Mondrian Forests for Large-Scale Regression when Uncertainty Matters

Balaji Lakshminarayanan
Gatsby Unit
University College London

Daniel M. Roy
Department of Statistical Sciences
University of Toronto

Yee Whye Teh
Department of Statistics
University of Oxford

Abstract

Many real-world regression problems demand a measure of the uncertainty associated with each prediction. Standard decision forests deliver efficient state-of-the-art predictive performance, but high-quality uncertainty estimates are lacking. Gaussian processes (GPs) deliver uncertainty estimates, but scaling GPs to large-scale data sets comes at the cost of approximating the uncertainty estimates. We extend Mondrian forests, first proposed by Lakshminarayanan et al. (2014) for classification problems, to the large-scale non-parametric regression setting. Using a novel hierarchical Gaussian prior that dovetails with the Mondrian forest framework, we obtain principled uncertainty estimates, while still retaining the computational advantages of decision forests. Through a combination of illustrative examples, real-world large-scale datasets, and Bayesian optimization benchmarks, we demonstrate that Mondrian forests outperform approximate GPs on large-scale regression tasks and deliver better-calibrated uncertainty assessments than decision-forest-based methods.

1 Introduction

Gaussian process (GP) regression is popular due to its ability to deliver both accurate non-parametric predictions and estimates of uncertainty for those predictions. The dominance of GP regression in applications such as Bayesian optimization, where uncertainty estimates are key to balance exploration and exploitation, is a testament to the quality of GP uncertainty estimates.

Unfortunately, the computational cost of GPs is cubic in the number of data points, making them computationally very expensive for large scale non-parametric regression tasks. (Specifically, we focus on the scenario where the number of data points $N$ is large, but the number of dimensions $D$ is modest.) Steady progress has been made over the past decade on scaling GP inference to big data, including some impressive recent work such as [6, 10, 12].

Ensembles of randomized decision trees, also known as decision forests, are popular for (non-probabilistic) non-parametric regression tasks, often achieving state-of-the-art predictive performance [3]. The most popular decision forest variants are random forests (Breiman-RF) introduced by Breiman [1] and extremely randomized trees (ERT) introduced by Geurts et al. [11]. The computational cost of learning decision forests is typically $O(N \log N)$ and the computation across the trees in the forest can be parallelized trivially, making them attractive for large scale regression tasks. While decision forests usually yield good predictions (as measured by, e.g., mean squared error or classification accuracy), the uncertainty estimates of decision forests are not as good as those produced by GPs. For instance, Jitkrit tum et al. [15] compare the uncertainty estimates of decision forests and GPs on a simple regression problem where the test distributions are different from the training distribution. As we move away from the training distribution, GP predictions smoothly return to the prior and exhibit higher uncertainty. However, the uncertainty estimates of decision forests are less smooth and do not exhibit this desirable property.

Our goal is to combine the desirable properties of GPs (good uncertainty estimates, probabilistic setup) with those of decision forests (computational speed). To this end, we extend Mondrian forests (MFs), introduced by Lakshminarayanan et al. [16] for classification tasks, to non-parametric regression tasks. Unlike usual decision forests, we use a probabilistic model within each tree to model the labels. Specifically, we use a hier-
archival Gaussian prior over the leaf node parameters and compute the posterior parameters efficiently using Gaussian belief propagation [19]. Due to special properties of Mondrian processes, their use as the underlying randomization mechanism results in a desirable uncertainty property: the prediction at a test point shrinks to the prior as the test point moves further away from the observed training data points. We demonstrate that, as a result, MFs yield better uncertainty estimates.

The paper is organized as follows: in Section 2, we briefly review decision trees and Mondrian trees. We present MFs for regression in Section 3 and discuss inference and prediction in detail. We present experiments in Section 5 that demonstrate that (i) MFs produce better uncertainty estimates than Breiman-RF and ERT when test distribution is different from training distribution, (ii) MFs outperform or achieve comparable performance to large scale approximate GPs in terms of mean squared error (MSE) or negative log predictive density (NLPD), thus making them well suited for large scale regression tasks where uncertainty estimates are important, and (iii) MFs outperform (or perform as well as) decision forests on Bayesian optimization tasks, where predictive uncertainty is important (since it guides the exploration-exploitation tradeoff). Finally, we discuss avenues for future work in Section 6.

2 Mondrian forests

Lakshminarayanan et al. [16] introduced Mondrian forests (MFs) for classification tasks. For completeness, we briefly review decision trees and Mondrian trees before describing how MFs can be applied to regression.

Our problem setup is the following: given $N$ labeled examples $(x_1, y_1), \ldots, (x_N, y_N) \in \mathbb{R}^D \times \mathbb{R}$ as training data, our task is to predict labels $y \in \mathbb{R}$ for unlabeled test points $x \in \mathbb{R}^D$ as well as provide corresponding estimates of uncertainty. Let $X_{1:n} := (x_1, \ldots, x_n)$, $Y_{1:n} := (y_1, \ldots, y_n)$, and $D_{1:n} := (X_{1:n}, Y_{1:n})$.

2.1 Decision trees

Following [16], a decision tree is a triple $(T, \delta, \xi)$ where $T$ is a finite, rooted, strictly binary tree and $\delta = (\delta_j)$ and $\xi = (\xi_j)$ specify, for every internal node $j \in T \setminus \text{leaves}(T)$, a split dimension $\delta_j \in \{1,\ldots,D\}$ and split location $\xi_j \in \mathbb{R}$. A decision tree represents a hierarchical partition of $\mathbb{R}^D$ into blocks, one for each node, as follows: At the root node, $\epsilon$, we have $B_\epsilon = \mathbb{R}^D$, while each internal node $j \in T \setminus \text{leaves}(T)$ represents a split of its parent’s block into two halves, with $\delta_j$ denoting the dimension of the split and $\xi_j$ denoting the location of the split. In particular,

$$B_{\text{left}(j)} := \{x \in B_j : x_{\delta_j} \leq \xi_j\} \quad \text{and} \quad B_{\text{right}(j)} := \{x \in B_j : x_{\delta_j} > \xi_j\}.$$

We will write $B_j := (\ell_{j1}, u_{j1}) \times \ldots \times (\ell_{jd}, u_{jd})$, where $\ell_{jd}$ and $u_{jd}$ denote the lower and upper bounds, respectively, of the rectangular block $B_j$ along dimension $d$. Put $\ell_j := (\ell_{j1}, \ell_{j2}, \ldots, \ell_{jd})$ and $u_j := (u_{j1}, u_{j2}, \ldots, u_{jd})$. Let $N(j)$ denote the indices of training data points at node $j$.

