Optimally Pruning Decision Tree Ensembles With Feature Cost

Feng Nan FNAN@BU.EDU
Joseph Wang JOEWANG@BU.EDU
Venkatesh Saligrama SRV@BU.EDU
Boston University, 8 Saint Mary's Street, Boston, MA

Abstract

We consider the problem of learning decision rules for prediction with feature budget constraint. In particular, we are interested in pruning an ensemble of decision trees to reduce expected feature cost while maintaining high prediction accuracy for any test example. We propose a novel 0-1 integer program formulation for ensemble pruning. Our pruning formulation is general - it takes any ensemble of decision trees as input. By explicitly accounting for feature-sharing across trees together with accuracy/cost trade-off, our method is able to significantly reduce feature cost by pruning subtrees that introduce more loss in terms of feature cost than benefit in terms of prediction accuracy gain. Theoretically, we prove that a linear programming relaxation produces the exact solution of the original integer program. This allows us to use efficient convex optimization tools to obtain an optimally pruned ensemble for any given budget. Empirically, we see that our pruning algorithm significantly improves the performance of the state of the art ensemble method BudgetRF.

1. Introduction

Many modern applications of supervised machine learning face the challenge of test-time budget constraints. For example, in internet search engines (Chapelle et al.), features of the query-document pair are extracted whenever a user enters a query at the cost of some CPU time in order to rank the relevant documents. The ranking has to be done in milliseconds to be displayed to the user, making it impossible to extract computationally expensive features for all documents. Rather than simply excluding these computationally expensive features, an adaptive decision rule is needed, so that only cheap features are extracted for the majority of queries and expensive features are extracted for only a small number of difficult queries. Many approaches have been proposed by various authors to solve such test-time budget constraint problem (Gao & Koller, 2011; Xu et al., 2012; Trapeznikov & Saligrama, 2013; Wang et al., 2014b;a; Nan et al., 2014; Wang et al., 2015).

Nan et al. (Nan et al., 2015) proposed a novel random forest approach for test-time feature cost reduction. During training, an ensemble of decision trees are built based on random subsampling the training data for each decision tree. A class of *admissible* (essentially monotone and supermodular) impurity functions together with the cost of each feature are used to greedily determine the data split at each internal node of the decision trees. During prediction, a test example is run through each of the trees in the ensemble and the majority label is assigned to the test example. Such a simple strategy is shown to yield a worst-case cost at most $O(\log(n))$ times the optimal cost for each decision tree built on n training samples. Empirically, it is shown to have state-of-the-art performance in terms of prediction-cost tradeoff.

The trees in these budgeted random forests are built independently, ignoring the fact that repeated use of the same feature does not incur repeated feature acquisition cost. We exploit interdependencies among the ensemble of trees to achieve better accuracy - cost tradeoff. Theoretically, we propose a general ensemble pruning formulation that solves the accuracy-cost tradeoff exactly; empirically, we demonstrate significant improvement.

The focus of this paper is on pruning ensembles of decision trees. We assume an ensemble of decision trees are given as inputs; such an ensemble can be obtained using the algorithm proposed by Nan et al. (Nan et al., 2015) or any other decision tree ensemble method. Our main contribution is the development of an efficient algorithm for pruning an ensemble of decision trees to explicitly tradeoff prediction accuracy and feature cost.

2. Related Work

Although decision tree pruning has been studied extensively to improve generalization performance, we are not aware of any existing pruning method that takes into account the feature costs.

A popular heuristic for pruning to reduce generalization error is Cost-Complexity Pruning (CCP), introduced by Breiman et al. (Breiman et al., 1984). It defines a *cost-complexity* measure for each subtree of the decision tree as sum of two terms: the number of misclassified examples in the subtree plus the number of leaves in the subtree times a tradeoff parameter. This measure is also computed when the subtree is pruned to become a leaf. As the tradeoff parameter increases, more emphasis is given to reducing the size of the subtree compared to minizing the number of misclassified examples. The CCP algorithm iteratively selects the subtree with the lowest cost-complexity measure if it were pruned as the tradeoff parameter gradually increases. At each iteration the selected subtree is pruned and the cost-complexity measures are re-computed for the next iteration. Each pruned tree produced in this procedure is optimal with respect to size - no other subtree of the same number of leaves would have a lower misclassification rate than the one obtained by this procedure. As pointed out by Li et al. (Li et al., 2001), CCP has undesirable "jumps" in the sequence of pruned tree sizes. To alleviate this, they proposed a Dynamic-Program-based Pruning (DPP) method for binary trees. The DPP algorithm is able to obtain optimally pruned trees of all sizes, however, faces the curse of dimensionality when pruning an ensemble of decision trees and taking feature cost into account.

Generally, pruning is not considered when constructing random forests as overfitting is avoided by constructing an ensemble of trees. The ensemble approach is a strong approach to avoiding overfitting, however test-time budget constraint problems require consideration of both cost and accuracy.

Kulkarni and Sinha (Kulkarni & Sinha, 2012) provide a survey of methods to prune random forests in order to reduce ensemble size. However, these methods do not explicitly account for feature costs.

3. Background and Notations

A training sample $S = \{(\mathbf{x}^{(i)}, y^{(i)}) : i = 1, \dots, N\}$ is generated i.i.d. from an unknown distribution, where $\mathbf{x}^{(i)} \in \Re^K$ is the feature vector with a cost assigned to each of the K features and $y^{(i)}$ is the label for the ith example. In the case of multi-class classification $y \in \{1, \dots, M\}$, where M is the number of classes. Given a decision tree $\mathcal{T}$, we index the nodes as $h \in \{1, \dots, |\mathcal{T}|\}$, where node 1 represents the root node. For any $h \in \mathcal{T}$, we define the following standard terminology:

$p(h) \equiv$ set of predecessor nodes of $h \equiv$ set of nodes (excluding h) that lie on the path from the root node to h.

