Making Tree Ensembles Interpretable: A Bayesian Model Selection Approach – Supplementary Material –

A EM Algorithm

For fixed K, we can estimate the model parameter Π using the maximum likelihood estimation:

$$\max_{\Pi} \log p(\mathcal{D}|\Pi, K). \tag{12}$$

The optimization problem (12) is solved by the EM algorithm. The lower bound of (12) is derived as

$$\sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{E}_{q(U)}[u_k^{(n)}] \log p(y^{(n)}|k,\phi)p(\boldsymbol{s}^{(n)}|k,\eta)p(k|\alpha) + H(q(U)), \tag{13}$$

where $q(U)$ is the distribution of U, and $H(q(U))$ is an entropy of $q(U)$. The EM algorithm is then formulated as an alternating maximization with respect to q (E-step) and the parameter Π (M-step).

E-Step In E-Step, we fix the parameter Π and maximize the lower bound (13) with respect to the distribution $q(U)$, which yields

$$q(u_k^{(n)} = 1) \propto p(y^{(n)}|k,\phi)p(\boldsymbol{s}^{(n)}|k,\eta)p(k|\alpha). \tag{14}$$

M-Step In M-Step, we fix the value of $q(u_k^{(n)}) = \beta_k^{(n)}$, and maximize the lower bound (13) with respect to the parameter Π. We then have, for η and α,

$$\eta_{k\ell} = \frac{\sum_{n=1}^{N}\beta_k^{(n)} s_\ell^{(n)}}{\sum_{n=1}^{N}\beta_k^{(n)}}, \qquad \alpha_k = \frac{1}{N}\sum_{n=1}^{N}\beta_k^{(n)}.$$

The parameter ϕ is also updated as

$$\mu_k = \frac{\sum_{n=1}^{N}\beta_k^{(n)} z^{(n)}}{\sum_{n=1}^{N}\beta_k^{(n)}}, \qquad \lambda_k = \frac{\sum_{n=1}^{N}\beta_k^{(n)}}{\sum_{n=1}^{N}\beta_k^{(n)}(z^{(n)}-\mu_k)^2},$$
$$\gamma_{kc} = \frac{\sum_{n=1}^{N}\beta_k^{(n)} z_c^{(n)}}{\sum_{n=1}^{N}\beta_k^{(n)}}.$$

B FAB Lower Bound

Here, we derive the lower bound of the marginal log-likelihood.

Theorem 1 *The marginal log-likelihood* $\log p(\mathcal{D}|K)$ *is lower bounded by (7) except the* $O(1)$ *term.*

The proof of this theorem follows the next three lemmas.

Lemma 1 *Let* $U = \{\boldsymbol{u}^{(n)}\}_{n=1}^{N}$ *and the complete data likelihood be* $p(\mathcal{D}, U|\Pi, K) = \prod_n p(y^{(n)}, \boldsymbol{s}^{(n)}, \boldsymbol{u}^{(n)}|\Pi, K)$. *The marginal log-likelihood* $\log p(\mathcal{D}|K)$ *is lower bounded by*

$$\mathbb{E}_{q(U)}\left[\log p(\mathcal{D}, U|\hat{\Pi}, K) - \frac{1}{2}\log\det F_{\hat{\Pi}}\right] + H(q(U)) - \frac{\dim\hat{\Pi}}{2}\log N + O(1), \tag{15}$$

where $q(U)$ *is the distribution of* U, $\hat{\Pi} = \operatorname{argmax}_{\Pi}\log p(\mathcal{D}, U|\Pi, K)$, $F_{\hat{\Pi}}$ *is the Hessian of* $-\log p(\mathcal{D}, U|\Pi, K)/N$ *at* $\Pi = \hat{\Pi}$, *and* $O(1)$ *is the term independent of* N.

(proof) Let $p(\mathcal{D}, U|K) = \int p(\mathcal{D}, U|\Pi, K)p(\Pi)d\Pi$ and $q^*(U) = p(U|\mathcal{D}, K)$. From the definition of $\log p(\mathcal{D}|K)$, the next equation holds:

$$\log p(\mathcal{D}|K) = \mathbb{E}_{q(U)}\left[\log p(\mathcal{D}, U|K)\right] + H(q(U)) + \mathrm{KL}[q(U)||q^*(U)], \tag{16}$$

where $\mathrm{KL}[q(U)||q^*(U)]$ is a KL-divergence defined as

$$\mathrm{KL}[q(U)||q^*(U)] = \mathbb{E}_{q(U)}\left[\log\frac{q(U)}{q^*(U)}\right]. \tag{17}$$

We note that the equation (16) can be easily verified from the next relationship

$$\begin{aligned}&\mathrm{KL}[q(U)||q^*(U)]\\ &= \mathbb{E}_{q(U)}[\log q(U)] - \mathbb{E}_{q(U)}[\log \underbrace{q^*(U)}_{=p(U|\mathcal{D},K)=\frac{p(\mathcal{D},U|K)}{p(\mathcal{D}|K)}}]\\ &= -H(q(U)) - \mathbb{E}_{q(U)}[\log p(\mathcal{D}, U|K)] + \log p(\mathcal{D}|K).\end{aligned} \tag{18}$$

Because the KL-divergence is non-negative, we have the lower bound of the marginal log-likelihood as

$$\log p(\mathcal{D}|K) \geq \mathbb{E}_{q(U)}\left[\log p(\mathcal{D}, U|K)\right] + H(q(U)). \tag{19}$$

We now apply Laplace's method to $\log p(\mathcal{D}, U|K)$, and derive the next equation [9]:

$$\log p(\mathcal{D}, U|K) = \log p(\mathcal{D}, U|\hat{\Pi}, K) - \frac{1}{2}\log\det F_{\hat{\Pi}} - \frac{\dim\hat{\Pi}}{2}\log N + O(1). \tag{20}$$

By substituting this result to (19), we derive the lower bound (15). □

Lemma 2 *The next inequality holds for any* Π*:*

$$\begin{aligned}
&\mathbb{E}_{q(U)}\left[\log p(\mathcal{D}, U|\hat{\Pi}, K)\right] \\
&\geq \mathbb{E}_{q(U)}\left[\log p(\mathcal{D}, U|\Pi, K)\right] \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K}\mathbb{E}_{q(U)}[u_k^{(n)}]\log p(y^{(n)}|k,\phi)p(\boldsymbol{s}^{(n)}|k,\eta)p(k|\alpha),
\end{aligned} \tag{21}$$

where $\hat{\Pi} = \mathrm{argmax}_{\Pi}\log p(\mathcal{D}, U|\Pi, K)$.