2.2 Mondrian trees and Mondrian forests

A Mondrian process [24] is a continuous-time Markov process $(M_t : t \geq 0)$, where, for every $t \geq s \geq 0$, $M_t$ is a hierarchical binary partition of $\mathbb{R}^D$ and a refinement of $M_s$. Mondrian trees [16] are restrictions of Mondrian processes to a finite set of points. (See Figure 1.) In particular, a Mondrian tree $T$ is a tuple $(T, \delta, \xi, \tau)$, where $(T, \delta, \xi)$ is a decision tree and $\tau = \{\tau_j\}_{j \in T}$ specifies a split time $\tau_j \geq 0$ with each node $j$. Split times increase with depth, i.e., $\tau_j > \tau_{\text{parent}(j)}$ and play an important role in online updates.

The expected depth of a Mondrian tree is parametrized by a non-negative lifetime parameter $\lambda > 0$. Since it is difficult to specify $\lambda$, Lakshminarayanan et al. [16] set $\lambda = \infty$ and stopped splitting a node if all the class labels of the data points within the node were identical. We follow a similar approach for regression: we do not split a node which has less than $\min \text{samples split}$ number of data points. Given a bound $\min \text{samples split}$ and training data $D_{1:n}$, the generative process for sampling Mondrian trees is described in Algorithms 1 and 2.

The process is similar to top-down induction of decision trees except for the following key differences: (i) splits

\[\text{Figure 1: (left) A Mondrian tree over six data points in } [0,1]^2. \text{ Every node in the tree represents a split and is embedded in time (vertical axis). (middle) An ordinary decision tree partitions the whole space. (right) In a Mondrian tree, splits are committed only within the range of the data in each block (denoted by gray rectangles). Let } j = \text{left}(\epsilon) \text{ be the left child of the root: then } B_j = (0,0.37) \times (0,1) \text{ is the block containing the red circles and } B_j^c \supseteq B_j \text{ is the smallest rectangle enclosing the two data points. (Adapted from [16, with permission.)}\]

\[\text{We refer to } y \in \mathbb{R} \text{ as label even though it is common in statistics to refer to } y \in \mathbb{R} \text{ as response instead of label.}\]
Algorithm 1 SampleMondrianTree($D_{1:n}$, min_samples_split)

1: Initialize: $T = \emptyset$, leaves($T$) = $\emptyset$, $\delta = \emptyset$, $\xi = \emptyset$, $\tau = \emptyset$, $N(\epsilon) = \{1, 2, \ldots, n\}$ $\triangleright$ initialize empty tree
2: SampleMondrianBlock($\epsilon, D_{N(\epsilon)}$, min_samples_split) $\triangleright$ Algorithm 2

Algorithm 2 SampleMondrianBlock($j, D_{N(j)}$, min_samples_split)

1: Add $j$ to $T$ and for all $d$, set $\ell_{jd} = \min \{X_{N(j),d}, u_{jd} = \max \{X_{N(j),d}\}$ $\triangleright$ dimension-wise min and max
2: if $|N(j)| \geq \text{min_samples_split}$ then $\triangleright$ $j$ is an internal node. $|N(j)|$ denotes \# data points.
3: Sample $E$ from exponential distribution with rate $\sum_d (u_{jd} - \ell_{jd})$
4: Set $\tau_j = \tau_{\text{parent(j)}} + E$
5: Sample split dimension $\delta_j$, choosing $d$ with probability proportional to $u_{jd} - \ell_{jd}$
6: Sample split location $\xi_j$ uniformly from interval $[\ell_{j\delta_j}, u_{j\delta_j}]$
7: Set $N(\text{left}(j)) = \{n \in N(j) : X_{n,\delta_j} \leq \xi_j\}$ and $N(\text{right}(j)) = \{n \in N(j) : X_{n,\delta_j} > \xi_j\}$
8: SampleMondrianBlock($\text{left}(j), D_{N(\text{left}(j))}$, min_samples_split)
9: SampleMondrianBlock($\text{right}(j), D_{N(\text{right}(j))}$, min_samples_split)
10: else $\triangleright$ $j$ is a leaf node
11: Set $\tau_j = \infty$ and add $j$ to leaves($T$)

in a Mondrian tree are committed only within the range of training data (see Figure 1), and (ii) the split dimensions and locations are chosen independent of the labels and uniformly within $B_j^d$ (see lines 5, 6 of Algorithm 2). A Mondrian forest consists of $M$ i.i.d. Mondrian trees $T_m = (T_m, \delta_m, \xi_m, \tau_m)$ for $m = 1, \ldots, M$. See [16] for further details.

Mondrian trees can be updated online in an efficient manner and remarkably, the distribution of trees sampled from the online algorithm is identical to the corresponding batch counterpart [16]. We use the batch version of Mondrian forests (Algorithms 1 and 2) in all of our experiments except the Bayesian optimization experiment in section 5.3. Since we do not evaluate the computational advantages of online Mondrian forest, using a batch Mondrian forest in the Bayesian optimization experiment would not affect the reported results. For completeness, we describe the online updates in Algorithms 3 and 4 in the supplementary material.

