Empirical Bayesian Forests

Matt Taddy @ Chicago Booth

Chun-Sheng Chen, Jun Yu, and Mitch Wyle @ eBay Trust Science

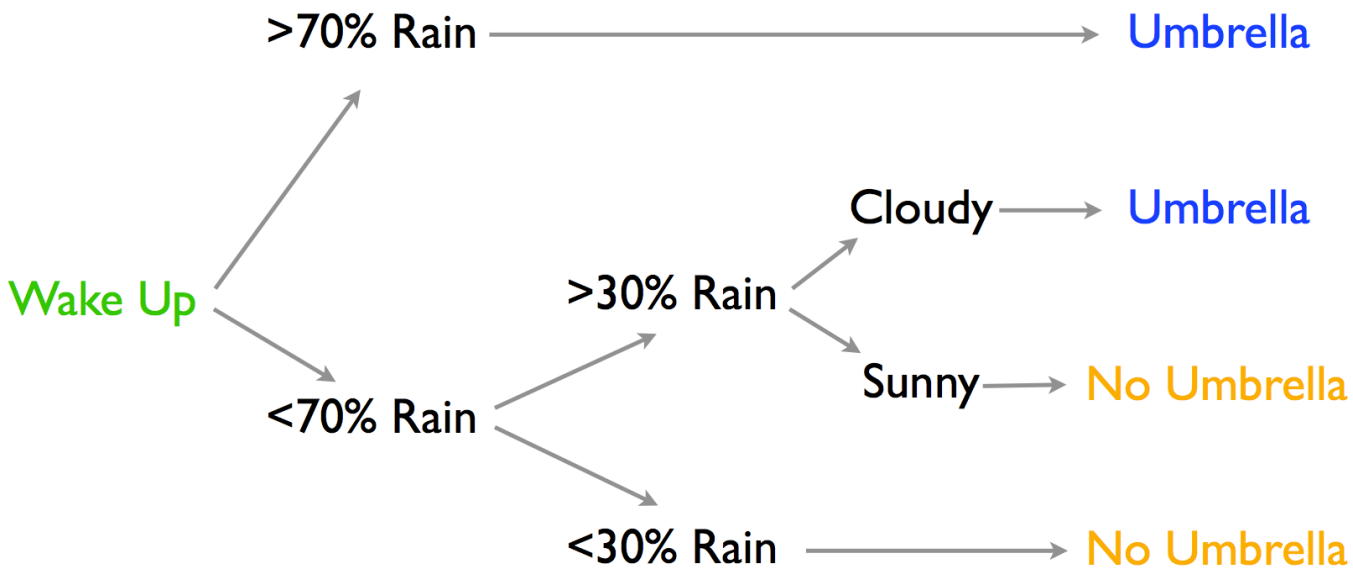
Outline

- Decision Tree and Random Forests
- Empirical Bayesian Forests
  - Bayesian Forests
  - Bootstrap vs. Bayesian Bootstrap
  - Truck stability
- Experimental Result
  - Benchmark datasets
  - eBay data

What is a Decision Tree?

Decision Tree (a.k.a CART[1]) is a predictive model, which maps from observations ($\mathbf{X}$) to a conclusion (Y) using a series of rules.

Example: Predict whether or not to bring an umbrella given the forecast and the current condition.