(proof) It directly follows from the definition of $\hat{\Pi}$. □

Lemma 3 *The next inequality holds:*

$$\begin{aligned}
&-\mathbb{E}_{q^(U)}\left[\frac{1}{2}\log\det F_{\hat{\Pi}}\right] - \frac{\dim\hat{\Pi}}{2}\log N \\
&\geq -\omega\sum_{k=1}^{K}\log\left(\sum_{n=1}^{N}\mathbb{E}_{q^(U)}[u_k^{(n)}] + 1\right) + O(1),
\end{aligned} \tag{22}$$

where $\omega = (\dim\phi/K + L + 1)/2$.

(proof) By expanding $\log\det F_{\Pi}$, we obtain

$$\begin{aligned}
&\log\det F_{\Pi} \\
&= \sum_{k=1}^{K}\underbrace{\log\det\left(-\frac{\partial^2}{\partial^2\phi}\frac{1}{N}\sum_{n=1}^{N}u_k^{(n)}\log p(y^{(n)}|k,\phi)\right)}_{(A)} \\
&+ \sum_{k=1}^{K}\underbrace{\log\det\left(-\frac{\partial^2}{\partial^2\eta}\frac{1}{N}\sum_{n=1}^{N}u_k^{(n)}\log p(\boldsymbol{s}^{(n)}|k,\eta)\right)}_{(B)} \\
&+ \sum_{k=1}^{K}\underbrace{\log\det\left(-\frac{\partial^2}{\partial^2\alpha}\frac{1}{N}\sum_{n=1}^{N}u_k^{(n)}\log p(k|\alpha)\right)}_{(C)}.
\end{aligned} \tag{23}$$

Here, we have

$$\begin{aligned}
(A) &= \frac{\dim\phi}{K}\log\left(\frac{1}{N}\sum_{n=1}^{N}u_k^{(n)}\right) + \log\det H_k \\
&= \underbrace{\frac{\dim\phi}{K}\log\left(\sum_{n=1}^{N}u_k^{(n)}\right) - \frac{\dim\phi}{K}\log N}_{O(\log N)} + \underbrace{\log\det H_k}_{O(1)},
\end{aligned} \tag{24}$$

$$\begin{aligned}
(B) &= \sum_{\ell=1}^{L}\log\left(\frac{1}{N}\sum_{n=1}^{N}u_k^{(n)}\left(\frac{1}{\eta_{k\ell}^2} + \frac{1}{(1-\eta_{k\ell})^2}\right)\right) \\
&= \underbrace{L\log\left(\sum_{n=1}^{N}u_k^{(n)}\right) - L\log N}_{O(\log N)} \\
&\quad + \underbrace{\sum_{\ell=1}^{L}\log\left(\frac{\sum_{n=1}^{N}u_k^{(n)}\left(\frac{1}{\eta_{k\ell}^2}\frac{1}{(1-\eta_{k\ell})^2}\right)}{\sum_{n=1}^{N}u_k^{(n)}}\right)}_{O(1)},
\end{aligned} \tag{25}$$

and

$$\begin{aligned}
(C) &= \log\left(\frac{1}{N}\sum_{n=1}^{N}u_k^{(n)}\frac{1}{\alpha_k^2}\right) \\
&= \underbrace{\log\left(\sum_{n=1}^{N}u_k^{(n)}\right) - \log N}_{O(\log N)} - \underbrace{\log\alpha_k^2}_{O(1)},
\end{aligned} \tag{26}$$

where $O(1)$ denotes terms independent of N. The matrix H_k is given as, for the regression case,

$$H_k = \begin{bmatrix} \lambda_k & \frac{\sum_{n=1}^{N}u_k^{(n)}(\mu_k - y^{(n)})}{\sum_{n=1}^{N}u_k^{(n)}} \\ \frac{\sum_{n=1}^{N}u_k^{(n)}(\mu_k - y^{(n)})}{\sum_{n=1}^{N}u_k^{(n)}} & \frac{1}{2\lambda_k^2} \end{bmatrix}$$

and for the classification case,

$$H_k = \mathrm{diag}\left(\frac{1}{\gamma_{k1}^2}, \frac{1}{\gamma_{k2}^2}, \ldots, \frac{1}{\gamma_{kC}^2}\right).$$

By using these results, we can express $\log\det F_{\Pi}$ as

$$\begin{aligned}
\log\det F_{\Pi} =& 2\omega\sum_{k=1}^{K}\log\left(\sum_{n=1}^{N}u_k^{(n)}\right) \\
&- \dim\Pi\log N + O(1).
\end{aligned} \tag{27}$$

We note that the only $O(1)$ term depends on Π and the first two terms are independent of the value of Π. Hence, the next equation holds for arbitrary Π:

$$\begin{aligned}
&-\frac{1}{2}\log\det F_{\hat{\Pi}} - \frac{\dim\hat{\Pi}}{2}\log N \\
&= -\frac{1}{2}\log\det F_{\Pi} - \frac{\dim\Pi}{2}\log N + O(1) \\
&= -\omega\sum_{k=1}^{K}\log\left(\sum_{n=1}^{N}u_k^{(n)}\right) + O(1).
\end{aligned} \tag{28}$$

Hence, the lower bound of $-\mathbb{E}_{q^*(U)}\left[\frac{1}{2}\log\det F_{\hat{\Pi}}\right] -$

$\frac{\dim\hat{\Pi}}{2}\log N$ can be derived as

$$
\begin{aligned}
&-\mathbb{E}_{q(U)}\left[\frac{1}{2}\log\det F_{\hat{\Pi}}\right]-\frac{\dim\hat{\Pi}}{2}\log N\\
&=-\omega\sum_{k=1}^{K}\mathbb{E}_{q(U)}\left[\log\left(\sum_{n=1}^{N}u_k^{(n)}\right)\right]+O(1)\\
&\geq-\omega\sum_{k=1}^{K}\log\left(\sum_{n=1}^{N}\mathbb{E}_{q(U)}[u_k^{(n)}]+1\right)+O(1), \qquad (29)
\end{aligned}
$$

where we used Jensen' inequality. □

By using these lemmas, we now prove our main claim.