3 Model, hierarchical prior, and predictive posterior for labels

In this section, we describe a probabilistic model that will determine the predictive label distribution, $p_T(y|x, D_{1:N})$, for a tree $T = (T, \delta, \xi, \tau)$, dataset $D_{1:N}$, and test point $x$. Let leaf($x$) denote the unique leaf node $j \in \text{leaves}(T)$ such that $x \in B_j$. Like with Mondrian forests for classification, we want the predictive label distribution at $x$ to be a smoothed version of the empirical distribution of labels for points in $B_{\text{leaf}(x)}$ and in $B_j$ for nearby nodes $j'$. We will also achieve this smoothing via a hierarchical Bayesian approach: every node is associated with a label distribution, and a prior is chosen under which the label distribution of a node is similar to that of its parent’s. The predictive $p_T(y|x, D_{1:N})$ is then obtained via marginalization.

As is common in the decision tree literature, we assume the labels within each block are independent of $X$ given the tree structure. Lakshminarayanan et al. [16] used a hierarchy of normalized stable processes (HNSP) prior for classification problems. In this paper, we focus on the case of real-valued labels. Let $N(\mu, \nu)$ denote a Gaussian distribution with mean $\mu$ and variance $\nu$. For every $j \in T$, let $\mu_j$ be a mean parameter (of a Gaussian distribution over the labels) at node $j$, and let $\mu = \{\mu_j : j \in T\}$. We assume the labels within a leaf node are Gaussian distributed:

$$y_n | T, \mu \sim N(\mu_{\text{leaf}(x_n)}, \sigma_y^2)$$

where $\sigma_y^2$ is a parameter specifying the variance of the (measurement) noise.

We use the following hierarchical Gaussian prior for $\mu$: For hyperparameters $\mu_H, \gamma_1, \gamma_2$, let

$$\mu \mid \mu_H \sim N(\mu_H, \phi_0), \quad \mu_j \mid \mu_{\text{parent}(j)} \sim N(\mu_{\text{parent}(j)}, \phi_j),$$

where $\phi_j = \gamma_1 \sigma(\gamma_2 \tau_j) - \gamma_1 \sigma(\gamma_2 \tau_{\text{parent}(j)})$ with the convention that $\tau_{\text{parent}(\epsilon)} = 0$, and $\sigma(t) = (1 + e^{-t})^{-1}$ denotes the sigmoid function.

Before discussing the details of posterior inference, we provide some justification for the details of this model: Recall that $\tau_j$ increases as we go down the tree, and so the use of the sigmoid $\sigma(\cdot)$ encodes the prior assumption that children are expected to be more similar to their parents as depth increases. The Gaussian hierarchy is closed under marginalization, i.e.,

$$\mu \mid \mu_H \sim N(\mu_H, \phi_0), \quad \mu_0 \mid \mu_H \sim N(\mu_H, \phi_0),$$

$$\mu_0 \mid \mu_H \sim N(\mu_H, \phi_0 + \phi_0),$$
where \( \phi_x + \phi_0 = \gamma_1 \sigma (\gamma_2 \tau) - \gamma_1 \sigma (\gamma_2 0) + \gamma_1 \sigma (\gamma_2 0) - \gamma_1 \sigma (\gamma_2 0) = \gamma_1 \sigma (\gamma_2 0) - \gamma_1 \sigma (\gamma_2 0) \). Therefore, we can introduce intermediate nodes without changing the predictive distribution. In Section 3.3, we show that a test data point can branch off into its own node: the hierarchical prior is critical for smoothing predictions.

Given training data \( \mathcal{D}_{1:N} \), our goal is to compute the posterior density over \( \mu \):

\[
    p_T(\mu | \mathcal{D}_{1:N}) \propto p_T(\mu) \prod_{n=1}^{N} \mathcal{N}(y_n | \mu_{\text{leaf}(x_n)}, \sigma^2_y). \tag{2}
\]

The posterior over \( \mu \) will be used during the prediction step described in Section 3.3. Note that the posterior over \( \mu \) is computed independently for each tree, and so can be parallelized trivially.

### 3.1 Gaussian belief propagation

We perform posterior inference using belief propagation [21]. Since the prior and likelihood are Gaussian, all the messages can be computed analytically and the posterior over \( \mu \) is also Gaussian. Since the hierarchy has a tree structure, the posterior can be computed in time that scales linearly with the number of nodes in the tree, which is typically \( O(N) \), hence posterior inference is efficient compared to non-tree-structured Gaussian processes whose computational cost is typically \( O(N^3) \). Message passing in trees is a well-studied problem, and so we refer the reader to [19, Chapter 20] for details.

### 3.2 Hyperparameter heuristic

In this section, we briefly describe how we choose the hyperparameters \( \theta = \{ \mu_H, \gamma_1, \gamma_2, \sigma^2_y \} \). More details can be found in Appendix B in the supplementary material. For simplicity, we use the same values of these hyperparameters for all the trees; it is possible to optimize these parameters for each tree independently, and would be interesting to evaluate this extra flexibility empirically. Ideally, one might choose hyperparameters by optimizing the marginal likelihood (computed as a byproduct of belief propagation) by, e.g., gradient descent. We use a simpler approach here: we maximize the product of the individual label marginals, assuming an individual label noise, which yields closed-form solutions for \( \mu_H \) and \( \gamma_1 \). This approach does not yield an estimate for \( \gamma_2 \), and so, using the fact that \( \tau \) increases with \( N \), we pre-process the training data to lie in \([0,1]^D\) and set \( \gamma_2 = D/(20 \log_2 N) \) based on the reasoning that (i) \( \tau \) is inversely proportional to \( D \) and (ii) \( \tau \) increases with tree depth and the tree depth is \( O(\log_2 N) \) assuming balanced trees.\(^{4}\)

\(^{4}\)In Lakshminarayanan et al. [16], it was observed that the average tree depths were 2-3 times \( \log_2(N) \) in practice.

