Statistics and Computing (2021) 31:20
https://doi.org/10.1007/s11222-021-09997-3

Bayesian additive regression trees with model trees

Estevão B. Prado[1,2] · Rafael A. Moral[1] · Andrew C. Parnell[1,2]

Received: 9 June 2020 / Accepted: 11 January 2021 / Published online: 3 March 2021

Abstract
Bayesian additive regression trees (BART) is a tree-based machine learning method that has been successfully applied to regression and classification problems. BART assumes regularisation priors on a set of trees that work as weak learners and is very flexible for predicting in the presence of nonlinearity and high-order interactions. In this paper, we introduce an extension of BART, called model trees BART (MOTR-BART), that considers piecewise linear functions at node levels instead of piecewise constants. In MOTR-BART, rather than having a unique value at node level for the prediction, a linear predictor is estimated considering the covariates that have been used as the split variables in the corresponding tree. In our approach, local linearities are captured more efficiently and fewer trees are required to achieve equal or better performance than BART. Via simulation studies and real data applications, we compare MOTR-BART to its main competitors. R code for MOTR-BART implementation is available at https://github.com/ebprado/MOTR-BART.

Keywords Bayesian Trees · Linear models · Machine learning · Bayesian nonparametric regression

1 Introduction

Bayesian additive regression trees (BART) is a statistical method proposed by Chipman et al (2010) that has become popular in recent years due to its competitive performance on regression and classification problems, when compared to other supervised machine learning methods, such as random forests (RF) (Breiman 2001) and gradient boosting (GB) (Friedman 2001). BART differs from other tree-based methods as it controls the structure of each tree via a prior distribution and generates the predictions via an MCMC backfitting algorithm that is responsible for accepting and rejecting the proposed trees along the iterations. In practice, BART can be used for predicting a continuous/binary response variable through R packages, such as `dbarts` (Chipman et al 2010), `BART` (McCulloch et al 2019) and `bartMachine` (Kapelner and Bleich 2016).

In essence, BART is a nonparametric Bayesian algorithm that generates a set of trees by choosing the covariates and the split points at random. To generate the predicted values for each terminal node, the normal distribution is adopted as the likelihood function as well as prior distributions are placed on the trees, predicted values and variance of the predictions. Through a backfitting MCMC algorithm, the predictions from each tree are obtained by combining Gibbs Sampler and Metropolis–Hastings steps. The final prediction is then calculated as the sum of the predicted values over all trees. In parallel, samples from the posterior distributions of the quantities of interest are naturally generated along the MCMC iterations.

In this paper, we introduce the algorithm MOTR-BART, which combines model trees (Quinlan 1992) with BART to deal with local linearity at node levels. In MOTR-BART, rather than estimating a constant as the predicted value as BART does, for each terminal node a linear predictor is estimated, including only the covariates that have been used as a split in the corresponding tree. With this approach, we aim to capture linear associations between the response and covariates and then improve the final prediction. We observe that MOTR-BART requires fewer trees to achieve equal or better performance than BART as well as reaches faster convergence, when we look at the overall log-likelihood. Through simulation experiments that consider different number of observations and covariates, MOTR-BART outperforms its main competitors in terms of RMSE on out-of-sample data, even using fewer trees. In the real

✉ Estevão B. Prado
estevao.prado@mu.ie

1 Hamilton Institute and Department of Mathematics and Statistics, Maynooth University, Maynooth, Ireland

2 Insight Centre for Data Analytics, Maynooth University, Maynooth, Ireland

data applications, MOTR-BART is competitive compared to BART and other tree-based methods.

This paper is organised as follows. In Sect. 2, we briefly introduce BART, some related works and model trees. Section 3 presents the mathematical details of BART and how it may be implemented in the regression context. In Sect. 4, we introduce the MOTR-BART, providing the mathematical expressions needed for regression and classification. Sect. 5 shows comparisons between MOTR-BART and other algorithms via simulated scenarios and real data applications. Finally, in Sect. 6, we conclude with a discussion.

2 Tree-based methods

2.1 Related works

BART considers that a univariate response variable can be approximated by a sum of predicted values from a set of trees as $\hat{y} = \sum_{t=1}^{m} g(\mathbf{X}; M_t, T_t)$, where $g(\cdot)$ is a function that assigns a predicted value based on $\mathbf{X}$ and T_t, $\mathbf{X}$ is the design matrix, M_t the set of predicted values of the tree t and T_t represents the structure of the tree t. In BART, a tree T_t can be modified using four moves (growing, pruning, changing or swapping), and the splitting rules that create the terminal/internal nodes are randomly chosen. To sample from the full conditional distribution of T_t, the Metropolis–Hastings algorithm is used. Further, each component $\mu_{t\ell} \in M_t$ is sampled from its full conditional via a Gibbs sampler step. Then, the final prediction is calculated by adding up the values of $\mu_{t\ell}$ from all the m trees. Further details are given in Sect. 3.

BART's versatility has made it an attractive option with applications in credit risk modelling (Zhang and Härdle 2010), identification of subgroup effects in clinical trials (Sivaganesan et al 2017; Schnell et al 2016), competing risk analysis (Sparapani et al 2019), survival analysis of stem cell transplantation (Sparapani et al 2016), proteomic biomarker discovery (Hernández et al 2015) and causal inference (Hill 2011; Green and Kern 2012; Hahn et al 2020). In this context, many extensions have been proposed, such as BART for estimating monotone and smooth surfaces (Starling et al 2019, 2020; Linero and Yang 2018), categorical and multinomial data (Murray 2017; Kindo et al 2016b), high-dimensional data (Hernández et al 2018; He et al 2019; Linero 2018), zero-inflated and semi-continuous responses (Linero et al 2018), heterocedastic data (Pratola et al 2017) and BART with quantile regression and varying coefficient models (Kindo et al 2016a; Deshpande et al 2020). Recently, some papers have developed theoretical aspects related to BART (Linero 2017b; Ročková and van der Pas 2017; Ročková and Saha 2018; Linero and Yang 2018).

Some of the works mentioned above are somewhat related to MOTR-BART. For instance, Linero and Yang (2018) introduce the soft BART in order to provide an approach suitable for both estimating a target smooth function and dealing with sparsity. In soft BART, the observations are not allocated deterministically to the terminal nodes, as it is commonly done in the conventional trees. Instead, the observations are assigned to the terminal nodes based on a probability measure, which is a function of a bandwidth parameter and of the distance between the values of the covariates and the cut-offs defined by the splitting rules. Through empirical and theoretical results, they show that soft BART is capable of smoothly approximating linear and nonlinear functions as well as that its posterior distribution concentrates, under mild conditions, at the minimax rate. The main differences between MOTR-BART and soft BART are: (i) MOTR-BART does not use the idea of soft trees, where the observations are assigned to the terminal based on a probability measure, and (ii) MOTR-BART uses a linear predictor rather than a piecewise constant to generate the predictions at node level.

In this sense, Starling et al (2020) propose the BART with targeted smoothing (tsBART) by introducing smoothness over a covariate of interest. In their approach, rather than predicting a piecewise constant as the standard BART, univariate smooth functions of a certain covariate of interest are used to generate the node-level predictions. In tsBART, they place a Gaussian process prior over the smooth function associated with each terminal node and grow the trees using all available covariates, apart from the one over which they wish to introduce the smoothness. Although tsBART and MOTR-BART have some similarities, since both do not base their predictions on piecewise constants and both aim to provide more flexibility at the node-level predictions, they differ as MOTR-BART allows for more than one covariate to be used in the linear predictors and do not assume a Gaussian process prior on each linear predictor.

In addition, Deshpande et al (2020) propose an extension named VCBART that combines varying coefficient models and BART. In their approach, rather than approximating the response variable itself, each covariate effect in the linear predictor is estimated by using BART. They also provide theoretical results about the near minimax optimal rate associated with the posterior concentration of the VCBART considering non-i.i.d errors. Although the linear model is a particular case of the varying coefficients model, MOTR-BART and VCBART are structurally different. For instance, VCBART considers that a univariate response variable can be approximated via an overall linear predictor in which the coefficients are estimated via BART. In contrast, MOTR-BART approximates the response by estimating a linear predictor for each terminal node in each tree, where normal priors are placed on the coefficients in order to estimate them.

Regarding non-Bayesian methods, we highlight the algorithms introduced by Friedberg et al (2018) and Künzel et al

(2019), named local linear forests (LLF) and linear random forests (LRF), respectively. In their work, RF-based algorithms are proposed, where the predictions for each terminal node are generated from a local ridge regression. Furthermore, the LLF algorithm also provides a pointwise confidence interval based on the RF delta method proposed by Athey et al (2019) and theoretical results related to asymptotic consistency and rates of convergence of the forest.

2.2 Model trees

Quinlan (1992) introduced the term model trees when proposing the M5 algorithm, which is a tree-based method that estimates a linear equation for each terminal node and then computes the final prediction based on piecewise linear models and a smoothing process. Initially introduced in the context of regression, extensions and generalisations for classification were presented by Wang et al (1997) and Landwehr et al (2005).

