. Recall that this constraint allows $\boldsymbol{\Theta}_m$ to be interpreted as sensor noise, thus contributing to the semantical interpretation of the model.

The actual learning of the parameters is performed by applying an EM-algorithm (Dempster, Laird, & Rubin, 1977) for LCMs. In the following we shall use α_k to denote the number of cases for which $Y = k$ in the database. Furthermore, we let $\vec{\boldsymbol{L}}_m$ be the matrix defined by $[\boldsymbol{L}_m, \boldsymbol{\eta}_m]$, and let $\vec{\boldsymbol{l}}_{m,i}^{\mathrm{T}}$ denote the i'th row of this matrix. In the same way, we define $\vec{\boldsymbol{Z}}$ as the augmented column vector of factors, i.e., $\vec{\boldsymbol{Z}} = [\boldsymbol{Z}^{\mathrm{T}}, 1]^{\mathrm{T}}$.[11] Finally, we let $\hat{\theta}_{m,k}$ denote the k'th element on the diagonal of $\hat{\boldsymbol{\Theta}}_m$ (in case of untied covariances); alternatively use $\hat{\theta}_{\cdot,k}$ to represent the k'th element on the diagonal of the tied covariance matrix. Based on this notation, the updating rules (M-step) for the EM-algorithm are given in the following.[12]

$$\hat{\boldsymbol{\mu}}_k \leftarrow \frac{1}{\alpha_k} \sum_{j:y_j=k} \sum_m P(M = m \mid \boldsymbol{D}_j)\, \mathbb{E}(\boldsymbol{Z} \mid M = m, \boldsymbol{D}_j)$$

$$\hat{\boldsymbol{\Gamma}}_k \leftarrow \frac{1}{\alpha_k} \sum_{j:y_j=k} \sum_m P(M = m \mid \boldsymbol{D}_j) \cdot \mathbb{E}[(\boldsymbol{Z} - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{Z} - \hat{\boldsymbol{\mu}}_k)^{\mathrm{T}} \mid M = m, \boldsymbol{D}_j]$$

$$\hat{\boldsymbol{l}}_{m,i} \leftarrow \left[\sum_{j=1}^{N} P(M = m \mid \boldsymbol{D}_j)\mathbb{E}(\vec{\boldsymbol{Z}}\vec{\boldsymbol{Z}}^{\mathrm{T}} \mid \boldsymbol{D}_j, M = m)\right]^{-1}$$

$$\left[\sum_{j=1}^{N} x_{i,j} P(M = m \mid \boldsymbol{D}_j)\mathbb{E}(\vec{\boldsymbol{Z}} \mid D_j, M = m)\right]$$

$$\hat{\theta}_{\cdot,i} \leftarrow \frac{1}{N}\sum_{j=1}^{N}\left[x_{i,j} - \sum_{m=1}^{|sp(M)|} P(M = m \mid \boldsymbol{D}_j)\, \hat{\boldsymbol{l}}_{m,i}^{\mathrm{T}}\, \mathbb{E}(\vec{\boldsymbol{Z}} \mid D_j, M = m)\right] x_{i,j}$$

$$\hat{\theta}_{m,i} \leftarrow \frac{1}{N}\sum_{j=1}^{N} P(M = m \mid \boldsymbol{D}_j)\big[x_{i,j} - \hat{\boldsymbol{l}}_{m,i}^{\mathrm{T}}\, \mathbb{E}(\vec{\boldsymbol{Z}} \mid D_j, M = m)\big] x_{i,j}$$

The derivation of these rules can be found in Appendix B together with the E-step for the algorithm. The computational complexity of one iteration of the EM algorithm is

$$O(\mid sp(M) \mid (N[q^2(\mid sp(Y) \mid + n) + q^3] + \mid sp(Y) \mid [n \cdot q^2 + q \cdot n^2 + n^3]));$$

further details can be found in Appendix C together with a complexity analysis of the overall algorithm.

5. Experimental results

In this section we investigate the classification accuracy of the proposed classifiers by considering 15 different datasets. Table 1 summarizes relevant characteristics of the datasets used in this experimental study, all of which are taken from the UCI Machine Learning Repository (Blake & Merz, 1998).

For each classifier, and each dataset, we have estimated the accuracy as the percentage of correctly classified instances through 5-fold cross-validation. We also give the theoretical standard deviations of these estimates calculated according to Kohavi (1995). In order to test whether a given classifier is better than another classifier we have used Nadeau and Bengio (2003)'s *corrected resampled t-test*, which incorporates information about classifier accuracy on each cross-validation fold; the same cross-validation folds were given to all classification algorithms.

When working with linear LCMs, we use the wrapper approach (Kohavi & John, 1997) to determine the number of latent variables, q, and following Proposition 4 we let $n \cdot \mid sp(Y) \mid$ be an upper bound. Specifically, we considered $q = 1, 2, \ldots, n \cdot \mid sp(Y) \mid$ and the q-value resulting in the highest (estimated) classification accuracy was selected.

When considering non-linear LCMs, we look for a value for both q and the number of mixture components, $\mid sp(M) \mid$. Again, we require that $q \leq n \cdot \mid sp(Y) \mid$, and to avoid (extreme) overfitting we also impose the constraints that $q \cdot \mid sp(M) \mid \leq N$. Unfortunately, if no other restrictions are placed on the parametrization of the model, then an exhaustive search over all parameter pairs will be computationally prohibitive for practical applications. We solve this problem by applying a semi-greedy search strategy: More precisely, we

Table 1. Description of datasets.

Database	#Att	#Cls	Inst	q	$\mid sp(M) \mid$
balance-scale	4	3	625	(4 – 6 – 10)	(2 – 20 – 25)
breast	10	2	683	(1 – 1 – 2)	(1 – 1 – 30)
crabs	5	4	200	(3 – 4 – 18)	(2 – 10 – 20)
diabetes	8	2	768	(3 – 6 – 14)	(1 – 20 – 35)
glass	9	7	214	(2 – 4 – 20)	(10 – 25 – 35)
glass2	9	2	163	(2 – 3 – 4)	(15 – 30 – 40)
heart	13	2	270	(2 – 5 – 6)	(1 – 1 – 2)
iris	4	3	150	(3 – 3 – 6)	(2 – 4 – 25)
liver	6	2	345	(10 – 12 – 12)	(4 – 15 – 40)
pima	8	2	768	(8 – 8 – 14)	(1 – 1 – 30)
sonar	60	2	208	(12 – 40 – 100)	(15 – 20 – 40)
tae	5	3	151	(6 – 8 – 14)	(20 – 35 – 40)
thyroid	5	3	215	(6 – 6 – 14)	(2 – 3 – 15)
vehicle	18	4	846	(35 – 40 – 40)	(20 – 25 – 40)
wine	13	3	178	(6 – 14 – 18)	(1 – 4 – 30)

The datasets which are used in the experimental study. For each dataset we have listed the number of attributes (#Att), the number of classes (#Cls), and the number of instances in the dataset (Inst.). The complexity of learning these domains is indicated by the number of latent variables used by the linear LCM classifier (q) and the number of mixtures used by the non-linear LCM classifier ($\mid sp(M) \mid$). The minimum, median and maximum values (over 5 cross-validation folds) are given.

perform a greedy search for q by scoring each q with the maximum accuracy it achieves when coupled with all the possible values of $\mid sp(M) \mid$. When the search terminates, the parameter pair with the highest estimated accuracy is chosen.[13]

In order to learn the probabilities in the models we applied the EM algorithm with standard parameter settings: The algorithm terminates when the relative increase in log-likelihood falls below 10^{-3} or after a maximum of 100 iterations. The EM algorithms were run with 100 restarts; this gives a number of different candidate models from which we should select one. The standard solution is to choose the candidate model with the highest log-likelihood on the training data, however, since our focus is classification we instead pick the model that obtains the *highest classification accuracy* on the training-set.[14]

A series of different classification algorithms have been implemented and tested on the datasets listed in Table 1. The classification algorithms have been selected with the goal of (i) providing a comparison with the LCMs, and (ii) exploring the effect of relaxing the two assumptions underlying the Naïve Bayes classifier (i.e., the conditional independence assumption and the assumption that all continuous variables follow a specific family of distributions). Before we present the accuracy results (in Tables 2 and 3) we give a brief description of each of the 16 different classifiers we consider:

NB: The NB classifier. Each attribute is assumed to be a realization from a Gaussian distribution, where the parameters are conditioned on the class variable.

