Learning Conditional Linear Gaussian Classifiers with Probabilistic Class Labels

Pedro L. López-Cruz, Concha Bielza, and Pedro Larrañaga

Computational Intelligence Group, Departamento de Inteligencia Artificial
Facultad de Informática, Universidad Politécnica de Madrid
pedro.lcruz@upm.es, {mcbielza,pedro.larranaga})@fi.upm.es

Abstract. We study the problem of learning Bayesian classifiers (BC) when the true class label of the training instances is not known, and is substituted by a probability distribution over the class labels for each instance. This scenario can arise, e.g., when a group of experts is asked to individually provide a class label for each instance. We particularize the generalized expectation maximization (GEM) algorithm in [1] to learn BCs with different structural complexities: naive Bayes, averaged one-dependence estimators or general conditional linear Gaussian classifiers. An evaluation conducted on eight datasets shows that BCs learned with GEM perform better than those using either the classical Expectation Maximization algorithm or potentially wrong class labels. BCs achieve similar results to the multivariate Gaussian classifier without having to estimate the full covariance matrices.

Keywords: Bayesian classifiers, probabilistic class labels, partially supervised learning, belief functions.

1 Introduction

A classification problem consists of assigning a class label to an object based on a set of characteristic features. Traditionally, machine learning research has focused on two problems: supervised and unsupervised learning. In supervised learning, the true class label of a set of training instances is known. In unsupervised learning settings, on the other hand, the true class label of the training instances is not available. It can be both hard and expensive to identify the true class label of all training instances. However, it is often easier to locate partial or incomplete information about the true class labels, and more sophisticated methods have been proposed for incorporating that information. Semi-supervised learning deals with the problem of learning classifiers when the true class labels of only a few training instances are known, and the rest of the training set is unlabeled. In partially supervised learning, a subset of possible class labels (including the true class) is given for each instance.

A general framework for learning multivariate Gaussian classifiers (MGC) is provided in [1], where the class information is modeled as belief functions [2], and a generalized expectation maximization (GEM) algorithm is proposed.

This approach includes supervised, unsupervised, semi-supervised and partially supervised learning as particular cases. Here we particularize the GEM algorithm to a specific scenario, where the information about the class for each instance is given as a probability distribution over the class labels. This is motivated by a problem in which it is hard to identify the true class labels of the training instances, perhaps because each label is not clearly defined, and a set of experts is asked to label the (same) training set to gain information about how the labels are assigned. Then, we summarize the information about the experts' classifications as probability distributions over the class labels.

Bayesian networks [3] are probabilistic graphical models which encode a factorization of the joint probability distribution over a set of variables, allowing for different kinds of reasoning and efficient computations. Bayesian classifiers (BC) [4] adapt Bayesian networks to classification problems. Here we adapt the GEM algorithm to fit BCs with different structures when the class information for each instance is given as a probability distribution.

In Sect. 2 we particularize the GEM algorithm to the case where the class information is given as probability distributions. Section 3 shows the use of the GEM algorithm to learn BCs. Section 4 includes the evaluation of the classifiers over eight datasets. Section 5 ends with conclusions and future work.

2 The GEM Algorithm for Probabilistic Class Labels

Our problem domain is modeled using n predictive univariate variables $\mathbf{X} = (X_1, \ldots, X_n)$ and a class variable C. The domain of each variable X_j is continuous and denoted as Ω_{X_j}. The class variable is discrete with $\Omega_C = \{1, \ldots, K\}$. We have a training dataset with N instances: $D = \{(\mathbf{x}_1, \boldsymbol{\pi}_1), \ldots, (\mathbf{x}_N, \boldsymbol{\pi}_N)\}$, where $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,n})$ are the values of the predictive variables for the ith instance, and $\boldsymbol{\pi}_i = (\pi_{i,1}, \ldots, \pi_{i,k})$ is the class information, i.e., a probability distribution over Ω_C so that $\pi_{i,k}$ is the probability of instance i belonging to class k, with $0 \leq \pi_{i,k} \leq 1$ and $\sum_{k=1}^{K} \pi_{i,k} = 1$. For instance, imagine that we ask 20 experts to classify each instance of a two-class problem and, for the ith instance, 15 experts classify it as belonging to class 1 and the rest assign the instance to class 2. We model that information as the probability distribution: $\boldsymbol{\pi}_i = (0.75, 0.25)$.

