A new algorithm for MCMC Bayesian CART sampling

Gilles Blanchard*

March 18, 2002

Abstract

Recent work ([11], [12]) has shown the interest of using Metropolis procedures in a Bayesian framework to search for good classification trees. For a particular class of prior distributions on the trees, we introduce a new algorithm for MCMC sampling of trees, similar to a Gibbs sampler, based on Tjalkens et al. "tree weighting" algorithm [14], which results in a dramatic increase of the number of models actually taken into account. The sampled tree models are then averaged to produce an aggregate estimator or classifier. We present results of simulations on three benchmark datasets, that show the practical interest of this procedure.

Keywords: CART model, multiple classifiers, Bayesian MCMC sampling, tree weighting procedure, mixture models.

1 Introduction

The use of classification and regression tree models (CART, [8]) enjoys a wide popularity for a broad range of classification problems involving a complex structure of features. Recently a lot of attention has been focused on using multiple tree classifiers: a "forest" of classification trees is built using some protocol, and the different outputs of the trees are usually just averaged (possibly using some weighted averaging). Use of such "aggregated classifiers" has been proven to be largely more efficient than a single tree, epecially in the context of classification.

*Département de mathématiques, université Paris 11 Orsay. E-mail: Gilles.Blanchard@math.u-psud.fr

Various protocols to build multiple trees include randomization procedures when building a tree, such as proposed by Amit and Geman in the context of handwritten digit recognition [2, 3], bootstrap resampling of the training set (Breiman's "bagging", [5]), and "boosting" (Freund and Shapire, [13]). One other approach is to consider tree sampling by the means of a Markov Chain Monte Carlo procedure, in a Bayesian context. More precisely, one can define a prior probability on tree structures and use a MCMC procedure to simulate the posterior distribution on trees. This idea has already been proposed and tested by several authors, see Chipman et al. [11], Denison et al. [12], Blanchard [4].

The two main points of this article are the following: first, we introduce a new algorithm for MCMC sampling of trees, similar to a Gibbs sampler, that allows a more efficient simulation (as compared to previous existing algorithms) of the posterior over the space of trees. This algorithm is valid for a particular, yet quite broad, class of prior distributions, and makes use of the "tree weighting algorithm" introduced by Tjalkens et al. [14].

As a second point, our goal is to truly average the tree outputs over the posterior. Most authors using MCMC chains of trees so far were mainly aimed at finding one single "good" tree by maximizing the posterior. Two distinct reasons led us to be eager to perform an average over the sampled trees: on the one hand, theoretical results on the performance of the related, Bayesian-flavoured "progressive mixture" estimator [4]; and on the other, to be able to present this algorithm as a legitimate of its own procedure for building "forests" of CARTs, as presented above. Our experiments show that actually keeping only a limited number of trees (of the order of 30 in the examples studied) among a long run of the MCMC is sufficient to achieve good classification rates.

On the other hand, it is still possible to use the algorithm to search the tree space for a MAP estimator, like that was the case in [11], [12], so that we can actually compare the performances of our algorithm to existing ones on a common ground.

We further introduced in our experiments an "inverse temperature" parameter that results in a modified posterior. This parameter, which we show has a dramatic influence on the results of the aggregated classifier, only makes sense in the context of averaging several trees. In particular, we show that it is closely related to the tradeoff between the average individual performance of one tree and the average conditional dependance of pairs of trees, which is put forward as a key quantity by Amit et al. [1] in the context of multiple randomized classifiers.

2 The Bayesian CART estimator

2.1 General framework

We will be focusing throughout this paper on a general classification problem. We consider a couple of random variables (X, Y) where X is the "observation" taking his values in some measurable space $\mathcal{X}$ and Y the "class" of X taking its values in a finite set of classes $\mathcal{Y} = \{1, \dots, c\}$. The general method for classification will be to compute an estimator for the conditional probability $P(Y|X)$: the associated classifier is then $C(X) = \mathrm{Argmax_Y P(Y|X)}$. This means our results can easily be extended to a more general regression framework as well. Also, we can design other rules for inferring the classifier from the estimator, for instance if the misclassification loss function actually depends on the class.

The abstract framework and notations will be the following. We assume that we have a set of training examples $Z_1^N = (X_1, Y_1), \dots, (X_N, Y_N)$ which we divide into two sets, the "learning set" $\Lambda = Z_1^L$ and the "estimation set" $E = Z_{L+1}^N$. The choice of the parameter L will be studied later. To perform the estimation, suppose that we first have to choose between a countable family of available models $\mathcal{M}$. With every choice of a model $m \in \mathcal{M}$, comes an associated estimator $Q_m^E(Y|X)$, that is, a conditional probability of Y given X, that depends on the estimation set E. We will detail the construction of the models and estimators in the next section, but basically one should think of m as a fixed CART structure (that is, a fixed tree structure with fixed questions attached to its internal nodes), and of the estimator $Q_m^E(Y|X)$ as an estimation of the distribution of Y given the leaf reached by X, estimated, for each leaf, from the empirical distribution of examples of the estimation set that reached that leaf.

The main problem regards of course how we should choose a "good" model m, that is, whose risk, measured through a certain loss function, is as low as possible. Instead of actually picking one model in $\mathcal{M}$, we work in the following Bayesian, or mixture, setting: let π be a prior on the set $\mathcal{M}$. We define the *mixture* (conditional) probability on the set $\mathcal{M} \times \mathcal{Y}^L \times \mathcal{Y}$ by

$$Q_{\text{mix.}}^E(m, Y_1^L, Y|X_1^L, X) \triangleq \pi(m) \prod_{i=1}^{L} Q_m^E(Y_i|X_i) Q_m^E(Y|X). \tag{1}$$

The term "mixture" refers to the following marginal distribution

$$Q_{\text{mix.}}^E(Y_1^L, Y|X_1^L, X) = \sum_{m \in \mathcal{M}} \pi(m) \prod_{i=1}^{L} Q_m^E(Y_i|X_i) Q_m^E(Y|X).$$

When the estimation set E is kept fixed, each model m defines a (conditional) product probability $\prod_{i=1}^{L} Q_m^E(Y_i|X_i) Q_m^E(Y|X)$ on $\Lambda \cup \{(X,Y)\}$ (here the additional element (X,Y) should be thought as a unknown test observation drawn independently), so that the marginal distribution defined in the above equation appears as a weighted mixture of product probabilities on the learning set.

We finally define the (predictive) Bayesian estimator Q_π

$$\begin{aligned} Q_\pi(Y|X, Z_1^N) &\triangleq Q_{\text{mix.}}^E(Y|X, Z_1^L) \\ &= \sum_{m \in \mathcal{M}} \frac{\pi(m) \prod_{i=1}^{L} Q_m^E(Y_i|X_i)}{\sum_{m' \in \mathcal{M}} \pi(m') \prod_{j=1}^{L} Q_{m'}(Y_j|X_j)} Q_m^E(Y|X). \end{aligned} \tag{2}$$

Defined this way, Q_π is an estimator of the conditional probability of the class Y given observation X, using the knowledge of the sample $(X_1, Y_1), \ldots, (X_N, Y_N)$.

