Bayesian Decision Theory on Decision Trees: Uncertainty Evaluation and Interpretability

Yuta Nakahara
Waseda University

Shota Saito
Gunma University

Naoki Ichijo
Waseda University

Koki Kazama
Shonan Institute of Technology

Toshiyasu Matsushima
Waseda University

Abstract

Deterministic decision trees have difficulty in evaluating uncertainty especially for small samples. To solve this problem, we interpret the decision trees as stochastic models and consider prediction problems in the framework of Bayesian decision theory. Our models have three kinds of parameters: a tree shape, leaf parameters, and inner parameters. To make Bayesian optimal decisions, we have to calculate the posterior distribution of these parameters. Previously, two types of methods have been proposed. One marginalizes out the leaf parameters and samples the tree shape and the inner parameters by Metropolis-Hastings (MH) algorithms. The other marginalizes out both the leaf parameters and the tree shape based on a concept called meta-trees and approximates the posterior distribution for the inner parameters by a bagging-like method. In this paper, we propose a novel MH algorithm where the leaf parameters and the tree shape are marginalized out by using the meta-trees and only the inner parameters are sampled. Moreover, we update all the inner parameters simultaneously in each MH step. This algorithm accelerates the convergence and mixing of the Markov chain. We evaluate our algorithm on various benchmark datasets with other state-of-the-art methods. Further, our model provides a novel statistical evaluation of feature importance.

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

1 INTRODUCTION

Decision trees are widely used in the field of machine learning as algorithms for handling regression and classification problems. Many of these decision trees represent deterministic functions for predicting new data without explicitly assuming the data observation process by a stochastic model, e.g., CART (Breiman et al., 1984), RandomForest (Breiman, 2001), LightGBM (Ke et al., 2017), and XGBoost (Chen and Guestrin, 2016). This can be seen as an attempt to handle data observation processes as widely as possible. However, because the data observation process is not expressed using a stochastic model, the following issues arise:

Issue 1: Difficulty in evaluating uncertainty It is difficult to optimally evaluate the uncertainty of predictions based on statistical theories such as Bayesian decision theory. This problem becomes particularly prominent when the sample size is small.

Issue 2: Difficulty in statistical interpretation of feature importance One of the advantages of decision tree-based algorithms is that they can calculate the importance of each feature, see, e.g., the implementation in scikit-learn (Pedregosa et al., 2011). However, such a feature importance often lacks a clear statistical interpretation.

Therefore, in this study, we propose a decision-tree-based stochastic model that represents the data observation process and assume that both training data and new data are observed according to this model. Under this assumption, we formulate the problem of evaluating the uncertainty of a new data point within the framework of Bayesian decision theory (Berger, 1985). Although similar models and problem settings have already been proposed by Chipman et al. (1998, 2010) and Dobashi et al. (2021), there are some differences between these and our approach, which will be de-

tailed in the next section. If we can calculate the optimal decision based on Bayesian decision theory for uncertainty evaluation, we can solve the aforementioned Issue 1.

However, there are still challenges in calculating the optimal decision under Bayesian decision theory. Our stochastic model has three types of parameters: tree shape, leaf node parameters, and inner node parameters. In order to calculate the optimal decision based on Bayesian decision theory, it is necessary to calculate the posterior distribution for all of them.

Conventionally, two types of methods have been proposed. The first type of method exactly calculates the posterior distribution of the leaf node parameters and approximates the posterior distribution of the tree shape and the inner node parameters by using the Metropolis-Hastings (MH) algorithm(Chipman et al., 1998, 2010). While this method is asymptotically able to calculate the exact posterior distribution as the number of MH steps increases, the parameter space of the posterior distribution to be approximated is quite large.

The second type of method exactly calculates the posterior distribution of both the leaf node parameters and the tree shape based on a concept called meta-trees and approximates the posterior distribution of the inner node parameters by using a bagging-like method (Dobashi et al., 2021). However, this approximation has no guarantee that it calculates the exact posterior distribution, even asymptotically.

In this paper, we propose a method that exactly calculates the posterior distribution of the leaf node parameters and the tree shape by using the meta-trees and approximates only the posterior distribution of the inner node parameters using the MH algorithm. Compared to the first approach, our method significantly reduces the dimensionality of the distribution to be approximated by the MH algorithm, and the convergence and mixing of the Markov chain will be accelerated. Furthermore, in our method, all inner node parameters can be updated simultaneously in a single MH step. This will also accelerate the convergence and mixing. Compared to the second approach, our method guarantees an asymptotically exact posterior distribution calculation as the number of MH steps increases.

Moreover, not only for our model, we believe that our method has a broader impact to accelerate the inference algorithms for various machine learning models with tree structures (Chipman et al., 2010; Jordan and Jacobs, 1994; Ghahramani et al., 2010) by eliminating the need for sampling of tree structures in Markov chain Monte Carlo (MCMC) methods.

For Issue 2, we propose a novel feature importance with a clear statistical interpretation, taking the advantage of the explicit assumption of the data observation process. Qualitatively, our feature importance represents a sum of the posterior expectation of the increase of the log Bayes factor at each inner node with that feature.

Finally, we evaluate the effectiveness of our method for these issues on various benchmark datasets and artificial datasets by comparing it with other state-of-the-art methods. Source code of our algorithm is publicly available in our library called BayesML (Nakahara et al., 2025).

2 RELATED WORK

2.1 Relationship with BayesianCART

The first study to consider decision trees as stochastic models is BayesianCART (Chipman et al., 1998). Our stochastic model is similar to the one in BayesianCART, but more general because our model can assume a wider range of models for leaf nodes, such as linear regression (LR) models, Poisson distributions, and exponential distributions. Furthermore, BayesianCART does not mention decision-making based on Bayesian decision theory. From the perspective of algorithms to calculate the posterior distribution, BayesianCART calculates the posterior distribution of leaf node parameters exactly, but approximates the posterior distribution of tree shape and inner node parameters using the MH algorithm. Chipman et al. (1998) pointed out this algorithm may struggle with multimodality of the posterior distribution and recommended repeatedly restarting the algorithm. In contrast, our method approximates only the inner node parameters using the MH algorithm. Moreover, we extend it to a replica exchange Monte Carlo (REMC) method (Swendsen and Wang, 1986) to deal with multimodality of the posterior distribution. This accelerates the convergence and mixing of the Markov chain. Lastly, the BayesianCART algorithm is not widely available as a library, while our method is publicly available in our library called BayesML (Nakahara et al., 2025).

2.2 Relationship with BART

BART (Chipman et al., 2010) is a study that extends BayesianCART. BART considers a model where additive noise ϵ is added to a nonlinear function $f(\boldsymbol{x})$ expressed as a sum of decision tree functions, known as the sum-of-trees model. There are two differences between this model and ours. First, our model considers a single-tree model. Second, the variance of the

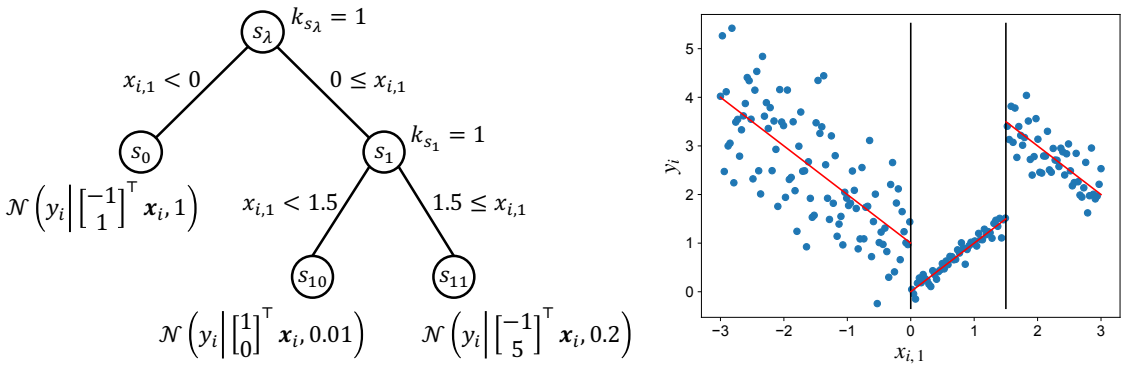


Figure 1: An example of $p(y_i|\boldsymbol{x}_i, \boldsymbol{k}, T, \boldsymbol{\theta})$ and an observed sample. Here, the model assigned to the leaf node s is the LR model $\mathcal{N}(y_i|\boldsymbol{w}_s^\top \boldsymbol{x}_i, \sigma_s^2)$, and the leaf node parameter is $\boldsymbol{\theta}_s = (\boldsymbol{w}_s, \sigma_s^2)$. Here, $x_{i,2}$ is an intercept and $x_{i,2} = 1$ holds for any i.

noise term ϵ in BART is assumed to be independent of the explanatory variables, i.e., BART's model is homoscedastic. In contrast, the variance of the noise in our model can differ for each leaf node, i.e., our model is heteroscedastic. Furthermore, BART does not mention decision-making based on Bayesian decision theory. Regarding the posterior distribution calculation algorithm, like BayesianCART, it approximates the posterior distribution for the tree shape by the MH algorithm. Therefore, by extending our model to a model consisting of multiple decision trees, our algorithm has a potential to contribute to accelerating the BART algorithm.

2.3 Relationship with MTRF

The first study to deal with decision trees within the framework of Bayesian decision theory is reported by Suko et al. (2003). Meta-tree random forest (MTRF) (Dobashi et al., 2021) is an extended version of this, and our stochastic model is basically equivalent to the model used in MTRF. However, MTRF in the original paper did not support continuous explanatory variables and non-binary objective variables. For the continuous explanatory variables, we published an algorithm that utilizes the inner node parameters (features and thresholds) obtained from a trained RF as hyperparameters of our model in our open-source library BayesML (Nakahara et al., 2025). For non-binary objective variables, MTRF will support them by applying the approach proposed in this paper. There are also differences in the problems solved under Bayesian decision theory. MTRF only deals with classification problems that minimize the 0-1 loss, while this study deals with regression problems that output prediction intervals (see Section 5) and classification problems that minimize the log loss (see Appendix E). Furthermore, there are differences in the algorithm for calculating Bayesian optimal decisions. In MTRF, not only the posterior distribution of leaf node parameters but also the posterior distribution of the tree shape are calculated exactly. Since this computation is based on a concept called a meta-tree, this method is called a meta-tree random forest. However, the posterior distribution of the inner node parameters is approximated by using a heuristic method based on the idea of bagging, which does not have any guarantees. In contrast, our method calculates the posterior distribution of the inner node parameters by the MH algorithm. Therefore, asymptotically exact posterior distribution is obtained after sufficient large number of MH steps.

3 DECISION TREES AS STOCHASTIC MODELS

In this section, we present an overview of our decision-tree-based stochastic model and the prior distribution. For more details, please refer to Appendix A.

3.1 Data observation process $p(y_i|\boldsymbol{x}_i, \boldsymbol{k}, T, \boldsymbol{\theta})$

In this section, we define the stochastic model $p(y_i|\boldsymbol{x}_i, \boldsymbol{k}, T, \boldsymbol{\theta})$ that represents the process of observing the target variable y_i given the explanatory variable $\boldsymbol{x}_i$. Figure 1 shows an overview of the model and an example of samples observed from that model.[1] This model is represented using a binary tree T, and each inner node s of T has inner node parameters k_s and t_s. Here, k_s is the index of the explanatory variable and represents the axis of the partitioning of the explanatory variable space, and t_s is the threshold for the partition. In this study, we assume that the par-

[1]This is a toy example to illustrate only the properties of our model and the data observation process. In a model with only one explanatory variable like this, we need not compute the posterior distribution of $\boldsymbol{k}$.

tition threshold t_s can be deterministically calculated by some rule[2] based only on the explanatory variables and k_s, and is not considered an unknown parameter of our model. This is a main limitation of our model. Let $\boldsymbol{k}$ be the tuple of k_s for all inner nodes, then the leaf node to which $\boldsymbol{x}_i$ belongs is uniquely determined by T and $\boldsymbol{k}$. This leaf node is denoted by $s_{\boldsymbol{k},T}(\boldsymbol{x}_i)$. Then, y_i is assumed to be observed independently according to the stochastic model $p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}_{s_{\boldsymbol{k},T}(\boldsymbol{x}_i)})$ assigned to $s_{\boldsymbol{k},T}(\boldsymbol{x}_i)$. That is, the likelihood function of $y^n \coloneqq (y_1, \ldots, y_n)$ given $\boldsymbol{x}^n \coloneqq (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ is given as follows:

$$p(y^n|\boldsymbol{x}^n, \boldsymbol{k}, T, \boldsymbol{\theta}) = \prod_{i=1}^{n} p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}_{s_{\boldsymbol{k},T}(\boldsymbol{x}_i)}), \qquad (1)$$

where $\boldsymbol{\theta}$ is the tuple of the leaf node parameter $\boldsymbol{\theta}_s$ for all leaf nodes.

Here, various models can be considered for the stochastic model $p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}_s)$ assigned to each leaf node s, such as Bernoulli distribution, categorical distribution, Poisson distribution, normal distribution, exponential distribution, and LR models. In particular, the expected value and variance of the distribution may differ for each node (see also Figure 1). In other words, our model can express heteroscedasticity. This flexibility of the model is an advantage compared to previous studies such as BayesianCART, BART, and MTRF.

3.2 Prior distribution $p(\boldsymbol{k}, T, \boldsymbol{\theta})$

In this study, we assume a prior distribution $p(\boldsymbol{k}, T, \boldsymbol{\theta})$ for the unknown parameters $(\boldsymbol{k}, T, \boldsymbol{\theta})$. This allows for discussion based on Bayesian decision theory. To define the prior distribution, we first assume the maximum tree $T_{\max}$. Then, for each inner node s in $T_{\max}$, we assume that k_s is independently and uniformly distributed over all feature indices. After that, T is generated from a distribution $p(T)$ over subtrees of $T_{\max}$ independently of $\boldsymbol{k}$. This tree distribution was first proposed by Matsushima and Hirasawa (1994), then mathematically summarized by Nakahara et al. (2022), and also used in MTRF. In this prior distribution $p(T)$, the probability of the tree decay with respect to its depth. Therefore, it prevents overfitting in the learning phase. Moreover, under some conditions, this prior distribution coincides with the tree prior distribution used in BayesianCART and BART. We will use this property for fair comparison in experiments in Section 5. In addition, $p(\boldsymbol{\theta})$ is assumed to be the conjugate prior distribution for $p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}_{s_{\boldsymbol{k},T}(\boldsymbol{x}_i)})$ and independent for each node. See Appendix A for details.

[2]For example, recursively bisecting the given maximum and minimum values of the explanatory variables. See Appendix G for more effective ways to determine the threshold.

4 META-TREE MARKOV CHAIN MONTE CARLO METHODS

In Bayesian decision theory, calculating the posterior distribution of unknown parameters is often necessary for optimal decision making (Berger, 1985). In the case of this study, it means that the calculation of $p(\boldsymbol{k}, T, \boldsymbol{\theta}|\boldsymbol{x}^n, y^n)$ is necessary. This distribution can be decomposed as follows:

$$\begin{aligned} &p(\boldsymbol{k}, T, \boldsymbol{\theta}|\boldsymbol{x}^n, y^n) \\ &= p(\boldsymbol{k}|\boldsymbol{x}^n, y^n) p(T|\boldsymbol{x}^n, y^n, \boldsymbol{k}) p(\boldsymbol{\theta}|\boldsymbol{x}^n, y^n, \boldsymbol{k}, T). \end{aligned} \qquad (2)$$

In conventional studies such as BayesianCART and BART, $\boldsymbol{k}$ and T are sampled using the MH algorithm where $p(\boldsymbol{\theta}|\boldsymbol{x}^n, y^n, \boldsymbol{k}, T)$ is exactly calculated and marginalized out. MTRF calculates both $p(T|\boldsymbol{x}^n, y^n, \boldsymbol{k})$ and $p(\boldsymbol{\theta}|\boldsymbol{x}^n, y^n, \boldsymbol{k}, T)$ exactly, but for $p(\boldsymbol{k}|\boldsymbol{x}^n, y^n)$, it only performs a heuristic approximation without any guarantees. Therefore, in this study, we propose a method to sample $\boldsymbol{k}$ using an MH algorithm where both $p(T|\boldsymbol{x}^n, y^n, \boldsymbol{k})$ and $p(\boldsymbol{\theta}|\boldsymbol{x}^n, y^n, \boldsymbol{k}, T)$ are exactly calculated and marginalized out. Therefore, the dimensionality of the distribution to be approximated by the MH algorithm is significantly reduced, and the convergence and mixing of the Markov chain will be accelerated. Furthermore, this method is able to update k_s for all the inner nodes simultaneously in a single MH step. This will also accelerate the convergence and mixing. However, updating many variables may cause a low acceptance probability in the MH step. To mitigate this negative effect, we carefully design a proposal distribution used in the MH step. Moreover, we can extend it to a REMC method to deal with multimodality of the posterior distribution. For this extension, please refer to Appendix D.

Since our method is a type of MH algorithm (Hastings, 1970), at each step t, we repeat sampling $\boldsymbol{k}^*$ from a proposal distribution $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$ and accepting $\boldsymbol{k}^*$ with the following probability:

$$\begin{aligned} &A(\boldsymbol{k}^*, \boldsymbol{k}^{(t-1)}) \\ &\coloneqq \min\left\{1, \frac{p(\boldsymbol{k}^*|\boldsymbol{x}^n, y^n) q(\boldsymbol{k}^{(t-1)}|\boldsymbol{k}^*)}{p(\boldsymbol{k}^{(t-1)}|\boldsymbol{x}^n, y^n) q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})}\right\}. \end{aligned} \qquad (3)$$

Here, we describe only the design of the proposal distribution $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$, which is the most important point of our method, and leave the proof that our method satisfies the detailed balance condition and the efficient calculation of our method to Appendices B and C.

In this study, we utilize $p(T|\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{k}^{(t-1)})$ in the design of $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$. We can exactly calculate $p(T|\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{k}^{(t-1)})$ in the same manner as MTRF by using the concept of meta-trees. Therefore, we call

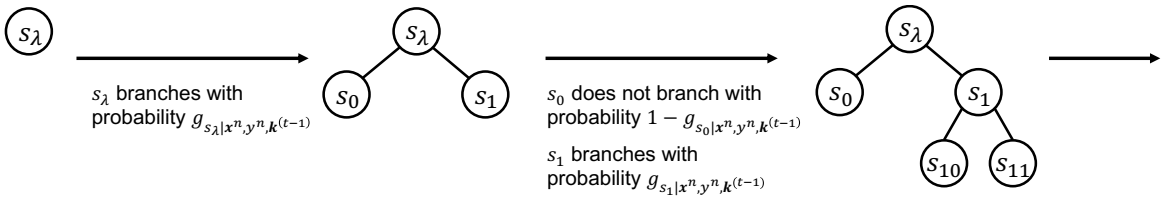


Figure 2: Node branching process representation of $p(T|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)})$ calculated by MTRF.

our sampling method Meta-tree MCMC (MTMCMC) method. The calculated $p(T|\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{k}^{(t-1)})$ has the same form as the tree distribution by Nakahara et al. (2022), and the stochastic tree generation process is represented as a node branching process from the root, as shown in Fig. 2. The posterior probability of node s branching is denoted as $g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}$. In other words, the larger $g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}$ is, the higher the posterior probability that the split by the $k_s^{(t-1)}$th explanatory variable was performed at the node s of the true model is. For such nodes, k_s^* should not be changed from $k_s^{(t-1)}$. Then, we define $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$ as follows (see also Fig. 3):

1. Starting from the root node, execute the branching process according to the posterior branching probability $g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}$ until all nodes are terminated or a pre-determined maximum depth $D_{\max}$ is reached. This generates a tree $\tilde{T}$.
2. At the inner nodes of $\tilde{T}$, set $k_s^* = k_s^{(t-1)}$.
3. At the leaf nodes of $\tilde{T}$, choose k_s^* uniformly from the feature indices other than $k_s^{(t-1)}$.
4. For descendant node of the leaf nodes of $\tilde{T}$, choose k_s^* uniformly from all feature indices.