Unlike BART, RF and GB, where multiple trees are generated to predict the outcome, the algorithm M5 generates only one tree. For the growing process, the variance reduction (VR) is adopted as the splitting criterion. When estimating the coefficients for the linear equation at a terminal node, the covariates are selected based on tests, and depending on their significance, the linear equation can be reduced to a constant, if all covariates do not show any significance. At the end, the prediction is calculated based on the linear predictors from all terminal nodes and then is averaged over the predictions from the terminal nodes along the path to the root.

3 BART

Introduced by Chipman et al (2010), BART is a tree-based machine learning method that considers that a univariate response variable $\mathbf{y} = (y_1, \dots, y_n)^\top$ can be approximated by a sum of trees as

$$y_i = \sum_{t=1}^{m} g(\mathbf{x}_i; T_t, M_t) + \epsilon_i, \ \epsilon_i \sim \mathrm{N}(0, \sigma^2),$$

where $g(\mathbf{x}_i; T_t, M_t)$ is a function that assigns a predicted value $\mu_{t\ell}$ based on $\mathbf{x}_i$, $\mathbf{x}_i = (x_{i1}, \dots, x_{id})$ represents the ith row of the design matrix $\mathbf{X}$, T_t is the set of splitting rules that defines the tth tree and $M_t = (\mu_{t1}, \dots, \mu_{tb_t})$ is the set of predicted values for all nodes in the tree t, with μ_{tb_t} representing the predicted value for the terminal node b_t. The splitting rules that define the terminal nodes for the tree t can be defined as partitions $\mathcal{P}_{t\ell}$, with $\ell = 1, \dots, b_t$, and $g(\mathbf{x}_i; T_t, M_t) = \mu_{t\ell}$ for all observations $i \in \mathcal{P}_{t\ell}$, based on the values of $\mathbf{x}_i$.

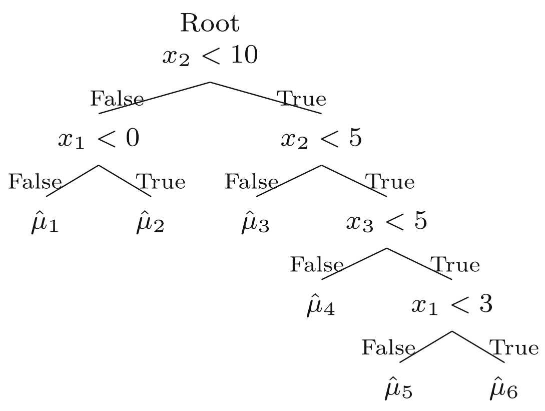

Fig. 1 An example of a single tree generated by BART. In practice, BART generates multiple trees for which the predictions are added together. The covariates and split points that define the terminal nodes are proposed uniformly and optimised via an MCMC algorithm. The quantities x_1, x_2 and x_3 represent covariates; $\hat{\mu}_\ell$ is the predicted value of node ℓ

In BART, each regression tree is generated as in Chipman et al (1998) (see Fig. 1) where, through a backfitting algorithm, a binary tree can be created or modified by four movements: grow, prune, change or swap. A new tree is created by one of these four movements and compared to the previous version via a Metropolis–Hastings step on the partial residuals. In the growing process, a terminal node is randomly selected and is separated into two new nodes. Here, the covariate that is used to create the new terminal nodes is picked uniformly as is its associated split point. In other words, the splitting rule is fully defined assuming the uniform distribution over both the set of covariates and the set of their split points. During a prune step, a parent of two terminal nodes is randomly chosen and then its child nodes are removed. In the change movement, a pair of terminal nodes is picked at random and its splitting rule is changed. In the swap process, two parents of terminal nodes are randomly selected and their splitting rules are exchanged.

In order to control the depth of the tree, a regularisation prior is considered as

$$p(T_t) = \prod_{\ell \in L_I} \left[\alpha(1 + d_{t\ell})^{-\beta}\right] \times \prod_{\ell \in L_T} \left[1 - \alpha(1 + d_{t\ell})^{-\beta}\right], \tag{1}$$

where L_I and L_T denote the sets of indices of the internal and terminal nodes, respectively, $d_{t\ell}$ is the depth of node ℓ in tree t, $\alpha \in (0, 1)$ and $\beta \geq 0$. Chipman et al (2010) recommend $\alpha = 0.95$ and $\beta = 2$. In essence, $\alpha(1 + d_{t\ell})^{-\beta}$ computes the probability of the node ℓ being internal at depth $d_{t\ell}$.

To estimate the terminal node parameters, $\mu_{t\ell}$, and overall variance, σ^2, conjugate priors are used:

$$\mu_{t\ell}|T_t \sim \mathrm{N}(0, \sigma_\mu^2),$$
$$\sigma^2 \sim \mathrm{IG}(\nu/2, \nu\lambda/2),$$

where $\sigma_\mu = 0.5/(c\sqrt{m})$, $1 \leq c \leq 3$, IG denotes the inverse gamma distribution and m is the number of trees. The division by m has the effect of reducing the predictive power of each tree and forcing each to be a weak learner. The joint posterior distribution of the trees and predicted values is given by

$$\begin{aligned} p((T, M), \sigma^2|\mathbf{y}, \mathbf{X}) &\propto p(\mathbf{y}|\mathbf{X}, T, M, \sigma^2)p(M|T)p(T)p(\sigma^2), \\ &\propto \left[\prod_{t=1}^{m}\prod_{\ell=1}^{b_t}\prod_{i:\mathbf{x}_i\in\mathcal{P}_{t\ell}} p(y_i|\mathbf{x}_i, T_t, M_t, \sigma^2)\right] \\ &\times \left[\prod_{t=1}^{m}\prod_{\ell=1}^{b_t} p(\mu_{t\ell}|T_t)p(T_t)\right] p(\sigma^2). \end{aligned}$$

Chipman et al (2010) initially decompose this joint posterior into two full conditionals. The first one generates all $\mu_{t\ell}$ for each tree $t = 1, \ldots, m$, and is given by

$$p(T_t, M_t|T_{(-t)}, M_{(-t)}, \sigma^2, \mathbf{X}, \mathbf{y}), \tag{2}$$

where $T_{(-t)}$ represents the set of all trees without the component t; similarly for $M_{(-t)}$. To sample from (2), Chipman et al (2010) noticed that the dependence of the full conditional of (T_t, M_t) on $T_{(-t)}, M_{(-t)}$ is given by the partial residuals through

$$R_t = \mathbf{y} - \sum_{k\neq t}^{m} g(\mathbf{X}; T_k, M_k).$$

Thus, rather than depending on the other trees and their predicted values, the joint full conditional of (T_t, M_t) may be rewritten as $p(T_t, M_t|R_t, \sigma^2, \mathbf{X})$, with R_t acting like the response variable. This simplification allows us to sample from $p(T_t, M_t|R_t, \sigma^2, \mathbf{X})$ in two steps:

(a) Propose a new tree either growing, pruning, changing or swapping terminal nodes via

$$\begin{aligned} p(T_t|R_t, \sigma^2) &\propto p(T_t)\int p(R_t|M_t, T_t, \sigma^2)p(M_t|T_t)dM_t, \\ &\propto p(T_t)p(R_t|T_t, \sigma^2), \\ &\propto p(T_t)\prod_{\ell=1}^{b_t}\left[\left(\frac{\sigma^2}{\sigma_\mu^2 n_{t\ell} + \sigma^2}\right)^{1/2} \right. \\ &\left. \times \exp\left(\frac{\sigma_\mu^2\left[n_{t\ell}\bar{R}_\ell\right]^2}{2\sigma^2(\sigma_\mu^2 n_{t\ell} + \sigma^2)}\right)\right], \end{aligned}$$

where $\bar{R}_\ell = \sum_{i\in\mathcal{P}_{t\ell}} r_i/n_{t\ell}$, $r_i \in R_t$ and $n_{t\ell}$ is the number of observations that belong to $\mathcal{P}_{t\ell}$. This sampling is carried out through a Metropolis–Hastings step, as the expression does not have a known distributional form;

(b) Generate the predicted values $\mu_{t\ell}$ for all terminal nodes in the corresponding tree. As all $\mu_{t\ell}$ are independent from each other, it is possible to write $p(M_t|T_t, R_t, \sigma^2) = \prod_{\ell=1}^{b_t} p(\mu_{t\ell}|T_t, R_t, \sigma^2)$. Hence,

$$\begin{aligned} p(\mu_{t\ell}|T_t, R_t, \sigma^2) &\propto p(R_t|M_t, T_t, \sigma^2)p(\mu_{t\ell}), \\ &\propto \exp\left(-\frac{1}{2\sigma_*^2}\left(\mu_{t\ell} - \mu_{t\ell}^*\right)^2\right), \end{aligned}$$

which is a

$$\mathrm{N}\left(\frac{\sigma^{-2}\sum_{i\in\mathcal{P}_{t\ell}} r_i}{n_{t\ell}/\sigma^2 + \sigma_\mu^{-2}}, \frac{1}{n_{t\ell}/\sigma^2 + \sigma_\mu^{-2}}\right).$$

Then, after generating all predicted values for all trees, σ^2 can be updated based on

$$\begin{aligned} p(\sigma^2|T, M, \mathbf{X}, \mathbf{y}) &\propto p(\mathbf{y}|\mathbf{X}, T, M, \sigma^2)p(\sigma^2) \\ &\propto (\sigma^2)^{-(\frac{n+\nu}{2}+1)}\exp\left(-\frac{S+\nu\lambda}{2\sigma^2}\right), \end{aligned} \tag{3}$$

where $S = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ and $\hat{y}_i = \sum_{t=1}^{m} g(\mathbf{x}_i; T_t, M_t)$. The expression in (3) is an $\mathrm{IG}((n+\nu)/2, (S+\nu\lambda)/2)$, and drawing samples from it is straightforward.