2.2 Relation to previous work

In [4], following ideas first introduced by Catoni [10] we have presented theoretical bounds for the Kullback-Leibler loss of the "progressive mixture" estimator, a modification of the plain Bayesian estimator defined by the following Cesàro mean:

$$
\begin{aligned}
Q_{\text{prog.}}(Y|X, Z_1^N) &\triangleq \frac{1}{L} \sum_{K=1}^{L} \sum_{m \in \mathcal{M}} \frac{\pi(m) \prod_{i=1}^{K} Q_m^E(Y_i|X_i)}{\sum_{m' \in \mathcal{M}} \pi(m') \prod_{j=1}^{K} Q_{m'}^E(Y_j|X_j)} Q_m^E(Y|X) \qquad (3) \\
&= \frac{1}{L} \sum_{K=1}^{L} Q_{\text{mix.}}^E(Y|X, Z_1^K).
\end{aligned}
$$

Let P_{N+1} be the joint probability of the variables $((Z_i)_{1 \le i \le N}, (X, Y))$ (that is, the training set plus an independent test observation), and let us denote by $R(Q)$ the Kullback-Leibler risk of an estimator $Q(Y|X, Z_1^N)$ with respect to P_{N+1}:

$$R(Q) = E_{P_{N+1}} D(P_{N+1}(Y|X, Z_1^N), Q(Y|X, Z_1^N)),$$

where D denotes the Kullback-Leibler divergence, i.e.

$$D(P, Q) = \begin{cases} E_P \log dP/dQ & \text{if } P \ll Q \\ +\infty & \text{otherwise.} \end{cases}$$

Then the following bound holds:

Theorem 1 *[4] If the joint probability P_{N+1} of the variables $((Z_i)_{1 \le i \le N}, (X, Y))$ is exchangeable,*

$$R(Q_{\text{prog.}}) \le \inf_{m \in \mathcal{M}} \left\{ R(Q_m) + \frac{1}{L} \log \frac{1}{\pi(m)} \right\}. \qquad (4)$$

This bound tells us that the K-L risk of the progressive mixture estimator $Q_{\text{prog.}}$ is not bigger than the risk of any model m plus a penalty term of order $\mathcal{O}(1/N)$, depending on m (when L is chosen to be of the form $L = \lfloor kN \rfloor$). Furthermore, while the result quoted here applies in principle only to a countable family $\mathcal{M}$, it can be extended to more general settings, for example classification trees where the set of questions is not necessarily countable, but is of finite Vapnik-Chervonenkis dimension. We refer the reader to [4] for more detail.

We have decided to restrict our attention throughout this paper to the experimental study of the Bayesian estimator Q_π (which is also the last term of the above Cesàro sum). Although bound (4) does not apply directly to this estimator, we hope that their behaviour are close enough in practice ; furthermore the estimator Q_π relates more directly to classical Bayesian alnalyses.

2.3 The CART model

We now define in more detail the model space $\mathcal{M}$ and estimators $Q_m^E(Y|X)$ as CART estimators [8]. We assume that we have a certain "library" $\mathcal{Q}$ of available questions on the observation X. A question is a measurable function $f : \mathcal{X} \to \{0,1\}$. For instance, a very typical setup is when $\mathcal{X} = \mathbb{R}^k$ and the questions are a set of separating hyperplanes $f_{v,b} : x \mapsto \mathbf{1}\{(x|v) \geq b\}$, where $\mathbf{1}$ denotes the indicator function of a set.

A classification tree is a binary tree whose nodes are labeled with questions $f \in \mathcal{Q}$. Given some observation X, we start from the root and choose at each node the left or right branch depending on the answer given to the question attached to the node. We continue until we reach a leaf; each leaf contains a certain distribution on Y that will be our estimator $Q_m^E(Y|X)$. The distributions contained in the leaves are estimated using the estimation set E.

Formally, a CART model m is defined by a complete binary tree whose internal nodes are labeled by elements of $\mathcal{Q}$ (by *complete* tree we mean that each node either is a leaf or has two descendants, there are no nodes having a single descendant). So $m = (T, K)$, where T is a binary tree structure, and K is the set of questions assigned to the internal nodes of T.

We denote by $m(X)$ the terminal node of T reached by letting the observation X follow the branches of the tree, as described earlier. Putting, for s a terminal node of

T,

$$n_s(i) = \sum_{(X,Y)\in E} \mathbf{1}\left\{m(X) = s, Y = i\right\} \ , \ i = 1, \ldots, c \ ,$$

we define the estimator Q_m by a Laplace estimator in each leaf of the tree, given by

$$Q_m^E(y|m(x) = s) = \frac{n_s(y) + 1}{\sum_i n_s(i) + c + 1}.$$

The main reason for which we chose a Laplace estimator at each terminal node, and not the simpler frequentist estimator, is that in the final Bayesian estimator (2), each model is weighted according to its posterior distribution on the learning set Λ and we don't want to allow this posterior to vanish. Also, simple bounds can be derived for the Kullback-Leibler risk of the Laplace estimator that do not hold for the frequentist estimator, and make sense in the setting recalled in section 2.2.

In principle, in the setting we have presented the set of models $\mathcal{M}$ should be countable, which implies that set of questions $\mathcal{Q}$ has to be countable in the first place. However, it can be shown that a bound similar to (4) holds in a more general setting when $\mathcal{Q}$ is of finite Vapnik-Cervonenkis dimension. Without getting into detail, the general principle is that we can regroup questions into sets of statistically equivalent questions with respect to the sample, that is to say, questions that divide the whole sample Z_1^N in the same way. For fixed training sample, we thus get a finite number of sets of regrouped questions and therefore a countable set of equivalent models on which we can define a prior in a suitable way.

In practice, the space of features is $\mathbb{R}^k$ and the questions are hyperplanes orthogonal to one of the coordinate axes, that is, of the type $\mathbf{1}\{x^{(i)} > t\}, t \in \mathbb{R}, i \in \{1, \ldots, k\}$, where $x^{(i)}$ is the i-th coordinate of x. We use the quite natural procedure of restricting the possible thresholds along coordinate i to midpoints of two consecutive elements of the reordered sample according to this coordinate.

3 Approximate computation through a MCMC procedure

3.1 Notations and hypotheses

The exact computation of the estimator (2) appears untractable because of the high number of models m involved. As an alternative method, common in Bayesian analysis, we will make use of a MCMC procedure, simulating the posterior on the set $\mathcal{M}$, to provide an approximate computation for the estimator. The idea of a MCMC simulation on classification trees in a Bayesian framework has already be suggested and studied by several authors [11], [12], and in the previous paper [4].

With the definition of a model m as a couple (T, K), a natural way to define a MCMC setup is to allow a move from m to m' by modifying either T or K. As a matter of fact, MCMC procedures proposed so far in the litterature for this particular setting seem to always involve stochastic movement on the tree structure itself: at each step of the MCMC algorithm, a pair of terminal nodes can be added or removed from the current tree. Our contribution to this point is to suggest a very different algorithm for which the structure of the tree is not altered, only the questions at the interior nodes of the maximal tree of depth d are changed at each step. The partial sum over all subtrees of this complete tree can then be computed in an exact and computationnaly efficient way at each step of the Monte-Carlo chain. This algorithm has proved to be more efficient and quick in practice than the previous ones.

At this point we will need to make a minor change in our notations. Let us first choose a maximal depth d for all the trees. As announced above, the principle of the algorithm is to decouple the averaging over subtree structures with fixed internal questions (performed analytically) and the averaging over questions (performed approximately with a MCMC). To handle this decoupling idea formally, from now on, in the notation $m = (T, K)$ we will assume that K actually stands for a set of questions attached to the internal nodes of the *maximal* complete binary tree $\overline{T}_d$ of depth d, even though some of them are not used in the actual tree structure T. By maximal tree we mean the tree whose leaves are all at depth d. This just amounts to splitting the original models into several equivalent models (namely, all models that have the same questions as the original one on the actual internal nodes of the structure T). An alternative way to look at this is to say that we first build a complete tree of maximal depth $\overline{T}_d$, with internal questions K, and then consider all possible prunings T of this maximal tree.

The following set of assumptions, denoted (H), will be necessary and assumed to hold in the sequel:

- The trees have a maximum depth d; in other words, the trees deeper than d have a prior probability zero.

- We assume that the prior π can be written as a product

$$\pi(m) = \pi(T, K) = \pi_1(T)\pi_2(K). \tag{5}$$

  This means that the prior on the structure is independent of the choice of internal questions.

- The prior π_1 on the tree structure T is exactly the distribution of the genealogy tree of a binary branching process of given parameter ρ, stopped at depth d (each individual gives birth to two descendants with probability ρ, or dies without children with probability $1-\rho$, except for individuals of the dth generation who never have children). We will show later that this hypothesis can be extended to a more general prior, namely a branching process where the parameter ρ can actually depend on the location of the node in the tree.