NB/M: An NB classifier, where the distribution of the attributes are given as mixtures of Gaussian distributions. The number of mixtures is (for each attribute) chosen according to the BIC score (Schwarz, 1978).

NB/D: The NB classifier using discretized data. Specifically, the data was discretized prior to learning by using the MLC++ (Kohavi et al., 1994) implementation of Fayyad and Irani (1993)'s algorithm.

TAN/D: A TAN classifier (Friedman, Geiger, & Goldszmidt, 1997) using data that has been discretized by the same method as NB/D.

TAN+/D: A TAN/D classifier with smoothing. We followed (Friedman, Geiger, & Goldszmidt, 1997) and set the smoothing parameter equal to 5.

k-NN: The k-Nearest Neighbor classifier (Aha, Kibler, & Albert, 1991); k was determined by leave-one-out cross-validation.[15]

FA/BIC: An NB classifier where the data is preprocessed to obtain approximately independent factors. The preprocessing amounts to fitting a factor analysis through maximum likelihood calculations (Rubin & Thayer, 1982). The number of latent variables was chosen according to the BIC score.

PCA/λ: This classifier employs the same idea as FA/BIC, but the preprocessing is performed using PCA. The number of latent variables were chosen such that an eigenvector $\boldsymbol{e}_l$ with corresponding eigenvalue λ_l is included in the loading matrix if and only if $\lambda_l \geq (\sum_j \lambda_j)/n$ (see, e.g., Kendall, 1980 for details on this procedure).

PCA/n: This classifier is similar to PCA/λ, but with the exception that PCA/n uses all n eigenvectors of the covariance matrix to build the loading matrix. There is therefore no loss of information during preprocessing; the latent variables are defined by a rotation of the attribute space.

CW/PCA/n: As noted by several authors, the preprocessing step should not aim at generating factors that are unconditionally independent, but rather independent given the class. CW/PCA/n obtains this by fitting one PCA/n transformation for each class, and then performs classification using Bayes rule. Gama (2000)'s linear-Bayes is equivalent to CW/PCA/n when all attributes are continuous.

CG/PCA/n: The attribute space was clustered (using unsupervised clustering with soft cluster assignments) by approximating the joint distribution over the attributes with a mixture of n-dimensional Gaussians. Each mixture component is then seen as defining a cluster. The number of clusters (that is, the number of mixtures) was chosen to maximize the BIC score. Finally, one PCA/n was fitted for each cluster, and classification was performed using Bayes rule.

LCM(q): The linear LCM model with full structure as described in Section 3.1; q was found using the wrapper approach with 5 folds.

LCM($\boldsymbol{q}$)/S: This classifier is identical to LCM(q), except that structural learning is also applied, see Section 4.1.

LCM(q, m; **T):** The full non-linear LCM model as defined in Section 3.2; both q and $\mid sp(M) \mid$ are found by the wrapper approach. The covariance matrices are tied, meaning that $\mathrm{Cov}(X \mid \boldsymbol{Z} = \boldsymbol{z}, M = m) = \mathrm{Cov}(X \mid \boldsymbol{Z} = \boldsymbol{z}, M = m')$ for all $m, m' \in sp\,(M)$.

LCM($\boldsymbol{q}$, $\boldsymbol{m}$; **U):** The same classifier as LCM(q, m; T), but with the covariance matrices untied.

Table 2. Summary of accuracy calculations.

Classifier	Average acc.	Winner	Top 3	Poor, $p < 10\%$	Poor, $p < 1\%$
NB	71.5	1	3	10	5
NB/M	72.8	0	2	9	5
NB/D	74.9	2	2	8	3
TAN/D	75.6	0	1	9	3
TAN+/D	76.0	0	4	8	3
k-NN	81.3	0	5	6	3
FA/BIC	68.7	0	0	13	10
PCA/λ	73.8	0	0	12	5
PCA/*n*	77.1	0	0	10	4
CW/PCA/*n*	78.2	0	1	7	4
CG/PCA/*n*	79.2	2	2	8	1
LCM(*q*)	81.3	**4**	*5*	*2*	1
LCM(*q*)/S	80.9	2	4	3	1
LCM(*q*; *m*; T)	**83.8**	**4**	**7**	**0**	**0**
LCM(*q*; *m*; U)	*83.2*	*3*	**7**	**0**	**0**
LCM(*q*; *m*; T)/S	*81.7*	1	2	3	**0**

LCM(*q*, *m*; T)/S: The LCM(*q*, *m*; T) classifier combined with structural learning. All mixture components share the same structure.

Note that w.r.t. the second goal of this empirical study (mentioned above), we see that NB/M and NB/D are related to the assumption that the attributes follow a specific family of distributions, whereas FA/BIC, PCA/λ, PCA/*n*, CW/PCA/*n*, CG/PCA/*n*, LCM(*q*) and LCM(*q*)/S relax the independence assumptions. TAN/D, TAN+/D, *k*-NN, LCM (*q*, *m*; T), LCM(*q*, *m*; U) and LCM(*q*, *m*; T)/S try to relax both assumptions simultaneously.

The estimated classification accuracies are summarized in Table 2. For each classifier we give the average accuracy (averaged over all datasets in Table 1), the number of times each classifiers gives the best results for a particular dataset, and the number of datasets where it is in the top three. The last two columns give the number of times the classifier is significantly poorer than the best one for a specific dataset at 10% level and 1% level respectively. The best results overall are given in **boldface**, results in top tree in *italics*.

Table 3 gives the estimated classification accuracies for a subset of the classifiers. A classifier was included in Table 3 if it gave the best classification accuracy for at least one dataset. Additionally, *k*-NN was included due to its good overall performance. The best result for a given dataset is given in **boldface**. Classifiers that obtain results significantly worse than the best classifier on a given dataset are marked with $^-$ (significant at 10% level) or * (significant at the 1% level).

Our results indicate that the independence assumptions embedded in the Naïve Bayes model structure have a higher impact on the classification accuracy than the distributional assumption: The average accuracy of NB/M is only marginally better than that of NB, and

Table 3. Classifier accuracies.

