Robust Bayesian Linear Classifier Ensembles

Jesús Cerquides[1] and Ramon López de Màntaras[2]

[1] Dept. de Matemàtica Aplicada i Anàlisi, Universitat de Barcelona
cerquide@maia.ub.es
[2] Artificial Intelligence Research Institute - IIIA,
Spanish Council for Scientific Research - CSIC
mantaras@iiia.csic.es

Abstract. Ensemble classifiers combine the classification results of several classifiers. Simple ensemble methods such as uniform averaging over a set of models usually provide an improvement over selecting the single best model. Usually probabilistic classifiers restrict the set of possible models that can be learnt in order to lower computational complexity costs. In these restricted spaces, where incorrect modeling assumptions are possibly made, uniform averaging sometimes performs even better than bayesian model averaging. Linear mixtures over sets of models provide an space that includes uniform averaging as a particular case. We develop two algorithms for learning maximum a posteriori weights for linear mixtures, based on expectation maximization and on constrained optimizition. We provide a nontrivial example of the utility of these two algorithms by applying them for one dependence estimators. We develop the conjugate distribution for one dependence estimators and empirically show that uniform averaging is clearly superior to Bayesian model averaging for this family of models. After that we empirically show that the maximum a posteriori linear mixture weights improve accuracy significantly over uniform aggregation.

1 Introduction

An ensemble of classifiers is a set of classifiers whose individual decisions are combined in some way (typically by weighted or unweighted voting) to classify new examples. Uniform averaging and other improper linear models have been demonstrated to be better than selecting a single best model [5].

Bayesian model averaging (BMA) [19,20] provides a coherent, theoretically optimal mechanism for accounting with model uncertainty. BMA, under the name Bayesian voting, is commonly understood as a method for learning ensembles [6]. With some exceptions [4,2], the application of BMA in machine learning has not proven as successful as expected [7]. A reasonable explanation of this mismatch between expected and real performance of BMA has been given in a short note by Minka [27], where it is clearly pointed out that BMA is not a model combination technique, and that it should be thought of as a method for 'soft model selection'. This understanding has led to the proposal of techniques for the bayesian combination of classifiers [13]. In spite of that, BMA is still being considered by many scientists as an ensemble learning technique and as such it is compared with other ensemble learning techniques such as stacking, bagging or boosting [3,9].

J. Gama et al. (Eds.): ECML 2005, LNAI 3720, pp. 72–83, 2005.

Accepting BMA as 'soft model selection', it can happen that uniform averaging improves over BMA when modeling assumptions are incorrect. Many times this is the case when classifiers are applied "out-of-the-box". However, an ensemble classifier should be able to recognize which models are right and which are incorrect. In order to do that, we propose two algorithms for adjusting the weights for a linear mixture of classifiers and are robust to incorrect modeling assumptions of the base classifiers.

The issue of generative versus discriminative classifiers has raised a lot of attention in the community in the last years [28,1,30,16,15,31]. It is widely believed that, provided enough data, discriminative classifiers outperform their generative pairs. Since both generative and discriminative classifiers are in use nowadays, two different initial settings are assumed in order to construct a linear ensemble of classifiers. In the first one, we are given a set of base classifiers that after receiving an unclassified observation, output the conditional probability distribution for each class. On the second setting, our base classifiers are assumed to output the joint probability for each class and the observation (instead of contioned to the observation). We could name the first setting linear averaging of discriminative classifiers and the second linear averaging of generative classifiers. We propose the usage of an expectation maximization algorithm for the first setting. The second setting is tougher and we propose the usage of augmented lagrangian techniques [14,29] for constrained nonlinear optimization.

In the last years there have been several attempts to improve the naive Bayes classifier by relaxing its restrictive independence assumption [10,22,38,2]. Averaged One Dependence Estimators (AODE) classifiers have been proposed [36] as an efficient and effective alternative to naive Bayes. They are based on k-dependence estimators [32], which are classifiers where the probability of each attribute value is conditioned by the class and at most k other attributes. AODE classifiers estimate the class probabilities by performing an equally weighted linear combination of the estimates of all possible 1-dependence estimators. Since AODE is a classifier based on uniform aggregation of simple classifiers that make very hard assumptions that are likely not to be fulfilled, it can act as a good test case for our algorithms. We describe AODE in section 4. In section 5 we find a conjugate distribution for the problem and we prove that it is possible to perform exact BMA over the set of 1-dependence estimators in polynomial time. After that, in section 6 we adapt our weight adjustment algorithms for ODEs and finally in section 7 we empirically compare the results of BMA with uniform averaging and our two linear mixtures, obtaining results that clearly confirm the previously exposed ideas.

