Optimization of Hierarchical Regression Model with Application to Optimizing Multi-Response Regression K-ary Trees

Pooya Tavallali
Electrical Engineering and Computer Science,
University of California, Merced
{ptavallali@ucmerced.edu}

Peyman Tavallali
Jet Propulsion Laboratory, California Institute
of Technology, Pasadena, CA 91109, USA
{peyman.tavallali@jpl.caltech.edu}

Mukesh Singhal
Electrical Engineering and Computer Science, University of California, Merced
{corresponding author:msinghal@ucmerced.edu}

Abstract

A fast, convenient and well-known way toward regression is to induce and prune a binary tree. However, there has been little attempt toward improving the performance of an induced regression tree. This paper presents a meta-algorithm capable of minimizing the regression loss function, thus, improving the accuracy of any given hierarchical model, such as k-ary regression trees. Our proposed method minimizes the loss function of each node one by one. At split nodes, this leads to solving an instance-based cost-sensitive classification problem over the node's data points. At the leaf nodes, the method leads to a simple regression problem. In the case of binary univariate and multivariate regression trees, the computational complexity of training is linear over the samples. Hence, our method is scalable to large trees and datasets. We also briefly explore possibilities of applying proposed method to classification tasks. We show that our algorithm has significantly better test error compared to other state-of-the-art tree algorithms. At the end, accuracy, memory usage and query time of our method are compared to recently introduced forest models. We depict that, most of the time, our proposed method is able to achieve better or similar accuracy while having tangibly faster query time and smaller number of nonzero weights.

Tree structured models are among the oldest ones in the supervised learning literature. Decision trees are popular in the scientific community due to their simple deployment, training and interpretation. In (Breiman et al. 1984), authors describe the fundamentals of binary decision trees and their applications, in both classification and regression. While decision trees can operate on problems with any number of labeled classes, other popular models, such as support vector machine, do not extend to multi-class problems naturally (Criminisi and Shotton 2013).

Despite mentioned advantages, tree training is a greedy algorithm that leads to sub-optimal structure and parameters. In this paper, we tackle this issue by providing a meta-algorithm that improves the performance of a given multi-response regression k-ary tree while keeping the training complexity linear for univariate and multivariate trees.

Conventionally, the goal is to minimize the following objective function

$$L_\alpha(T) = L(T) + \alpha|T|, \qquad \alpha > 0.$$

Here $L(T)$ is the loss function of a tree T. $|T|$ is the number of leaf nodes, and $\alpha > 0$ is a complexity cost. Pruning is performed in order to achieve a minimum of this cost function $L_\alpha(T)$ over the validation set. Finding the optimum of this objective function is an NP-complete problem (Laurent and Rivest 1976). Therefore, traditionally, it is not surprising that standard algorithms to train a tree are applied greedily.

A decision tree is trained by recursively partitioning the feature space into two (or several) parts, optimally with respect to some criteria (Breiman et al. 1984). The criteria are usually mean squared error for regression and purity cost functions for classification. the induction algorithm stops as soon as some stopping criteria are triggered, e.g., maximum depth and minimum number of samples at nodes. At test time, an input sample follows a path from the root to a leaf based on the hierarchy of decisions made at the split nodes. Finally, the prediction happens at a leaf node. In regression trees, the model at a leaf is a constant function in the space of target values.

There are two common split functions at nodes of a decision tree. The first type is called a univariate, a.k.a axis-aligned, node that consists of a single feature and a threshold to drive data to left or right. Finding optimal feature and threshold with respect to a criterion can be done by exhaustive search over all possible feature and threshold combinations. At node j consisting of N_j samples with a dimensionality of p, optimal combination is found in $\mathcal{O}(N_j p)$ using an incremental algorithm (Breiman et al. 1984). We have to mention that this complexity is based on assuming samples are sorted along each feature dimension before training. The second type of split function is termed multivariate node which consists of linear combination of features and a threshold. The complexity of finding an optimal linear combination is NP-complete (Heath, Kasif, and Salzberg 1993).

Here, the optimization method is performed by applying coordinate descent over weights one at a time (Breiman et al. 1984; Murthy, Kasif, and Salzberg 1994). The trees using second type of split nodes are called multivariate or oblique trees.

A main limitation of above-mentioned methods is that the greedy algorithms lead to sub-optimal and usually inaccurate trees. In addition, for multivariate trees, the coordinate descent optimization used at the split nodes may get stuck at a local minimum. Therefore, researchers have investigated several heuristics to avoid local optima such as restarting from a random initialization or applying random perturbations to escape from them (Murthy, Kasif, and Salzberg 1994). Unfortunately, these heuristics have shown marginal improvements(Murthy, Kasif, and Salzberg 1994).

To tackle above-mentioned shortcomings, we propose a meta-algorithm framework addressing the issue of optimizing a hierarchical model that is constrained to drive data down to only one child at a split node (e.g., k-ary tree). This means that each sample can only follow one path through the structure. Inspired by (Jordan and Jacobs 1994), our proposed framework optimizes a given multi-response regression k-ary tree by optimizing one node at a time while keeping the rest of structure fixed. Optimizing a split node j only depends on the set of samples S_j passing through the node. Minimizing the loss function depends on the traversed route of a sample from the node j. This means that node j should be able to route the samples in S_j correctly to one of its children such that the loss function is decreased optimally. We show that this leads to an instance-based cost-sensitive classification problem and solve it for cases of multivariate and univariate multi-response regression binary trees. At the leaf nodes, the prediction function is a constant function whose optimal value is the mean of the target responses in S_j. The algorithm iterates over tree nodes until there is no further improvement in the loss function. Generally, this meta-algorithm can be applied to different supervised learning problems and loss functions. At the end of proposed method, we explore possibility of applying our method to classification task and show that for a specific case of using 0-1 misclassification loss for classification binary tree will reduce to TreeAlternatingOptimization (TAO) (Carreira-Perpiñán and Tavallali 2018).

In the literature of multi-response regression trees, the convention is to induce an individual tree for each output response. During the test time, each output is predicted by its corresponding tree (Breiman et al. 1984). This approach leads to large models. In this paper, we do not follow this convention. Instead, a single tree is induced to predict all output values (De'Ath 2002), simultaneously.

In the rest of this paper, we first present related work. The proposed method is explained in detail and solutions to different kinds of splitting nodes are proposed. In "Experiments and Results" section , proposed method is compared with the state-of-the-art regression algorithms on various datasets. We also compare optimized regression trees with regression forest models. These experiments demonstrate generally better accuracy, query complexity and memory size.