In Algorithm 1, we present the full structure of the BART algorithm. Firstly, the response variable and design matrix are required. The trees, hyper-parameters, partial residuals and the number of MCMC iterations have to be initialised. Later, within each MCMC iteration, candidate trees (T_t^*) are sequentially generated, which might be accepted (or rejected) as the current trees with probability $\alpha(T_t, T_t^*)$. After that, the predicted values $\mu_{t\ell}$ are generated for all terminal nodes, and then, the partial residuals are updated. Finally, the final predictions and σ^2 are obtained.

4 Model trees BART

In MOTR-BART, we consider that the response variable is a sum of trees in the form of

$$\mathbf{y} = \sum_{t=1}^{m} g(\mathbf{X}; T_t, B_t) + \epsilon,$$

where B_t is the set of parameters of all linear predictors of the tree t. In terms of partial residuals, MOTR-BART can be represented as

$$r_i|\mathbf{x}_i, \boldsymbol{\beta}_{t\ell}, \sigma^2 \sim \mathrm{N}(\mathbf{x}_i\boldsymbol{\beta}_{t\ell}, \sigma^2),$$

Algorithm 1: BART Algorithm

```
Result: A posterior distribution of trees $T$
Data: $\mathbf{y}$ (response variable) and $\mathbf{X}$ (design matrix);
Initialise: $T = (T_1, \ldots, T_m)$ to stumps, $\{\mu_{t\ell}\} = 0$, $R_1^{(1)} = \mathbf{y}$, $\alpha$, $\beta$, $\sigma_\mu^2$, $\nu$, $\lambda$, $\sigma^2$, number of trees ($m$) and the number of MCMC iterations (burn-in and post-burn-in) ($nIter$).
for $k$ in 1:nIter do
    for $t$ in 1:m do
        Propose a new tree $T_t^*$ by growing, pruning, changing or swapping, where each movement has probability of 0.25 to be chosen;
        Compute $\alpha(T_t, T_t^*) = \min\left\{1, \frac{p(R_t^{(k)}|T_t^*, \sigma^2)p(T_t^*)}{p(R_t^{(k)}|T_t, \sigma^2)p(T_t)}\right\}$;
        Sample $u \sim U[0, 1]$;
        if $u < \alpha(T_t, T_t^*)$ then $T_t = T_t^*$ else $T_t = T_t$;
        for $\ell$ in 1:$b_t$ do
            Update $\mu_{t\ell}$ from $p(\mu_{t\ell}|T_t, R_t, \sigma^2)$;
        end
        Update $R_t^{(k)} = \mathbf{y} - \sum_{j\neq t}^m g(\mathbf{X}; T_j, M_j)$;
    end
    Update $\hat{\mathbf{y}}^{(k)} = \sum_{t=1}^m g(\mathbf{X}, T_t, M_t)$;
    Update $\sigma^2$ sampling from $p(\sigma^2|T, M, \mathbf{X}, \mathbf{y})$.
end
```

where $r_i = y_i - \sum_{j\neq t}^m g(\mathbf{x}_i; T_j, B_j)$, $\boldsymbol{\beta}_{t\ell}$ is the parameter vector associated with the terminal node ℓ of the tree t. In this sense, all observations $i \in \mathcal{P}_{t\ell}$ will have predicted values based on $\boldsymbol{\beta}_{t\ell}$ and the values of their covariates $\mathbf{X}_{t\ell}$. Thus, each observation $i \in \mathcal{P}_{t\ell}$ may have different predicted value.
The priors for $\boldsymbol{\beta}_{t\ell}$ and σ^2 are

$$\boldsymbol{\beta}_{t\ell}|T_t \sim \mathbf{N}_q(\mathbf{0}, \sigma^2\mathbf{V}),$$
$$\sigma^2|T_t \sim \text{IG}(\nu/2, \nu\lambda/2), \tag{4}$$

where $\mathbf{V} = \tau_b^{-1} \times \mathbf{I}_q$ and $q = p_{t\ell} + 1$, with $p_{t\ell}$ representing the number of covariates in the linear predictor of the terminal node ℓ of the tree t. The additional dimension in $\mathbf{V}$ is due to a column filled with 1's in the design matrix $\mathbf{X}_{t\ell}$. Here, the role of the parameter τ_b is to balance the importance of each tree on the final prediction by keeping the components of $\boldsymbol{\beta}_{t\ell}$ close to zero, thus avoiding that one tree contributes more than other. Since the prior on the vector $\boldsymbol{\beta}_{t\ell}$ assumes that all entries have the same variance, we scale the predictors in $\mathbf{X}_{t\ell}$ in order to make this assumption valid. In our simulations and real data applications, we have found that $\tau_b = m$ worked well.

Another possibility is to penalise the intercept and the slopes differently. In this sense, the specification of intercept- and slope-specific variances may be done by setting $\mathbf{V}$ as a $q\times q$ diagonal matrix with $\mathbf{V}_{1,1} = \tau_{\beta_0}^{-1}$ and $\mathbf{V}_{j+1,j+1} = \tau_\beta^{-1}$. In addition, we may assume conjugate priors such as $\tau_{\beta_0} \sim \text{G}(a_0, b_0)$ and $\tau_\beta \sim \text{G}(a_1, b_1)$ to be able to estimate both variances via Gibbs sampling steps. In this case, we would end up with the following full conditionals:

$$\tau_{\beta_0}|- \sim \text{G}\left(\frac{\sum_{t=1}^m b_t}{2} + a_0, \frac{\boldsymbol{\beta}_0^\top\boldsymbol{\beta}_0}{2\sigma^2} + b_0\right),$$

$$\tau_\beta|- \sim \text{G}\left(\frac{\sum_{t=1}^m \sum_{\ell=1}^{b_t} p_{t\ell}}{2} + a_1, \frac{\boldsymbol{\beta}_*^\top\boldsymbol{\beta}_*}{2\sigma^2} + b_1\right),$$

where $\boldsymbol{\beta}_0$ is a vector with the intercepts from all terminal nodes of all trees and $\boldsymbol{\beta}_*$ contains the slopes from all linear predictors of all trees. In our software, we have implemented an option, through the argument `vars_inter_slope = TRUE/FALSE`, that allows the user to either estimate τ_{β_0} and τ_β or use $\tau_b = m$. In Sect. 5, we show the results of MOTR-BART using both approaches.

Hence, the full conditionals are

$$p(\boldsymbol{\beta}_{t\ell}|\mathbf{X}_{t\ell}, R_t, \sigma^2, T_t) \propto p(R_t|\mathbf{X}_{t\ell}, \boldsymbol{\beta}_{t\ell}, \sigma^2, T_t)p(\boldsymbol{\beta}_{t\ell}),$$

which is a

$$\mathbf{N}_q\left(\boldsymbol{\mu}_{t\ell}, \sigma^2\boldsymbol{\Lambda}_{t\ell}\right),$$

where $\boldsymbol{\mu}_{t\ell} = \boldsymbol{\Lambda}_{t\ell}(\mathbf{X}_{t\ell}^\top\mathbf{r}_{t\ell})$, $\boldsymbol{\Lambda}_{t\ell} = (\mathbf{X}_{t\ell}^\top\mathbf{X}_{t\ell} + \mathbf{V}^{-1})^{-1}$ and $\mathbf{X}_{t\ell}$ is an $n_{t\ell} \times q$ matrix with all elements of the design matrix such that $i \in \mathcal{P}_{t\ell}$. The full conditional of σ^2 is similar to the expression in (3), but with $\hat{y}_i = \sum_{t=1}^m g(\mathbf{x}_i; T_t, B_t)$. Finally, the full conditional for T_t is given by

$$p(T_t|\mathbf{X}, R_t, \sigma^2) \propto p(T_t)\int p(R_t|\mathbf{X}, B_t, \sigma^2, T_t)p(B_t)dB_t,$$
$$\propto p(T_t)p(R_t|\mathbf{X}, \sigma^2, T_t),$$

where

$$p(R_t|\mathbf{X}, \sigma^2, T_t) = (\sigma^2)^{-n/2}\prod_{\ell=1}^{b_t}\Bigg[|\mathbf{V}|^{-1/2}|\boldsymbol{\Lambda}_{t\ell}|^{1/2} \times$$
$$\times \exp\left(-\frac{1}{2\sigma^2}\left[-\boldsymbol{\mu}_{t\ell}^\top\boldsymbol{\Lambda}_{t\ell}^{-1}\boldsymbol{\mu}_{t\ell} + \mathbf{r}_{t\ell}^\top\mathbf{r}_{t\ell}\right]\right)\Bigg].$$

The main difference between BART and MOTR-BART is shown in Fig. 2. Now, rather than having a constant as the predicted value for each terminal node, the prediction will be obtained from a linear predictor at node level. The purpose of introducing a linear predictor is to try to capture local linearity, reduce the number of trees and then possibly improve the prediction at node level.