3.2 A Gibbs sampler for trees

The key point of the algorithm is that, under (H), when the set of questions K is fixed, a sum of the form

$$S(K, Z_1^P) \triangleq \sum_{T \in \mathcal{T}_d} \pi_1(T) \prod_{i=1}^{P} Q^E_{(T,K)}(Y_i|X_i), \tag{6}$$

(where $\mathcal{T}_d$ denotes the set of all complete binary tree structures of depth less than d, or equivalently the set of all subtrees of $\overline{T}_d$), can be computed analytically and in a fast way, using the recursive "tree-weighting" algorithm designed by Tjalkens et al. [14] in a data compression framework. We describe the algorithm in a very general setting in Appendix A. The computation requires one single sweep through all the nodes of the maximal tree $\overline{T}$.

To take advantage of this algorithm, we develop the sum (2) defining Q_π into two stages, namely a sum over all the possible values of the set of questions K, and a sum over the tree structure $T \in \mathcal{T}_d$ when K is fixed. The principle is that the tree weighting algorithm allows us to compute efficiently this second stage, while a Gibbs sampler algorithm only acting on the values of K is used to perform the first stage.

More formally, let $\mathcal{Q}_d$ denote the set of all possible values of K, that is, all possible assignments of questions to the internal nodes of $\overline{T}_d$, the maximal complete binary tree of depth d.

We develop the definition (2) of Q_π into a sum over K only, getting

$$\begin{aligned} Q_\pi(Y|X, Z_1^N) &= Q^E_{\text{mix.}}(Y|X, Z_1^L) \\ &= \sum_{K \in \mathcal{Q}_d} Q^E_{\text{mix.}}(K|Z_1^L) Q^E_{\text{mix.}}(Y|X, Z_1^L, K). \end{aligned} \tag{7}$$

Here $Q^E_{\text{mix.}}(K|Z_1^L)$ is exactly the Bayesian posterior distribution on K given the sample. On the other hand, the quantities $Q^E_{\text{mix.}}(Y|X, Z_1^L, K)$ can actually be seen as a family of estimators indexed by K; K can therefore be seen as a super-model index on which the MCMC transitions will operate. Once K is fixed, we will call informally

$Q^E_{\text{mix.}}(Y|X, Z_1^L, K)$ an "s-tree" estimator defined by K. An s-tree estimator is obtained by averaging according to the posterior over all the possible subtree structures of the maximal tree $\overline{T}_d$ when the questions K are fixed; it can be written as

$$
\begin{aligned}
Q^E_{\text{mix.}}(Y|X, Z_1^L, K) &= \frac{Q^E_{\text{mix.}}(K, Y, Y_1^L | X, X_1^L)}{Q^E_{\text{mix.}}(K, Y_1^L | X_1^L)} \\
&= \frac{\sum_T Q^E_{\text{mix.}}(\,(T, K), Y, Y_1^L, Y | X, X_1^L)}{\sum_T Q^E_{\text{mix.}}(\,(T, K), Y_1^L | X_1^L)} \qquad (8) \\
&= \frac{S(K, (Z, Z_1^L))}{S(K, Z_1^L)}, \qquad (9)
\end{aligned}
$$

where we have used definition (1) of Q^E_m. This means that an s-tree estimator for an unknown test example (X, Y) can be analytically computed using only two sweeps through the the nodes of $\overline{T}$ (one for each $S(K, \cdot)$ above).

Now, the posterior on the s-tree index K can in turn be written as

$$
\begin{aligned}
Q^E_{\text{mix.}}(K|Z_1^L) &= \frac{Q^E_{\text{mix.}}(K, Y_1^L | X_1^L)}{\sum_{K'} Q^E_{\text{mix.}}(K', Y_1^L | X_1^L)} \\
&= \frac{\sum_T Q^E_{\text{mix.}}(\,(T, K), Y_1^P | X_1^P)}{\sum_{K'} Q^E_{\text{mix.}}(K', Y_1^L | X_1^L)} \\
&= \frac{\pi_2(K) S(K, Z_1^L)}{\sum_{K'} Q^E_{\text{mix.}}(K', Y_1^L | X_1^L)} \qquad (10)
\end{aligned}
$$

To sum things up, for every K we are able to compute the second factor of (7) through the tree weighting algorithm, using eq. (9). Therefore, we only need to simulate the marginal of the posterior of K on $\mathcal{Q}_d$, given by equation (10). To build a MCMC procedure, what we actually need to compute is the ratio of the posterior of $K, K' \in \mathcal{Q}_d$, given by

$$
\frac{Q^E_{\text{mix.}}(K|Z_1^L)}{Q^E_{\text{mix.}}(K'|Z_1^L)} = \frac{\pi_2(K) S(K, Z_1^L)}{\pi_2(K') S(K', Z_1^L)}, \qquad (11)
$$

so that this ratio can in turn be computed efficiently. We propose to consider the MCMC transitions obtained by changing one single question at a time at one internal node of $\overline{T}$, which we can see as a kind of Gibbs sampler.

We describe the algorithm obtained on fig. 1, in the classical case where $X \in \mathbb{R}k$ and the questions are of the form $\mathbf{1}\{x^{(p)} > t\}$.

1. Initialize randomly the questions at the internal nodes of the maximal tree of depth d.

   Initialize at each node counters for the number of examples in each category that reached this node, separately for the learning and estimation sets. Compute the corresponding Laplace estimators in each node using the counters for the estimation set.

2. Choose a depth ℓ as a uniform random number in $\{1,\dots,d-1\}$, then choose an internal node s at random, uniformly among all possible nodes at depth ℓ.

3. Draw a new question $f \in \mathcal{Q}$ for this node, uniformly among available questions: draw a coordinate p at random in $\{1,\dots,k\}$ and a threshold t at random among the midpoints of two consecutive elements of the reordered set of p-th coordinates of the training sample. The new question is then $\mathbf{1}\{x^{(p)} > t\}$.

   Denote by K' the new set of questions (i.e. question attached to node s beeing replaced by f). Update the counters and the estimators over the subtree rooted in s.

4. Accept the new set of questions K' with probability

$$1 \wedge \frac{S(K', Z_1^L)}{S(K, Z_1^L)}, \tag{12}$$

   (computed using the Laplace estimators and the learning counters), or keep the old set of questions K with the complementary probability.

5. Record the new s-tree obtained, and go to step 2.

Figure 1: The s-tree sampling procedure

Note that alternatively, for step 2, one could sweep successively through all the internal nodes according to a fixed deterministic order, as in a classical Gibbs sampler. However, with this method half of the computation time will be used to change questions at the deepest internal nodes, which does not appear as a good strategy since the most relevant nodes are the top ones. For the same reason, a uniform random choice between the internal nodes is probably not optimal. The solution we adopted here is to draw a depth p uniformly between 1 and $d-1$, and then to draw a node uniformly among nodes at depth p. This ensures that the questions at the top nodes will have a reasonable variability, which certainly is a crucial point if we are to expect the algorithm to have good mixing properties.

Finally, once the sample of s-trees has been recorded, it is possible to compute the approximate value of the Bayesian estimator on a new example (X, Y) by averaging the values of the s-tree estimators over the sample obtained, using eq. (9). Alternatively, to speed things up, only a small proportion of the sample may be used (this will be discussed below).

The complexity of the algorithm can be studied as follows: at each step of the algorithm we have to compute the sum $S(K', Z_1^L)$ which is done in less than $\mathcal{O}(2^d)$ operations (more precisely, if we choose to change node s which is at depth h, we have to update the entire subtree rooted at s and then update the nodes on the branch from the root to s, resulting in $\mathcal{O}(2^{d-h}+h)$ operations. This a straightforward consequence of the form of the tree weighting algorithm, see appendix A); for the computation of the estimator on a unknown example (X, Y), we then may also possibly have to compute $S(K', (Z_1^L, (X, Y)))$ in the case the transition is accepted, but this one is done in $\mathcal{O}(d)$ operations since we only have to update the counters along the branch followed by observation X.

