To Bag is to Prune

Philippe Goulet Coulombe

gouletc@sas.upenn.edu

University of Pennsylvania

Saint-Zénon, May 7, 2021

Random Forest (RF) Crash Course

What is a tree?

RF is a diversified ensemble of regression trees. What is a tree?

- Let π_t be inflation at time t.
- t^* is inflation targeting implementation date.
- Let g_t be some measure of output gap.

$$\begin{tikzcd}
 & & \text{Full Sample} \arrow[dl, dash] \arrow[dr, dash] & \\
 & t < t^* \arrow[dl, dash] \arrow[dr, dash] & & t \geq t^* \arrow[d, dash] \\
g_{t-1} < 0 \arrow[d, dash] & & g_{t-1} \geq 0 \arrow[d, dash] & \pi_t = 2 + \epsilon_t \\
\pi_t = 1 + \epsilon_t & & \pi_t = 4 + \epsilon_t &
\end{tikzcd}$$

Estimating a tree

$$y_i = \mathcal{T}(X_i) + \epsilon_i$$

- A regression tree is an algorithm that recursively partitions the data until some stopping criterion is met. A *greedy* algorithm is used:

$$\min_{k \in \mathcal{K}, c \in \mathbb{R}} \left[\min_{\mu_1} \sum_{\{i \in L | X_i^k \leq c\}} (y_i - \mu_1)^2 + \min_{\mu_2} \sum_{\{i \in L | X_i^k > c\}} (y_i - \mu_2)^2 \right] \quad (1)$$

- The prediction for j is the average of y_i for all i that are members of the same "leaf" as j.
- A single tree typically has low bias and very high variance.
- There exists ways to decrease tree's variance by "pruning", which means stopping the greedy algorithm "early".

RF Crash Course

3 ingredients to go from a single tree to a forest

For each tree:

1. **Let the trees run deep**: even though that would surely imply overfitting for a single tree, let each tree run until leafs contain very few observations (usually < 5).

Diversifying the Portfolio (i.e., creating the ensemble)

2. **Bagging**: Create B nonparametric bootstrap samples of the data. That is, we are picking $[y_i \;\; X_i]$ pairs with replacement.

3. **Perturbation**: At each splitting point, we only consider a subset of all predictors ($\mathcal{J}^- \subset \mathcal{J}$) for the split.

RF prediction is the simple average of all the B tree predictions.

RF Crash Course

Why do we like it

- It works tremendously well on all sorts of data, even macro data (Chen et al., 2019; Goulet Coulombe et al., 2019; Medeiros et al., 2019; Goulet Coulombe, 2020; Goulet Coulombe et al., 2020).
- More often than not, it's better than Neural Networks – which require careful tuning.
- Can approximate a wide range of nonlinearities
- Tuning parameters do not alter prediction much
- Can easily deal with a very large X (no matrix operation involved)
- **Most importantly, it does not seem to overfit. How can that be?**