The key point in MOTR-BART is which covariates should be considered in the linear predictor of each terminal node. At first glance, one might think that it would be advantageous to use variable selection techniques for regression models, such as ridge regression, lasso (Tibshirani 1996) or horseshoe (Carvalho et al 2010). Under the Bayesian perspective, these methods assume different priors on the regression coefficient

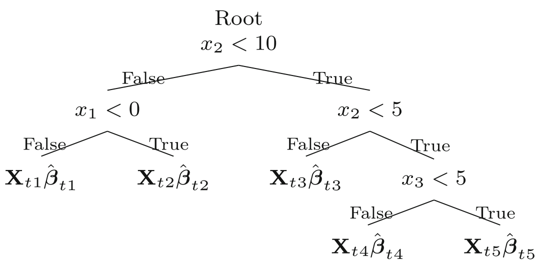
Root
$x_2 < 10$
False
True
$x_1 < 0$
$x_2 < 5$
False
True
False
True
$\mathbf{X}_{t1}\hat{\boldsymbol\beta}_{t1}$
$\mathbf{X}_{t2}\hat{\boldsymbol\beta}_{t2}$
$\mathbf{X}_{t3}\hat{\boldsymbol\beta}_{t3}$
$x_3 < 5$
False
True
$\mathbf{X}_{t4}\hat{\boldsymbol\beta}_{t4}$
$\mathbf{X}_{t5}\hat{\boldsymbol\beta}_{t5}$

Fig. 2 An example of a tree generated based on MOTR-BART. The quantities x_1, x_2 and x_3 represent covariates; $\mathbf{X}_{t\ell}$ is a subset of the design matrix $\mathbf{X}$ such that $i \in \mathcal{P}_{t\ell}$; and $\hat{\boldsymbol\beta}_{t\ell} = (\hat\beta_{0t\ell}, \hat\beta_{1t\ell}, \ldots, \hat\beta_{p_{t\ell}t\ell})^\top$ is the parameter vector associated with the node ℓ of the tree t

vector and then estimate its components. In the ridge and horseshoe regressions, a Gaussian with mean zero is assumed as the prior on the parameter vector. For lasso regression, a Laplace distribution is considered. For MOTR-BART, we assume a normal distribution with mean zero on $\boldsymbol\beta_{t\ell}$, but as the trees might change their dimensions depending on the moves growing and pruning, it is not possible to obtain the posterior distribution associated with each component of $\boldsymbol\beta_{t\ell}$ and then perform the variable selection.

Our idea to circumvent this issue is to consider in the linear predictor only covariates that have been used as a split in the corresponding tree. For instance, in Fig. 2 three covariates are used as a split (x_1, x_2 and x_3). The plan is to include these three covariates in each of the five linear predictors. The intuition in doing so is that if a covariate has been utilised as a split, it means that it improves the prediction either because it has a linear or a nonlinear relation with the response variable. If this relation is linear, this will be captured by the linear predictor. However, if the relation is nonlinear, the coefficient associated with this covariate will be close to zero and the covariate will not have impact on the prediction.

We have also explored using only the ancestors of the terminal nodes in the linear predictor as well as replacing the uniform branching process, where the covariates are selected with equal probability, by the Dirichlet branching process proposed by Linero (2018). To illustrate the first approach, we recall Fig. 2, where there are five terminal nodes and three covariates are used in the splitting rules. For the two leftmost terminal nodes, only the covariates x_1 and x_2 would be considered in both linear predictors. For the terminal node 3, only x_2. For the rightmost terminal nodes, x_3 and x_2 would be used. In relation to Linero's approach, rather than selecting the covariates with probability $1/p$, a Dirichlet prior is placed on the vector of splitting probability so that the covariates that are frequently used to create the internal nodes are more likely to be chosen. In the supplementary material, we show the performance of these and other strategies that we investigated to select the covariates that should be considered in the linear predictor.

4.1 MOTR-BART for classification

The version of MOTR-BART that was presented in Sect. 4 assumes that the response variable is continuous. In this section, we provide the extension to the case when it is binary following the idea of Chipman et al (2010), which used the strategy of data augmentation (Albert and Chib 1993). Firstly, we consider that $y_i \in \{0, 1\}$ and we introduce a latent variable

$$z_i \sim \mathrm{N}\left(\sum_{t=1}^{m} g(\mathbf{x}_i, T_t, B_t), 1\right), \text{ with } i = 1, \ldots, n$$

such that $y_i = 1$ if $z_i > 0$ and $y_i = 0$ if $z_i \leq 0$. With this formulation, we have that $p(y_i = 1|\mathbf{x}_i) = 8(\sum_{t=1}^m g(\mathbf{x}_i, T_t, B_t))$, where $8(\cdot)$ is the cumulative distribution function (cdf) of the standard normal, which works as the link function that limits the output to the interval $(0, 1)$. Here, there is no need to estimate the variance component as it is equal to 1. The priors on T_t and B_t are the same as in (1) and (4), respectively. Finally, as the latent variable z_i is introduced, it is necessary to compute its full conditional, which is given by

$$z_i|[y_i = 0] \sim \mathrm{N}_{(-\infty,0)}\left(\sum_{t=1}^{m} g(\mathbf{x}_i, T_t, B_t), 1\right),$$

$$z_i|[y_i = 1] \sim \mathrm{N}_{(0,\infty)}\left(\sum_{t=1}^{m} g(\mathbf{x}_i, T_t, B_t), 1\right),$$

where $\mathrm{N}_{(a,b)}(\cdot)$ denotes a truncated normal distribution constrained to the interval (a, b). Going back to Algorithm 1, some steps need to be modified or included:

1. The update of σ^2 is no longer needed, because we set $\sigma^2 = 1$;
2. The predicted values now consider the cdf of the standard normal as a probit model in the form of $\hat{\mathbf{y}}^{(k)} = 8\left(\sum_{t=1}^m g(\mathbf{X}, T_t, B_t)\right)$;
3. A Gibbs sampling step needs to be created to update the latent variables at each MCMC iteration. The update is done by drawing samples from $p(z_i|y_i)$;
4. Rather than calculating the partial residuals taking into account the response variable, we have that $R_t^{(k)} = \mathbf{z}^{(k)} - \sum_{j \neq t}^m g(\mathbf{X}, T_j, B_j)$, where $\mathbf{z}^{(k)}$ is the vector with the all latent variables at iteration k. For the first iteration, the vector $\mathbf{z}^{(1)}$ needs to be initialised and $R_1^{(1)} = \mathbf{z}^{(1)}$.

5 Results

In this section, we compare MOTR-BART to BART, RF, GB, lasso regression, soft BART and LLF via simulation scenarios and real data applications using the root mean square error (RMSE) as the accuracy measure. All results were generated by using R (R Core Team 2020) version 3.6.3 and the packages `dbarts` (Chipman et al 2010), `ranger` (Wright and Ziegler 2017), `gbm` (Greenwell et al 2019), `glmnet` (Friedman et al 2010), `SoftBart` (Linero 2017a) and `grf` (Tibshirani et al 2020). We use the default behaviour of these packages, except where otherwise specified below. We also tried running the linear random forests (LRF, Künzel et al (2019)) algorithm. However, we got errors when using the `forestry` R package and then we decided not to consider the LRF in our comparisons.

Throughout this section, we present results for two versions of our method. The first one is MOTR-BART (10 trees), which uses the Dirichlet branching process and estimates τ_{β_0} and τ_β, while the second is MOTR-BART (10 trees, fixed var), which uses the uniform branching process and sets $\tau_b = m$. As a default version, we recommend the MOTR-BART (10 trees).

