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($T, \mathcal{D}$, min_samples_split)

```
1: Input: Tree $T = (\mathsf{T}, \boldsymbol{\delta}, \boldsymbol{\xi}, \boldsymbol{\tau})$, new training instance $\mathcal{D} = (\boldsymbol{x}, y)$
2: ExtendMondrianBlock($T, \epsilon, \mathcal{D}$, min_samples_split)    ▷ Algorithm 4
```

Algorithm 4 ExtendMondrianBlock($T, j, \mathcal{D}$, min_samples_split)

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

Algorithm 5 Predict($T, \boldsymbol{x}$)

```
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 $p_{\mathsf{NotSeparatedYet}} = 1$
4: while True do
5:     Set $\Delta_j = \tau_j - \tau_{\mathsf{parent}(j)}$ and $\eta_j(\boldsymbol{x}) = \sum_d \big(\max(x_d - u_{jd}^x, 0) + \max(\ell_{jd}^x - x_d, 0)\big)$
6:     Set $p_j^s(\boldsymbol{x}) = 1 - \exp\big(-\Delta_j \eta_j(\boldsymbol{x})\big)$
7:     if $p_j^s(\boldsymbol{x}) > 0$ then
8:         $w_j = p_{\mathsf{NotSeparatedYet}}\ p_j^s(\boldsymbol{x})$
9:     if $j \in \mathsf{leaves}(\mathsf{T})$ then
10:         $w_j = p_{\mathsf{NotSeparatedYet}}(1 - p_j^s(\boldsymbol{x}))$
11:         return
12:     else
13:         $p_{\mathsf{NotSeparatedYet}} \leftarrow p_{\mathsf{NotSeparatedYet}}(1 - p_j^s(\boldsymbol{x}))$
14:         if $x_{\delta_j} \le \xi_j$ then $j \leftarrow \mathsf{left}(j)$ else $j \leftarrow \mathsf{right}(j)$    ▷ recurse to the child where $\boldsymbol{x}$ lies
```

B Choosing the hyperparameters

In this appendix, we give more details on how we choose the hyper parameters $\boldsymbol{\theta} = \{\mu_H, \gamma_1, \gamma_2, \sigma_y^2\}$. 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 $\boldsymbol{\mu}$ for each label individually, i.e.,

$$q(Y|\boldsymbol{\theta}, T) = \prod_{j \in \mathsf{leaves(T)}} \prod_{n \in N(j)} \mathcal{N}(y_n | \mu_H, \phi_j - \phi_{\mathsf{parent}(\epsilon)} + \sigma_y^2).$$

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

$$\begin{aligned} \phi_j - \phi_{\mathsf{parent}(\epsilon)} &= \gamma_1 \sigma(\gamma_2 \tau_j) - \gamma_1 \sigma(\gamma_2 0) \\ &= \gamma_1 (\sigma(\infty) - \sigma(0)) \\ &= \frac{\gamma_1}{2}. \end{aligned}$$

If the noise variance is known, σ_y^2 can be set to the appropriate value. In our case, the noise variance is unknown; hence, we parametrize σ_y^2 as γ_1/K and set $K = \mathsf{min}(2000, 2N)$ to ensure that the noise variance σ_y^2 is a non-zero fraction of the total variance $\gamma_1/2 + \gamma_1/K$.

We maximize $q(Y|\boldsymbol{\theta}, T)$ over μ_H, γ_1, and K, leading to

$$\begin{aligned} \mu_H &= \frac{1}{N} \sum_n y_n, \\ \gamma_1 (\frac{1}{2} + \frac{1}{K}) &= \frac{1}{N} \sum_n (y_n - \mu_H)^2. \end{aligned}$$

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 $q(Y|\boldsymbol{\theta}, T)$ does not depend on γ_2, and so we cannot choose γ_2 by optimizing it. We know, however, that τ 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) τ increases with tree depth and the tree depth is $\mathcal{O}(\log_2 N)$ assuming balanced trees and (ii) τ is inversely proportional to D. In Appendix C, we describe a fast approximation which does not involve estimation of γ_1, γ_2.

C Fast approximation to message passing and hyperparameter estimation

In Section 5.3, we suggested a fast $\mathcal{O}(\log n)$ approximation to exact message passing which costs $\mathcal{O}(n)$. 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 σ_y^2 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.