Database	NB	NB/D	*k*-NN	CG/PCA/*n*	LCM(*q*)	LCM(*q*)/S	LCM(*q*, *m*; T)	LCM(*q*, *m*; U)	LCM(*q*, *m*; T)/S
balance-scale	$^-$86.9 ± 1.4	$^-$65.8 ± 1.9	89.8 ± 1.2	$^-$76.3 ± 1.7	88.0 ± 1.3	87.8 ± 1.3	**90.9 ± 1.2**	89.0 ± 1.3	87.4 ± 1.3
breast	$^-$96.2 ± 0.7	**97.4 ± 0.6**	$^-$95.9 ± 0.8	$^-$94.6 ± 0.9	96.8 ± 0.7	96.5 ± 0.7	96.5 ± 0.7	96.5 ± 0.7	96.5 ± 0.7
crabs	*39.5 ± 3.5	*44.5 ± 3.5	*89.5 ± 2.2	94.5 ± 1.6	94.5 ± 1.6	**95.5 ± 1.5**	**95.5 ± 1.5**	**95.5 ± 1.5**	94.5 ± 1.6
diabetes	75.9 ± 1.5	75.6 ± 1.5	75.1 ± 1.6	74.5 ± 1.6	75.9 ± 1.5	**76.6 ± 1.5**	75.9 ± 1.6	75.5 ± 1.6	72.9 ± 1.6
glass	*36.4 ± 3.3	71.0 ± 3.1	71.5 ± 3.1	**72.0 ± 3.1**	$^-$57.0 ± 3.4	62.1 ± 3.3	70.1 ± 3.2	64.5 ± 3.3	67.3 ± 3.2
glass2	*62.0 ± 3.8	81.6 ± 3.0	$^-$77.3 ± 3.2	$^-$72.4 ± 3.5	*66.9 ± 3.7	*67.6 ± 3.7	**85.3 ± 3.0**	81.0 ± 3.3	79.8 ± 3.1
heart	84.8 ± 2.2	83.7 ± 2.2	82.6 ± 2.3	$^-$77.4 ± 2.5	**85.2 ± 2.2**	83.7 ± 2.2	83.3 ± 2.3	83.3 ± 2.3	82.6 ± 2.3
iris	$^-$95.3 ± 1.7	*93.3 ± 2.0	*94.7 ± 1.7	96.7 ± 1.5	**98.0 ± 1.1**	96.7 ± 1.5	96.7 ± 1.6	97.3 ± 1.3	$^-$95.3 ± 1.7
liver	$^-$58.0 ± 2.7	$^-$58.0 ± 2.7	64.6 ± 2.7	**69.0 ± 2.5**	65.2 ± 2.6	66.4 ± 2.5	68.4 ± 2.5	**69.0 ± 2.5**	**69.0 ± 2.5**
pima	75.0 ± 1.6	**76.2 ± 1.5**	75.3 ± 1.5	72.8 ± 1.6	74.0 ± 1.6	74.4 ± 1.6	75.0 ± 1.6	75.6 ± 1.5	73.7 ± 1.6
sonar	*70.7 ± 3.2	$^-$76.5 ± 2.9	83.6 ± 2.6	*70.7 ± 3.2	81.2 ± 2.7	$^-$76.4 ± 2.9	80.2 ± 2.8	**84.1 ± 2.5**	83.2 ± 2.6
tae	54.3 ± 4.1	$^-$49.6 ± 4.1	57.0 ± 4.0	53.6 ± 4.1	56.9 ± 4.0	53.6 ± 4.1	**61.5 ± 4.0**	58.9 ± 4.0	56.9 ± 4.0
thyroid	**96.3 ± 1.3**	$^-$92.6 ± 1.8	95.8 ± 1.4	$^-$89.8 ± 2.1	95.3 ± 1.4	95.8 ± 1.4	94.4 ± 1.6	95.4 ± 1.4	$^-$89.3 ± 2.1
vehicle	*44.3 ± 1.7	*59.1 ± 1.7	*70.7 ± 1.6	$^-$77.3 ± 1.4	**84.3 ± 1.3**	$^-$80.4 ± 1.4	83.5 ± 1.3	84.2 ± 1.3	$^-$79.2 ± 1.4
wine	$^-$97.2 ± 1.2	98.9 ± 0.8	$^-$95.5 ± 1.5	$^-$97.2 ± 1.2	**100.0 ± 0.0**	99.4 ± 0.6	99.4 ± 1.0	98.9 ± 0.8	97.7 ± 1.1
Average	71.5	79.2	74.9	81.3	81.3	80.9	83.8	83.2	81.7

Classifiers used: NB: Naïve Bayes with Gaussian leaves. NB/D: Naïve Bayes with discretized data. *k*-NN: *k* nearest neighbor; *k* found by leave-one-out cross-validation. CG/PCA/*n*: Unsupervised clustering of attribute space by fitting multidimensional Gaussians; number of clusters decided by the BIC score; one PCA/*n* fitted per cluster. LCM(*q*): Linear LCM model where *q* was found by the wrapper approach. LCM(*q*)/S: Structural learning of LCM(*q*). LCM(*q*, *m*; T): Non-linear LCM, *q* and $| sp(M) |$ found by wrappers. Covariances tied. LCM(*q*, *m*; U): As LCM(*q*, *m*; T), but covariances untied. LCM(*q*, *m*; T)/S: Structural learning of LCM(*q*, *m*; T). See text for details.

The best result for a given dataset is given in **boldface**. Classifiers that obtain a result which is significantly worse than the best classifier on a given dataset are marked with - (significant at 10% level) or * (significant at the 1% level). The significance levels are calculated using the corrected resampled *t*-test (Nadeau & Bengio, 2003).

although NB/M is significantly better than NB in some cases (e.g., the glass domain), it is also significantly worse in other domains (e.g., balance-scale). Our results point towards discretization being slightly better than using mixtures of Gaussians in the Naïve Bayes model, but, on the other hand, even NB/D performs rather poorly on average.

Preprocessing the database can be quite beneficial, but the information loss incurred by this procedure can also jeopardize classification quality. FA/BIC is the classifier with the worst overall performance, and is only competitive (at the 10% level) in 2 of the 15 domains. On the other hand, PCA/n either as-is or combined with clustering (CW/PCA/n and CG/PCA/n) can supply reasonable solutions in domains where we have a strong correlation between the attributes (see, e.g., the crabs domain). LCM(q) and LCM(q)/S work well overall, but have (significant) problems with highly non-Gaussian domains like glass and glass2.

Structural learning does not appear to increase classifier accuracy on average, but can improve classification quality for some datasets. LCM(q)/S is, for instance, the best classifier overall on the diabetes dataset, but also significantly worse than LCM(q) in the vehicle domain. This is also in accordance with the claims by Greiner, Grove, and Schuurmans (1997), as our structural learning procedure focuses on the Bayesian metric for Gaussian networks (Geiger & Heckerman, 1994) and not classification accuracy.

The main conclusion is that the two non-linear LCM models appear to be superior to the other classification algorithms. Neither of these classifiers are significantly worse than any of the other classifiers at *any* dataset (at 10% level), and they also come out as the two best classifiers overall.

6. Conclusion

In this paper we have proposed a new class of models, termed Latent Classification Models (LCMs), for probabilistic classification in continuous domains. An LCM can roughly be seen as a mixture of factor analyzers integrated with a Naïve Bayes model. This combination enables concurrent clustering and, within each cluster, localized classification. LCMs extend the basic Naïve Bayes models in two ways: (i) The conditional independence assumptions embedded in the Naïve Bayes models are relaxed, and (ii) the distributional assumptions on the attributes are lifted. LCMs therefore constitute a flexible class of models in the sense that an LCM can approximate any continuous distribution over the attributes arbitrarily well. This is also confirmed in our experimental study, where it is demonstrated that LCMs provide good results in all the domains we investigate. Moreover, we find that LCMs are significantly better than a wide range of other probabilistic classifiers.

Appendix A: Proofs

Proof of Proposition 4: By definition of an LCM we have a q-dimensional random vector $\boldsymbol{Z} = (Z_1, \ldots, Z_q)$ of latent variables ($\boldsymbol{Z} \mid \{Y = j\} \sim \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Gamma}_j)$ for $j = 1, \ldots, \mid sp(Y) \mid$) and a linear mapping $\boldsymbol{L}$ represented by a $(n \times q)$-matrix such that $\boldsymbol{X} = \boldsymbol{L}\boldsymbol{Z} + \boldsymbol{\epsilon}$; $\boldsymbol{\epsilon}$ is white noise, i.e., $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Theta})$ and $\boldsymbol{\Theta}$ is assumed diagonal. Note that as $\boldsymbol{Z} \mid \{Y = j\}$ is Gaussian, it follows that $\boldsymbol{LZ} \mid \{Y = j\}$ is Gaussian as well, and $\boldsymbol{X} \mid \{Y = j\}$ is therefore fully described by its two first moments.