In [33,34] a Bayesian technique for finding maximum likelihood ensembles of Bayesian networks is described. In [26,25] an EM algorithm for finding linear mixtures of trees is proposed. Those works were stated in the setting of density estimation (mixtures were learned with a generative approach in mind) and our explicitly deals with the problem of classification or conditional density estimation (discriminative approach). Ghahramani et al. [13] presented Bayesian methods for averaging classifiers. They assume the predicted class to be the only information available as output from the classifiers to be averaged. We assume a bit more and ask classifiers to output a probability distribution. This setting was already proposed by them as a rellevant line of future work.

To summarize, the main contribution of the paper is the proposal of two maximum a posteriori algorithms for averaging probability distributions in a supervised setting.

As side results, we provide the conjugate distribution for ODEs and empirically confirm the limitations of BMA when understood as an ensemble learning technique in a nontrivial case. A more detailed study of the two algorithms proposed and a comparison with other general ensemble learning methods will be the subject of future work.

2 Formalization and Notation

A *discrete attribute* is a finite set, for example we can define attribute $Pressure$ as $Pressure = \{Low, Medium, High\}$. A *discrete domain* is a finite set of discrete attributes. We denote $\Omega_C = \{A_1, \ldots, A_n, C\}$ for a classified discrete domain where A_u are attributes other than the class and C is the class attribute. We will use i and j as values of an attribute and u and v as indexes over attributes in a domain. We denote $X_{-C} = \{A_1, \ldots, A_n\}$ the set that contains all the attributes in a classified discrete domain except the class attribute.

Given an attribute X, we denote $\#X$ as the number of different values of X. An *observation* x in Ω_C is an ordered tuple $x = (x_1, \ldots, x_n, x_C) \in A_1 \times \ldots \times A_n \times C$. An *unclassified observation* x_{-C} in Ω_C is an ordered tuple $x_{-C} = (x_1, \ldots, x_n) \in A_1 \times \ldots \times A_n$. To be homogeneous we will abuse this notation a bit noting x_C for a possible value of the class for x_{-C}. A *dataset* $\mathcal{D}$ in Ω_C is a multiset of classified observations in Ω_C.

We will denote N for the number of observations in the dataset. We will also denote $N_u(i)$ for the number of observations in $\mathcal{D}$ where the value for A_u is i, $N_{u,v}(i,j)$ the number of observations in $\mathcal{D}$ where the value for A_u is i and the value for A_v is j and similarly for $N_{u,v,w}(i,j,k)$ and so on.

3 Learning Mixtures of Probability Distributions

In order to aggregate the predictions of a set of different models, we can use a linear mixture assigning a weigth to each model. If modeling assumptions are correct, BMA provides the best linear mixture. Otherwise, uniform averaging has been demonstrated to improve over single model selection and many times also over BMA. We would like to develop an algorithm for assigning weigths to models in a linear mixture that improves over uniform averaging while being robust to incorrect modeling assumptions of the base classifiers.

3.1 Formalization of the Problem

On a classified discrete domain Ω_C, we define two different types of probability distributionss. A generative probability distribution (GPD) is a probability distribution over Ω_C. A discriminative probability distribution (DPD) is a probability distribution over C given X_{-C}. Obviously, from every GPD, we can construct a DPD, but not *vice versa*.

A linear mixture of n DPDs (LMD in the following) is defined by the equation:

$$P_{LMD}(x_C|x_{-C}) = \sum_{u=1}^{n} \alpha_u P_{DPD_u}(x_C|x_{-C}). \quad (1)$$

The model is more widely known as the *linear opinion pool* [12,11].

A linear mixture of n GPDs (LMG in the following) is defined by the equation:

$$P_{LMG}(x_C, x_{-C}) = \sum_{u=1}^{n} \alpha_u P_{GPD_u}(x_C, x_{-C}), \tag{2}$$

in both cases $\sum_{u=1}^{n} \alpha_u = 1$ and $\forall u\ \alpha_u > 0$.