Related Work

Classification and regression trees (CART) baseline methods are presented in (Breiman 2001). These methods are based on greedily growing and then pruning a tree. Despite being sub-optimal, they provide a good approximation for axis-aligned trees.

A recent survey by (Borchani et al. 2015) presents different state-of-the-art approaches toward multi-response regression tasks. The authors categorize the multi-regression to two groups. The first is a set of problem transformation methods that transform the multi-output problem into independent single-output problems solved by a single-output regression algorithm. The second group is a set of algorithm adaptation methods that adapt a specific single-output method to directly handle multi-output data sets. In this group, an example is the work done by (De'Ath 2002) that builds a regression tree following the same steps as CART. The only difference from the original CART is the redefinition of splitting criterion of a node as the sum of squared error over the multi-responses.

Authors in (Kocev et al. 2009) have explored and compared both mentioned groups of learning multi-response regression. In (Kocev et al. 2013), for purpose of improving performance, authors have considered the comparison of both previously mentioned groups using ensemble techniques of bagging (Breiman 1996) and random forests (Breiman 2001).

There are also papers aiming at different approaches toward inducing a regression tree. The concern of these papers is mainly concentrated around the splitting criterion used at the growing phase of a tree (Ikonomovska, Gama, and Džeroski 2011; Levatić et al. 2014).

Beside the regression trees, there is an extensive literature on classification trees that aim at exploring different splitting criterion and pruning schemes of a classification tree (Quinlan 2014; Loh and Shih 1997).

There have been many attempts toward optimal learning of fixed structure decision trees, specially, multivariate ones. For example, (Bennett 1992; 1994) has proposed a linear/multi-linear programming formulation towards finding a global optimum. This method was only proposed for binary classification and is also restricted to trees with few splits due to high complexity of the proposed optimization method.

(Nijssen and Fromont 2007) demonstrates how frequent itemsets can be used for constructing optimal univariate decision trees under a variety of constraints. The paper presents different types of constraints and proposes an approach for generating decision trees from lattices of itemsets that have been mined under these constraints. In (Garofalakis et al. 2003; Struyf and Džeroski 2005), authors are aiming toward inducting constrained based trees. In the former, constraints are considered during the tree induction, while in the latter, a large tree is built then pruned to satisfy the user constraints.

In (Norouzi et al. 2015a), the authors present an algorithm capable of optimizing the split and leaf parameters of the tree jointly, based on a global objective. Their proposed method is related to structured prediction with latent

variables. The optimization is initialized from an already induced tree called CO2 (Norouzi et al. 2015b) that utilizes a similar approach, this time in the induction phase of a multivariate tree.

Recently, in (Bertsimas and Dunn 2017), the authors have attempted to find an optimal decision tree with a fixed depth. It is done by linearizing the tree loss function along linear and binary constraints. This results is a mixed integer optimization (MIO). However, the algorithm has no guarantee on the computational complexity estimate since the proposed MIO solver attempts to find an exact solution to an NP-complete problem.

Compared to the conventional decision trees, soft decision trees incorporate probabilistic routes from the root node to the leaf nodes, (Jordan and Jacobs 1994). The learning process of these trees is cast as a maximum likelihood estimation. Log likelihood of a data set is obtained by multiplying densities at the leaf nodes. Later, it is optimized over weights. Generally, there are two approaches for solving this problem. One is through introducing missing data and then applying expectation maximization algorithm. The other consists of using backpropagation and applying a gradient based method (Jordan and Jacobs 1994). However, the main drawback of a soft decision tree is the computational complexity at the test time since all nodes have to be evaluated for a prediction. In contrast, a decision tree just needs to evaluate one path from the root node to a leaf, for each input data point.

A well-known application of decision trees is in ensemble methods. Shallow trees are usually used in boosting methods (Viola and Jones 2004; Freund, Schapire, and Abe 1999), while deep trees are used in random forests (Breiman 2001; Criminisi and Shotton 2013). One of the benefits of using boosting, or random forest models, is that they provide far better test accuracy compared to a single tree. On the other hand, a single tree is more interpretable and faster in test time.

Regression Trees

Consider a training data set (X, Y) containing N pairs of observations (x_i, y_i), $i = 1, ..., N$, where $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}^d$. The task of fitting a regression tree can be stated as the following minimization problem (Breiman et al. 1984):

$$\min_{F} \frac{1}{N} \sum_{i=1}^{N} L(y_i, T(x_i; F)) \quad \text{s.t.} \begin{cases} N_j \geq N_{min} \; \forall j \in leaves(T) \\ D \leq D_{max} \end{cases} \tag{1}$$

where T is the tree function, $F = \{f_1, ..., f_{|T|}\}$ is the decision functions at each node $1, ..., |T|$, $leaves(T)$ is set of leaf nodes in T, and $L(y_i, T(x_i; F))$ is the regression error function (mean square error, absolute error, etc.). The constraints in (1) contain the limit on the number of minimum acceptable samples N_{min} at each leaf N_j and tree's depth D smaller than maximum depth D_{max}. Finding optimal partitioning of the data that minimizes the loss function is NP-complete. Therefore, standard methods approximate it greedily through locally minimizing some cost function and recursively partitioning the data to grow the tree.

For regression, the cost function is usually squared error. Hence, the decision function parameters at node j are found through solving the following problem:

$$\begin{aligned} f_j^* = \underset{f_j}{\text{argmin}} \{ & \min_{c_1, c_2} \sum_{i: f_j(x_i) \leq 0} ||y_i - c_1||_2 + \\ & \sum_{i: f_j(x_i) > 0} ||y_i - c_2||_2 \} \\ \text{s.t} \quad & (x_i, y_i) \in S_j \quad \forall \quad i = 1...N. \end{aligned} \tag{2}$$

The decision function f_j can be $W_j^T x_i - b_j$, for a weight vector W_j, in a multivariate tree or simply $x_i^P - b_j$ for a univariate tree with P as one of the features, b_j and S_j are the threshold and subset of samples in the j^{th} node. Further, f_j corresponds to decision function at the j^{th} node. Also, c_1 and c_2 are constant vectors with the same dimension as the output responses.

