

Continuous-Time Birth-Death MCMC for Bayesian Regression Tree Models

Reza Mohammadi

*Amsterdam Business School
University of Amsterdam
Amsterdam, The Netherlands*

A.MOHAMMADI@UVA.NL

Matthew Pratola

*Department of Statistics
The Ohio State University
Ohio, USA*

MPRATOLA@STAT.OSU.EDU

Maurits Kaptein

*Statistics and Research Methods
University of Tilburg
Tilburg, The Netherlands*

M.C.KAPTEIN@UVT.NL

Editor: Sayan Mukherjee

Abstract

Decision trees are flexible models that are well suited for many statistical regression problems. In the Bayesian framework for regression trees, Markov Chain Monte Carlo (MCMC) search algorithms are required to generate samples of tree models according to their posterior probabilities. The critical component of such MCMC algorithms is to construct “good” Metropolis-Hastings steps to update the tree topology. Such algorithms frequently suffer from poor mixing and local mode stickiness; therefore, the algorithms are slow to converge. Hitherto, authors have primarily used *discrete*-time birth/death mechanisms for Bayesian (sums of) regression tree models to explore the tree-model space. These algorithms are efficient, in terms of computation and convergence, only if the rejection rate is low which is not always the case. We overcome this issue by developing a novel search algorithm which is based on a *continuous*-time birth-death Markov process. The search algorithm explores the tree-model space by jumping between parameter spaces corresponding to different tree structures. The jumps occur in continuous time corresponding to the birth-death events which are modeled as independent Poisson processes. In the proposed algorithm, the moves between models are *always* accepted which can dramatically improve the convergence and mixing properties of the search algorithm. We provide theoretical support of the algorithm for Bayesian regression tree models and demonstrate its performance in a simulated example.

Keywords: Bayesian Regression trees, Decision trees, Continuous-time MCMC, Bayesian structure learning, Birth-death process, Bayesian model averaging, Bayesian model selection.

1. Introduction

Classification and regression trees (Breiman et al., 1984) provide a flexible modeling approach using a binary decision tree via splitting rules based on a set of predictor variables. Tree models often perform well on benchmark data sets, and they are, at least conceptually, easy to understand (De’ath and Fabricius, 2000). Tree-based models, and their extensions such as ensembles of trees (Prasad et al., 2006) and sums of trees (Chipman et al., 2010) are an active research area and arguably some of the most popular machine learning tools (Biau, 2012; Biau et al., 2008; Chipman et al., 1998; Denison et al., 1998; Chipman et al., 2002; Wu et al., 2007; Linero, 2018; Au, 2018; Probst and Boulesteix, 2017; Pratola et al., 2014).

Much contemporary research work has focused on Bayesian formulations of regression trees (see, e.g., Denison et al., 1998; Chipman et al., 2010). The Bayesian paradigm provides, next to a good predictive performance, a principled method for quantifying uncertainty (Robert, 2007). This Bayesian formulation can, amongst other uses, be extremely valuable in sequential decision problems (Robbins, 1985; Gittins et al., 2011) and active learning (Cohn et al., 1996) for which popular approaches include Thompson sampling (Thompson, 1933; Agrawal and Goyal, 2012). It is vital to know not merely the expected values (or some other point estimate) of the modeled outcome, but rather to obtain a quantitative formulation of the associated uncertainty (Eckles and Kaptein, 2014, 2019). This is exactly what Bayesian methods readily provide (Robert, 2007).

Recent Bayesian formulations of regression trees have already found their way into many applications (Gramacy and Lee, 2008), but computationally efficient sampling algorithms for tree models and sum-of-tree models have proven non-trivial: the tree-model space of possible trees grows rapidly as a function of the number of features and efficient exploration of this space has proven cumbersome (Pratola, 2016). Numerous methods have been proposed to address this problem; indeed, the popular sums-of-trees model specification proposed by Chipman et al. (2010) is itself an attempt to reduce the tree depth and thereby partly mitigate the problem. Other recent approaches have focussed on efficiently generating Metropolis-Hasting (MH) proposals in the Markov Chain Monte Carlo (MCMC) algorithm (see Pratola, 2016; Wu et al., 2007, for examples), or alternatives to the MH sampler such as sequential MCMC (Taddy et al., 2011) and particle-based approaches (Lakshminarayanan et al., 2013).

To the best of our knowledge, the most effective search algorithm known at this point in time is provided by Pratola (2016), who efficiently integrates earlier advances and adds a number of novel methods to generate tree proposals. Pratola (2016) implements these methods to explore the tree-space by using a search algorithm that is known as reversible jump MCMC (RJ-MCMC) (Green, 1995) which is based on an ergodic, *discrete*-time Markov chain. The RJ-MCMC algorithm often suffers from high rejection rates especially when the model space is large which is the case for the decision tree models. Therefore, these algorithms often are poor mixing and slow to converge.

In this paper, to overcome this issue, we make a significant contribution to the Bayesian decision tree literature by proposing a novel *continuous*-time MCMC (CT-MCMC) search algorithm which is essentially the continuous version of the RJ-MCMC algorithm. The main advantage of the CT-MCMC algorithm is that each step of the MCMC algorithm considers the whole set of transitions and a transition always occurs; in fact, there is no rejection.

Thus, the CT-MCMC has clearly better performances in terms of computational time and convergence rate. The proposed CT-MCMC search algorithm is based on the construction of continuous-time Markov birth-death processes (introduced by Preston 1977) with the appropriate stationary distribution. Sampling algorithms based on these processes have already been used successfully in the context of mixture distributions by Stephens (2000); Cappé et al. (2003); Mohammadi et al. (2013). In the case of mixture distributions, the birth-death mechanisms have been implemented in the MCMC algorithm in such a way that the algorithm explores the model space by adding/removing a component for the case of a birth/death event. More recently, such MCMC algorithms have been used in the field of (Gaussian) graphical models (Mohammadi and Wit, 2015; Mohammadi et al., 2017a; Dobra and Mohammadi, 2018; Wang et al., 2020; Dobra and Mohammadi, 2018; Hinne et al., 2014; Mohammadi and Wit, 2019; Mohammadi et al., 2017b). For the case of graphical models, the birth-death mechanisms have been implemented in the MCMC algorithm in such a way that the algorithm explores the graph space by adding/removing a link for the case of a birth/death event.

We apply this continuous-time MCMC mechanism to the classification and regression tree (CART) model context, by considering the parameters of the model as a point process, in which the points represent the nodes in the tree model. The MCMC algorithm explores the tree space by allowing new terminal nodes to be *born* and existing terminal nodes to *die*. These birth and death events occur in continuous time, as independent Poisson processes; see Figure 3. We design the MCMC algorithm in such a way that the relative rates of the birth/death events determine the stationary distribution of the process. In Section 3 we formalize the relationship between the birth/death rates and the stationary distribution. Based on this we construct the MCMC search algorithm in which the birth/death rates are the ratios of the posterior distributions. We show how to use the advantage of continuous-time sampling to efficiently estimate the parameter of interest based on model averaging, using Rao-Blackwellization Cappé et al. (2003).

This paper is structured as follows. In the next section we introduce the tree and sums of tree models more formally and introduce the sampling challenges associated with this model in more detail. Next, in Section 3 we detail our suggested alternative birth-death approach and provide both an efficient algorithm and the theoretical justification for our proposal. Subsequently, we extend this proposal to also include the rotation moves suggested by Pratola (2016). In Section 4 we compare the performance of our method—in terms of both its statistical properties and its computation time—to the current state of the art (Pratola, 2016) using a simple, well-known, example that is notoriously challenging for tree models (Wu et al., 2007). Finally, in Section 5 we discuss the limitations of our contribution and provide pointers for future work.

2. Bayesian tree models

We consider binary regression or classification trees and sum of trees models. Given a feature vector $x = (x_1, \dots, x_d)$, and a scalar output of interest y we can denote the tree model as follows

$$y = g(x; T, \theta_T) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

where T denotes the interior nodes of the tree, θ_T denotes a set of maps associated with the terminal nodes. Effectively, T encodes all the (binary) split rules that jointly generate the tree structure. This is often expressed using a list of tuples $\{(\nu_1, c_1), (\nu_2, c_2), \dots\}$ where $\nu_i \in \{1, \dots, d\}$ indicates which element of the feature vector to split on, and c_i denotes the associated value of the split (see, e.g., Pratola, 2016). This way of expressing the tree is however limited since it does not encode the actual topology τ of the tree, which encodes the number of nodes in a tree, whether a node is internal or terminal, parent/child edges, and node depths. Hence, more precisely τ and $\{(\nu_1, c_1), (\nu_2, c_2), \dots\}$ *jointly* make up the full tree structure T . Figure 1 illustrates our notation at this point in the paper; in Section 3 we will gradually introduce some additional notation necessary for our theoretical justification.