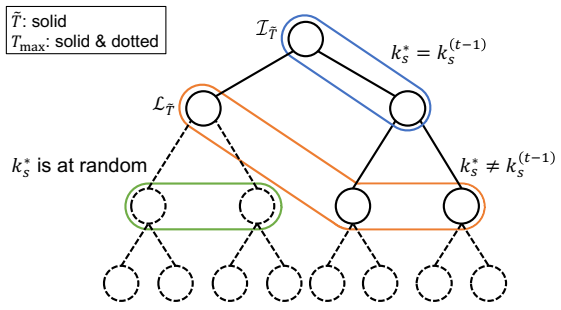


Figure 3: An example of $\tilde{T}$ and properties of $\boldsymbol{k}^*$. Here, the depth $D_{\max}$ of the maximum tree $T_{\max}$ is 3, and the set of all the inner nodes of $T_{\max}$ is represented as $\mathcal{I}_{\max}$. The tree with solid line represents $\tilde{T}$, and let $\mathcal{S}_{\tilde{T}}$, $\mathcal{I}_{\tilde{T}}$, and $\mathcal{L}_{\tilde{T}}$ denote the sets of all the nodes, all the inner nodes, and all the leaf nodes of $\tilde{T}$, respectively. For $s \in \mathcal{I}_{\tilde{T}}$, we have $k_s^* = k_s^{(t-1)}$. For $s \in \mathcal{L}_{\tilde{T}} \cap \mathcal{I}_{\max}$, we have $k_s^* \neq k_s^{(t-1)}$. For $s \in \mathcal{I}_{\max} \setminus \mathcal{S}_{\tilde{T}}$, k_s^* follows the uniform distribution on all the feature indices.

This allows updating $k_s^{(t-1)}$ for all nodes simultaneously in a single MH step. This method satisfies the detailed balance (see Appendix B). Moreover, by making slight modifications to $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$, we can significantly reduce the complexity of sampling $\boldsymbol{k}^*$ and evaluating $A(\boldsymbol{k}^*, \boldsymbol{k}^{(t-1)})$ while maintaining the detailed balance condition (see Appendix C). Then, the computational complexity of each MH step is upper bounded by $O(nD_{\max})$ (see Appendix C). Furthermore, in Appendix F, we conduct experiments on tractable examples to confirm the obtained empirical distribution converges to the true posterior distribution and to compare the convergence speed with other proposal distributions.

5 APPLICATION TO UNCERTAINTY EVALUATION

In the following, we apply the MTMCMC method to the problem of evaluating the uncertainty of a new data point. We believe that simply adding a certain width above and below the predicted value or having the output in the form like a probability distribution is not sufficient for uncertainty evaluation. Therefore, in this study, we formulate the problem of uncertainty evaluation within the framework of Bayesian decision theory. In other words, we clarify an evaluation criterion to judge what constitutes a successful uncertainty evaluation and minimize that evaluation criterion. In the following, we focus on regression problems, i.e., continuous objective variables, and describe the problem formulation under Bayesian decision theory and the optimal decision. Further, we show the results of applying it to benchmark data. Uncertainty evaluation in classification problems is in Appendix E.

There are various methods for evaluating uncertainty when the objective variable is continuous, but here we consider evaluation using prediction intervals. Note that we allow outputting multiple disconnected intervals for a single data point. Therefore, the output of the decision function $\delta(\boldsymbol{x}^n, y^n, \boldsymbol{x}_{n+1})$ is a subset C of the set of all real numbers $\mathbb{R}$. In Bayesian decision theory, a decision function represents a kind of decision-making, and basically, it is a function that takes all

given data as input and outputs some action (here, estimation of prediction intervals).

In Bayesian decision theory, an evaluation criterion is predetermined to judge the goodness of the decision function. Here, we evaluate the goodness of the output prediction intervals based on two evaluation criteria. The first is whether the true objective variable is included in the prediction intervals, and the second is the length of the prediction intervals. For the first criterion, we assume the following 0-1 loss function:[3]

$$\begin{aligned} &L_{0\text{-}1}(\boldsymbol{k}, T, \boldsymbol{\theta}, \delta(\boldsymbol{x}^n, y^n, \boldsymbol{x}_{n+1})) \\ &\coloneqq \int I\{y_{n+1} \notin \delta(\boldsymbol{x}^n, y^n, \boldsymbol{x}_{n+1})\} \\ &\qquad \times p(y_{n+1}|\boldsymbol{x}_{n+1}, \boldsymbol{k}, T, \boldsymbol{\theta}) \mathrm{d}y_{n+1}, \end{aligned} \tag{4}$$

where $I\{\cdot\}$ denotes the indicator function. However, this function depends on the data and unknown parameters $(\boldsymbol{k}, T, \boldsymbol{\theta})$, so it cannot be directly used as an evaluation function for the decision function δ.

Therefore, we use the Bayes risk function $BR(\delta)$ as the evaluation criterion for δ. The Bayes risk function $BR(\delta)$ is defined as the expectation of the loss function with respect to the prior distribution of the data[4] and unknown parameters in the following manner.

$$\begin{aligned} BR_{0\text{-}1}(\delta) \coloneqq &\sum_{\boldsymbol{k}} \sum_{T} \iint p(\boldsymbol{k}, T, \boldsymbol{\theta}) p(y^n|\boldsymbol{x}^n, \boldsymbol{k}, T, \boldsymbol{\theta}) \\ &\times L_{0\text{-}1}(\boldsymbol{k}, T, \boldsymbol{\theta}, \delta(\boldsymbol{x}^n, y^n, \boldsymbol{x}_{n+1})) \mathrm{d}y^n \mathrm{d}\boldsymbol{\theta}. \end{aligned} \tag{5}$$

It is known that the decision function δ that minimizes $BR(\delta)$ is equivalent to the decision function δ that minimizes the posterior expected loss. If we simply want to minimize $BR(\delta)$ or the posterior expected loss, we can output a sufficiently large interval, such as $\mathbb{R}$, as the prediction interval. However, it is clear that such uncertainty evaluation is meaningless. Therefore, we define the optimal decision function as the one that minimizes the length of the prediction intervals, which is the second evaluation criterion, while keeping the posterior expected loss at a certain value of $100(1-\alpha)\%$.

In this case, as a general result of Bayesian decision theory, it is known that the optimal decision function is given by the highest posterior predictive density credible intervals (Berger, 1985). That is, let $C(l)$ be the set of y_{n+1} such that the density of the posterior predictive distribution is greater than l, i.e., $C(l) \coloneqq \{y_{n+1} \in \mathbb{R} \mid p(y_{n+1}|\boldsymbol{x}^n, y^n, \boldsymbol{x}_{n+1}) > l\}$, and let l^* be the largest l that satisfies $\Pr\{C(l)\} \geq 1-\alpha$. Then, $C(l^*)$ becomes the output of the optimal decision function. In general, since the posterior predictive distribution $p(y_{n+1}|\boldsymbol{x}^n, y^n, \boldsymbol{x}_{n+1})$ can be multimodal, the optimal prediction intervals may consist of multiple disjoint intervals. If a continuous interval is preferred for practical reasons, an equal-tailed posterior prediction interval can be used, as in conventional Bayesian statistical models (Berger, 1985). However, please note that this will result in a wider prediction interval.

In this study, we approximate the posterior predictive distribution required for calculating the highest posterior predictive density credible intervals by using the MTMCMC method as follows:

$$\begin{aligned} &p(y_{n+1}|\boldsymbol{x}^n, y^n, \boldsymbol{x}_{n+1}) \\ &\approx \frac{1}{t_{\mathrm{end}} - t_{\mathrm{start}}} \sum_{t=t_{\mathrm{start}}}^{t_{\mathrm{end}}} \sum_{T} p(T|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t)}) \\ &\quad \times \int p(y_{n+1}|\boldsymbol{x}_{n+1}, \boldsymbol{k}^{(t)}, T, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{x}^n, y^n, T, \boldsymbol{k}^{(t)}) \mathrm{d}\boldsymbol{\theta}, \end{aligned} \tag{6}$$

where t_{start} and t_{end} represent the length of the burn-in and the total number of sampling iterations, respectively. Note that the right-hand side of the above equation converges to the left-hand side under a sufficiently large number of sampling iterations since the MH method described in the previous section satisfies the detailed balance condition.

In a numerical experiment, we calculated 95% prediction intervals for various benchmark datasets (Dua and Graff, 2017; Pedregosa et al., 2011; Meyer, 1989) and compared them with some state-of-the-art methods, as shown in Table 1. See Appendix I for details of the datasets. When the leaf model in our model is the LR model, we calculated the highest posterior predictive density credible intervals by two methods. One was based on the value of probability density function approximated by Eq. (6). The other was based on a sample of y_{n+1} obtained from Eq. (6). Please refer to Appendix I for more details. Since our model assumes a single tree model, we set the option to specify the number of trees in BART to `ntree=1`. Then, except for the heteroscedasticity mentioned in Section 2, the model of BART becomes almost equivalent to our model when our leaf node model is a normal distribution. Furthermore, we have aligned the values of the hyperparameters of the prior distribution as much as possible (see Appendix I). Additionally, we also included experimental results using BART with the default settings assuming a sum-of-trees model for reference. Note that we used the interval from the 0.025% quantile to the 0.975% quantile as the prediction inter-

[3] This is the expectation of usual 0-1 loss in machine learning with respect to a new data point.

[4] Throughout this paper, we consider the explanatory variables are given constants and not a random variable. So, we do not take the expectation for them.

Table 1: Average frequency that the prediction interval contains the true y_{n+1}, average length of the prediction interval, and their standard errors (SE).

Avg. frequency (SE) Avg. length (SE)	LightGBM	XGBoost	GBDT	BART `ntree=1`	MTMCMC Leaf model: Normal	MTMCMC Leaf model: LR model	MTMCMC Leaf model: LR model (sampling)	BART
automobile (n:159, dim:57)	0.911 (0.024) 3.111 (0.107)	0.914 (0.025) 3.047 (0.126)	0.850 (0.021) 3.093 (0.131)	0.433 (0.035) 0.926 (0.055)	0.992 (0.004) 3.263 (0.034)	**0.938** (0.017) 3.163 (0.137)	**0.943** (0.016) 3.276 (0.139)	0.706 (0.022) 1.234 (0.029)
servo (n:167, dim:12)	**0.908** (0.011) 1.938 (0.221)	0.839 (0.021) 2.025 (0.161)	0.869 (0.016) 2.015 (0.162)	0.558 (0.045) 0.662 (0.092)	**0.990** (0.005) 2.204 (0.066)	0.992 (0.004) 2.215 (0.064)	0.993 (0.004) 2.302 (0.065)	0.699 (0.013) 0.934 (0.023)
cpu (n:209, dim:36)	**0.935** (0.010) 3.888 (0.204)	0.756 (0.022) 0.938 (0.078)	0.850 (0.016) 1.333 (0.150)	0.428 (0.029) 0.427 (0.022)	0.983 (0.005) 2.028 (0.102)	**0.975** (0.008) 1.650 (0.138)	0.977 (0.008) 1.692 (0.141)	0.672 (0.030) 0.586 (0.031)
liver (n:345, dim:5)	0.917 (0.023) 3.270 (0.114)	0.898 (0.018) 2.785 (0.098)	0.833 (0.037) 2.899 (0.085)	0.128 (0.010) 0.522 (0.023)	**0.950** (0.005) 3.632 (0.062)	0.943 (0.006) 3.703 (0.070)	**0.949** (0.006) 3.869 (0.070)	0.282 (0.017) 0.833 (0.018)
Mpg (n:392, dim:11)	0.854 (0.011) 1.379 (0.052)	0.777 (0.027) 1.683 (0.113)	0.897 (0.013) 2.117 (0.122)	0.474 (0.020) 0.643 (0.031)	0.979 (0.005) 2.181 (0.028)	**0.955** (0.009) 1.562 (0.028)	**0.962** (0.007) 1.636 (0.030)	0.765 (0.013) 0.738 (0.008)
student (n:395, dim:43)	0.865 (0.012) 3.098 (0.077)	0.829 (0.024) 2.863 (0.138)	0.901 (0.013) 3.276 (0.071)	0.189 (0.015) 0.500 (0.035)	0.967 (0.005) 3.784 (0.024)	**0.950** (0.005) 3.643 (0.021)	**0.959** (0.005) 3.819 (0.022)	0.522 (0.012) 1.475 (0.016)
diabetes (n:442, dim:10)	0.832 (0.010) 2.303 (0.035)	0.674 (0.016) 1.996 (0.043)	0.878 (0.011) 2.677 (0.061)	0.251 (0.012) 0.679 (0.034)	0.970 (0.004) 3.251 (0.037)	**0.957** (0.004) 2.832 (0.032)	**0.967** (0.003) 2.971 (0.034)	0.576 (0.007) 1.211 (0.013)
boston (n:506, dim:13)	0.882 (0.010) 3.222 (0.243)	0.800 (0.021) 3.172 (0.291)	0.880 (0.013) 2.988 (0.313)	0.331 (0.025) 0.501 (0.026)	0.980 (0.003) 2.270 (0.052)	**0.965** (0.004) 1.631 (0.052)	**0.969** (0.003) 1.701 (0.054)	0.852 (0.007) 0.864 (0.023)
cps (n:534, dim:19)	0.845 (0.012) 2.696 (0.050)	0.718 (0.021) 1.903 (0.055)	0.891 (0.010) 2.765 (0.061)	0.211 (0.016) 0.507 (0.037)	**0.954** (0.004) 3.415 (0.033)	0.943 (0.005) 3.329 (0.044)	**0.948** (0.004) 3.470 (0.045)	0.572 (0.022) 1.131 (0.034)
strikes (n:625, dim:23)	0.820 (0.011) 2.012 (0.060)	0.703 (0.016) 1.182 (0.050)	0.887 (0.009) 1.905 (0.061)	0.198 (0.019) 0.344 (0.022)	0.953 (0.005) 2.608 (0.049)	**0.949** (0.004) 2.551 (0.052)	**0.952** (0.004) 2.643 (0.053)	0.851 (0.012) 1.289 (0.024)

val of deterministic methods since the highest posterior predictive density credible intervals cannot be defined for deterministic decision trees. Other hyperparameters are described in Appendix I.

We repeated 2-fold cross-validation 10 times and measured how frequently the test data was included in the output prediction intervals and the length of the prediction intervals for the test data. The number of splits in cross-validation was kept small to more accurately evaluate the generalization performance for a small sample, which is an advantage of the optimal decision based on Bayesian decision theory.

Each cell in Table 1 shows the mean and standard error of those values. The frequency closest to 0.95 among the methods for each dataset is shown in bold with a shaded background, and the second closest frequency is shown in bold. We can see that the frequency of our method is very close to 95%. Moreover, the lengths of the prediction intervals are simultaneously shorter for some datasets. Note that BART, which assumes a model represented by more trees than our model, often outputs prediction intervals that are too narrow. Regarding the computational cost, using Python on a laptop detailed in Appendix J, each MH step on a single chain required approximately 30 msec for boston data, where $D_{\max} = 10$.

The key insights obtained from this experiment are as follows. Our model with normal distributions at leaf nodes shows significant improvements compared to BART with `ntree=1`. As mentioned earlier, since both models are almost equivalent, this difference is likely due to the efficiency of the MCMC method. As pointed out by Chipman et al. (1998), BART with `ntree=1` may struggle with multimodality of the posterior distributions. In contrast, MTMCMC potentially addresses this issue through exact marginalization of T, simultaneous updating of multiple k_s while maintaining acceptance ratio, and the extension to the REMC methods. Another possibility is that it could be due to the heteroscedasticity of our model, which is also one of the advantages of our model. Furthermore, by setting the leaf node models as LR models, the performance of our model is further improved. This flexibility in the leaf node models is another advantage of our model.

6 STATISTICALLY INTERPRETABLE FEATURE IMPORTANCE

The ability to calculate the importance of each feature is considered as an advantage of conventional decision tree methods, see, e.g., the implementation in scikit-learn (Pedregosa et al., 2011). However, such a feature importance often lacks a clear statistical interpretation. In contrast, this study enables the calculation of feature importance with clearer statistical meaning. This is a byproduct of treating decision trees as stochastic models and being able to compute the posterior distribution of their unknown parameters. Although various definitions can be considered, we describe an example here. We define the importance $F(k)$ of the kth explanatory variable as follows.

First, we define $\boldsymbol{x}_{s,\boldsymbol{k}}$ as the set of data points that pass through s for given $\boldsymbol{k}$. Mathematically, $\boldsymbol{x}_{s,\boldsymbol{k}}$ is represented as $\{\boldsymbol{x}_i\}_{i:s\preceq s_{T_{\max},\boldsymbol{k}}(\boldsymbol{x}_i)}$, where $s_{T_{\max},\boldsymbol{k}}(\boldsymbol{x}_i)$ represents the leaf node that $\boldsymbol{x}_i$ reaches on the maximum tree $T_{\max}$ with $\boldsymbol{k}$ and $s \prec s'$ means s is an ancestor node of s'. In a similar manner, we define $y_{s,\boldsymbol{k}} \coloneqq \{y_i\}_{i:s\preceq s_{T_{\max},\boldsymbol{k}}(\boldsymbol{x}_i)}$. In addition, let $p(y_{s,\boldsymbol{k}}|\boldsymbol{x}_{s,\boldsymbol{k}},\boldsymbol{\theta}_s)$ denote $\prod_{i:s\preceq s_{T_{\max},\boldsymbol{k}}(\boldsymbol{x}_i)} p(y_i|\boldsymbol{x}_i,\boldsymbol{\theta}_s)$. Then, our feature importance metric is defined as follows:

$$F(k) \coloneqq \sum_{s\in\mathcal{S}_{\max}} \mathbb{E}_{p(\boldsymbol{k}',T|\boldsymbol{x}^n,y^n)}\Bigg[I\{s \text{ is inner node of } T\} \times I\{k'_s = k\}\log\frac{\prod_{s_{\mathrm{ch}}\in\mathrm{Ch}(s)} f(y_{s_{\mathrm{ch}},\boldsymbol{k}'}|\boldsymbol{x}_{s_{\mathrm{ch}},\boldsymbol{k}'},s_{\mathrm{ch}},\boldsymbol{k}')}{f(y_{s,\boldsymbol{k}'}|\boldsymbol{x}_{s,\boldsymbol{k}'},s,\boldsymbol{k}')}\Bigg], \tag{7}$$

where $\mathrm{Ch}(s)$ denotes the set of the child nodes of s and $f(y_{s,\boldsymbol{k}'}|\boldsymbol{x}_{s,\boldsymbol{k}'},s,\boldsymbol{k}')$ is defined as follows:

$$f(y_{s,\boldsymbol{k}'}|\boldsymbol{x}_{s,\boldsymbol{k}'},s,\boldsymbol{k}') \coloneqq \int p(y_{s,\boldsymbol{k}'}|\boldsymbol{x}_{s,\boldsymbol{k}'},\boldsymbol{\theta}_s)p(\boldsymbol{\theta}_s)\mathrm{d}\boldsymbol{\theta}_s. \tag{8}$$

Here, $f(y_{s,\boldsymbol{k}'}|\boldsymbol{x}_{s,\boldsymbol{k}'},s,\boldsymbol{k}')$ represents the marginal likelihood when the node s is a leaf node and all y_i for the data passing through the node s for given $\boldsymbol{k}'$ are generated from s. For example, when assuming the leaf node model is a normal distribution, this value becomes large if the histogram of $y_{s,\boldsymbol{k}'}$ is unimodal and symmetric, i.e., it is likely that all the y_i in $y_{s,\boldsymbol{k}'}$ were generated from a single normal distribution. In contrast, if the histogram has two peaks, the value of $f(y_{s,\boldsymbol{k}'}|\boldsymbol{x}_{s,\boldsymbol{k}'},s,\boldsymbol{k}')$ for node s would be small, while the values for s's child nodes would likely be large.