Lakshminarayanan et al. [16] proposed to stop splitting a Mondrian block whenever all the class labels were identical.\(^{4}\) We adopt a similar strategy here and stop splitting a Mondrian block if the number of samples is fewer than a parameter \( \text{min}_\text{samples}_\text{split} \). It is common in decision forests to require a minimum number of samples in each leaf, for instance Breiman [1] and Geurts et al. [11] recommend setting \( \text{min}_\text{samples}_\text{leaf} = 5 \) for regression problems. We set \( \text{min}_\text{samples}_\text{split} = 10 \).

### 3.3 Predictive variance computation

The prediction step in a Mondrian regression tree is similar to that in a Mondrian classification tree [16, Appendix B] except that at each node of the tree, we predict a Gaussian posterior over \( y \) rather than predict a posterior over class probabilities. Recall that a prediction from a vanilla decision tree is just the average of the training labels in \text{leaf}(x) \). Unlike decision trees, in a Mondrian tree, a test point \( x \) can potentially ‘branch off’ the existing Mondrian tree at any point along the path from root to \text{leaf}(x) \). Hence, the predictive posterior over \( y \) from a given tree \( T \) is a mixture of Gaussians of the form

\[
    p_T(y, \mathcal{D}_{1:N}) = \sum_{j \in \text{path(leaf}(x)))} w_j \mathcal{N}(y | m_j, v_j) \tag{3}
\]

where \( w_j \) denotes the weight of each component, given by the probability of branching off just before reaching node \( j \), and \( m_j, v_j \) respectively denote the predictive mean and variance. The probability of branching off increases as the test point moves further away from the training data at that particular node; hence, the predictions of MFs exhibit higher uncertainty as we move farther from the training data. For completeness, we provide pseudocode for computing (3) in Algorithm 5 of the supplementary material.

If a test data point branches off to create a new node, the predictive mean at that node is the posterior of the parent of the new node; if we did not have a hierarchy and instead assumed the predictions at leaves were i.i.d, then branching would result in a prediction from the prior, which would lead to biased predictions in most applications. The predictive mean and variance for the mixture of Gaussians are

\[
    \mathbb{E}_T[y] = \sum_j w_j m_j \quad \text{and} \quad \text{Var}_T[y] = \sum_j w_j (v_j + m^2_j) - (\mathbb{E}_T[y])^2, \tag{4}
\]
and the prediction of the ensemble is then

$$p(y|x, D_{1:N}) = \frac{1}{M} \sum_{m} p_{T_m}(y|x, D_{1:N}). \quad (5)$$

The prediction of the ensemble can be thought of as being drawn from a mixture model over $M$ trees where the trees are weighted equally. The predictive mean and variance of the ensemble can be computed using the formula for mixture of Gaussians similar to (4). Similar strategy has been used in [4, 14] as well.

4 Related work

The work on large scale Gaussian processes can be broadly split into approaches that optimize inducing variables using variational approximations and approaches that distribute computation by using experts that operate on subsets of the data. We refer to [6] for a recent summary of large scale Gaussian processes. Hensman et al. [12] and Gal et al. [10] use stochastic variational inference to speed up GPs, building on the variational bound developed by Titsias [28]. Deisenroth and Ng [6] present the robust Bayesian committee machine (rBCM), which combines predictions from experts that operate on subsets of data.

Hutter [13] investigated the use of Breiman-RF for Bayesian optimization and used the empirical variance between trees in the forest as a measure of uncertainty. (Hutter et al. [14] proposed a further modification, see Appendix C.) Esfami et al. [9] used a non-standard decision forest implementation where a quadratic regressor is fit at each leaf node, rather than a constant regressor as in popular decision forest implementations. Their uncertainty measure—a sum of the Kullback-Leibler (KL) divergence—is highly specific to their application of accelerating expectation propagation, and so it seems their method is unlikely to be immediately applicable to general non-parametric regression tasks. Indeed, Jitkrittum et al. [15] demonstrate that the uncertainty estimates proposed by [9] are not as good as kernel methods in their application domain when the test distribution is different from the training distribution. As originally defined, Mondrian forests produce uncertainty estimates for categorical labels, but Lakshminarayanan et al. [16] evaluated their performance on (online) prediction (classification accuracy) without any assessment of the uncertainty estimates.

5 Experiments

5.1 Comparison of uncertainty estimates of MFs to popular decision forests

In this experiment, we compare uncertainty estimates of MFs to those of popular decision forests. The prediction of MFs is given by (5), from which we can compute the predictive mean and predictive variance.

For decision forests, we compute the predictive mean as the average of the predictions from the individual trees and, following Hutter [13, §11.1.3], compute the predictive variance as the variance of the predictions from the individual trees. We use 25 trees and set $\text{min\_samples\_leaf} = 5$ for decision forests to make them comparable to MFs with $\text{min\_samples\_split} = 10$. We use the ERT and Breiman-RF implementation in scikit-learn [22] and set the remaining hyperparameters to their default values.

We use a simplified version of the dataset described in [15], where the goal is to predict the outgoing message in expectation propagation (EP) from a set of incoming messages. When the predictions are uncertain, the outgoing message will be re-computed (either numerically or using a sampler), hence predictive uncertainty is crucial in this application. Our dataset consists of two-dimensional features (which are derived from the incoming message) and a single target (corresponding to mean of the outgoing message). The scatter plot of the training data features is shown in Fig. 2(a). We evaluate predictive uncertainty on two test distributions, shown in red and blue in Fig. 2(a), which contain data points in unobserved regions of the training data.

The mean squared error of all the methods are comparable, so we focus just on the predictive uncertainty. Figures 2(b), 2(c), and 2(d) display the predictive uncertainty of MF, ERT and Breiman-RF as a function of $x_1$. We notice that Breiman-RF’s predictions are over-confident compared to MF and ERT. The predictive variance is quite low even in regions where training data has not been observed. The predictive variance of MF is low in regions where training data has been observed ($-5 < x_1 < 5$) and goes up smoothly as we move farther away from the training data; the red test dataset is more similar to the training data than the blue test data and the predictive uncertainty of MF on the blue dataset is higher than that of the red dataset, as one would expect. ERT is less overconfident than Breiman-RF, however its predictive uncertainty is less smooth compared to that of MF.