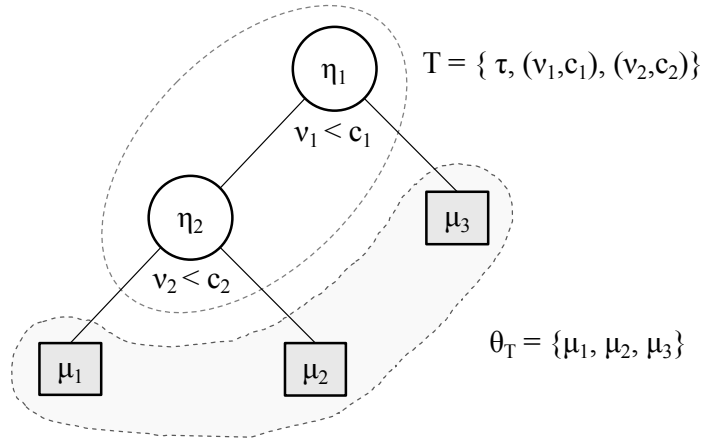


Figure 1: A simple example of our main notation for a tree model which has 2 interior nodes (η_1, η_2). T encodes both the split rules $\{(\nu_1, c_1), (\nu_2, c_2), \dots\}$ as well as the topology τ . The set of maps $\theta_T = \{\mu_1, \mu_2, \dots, \mu_{n_t}\}$ determines the values of the terminal nodes.

Given the number of terminal nodes, n_t , the maps $\theta_T = \{\mu_1, \mu_2, \dots, \mu_{n_t}\}$ take as input a feature vector x and produce a response $\mu_j(x)$. In typical tree regression models the maps are constants; $\mu_j(x) = \mu_j$. Taken together, T represents a partitioning of the feature space and a mapping from an input feature x to a response value encoded in θ_T .

The Bayesian formulation of the tree model is completed by using priors of the form

$$\pi(T, \theta_T, \sigma^2) = \pi(\theta_T | T) \pi(T) \pi(\sigma^2).$$

Note that the sum-of-trees model (Chipman et al., 2010) provides a conceptually straightforward extension of the above specified single tree model

$$y = \sum_{m=1}^M g(x; T_m, \theta_{T_m}) + \epsilon \quad (1)$$

where the sum runs over M distinct trees whose outputs are added. In the case of the sum-of-trees model we have

$$\pi(T_1, \theta_{T_1}, \dots, T_M, \theta_{T_M}, \sigma^2) = \left[\prod_{m=1}^M \pi(\theta_{T_m} | T_m) \pi(T_m) \right] \pi(\sigma^2).$$

For more details related to the sum-of-tree models we refer to Pratola (2016).

2.1 Specification of the tree prior

We specify the prior $\pi(T)$ by three parts,

- The distribution on the splitting variable assignments at each interior node ν as a discrete random quantity in $\{1, \dots, d\}$.
- The distribution on the cut-point c as a discrete random quantity in

$$\left\{ 0, \frac{1}{n_\nu - 1}, \dots, \frac{n_\nu - 2}{n_\nu - 1} \right\}$$

where n_ν is the resolution of discretization for variable η .

- The prior probability that a node i (η_i) at depth d_i is non-terminal to be

$$\pi(\eta_i) \propto \frac{\alpha}{(1 + d_i)^\beta}, \quad \alpha \in (0, 1), \beta \geq 0.$$

To specify the prior distributions on bottom-node ν 's, we use standard conjugate form

$$\theta_T = \{\mu_1, \mu_2, \dots, \mu_{n_t}\} | T \stackrel{iid}{\sim} \mathcal{N}(\hat{\mu}, \sigma_\mu^2).$$

In practice, the observed Y can be used to guide the choice of the prior parameter values for $(\hat{\mu}, \sigma_\mu^2)$; See e.g. Chipman et al. (1998). Here for simplicity we assume $\hat{\mu} = 0$.

For a prior specification of the σ , we also use a conjugate inverse chi-square distribution prior

$$\sigma^2 \sim \mathcal{IG} \left(\frac{\nu}{2}, \frac{\nu\lambda}{2} \right)$$

which results in simple Gibbs updates for the variance. In practice, the observed Y can be used to guide the choice of the prior parameter values for (ν, λ) ; See e.g. Chipman et al. (1998).

2.2 Sampling from tree-models

For a single tree the full posterior of the model, for given tree (T, θ_T) , σ , and data $\mathcal{D}$ is

$$\Pr(T, \theta_T, \sigma | \mathcal{D}) \propto L(T, \theta_T, \sigma) \pi(\theta_T | T) \pi(T) \pi(\sigma). \quad (2)$$

For a single tree—which could easily be extended to the sum-of-tree case—sampling from the full posterior of the model 2 is conceptually carried out by iterating the following steps

1. Draw a new topology $\tau|y, \sigma^2, \{(\nu_i, c_i)\}$ using some method of generating new topologies such as a birth/death or rotation and subsequently accepting or rejecting the proposal.
2. Draw the split rules $(\nu_i, c_i)|y, \tau, \sigma^2, \{(\nu_{-i}, c_{-i})\}, \forall i$ using perturb or perturb within change-of-variable proposals.
3. Draw $\mu_j|y, \tau, \sigma^2, \{(\nu_i, c_i)\}$ using conjugate Gibbs sampling.
4. Draw $\sigma^2|y, \tau, \{\mu_j\}, \{(\nu_i, c_i)\}$ also using conjugate Gibbs scheme.

The above algorithm has been implemented successfully in earlier work (see Pratola, 2016). Steps 3 and 4 are the standard Gibbs sampling using conjugate priors. Also Step 2 is efficiently implemented by (Pratola, 2016, Section 4). For the sampling of $\tau|y, \sigma^2, \{(\nu_i, c_i)\}$ (i.e., in step 1 above) the current state-of-the art is to use an RJ-MCMC search algorithm. In practice the RJ-MCMC algorithm performs well if the rejection rate is low (the computation of which is detailed in Equation 4 of Pratola, 2016). However, when the rejection rate is not low—which is often the case—the mixing of the chain is poor and the exploration of the full tree-model space is notoriously slow. To overcome this issue, in the next section, we introduce a novel search algorithm which has basically no rejection rate. So the novelty of our work mainly lies in the new search algorithm for step 1.

3. Continuous-time birth-death MCMC search algorithm

The issue of a low acceptance rate in step 1 of the algorithm mentioned in the previous section is surprisingly common: as the tree space is extremely large, proposals with a low likelihood are frequent. This specific issue can however be overcome by adopting a *continuous*-time Markov process—or a CT-MCMC search algorithm—as an alternative to RJ-MCMC. In this sampling scheme the algorithm explores the tree-model space by either jumping to a larger dimension (birth) or lower dimension (death) as in step 1 above. But this time each of these events is modeled as an independent Poisson process and the time between two successive events is exponentially distributed. The change events thus occur in continuous time and their rates determine the stationary distribution of the process; see Figure 2 for a graphical overview of possible birth and deaths from a given tree. Unlike the RJ-MCMC, in the CT-MCMC search algorithm the moves between models are *always* accepted making the algorithm more efficient.

Cappé et al. (2003) have shown, on appropriate re-scaling of time, that the RJ-MCMC converges to a continuous-time birth-death chain. One advantage of CT-MCMC is its ability to transit to low probability regions that can form a kind of “springboard” for the algorithm to flexibly move from one mode to another.

Our strategy is to view each component of the terminal nodes of the tree as a point in parameter space, and construct a Markov chain with the posterior distribution of the parameters as its stationary distribution. For given tree (T, θ_T) and data $\mathcal{D}$, the target posterior distribution is

$$\Pr(T, \theta_T | \mathcal{D}) \propto L(T, \theta_T) \pi(T) \pi(\theta_T) \quad (3)$$

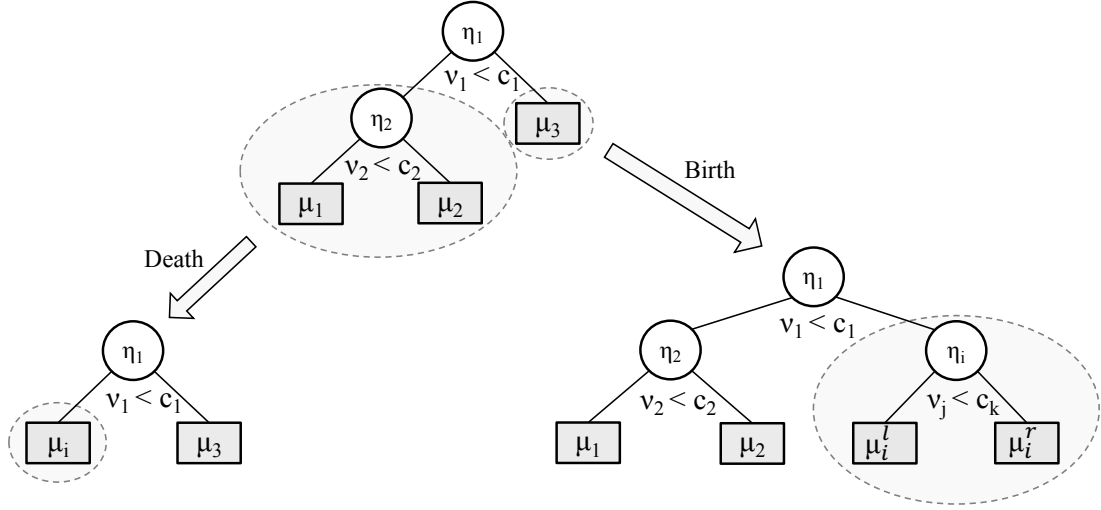


Figure 2: The birth-death mechanism for adding or deleting nodes of the tree. On the bottom left a death occurs at node η_2 from the original resulting in the removal of maps (μ_2, μ_3) and the emergence of a new map μ_i . On the bottom right a new node η_i is born at map μ_1 resulting in the removal of this map and the addition of ν_j , c_k , and (μ_i^l, μ_i^r) .

where $L(T, \theta_T)$ is the likelihood. Note that the proposed search algorithm for sampling the tree model can then be combined with conjugate Gibbs updates of the continuous parameters such as σ^2 similar to the Metropolis-within-Gibbs algorithm; see for example Stephens (2000).