In an axis-aligned tree, the best features and thresholds are found through exhaustive search. This can be done through an incremental algorithm in linear time $\mathcal{O}(Np)$. Problem (2) consists of selecting a feature and threshold such that loss of left and right children are minimized. We have $f_j = sign(x^P - b)$. Function f_j routes the samples down to the left or right child. The amount of objective function in (2) does not change when threshold b is a value between two consecutive values of x^P of samples. Therefore, in general there are $\mathcal{O}(Np)$ different combination of features and thresholds that have to be evaluated and the best combination is picked. To do so, exhaustive search is applied (Breiman et al. 1984). A naive approach is to simply calculate c_1 and c_2 for each partition incrementally and then evaluate objective function in (2) that takes $\mathcal{O}(N)$. Since there are also N different partitions along each dimension, this takes $\mathcal{O}(N^2 p)$. But one can find the best combination in $\mathcal{O}(Np)$. Assume along one dimension like P samples are sorted and sample indexes correspond to the place of a sample after being sorted. E.g. y_i corresponds to the target value of i^{th} sample after sorting. The loss function for each threshold between $i - 1^{th}$ and i^{th} for the left child can be written as follows(assuming y_i is one dimensional):

$$E_{i-1} = \sum_{i'=1}^{i-1} (y_{i'} - \hat{y}_{i-1})^2 \tag{3}$$

For each partition, optimal c_1 is the mean of target values in its partition. Therefore, in (3) we have $\hat{y}_{i-1} = \sum_{i''=1}^{i-1} \frac{y_{i'}}{i-1}$. For the next partition(or threshold) it can be written that:

$$E_i = \sum_{i'=1}^{i} (y_{i'} - \hat{y}_i)^2 \tag{4}$$

It is trivial that $\hat{y}_i = \frac{(i-1)\hat{y}_{i-1} + y_i}{i}$. By replacing it in (4), then using (3)and using $\sum_{i'=1}^{i-1} (y_{i'} - \hat{y}_{i-1}) = 0$ we have:

$$E_i = E_{i-1} + (y_i - \hat{y}_{i-1})^2 (1 - \frac{1}{i}). \tag{5}$$

The equation (5) enables the incremental calculation of error for left side of each threshold along one dimension. The error for the right side of each threshold should also be calculated and summed with error of the threshold's left side to have total error for each threshold. Then the error for all dimensions of output and input must be calculated. In total, this takes $\mathcal{O}(Npd)$. Note that since the output dimensions are independent, applying equation (5) to each output dimension individually is easily possible. Also, in this section, the assumption is that samples are sorted before induction of the tree.

Throughout the rest of paper, we call this method of solving problem (2) as multi-response regression tree (MRT) as in (De'Ath 2002). To the best of our knowledge, we have not found any method that constructs oblique multi-response regression tree. However, in case of classification, for various cost functions, the minimization problem (2) is found by applying coordinate descent optimization over weights W_j (Murthy, Kasif, and Salzberg 1994; Breiman et al. 1984). In this paper, the regression oblique trees are called OC1-reg since the same optimization techniques, as OC1 in (Murthy, Kasif, and Salzberg 1994), are used to construct oblique regression tree.

Optimization of Hierarchical Model for Regression (OHMR)

Given a multi-response regression k-ary tree, the objective function can be written as

$$\min_{F} \sum_{i=1}^{n} ||y_i - T(x_i; F)||_2. \tag{6}$$

Inspired by (Jordan and Jacobs 1994), the idea of our method is to iterate over nodes of the tree and optimize them one at a time while keeping the rest fixed. The algorithm terminates when no further improvement is possible after two consecutive passes, over all nodes, or the maximum allowed iterations is reached. At node j, the node receives sample points S_j from its parent and if only operates on these samples. Therefore, the optimization of node j only depends on S_j and no other samples. The task of a node is to distribute its samples among its children, hence, the optimization is performed on the node parameters in f_j. This can be cast as the following problem

$$\min_{f_j} \sum_{x_i \in S_j} ||y_i - T_j(x_i; F_j)||_2. \tag{7}$$

where, T_j is the subtree hanging from node j, F_j is the set of decision functions in T_j and $f_j (\in F_j)$ is the splitting function at node j. Now, minimization (7) can be rewritten as

$$\min_{f_j} \sum_{l} \sum_{(x_i, y_i) \in S_j^l} ||y_i - T_j^l(x_i; F_j^l)||_2 \tag{8}$$

Here T_j^l and F_j^l correspond to the tree and set of decision functions at the l^{th} child. First summation is over all children of the node. Also, we have $S_j^l = \{(\alpha, \beta) \mid (\alpha, \beta) \in S_j \quad \& \quad f_j(\alpha) = l)\}$. Tuple of (α, β) corresponds to an input sample at α and its target value β. In other words, S_j^l is set of samples routed to the l^{th} child by decision function f_j. In short, the problem consists of driving the arrived data at node j to its children minimizing the loss function. In this setting, there are two parameter sets to optimize: The first is the partitions S_j^l and the second is the decision node parameters f_j. In order to find S_j^l for each l, computationally, we introduce a variable z_i^l for each sample point defined as the amount of loss for sample x_i sent to the l^{th} child. In mathematical terms, $z_i^l = ||y_i - T_j^l(x_i; F_j^l)||_2$. Consequently, the optimization problem we have is

$$\min_{f_j} \sum_{l} \sum_{(x_i, y_i) \in S_j} z_i^l I(f_j(x_i), l) \tag{9}$$

where, I is an indicator function producing 0 if $f_j(x_i) \neq l$, and 1 if $f_j(x_i) = l$. Problem (9) is an instance-based cost-sensitive classification task because cost of a sample being sent to a child can be different from other samples. The solution to (9) depends on the decision function at each split node. Exact solution to the problem (9) could be an NP-hard problem depending on f_j. However, one can approximate it with existing classification methods in the literature. This approximation for multivariate and exact solution for axis-aligned regression trees are described later. Similarly, in the leaf nodes, the solution is based on the model at the leaf node. For example, in case of a constant function, the optimal value for the constant value is the mean of response values at the leaf node. The psudo-algorithm for our method is given in fig 1. The psudo-algorithm resembles the EM algorithm in (Jordan and Jacobs 1994) but the difference is the solution to the nodes (or gating and expert networks in (Jordan and Jacobs 1994)) and the fact that OHMR is applicable to k-ary and non-probabilistic regression trees. The convergence of our proposed algorithm is guaranteed based on the following theorem.

input data $\{(x_i, y_i)\}_{i=1}^N$, initial tree T,
-$\tau = 0$, $E_\tau = \sum_{i=1}^n ||y_i - T(x_i; F)||_2$,$E_{\tau-1} = inf$
while ($E_\tau \neq E_{\tau-1}$ and
  $\tau <$ maximum allowed passes)
  for each depth= $level$ of the tree
    for each split node "j" at $level$
      for each child l (there are k children in a k-ary tree)
        -compute z_i^l for each tuple of $(x_i, y_i) \in S_j$
      endfor
      -optimize cost-sensitive classification problem (9)
      considering sample costs
    endfor
    for each leaf node "j"
      -y_j = mean of target responses in S_j
    endfor
  endfor
-$\tau = \tau + 1$, $E_\tau = \sum_{i=1}^n ||y_i - T(x_i; F)||_2$
endwhile

Figure 1: OHMR applied to a regression tree

Theorem 1 (algorithm convergence)**.** *For any set of samples* (X, Y) *and any number of leaf nodes* $|leaves(T)|$ *of a tree*

T, the OHMR converges in a finite number of iterations for a fixed structure.