5.1 Simulation

To compare the algorithms, we simulate data from the equation proposed by Friedman (1991). This data set is widely used in testing tree-based models and has been used repeatedly to evaluate the performance of BART and extensions (Friedman 1991; Chipman et al 2010; Linero 2018). We generate the response variable considering five covariates via:

$$y_i = 10\sin(\pi x_{i1} x_{i2}) + 20(x_{i3} - 0.5)^2 + 10x_{i4} + 5x_{i5} + \epsilon_i,$$

where the covariates $x_{ip} \sim \text{U}(0, 1)$, with $p = 1, \ldots, 5$, and $\epsilon_i \sim \text{N}(0, 1)$. For the simulation, we created 9 data sets with different numbers of observations (200, 500 and 1000) and covariates (5, 10 and 50). For those scenarios with 10 and 50 covariates, the additional x values do not have any impact on the response variable.

Each simulated data set was split into 10 different training (80%) and test (20%) partitions. For MOTR-BART, 10 trees were considered, 1000 iterations as burn-in, 5000 as post-burn-in, `alpha` $= \alpha = 0.95$ and `beta` $= \beta = 2$. To choose the number of trees (10) for MOTR-BART, we initially tested a range of possible values, such as 3, 10 and 50, and then, we used cross-validation to select that setting that presented the lowest RMSE. The set up for `dbarts` was similar to MOTR-BART, except for the number of trees (10 and 200, the default). For the packages `ranger` and `gbm`, the default options were kept, except for the number of trees (200) and the parameter `interaction.depth = 3`. For the `glmnet`, we followed the manual and used a 10-fold cross-validation with `type.measure = 'mse'` to obtain the estimate of the regularisation parameter `lambda.min`, which is the value that minimises the cross-validated error under the loss function chosen in `type.measure`. As in Chipman et al (2010), we evaluate the convergence of MOTR-BART and BART by eye from the plot of σ^2 after the burn-in period.

In Fig. 3, we present the comparison of the algorithms MOTR-BART, BART, RF, GB, lasso, soft BART and LLF in terms of RMSE on test data. Note that we have BART (10 trees) and BART (200 trees; default). The first version considers 10 trees and was run to see how BART would perform with the equivalent number of trees of MOTR-BART. We can see that for different combinations of number of observations (n) and covariates (p), soft BART and MOTR-BART (10 trees) consistently presented the best results for all scenarios. When compared to both versions of the original BART, both versions of MOTR-BART present lower median values of RMSE and slightly greater variability. However, the variability reduces as n and p increase. Further, we notice that MOTR-BART (10 trees) benefits from penalising the intercepts and slopes differently. In addition, it is possible to observe that the number of noisy covariates impacts on the performance of RF and LLF. For all values of n, their RMSEs increase with the number of covariates.

To further analyse the improvements given by MOTR-BART over standard BART, in Appendix A we present Tables 1, 2, 3, which shows the mean of the total number of terminal nodes utilised for BART to calculate the final prediction taking into account all the 5000 iterations. The idea of Table 3 is to show that MOTR-BART has similar or better performance while using fewer parameters than standard BART. As the default version of BART defaults to 200 trees, which is far more trees than MOTR-BART uses, we created Table 3 to highlight that although MOTR-BART estimates fewer parameters, it still remains competitive to the default BART. For MOTR-BART, we consider the mean of the number of parameters estimated in the linear predictors. As BART and soft BART predict a constant for each terminal node, the number of 'parameters' estimated is equal to the number of terminal nodes. On the other hand, MOTR-BART estimates an intercept, which is equivalent to the constant that BART predicts, plus the parameters associated with those covariates that have been used as a split in the corresponding tree. For instance, if a tree has 5 terminal nodes and 2 numeric variables are used in the splitting rules, MOTR-BART will estimate 15 parameters. For BART, we set the argument `keepTrees = TRUE`, and then, we extracted from the sampler object `fit` the content of `getTrees()`. For both MOTR-BART and BART, we firstly summed the number of parameters for all trees along the MCMC iterations and then averaged it over the 10 sets.

In Table 3, we can observe, for example, for the Friedman data set with $n = 1000$ and $p = 50$ that BART (10 trees) utilised 212,421 parameters on average to calculate the final prediction, while MOTR-BART (10 trees), BART (200 trees)

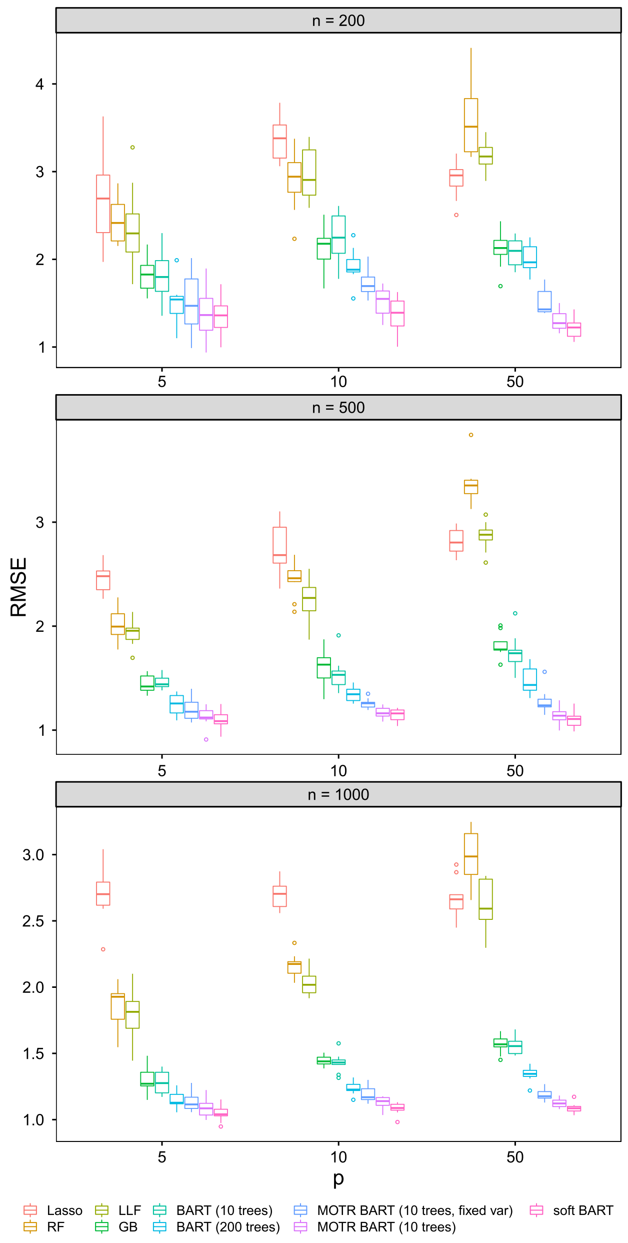

Fig. 3 Comparison of RMSE for Friedman data sets on test data for different combinations of n (200, 500 and 1000) and p (5, 10 and 50)

and soft BART used 391,193, 2,371,140 and 255,155, respectively. For all simulated data sets, MOTR-BART presented lower RMSE than BART (200 trees), even though it estimates far fewer parameters. From Table 3, it is possible to obtain the mean number of terminal nodes per tree by dividing the column 'Mean' by the number of MCMC iterations (5000) times the number of trees (10 or 200). In this case, we note that both versions of BART produce small trees with the mean number of terminal nodes per tree varying between 2 and 5. Due to the greater number of trees, BART (200 trees) has the lowest mean, regardless the number of observations and covariates.

In contrast, MOTR-BART has the mean number of parameters per tree varying from 5 to 8. Comparatively speaking, this is somewhat expected once MOTR-BART estimates a linear predictor. In this way, the trees from MOTR-BART tend to be shallower than those from BART (10 trees), but with more parameters estimated overall. It is important to highlight that the numbers from BART and MOTR-BART cannot be compared to those from RF, as the former work with the residuals and the latter with the response variable itself. The numbers for GB are not shown as the quantity of terminal nodes in each tree is fixed due to the parameter settings `interaction.depth = 3`.

In our simulations, MOTR-BART utilised just 10 trees and its results were better than RF, GB, BART (10 and 200 trees) and LLF. In practice, different number of trees may be compared via cross-validation and hence a choice can be made such that the cross-validated error is minimised.

5.2 Application

In this section, we compare the predictive performance of MOTR-BART to RF, GB, BART, soft BART and LLF in terms of RMSE on four real data sets. The first one (Ankara) has 1609 rows and contains weather information for the city of Ankara from 1994 to 1998. The goal is to predict the mean temperature based on 9 covariates. The second is the Boston Housing data set, where the response variable is the median value of properties in suburbs of Boston according to the 1970 US census. This data set has 506 rows and 18 explanatory variables. The third data set (Ozone) has 330 observations and 8 covariates and is about ozone concentration in Los Angeles in 1976. The aim is to predict the amount of ozone in parts per million (ppm) based on wind speed, air temperature, pressure gradient, humidity and other covariates. The fourth data set (Compactiv) refers to a multi-user computer that had the time of its activity measured under different tasks. The goal is to predict the portion of time that the computer runs in user mode for 8192 observations based on 21 covariates. These data sets are a subset of 9 sets considered by Kapelner and Bleich (2016).