The R^2_{test} vs R^2_{train} Puzzle

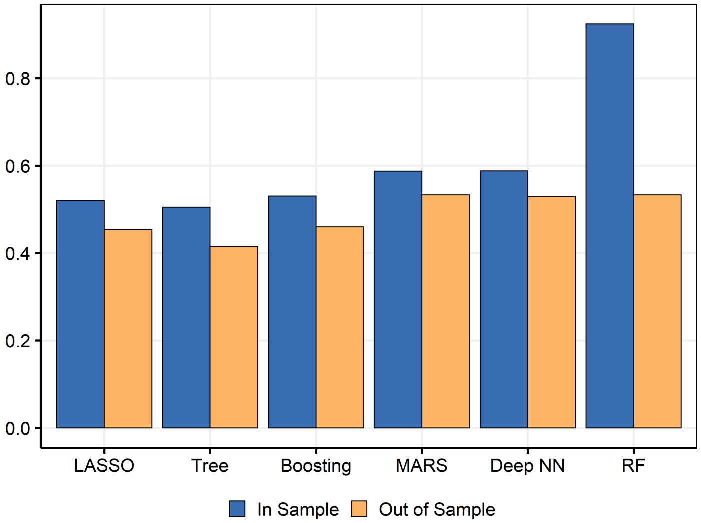


Figure: *Abalone* data set example

Usual explanations for RF's success don't explain it

- (Breiman, 2001) originally derived an upper bound on the generalization error of RF — it decreases as $\mathcal{T}$ strength increases, and increases as correlation between them increases.
  - $\Rightarrow$ Nice to have, but it does not say much about results obtained in practice.
- (Bühlmann et al., 2002): bagging brings smoothness (hence regularization)
  - $\Rightarrow$ If that was just that, then $R^2_{\text{test}} \approx R^2_{\text{train}}$ like for any usual smoothing method
- (Mentch and Zhou, 2019) (and ESL): randomization implies a ridge-like regularization obtained by model averaging – an adequate argument for *global linear* models (reminiscent of (Elliott et al., 2013)'s CSR)
  - $\Rightarrow$ If that was just that, then $R^2_{\text{test}} \approx R^2_{\text{train}}$ like for Ridge
- (Belkin et al., 2019) claims RF has a "double-descent" risk curve, like Neural Nets.
  - $\Rightarrow$ Their construction confused additional trees with additional complexity, which is true for Boosted Trees, but not RF. In fact, RF has a single, never-ascending, descent.

Roadmap

- Why the puzzle occurs and what it tells us about RF's legendary robustness to overfitting
    1. What happens in the overfitting zone stays in the overfitting zone
    2. Bagging + Perturbation (`B` & `P`) as an approximation to population sampling (and a *Perfectly* Random Forest)
- Those ideas should apply to any *randomized greedy algorithm* $\rightarrow$ leverage those to develop two new "self-tuning" algorithms
    1. Booging
    2. MARSquake

——————————- **If time allows** ——————————-

- Why RF implicit pruning is better than pruning CART directly
    1. Insights from nonlinear time series models: it prunes the true latent $\mathcal{T}$.
    2. **Extra**: Slow-Growing Trees

Greed is Good

The Key: *A greedy algorithm treats what has already happened as given and what comes next as if it will never happen.*

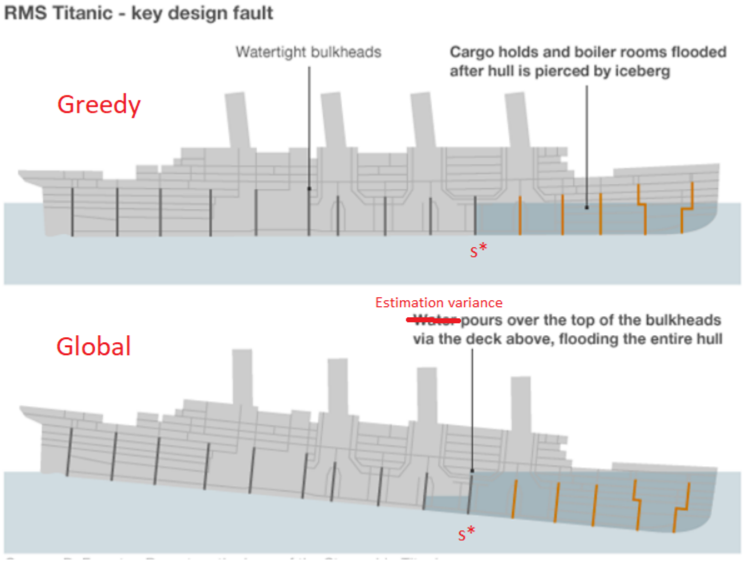
- **Old song**: greedy optimization is an inevitable (but suboptimal) practical approach in the face of computational adversity (see ESL) — bad because no guarantee to get the "optimal" tree.
- **New song**: by building recursively a model of increasing complexity (when true complexity s^* is unknown) in a stepwise fashion, what is estimated in early steps is immune to the "pollution" brought by the latter steps (which are likely overfitting).

