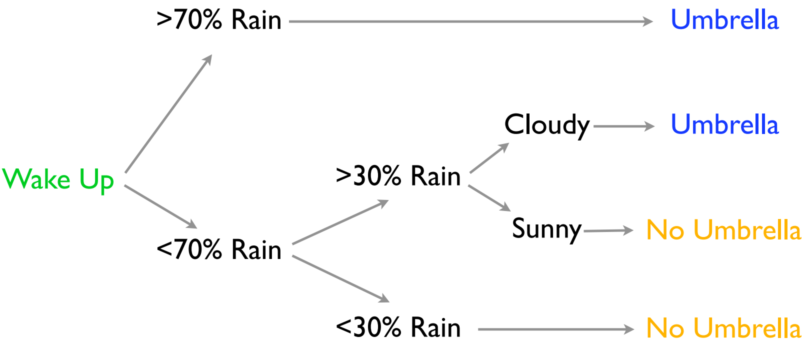
Empirical Bayesian Forests

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

Neighborhoods: Parents, Siblings, Children, and Leaves

$$
\begin{tikzcd}
 & & P(\eta) \arrow[dl] \arrow[dr] & \\
 & \eta \arrow[dl] \arrow[dr] & & S(\eta) \\
C_l(\eta) & & C_r(\eta) &
\end{tikzcd}
$$

Every node has a *parent* (except root).

Nodes sharing parents are *siblings*.

Each node can have its own *children*.

After descending through splits, leaf nodes hold a data subset

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

Estimation of Decision Trees

We maximize data likelihood by thinking recursively:

Split the data into two different decisions about y.

Take each new partition and split again.

Growing your tree with the CART algorithm:

- Find the split location in $\mathbf{x}$ that minimizes deviance.
  - $\hat{y}_i$ or $\hat{p}_i$ change depending on whether $x_i < x_{\text{split}}$.
- You then grow the tree at this point
  - Each new child node contains a subset of the data.
- View each child as a new dataset, and try to grow again.
  - Stop splitting/growing when there are some fixed minimum number of observations in each leaf node.

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')

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

$$\mathrm{p}(\mathbf{Z}) = \sum_{l=1}^{L} \omega_l \mathbb{1}[\mathbf{Z} = \mathbf{z}_l]$$

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

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

Our statistic of interest is the CART fit...

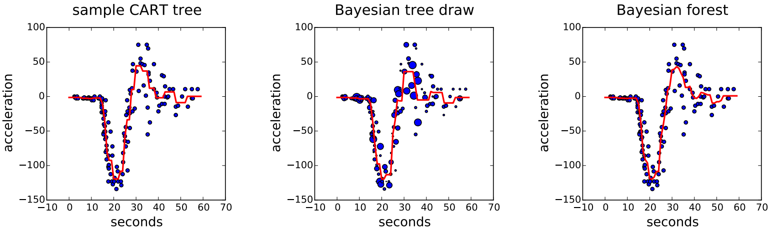
Bayesian Forest

for $b = 1$ **to** B **do**

draw $\boldsymbol{\theta}^b \stackrel{iid}{\sim} \mathrm{Exp}(\mathbf{1})$

run weighted-sample CART to get $\mathcal{T}_b = \mathcal{T}(\boldsymbol{\theta}^b)$

end for



Given this expression of forests as a posterior,
we can start talking about *variance*