We take advantage of the theory on general classes of Markov birth-death processes from Preston (1977, Section 7 and 8). This class of Markov jump processes evolve in jumps which occur a finite number of times in any finite time interval. These jumps are of two types: (i) *birth* in which a single point is added, and the process jumps to a state that contains the additional point; and (ii) *death* in which one of the points in the current state is deleted, and the process jumps to a state with one less point. Preston (1977) shows that this process converges to a unique stationary distribution provided if the detailed balance conditions hold.

To properly define the birth and death events in our case we need to introduce some additional notation identifying the different nodes in the tree and their respective variables and cut-points. Let (T, θ_T) define the tree model as before, additionally let n_t be the number of terminal nodes, n_ν the number of variables, and n_c the number of cut-points. Given the current state (T, θ_T) :

Birth: A new terminal node is created (born) in continuous time with a *birth rate* $B_{ijk}(T, \theta_T)$; we denote this operation by ‘ $\cup$ ’. In this case, the process transits to a new state

$$(T^{b_{ijk}}, \theta_{T^{b_{ijk}}}) = (T \cup (\eta_i, \nu_{ij}, c_{ijk}), \theta_T \cup (\mu_i^l, \mu_i^r) \setminus \mu_i)$$

where η_i denotes to internal node i , $i \in 1, \dots, n_t$, $j \in 1, \dots, n_\nu$, and $k \in 1, \dots, n_c$. Further, we define the total birth rate as

$$B(T, \theta_T) = \sum_{i=1}^{n_t} \sum_{j=1}^{n_\nu} \sum_{k=1}^{n_c} B_{ijk}(T, \theta_T).$$

Hence, a birth event changes the topology τ of the current tree T by adding a terminal node i . Accordingly, to complete the specification of the new tree $(T^{b_{ijk}}, \theta_{T^{b_{ijk}}})$ we also need to add variable ν_j and cut-point c_k as well as the new terminal maps (μ_i^l, μ_i^r) . This process is illustrated in Figure 2 on the bottom right where a birth occurs at map μ_1 .

Death: In the current state (T, θ_T) with n_d terminal nodes, one of the terminal nodes is killed in continuous time with *death rate* $D_i(T, \theta_T)$; we denote this operation by ‘ $\setminus$ ’. In this case, the process transits to state

$$(T^{d_i}, \theta_{T^{d_i}}) = (T \setminus (\eta_i, \nu_i, c_i), \theta_T \setminus (\mu_i^l, \mu_i^r) \cup \mu_i)$$

where $i \in 1, \dots, n_d$ and n_d is the number of possible deaths. Also, we define the total death rate as

$$D(T, \theta_T) = \sum_{i=1}^{n_d} D_i(T, \theta_T).$$

Hence, a death event changes the topology τ by removing node i , including its associated variable and cut-point, (ν_i, c_i) and their respective maps (μ_i^l, μ_i^r) . Accordingly, to complete the specification of the tree, we need to add a new map μ_i . This process is illustrated in Figure 2 on the bottom left where a death occurs at node η_2 .

Since birth and death events are independent Poisson processes, the time between two consecutive events has an exponential distribution with mean

$$W(T, \theta_T) = \frac{1}{B(T, \theta_T) + D(T, \theta_T)} \quad (4)$$

which is the *waiting time*. Note that the waiting times are calculated based on all the possible birth and death moves from the current state (T, θ_T) to a new state which would be a tree with one more or less terminal nodes regarding to the birth/death rates. Therefore, the waiting times essentially capture all the possible moves of each step of the CT-MCMC search algorithm. If the waiting time from (T, θ_T) is large then the process tends to stay longer in the current state while if the waiting time is small, the process tends to transition away from the current state. The birth and death probabilities involved are

$$\Pr(\text{birth at node } \eta_i \text{ for variable } \nu_j \text{ and cut-point } c_k) = \frac{B_{ijk}(T, \theta_T)}{B(T, \theta_T) + D(T, \theta_T)}, \quad (5)$$

$$\Pr(\text{death at node } \eta_i) = \frac{D_i(T, \theta_T)}{B(T, \theta_T) + D(T, \theta_T)}. \quad (6)$$

The corresponding Markov process converges to the target posterior distribution in Equation 3 given sufficient conditions that are provided in the following theorem.

Theorem 1 *The birth-death process defined by the birth and death probabilities in Equations 5 and 6 has stationary distribution $Pr(T, \theta_T \mid \mathcal{D})$, provided birth and death rates satisfy*

$$B_{ijk}((T, \theta_T)Pr(T, \theta_T \mid \mathcal{D})Pr(\mu_l^n)Pr(\mu_r^n) = D_i(T^{b_{ijk}}, \theta_{T^{b_{ijk}}})Pr(\mu_i)Pr(T^{b_{ijk}}, \theta_{T^{b_{ijk}}} \mid \mathcal{D})).$$

Proof. Our proof draws on the theory of general continuous-time Markov birth-death processes derived by (Preston, 1977, Section 7 and 8). The process evolves by jumps which occur a finite number of times in any finite time interval. The jumps are of two types: a *birth* in which the process jumps to a state with the additional point, whereas a *death* in which the process jumps to a state with one less point by deleting one of the points in the current state. For the general case, Preston (1977, Theorem 7.1) proves the process converges to the target stationary distribution if the detailed balance conditions hold, as described in Theorem 1. For the case of a decision tree, if a birth occurs then we add one node to the current tree and if a death occurs we remove one node from the current tree. We design the CT-MCMC search algorithm in such a way that the stationary distribution of the process is our target posterior distribution Equation 3. For a detailed proof see the Appendix A.

Based on Theorem 1, we can derive the birth and death rates as a function of the ratio of the target posterior distribution as follows

$$B_{ijk}(T, \theta_T) = \min \left\{ 1, \frac{\Pr(T^{b_{ijk}}, \theta_{T^{b_{ijk}}} \mid \mathcal{D})Pr(\mu_i)}{\Pr(T, \theta_T \mid \mathcal{D})Pr(\mu_l^n)Pr(\mu_r^n)} \right\} \quad (7)$$

and

$$D_i(T, \theta_T) = \min \left\{ 1, \frac{\Pr(T^{d_i}, \theta_{T^{d_i}}} \mid \mathcal{D})Pr(\mu_l^i)Pr(\mu_r^i)}{\Pr(T, \theta_T \mid \mathcal{D})Pr(\mu_n)} \right\}. \quad (8)$$

Given the results provided above, our proposed algorithm for the posterior sampling from a (sums of) tree model is presented in the Algorithm 1.

Algorithm 1 . CT-MCMC search algorithm

Input: A tree (T, θ_T) , data $\mathcal{D}$.

for N iterations **do**

for all the possible moves (for $i \in 1, \dots, n_t, j \in 1, \dots, n_\nu, k \in 1, \dots, n_c$) in parallel **do**

 Draw the new split rules (ν_j, c_k) .

 Draw the new μ_i 's.

 Calculate the birth rates $B_{ijk}(T, \theta_T)$ and death rates $D_i(T, \theta_T)$ according to Equations 7 and 8.

end for

 Calculate the waiting time $W(T, \theta_T)$ given by Equation 4.

 Update the new topology τ based on birth/death probabilities in Equations 5 and 6.

 Update σ^2 using standard Gibbs sampling scheme.

end for

Output: Samples from the full posterior distribution, Equation 2.

Algorithm 1 presents the pseudo-code for the CT-MCMC search algorithm which samples from the posterior distribution in Equation 2 by using the continuous-time birth-death mechanism that is described above. Basically, in the CT-MCMC search algorithm, we only

simulate the jump chain and store each tree which visits and the corresponding waiting time. For the graphical visualization of the algorithm see Figure 3.