$\mathcal{T}_h \equiv$ subtree of $\mathcal{T}$ that is rooted at node h.

$\tilde{\mathcal{T}} \equiv$ set of leaf nodes of tree $\mathcal{T}$.

$b(h) \equiv$ set of brother (sibling) nodes of $h \equiv$ set of nodes who share the same immediate parent node as h.

$S_h \equiv$ the set of examples in S routed to or through h on $\mathcal{T}$.

$\text{Pred}_h \equiv$ predicted label at node h on $\mathcal{T}$ based on the class distribution of S_h. It is equal to the class with the most number of training examples at h.

$e_h \equiv$ number of misclassified examples in S_h based on Pred_h. It is equal to $\sum_{i \in S_h} \mathbb{1}_{[y^{(i)} \neq \text{Pred}_h]}$.

Finally, the corresponding definitions for $\mathcal{T}$ can be extended to an ensemble of T decision trees $\{\mathcal{T}_t : t = 1, \dots, T\}$ by adding an subscript t.

The process of pruning $\mathcal{T}$ at h involves *collapsing* $\mathcal{T}_h$ and making h a leaf node. We say a pruned tree $\mathcal{T}_{\text{Prune}}$, having $\tilde{\mathcal{T}}_{\text{Prune}}$ as its set of leaf nodes, is a *valid* pruned tree of $\mathcal{T}$ if (1) $\mathcal{T}_{\text{Prune}}$ is a subtree of $\mathcal{T}$ containing root node 1 and (2) for any $h \neq 1$ contained in $\mathcal{T}_{\text{Prune}}$, the sibling nodes $b(h)$ must also be contained in $\mathcal{T}_{\text{Prune}}$.

For a given tree $\mathcal{T}$, let us define the following binary variable for each node $h \in \mathcal{T}$

$$z_h = \begin{cases} 1 & \text{if node } h \text{ is a leaf in the pruned tree,} \\ 0 & \text{otherwise.} \end{cases}$$

Proposition 1 of (Sherali et al., 2009) showed that the following set of constraints completely characterize the set of valid pruned trees of $\mathcal{T}$.

$$z_h + \sum_{u \in p(h)} z_u = 1 \qquad \forall h \in \tilde{\mathcal{T}},$$
$$z_h \in \{0, 1\} \qquad \forall h \in \mathcal{T}.$$

A common decision tree pruning objective is to keep the probability of prediction error in the pruned tree as low as possible while reducing the number of tree nodes. Given a decision tree $\mathcal{T}$, it is easy to see that the overall probability of prediction error is of the pruned tree $\mathcal{T}_{\text{Prune}}$ is

$$\frac{1}{N} \sum_{h \in \mathcal{T}} e_h z_h. \tag{1}$$

Therefore a decision tree pruning problem can be formulated as the following integer program

$$\begin{aligned} \min_{z_h} \quad & \tfrac{1}{N} \textstyle\sum_{h \in \mathcal{T}} e_h z_h \\ \text{s.t.} \quad & z_h + \textstyle\sum_{u \in p(h)} z_u = 1 \quad \forall h \in \tilde{\mathcal{T}}, \\ & z_h \in \{0, 1\} \qquad\qquad\quad \forall h \in \mathcal{T}. \end{aligned} \tag{IP0}$$

By showing that the constraint matrix can be turned into a network matrix form, (Sherali et al., 2009) showed the above integer problem can be solved exactly by linear program relaxation.

4. Pruning with Feature Costs

Suppose the feature costs are given by $\{c_k : k = 1, \dots, K\}$. The feature cost incurred by an example is the total costs of *unique* features it encounters in all trees. This is because we assume whenever a feature is acquired its value is cached and subsequent usage incurs no additional cost. Specifically, the cost of classifying an example i on decision tree $\mathcal{T}$ is given by

$$c(\mathcal{T}, \mathbf{x}^{(i)}) = \sum_{k=1}^{K} c_k \mathbb{1}_{\left[\text{feature } k \text{ is used by } \mathbf{x}^{(i)} \text{ in } \mathcal{T}\right]} = \sum_{k=1}^{K} c_k w_{k,i},$$

where the binary variables $w_{k,i}$ serve as the indicator variables:

$$w_{k,i} = \begin{cases} 1 & \text{if feature } k \text{ is used by } \mathbf{x}^{(i)} \text{ in } \mathcal{T}, \\ 0 & \text{otherwise.} \end{cases}$$

Similarly, the cost of classifying $\mathbf{x}^{(i)}$ on an ensemble of T trees is

$$c(\mathcal{T}_{[T]}, \mathbf{x}^{(i)}) = \sum_{k=1}^{K} c_k \mathbb{1}_{\left[\text{feature } k \text{ is used by } \mathbf{x}^{(i)} \text{ in any } \mathcal{T}_t, t=1,\dots,T\right]}.$$

