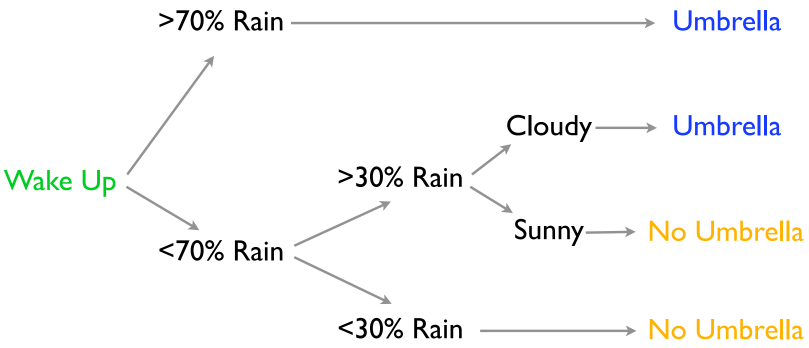
Empirical Bayesian Forests

Matt Taddy, Chicago Booth

with Chun-Sheng Chen, Jun Yun, and Mitch Wyle at eBay

`faculty.chicagobooth.edu/matt.taddy/research`

What is a Decision Tree?



Tree-logic uses a series of steps to come to a conclusion.
The trick is to have mini-decisions combine for good choices.
Each decision is a node, and the final prediction is a leaf node

Decision Trees are a Regression Model

You have inputs $\mathbf{x}$ (forecast, current conditions)
and an output of interest y (need for an umbrella).

Based on previous data, the goal is to specify branches of choices that lead to good predictions in new scenarios.
In other words, you want to estimate a Tree Model.

Instead of linear coefficients, we need to find 'decision nodes':
split-rules defined via thresholds on some dimension of $\mathbf{x}$.

Nodes have a parent-child structure: every node except the root has a parent, and every node except the leaves has two children.

Decision trees are like a game of mousetrap

You drop your **x** covariates in at the top, and each decision node bounces you either left or right. Finally, you end up in a leaf node which contains the data subset defined by these decisions (splits).

$$\begin{tikzcd}
 & & x_i = 0 \arrow[dl] \arrow[dr] & \\
 & x_j = 2 \arrow[dl] \arrow[dr] & & \{\mathbf{x} : x_i > 0\} \\
\{\mathbf{x} : x_i \leq 0, x_j \leq 2\} & & \{\mathbf{x} : x_i \leq 0, x_j > 2\} &
\end{tikzcd}$$

The prediction rule at each leaf (a class probability or predicted $\hat{y}$) is the average of the sample y values that ended up in that leaf.

Given a parent set of data $\{\mathbf{x}_i, y_i\}_{i=1}^n$, the optimal split is that location x_{ij} on some dimension j on some observation i, so that the child sets

$$\text{left: } \{\mathbf{x}_k, y_k : \; x_{kj} \leq x_{ij}\} \;\text{ and }\; \text{right: } \{\mathbf{x}_k, y_k : \; x_{kj} > x_{ij}\}$$

are as homogeneous in response y as possible.

For example, we will minimize the sum of squared errors

$$\sum_{k \in \text{left}} (y_k - \bar{y}_{\text{left}})^2 + \sum_{k \in \text{right}} (y_k - \bar{y}_{\text{right}})^2$$

for *regression trees*, or gini impurity for *classification trees* (e.g., the sum across children 'c' of $n_\text{c}\bar{y}_\text{c}(1 - \bar{y}_\text{c})$ if $y \in \{0, 1\}$).

We estimate decision trees by being recursive and greedy

CART grows the tree through a sequence of splits:

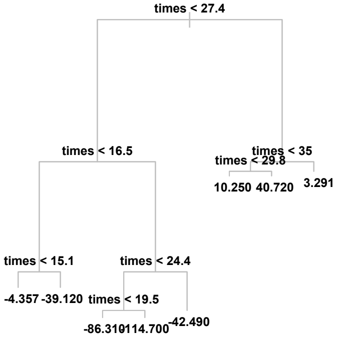
- Given any set (node) of data, you can find the optimal split (the error minimizing split) and divide into two child sets.
- We then look at each child set, and again find the optimal split to divide it into two homogeneous subsets.
- The children become parents, and we look again for the optimal split on their new children (the grandchildren!).