What happens past s^* stays past s^*

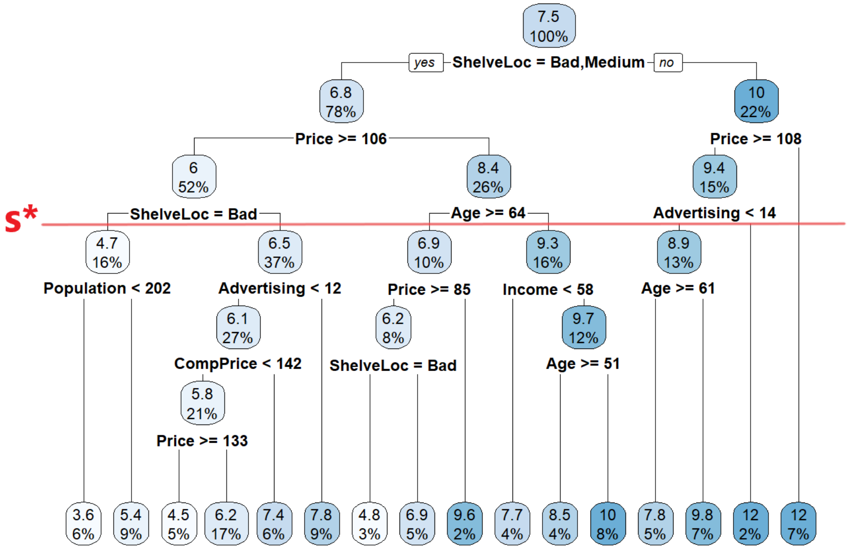
$$\hat{y}_i = \underbrace{\underbrace{\beta_1 x_{1,i}}_{s=1} + \underbrace{\beta_2 x_{2,i}}_{s=2} + \underbrace{\beta_3 x_{3,i}}_{s=3}}_{OLS}$$

- **Global** optimization (think OLS): overfitting weakens the whole prediction function
  - $\Rightarrow$ estimating many useless coefficients inflates the generalization error by increasing the variance of *both* the useful coefficients and the useless ones.
- **Greedy** optimization (think tree, or boosting): the function estimated before s is *treated as given*.
  - $\Rightarrow$ the algorithm eventually reach s^* where the only thing left to fit is the unshrinkable "true" error $\epsilon_i = y_i - \hat{f}_{s^*}(x_i)$, i.e., overfitting.
  - $\Rightarrow$ But this does not alter $\hat{f}_{s-1}$ since it is not re-evaluated. Only useless stuff is added on "top" of it.
  - $\Rightarrow$ More concretely, $\hat{\beta}$'s estimated or tree splits estimated before s^* cannot be revoked, and the predictive structure attached to them cannot weaken by ulterior steps.

RMS Titanic, Compartments, and ML Algorithms


RMS Titanic - key design fault
Watertight bulkheads
Cargo holds and boiler rooms flooded after hull is pierced by iceberg
Greedy
s*
Estimation variance
~~Water~~ pours over the top of the bulkheads via the deck above, flooding the entire hull
Global
s*

A Less Maritime Example



What is happening beyond s^*?

- At s^*, the unknown point of optimal early stopping (aka the *true* terminal node in the case of a tree), the DGP is

$$y_i = \mu + \epsilon_i. \tag{2}$$

  and the best prediction is clearly the sample average. And yet, the algorithm continues to fit beyond s^*.

- Two questions:
  1. What is the prediction of a "perfectly random forest"? That is, one where we replaced `B` & `P` by population sampling – fitting fully overfitted greedy trees on *non-overlapping* samples of the same DGP?
  2. Can `B` & `P` provide a good approximation to the ideal PRF when applied to trees? (This is an empirical matter.)

The Perks of a *Perfectly* Random Forest

- We are looking at the prediction for a new data point j using f trained on observations $i \neq j$.
- Assume fully grown trees – terminal nodes include one observation.
- Since the tree is fitting noise, each out-of-sample tree prediction is a randomly chosen y_i for each b.
- Define $r = {}^{B}/_{N}$ where N is the number of training observations and r will eventually stand for "replicas".
- Since the $y_{i(b)}$'s amount to random draws of $y_{1:N}$, for a large enough B, we know with certainty that the vector to be averaged will contain r times the same observation y_i.
- Remembering that $r = {}^{B}/_{N}$, the prediction is

$$\hat{\mu}_j^{\text{RF}} = \frac{1}{B}\sum_{b=1}^{B} y_{i(b)} = \frac{1}{B}\sum_{i=1}^{N}\sum_{r'=1}^{r} y_{i,r'} = \frac{1}{B}\sum_{i=1}^{N}\sum_{r'=1}^{r} y_i = \frac{r}{B}\sum_{i=1}^{N} y_i = \frac{1}{N}\sum_{i=1}^{N} y_i$$

- When a PRF is starting to fit pure noise, its out-of-sample prediction collapses to $\bar{y}$, which is *optimal*.