In a pruned tree $\mathcal{T}_{\text{Prune}}$ we can encode the conditions for $w_{k,i}$'s using the leaf indicator variable z_h's. If $z_h = 1$ for some node h, then the examples that are routed to h must have used all the features in the predecessor nodes $p(h)$. We use $k \sim p(h)$ to denote feature k is used in any predecessor of h. Then for each feature k and example i, we must have $w_{k,i} \geq z_h$ for all nodes h such that $i \in S_h$ and $k \sim p(h)$. Combining the error term (1) and feature cost in the objective, we arrive at the following integer program:

$$\begin{array}{lll} \min\limits_{z_h, w_{k,i}} & \frac{1}{N} \sum\limits_{h \in \mathcal{N}} e_h z_h + \lambda \sum\limits_{k=1}^{K} c_k \left(\frac{1}{N} \sum\limits_{i=1}^{N} w_{k,i}\right) & \\ \text{s.t.} & z_h + \sum_{u \in p(h)} z_u = 1 & \forall h \in \tilde{\mathcal{T}}, \\ & z_h \in \{0,1\} & \forall h \in \mathcal{T}, \\ & w_{k,i} \geq z_h & \forall h : i \in S_h \wedge k \sim p(h), \\ & & \forall k \in [K], \forall i \in S, \\ & w_{k,i} \in \{0,1\} & \forall k \in [K], \forall i \in S. \end{array} \tag{IP1}$$

Again, the constraint $w_{k,i} \geq z_h$ ensures that if h is a leaf node in the pruned tree ($z_h = 1$) and the ith example encounters feature k along the way before arriving at h then $w_{k,i}$ must be 1.

Unfortunately, unlike (IP0), the constraint set in (IP1) has fractional extreme points, leading to possibly fractional solutions to the relaxed problem. Consider Tree 1 in Figure 1. Feature 1 is used at the root node and feature 2 is used at node 3. There are 7 variables (assuming there is only one example and it goes to leaf 4):

$$z_1, z_2, z_3, z_4, z_5, w_{1,1}, w_{2,1}.$$

The LP relaxed constraints are:

$$z_1 + z_3 + z_4 = 1, z_1 + z_3 + z_5 = 1, z_1 + z_2 = 1,$$
$$w_{1,1} \geq z_4, w_{1,1} \geq z_3, w_{2,1} \geq z_4, 0 \leq z \leq 1.$$

The following is a basic feasible solution:

$$z_1 = 0, z_2 = 1, z_3 = z_4 = z_5 = 0.5, w_{1,1} = w_{2,1} = 0.5,$$

because the following set of 7 constraints are active:

$$z_1 + z_3 + z_4 = 1, z_1 + z_3 + z_5 = 1,$$
$$w_{1,1} \geq z_4, w_{1,1} \geq z_3, w_{2,1} \geq z_4, z_1 = 0, z_2 = 1.$$

Even if we were to interpret the fractional solution of z_h as probabilities of h being a leaf node, we see an issue with this formulation: the example has 0.5 probability of stopping at node 3 or 4 ($z_3 = z_4 = 0.5$). In both cases feature 1 at the root node has to be used; but $w_{1,1} = 0.5$ indicates that it's only being used half of the times, which is undesirable at all.

We have seen the LP relaxation of (IP1) fails to capture the desired behavior of the integer program. We now examine an alternative formulation and show that the optimal solution of its LP relaxation is exactly that of the integer program.

Given a tree $\mathcal{T}$, feature k and example $\mathbf{x}^{(i)}$, let $u_{k,i}$ be the first node associated with feature k on the root-to-leaf path the example follows in $\mathcal{T}$. Clearly, feature k is used by $\mathbf{x}^{(i)}$ if and only if none of the nodes between root and $u_{k,i}$ is leaf. In terms of constraints, we have

$$w_{k,i} + z_{u_{k,i}} + \sum_{h \in p(u_{k,i})} z_h = 1 \tag{2}$$

as long as feature k is used by $\mathbf{x}^{(i)}$ in $\mathcal{T}$. Intuitively, this constraint ensures that for the binary variable $w_{k,i}$ to be non-zero, the tree cannot be pruned before the feature k is obtained (the summation in the constraint equal to zero) and the feature k must be used in order to split the data (the term $z_{u_{k,i}}$ in the constraint equal to zero).

For a given tree $\mathcal{T}$ we arrive at the following formulation.

$$\begin{aligned}
\min_{z_h, w_{k,i}} \quad & \frac{1}{N}\sum_{h\in\mathcal{N}} e_h z_h + \lambda \sum_{k=1}^{K} c_k\Big(\frac{1}{N}\sum_{i=1}^{N} w_{k,i}\Big) \\
\text{s.t.} \quad & z_h + \textstyle\sum_{u\in p(h)} z_u = 1 && \forall h \in \tilde{\mathcal{T}}, \\
& z_h \in \{0,1\} && \forall h \in \mathcal{T}, \\
& w_{k,i} + z_{u_{k,i}} + \sum_{h\in p(u_{k,i})} z_h = 1, \forall k \in K_i, \forall i \in S, \\
& w_{k,i} \in \{0,1\} && \forall k \in [K], \forall i \in S,
\end{aligned} \tag{IP2}$$

where K_i denotes the set of features the ith example uses on tree $\mathcal{T}$.