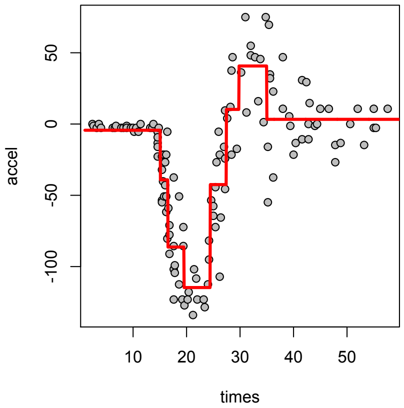
You stop splitting and growing when the size of the leaf nodes hits some minimum threshold (e.g., say no less than 10 obsv per leaf).

Trees are awesome

They automatically learn non-linear response functions and will discover interactions between variables.

Example: Motorcycle Crash Test Dummy Data
x is time from impact, y is acceleration on the helmet.





Random Forests

CART is an effective way to choose a single tree, but often there are many possible trees that fit the data similarly well.

An alternative approach is to make use of random forests.

- Sample B subsets of the data + variables:
  e.g., observations $1, 5, 20, \ldots$ and inputs $2, 10, 17, \ldots$
- Fit a tree to each subset, to get B fitted trees is $\mathcal{T}_b$.
- Average prediction across trees:
  - for regression average $\mathbb{E}[y|\mathbf{x}] = \frac{1}{B}\sum_{b=1}^{B} \mathcal{T}_b(\mathbf{x})$.
  - for classification let $\{\mathcal{T}_b(\mathbf{x})\}_{b=1}^{B}$ vote on $\hat{y}$.

The observation resample is usually *with-replacement*, so that this is taking the *average of bootstrapped trees* (i.e., 'bagging')

'distribution free' nonparametric statistics

1: Find some statistic that matters for your problem,
regardless of the 'data generating process' (DGP).

2: Derive the distribution for this stat under minimal assumptions.

For (2): say $\mathbf{z}_l = \{\mathbf{x}_l, y_l\}$ is a possible data point. Then

$$\mathrm{p}(\mathbf{Z}) = \frac{1}{|\boldsymbol{\theta}|} \sum_{l=1}^{L} \theta_l \mathbb{1}[\mathbf{Z} = \mathbf{z}_l]$$

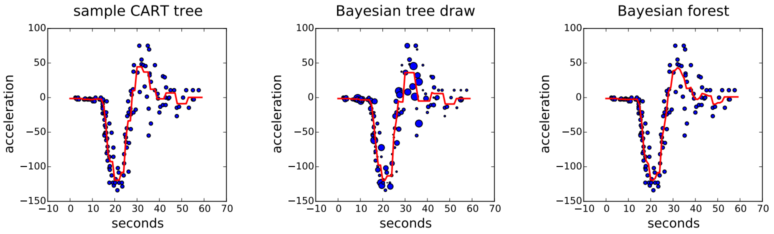
where L is a *large* number of possible values.

- Use sample as stand-in for the L points.
- Bayesian model for the θ_l weights.
- *This is essentially a bootstrap.*

Our statistic of interest is the CART fit...

Bayesian Forest

```
for $b = 1$ to $B$ do
    draw $\boldsymbol{\theta}^b \overset{iid}{\sim} \mathrm{Exp}(\mathbf{1})$
    run weighted-sample CART to get $\mathcal{T}_b = \mathcal{T}(\boldsymbol{\theta}^b)$
end for
```



Given forests as a posterior, we can start talking about *variance*.

Consider data at a given node in full-sample CART.

Say $\mu_j(x_{ij})$ is the leaf rule implied by next splitting on a given x_j.