We must show that for a given $\{\boldsymbol{\alpha}_j, \boldsymbol{\Sigma}_j\}_{j=1}^{|sp(Y)|}$ we can define $\boldsymbol{L}$, $\boldsymbol{\Theta}$, $\{\boldsymbol{\mu}_j\}_{j=1}^{|sp(Y)|}$ and $\{\boldsymbol{\Gamma}_j\}_{j=1}^{|sp(Y)|}$ such that $\boldsymbol{X} \mid \{Y = j\} \sim \mathcal{N}(\boldsymbol{\alpha}_j, \boldsymbol{\Sigma}_j)$, for all $j = 1, \ldots, \mid sp(Y) \mid$. The proof is constructive, and we shall choose $q = n \cdot \mid sp(Y) \mid$ and $\boldsymbol{\Theta} = \boldsymbol{0}$ throughout. First, assume that $\boldsymbol{L}$ is a fixed $(n \times q)$-matrix with $\text{rank}(\boldsymbol{L}) = n$. Then, for any given $\boldsymbol{\alpha} \in \mathbb{R}^n$, we can find at least one $\boldsymbol{\mu} \in \mathbb{R}^q$ such that $\boldsymbol{L\mu} = \boldsymbol{\alpha}$. Repeat this $\mid sp(Y) \mid$ times, and let $\boldsymbol{\mu}_j$ be defined as a solution of $\boldsymbol{L\mu}_j = \boldsymbol{\alpha}_j$ for $j = 1, \ldots, \mid sp(Y) \mid$. By this construction we have that $\mathbb{E}[\boldsymbol{X} \mid Y = j] = \boldsymbol{L\mu}_j = \boldsymbol{\alpha}_j$; as long as $\text{rank}(\boldsymbol{L}) = n$ we can therefore always choose $\boldsymbol{\mu}_j$ so that the first moment of $\boldsymbol{X} \mid \{Y = j\}$ "fits".

To define $\boldsymbol{L}$ and $\{\boldsymbol{\Gamma}_j\}_{j=1}^{|sp(Y)|}$ we proceed by fixing a value of j, $j = j_0$, say. As $\boldsymbol{\Sigma}_{j_0}$ is positive semi-definite and symmetric matrix we can create the singular value decomposition of $\boldsymbol{\Sigma}_{j_0}$ as $\boldsymbol{\Sigma}_{j_0} = \boldsymbol{L}_{j_0}\boldsymbol{\Lambda}_{j_0}\boldsymbol{L}_{j_0}^{\mathrm{T}}$; $\boldsymbol{L}_{j_0}$ contains the eigenvectors of $\boldsymbol{\Sigma}_{j_0}$ and $\boldsymbol{\Lambda}_{j_0}$ is the diagonal matrix holding the corresponding eigenvalues, see, e.g., Kendall (1980). We do this for $j = 1, \ldots, \mid sp(Y) \mid$. Now, we define $\boldsymbol{L}$ as the $(n \times n\cdot \mid sp(Y) \mid)$-matrix made up by these $\boldsymbol{L}_j$ matrices, $\boldsymbol{L} = \left[\boldsymbol{L}_1 \mid \ldots \mid \boldsymbol{L}_{|sp(Y)|}\right]$, and let $\boldsymbol{\Gamma}_j$ be the block diagonal matrix with blocks made up by $j - 1$ $(n \times n)$ $\boldsymbol{0}$-matrices, followed by $\boldsymbol{\Lambda}_j$ and then $\mid sp(Y) \mid -j$ $(n \times n)$ $\boldsymbol{0}$-matrices,

$$\boldsymbol{\Gamma}_j = \begin{bmatrix} \boldsymbol{0}^{(j-1)n\times(j-1)n} & \boldsymbol{0}^{(j-1)n\times n} & \boldsymbol{0}^{(j-1)n\times(|sp(Y)|-j)n} \\ \boldsymbol{0}^{n\times(j-1)n} & \boldsymbol{\Lambda}_j & \boldsymbol{0}^{n\times(|sp(Y)|-j)n} \\ \boldsymbol{0}^{(|sp(Y)|-j)n\times(j-1)n} & \boldsymbol{0}^{(|sp(Y)|-j)n\times n} & \boldsymbol{0}^{(|sp(Y)|-j)n\times(|sp(Y)|-j)n} \end{bmatrix}.$$

By this construction we get

$$\begin{aligned} \text{Cov}(\boldsymbol{X} \mid Y = j) &= \boldsymbol{L\Gamma}_j\boldsymbol{L}^{\mathrm{T}} + \boldsymbol{\Theta} \\ &= \boldsymbol{L\Gamma}_j\boldsymbol{L}^{\mathrm{T}} \\ &= \boldsymbol{L}_j\boldsymbol{\Lambda}_j\boldsymbol{L}_j^{\mathrm{T}} \\ &= \boldsymbol{\Sigma}_j. \end{aligned}$$

Finally, we note that $\boldsymbol{L}$ will have rank n. This follows because there exists an ℓ where $\text{rank}(\boldsymbol{\Sigma}_\ell) = n$, and therefore $\text{rank}(\boldsymbol{L}_\ell) = n$. □

Proof of Proposition 6: Let us fix a $q \geq 1$, and for simplicity assume that $\boldsymbol{L}_m$ is a matrix where all elements are 0 for all $m = 1, \ldots, \mid sp(M) \mid$. Now, $\boldsymbol{X} \mid \{Y = k, M = m\} \sim \mathcal{N}(\boldsymbol{\eta}_m, \boldsymbol{\Theta}_m)$, and $\boldsymbol{X} \mid \{Y = k\}$ is therefore a mixture of (multivariate) Gaussians, where each mixture component is parameterized independently of the others. The result now follows trivially as any distribution over continuous variables can be approximated arbitrarily well by such a mixture of multivariate Gaussians (Bishop, 1995). □

Appendix B: The EM algorithm for LCMs

In this section we describe the EM algorithm for learning the parameters in an LCM. First of all, observe that the joint probability distribution over $(\boldsymbol{X}, \boldsymbol{Z}, M, Y)$ can be expressed as:

$$f(\boldsymbol{x}, z, m, y) = f(y)f(m \mid y)f(z \mid y)\prod_{i=1}^{n} f(x_i \mid m, z),$$

where

$$\begin{aligned}
f(y) &= P(Y = y);\\
f(m \mid y) &= P(M = m \mid Y = y);\\
f(z \mid y) &= \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Gamma}_y);\\
&= (2\pi)^{-q/2} \mid \boldsymbol{\Gamma}_y \mid^{-1/2} \exp\left(-\frac{1}{2}(z - \boldsymbol{\mu}_y)^{\mathrm{T}}\boldsymbol{\Gamma}_y^{-1}(z - \boldsymbol{\mu}_y)\right);\\
f(x_i \mid m, z) &= \mathcal{N}\big(\boldsymbol{l}_{m,i}^{\mathrm{T}} z + \eta_{m,i}, \theta_{m,i}\big)\\
&= (2\pi)^{-1/2}\theta_{m,i}^{-1/2} \exp\left(-\frac{1}{2}\big(\big(x_i - \big(\boldsymbol{l}_{m,i}^{\mathrm{T}} z + \eta_{m,i}\big)\big)\theta_{m,i}^{-1/2}\big)^2\right);
\end{aligned}$$

$\boldsymbol{l}_{m,i}^{\mathrm{T}}$ is the i'th row of $\boldsymbol{L}_m$ and $\theta_{m,i}$ is the i'th element on the diagonal of $\boldsymbol{\Theta}_m$. That is, both the regression vector and the offset for $f(x_i \mid m, z)$ depend on the state of the mixture variable M.