(proof of Theorem 1) By substituting (21) and (22) into (15) and removing the $O(1)$ term, the claim follows. □

C FAB Inference Algorithm Derivation

E-Step In E-Step, we update the distribution $q(U)$ so that the lower bound (7) to be maximized. Let $\beta_k^{(n)}=\mathbb{E}_{q(U)}[u_k^{(n)}]=q(u_k^{(n)})$. The maximization problem can then be expressed as

$$
\begin{aligned}
\max_{\beta}\sum_{n=1}^{N}\sum_{k=1}^{K}\beta_k^{(n)}\log f_k^{(n)}-\omega\sum_{k=1}^{K}\log\left(\sum_{n=1}^{N}\beta_k^{(n)}+1\right)\\
-\sum_{n=1}^{N}\sum_{k=1}^{K}\beta_k^{(n)}\log\beta_k^{(n)},\\
\text{s.t.}\ \sum_{k=1}^{K}\beta_k^{(n)}=1, \qquad (30)
\end{aligned}
$$

where $f_k^{(n)}=p(y^{(n)}|k,\phi)p(\boldsymbol{s}^{(n)}|k,\eta)p(k|\alpha)$. We note that the problem (30) is smooth concave maximization, and a unique global optimum exists. Such an optimum can be found by iterative maximization of the lower bound of (30). Recall that $\log\left(\sum_{n=1}^{N}\beta_k^{(n)}+1\right)\leq\log\left(\sum_{n=1}^{N}\psi_k^{(n)}+1\right)+\sum_{n=1}^{N}\frac{\beta_k^{(n)}-\psi_k^{(n)}}{\sum_{n'=1}^{N}\psi_k^{(n')}+1}$ holds for any $\psi_k^{(n)}$ from the concavity. Once the value of $\psi_k^{(n)}$ is fixed, we can maximize the lower bound of (8) by solving

$$
\begin{aligned}
\max_{\beta}\sum_{n=1}^{N}\sum_{k=1}^{K}\beta_k^{(n)}\log f_k^{(n)}-\omega\sum_{k=1}^{K}\sum_{n=1}^{N}\frac{\beta_k^{(n)}}{\sum_{n'=1}^{N}\psi_k^{(n')}+1}\\
-\sum_{n=1}^{N}\sum_{k=1}^{K}\beta_k^{(n)}\log\beta_k^{(n)},\\
\text{s.t.}\ \sum_{k=1}^{K}\beta_k^{(n)}=1, \qquad (31)
\end{aligned}
$$

which results in

$$
\beta_k^{(n)}\propto f_k^{(n)}\exp\left(-\frac{\omega}{\sum_{n=1}^{N}\psi_k^{(n)}+1}\right), \qquad (32)
$$

Using this result, we can solve the original maximization problem (30) by iteratively setting $\psi_k^{(n)}\leftarrow\beta_k^{(n)}$ and updating β by (32). Because the lower bound (31) increases in every iteration, the iteration procedure converges to the global optimum.

M-Step In M-Step, we update the parameter Π so that the lower bound (7) to be maximized. Let $\beta_k^{(n)}=\mathbb{E}_{q(U)}[u_k^{(n)}]=q(u_k^{(n)})$. The maximization problem (7) can then be decomposed into subproblems:

$$
\begin{aligned}
&\max_{\phi}\sum_{n=1}^{N}\beta_k^{(n)}\log p(y^{(n)}|k,\phi), \qquad (33)\\
&\max_{\eta}\sum_{n=1}^{N}\beta_k^{(n)}\log\eta_{k\ell}^{s_\ell^{(n)}}(1-\eta_{k\ell})^{1-s_\ell^{(n)}}, \qquad (34)\\
&\max_{\alpha}\sum_{k=1}^{K}\left(\sum_{n=1}^{N}\beta_k^{(n)}\right)\log\alpha_k,\ \text{s.t.}\ \sum_{k=1}^{K}\alpha_k=1,. \qquad (35)
\end{aligned}
$$

These problems can be solved analytically. The solution to the problem (33) are derived as

$$
\text{(regression):}\begin{cases}\mu_k=\frac{\sum_{n=1}^{N}\beta_k^{(n)}y^{(n)}}{\sum_{n=1}^{N}\beta_k^{(n)}},\\ \lambda_k=\frac{\sum_{n=1}^{N}\beta_k^{(n)}}{\sum_{n=1}^{N}\beta_k^{(n)}(y^{(n)}-\mu_k)^2},\end{cases} \qquad (36)
$$

$$
\text{(classification):}\quad \gamma_{kc}=\frac{\sum_{n=1}^{N}\beta_k^{(n)}y_c^{(n)}}{\sum_{n=1}^{N}\beta_k^{(n)}}. \qquad (37)
$$

The solutions to the problem (34) and (35) are derived as

$$
\eta_{k\ell}=\frac{\sum_{n=1}^{N}\beta_k^{(n)}s_\ell^{(n)}}{\sum_{n=1}^{N}\beta_k^{(n)}}, \qquad \alpha_k=\frac{1}{N}\sum_{n=1}^{N}\beta_k^{(n)}. \qquad (38)
$$

D Scalable FAB Inference with Statement Sampling

The time complexity of the proposed FAB inference is $O(K_{\max}LN+\zeta K_{\max}N)$, where ζ is the number of iterations in E-Step. This time complexity sometimes gets prohibitive when L is large, i.e., when the tree ensemble is huge.

To scale up the proposed FAB inference to large L, we propose a simple heuristic, a sampling FAB inference, based on the random sampling of the statements: we do not use all the statements in the tree ensemble but the randomly sampled subset. If the number of statements is reduced and L gets very small, FAB inference gets highly efficient and scalable.

The sampling idea follows from the intuition that most of the statements in the tree ensemble are redundant when we make a simplified expression. This is because we usually have similar statements such as $x_1 < 0.500$ and $x_1 < 0.501$ in the tree ensemble. When this subtle change on the threshold is ignorable, removing one of these two statements will have almost no impact to the resulting simplified expression. Random sampling of the statements can remove these redundant statements effectively.