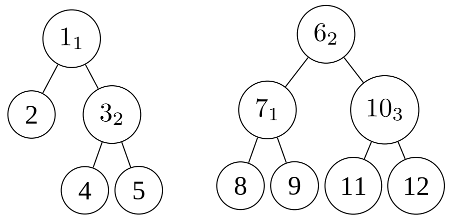
From tree to ensemble: we generalize (IP2) to ensemble pruning with tree index t: $z_h^{(t)}$ indicates whether node h in $\mathcal{T}_t$ is a leaf; $w_{k,i}^{(t)}$ indicates whether feature k is used by the ith example in $\mathcal{T}_t$; $w_{k,i}$ indicates whether feature k is used by the ith example in any of the T trees $\mathcal{T}_1, \ldots, \mathcal{T}_T$; $u_{t,k,i}$ is the first node that associated with feature k on the root-to-leaf path the example follows in $\mathcal{T}_t$. Note that we minimize the average empirical probability of error across all trees, which corresponds to the error of prediction based on averaging the leaf distributions across the ensemble for a given example.

$$\begin{aligned}
\min_{z_h^{(t)}, w_{k,i}^{(t)}} \quad & \frac{1}{NT}\sum_{t=1}^{T}\sum_{h\in\mathcal{N}^{(t)}} e_h^{(t)} z_h^{(t)} + \lambda \sum_{k=1}^{K} c_k\Big(\frac{1}{N}\sum_{i=1}^{N} w_{k,i}\Big) \\
\text{s.t.} \quad & z_h^{(t)} + \textstyle\sum_{u\in p(h)} z_u^{(t)} = 1 && \forall h \in \tilde{\mathcal{T}}_t, \forall t \in [T], \\
& z_h^{(t)} \in \{0,1\} && \forall h \in \mathcal{T}_t, \forall t \in [T], \\
& w_{k,i}^{(t)} + z_{u_{t,k,i}}^{(t)} + \sum_{h\in p(u_{t,k,i})} z_h^{(t)} = 1, \\
& && \forall k \in K_{t,i}, \forall i \in S, \forall t \in [T], \\
& w_{k,i}^{(t)} \in \{0,1\} && \forall k \in [K], \forall i \in S \forall t \in [T], \\
& w_{k,i}^{(t)} \le w_{k,i} && \forall k \in [K], \forall i \in S \forall t \in [T], \\
& w_{k,i} \in \{0,1\} && \forall k \in [K], \forall i \in S.
\end{aligned} \tag{IP3}$$

Lemma 4.1 *The equality constraints in (IP3) can be turned into an equivalent network matrix form for each tree.*

Proof This is simply due to an observation that $w_{k,i}^{(t)}$ can be regarded as just another z variable for a fictitious child node of $u_{t,k,i}$ and the rest of proof follows directly from the construction in Proposition 3 of (Sherali et al., 2009).

Figure 1 illustrate such a construction. For simplicity we consider only one example being routed to nodes 4 and 11 respectively on the two trees. The equality constraints in (IP3) can be separated based on the trees and put in matrix form:



(a) Tree 1 (b) Tree 2

Figure 1. An ensemble of two decision trees with node numbers and associated feature in subscripts

$$\begin{array}{c|ccccccc}
 & z_1 & z_2 & z_3 & z_4 & z_5 & w_{1,1}^{(1)} & w_{2,1}^{(1)} \\
\hline
r_1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
r_2 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
r_3 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
r_4 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
r_5 & 1 & 0 & 0 & 0 & 0 & 1 & 0
\end{array},$$

for tree 1 and

$$\begin{array}{c|ccccccccc}
 & z_6 & z_7 & z_8 & z_9 & z_{10} & z_{11} & z_{12} & w_{2,1}^{(2)} & w_{3,1}^{(2)} \\
\hline
r_1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
r_2 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
r_3 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
r_4 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
r_5 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
r_6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{array},$$

for tree 2. Through row operations they can be turned into network matrices, where there is exactly two non-zeros in each column, a 1 and a -1.

$$\begin{array}{c|ccccccc}
 & z_1 & z_2 & z_3 & z_4 & z_5 & w_{1,1}^{(1)} & w_{2,1}^{(1)} \\
\hline
-r_1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\
r_1-r_2 & 0 & 1 & -1 & -1 & 0 & 0 & 0 \\
r_2-r_3 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
r_3-r_4 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\
r_4-r_5 & 0 & 0 & 1 & 0 & 0 & -1 & 1 \\
r_5 & 1 & 0 & 0 & 0 & 0 & 1 & 0
\end{array},$$

for tree 1 and

$$\begin{array}{c|ccccccccc}
 & z_6 & z_7 & z_8 & z_9 & z_{10} & z_{11} & z_{12} & w_{2,1}^{(2)} & w_{3,1}^{(2)} \\
\hline
-r_1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
r_1-r_2 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
r_2-r_3 & 0 & 1 & 0 & 1 & -1 & -1 & 0 & 0 & 0 \\
r_3-r_4 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
r_4-r_5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\
r_5-r_6 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 1 \\
r_6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{array}$$

for tree 2. Note the above transformation to network matrices can always be done as long as the nodes are numbered in a pre-order fashion. Now we are ready to state the main theoretical result of this paper.

Theorem 4.2 *The linear program relaxation of* (IP3) *has only integral optimal solutions.*

Proof Denote the equality constraints of (IP3) with index set J_1. They can be divided into each tree. Each constraint matrix in J_1 associated with a tree can be turned into a network matrix according to Lemma 4.1. Stacking these matrices leads to a larger network matrix. Denote the $w_{k,i}^{(t)} \leq w_{k,i}$ constraints with index set J_2. Consider the constraint matrix for J_2. Each $w_{k,i}^{(t)}$ only appears once in J_2, which means the column corresponding to $w_{k,i}^{(t)}$ has only one element equal to 1 and the rest equal to 0. If we arrange the constraints in J_2 such that for any given k, i $w_{k,i}^{(t)} \leq w_{k,i}$ are put together for $t \in [T]$, the constraint matrix for J_2 has interval structure such that the non-zeros in each column appear consecutively. Finally, putting the network matrix from J_1 and the matrix from J_2 together. Assign J_1 and the odd rows of J_2 to the first partition Q_1 and assign the even rows of J_2 to the second partition Q_2. Note the upper bound constraints on the variables can be ignored as this is an minimization problem. We conclude that the constraint matrix of (IP3) is totally unimodular according to Theorem 2.7, Part 3 of (Nemhauser & Wolsey, 1988) with partition Q_1 and Q_2. By Proposition 2.1 and 2.2, Part 3 of (Nemhauser & Wolsey, 1988) we can conclude the proof.