3.3 Variants and extensions

3.3.1 More general priors on the tree structure

As announced previously, the prior on the tree structure T can be made more flexible if the following way: the parameter ρ of the Galton-Watson process (i.e. the probability for a node to have two descendants) can be a function depending on the node itself, this appears clearly when looking at the recursive algorithm in Appendix A.

This way, we have some control over the prior; for example, we can tune the parameters ρ so that trees of a given size have a higher prior, lower the prior of the shallow trees which we don't expect to be very good by taking ρ close to 1 for the root node and its descendants, etc. For example, the prior defined by the splitting criterion p_{SPLIT} proposed in [12] fits into this setting.

3.3.2 Other MCMC moves

Chipman et al. suggest in [11] that in addition to changing the question at a given node, allowing to swap the questions between one internal node and his parent yields significant improvement in terms of speed of convergence. Of course this move can be added to our version as well—it is not a Gibbs sampler any more in this case though.

3.3.3 Same estimation and learning sets

While the separation between an estimation and a learning set allowed us, in [4], to derive bounds on the Kullback-Leibler risk for the "progressive estimator" given by equation (3), we are naturally led, in practice, to try to use the same set for estimation and learning. This is, of course, going one more step away from the theory. Our experiments on standard databases show that this, heuristically, usually yields a noticeable improving of results.

3.3.4 "Temperature" parameter

Because we wanted to get more variability in the Monte Carlo chain used to simulate the posterior, we introduced a "inverse temperature" parameter β and replaced the probability of accepting a new question in equation (12) by

$$1 \wedge \left(\frac{S(K', Z_1^L)}{S(K, Z_1^L)} \right)^{\beta} . \tag{13}$$

Of course, doing this we are not simulating the true posterior anymore; rather, we will get an approximate computation for the modified estimator

$$Q_{\pi,\beta}(Y|X, Z_1^N) = \sum_{K \in \mathcal{Q}_d} \frac{\pi_2(K) S(K, Z_1^L)^{\beta}}{\sum_{K'} \pi_2(K') S(K', Z_1^L)^{\beta}} \left(\frac{S(K, (Z_1^L, (X, Y)))}{S(K, Z_1^L)} \right) . \tag{14}$$

(The introduction of this parameter and this modified version of the posterior has been suggested to us by the work of Catoni [9] on "Gibbs estimators").

It appeared experimentally that usually taking some suitable $\beta \in (0, 1)$ provided noticeably better classifiers than the original "plain" algorithm (equivalent to taking $\beta = 1$). This our opinion, which we will discuss later in the paper, than this improved performance is not (or at least not only), as one could think at first glance, due to a faster convergence of the Monte-Carlo chain (and thus better approximation) when $\beta < 1$ (i.e. "higher" temperature), but is an intrisic property of the estimator $Q_{\pi,\beta}$.

3.4 A remark about the MAP estimator

Although our focus in this paper is about Bayesian model averaging, an alternate classical Bayesian procedure (MAP) is to pick a single model for which the posterior probability is maximized. It is also common in this setting, when it is not possible to find directly this maximum, to perform an approximate computation based on a MCMC procedure (e.g. Metropolis algorithm or simulated annealing). In the case of classification trees, while multiple tree models are known to perform better in general, one single tree model may still be preferable in some cases, for example if one wants to have an easy to understand interpretation of how the estimator actually handles the data. In the papers [11, 12] for example, the emphasis was on finding one single tree.

However, the algorithm presented here can readily be used in this setting: once we have sampled the s-trees, the only problem remaining is then to be able to extract the best sub-tree model (in terms of the posterior) from a given s-tree, but it appears that we can perform that operation by a straightforward modification of the "tree weighting" algorithm, also presented in Appendix A. This Bayesian MAP subtree extracting can be seen as a kind of automatic pruning procedure.

4 Experimental results

We tested the algorithm on three "classical" datasets taken from the UCI database (available at `ftp.ics.uci.edu/pub/machine-learning-databases`), namely "Breast-cancer-wisconsin", "Pima-Indian-Diabetes", and "Yeast". A summary of these datasets is given in table 1. All the features are real-valued. For the Breast Cancer database, 16 examples with missing data (out of an original 699) where thrown away.

Data Set	Size	Nb. features	Nb. classes
Breast Cancer	683	9	2
Pima Diabetes	768	8	2
Yeast	1484	8	10

Table 1: The datasets used in the experiments.

4.1 Base protocol

The base protocol makes use of the extension mentioned above under the paragraph "continuous set of questions". Half of the examples of the sets are used for training and the others are kept as a test set. The default parameters used in the experiments are the following: 5000 iterations of "burn-in" and then 5000 recorded iterations; the training examples are split into estimation and learning sets of equal sizes; the branching parameter ρ is 0.4; the maximal tree depth is 8. The same 5000 recorded iterations are then used for estimation and classification of all examples of the test set. Each experiment is repeated 200 times (re-drawing randomly the training and test sets each time) and the error bars in the graphics show a 95% confidence interval of the estimated quantities for a gaussian approximation. We report classification errors which are our primary interest, but also the empirical redundance[1] of the classifier, since the classifier was built in [4] to minimize this loss criterion. One can see on the figures that these two criterions are closely related in behaviour except in some particular cases.

4.2 Number of iterations

The choice of the number of iterations for a Metropolis algorithm is always delicate. Due to the complex structure of the space considered, it is difficult to give *a priori* estimates of the number of steps required. The results in figure 2 show the evolution of

[1]For two discrete probability distributions P, Q the redundance of Q with respect to P is defined as $E_P[-\log(Q)]$. It is, up to the constant $E_P[-\log(P)]$, the same quantity as the K-L divergence. We estimate it by replacing P by the empirical distribution of the test set.

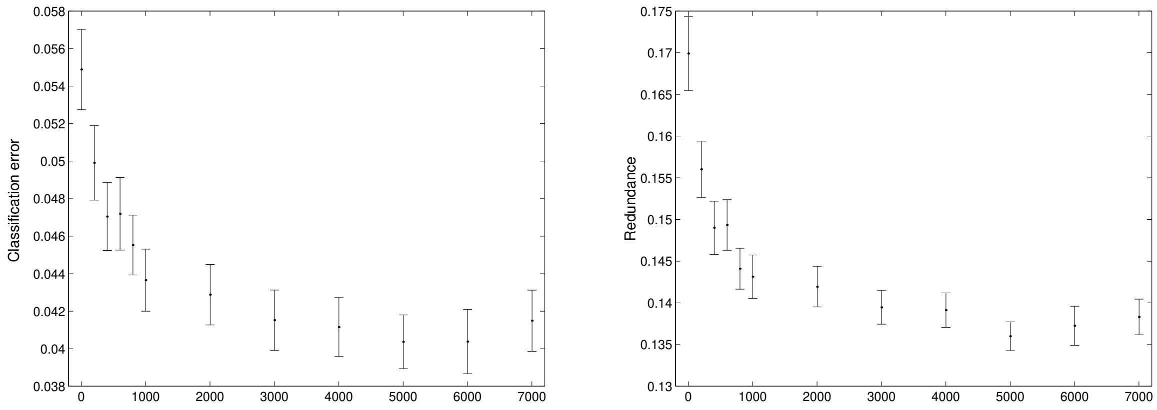

Figure 2: Classification error and redundance vs. number of iterations (after burn-in) obtained on the breast cancer dataset.

the error rates as the number of iterations is increased for two of the datasets. While the error rate looks stationnary after 5000 steps (a first 5000 steps are performed before as a "burn-in"), it is not necessarily an evidence that the MCMC has reached its stationnary distribution, because the space considered obviously has many local extrema. However, the mixing properties of this chain are certainly better than those of a chain where the tree structure itself is changed at each step.

4.3 Repartition between learning and estimation sets

The choice of the parameter L, that is, of the respective size of the estimation and learning sets, is a question arising naturally. Figure 3 shows the error rates drawn with respect to the ratio L/N, as well as the mean size of a tree and the average depth reached by a test example[2]. The conclusion is that in general the algorithm is quite robust with respect to the choice of L, although we can guess that the best value is for L/N probably a little over 0.5. It is interesting to note as well that in this case the best performance also corresponds to the larger average size of the trees.