We now turn to the proposed heuristic, a sampling FAB inference. Suppose that we randomly sampled $\tilde{L}$ statements out of L statements in the tree ensemble. Then, the binary feature defined on the sampled statements is given by $\tilde{\boldsymbol{s}} \in \{0,1\}^{\tilde{L}}$ where $\tilde{s}_{\ell'} = \mathbb{I}(x_{\tilde{d}_{\ell'}} > \tilde{b}_{\ell'})$, and $\tilde{d}_{\ell'}$ and $\tilde{b}_{\ell'}$ denote the feature index and the threshold of each of the sampled statement, respectively. The generative model of $\tilde{\boldsymbol{s}}$ is given by

$$p(\tilde{\boldsymbol{s}}|g) := \prod_{\ell'=1}^{\tilde{L}} {\eta'}_{g\ell'}^{\tilde{s}_{\ell}} (1 - {\eta'}_{g\ell'})^{1-\tilde{s}_{\ell'}}, \tag{39}$$

where $\eta'_{g\ell'} = p(\tilde{s}_{\ell'} = 1) = p(x_{\tilde{d}_{\ell'}} > \tilde{b}_{\ell'})$. Here, we assume that the sampled version of the generative model (39) becomes a good approximation of the original generative model (4) when we virtually increase the number of statements from $\tilde{L}$ to L, i.e.,

$$p(\tilde{\boldsymbol{s}}|g)^{L/\tilde{L}} \approx p(\boldsymbol{s}|g). \tag{40}$$

With the approximation (40), the approximate FAB lower bound becomes

$$\sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{E}_{q(U)}[u_k^{(n)}] \log p(y^{(n)}|k,\phi) p(\tilde{\boldsymbol{s}}^{(n)}|k,\eta)^{L/\tilde{L}} p(k|\alpha) - \omega \sum_{k=1}^{K} \log\left(\sum_{n=1}^{N} \mathbb{E}_{q(U)}[u_k^{(n)}] + 1\right) + H(q(U)), \tag{41}$$

where $\omega = (\dim\phi/K + L + 1)/2$. The sampling FAB inference algorithm can be derived by a slight modification of the original E-step (32): we replace $f_k^{(n)} = p(y^{(n)}|k,\phi)p(\boldsymbol{s}^{(n)}|k,\eta)p(k|\alpha)$ with $\tilde{f}_k^{(n)} = p(y^{(n)}|k,\phi)p(\tilde{\boldsymbol{s}}^{(n)}|k,\eta)^{L/\tilde{L}}p(k|\alpha)$. The M-step remains the same as the original FAB inference except that η is replaced with the sampled version η' in (39). Because the number of statements appearing in the sampling FAB inference is $\tilde{L}$ rather than L, its time complexity is $O(K_{\max}\tilde{L}N + \zeta K_{\max}N)$, which can be significantly smaller than the original FAB inference. For example, when the number of original statements is $L = 10,000$, the sampling FAB inference can be nearly 100 times faster when we set $\tilde{L} = 100$. We show in Appendix E that the speed up can be achieved with almost no effects on the prediction performance of the resulting simplified expression.

Table 4: [Datasets] Four real world datasets are obtained from the UCI Machine Learning Repository [23]. The task of the first five data are binary classification, while the task of the last Energy data is regression. D is the data dimensionality, N_{all} is the number of data points in the original dataset, and N_{train} and N_{test} denote the number of data points randomly sampled for training and testing the models in the experiment.

	D	N_{all}	$N_{\text{train}}, N_{\text{test}}$
Synthetic1	2	-	1,000
Synthetic2	2	-	1,000
Spambase	57	4,601	1,000
MiniBooNE	50	130,065	5,000
Higgs	28	11,000,000	5,000
Energy	8	768	384

E Experiments

E.1 Implementations

In all experiments, we used `randomForest` package in R to train tree ensembles with 100 trees. The tree ensemble simplification methods are then applied to extract rules from the learned tree ensembles. The proposed method is implemented in Python. For the proposed method, we used the statement sampling heuristic with the sampling size 100. We set $K_{\max} = 10$ and ran FAB inference for 20 different random initial parameters, and derived learned 20 parameters $\{\Pi_m\}_{m=1}^{20}$. We then adopted the parameter with the smallest training error, i.e., $\tilde{\Pi} = \text{argmin}_{\Pi_m} \text{Error}(\mathcal{D}, \Pi_m)$ with $\text{Error}(\mathcal{D},\Pi) := \sum_{n=1}^{N}\left(y^{(n)} - \hat{y}^{(n)}\right)^2$ for regression, and $\text{Error}(\mathcal{D},\Pi) := \sum_{n=1}^{N}\mathbb{I}(y^{(n)} \neq \hat{y}^{(n)})$ for classification. The BATrees is implemented also in Python. The depth of BATrees is chosen from $\{2,3,4,6,8,10\}$ using 5-fold cross validation. For inTrees and Node Harvest, we used their R implementations with their default settings. For DTree2, we used `DecisionTreeRegressor` and `DecisionTreeClassifier` of scikit-learn in Python while fixing their depth to two. All experiments were conducted on 64-bit CentOS 6.7 with an Intel Xeon E5-2670 2.6GHz CPU and 512GB RAM.

E.2 Datasets

In the experiments, we used six datasets summarized in Table 4. Because BATrees is not scalable to large datasets, we limited the number of dataset size to 5,000 for MiniBooNE and Higgs. Later in Appendix E.3.3, we show that the proposed method scales to the large dataset.

The first synthetic data (Synthetic1) is generated from

the following procedure:

$$\begin{aligned} \boldsymbol{x} &= (x_1, x_2) \sim \text{Uniform}[0,1], \\ y^* &= \text{XOR}(x_1 > 0.5, x_2 > 0.5), \\ y &= \text{XOR}(y^*, \theta), \end{aligned}$$

where $\theta \in \{0,1\}$ with $p(\theta = 1) = 0.1$, which corresponds to the 10% label noise. Similarly, the second synthetic data (Synthetic2) is generated by replacing the second step with $y^* = \mathbb{I}(x_2 > r(x_1))$ where $r(x_1) = 0.25 + 0.5/(1 + \exp(-20 * (x_1 - 0.5))) + 0.05\cos(2\pi x_1)$. Synthetic1 has a box-shaped class boundary (upper figure of Figure 1(a)), and can be expressed by the region-based model using four regions. On the other hand, Synthetic2 has a more complicated class boundary (bottom figure of Figure 1(a)). Hence, it is more difficult to simplify the tree ensemble and derive a good approximate model.