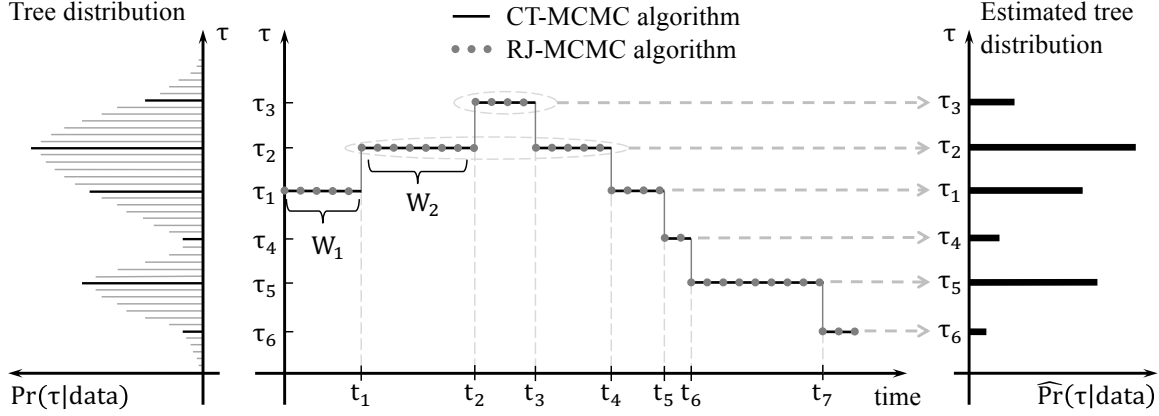


Figure 3: Graphical representation of the CT-MCMC algorithm versus RJ-MCMC algorithm. The left panel presents the target posterior distribution of the tree topologies. The middle panel shows the sampling scheme of CT-MCMC and RJ-MCMC search algorithms. CT-MCMC algorithm samples in continuous time and $\{W_1, W_2, \dots\}$ stand for waiting times (or holding times) and $\{t_1, t_2, \dots\}$ stand for jumping times of the CT-MCMC algorithm; while the RJ-MCMC algorithm samples in discrete-time and dots visualize the sampling scheme of the algorithm. The right panel shows the estimated posterior distribution of the tree topologies based on the CT-MCMC sampler which are the proportional to the total waiting times of the visited trees, according to the Rao-Blackwellized estimator; see Subsection 3.2.

One important feature of the CT-MCMC search algorithm is that a continuous time jump process is associated with the birth and death rates (Equations 7 and 8): whenever a jump occurs, the corresponding move is always accepted. In fact, the acceptance probability of usual RJ-MCMC search algorithm is replaced by the waiting times (4) in the CT-MCMC search algorithm. Particularly, implausible trees, i.e. trees with low posterior probabilities have small waiting times and as a result die quickly; Conversely, plausible trees, i.e. trees with high posterior probabilities have larger waiting times. Thus, the CT-MCMC search algorithm are efficient to detect the high posterior probabilities regimes particularly for high-dimensional space models.

3.1 Computational improvements and further additions

The key computational bottleneck of the CT-MCMC search algorithm is the computation of the birth and death rates over all the possible moves of the next step; The number of all possible moves exponentially increases with respect to the size of the tree-topology. Fortunately, in each step of the search algorithm, the birth and death rates can be calculated independently of each other; Thus, the rates can be computed in parallel which represents

a key computational improvement of the CT-MCMC search algorithm with respect to RJ-MCMC. We implement this step of the above algorithm in parallel using OpenMP in C to speed up the computations.

While Algorithm 1 is feasible, in practice, it can be improved by

- a) exploiting conjugacy,
- b) including rotation proposals (as initially suggested by Pratola, 2016, for the RJ-MCMC case).

Below we detail each in turn.

Conjugate priors on the terminal node parameters $\mu_i \in \Theta_T$, $i \in \{1, \dots, n_t\}$, can simplify the CT-MCMC algorithm. In the example below we are interested in modeling a continuous response which leads to i.i.d. priors $\pi(\mu_i) \sim N(0, \tau^2)$ (Chipman et al., 2010). Marginalizing out a single terminal node parameter μ_i , the integrated likelihood is given by

$$Pr(T, \overline{\theta_T} | \mathcal{D}) = \int_{\mu_i} Pr(T, \theta_T) \pi(\mu_i) d\mu_i$$

which is available in closed form for conjugate priors (similarly for integrating two terminal node parameters). Applying this marginalization to Equations 7 and 8, the updated birth and death rates for CT-MCMC search algorithm are

$$B_{ijk}(T, \overline{\theta_T}) = \min \left\{ 1, \frac{\Pr(T^{b_{ijk}}, \overline{\theta_{T^{b_{ijk}}}} | \mathcal{D})}{Pr(T, \overline{\theta_T} | \mathcal{D})} \right\} \quad (9)$$

and

$$D_i(T, \overline{\theta_T}) = \min \left\{ 1, \frac{\Pr(T^{d_i}, \overline{\theta_{T^{d_i}}}} | \mathcal{D})}{Pr(T, \overline{\theta_T} | \mathcal{D})} \right\}. \quad (10)$$

Considering the above birth and death rates, we present our implemented CT-MCMC search algorithm in Algorithm 2.

Algorithm 2 . CT-MCMC search algorithm - exploiting conjugacy

Input: A tree $(T, \overline{\theta_T})$, data $\mathcal{D}$.

for N iterations **do**

for all the possible moves (for $i \in 1, \dots, n_t$, $j \in 1, \dots, n_\nu$, $k \in 1, \dots, n_c$) in parallel **do**

 Draw the new split rules (ν_j, c_k) .

 Calculate the birth rates $B_{ijk}(T, \overline{\theta_T})$ and death rates $D_i(T, \overline{\theta_T})$ according to Equations 9 and 10.

end for

 Calculate the waiting time $W(T, \overline{\theta_T})$ given by Equation 4, using Equations 9 and 10.

 Update the new topology τ based on birth/death probabilities in Equations 5 and 6, using Equations 9 and 10.

 Update σ^2 using standard Gibbs sampling scheme.

end for

Output: Samples from the full posterior distribution, Equation 2.

By integrating out the terminal node parameters $\mu_i \in \Theta_T$, $i \in \{1, \dots, n_t\}$ in our tree model, we essentially exclude a sampling step inside the nested for loop in the Algorithm 2; Thus, this algorithm is computationally more efficient than Algorithm 1.

While until now we have introduced our main results focusing merely on birth-death moves for simplicity, building on recent work by Pratola (2016) we can extend our sampling approach to so-called rotate proposals: rotate proposals can be thought of as a multivariate generalization of the simple univariate rotation mechanism found in the binary search tree literature (see, e.g., Sleator et al., 1988) and implemented in Gramacy and Lee (2008). This generalization allows dimension-changing proposals to occur at any interior node of a tree, and directly moves between modes of high likelihood and is described in detail in Pratola (2016). In Appendix B, we demonstrate the correctness of this approach once added to the proposed birth-death mechanism in the CT-MCMC case. Moreover, we present an efficient way of implementing rotate proposals within Algorithm 1 and 2 using marginalization.

3.2 Posterior inference by samples in continuous time

Figure 3 shows the sampling scheme of CT-MCMC versus RJ-MCMC algorithm and how to estimate posterior quantities of interest using sampling in continuous time, based on model averaging.

Basically, for the case of discrete time RJ-MCMC sampler, we monitor its output after each iteration. In this case, based on model averaging, the posterior means are estimated by sample means

$$E[g(T, \theta_T)] \approx \frac{1}{N} \sum_{i=1}^N g(T_i, \theta_{T_i}) \quad (11)$$

in which N is the number of MCMC iterations. For the CT-MCMC sampler, at each jump, we store each state that it visits and the corresponding waiting time which are $\{W_1, W_2, \dots\}$ in Figure 3. Note that alternative sampling schemes have been proposed – for instance, similar to Stephens (2000), the process may be sampled at regular times; see Cappé et al. (2003).

We use the Rao-Blackwellized estimator (Cappé et al., 2003) to estimate parameters of the models, based on model averaging. It is proportional to the expectation of length of the holding time in that tree which is estimated as the sum of the waiting times in that tree. In this case, the posterior means are estimated by sample means

$$E[g(T, \theta_T)] \approx \frac{\sum_{i=1}^N W_i(T_i, \theta_{T_i}) g(T_i, \theta_{T_i})}{\sum_{i=1}^N W_i(T_i, \theta_{T_i})}. \quad (12)$$

Effectively, the Rao-Blackwellized estimator depends on the waiting times (4) of the visited trees by the CT-MCMC sampler. The waiting times are calculated based on all the possible birth and death moves from the current state (T, θ_T) ; Therefore, the waiting times essentially capture all the possible moves in each step. Therefore, by containing the waiting times in the Rao-Blackwellized estimator, all possible moves are incorporated into our estimation, not only those that are selected. Moreover, according to the Rao-Blackwell theorem, the variances of estimators built from the sampler output are decreased (Cappé et al., 2003).

Note that the Rao-Blackwellized estimator is based on model averaging, which has the advantage that it provides a coherent way of combining results over different models. As a

result, the estimation of the parameter of interest is not based on only one single tree. In fact, the estimation is based on the all trees that are visited by the MCMC search algorithm.

4. Empirical evaluation of our sampling approach

We examine here the performance of the proposed CT-MCMC search algorithm based on a simulation scenario that is often used in the regression tree literature. This simulation scenario serves as a simple demonstration where proper mixing of the regression trees topological structure is important (Wu et al., 2007). The synthetic data set consists of $n = 300$ data points with (x_1, x_2, x_3) covariates where

$$x_{1i} \sim \begin{cases} \text{Unif}(0.1, 0.4), & \text{for } i = \{1, \dots, 200\} \\ \text{Unif}(0.6, 0.9), & \text{for } i = \{201, \dots, 300\} \end{cases} \quad (13)$$

$$x_{2i} \sim \begin{cases} \text{Unif}(0.1, 0.4), & \text{for } i = \{1, \dots, 100\} \\ \text{Unif}(0.6, 0.9), & \text{for } i = \{101, \dots, 200\} \\ \text{Unif}(0.1, 0.9), & \text{for } i = \{201, \dots, 300\} \end{cases} \quad (14)$$

$$x_{3i} \sim \begin{cases} \text{Unif}(0.6, 0.9), & \text{for } i = \{1, \dots, 200\} \\ \text{Unif}(0.1, 0.4), & \text{for } i = \{201, \dots, 300\} \end{cases} \quad (15)$$

Figure 4 shows the partition of the simulation data set with respect to the covariates. Note that, following Wu et al. (2007); Pratola (2016), we generate covariates such that the effects of x_1 and x_3 (see the middle panel in the Figure 4) are confounded which makes this data generating scheme particularly challenging.