In [1], the information about the class of each instance $\mathbf{x}_i$ is modeled as a basic belief assignment (bba), which is a function $m_i^{\Omega_C} : 2^{\Omega_C} \rightarrow [0, 1]$ over the powerset 2^{Ω_C}, verifying $\sum_{\omega \subseteq \Omega_C} m_i^{\Omega_C}(\omega) = 1$. Table 1 shows an example of a general bba (top) from [5]. Using the belief function theory in the context of the transferable belief model [2], a generalization of the Expectation Maximization (EM) algorithm [6] is derived in [1] for fitting a finite mixture of multivariate Gaussian distributions with K components

$$f_{\mathbf{X}}(\mathbf{x}) = \sum_{k=1}^{K} p_C(k; \boldsymbol{\theta}_C) f_{\mathbf{X}|k}(\mathbf{x}; \boldsymbol{\mu}_{\mathbf{X}|k}, \boldsymbol{\Sigma}_{\mathbf{X}|k}) \ , \tag{1}$$

which is used as a MGC, where $p_C(k; \boldsymbol{\theta}_C)$ is the prior probability of $C = k$ and $f_{\mathbf{X}|k}(\mathbf{x}; \boldsymbol{\mu}_{\mathbf{X}|k}, \boldsymbol{\Sigma}_{\mathbf{X}|k})$ is the conditional multivariate Gaussian density function of

Table 1. Example of a general bba $m_i^{\Omega_C}(\omega)$ taken from [5] (top) and a Bayesian bba (bottom). The class variable C has three values $\Omega_C = \{1, 2, 3\}$.

	ω							
	$\emptyset$	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	Ω_C
General bba	0	0.1	0	0.3	0.2	0.3	0	0.1
Bayesian bba	0	0.3	0.2	0.5	0	0	0	0

the predictive variables $\mathbf{X}$ given $C = k$. The GEM algorithm finds the parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_C\} \cup \left\{\boldsymbol{\mu}_{\mathbf{X}|k}, \boldsymbol{\Sigma}_{\mathbf{X}|k}\right\}_{k=1,\ldots,K}$ that maximize a generalized log-likelihood (LL) criterion

$$\ln(pl^{\boldsymbol{\Theta}}(\boldsymbol{\Theta}|D)) = \sum_{i=1}^{N} \ln\left(\sum_{k=1}^{K} pl_{i,k} p_C(k; \boldsymbol{\theta}_C) f_{\mathbf{X}|k}(\mathbf{x}_i; \boldsymbol{\mu}_{\mathbf{X}|k}, \boldsymbol{\Sigma}_{\mathbf{X}|k})\right) + \nu \ , \qquad (2)$$

where ν is a constant and $pl_{i,k} = pl_i^{\Omega_C}(\{k\})$ are the plausibilities of the ith instance for the set $\{k\}$, with $pl_i^{\Omega_C}(\omega) = \sum_{\gamma \subseteq \Omega_C, \gamma \cap \omega \neq \emptyset} m_i^{\Omega_C}(\gamma), \forall \omega \subseteq \Omega_C$.

In our scenario, each bba is a probability distribution over Ω_C, so all the focal sets (subsets ω with $m_i^{\Omega_C}(\omega) > 0$) are singletons, and the bba is called a Bayesian bba (bottom row in Table 1). Since our $m_i^{\Omega_C}$ are Bayesian, the plausibility functions $pl_i^{\Omega_C}$ are probability measures: $pl_{i,k} = pl_i^{\Omega_C}(\{k\}) = m_i^{\Omega_C}(\{k\}) = \pi_{i,k}$. Then, the generalized LL criterion (2) is rewritten as

$$LL = \ln(p(\boldsymbol{\Theta}|D)) = \sum_{i=1}^{N} \ln\left(\sum_{k=1}^{K} \pi_{i,k} p_C(k; \boldsymbol{\theta}_C) f_{\mathbf{X}|k}(\mathbf{x}_i; \boldsymbol{\mu}_{\mathbf{X}|k}, \boldsymbol{\Sigma}_{\mathbf{X}|k})\right) + \nu \ . \quad (3)$$