Supervised Posteriors. From a frequentistic point of view, in order to learn conditional probability distributions we need to maximize conditional likelihood. In [17] the concept of *supervised posterior* is introduced as a Bayesian response to this frequentistic idea. The proposal in [17] is that from a Bayesian point of view, in order to learn conditional probability distributions, given a family of models $\mathcal{M}$, we need to compute the BMA over models using the supervised posterior $P^s(M|\mathcal{D})$:

$$P(x_C|x_{-C}, \mathcal{D}, \xi) = \int_{M \in \mathcal{M}} P(x_C|x_{-C}, M, \xi) P^s(M|\mathcal{D}, \xi), \tag{3}$$

where

$$P^s(M|\mathcal{D}, \xi) = P^s(\mathcal{D}|M, \xi) P(M|\xi) \tag{4}$$

and

$$P^s(\mathcal{D}|M, \xi) = \prod_{x \in \mathcal{D}} P(x_C|x_{-C}, M, \xi). \tag{5}$$

Supervised posterior for LMD. In order to perform Bayesian learning over LMD and LMG we define a prior distribution over $\boldsymbol{\alpha}$. A natural choice in this case is to use a Dirichlet distribution. For conciseness we fix the Dirichlet hyperparameters to 1, that is $P(\boldsymbol{\alpha}|\xi) \propto \prod_{u=1}^{n} \alpha_u$, although the development can be easily generalized to any Dirichlet prior. The supervised posterior after an i.i.d. dataset $\mathcal{D}$ for a LMD is:

$$P_{LMD}(\boldsymbol{\alpha}|\mathcal{D}, \xi) = \frac{P(\mathcal{D}|\boldsymbol{\alpha}, \xi) P(\boldsymbol{\alpha}|\xi)}{P(\mathcal{D}|\xi)} = \frac{\prod_{x \in \mathcal{D}} P(x_C|x_{-C}, \boldsymbol{\alpha}, \xi) P(x_{-C}|\boldsymbol{\alpha}, \xi) P(\boldsymbol{\alpha}|\xi)}{P(\mathcal{D}|\xi)} =$$

$$= \prod_{x \in \mathcal{D}} P(x_C|x_{-C}, \boldsymbol{\alpha}, \xi) P(\boldsymbol{\alpha}|\xi) \frac{\prod_{x \in \mathcal{D}} P(x_{-C}|\boldsymbol{\alpha}, \xi)}{P(\mathcal{D}|\xi)}. \tag{6}$$

Assuming that $P(x_{-C}|\boldsymbol{\alpha}, \xi)$ does not depend on $\boldsymbol{\alpha}$ we can conclude that

$$P_{LMD}(\boldsymbol{\alpha}|\mathcal{D}, \xi) \propto \prod_{x \in \mathcal{D}} \sum_{u=1}^{n} \alpha_u P_{DPD_u}(x_C|x_{-C}) \prod_{u=1}^{n} \alpha_u. \tag{7}$$

The exact BMA prediction in this setting will be given by:

$$P_{LMD}(x_C|x_{-C}, \mathcal{D}, \xi) = \int_{\boldsymbol{\alpha}} P_{LMD}(\boldsymbol{\alpha}|\mathcal{D}, \xi) \sum_{u=1}^{n} \alpha_u P_{DPD_u}(x_C|x_{-C}) d\boldsymbol{\alpha}. \tag{8}$$

Supervised Posterior for LMG. The supervised posterior after an i.i.d. dataset $\mathcal{D}$ for LMG is

$$P_{LMG}(\boldsymbol{\alpha}|\mathcal{D},\xi) = \prod_{x\in\mathcal{D}} \frac{\sum_{u=1}^{n} \alpha_u P_{GPD_u}(x_C, x_{-C})}{\sum_{c\in C}\sum_{u=1}^{n} \alpha_u P_{GPD_u}(c, x_{-C})} \prod_{u=1}^{n} \alpha_u, \tag{9}$$

and the exact BMA prediction in this setting

$$P_{LMG}(x_C|x_{-C},\mathcal{D},\xi) = \int_{\boldsymbol{\alpha}} P_{LMG}(\boldsymbol{\alpha}|\mathcal{D},\xi) \frac{\sum_{u=1}^{n} \alpha_u P_{GPD_u}(x_C, x_{-C})}{\sum_{c\in C}\sum_{u=1}^{n} \alpha_u P_{GPD_u}(c, x_{-C})} d\boldsymbol{\alpha}. \tag{10}$$

3.2 Proposed Solutions

MAPLMD. To the best of our knowledge there is no closed form solution for computing the result of equation 8. Hence, we will have to approximate its value. A first possibility would be to directly approximate it using Markov Chain Monte Carlo (MCMC). However, each iteration of the model will require the computation of the product in equation 8 that ranges over all the observations in the dataset, resulting in a heavy use of computational resources. A second possibility is approximating the expression using only the maximum a posteriori (MAP) value for $\boldsymbol{\alpha}$ (which we denote $\boldsymbol{\alpha}^{\boldsymbol{MAP}}_{\boldsymbol{LMD}}$) as

$$P(x_C|x_{-C},\mathcal{D},\xi) \approx \sum_{u=1}^{n} \alpha_u{}^{MAP}_{LMD} P_{DPD_u}(x_C|x_{-C}). \tag{11}$$