We say a pruned tree of $\mathcal{T}$ is *optimal* for a given budget constraint if it has the lowest empirical error among all pruned trees of $\mathcal{T}$ that satisfy the budget constrain.

Corollary 4.3 *The linear program relaxation of* (IP3) *produces an* optimally pruned tree *for a given budget B.*

Proof Let the optimal value of (IP3) be $f(\lambda)$. As λ increases, a higher penalty is applied to the feature cost compared to the classification error; therefore, the optimal solution will have feature cost decreasing to 0 as a function of λ. Let λ^* be such that the feature cost $\sum_{k=1}^K c_k(\frac{1}{N}\sum_{i=1}^N w_{k,i}^*) = B$. Therefore,

$$\begin{aligned} f(\lambda^*) &= \frac{1}{NT}\sum_{t=1}^T \sum_{h\in\mathcal{N}^{(t)}} e_h^{(t)} z_h^{(t)*} + \lambda^* \sum_{k=1}^K c_k(\frac{1}{N}\sum_{i=1}^N w_{k,i}^*) \\ &= \frac{1}{NT}\sum_{t=1}^T \sum_{h\in\mathcal{N}^{(t)}} e_h^{(t)} z_h^{(t)*} + \lambda^* B. \end{aligned}$$

On the other hand, consider (IP3) with explicit budget constraint:

$$\begin{aligned} \min_{z_h^{(t)}, w_{k,i}^{(t)} \in Q} \quad & \tfrac{1}{NT}\sum_{t=1}^T \sum_{h\in\mathcal{N}^{(t)}} e_h^{(t)} z_h^{(t)} \quad , \qquad \text{(LP1)} \\ \text{s.t.} \quad & \textstyle\sum_{k=1}^K c_k(\frac{1}{N}\sum_{i=1}^N w_{k,i}) \leq B \end{aligned}$$

where Q denotes the constraint set of (IP3). Let opt be the optimal value of (LP1). Then we have

$$\begin{aligned} \text{opt} = \min_{z_h^{(t)}, w_{k,i} \in Q} \max_{\lambda \geq 0} \quad & (\frac{1}{NT}\sum_{t=1}^T \sum_{h\in\mathcal{N}} e_h^{(t)} z_h^{(t)} \\ & + \lambda(\sum_{k=1}^K c_k \frac{1}{N}\sum_{i=1}^N w_{k,i} - B)) \\ = \max_{\lambda \geq 0} \min_{z_h^{(t)}, w_{k,i} \in Q} \quad & (\frac{1}{NT}\sum_{t=1}^T \sum_{h\in\mathcal{N}} e_h^{(t)} z_h^{(t)} \\ & + \lambda(\sum_{k=1}^K c_k \frac{1}{N}\sum_{i=1}^N w_{k,i} - B)). \end{aligned}$$

By the definition of $f(\lambda)$ we have

$$\begin{aligned} \text{opt} &= \max_{\lambda\geq 0} f(\lambda) - \lambda B \\ &\geq f(\lambda^*) - \lambda^* B \\ &= \frac{1}{NT}\sum_{t=1}^T \sum_{h\in\mathcal{N}^{(t)}} e_h^{(t)} z_h^{(t)*} + \lambda^* B - \lambda^* B \\ &= \frac{1}{NT}\sum_{t=1}^T \sum_{h\in\mathcal{N}^{(t)}} e_h^{(t)} z_h^{(t)*} \end{aligned}$$

Thus we obtain the desired inequality.

Complexity: The number of $z_h^{(t)}$ variables is at most $T \times |\mathcal{T}_{\max}|$, where $|\mathcal{T}_{\max}|$ is the maximum number of nodes in a tree. The number of $w_{k,i}^{(t)}$ variables is at most $N \times T \times K_{\max}$, where $K_{\max}$ is the maximum number of features an example uses in a tree. The number of $w_{k,i}$ variables is at most $N \times \min\{T \times K_{\max}, K\}$. In total there are $T \times |\mathcal{T}_{\max}| + N \times T \times K_{\max} + N \times \min\{T \times K_{\max}, K\}$ variables. The number of $z_h^{(t)} + \sum_{u\in p(h)} z_u^{(t)} = 1$ constraints is at most $T \times |\tilde{\mathcal{T}}_{\max}|$, where $|\tilde{\mathcal{T}}_{\max}|$ is the maximum number of leaf nodes in any tree. The number of $w_{k,i}^{(t)} + z_{u_{t,k,i}}^{(t)} + \sum_{h\in p(u_{t,k,i})} z_h^{(t)} = 1$ constraints is at most $N \times T \times K_{\max}$. The number of $w_{k,i}^{(t)} \leq w_{k,i}$ constraints is again at most $N \times T \times K_{\max}$. In total there are at most $T \times |\tilde{\mathcal{T}}_{\max}| + 2 \times N \times T \times K_{\max}$ constraints besides the positivity constraints on all variables.