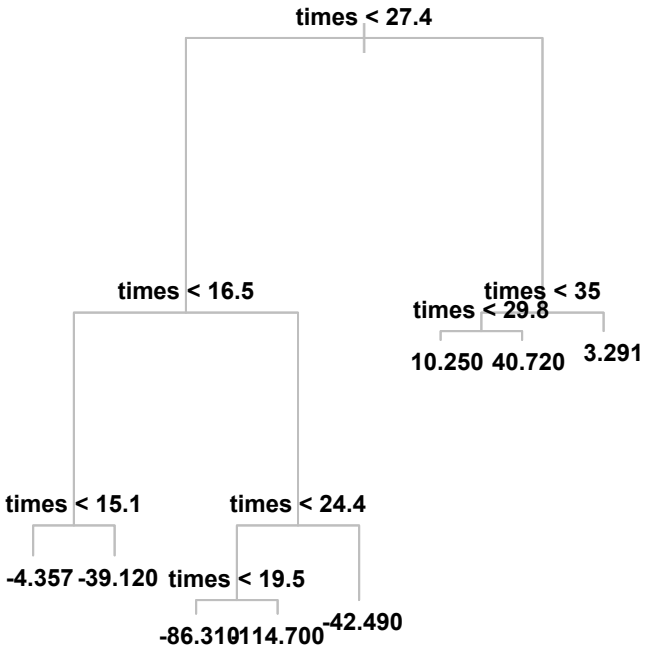


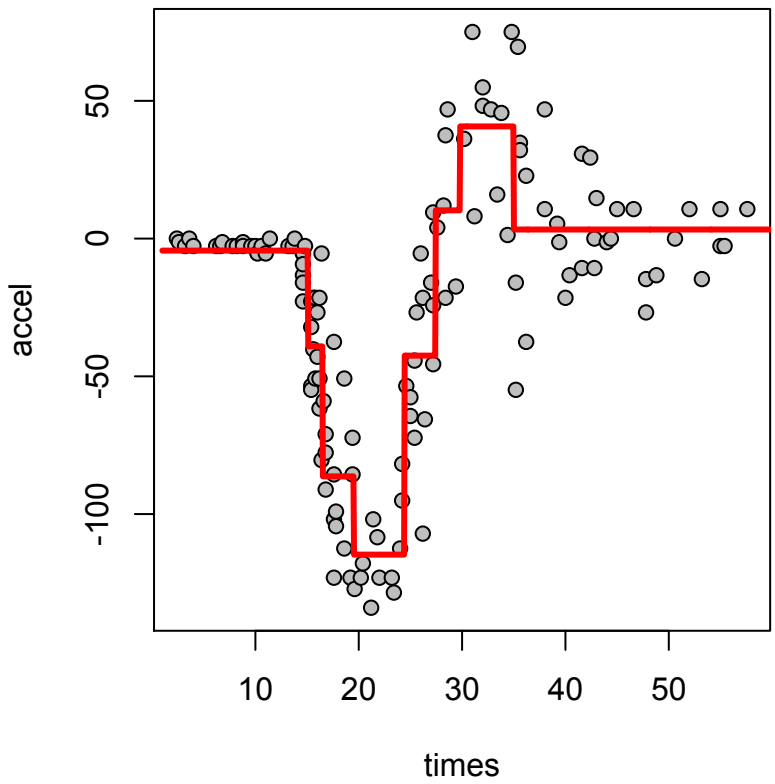
[1]CART: Classification And Regression Trees

Decision tree works well out of box

Decision tree automatically learns **non-linear** response functions and discovers **interactions** between variables.

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





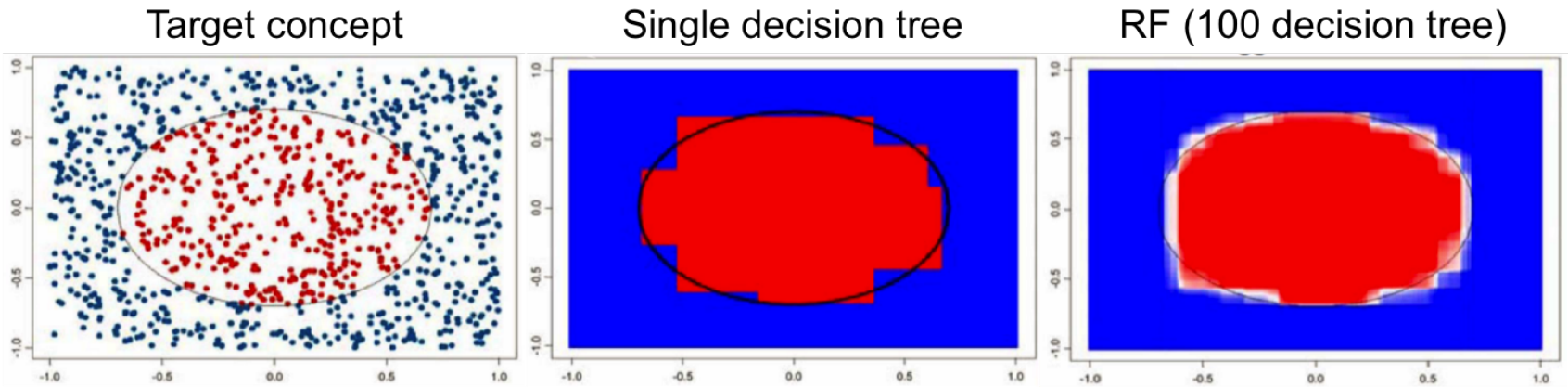
Random Forests (RF)

Unfortunately, CART has high variance and can overfit because deep tree structure is unstable.

Instead, we can construct an ensemble of decision trees (Random Forests):

- Bootstrap the original dataset by sampling with replacement.
- For each bootstrapped data, fit a CART tree.
- For a testing data point x, take the average prediction from this forest of trees.

Random Forests helps



Real structure that persists across datasets shows up in the average. Noisy useless signals will average out to have no effect.

Random Forests are awesome, BUT ...

- Search for *optimal splits* is expensive on Big data.
- Sub-Sampling Forests hurts the performance because deep structure *needs* big samples.

To cut computation without hurting performance, we need to figure out what portions of the tree are hard or easy to learn. Then, we can use a little bit of the data to learn the easy stuff and direct our full data at the hard stuff.

Bayesian Forests

Our framework for thinking about easy vs hard learning derives Random Forests as a Bayesian posterior over the optimal tree.