It is known [24,23] that, since we are dealing with a finite mixture model, we can determine $\boldsymbol{\alpha}^{\boldsymbol{MAP}}_{\boldsymbol{LMD}}$ by means of the Expectation-Maximization (EM) algorithm by posing the problem into an incomplete-data one introducing an additional unobservable variable for each observation corresponding to the mixture component that generated the data. This gives us a reasonably efficient procedure for determining $\boldsymbol{\alpha}^{\boldsymbol{MAP}}_{\boldsymbol{LMD}}$. The aggregation method resulting from finding $\boldsymbol{\alpha}^{\boldsymbol{MAP}}_{\boldsymbol{LMD}}$ and then applying it in equation 1 is MAPLMD.

MAPLMG. The case of LMG is not so simple. As we did for LMD, we can approximate the exact BMA prediction using only the MAP value for $\boldsymbol{\alpha}$ (that we denote $\boldsymbol{\alpha}^{\boldsymbol{MAP}}_{\boldsymbol{LMG}}$). However, in this case, there is no straightforward way to use the EM algorithm. From an optimization point of view, we have to find $\boldsymbol{\alpha}^{\boldsymbol{MAP}}_{\boldsymbol{LMG}}$, under the inequality constraints that each component of the vector $\boldsymbol{\alpha}^{\boldsymbol{MAP}}_{\boldsymbol{LMG}}$ should be greater that 0 and the equality constraint that the components of $\boldsymbol{\alpha}^{\boldsymbol{MAP}}_{\boldsymbol{LMG}}$ should add up to 1. This is a constrained nonlinear optimization problem that can be solved by using the augmented (or penalized) lagrangian method [14,29] for constrained nonlinear optimization. This method transforms a constrained nonlinear optimization problem into a sequence of unconstrained optimization problems, progresively adjusting the penalization provided

by not fulfilling the constraints. For solving each of the resulting unconstrained optimization problems several efficient methods are available. In our case we have used the well known Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. It is a quasi-Newton method which builds up an approximation to the second derivatives of the function using the difference between successive gradient vectors. By combining the first and second derivatives the algorithm is able to take Newton-type steps towards the function minimum, assuming quadratic behavior in that region. This technique requires the computation of the partial derivative of the function to be optimized with respect to each of the α_i. Fortunately this can be done efficiently if we calculate it together with the function. By simple algebraic manipulations it can be seen that the derivative of equation 9 is:

$$\frac{\partial P(\boldsymbol{\alpha}|\mathcal{D},\xi)}{\partial \alpha_u} = P(\boldsymbol{\alpha}|\mathcal{D},\xi)\left(\sum_{x\in\mathcal{D}} \frac{p_{u,x_C}\sum_{u=1}^{n}\alpha_u p_u - p_u\sum_{u=1}^{n}\alpha_u p_{u,x_C}}{\sum_{u=1}^{n}\alpha_u p_u \sum_{u=1}^{n}\alpha_u p_{u,x_C}} + \frac{1}{\alpha_u}\right), \quad (12)$$

where $p_{u,c} = P_{GPD_u}(c, x_{-C})$ and $p_u = \sum_{c\in C} P_{GPD_u}(c, x_{-C})$. In order to complete the Lagrangian, we also need to compute the derivatives of the constraints, $\sum u = 1^n \alpha_u = 1$ and $\forall u\ \alpha_u > 0$, with respect to each α_u, that are very simple. The aggregation method resulting from finding $\boldsymbol{\alpha}_{\boldsymbol{LMG}}^{\boldsymbol{MAP}}$ and then predicting using equation 2 is named MAPLMG.