While this results give us more insight onto the general behaviour of the algorithm, this issue is however of reduced practical interest, because when we choose to use the

[2]A description of how to compute the average size of the tree models and the average depth reached by an example on a given s-tree is given in Appendix C

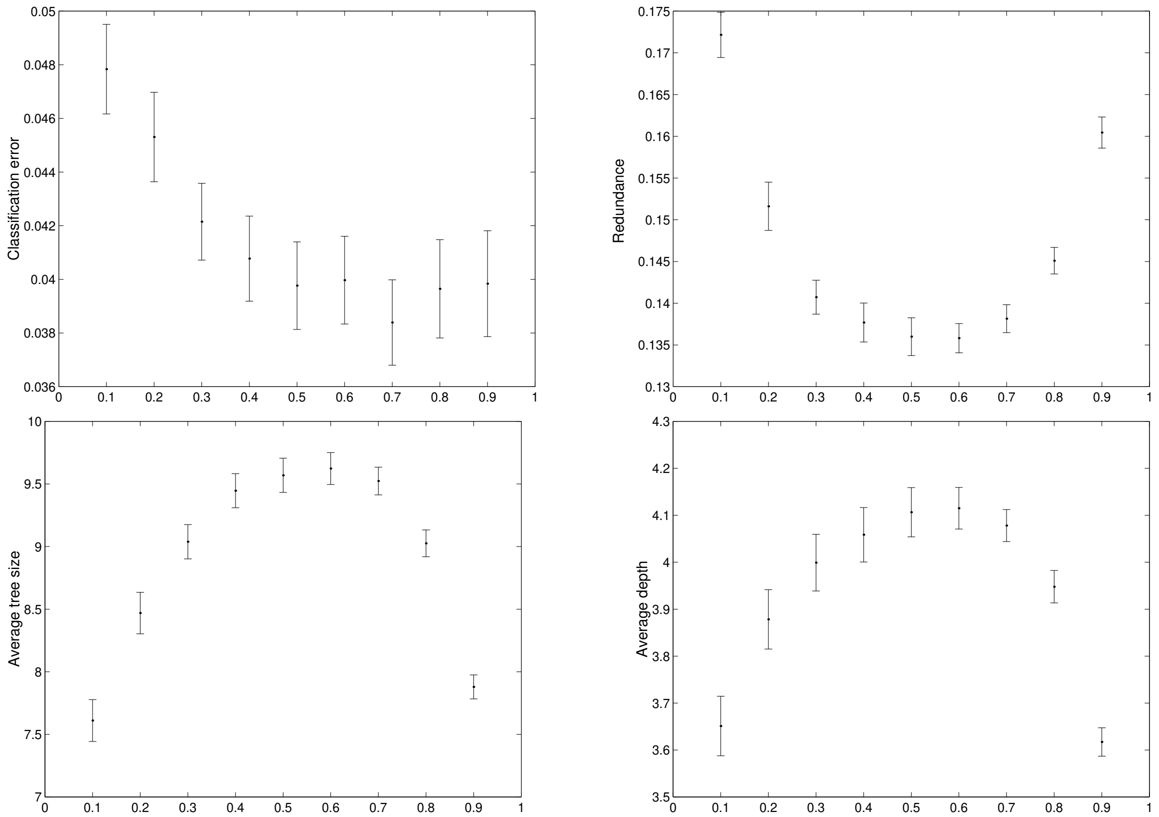

Figure 3: Influence of the repartition of training examples between learning and estimation on various characteristic quantities. X axis: fraction of the training sample used for estimation. (Breast Cancer dataset)

Data Set	2 set	1 set
Breast Cancer	3.8%	2.7%
Pima Diabetes	25.3%	24.3%
Yeast	44.3%	41.2%

Table 2: Dividing (equally) the training set into estimating and learning set versus using the same set (the full training set) for both (table shows best results wrt. choice of the inverse temperature β in each case)

full set Z_1^N both as estimation and learning set as suggested in the previous section, we get systematically better classifications rates (see table 2). This is the protocol we will use from now on.

4.4 Branching parameter and other priors

Another parameter we have to choose is ρ — the parameter of the binary branching process. As shown in figure 4, while as a rule of thumb $\rho \approx 0.4$ seems to be a good value in general, the algorithm appears to be quite robust with respect to the choice of ρ. In our opinion, this reduces the interest of an in depth discussion about the choice of the prior π_1. We also tried, in experiments not shown, the extended prior suggested in the previous section, namely, since we can actually let ρ depend on the node considered in the tree, it seems a reasonable try to choose ρ_s to be close to 1 for the nodes s close to the root in order to downweight the very shallow trees (which are given a high prior if ρ is fixed). Nevertheless, using this kind of prior we noticed a but very slight improvement of performance, of little statistical significance, which strenghtens the evidence that the algorithm is robust with respect to the choice of the prior. It is noticeable though that the parameter has an impact on the average size of the trees, which was to be expected (A larger ρ gives a relatively higher *a priori* probability to large trees). Although it seems profitable at first to rise the weight of larger trees, it seems that for $\rho > 0.4$ the classification error stays stationnary while ρ and the average tree size are still growing.

4.5 Reduction of the number of trees

An obvious drawback of the method proposed it that it is computationnaly intensive. In our experiments, we usually drew a single sample path of the chain using the training and estimation sets, recorded it and then used the average of all the distributions predicted by the s-trees along this path to classify all the examples of the test set. This usually means a few thousand s-trees. On the other hand, other authors dealing with some other algorithms to generate random tress (see e.g. [2]) usually achieve their best performance with only a few dozens of trees. In order to reduce the number of trees used, we tried the following procedure: sample k independant

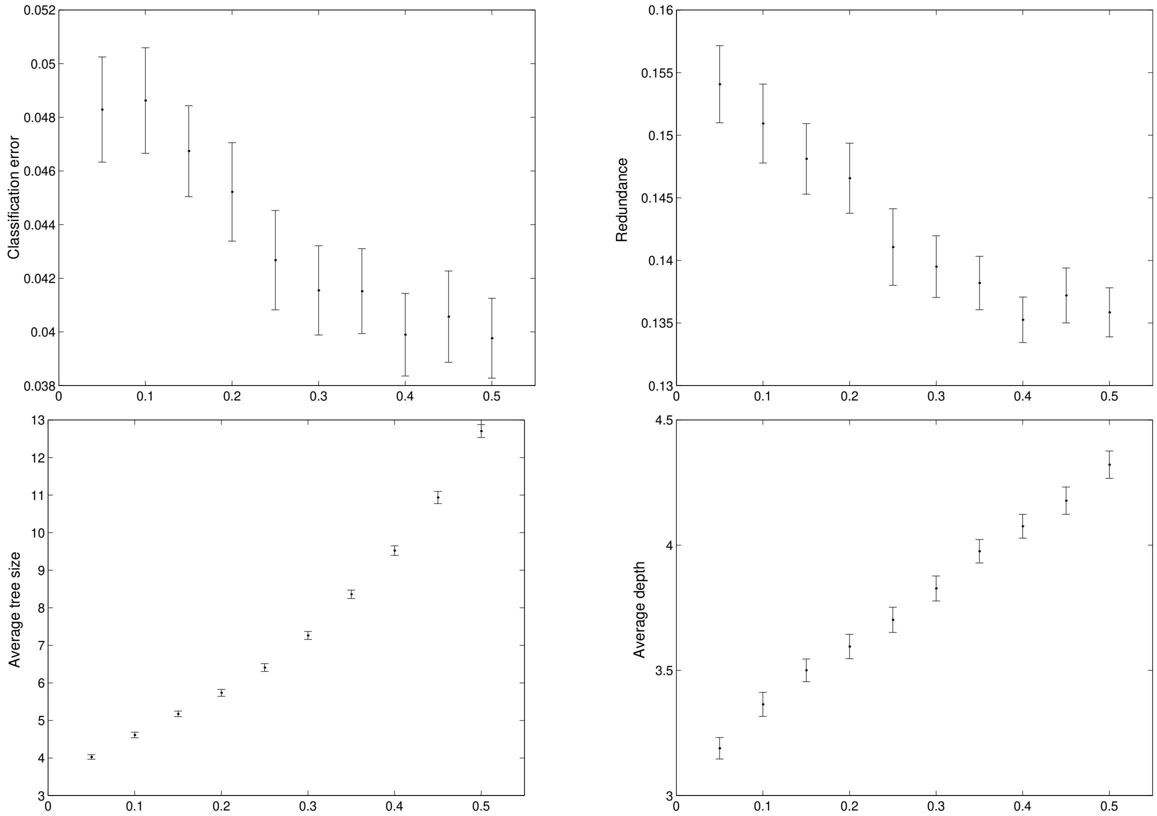

Figure 4: Influence of the branching parameter ρ. (Breast Cancer dataset)