The advantage of this feature importance is that it can be defined in the same way regardless of the distribution $p(y_i|\boldsymbol{x}_i,\boldsymbol{\theta}_{s_{\boldsymbol{k},T}(\boldsymbol{x}_i)})$ at leaf nodes. It can be used in binary classification problems assuming a Bernoulli distribution and can also be used similarly in regression problems assuming a LR model. When calculating this feature importance, the expectation with respect to $p(T|\boldsymbol{x}^n,y^n,\boldsymbol{k}')$ is calculated exactly, and the expectation with respect to $p(\boldsymbol{k}'|\boldsymbol{x}^n,y^n)$ is approximated by the sample mean over the samples obtained by the MTMCMC method.

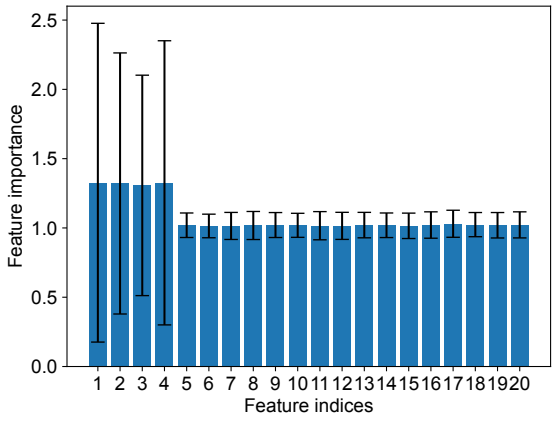


Figure 4: Average of $\exp(F(k)/n)$ and their standard deviations for each k.

The usefulness of this feature importance is confirmed through experiments on synthetic data. In this experiment, $\boldsymbol{x}_i$ is 20-dimensional and generated according to a uniform distribution on $[-3, 3]$ independently for $i = 1, 2, \ldots, 200$. Next, $(\boldsymbol{k}, T, \boldsymbol{\theta})$ are generated according to the prior distribution, where the explanatory variables used in the inner nodes are limited to the first 4 of the 20 elements. Then, y^n is generated. Here, the stochastic model of the leaf nodes is assumed to be a normal distribution. Finally, the feature importance $F(k)$ is calculated from $\boldsymbol{x}^n$ and y^n. This operation is repeated 500 times. Figure 4 shows the average of $\exp(F(k)/n)$ for these 500 experiments. Note that the transformation $\exp(\cdot/n)$ is simply for making the graph of the average of 500 experiments easier to view and is not necessary when comparing the feature importance within a single sample. We can see the $F(k)$ values of the first 4 explanatory variables are large from Fig. 4.

7 CONCLUSION

In this study, we defined a stochastic model based on decision trees and evaluated the uncertainty of a new data point within the framework of Bayesian decision theory. Since our model has three types of parameters: tree shape, inner node parameters, and leaf node parameters, it is necessary to compute the posterior distribution for all of these to calculate the optimal decision under Bayesian decision theory. Therefore, we proposed a new MH algorithm that accelerates the convergence and mixing of the Markov chain. Our algorithm exactly calculates the posterior distribution of the leaf node parameters and tree shape and approximates only the posterior distribution of the inner node parameters by sampling. Furthermore, in this MH algorithm, all inner node parameters can be updated simultaneously in a single MH step. We believe that our method has a broader impact to accelerate the inference algorithm for various machine learning models with tree structures by eliminating the need for sampling of tree structures in MCMC methods. We also compared this method with state-of-the-art methods on various benchmark datasets. Additionally, taking advantage of the fact that our decision tree represents a stochastic model, we proposed a feature importance with clear statistical meaning.

Acknowledgments

This work was supported in part by JSPS KAKENHI Grant Numbers JP22K02811, JP22K14254, JP23H00468, JP23K03863, JP23K04293, and JP23K11097.

References

Berger, J. O. (1985). *Statistical Decision Theory and Bayesian Analysis*. Springer New York, New York, NY.

Berndt, E. (1991). *The Practice of Econometrics: Classic and Contemporary*. Addison-Wesley Publishing Company.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer.

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

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and Regression Trees*. CRC press.

Cason, T. (1999). Titanic data (titanic3). Vanderbilt Biostatistics Datasets. `https://hbiostat.org/data/repo/titanic.html`, Accessed: March 10, 2025.

Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. Association for Computing Machinery.

Chipman, H. A., George, E. I., and McCulloch, R. E. (1998). Bayesian cart model search. *Journal of the American Statistical Association*, 93(443):935–948.

Chipman, H. A., George, E. I., and McCulloch, R. E. (2010). BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266 – 298.

Cortez, P. (2014). Student Performance. UCI Machine Learning Repository. DOI: `https://doi.org/10.24432/C5TG7T`.

Dobashi, N., Saito, S., Nakahara, Y., and Matsushima, T. (2021). Meta-tree random forest: Probabilistic data-generative model and Bayes optimal prediction. *Entropy*, 23(6).

Dua, D. and Graff, C. (2017). UCI machine learning repository. http://archive.ics.uci.edu/ml.

Feldmesser, J. (1987). Computer Hardware. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5830D.

Fisher, R. A. (1988). Iris. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C56C76.

German, B. (1987). Glass Identification. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5WW2P.

Ghahramani, Z., Jordan, M., and Adams, R. P. (2010). Tree-structured stick breaking for hierarchical data. In Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., and Culotta, A., editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc.

Harrison, D. and Rubinfeld, D. L. (1978). Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109.

Hofmann, H. (1994). Statlog (German Credit Data). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5NC77.

Jordan, M. I. and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154.

Matsushima, T. and Hirasawa, S. (1994). A Bayes coding algorithm using context tree. In *1994 IEEE International Symposium on Information Theory*, page 386.

Matsushima, T., Inazumi, H., and Hirasawa, S. (1991). A class of distortionless codes designed by Bayes decision theory. *IEEE Transactions on Information Theory*, 37(5):1288–1293.

Meyer, M. (1989). StatLib—Datasets Archive. http://lib.stat.cmu.edu/datasets/, Accessed: March 10, 2025.

Nakahara, Y., Ichijo, N., Shimada, K., Iikubo, Y., Saito, S., Kazama, K., Matsushima, T., and BayesML Developers (2025). BayesML. Python package version 0.3.0. https://github.com/bayesml/BayesML.

Nakahara, Y. and Matsushima, T. (2024). Batch updating of a posterior tree distribution over a meta-tree. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E107.A(3):523–525.

Nakahara, Y., Saito, S., Kamatsuka, A., and Matsushima, T. (2022). Probability distribution on full rooted trees. *Entropy*, 24(3).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Quinlan, R. (1993). Auto MPG. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5859H.

Schlimmer, J. (1987). Automobile. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5B01C.

Sejnowski, T. and Gorman, R. (1988). Connectionist Bench (Sonar, Mines vs. Rocks). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5T01Q.

Sigillito, V., Wing, S., Hutton, L., and Baker, K. (1989). Ionosphere. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5W01B.

Sparapani, R., Spanbauer, C., and McCulloch, R. (2021). Nonparametric machine learning and efficient computation with Bayesian additive regression trees: The BART R package. *Journal of Statistical Software*, 97(1):1–66.

Suko, T., Nomura, R., Matsushima, T., and Hirasawa, S. (2003). Prediction algorithm for decision tree model. *IEICE technical report. Theoretical foundations of Computing*, 103:93–98. (in Japanese).

Swendsen, R. H. and Wang, J.-S. (1986). Replica monte carlo simulation of spin-glasses. *Phys. Rev. Lett.*, 57:2607–2609.

Ulrich, K. (1993). Servo. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5Q30F.

Western, B. (1996). Vague theory and model uncertainty in macrosociology. *Sociological Methodology*, 26:165–192.

Wolberg, W., Mangasarian, O., Street, N., and Street, W. (1995). Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5DW2B.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Appendix

A FORMAL DEFINITION OF STOCHASTIC MODELS

A.1 Basic Notations

Let the dimension of the continuous features be $p \in \mathbb{N} \coloneqq \{1, 2, 3, \ldots\}$. Let the dimension of the binary features be $q \in \mathbb{N}$. Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_p, x_{p+1}, \ldots, x_{p+q}) \in \mathbb{R}^p \times \{0,1\}^q \subset \mathbb{R}^{p+q}$ be an explanatory variable, where $x_1, \ldots, x_p$ take continuous values and $x_{p+1}, \ldots, x_{p+q}$ take binary values. Further, $\mathcal{Y}$ denotes a set of possible values of objective variables. Our discussion can be applied to both a discrete set (e.g., $\mathcal{Y} = \{0,1\}$) and a continuous set (e.g., $\mathcal{Y} = \mathbb{R}$). Let Y be a random variable taking values in $\mathcal{Y}$ and $y \in \mathcal{Y}$ be a realization of Y.

Regarding a tree, we use the following notations. See also Fig. 5. Let $D_{\max} \in \mathbb{N}$ be the maximum depth of trees. The perfect binary tree whose depth is $D_{\max}$ is denoted by $T_{\max}$. Here, the perfect binary tree means the tree where all the inner nodes have exactly two children and all the leaf nodes have the same depth. The set of all the nodes of $T_{\max}$ is denoted by $\mathcal{S}_{\max}$. The set $\mathcal{S}_{\max}$ can be divided into two disjoint subsets: $\mathcal{L}_{\max} \subset \mathcal{S}_{\max}$ and $\mathcal{I}_{\max} \subset \mathcal{S}_{\max}$, where $\mathcal{L}_{\max}$ is the set of all the leaf nodes of $T_{\max}$ and $\mathcal{I}_{\max}$ is the set of all the inner nodes of $T_{\max}$. In this paper, we consider a rooted tree, i.e., a tree that has a root node $s_\lambda \in \mathcal{S}_{\max}$. Let T be a full (also called proper) subtree of $T_{\max}$, where T's root node is s_λ and all the inner nodes have exactly two children. The set of all the nodes of T is denoted by $\mathcal{S}_T \subset \mathcal{S}_{\max}$. It can be divided into $\mathcal{L}_T \subset \mathcal{S}_T$ and $\mathcal{I}_T \subset \mathcal{S}_T$, where $\mathcal{L}_T$ is the set of all the leaf nodes of T and $\mathcal{I}_T$ is the set of all the inner nodes of T. The set of all full subtrees T is denoted by $\mathcal{T}$. As we will describe later in detail, a feature index $k_s \in \{1, 2, \ldots, p+q\}$ is assigned to an inner node $s \in \mathcal{I}_{\max}$, and a feature assignment vector is denoted by $\boldsymbol{k} \coloneqq (k_s)_{s \in \mathcal{I}_{\max}} \in \mathcal{K} \coloneqq \{1, 2, \ldots, p+q\}^{|\mathcal{I}_{\max}|}$. Further, we assume a node $s \in \mathcal{S}_{\max}$ has a parameter $\boldsymbol{\theta}_s$, and let $\boldsymbol{\theta}$ denote $(\boldsymbol{\theta}_s)_{s \in \mathcal{S}_{\max}}$. Lastly, the set of $\boldsymbol{\theta}$ is denoted by $\boldsymbol{\Theta}$.

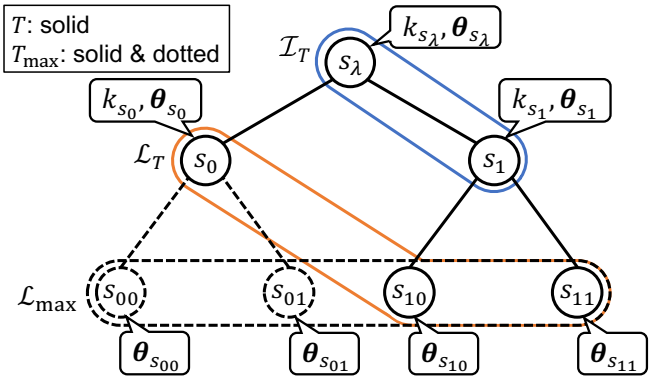


Figure 5: The basic notations for a binary tree. Here, $D_{\max}$ is 2.

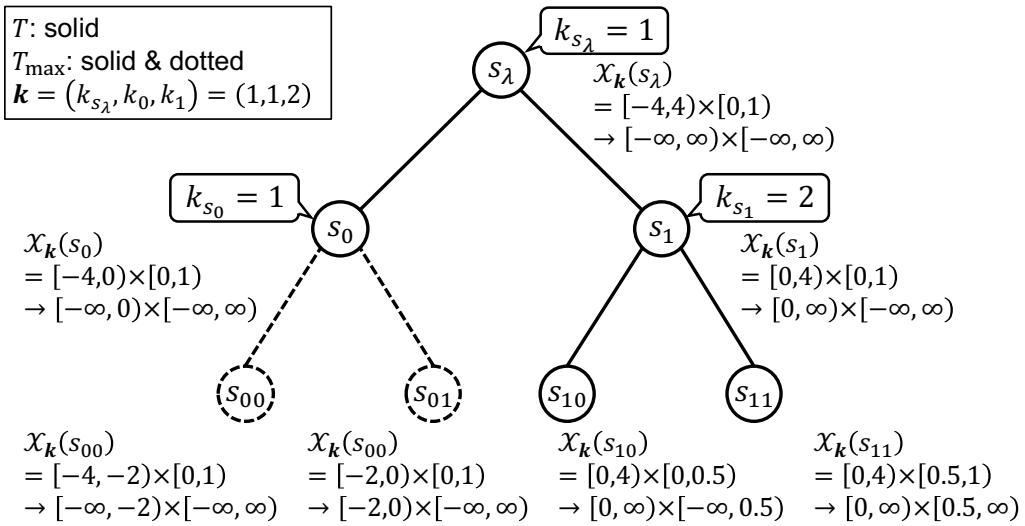


Figure 6: An example of subspace division procedure. Here, $D_{\max} = 2$, $p = 1$, $q = 1$, and $\boldsymbol{k} = (k_{s_\lambda}, k_{s_0}, k_{s_1}) = (1, 1, 2)$. First, the root node s_λ has a subspace $\mathcal{X}_{\boldsymbol{k}}(s_\lambda) = [a_{1,s_\lambda}, b_{1,s_\lambda}) \times [a_{2,s_\lambda}, b_{2,s_\lambda}) = [-4, 4) \times [0, 1)$. Next, $\mathcal{X}_{\boldsymbol{k}}(s_\lambda)$ is divided into $\mathcal{X}_{\boldsymbol{k}}(s_0)$ and $\mathcal{X}_{\boldsymbol{k}}(s_1)$. Its threshold is a midpoint of a_{1,s_λ} and b_{1,s_λ} because $k_{s_\lambda} = 1$. Similarly, $\mathcal{X}_{\boldsymbol{k}}(s_0)$ and $\mathcal{X}_{\boldsymbol{k}}(s_1)$ are divided into $\mathcal{X}_{\boldsymbol{k}}(s_{00})$, $\mathcal{X}_{\boldsymbol{k}}(s_{01})$, $\mathcal{X}_{\boldsymbol{k}}(s_{10})$, and $\mathcal{X}_{\boldsymbol{k}}(s_{11})$. After that, temporary minimum and maximum values are replaced with $-\infty$ and ∞, respectively. As a result, $\bigcup_{s\in\mathcal{L}_T} \mathcal{X}_{\boldsymbol{k}}(s) = \mathbb{R}^{p+q}$ holds for any $T \in \mathcal{T}$, e.g., if T is a tree represented with solid lines, then $\mathcal{X}_{\boldsymbol{k}}(s_0) \cup \mathcal{X}_{\boldsymbol{k}}(s_{10}) \cup \mathcal{X}_{\boldsymbol{k}}(s_{11}) = \mathbb{R}^2$ since $\mathcal{L}_T = \{s_0, s_{10}, s_{11}\}$.

A.2 Stochastic Data Observation Process

We assume the following probability distribution on an objective variable y given an explanatory variable $\boldsymbol{x}$. Note that $\boldsymbol{\theta}$, T, and $\boldsymbol{k}$ are unobservable parameters and their posterior distribution should be calculated later in a Bayesian manner. First, we define the following subspace division procedure and a leaf node corresponding to a given explanatory variable. Note that this procedure is not a tree construction method from given data but a definition of stochastic data observation process behind the given data.

Definition 1 ($\mathcal{X}_{\boldsymbol{k}}(s)$ and $s_{\boldsymbol{k},T}(\boldsymbol{x})$)**.** Given $\boldsymbol{k}$, let $\mathcal{X}_{\boldsymbol{k}}(s)$ for $s \in \mathcal{S}_{\max}$ denote a subspace of $\mathbb{R}^{p+q}$, which is recursively defined in the following manner (see also Fig. 6).

First, for the root node s_λ, we assume

$$\mathcal{X}_{\boldsymbol{k}}(s_\lambda) = [a_{1,s_\lambda}, b_{1,s_\lambda}) \times \cdots \times [a_{p+q,s_\lambda}, b_{p+q,s_\lambda}). \tag{9}$$

Here, $a_{k,s_\lambda}, b_{k,s_\lambda} \in \mathbb{R}$ are just initial values to determine thresholds and they do not restrict the acceptable range of the features. At this point, we use temporary minimum and maximum values for continuous features and $a_{k,s_\lambda} = 0$ and $b_{k,s_\lambda} = 1$ for binary features.

Next, if the following holds for any inner node $s \in \mathcal{I}_{\max}$,

$$\mathcal{X}_{\boldsymbol{k}}(s) = [a_{1,s}, b_{1,s}) \times \cdots \times [a_{p+q,s}, b_{p+q,s}), \tag{10}$$

then the subspace assigned to the left child s_l and the right child s_r of s is defined as follows, based on the feature index k_s assigned to s.

$$\mathcal{X}_{\boldsymbol{k}}(s_l) = \{\boldsymbol{x} \in \mathcal{X}_{\boldsymbol{k}}(s) \mid a_{k_s,s} \le x_{k_s} < t_{k_s,s}\}, \tag{11}$$
$$\mathcal{X}_{\boldsymbol{k}}(s_r) = \{\boldsymbol{x} \in \mathcal{X}_{\boldsymbol{k}}(s) \mid t_{k_s,s} \le x_{k_s} < b_{k_s,s}\}. \tag{12}$$

Here, we have many possible ways to define the threshold $t_{k_s,s}$. For example, we can use the midpoint of the assigned subspace, i.e., $t_{k_s,s} \coloneqq (a_{k_s,s} + b_{k_s,s})/2$. Other ideas are described in Appendix G.

Lastly, we replace $a_{k,s}$ with $-\infty$ for any $k \in \{1, \ldots, p+q\}$ and $s \in \mathcal{S}_{\max}$ such that $a_{k,s} = a_{k,s_\lambda}$. Similarly, we replace $b_{k,s}$ with ∞ for any $k \in \{1, \ldots, p+q\}$ and $s \in \mathcal{S}_{\max}$ such that $b_{k,s} = b_{k,s_\lambda}$.