The GEM algorithm is then particularized to maximize the LL criterion (3) by alternating the two steps:

- Expectation step in iteration q: compute the expected posterior probabilities

$$t_{i,k}^{(q)} = \frac{\pi_{i,k} p_C(k; \boldsymbol{\theta}_C) f_{\mathbf{X}|k}(\mathbf{x}_i; \boldsymbol{\mu}_{\mathbf{X}|k}, \boldsymbol{\Sigma}_{\mathbf{X}|k})}{\sum_{k'=1}^{K} \pi_{i,k'} p_C(k'; \boldsymbol{\theta}_C) f_{\mathbf{X}|k'}(\mathbf{x}_i; \boldsymbol{\mu}_{\mathbf{X}|k'}, \boldsymbol{\Sigma}_{\mathbf{X}|k'})} \ , \qquad (4)$$

- Maximization step in iteration q: find the parameters which maximize the expected LL of the complete data

$$\begin{aligned} \theta_{C=k}^{(q+1)} &= \frac{1}{N} \sum_{i=1}^{N} t_{i,k}^{(q)} \ , \boldsymbol{\mu}_{\mathbf{X}|k}^{(q+1)} = \frac{1}{\sum_{i=1}^{N} t_{i,k}^{(q)}} \sum_{i=1}^{N} t_{i,k}^{(q)} \mathbf{x}_i \ , \\ \boldsymbol{\Sigma}_{\mathbf{X}|k}^{(q+1)} &= \frac{1}{\sum_{i=1}^{N} t_{i,k}^{(q)}} \sum_{i=1}^{N} t_{i,k}^{(q)} \left(\mathbf{x}_i - \boldsymbol{\mu}_{\mathbf{X}|k}^{(q+1)}\right) \left(\mathbf{x}_i - \boldsymbol{\mu}_{\mathbf{X}|k}^{(q+1)}\right)^T \ . \end{aligned} \qquad (5)$$

Like the EM algorithm, the GEM algorithm guarantees that the generalized LL (3) increases in each iteration q up to a local maximum. To avoid local maxima, several runs of the algorithm are usually performed with different randomized initializations of the parameters $\boldsymbol{\Theta}$, and the model with the highest LL is returned. Instead, we consider the $\pi_{i,k}$ as initial values for $t_{i,k}^{(1)}$ in the first expectation step (4) of the algorithm. Therefore, the algorithm is only run once. The stopping criterion used to check the convergence of the algorithm is $(LL^q - LL^{q-1})/|LL^{q-1}| < \epsilon$. We set $\epsilon = 10^{-6}$.

3 Learning Bayesian Classifiers with GEM

Multivariate Gaussian classifiers, such as the ones used in [1], need to estimate a full covariance matrix $\boldsymbol{\Sigma}_{\mathbf{X}|k}$ for each class label. When the number of training instances N is low or a high number of predictive variables n are available in the dataset, the estimated covariance matrices might not be very accurate. BCs are able to exploit the conditional independence relationships between the predictive variables given the class variable, reducing the number of parameters for estimation. We focus on BCs which conform with the conditional linear Gaussian (CLG) network' structure [7], i.e., discrete variables cannot have continuous parents. Therefore, the class variable is a parent of all the predictive variables and the predictive variables can only have other (continuous) predictive variables as parents. In a BC with a CLG structure, the conditional density function for a continuous variable X_j having parents $\mathbf{Pa}(X_j) = (\mathbf{Y}_j, C)$, where $\mathbf{Y}_j$ is continuous, is defined as $f_{X_j|\mathbf{y}_j,k}(x_j) = \mathcal{N}(x_j\ ;\ \beta_{0,X_j|\mathbf{Y}_j,k} + \boldsymbol{\beta}^T_{X_j|\mathbf{Y}_j,k}\mathbf{y}_j, \sigma^2_{X_j|\mathbf{Y}_j,k})$, with