First, define the augmented column vector of factors ($\vec{z} = [z^{\mathrm{T}}, 1]^{\mathrm{T}}$) and the augmented regression vector ($\vec{\boldsymbol{l}}_{m,i} = [\boldsymbol{l}_{m,i}^{\mathrm{T}}, \eta_{m,i}]^{\mathrm{T}}$). By using that $(z - \boldsymbol{\mu}_y)^{\mathrm{T}}\boldsymbol{\Gamma}_y^{-1}(z - \boldsymbol{\mu}_y)$ can be rewritten as

$$(z - \boldsymbol{\mu}_y)^{\mathrm{T}}\boldsymbol{\Gamma}_y^{-1}(z - \boldsymbol{\mu}_y) = \mathrm{tr}\big(\boldsymbol{\Gamma}_y^{-1} z z^{\mathrm{T}}\big) - 2\boldsymbol{\mu}_y^{\mathrm{T}}\boldsymbol{\Gamma}_y^{-1} z + \boldsymbol{\mu}_y^{\mathrm{T}}\boldsymbol{\Gamma}_y^{-1}\boldsymbol{\mu}_y,$$

we get:

$$\begin{aligned}
\log f(\boldsymbol{x}, z, m, y) = \log P(y) &+ \log P(m \mid y) - \frac{n+q}{2}\log(2\pi)\\
&- \frac{1}{2}\log \mid \boldsymbol{\Gamma}_y \mid - \frac{1}{2}\sum_{i=1}^{n}\log\theta_{m,i}\\
&- \frac{1}{2}\mathrm{tr}\big(\boldsymbol{\Gamma}_y^{-1} z z^{\mathrm{T}}\big) + \boldsymbol{\mu}_y^{\mathrm{T}}\boldsymbol{\Gamma}_y^{-1} z - \frac{1}{2}\boldsymbol{\mu}_y^{\mathrm{T}}\boldsymbol{\Gamma}_y^{-1}\boldsymbol{\mu}_y\\
&- \frac{1}{2}\sum_{i=1}^{n}\big[(x_i - \vec{\boldsymbol{l}}_{m,i}^{\mathrm{T}}\vec{z})\theta_{m,i}^{-1/2}\big]^2.
\end{aligned}$$

The expected log-likelihood of the database $\mathcal{D}_N = (\boldsymbol{D}_1, \boldsymbol{D}_2, \ldots, \boldsymbol{D}_N)$ can now be expressed as:

$$\begin{aligned}
\mathcal{Q} &= \mathbb{E}\log\left(\prod_{j=1}^{N} f(\cdot \mid \boldsymbol{D}_j)\right) = \mathbb{E}\left(\sum_{j=1}^{N}\log f(\cdot \mid \boldsymbol{D}_j)\right)\\
&= \sum_{j=1}^{N}\log P(y_j) + \sum_{j=1}^{N}\mathbb{E}\log P(M \mid y_j) - N\frac{n+q}{2}\log(2\pi)\\
&\quad - \sum_{h=1}^{|sp(Y)|}\frac{\alpha_h}{2}\log \mid \boldsymbol{\Gamma}_h \mid - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{N}\mathbb{E}(\log\theta_{M,i} \mid \boldsymbol{D}_j)\\
&\quad -\frac{1}{2}\sum_{j=1}^{N}\operatorname{tr}\!\left(\boldsymbol{\Gamma}_{y_j}^{-1}\mathbb{E}[\boldsymbol{Z}\boldsymbol{Z}^{\mathrm{T}} \mid \boldsymbol{D}_j]\right) + \sum_{j=1}^{N}\boldsymbol{\mu}_{y_j}^{\mathrm{T}}\boldsymbol{\Gamma}_{y_j}^{-1}\mathbb{E}[\boldsymbol{Z} \mid \boldsymbol{D}_j]\\
&\quad - \sum_{h=1}^{|sp(Y)|}\frac{\alpha_h}{2}\boldsymbol{\mu}_h^{\mathrm{T}}\boldsymbol{\Gamma}_h^{-1}\boldsymbol{\mu}_h\\
&\quad -\sum_{i=1}^{n}\sum_{j=1}^{N}\left[\frac{1}{2}{x_{i,j}}^2\mathbb{E}\!\left(\theta_{M,i}^{-1} \mid \boldsymbol{D}_j\right) + \frac{1}{2}\mathbb{E}\!\left(\vec{\boldsymbol{Z}}^{\mathrm{T}}\vec{\boldsymbol{l}}_{M,i}\vec{\boldsymbol{l}}_{M,i}^{\mathrm{T}}\vec{\boldsymbol{Z}}\theta_{M,i}^{-1} \mid \boldsymbol{D}_j\right)\right.\\
&\quad \left. - x_{i,j}\mathbb{E}\!\left(\vec{\boldsymbol{l}}_{M,i}^{\mathrm{T}}\vec{\boldsymbol{Z}}\theta_{M,i}^{-1} \mid \boldsymbol{D}_j\right)\right].
\end{aligned}$$

Based on the above expression for the expected log-likelihood, we can derive the updating rules which are used in the M-step of the algorithm. First of all, for $\vec{\boldsymbol{l}}_{m,i} = [\boldsymbol{l}_{m,i}^{\mathrm{T}}, \eta_{m,i}]^{\mathrm{T}}$ we have:

$$\begin{aligned}
\frac{\partial\mathcal{Q}}{\partial\vec{\boldsymbol{l}}_{m,i}} = \theta_{m,i}^{-1}\sum_{j=1}^{N}[&P(M = m \mid \boldsymbol{D}_j)\mathbb{E}(\vec{\boldsymbol{Z}}\vec{\boldsymbol{Z}}^{\mathrm{T}} \mid \boldsymbol{D}_j, M = m)\vec{\boldsymbol{l}}_{m,i}\\
&-x_{i,j}P(M = m \mid \boldsymbol{D}_j)\mathbb{E}(\vec{\boldsymbol{Z}} \mid D_j, M = m)].
\end{aligned}$$

Thus, we get the following update rule for $\vec{\boldsymbol{l}}_{m,i}$:

$$\begin{aligned}
\hat{\vec{\boldsymbol{l}}}_{m,i} \leftarrow &\left[\sum_{j=1}^{N}P(M = m \mid \boldsymbol{D}_j)\mathbb{E}(\vec{\boldsymbol{Z}}\vec{\boldsymbol{Z}}^{\mathrm{T}} \mid \boldsymbol{D}_j, M = m)\right]^{-1}\\
&\left[\sum_{j=1}^{N}x_{i,j}P(M = m \mid \boldsymbol{D}_j)\mathbb{E}(\vec{\boldsymbol{Z}} \mid D_j, M = m)\right].
\end{aligned}$$

Analogously, for $\theta_{m,i}$ we get:

$$\frac{\partial Q}{\partial \theta_{m,i}} = -\frac{1}{2}\sum_{j=1}^{N} P(M=m \mid \boldsymbol{D}_j)\theta_{m,i}^{-1} + \theta_{m,i}^{-2}\Omega_{m,i},$$

where

$$\begin{aligned}\Omega_{m,i} &= \frac{1}{2}\sum_{j=1}^{N} x_{i,j}^2 P(M=m \mid \boldsymbol{D}_j)\\ &\quad+\frac{1}{2}\sum_{j=1}^{N} P(M=m \mid \boldsymbol{D}_j)\vec{\boldsymbol{l}}_{m,i}^{\mathrm{T}}\mathbb{E}(\vec{\boldsymbol{Z}}\vec{\boldsymbol{Z}}^{\mathrm{T}} \mid \boldsymbol{D}_j, M=m)\vec{\boldsymbol{l}}_{m,i}\\ &\quad-\sum_{j=1}^{N} x_{i,j} P(M=m \mid \boldsymbol{D}_j)\vec{\boldsymbol{l}}_{m,i}^{\mathrm{T}}\mathbb{E}(\vec{\boldsymbol{Z}} \mid \boldsymbol{D}_j, M=m).\end{aligned}$$

