Pros and cons of soft vs hard decision trees

Kuat Gazizov[1] and **Arman Zharmagambetov**[2] and
Miguel Á. Carreira-Perpiñán[1], [1]EECS, UC Merced, [2] FAIR at Meta

BayLearn
Bay Area Machine Learning Symposium

1 Introduction

Decision trees come in two basic types: soft (SDTs) and hard (HDTs). An input instance follows each root-leaf path with a positive probability in SDTs, and exactly one root-leaf path in HDTs (because each decision node picks only one child). We focus on oblique trees (having hyperplane decision nodes) for classification. Effective training of oblique trees can be done by gradient descent for SDTs and, since recently, by alternating optimization for HDTs. This makes it possible to perform an objective comparison between both models along multiple dimensions, including: the accuracy as a function of tree size and inference time; the scalability to training on large datasets; the effect of singularities of SDTs; the impact of hardening a SDT into a HDT; the extent to which experts specialize and its effect on tree interpretability; and the benefit of using sparsity in the weight vectors and of pruning. We conclude that HDTs are generally preferable to SDTs in most use cases.

Work partially supported by NSF award IIS–2007147.

2 Training trees

Soft tree. Each decision node i assigns a probability to its right child of $\sigma(\mathbf{w}_i^T\mathbf{x}+w_{i0})$, The output is computed as follows: $\mathbf{S}\left(\sum_{j\in\text{leaves}} p_j(\mathbf{x};\mathbf{W})\,\mathbf{q}_j\right)$ where $p_j(\mathbf{x};\mathbf{W})$ is a leaf probability. We use a cross-entropy loss to optimize the tree.

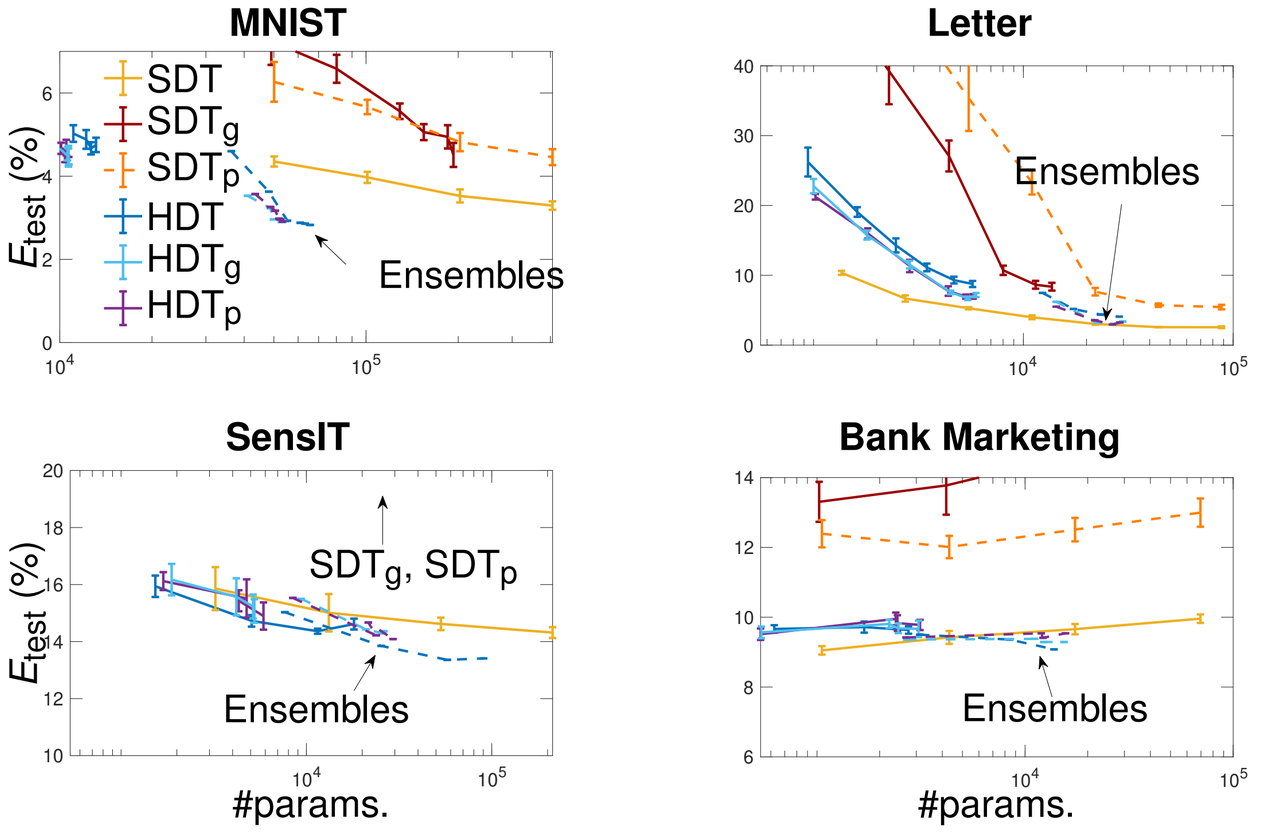
Hard tree. We use the TAO (Tree Alternation Optimization) algorithm to train a hard decision tree.