$$\begin{aligned} \beta_{0,X_j|\mathbf{Y}_j,k} &= \mu_{X_j|k} - \boldsymbol{\Sigma}_{X_j,\mathbf{Y}_j|k}\boldsymbol{\Sigma}^{-1}_{\mathbf{Y}_j|k}\boldsymbol{\mu}_{\mathbf{Y}_j|k}\ , \\ \boldsymbol{\beta}_{X_j|\mathbf{Y}_j,k} &= \boldsymbol{\Sigma}^{-1}_{\mathbf{Y}_j|k}\boldsymbol{\Sigma}_{\mathbf{Y}_j,X_j|k}\ , \\ \sigma^2_{X_j|\mathbf{Y}_j,k} &= \Sigma_{X_j|k} - \boldsymbol{\Sigma}_{X_j,\mathbf{Y}_j|k}\boldsymbol{\Sigma}^{-1}_{\mathbf{Y}_j|k}\boldsymbol{\Sigma}_{\mathbf{Y}_j,X_j|k}\ , \end{aligned} \tag{6}$$

where $\mu_{X_j|k}$ and $\boldsymbol{\mu}_{\mathbf{Y}_j|k}$ are the mean values of variables X_j and $\mathbf{Y}_j$ given the class label k. Therefore, we only need to estimate, for each class label k, the covariances of each variable with its parents ($\boldsymbol{\Sigma}_{X_j,\mathbf{Y}_j|k}$), and the covariances between the parents of the same variables ($\boldsymbol{\Sigma}_{\mathbf{Y}_j|k}$) in the maximization step (5).

In this paper, we consider four BCs with different structures, and fit their parameters with GEM:

- The naive Bayes (NB) classifier [8] assumes that all the predictive variables are conditionally independent given the class variable. Therefore, the covariance matrices for each component are reduced to diagonal matrices, so only the main diagonal of $\boldsymbol{\Sigma}_{\mathbf{X}|k}$ has to be estimated for each class label k in GEM (5). The updating equations for the mean values $\boldsymbol{\mu}_{\mathbf{X}|k}$ of the conditional densities in (5) are unchanged.
- The averaged one-dependence estimators (AODE) classifier [9] learns n BCs with a tree-augmented naive Bayes (TAN) structure [4]. The variable X_j is a parent of all the other predictive variables in the jth BC. When classifying a new instance, we compute the posterior probability of each class label as the mean of the posterior probabilities yielded by each TAN classifier.

- The structural EM (SEM) algorithm [10] is used to find the structure of the BC. Since we want to find the conditional (in)dependence relationships between the predictive variables given the class variable, we search for the structure in the space of the predictive variables. The SEM algorithm alternates between a structural search step and a parameter search step. The structural search starts with a NB structure and greedily evaluates all the possible structures that can be obtained by adding, deleting or reversing an arc between two predictive variables. The arcs from the class variable C to each predictive variable X_j are fixed. In the parametric search, the GEM algorithm finds the maximum likelihood estimates of the parameters. The process is iterated until there is no further increase in the BIC score. We implemented a BIC score which uses the generalized LL (3): $BIC(\mathcal{M} : D) = \ln(p(\boldsymbol{\Theta}|D)) - 0.5 dim(\mathcal{M}) \ln N$, where $\mathcal{M}$ represents a classifier and $dim(\mathcal{M})$ is the number of free parameters in the classifier.
- The performance of BCs is known to suffer when irrelevant or redundant variables are not removed from the problem. Therefore, we have also considered including feature subset selection in the structural search step of the SEM algorithm. We call this algorithm the feature subset selected structural EM (FSSSEM). FSSSEM includes the class variable in the structural search step and introduces some restrictions to ensure that the BC structure is valid. First, arcs including the class variable have to be directed towards the predictive variables and cannot be reversed. Second, if an arc from the class variable to a predictive feature is deleted, we consider that the variable has been erased and we delete all the arcs including the predictive variable. Like SEM, the search procedure alternates between the structural search and the parameter search steps until the generalized BIC score does not increase.