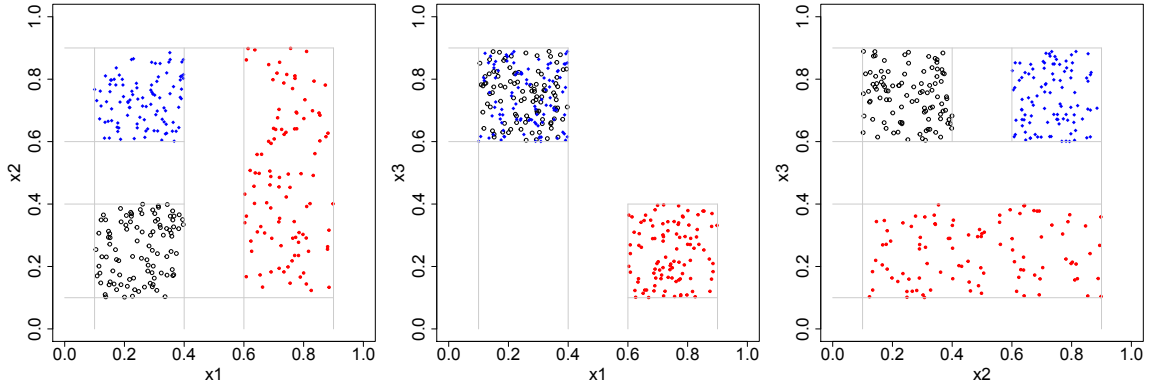


Figure 4: Partition of the simulation data set with respect to the covariates x_1 , x_2 , and x_3 with the three regions defined in 13, 14, and 15.

The response y is calculated for $n = 300$ data points as:

$$y = \begin{cases} 1 + \mathcal{N}(0, \sigma^2) & \text{if } x_1 \leq 0.5, x_2 \leq 0.5 \\ 3 + \mathcal{N}(0, \sigma^2) & \text{if } x_1 \leq 0.5, x_2 > 0.5 \\ 5 + \mathcal{N}(0, \sigma^2) & \text{if } x_1 > 0.5. \end{cases} \quad (16)$$

Figure 5 presents the above regression tree model with the partitions based on the covariates x_1 and x_2 .

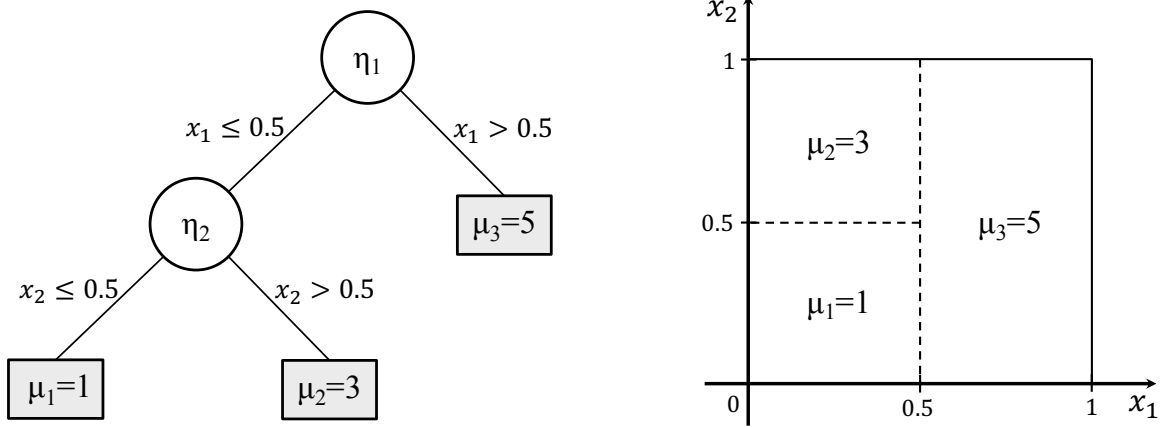


Figure 5: True tree model (left) for the regression tree model in 16 where $y \sim \mathcal{N}(\mu, \sigma^2)$ with three partitions (right) based on the covariates x_1 and x_2 .

Following Pratola (2016) we fit a single tree model (thus $M = 1$ in Equation 1) to this data using the following approaches:

- **RJ-A:** Here we use a straightforward RJ-MCMC algorithm which is based on discrete time birth-death proposals as described in Pratola (2016).
- **RJ-B:** Here we use discrete time RJ-MCMC algorithm to which we add the rotation proposals as described in Pratola (2016).
- **RJ-C:** Here we use discrete time RJ-MCMC algorithm including rotation proposals and perturbation. The latter addition concerns the second step of the sampling procedure as outlined in Section 2 which concerns the sampling of the split rules (ν_i, c_i) . This is not the main focus of this paper, however, we want to see whether this additional mechanism is also useful for the CT-MCMC approach (see Pratola, 2016, for detail).
- **CT-A:** Here we use our proposed CT-MCMC algorithm which is based on continuous-time birth-death approach but without perturbation; see Algorithm 2.
- **CT-B:** Here we add rotation proposals to the CT-MCMC algorithm described in Algorithm 2, again without perturbation; for details we refer to Appendix B.

- **CT-C:** Here we use both birth-death and rotation proposals and we add perturbation proposals to the second step of Algorithm 2.

To evaluate the performance of the CT-MCMC search algorithm with compare with the RJ-MCMC, we run all the above search algorithms in the same conditions with 20,000 iterations and 1,000 as a burn-in. We perform all the computations in R and the computationally intensive tasks are implemented in parallel in C and interfaced in R. All the computations were carried out on an MacBook Pro with 2.9 GHz processor and Quad-Core Intel Core i7.

For each of the above search algorithms, we report the following measurements:

- **MSE:** This is the Mean Square Error. To calculate the MSE, we generate another synthetic data set consists of $n = 300$ data points as a test set. Then, we compute the MSE of these test set based on the estimated tree models from the above MCMC search algorithms.
- **Effective Sample Size:** This is the number of effective independent draws that the algorithm generates.
- **Activity:** This is the proportion of splits on a given variable making up the tree decision rules. In this synthetic example it is possible to derive the variable activity analytically which should be approximately 0.3, 0.4, and 0.3 respectively.
- **Unique Trees:** The number of unique trees the algorithm generates.
- **Effective Sample Size per Second:** The number of effective samples drawn per second of computation time.

For the CT-MCMC approach, we estimate the parameter of interest based on the Rao-Blackwellized estimator in Equation 12 and for the case of RJ-MCMC it is based on sample means in Equation 11. Thus, all estimates are based on model averaging across all visited tree models; see Figure 3.

Table 1 presents the results for $\sigma^2 = 1$ which is a relatively challenging, high-noise, scenario. On average, over 100 replications, the prediction error of each of the models is similar, and hence, as expected, the different sampling methods do not differentiate in terms of predictive performance. However, in terms of computational efficiency, measured as the effective sample size computed based on the posterior draws, it is clear that the CT-MCMC methods perform better than the RJ-MCMC approaches across the board. Our most elaborate proposal—combining CT-MCMC with both rotation proposals and perturbation proposals (CT-C) —provides the best performance. This is especially prominent when looking at the exploration behavior of the different sampling methods as summarized by the number of unique trees visited.

Finally, it is clear that the computation time, measured as effective samples per second, of our newly proposed methods is on par, or faster, than the current state-of-art methods. To summarize, across the board we find a good empirical performance of our suggested CT-MCMC method(s). Appendix C provides additional simulation results for the cases $\sigma^2 \in (0.01, 0.1)$ showing that in both of these cases our suggested method again outperforms the RJ methods. In fact, in these lower noise scenarios the RJ methods fail to properly explore the parameters space while our suggested CT method maintains proper variable

Method	MSE	Effective Sample Size				Activity		
		σ^2	x_1	x_2	x_3	x_1	x_2	x_3
RJ-A	1.02	19758	1037	2370	1265	0.27	0.45	0.28
RJ-B	1.02	19774	1419	2899	1482	0.25	0.49	0.26
RJ-C	1.02	19625	13134	1306	13144	0.25	0.5	0.25
CT-A	1.02	19282	14160	32374	14128	0.28	0.43	0.28
CT-B	1.02	19577	40041	74608	37925	0.27	0.48	0.25
CT-C	1.02	20474	14452	51557	14518	0.26	0.47	0.26

Method	Unique Trees	Effective Sample Size per Second			
		σ^2	x_1	x_2	x_3
RJ-A	1.83	26344	1382	3160	1686
RJ-B	3.07	21971	1576	3221	1646
RJ-C	8.29	11410	7636	759	7642
CT-A	11.44	5371	3944	9018	3935
CT-B	3.82	5515	11279	21016	10683
CT-C	11.71	4611	3255	11612	3270

Table 1: Overview of the performance measures of different sampling methods for simulation example for the case $\sigma^2 = 1$ in Equation 16 . The table reports the average over 100 replications of the prediction error, the sampling efficiency, the exploration behavior in terms of variable activity measured as the average proportion of internal rules involving each variable, the exploration behavior in terms of the average number of unique trees visited, and computational efficiency in effective samples per second.

activity while better sampling tree-space. We also note that the Rao-Blackwellization makes a greater impact in these low-noise scenarios, with an unweighted MSE of 0.014 for CT-C versus Rao-Blackwellized estimate of 0.01 reported in Table 2 and an unweighted MSE of 1.98E-3 for CT-C versus the Rao-Blackwellized estimate of 1.04E-4 reported in Table 3.