Now substitute $\vec{\boldsymbol{l}}_{m,i}$ with $\hat{\vec{\boldsymbol{l}}}_{m,i}$, and we get:

$$\Omega_{m,i} = \frac{1}{2}\sum_{j=1}^{N}\left[x_{i,j} - \hat{\vec{\boldsymbol{l}}}_{m,i}^{\mathrm{T}}\mathbb{E}(\vec{\boldsymbol{Z}} \mid D_j, M=m)\right]P(M=m \mid \boldsymbol{D}_j)\cdot x_{i,j}.$$

Thus, we get the following update rule for $\theta_{m,i}$:

$$\hat{\theta}_{m,i} \leftarrow \frac{1}{N}\sum_{j=1}^{N}\left[x_{i,j} - \hat{\vec{\boldsymbol{l}}}_{m,i}^{\mathrm{T}}\mathbb{E}(\vec{\boldsymbol{Z}} \mid D_j, M=m)\right]P(M=m \mid \boldsymbol{D}_j)\cdot x_{i,j}.$$

In the special case where $\boldsymbol{\Theta}_m$ is constant over $m \in sp(M)$ ($\boldsymbol{\Theta}_m$ is interpreted as sensor noise) we have:

$$\hat{\theta}_{\cdot,i} \leftarrow \frac{1}{N}\sum_{j=1}^{N}\left[x_{i,j} - \sum_{m\in sp(M)} P(m \mid \boldsymbol{D}_j)\,\hat{\vec{\boldsymbol{l}}}_{m,i}^{\mathrm{T}}\,\mathbb{E}(\vec{\boldsymbol{Z}} \mid D_j, M=m)\right]x_{i,j}.$$

For $\boldsymbol{\mu}_y$ we have:

$$\begin{aligned}\frac{\partial Q}{\partial \boldsymbol{\mu}_y} &= \sum_{j:y_j=y}\boldsymbol{\Gamma}_y^{-1}\mathbb{E}(\boldsymbol{Z} \mid \boldsymbol{D}_j) - \frac{1}{2}\sum_{j:y_j=y}\left(\boldsymbol{\Gamma}_y^{-1} + \boldsymbol{\Gamma}_y^{-1^{\mathrm{T}}}\right)\boldsymbol{\mu}_y\\ &= \boldsymbol{\Gamma}_y^{-1}\sum_{j:y_j=y}\mathbb{E}(\boldsymbol{Z} \mid \boldsymbol{D}_j) - \alpha_y\boldsymbol{\Gamma}_y^{-1}\boldsymbol{\mu}_y,\end{aligned}$$

where $\alpha_y = |\{\boldsymbol{D} : \boldsymbol{D} \in \mathcal{D}_N, \boldsymbol{D}^{\downarrow Y} = y\}|$ is the number of observations from class y. Thus we get the following updating rule:

$$\begin{aligned}\hat{\boldsymbol{\mu}}_y \leftarrow & \frac{1}{\alpha_y} \sum_{j:y_j=y} \mathbb{E}(\boldsymbol{Z} \mid \boldsymbol{D}_j) \\ &= \frac{1}{\alpha_y} \sum_{j:y_j=y} \sum_{m} P(M = m \mid \boldsymbol{D}_j)\mathbb{E}(\boldsymbol{Z} \mid M = m, \boldsymbol{D}_j).\end{aligned}$$

For $\boldsymbol{\Gamma}_y$ we have that (recall that $\boldsymbol{\Gamma}_y = \boldsymbol{\Gamma}_y^{\mathrm{T}}$):

$$\begin{aligned}\frac{\partial Q}{\partial \boldsymbol{\Gamma}_y} = -\frac{\alpha_y}{2}\boldsymbol{\Gamma}_y^{-1}\Bigg(&\boldsymbol{I} - \frac{1}{\alpha_y} \sum_{j:y_j=y} \mathbb{E}(\boldsymbol{Z}\boldsymbol{Z}^{\mathrm{T}} \mid \boldsymbol{D}_j)\boldsymbol{\Gamma}_y^{-1} \\ &+ \frac{2}{\alpha_y} \sum_{j:y_j=y} \boldsymbol{\mu}_y \mathbb{E}(\boldsymbol{Z} \mid \boldsymbol{D}_j)^T \boldsymbol{\Gamma}_y^{-1} - \boldsymbol{\mu}_y \boldsymbol{\mu}_y^{\mathrm{T}} \boldsymbol{\Gamma}_y^{-1}\Bigg),\end{aligned}$$

which gives the following update rule:

$$\begin{aligned}\boldsymbol{\Gamma}_y \leftarrow \frac{1}{\alpha_y} \sum_{j:y_j=y} \sum_{m} P(M = m \mid \boldsymbol{D}_j) \cdot \\ \mathbb{E}[(\boldsymbol{Z} - \hat{\boldsymbol{\mu}}_y)(\boldsymbol{Z} - \hat{\boldsymbol{\mu}}_y)^{\mathrm{T}} \mid \boldsymbol{D}_j, M = m].\end{aligned}$$

Finally, in order to obtain the probabilities for the mixture variable we use (see also Ghahramani & Hinton, 1996; Murphy, 1998):

$$P(M = m \mid Y = y) = \int P(M = m \mid Y = y, \boldsymbol{X})P(\boldsymbol{X} \mid Y = y)d\boldsymbol{X}.$$

To estimate this integral we use the empirical distribution of the data as an estimate for $P(\boldsymbol{X} \mid Y = y)$:

$$P(M = m \mid Y = y) \leftarrow \frac{1}{\alpha_y} \sum_{j:y_j=y} P(M = m \mid \boldsymbol{D}_j).$$

In order to perform the expectation step we first note that given $Y = y$ and $M = m$, the joint $(\boldsymbol{X}, \boldsymbol{Z})$ has a Gaussian distribution:

$$(\boldsymbol{X}, \boldsymbol{Z}) \mid \{Y = y, M = m\} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{L}_m \boldsymbol{\mu}_y \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{L}_m \boldsymbol{\Gamma}_y \boldsymbol{L}_m^{\mathrm{T}} + \boldsymbol{\Theta}_m & \boldsymbol{L}_m \boldsymbol{\Gamma}_y \\ \boldsymbol{\Gamma}_y \boldsymbol{L}_m^{\mathrm{T}} & \boldsymbol{\Gamma}_y \end{bmatrix}\right).$$