When classifying a new instance $\mathbf{x}$, any BC yields a posterior probability $p(C = k|\mathbf{X} = \mathbf{x})$ for each class label k. We use the maximum a posteriori decision rule, so that $\mathbf{x}$ is assigned to the class with maximum posterior probability $k^* = \arg\max_{k \in \Omega_C} p(C = k|\mathbf{X} = \mathbf{x})$.

4 Experiments

This section includes the evaluation of the classifiers on eight datasets taken from the UCI[1] and KEEL[2] repositories (see Fig. 1). Each variable in the datasets was standardized by subtracting the mean and dividing by the standard deviation. We erased the eighth variable in the `glass` dataset because 82.24% of the values were zero and the estimated covariance matrices were not positive definite in some runs. Also, we erased the first variable in the `ion` dataset because it was discrete.

Five classifiers (MGC, NB, AODE, SEM and FSSSEM) were learned in four different scenarios according to the available data and the algorithm used:

[1] Available at: http://archive.ics.uci.edu/ml/

[2] Available at: http://keel.es

- GEM: The parameters of the BCs were found using the probability distribution for the class labels with the GEM algorithm.
- EM: The parameters of the BCs were found with the classical EM algorithm [6]. The probability distributions for the class labels were used to initialize $t_{i,k}^{(1)}$ in the E-step of the first iteration of the algorithm.
- Wrong labels (WL): The BCs were fitted as in a common supervised classification problem, but the class labels of some instances were flipped to a wrong label (see Sect. 4.1).
- True labels (TL): The BCs were fitted using the true class labels of the instances. This corresponds to an utopian scenario where the class labels of the instances are known to be correct.

4.1 Dataset Generation and Stratified l-Fold Cross-Validation

We artificially modified the real datasets by transforming the true class label of each instance into a probability distribution over the class labels. For each instance, we sampled a value b_i from a beta distribution with mean μ_B and standard deviation σ_B. If the true class label for the ith instance was k, then we set $\pi_{i,k} = 1 - b_i$ and $\pi_{i,k'} = b_i/(K-1), k' \neq k$. The beta distribution models the mistakes made by the experts when classifying the instances. The probability of the true class label was high with low values of μ_B, whereas high values of μ_B yielded probability distributions where the true class label did not have the maximum probability. Similarly, in the WL setting, we randomly modified the class label for some instances in the dataset. For each instance, we drew a value u_i from a uniform distribution in $[0,1]$. If $u_i < b_i$, then the true class label was changed to any other class label in Ω_C with equal probability.

Stratified l-fold cross-validation was used to honestly estimate the classification error of the models. We assumed that the true class label of the instances was not available, so we based the stratified cross-validation on the probability distributions over the class labels. We proposed a simple greedy algorithm for generating the folds in the cross-validation process. The goal was to generate folds with the same mean probabilities as the complete dataset. First, for each class label k, the instances were ranked in decreasing order using the probabilities $\pi_{i,k}$. Then, a mean rank was computed for each instance using $K-1$ rankings. We ordered the instances according to the mean rank and assigned each instance to the fold with the lowest sum of mean ranks at any time. The proposed stratified l-fold cross-validation algorithm yielded folds with similar proportions to the complete dataset, even when the class labels were unbalanced (not shown). Once the folds were generated, we proceeded as in a classical stratified cross-validation setting. Each fold was considered once to test the classifier learned using the other $l-1$ folds. The estimated error of the classifier was the mean of the errors of the classifiers learned for each fold.