5.2 Comparison to GPs and decision forests on flight delay dataset

In this experiment, we compare decision forest variants to large scale Gaussian processes. Deisenroth and Ng [6] evaluated a variety of large scale Gaussian processes on the flight delay dataset, processed by Hensman et al. [12], and demonstrate that their method achieves state-of-the-art predictive performance; we evaluate decision

$^5$ Code available from the authors’ websites.
We evaluate performance by measuring the root mean squared error (RMSE) and negative log predictive density (NLPD). NLPD, defined as the negative logarithm of (5), is a popular measure for evaluating predictive uncertainty (cf. [23, section 4.2]). NLPD penalizes over-confident as well as under-confident predictions since it is not only accounts for predictive mean but also the predictive variance. RF and ERT do not offer a principled way to compute NLPD. But, as a simple approximation, we computed NLPD for RF and ERT assuming a Gaussian distribution with mean equal to the average of trees’ predictions, variance equal to the variance of trees’ predictions.

Table 1 presents the results. The RMSE and NLPD results for SVI-GP, Dist-VGP and rBCM results were taken from [6], who report a standard error lower than 0.3 for all of their results. Table 1 in [6] shows that rBCM achieves significantly better performance than other large scale GP approximations; hence we only report the performance of rBCM here. It is important to note that the dataset exhibits non-stationarity: as a result, the performance of decision forests as well as GPs is worse on the larger datasets. (This phenomenon was observed by Gal et al. [10] and Deisenroth and Ng [6] as well.) On the 700K and 2M dataset, we observe that decision forests achieve significantly lower RMSE than rBCM. MF achieves significantly lower NLPD compared to rBCM, which suggests that its uncertainty estimates are useful for large scale regression tasks. However, all the decision forests, including MFs, achieve poorer RMSE performance than rBCM on the larger 5M dataset. We believe that this is due to the non-stationary nature of the data. To test this hypothesis, we shuffled the 5,100,000 data points to create three new training (test) data sets with 5M (100K) points; all the decision forests achieved a RMSE in the range 31-34 on these shuffled datasets.

MF outperforms rBCM in terms of NLPD on all three datasets. On the 5M dataset, the NLPD of Breiman-RF is similar to that of MF, however Breiman-RF’s uncertainty is not computed in a principled fashion. As an additional measure of uncertainty, we report probability calibration measures (akin to those for binary classification cf. http://scikit-learn.org/stable/modules/calibration.html), also known as reliability diagrams [5], for MF, Breiman-RF and ERT. First, we compute the z% (e.g. 90%) prediction interval for each test data point based on Gaussian quantiles using predictive mean and variance. Next, we measure what fraction of test observations fall within this prediction interval. For a well-calibrated regressor, the observed fraction should be close to z%. We compute observed fraction for z = 10% to z = 90% in increments of 10. We report observed fraction minus ideal fraction since it is easier to interpret—a value of zero implies perfect calibration, a negative value implies over-confidence (a
Table 1: Comparison of MFs to popular decision forests and large scale GPs on the flight delay dataset. We report
average results over 3 runs (with random initializations), along with standard errors. MF achieves significantly better
NLPD than rBCM. RF and ERT do not offer a principled way to compute NLPD, hence they are marked with an asterisk.

| Dataset | Method  | 700K/100K | 2M/100K | 5M/100K |
|---------|---------|-----------|---------|---------|
|         | RMSE    | NLPD      | RMSE    | NLPD    | RMSE    | NLPD    |
| SVI-GP  | 33.0    | -         | -       | -       | -       | -       |
| Dist-VGP| 33.0    | -         | -       | -       | -       | -       |
| rBCM    | 27.1    | 9.1       | 34.4    | 8.4     | 35.5    | 8.8     |
| RF      | 24.07 ± 0.02 | 5.06 ± 0.02* | 27.3 ± 0.01 | 5.19 ± 0.02* | 39.47 ± 0.02 | 6.90 ± 0.05* |
| ERT     | 24.32 ± 0.02 | 6.22 ± 0.03* | 27.95 ± 0.02 | 6.16 ± 0.01* | 38.38 ± 0.02 | 8.41 ± 0.09* |
| MF      | 26.57 ± 0.04 | 4.89 ± 0.02 | 29.46 ± 0.02 | 4.97 ± 0.01 | 40.13 ± 0.05 | 6.91 ± 0.06 |

Table 2: Comparison of MFs to popular decision forests on the flight delay dataset. Each entry denotes the difference
between the observed fraction minus the ideal fraction (which is shown at the top of the column). Hence, a value of zero
implies perfect calibration, a negative value implies overconfidence and a positive value implies under-confident predictor.
MF is better calibrated than Breiman-RF and ERT, which are consistently over-confident.

| Dataset | Method  | 0.1  | 0.2  | 0.3  | 0.4  | 0.5  | 0.6  | 0.7  | 0.8  | 0.9  |
|---------|---------|------|------|------|------|------|------|------|------|------|
| 700K    | Breiman-RF | -0.02 | -0.04 | -0.05 | -0.06 | -0.06 | -0.06 | -0.05 | -0.06 | -0.07 |
| 700K    | ERT     | -0.04 | -0.07 | -0.11 | -0.14 | -0.16 | -0.18 | -0.19 | -0.19 | -0.18 |
| 700K    | MF      | -0.01 | -0.02 | -0.01 | 0     | 0.02 | 0.03 | 0.03 | 0.02 | 0    |
| 2M      | Breiman-RF | -0.02 | -0.04 | -0.05 | -0.06 | -0.05 | -0.04 | -0.03 | -0.03 | -0.04 |
| 2M      | ERT     | -0.04 | -0.08 | -0.12 | -0.15 | -0.17 | -0.18 | -0.19 | -0.18 | -0.16 |
| 2M      | MF      | -0.02 | -0.04 | -0.05 | -0.05 | -0.03 | 0    | 0.02 | 0.03 | 0.01 |
| 5M      | Breiman-RF | -0.03 | -0.06 | -0.08 | -0.09 | -0.1  | -0.1  | -0.11 | -0.1  | -0.1  |
| 5M      | ERT     | -0.04 | -0.07 | -0.11 | -0.14 | -0.16 | -0.18 | -0.19 | -0.19 | -0.18 |
| 5M      | MF      | -0.02 | -0.04 | -0.05 | -0.06 | -0.06 | -0.05 | -0.05 | -0.05 | -0.07 |