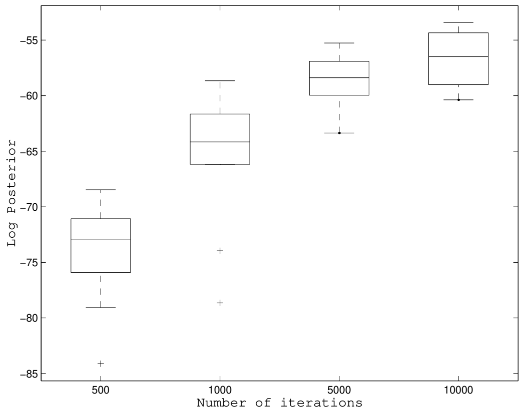

Figure 5: Boxplots for the maximum log posterior among the sampled trees versus number of iterations (Breast Cancer dataset, using the full dataset), 10 runs per boxplot.

paths using the Metropolis algorithm, every time keeping only the last tree. Then for classification of the test set use only the k trees thus built. On the databases used, we were able to obtain results as good as averaging over one single sample path with only 20 to 30 trees. Since we have to run 30 times the Metropolis algorithm, of course the part involving the tree building is now even more computationnaly intensive. But the important point is that once the classifier (the k trees) is built, we only have to deal with a reduced number of trees for effective classification (and this is the part where we are more interested to reduce the computation time in the first place). An alternative method is to keep only a fixed number of trees on a single sample path, for example one every thousand.

4.6 Maximum a posteriori

In relation to the possible use of the algorithm to find a good approximation to the MAP estimator, we investigated the behavior of the maximum a posteriori among the trees sampled by the algorithm, using the procedure described in section 3.4. This is in particular useful to evaluate if the algorithm is able to find regions of high posterior distribution. Figure 5 shows the results obtained for the Breast Cancer dataset (used in totality in those specific experiments). It is interesting to note that

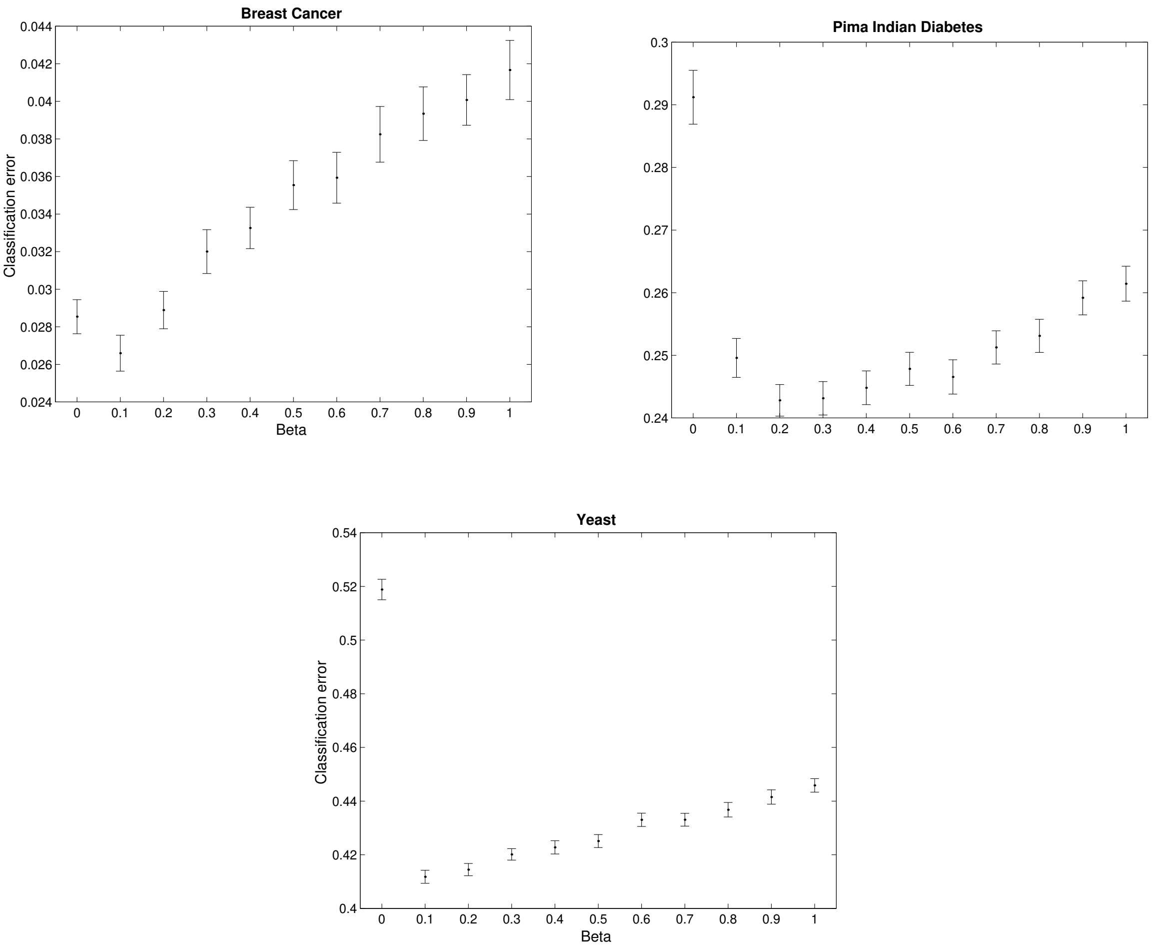

Figure 6: Influence of the inverse temperature β on classification error

after a few thousand iterations, all different runs of the algorithm have found regions with relatively high posterior, so that the values of the maximum log posterior found are quite concentrated. This is in contrast with previous existing algorithms where the MCMC moves acted on the tree structure itself, which can easily be trapped into regions of lower posterior for a long time (see for example the experiments shown in [11]).

4.7 Temperature parameter

The temperature parameter β introduced in the previous section was found to have a strong influence on the results. Figure 6 shows curves of classification rates vs. β. These appear to be coarsely convex for $\beta \in [0, 1]$, and usually have a minimum for some intermediate value of β. Surprisingly enough, for the Breast Cancer data set, the value $\beta = 0$ is better than $\beta = 1$ — while $\beta = 1$ corresponds to the standard Bayesian estimator studied so far, $\beta = 0$ is equivalent to drawing questions completely randomly at every node of the tree. (In practice, it means that the new question drawn at a node of the tree is always accepted at each step of the Metropolis chain.) Note, however, that at each step, once the questions are fixed, the weighted average over all the subtrees of the complete maximal tree is still done according to the "true" posterior, that is to say, the parameter β only influences the posterior distribution of the *questions* while the posterior on the *tree structure*, given the questions, remains the same. In other words, β has an influence on the distribution simulated on the space of s-trees, but not on the way an s-tree estimator is computed. We have tried to introduce this temperature parameter for the posterior on the tree structure as well, but it yielded a dramatic deterioration of the classification rate for $\beta < 1$, negating the beneficial effects on the questions's posterior.