5. Parallel Ensemble Pruning

In this section we further explore the special structure of (IP3) and show that it admits a Dantzig-Wolfe decomposition that can be massively parallelized. The key observation is that pruning each tree is a shortest-path problem on directed graphs that can be efficiently solved ($O(|\tilde{\mathcal{T}}|^2)$). First, group the variables $\{z_h^{(t)}, w_{k,i}^{(t)}, \forall h \in$

$\tilde{\mathcal{T}}_t, \forall k \in K_{t,i}, \forall i \in S\}$ into a vector $\theta^{(t)}$ for each tree $\mathcal{T}_t, t = 1, \ldots, T$. Let P_t denote the feasible set corresponding to the first 4 sets of (LP-relaxed) constraints in (IP3) for tree $\mathcal{T}_t$:

$$\begin{aligned} P^{(t)} = \{\theta^{(t)} = (z_h^{(t)}, w_{k,i}^{(t)}) | z_h^{(t)} + \sum_{u \in p(h)} z_u^{(t)} = 1, \forall h \in \tilde{\mathcal{T}}_t, \\ w_{k,i}^{(t)} + z_{u_{t,k,i}}^{(t)} + \sum_{h \in p(u_{t,k,i})} z_h^{(t)} = 1, \forall k \in K_{t,i}, \forall i \in S, \\ z_h^{(t)} \geq 0, w_{k,i}^{(t)} \geq 0, \forall h \in \mathcal{T}_t, \forall k \in K_{t,i}, \forall i \in [N]\}. \end{aligned}$$

Thus, the LP relaxation of (IP3) can be re-written as

$$\begin{array}{lll} \min\limits_{z_h^{(t)}, w_{k,i}^{(t)}} & \frac{1}{NT} \sum\limits_{t=1}^{T} \sum\limits_{h \in \mathcal{T}_t} e_h^{(t)} z_h^{(t)} + \lambda \sum\limits_{k=1}^{K} c_k (\frac{1}{N} \sum\limits_{i=1}^{N} w_{k,i}) & \\ \text{s.t.} & \theta^{(t)} \in P^{(t)} & \forall t \in [T], \\ & w_{k,i}^{(t)} \leq w_{k,i} & \forall k \in [K], \forall i \in S \forall t \in [T], \\ & w_{k,i} \geq 0 & \forall k \in [K], \forall i \in S. \end{array} \tag{LP2}$$

Let $\hat{\theta}_i^{(t)}$,$i = 1, \ldots, I_t$ be the extreme points of $P^{(t)}$. Any point in $P^{(t)}$ can be written as a convex combination of these extreme points: $\theta^{(t)} = \sum_{j=1}^{I_t} \alpha_j^{(t)} \hat{\theta}_j^{(t)}, \sum_{j=1}^{I_t} \alpha_j^{(t)} = 1, \alpha_j^{(t)} \geq 0$. Thus we re-write (LP2) in terms of the extreme points of $P^{(t)}$:

$$\begin{array}{lll} \min\limits_{\alpha_j^{(t)}, w_{k,i}} & \frac{1}{NT} \sum\limits_{t=1}^{T} \sum\limits_{h \in \mathcal{T}_t} \sum\limits_{j=1}^{I_t} e_h^{(t)} \alpha_j^{(t)} \hat{z}_{h,j}^{(t)} + \lambda \sum\limits_{k=1}^{K} c_k (\frac{1}{N} \sum\limits_{i=1}^{N} w_{k,i}) & \\ \text{s.t.} & \sum_{j=1}^{I_t} \alpha_j^{(t)} \hat{w}_{k,i,j}^{(t)} \leq w_{k,i} & \forall i \in S, \forall k \in [K], \forall t \in [T], \\ & \sum_{j=1}^{I_t} \alpha_j^{(t)} = 1, \alpha_j^{(t)} \geq 0, & \forall t \in [T], \\ & w_{k,i} \geq 0 & \forall k \in [K], \forall i \in S, \end{array} \tag{LP3}$$

where $\hat{z}_{h,j}^{(t)}$ is the jth extreme point value of the node h on tree $\mathcal{T}_t$ and $\hat{w}_{k,i,j}^{(t)}$ is the jth extreme point value of $w_{k,i}^{(t)}$. In a more compact form, we can write (LP3) as

$$\begin{aligned} \min_{\alpha_i^{(t)}, w_{k,i}} \; & \frac{1}{NT} \sum_{t=1}^{T} \sum_{j=1}^{I_t} \alpha_j^{(t)} \mathbf{c}_t' \hat{\theta}_j^{(t)} + \lambda \sum_{k=1}^{K} c_k (\frac{1}{N} \sum_{i=1}^{N} w_{k,i}) \\ \text{s.t.} & \sum_{j=1}^{I_1} \alpha_j^{(1)} \begin{pmatrix} D^{(1)} \hat{\theta}_j^{(1)} \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \cdots + \sum_{j=1}^{I_T} \alpha_j^{(T)} \begin{pmatrix} D^{(T)} \hat{\theta}_j^{(T)} \\ 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} \\ & + \begin{pmatrix} D_w \mathbf{w} \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} D_s \mathbf{s} \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}, \\ & \alpha_j^{(t)} \geq 0, \forall t \in [T], \forall j \in [I_t] \\ & w_{k,i} \geq 0, \forall k \in [K], \forall i \in [N], \end{aligned} \tag{LP4}$$

where $D^{(t)}$ is the constant matrix selecting the $\hat{w}_{k,i,j}^{(t)}$ components of $\hat{\theta}_j^{(t)}$; $\mathbf{w}$ is the vector of $w_{k,i}$'s and $\mathbf{s}$ is the vector of slack variables $s_{k,i}^{(t)}$'s. The number of equality constraints is at most $N \times T \times K_{\max} + T$, much less than the number of constraints in (IP3). However, the number of variables can be huge.