5.3 Scalable Bayesian optimization

Next, we showcase the usefulness of MFs in a Bayesian optimization (BayesOpt) task. We briefly review the
Bayesian optimization setup for completeness and refer the interested reader to [2, 25] for further details.
Bayesian optimization deals with the problem of identifying the global maximizer (or minimizer) of an
unknown (a.k.a. black-box) objective function which is computationally expensive to evaluate.\footnote{For a concrete example, consider the task of optimizing the hyperparameters of a deep neural network to maximize validation accuracy.} Our goal is to
identify the maximizer in as few evaluations as possible. Bayesian optimization is a model-based sequential
search approach to solve this problem. Specifically, given $n$ noisy observations, we fit a surrogate model
such as a Gaussian process or a decision forest and choose the next location based on an acquisition function such as upper confidence bound (UCB) [27] or expected improvement (EI) [18]. The acquisition function
trades off exploration versus exploitation by choosing input locations where the predictive mean from the
surrogate model is high (exploitation) or the predictive uncertainty of the surrogate model is high (exploration).
Hence, a surrogate model with well-calibrated predictive uncertainty is highly desirable. Moreover, the
surrogate model has to be re-fit after every new observation is added; while this is not a significant limitation
for a few (e.g. 50) observations and scenarios where the evaluation of the objective function is significantly
more expensive than re-fitting the surrogate model, the re-fitting can be computationally expensive if one is
interested in scalable Bayesian optimization [26].

Hutter [13] proposed sequential model-based algorithm configuration (SMAC), which uses Breiman-RF as the
surrogate model and the uncertainty between the trees as a heuristic measure of uncertainty.\footnote{Hutter et al. [14, §4.3.2] proposed a further modification to the variance estimation procedure, where each
tree outputs a predictive mean and variance, in the spirit of \textit{quantile regression forests} [17]. See Appendix C for a discussion on how this relates to MFs.} Nickson et al.
We report results on four Bayesian optimization benchmarks used in [7, 26], consisting of two synthetic functions namely the Branin and Hartmann functions, and two real-world problems, namely optimizing the hyper-parameters of online latent Dirichlet allocation (LDA) and structured support vector machine (SVM). LDA and SVM datasets consist of 288 and 1400 grid points respectively; we sampled Branin and Hartmann functions at 250,000 grid points (to avoid implementing a separate optimizer for optimizing over the acquisition function). For SVM and LDA, some dimensions of the grid vary on a non-linear scale (e.g. $10^0, 10^{-1}, 10^{-2}$); we log-transformed such dimensions and scaled all dimensions to [0, 1] so that all features are on the same scale. We used 10 trees, set min_samples_split = 2 and use UCB as the acquisition function\(^9\) for MFs. We repeat our results 15 times (5 times each with 3 different random grids for Branin and Hartmann) and report mean and standard deviation.

Following Eggensperger et al. [7], we evaluate a fixed number of evaluations for each benchmark and measure the maximum value observed. The results are shown in Table 3. The SMAC results (using Breiman-RF) were taken from Table 2 of [7]. Both MF and SMAC identify the optimum for LDA-grid. SMAC does not identify the optimum for Branin and Hartmann functions. We observe that MF finds maxima very close to the true maximum on the grid, thereby suggesting that better uncertainty estimates are useful for better exploitation.

\(^9\)Specifically, we set acquisition function = predictive mean + predictive standard deviation.

### Table 3: Results on BayesOpt benchmarks

| Dataset (D, #evals) | Oracle       | MF           | SMAC [7]     |
|---------------------|--------------|--------------|--------------|
| Branin (2, 200)     | -0.398       | -0.400 ± 0.005 | -0.655 ± 0.27 |
| Hartmann (6, 200)   | -3.322       | 3.247 ± 0.109  | 2.977 ± 0.11  |
| SVM-grid (3, 100)   | -1266.2      | -1266.36 ± 0.52 | -1269.6 ± 2.9 |
| LDA-grid (3, 50)    | -24.1        | -24.1 ± 0     | -24.1 ± 0.1   |

No method is a panacea: here we discuss two failure modes of our approach that would be important to address in future work. First, we expect GPs to perform better than decision forests on extrapolation tasks: a GP with an appropriate kernel (and well-estimated hyperparameter values) can extrapolate beyond the observed range of training data; however, the predictions of decision forests with constant predictors at leaf nodes are confined to the range of minimum and maximum observed $y$. If extrapolation is desired, we need complex regressors (that are capable of extrapolation) at leaf nodes of the decision forest. However, this will increase the cost of posterior inference. Second, MFs choose splits independent of the labels; hence irrelevant features can hurt predictive performance [16]; in the batch setting, one can apply feature selection to filter or down weight the irrelevant features.

### 6 Discussion

We developed a novel and scalable methodology for regression based on Mondrian forests that provides both good predictive accuracy as well as sensible estimates of uncertainty. These uncertainty estimates are important in a range of application areas including probabilistic numerics, Bayesian optimization and planning. Using a large-scale regression application on flight delay data, we demonstrate that our proposed regression framework can provide both state-of-the-art RMSE and estimates of uncertainty as compared to recent scalable GP approximations. We demonstrate that Mondrian forests deliver better-calibrated uncertainty estimates than existing decision forests, especially in regions far away from the training data. Since Mondrian forests deliver good uncertainty estimates and can be trained online efficiently, they seem promising for applications such as Bayesian optimization and reinforcement learning.
Acknowledgments

We thank Wittawat Jitkrittum for sharing the dataset used in [15] and helpful discussions. We thank Katharina Eggensperger, Frank Hutter and Ziyu Wang for helpful discussions on Bayesian optimization. BL gratefully acknowledges generous funding from the Gatsby Charitable Foundation. This research was carried out in part while DMR held a Research Fellowship at Emmanuel College, Cambridge, with funding also from a Newton International Fellowship through the Royal Society. YWT’s research leading to these results has received funding from EPSRC (grant EP/K009362/1) and the ERC under the EU’s FP7 Programme (grant agreement no. 617411).