B & P as an Approximation to Population Sampling

- Intuitively, at s^*, the test set behavior is identical to that of doing (random) subsampling with subsamples containing one observation.
- Averaging the results of the latter (over a large B) is just a complicated way to compute an ***average*** — equivalent to stopping at s^*.

Let's recapitulate

- For the prediction function to be close to optimal without tuning it, we needed stuff past s^* to *efficiently* averages out to 0 in *the hold-out sample*.
- We also needed the estimated function before s^* to be protected against what comes next. We have both.
  - ⇒ Immediate implication: there is no need to find s^* through cross-validation to obtain optimal predictions.
- Simulations will ask "How close to population sampling are we when fitting B & P trees?" and the answer will be "surprisingly close".

New Kids on the Block: *Booging* and *MARSquake*

- The key ingredients for an ensemble to completely overfit in-sample while maintaining a stellar generalization error are
  - (i) the base learner prediction function is obtained by greedy/recursive optimization and
  - (ii) enough randomization in the fitting process.
- (i) means `B` & `P` variants of Boosted Trees and MARS are eligible for self-pruning.
- (ii) means its success depends on the capacity of the algorithm for randomization
- Tree constructions are completely irrevocable, additive structure are partly revocable (making (i) and (ii) maybe not as applicable as for RF)
- Let the data decide.

Simulations — Results

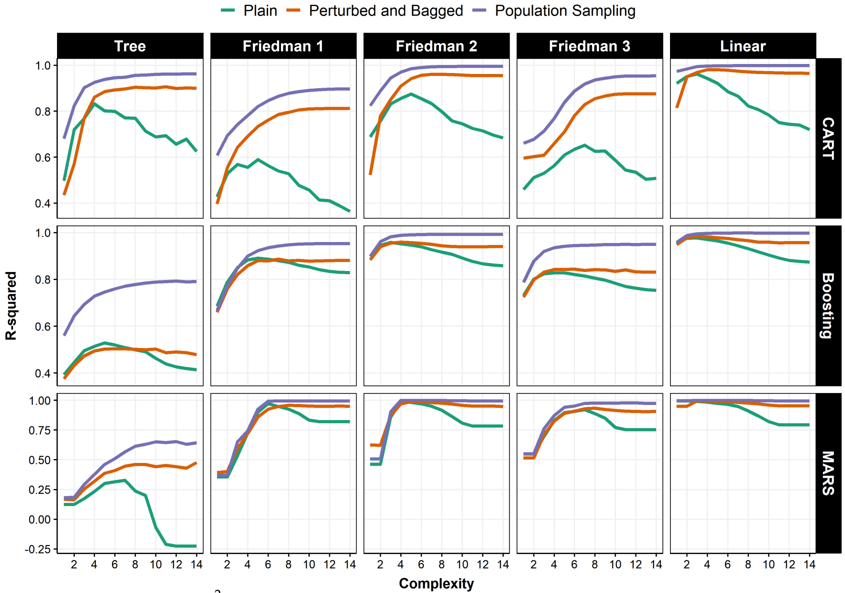
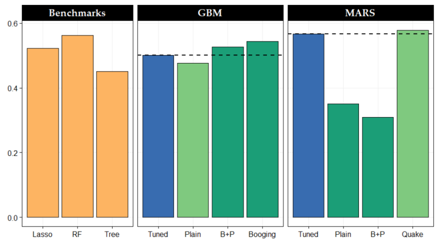
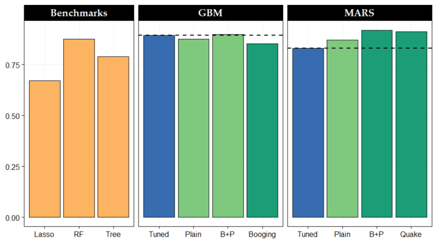


Figure: This plots hold-out sample R^2's between the prediction **and the true conditional mean**. The level of noise is calibrated so the signal-to-noise ratio is 4. Column facets are DGPs and row facets are base learners. The *x*-axis is an index of depth. For CART, it is a decreasing minimal size node $\in 1.4^{\{16,..,2\}}$, for Boosting, an increasing number of steps $\in 1.5^{\{4,..,18\}}$ and for MARS, it is an increasing number of included terms $\in 1.4^{\{2,..,16\}}$.