$$E(\mathbf{W},\mathbf{Q}) = \sum_{n=1}^{N} L(y_n, T(\mathbf{x}_n;\mathbf{W},\mathbf{Q})) + \lambda \sum_{i\in\text{nodes}} \phi_i(\mathbf{w}_i) \qquad (1)$$

where L is a loss function (here, the 0/1 loss) and ϕ_i a regularization on the decision weights (here, ℓ_1)

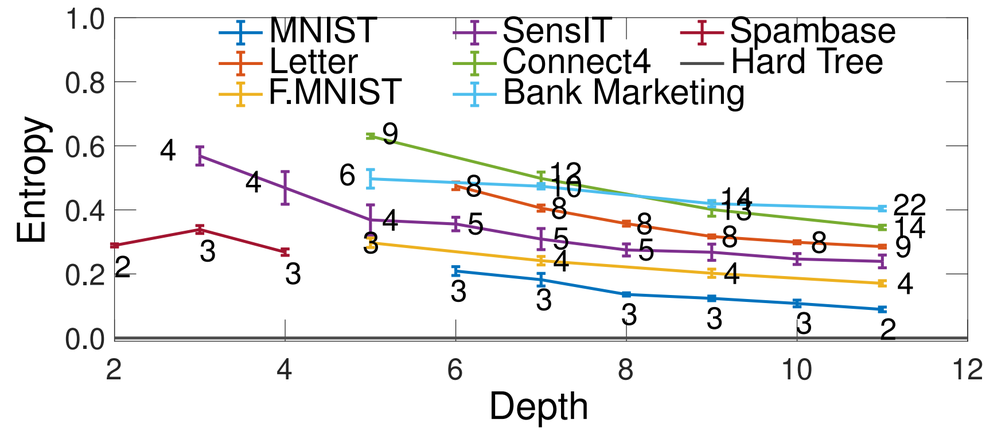
3 Predictive accuracy

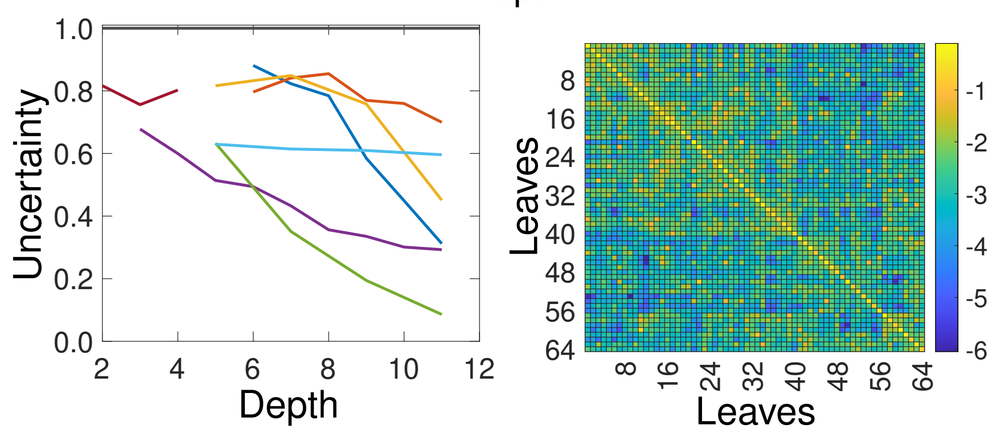
For the exact same tree structure, a SDT has a strictly higher representation power than a HDT. But, depending on the dataset, this need not result in a more accurate tree, particularly if one accounts for inference time and number of parameters.