References

[1] L. Breiman. Random forests. Mach. Learn., 45: 5–32, 2001.
[2] E. Brochu, V. M. Cora, and N. De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv preprint arXiv:1012.2599, 2010.
[3] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In Proc. Int. Conf. Mach. Learn. (ICML), 2006.
[4] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. Found. Trends Comput. Graphics and Vision, 7(2–3):81–227, 2012.
[5] M. H. DeGroot and S. E. Fienberg. The comparison and evaluation of forecasters. The statistician, pages 12–22, 1983.
[6] M. P. Deisenroth and J. W. Ng. Distributed Gaussian processes. In Proc. Int. Conf. Mach. Learn. (ICML), 2015.
[7] K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown. Towards an empirical foundation for assessing Bayesian optimization of hyperparameters. In NIPS workshop on Bayesian Optimization in Theory and Practice, 2013.
[8] D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. J. Mach. Learn. Res. (JMLR), 6:503–556, 2005.
[9] S. A. Eslami, D. Tarlow, P. Kohli, and J. Winn. Just-in-time learning for fast and flexible inference. In Adv. Neural Information Proc. Systems (NIPS), 2014.
[10] Y. Gal, M. van der Wilk, and C. Rasmussen. Distributed variational inference in sparse Gaussian process regression and latent variable models. In Adv. Neural Information Proc. Systems (NIPS), 2014.
[11] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. Mach. Learn., 63(1):3–42, 2006.
[12] J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In Conf. Uncertainty Artificial Intelligence (UAI), 2013.
[13] F. Hutter. Automated configuration of algorithms for solving hard computational problems. PhD thesis, University of British Columbia, 2009.
[14] F. Hutter, L. Xu, H. H. Hoos, and K. Leyton-Brown. Algorithm runtime prediction: Methods & evaluation. Artificial Intelligence, 206:79–111, 2014.
[15] W. Jitkrittum, A. Gretton, N. Heess, S. Eslami, B. Lakshminarayanan, D. Sejdinovic, and Z. Szabó. Kernel-based just-in-time learning for passing expectation propagation messages. In Conf. Uncertainty Artificial Intelligence (UAI), 2015.
[16] B. Lakshminarayanan, D. M. Roy, and Y. W. Teh. Mondrian forests: Efficient online random forests. In Adv. Neural Information Proc. Systems (NIPS), 2014.
[17] N. Meinshausen. Quantile regression forests. J. Mach. Learn. Res. (JMLR), 7:983–999, 2006.
[18] J. Mockus, V. Tiesis, and A. Zilinskas. The application of Bayesian methods for seeking the extremum. Towards Global Optimization, 2(117-129): 2, 1978.
[19] K. P. Murphy. Machine Learning: a Probabilistic Perspective. MIT press, 2012.
[20] T. Nickson, M. A. Osborne, S. Reece, and S. J. Roberts. Automated machine learning on big data using stochastic algorithm tuning. arXiv preprint arXiv:1407.7969, 2014.
[21] J. Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., 1988.
[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. (JMLR), 12:2825–2830, 2011.
[23] J. Quinonero-Candela, C. E. Rasmussen, F. Sinz, O. Bousquet, and B. Schölkopf. Evaluating predictive uncertainty challenge. In Machine Learning
Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment, pages 1–27. Springer, 2006.

[24] D. M. Roy and Y. W. Teh. The Mondrian process. In Adv. Neural Information Proc. Systems (NIPS), 2009.

[25] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In Adv. Neural Information Proc. Systems (NIPS), 2012.

[26] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Ali, and R. P. Adams. Scalable Bayesian optimization using deep neural networks. In Proc. Int. Conf. Mach. Learn. (ICML), 2015.

[27] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In Proc. Int. Conf. Mach. Learn. (ICML), 2010.

[28] M. K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In Int. Conf. Artificial Intelligence Stat. (AISTATS), 2009.
Mondrian Forests for Large-Scale Regression when Uncertainty Matters: Supplementary material

A  Pseudocode for online learning and prediction

The online updates are shown in Algorithms 3 and 4. The prediction step is detailed in Algorithm 5.

**Algorithm 3** ExtendMondrianTree($T, D, \text{min}_\text{samples}_\text{split}$)

1: Input: Tree $T = (T, \delta, \xi, \tau)$, new training instance $D = (x, y)$
2: ExtendMondrianBlock($T, \epsilon, D, \text{min}_\text{samples}_\text{split}$) \> Algorithm 4