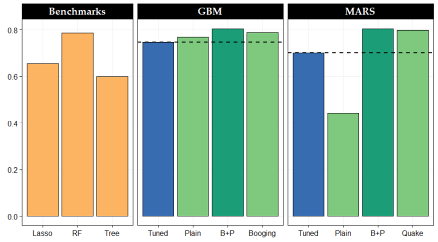
Real Data Results (1)



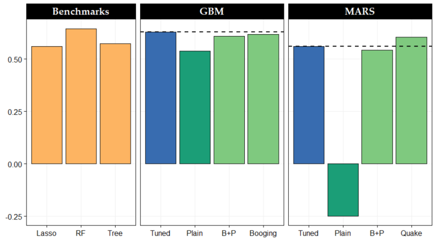
(a) *Abalone*



(b) *Boston Housing*



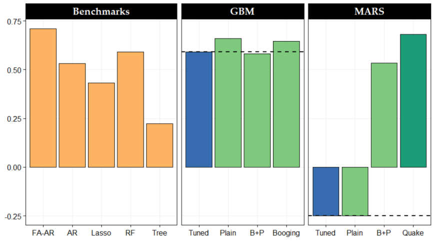
(c) *Crime Florida*



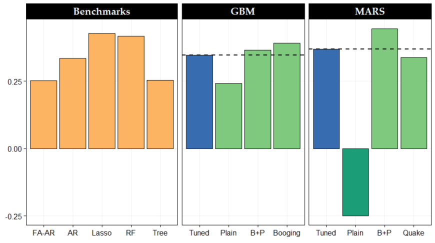
(d) *Fish Toxicity*

Figure: Performance metric is R^2_{test}. Darker green bars means the performance differential between the tuned version and the three others is significant at the 5% level.

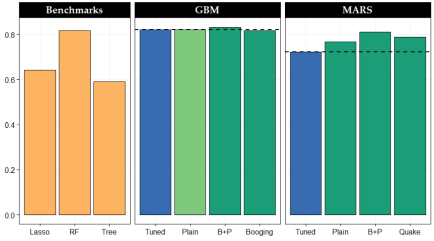
Real Data Results (2)



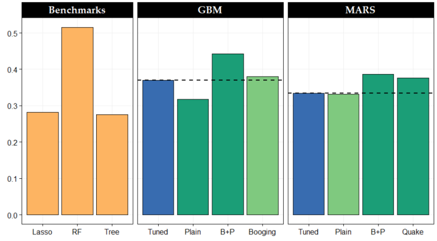
(e) *US Unemployment Rate* $(h = 1)$



(f) *US Inflation* $(h = 1)$



(g) *California Housing*



(h) *White Wine*

Figure: Performance metric is R^2_{test}. Darker green bars means the performance differential between the tuned version and the three others is significant at the 5% level.

Conclusion

1. `B` & `P` as implemented by RF automatically *prune* a (latent) true underlying tree.
2. This gives rise to the R^2_{test} vs R^2_{train} puzzle, which traditional explanations do not account for
3. More generally, there is no need to tune the stopping point of a properly randomized ensemble of greedily optimized base learners.
4. Boosting and MARS are also eligible for automatic (implicit) tuning.

Not discussed, but of interest:

- Why pruning CART $\neq$ RF: because RF "simulates" the true $\mathcal{T}$ through Bagging (intuition bases on nonlinear time series forecasting)
- Stabilizing the greedy algorithm can also be done with slow-learning (the traditional boosting way) and is developed in (Goulet Coulombe, 2021) where a single "Slow-Growing" Tree can match RF.

