Bivariate decision trees

Rasul Kairgeldin
Dept. CSE, University of California, Merced
rkairgeldin@ucmerced.edu

Miguel Á. Carreira-Perpiñán
Dept. CSE, University of California, Merced
mcarreira-perpinan@ucmerced.edu

Univariate decision trees Traditional decision trees, in widespread use since the 1950s, make a prediction by asking a tree-structured set of questions (decision nodes), each of the form "is $x_d > \theta$", where x_d is an input feature and θ a threshold. They are sometimes called *univariate* or *axis-aligned* trees because of using a single feature in each decision node. Because of the simple logic involved in making a prediction, this type of trees are generally considered among the most interpretable of all machine learning models, and are widely used in applications, particularly when interpretability is critical (such as credit scoring). However, this is only true if the tree is small enough. One critical problem with univariate decision trees is that they typically do not achieve a predictive accuracy (in classification or regression) that is competitive with other models. This is due to several factors. First, the univariate splits are ill-suited for modeling feature correlations, particularly with high-dimensional feature vectors. For example, linearly separable classes along an oblique hyperplane will result in a large univariate tree that tries to approximate the hyperplane with a zigzagging series of splits. Second, univariate trees can only use a small number of input features (in the whole tree, or in a single root-leaf path), because each decision node uses one feature, and the number of leaves must be smaller or equal than the sample size (usually much smaller). With high-dimensional data, this means the tree can only use a subset of the input features to make a prediction. Third, univariate trees are traditionally trained with heuristic, greedy recursive partitioning algorithms such as CART [3]. As CART grows the tree, it fixes the feature and threshold at each node by optimizing a local, purity criterion. However, this procedure does not optimize any loss function over the whole tree. The combination of these issues results in trees that are both suboptimal and large (deep and with many nodes), hence hard to interpret.

Multivariate decision trees On the other hand, multivariate (oblique) trees can use potentially all features in each decision node, thus capturing high-dimensional correlations well. Until recently, oblique trees were trained using CART, which (as with univariate trees) results is overly large yet not quite accurate trees. However, the recently introduced *Tree Alternating Optimization (TAO)* [4] algorithm can train oblique trees properly which do live up to their expectations: they are both much smaller and much more accurate than univariate trees. However, an oblique tree can still be hard to interpret because of using many features in each node.

Bivariate decision trees Here we consider bivariate trees, where each decision node can use (up to) two features. We think they occupy a sweet spot between univariate and fully multivariate splits which makes them practically attractive. While not as powerful as oblique trees, the ability to use pairs of features significantly helps with modeling correlations, and this translates into a higher accuracy and a much smaller tree size (in depth and number of nodes), which helps with interpretability. The only, minor drawback is the use of two features instead of one in most nodes. This results in trees that are both more accurate and more interpretable, as shown in our experiments. Bivariate trees have actually been considered before with CART [5, 1, 2], but the resulting trees are still large and inaccurate.

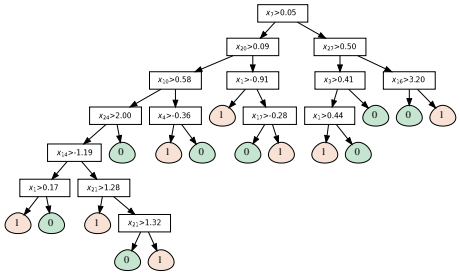
Scalable training of bivariate trees We adapt the TAO algorithm. Unlike CART-style algorithms, TAO takes an initial tree of fixed structure and iteratively updates its node parameters so that a well-defined objective function over the whole tree is monotonically reduced. The objective is a loss function (say, the 0/1 loss for classification) plus a regularization term. The latter is the sum of a

Preprint. Under review.

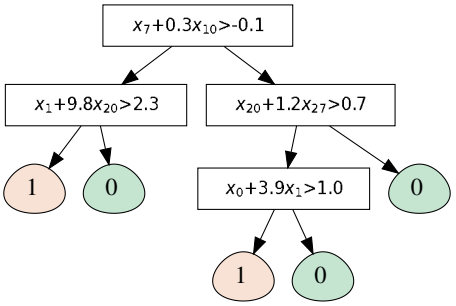
penalty per decision node, which equals C/1/0 if it uses 2/1/0 features, respectively. If a node uses 0 features, it becomes redundant and can be pruned at the end, thus shrinking the tree structure. The tradeoff between loss and tree size is controlled by a hyperparameter, which we usually set by cross-validation. TAO works by optimizing over each node given the others are fixed and iterating. Over a leaf this involves picking the majority class or the mean of the instances reaching the leaf. Over a decision node it involves a weighted 0/1 loss binary classification over a bivariate linear classifier. While NP-hard, we can approximate this by considering a small, fixed subset of hyperplane inclinations for every pair of features. The algorithm is slower than CART but still scales well to large datasets. A separability condition holds so that non-descendant nodes can be optimized in parallel. We think our algorithm makes bivariate trees practical and scalable and hope they will find a permanent place in the data analyst's toolbox.