The resulting *impurity is a posterior functional*:

$$\sigma_j^2(\boldsymbol{\theta}) = \frac{1}{n} \sum_i \theta_i \left[y_i - \mu_j(x_{ij}) \right]^2 .$$

Say $\sigma_1(\boldsymbol{\theta})$ is the full CART split's impurity,
and $\sigma_j(\boldsymbol{\theta})$ is that for any other location.

Then we can derive a first order approx to $\Delta_j(\boldsymbol{\theta}) = \sigma_1^2(\boldsymbol{\theta}) - \sigma_j^2(\boldsymbol{\theta})$

and we then have $\mathrm{p}(\Delta_j) < 0$.

Theoretical trunk stability

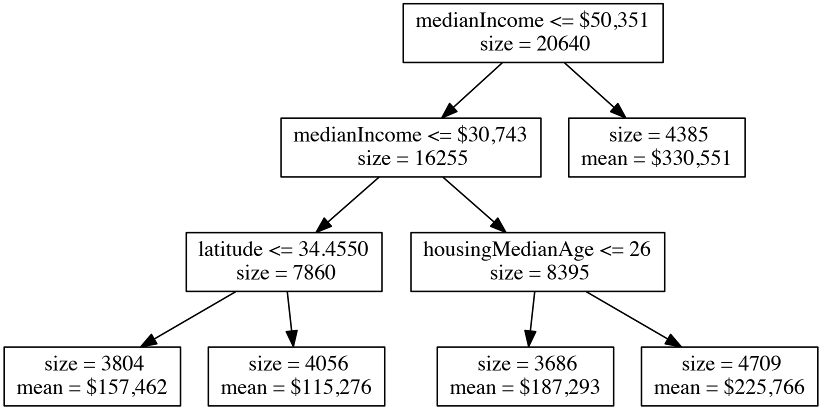
For the data at a given node on the sample CART tree, the probability that the next split for a posterior DGP realization matches the sample split location is

$$\mathrm{p}\,(\text{split matches sample CART}) \gtrsim 1 - \frac{p}{\sqrt{n}} e^{-n},$$

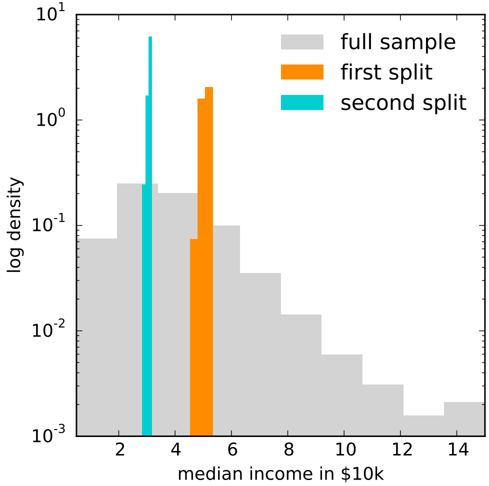
where p is the number of possible split locations and n the number of observations on the current node.

So things are pretty stable (until they aren't).

California Housing Data



20k observations.



- sample tree occurs 62% of the time.
- 90% of trees split on income twice, and then latitude.
- 100% of trees have 1st 2 splits on median income.

So trees are stable, at the trunk.

A big data problem

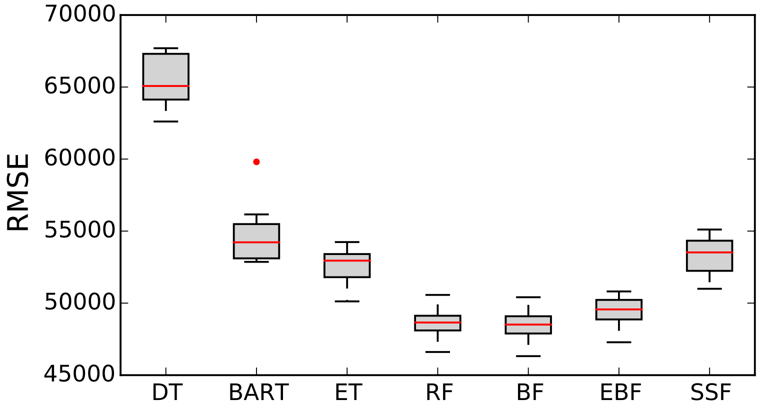
RFs are expensive when data is too big to fit in memory.