**Algorithm 4** ExtendMondrianBlock($T, j, D, \text{min}_\text{samples}_\text{split}$)

1: Set $e^f = \max(\ell^f_j - x, 0)$ and $e^u = \max(x - u^f_j, 0)$ \> $e^f = e^u = 0_D$ if $x \in B_j^\tau$
2: Sample $E$ from exponential distribution with rate $\sum_d(e^f_d + e^u_d)$
3: if $\tau_{\text{parent}(j)} + E < \tau_j$ then \> introduce new parent for node $j$
4: Sample split dimension $\delta$, choosing $d$ with probability proportional to $e^f_d + e^u_d$
5: Sample split location $\xi$ uniformly from interval $[u^f_j, \delta, x_j]$ if $x_j > u^f_j, \delta$ else $[x_j, \ell^f_j, \delta]$.
6: Insert a new node $j$ just above node $j$ in the tree, and a new leaf $j''$, sibling to $j$, where
7: $\delta_j = \delta, \xi_j = \xi, \tau_j = \tau_{\text{parent}(j)} + E$, $\ell^f_j = \min(\ell^f_j, x)$, $u^f_j = \max(u^f_j, x)$
8: $j'' = \text{left}(j)$ if $x_{\delta_j} \leq \xi_j$
9: SampleMondrianBlock($j'', D, \text{min}_\text{samples}_\text{split}$)
10: else
11: Update $\ell^f_j \leftarrow \min(\ell^f_j, x), u^f_j \leftarrow \max(u^f_j, x)$ \> update extent of node $j$
12: if $j \notin \text{leaves}(T)$ then \> return if $j$ is a leaf node, else recurse down the tree
13: if $x_{\delta_j} \leq \xi_j$ then child$(j) = \text{left}(j)$ else child$(j) = \text{right}(j)$
14: ExtendMondrianBlock($T, \text{child}(j), D, \text{min}_\text{samples}_\text{split}$) \> recurse on child containing $D$

**Algorithm 5** Predict($T, x$)

1: \> Description of prediction using a Mondrian tree given by (3).
2: \> The predictive mean, predictive variance and NLPD computation are not shown, but they can be computed easily during the top-down pass using the weights $w_j$ and posterior moments $m_j, v_j$ at node $j$.
3: Initialize $j = \epsilon$ and $p_{\text{NotSeparatedYet}} = 1$
4: while True do
5: Set $\Delta_j = \tau_j - \tau_{\text{parent}(j)}$ and $\eta_j(x) = \sum_d(\max(x_d - u^f_{\delta_d}, 0) + \max(\ell^f_{\delta_d} - x_d, 0))$
6: Set $p^f_j(x) = 1 - \exp(-\Delta_j \eta_j(x))$
7: if $p^f_j(x) > 0$ then
8: $w_j = p_{\text{NotSeparatedYet}} p^f_j(x)$
9: if $j \in \text{leaves}(T)$ then
10: $w_j = p_{\text{NotSeparatedYet}} (1 - p^f_j(x))$
11: return
12: else
13: if $x_{\delta_j} \leq \xi_j$ then $j \leftarrow \text{left}(j)$ else $j \leftarrow \text{right}(j)$ \> recurse to the child where $x$ lies

B  Choosing the hyperparameters

In this appendix, we give more details on how we choose the hyper parameters $\theta = \{\mu_H, \gamma_1, \gamma_2, \sigma_x^2\}$. For simplicity, we used the same values of these hyperparameters for all the trees; it is possible to optimize these parameters for each tree independently.
We optimize the product of label marginals, integrating out $\mu$ for each label individually, i.e.,

$$q(Y|\theta, T) = \prod_{j \in \text{leaves}(T)} \prod_{n \in N(j)} N(y_n|\mu_H, \phi_j - \phi_{\text{parent}(e)} + \sigma^2_y).$$

Since $\tau_j = \infty$ at the leaf nodes, we have

$$\phi_j - \phi_{\text{parent}(e)} = \gamma_1 \sigma(\gamma_2 \tau_j) - \gamma_1 \sigma(\gamma_2 0) = \gamma_1 (\sigma(\infty) - \sigma(0)) = \gamma_1 2.$$

If the noise variance is known, $\sigma^2_y$ can be set to the appropriate value. In our case, the noise variance is unknown; hence, we parametrize $\sigma^2_y$ as $\gamma_1 / K$ and set $K = \min(2000, 2N)$ to ensure that the noise variance $\sigma^2_y$ is a non-zero fraction of the total variance $\gamma_1 / 2 + \gamma_1 / K$.

We maximize $q(Y|\theta, T)$ over $\mu_H$, $\gamma_1$, and $K$, leading to

$$\mu_H = \frac{1}{N} \sum_n y_n,$$

$$\gamma_1 \left( \frac{1}{2} + \frac{1}{K} \right) = \frac{1}{N} \sum_n (y_n - \mu_H)^2.$$

Note that we could have instead performed gradient descent on the actual marginal likelihood produced as a byproduct of belief propagation. It would be interesting to investigate this.

The likelihood $q(Y|\theta, T)$ does not depend on $\gamma_2$, and so we cannot choose $\gamma_2$ by optimizing it. We know, however, that $\tau$ increases with $N$. Moreover, Lakshminarayanan et al. [16] observed that the average tree depths were 2-3 times $\log_2(N)$ in practice. We therefore pre-process the training data to lie in $[0, 1]^D$ and set $\gamma_2 = \frac{D}{20 \log_2 N}$ since (i) $\tau$ increases with tree depth and the tree depth is $O(\log_2 N)$ assuming balanced trees and (ii) $\tau$ is inversely proportional to $D$. In Appendix C, we describe a fast approximation which does not involve estimation of $\gamma_1, \gamma_2$.

## Fast approximation to message passing and hyperparameter estimation

In Section 5.3, we suggested a fast $O(\log n)$ approximation to exact message passing which costs $O(n)$. Under this approximation, the Gaussian posterior at each node is approximated by a Gaussian distribution whose mean and variance are given by the empirical mean and variance of the data points at that node. This approximation is better suited for online applications since adding a new data point involves just updating mean and variance for all the nodes along the path from root to a leaf. Another advantage of this approximation is that we only need to set the noise variance $\sigma^2_y$ and do not need to set the hyper-parameters $\{\mu_H, \gamma_1, \gamma_2\}$.

Since our initial publication, we have learnt that this Gaussian posterior approximation is similar to a random forest modification independently proposed in Hutter et al. [14, §4.3.2]. In [14], each tree outputs a predictive mean and variance equal to the empirical mean and variance of the labels at the leaf node of the decision tree. However, there is an additional level of smoothing in MFs that is not present in [14]. Specifically, the prediction from a Mondrian tree, described in (3), is a weighted mixture of predictions from nodes along the path from the root to the leaf. Moreover, the weights account for the distance between the test point from the training data, thereby ensuring that the predictions shrink to the prior as we move farther away from the training data.