The Danzig-Wolfe algorithm works as follows. Start with a feasible basis B of (LP4) and a dual vector $\mathbf{p}' = (\mathbf{q}', r_1, \ldots, r_T) = \mathbf{c}_B' B^{-1}$, where $\mathbf{q}$ corresponds to the constraints involving w's and $r_1, \ldots, r_T$ corresponds to the convexity constraints of the α's. For each tree t, solve the sub-problem

$$\begin{array}{lll} OPT_t = \min\limits_{\mathbf{x}} & (\mathbf{c}_t' - \mathbf{q}' D^{(t)}) \mathbf{x} \\ \text{subject to} & \mathbf{x}^{(t)} \in P^{(t)}. \end{array} \tag{SUB1}$$

If $OPT_t < r_t$, and the extreme point $\hat{\mathbf{x}}_i^{(t)}$ is optimal to the above sub-problem then it is easy to check that the reduced cost for the variable $\alpha_j^{(t)}$ is less than 0:

$$\begin{aligned} & \mathbf{c}_t' \hat{\theta}_j^{(t)} - \begin{bmatrix} \mathbf{q}' & r_1 & \ldots & r_T \end{bmatrix} \begin{bmatrix} D^{(t)} \hat{\theta}_j^{(t)} \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \\ & = (\mathbf{c}_t' - \mathbf{q}' D^{(t)}) \hat{\theta}_j^{(t)} - r_t < 0. \end{aligned}$$

Therefore, generate column

$$(D^{(t)} \hat{\theta}_i^{(t)}, 0, \ldots, 1, \ldots, 0)^T$$

and bring it into basis. Note due to the network matrix structure (Lemma 4.1), these subproblems can be solved very efficiently. Similarly, check the reduced costs for all $w_{k,i}$'s and $s_{k,i}^{(t)}$'s, and if any of them are negative, generate the corresponding columns and bring them into the basis. For the above decomposition, the main computational burden of pruning individual trees can be distributed to separate computional nodes that communicate to adjust for shared features. This can lead to dramatic efficiency improvement when the number of trees in the ensemble becomes large. Efficient implementation of the Dantzig-Wolfe decomposition has been shown to yield significant speedup through parallelism (Tebboth, 2001).

6. Experiments

We test our pruning algorithm on a number of benchmark datasets to show its advantage. Our pruning takes the ensembles from BudgetRF algorithm (Nan et al., 2015) as input. The datasets are CIFAR Krizhevsky, 2009, MiniBooNE, Forest Covertype, Sonar and Heart (Frank & Asuncion).

		no pruning	ens.pru.low	ind.pru.low	ens.pru.high	ind.pru.high
MiniB	cost	37.0671$\pm$0.3108	(68.24)25.2960$\pm$0.3157	(95.02)35.2219$\pm$0.3667	(43.17)16.0018$\pm$0.2498	(55.19)20.4584$\pm$0.1270
	error	0.0725$\pm$0.0004	0.0724$\pm$0.0005	0.0727$\pm$0.0004	0.0766$\pm$0.0004	0.0766$\pm$0.0008
Forest	cost	13.9005$\pm$0.0498	(88.10)12.2463$\pm$0.0834	(93.24)12.9604$\pm$0.1004	(65.16)9.0577$\pm$0.6481	(78.82)10.9565$\pm$0.0729
	error	0.1122$\pm$0.0009	0.1135$\pm$0.0010	0.1137$\pm$0.0010	0.1220$\pm$0.0025	0.1228$\pm$0.0010
Cifar	cost	186.5456$\pm$1.3180	(92.40)172.3720$\pm$1.8741	(93.02)173.5255$\pm$1.4516	(75.39)140.6308$\pm$2.5059	(77.89)145.2933$\pm$2.4797
	error	0.3152$\pm$0.0031	0.3165$\pm$0.0021	0.3158$\pm$0.0024	0.3227$\pm$0.0026	0.3236$\pm$0.0016
Sonar	cost	49.9715 $\pm$1.1103	(45.20)22.5860$\pm$3.9528	(74.31)37.1355$\pm$2.0425	(16.48)8.2349$\pm$1.7930	(28.11)14.0479$\pm$1.6909
	error	0.1539$\pm$0.0641	0.1838$\pm$0.0722	0.1890$\pm$0.0691	0.2121$\pm$0.0668	0.2139$\pm$0.0676
Heart	cost	12.1670$\pm$0.2341	(73.26)8.9133$\pm$0.7524	(96.20)11.7052$\pm$0.2742	(47.75)5.8094$\pm$2.5589	(75.86)9.2301$\pm$0.7036
	error	0.1721$\pm$0.0756	0.1711$\pm$0.0727	0.1719$\pm$0.0680	0.1977$\pm$0.0807	0.1973$\pm$0.0724

Table 1. Comparison of no pruning, ensemble pruning and individual pruning in terms of average feature costs and test error. Two different error levels for both ensemble and individual pruning methods are reported. Cost of the pruned trees are also reported as percentages of the cost of the unpruned trees in parenthesis.

Note the cost of each feature is 1 uniformly in all datasets and therefore cost is equivalent to the average number of unique features used for each example. For each dataset, we present the average cost and average error on test data in Table 1. As a baseline, we provide the performance of BudgetRF without pruning in the third column of Table 1. The results of our proposed ensemble pruning methods are in columns 4 and 6 under the title “ens.pru.”. We also compare to the same pruning algorithm that we propose but applied to individual trees separately rather than the entire ensemble. The results are given in column 5 and 7 of Table 1 under the title “ind.pru.”. Intuitively, pruning as an ensemble exploits the interdependencies among trees, potentially leading to better accuracy-cost trade-offs compared to pruning individual trees separately. We present pruning results at two different error levels: low and high. A low error level corresponds to little pruning with an implicitly larger average cost, while a high error level corresponds to pruning away much of the original trees, reducing the average cost at the expense of reduced classification performance.