Experiments In our experiments, we use as initial tree a univariate CART tree (fully grown without postpruning). We show the univariate (CART postpruned) and bivariate trees and the corresponding rules resulting for the Breast Cancer dataset ($N = 559$ instances of dimension $D = 9$ in 2 classes). It is clear the bivariate tree is much more interpretable and accurate. We also show a plot of the error as a function of tree size for the Breast Cancer dataset, and a table with test error and tree size for other datasets; the training time was at most a couple of minutes per dataset.

univariate tree
$\Delta = 7$, #leaves=16, $E_{\text{test}} = 6.4\%$

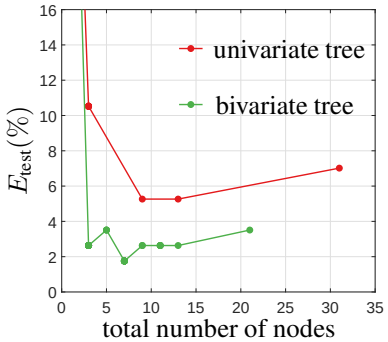


bivariate tree
$\Delta = 3$, #leaves=5, $E_{\text{test}} = 2.6\%$



1: **if** $x_7 > 0.05$ & $x_{20} > 0.09$ & $x_{10} > 0.58$ & $x_{24} > 2.00$ & $x_{14} > -1.19$ & $x_1 > 0.17$ **then** PREDICT 1 **end if**
2: **if** $x_7 > 0.05$ & $x_{20} > 0.09$ & $x_{10} > 0.58$ & $x_{24} > 2.00$ & $x_{14} > -1.19$ & $x_1 \leq 0.17$ **then** PREDICT 0 **end if**
3: **if** $x_7 \leq 0.05$ & $x_{27} \leq 0.50$ & $x_{16} \leq 3.20$ **then** PREDICT 1 **end if**
4: **if** $x_7 \leq 0.05$ & $x_{27} \leq 0.50$ & $x_{16} > 3.20$ **then** PREDICT 0 **end if**
5: **else**
6: ...**//12 MORE RULES**...

1: **if** $x_7 + 0.3x_{10} > -0.1$ & $x_1 + 9.8x_{20} > 2.3$ **then** PREDICT 1 **end if**
2: **if** $x_7 + 0.3x_{10} > -0.1$ & $x_1 + 9.8x_{20} \leq 2.3$ **then** PREDICT 0 **end if**
3: **if** $x_7 + 0.3x_{10} \leq -0.1$ & $x_{20} + 1.2x_{27} \leq 0.7$ **then** PREDICT 0 **end if**
4: **if** $x_7 + 0.3x_{10} \leq -0.1$ & $x_{20} + 1.2x_{27} > 0.7$ & $x_0 + 3.9x_1 > 1.0$ **then** PREDICT 1 **end if**
5: **else**
6: PREDICT 0



Dataset		bivariate tree	univariate tree
Breast Cancer	training (%)	99.12±0.01	99.26±1.22
	test (%)	97.37±0.12	93.64±2.80
	Δ/#nodes	3.1/9.5	7.0/31.0
Segment	training (%)	99.03±0.30	99.34±0.58
	test (%)	97.12±1.10	94.17±3.85
	Δ/#nodes	12.1/45.3	15.0/127.8
Optical recog.	training (%)	97.17±0.05	99.85±0.12
	test (%)	87.65±0.11	85.45±0.11
	Δ/#nodes	11.0/123.0	15.7/493.0
Pageblock	training (%)	98.24±0.01	99.74±0.01
	test (%)	97.81±0.10	96.52±0.17
	Δ/#nodes	5.0/23.0	18/268.2

Acknowledgments: Work funded in part by NSF award IIS–2007147.

References

[1] J. C. Bioch, O. van der Meer, and R. Potharst. Bivariate decision trees. In J. Komorowski and J. Zytkow, editors, *Proc. European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD 1997)*, pages 232–242, 1997.

[2] F. Bollwein and S. Westphal. A branch & bound algorithm to determine optimal bivariate splits for oblique decision tree induction. *Applied Intelligence*, 51(10):7552–7572, Oct. 2021.

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

[4] M. Á. Carreira-Perpiñán and P. Tavallali. Alternating optimization of decision trees, with application to learning sparse oblique trees. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NEURIPS)*, volume 31, pages 1211–1221. MIT Press, Cambridge, MA, 2018.

[5] D. Lubinsky. Classification trees with bivariate splits. *Applied Intelligence*, 4(3):283–296, July 1994.