How can we explain these results? Obviously, a lesser value for β has a "flattening" effect on the posterior — in some sense, it reduces artificially the size of a test sample from L to βL. So why do we get better results? The first thought that comes in mind is that the number of iterations of our Metropolis-Hastings chain is not sufficient for $\beta = 1$ and that it keeps beeing caught in some local minima. When $\beta < 1$, the energy landscape is flatter and the chain would reach its stationnary state sooner. This would compensate for the formal reduction of the sample size, acting much like a "simulated annealing" effect. To check for this hypothesis we did the following experiment: start by performing the modified algorithm with $\beta < 1$ chosen to be heuristically the "best" one (from the previous table of results). Then, switch to $\beta = 1$ and perform a few more thousand iterations. Keep the last tree obtained, and start this procedure over again in order to obtain 25 trees. Remember that in the previous paragraph we showed that drawing a limited number of trees from such independant Metropolis chains does as well as averaging over thousands of trees along one path of the chain.

If the "simulated annealing effect" explanation was right, the iterations made at $\beta = 1$ should not drive the chain too far away from the region it first reached with a lower β and thus we should get at least equivalent (or even better, since this procedure is closer to a true simulated annealing procedure) results than those obtained with a fixed $\beta < 1$. But his was not the case: by acting this way we got back the same performance results that with the "plain" $\beta = 1$ algorithm. This must show, as claimed earlier in the paper, that the better performance obtained with low β is not caused (or at least mainly not) by better convergence rates of the MCMC chain.

We suggest two tracks for explaining the better classification rate at "higher" temperature. First, as noted earlier, the idea of introducing such a temperature parameter was suggested by Catoni's "Gibbs estimators" [9]. In this paper Catoni presents an in-depth study of a very general form of such modified Bayesian estimators where the likelihood part of the posterior is modified by an similar inverse temperature exponent β. However, as pointed out earlier, a full Gibbs estimator in our case would involve to use the β exponent also when computing the partial sums over all subtrees when the questions are fixed, and when we tried to do so the classification rates deteriorated.

A more heuristic explanation relies on the work of Geman and Amit [3], and more recently Amit and Blanchard [1], and Breiman [7], where they suggest that "aggregated trees" (i.e. precisely a classifier obtained by an average of trees) perform better when the classifiers are *weakly dependant conditional on class.* On a very broad class of algorithms generating multiple classifiers, there is actually a tradeoff between this weak dependance and the individual performance of each tree.

We show in figure 7 the average covariances (conditional on class) betweeen pairs of s-trees when the parameter β varies, and notice that the dependance is indeed

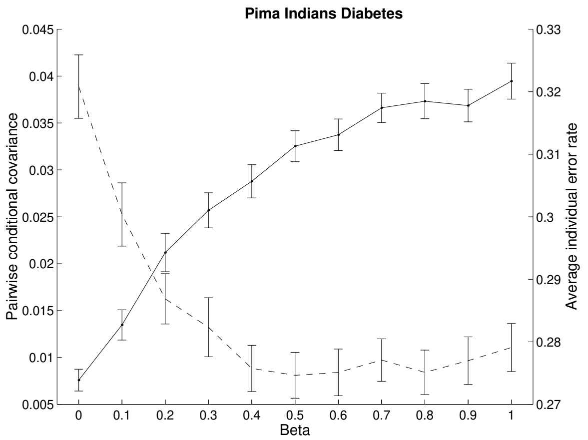

Figure 7: Solid line shows pairwise conditional covariance of s-trees, dashed line shows average error rate of one single s-tree versus β.

weaker when β gets smaller. From this point of view, it becomes more sensible to use some $\beta < 1$ for the posterior probability of the questions, but not when performing the sums over subtrees once questions are fixed: we want to have more freedom on the choice of questions since it will very probably decrease the dependance between trees. But if we fix the questions, then all the classifiers obtained as subtrees of the complete tree are likely to be very correlated whatsoever, so our better choice is probably to concentrate on the better ones, i.e. to have a more peaked (conditional) posterior, rather than a spread out posterior.

5 Discussion

Generation of classifier trees by MCMC procedures has received increased interest recently. One of the main problems encountered so far is that very often, when the chain movements concern the tree structure itself, the chain can get caught into local minima of the energy landscape (see for example [11]). The algorithm we propose has several advantages: we only have transitions on the questions asked at the internal nodes of the tree. Taking into account the different possible tree structures is made automatically and in an exact way. As a consequence we avoid many of the problems linked to the chain beeing caught locally, and obtain better mixing properties. Also the number of "trees" truly averaged is much greater in sheer number (remember that for a fixed set of questions on a complete tree of depth d, there are of the order of 2^{2^d} subtrees which are all taken into account). The classification rates obtained on the "breast-cancer" and "diabetes" datasets (see table 2) are similar or better than those of various protocols generating multiple tree classifiers to be found in e.g. [6]. (The third dataset "Yeast" does not appear in this last reference)

In terms of computing power needed, one step of the chain requires of course more computation than when single trees are considered. The structure averaging algorithm is very fast and efficient however. On the examples studied, we were able to run 10000 iterations of the chain in the order of one minute on a SPARC Ultra5. For practical classification problems, the critical point should be the computation time needed to "ask" a question on a training example. Namely, when we change a question at an internal node s of the tree we need to collect all training examples present at that node and let them follow the branches again until they reach a leaf. This means asking $d - d_s$ questions per training example present at node s, where d_s is the depth of node s. In the examples studied here, asking a question does not require any computation so this was not much of a problem; in other situations (for example the complicated queries taken into consideration by Amit and Geman [2] for handwritten digit recognition), this might be a serious bottleneck. Note however that this problem has always been present in previous Bayesian trees MCMC procedures in the litterature so far.

On the other hand, if true that constructing the chain is relatively computationaly intensive, classifying a new test example is actually very fast (less than 0.1 sec.) if we only finally keep 30 trees.

Finally, we want to point out once again the key role of the inverse temperature parameter β. In practice it works as a tuning parameter that allows the user to balance individual classifier performance versus pairwise conditional dependance. It controls the spread of the distribution on the space of s-trees in a straightforward way, as recommended by Amit et al. [1] for multiple randomized classifiers. In practice a good value of β can be determined by cross-validation, but a more theoretical explanation on why and how to choose the parameter is probably an interesting track for future research.

A The tree weighting algorithm

We give here a version of Tjalkens, Willems and Shtarkov's recursive "tree weighting" algorithm [14], extended for our purposes in a general abstract setting.

We want to perform an exact calculation of a sum of the following form:

$$S(f,g) = \sum_{T \in \mathcal{T}_d} \left(\prod_{s \in \delta T} f(s) \prod_{s \in T \setminus \delta T} g(s) \right), \tag{15}$$

where $\mathcal{T}_d$ is the set of all complete binary trees of depth less than d, δT is the set of terminal nodes (leaves) of a tree T, f is some real function defined on the nodes (both internal and terminal) of the maximal complete binary tree of depth d, denoted by $\overline{T}_d$, and g is another real function defined on $\overline{T}_d \setminus \delta\overline{T}_d$. A node of a tree is defined as a finite binary chain, element of $\{\emptyset\} \cup \{0,1\}^*$ ($\emptyset$ is the tree root; if s is a node, by definition its two descendants are $s0$ and $s1$). A tree T is defined as the collection of its nodes.

Let us define recursively the function σ on the nodes of $\overline{T}_d$ by:

$$\sigma(s) = \begin{cases} f(s) & \text{if } s \text{ is a terminal node of } \overline{T}_d \\ f(s) + g(s)\sigma(s0)\sigma(s1) & \text{otherwise.} \end{cases} \tag{16}$$

Then it can be shown by a straightforward recursion on the tree depth that $\sigma(\emptyset) = S(f,g)$, and so we can compute recursively $S(f,g)$ in a single sweep through all the nodes of $\overline{T}_d$, therefore $\mathcal{O}(2^d)$ operations.