As with the Friedman data, we consider two versions of BART (10 and 200 trees) and MOTR-BART (10 trees and 10 trees, fixed var), and we split the data into 10 different train (80%) and test (20%) sets. Furthermore, no transformations were applied to the response variables and all results are based on the test data.

Figure 4 shows the results of RMSE on test sets. It is possible to note that MOTR-BART (10 trees) presents the lowest or second lowest median RMSE on all data sets, except for Ozone. For Ankara, RF and GB have quite similar results and lasso presents the highest RMSE. For Boston, lasso regression shows the highest RMSE, while MOTR-BART (10 trees) and soft BART do not differ much in terms of median and

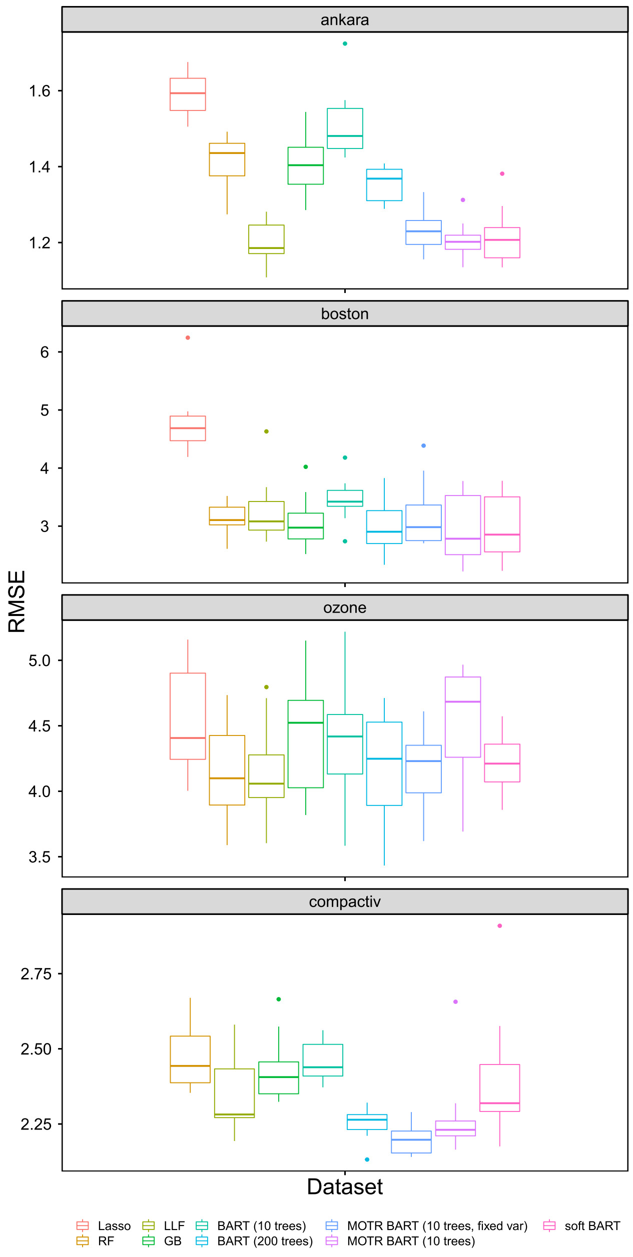

Fig. 4 Comparison of RMSE for Ankara, Boston, Ozone and Compactiv data sets on test data

quartiles. For Ozone, it can be seen that MOTR-BART (10 trees) presents the highest RMSE and that LLF, RF and MOTR-BART (10 trees) have the lowest median values. For Compactiv, RF and GB show similar results, while MOTR-BART (10 trees, fixed var) presents the lowest RMSE. To facilitate the visualisation, the results for lasso are not shown for the data set Compactiv, as it has RMSEs greater than 9. In Appendix B, however, Table 4 reports the median and the first and third quartiles of the RMSE for all algorithms and data sets.

In Table 5 (see Appendix B), we show the mean of the total number of parameters/terminal nodes created for BART to generate the final prediction for each data set. For MOTR-BART, the numbers correspond to the mean of the total of parameters estimated. For instance, for the data set Ankara, 304,696 terminal nodes were used on average by BART (10 trees), while BART (200 trees) estimated 2,250,599 and MOTR-BART 546,959. As can be seen, MOTR-BART estimates more parameters than BART (10 trees) for all data sets, as we expect. However, when compared to BART (200 trees), MOTR-BART utilises significantly fewer parameters to obtain similar or better performance, except for Compactiv.

6 Discussion

In this paper, we have proposed an extension of BART, called MOTR-BART, that can be seen as a combination of BART and model trees. In MOTR-BART, rather than having a constant as predicted value for each terminal node, a linear predictor is estimated considering only those covariates that have been used as a split in the corresponding tree. Furthermore, MOTR-BART is able to capture linear associations between the response and covariates at node level and it requires fewer trees to achieve equivalent or better performance when compared to other methods.

Via simulation studies and real data applications, we showed that MOTR-BART is highly competitive when compared to BART, random forests, gradient boosting, lasso regression, soft BART and LLF. In simulation scenarios, MOTR-BART outperformed the other tree-based methods, except soft BART. In the real data applications, four data sets were considered and MOTR-BART provided great predictive performance.

Due to the structure of MOTR-BART, to evaluate variable importance or even to select the covariates that should be included in the linear predictors is not straightforward. Recall that model trees were introduced in the context of one tree, where statistical methods of variable selection such as forward, backwards or stepwise can be performed at node level. Compared to other tree-based methods that consider only one tree, model trees produces much smaller trees (Landwehr et al 2005), which helps to alleviate the computational time required by the variable selection procedures. In theory, one might think that it would be possible to use such a procedure for MOTR-BART, but in practice they would be a burden as they would be performed for each terminal node out of all trees within every MCMC iteration.

In the Bayesian context, Chipman et al (2010) propose to use the inclusion probability as a measure of variable importance. Basically, this metric is the proportion of times that a covariate is used as a split out of all splitting rules over all trees and MCMC iterations. However, this measure gives us an idea about the covariates that are important for the splitting rules and does not say anything about which covariates that should be included in the linear predictors.

In this sense, the variable selection/importance remains as a challenge that may be investigate in future work, once conventional procedures are not suitable. One might try proposing adaptations of ridge, lasso or horseshoe regressions for trees. Another extension could be replacing the linear functions by Splines to provide even further flexibility and capture local nonlinear behaviour, which is a subject of ongoing work. Finally, model trees can be incorporated to other BART extensions, such as BART for log-linear models (Murray 2017), soft BART (Linero and Yang 2018) and BART for log-normal and gamma hurdle models (Linero et al 2018). We hope to produce an R package that implements our methods shortly; current code is available at https://github.com/ebprado/MOTR-BART.

Acknowledgements We thank the editors and the two anonymous referees for their comments that greatly improved the earlier version of the paper. This work was supported by a Science Foundation Ireland Career Development Award grant number 17/CDA/4695.

Appendix A: Simulation results

In this section, we present results related to the simulation scenarios shown in Sect. 5.1. In total, 9 data sets were created based on Friedman's equation considering some combinations of sample size (n) and number of covariates (p). In Tables 1 and 2 , the medians and quartiles of the RMSE are shown for the algorithms MOTR-BART, BART, GB, RF, lasso, soft BART and LLF. The values in this table were graphically shown in Fig. 3. In addition, Table 3 presents the mean number of parameters utilised by BART, MOTR-BART and soft BART to calculate the final prediction.

Table 1 Median of the RMSE on test data of the Friedman data sets when $n = 200$ and 500