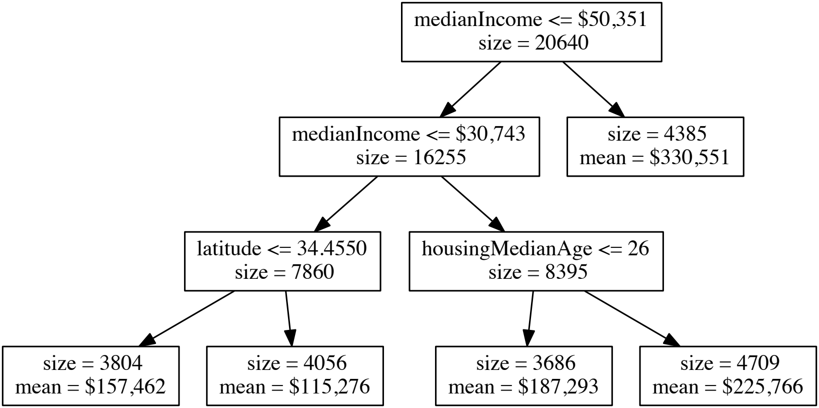
For the data at a given node on the sample CART tree, the probability that the next split for a posterior DGP realization matches the sample split location is

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

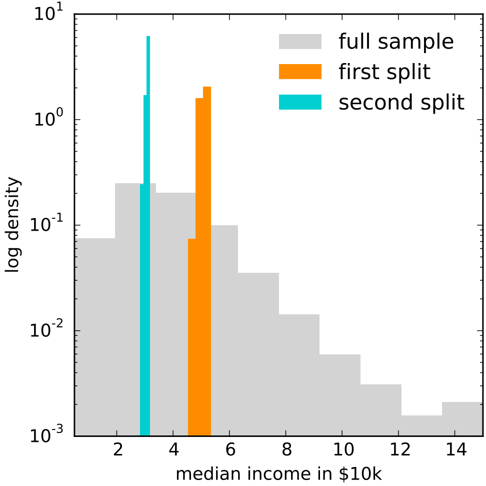
where p is the number of possible split locations and n the number of observations on the current node.

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

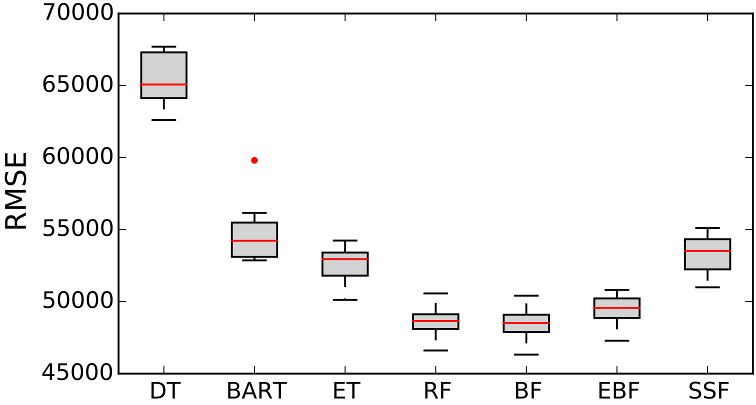
RFs are expensive when data is too big to fit in memory.

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?

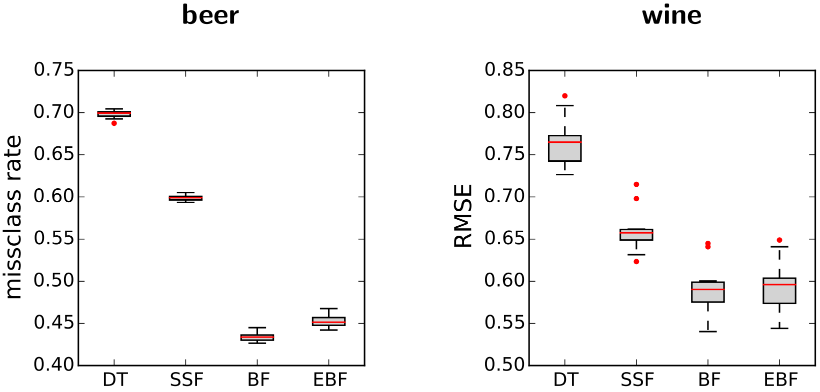
This is classic Empirical Bayes: fix higher levels in a *hierarchical model*, and direct your machinery at learning the hard bits.

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



Predicting beer choice from demographics,
and wine rating from chemical profile

EBFs at eBay: predicting Bad Buyer Experiences

- 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.

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.

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!