Now we show that the quantity $S(K, Z_1^P)$ defined by

$$S(K, Z_1^P) \triangleq \sum_{T \in \mathcal{T}_d} \pi_1(T) \prod_{i=1}^{P} Q^E_{(T,K)}(Y_i|X_i), \tag{17}$$

is of the form $S(f,g)$ for a certain couple (f,g): on the one hand, the specific prior π_1 on the tree structure can be written as

$$\pi_1(T) = \rho^{|T \backslash \delta T|}(1-\rho)^{|\delta T \backslash \delta \overline{T}_d|}.$$

On the other hand, let us define the counters $n_s(i)$ as the number of examples of class i of the estimation set E whose path down the tree (recall that the set of questions K attached to the nodes is fixed) goes through node s, and $k_s(i)$ as the similar counters for the set Z_1^L (i.e. the learning set Λ), then we have

$$\prod_{i=1}^{L} Q^E_{(T,K)}(Y_i|X_i) = \prod_{s \in \delta T} \prod_{i=0}^{c} \left(\frac{n_s(i)+1}{\sum_j n_s(j) + c + 1} \right)^{k_s(i)}.$$

We finally can compute the quantity $S(K, Z_1^P)$ if we define the functions f, g as

$$f(s) = \begin{cases} (1-\rho) \prod_{i=0}^{c} \left(\frac{n_s(i)+1}{\sum_j n_s(j)+c+1} \right)^{k_s(i)} & \text{for } s \in \overline{T}_d \backslash \delta\overline{T}_d, \\ \prod_{i=0}^{c} \left(\frac{n_s(i)+1}{\sum_j n_s(j)+c+1} \right)^{k_s(i)} & \text{for } s \in \delta\overline{T}_d, \end{cases}$$

$$g(s) = \rho \text{ for all } s \in \overline{T}_d \backslash \delta\overline{T}_d.$$

B Finding the MAP subtree

A straightforward modification of the algorithm presented in the previous appendix can allow the computation of the quantity

$$M(f,g) = \max_{T \in \mathcal{T}_d} \left(\prod_{s \in \delta T} f(s) \prod_{s \in T \backslash \delta T} g(s) \right), \tag{18}$$

where the notations are the same as in Appendix A. This is in particular useful to find the best subtree of $\overline{T}_d$ in terms of the posterior (with the same choice for the functions (f,g) as previously).

Namely, let us define recursively the function θ on the nodes of $\overline{T}_d$ by:

$$\theta(s) = \begin{cases} f(s) & \text{if } s \text{ is a terminal node of } \overline{T}_d \\ \max(f(s), g(s)\theta(s0)\theta(s1)) & \text{otherwise.} \end{cases} \tag{19}$$

It can be shown by a straightforward recursion on the tree depth that $\theta(\emptyset) = M(f,g)$, and so again we can compute recursively $M(f,g)$ in a single sweep through all the nodes of $\overline{T}_d$. Moreover, we can keep track of the different "max" operations in each node so that we actually know at the end exactly which subtree attains the maximum.

C Calculation of the average tree size

During the run of the MCMC chain we want to keep trace of relevant quantities such as the average number of leaves in the tree $|\delta T|$ (or the average total number of nodes in T which is $2|\delta T| - 1$ for a complete binary tree). When the set of questions K on the tree are fixed, the average number of leaves (conditional on these questions) is

$$L(K) = \frac{1}{Z_0(K)} \sum_{T \in \mathcal{T}_d} \pi_1(T) |\delta T| \prod_{s \in \delta T} Q^E_{(T,K)}(Y^s | X^s),$$

where $|\delta T|$ denotes the set of leaves (terminal nodes) of the tree T, Z_0 is the normalizing constant, and the superscript s means that we take the subset of examples that have reached node s: $X^s = \{X_i | T(X_i) = s\}$.

Although this sum cannot be directly factorized as in appendix A, we have

$$\begin{aligned} L(K) &= \frac{1}{Z_0(K)} \frac{\partial}{\partial \xi} \left(\sum_{T \in \mathcal{T}_d} \pi_1(T)(1+\xi)^{|\delta T|} \prod_{s \in \delta T} Q^E_{(T,K)}(Y^s|X^s) \right) \Bigg|_{\xi=0} \\ &= \frac{\partial}{\partial \xi} \left(\log Z_\xi(K) \right) \Big|_{\xi=0}, \end{aligned}$$

where

$$Z_\xi(K) = \sum_{T \in \mathcal{T}_d} \pi_1(T) \prod_{s \in \delta T} (1+\xi) Q^E_{(T,K)}(Y^s|X^s), \tag{20}$$

which now can be computed using the factorization method given in appendix A, so that we can approximate $L(K)$ by $(Z_\xi(K) - Z_0(K))/\xi Z_0(K)$ for ξ small enough. To control the precision we can write

$$\begin{aligned} |(1+\xi)^n - 1 - n\xi| &= \int_0^\xi x \, \frac{\partial^2}{\partial u^2} (1+u)^n \bigg|_{u=x} dx \\ &\leq \frac{n(n-1)}{2} \xi^2 (1+\xi)^{n-2} \\ &\leq \frac{(n\xi)^2}{2} \exp{(n\xi)} \,, \end{aligned}$$

so that in pratice for example with $n \leq 256$ (i.e. for a maximum tree depth of 8) $\xi = 1.10^{-3}$ gives us a .05 accurate value (note that since we are integrating $(1+\xi)^{|\delta T|}$ with respect to a probability distribution we get the same precision on $L(K)$). We can then average this quantity over the run of the MCMC chain.

Similarly, we can compute the empirical average depth reached by the test examples: given one test example X, when the questions K are fixed, the average depth reached by X can be approximated by the same principle as above if we replace the factor $(1+\xi)$ in (20) by the function

$$h(s,\xi) = \begin{cases} (1+\xi)^{|s|} & \text{if } s \text{ is on the branch followed by } X \\ 1 & \text{otherwise,} \end{cases}$$

where $|s|$ denotes the depth of the node s.

References

[1] Y. Amit, G. Blanchard, and K. Wilder. Multiple randomized classifiers: MRCL. Technical report, University of Chicago, nov 1999. available at `http://galton.uchicago.edu/~amit/Papers/mrcl.ps.gz`.

[2] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9:1545–1588, 1997.

[3] Y. Amit, D. Geman, and K. Wilder. Joint induction of shape features and tree classifiers. *IEEE Trans. PAMI*, 19(11):1300–1306, 1997.

[4] G. Blanchard. The "progressive mixture" estimator for regression trees. *Annales de l'I.H.P.*, 35(6):793–820, 1999.

[5] L. Breiman. Bagging predictors. *Machine Learning*, 26:123–140, 1996.

[6] L. Breiman. Arcing classifiers. *The annals of statistics*, 26(3):801–849, 1998.

[7] L. Breiman. Random forests - random features. Technical report, University of California, Berkeley, 1999.

[8] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and regression Trees*. Wadsworth, Belmont, California, 1984.

[9] O. Catoni. Gibbs estimators. *Preprint of the Laboratoire de Mathmatiques de l'Ecole Normale Suprieure*, LMENS-98(21):1–23, may 1998. available at `http://www.dma.ens.fr/edition/publis/index.html`.

[10] O. Catoni. "Universal" aggregation rules with exact bias bounds. *Preprint of the laboratoire de Probabilités et modèles aléatoires, Université Pierre et Marie Curie*, 1999. available at `http://www.proba.jussieu.fr/mathdoc/preprints/index.html#1999` (to appear in *Annals of Statistics*).

[11] H. Chipman, E. I. George, and E. McCulloch. Bayesian CART model search. *JASA*, 93:935–947, September 1998.

[12] D. Denison, B. Mallick, and A. F. M. Smith. A bayesian CART algorithm. *Biometrika*, (85-2):363–377, June 1998.

[13] Y. Freund and R. Schapire. Experiments on a new boosting algorithm. *Machine Learning: proceedings of the 13th international conference*, pages 148–156, 1996.

[14] F. M. J. Willems, Y. M. Shtarkov, and T J. Tjalkens. The context-tree weighting method: basic properties. *IEEE transactions on information theory*, 41(3):653–664, 1995.