For ease of comparison, we match the error levels for both ensemble and individual pruning methods and focus on the difference in cost. We also compute the cost as a percentage of the cost of the unpruned ensemble, shown in parenthesis in Table 1.

In MiniBooNE, Forest and CIFAR datasets, we run BudgetRF to obtain an ensemble of 40 trees following the given training/validation/test data splits (Nan et al., 2015). We report the mean and standard deviations based on 10 repeated runs. We observe that ensemble pruning reduces cost of the BudgetRF ensembles significantly while keeping the same level of test error. For example, the unpruned ensemble on MiniBooNE uses about 37 features on an average test example with an average test error of 0.0725; our ensemble pruning method reduces the average number of features to about 25, about 68% of the unpruned cost, with test error 0.0724. Further reduction of the cost to 43% of the original budget maintains approximately the same level of accuracy.

In Sonar and Heart datasets, we run BudgetRF to obtain an ensemble of 90 trees. Because of the small sizes, we perform 10-fold cross validation to obtain training/test splits and report the mean as well as standard deviation of test cost and error over 100 repeated runs. Again we observe the effectiveness of our pruning algorithm. For example in Heart the ensemble pruning uses 73% of the unpruned cost without losing accuracy.

We observe that ensemble pruning always performs better than individual pruning: fixing the error levels, Table 1 shows that ensemble pruning always incurs less feature cost than individual pruning. The advantage is quite significant in most of the datasets. This is expected because pruning individual trees does not exploit the inter-dependencies among trees.

7. Conclusion

We propose a novel ensemble pruning formulation with feature costs involving a 0-1 integer program. We prove that the linear program relaxation produces the optimal solution to the original integer program. This allows us to use efficient convex optimization tools to obtain the optimally pruned ensemble for any given budget. Our pruning formulation is general - it can take any ensemble of decision trees as input. As the pruning formulation explicitly account for feature sharing across trees together with accuracy/cost trade-off, it is able to significantly reduce feature cost by pruning subtrees that introduce more loss in terms of feature cost than benefit in terms of prediction accuracy gain. Empirically we see that our pruning algorithm indeed significantly improves the performance of the state of the art ensemble method BudgetRF.

References

Breiman, Leo, Friedman, Jerome, Stone, Charles J, and Olshen, Richard A. *Classification and regression trees*.

CRC press, 1984.

Chapelle, O, Chang, Y, and Liu, T (eds.). *Proceedings of the Yahoo! Learning to Rank Challenge, held at ICML 2010, Haifa, Israel, June 25, 2010*.

Frank, A. and Asuncion, A. UCI machine learning repository.

Gao, T. and Koller, D. Active classification based on value of classifier. In *Advances in Neural Information Processing Systems (NIPS 2011)*, 2011.

Krizhevsky, Alex. Learning Multiple Layers of Features from Tiny Images. Master's thesis, 2009.

Kulkarni, V.Y. and Sinha, P.K. Pruning of random forest classifiers: A survey and future directions. In *Data Science Engineering (ICDSE), 2012 International Conference on*, pp. 64–68, July 2012. doi: 10.1109/ICDSE.2012.6282329.

Li, Xiao-Bai, Sweigart, James, Teng, James, Donohue, Joan, and Thombs, Lori. A dynamic programming based pruning method for decision trees. *INFORMS J. on Computing*, 13(4):332–344, September 2001. ISSN 1526-5528.

Nan, F, Wang, J, Trapeznikov, K, and Saligrama, V. Fast margin-based cost-sensitive classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*, 2014.

Nan, Feng, Wang, Joseph, and Saligrama, Venkatesh. Feature-budgeted random forest. In Blei, David and Bach, Francis (eds.), *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1983–1991. JMLR Workshop and Conference Proceedings, 2015.

Nemhauser, George L. and Wolsey, Laurence A. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA, 1988. ISBN 0-471-82819-X.

Sherali, Hanif D., Hobeika, Antoine G., and Jeenanunta, Chawalit. An optimal constrained pruning strategy for decision trees. *INFORMS Journal on Computing*, 21 (1):49–61, 2009. doi: 10.1287/ijoc.1080.0278. URL http://dx.doi.org/10.1287/ijoc.1080.0278.

Tebboth, James Richard. A computational study of dantzig-wolfe decomposition. 2001.

Trapeznikov, K and Saligrama, V. Supervised sequential classification under budget constraints. In *International Conference on Artificial Intelligence and Statistics*, pp. 581–589, 2013.

Wang, J., Bolukbasi, T., Trapeznikov, K, and Saligrama, V. Model selection by linear programming. In *European Conference on Computer Vision*, pp. 647–662, 2014a.

Wang, J, Trapeznikov, K, and Saligrama, V. An lp for sequential learning under budgets. In *International Conference on Artificial Intelligence and Statistics*, 2014b.

Wang, Joseph, Trapeznikov, Kirill, and Saligrama, Venkatesh. Efficient learning by directed acyclic graph for resource constrained prediction. In Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 2143–2151. Curran Associates, Inc., 2015.

Xu, Zhixiang Eddie, Weinberger, Kilian Q., and Chapelle, Olivier. The greedy miser: Learning under test-time budgets. In *Proceedings of the 29th International Conference on Machine Learning, ICML*, 2012.