E.3 Results

E.3.1 FAB Inference vs. EM Algorithm

We compared the runtimes of FAB inference and the EM algorithm. For the EM algorithm, we ran the algorithm by varying the value of K from 1 to 10, and reported the total runtime. For both methods, we used the sampling heuristic described in Appendix D.

Table 5 summarizes that FAB inference was from three to nearly ten times faster than the EM algorithm. FAB inference attained smaller runtimes by avoiding searching over several possible number of rules K and deciding the number automatically. Figure 5 shows the comparison of the test errors of the found rules: they show that FAB inference could find an appropriate number of rules K with small prediction errors. These results suggest the superiority of FAB inference over the EM algorithm as it could find an appropriate number of rules by avoiding redundant computations for searching the number of rules K.

E.3.2 Comparison with Baseline Methods

Figure 6 shows the trade-off between the number of found rules K and the test errors of each method. The figures show that DTree2 tended to attain the smallest number of rules (i.e., four), and the proposed method was second (from four to ten). The number of rules found by inTrees and Node Harvest tended to be around 30 to 100, while the number of rules found by BATrees sometimes exceeded 100. The figures also show inTrees and BATrees tended to attain the smallest errors while DTree2 tended to be the worst on most of the datasets. The proposed method attained a good trade-off between the number of rules and the test errors: it tended to score smaller errors than DTree2 while using only a few rules, which is significantly smaller than the other baseline methods. These results suggest that the proposed method is favorable for interpretation as it generates only a few rules with small test errors. Smaller number of rules helps users to check the found rules easily. The small test errors support that the found rules are reliable, i.e., the rules well explain the original tree ensemble.

E.3.3 Naive FAB Inference vs. Sampling FAB Inference

Table 6 shows how the samling FAB inference help to improve the computational scalability. For the sampling FAB inference, we fixed the number of statement sampling $\tilde{L}$ to be 100. For the naive FAB inference, we used all the statements taken from the original tree ensemble. The table shows that the sampling FAB inference ran from 2 to 20 times faster than the naive FAB inference. In particular, we can find that the sampling FAB inference is computationally efficient when the data size is large such as in MiniBooNE and Higgs. It also shows that, compared to the naive FAB inference, there is almost no changes on the number of found rules and the test errors even if we use the sampling FAB inference. This result suggests that we can use the sampling FAB inference for efficient computation with almost no sacrifices on the resulting performances.

To see the success of the sampling FAB inference in detail, we increased the training size of MiniBooNE to $N_{\text{train}} = 100,000$, and conducted the experiment over ten random data realizations. Because BATrees was too slow, we omitted it from the experiment. Figure 7 and Table 7 show that the sampling FAB inference was still effective under the large sample size. Figure 7 shows that the sampling FAB inference attained the good trade-off between the number of rules and the test errors as we have observed in Appendix E.3.2. Table 7 reports that the sampling FAB inference was computationally efficient even under the large sample size. Indeed, compared to the naive FAB inference in Table 6, the sampling FAB inference was more than two times faster even though the training size was increased by twenty times. We also note that, compared to inTrees and Node Harvest, the sampling FAB inference was comparably fast: it was slightly slower than inTrees and a few times faster than Node Harvest.

E.3.4 Example of Found Rules

Synthetic Data: Figure 8 and 9 show the simplified rules of the learned tree ensemble on Synthetic1 and Synthetic2, respectively. The results on Node Harvest can be found in Figure 10 and 11. The proposed method well simplified the boundary using only a few

Table 5: Average runtimes in seconds for one restart: the EM algorithm ran over $K = 1, 2, \ldots, 10$, and its total time is reported.

	Synthetic1	Synthetic2	Spambase	MiniBooNE	Higgs	Energy
FAB	3.01 ± 1.73	3.30 ± 1.43	5.32 ± 1.98	93.0 ± 49.2	54.3 ± 13.7	0.14 ± 0.12
EM	19.1 ± 5.99	18.3 ± 4.38	35.7 ± 3.47	282 ± 60.1	227 ± 63.6	1.79 ± 0.64

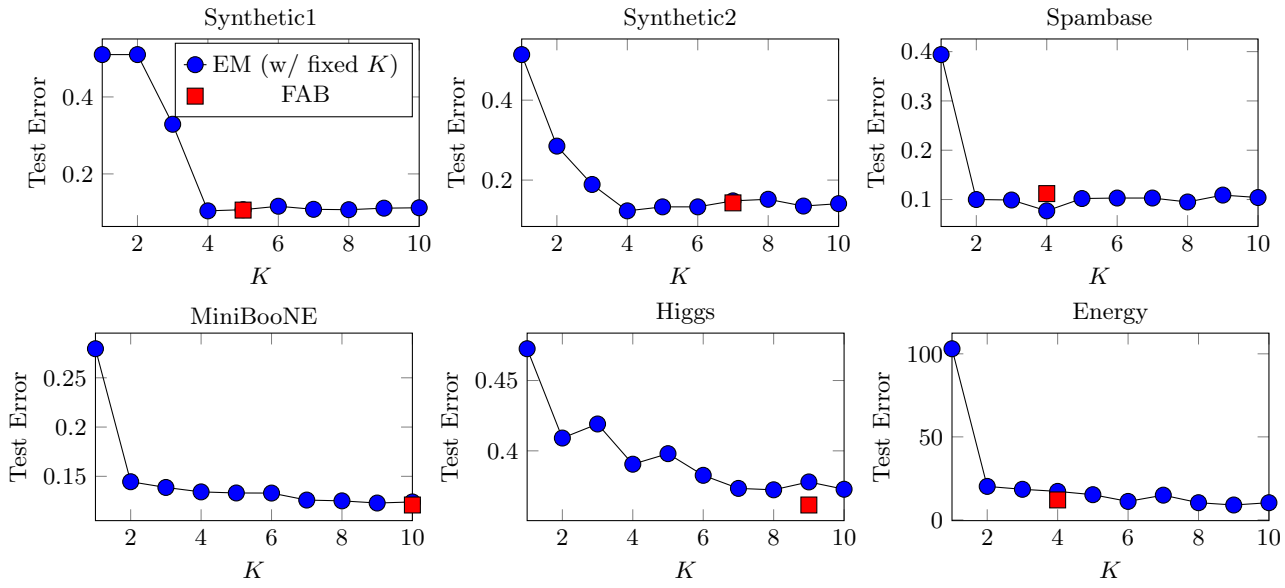


Figure 5: Test Errors of FAB inference and the EM algorithm.

