Alternating Decision Forests

Yufeng Jiang

1. Training Alternating Decision Forests

In order to incorporate any diffenentiable, global loss function into Random Forests, authors adopt the idea of the above reviewed Gradient Boosting method, *i.e.* they perfoem Gradient Descent in function space. While they could easily train Boosted Trees that respect a global loss function, *i.e.* Gradient Boosting having decision trees as weak learners, the goal of Althernating Decision Forests is to explicitly integrate the global loss into the tree growing scheme, thus preserving the parallel training of standard RFs. For that purpose, they need (i) a stage-wise tree growing scheme during which they update the weight distribution and (ii) spliting functions taking these weights into account.

To get a $stage-wise\ classifier$, they let their trees grow in an iterative, breadth-first manner, contrary to a typical depthfirst scheme in standard RFs. Each stage in ADFs corresponds to a single depth of the forest, *i.e.* the iterations are now indexed with $d = 1, \ldots, D_{max}$, where D_{max} is the maximum tree depth of the forest. After training each single iteration d, they have a classification model $\mathcal{T}^d(\mathbf{x}_i) = \frac{1}{T}\sum_{t=1}^{T} p_t^d(y_i|\mathbf{x}_i)$, where $p_t^d(y_i|\mathbf{x}_i)$ is the class probability estimate of sample $\mathbf{x}_i$ returned by tree $\mathcal{T}_t^d$, which denotes tree $\mathcal{T}_t$ grown up to depth d. They can now use this strong clasifier and a given loss $l(\cdot)$ to update the weights w_i^{d+1} of all training samples for the next iteration as illustrated in Eq.1. In the first iteration, all weights are set uniformly. After the weight updates, they train the next stage of the forest in parallel. This is done by convering all leaf nodes in the current iteration to split nodes.

To let the $splitting\ functions\ consider\ the\ sample\ weights$, they change the standard entropy claculation to become a weighted entropy by changing the estimation of the class distributions $p(k|S)$ for the set S:

$$p(k|S) = \frac{\sum_{i=1}^{|S|}[y_i = k] \cdot w_i^d}{\sum_{i=1}^{|S|} w_i^d}. \qquad (1)$$

Here, $[y_i = k]$ is the indicator function returning 1 if the label y_i of the sample $\mathbf{x}_i \in S$ is equal to k, and 0 otherwise.

2. Comparison of ADFs with Other Classifiers

First, authors directly compare all methods against each other on all 5 data sets. As the common parameters of ADFs, RFs and BTs are set equally, as mentioned before, they directly compare the way the tree structure is built. For ADFs and BTs they also evaluate 5 different loss functiions that can be integrated in the gradient Boosting formulation: Logit, Hinge, Exponential, Savage [3] and Tangent [2].

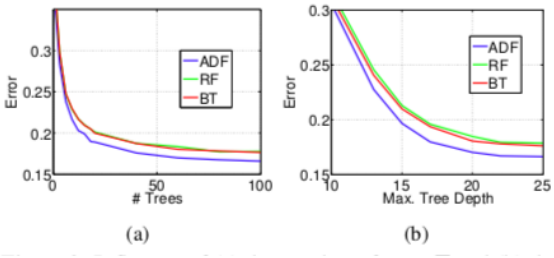

Figure 1. Influence of (a) the number of trees T and (b) the maximum tree depth D_{max} on the three classifiers ADFs, RFs, and BTs on the $Char74k$ data set.

Tab. 1 depicts their results. As can be seen, the combination of ADFs and the Tangent loss function yields the best results on all 5 data sets. ADFs with Savage loss and Exponential loss are the second best choices on 3, respectively 1 data sets. Only for $G50c$, BTs with Savage loss gives the second best results. Most interesting, however, is the fact that ADFs (Tangent loss) constantly outperform both RFs and BTs, *i.e.* the main competitors.

They furhter evaluate the computational costs of training ADFs and give a relative comparison to RFs and BTs in Tab. 2. As expected, BTs are much slower (around 7 times slower), while ADFs and RFs show similar training time.

3. Comparison of Common Parameters

Next, they evaluate the influene of two important parameters of all evaluated classifiers on the $Char74k$ data set. They investigate the number of trees T and the maximum tree depth D_{max}, which they vary in the ranges [1,100] and [10,25], respevtively. As mentioned before, they choose the Tangent loss for both ADFs and BTs due to the good overall performance in the last experiment.

It is interesting that ADFs improve over RFs and BTs only for larger number of trees, just as shown at Fig. 1. For

Method	Loss	G50c	Letter	USPS	MINIST	Char74k
Alternating	Logit	19.83±1.34	6.81±0.34	6.31±0.23	3.82±0.08	17.38±0.16
Decision	Hinge	19.47±1.26	4.89±0.16	5.87±0.19	3.20±0.06	17.00±0.10
Forests	Exp	19.09±1.17	4.27±0.13	6.03±0.29	2.96±0.05	16.82±0.15
	Savage [3]	19.00±1.32	3.94±0.14	5.76±0.16	2.78±0.09	16.92±0.15
	Tangent [2]	**18.71±1.27**	**3.52±0.12**	**5.59±0.16**	**2.71±0.10**	**16.67±0.21**
Boosted Trees [1]	Logit	18.99±1.51	4.98±0.09	5.99±0.21	3.18±0.08	17.98±0.19
	Hinge	18.97±1.33	4.87±0.15	5.90±0.15	3.23±0.05	17.65±0.19
	Exp	18.91±1.30	4.78±0.12	5.83±0.19	3.17±0.07	17.57±0.10
	Savage [3]	18.87±1.31	4.65±0.12	5.92±0.19	3.19±0.07	17.62±0.25
	Tangent [2]	18.90±1.31	4.70±0.18	5.93±0.27	3.15±0.05	17.59±0.29
Random Forests	-	18.91±1.33	4.75±0.10	5.96±0.21	3.21±0.07	17.76±0.13

Table 1. Alternating Decision Forests compared with the two main competitors, Random Forests and Boosted Trees, on 5 data sets. Best performing methods are highlighted and marked **bold-face**, second best methods are highlighted only.

Method	G50c	Letter	USPS	MINIST	Char74k
ADF	1.0	1.0	1.0	1.0	1.0
BT	3.99	6.55	7.32	7.05	7.09
RF	1.55	0.45	0.70	0.79	1.52

Table 2. Training time comparison between the three main competitors on all 5 data sets, relative to ADFs.

a small number of trees, the performance is more or less the same. This could be explained by the weight update in ADF, which follow the predictions of the current state of the classifier. However, due to the small number of trees and the fact that the trees are not fully grown yet, these predictions are unreliable. This might decrease the performance as the weights take unreasonable values. However, as soon as the number of trees increases, also the performance of ADFs increases and outperforms both RFs and BTs.

References

[1] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2009. 2

[2] H. Masnadi-Shirazi, V. Mahadevan, and N. Vasconcelos. On the design of robust classifiers for computer vision. In *CVPR*, 2010. 1, 2

[3] F. Moosmann, B. Triggs, and F. Jurie. Fast discriminative visual codebooks using randomized clustering forests. In *NIPS*, 2006. 1, 2