4.2 Results

Figure 1 shows the mean classification error achieved in each dataset for different values of $\mu_B = \{0.1, 0.2, 0.3, 0.4\}$ in the beta distribution used to generate

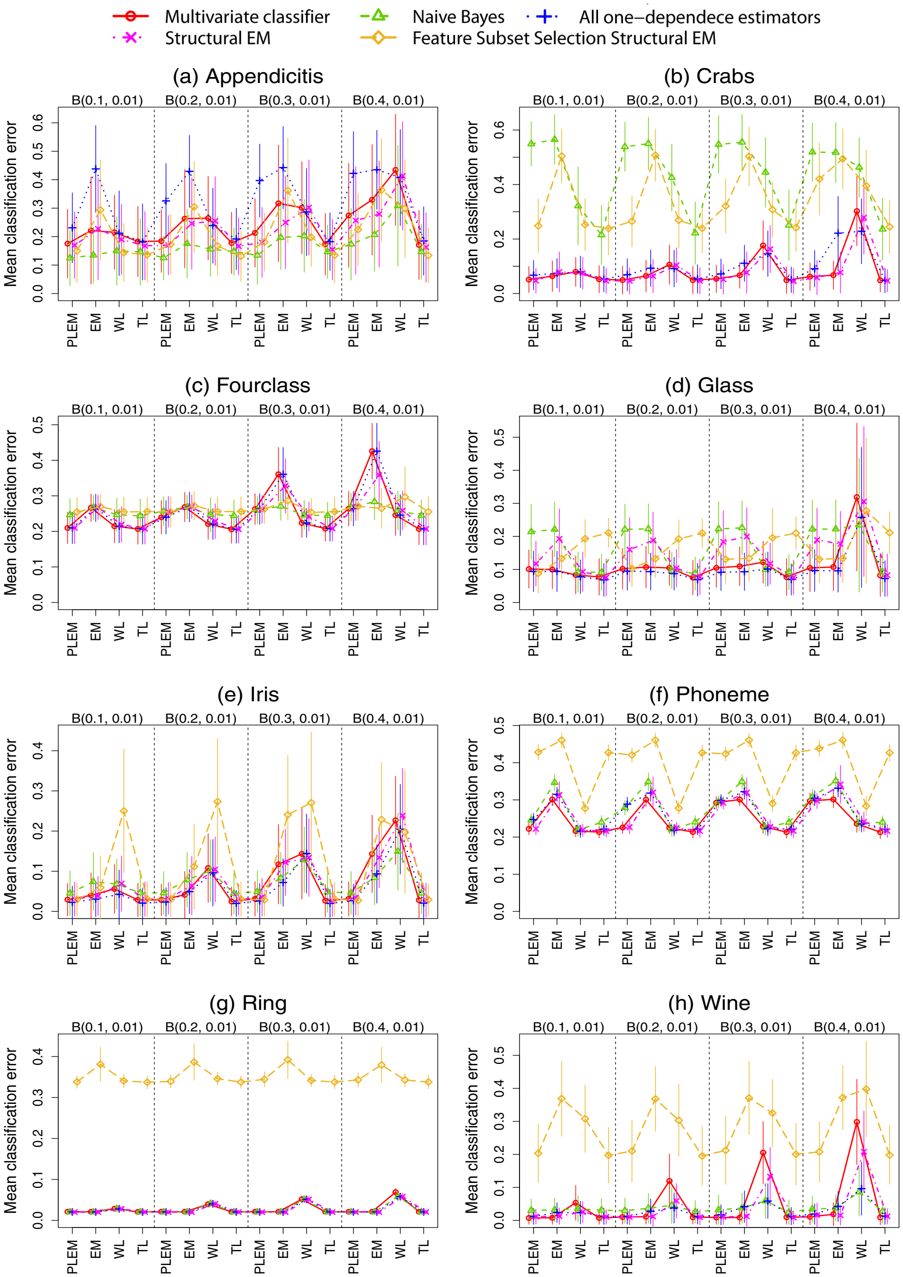

Fig. 1. Mean classification error and standard deviation bars in ten repetitions of a 10-fold stratified cross-validation procedure. The probability distributions for the class labels $\boldsymbol{\pi}_i$ were generated from beta distributions $B(\mu_B, 0.01)$ with $\mu_B = \{0.1, 0.2, 0.3, 0.4\}$.

Table 2. Comparison between algorithms considering the results for all the classifiers (5), datasets (8) and values of μ_B (4).

