ALTERNATING OPTIMIZATION OF DECISION TREES, WITH APPLICATION TO LEARNING SPARSE OBLIQUE TREES

Miguel Á. Carreira-Perpiñán and **Pooya Tavallali**, EECS, University of California, Merced

1 Abstract

Learning a decision tree from data is a difficult optimization problem. The most widespread algorithm in practice, dating to the 1980s, is based on a greedy growth of the tree structure by recursively splitting nodes, and possibly pruning back the final tree. The parameters (decision function) of an internal node are approximately estimated by minimizing an impurity measure. We give an algorithm that, given an input tree (its structure and the parameter values at its nodes), produces a new tree with the same or smaller structure but new parameter values that provably lower or leave unchanged the misclassification error. This can be applied to both axis-aligned and oblique trees and our experiments show it consistently outperforms various other algorithms while being highly scalable to large datasets and trees. Further, the same algorithm can handle a sparsity penalty, so it can learn *sparse oblique trees*, having a structure that is a subset of the original tree and few nonzero parameters. This combines the best of axis-aligned and oblique trees: flexibility to model correlated data, low generalization error, fast inference and interpretable nodes that involve only a few features in their decision.

Work supported by NSF award IIS–1423515.

2 Decision trees

The prediction made by a decision tree is given by a path from the root to a leaf consisting of a sequence of decisions, each involving a question of the type "$x_j > b_i$" (is feature j bigger than threshold b_i?) for axis-aligned trees, or "$\mathbf{w}_i^T \mathbf{x} > b_i$" for oblique trees.
Decision trees enjoy several unique advantages:

- Among the most widely used statistical models in practice.
- Able to model nonlinear data.
- Very fast inference.
- May not need to use all input features to make a prediction.
- Interpretable: the path can be understood as a sequence of IF-THEN rules, which is intuitive to humans. We can equivalently turn the tree into a database of rules.

This can make decision trees preferable over more accurate models such as neural nets in some applications, such as medical diagnosis or legal analysis.

3 Traditional optimization of decision trees

Learning the tree from data is a very difficult optimization problem, involving a search over a complex, large set of tree structures, and over the parameters at each node. In practice, the most widespread algorithm for learning trees (such as CART or C4.5) consists of growing a tree by recursively splitting each node into two children. The decision parameters at each split are obtained by approximately maximizing the class purity of the split (e.g. via the Gini index or class distribution entropy).
This approach has two problems:

- optimizing the purity is still difficult (it is a non-differentiable function of the parameters)
- it does not optimize the desired loss (e.g. misclassification error).

As a result, this produces suboptimal trees, particularly with oblique trees. This has caused oblique trees to remain little used in practice.
We provide a new algorithm, TAO, that can learn much better trees (axis-aligned, oblique or otherwise).

4 Tree alternating optimization (TAO)

TAO seeks a local minimum of the misclassification error $\mathcal{L}$ (on the training set) over the parameters of all nodes i of the tree. TAO uses an initial tree (CART or random) with a fixed structure. Define the *reduced set of node* i (whether decision node or leaf) as the training instances that reach i under the current tree. Assume the loss $\mathcal{L}$ is additive over the training instances and that the tree parameters are not shared across nodes. TAO is based on two theorems:

Separability condition. If nodes i and j in the tree (whether decision nodes or leaves) are not descendants of each other, then $\mathcal{L}$ can be written as a separable function of the parameters of i and j. (This follows from the fact that the reduced sets of i and j are disjoint, because the tree makes hard decisions.)

Reduced problem. The problem of optimizing $\mathcal{L}$ over the parameters at a decision node is equivalent to a binary classification problem over the node's decision function on its reduced set, where each instance is labeled as the child (left or right) that leads to a lower value of $\mathcal{L}$ under the current tree. (This follows from the fact that all a decision node can do with an instance is send it down its left or right child, and the ideal choice is the one that results in the best prediction downstream from that node.)
Optimizing over a leaf is simple: just train its model on its reduced set to optimize $\mathcal{L}$.

This has the following consequences:

- The separability condition allows us to optimize separately (and in parallel) over the parameters of any set of nodes that are not descendants of each other (fixing the parameters of the remaining nodes). This has two advantages: 1) we expect a deeper decrease of the loss, because we optimize over a large set of parameters exactly; 2) the computation is fast: the joint problem over the set of nodes becomes a collection of smaller independent problems each over one node that can be solved in parallel. Typically, we cycle over depth levels from the bottom (leaves) to the top (root) and iterate bottom-top, bottom-top, etc.
- The reduced problem theorem implies that optimizing the tree loss over a decision node's parameters reduces to optimizing a binary classifier (defined by those parameters) with the 0/1 loss with certain labels. While optimizing the 0/1 loss in general is NP-hard, we can approximate it by a surrogate loss (e.g. logistic or hinge loss). Regularization terms over the tree parameters carry over to the reduced problem.
- Effectively, what TAO does is repeatedly train a simple binary classifier at each decision node and a simple predictor (K-class classifier) at each leaf while monotonically decreasing the loss $\mathcal{L}$. What changes over iterations is the reduced set on which each classifier or predictor is trained.
- Finally, as TAO iterates, the root-leaf path followed by each training instance changes and so does the reduced set at each node. This can cause *dead branches* (whose reduced set is empty) and *pure subtrees* (which contain instances from a single class). They can be pruned after convergence. This means that TAO can actually modify the tree structure, by reducing the size of the tree; this is very significant with sparsity penalties. We can handle model selection as usual, by cross-validation over the tree hyperparameters (depth and sparsity).