By this procedure, each $s \in \mathcal{S}_{\max}$ is assigned to a subspace of $\mathbb{R}^{p+q}$ and the following holds: for any $T \in \mathcal{T}$, $\bigcup_{s \in \mathcal{L}_T} \mathcal{X}_{\boldsymbol{k}}(s) = \mathbb{R}^{p+q}$, and for any $s, s' \in \mathcal{L}_T$, $s \neq s' \Rightarrow \mathcal{X}_{\boldsymbol{k}}(s) \cap \mathcal{X}_{\boldsymbol{k}}(s') = \emptyset$. Therefore, given $\boldsymbol{k}$ and T, for any $\boldsymbol{x} \in \mathbb{R}^{p+q}$, we can uniquely determine a node $s \in \mathcal{L}_T$ such that $\boldsymbol{x} \in \mathcal{X}_{\boldsymbol{k}}(s)$ holds. Let $s_{\boldsymbol{k},T}(\boldsymbol{x})$ represents this node.

Using the above notation, we impose the following assumptions on the probability distribution of an objective variable y given an explanatory variable $\boldsymbol{x}$.

Assumption 1. Given $\boldsymbol{k}$ and T, let $s_{\boldsymbol{k},T}(\boldsymbol{x}) \in \mathcal{L}_T$ denote the leaf node defined in Def. 1, which is uniquely and deterministically obtained from the explanatory variable $\boldsymbol{x}$. Then, we assume

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}, T, \boldsymbol{k}) = p(y|\boldsymbol{x}, \boldsymbol{\theta}_{s_{\boldsymbol{k},T}(\boldsymbol{x})}). \tag{13}$$

That is, we assume that y is independent of any other parameter than that assigned to $s_{\boldsymbol{k},T}(\boldsymbol{x})$.

A.3 Prior Distributions

Assumption 2. We assume each element $\boldsymbol{\theta}_s$ is independent and identically distributed with a conjugate prior distribution $p(\boldsymbol{\theta}_s)$ for $p(y|\boldsymbol{x}, \boldsymbol{\theta}_s)$, i.e.,

$$p(\boldsymbol{\theta}) = \prod_{s \in \mathcal{S}_{\max}} p(\boldsymbol{\theta}_s). \tag{14}$$

In addition, we assume we can calculate its predictive distribution $p(y) = \int p(y|\boldsymbol{x}, \boldsymbol{\theta}_s) p(\boldsymbol{\theta}_s) \mathrm{d}\boldsymbol{\theta}_s$ with an acceptable cost.

The following examples fulfill the above assumptions.

Example 1. For example, when $\mathcal{Y}$ is finite, we can assume the categorical distribution $\mathrm{Cat}(y|\boldsymbol{\pi}_s)$ and the Dirichlet prior $\mathrm{Dir}(\boldsymbol{\pi}_s|\boldsymbol{\alpha})$. When y is a count data, i.e., $\mathcal{Y} = \{0, 1, \dots\}$, we can assume the Poisson distribution $\mathrm{Po}(y|\nu_s)$ and the gamma prior $\mathrm{Gam}(\nu_s|\alpha, \beta)$. When $\mathcal{Y}$ is continuous, we can assume the normal distribution $\mathcal{N}(y|\mu_s, \sigma_s^2)$ and the normal-gamma prior $\mathcal{N}(\mu_s|m, \gamma\sigma_s^2)\mathrm{Gam}(1/\sigma_s^2|\alpha, \beta)$. Further, we can also assume a more complicated model, e.g., linear regression (LR) model $\mathcal{N}(y|\boldsymbol{w}_s^\top \boldsymbol{x}, \sigma_s^2)$ and the normal-gamma prior $\mathcal{N}(\boldsymbol{w}_s|\boldsymbol{m}, \sigma_s^2\boldsymbol{\Sigma})\mathrm{Gam}(1/\sigma_s^2|\alpha, \beta)$, as long as it satisfies Assumption 2. This flexibility is one of advantages of our model. Here, $\boldsymbol{\pi}_s$, ν_s, (μ_s, σ_s^2), and $(\boldsymbol{w}_s, \sigma_s^2)$ correspond to $\boldsymbol{\theta}_s$ in the general notation.

We assume the following prior distribution of $T \in \mathcal{T}$, which was first proposed for text compression in information theory by Matsushima and Hirasawa (1994), then applied to decision trees by Dobashi et al. (2021), and mathematically summarized by Nakahara et al. (2022).

Assumption 3. Given $D_{\max}$, we assume the following probability distribution on the set $\mathcal{T}$ of full trees T, which are subtrees of the perfect binary tree $T_{\max}$ whose depth is $D_{\max}$:

$$p(T) \coloneqq \prod_{s \in \mathcal{I}_T} g_s \prod_{s' \in \mathcal{L}_T} (1 - g_{s'}), \tag{15}$$

where $g_s \in [0, 1]$ is a given hyperparameter representing an branching probability of a node s. For $s \in \mathcal{L}_{\max}$, we assume $g_s = 0$.

Properties of this distribution are discussed by Nakahara et al. (2022), e.g., Eq. (15) satisfies $\sum_{T \in \mathcal{T}} p(T) = 1$.

Example 2. Figure 7 shows an example of $p(T)$. The hyperparameter g_s represents the branching probability under the condition that all the ancestor nodes of s extend their edges. In other words, the data observation process includes the explanatory variable x_{k_s} with the prior probability g_s under the condition that it includes all the explanatory variables assigned to the ancestor nodes of s (Nakahara et al., 2022, Remark 2). Therefore, the prior probability that an explanatory variable on a node is included in the model decreases exponentially with its depth. In the learning phase, this property of the prior distribution prevents overfitting.

Remark 1. The node branching parameter g_s in Eq. (15) in Assumption 3 plays the same role as p_{SPLIT} in BayesianCART. Therefore, by setting $g_s = \alpha(1 + d_s)^\beta$ and assuming the maximum depth, our prior distribution coincides with the prior distribution in BayesianCART, where d_s is the depth of node s. This relationship allowed us to conduct a fairer numerical experiment in Section 5 and Appendix E. While the BayesianCART prior is

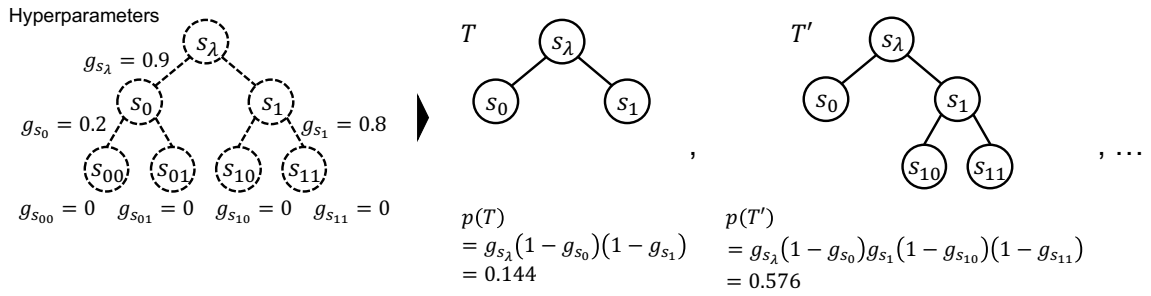


Figure 7: An example of the prior distribution on $T \in \mathcal{T}$. Its hyperparameters are given as shown in the left, e.g., s_λ becomes an inner node with probability $g_{s_\lambda} = 0.9$. Therefore, the data observation process includes $x_{k_{s_\lambda}}$ with probability $g_{s_\lambda} = 0.9$. Similarly, s_1 becomes an inner node and the data observation process includes $x_{k_{s_1}}$ with probability $g_{s_\lambda} g_{s_1} = 0.9 \cdot 0.8$ (Nakahara et al., 2022, Remark 2).

more general in the sense that it allows infinite depth, our prior is more general in the sense that the functional form of g_s is not restricted. However, we can set $D_{\max}$ arbitrarily large in our prior distribution, as we will describe in Appendix B. Therefore, in practice, our prior distribution can be considered to include the prior distribution in BayesianCART by setting $D_{\max}$ sufficiently large, e.g., one million.

Lastly, the prior distribution of $\boldsymbol{k}$ is as follows.

Assumption 4. We assume that k_s is independently assigned to each $s \in \mathcal{I}_{\max}$ with probability $1/(p+q)$, that is, $p(\boldsymbol{k})$ is the uniform distribution on $\mathcal{K}$.

In other words, each feature can be assigned multiple times on a path from the root node to a leaf node.

Remark 2 (Guidelines for hyperparameter tuning). Regarding hyperparameter tuning, we can provide some general guidelines. Our model's complexity is largely determined by the following three hyperparameters:

- The maximum tree depth $D_{\max}$
  Increasing this allows for deeper models to be included as candidates, while we can set $D_{\max}$ arbitrarily large in practice (see Appendix B).
- The branching probability g_s
  Setting this higher emphasizes deeper trees, i.e., more complex models.
- Hyperparameters of the prior distribution $p(\boldsymbol{\theta}_s)$
  If the variance of $p(y|\boldsymbol{x}, \boldsymbol{\theta}_s)$ at each leaf node s is small, the model tends to explain the overall data variability through a combination of multiple leaf nodes. Thus, the small variance of $p(y|\boldsymbol{x}, \boldsymbol{\theta}_s)$ emphasizes deeper and more complex models. We can control the probability that the variance of $p(y|\boldsymbol{x}, \boldsymbol{\theta}_s)$ becomes small by tuning the hyperparameters of the prior distribution $p(\boldsymbol{\theta}_s)$.

B META-TREE MARKOV CHAIN MONTE CARLO METHODS

In Bayesian decision theory, calculating the posterior distribution of unknown parameters is often necessary for optimal decision making (Berger, 1985). In the case of this study, it means that the calculation of $p(\boldsymbol{k}, T, \boldsymbol{\theta}|\boldsymbol{x}^n, y^n)$ is necessary. This section describes an algorithm to calculate the posterior distribution $p(\boldsymbol{k}, T, \boldsymbol{\theta}|\boldsymbol{x}^n, y^n)$. Here, it should be noted that we do not learn the thresholds for subspace partitioning because they are deterministicly derived from $\boldsymbol{k}$ and $\boldsymbol{x}^n$ in our setup (see Definition 1). In other words, we regard the problem of threshold learning as the problem of learning how many times the same k is assigned on a path from the root node to a leaf node, and optimally solve it in a Bayesian manner.

B.1 Overview of the Algorithm

First, the posterior distribution $p(\boldsymbol{k}, T, \boldsymbol{\theta}|\boldsymbol{x}^n, y^n)$ can be decomposed as follows:

$$p(\boldsymbol{k}, T, \boldsymbol{\theta}|\boldsymbol{x}^n, y^n) = p(\boldsymbol{k}|\boldsymbol{x}^n, y^n) p(T|\boldsymbol{x}^n, y^n, \boldsymbol{k}) p(\boldsymbol{\theta}|\boldsymbol{x}^n, y^n, \boldsymbol{k}, T). \tag{16}$$

In this study, we propose a method to sample $\boldsymbol{k}$ from $p(\boldsymbol{k}|\boldsymbol{x}^n, y^n)$ using an MH algorithm where both $p(T|\boldsymbol{x}^n, y^n, \boldsymbol{k})$ and $p(\boldsymbol{\theta}|\boldsymbol{x}^n, y^n, \boldsymbol{k}, T)$ are exactly calculated and marginalized out. This marginalization significantly reduces the dimensionality of the distribution to be approximated by the MH algorithm, and the convergence and mixing of the Markov chain will be accelerated. Furthermore, this method is able to update k_s for all inner nodes simultaneously in a single MH step. This will also accelerate the convergence and mixing. However, updating many variables may cause a low acceptance probability in the MH step. To mitigate this negative effect, we carefully design a proposal distribution used in the MH step. Moreover, we extend it to a replica exchange Monte Carlo (REMC) method (Swendsen and Wang, 1986) to deal with multimodality of the posterior distribution. Herein, we only describe the underlying MH method. The extension to the REMC method is described in Appendix D.

The overview of our algorithm is as follows. Since our algorithm is a kind of MH algorithm, we follow the general framework of the MH algorithm (Hastings, 1970). Therefore, we repeat generating $\boldsymbol{k}^*$ from a proposal distribution $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$ and accepting it according to the following acceptance probability for each iteration number t.

$$A(\boldsymbol{k}^*, \boldsymbol{k}^{(t-1)}) \coloneqq \min\left\{1, \frac{p(\boldsymbol{k}^*|\boldsymbol{x}^n, y^n) q(\boldsymbol{k}^{(t-1)}|\boldsymbol{k}^*)}{p(\boldsymbol{k}^{(t-1)}|\boldsymbol{x}^n, y^n) q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})}\right\}. \tag{17}$$

If $\boldsymbol{k}^*$ is accepted, we make $\boldsymbol{k}^{(t)} \leftarrow \boldsymbol{k}^*$, otherwise $\boldsymbol{k}^{(t)} \leftarrow \boldsymbol{k}^{(t-1)}$. In general, the following proposition holds for MH methods.

Proposition 1. If $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$ is time-invariant[5] and $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}) > 0$ holds for any $\boldsymbol{k}^*$ and $\boldsymbol{k}^{(t-1)}$ through this process, then the *detailed balance* condition is satisfied and an empirical distribution of the obtained sample $\{\boldsymbol{k}^{(t)}\}_{t=1,2,\dots}$ converges to the objective distribution $p(\boldsymbol{k}|\boldsymbol{x}^n, y^n)$ after sufficient iterations.

For the readers not familiar with MCMC methods, we briefly summarize the proof for this proposition (Bishop, 2006, Chapter 11) in Appendix K.

On the other hand, the unique feature of our algorithm is that we use the exactly calculated posterior distribution of T and $\boldsymbol{\theta}$ to sample $\boldsymbol{k}^*$ and evaluate $A(\boldsymbol{k}^*, \boldsymbol{k}^{(t-1)})$. Thus, our algorithm is summarized as follows: we repeat the following procedure for each iteration number t from any initial value $\boldsymbol{k}^{(0)}$.

1. Calculate $p(\boldsymbol{\theta}|\boldsymbol{x}^n, y^n, T, \boldsymbol{k}^{(t-1)})$ and $p(y^n|\boldsymbol{x}^n, T, \boldsymbol{k}^{(t-1)})$ (see Section B.2).
2. Calculate $p(T|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)})$ and $p(y^n|\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)})$ by using $p(y^n|\boldsymbol{x}^n, T, \boldsymbol{k}^{(t-1)})$ (see Section B.3).
3. Sample $\boldsymbol{k}^*$ according to a proposal distribution $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$ based on $p(T|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)})$ (see Section B.4).
4. Accept $\boldsymbol{k}^*$ according to a probability $A(\boldsymbol{k}^*, \boldsymbol{k}^{(t-1)})$ calculated by using $p(y^n|\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)})$ (see Section B.5).

If our purpose was just calculating the posterior distribution $p(\boldsymbol{k}, T, \boldsymbol{\theta}|\boldsymbol{x}^n, y^n)$, the above procedure would be enough, but if our purpose is to predict a new data point y_{n+1}, we often need to calculate the posterior predictive distribution $p(y_{n+1}|\boldsymbol{x}^n, y^n, \boldsymbol{x}_{n+1})$ after sufficient iterations of the above procedure. The method for this calculation will be described in Section B.6. The computational complexity of the above procedure is roughly $O(nD_{\max} + 2^{D_{\max}})$ for each iteration. In Appendix C, we reduce this to $O(nD_{\max})$.

B.2 Calculation of $p(\boldsymbol{\theta}|\boldsymbol{x}^n, y^n, T, \boldsymbol{k}^{(t-1)})$ and $p(y^n|\boldsymbol{x}^n, T, \boldsymbol{k}^{(t-1)})$

First, we define $\boldsymbol{x}_{s,\boldsymbol{k}}$ as $\{\boldsymbol{x}_i\}_{i:s\preceq s_{T_{\max},\boldsymbol{k}}(\boldsymbol{x}_i)}$, where $s \prec s'$ means s is an ancestor node of s'. Namely, $\boldsymbol{x}_{s,\boldsymbol{k}}$ is the set of data points that pass through s in the data generating process for given $\boldsymbol{k}$. Note that $\bigcup_{s\in\mathcal{L}(T)} \boldsymbol{x}_{s,\boldsymbol{k}} = \boldsymbol{x}^n$ holds for any $T \in \mathcal{T}$ and $\boldsymbol{k} \in \mathcal{K}$. In a similar manner, we define $y_{s,\boldsymbol{k}} \coloneqq \{y_i\}_{i:s\preceq s_{T_{\max},\boldsymbol{k}}(\boldsymbol{x}_i)}$. In addition, let $p(y_{s,\boldsymbol{k}}|\boldsymbol{x}_{s,\boldsymbol{k}}, \boldsymbol{\theta}_s)$ denote $\prod_{i:s\preceq s_{T_{\max},\boldsymbol{k}}(\boldsymbol{x}_i)} p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}_s)$. Note that, $\boldsymbol{x}_{s,\boldsymbol{k}}$ and $y_{s,\boldsymbol{k}}$ may be empty. In such a case, we define $p(y_{s,\boldsymbol{k}}|\boldsymbol{x}_{s,\boldsymbol{k}}, \boldsymbol{\theta}_s) = 1$, which is similar to usual empty products.

Hereafter, we fix $\boldsymbol{k}^{(t-1)}$. Using the above notations, $p(\boldsymbol{\theta}|\boldsymbol{x}^n, y^n, T, \boldsymbol{k}^{(t-1)})$ can be expressed as follows:

$$p(\boldsymbol{\theta}|\boldsymbol{x}^n, y^n, T, \boldsymbol{k}^{(t-1)}) = \frac{p(y^n|\boldsymbol{x}^n, \boldsymbol{\theta}, T, \boldsymbol{k}^{(t-1)}) p(\boldsymbol{\theta})}{p(y^n|\boldsymbol{x}^n, T, \boldsymbol{k}^{(t-1)})} = \prod_{s\in\mathcal{L}_T} \frac{p(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}}, \boldsymbol{\theta}_s) p(\boldsymbol{\theta}_s)}{\int p(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}}, \boldsymbol{\theta}_s) p(\boldsymbol{\theta}_s) \mathrm{d}\boldsymbol{\theta}_s}. \tag{18}$$

[5]For $t \neq t'$, if $\boldsymbol{k}^{(t-1)} = \boldsymbol{k}^{(t'-1)}$ holds, then $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}) = q(\boldsymbol{k}^*|\boldsymbol{k}^{(t'-1)})$ holds for any $\boldsymbol{k}^*$.

By Assumption 2, we can calculate $\int p(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}},\boldsymbol{\theta}_s)p(\boldsymbol{\theta}_s)\mathrm{d}\boldsymbol{\theta}_s$ with an acceptable computational cost. Therefore, we can calculate $p(\boldsymbol{\theta}|\boldsymbol{x}^n,y^n,T,\boldsymbol{k}^{(t-1)})$.