5. Discussion

In this paper we introduced a continuous-time MCMC search algorithm for posterior sampling from Bayesian regression trees and sums of trees (BART). Our work is inspired by earlier work in this space demonstrating the efficiency of continuous-time MCMC search algorithms (see, e.g., Mohammadi and Wit, 2015; Mohammadi et al., 2017a). Using the general theory described by Preston (1977) we have shown analytically that our proposed sampling approach converges to our desired target posterior distribution $\Pr(T, \theta_T \mid \mathcal{D})$ in the case of birth-death proposals. Next, we extended this result to also include the novel rotate proposals initially proposed by Pratola (2016). Jointly, these suggestions lead to an efficient sampling mechanism for Bayesian (additive) regression trees; a model that is gaining popularity in applied studies (see, e.g., Logan et al., 2019) and hence effective sampling methods are sought after.

The current work provides theoretical guarantees regarding the convergence of the CT-MCMC search algorithm. There is still room for additional computational improvements: while our marginalizing approach, combined with our mixture approach to include rotation proposals (see Appendix B), provide important steps in providing a computationally feasible CT-MCMC method, we believe additional gains might be possible. Furthermore, while our current implementation parallelizes parts of the sampling process, additional gains might be achieved here. The current implementation of the methods proposed in this paper are available at <https://bitbucket.org/mpratola/openbt>.

We hope our current results improve the practical usability of Bayesian regression tree models for applied researchers by speeding up, and improving the accuracy, of the sampling process. Our methods seem to work well for reasonably sized problems (e.g., thousands of observations, tens of variables); we think their actual performance on big data sets needs to be further investigated.

Appendix A. Proof of Theorem 1

Our proof here is based on the theory of general continuous-time Markov birth-death processes derived by Preston (1977). We use the notation defined in the body of this paper. Assume that at a given time, the process is in a tree state (T, θ_T) . The process is characterized by the *birth rates* $B_{ijk}(T, \theta_T)$, the *death rates* $D_i(T, \theta_T)$, and the birth and death *transition kernels* $K_B((T, \theta_T) \rightarrow (T^*, \theta_{T^*}))$ and $K_D((T, \theta_T) \rightarrow (T^*, \theta_{T^*}))$.

Birth and death events occur as independent Poisson processes with rates $B_{ijk}(T, \theta_T)$ and $D_i(T, \theta_T)$ respectively. Given that a specific birth occurs, the probability that the following jump leads to a point in $H \subset \Omega_{T^{b_{ijk}}}$ (where $\Omega_{T^{b_{ijk}}}$ is the space of $\theta_{T^{b_{ijk}}}$) is

$$\begin{aligned} K_B((T, \theta_T) \rightarrow (T^{b_{ijk}}, H)) &= Pr(T \rightarrow T^{b_{ijk}}) \times Pr(\theta_{T^{b_{ijk}}} \rightarrow H | T \rightarrow T^{b_{ijk}}) \\ &= \frac{B_{ijk}(T, \theta_T)}{B(T, \theta_T)} \int \mathbb{I}(\theta_{T^{b_{ijk}}} \in H) Pr(\mu_{n_l}) Pr(\mu_{n_r}) d\mu_{n_l} d\mu_{n_r} \end{aligned}$$

in which $B(T, \theta_T) = \sum_{ijk} B_{ijk}(T, \theta_T)$ and $Pr(\cdot)$ is a proposal distribution for μ 's.

Similarly, given a specific death occurs, the probability that the following jump leads to a point in $F \subset \Omega_{T^{d_i}}$ (where $\Omega_{T^{d_i}}$ is the space of $\theta_{T^{d_i}}$) is

$$\begin{aligned} K_D((T, \theta_T) \rightarrow (T^{d_i}, F)) &= Pr(T \rightarrow T^{d_i}) \times Pr(\theta_{T^{d_i}} \rightarrow F | T \rightarrow T^{d_i}) \\ &= \frac{D_i(T, \theta_T)}{D(T, \theta_T)} \int \mathbb{I}(\theta_{T^{d_i}} \in F) Pr(\mu_i) d\mu_i \end{aligned} \quad (17)$$

in which $D(T, \theta_T) = \sum_i D_i(T, \theta_T)$ and $Pr(\cdot)$ is a proposal distribution for μ 's.

This birth-death process satisfies the detailed balance conditions if

$$\begin{aligned} &\int_H B(T, \theta_T) Pr(T, \theta_T | \mathcal{D}) d\theta_T = \\ &\sum_{ijk} \int_{\theta_{T^{b_{ijk}}}} D(T^{b_{ijk}}, \theta_{T^{b_{ijk}}}) K_D((T^{b_{ijk}}, \theta_{T^{b_{ijk}}}) \rightarrow (T, F)) Pr(T^{b_{ijk}}, \theta_{T^{b_{ijk}}} | \mathcal{D}) d\theta_{T^{b_{ijk}}}, \end{aligned} \quad (18)$$

and

$$\int_F D(T, \theta_T) \Pr(T, \theta_T \mid \mathcal{D}) d\theta_T = \sum_i \int_{\theta_{T^{d_i}}} B(T^{d_i}, \theta_{T^{d_i}}) K_B((T^{d_i}, \theta_{T^{d_i}}) \rightarrow (T, F)) \Pr(T^{d_i}, \theta_{T^{d_i}} \mid \mathcal{D}) d\theta_{T^{d_i}}.$$

We check the first part of the detailed balance conditions (Equation 18) as follows. For the left hand side (LHS) we have

$$\begin{aligned} LHS &= \int_F B(T, \theta_T) \Pr(T, \theta_T \mid \mathcal{D}) d\theta_T \\ &= \int_{\theta_T} \mathbb{I}(\theta_T \in F) B(T, \theta_T) \Pr(T, \theta_T \mid \mathcal{D}) d\theta_T \\ &= \int_{\theta_T} \mathbb{I}(\theta_T \in F) \sum_{ijk} B_{ijk}(T, \theta_T) \Pr(T, \theta_T \mid \mathcal{D}) d\theta_T \\ &= \sum_{ijk} \int_{\theta_T} \mathbb{I}(\theta_T \in F) B_{ijk}(T, \theta_T) \Pr(T, \theta_T \mid \mathcal{D}) d\theta_T \\ &= \sum_{ijk} \int_{\theta_T} \mathbb{I}(\theta_T \in F) B_{ijk}(T, \theta_T) \Pr(T, \theta_T \mid \mathcal{D}) \left[\int_{(\mu_l^n, \mu_r^n)} Pr(\mu_l^n) Pr(\mu_r^n) d\mu_l^n d\mu_r^n \right] d\theta_T \\ &\quad [Pr(.) \text{ must integrate to } 1] \\ &= \sum_{ijk} \int_{\theta_T} \int_{(\mu_l^n, \mu_r^n)} \mathbb{I}(\theta_T \in F) B_{ijk}(T, \theta_T) \Pr(T, \theta_T \mid \mathcal{D}) Pr(\mu_l^n) Pr(\mu_r^n) d\mu_l^n d\mu_r^n d\theta_T. \end{aligned}$$

Furthermore, for the right hand side (RHS) of Equation 18, by using Equation 17 we have

$$\begin{aligned} RHS &= \sum_{ijk} \int_{\theta_{T^{b_{ijk}}}} D(T^{b_{ijk}}, \theta_{T^{b_{ijk}}}) K_D((T^{b_{ijk}}, \theta_{T^{b_{ijk}}}); (T, F)) \Pr(T^{b_{ijk}}, \theta_{T^{b_{ijk}}} \mid \mathcal{D}) d\theta_{T^{b_{ijk}}} \\ &\quad [\text{Equation 17}] \\ &= \sum_{ijk} \int_{\theta_{T^{b_{ijk}}}} D_i(T^{b_{ijk}}, \theta_{T^{b_{ijk}}}) \left[\int_{\mu_i} \mathbb{I}(\theta_T \in F) Pr(\mu_i) d\mu_i \right] \Pr(T^{b_{ijk}}, \theta_{T^{b_{ijk}}} \mid \mathcal{D}) d\theta_{T^{b_{ijk}}} \\ &= \sum_{ijk} \int_{\theta_{T^{b_{ijk}}}} \int_{\mu_i} \mathbb{I}(\theta_T \in F) D_i(T^{b_{ijk}}, \theta_{T^{b_{ijk}}}) Pr(\mu_i) \Pr(T^{b_{ijk}}, \theta_{T^{b_{ijk}}} \mid \mathcal{D}) d\mu_i d\theta_{T^{b_{ijk}}}. \end{aligned}$$

Note that the number of terminal nodes n_t for performing a birth in the original tree T equals the number of ways we can return by deaths n_d .