Proof. The proof is based on k-means clustering method and is also similarly done in (Carreira-Perpiñán and Tavallali 2018). It is trivial that, at each iteration, the loss function of a node is minimized. Therefore, at each node the loss function may decrease or at least remain the same as before. It is also important to mention that since the structure is fixed, the maximum number of leaf nodes are fixed $|leaves(T)|$. Hence, the maximum number of different assignments of points to the leaf nodes is fixed and is $N^{|leaves(T)|}$. In addition, the regression loss function is bounded from below by 0. Together, this means that the loss function cannot decrease more than a finite number of passes over the structure. Thus, it must stop decreasing at some point. □

In case of using an approximate algorithm, in order to guarantee convergence, one can introduce a rejection step in which new parameters to a node are accepted only if the objective function of problem (9) decreases. In practice, we may notice marginal difference for different orders of optimizing the nodes. In what follows next, we will propose how to apply the OHMR framework to binary multi-response regression univariate and multivariate trees.

OHMR for Multi-response Regression Multivariate Binary Trees

The decision function at nodes of a multivariate regression tree can be expressed as $f_j(x; W_j, b_j) = sign(W_j^T x - b_j)$. Accordingly, we can rewrite problem (9) as

$$\min_{W_j, b_j} \sum_{(x_i, y_i) \in S_j} t_i L(\operatorname{argmin}_l z_i^l, f_j(x_i)). \qquad (10)$$

Problem (10) is in fact a cost-sensitive binary classification. t_i is the cost of misclassifying a sample and is equal to $\max_l(z_i^l) - \min_l(z_i^l)$. In other words, if a sample is classified wrongly, there will be t_i increase in the loss function. Here, $L(.)$ is a 0-1 misclassification function. Since the objective function of (10) is piecewise constant and discontinuous, the problem is NP-hard and in practice is approximated using surrogate loss (Maibing and Igel 2015). Here, we approximate the solution to problem (10) using a modified logistic regression.

The easiest way to approximate (10) is to replace the 0-1 loss $L(.)$ with a logistic loss. For simplicity of notation, we replace $\operatorname{argmin}_l z_i^l$ with $l^* = \{-1, 1\}$. Therefore, we have:

$$\min_{W_j, b_j} \sum_{(x_i, y_i) \in S_j} t_i log(1 + e^{-(l^*(W^T x_i - b))}). \qquad (11)$$

Objective function in problem (11) is a convex function and one can easily optimize it using any gradient-based algorithm.However, implementation-wise, one might be interested in using highly optimized existing toolboxes for logistic regression such as liblinear (Fan et al. 2008). As per our search, no toolbox has the option for instance based cost-sensitive binary classification such as in (11). Therefore, we propose the following useful heuristic to use. For each datapoint multiply it's features and augmented bias by the misclassification cost of the datapoint (instead of x_i we feed $t_i \times [x_i, \quad 1]$) and then feed it to the toolbox. The intuition is that the shape of loss function changes such that the loss for misclassified samples with higher costs (heavy samples) rises faster as they get further away from decision boundary($f(x) = 0$) than the samples with lower costs (light samples). This still makes the loss function a surrogate to original 0-1 loss. Figure 2 shows 0-1 loss functions and our proposed modification.

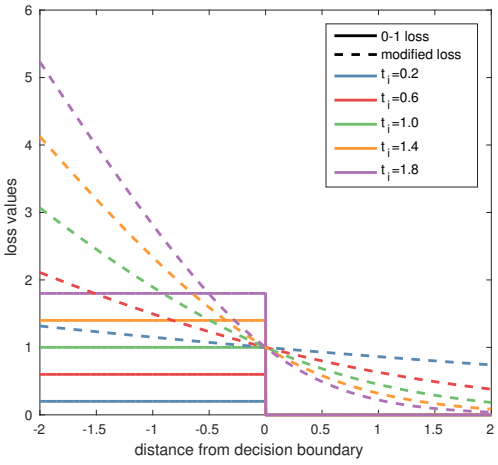

Figure 2: 0-1 loss and our proposed surrogate loss functions for different sample weights. Intuitively, as cost of a sample gets higher, the slope of surrogate loss gets steeper and essentially making it easier to classify.

OHMR for Multi-response Regression Univariate Binary Trees

In univariate regression trees, the decision function at a split node is $f_j = sign(x^{P_j} - b_j)$. This formulation is similar to problem (10) in which all the possible combinations of feature and threshold are $\mathcal{O}(N_j p)$. In this case, the solution can be found efficiently in $\mathcal{O}(N_j p)$ by using an incremental algorithm. We are interested in solving (10) with $f_j = sign(x^P - b)$. This is a selection of a feature and threshold. Again similar to section case of splitting a univariate node, there are $\mathcal{O}(N)$ available thresholds along each feature and $\mathcal{O}(Np)$ in total. One needs to calculate error for each threshold along a feature and the error. Error of left child is equal to summation $E_i = \sum_{i'=1}^{i} z_{i'}^{-1}$ (it is because z_i^l is equal to error of sample sent to l^{th} child and $l = -1$ corresponds to left child). Therefore, by shifting indexes, we have $E_i = E_i + z_i^{-1}$ which in practice is calculating errors incrementally and takes $\mathcal{O}(N)$. The same procedure can be applied for the right child in reverse order. Afterwards, adding both errors of left and right children and then finding the global minimum error and picking its respective feature and threshold. This has to be applied to all p input features, hence, total computational complexity is $\mathcal{O}(Np)$. Note that

since z_i^l is one dimensional, computational complexity does not depend on the dimensionality of y_i.

OHMR for Classification Multivariate Binary Trees

In order to tackle a classification problem, 0-1 loss misclassification can be used instead of regression loss in (6) and follow the procedure to problem (9). It is trivial to see that z_i^l at the node is either 0 or 1 (because the loss function produces 0 or 1). Therefore, it is possible to have similar costs for several children while in the regression case z_i^l can be any positive real value. Further for case of classification binary trees, by following the same analysis as for case of binary regression trees, it can be seen that sample cost t_i becomes either 0 or 1. Essentially meaning the problem only depends on samples that have sample cost of 1; hence, leading to a binary classification(TAO algorithm in (Carreira-Perpiñán and Tavallali 2018)).