Convergence and computational complexity. Optimizing the misclassification loss $\mathcal{L}$ over a tree is NP-hard in general and we have no approximation guarantees for TAO at present. TAO does converge to a local optimum in the sense of alternating optimization (as in k-means), i.e., when no more progress can be made by optimizing one subset of nodes given the rest. For oblique trees, the complexity of one TAO iteration (pass over all nodes) is upper bounded by the tree depth times the cost of solving an SVM on the whole training set, and is typically quite smaller than that.

Pseudocode for the tree alternating optimization (TAO) algorithm. Visiting each node in reverse breadth-first search (BFS) order means scanning depths from depth(T) down to 1, and at each depth processing (in parallel) all nodes at that depth.

input training set and initial tree T (from CART, or random)
repeat
  for $i \in$ nodes of T, visited in reverse BFS
    if i is a leaf **then**
      train predictor model at i to optimize $\mathcal{L}$ on its reduced set
    else
      train decision model to solve the reduced problem at i
until stop
postprocess T: remove dead branches & pure subtrees
return T

5 Sparse oblique trees

In TAO, penalties or constraints on each node's parameters appear in the corresponding node's optimization only. This makes it computationally easy to introduce constraints over the weight vector and hence learn more flexible types of oblique trees than was possible before. Here we propose to learn oblique nodes involving few features. We can do this by adding an ℓ_1 penalty "$\lambda \sum_{\text{nodes } i} \|\mathbf{w}_i\|_1$" to the misclassification error $\mathcal{L}$ where $\lambda \geq 0$ controls the sparsity: from no sparsity for $\lambda = 0$ to all weight vectors in all nodes being zero if λ is large enough. The only change in TAO is that the optimization over node i has a penalty "$\lambda \|\mathbf{w}_i\|_1$". For linear nodes this results in an ℓ_1-regularized linear SVM or logistic regression, a convex problem for which well-developed code exists.

6 Sparse oblique trees for MNIST digits

TAO produces trees that significantly improve over the CART baseline for axis-aligned and especially oblique trees, and also consistently beat the other methods (see paper). Here we show results on MNIST. The figure shows an initial CART tree of depth 12 and two sparse oblique trees obtained by running TAO on the CART tree for $C = 0.09$ (optimal validation error) and $C = 0.01$ (an oversparse tree). Runtime was a few minutes per tree.

- Accuracy: TAO improves considerably over CART trees: from a training/test error of 1.95%/11.03% down to 0.09%/5.66%. The tree is pruned from 410 to 230 decision nodes. The TAO tree is more balanced: the samples are distributed more evenly over the tree branches and the training subset that a node receives is more pure.
- Inference time: extremely fast because the root-leaf path involves a handful of nodes and each node's decision function looks at a few pixels of the image. This is orders-of-magnitude faster than kernel machines, deep nets, random forests or nearest-neighbor classifiers.
- The TAO trees operate directly on the image pixels with no special transformation and are astoundingly small and fast given their accuracy. For example, our tree achieves about the same test error as a 3-nearest-neighbor classifier, but the tree compares the input image with at most ≈6 sparse "images" (weight vectors), rather than with the 60 000 dense training images.
- Interpretable trees: each decision node pays attention to a simple but high-level concept so the tree classifies digits by asking a few conceptual questions about the relative spatial presence of strokes or ink. A root-leaf path can then be seen as a sequence of conceptual questions that "define" a class. This is very different from the way convolutional neural networks operate.

The bottom plot shows test error vs. number of parameters for sparse oblique trees of different depths (color-coded), initialized from a CART tree that is either axis-aligned (dotted line) or oblique (solid line). The markers correspond to the initial CART trees ($\circ$, $+$) or to other models: 1) linear classifiers, 2) one-vs-all linear classifiers, 3) 2-layer neural net with 300 hidden units, 4) 2-layer neural net with 1 000 hidden units, 5) 3-nearest-neighbor classifier, 6) one-vs-all classifiers where each classifier consists of 50 000 boosted decision stumps (each operating over a feature and threshold), 7) 3-layer neural net with 500+100 hidden units. Values outside the axes limits are projected on the boundary of the plots.

7 Conclusion

Tree Alternating Optimization (TAO) is a scalable algorithm that can find a local optimum of oblique trees given a fixed structure, in the sense of repeatedly decreasing the misclassification loss until no more progress can be done. A critical difference with the standard tree induction algorithm is that we do not optimize a proxy measure (the impurity) greedily one node at a time, but the misclassification error itself, jointly and iteratively over all nodes.
TAO could make oblique trees widespread in practice and replace to some extent the considerably less flexible axis-aligned trees. Even more interesting are the sparse oblique trees we propose. These can strike a good compromise between flexible modeling of features (involving complex local correlations, as with image data) and using few features in each node, hence producing a relatively accurate tree that is very small, fast and interpretable. For MNIST, we believe this is the first time that a single decision tree achieves such high accuracy, comparable to that of much larger models.