Thus, we get:

$$\boldsymbol{Z} \mid \{Y = y, \boldsymbol{X} = \boldsymbol{x}, M = m\} \sim \mathcal{N}(\boldsymbol{\beta}_{m,y}\boldsymbol{x} + (\boldsymbol{I} - \boldsymbol{\beta}_{m,y}\boldsymbol{L}_m)\boldsymbol{\mu}_y, (\boldsymbol{I} - \boldsymbol{\beta}_{m,y}\boldsymbol{L}_m)\boldsymbol{\Gamma}_y),$$

where $\boldsymbol{\beta}_{m,y} = (\boldsymbol{L}_m\boldsymbol{\Gamma}_y)^{\mathrm{T}}(\boldsymbol{L}_m\boldsymbol{\Gamma}_y\boldsymbol{L}_m^{\mathrm{T}} + \boldsymbol{\Theta}_m)^{-1}$. More specifically,

$$\mathbb{E}(\boldsymbol{Z} \mid \boldsymbol{D}_j, M = m) = \boldsymbol{\beta}_{m,y}\boldsymbol{D}_j^{\downarrow \boldsymbol{x}} + (\boldsymbol{I} - \boldsymbol{\beta}_{m,y}\boldsymbol{L}_m)\boldsymbol{\mu}_y$$

and

$$\begin{aligned}&\mathbb{E}(\boldsymbol{Z}\boldsymbol{Z}^{\mathrm{T}} \mid \boldsymbol{D}_j, M = m)\\ &\quad= (\boldsymbol{I} - \boldsymbol{\beta}_{m,y}\boldsymbol{L}_m)\boldsymbol{\Gamma}_y + \mathbb{E}(\boldsymbol{Z} \mid \boldsymbol{D}_j, M = m)\mathbb{E}(\boldsymbol{Z} \mid \boldsymbol{D}_j, M = m)^{\mathrm{T}}.\end{aligned}$$

Appendix C: Complexity analysis

For fixed values for q and $\mid sp(M) \mid$, the time complexity of the individual M-steps are given by: $\hat{\boldsymbol{\mu}}_k \sim O(N \cdot \mid sp(M) \mid \cdot q)$, $\hat{\boldsymbol{\Gamma}}_k \sim O(N \cdot \mid sp(M) \mid \cdot q^2)$, $\hat{\boldsymbol{l}}_{m,i} \sim O(N \cdot q^2 + n^3)$, $\hat{\theta}_{\cdot,i} \sim O(N \cdot \mid sp(M) \mid \cdot q)$ and $\hat{\theta}_{m,i} \sim O(N \cdot q)$; for simplicity we let inversion of an $n \times n$ matrix have time complexity $O(n^3)$. Note that, e.g., $\hat{\boldsymbol{\Gamma}}_k$ must be calculated for each possible value of $1 \leq k \leq \mid sp(Y) \mid$ hence, we get:

$$\begin{aligned}&\hat{\boldsymbol{\mu}} \sim O(N \cdot \mid sp(M) \mid \cdot q \cdot \mid sp(Y) \mid); \quad \hat{\boldsymbol{\Theta}}_{\text{tied}} \sim O(N \cdot \mid sp(M) \mid \cdot q \cdot n);\\ &\hat{\boldsymbol{\Gamma}} \sim O(N \cdot \mid sp(M) \mid \cdot q^2 \cdot \mid sp(Y) \mid); \quad \hat{\boldsymbol{\Theta}}_{\text{untied}} \sim O(N \cdot q \cdot n \mid sp(M) \mid);\\ &\hat{\boldsymbol{L}} \sim O((N \cdot q^2 \cdot n + n^3) \cdot \mid sp(M) \mid).\end{aligned}$$

For the E-step, the time complexity is determined by the complexity of calculating $\mathbb{E}(\boldsymbol{Z} \mid \boldsymbol{D}_j, M = m)$ and $\mathbb{E}(\boldsymbol{Z}\boldsymbol{Z}^{\mathrm{T}} \mid \boldsymbol{D}_j, M = m)$, which both depend on $\boldsymbol{\beta}_{m,y}$ (see Appendix B). Specifically, when also incorporating the iterations over the conditioning sets, we get:

$$\begin{aligned}&\{\boldsymbol{\beta}_{m,y} : 1 \leq m \leq \mid sp(M) \mid, 1 \leq y \leq \mid sp(Y) \mid\}\\ &\quad\sim O(\mid sp(M) \mid \cdot \mid sp(Y) \mid \cdot (n \cdot q^2 + q \cdot n^2 + n^3));\\ &\{\mathbb{E}(\boldsymbol{Z} \mid \boldsymbol{D}_j, M = m) : 1 \leq j \leq N, 1 \leq m \leq \mid sp(M) \mid\}\\ &\quad\sim O(\mid sp(M) \mid \cdot (N \cdot n \cdot q + n \cdot q^2));\\ &\{\mathbb{E}(\boldsymbol{Z}\boldsymbol{Z}^{\mathrm{T}} \mid \boldsymbol{D}_j, M = m) : 1 \leq j \leq N, 1 \leq m \leq \mid sp(M) \mid\}\\ &\quad\sim O(\mid sp(M) \mid \cdot (n \cdot q^2 + N \cdot q^3)).\end{aligned}$$

Thus, the overall time complexity of (one iteration of) the EM-algorithm is:

$$O(\mid sp(M) \mid (N[q^2(\mid sp(Y) \mid + n) + q^3] + \mid sp(Y) \mid [n \cdot q^2 + q \cdot n^2 + n^3])).$$

The time complexity of the inference procedure is determined by the complexity of the initialization phase and the updating phase. During initialization we calculate $\mathbb{E}(\boldsymbol{X} \mid Y =$

$y, M = m) = \vec{\boldsymbol{L}}_m \vec{\boldsymbol{\mu}}_y$, $\mathrm{Cov}(\boldsymbol{X} \mid Y = y, M = m) = \vec{\boldsymbol{L}}_m \vec{\boldsymbol{\Gamma}}_y \vec{\boldsymbol{L}}_m^{\mathrm{T}} + \boldsymbol{\Theta}_m)$, $\mathrm{Cov}(\boldsymbol{X} \mid Y = y, M = m)^{-1}$, and $\det(\mathrm{Cov}(\boldsymbol{X} \mid Y = y, M = m))$, where we have defined $\vec{\boldsymbol{\mu}}_y = \mathbb{E}(\vec{\boldsymbol{Z}} \mid Y = y)$ and $\vec{\boldsymbol{\Gamma}}_y = \mathrm{Cov}(\vec{\boldsymbol{Z}} \mid Y = y)$. It follows that initialization has time complexity $O(\mid sp(Y) \mid \cdot \mid sp(M) \mid \cdot (n^2 \cdot q + q^2 \cdot n + n^3))$. The time complexity of performing one propagation in an LCM (after initialization) is $O(n^2 \cdot \mid sp(Y) \mid \cdot \mid sp(M) \mid)$, hence the complexity of performing both initialization and N propagations is $O(\mid sp(Y) \mid \cdot \mid sp(M) \mid \cdot (n^2 \cdot q + q^2 \cdot n + n^3 + N \cdot n^2))$.

Generally, the time complexity of Algorithm 1 (without learning the edge-set of the classifier) is defined as above together with the time complexity of the search procedure being applied w.r.t. q and $\mid sp(M) \mid$ (we propose a semi-greedy search procedure in Section 5). Note that when structural learning is also performed, the time complexity also depends on the SEM algorithm (Friedman, 1998). However, as opposed to standard learning problems we have a tight upper bound on the number of different structures to investigate, i.e., we only need to focus on the $q \cdot n$ possible edges from the latent variables to the attributes.

Acknowledgments

We would like to thank David Heckerman for clarifying the assumptions underlying the metric for Gaussian networks by Geiger and Heckerman (1994). We have also benefitted from interesting discussions with the members of the Decision Support Systems group at Aalborg University, and from the comments of the anonymous referees. Finally, we would like to thank *Hugin Expert* (www.hugin.com) for giving us access to the *Hugin Decision Engine*, which forms the basis of our implementation.