H_0	W/T/L	Binomial test		Bergmann-Hommel test	
		H_1	p-value	H_1	p-value
GEM = EM	141/1/18	$> *$	0.0000	$\neq *$	0.0000
GEM = WL	104/0/56	$> *$	0.0000	$\neq *$	0.0000
GEM = TL	32/5/123	$< *$	0.0000	$\neq *$	0.0000
EM = WL	80/0/80	$\neq$	1.0000	$\neq$	0.1530
EM = TL	20/0/140	$< *$	0.0000	$\neq *$	0.0000
WL = TL	16/0/144	$< *$	0.0000	$\neq *$	0.0000

the artificial datasets. The value $\sigma_B = 0.01$ was used in all experiments. We performed ten repetitions of 10-fold stratified cross-validation and computed the classification error with respect to the true class labels of the instances. The GEM algorithm frequently outperformed the classical EM algorithm, the only exception being the `ring` dataset, where EM and GEM algorithms won 10 times each. Interestingly, GEM achieved better results than TL in some experiments, e.g., MGC in `appendicitis` and `crabs` datasets, or NB in `appendicitis`, `iris`, `ring` and `wine` datasets ($\mu_B = 0.1$). A possible explanation is that GEM uses the information about an instance to estimate the parameters of the conditional probabilities for all the class labels where the probability $\pi_{i,k}$ is higher than zero. Therefore, more information was available to fit the classifiers and higher accuracies could be achieved. The accuracy in the WL scenario deteriorated as we increased the mean value of the beta distributions, e.g., in the `appendicitis`, `crabs`, `glass`, `iris` or `wine` datasets. On the contrary, the accuracy of GEM remained rather stable or decreased slightly (e.g., `fourclass` or `phoneme`) when increasing μ_B. These behaviors could be observed for all but the classifiers learned with FSSSEM. In general, MGC, SEM and AODE were the classifiers that performed better for the different algorithms and datasets. NB yielded poor results in the `crabs` and `glass` datasets but seemed to outperform the other classifiers in `appendicitis`. FSSSEM's performance was not very good across all the datasets.

Table 2 compares the four learning algorithms (GEM, EM, WL and TL). The number of times the first algorithm wins, ties or loses against the second are shown. The binomial test checks whether or not the number of wins is equal to the number of losses. The non-parametric Bergmann-Hommel post-hoc test [11] checks whether or not the mean accuracy of the methods is the same. The p-value and the alternative hypothesis (H_1) are reported for each test. Statistically significant results at $\alpha = 0.05$ are shown with an asterisk. We found that GEM significantly outperformed both EM and WL. Not surprisingly, we found significant differences between TL and all the other learning scenarios. On the other hand, no significant differences were found between EM and WL.

Similarly, Table 3 compares the five BCs. We did not find significant differences between the pairwise performances of AODE, MGC and SEM. All the

Table 3. Comparison between classifiers considering the results for all the algorithms (4), datasets (8) and values of μ_B (4)

H_0	W/T/L	Binomial test		Bergmann-Hommel test	
		H_1	p-value	H_1	p-value
MGC = NB	80/0/48	$> *$	0.0030	$\neq *$	0.0022
MGC = AODE	49/17/62	$<$	0.1273	$\neq$	1.0000
MGC = SEM	57/7/64	$<$	0.2928	$\neq$	1.0000
MGC = FSSSEM	108/1/19	$> *$	0.0000	$\neq *$	0.0000
NB = AODE	41/0/87	$< *$	0.0000	$\neq *$	0.0003
NB = SEM	39/14/75	$< *$	0.0005	$\neq *$	0.0008
NB = FSSSEM	89/0/39	$> *$	0.0000	$\neq *$	0.0000
AODE = SEM	59/5/64	$<$	0.3593	$\neq$	1.0000
AODE = FSSSEM	108/0/20	$> *$	0.0000	$\neq *$	0.0000
SEM = FSSSEM	100/1/27	$> *$	0.0000	$\neq *$	0.0000

classifiers significantly outperformed FSSSEM according to both tests. The feature subset selection method in FSSSEM is rather naive and uninformative and the number of selected variables in the final classifiers was usually low. This could explain FSSSEM's poor performance. Also, NB was outperformed by MGC, AODE and SEM.