Subsampling forests (fitting CART on *without replacement* samples) leads to a big drop in performance.

But wait: if the trunks are stable, can we just fit that once and then fit forests at each branch?

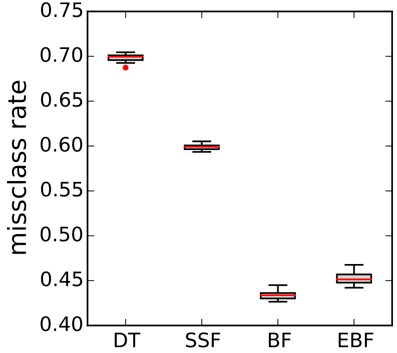
This is classic Empirical Bayes: fix higher levels in a *hierarchical model*, and direct your machinery at learning the hard bits.

Emperical Bayesian Forests



Since the trunks in a full forest are all similar, EBF and BF give nearly the same results. *SSF does not.*

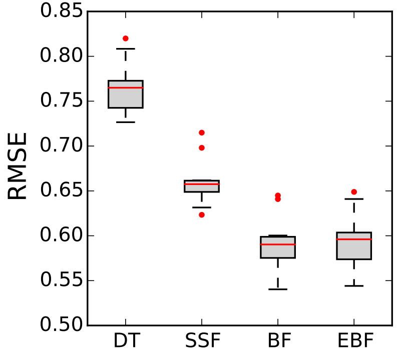
EBFs work all over the place



$\overline{\text{MCR}}$	% WTB	
0.4341	0.0	BF
0.4531	4.4	EBF
0.5989	38.0	SSF
0.6979	60.8	DT

Predicting beer choice from demographics

EBFs work all over the place



$\overline{\text{RMSE}}$	% WTB	
0.5905	0.0	BF
0.5953	0.8	EBF
0.6607	11.9	SSF
0.7648	29.5	DT

or wine rating from chemical profile

Choosing the trunk depth

Distributed computing perspective: fix only as deep as you must!

	CA housing			Wine			Beer		
Min Leaf Size in 10^3	6	3	1.5	2	1	0.5	20	10	5
% Worse Than Best	1.6	2.4	4.3	0.3	0.8	2.2	1.0	4.4	7.6

Still, open questions. e.g., more trees vs shallower trunk?

No rigorous answer yet... maybe it's in the npB framework.

EBFs at EBay: predicting Bad Buyer Experiences

A BBE could be receiving something that is 'significantly not as described', or shipping delays, or any of many other problems.

The models are updated frequently, and information about p(BBE) is an input to search rankings and more.

Big Data axiom: more data beats fancy model.

The best thing to improve predictions is more data.
With millions of daily transactions, there's little limit on data.

EBFs at EBay

Full random forest runs take too long on full data
(even using distributed tree algorithms).

Subsampling led to a noticeable and big drop in performance.

So: EBFs!

- trunk can be fit in distribution using Spark `MLLib`.
- this trunk acts as a sorting function to map observations to separate locations corresponding to each branch.
- Forests are then fit on a machine for each branch.

EBFs at EBay

On 12 million transactions, EBF with 32 branches yields a 1.3% drop in misclassification over the SSF alternatives.

This amounts to more than 20,000 extra detected BBE occurrences over this short time window.

Putting it into production requires some careful engineering, but this really is *a very simple algorithm*.

If you already fit RFs and are hitting time/space constraints, then an EBF is lots of gain for little pain.

The key to big data

Use plug-in estimates for the stuff that is easy to measure.

Partition conditional on these plug-ins.

Direct the full data towards the stuff that is tough to learn.

Thanks!