It should be noted that $\int p(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}},\boldsymbol{\theta}_s)p(\boldsymbol{\theta}_s)\mathrm{d}\boldsymbol{\theta}_s$ does not depend on T but only on s and $\boldsymbol{k}^{(t-1)}$. Therefore, we need not to calculate $\int p(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}},\boldsymbol{\theta}_s)p(\boldsymbol{\theta}_s)\mathrm{d}\boldsymbol{\theta}_s$ for all $T\in\mathcal{T}$ but only for all s where $\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}}$ and $y_{s,\boldsymbol{k}^{(t-1)}}$ are not empty. Lastly, we define the following notation.

$$f(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}},s,\boldsymbol{k}^{(t-1)}) \coloneqq \int p(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}},\boldsymbol{\theta}_s)p(\boldsymbol{\theta}_s)\mathrm{d}\boldsymbol{\theta}_s. \tag{19}$$

Using this notation, $p(y^n|\boldsymbol{x}^n,T,\boldsymbol{k}^{(t-1)})$ can be expressed as follows:

$$p(y^n|\boldsymbol{x}^n,T,\boldsymbol{k}^{(t-1)}) = \prod_{s\in\mathcal{L}_T} f(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}},s,\boldsymbol{k}^{(t-1)}). \tag{20}$$

B.3 Calculation of $p(T|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)})$ and $p(y^n|\boldsymbol{x}^n,\boldsymbol{k}^{(t-1)})$

From Assumption 3 and Eq. (20), $p(T|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)})$ can be expressed as follows:

$$p(T|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}) = \frac{p(y^n|\boldsymbol{x}^n,T,\boldsymbol{k}^{(t-1)})p(T)}{p(y^n|\boldsymbol{x}^n,\boldsymbol{k}^{(t-1)})} \tag{21}$$

$$= \frac{1}{p(y^n|\boldsymbol{x}^n,\boldsymbol{k}^{(t-1)})}\left(\prod_{s\in\mathcal{I}_T} g_s\right)\left(\prod_{s'\in\mathcal{L}_T}(1-g_{s'})f(y_{s',\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s',\boldsymbol{k}^{t-1}},s',\boldsymbol{k}^{(t-1)})\right). \tag{22}$$

Therefore, the posterior distribution $p(T|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)})$ has the same form as the prior distribution in Eq. (15) except for the normalization term $p(y^n|\boldsymbol{x}^n,\boldsymbol{k}^{(t-1)})$. Here, using an algorithm based on the concept of meta-trees proposed by Dobashi et al. (2021) as well as Nakahara and Matsushima (2024), we can calculate this normalization term and distribute it to each factor to represent the posterior distribution in the same form as the prior distribution. In other words, $p(T)$ is a conjugate prior distribution for our model (Nakahara et al., 2022). Since we also use this algorithm, we call our sampling method Meta-Tree MCMC (MTMCMC) method.

Before describing this algorithm, we define a set of *active* inner nodes for $\boldsymbol{x}^n$ and $\boldsymbol{k}^{(t-1)}$ as follows:

$$\mathcal{I}_{\boldsymbol{x}^n,\boldsymbol{k}^{(t-1)}} \coloneqq \{s\in\mathcal{I}_{T_{\max}} \mid \exists \boldsymbol{x}_i\in\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}},\exists \boldsymbol{x}_j\in\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}} \text{ s.t. } \boldsymbol{x}_i\neq\boldsymbol{x}_j\}. \tag{23}$$

In other words, the active node is the inner node that has at least two unique data points, and it can be divided further. Using this notation, the algorithm to calculate $p(T|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)})$ is described as the following proposition.

Proposition 2 (Dobashi et al. 2021; Nakahara and Matsushima 2024). For any $\boldsymbol{x}^n$, y^n, and $\boldsymbol{k}^{(t-1)}$, the posterior distribution of T is represented as follows:

$$p(T|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}) = \prod_{s\in\mathcal{I}_T} g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}} \prod_{s'\in\mathcal{L}_T}(1-g_{s'|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}), \tag{24}$$

where $g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}\in[0,1]$ can be calculated from $\boldsymbol{x}^n$, y^n, and $\boldsymbol{k}^{(t-1)}$ as follows:

$$g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}} \coloneqq \begin{cases} \frac{g_s\prod_{s'\in\mathrm{Ch}(s)} q(y_{s',\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s',\boldsymbol{k}^{(t-1)}},s',\boldsymbol{k}^{(t-1)})}{q(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}},s,\boldsymbol{k}^{(t-1)})}, & s\in\mathcal{I}_{\boldsymbol{x}^n,\boldsymbol{k}^{(t-1)}}, \\ g_s, & \text{otherwise}, \end{cases} \tag{25}$$

where $\mathrm{Ch}(s)$ denotes the set of child nodes of s on $T_{\max}$ and $q(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}},s,\boldsymbol{k}^{(t-1)})$ is defined for any $s\in\mathcal{S}_{T_{\max}}$ as follows.

$$q(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}},s,\boldsymbol{k}^{(t-1)}) \coloneqq \begin{cases} (1-g_s)f(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}},s,\boldsymbol{k}^{(t-1)}) \\ \qquad + g_s\prod_{s'\in\mathrm{Ch}(s)} q(y_{s',\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s',\boldsymbol{k}^{(t-1)}},s',\boldsymbol{k}^{(t-1)}), & s\in\mathcal{I}_{\boldsymbol{x}^n,\boldsymbol{k}^{(t-1)}}, \\ f(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}},s,\boldsymbol{k}^{(t-1)}), & \text{otherwise}. \end{cases} \tag{26}$$

Intuitively, $q(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}}, s, \boldsymbol{k}^{(t-1)})$ is a kind of marginal likelihood (actually, $p(y^n|\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)}) = q(y_{s_\lambda,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s_\lambda,\boldsymbol{k}^{(t-1)}}, s_\lambda, \boldsymbol{k}^{(t-1)})$ holds). It is because $q(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}}, s, \boldsymbol{k}^{(t-1)})$ is a weighted sum of two marginal likelihoods $f(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}}, s, \boldsymbol{k}^{(t-1)})$ and $\prod_{s'\in \mathrm{Ch}(s)} q(y_{s',\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s',\boldsymbol{k}^{(t-1)}}, s', \boldsymbol{k}^{(t-1)})$. The former represents the marginal likelihood when the node s is a leaf node, and it is weighted with its prior probability $(1-g_s)$. The latter represents the marginal likelihood when the node s is a inner node, and it is weighted with its prior probability g_s. Then, the updating formula of $g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}$ is the Bayes' theorem itself because the posterior branching probability $g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}$ is obtained as a product of the prior branching probability g_s and the marginal likelihood $\prod_{s'\in \mathrm{Ch}(s)} q(y_{s',\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s',\boldsymbol{k}^{(t-1)}}, s', \boldsymbol{k}^{(t-1)})$ divided by the normalization term $q(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}}, s, \boldsymbol{k}^{(t-1)})$.

In this algorithm, we need not to calculate $g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}$ and $q(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}}, s, \boldsymbol{k}^{(t-1)})$ for $s \notin \mathcal{I}_{\boldsymbol{x}^n,\boldsymbol{k}^{(t-1)}}$. Since $|\mathcal{I}_{\boldsymbol{x}^n,\boldsymbol{k}^{(t-1)}}|$ is upper bounded by some function of only n, we can run this algorithm for arbitrarily large $D_{\max}$.

B.4 Sampling of $\boldsymbol{k}^*$ From $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$

Asymptotically, we can use any time-invariant distribution that satisfies $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}) > 0$ for any $\boldsymbol{k}^*$ and $\boldsymbol{k}^{(t-1)}$, e.g., the uniform distribution $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}) = (p+q)^{-|\mathcal{I}_{\max}|}$. However, its design crucially affects the practical performance. This can be explained from a viewpoint of an analogy of the MH algorithm and a neighborhood searching algorithm. In the MH algorithm, $\boldsymbol{k}^*$ is proposed from a kind of neighborhood of $\boldsymbol{k}^{(t-1)}$ according to $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$. Roughly speaking, it will be accepted if it increases the probability of the objective distribution, i.e., $p(\boldsymbol{k}^*|\boldsymbol{x}^n, y^n) > p(\boldsymbol{k}^{(t-1)}|\boldsymbol{x}^n, y^n)$. Since the entropy of $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$ corresponds to the step size of neighborhood search, it should be larger to accelerate the search but it should be smaller to increase the acceptance ratio. Therefore, a desirable $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$ should induce many changes in the elements of $\boldsymbol{k}^{(t-1)}$ when $p(\boldsymbol{k}^{(t-1)}|\boldsymbol{x}^n, y^n)$ is small and a few changes when $p(\boldsymbol{k}^{(t-1)}|\boldsymbol{x}^n, y^n)$ is large. Note that $\boldsymbol{k}$ is discrete and hierarchically structured. Therefore, we cannot use the derivative of $p(\boldsymbol{k}^{(t-1)}|\boldsymbol{x}^n, y^n)$, and any Gibbs sampler for our model has not been reported to our best knowledge. Then, we use the posterior distribution $p(T|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)})$ as a heuristic to design $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$.

By Proposition 2, in the tth iteration, we can represent the posterior distribution $p(T|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)})$ in the same form as the prior distribution $p(T)$ with a posterior branching probability $g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}$ of each node s. In other words, the larger $g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}$ is, the higher the posterior probability that the split by the $k_s^{(t-1)}$th explanatory variable was performed at the node s of the true model is. For such nodes, k_s^* should not be changed from $k_s^{(t-1)}$. Then, we use this probability $g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}$ as a heuristic to determine the fixed elements of $\boldsymbol{k}^*$, that is, the smaller $g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}$ the node s has, the more frequently $k_s^{(t-1)}$ is changed. Consequently, we generate $\boldsymbol{k}^*$ according to the following procedure, see also Fig. 8. (The initial value $\boldsymbol{k}^{(0)}$ is generated from the uniform distribution on $\mathcal{K}$.)

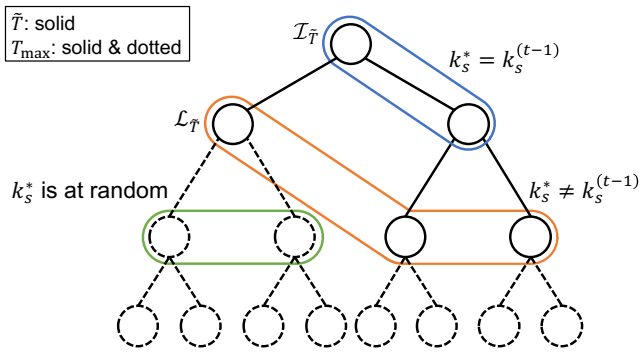


Figure 8: An example of $\tilde{T}$ and properties of $\boldsymbol{k}^*$. Here, we assume $D_{\max} = 3$. The tree with solid line represents $\tilde{T}$. For $s \in \mathcal{I}_{\tilde{T}}$, we have $k_s^* = k_s^{(t-1)}$. For $s \in \mathcal{L}_{\tilde{T}} \cap \mathcal{I}_{\max}$, we have $k_s^* \neq k_s^{(t-1)}$. For $s \in \mathcal{I}_{\max} \setminus \mathcal{S}_{\tilde{T}}$, k_s^* follows the uniform distribution on $\{1, 2, \ldots, p+q\}$.

1. $\tilde{T}$ is generated according to

$$q(\tilde{T}|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)}) \coloneqq \prod_{s\in\mathcal{I}_{\tilde{T}}} \min\{g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}, \bar{g}\} \prod_{s'\in\mathcal{L}_{\tilde{T}}} (1-\min\{g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}, \bar{g}\}), \tag{27}$$

where $\bar{g}\in[0,1]$ is predetermined in a burn-in phase (see also Appendix H).

2. For $s\in\mathcal{I}_{\tilde{T}}$, $k_s^{(t-1)}$ is fixed, i.e., $k_s^*=k_s^{(t-1)}$ holds.

3. For $s\in\mathcal{L}_{\tilde{T}}\cap\mathcal{I}_{\max}$, $k_s^{(t-1)}$ is changed according to the uniform distribution on $\{1,2,\dots,p+q\}\setminus\{k_s^{(t-1)}\}$.

4. The others are generated according to the uniform distribution on $\{1,2,\dots,p+q\}$.

Note that $\tilde{T}$ is uniquely determined from $\boldsymbol{k}^*$ and $\boldsymbol{k}^{(t-1)}$ as the maximum tree that satisfies $k_s^*=k_s^{(t-1)}$ for all $s\in\mathcal{I}_{\tilde{T}}$. Therefore, the proposal distribution $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$ is represented as follows:

$$q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}) = q(\tilde{T}|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)})(p+q-1)^{-|\mathcal{L}_{\tilde{T}}\cap\mathcal{I}_{\max}|}(p+q)^{-|\mathcal{I}_{\max}\setminus\mathcal{S}_{\tilde{T}}|}. \tag{28}$$

Moreover, the following theorem holds.

Theorem 1. *The empirical distribution of $\{\boldsymbol{k}^{(t)}\}_{t=1,2,\dots}$ obtained from our algorithm converges to $p(\boldsymbol{k}|\boldsymbol{x}^n,y^n)$ after sufficient large number of MH steps.*

Proof. If $\boldsymbol{k}^{(t-1)}=\boldsymbol{k}^{(t'-1)}$ holds, then $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})=q(\boldsymbol{k}^*|\boldsymbol{k}^{(t'-1)})$ clearly holds for any $\boldsymbol{k}^*$ even when $t\neq t'$. Therefore, a Markov chain of $\boldsymbol{k}^{(t)}$ induced from $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$ and $A(\boldsymbol{k}^*,\boldsymbol{k}^{(t-1)})$ is time-invariant. Moreover, $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})>0$ holds for any $\boldsymbol{k}^*$ and $\boldsymbol{k}^{(t-1)}$. Therefore, the induced Markov chain of $\boldsymbol{k}^{(t)}$ is ergodic. Then, $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$ satisfies the condition of Proposition 1. Therefore, empirical distribution of the obtained sample converges to $p(\boldsymbol{k}|\boldsymbol{x}^n,y^n)$. □

Remark 3. $\bar{g}$ is an additional parameter to control the entropy of $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$. When n is large, $g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}$ for the nodes near the root s_λ numerically equals to 1. Then, k_s for them tends to be fixed and ergodicity will be collapsed. Introducing $\bar{g}$, all the elements of $\boldsymbol{k}^{(t-1)}$ are refreshed with the probability $1-\bar{g}$ and the ergodicity is ensured. This induces a "jump" of $\boldsymbol{k}^*$ and has some effects to deal with multimodality of the posterior distribution. A more effective approach to multimodality is extending our MH method to the REMC method, which is described in Appendix D.

B.5 Evaluation of the Acceptance Probability $A(\boldsymbol{k}^*,\boldsymbol{k}^{(t-1)})$

First, from the Bayes' theorem and Assumption 4, Eq. (17) is further transformed as follows:

$$A(\boldsymbol{k}^*,\boldsymbol{k}^{(t-1)}) = \min\left\{1, \frac{p(y^n|\boldsymbol{x}^n,\boldsymbol{k}^*)q(\boldsymbol{k}^{(t-1)}|\boldsymbol{k}^*)}{p(y^n|\boldsymbol{x}^n,\boldsymbol{k}^{(t-1)})q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})}\right\}. \tag{29}$$

Here, $p(y^n|\boldsymbol{x}^n,\boldsymbol{k}^{(t-1)})$ is already calculated as $p(y^n|\boldsymbol{x}^n,\boldsymbol{k}^{(t-1)}) = q(y_{s_\lambda,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s_\lambda,\boldsymbol{k}^{(t-1)}},s_\lambda,\boldsymbol{k}^{(t-1)})$ by using Eq. (26). In a similar manner, we can calculate $p(y^n|\boldsymbol{x}^n,\boldsymbol{k}^*)$.

Furthermore, note that the following holds for $q(\boldsymbol{k}^{(t-1)}|\boldsymbol{k}^*)$.

Remark 4. Because of the uniqueness of $\tilde{T}$, transition from $\boldsymbol{k}^{(t-1)}$ to $\boldsymbol{k}^*$ cannot occur through any other tree than $\tilde{T}$, and vice versa. Therefore, $q(\boldsymbol{k}^{(t-1)}|\boldsymbol{k}^*)$ is represented as follows.

$$q(\boldsymbol{k}^{(t-1)}|\boldsymbol{k}^*) = q(\tilde{T}|\boldsymbol{x}^n, y^n, \boldsymbol{k}^*)(p+q-1)^{-|\mathcal{L}_{\tilde{T}}\cap\mathcal{I}_{\max}|}(p+q)^{-|\mathcal{I}_{\max}\setminus\mathcal{S}_{\tilde{T}}|}, \tag{30}$$

where $\tilde{T}$ is the same tree as that in Eq. (28).

As a result, we can efficiently evaluate Eq. (29) because many terms in the numerator and the denominator of Eq. (29) are canceled by substituting Eqs. (28) and (30) as follows:

$$A(\boldsymbol{k}^*,\boldsymbol{k}^{(t-1)}) = \min\left\{1, \frac{p(y^n|\boldsymbol{x}^n,\boldsymbol{k}^*)q(\tilde{T}|\boldsymbol{x}^n,y^n,\boldsymbol{k}^*)}{p(y^n|\boldsymbol{x}^n,\boldsymbol{k}^{(t-1)})q(\tilde{T}|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)})}\right\}. \tag{31}$$

B.6 Calculation of $p(y_{n+1}|\boldsymbol{x}^n, y^n, \boldsymbol{x}_{n+1})$

If our purpose is to predict a new data point y_{n+1}, we often need to calculate the posterior predictive distribution $p(y_{n+1}|\boldsymbol{x}^n, y^n, \boldsymbol{x}_{n+1})$. Using the obtained sample of $\boldsymbol{k}$, we can calculate it as follows:

$$\begin{aligned}
&p(y_{n+1}|\boldsymbol{x}^n, y^n, \boldsymbol{x}_{n+1}) \\
&\approx \frac{1}{t_{\mathrm{end}} - t_{\mathrm{start}}} \sum_{t=t_{\mathrm{start}}}^{t_{\mathrm{end}}} \underbrace{\sum_{T} p(T|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t)}) \underbrace{\int p(y_{n+1}|\boldsymbol{x}_{n+1}, \boldsymbol{k}^{(t)}, T, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{x}^n, y^n, T, \boldsymbol{k}^{(t)}) \mathrm{d}\boldsymbol{\theta}}_{(a)}}_{(b)},
\end{aligned} \tag{32}$$

where t_{start} and t_{end} represent the length of the burn-in and the total number of sampling iterations, respectively. Since $p(T)$ and $p(\boldsymbol{\theta})$ are the conjugate prior distribution of our model, $p(T|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t)})$ and $p(\boldsymbol{\theta}|\boldsymbol{x}^n, y^n, T, \boldsymbol{k}^{(t)})$ have the same forms as the prior distributions. Therefore, we can calculate (a) and (b) in the above formula in a similar manner to Eqs. (19) and (26), respectively.

C AN EFFICIENT PROPOSAL DISTRIBUTION AND ITS COMPLEXITY

C.1 A Computationally Efficient Proposal Distribution

In the algorithm described in Appendix B, we have to remember $k_s^{(t-1)}$ for any node $s \in \mathcal{S}_{T_{\max}}$ to sample $\boldsymbol{k}^*$. It is because any node $s \in \mathcal{S}_{T_{\max}}$ may becomes a leaf node of $\tilde{T}$ with non-zero probability, and for such cases, k_s^* must be different from $k_s^{(t-1)}$. In this section, we describe a method to reduce this complexity. Although the explanation is based on the proposal distribution (28), this method is also applied for other proposal distributions that will be described later in Eqs. (44) and (45).

First, we define the following parameter for all $s \in \mathcal{S}_{T_{\max}}$:

$$\tilde{g}_{s|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)}} \coloneqq \begin{cases} \min\{g_{s|\boldsymbol{x}^n, y^n, \boldsymbol{k}}, \bar{g}\}, & s \in \mathcal{I}_{\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)}}, \\ 0, & \text{otherwise}, \end{cases} \tag{33}$$

where $\mathcal{I}_{\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)}}$ is the set of the active nodes for $\boldsymbol{x}^n$ and $\boldsymbol{k}^{(t-1)}$ defined in Eq. (23).