4 AODE

In this section we review the AODE classifier as presented in [36]. Given a classified domain, AODE learns a set of 1-dependence probability distribution estimators (ODE) containing those where the class attribute and another single attribute are the parents of all other attributes. Obviously there are n ODEs satisfying our condition, one for each choice of root attribute. The probablity estimates for an ODE are:

$$P_u(x) = P_u(x_C, x_{-C}) = P_u(x_C, x_u)\prod_{\substack{v=1\\ v\neq u}}^{n} P_u(x_v|x_C, x_u), \quad (13)$$

where $P_u(x_C, x_u) = \frac{N_{C,u}(x_C,x_u)+1}{N+\#C\#A_u}$ and $P_u(x_v|x_C, x_u) = \frac{N_{C,u,v}(x_C,x_u,x_v)+1}{N_{C,u}(x_C,x_u)+\#A_v}$ (these equations are slightly different to the ones presented in [36] and correspond to the AODE classifier implemented in Weka[37] version 3.4.3). After learning these models , AODE uniformly combines the probabilities for each of them:

$$P_{AODE}(x_C, x_{-C}) = \sum_{\substack{u=1\\ N_u(x_u)>t}}^{n} P_u(x_C, x_{-C}). \quad (14)$$

In equation 14, the condition $N_u(x_u) > t$ is used as a threshold in order to avoid making predictions from attributes having few observations. If no attribute fulfills the condition, AODE returns the results of predicting using naive Bayes.

5 Exact Bayesian Model Averaging of ODEs

In this section we provide a conjugate distribution for ODEs and show how it can be used to efficiently perform BMA over ODEs.

5.1 Conjugate Distribution for One Dependence Estimators

In order to define a probability distribution over ODEs, we define how we compute the probability that an ODE is the generating model. After that, we define the probability distribution over the parameters of that ODE. Probability distribution over the parameters of two different ODEs u and v (denoted ${}^u\boldsymbol{\Theta}$ and ${}^v\boldsymbol{\Theta}$) are assumed independent.

Definition 1 (Decomposable distribution over ODEs). *The probability of an ODE with concrete structure and parameters under a decomposable distribution over ODEs with hyperparameters* $\boldsymbol{\alpha}, \boldsymbol{N'} = \bigcup_{u=1}^{n} {}^u\boldsymbol{N'}$ *is the product of the probability that its root is the selected root* ($P(\rho_B|\xi)$) *times the probability that its parameters are the right parameters* ($P({}^{\rho_B}\boldsymbol{\Theta}|\xi)$)*:*

$$P(B|\xi) = P(\rho_B|\xi)P({}^{\rho_B}\boldsymbol{\Theta}|\xi). \tag{15}$$

The probability distribution for the root is a multinomial with hyperparameter $\boldsymbol{\alpha}$*. The probability for the parameter set ,*${}^u\boldsymbol{\Theta}$*, for each possible root* u *factorizes following the ODE structure:*

$$P({}^u\boldsymbol{\Theta}|\xi) = P({}^u\boldsymbol{\theta}_{u,C}|\xi) \prod_{\substack{v=1 \\ v\neq u}}^{m} P({}^u\boldsymbol{\theta}_{v|u,C}|\xi) \tag{16}$$

and the distribution over each conditional probability table follows a Dirichlet distribution (where the needed hyperparameters are given by ${}^u\boldsymbol{N'}$*):*

$$P({}^u\boldsymbol{\theta}_{u,C}|\xi) = D({}^u\theta_{u,C}(.,.); {}^uN'_{u,C}(.,.)) \tag{17}$$

$$P({}^v\boldsymbol{\theta}_{v|u,C}|\xi) = D({}^u\theta_{v|u,C}(.,i,c); {}^uN'_{v,u,C}(.,i,c)) \tag{18}$$

.

5.2 Learning Under Decomposable Distributions over ODEs

If a decomposable distribution over ODEs is accepted as prior, we can efficiently calculate the posterior after a complete i.i.d. dataset:

Theorem 1. *If* $P(B|\xi)$ *follows a decomposable distribution over ODEs with hyperparameters* $\boldsymbol{\alpha}$,$\mathbf{N}'$*, the posterior distribution given an i.i.d. dataset D is a decomposable distribution over ODEs with hyperparameters* $\boldsymbol{\alpha}^*$,$\mathbf{N}'^*$ *given by:*

$$\alpha^*_u = \alpha_u W_u \tag{19}$$

$${}^uN'^*_{u,C}(i,c) = {}^uN'_{u,C}(i,c) + N_{u,C}(i,c) \tag{20}$$

$${}^uN'^*_{v,u,C}(j,i,c) = {}^uN'_{v,u,C}(j,i,c) + N_{v,u,C}(j,i,c) \tag{21}$$

where

$$W_u = \frac{\Gamma(N')}{\Gamma(N'^*)} \prod_{c \in C} \prod_{i \in A_u} \left[\frac{\Gamma({}^uN'^*_{u,C}(i,c))}{\Gamma({}^uN'_{u,C}(i,c))} \prod_{\substack{v=1 \\ v \neq u}}^{m} \left(\frac{\Gamma({}^{u,s(v)}N'_{u,C}(i,c))}{\Gamma({}^{u,s(v)}N'^*_{u,C}(i,c))} \prod_{j \in A_v} \frac{\Gamma({}^uN'^*_{v,u,C}(j,i,c))}{\Gamma({}^uN'_{v,u,C}(j,i,c))} \right) \right], \tag{22}$$

and

$${}^uN' = \sum_{c \in C} \sum_{i \in A_u} {}^uN'_{u,C}(i,c) \tag{23}$$

$${}^{u,s(v)}N'_{u,C}(i,c)) = \sum_{j \in A_v} {}^uN'_{v,u,C}(j,i,c), \tag{24}$$

and the equivalent of equations 23 and 24 hold for N'^*.