Algorithm	p	RMSE
$n = 200$		
MOTR-BART	5	**1.36 (1.19;1.55)**
MOTR-BART (fixed var)	5	1.47 (1.26;1.78)
BART (10 trees)	5	1.80 (1.63;1.99)
BART (200 trees)	5	1.54 (1.38;1.58)
GB	5	1.83 (1.67;1.93)
RF	5	2.41 (2.21;2.63)
Lasso	5	2.69 (2.30;2.96)
soft BART	5	**1.36 (1.22;1.47)**
LLF	5	2.30 (2.08;2.52)
MOTR-BART	10	**1.55 (1.39;1.64)**
MOTR-BART (fixed var)	10	1.70 (1.63;1.80)
BART (10 trees)	10	2.25 (2.07;2.49)
BART (200 trees)	10	1.88 (1.86;2.00)
GB	10	2.18 (2.00;2.24)
RF	10	2.94 (2.76;3.10)
Lasso	10	3.38 (3.15;3.53)
soft BART	10	**1.39 (1.24;1.52)**
LLF	10	2.91 (2.73;3.25)
MOTR-BART	50	**1.27 (1.21;1.38)**
MOTR-BART (fixed var)	50	1.43 (1.40;1.63)
BART (10 trees)	50	2.10 (1.94;2.21)
BART (200 trees)	50	1.97 (1.90;2.14)
GB	50	2.13 (2.06;2.22)
RF	50	3.51 (3.23;3.83)
Lasso	50	2.96 (2.84;3.02)
soft BART	50	**1.22 (1.12;1.27)**
LLF	50	3.17 (3.09;3.28)
$n = 500$		
MOTR-BART	5	**1.12 (1.11;1.19)**
MOTR-BART (fixed var)	5	1.18 (1.11;1.27)
BART (10 trees)	5	1.44 (1.42;1.50)
BART (200 trees)	5	1.26 (1.17;1.33)
GB	5	1.42 (1.38;1.52)
RF	5	2.00 (1.92;2.12)
Lasso	5	2.48 (2.35;2.53)
soft BART	5	**1.09 (1.06;1.15)**
LLF	5	1.96 (1.87;1.98)
MOTR-BART	10	**1.16 (1.14;1.21)**
MOTR-BART (fixed var)	10	1.26 (1.22;1.27)
BART (10 trees)	10	1.53 (1.44;1.57)
BART (200 trees)	10	1.35 (1.28;1.39)
GB	10	1.63 (1.50;1.70)
RF	10	2.46 (2.43;2.53)
Lasso	10	2.68 (2.61;2.95)
soft BART	10	**1.16 (1.10;1.19)**
LLF	10	2.27 (2.15;2.37)
MOTR-BART	50	**1.14 (1.10;1.18)**
MOTR-BART (fixed var)	50	1.24 (1.22;1.30)
BART (10 trees)	50	1.74 (1.66;1.77)
BART (200 trees)	50	1.43 (1.38;1.59)
GB	50	1.78 (1.77;1.85)
RF	50	3.35 (3.27;3.40)
Lasso	50	2.80 (2.72;2.92)

Table 1 continued

Algorithm	p	RMSE
soft BART	50	**1.11 (1.05;1.13)**
LLF	50	2.88 (2.83;2.92)

The values in parentheses are the first and third quartiles, respectively

Table 2 Median of the RMSE on test data of the Friedman data sets when $n = 1000$

Algorithm	p	RMSE
$n = 1000$		
MOTR-BART	5	**1.09 (1.03;1.12)**
MOTR-BART (fixed var)	5	1.11 (1.08;1.17)
BART (10 trees)	5	1.28 (1.20;1.36)
BART (200 trees)	5	1.13 (1.12;1.19)
GB	5	1.27 (1.25;1.36)
RF	5	1.93 (1.76;1.95)
Lasso	5	2.70 (2.62;2.79)
soft BART	5	**1.04 (1.03;1.08)**
LLF	5	1.81 (1.69;1.89)
MOTR-BART	10	**1.14 (1.11;1.17)**
MOTR-BART (fixed var)	10	1.17 (1.15;1.23)
BART (10 trees)	10	1.43 (1.42;1.45)
BART (200 trees)	10	1.23 (1.22;1.27)
GB	10	1.44 (1.42;1.47)
RF	10	2.17 (2.10;2.19)
Lasso	10	2.70 (2.61;2.76)
soft BART	10	**1.09 (1.07;1.12)**
LLF	10	2.02 (1.96;2.08)
MOTR-BART	50	**1.12 (1.10;1.15)**
MOTR-BART (fixed var)	50	1.18 (1.16;1.21)
BART (10 trees)	50	1.55 (1.50;1.59)
BART (200 trees)	50	1.35 (1.33;1.37)
GB	50	1.57 (1.55;1.61)
RF	50	2.99 (2.85;3.16)
Lasso	50	2.66 (2.59;2.70)
soft BART	10	**1.09 (1.07;1.10)**
LLF	10	2.59 (2.51;2.81)

The values in parentheses are the first and third quartiles, respectively

Table 3 Friedman data sets: mean and standard deviation of the total number of terminal nodes created for BART and soft BART to generate the final prediction over 5000 iterations

Algorithm	p	Mean	SD
$n = 200$			
MOTR-BART	5	302,447	18,210
MOTR-BART (fixed var)	5	263,079	11,990
BART (10 trees)	5	163,468	7393
BART (200 trees)	5	2,468,707	6734
soft BART	5	250,615	6599
MOTR-BART	10	326,678	20,309
MOTR-BART (fixed var)	10	258,380	13,530
BART (10 trees)	10	145,458	7380
BART (200 trees)	10	2,470,333	3670
soft BART	10	256,391	5577
MOTR-BART	50	327,751	28,627
MOTR-BART (fixed var)	50	251,469	12,376
BART (10 trees)	50	134,809	4447
BART (200 trees)	50	2,428,259	5368
soft BART	50	256,184	6754
$n = 500$			
MOTR-BART	5	364,786	21,978
MOTR-BART (fixed var)	5	364,258	17,478
BART (10 trees)	5	203,625	7950
BART (200 trees)	5	2,470,900	8739
soft BART	5	257,769	6727
MOTR-BART	10	382,528	24,768
MOTR-BART (fixed var)	10	354,755	25,238
BART (10 trees)	10	206,394	8694
BART (200 trees)	10	2,448,212	9171
soft BART	10	256,465	2829
MOTR-BART	50	384,828	18,099
MOTR-BART (fixed var)	50	330,434	33,566
BART (10 trees)	50	178,779	8700
BART (200 trees)	50	2,407,661	14,314
soft BART	50	254,164	9334
$n = 1000$			
MOTR-BART	5	389,479	23,247
MOTR-BART (fixed var)	5	396,280	29,040
BART (10 trees)	5	271,878	8977
BART (200 trees)	5	2,425,863	8276
soft BART	5	257,656	9,881
MOTR-BART	10	410,517	30,346
MOTR-BART (fixed var)	10	390,274	22,442
BART (10 trees)	10	256,511	7,812
BART (200 trees)	10	2,415,372	8,575
soft BART	10	255,604	6549
MOTR-BART	50	391,193	16,127
MOTR-BART (fixed var)	50	380,365	40,069
BART (10 trees)	50	212,421	5959
BART (200 trees)	50	2,371,140	14,287
soft BART	50	255,155	4400

For MOTR-BARTs, the values correspond to the mean of the total number of parameters estimated in the linear predictors

Appendix B: Real data results

This appendix presents two tables with results associated with the data sets Ankara, Boston, Ozone and Compactiv. In Table 4, it is reported the median and quartiles of the RMSE computed on 10 test sets. The values in this table are related to the Fig. 4 from Sect. 5.2. Further, Table 5 shows the mean number of parameters utilised by BART, MOTR-BART and soft BART to calculate the final prediction for the aforementioned data sets.

Table 4 Real data sets: comparison of the median RMSE (and first and third quartiles) for Ankara, Boston, Ozone and Compactiv data sets on test data

Data set	Algorithm	RMSE	rank
Ankara	MOTR-BART	**1.20 (1.18;1.22)**	2
	MOTR-BART (fv)	1.23 (1.20;1.26)	4
	BART (200 trees)	1.37 (1.31;1.39)	5
	BART (10 trees)	1.48 (1.45;1.55)	8
	GB	1.40 (1.35;1.45)	6
	RF	1.44 (1.38;1.46)	7
	Lasso	1.59 (1.55;1.63)	9
	soft BART	1.21 (1.16;1.24)	3
	LLF	**1.19 (1.17;1.25)**	1
Boston	MOTR-BART	**2.78 (2.51;3.53)**	1
	MOTR-BART (fv)	2.98 (2.75;3.36)	5
	BART (200 trees)	2.90 (2.70;3.27)	3
	BART (10 trees)	3.42 (3.34;3.62)	8
	GB	2.97 (2.78;3.22)	4
	RF	3.10 (3.02;3.33)	6
	Lasso	4.69 (4.47;4.89)	9
	soft BART	**2.85 (2.56;3.50)**	2
	LLF	3.08 (2.93;3.42)	7
Ozone	MOTR-BART	4.68 (4.26;4.87)	9
	MOTR-BART (fv)	4.23 (3.99;4.35)	3
	BART (200 trees)	4.25 (3.89;4.53)	4
	BART (10 trees)	4.42 (4.13;4.59)	6
	GB	4.52 (4.03;4.69)	8
	RF	**4.10 (3.89;4.43)**	2
	Lasso	4.41 (4.24;4.90)	7
	soft BART	4.21 (4.07;4.36)	5
	LLF	**4.06 (3.95;4.28)**	1
Compactiv	MOTR-BART	**2.23 (2.21;2.26)**	2
	MOTR-BART (fv)	**2.20 (2.15;2.23)**	1
	BART (200 trees)	2.26 (2.23;2.28)	3
	BART (10 trees)	2.44 (2.41;2.51)	7
	GB	2.41 (2.35;2.46)	6
	RF	2.44 (2.39;2.54)	8
	Lasso	9.97 (9.51;10.09)	9
	soft BART	2.32 (2.29;2.45)	5
	LLF	2.28 (2.27;2.43)	4

The acronym ‘fv’ stands for ‘fixed var’

Table 5 Real data sets: mean and standard deviation of the total number of terminal nodes created for BART and soft BART to generate the final prediction over 5000 iterations