Then, we generate $\tilde{T}_{\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}}$ according to the following distribution instead of Eq. (27).

$$q(\tilde{T}_{\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)}) = \prod_{s \in \mathcal{I}_{\tilde{T}_{\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}}}} \tilde{g}_{s|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)}} \prod_{s' \in \mathcal{L}_{\tilde{T}_{\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}}}} (1 - \tilde{g}_{s'|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)}}). \tag{34}$$

Lastly, we generate $\boldsymbol{k}^*$ as follows. For $s \in \mathcal{I}_{\tilde{T}_{\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}}}$, $k_s^{(t-1)}$ is fixed and $k_s^* = k_s^{(t-1)}$. For $s \in \mathcal{L}_{\tilde{T}_{\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}}} \cap \mathcal{I}_{\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)}}$, $k_s^{(t-1)}$ is changed according to the uniform distribution on $\{1, 2, \ldots, p+q\} \backslash \{k_s^{(t-1)}\}$. For the other nods, k_s^* is generated according to the uniform distribution on $\{1, 2, \ldots, p+q\}$. Therefore, we do not require $k_s^{(t-1)}$ on $s \notin \mathcal{I}_{\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)}}$ to sample $\boldsymbol{k}^*$.

Hereafter, we confirm the efficiency of the above proposal distribution in evaluating $A(\boldsymbol{k}^*, \boldsymbol{k}^{(t-1)})$. To evaluate $A(\boldsymbol{k}^*, \boldsymbol{k}^{(t-1)})$, we have to calculate not only $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$ but also $q(\boldsymbol{k}^{(t-1)}|\boldsymbol{k}^*)$. First, we show the specific form of $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$ and $q(\boldsymbol{k}^{(t-1)}|\boldsymbol{k}^*)$. Given $\boldsymbol{k}^*$ and $\boldsymbol{k}^{(t-1)}$, $\tilde{T}_{\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}}$ is uniquely determined in the following manner. Let $\tilde{T}'$ be the maximum tree that satisfies $k_s^* = k_s^{(t-1)}$ for all $s \in \mathcal{I}_{\tilde{T}'}$. Then, $\tilde{T}_{\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}}$ is uniquely determined as the tree whose inner node set is the intersection of $\mathcal{I}_{\tilde{T}'}$ and $\mathcal{I}_{\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)}}$. Therefore, $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$ is represented as follows:

$$\begin{aligned}
&q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}) \\
&= q(\tilde{T}_{\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)}) (p+q-1)^{-\left|\mathcal{L}_{\tilde{T}_{\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}}} \cap \mathcal{I}_{\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)}}\right|} (p+q)^{-\left|\mathcal{I}_{\max} \backslash \left(\mathcal{S}_{\tilde{T}_{\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}}} \cap \mathcal{I}_{\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)}}\right)\right|}.
\end{aligned} \tag{35}$$

In a similar manner, $q(\boldsymbol{k}^{(t-1)}|\boldsymbol{k}^*)$ is represented as follows:

$$q(\boldsymbol{k}^{(t-1)}|\boldsymbol{k}^*) = q(\tilde{T}_{\boldsymbol{k}^{(t-1)}|\boldsymbol{k}^*}|\boldsymbol{x}^n, y^n, \boldsymbol{k}^*)(p+q-1)^{-\left|\mathcal{L}_{\tilde{T}_{\boldsymbol{k}^{(t-1)}|\boldsymbol{k}^*}} \cap \mathcal{I}_{\boldsymbol{x}^n, \boldsymbol{k}^*}\right|}(p+q)^{-\left|\mathcal{I}_{\max} \setminus \left(\mathcal{S}_{\tilde{T}_{\boldsymbol{k}^{(t-1)}|\boldsymbol{k}^*}} \cap \mathcal{I}_{\boldsymbol{x}^n, \boldsymbol{k}^*}\right)\right|}. \quad (36)$$

Next, we show $\tilde{T}_{\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}} = \tilde{T}_{\boldsymbol{k}^{(t-1)}|\boldsymbol{k}^*}$ holds and most terms in Eqs. (35) and (36) are canceled when evaluating $A(\boldsymbol{k}^*, \boldsymbol{k}^{(t-1)})$. For any node $s \in \mathcal{S}_{T_{\max}}$, the necessary and sufficient condition that s is included in $\tilde{T}_{\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}}$ is as follows:

1. For any ancestor node s' of s, $k^*_{s'} = k^{(t-1)}_{s'}$ holds.
2. The parent node of s is included in $\mathcal{I}_{\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)}}$.

Let s be a node of $\tilde{T}_{\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}}$. Since $k^*_{s'} = k^{(t-1)}_{s'}$ holds for any ancestor node s' of s, all the data points that reach the parent node of s under $\boldsymbol{k}^{(t-1)}$ also reach the parent node of s under $\boldsymbol{k}^*$. Therefore, the parent node of s is included in $\mathcal{I}_{\boldsymbol{x}^n, \boldsymbol{k}^*}$, i.e., any node of $\tilde{T}_{\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}}$ is also included in $\tilde{T}_{\boldsymbol{k}^{(t-1)}|\boldsymbol{k}^*}$, and vice versa. Therefore, $\tilde{T}_{\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}} = \tilde{T}_{\boldsymbol{k}^{(t-1)}|\boldsymbol{k}^*}$ holds. Then, let $\tilde{T}$ represent both $\tilde{T}_{\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}}$ and $\tilde{T}_{\boldsymbol{k}^{(t-1)}|\boldsymbol{k}^*}$. Moreover, for any $s \in \mathcal{L}_{\tilde{T}}$, if $s \in \mathcal{I}_{\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)}}$ holds, then $s \in \mathcal{I}_{\boldsymbol{x}^n, \boldsymbol{k}^*}$ also holds because $k^*_{s'} = k^{(t-1)}_{s'}$ holds for any ancestor node s' of s. Therefore, $(\mathcal{L}_{\tilde{T}} \cap \mathcal{I}_{\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)}}) = (\mathcal{L}_{\tilde{T}} \cap \mathcal{I}_{\boldsymbol{x}^n, \boldsymbol{k}^*})$ holds. Similarly, $(\mathcal{S}_{\tilde{T}} \cap \mathcal{I}_{\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)}}) = (\mathcal{S}_{\tilde{T}} \cap \mathcal{I}_{\boldsymbol{x}^n, \boldsymbol{k}^*})$ holds.

Consequently, most terms of Eqs. (35) and (36) in $A(\boldsymbol{k}^*, \boldsymbol{k}^{(t-1)})$ are canceled, and $A(\boldsymbol{k}^*, \boldsymbol{k}^{(t-1)})$ is represented as follows:

$$A(\boldsymbol{k}^*, \boldsymbol{k}^{(t-1)}) = \min\left\{1, \frac{p(y^n|\boldsymbol{x}^n, \boldsymbol{k}^*)q(\tilde{T}|\boldsymbol{x}^n, y^n, \boldsymbol{k}^*)}{p(y^n|\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)})q(\tilde{T}|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)})}\right\}. \quad (37)$$

In summary, by using the proposal distribution in Eq. (35), we can sample $\boldsymbol{k}^*$ without using $k^{(t-1)}_s$ on $s \notin \mathcal{I}_{\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)}}$ and we can evaluate $q(\tilde{T}|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)})$ and $q(\tilde{T}|\boldsymbol{x}^n, y^n, \boldsymbol{k}^*)$ using only the parameters on the nodes in $\mathcal{S}_{\tilde{T}}$.

In addition, we confirm the validity of $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$. It is clear that $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$ is time-invariant. For any $\boldsymbol{k}^*$ and $\boldsymbol{k}^{(t-1)}$, $\tilde{T}$ is uniquely determined in the aforementioned manner, and for such $\tilde{T}$, $q(\tilde{T}|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)}) > 0$ holds. Therefore, $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}) > 0$ holds for any $\boldsymbol{k}^*$ and $\boldsymbol{k}^{(t-1)}$. Thus, $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$ satisfies the condition of Proposition 1.

C.2 Complexity Analysis

We summarize the computational complexity of MTMCMC methods based on the proposal distribution in this section. In each iteration of the MTMCMC methods, we have to do the following procedure.

1. Calculate $\int p(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}}, \boldsymbol{\theta}_s)p(\boldsymbol{\theta}_s)\mathrm{d}\boldsymbol{\theta}_s$ for all s s.t. $\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}} \neq \emptyset$.
2. Calculate $p(T|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)})$ and $p(y^n|\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)})$.
3. Generate $\boldsymbol{k}^*$ according to $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$.
4. Evaluate $A(\boldsymbol{k}^*, \boldsymbol{k}^{(t-1)})$.

In the following, we evaluate the computational complexity of these procedures.

Calculation of $\int p(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}}, \boldsymbol{\theta}_s)p(\boldsymbol{\theta}_s)\mathrm{d}\boldsymbol{\theta}_s$ for all s s.t. $\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}} \neq \emptyset$: we consider the worst case where $\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}} \neq \emptyset$ holds for all $s \in \mathcal{S}_{T_{\max}}$. For each node $s \in \mathcal{S}_{T_{\max}}$, the computational cost to calculate $\int p(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}}, \boldsymbol{\theta}_s)p(\boldsymbol{\theta}_s)\mathrm{d}\boldsymbol{\theta}_s$ is usually proportional to the number of data points when $p(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}}, \boldsymbol{\theta}_s)$ is an usual exponential family distribution. Although the number of data points assigned to each node s depends on $\boldsymbol{k}^{(t-1)}$, the sum of the number of data points assigned to all the nodes at

each depth is always n. Therefore, the computational cost to calculate $\int p(y_{s,\boldsymbol{k}^{(t-1)}}|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}},\boldsymbol{\theta}_s)p(\boldsymbol{\theta}_s)\mathrm{d}\boldsymbol{\theta}_s$ for all $s \in \mathcal{S}_{T_{\max}}$ is $O(nD_{\max})$. We can simultaneously calculate $p(\boldsymbol{\theta}_s|\boldsymbol{x}_{s,\boldsymbol{k}^{(t-1)}}, y_{s,\boldsymbol{k}^{(t-1)}})$ in this procedure.

Calculation of $p(T|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)})$ and $p(y^n|\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)})$: using the algorithm in Proposition 2, we can calculate $p(T|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)})$ with a complexity of $O(|\mathcal{I}_{\boldsymbol{x}^n,\boldsymbol{k}^{(t-1)}}|)$. Note that $|\mathcal{I}_{\boldsymbol{x}^n,\boldsymbol{k}^{(t-1)}}| \leq n(D_{\max}-1)$ always holds. It is because each data point $\boldsymbol{x}_i$ is assigned at most $(D_{\max}-1)$ inner nodes on the path from the root node s_λ to the leaf node $s_{\boldsymbol{k}^{(t-1)},T_{\max}}(\boldsymbol{x}_i)$. We can simultaneously calculate $p(y^n|\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)})$ in this procedure.

Generation of $\boldsymbol{k}^*$ according to $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$: using the proposal distribution described in this section, we need not generate k^*_s for $s \notin \mathcal{I}_{\boldsymbol{x}^n,\boldsymbol{k}^*}$. Therefore, the computational complexity is $O(|\mathcal{I}_{\boldsymbol{x}^n,\boldsymbol{k}^*}|)$.

Evaluation of $A(\boldsymbol{k}^*, \boldsymbol{k}^{(t-1)})$: to evaluate $A(\boldsymbol{k}^*, \boldsymbol{k}^{(t-1)})$, we have to calculate $p(y^n|\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)})$, $p(y^n|\boldsymbol{x}^n, \boldsymbol{k}^*)$, $q(\tilde{T}|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)})$, and $q(\tilde{T}|\boldsymbol{x}^n, y^n, \boldsymbol{k}^*)$. We have already calculated $p(y^n|\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)})$. We can calculate $p(y^n|\boldsymbol{x}^n, \boldsymbol{k}^*)$ in a similar manner to $p(y^n|\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)})$ by using the algorithm in Proposition 2. Its computational complexity is $O(|\mathcal{I}_{\boldsymbol{x}^n,\boldsymbol{k}^*}|)$. The computational complexity to calculate $q(\tilde{T}|\boldsymbol{x}^n, y^n, \boldsymbol{k}^{(t-1)})$ is $O(|\mathcal{S}_{\tilde{T}}|) = O(|\mathcal{I}_{\boldsymbol{x}^n,\boldsymbol{k}^{(t-1)}}|)$ when using the proposal distribution in this section. The computational cost to calculate $q(\tilde{T}|\boldsymbol{x}^n, y^n, \boldsymbol{k}^*)$ is similarly evaluated.

Consequently, the total complexity of the MTMCMC method is roughly $O(t_{\mathrm{end}}nD_{\max})$.

D EXTENSION TO REPLICA EXCHANGE MONTE CARLO METHODS

In this section, we extend our MH method to REMC methods (Swendsen and Wang, 1986) to deal with multimodality of the posterior distribution. First, we define the following joint distribution over $\mathcal{K}^J$ for $J \in \mathbb{N}$.

$$q(\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_J) \coloneqq \prod_{j=1}^{J} q_j(\boldsymbol{k}_j) \coloneqq \prod_{j=1}^{J} \frac{p(\boldsymbol{k}_j|\boldsymbol{x}^n, y^n)^{\beta_j}}{\sum_{\boldsymbol{k}_j \in \mathcal{K}} p(\boldsymbol{k}_j|\boldsymbol{x}^n, y^n)^{\beta_j}}, \tag{38}$$

where $0 \leq \beta_1 < \beta_2 < \cdots < \beta_J = 1$. Since $\beta_J = 1$, the marginal distribution $q_J(\boldsymbol{k}_J)$ is equivalent to the posterior distribution $p(\boldsymbol{k}|\boldsymbol{x}^n, y^n)$ required to calculate the Bayesian optimal prediction. Therefore, we construct an MCMC method for this joint distribution $q(\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_J)$ and use the sample for only $\boldsymbol{k}_J$, ignoring those for $\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_{J-1}$. In REMC methods, the sample from the joint distribution is obtained as follows:

1. For each $q_j(\boldsymbol{k}_j)$, run the MH method and obtain the sample $\boldsymbol{k}_j^{(1)}, \boldsymbol{k}_j^{(2)}, \ldots$, where the proposal distribution and the acceptance probability are similar to those in the usual MH method in the MTMCMC method.

2. Let $m \in \mathbb{N}$ be a predetermined number. For every m iterations of the MH method, we randomly choose $j \in \{1, \ldots, J-1\}$ and exchange $\boldsymbol{k}_j^{(t)}$ and $\boldsymbol{k}_{j+1}^{(t)}$ with probability

$$\frac{q_j(\boldsymbol{k}_{j+1}^{(t)})q_{j+1}(\boldsymbol{k}_j^{(t)})}{q_j(\boldsymbol{k}_j^{(t)})q_{j+1}(\boldsymbol{k}_{j+1}^{(t)})} = \frac{p(y^n|\boldsymbol{x}^n, \boldsymbol{k}_{j+1}^{(t)})^{\beta_j} p(y^n|\boldsymbol{x}^n, \boldsymbol{k}_j^{(t)})^{\beta_{j+1}}}{p(y^n|\boldsymbol{x}^n, \boldsymbol{k}_{j+1}^{(t)})^{\beta_{j+1}} p(y^n|\boldsymbol{x}^n, \boldsymbol{k}_j^{(t)})^{\beta_j}}, \tag{39}$$

where we used the Bayes' theorem and Assumption 4. (This procedure can be applied multiple times at the same tth iteration of the MH method.)

It is known that the above procedure satisfies the detailed balance condition and the obtained sample asymptotically follows $q(\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_J)$ after sufficient iterations. Since β_j is monotonically increasing, the effect of multimodality of $p(\boldsymbol{k}|\boldsymbol{x}^n, y^n)$ is reduced for small j. Therefore, $\boldsymbol{k}_j^{(t)}$ for small j tends to move over the multiple modes. Exchanging these samples with probability (39), the REMC methods ensure the detailed balance and provide the sample from the multiple modes.

E APPLICATION TO UNCERTAINTY EVALUATION IN CLASSIFICATION PROBLEMS

In the following, we apply the MTMCMC method to evaluate the uncertainty of a new data point in classification problems. In this study, we formulate the problem of uncertainty evaluation within the framework of Bayesian

decision theory. Hereafter, we describe the problem formulation of evaluating the uncertainty under Bayesian decision theory and the optimal decision. Further, we show the results of applying it to benchmark data.

There are various approaches to evaluate uncertainty when the set of objective variables is a finite set, but here we consider the problem of outputting a predictive distribution. Therefore, the output of the decision function $\delta(\boldsymbol{x}^n, y^n, \boldsymbol{x}_{n+1})$ is a vector $\hat{\boldsymbol{p}} \coloneqq (\hat{p}_1, \dots, \hat{p}_{|\mathcal{Y}|})$ representing the predictive distribution. Here, $\hat{p}_j$ represents the predictive probability that $y_{n+1} = j$.

As an evaluation criterion for the output predictive distribution, we consider the following log loss.[6]

$$L_{\log}(\boldsymbol{k}, T, \boldsymbol{\theta}, \delta(\boldsymbol{x}^n, y^n, \boldsymbol{x}_{n+1})) \coloneqq -\sum_{y_{n+1}} p(y_{n+1}|\boldsymbol{x}_{n+1}, \boldsymbol{k}, T, \boldsymbol{\theta}) \sum_{j=1}^{|\mathcal{Y}|} I\{y_{n+1} = j\} \log \hat{p}_j. \tag{40}$$

Since this function depends on the data and unknown parameters $(\boldsymbol{k}, T, \boldsymbol{\theta})$, we use the Bayes risk function $BR(\delta)$ as the evaluation criterion for δ. The Bayes risk function $BR(\delta)$ is defined as the expected value of this loss function with respect to the prior distribution of the data and unknown parameters in the following manner.

$$BR_{\log}(\delta) \coloneqq \sum_{\boldsymbol{k}} \sum_{T} \iint p(\boldsymbol{k}, T, \boldsymbol{\theta}) p(y^n|\boldsymbol{x}^n, \boldsymbol{k}, T, \boldsymbol{\theta}) L_{\log}(\boldsymbol{k}, T, \boldsymbol{\theta}, \delta(\boldsymbol{x}^n, y^n, \boldsymbol{x}_{n+1})) \mathrm{d}y^n \mathrm{d}\boldsymbol{\theta}. \tag{41}$$

In this case, as a general result of Bayesian decision theory, it is known that the optimal decision function is given by the posterior predictive distribution (Berger, 1985; Matsushima et al., 1991). Similar to the case of regression problems, we approximate the posterior predictive distribution by Eq. (6).

In the numerical experiments, we computed predictive distributions for various datasets (Dua and Graff, 2017; Cason, 1999) and evaluated them using the log loss, as shown in Table 2. See Appendix I for details of the datasets. For BART, we used both the default setting and the setting where `ntree=1` because the MTMCMC method is based on the single tree model. For LightGBM and XGBoost, we tuned the normalization parameter (`reg_lambda` option) because these methods caused overfitting in their default setting. The main limitation of the MTMCMC method is that the threshold at the inner node is fixed. This may become more problematic for classification problems than regression problems. Moreover, this may lead to unnecessarily deep tree and reduce the number of data points at each leaf node. In such a case, the effect of the prior distribution to the predictive distribution becomes more significant. For this problem, we tune the MTMCMC method to use a more complicated threshold, which minimizes the sum of squared deviations of the k_sth explanatory variable on both sides of the threshold (see also Appendix G). Moreover, we use a more sophisticated prior distribution $p(\boldsymbol{\theta}_s)$ based on the empirical distribution of y^n. Regarding hyperparameter tuning, see also some general guidelines in Remark 2 in Appendix A. Similarly to Section 5, we repeated 2-fold cross-validation 10 times to more accurately evaluate the generalization performance for small sample. Please refer to Appendix I for the other hyperparameters.