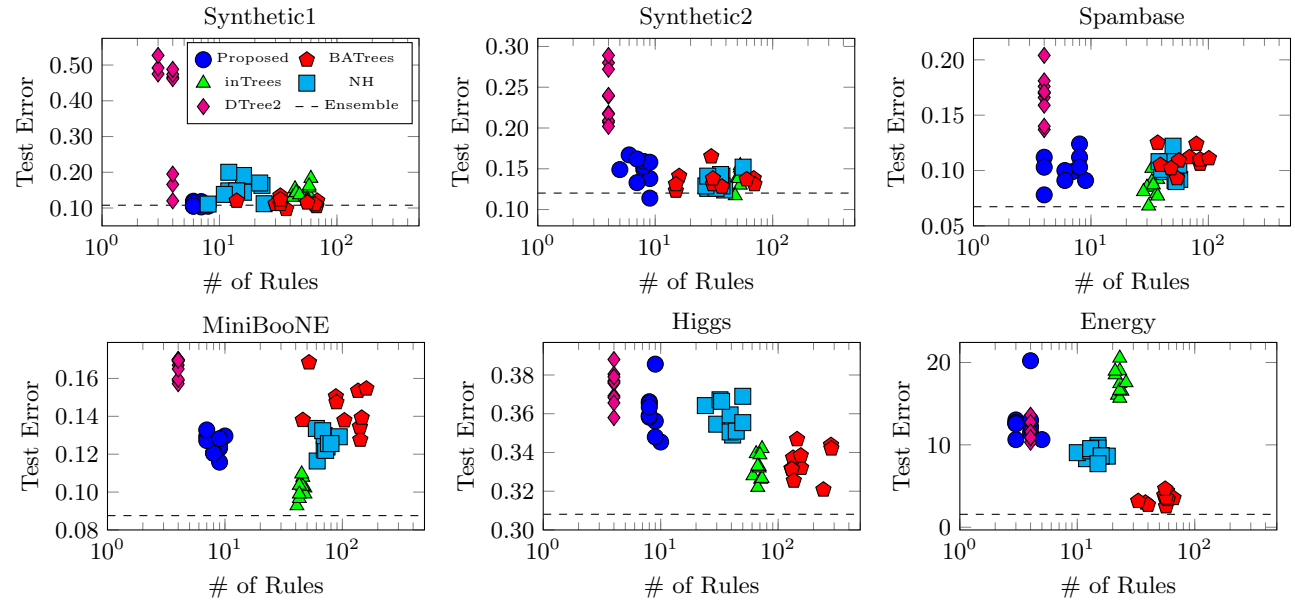


Figure 6: Comparison of the simplification methods: # of rules vs. test error. *Ensemble* denotes the average error of the tree ensemble over the ten random data realizations.

rules. On the other hand, BATrees and inTrees required more rules. It is important to note that inTrees and Node Harvest found rules that highly overlap each other. The rule overlapping is not favorable for interpretation: if the prediction of the overlapping rules are distinct, we cannot decide which rule to trust. Although there are some overlaps between the rules in the proposed method, this is not that critical as ob-

Table 6: Comparison of the naive FAB inference and the sampling FAB inference: The average runtime in seconds for one restart, the average number of found rules, and the average test error are reported.

	Synthetic1 ($N_{\text{train}} = 1,000$)			Synthetic2 ($N_{\text{train}} = 1,000$)		
	Time	# of Rules	Test Error	Time	# of Rules	Test Error
Naive FAB	11.7 ± 4.69	5.10 ± 1.76	0.11 ± 0.01	17.8 ± 4.75	6.40 ± 1.02	0.15 ± 0.02
Sampl. FAB	3.63 ± 0.85	6.60 ± 0.80	0.11 ± 0.01	4.09 ± 0.50	7.60 ± 1.28	0.15 ± 0.02
	Spambase ($N_{\text{train}} = 1,000$)			MiniBooNE ($N_{\text{train}} = 5,000$)		
	Time	# of Rules	Test Error	Time	# of Rules	Test Error
Naive FAB	19.5 ± 4.30	5.40 ± 1.56	0.09 ± 0.01	1962 ± 509	9.10 ± 0.83	0.12 ± 0.00
Sampl. FAB	7.77 ± 2.02	6.40 ± 1.80	0.10 ± 0.01	81.5 ± 7.74	9.30 ± 0.90	0.12 ± 0.01
	Higgs ($N_{\text{train}} = 5,000$)			Energy ($N_{\text{train}} = 384$)		
	Time	# of Rules	Test Error	Time	# of Rules	Test Error
Naive FAB	1503 ± 519	9.10 ± 0.83	0.36 ± 0.01	0.25 ± 0.02	3.60 ± 0.66	12.9 ± 2.58
Sampl. FAB	56.5 ± 8.04	9.40 ± 0.66	0.37 ± 0.01	0.11 ± 0.01	3.60 ± 0.66	12.9 ± 2.58

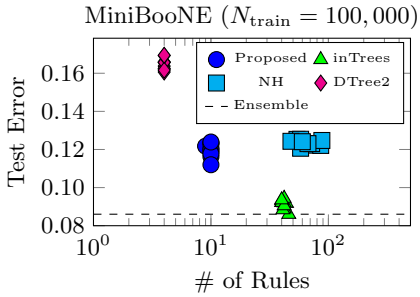


Figure 7: Comparison of the simplification methods on the large MiniBooNE dataset ($N_{\text{train}} = 100,000$): # of rules vs. test error. *Ensemble* denotes the average error of the tree ensemble over the ten random data realizations.