Appendix

Simulation Results with more noise

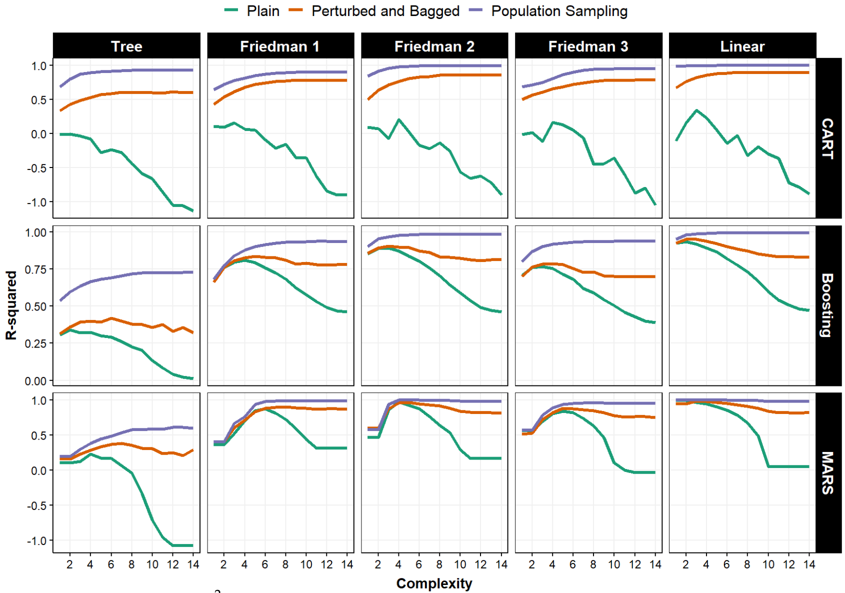


Figure: This plots hold-out sample R^2's between the prediction **and the true conditional mean**. The level of noise is calibrated so the signal-to-noise ratio is 1. Column facets are DGPs and row facets are base learners. The x-axis is an index of depth. For CART, it is a decreasing minimal size node $\in 1.4^{\{16,..,2\}}$, for Boosting, an increasing number of steps $\in 1.5^{\{4,..,18\}}$ and for MARS, it is an increasing number of included terms $\in 1.4^{\{2,..,16\}}$.

Why is RF typically much better than pruned CART?

Some insights from nonlinear time series forecasting

- It's been known for a while that RF (or bagged trees) performs orders of magnitude better than a single pruned true (Breiman, 1996).
- RF "pruning via inner randomization" is applied on the true *latent* tree $\mathcal{T}$ which itself can only be constructed from randomization — the greedy fitting procedure itself that generates the need for Bagging.
- The inspiration for the following argument comes from forecasting with nonlinear time series models. An illustrative SETAR DGP is

$$y_{t+1} = \eta_t \phi_1 y_t + (1-\eta_t)\phi_2 y_t + \epsilon_t, \quad \eta_t = I(y_t > 0) \tag{3}$$

- Forecasts are obtained y_{t+h} by iterating forward starting from t.
- From $h > 1$ on, only an estimate $\hat{y}_{t+1} = E(y_{t+1}|y_t)$ is available. By construction, $E(\hat{y}_{t+1}) = y_{t+1}$. However, by properties of expectations, $E(f(\hat{y}_{t+1})) \neq f(y_{t+1})$ if f is non-linear.
- Iterating forward using $\hat{y}_{t+h}$'s as substitutes for y_{t+h} leads to bias.

Why is RF typically much better than pruned CART?

Back to the tree algo

- To get the next finer subset that includes i, the "cutting" operator is applied to the latest available subset $\boldsymbol{S}' = \mathcal{C}(\boldsymbol{S}; y, X, i)$.
- The prediction for i can be obtained by using $\mathcal{C}$ recursively starting from $\boldsymbol{S}_0$ (the full data set) and taking the mean in the final $\boldsymbol{S}$.
- As such, the true latent tree is $\mathcal{T}(X_i) = E\left(y_{i'} | i' \in \mathcal{C}^D(\boldsymbol{S}_0; y, X, i)\right)$ where D is the number of times the cutting operator must be applied to obtain the final subset in which i resides.
- **But** using $\hat{y}_{t+1}$ *in situ* of y_{t+1} in SETAR and $\hat{\boldsymbol{S}}$ *in situ* of $\boldsymbol{S}$ in a tree generate problems of the same nature.
- The direct CART procedure produces an unreliable estimate of $\mathcal{T}$ because it takes as given at each step something that is not given, but estimated. Since $\mathcal{C}$ is a non-linear operator, this implies that the mean itself is not exempted from bias.
- Like in the case of forecasting SETARs, simulating the expectation numerically via bootstrapping circumvents the problem. In the context of a tree, this has a different name: **Bagging**.