It follows that we have LHS=RHS provided that

$$B_{ijk}((T, \theta_T) Pr(T, \theta_T \mid \mathcal{D}) Pr(\mu_l^n) Pr(\mu_r^n) = D_i(T^{b_{ijk}}, \theta_{T^{b_{ijk}}}) Pr(\mu_i) \Pr(T^{b_{ijk}}, \theta_{T^{b_{ijk}}} \mid \mathcal{D}).$$

Appendix B. Extending of CT-MCMC algorithm to rotate mechanism

Here we consider extending the CT-MCMC algorithm to include the rotate mechanism. Following the construction of Preston (1977), let the state space be $\Omega = \cup_{n=0}^{\infty} \Omega_n$ where Ω_n is made up of all states of cardinality n and are disjoint. Further, let $\Omega^{b(n)}$ be the states from which a birth into Ω_n originates, let $\Omega^{d(n)}$ be the states from which a death into Ω_n originates and let $\Omega^{r(n)}$ be the states from which a rotate into Ω_n originates where $\Omega^{b(n)}, \Omega^{d(n)}, \Omega^{r(n)}$ are disjoint; that is $\Omega^{b(n)} \equiv \Omega_{n-1}$, $\Omega^{d(n)} \equiv \Omega_{n+1}$ and $\Omega^{r(n)} \subset \Omega \setminus (\Omega^{b(n)} \cup \Omega^{d(n)})$.

Let $\mathcal{F}_n$ be the σ -field of subsets of Ω_n and let $\mathcal{F}$ be the σ -field on Ω generated by the $\mathcal{F}_n$. We consider a jump process that can jump from state $x \in \Omega_n$ to a point in one of $\Omega^{b(n)}, \Omega^{d(n)}, \Omega^{r(n)}$. Let μ denote a measure on $(\Omega, \mathcal{F})$ and μ_n denote μ restricted to Ω_n . Let $B, D, R : \Omega \rightarrow \mathbb{R}^+$ be $\mathcal{F}$ -measurable with $D(x) = R(x) = 0$ for $x \in \Omega_0$ and let $\alpha = B + D + R$. For $n \geq 1$ we define the transition probability kernels

$$K_B^{(n)} : \Omega_n \times \mathcal{F}_{b(n)} \rightarrow \mathbb{R}^+,$$

$$K_D^{(n)} : \Omega_n \times \mathcal{F}_{d(n)} \rightarrow \mathbb{R}^+,$$

and

$$K_R^{(n)} : \Omega_n \times \mathcal{F}_{r(n)} \rightarrow \mathbb{R}^+.$$

Then the overall transition kernel is given by (Preston, 1977)

$$K(x, F) = \frac{B(x)}{\alpha(x)} K_B^{(n)}(x, F_{b(n)}) + \frac{D(x)}{\alpha(x)} K_D^{(n)}(x, F_{d(n)}) + \frac{R(x)}{\alpha(x)} K_R^{(n)}(x, F_{r(n)})$$

for $x \in \Omega_n, n \geq 1$ and let $\frac{B(x)}{\alpha(x)} = \frac{D(x)}{\alpha(x)} = \frac{R(x)}{\alpha(x)} = \frac{1}{2}$ if $\alpha(x) = 0$, and

$$K(x, F) = K_B^{(0)}(x, F_{b(0)})$$

if $x \in \Omega_0$.

A *rotate event* goes to state $(T^{r_{ij}}, \theta_{T^{r_{ij}}})$ with *rotate rate* $R_{ij}(T, \theta_T)$ where $i \in 1, \dots, n_r$ and n_r is the number of possible rotatable nodes (see Pratola, 2016, for details) and $j \in 1, \dots, n_j$ is the number of possible outcomes from a rotate at the i 'th rotatable node. Furthermore we define $R(T, \theta_T) = \sum_{i=1}^{n_r} \sum_{j=1}^{n_i} R_{ij}(T, \theta_T)$. Hence, a rotate event changes the topology τ by rearranging internal nodes according to the rules described in Pratola (2016).

In total, we consider the overall number of topological changes to the tree to occur via birth and death moves (as defined earlier) and rotate moves which occur with respective rates $B_{ijk}(T, \theta_T)$, $D_i(T, \theta_T)$ and $R_{ij}(T, \theta_T)$ given the tree is in state (T, θ_T) . With rotate, we do not know how many of the j possible outcomes of a rotate at node i will increase the dimension of θ_T thereby creating a new μ parameter. So, to make things easier—and since this is what we do in practice—we integrate out all of these parameters and work directly with the marginal likelihood. In this case, the birth/death transition kernels from above become:

$$K_B(T \rightarrow T^{b_{ijk}}) = \frac{B_{ijk}(T)}{B(T)},$$

$$K_D(T \rightarrow T^{d_i}) = \frac{D_i(T)}{D(T)},$$

and

$$K_R(T \rightarrow T^{r_{ij}}) = \frac{R_{ij}(T)}{R(T)}.$$

One of the things we need is that birth is inverse of death, death is inverse of birth and rotate is inverse of rotate. This means that in this case our detailed balance condition will consist of 3 equations, essentially the birth/death balances from earlier as well as a rotate balance condition

$$B(T) \Pr(T \mid \mathcal{D}) = \sum_{ijk} D(T^{b_{ijk}}) K_D((T^{b_{ijk}}) \rightarrow T) \Pr(T^{b_{ijk}} \mid \mathcal{D}),$$

$$D(T) \Pr(T \mid \mathcal{D}) = \sum_i B(T^{d_i}) K_B((T^{d_i}) \rightarrow T) \Pr(T^{d_i} \mid \mathcal{D}),$$

and

$$R(T) \Pr(T \mid \mathcal{D}) = \sum_{ij} R(T^{r_{ij}}) K_R((T^{r_{ij}}) \rightarrow T) \Pr(T^{r_{ij}} \mid \mathcal{D}),$$

where T_{ij}^r is the tree state generated from previously choosing the j 'th rotate generated at rotatable node i and $Pr(T \mid \mathcal{D}) = \int_{\theta^T} Pr(T, \theta_T \mid \mathcal{D})$ is the marginal posterior.

For the rotate balance, we have

$$\begin{aligned} R(T) \Pr(T \mid \mathcal{D}) &= \sum_{ij} R(T^{r_{ij}}) K_R(T^{r_{ij}} \rightarrow T) \Pr(T^{r_{ij}} \mid \mathcal{D}) \\ \sum_{ij} R_{ij}(T) \Pr(T \mid \mathcal{D}) &= \sum_{ij} R_{ij}(T^{r_{ij}}) P(T^{r_{ij}} \mid \mathcal{D}) \end{aligned}$$

which is satisfied if

$$R_{ij}(T) P(T \mid \mathcal{D}) = R_{ij}(T^{r_{ij}}) P(T^{r_{ij}} \mid \mathcal{D}).$$

Thus, the corresponding rate for the rotate move is

$$R_{ij}(T) = \min \left\{ 1, \frac{\Pr(T^{r_{ij}} \mid \mathcal{D})}{Pr(T \mid \mathcal{D})} \right\}$$

and similarly working with the integrated posterior, the corresponding rates for the birth/death moves become

$$B_{ijk}(T) = \min \left\{ 1, \frac{\Pr(T^{b_{ijk}} \mid \mathcal{D})}{Pr(T \mid \mathcal{D})} \right\}$$

and

$$D_i(T) = \min \left\{ 1, \frac{\Pr(T^{d_i} \mid \mathcal{D})}{Pr(T \mid \mathcal{D})} \right\}.$$

Given this construction, the probability of birth, death and rotate moves occur with probabilities given by

$$\Pr(\text{birth at node } \eta_i \text{ for variable } \nu_j \text{ and cut-point } c_k) = \frac{B_{ijk}(T)}{B(T) + D(T) + R(T)},$$

$$\Pr(\text{death at node } \eta_i) = \frac{D_i(T)}{B(T) + D(T) + R(T)}$$

and

$$\Pr(\text{rotate } j \text{ at node } \eta_i) = \frac{R_{ij}(T)}{B(T) + D(T) + R(T)}.$$

Note that in practice this approach is too expensive because we have to calculate $B(T) + D(T) + R(T)$ at each iteration. To address this problem we split this move into two moves: a birth/death part and a rotate part can be performed separately to reduce computational burden. To do so, we introduce parameter α . The idea is that with probability $\alpha \in [0, 1]$ we perform a birth/death move via CT-MCMC, and with probability $1 - \alpha$ we perform a rotate move via CT-MCMC. That is, our move corresponds to the mixture distribution

$$\alpha \left[\frac{B_{ijk}(T)}{B(T) + D(T)} + \frac{D_i(T)}{B(T) + D(T)} \right] + (1 - \alpha) \frac{R_{ij}(T)}{R(T)}$$

for some fixed, known α . Note that if

$$\alpha = \frac{B(T) + D(T)}{B(T) + D(T) + R(T)}$$

then this mixture distribution corresponds exactly to the distribution for the full CT-MCMC algorithm.