An alternative method to solve a classification problem using regression loss is to change the categorical variable outputs to hot-one-vectors. By doing so sample costs at the nodes will remain positive real value; hence, terminating confusion caused by 0 cost samples. For classification experiments, in this paper the second method is applied. However, focus of the current paper is on regression problems and We are not trying to address classification in this paper, as it has been done in some previous work (Norouzi et al. 2015b; 2015a; Frosst and Hinton 2017; Carreira-Perpiñán and Tavallali 2018; Bertsimas and Dunn 2017). However, we are trying to show the capabilities of OHMR in treating classification as regression.

Computational Complexity

Theorem 2 (computational complexity)**.** *Computational complexity of one pass of OHMR over the k-ary tree (k denotes the number of children each node has) structure is $\mathcal{O}(Dg(N,p)+kD^2Nh(p))$, if following condition holds for $g(N,p)$:*

$$\sum_j g(N_j,p) \leq g(N,p) \; s.t \begin{cases} \sum_j N_j \leq N \\ N_j, N \in \mathbb{Z}^+ \end{cases} . \qquad (12)$$

Here $g(.)$ and $h(.)$ are the computational complexities of training over N samples and evaluating a sample for the model at the nodes, respectively.

Proof. First, we define the computational complexity of one pass over the tree.

$$\sum_{n=1}^{D} \sum_{depth(j)=n} g(N_j,p) + k(D-n)N_j h(p). \qquad (13)$$

Here, the index of first summation is over the depth of the tree and second summation is over the nodes at the same depth of n. $depth(j) = n$ denotes index of any node j that is at depth n. First term, $g(N_j,p)$ is due to training model at node j. Second term, $k(D-n)N_j h(p)$ is due to propagating each sample (these are N_j samples) down each k children (which are a subtree of depth $D-n$) and calculating its costs at node j. Since each input sample in the structure can only follow a fixed path, at a certain depth, a sample can be present in at most only at one node. As a result, we have $\sum_{depth(j)=n} N_j \leq N$. Hence, $\sum_{depth(j)=n} g(N_j,p) \leq g(N,p)$. This leads to the fact that at depth n, we have $\sum_{depth(j)=n} g(N_j,p) + k(D-n)N_j h(p) \leq g(N,p) + k(D-n)Nh(p)$. So, the total computational complexity of training models is $\sum_{n=1}^{D} g(N,p) + k(D-n)Nh(p) = Dg(N,p) + k\frac{D(D+1)}{2}Nh(p)$. Therefore, asymptotically total computational complexity is $\mathcal{O}(Dg(N,p) + kD^2Nh(p))$. □

computational complexity of applying OHMR to regression binary trees: for multivariate case: solving a logistic loss function using truncated Newton method costs $\mathcal{O}(\alpha Np)$ (Hsia, Zhu, and Lin 2017) ($g(N,p) = \alpha Np$). α is the number of iterations performed. Evaluating each split node takes $h(p) = p$. Therefore, by plugging h and g in theorem 2, OHMR's learning complexity is $\mathcal{O}(D\alpha Np + 2D^2Np))$. The algorithm is linear on the number of samples. Similarly, for univariate case, the computational complexity is $\mathcal{O}(DNp + 2D^2)$.

Experiments and Results

In this section, we present results of experiments to show the merits of our proposed method. In all these experiments, OHMR is applied to an induced MRT or OC1-Reg tree. MRT is similar to the CART algorithm, except that the this algorithm is applied to a multi-response tree like the work in (De'Ath 2002). OC1-reg is also OC1 optimization algorithm for a single oblique tree applied for multi-response regression. To the best of our knowledge, it is the first time this method has been applied to create one OC1 tree for a multi-response regression task. In all experiments, at the split optimization, logistic regression was trained using (Fan et al. 2008) library. In each figure, the dataset and its dimensionality is written left to the figure as $datasetname(N \times p, d)$. All reported errors are mean squared error.

The optimized MRT and OC1-Reg are called MRT-OHMR and OC1-OHMR, respectively. Maximum number of allowed iterations over tree was set to 10. As a reminder, we mention that the OHMR model for the experiments is the model introduced for multivariate trees.

Comparison With Regression Trees

During the regression experiments, the datasets were randomly partitioned to 64% train, 16% validation and 20% test sets. Concrete Compressive Strength(CCS), Airfoil, CT slice localization and California housing(CADATA) datasets were downloaded from (Dheeru and Karra Taniskidou 2017). Classification datasets and regression dataset of Year Prediction were downloaded from (Chang and Lin 2011).

Figure 3 shows the results of applying OHMR to different datasets. OHMR has been able to decrease test and train error by $25-50\%$ in most datasets. We can see from these figures that as trees get deeper, overfitting might happen. This issue can be handled by simply applying cross-validation

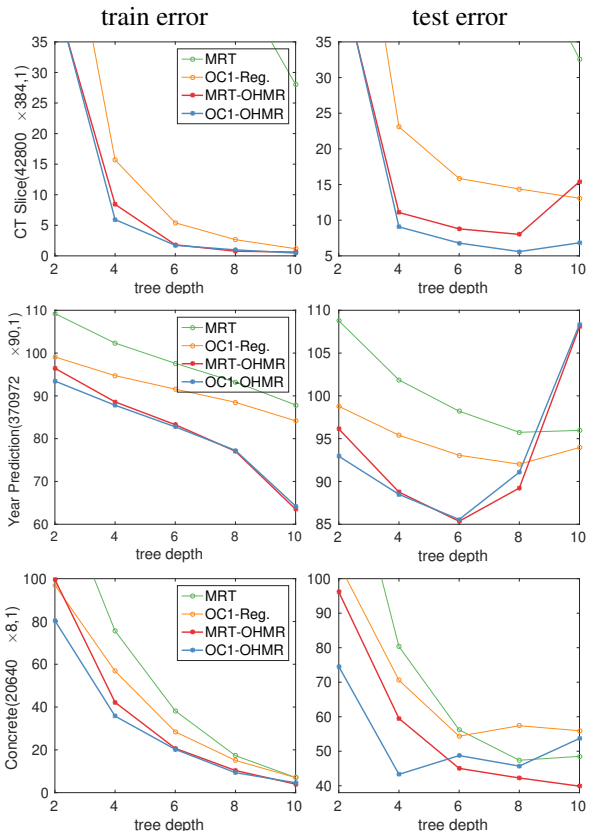

Figure 3: Train and test errors(mean squared error) of the proposed method as compared to other greedy algorithms in the literature. OHMR has achieved the smallest test error in all 3 datasets, winning all time in 2 datasets and best train in all of them.

and picking the best tree among trees of different depths. In addition, the algorithm is also fast and usually takes 25 seconds to optimize a tree of depth 8 over a dataset like CT Slice using a device with 4 core i7 cpu and 8 GB of ram.