5.3 Classifying Under Decomposable Distributions over ODEs

Under a decomposable distribution over ODEs, we can efficiently calculate the probability of an observation by averaging over both structure and parameters:

Theorem 2. *If $P(B|\xi)$ follows a decomposable distribution over ODEs with hyperparameters $\boldsymbol{\alpha}$,$\mathbf{N}'$, the probability of an observation given ξ is*

$$P(\mathcal{X} = x|\xi) = \sum_{u=1}^{m} \alpha_u P(\mathcal{X} = x|\rho_B = u, \xi)$$

where $P(\mathcal{X} = x|\rho_B = u, \xi) = \frac{{}^uN'_{u,C}(x_u,x_C)}{{}^uN'} \prod_{\substack{v=1 \\ v \neq u}}^{m} \frac{{}^uN'_{v,u,C}(x_v,x_u,x_C)}{{}^{u,s(v)}N'_{u,C}(x_u,x_C)}$.

Theorems 1 and 2 demonstrate that exact learning can be performed in polynomial time under the assumption of decomposable distributions over ODEs. Furthermore, the overhead with respect to the standard AODE algorithm in terms of computational complexity can be considered very small. Proofs are omitted due to space limitations. For domains where we do not have prior information we will assign a value of 1 to each of the hyperparameters in $\boldsymbol{\alpha}$ and $\mathbf{N}'$. We name the resulting classifier BMAAODE.

6 Learning Mixtures of ODEs

It is worth noting that the development in section 3 was done under the assumption that the dataset $\mathcal{D}$ used for determining $\boldsymbol{\alpha}^{\boldsymbol{MAP}}$ is assumed to be independent of the dataset used to learn the individual classifiers. To allow the successful application of this results to ODEs, instead of using $P_u(c, x_{-C})$ as the probability distribution being averaged, we will use $P_u^{LOO}(c, x_{-C})$ (from Leave-One-Out), where the observation being classified (x) is excluded from the training set. After computing the counts $N_{C,u,v}(c,i,j)$,$N_{C,u}(c,x_u)$, and N, P_u^{LOO} is simply:

$$P_u^{LOO}(c, x_{-C}) = P_u^{LOO}(c, x_u) \prod_{\substack{v=1 \\ v \neq u}}^{n} P_u^{LOO}(x_v|c, x_u) \tag{25}$$

$$P_u^{LOO}(c, x_u) = \frac{N_{C,u}(c, x_u) + 1 - \delta(c = x_C)}{N + \#C\#A_u - 1} \tag{26}$$

$$P_u^{LOO}(x_v|c, x_u) = \frac{N_{C,u,v}(c, x_u, x_v) + 1 - \delta(c = x_C)}{N_{C,u}(c, x_u) + \#A_v - \delta(c = x_C)} \tag{27}$$

so almost no computational burden is introduced by this strategy. This can be understood as performing the best possible stacking [35] strategy with the data at hand, with an ODE for each attribute as the set of *level-0 models* and MAPLMD or MAPLMG as the *level-1 generalizer*. This particularization of MAPLMD and MAPLMG for ODE are named MAPLMDODE and MAPLMGODE respectively.

7 Empirical Results

In this section we compare AODE with BMAAODE, MAPLMGODE and MAPLMDODE on two different scenarios. On the first one we compare performance over Irvine datasets and on the secondone over randomly generated Bayesian networks with different sets of parameters. In the following sections, we explain the experimental setup and then show the results and draw some conclusions.

7.1 Experimental Setup

We used three different measures to compare the performance of the algorithms: the error rate, the conditional log-likelihood and the area under the ROC curve [8] which we will refer to as AUC. For this last measure, when the class is multivalued, we use the formula provided in [18].

Irvine Setup. We ran each algorithm on 38 datasets from the Irvine repository repeating 10 runs of 10 fold cross validation. Continuous attributes were discretized into 5 equal frequency intervals.