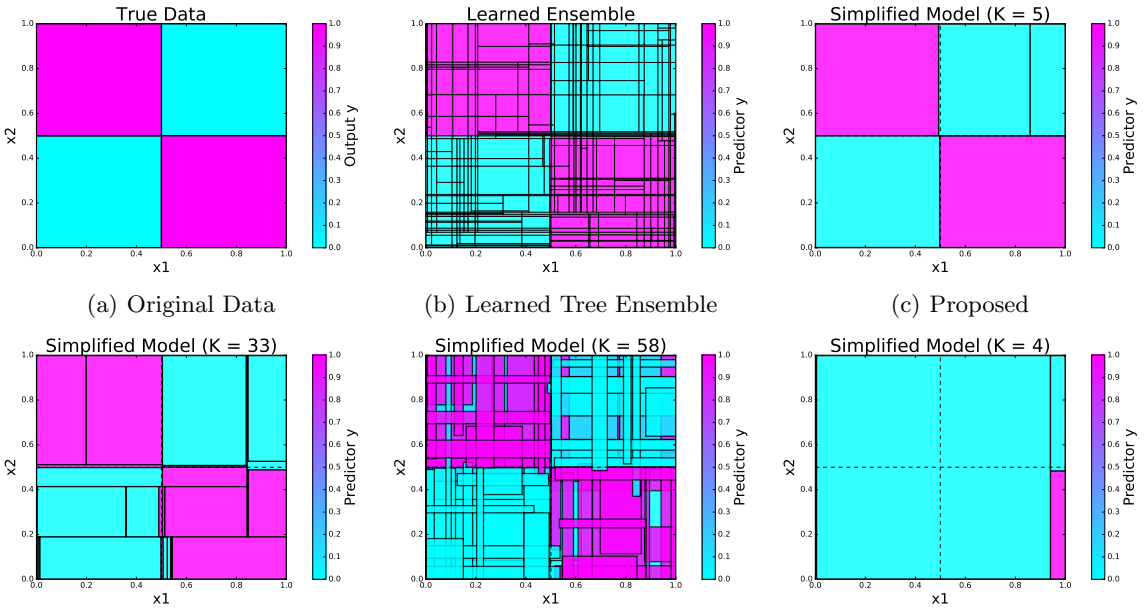
Table 7: Performance of the sampling FAB inference, inTrees, and Node Harvest: The average runtime in seconds (per one restart for the sampling FAB inference), the average number of found rules, and the average test error are reported.

	MiniBooNE ($N_{\text{train}} = 100,000$)		
	Time	# of Rules	Test Error
Sampl. FAB	790 ± 71.2	9.90 ± 0.30	0.12 ± 0.00
inTrees	588 ± 40.0	41.9 ± 1.81	0.09 ± 0.00
Node Harvest	2902 ± 117	66.5 ± 12.7	0.12 ± 0.00

served for inTrees and Node Harvest because the overlapped regions are limited. Table 8 shows the average number of overlapped rules where the ideal value is one. While the proposed method attained values close to one, inTrees and Node Harvest scored far larger values.

Energy Data: Energy efficiency data is a simulation data sampled from 12 different building shapes. The dataset comprises eight numeric features which are Relative Compactness, Surface Area, Wall Area, Roof Area, Overall Height, Orientation, Glazing Area, and Glazing Area Distribution. The task is regression, which aims to predict the heating load of the building from these eight features.

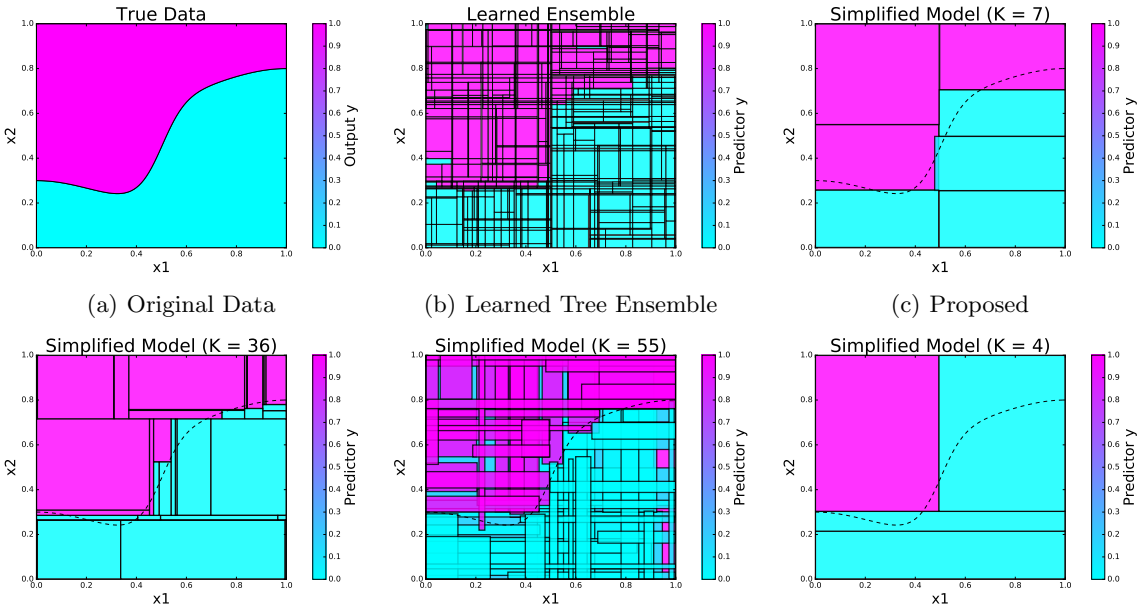
In Table 9, the four rules found by the proposed method are characterized by the two features Overall Height and Wall Area. The four rules are expressed as a direct product of the two statements; (i) Overall Height $\in \{\text{low}, \text{high}\}$, and (ii) Wall Area $\in \{\text{small}, \text{large}\}$. The resulting rules are intuitive such that the load is small when the building is small, while the load is large when the building is huge. Hence, from these simplified rules, we can infer that the tree ensemble is learned in accordance with our intuition about the data. In contrast to the simple rules found by the proposed method, the baseline methods found more rules: BATrees learned 66 rules, inTrees enumerated 23 rules, and Node Harvest found 10 rules, respectively. Table 9 shows four example rules found by each method.



(a) Original Data (b) Learned Tree Ensemble (c) Proposed

(d) BATrees (e) inTrees (f) DTree2

Figure 8: Synthetic1: Original data, leaned tree ensemble, and simplified rules.



(a) Original Data (b) Learned Tree Ensemble (c) Proposed

(d) BATrees (e) inTrees (f) DTree2

Figure 9: Synthetic2: Original data, leaned tree ensemble, and simplified rules.