Comparison With Forest Models

The fact with the regression trees is that they are fast in query complexity. Hence, one important problem is to see if an oblique tree can do as good as, or on par as, a forest model while preserving the tree's faster query complexity. To check this, we compared OHMR with a few recently proposed forest models. For a fair comparison, we followed the same dataset partition as in (Begon, Joly, and Geurts 2017; Li and Martin 2017) for each dataset. The query complexity consists of the number of element-wise multiplications and additions needed to propagate a sample down the trees or forest. The number of parameters consists of all existing parameters throughout the model nodes. Methods that do not fit inside the plots are put on the boundaries of them.

For CADATA, dataset was partitioned into 66% train and 34% test sets, each experiment was done 10 times and the averages of the test errors are reported. For Airfoil and Concrete, datasets were partitioned into 60% train and 40% test sets and each experiment was conducted 20 times. OHMR was applied to trees of depth 2 to 10. The errors of these trees are reported as a function of query complexity and number of parameters; see figure 4.

In Figure 4, red and blue lines show MRT-OHMR and OC1-OHMR,respectively. Each two consecutive dots on the line represent two trees at consecutive depths. The trees on the lines start from depth 2 and increases up to depth 10. Huber, and Tukey were forest models from (Li and Martin 2017). The forests in this paper were first induced by a random forest of 1000 trees as a kernel for non-parametric models. Then the non-parametric models were used to predict target values with respect to different loss functions such as Huber (Huber and others 1964) and Tukey (Huber 2011). Random forest (RF) in all datasets consist of 100 trees of depth 9. ET100, ET10, ET1, GIF10, GIF1 are forest models from (Begon, Joly, and Geurts 2017). ET100 is a random forest of 1000 fully grown trees. ET10 and ET1 are the same as ET100, except that they are built such that they have as 10% and 1% many nodes as ET100, respectively. GIF10 and GIF1 are the authors proposed method greedily inducing forests that has as 10% and 1% many nodes as ET100, respectively. Since the information about query complexity of ET100, E10 and ET1 were not available, they were not reported in figures consisting of query complexity.

The query complexity and size of other Huber and Tukey forests are estimated in the experiments based on the minimum needed query complexity or number of parameters reported in the (Huber and others 1964). Figure 4 ,the graphs in the left column, demonstrates the fact that a single oblique tree can outperform, or do almost as good as, a forest model. However, in all cases, the query complexity of OHMR is orders of magnitude smaller.

Figure 4,the graphs in the right column, shows the performance of different models with respect to the number of parameters of each model. As a tree grows larger, the number of its parameters increase exponentially. However, still with small trees, OHMR was able to achieve comparable or smaller error compared to the forest models.

Compressed multivariate trees One direct application of OHMR is to use it toward building sparse multivariate trees by imposing l_1 penalty over the weights $\lambda_j||W_j||_1$ of each node. λ_j is the regularization parameter for each node. This helps achieve trees with smaller size and faster query complexity, specifically useful for large datasets such as CT slice. To do so, two general approaches can be followed. One is to set λ_j equal for all nodes and similar to lasso (Hastie, Tibshirani, and Friedman 2009) follow the regularization path (increase λ_j from a small value to a large value) and at each value of λ_j optimizing the tree. However, since the lower level nodes in the tree receive smaller number of samples, their weights get sparser with smaller values of λ_j, resulting in subtrees with no samples; hence, a post processing procedure is needed to prune the inactive subtrees (e.g. (Carreira-Perpiñán and Tavallali 2018)). Second approach is to set λ_j for each node individually. To do so, a balance between λ_j and N_j has to be scheduled to make sure lower

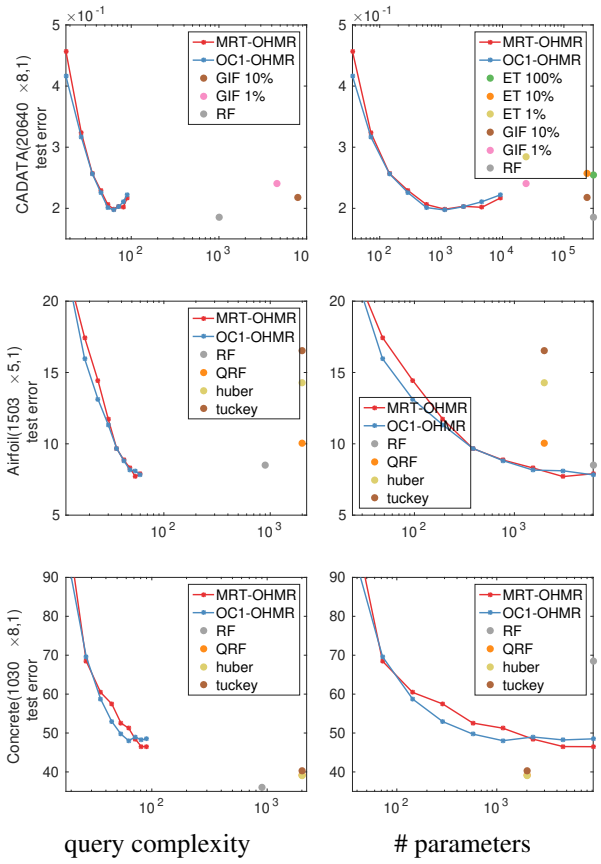

Figure 4: Test error(mean squared error) versus query complexity and number of parameters are shown over different data sets for OHMR and several forest models.

level nodes do not get heavily penalized. For this we set $\lambda_j/N_j = C$ in which C is a constant value. As a result, λ_j will change proportional to number of samples the node receives. Then, follow the regularization path based on C, increasing C from a small value to a large one and optimize the tree for each value of C. In practice, we noticed the second approach results in faster and similar size trees compared to first approach. Further, each node gets similar sparsity ratio to other nodes through the tree; hence resulting in no structural changes. Nodes may occasionally get empty (unless regularization penalty is high). Note that the rejection step must not be applied, otherwise, updates are rejected and sparsity is not achieved. From the trees along the regularization path, the best tree can be picked by cross-validation or in case the validation set is not provided, by looking at the train error of the trees versus their sparsity and picking the sparsest tree among trees with lowest train error. The second approach experiment is applied over CT slice and is compared with other forest, nearest neighbor and radial basis function (RBF) models. Same data partition as (Begon, Joly, and Geurts 2017) was followed. Figure 5 shows different methods compared to OHMR. Both top graphs of figure 5 show that sparse OHMR was able to achieve mostly smaller test error while having order of exponent faster query and