Slow-Growing Trees

Filling the Missing Corner

Table: A Tree Ensemble Quaternity

		Model Structure	
		Additive Shallow Trees	One Deep Tree
Regularizer	Slow Learning	Boosting	**Slow-Growing Tree**
	`B` & `P`	Booging	Random Forest

Implementation

Consider the deceptively simple tree below, which is obtained after one recursion of CART.

$$\begin{tikzcd}[column sep=small] & \text{Full Sample} \arrow[dl, dash] \arrow[dr, dash] & \\ A \equiv \{i | x_i < 0\} & & B \equiv \{i | x_i \geq 0\} \end{tikzcd}$$

The subsequent problem of finding k_A^* and c_A^* to further grow the tree on the A side is

$$\min_{k \in \mathcal{K}, c \in \mathbb{R}} \left[\min_{\mu_1} \sum_{\{i | X_i^k \leq c\}} \omega_i^A \left(y_i - \mu_1\right)^2 + \min_{\mu_2} \sum_{\{i | X_i^k > c\}} \omega_i^A \left(y_i - \mu_2\right)^2 \right] \tag{4}$$

where $\omega_i = I(i \in A)$. This can be generalized to

$$\omega_i = I(i \in A) + (1 - \eta) I(i \in B)$$

where $\eta \in (0,1]$ is a learning rate. ω_i's collapse to CART when $\eta = 1$.

SGT Algorithm

Algorithm 1 Slow-Growing Tree

```
Input: Training data $[y_i \;\; \boldsymbol{X}_i]$, test set predictors $\boldsymbol{X}_j$, learning rate $\eta \in (0,1]$, maximal Gini coefficient $\bar{G}$
Initialize $\omega_i^0 = 1 \;\; \forall i$.
for $l$'s such that $G_l < \bar{G}$ do
    $(k_l^*, c_l^*) = \underset{k \in \mathcal{K}, c \in \mathbb{R}}{\arg\min} \left[ \min_{\mu_1} \sum_{\{i | X_i^k \leq c\}} \omega_i^l (y_i - \mu_1)^2 + \min_{\mu_2} \sum_{\{i | X_i^k > c\}} \omega_i^l (y_i - \mu_2)^2 \right]$
    Create 2 children nodes with $\omega_i^{l,+} = \omega_i^l (1 - \eta I(X_i^{k_l^*} \leq c_l^*))$ and $\omega_i^{l,-} = \omega_i^l (1 - \eta I(X_i^{k_l^*} > c_l^*))$.
end for
Return: $\hat{y}_j = \sum_{l=1}^{L} \left( w_j^l(\boldsymbol{X}_j) \sum_{i=1}^{N} \omega_i^l y_i \right)$ where $w_j^l(\boldsymbol{X}_j) = \frac{\omega_j^l(\boldsymbol{X}_j)}{\sum_{l=1}^{L} \omega_j^l(\boldsymbol{X}_j)}$
```

Simulations

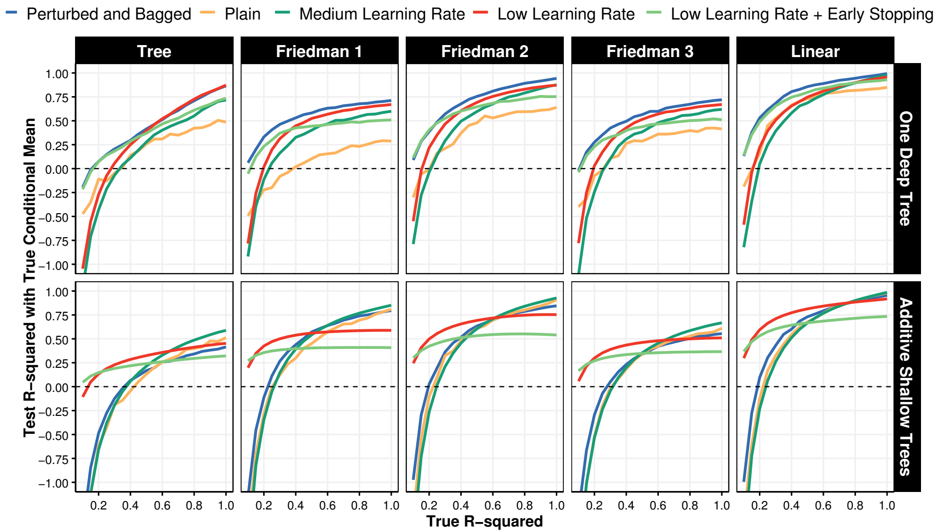


Figure: This plots the hold-out sample R^2 between the prediction **and the true conditional mean**. The level of noise is decreasing along the x-axis. Column facets are DGPs and row facets are "models". The y-axis is cut at -1 to favor readability because a few models go largely below it for the lowest true R^2 case.