Random Bayesian Networks Setup. We compared the algorithms over random Bayesian networks varying the number of attributes in $\{5, 10, 20, 40\}$, the number of maximum values of an attribute in $\{2, 5, 10\}$ and the maximum induced width in $\{2, 3, 4\}$. For each configuration of parameters we generated randomly 100 Bayesian networks using BNGenerator [21]. For each Bayesian network we obtained 5 learning samples of sizes $\{25, 100, 400, 1600, 6400\}$ and a testing sample of size of 500.

7.2 Results and Conclusions

A summary of the results can be seen in tables 1 and 2. The tables describe the number of Wins/Draws/Loses at a 95% statistical t-test confidence level for each measure. AODE0 and AODE30 are two versions of AODE, with different thresholds $t = 0$ and $t = 30$ respectively. The results show that the condition $N_u(x_u) > t$ proposed in [36] although intuitively appealing, does not improve performance on none of both settings and can safely be simplified.

Table 1. Empirical results over Irvine datasets

Algorithms	AUC	ER	LogP
AODE0-AODE30	7/24/7	10/22/6	13/18/7
AODE0-BMAAODE	26/11/1	25/8/5	29/4/5
MAPLMGODE-AODE0	12/20/6	18/18/2	29/5/4
MAPLMDODE-AODE0	14/9/15	17/11/10	26/6/6

Table 2. Empirical results over random Bayesian networks

Algorithms	AUC	ER	LogP
AODE0-AODE30	38/124/18	45/128/7	85/92/3
AODE0-BMAAODE	101/77/2	90/83/7	143/26/11
MAPLMGODE-AODE0	155/24/1	138/41/1	151/17/12
MAPLMDODE-AODE0	176/4/0	145/27/8	177/2/1

It can be seen that BMAAODE performance is significantly worse than uniform aggregation in both settings. In order to understand the reason why, we note that in our Bayesian formalization of the problem an additional assumption has been introduced 'unnoticed': the assumption that one of the ODEs is the right model generating the data. This assumption has the effect that the posterior after a small number of observations concentrates most of its weight in a single model. AODE also makes a strong assumption: that the right model generating the data is a uniform aggregation of ODEs. This assumption turns out to be less restrictive that the one made by BMAAODE. Obviously, neither AODE nor BMAAODE assumptions are fulfilled by the datasets nor by the Bayesian networks used for the experimentation, but AODE is able to provide a better approximation than BMAAODE to their probability distributions most of the times. This result obviously does not change the fact that the assumption of a single generating model, as a generic assumption underlying Bayesian learning, is completely reasonable. However, it points out that we should be careful and understand that BMA provides the optimal linear ensemble only when the assumption is fulfilled.

Comparing AODE0 with MAPLMDODE and MAPLMGODE we can see that, with the only exception of MAPLMDODE over Irvine datasets and the AUC measure, both algorithms consistently improve AODE0 in a statistically significant way. Hence, we have shown that the general scheme for determining weights of linear mixtures developed in section 3, when particularized for ODEs, improves uniform aggregation significantly, even when the models make incorrect modeling assumptions.

8 Conclusions

We have argued that under incorrect modeling assumptions BMA can be worse than uniform aggregation. We have provided two maximum a posteriori algorithms to improve over uniform aggregation even in the case that the classifiers make incorrect modeling assumptions. We have shown by means of a nontrivial example that the algorithms can

be applied with significant accuracy gains. A more detailed study of these algorithms and a comparison with other general ensemble learning methods will be the subject of future work.