5 Conclusions

In this paper we have adapted the GEM algorithm [1] to the particular scenario where the information about the class of the training instances is given as probability distributions over the class labels. We used this particularization of the GEM algorithm to learn Bayesian network classifiers with different structural complexities: multivariate Gaussian classifiers, naive Bayes, AODE or conditional linear Gaussian classifiers. We evaluated the classifiers on eight real datasets. BCs learned with GEM outperform others learned with the classical EM algorithm or with potentially wrong labels. We found no significant differences between the performances of MGC, AODE and CLG classifiers learned with SEM. In general, both AODE and SEM require a lower number of parameters than MGC to be estimated from data. Therefore, these classifiers might be more appropriate when the number of instances in the training datasets are low with respect to the number of variables.

Future work includes the extension to other BCs, e.g., TAN, k-DB, selective NB, etc. These methods are far more efficient than SEM or FSSSEM because the structural search uses the conditional mutual information between the predictive variables to find interrelationships. However, estimating the conditional mutual information between two variables when the class values are provided as probability distributions or belief functions is a matter of research. Other more informative methods could be used for feature subset selection. Adapting classical measures of the information that a variable (or a set of variables) provides about the class, such as the mutual information or correlation-based measures, to work with uncertain class labels is also challenging. In this paper, we have only considered score+search methods for learning BCs. However, there are approaches which are based on statistical tests for conditional independence between the variables. To the best of our knowledge, how to adapt conditional independence tests to work with probabilistic class labels is also an open question.

Finally, the GEM algorithm does not explicitly model class uncertainty, i.e., the probabilities $\pi_{i,k}$ remain constant throughout the whole algorithm and they do not appear in the final model (1). Other approaches that explicitly model these probabilities (e.g., using Dirichlet distributions) would be useful for studying and considering the interactions between the different class values.

Acknowledgments. This work has been partially supported by the Spanish Ministry of Economy and Competitiveness through Cajal Blue Brain (C080020-09) and TIN2010-20900-C04-04 projects. PLLC is supported by a Fellowship (FPU AP2009-1772) from the Spanish Ministry of Education, Culture and Sport.

References

1. Côme, E., Oukhellou, L., Denoeux, T., Aknin, P.: Learning from partially supervised data using mixture models and belief functions. Pattern Recognit. 42, 334–348 (2009)
2. Smets, P., Kennes, R.: The transferable belief model. Artif. Intell. 66, 191–243 (1994)
3. Pearl, J.: Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann (1988)
4. Friedman, N., Goldszmidt, M., Lee, T.J.: Bayesian network classification with continuous attributes: Getting the best of both discretization and parametric fitting. In: Shavlik, J.W. (ed.) Proceedings of the 15th ICML, pp. 179–187. Morgan Kaufmann (1998)
5. Vannoorenberghe, P., Smets, P.: Partially supervised learning by a credal EM approach. In: Godo, L. (ed.) ECSQARU 2005. LNCS (LNAI), vol. 3571, pp. 956–967. Springer, Heidelberg (2005)
6. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. Ser. B-Stat. Methodol. 39, 1–38 (1977)
7. Lauritzen, S.L., Wermuth, N.: Graphical models for associations between variables, some of which are qualitative and some quantitative. Ann. Stat. 17, 31–57 (1989)
8. Minsky, M.: Steps toward artificial intelligence. Proc. Inst. Radio Eng. 49, 8–30 (1961)
9. Webb, G.I., Boughton, J.R., Wang, Z.: Not so naive Bayes: Aggregating one-dependence estimators. Mach. Learn. 58, 5–24 (2005)
10. Friedman, N.: Learning belief networks in the presence of missing values and hidden variables. In: Fisher, D.H. (ed.) 14th ICML, pp. 125–133. Morgan Kaufmann (1997)
11. García, S., Herrera, F.: An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. J. Mach. Learn. Res. 9, 2677–2694 (2008)