4 Do the experts specialize in a SDT?

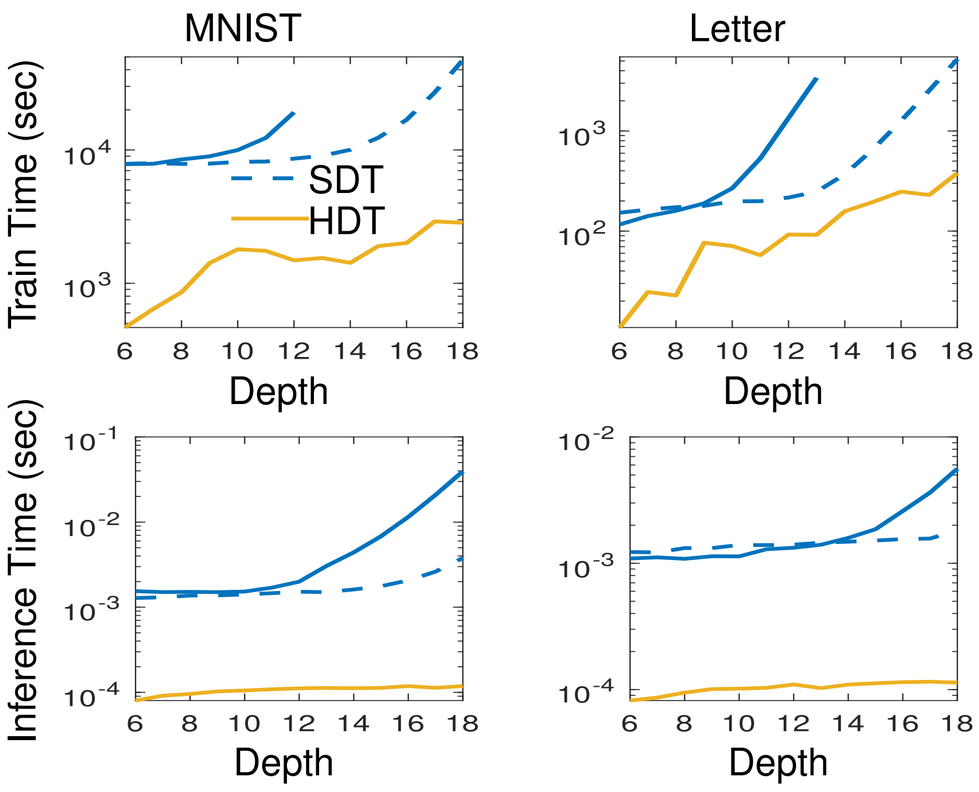
Empirically, the leaves ("experts") in a SDT overlap heavily, rather than specializing narrowly, and an input instance reaches multiple leaves with significant probability. This makes interpreting SDTs very hard.





5 Training and inference time

For a complete tree of depth Δ, training a HDT with TAO is linear on Δ while training a SDT with SGD is exponential on Δ. The same holds for inference time. HDTs are thus scalable to large datasets, while SDTs cannot be very deep.



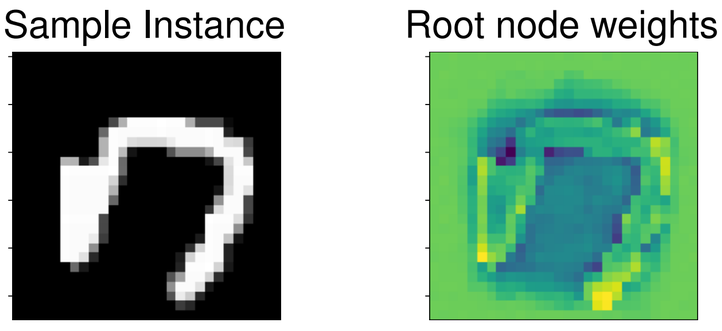
6 Hardening a SDT into a HDT

Hardening a SDT into a HDT (as sometimes done to make it faster and interpretable) significantly decreases its accuracy and is much worse than training a HDT with TAO.

We consider the following hardening techniques: *Greedy Hardening* (SDT_g), *Path-Based Hardening* (SDT_p), *Weight-Based Hardening* (HDT_g), *Sample-Leaf Probability Hardening* (HDT_p).

7 Singularities of SDTs

The SDT cross-entropy has singularities corresponding to hard splits (in the limit). These typically result in portions of the SDT being wasted and, being intrinsic to the loss function, are hard to avoid during the optimization.



8 Sparsifying and pruning

Sparsifying the weight vectors via an ℓ_1 penalty results in fewer nonzero parameters and a smaller tree structure. Specifically, when a decision node's weights $\mathbf{w}_i = \mathbf{0}$, it makes this node redundant. However, in SDTs, this node continues to send each instance to both children, so it cannot be pruned.