References

1. G. Bouchard and B. Triggs. The tradeoff between generative and discriminative classifiers. In *IASC International Symposium on Computational Statistics (COMPSTAT)*, pages 721–728, Prague, August 2004.
2. J. Cerquides and R. López de Màntaras. Tan classifiers based on decomposable distributions. *Machine Learning- Special Issue on Graphical Models for Classification*, 59(3):323–354, 2005.
3. B. Clarke. Comparing bayes model averaging and stacking when model approximation error cannot be ignored. *Journal of Machine Learning Research*, 4:683–712, 2003.
4. D. Dash and G. F. Cooper. Model averaging for prediction with discrete bayesian networks. *Journal of Machine Learning Research*, 5:1177–1203, 2004.
5. R. Dawes. The robust beauty of improper linear models. *American Psychologist*, 34:571–582, 1979.
6. T. G. Dietterich. Ensemble methods in machine learning. In *MCS '00: Proceedings of the First International Workshop on Multiple Classifier Systems*, pages 1–15. Springer-Verlag, 2000.
7. P. Domingos. Bayesian averaging of classifiers and the overfitting problem. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 223–230, 2000.
8. T. Fawcett. Roc graphs: Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4, HP Laboratories Palo Alto, 2003.
9. J. Friedman. Importance sampling: An alternative view of ensemble learning. Workshop on Data Mining Methodology and Applications, October 2004.
10. N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.
11. C. Genest and K. McConway. Allocating the weights in the linear opinion pool. *Journal of Forecasting*, 9:53–73, 1990.
12. C. Genest and J. Zidek. Combining probability distributions: A critique and an annotated bibliography. *Statistical Science*, 1(1):114–148, 1986.
13. Z. Ghahramani and H.-C. Kim. Bayesian classifier combination. Gatsby Technical report, 2003.
14. P. Gill, W. Murray, M. Saunders, and M. Wright. Constrained nonlinear programming. In G. Nemhauser, A. Rinnooy Kan, and M. Todd, editors, *Optimization*, Handbooks in Operations Research and Management Science. North-Holland, 1989.
15. R. Greiner, X. Su, B. Shen, and W. Zhou. Structural extension to logistic regression: Discriminant parameter learning of belief net classifiers. *Machine Learning - Special Issue on Graphical Models for Classification*, 59(3):297–322, 2005.
16. D. Grossman and P. Domingos. Learning bayesian network classifiers by maximizing conditional likelihood. In C. E. Brodley, editor, *ICML*. ACM, 2004.
17. P. Gruenwald, P. Kontkanen, P. Myllymäki, T. Roos, H. Tirri, and H. Wettig. Supervised posterior distributions. presented at the Seventh Valencia International Meeting on Bayesian Statistics, Tenerife, Spain, 2002.
18. D. Hand and R. Till. A simple generalization of the area under the roc curve to multiple class classification problems. *Machine Learning*, 45(2):171–186, 2001.

19. J. Hoeting, D. Madigan, A. Raftery, and C. Volinsky. Bayesian model averaging: A tutorial (with discussion). *Statistical science*, 14:382–401, 1999.
20. J. Hoeting, D. Madigan, A. Raftery, and C. Volinsky. Bayesian model averaging: A tutorial (with discussion) - correction. *Statistical science*, 15:193–195, 1999.
21. J. Ide and F. Cozman. Generation of random bayesian networks with constraints on induced width, with applications to the average analysis od d-connectivity, quasi-random sampling, and loopy propagation. Technical report, University of Sao Paulo, June 2003.
22. E. Keogh and M. Pazzani. Learning augmented bayesian classifiers: A comparison of distribution-based and classification-based approaches. In *Uncertainty 99: The Seventh International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, 1999.
23. G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, 1997.
24. G. J. McLachlan and K. E. Basford. *Mixture Models*. Marcel Dekker, 1988.
25. M. Meila and M. I. Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48, 2000.
26. M. Meila-Predoviciu. *Learning with mixtures of trees*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, 1999.
27. T. Minka. Bayesian model averaging is not model combination. MIT Media Lab note, December 2002.
28. A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 841–848, Cambridge, MA, 2002. MIT Press.
29. P. Pedregal. *Introduction to Optimization*. Number 46 in Texts in Applied Mathematics. Springer, 2004.
30. R. Raina, Y. Shen, A. Y. Ng, and A. McCallum. Classification with hybrid generative/discriminative models. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.
31. T. Roos, H. Wettig, P. Grünwald, P. Myllymäki, and H. Tirri. On discriminative bayesian network classifiers and logistic regression. *Machine Learning - Special Issue on Graphical Models for Classification*, 59(3):267–296, 2005.
32. M. Sahami. Learning limited dependence Bayesian classifiers. In *Second International Conference on Knowledge Discovery in Databases*, pages 335–338, 1996.
33. B. Thiesson, C. Meek, D. Chickering, and D. Heckerman. Learning mixtures of bayesian networks, 1997.
34. B. Thiesson, C. Meek, D. Chickering, and D. Heckerman. Learning mixtures of dag models. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 504–513, 1998.
35. K. Ting and I. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, 1999.
36. G. I. Webb, J. Boughton, and Z. Wang. Not so naive bayes: Aggregating one-dependence estimators. *Machine Learning*, 58(1):5–24, 2005.
37. I. H. Witten and E. Frank. *Data Mining: practical machine learning tools and techniques with java implementations*. Morgan Kaufmann, 2000.
38. Z. Zheng and G. I. Webb. Lazy learning of bayesian rules. *Machine Learning*, 41(1):53–84, 2000.