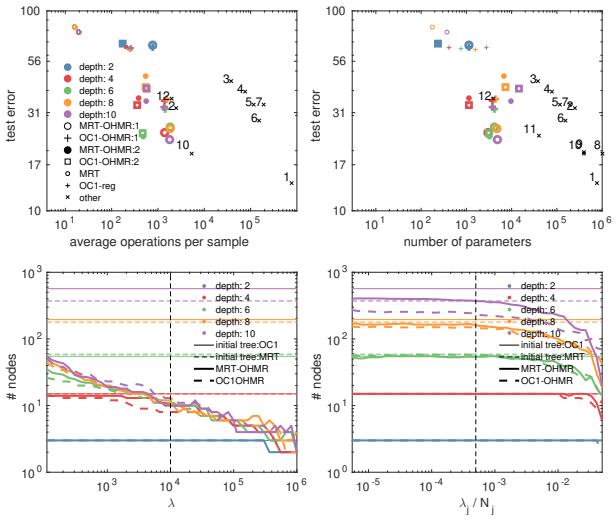

Figure 5: Test error versus query complexity and size are shown over CT slice dataset in top two curves. Other models are shown by $\times$ symbol and a number beside it. MRT-OHMR:1, OC1-OHMR:1 and MRT-OHMR:2, OC1-OHMR:2 correspond to first and second setup of experiments mentioned in compressed trees paragraph respectively. 1 is nearest neighbor. 2 is a bagging ensemble of 250 trees of depth 10. Sampling rate was 0.7. models 3,4,5,6 and 7 are RBF models with 100,200,300,400,500 basis functions and used as transformation to a linear regression. Gaussian functions with same variance were used and centers were found using k-means. 8,9,10,11,12, are ET100, ET10, GIF10, ET1 and GIF1 respectively. The bottom two curves show number of nodes remained as λ_j/N_j or λ increases. Left and right figures of second row correspond to first and second setup of experiments mentioned in compressed trees paragraph respectively. The black horizontal line displays the selected tree models shown in the bottom two curves.

smaller size. Some nodes getting empty due to receiving no training samples were pruned after the experiment procedure. The bottom two graphs of figure 5 have explored number of nodes through the experiments procedure. # nodes is the number of nodes.

OHMR Applied to Axis-aligned Regression Trees

In Tables 1 and 2, accuracy and test error of applying OHMR to MRT is shown for classification and regression tasks, respectively. The classification tasks were changed to regression problem as explained in classification section of the paper. All datasets were randomly partitioned to 50% train, 25% test and 25% validation. The algorithm was compared to recently introduced optimal classification trees (OCT) (Bertsimas and Dunn 2017). OHMR is able to achieve better or similar test accuracy compared to OCT. OCT's run time in practice could take up to half an hour, whereas OHMR terminates in order of 0.1 second. All datasets are downloaded from (Dheeru and Karra Taniskidou 2017).

Table 1: Test classification accuracy (avg $\pm$ stdev) for different methods and datasets (sample size $\times$ dimensionality, # classes). All models are axis-aligned trees.

DATA SET	DEPTH	MRT	OHMR	OCT
BREAST	2	90.8 ± 1.2	**92.9** ± 2.0	91.9
CANCER	3	92.9 ± 0.5	**93.2** ± 1.2	91.5
$(569 \times 30, 2)$	4	92.8 ± 0.6	**93.1** ± 0.5	91.5
SPAMBASE	2	82.5 ± 2.4	**85.9** ± 0.9	84.3
$(4601 \times 57, 2)$	3	87.2 ± 1.1	**89.1** ± 1.0	86.0
	4	89.7 ± 0.9	**90.4** ± 0.7	86.1

Table 2: Mean squared error over test set (avg $\pm$ stdev) for different methods and datasets (sample size $\times$ dimensionality, # output dimensionality). All models are axis-aligned trees.

DATA SET	DEPTH	MRT	OHMR
PARKINSON	2	117.8 ± 3.4	**116.0** ± 3.5
TELEMONITORING	3	89.2 ± 5.1	**85.1** ± 4.4
$(5875 \times 20, 2)$	4	61.3 ± 6.3	**53.4** ± 6.2
GAS$\times 10^{-2}$	2	13.1 ± 6.9	**7.8** ± 3.5
$(58 \times 432, 2)$	3	10.7 ± 10.2	**8.0** ± 6.1
	4	**10.9** ± 8.5	13.2 ± 8.6
OES$\times 10^{-6}$	2	29.2 ± 17.1	**27.6** ± 14.4
$(267 \times 263, 16)$	3	25.1 ± 12.2	**24.3** ± 14.9
	4	23.7 ± 11.4	**21.1** ± 10.7

Classification as Regression

In this section, we present examples on how a classification problem can be handled as a regression using OHMR. The target classes are changed to hot-one-vector in which the dimension is 1 if the class of a point belongs to that dimension and other dimensions are zero. This enables a regression tree to handle a classification problem in the context of regression. Datasets are from in (Chang and Lin 2011). Our proposed method was able to improve significantly over test and train accuracies and outperform recently introduced optimization method in (Norouzi et al. 2015a) and greedy tree in (Norouzi et al. 2015b) called non-greedy and CO2, respectively. Specifically, in MNIST, OHMR was able to pass accuracy 90% test accuracy at depth of 4 and also achieve the best accuracy of 95.39% at depth 12.

Conclusion

This paper tackled the problem of optimizing a multi-response regression tree through a meta-algorithm that proposes general procedure for optimizing any hierarchical (e.g., k-ary trees) structure for regression (OHMR). OHMR provides opportunity for also learning sparse weights at the nodes. Inspired by (Jordan and Jacobs 1994), OHMR is done by iterating over the nodes and optimizing one at a time which leads to instance-based cost-sensitive classification at the split nodes and a regular regression problem at the leaf nodes. Proposed method has an efficient training procedure making it possible to scale the algorithm to larger datasets