Table 8: Average Number of Rules Covering Each Test Point: BATrees is omitted because its value is always one.

	Synthetic1	Synthetic2	Spambase	MiniBooNE	Higgs	Energy
Proposed	1.03 ± 0.04	1.05 ± 0.06	2.03 ± 0.26	4.53 ± 0.86	1.56 ± 0.20	0.95 ± 0.08
inTrees	4.65 ± 0.38	3.73 ± 0.41	5.39 ± 0.34	6.25 ± 0.28	5.53 ± 0.27	3.30 ± 0.19
Node Harvest	3.53 ± 0.95	8.78 ± 2.02	12.4 ± 1.76	17.9 ± 2.24	9.42 ± 1.85	3.61 ± 0.19

Figure 10: Synthetic1: Found rules by Node Harvest.

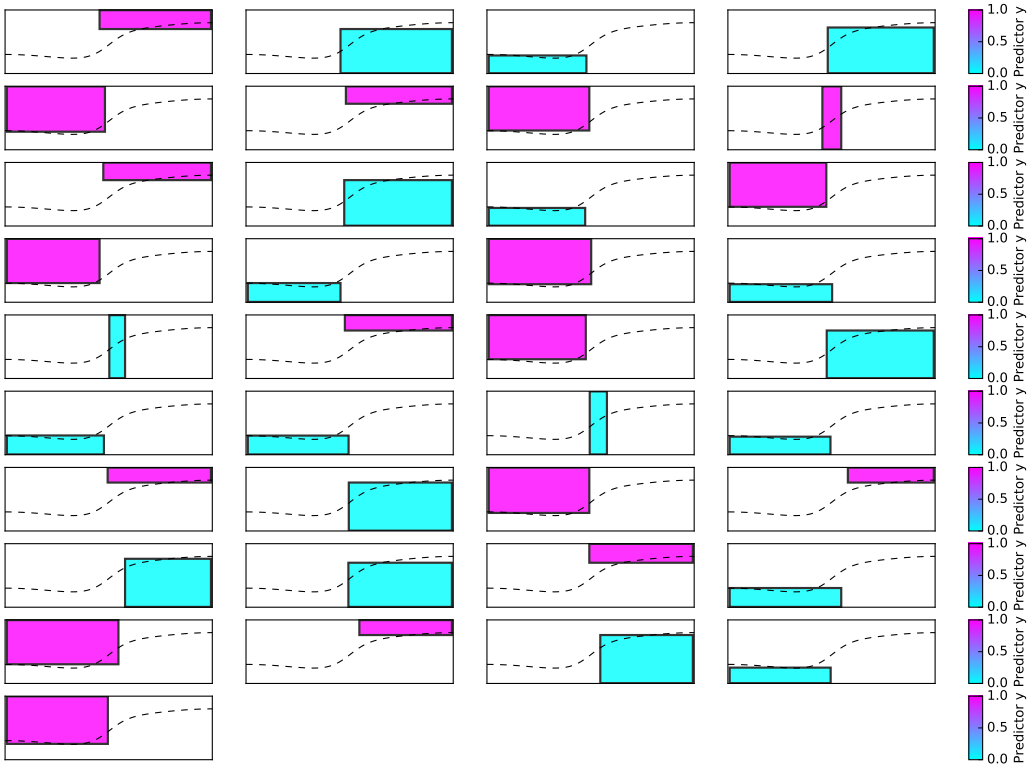


Figure 11: Synthetic2: Found rules by Node Harvest.

Table 9: Examples of extracted rules using the tree ensemble simplification methods on Energy data.

	y	Rule
Proposed	12.33	OverallHeight $<$ 5.25, WallArea $<$ 306.25
	14.39	OverallHeight $<$ 5.25, WallArea $\geq$ 318.50
	28.17	OverallHeight $\geq$ 5.25, WallArea $<$ 330.75
	37.38	OverallHeight $\geq$ 5.25, WallArea $\geq$ 343.00
BATrees	17.18	RelativeCompactness $<$ 0.84, WallArea $<$ 330.75, RoofArea $<$ 183.75, GlazingArea $<$ 0.33, GlazingAreaDistribution $<$ 0.50
	24.50	RelativeCompactness $<$ 0.84, WallArea $<$ 330.75, RoofArea $<$ 183.75, Orientation $<$ 3.50, GlazingArea $<$ 0.33, 0.50 $\leq$ GlazingAreaDistribution $<$ 3.50
	24.20	RelativeCompactness $<$ 0.84, WallArea $<$ 330.75, RoofArea $<$ 183.75, 3.50 $\leq$ Orientation $<$ 4.50, GlazingArea $<$ 0.33, 0.50 $\leq$ GlazingAreaDistribution $<$ 3.50
	23.94	RelativeCompactness $<$ 0.84, WallArea $<$ 330.75, RoofArea $<$ 183.75, Orientation $\geq$ 4.50, GlazingArea $<$ 0.33, 0.50 $\leq$ GlazingAreaDistribution $<$ 3.50
inTrees	12.58	RelativeCompactness $\geq$ 0.65, OverallHeight $<$ 5.25, GlazingArea $<$ 0.33
	33.20	RelativeCompactness $\geq$ 0.75, SurfaceArea $\geq$ 624.75, GlazingAreaDistribution $\geq$ 0.50
	33.20	RelativeCompactness $\geq$ 0.84, GlazingArea $\geq$ 0.33
	23.61	673.75 $\leq$ SurfaceArea $<$ 796.25, WallArea $\geq$ 306.25, GlazingArea $\geq$ 0.17
NH	14.53	SurfaceArea $\geq$ 674.00, GlazingArea $\geq$ 0.17
	28.17	RelativeCompactness $\geq$ 0.81, SurfaceArea $<$ 674.00
	37.38	RelativeCompactness $<$ 0.81, SurfaceArea $<$ 674.00
	11.21	SurfaceArea $\geq$ 674.00, GlazingArea $<$ 0.17
DTree2	11.21	SurfaceArea $\geq$ 673.75, GlazingArea $<$ 0.17
	14.53	SurfaceArea $\geq$ 673.75, GlazingArea $\geq$ 0.17
	28.17	SurfaceArea $<$ 624.75
	37.38	624.75 $\leq$ SurfaceArea $<$ 673.75