Notes

1. The potential information loss should be compared to the situation where the distribution is modelled by some appropriate parametric family of continuous probability distributions.
2. As separate PCAs are used for each class, the generated factors are not necessarily comparable. A similar approach was nevertheless pursued by Bressan and Vitrià (2002), and they reported promising results regarding classification accuracies.
3. Note that $\boldsymbol{L}$ determines the graphical structure between $\boldsymbol{Z}$ and $\boldsymbol{X}$, i.e., if $l_{i,j} = 0$, then there is no arc from Z_j to X_i and consequently $X_i \perp\!\!\!\perp Z_j \mid Y$.
4. Observe that the assumption $q \leq n$ (common in FA) does not apply here.
5. In what follows, we will refer to the model described in Section 3.1 as a linear LCM because of the linear mapping, $\boldsymbol{L}$, from $\boldsymbol{Z}$ to $\boldsymbol{X}$.
6. Note that if $\boldsymbol{\Theta}_i \neq \boldsymbol{\Theta}_j$, for some $1 \leq i \neq j \leq m$, then we no longer have an immediate interpretation of $\boldsymbol{\Theta}_i$ as being sensor noise (we shall return to this discussion in Section 4).
7. Fokoué and Titterington (2003) consider learning mixture of FAs by Markov Chain Monte Carlo methods.
8. Minka (2000) describes a global method for learning the dimensionality for PCA. Unfortunately, such methods are not necessarily suitable within the proposed classification context; this is also illustrated in Section 5 in the form of the PCA/λ classifier.
9. Recall that the structure between $\boldsymbol{Z}$ and $\boldsymbol{X}$ is determined by the zero elements in the regression matrices.
10. In an LCM, the class variable Y is a parent of the latent variables, but not the attributes. This violates Geiger and Heckerman (1994)'s structural constraint, since the latent variables and the attributes are connected. Similarly, M causes problems as it is a parent of all the attributes, but it is not a parent of the latent variables.

11. For linear LCMs we simply use $\vec{L}_m = L_m$ and $\vec{Z} = Z$.
12. Observe that, except for the difference between $\hat{\theta}_{\cdot,k}$ and $\hat{\theta}_{m,k}$, we have the same updating rules for the tied and the untied models.
13. Note that for the tests reported in Table 3, the possible values for $\mid sp(M) \mid$ was restricted to the set $\{1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 35, 40\}$.
14. This is motivated by Vapnik's bound (see, e.g., Burges, 1998, Section 2) and the fact that all candidate models per definition have the same VC-dimension.
15. We used the k-NN implementation in Weka 3.4 (Witten & Frank, 2000).

References

Aha, D., Kibler, D. F., & Albert, M. (1991). Instance-based learning algorithms. *Machine Learning, 6*, 37–66.

Bishop, C. M. (1995). *Neural networks for pattern recognition* Oxford, UK: Oxford University Press.

Blake, C., & Merz, C. (1998). UCI Repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Bøttcher, S. G. (2001). Learning Bayesian networks with mixed variables. In *Proceedings of the Eighth International Workshop in Artificial Intelligence and Statistics* (pp. 149–156). Morgan Kaufmann Publishers.

Bressan, M., & Vitrià, J. (2002). Improving Naive Bayes using class-conditional ICA. In *Advances in Artificial Intelligence* (pp. 1–10). Vol. 2527 of *Lecture Notes in Artificial Intelligence*. Berlin, Germany: Springer-Verlag.

Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery, 2:2*, 955–974.

Campbell, N., & Mahon, R. (1974). A multivariate study of variation in two species of rock crab of genus Leptograpsus. *Australian Journal of Zoology, 22*, 417–425.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B, 39*, 1–38.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: John Wiley & Sons.

Fayyad, U. M., & Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence* (pp. 1022–1027). San Mateo, CA: Morgan Kaufmann Publishers.

Fokoué, E., & Titterington, D. M. (2003). Mixtures of factor analysers. Bayesian estimation and inference by stochastic simulation. *Machine Learning, 50:1/2*, 73–94.

Friedman, N. (1998). The Bayesian structural EM algorithm. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence* (pp. 129–138). San Fransisco, CA: Morgan Kaufmann Publishers.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning, 29:2/3*, 131–163.

Friedman, N., Goldszmidt, M., & Lee, T. J. (1998). Bayesian network classification with continuous attributes: Getting the best of both discretization and parametric fitting. In *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 179–187). San Francisco, CA: Morgan Kaufmann Publishers.

Friedman, N., Linial, M., Nachman, I., & Pe'er, D. (2000). Using Bayesian networks to analyze expression data. *Journal of Computational Biology, 7*, 601–620.

Gama, J. (2000). A linear-Bayes classifier. In *IBERAMIA-SBIA* (pp. 269–279). Vol. 1952 of *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag.

Geiger, D., & Heckerman, D. (1994). Learning Gaussian networks. Technical Report MSR-TR-94-10, Microsoft Research.

Ghahramani, Z., & Hinton, G. E. (1996). The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, Department of Computer Science, University of Toronto, Canada.

Greiner, R., Grove, A. J., & Schuurmans, D. (1997). Learning Bayesian nets that perform well. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence* (pp. 198–207). San Fransisco, CA: Morgan Kaufmann Publishers.

Hinton, G., Dayan, P., & Revow, M. (1997). Modelling the manifolds of images of handwritten digits. *IEEE Transactions on Neural Networks, 8:1*, 65–74.

Hyvärinen, A., Karhunen, J., & Oja, E. (2001). *Independent component analysis, adaptive and learning systems for signal processing, communications, and control*. New York: John Wiley & Sons.

John, G. H., & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* (pp. 338–345). San Fransisco, CA: Morgan Kaufmann Publishers.

Kendall, M. (1980). *Multivariate analysis*. London, UK: Charles Griffin & Co., 2nd edition.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (pp. 1137–1143). San Mateo, CA: Morgan Kaufmann Publishers.

Kohavi, R., John, G., Long, R., Manley, D., & Pfleger, K. (1994). MLC++: A machine learning library in C++. In *Proceedings of the Sixth International Conference on Tools with Artificial Intelligence* (pp. 740–743). IEEE Computer Society Press.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence, 97:1/2*, 273–324.

Lam, W., & Bacchus, F. (1994). Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence, 10:4*, 269–293.

McLachlan, G. J. (2004). *Discriminant analysis and statistical pattern recognition*. New York: John Wiley & Sons.

Minka, T. P. (2000). Automatic choice of dimensionality for PCA. *Advances in Neural Information Processing Systems, 13*, 598–604.

Mitchell, T. M. (1997). *Machine learning* Boston, MA: McGraw Hill.

Monti, S., & Cooper, G. F. (1999a). A Bayesian network classifier that combines a finite mixture model and a Naive Bayes model. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence* CA., (pp. 447–456). San Fransisco: Morgan Kaufmann Publishers.

Monti, S. & Cooper, G. F. (1999b). Learning hybrid Bayesian networks from data. In M. I. Jordan (Ed.), *Learning in graphical models* (pp. 521–540). Cambridge, MA: The MIT Press.

Murphy, K. (1998). Fitting a conditional Gaussian distribution. Technical report, Department of Computer Science, University of California at Berkeley.

Nadeau, C., & Bengio, Y. (2003). Inference for the generalization error. *Machine Learning, 52:3*, 239–281.

Ripley, B. D. (1996). Pattern recognition and neural networks Cambridge, UK: Cambridge University Press.

Rissanen, J. (1978). Modelling by shortest data description. *Automatica, 14*, 465–471.

Rubin, D. B., & Thayer, D. T. (1982). EM algorithms for ML factor analysis. *Psychometrika, 47:1*, 69–76.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics, 6*, 461–464.

Vilalta, R., & Rish, I. (2003). A decomposition Of classes via clustering to explain and improve Naive Bayes. In *Proceedings of the Fourteenth European Conference on Machine Learning* (pp. 444–455). Vol. 2837 of *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag.

Witten, I. H. & Frank, E. (2000). *Data mining: Practical machine learning tools with Java implementations* San Francisco, CA: Morgan Kaufmann Publishers.

Accepted November 26, 2004