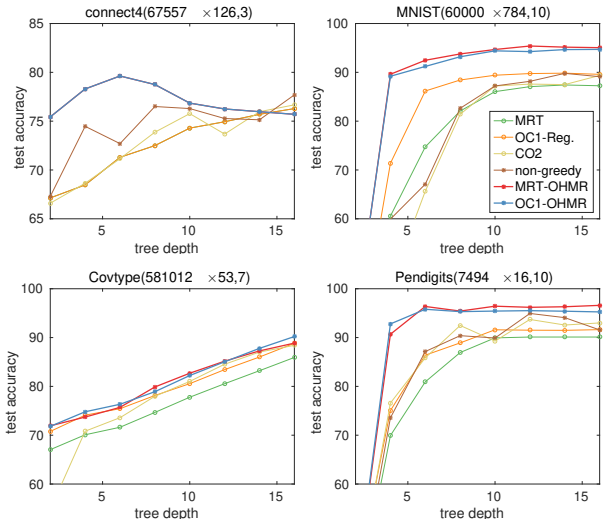

Figure 6: Test accuracy of the proposed method compared to other greedy and optimization algorithms in the literature. OHMR has almost achieved the best test accuracy in most data sets.

and models. OHMR is also capable of competing with existing ensemble methods and even achieving smaller test error, hence, possibility of essentially decreasing query complexity and size of the model. Because of OHMR other research topics using trees such as using other loss functions toward other tasks such as density estimation, manifold learning, multi-class labeling and using more complicated models at the split nodes have now become easily possible. All that is needed is to approximate or solve problem (9) for any given task.

Acknowledgment

Peyman Tavallali's research contribution to this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

Begon, J.-M.; Joly, A.; and Geurts, P. 2017. Globally induced forest: A prepruning compression scheme. In *International Conference on Machine Learning*, 420–428.

Bennett, K. P. 1992. Decision tree construction via linear programming. In *Proc. 4th Midwest Artificial Intelligence and Cognitive Sience Society Conference*, 97–101.

Bennett, K. P. 1994. Global tree optimization: A non-greedy decision tree algorithm. *Computing Science and Statistics* 26:156–160.

Bertsimas, D., and Dunn, J. 2017. Optimal classification trees. *Machine Learning* 106(7):1039–1082.

Borchani, H.; Varando, G.; Bielza, C.; and Larrañaga, P. 2015. A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 5(5):216–233.

Breiman, L. J.; Friedman, J. H.; Olshen, R. A.; and Stone, C. J. 1984. *Classification and Regression Trees*. Belmont, Calif.: Wadsworth.

Breiman, L. 1996. Bagging predictors. *Machine learning* 24(2):123–140.

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

Carreira-Perpiñán, M. Á., and Tavallali, P. 2018. Alternating optimization of decision trees, with application to learning sparse oblique trees. In *Advances in Neural Information Processing Systems*, 1219–1229.

Chang, C.-C., and Lin, C.-J. 2011. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)* 2(3):27.

Criminisi, A., and Shotton, J. 2013. *Decision Forests for Computer Vision and Medical Image Analysis*. Advances in Computer Vision and Pattern Recognition. Springer-Verlag.

De'Ath, G. 2002. Multivariate regression trees: a new technique for modeling species–environment relationships. *Ecology* 83(4):1105–1117.

Dheeru, D., and Karra Taniskidou, E. 2017. UCI machine learning repository.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin, C.-J. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research* 9(Aug):1871–1874.

Freund, Y.; Schapire, R.; and Abe, N. 1999. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence* 14(771-780):1612.

Frosst, N., and Hinton, G. 2017. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*.

Garofalakis, M.; Hyun, D.; Rastogi, R.; and Shim, K. 2003. Building decision trees with constraints. *Data Mining and Knowledge Discovery* 7(2):187–214.

Hastie, T.; Tibshirani, R.; and Friedman, J. H. 2009. *The elements of statistical learning: data mining, inference, and prediction*. New York, NY: Springer, second edition.

Heath, D.; Kasif, S.; and Salzberg, S. 1993. Induction of oblique decision trees. In *IJCAI*, volume 1993, 1002–1007.

Hsia, C.-Y.; Zhu, Y.; and Lin, C.-J. 2017. A study on trust region update rules in newton methods for large-scale linear classification. In *Asian Conference on Machine Learning*, 33–48.

Huber, P. J., et al. 1964. Robust estimation of a location parameter. *The annals of mathematical statistics* 35(1):73–101.

Huber, P. J. 2011. Robust statistics. In *International Encyclopedia of Statistical Science*. Springer. 1248–1251.

Ikonomovska, E.; Gama, J.; and Džeroski, S. 2011. Incremental multi-target model trees for data streams. In *Proceedings of the 2011 ACM symposium on applied computing*, 988–993. ACM.

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

Kocev, D.; Džeroski, S.; White, M. D.; Newell, G. R.; and Griffioen, P. 2009. Using single-and multi-target regression trees and ensembles to model a compound index of vegetation condition. *Ecological Modelling* 220(8):1159–1168.

Kocev, D.; Vens, C.; Struyf, J.; and Džeroski, S. 2013. Tree ensembles for predicting structured outputs. *Pattern Recognition* 46(3):817–833.

Laurent, H., and Rivest, R. L. 1976. Constructing optimal binary decision trees is np-complete. *Information processing letters* 5(1):15–17.

Levatić, J.; Ceci, M.; Kocev, D.; and Džeroski, S. 2014. Semi-supervised learning for multi-target regression. In *International Workshop on New Frontiers in Mining Complex Patterns*, 3–18. Springer.

Li, A. H., and Martin, A. 2017. Forest-type regression with general losses and robust forest. In *International Conference on Machine Learning*, 2091–2100.

Loh, W.-Y., and Shih, Y.-S. 1997. Split selection methods for classification trees. *Statistica sinica* 815–840.

Maibing, S. F., and Igel, C. 2015. Computational complexity of linear large margin classification with ramp loss. In *Artificial Intelligence and Statistics*, 259–267.

Murthy, S. K.; Kasif, S.; and Salzberg, S. 1994. A system for induction of oblique decision trees. *Journal of artificial intelligence research* 2:1–32.

Nijssen, S., and Fromont, E. 2007. Mining optimal decision trees from itemset lattices. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 530–539. ACM.

Norouzi, M.; Collins, M.; Johnson, M. A.; Fleet, D. J.; and Kohli, P. 2015a. Efficient non-greedy optimization of decision trees. In *Advances in Neural Information Processing Systems*, 1729–1737.

Norouzi, M.; Collins, M. D.; Fleet, D. J.; and Kohli, P. 2015b. Co2 forest: Improved random forest by continuous optimization of oblique splits. *arXiv preprint arXiv:1506.06155*.

Quinlan, J. R. 2014. *C4. 5: programs for machine learning*. Elsevier.

Struyf, J., and Džeroski, S. 2005. Constraint based induction of multi-objective regression trees. In *International Workshop on Knowledge Discovery in Inductive Databases*, 222–233. Springer.

Viola, P., and Jones, M. J. 2004. Robust real-time face detection. *International journal of computer vision* 57(2):137–154.