In Table 2, the smallest average log loss value among the methods excluding the default BART for each dataset is shown in bold with a shaded background, and the second smallest value is shown in bold. Comparing with BART, our method outperforms it when `ntree` option is set to 1. This is likely due to the convergence and mixing speed of the MTMCMC method, although there is a little difference in the models of BART and ours. It is because BART constructs discrete distributions by applying the logistic sigmoid function to the output of tree-based nonlinear function while our model has the categorical distribution at each leaf node. In the default setting with the sum-of-trees model, BART shows good performance especially for the binary classification problems. This gap may be decreased by extending our model to a sum-of-trees model. Within the MTMCMC methods, by tuning the definition of the thresholds and the hyperparamerer of the prior distribution for the leaf node parameter, the performance is significantly improved. These modifications may be important in the classification problem under the log loss. In average, the tuned MTMCMC shows almost the same performance as LightGBM and better than the other methods. Overall, we believe that our method demonstrates performance comparable to other state-of-the-art methods. Regarding the computational cost, using Python on a laptop detailed in Appendix J, each MH step on a single chain in the tuned MTMCMC required approximately 25 msec for german data, where $D_{\max} = 10$.

[6] This is the expectation of usual log loss or cross entropy in machine learning for a new data point.

Table 2: Average log loss values and standard errors (SE)

Avg. log loss (SE)	LightGBM	XGBoost	RF	MTRF	BART `ntree=1`	MTMCMC Leaf model: Categorical	MTMCMC (tuned) Leaf model: Categorical	BART
sonar (n:208, dim:60)	**0.455** (0.021)	0.479 (0.021)	0.465 (0.008)	0.674 (0.056)	0.597 (0.016)	0.490 (0.010)	**0.457** (0.014)	0.474 (0.012)
ionosphere (n:351, dim:33)	**0.254** (0.015)	**0.266** (0.015)	0.305 (0.048)	0.332 (0.033)	0.402 (0.022)	0.298 (0.008)	0.278 (0.013)	0.276 (0.007)
votes (n:435, dim:48)	**0.129** (0.009)	0.133 (0.009)	**0.124** (0.005)	0.151 (0.009)	0.234 (0.020)	0.141 (0.006)	0.147 (0.008)	0.118 (0.005)
breast cancer (n:569, dim:30)	**0.113** (0.008)	**0.115** (0.008)	0.163 (0.020)	0.183 (0.012)	0.257 (0.013)	0.161 (0.011)	0.147 (0.008)	0.126 (0.005)
diabetes (n:768, dim:8)	0.523 (0.011)	0.538 (0.011)	**0.496** (0.009)	0.617 (0.021)	0.537 (0.008)	**0.514** (0.007)	0.529 (0.008)	0.474 (0.005)
german (n:1000, dim:24)	0.530 (0.008)	0.543 (0.010)	**0.513** (0.007)	0.619 (0.019)	0.555 (0.005)	**0.529** (0.006)	0.548 (0.009)	0.505 (0.005)
titanic3[7] (n:1309, dim:9)	0.457 (0.006)	0.459 (0.006)	0.777 (0.068)	0.478 (0.010)	0.465 (0.006)	**0.452** (0.005)	**0.453** (0.005)	0.449 (0.005)
rice (n:3810, dim:7)	0.214 (0.004)	0.215 (0.004)	0.324 (0.015)	0.228 (0.005)	0.213 (0.003)	**0.194** (0.003)	**0.193** (0.003)	0.189 (0.003)
iris (3 classes) (n:150, dim:4)	0.165 (0.014)	0.194 (0.010)	**0.118** (0.010)	0.163 (0.011)	0.341 (0.025)	0.189 (0.011)	**0.157** (0.015)	0.302 (0.015)
glass (6 classes) (n:214, dim:9)	**0.833** (0.024)	0.841 (0.027)	0.848 (0.067)	1.004 (0.023)	1.482 (0.065)	1.030 (0.018)	**0.773** (0.032)	1.354 (0.036)
Average	0.367	0.378	0.413	0.445	0.508	0.400	0.368	0.427

F IN-DEPTH EXPERIMENTS ON ALGORITHM BEHAVIOR

F.1 Other Examples of Proposal Distributions

We show other examples of the proposal distributions of $\boldsymbol{k}^*$ and numerically compare their effectiveness.

F.1.1 Uniform Proposal Distribution

For comparison, we utilize the uniform distribution on $\mathcal{K}$ as the proposal distribution, i.e., $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}) = (p+q)^{-|\mathcal{I}_{\max}|}$. For this type of proposal distribution, the acceptance probability that satisfies the detailed balance is derived as follows:

$$A(\boldsymbol{k}^*, \boldsymbol{k}^{(t-1)}) = \min\left\{1, \frac{p(y^n|\boldsymbol{x}^n, \boldsymbol{k}^*)}{p(y^n|\boldsymbol{x}^n, \boldsymbol{k}^{(t-1)})}\right\}. \tag{42}$$

F.1.2 Tree Prior Based Proposal Distribution

We can utilize the tree prior (15) to generate $\tilde{T}$ instead of Eq. (27). Then, the proposal distribution $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$ is represented as follows:

$$q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)}) = p(\tilde{T})(p+q-1)^{-|\mathcal{L}_{\tilde{T}} \cap \mathcal{I}_{\max}|}(p+q)^{-|\mathcal{I}_{\max} \setminus \mathcal{S}_{\tilde{T}}|}. \tag{43}$$

The acceptance probability that satisfies the detailed balance for this proposal distribution is the same as Eq. (42).

[7]Data obtained from `http://hbiostat.org/data` courtesy of the Vanderbilt University Department of Biostatistics.

F.1.3 Other Examples of Tree Posterior Based Proposal Distribution

In Eq. (27), we truncated the hyperparameter $g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}$ by $\bar{g}$ to ensure the ergodicity and induce a jump. We also utilize a reduced one, such as,

$$q(\tilde{T}|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}) = \prod_{s\in\mathcal{I}_{\tilde{T}}} \alpha g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}} \prod_{s'\in\mathcal{L}_{\tilde{T}}} (1-\alpha g_{s'|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}), \tag{44}$$

where α is in the range of $[0,1]$.

Further, not only reducing the large $g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}$, we can also amplify the small $g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}$ as follows.

$$\begin{aligned} &q(\tilde{T}|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}) \\ &= \prod_{s\in\mathcal{I}_{\tilde{T}}} \left((g_s+\alpha(g_{s|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}-g_s)\right) \prod_{s'\in\mathcal{L}_{\tilde{T}}} \Big(1-\big(g_{s'}+\alpha(g_{s'|\boldsymbol{x}^n,y^n,\boldsymbol{k}^{(t-1)}}-g_{s'})\big)\Big), \end{aligned} \tag{45}$$

where g_s is the hyperparameter of the prior (15).

For Eqs. (44) and (45), the acceptance probability is defined in a similar manner to Eq. (29). Many terms in the numerator and the denominator of Eq. (29) are canceled in a similar manner to Remark 4. We can tune α in the same algorithm as Algorithm 1 in Appendix H.

F.2 Experiment 1: Convergence to Exact Posterior

Purpose: we confirm the convergence of the empirical distribution of the MCMC sample to the exact posterior distribution. Since our proposal distributions satisfy the detailed balance condition, the approximated posterior distributions are expected to converge to the exact one. Further, we confirm the effectiveness of the design policy of the proposal distribution, compared with the uniform proposal distribution.

Conditions: we assume a model with small p, q and $D_{\max}$ as a true model to calculate the exact posterior distribution. Specifically, we perform the experiment under the following conditions. We assume $p=0$ and $q=5$. Therefore, all the explanatory variables are binary. In addition, $\mathcal{Y}$ is also assumed to be the binary set $\{0,1\}$. We assume $D_{\max}=3$. Then, we have $|\mathcal{K}|=5^7=78125$. Specific values of $\boldsymbol{k}$, T, and $\boldsymbol{\theta}$ are shown in Fig. 9. The data generative model $p(y|\boldsymbol{x},\boldsymbol{\theta}_s)$ is the Bernoulli distribution $\mathrm{Bern}(y|\theta_s)$. The ith explanatory variable $\boldsymbol{x}_i$ is independently generated according to the uniform distribution on $\{0,1\}^q$. Then, y_i is generated from the model shown in Fig. 9. The sample size n is 100 and the number of generated samples is 10.

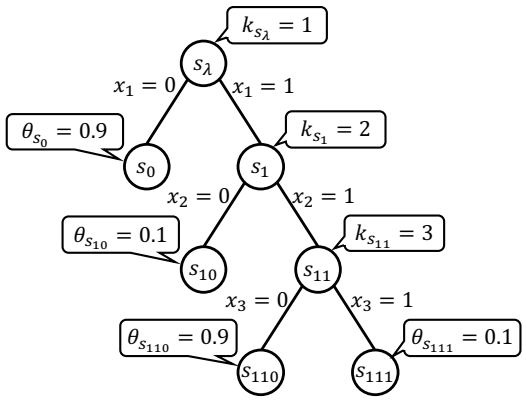


Figure 9: The true model assumed in Experiment 1. Here, $\mathcal{Y}=\{0,1\}$ and the parameters θ_s on the leaf nodes represent the probability that $y=1$.

For posterior learning, we independently assume the beta distribution $\mathrm{Beta}(\theta_s|0.5,0.5)$ as the prior distribution for each θ_s. The hyperparameter of $p(T)$ is fixed at $g_s=0.5$ for each $s\in\mathcal{I}_{\max}$. Herein, we utilize two proposal distributions: the uniform distribution and Eq. (28). The tuning parameter $\bar{g}$ in Eq. (27) of the tree posterior based proposal distribution $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$ is fixed at 0.75. The burn-in length is 500 and the MCMC process is continued until 1000 samples are accepted.

Results: we evaluate the distance $d(p, \hat{p})$ between the exact posterior distribution $p(\boldsymbol{k}|\boldsymbol{x}^n, y^n)$ and the approximated posterior distribution $\hat{p}(\boldsymbol{k}|\boldsymbol{x}^n, y^n)$ obtained from the MCMC sample by the following Jensen-Shannon divergence.[8]

$$d(p, \hat{p}) \coloneqq \frac{1}{2} \sum_{\boldsymbol{k} \in \mathcal{K}} p(\boldsymbol{k}|\boldsymbol{x}^n, y^n) \log \frac{p(\boldsymbol{k}|\boldsymbol{x}^n, y^n)}{r(\boldsymbol{k}|\boldsymbol{x}^n, y^n)} + \frac{1}{2} \sum_{\boldsymbol{k} \in \mathcal{K}} \hat{p}(\boldsymbol{k}|\boldsymbol{x}^n, y^n) \log \frac{\hat{p}(\boldsymbol{k}|\boldsymbol{x}^n, y^n)}{r(\boldsymbol{k}|\boldsymbol{x}^n, y^n)}, \tag{46}$$

where $r(\boldsymbol{k}|\boldsymbol{x}^n, y^n) \coloneqq (p(\boldsymbol{k}|\boldsymbol{x}^n, y^n) + \hat{p}(\boldsymbol{k}|\boldsymbol{x}^n, y^n))/2$ and we use the convention that $0 \log 0 = 0$.

Figure 10 shows the transition of the distance $d(p, \hat{p})$ for the increase of the number of the accepted tests. Both of the approximated posteriors converge to the exact one as expected. The convergence speed of the tree posterior based proposal distribution is faster than that of the uniform proposal distribution. In addition, the acceptance ratio of the tree posterior based proposal distribution was 0.274, while that of the uniform proposal distribution was 0.0118. These results support the effectiveness of our design policy of the proposal distribution. We also obtained similar results for other data generative models described in the next subsection.

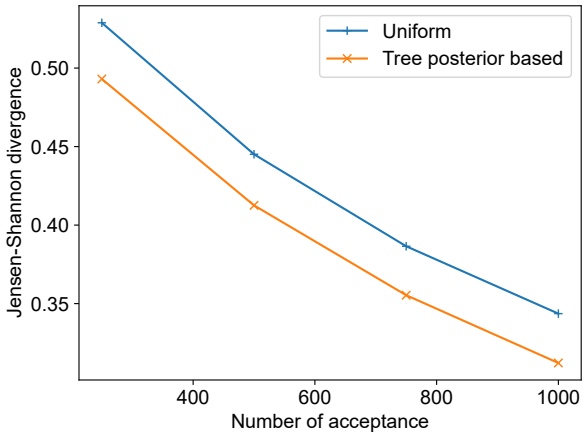


Figure 10: The Jensen-Shannon divergence between the exact posterior and the approximated posterior to the number of accepted tests of the MCMC.

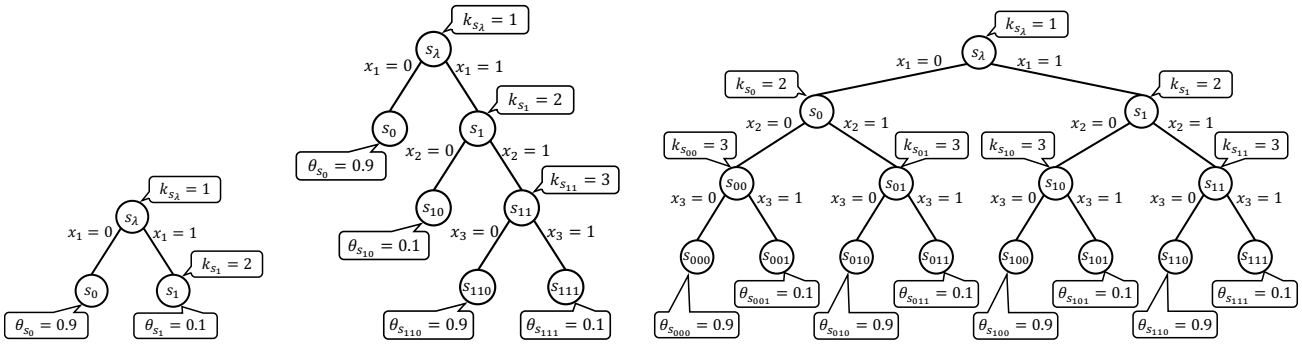
F.3 Experiment 2: Comparison of Convergence

We compare the convergence of the empirical distribution of the MCMC sample obtained from the aforementioned four proposal distributions $q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$: the uniform distribution and Eqs. (43), (28) and (45). We assumed three models shown in the upper side of Fig. 11. Herein, we assumed $p = 0$. The other hyperparameters are the same as those for Experiment 1. Resuls are shown in the lower side of Fig. 11 and Table 3. The tree posterior based proposal distributions (28) and (45) showed better performances, i.e., they showed faster convergence and higher acceptance ratio than the others. In particular, the uniform proposal distribution and Eq. (43) showed extremely low acceptance ratio for Model B, which has an unbalanced shape.

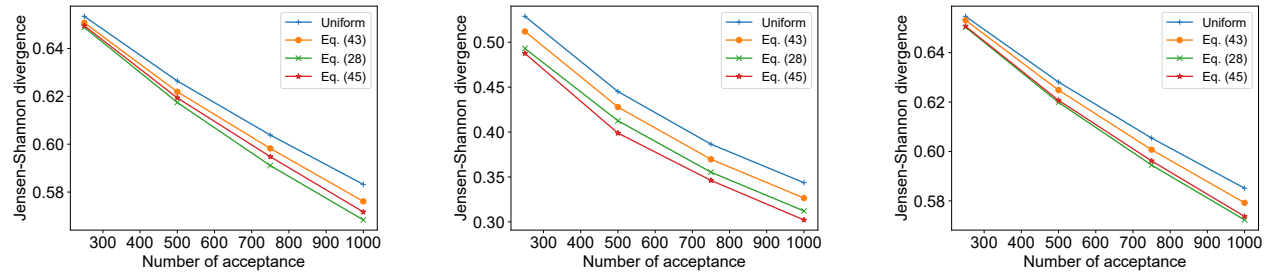
Table 3: Acceptance ratios for the compared proposal distributions.

	Acceptance ratio		
$q(\boldsymbol{k}^*|\boldsymbol{k}^{(t-1)})$	Model A	Model B	Model C
$(p+q)^{-\lvert\mathcal{I}_{\max}\rvert}$	0.11	0.012	0.11
Eq. (43)	0.27	0.073	0.29
Eq. (28)	0.49	0.27	0.50
Eq. (45)	0.46	0.25	0.46

[8] Since $\hat{p}(\boldsymbol{k}|\boldsymbol{x}^n, y^n)$ is a empirical distribution and takes 0 on some points in $\mathcal{K}$, the usual Kullback–Leibler divergence cannot be evaluated.



(a) Model A (b) Model B (c) Model C



(d) Results for Model A (e) Results for Model B (f) Results for Model C

Figure 11: The models assumed in the experiment and the results of the Jensen-Shannon divergence for each proposal distributions.

G VARIOUS DEFINITIONS OF THRESHOLDS

First, it should be noted that the explanatory variable $\boldsymbol{x}^n$ is considered a known constant and not treated as a random variable throughout this study. In this case, the threshold $t_{k_s,s}$ that determines the space partitioning assigned to the inner node s can be defined in any way as long as it is a value that is deterministically determined from $\boldsymbol{x}^n$ and the feature assignment vector $\boldsymbol{k}$. The reason why it doesn't matter is that, regardless of how $t_{k_s,s}$ is defined, asymptotically exact posterior distribution can be calculated for that definition using the MTMCMC method.

In the example shown in Fig. 6, we defined $t_{k_s,s} := (a_{k_s,s} + b_{k_s,s})/2$, but this was a definition where $t_{k_s,s}$ was determined solely from $\boldsymbol{k}$. By using $\boldsymbol{x}^n$ in the definition of $t_{k_s,s}$, we can consider more flexible models. For example, we can define $t_{k_s,s}$ as the mean, median, or midpoint of the maximum and minimum values of $\boldsymbol{x}_i$ contained in $\mathcal{X}_{\boldsymbol{k}}(s)$. A more complex threshold can be defined as a threshold based on some descriptive statistical clustering method. For example, we can use the threshold that minimizes the sum of squared deviations of the data on both sides of the threshold, i.e., a one-dimensional k-means method along the k_sth explanatory variable.

It should be noted that the discussion here is merely defining the observation process of y^n given $\boldsymbol{x}^n$ as a known constant, and it is not estimating $t_{k_s,s}$ from $\boldsymbol{x}^n$. The method of treating $t_{k_s,s}$ as an unknown parameter of the model and estimating it from the data is a topic for future research.

H TUNING ALGORITHM OF $\bar{g}$ in Eq. (27)

The additional parameter $\bar{g}$ in Eq. (27) should be tuned in the burn-in phase. The following Algorithm 1 can be used. We set $r_{\mathrm{obj}} = 0.3$, $\rho = 0.99$, and $\phi = 0.999$, but these values should be changed depending on the length of the burn-in phase.

Algorithm 1 Tuning algorithm of $\bar{g}$

Require: $r_{\text{obj}} \in [0,1]$, $\rho \in [0,1]$, $\phi \in [0,1]$
Ensure: $\bar{g} \in [0,1]$
$\bar{g} \leftarrow 0$
$N_{\text{accept}} \leftarrow 1$
$N_{\text{propose}} \leftarrow 1$
$N'_{\text{propose}} \leftarrow 1$
while Burn-in phase **do**
  Propose $\boldsymbol{k}^*$
  if $\boldsymbol{k}^*$ is accepted **then**
    $N_{\text{accept}} \leftarrow \rho N_{\text{accept}} + 1$
  else
    $N_{\text{accept}} \leftarrow \rho N_{\text{accept}}$
  end if
  $N_{\text{propose}} \leftarrow \rho N_{\text{propose}} + 1$
  $\hat{r} \leftarrow N_{\text{accept}} / N_{\text{propose}}$
  if $\hat{r} > r_{\text{obj}}$ **then**
    $\bar{g}_{\text{tmp}} \leftarrow \bar{g} \cdot r_{\text{obj}} / \hat{r}$
  else
    $\bar{g}_{\text{tmp}} \leftarrow 1 - (1 - \bar{g})(1 - r_{\text{obj}})/(1 - \hat{r})$
  end if
  $N'_{\text{propose}} \leftarrow \phi N'_{\text{propose}} + 1$
  $\bar{g} \leftarrow (\phi \bar{g} + \bar{g}_{\text{tmp}}) / N'_{\text{propose}}$
end while
return $\bar{g}$

I DETAILED CONDITIONS OF THE EXPERIMENTS

I.1 Regression

I.1.1 Datasets

The datasets and their references used in the experiment in Section 5 are as follows. If the author information is missing, we show only the URL of the dataset.