Data set	Algorithm	Mean	SD
Ankara	MOTR-BART	546,959	36,977
	MOTR-BART (fv)	485,743	40,840
	BART (10 trees)	304,696	8872
	BART (200 trees)	2,250,599	11,798
	soft BART	312,927	11,539
Boston	MOTR-BART	748,468	58,945
	MOTR-BART (fv)	414,762	50,705
	BART (10 trees)	204,038	10,143
	BART (200 trees)	2,389,130	14,244
	soft BART	318,171	17,078
Ozone	MOTR-BART	272,370	42,809
	MOTR-BART (fv)	182,189	8093
	BART (10 trees)	137,239	2642
	BART (200 trees)	2,343,350	5128
	soft BART	268,948	5667
Compactiv	MOTR-BART	2,990,494	298,221
	MOTR-BART (fv)	1,529,666	102,940
	BART (10 trees)	539,621	15,759
	BART (200 trees)	2,649,167	29,989
	soft BART	711,860	49,087

For MOTR-BARTs, the values correspond to the mean of the total number of parameters estimated in the linear predictors. The acronym ‘fv’ stands for ‘fixed var’

References

Albert, J.H., Chib, S.: Bayesian analysis of binary and polychotomous response data. J. Am. Stat. Assoc. **88**(422), 669–679 (1993)

Athey, S., Tibshirani, J., Wager, S., et al.: Generalized random forests. Ann. Stati. **47**(2), 1148–1178 (2019)

Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)

Carvalho, C.M., Polson, N.G., Scott, J.G.: The horseshoe estimator for sparse signals. Biometrika **97**(2), 465–480 (2010)

Chipman, H.A., George, E.I., McCulloch, R.E.: Bayesian cart model search. J. Am. Stat. Assoc. **93**(443), 935–948 (1998)

Chipman, H.A., George, E.I., McCulloch, R.E., et al.: Bart: Bayesian additive regression trees. Ann. Appl. Stat. **4**(1), 266–298 (2010)

Deshpande, S.K., Bai, R., Balocchi, C., Starling, J.E.: (2020) Vc-bart: Bayesian trees for varying coefficients. arXiv preprint arXiv:2003.06416

Friedberg, R., Tibshirani, J., Athey, S., Wager, S.: Local linear forests. (2018) arXiv preprint arXiv:1807.11408

Friedman, J., Hastie, T., Tibshirani, R.: Regularization paths for generalized linear models via coordinate descent. J. Stat. Softw. **33**(1), 1 (2010)

Friedman, J.H.: Multivariate adaptive regression splines. The annals of statistics pp 1–67 (1991)

Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Ann. Stat. pp. 1189–1232 (2001)

Green, D.P., Kern, H.L.: Modeling heterogeneous treatment effects in survey experiments with bayesian additive regression trees. Public Opin. Quart. **76**(3), 491–511 (2012)

Greenwell, B., Boehmke, B., Cunningham, J., Developers, G.: gbm: Generalized boosted regression models. https://CRAN.R-project.org/package=gbm, r package version 2.1.5 (2019)

Hahn, P.R., Murray, J.S., Carvalho, C.M., et al.: Bayesian regression tree models for causal inference: regularization, confounding, and heterogeneous effects. Bayesian Analysis (2020)

He, J., Yalov, S., Hahn, P.R.: Xbart: Accelerated Bayesian additive regression trees. In: Proceedings of the 22nd international conference on artificial intelligence and statistics 89 (2019)

Hernández, B., Pennington, S.R., Parnell, A.C.: Bayesian methods for proteomic biomarker development. EuPA Open Proteom. **9**, 54–64 (2015)

Hernández, B., Raftery, A.E., Pennington, S.R., Parnell, A.C.: Bayesian additive regression trees using bayesian model averaging. Stat. Comput. **28**(4), 869–890 (2018)

Hill, J.L.: Bayesian nonparametric modeling for causal inference. J. Comput. Gr. Stat. **20**(1), 217–240 (2011)

Kapelner, A., Bleich, J.: bartMachine: Machine learning with Bayesian additive regression trees. J. Stat. Softw. **70**(4), 1–40 (2016). https://doi.org/10.18637/jss.v070.i04

Kindo, B.P., Wang, H., Hanson, T., Peña, E.A.: (2016a) Bayesian quantile additive regression trees. arXiv preprint arXiv:1607.02676

Kindo, B.P., Wang, H., Peña, E.A.: Multinomial probit bayesian additive regression trees. Stat **5**(1), 119–131 (2016b)

Künzel, S.R., Saarinen, T.F., Liu, E.W., Sekhon, J.S.: Linear aggregation in tree-based estimators (2019) arXiv preprint arXiv:1906.06463

Landwehr, N., Hall, M., Frank, E.: Logistic model trees. Mach. Learn. **59**(1–2), 161–205 (2005)

Linero, A.: SoftBart: A package for implementing the SoftBart algorithm. R package version **1**, (2017a)

Linero, A.R.: A review of tree-based bayesian methods. Commun. Stat. Appl. Methods **24**(6), (2017b)

Linero, A.R.: Bayesian regression trees for high-dimensional prediction and variable selection. J. Am. Stat. Assoc. **113**(522), 626–636 (2018)

Linero, A.R., Yang, Y.: Bayesian regression tree ensembles that adapt to smoothness and sparsity. J. R. Stat. Soc. Ser. B (Stat. Methodol.) **80**(5), 1087–1110 (2018)

Linero, A.R., Sinha, D., Lipsitz, S.R.: Semiparametric mixed-scale models using shared bayesian forests (2018) arXiv preprint arXiv:1809.08521

McCulloch, R., Sparapani, R., Gramacy, R., Spanbauer, C., Pratola, M.: BART: Bayesian Additive Regression Trees. https://CRAN.R-project.org/package=BART, r package version 2.7 (2019)

Murray, J.S.: Log-linear bayesian additive regression trees for categorical and count responses.(2017) arXiv preprint arXiv:1701.01503

Pratola, M., Chipman, H., George, E., McCulloch, R.: Heteroscedastic bart using multiplicative regression trees (2017). arXiv preprint arXiv:1709.07542

Quinlan, J.R.: Learning with continuous classes. In: 5th Australian joint conference on artificial intelligence, World Scientific, vol 92, pp 343–348 (1992)

R Core Team R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna (2020) https://www.R-project.org/

Ročková, V., van der Pas, S.: Posterior concentration for bayesian regression trees and forests (2017). arXiv preprint arXiv:1708.08734

Ročková, V., Saha, E.: On theory for bart (2018). arXiv preprint arXiv:1810.00787

Schnell, P.M., Tang, Q., Offen, W.W., Carlin, B.P.: A bayesian credible subgroups approach to identifying patient subgroups with positive treatment effects. Biometrics **72**(4), 1026–1036 (2016)

Sivaganesan, S., Müller, P., Huang, B.: Subgroup finding via bayesian additive regression trees. Stat. Med. **36**(15), 2391–2403 (2017)

Sparapani, R., Logan, B.R., McCulloch, R.E., Laud, P.W.: Nonparametric competing risks analysis using bayesian additive regression trees. Stat. Methods Med. Res. p 0962280218822140 (2019)

Sparapani, R.A., Logan, B.R., McCulloch, R.E., Laud, P.W.: Nonparametric survival analysis using bayesian additive regression trees (bart). Stat. Med. **35**(16), 2741–2753 (2016)

Starling, J.E., Aiken, C.E., Murray, J.S., Nakimuli, A., Scott, J.G.: Monotone function estimation in the presence of extreme data coarsening: Analysis of preeclampsia and birth weight in urban uganda (2019). arXiv preprint arXiv:19120.6946

Starling, J.E., Murray, J.S., Carvalho, C.M., Bukowski, R.K., Scott, J.G., et al.: Bart with targeted smoothing: an analysis of patient-specific stillbirth risk. Ann. Appl. Stat. **14**(1), 28–50 (2020)

Tibshirani, J., Athey, S., Wager, S.: grf: Generalized Random Forests (2020). https://CRAN.R-project.org/package=grf, r package version 1.2.0

Tibshirani, R.: Regression shrinkage and selection via the lasso. J. Roy. Stat. Soc.: Ser. B (Methodol.) **58**(1), 267–288 (1996)

Wang, Y., Witten, I., van Someren, M., Widmer, G.: Inducing models trees for continuous classes. In: Proceedings of the Poster Papers of the European Conference on Machine Learning, Department of Computer Science, University of Waikato, New Zeland (1997)

Wright, M.N., Ziegler, A.: ranger: A fast implementation of random forests for high dimensional data in C++ and R. J. Stat. Softw. **77**(1), 1–17 (2017). https://doi.org/10.18637/jss.v077.i01

Zhang, J.L., Härdle, W.K.: The bayesian additive classification tree applied to credit risk modelling. Comput. Stat. Data Anal. **54**(5), 1197–1205 (2010)