Appendix C. Additional simulation results

Here we present a number of additional simulation results for the simulation scenario in the Section 4 and described in the main text for $\sigma^2 \in (0.1, 0.01)$. Tables 2 and 3 demonstrate that also in these cases our proposed CT-MCMC method performs well.

Method	MSE	Effective Sample Size				Activity		
		σ^2	x_1	x_2	x_3	x_1	x_2	x_3
RJ-A	0.01	19735	1279	2275	996	0.29	0.45	0.26
RJ-B	0.01	19707	1203	3021	1818	0.23	0.47	0.30
RJ-C	0.01	19660	13247	2141	13255	0.25	0.5	0.25
CT-A	0.01	24759	14276	39968	14333	0.28	0.44	0.28
CT-B	0.01	19703	36275	75526	39250	0.25	0.48	0.27
CT-C	0.01	29061	14470	55343	14467	0.26	0.47	0.26

Method	Unique Trees	Effective Sample Size per Second			
		σ^2	x_1	x_2	x_3
RJ-A	1.67	26669	1729	3074	1346
RJ-B	2.68	21656	1322	3319	1997
RJ-C	7.98	11702	7885	1274	7890
CT-A	10.24	6897	3977	11133	3993
CT-B	3.00	5582	10276	21395	11119
CT-C	10.0	6590	3281	12550	3280

Table 2: Overview of the performance measures of different sampling methods for simulation example for the case $\sigma^2 = 0.1$ in Equation 16 . The table reports the average over 100 replications of the prediction error, the sampling efficiency, the exploration behavior in terms of variable activity measured as the average proportion of internal rules involving each variable, the exploration behavior in terms of the average number of unique trees visited, and computational efficiency in effective samples per second.

Method	MSE	Effective Sample Size				Activity		
		σ^2	x_1	x_2	x_3	x_1	x_2	x_3
RJ-A	1.02E-4	19739	1410	3029	1620	0.25	0.44	0.31
RJ-B	1.02E-4	19742	2687	5290	2603	0.29	0.49	0.22
RJ-C	1.02E-4	19719	13315	3602	13319	0.25	0.5	0.25
CT-A	1.03E-4	15230	14095	21063	14131	0.28	0.43	0.28
CT-B	1.02E-4	19713	34644	75182	40538	0.24	0.48	0.28
CT-C	1.04E-4	19254	14308	43675	14283	0.27	0.46	0.27

Method	Unique Trees	Effective Sample Size per Second			
		σ^2	x_1	x_2	x_3
RJ-A	1.61	26675	1905	4094	2189
RJ-B	2.80	22434	3053	6011	2958
RJ-C	7.71	11738	7926	2144	7928
CT-A	15.8	4290	3971	5933	3981
CT-B	3.00	5585	9814	21298	11484
CT-C	10.0	4269	3173	9684	3167

Table 3: Overview of the performance measures of different sampling methods for simulation example for the case $\sigma^2 = 0.01$ in Equation 16 . The table reports the average over 100 replications of the prediction error, the sampling efficiency, the exploration behavior in terms of variable activity measured as the average proportion of internal rules involving each variable, the exploration behavior in terms of the average number of unique trees visited, and computational efficiency in effective samples per second.

References

- Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory*, pages 39–1, 2012.
- Timothy C Au. Random forests, decision trees, and categorical predictors: the absent levels problem. *The Journal of Machine Learning Research*, 19(1):1737–1766, 2018.
- GAŠrard Biau. Analysis of a random forests model. *Journal of Machine Learning Research*, 13(Apr):1063–1095, 2012.
- GAŠrard Biau, Luc Devroye, and GÃAŁbor Lugosi. Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, 9(Sep):2015–2033, 2008.
- Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. Classification and regression trees. 1984.
- Olivier Cappé, Christian P Robert, and Tobias Rydén. Reversible jump, birth-and-death and more general continuous time markov chain monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(3):679–700, 2003.
- Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayesian cart model search. *Journal of the American Statistical Association*, 93(443):935–948, 1998.
- Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayesian treed models. *Machine Learning*, 48(1):299–320, 2002.
- Hugh A Chipman, Edward I George, Robert E McCulloch, et al. Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.
- David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 1996.
- Glenn De’ath and Katharina E Fabricius. Classification and regression trees: a powerful yet simple technique for ecological data analysis. *Ecology*, 81(11):3178–3192, 2000.
- David GT Denison, Bani K Mallick, and Adrian FM Smith. A bayesian cart algorithm. *Biometrika*, 85(2):363–377, 1998.
- Adrian Dobra and Reza Mohammadi. Loglinear model selection and human mobility. *The Annals of Applied Statistics*, 12(2):815–845, 2018.
- Dean Eckles and Maurits Kaptein. Thompson sampling with the online bootstrap. *arXiv preprint arXiv:1410.4009*, 2014.
- Dean Eckles and Maurits Kaptein. Bootstrap thompson sampling and sequential decision problems in the behavioral sciences. *Sage Open*, 9(2):1–12, 2019.
- John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

- Robert B Gramacy and Herbert K H Lee. Bayesian treed gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483):1119–1130, 2008.
- Peter J Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732, 1995.
- Max Hinnie, Alex Lenkoski, Tom Heskes, and Marcel van Gerven. Efficient sampling of gaussian graphical models using conditional bayes factors. *Stat*, 3(1):326–336, 2014.
- Balaji Lakshminarayanan, Daniel Roy, and Yee Whye Teh. Top-down particle filtering for bayesian decision trees. In *International Conference on Machine Learning*, pages 280–288, 2013.
- Antonio R Linero. Bayesian regression trees for high-dimensional prediction and variable selection. *Journal of the American Statistical Association*, 113(522):626–636, 2018.
- Brent R Logan, Rodney Sparapani, Robert E McCulloch, and Purushottam W Laud. Decision making and uncertainty quantification for individualized treatments using bayesian additive regression trees. *Statistical methods in medical research*, 28(4):1079–1093, 2019.
- Abdolreza Mohammadi and Ernst C Wit. Bayesian structure learning in sparse Gaussian graphical models. *Bayesian Analysis*, 10(1):109–138, 2015.
- Abdolreza Mohammadi, Mohammadi-Reza Salehi-Rad, and Ernst C Wit. Using mixture of Gamma distributions for Bayesian analysis in an M/G/1 queue with optional second service. *Computational Statistics*, 28(2):683–700, 2013.
- Abdolreza Mohammadi, Fentaw Abegaz, Edwin van den Heuvel, and Ernst C. Wit. Bayesian modeling of Dupuytren disease using Gaussian copula graphical models. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 66(3):629–645, 2017a.
- Reza Mohammadi and Ernst C Wit. BDgraph: An R package for Bayesian structure learning in graphical models. *Journal of Statistical Software*, 89(3):1–30, 2019.
- Reza Mohammadi, Helene Massam, and Gerard Letac. The ratio of normalizing constants for bayesian graphical gaussian model selection. *arXiv preprint arXiv:1706.04416*, 110:116, 2017b.
- Anantha M Prasad, Louis R Iverson, and Andy Liaw. Newer classification and regression tree techniques: bagging and random forests for ecological prediction. *Ecosystems*, 9(2):181–199, 2006.
- Matthew T Pratola. Efficient metropolis–hastings proposal mechanisms for bayesian regression tree models. *Bayesian analysis*, 11(3):885–911, 2016.
- Matthew T Pratola, Hugh A Chipman, James R Gattiker, David M Higdon, Robert McCulloch, and William N Rust. Parallel bayesian additive regression trees. *Journal of Computational and Graphical Statistics*, 23(3):830–852, 2014.

- Chris Preston. Spatial birth-and-death processes. *Bulletin of the International Statistical Institute*, 46:371–391, 1977.
- Philipp Probst and Anne-Laure Boulesteix. To tune or not to tune the number of trees in random forest. *Journal of Machine Learning Research*, 18:181–1, 2017.
- Herbert Robbins. Some aspects of the sequential design of experiments. In *Herbert Robbins Selected Papers*, pages 169–177. Springer, 1985.
- Christian Robert. *The Bayesian choice: from decision-theoretic foundations to computational implementation*. Springer Science & Business Media, 2007.
- Daniel D Sleator, Robert E Tarjan, and William P Thurston. Rotation distance, triangulations, and hyperbolic geometry. *Journal of the American Mathematical Society*, 1(3): 647–681, 1988.
- Matthew Stephens. Bayesian analysis of mixture models with an unknown number of components-an alternative to reversible jump methods. *Annals of statistics*, pages 40–74, 2000.
- Matthew A Taddy, Robert B Gramacy, and Nicholas G Polson. Dynamic trees for learning and design. *Journal of the American Statistical Association*, 106(493):109–123, 2011.
- William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.
- Nanwei Wang, Laurent Briollais, and Helene Massam. The scalable birth-death mcmc algorithm for mixed graphical model learning with application to genomic data integration. *arXiv preprint arXiv:2005.04139*, 2020.
- Yuhong Wu, Håkon Tjelmeland, and Mike West. Bayesian cart: Prior specification and posterior simulation. *Journal of Computational and Graphical Statistics*, 16(1):44–66, 2007.