- automobile (Schlimmer, 1987)
- servo (Ulrich, 1993)
- cpu (Feldmesser, 1987)
- liver (`https://doi.org/10.24432/C54G67`, 2016)
- Mpg (Quinlan, 1993)
- student (Cortez, 2014)
- diabetes (Pedregosa et al., 2011)
- boston (Harrison and Rubinfeld, 1978)
- cps (Berndt, 1991)
- strikes (Western, 1996)

I.1.2 Preprocessing

Basically, the following preprocessing was performed on all the data:

- Rows with missing values were removed.
- Variables with two possible values were converted to 0 and 1.
- Categorical variables with three or more possible values were one-hot encoded.
- Continuous explanatory variables and target variables were standardized to have mean 0 and variance 1. Usually, this standardization does not affect the results for decision trees, but as we will discuss later, since our model assumes a linear regression model at the leaf nodes, this improves the numerical stability of the Bayesian estimation of the regression coefficients.

For the datasets below, the following exceptional preprocessing was performed:

automobile The string-type explanatory variable "num-of-cylinders" was converted to a numerical value by converting 'eight' to 8, for example.

Mpg The explanatory variable "car name" contained various proper nouns of cars, but since there were a very large number of possible values, it was not one-hot encoded and was removed from the explanatory variables.

student This dataset contains multiple candidate target variables, but here we used the variable "G1" as the target variable.

I.1.3 Settings for Each Method

LightGBM All hyperparameters were set to the default values of the Python library lightgbm 3.3.5 (Ke et al., 2017). To output prediction intervals, the left endpoint of the interval was obtained by specifying the options `objective='quantile'` and `alpha=0.025` when creating an instance from the `LGBMRegressor` class. This setting outputs the 0.25% quantile. The right endpoint of the interval was obtained by specifying the options `objective='quantile'` and `alpha=0.975`. This setting outputs the 0.975% quantile.

XGBoost All hyperparameters were set to the default values of the Python library py-xgboost 2.0.3 (Chen and Guestrin, 2016). To output prediction intervals, the left endpoint of the interval was obtained by specifying the options `objective='reg:quantileerror'` and `quantile_alpha=0.025` when creating an instance from the `XGBRegressor` class. This setting outputs the 0.25% quantile. The right endpoint of the interval was obtained by specifying the options `objective='reg:quantileerror'` and `quantile_alpha=0.975`. This setting outputs the 0.975% quantile.

GBDT All hyperparameters were set to the default values of the Python library scikit-learn 1.2.2 (Pedregosa et al., 2011). To output prediction intervals, the left endpoint of the interval was obtained by specifying the options `loss='quantile'` and `alpha=0.025` when creating an instance from the `GradientBoostingRegressor` class. This setting outputs the 0.25% quantile. The right endpoint of the interval was obtained by specifying the options `loss='quantile'` and `alpha=0.975`. This setting outputs the 0.975% quantile.

BART `ntree=1`

- The `wbart` function from the R package BART 2.9.7 (Sparapani et al., 2021) was used.
- Since our method is a single tree model, the `ntree` option was set to 1 to ensure consistent conditions.
- The `power` and `base` options was set to `power=0` and `base=0.75` to use the same tree prior distribution as ours. This setting corresponds to $p_{\mathrm{SPLIT}} = \alpha(1+d_s)^{\beta} = 0.75(1+d_s)^{0} = 0.75$, and equivalent to $g_s = 0.75$ in our model. See also Remark 1 in Appendix A.
- We set the prior distribution of the noise term to be common between BART and MTMCMC. Since MTMCMC assumes an inverse gamma distribution $\mathrm{InvGam}(\sigma_s^2|2.1,1)$ as the prior distribution for the variance σ_s^2 of the noise term, we aligned both by utilizing the equivalence between the inverse gamma distribution $\mathrm{InvGam}\left(\frac{\nu}{2},\frac{\nu\lambda}{2}\right)$ and the scaled inverse chi-squared distribution $\nu\lambda\text{Inv-}\chi^2(\nu)$. Therefore, we set `sigdf=4.2` and `lambda=1/2.1` in the BART package.

- The length of the burn-in period and the number of post-burn-in samples were set to 100 and 500, respectively, to match our method.
- To output the 95% highest posterior predictive density credible intervals, we used the `hdi` function in the Python library arviz 0.19.0 for the `yhat.test` (samples from the posterior predictive distribution of the target variable for the test data), which is one of the return values of the `wbart`. The `multimodal` option of the `hdi` was set to `True`.

MTMCMC Leaf model: Normal

- The maximum depth $D_{\max}$ was set to 10.
- The thresholds at inner nodes were defined as the midpoint of the maximum and minimum values of $\boldsymbol{x}_i$ passing through s.
- The model assigned to each leaf node s was a normal distribution $\mathcal{N}(y_i|\mu_s, \sigma_s^2)$.
- The prior branching probability g_s for each node s in the tree prior distribution $p(T)$ was set to 0.75.
- The prior distribution for the parameters of the normal distribution at each leaf node s was set to the normal-gamma distribution $\mathcal{N}(\mu_s|0, \sigma_s^2)\mathrm{Gam}(1/\sigma_s^2|2.1, 1)$. The shape parameter of the gamma distribution is set to 2.1 to ensure the existence of the variance of the t-distribution of the predictive distribution.
- Instead of a simple MH method, we used the REMC method discussed in Appendix D. The number of chains was $J = 8$, the length of the burn-in period was 100, the number of post-burn-in samples was 500, and the tuning parameter $\bar{g}$ was set to 0.9. We set $\beta_j = j/J$ in Eq. (38). Replica exchange process was performed every 10 MH steps. In one replica exchange process, four replicas were randomly selected in order and subjected to an exchange test.
- When the confidence level is $(1-\alpha)$, the highest posterior predictive density credible interval is calculated as follows. Set the initial interval by $(\mu - \sigma/\sqrt{\alpha}, \mu + \sigma/\sqrt{\alpha})$ for the expected value μ and standard deviation σ of the posterior predictive distribution, and divide this interval into 1000 equal parts. Then, the endpoints of highest posterior predictive density interval will be selected from these 1000 candidates using numerical integration. The initial interval is the upper and lower bounds of the endpoints of the prediction interval obtained by Chebyshev's inequality.

MTMCMC Leaf model: LR model

- All the settings other than the leaf node model and its prior distribution are the same as MTMCMC Normal.
- The model assigned to each leaf node s was a linear regression model $\mathcal{N}(y_i|\boldsymbol{w}_s^\top \tilde{\boldsymbol{x}}_i, \sigma_s^2)$, where $\tilde{\boldsymbol{x}}_i$ is a vector obtained by extracting the continuous variables from the ith explanatory variable and appending an intercept term of 1 at the end.
- The prior distribution for the parameters of the linear regression model at each leaf node s was set to the normal-gamma distribution $\mathcal{N}(\boldsymbol{w}_s|\boldsymbol{0}, \sigma_s^2\boldsymbol{I})\mathrm{Gam}(1/\sigma_s^2|2.1, 1)$. The shape parameter of the gamma distribution is set to 2.1 to ensure the existence of the variance of the t-distribution of the predictive distribution.

MTMCMC Leaf model: LR model (sampling)

- All the settings other than the method to calculate the highest posterior predictive density credible interval are the same as MTMCMC LR.
- To output 95% highest posterior predictive density credible intervals, we used the `hdi` function in the Python library arviz 0.19.0 for the sample of y_{n+1} obtained from the posterior predictive distribution approximated by Eq. (6). The size of the sample was 500. The `multimodal` option of the `hdi` was set to `True`.

BART The `wbart` function from the R package BART 2.9.7 was used. All hyperparameters, including the length of the burn-in period and the number of samples, were set to the package's default values. The method for outputting 95% highest posterior predictive density credible intervals is the same as in the case of `ntree=1`.

I.2 Classification

I.2.1 Datasets

The datasets and their references used in the experiment in Appendix E are as follows. If the author information is missing, we show only the URL of the dataset.

- sonar (Sejnowski and Gorman, 1988)
- ionosphere (Sigillito et al., 1989)
- votes (`https://doi.org/10.24432/C5C01P`, 1987)
- breast cancer (Wolberg et al., 1995)
- diabetes (`https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database`, 1988)
- german (Hofmann, 1994)
- titanic3 (Cason, 1999)
- rice (`https://doi.org/10.24432/C5MW4Z`, 2019)
- iris (Fisher, 1988)
- glass (German, 1987)

I.2.2 Preprocessing

Basically, the following preprocessing was performed on all the data:

- Rows with missing values were removed.
- Variables with two possible values were converted to 0 and 1.
- Categorical variables with three or more possible values were one-hot encoded.

For the datasets below, the following exceptional preprocessing was performed:

ionosphere The second explanatory variable had a value of 0 for all data points, so it was removed from the explanatory variables.

german While several formats of data can be downloaded from the UCI repository (Hofmann, 1994), we used the data called "german.data-numeric."

titanic3 The used explanatory variables are "pclass", "age", "sibsp", "parch", "fare", "sex", and "embarked". Missing values were filled with the mode of each variable.

glass The possible values for the target variable were $[1, 2, 3, 5, 6, 7]$, but these were converted to $[0, 1, 2, 3, 4, 5]$ and treated as a 6-class classification problem.

I.2.3 Settings for Each Method

LightGBM We used the Python library lightgbm 3.3.5 (Ke et al., 2017). Since the default settings showed a clear overfitting, the `reg_lambda` option was adjusted. We tried `reg_lambda=1,5,10,25,50,100` and adopted `reg_lambda=5`, which produced the smallest average log loss value for the test data. The `predict_proba` function of `LGBMClassifier` was used to output the predictive distribution.

XGBoost We used the Python library py-xgboost 2.0.3 (Chen and Guestrin, 2016). Since the default settings showed a clear overfitting, the `reg_lambda` option was adjusted. We tried `reg_lambda=1,5,10,25,50,100` and adopted `reg_lambda=25`, which produced the smallest average log loss value for the test data. The `predict_proba` function of `XGBClassifier` was used to output the predictive distribution.

RF All hyperparameters were set to the default values of the Python library scikit-learn 1.2.2. The `predict_proba` function of `RandomForestClassifier` was used to output the predictive distribution.

MTRF

- The maximum depth $D_{\max}$ was set to 10.
- The model assigned to leaf node s was a categorical distribution $\mathrm{Cat}(y_i|\boldsymbol{\theta}_s)$.
- The prior branching probability g_s for each node in the tree prior distribution $p(T)$ was set to 0.75.
- The prior distribution for the parameters of the categorical distribution at leaf node s was set to the non-informative Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\theta}_s|(0.5,\ldots,0.5))$.
- The other hyperparameters were set to the default values of the Python library BayesML 0.3.0 (Nakahara et al., 2025).

BART `ntree=1`

- The `mbart` function from the R package BART 2.9.7 (Sparapani et al., 2021) was used.
- Since our method is a single tree model, the `ntree` option was set to 1 to ensure consistent conditions.
- The `power` and `base` options was set to `power=0` and `base=0.75` to use the same tree prior distribution as ours. This setting corresponds to $p_{\mathrm{SPLIT}} = \alpha(1+d_s)^{\beta} = 0.75(1+d_s)^0 = 0.75$, and equivalent to $g_s = 0.75$ in our model. See also Remark 1 in Appendix A.
- The length of the burn-in period and the number of post-burn-in samples were set to 100 and 500, respectively, to match the conditions of our method.
- To output the predictive distribution, the `prob.test.mean` was used. This is the mean value of the samples from the posterior predictive distribution of the target variable for the test data and one of the return values of the `mbart`.

MTMCMC

- The maximum depth $D_{\max}$ was set to 10.
- The thresholds at inner nodes were defined as the midpoint of the maximum and minimum values of $\boldsymbol{x}_i$ passing through s.
- The model assigned to each leaf node s was a categorical distribution $\mathrm{Cat}(y_i|\boldsymbol{\theta}_s)$.
- The prior branching probability g_s for each node s in the tree prior distribution $p(T)$ was set to 0.75.
- The prior distribution for the parameters of the categorical distribution at each leaf node s was set to the non-informative Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\theta}_s|(0.5,\ldots,0.5))$.
- Instead of a simple MH method, we used the REMC method discussed in Appendix D. The number of chains was $J = 8$, the length of the burn-in period was 100, the number of post-burn-in samples was 500, and the tuning parameter $\bar{g}$ was set to 0.9. We set $\beta_j = j/J$ in Eq. (38). Replica exchange process was performed every 10 MH steps. In one replica exchange process, four replicas were randomly selected in order and subjected to an exchange test.

MTMCMC (tuned)

- We tuned the definition of the thresholds and the hyperparameter of the prior distribution for the leaf node parameters. The other settings are the same as the above MTMCMC.
- The thresholds at inner nodes are defined as the point that minimizes the sum of squared deviations of the k_sth explanatory variable of the unique data points on both sides of the threshold, i.e., a one-dimensional k-means method (see also Appendix G).
- The prior distribution for the parameters of the categorical distribution at leaf node s was set to the Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\theta}_s|\boldsymbol{\alpha})$, where $\boldsymbol{\alpha}$ was tuned based on the empirical distribution of y^n as follows:

$$\alpha_l = \frac{\lambda}{n}\sum_{i=1}^{n} I\{y_i = l\}. \tag{47}$$

Here, λ is a tuning parameter to control the significance of the prior distribution. We tried $\lambda = 1, 0.5, 0.3, 0.2$ and adopted $\lambda = 0.3$, which produced the smallest average log loss value for the test data. See also the hyperparameter tuning guidelines in Remark 2 in Appendix A.

BART The `mbart` function from the R package BART 2.9.7 (Sparapani et al., 2021) was used. All hyperparameters, including the length of the burn-in period and the number of samples, were set to the package's default values. The method for outputting the predictive distribution is the same as in the case of `ntree=1`.

I.3 Feature Importance

The hyperparameters of the model were set as follows:

- The maximum depth $D_{\max}$ was set to 10.
- In data generation, the thresholds at inner nodes were given by recursively bisecting $[-3, 3]$.
- In posterior estimation and feature importance calculation, the thresholds at inner nodes were defined as the midpoint of the maximum and minimum values of $\boldsymbol{x}_i$ passing through s.
- The model assigned to each leaf node s was a normal distribution $\mathcal{N}(y_i|\mu_s, \sigma_s^2)$.
- The prior branching probability g_s for each node s in the tree prior distribution $p(T)$ was set to 0.75.
- The prior distribution for the parameters of the normal distribution at each leaf node s was set to the normal-gamma distribution $\mathcal{N}(\mu_s|0, 0.01 \times \sigma_s^2)\mathrm{Gam}(1/\sigma_s^2|10, 10)$.
- Instead of a simple MH method, we used the REMC method discussed in Appendix D. The number of chains was 4, the length of the burn-in period was 50, the number of post-burn-in samples was 100, and the tuning parameter $\bar{g}$ was set to 0.9. Replica exchange process was performed every 10 MH steps. In one replica exchange process, four replicas were randomly selected in order and subjected to an exchange test.

J COMPUTING RESOURCES

The main computing resources used in our experiments are as follows:

- Laptop
  - CPU: Apple M2
  - Memory: 24GB
  - OS: macOS Ventura 13.6.9
- Desktop 1
  - CPU: Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz
  - Memory: 64GB
  - OS: Windows 10 Pro
- Desktop 2
  - CPU: Intel(R) Xeon(R) w7-3445 2.59 GHz
  - Memory: 64GB
  - OS: Windows 11 Pro for Workstations

K PROOF OF PROPOSITION 1

Herein, we prove Proposition 1 in general, i.e., we consider a general process to obtain an MCMC sample $\{z^{(t)}\}_{t=1,2,\dots}$ from an objective distribution $p(z)$ by the MH algorithm. (In our case, z and $p(z)$ should be replaced with $\boldsymbol{k}$ and $p(\boldsymbol{k}|\boldsymbol{x}^n, y^n)$, respectively.) In the tth iteration of the MH algorithm, z^* is generated from a proposal distribution $q(z^*|z^{(t-1)})$. Then, it will be accepted according to the following acceptance probability

$$A(z^*, z^{(t-1)}) \coloneqq \min\left\{1, \frac{p(z^*)q(z^{(t-1)}|z^*)}{p(z^{(t-1)})q(z^*|z^{(t-1)})}\right\}. \tag{48}$$

If z^* is accepted, we make $z^{(t)} \leftarrow z^*$, otherwise $z^{(t)} \leftarrow z^{(t-1)}$.

According to the above process, the transition probability $T(z^{(t)}|z^{(t-1)})$ is represented as follows.

$$T(z^{(t)}|z^{(t-1)}) = \begin{cases} q(z^{(t)}|z^{(t-1)})A(z^{(t)}, z^{(t-1)}), & z^{(t)} \neq z^{(t-1)}, \\ 1 - \sum_{z \neq z^{(t)}} q(z|z^{(t-1)})A(z, z^{(t-1)}), & z^{(t)} = z^{(t-1)}. \end{cases} \tag{49}$$

Therefore, if $q(z^*|z^{(t-1)})$ is time-invariant[9], then $T(z^{(t)}|z^{(t-1)})$ is also time-invariant. Moreover, if $q(z^*|z^{(t-1)}) > 0$ holds for any z^* and $z^{(t-1)}$, the Markov chain of $z^{(t)}$ is ergodic.

It is known that any time-invariant and ergodic Markov chain has a unique stationary distribution $p^*(z)$. It is also known that if any distribution $\tilde{p}(z)$ satisfies the following condition called detailed balance,

$$\tilde{p}(z)T(z'|z) = \tilde{p}(z')T(z|z') \quad \text{for any } z \text{ and } z', \tag{50}$$

then $\tilde{p}(z)$ is the stationary distribution, i.e., $\tilde{p}(z) = p^*(z)$. (It is a sufficient condition but not a necessary condition.)

Then, we prove the objective distribution $p(z)$ is the stationary distribution of the Markov chain whose transition probability is $T(z^{(t)}|z^{(t-1)})$ by showing the objective distribution satisfies the detailed balance. If $z = z'$, Eq. (50) clearly holds. If $z \neq z'$, we have

$$\begin{aligned} p(z)T(z'|z) &= p(z)q(z'|z)A(z', z) && (51) \\ &= p(z)q(z'|z) \min\left\{1, \frac{p(z')q(z|z')}{p(z)q(z'|z)}\right\} && (52) \\ &= \min\{p(z)q(z'|z), p(z')q(z|z')\} && (53) \\ &= p(z')q(z|z') \min\left\{\frac{p(z)q(z'|z)}{p(z')q(z|z')}, 1\right\} && (54) \\ &= p(z')q(z|z')A(z, z') && (55) \\ &= p(z')T(z|z'). && (56) \end{aligned}$$

Therefore, the objective distribution $p(z)$ is the stationary distribution of Markov chain induced from the aforementioned process. Consequently, the empirical distribution of $\{z^{(t)}\}_{t=1,2,\dots}$ converges to the objective distribution $p(z)$ after sufficient iterations.

[9]For $t \neq t'$, if $z^{(t-1)} = z^{(t'-1)}$ holds, then $q(z^*|z^{(t-1)}) = q(z^*|z^{(t'-1)})$ holds for any z^*.