Imagine you see all data and fit the best 'population' CART tree. The Bayesian/Random Forest represents your uncertainty about this optimal tree. (you are uncertain because you don't actually get to see all data.)

Classic Bootstrap (RF) vs. Bayesian Bootstrap (BF)

- Operationally similar. BF fits CART to randomly weighted data whereas RF fits to randomly re-sampled data.
- Interpretatively different. BB simulates the posterior distribution of a statistic ϕ whereas the bootstrap simulates the sampling distribution of a statistic ϕ.

Theoretical trunk stability

We are able to derive theoretically that the earliest structure in the tree – the trunk – should be very stable for large samples.

For the data at a given node, the probability that the optimal split on resampled data matches the true optimal split is

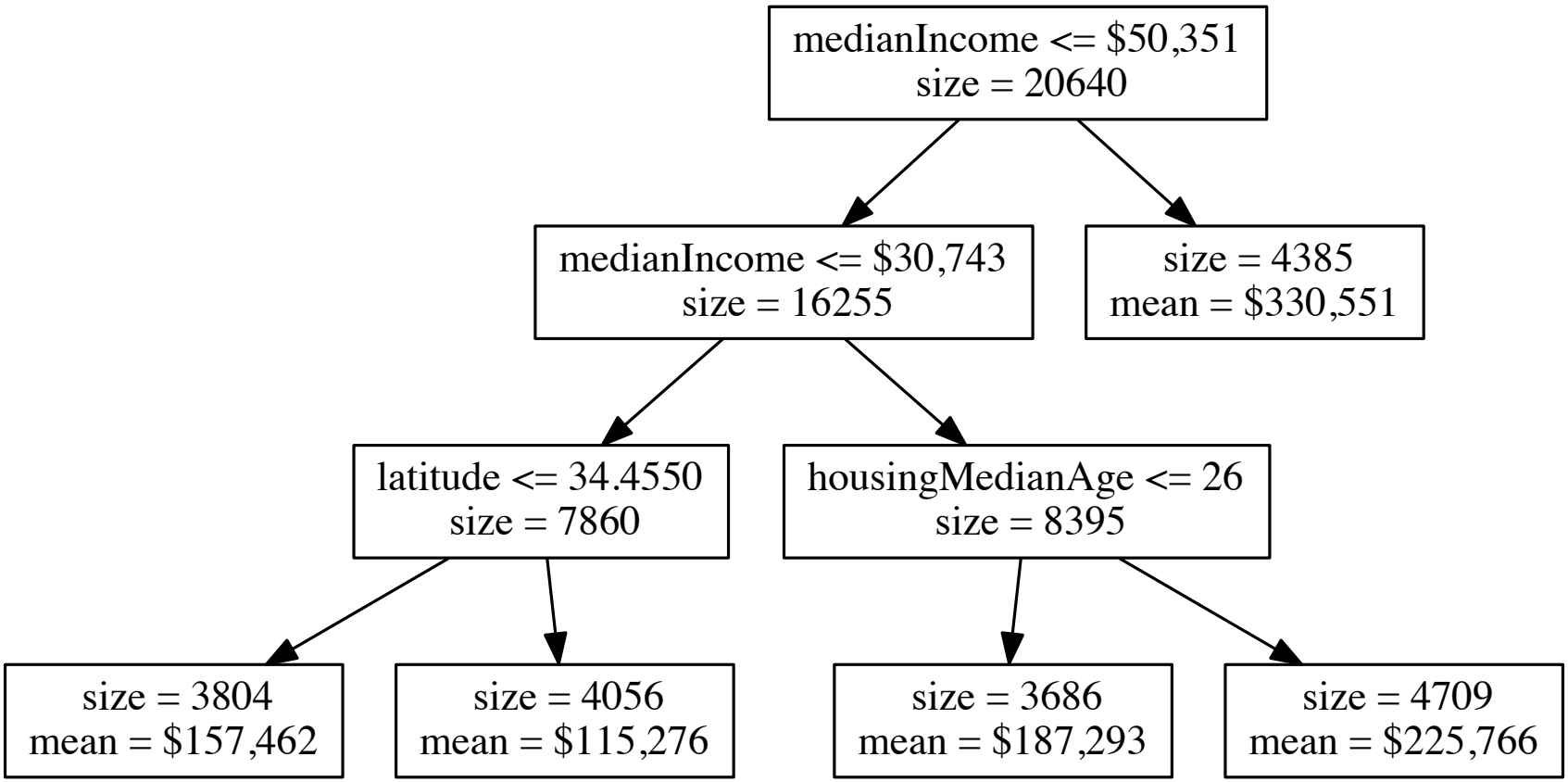
$$\mathrm{p}\,(\text{split matches sample CART}) \gtrsim 1 - \frac{p}{\sqrt{n}} e^{-n},$$

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

Things are pretty stable, until they aren't: as the tree grows, node sizes get smaller and chance of a non-optimal split multiplies.

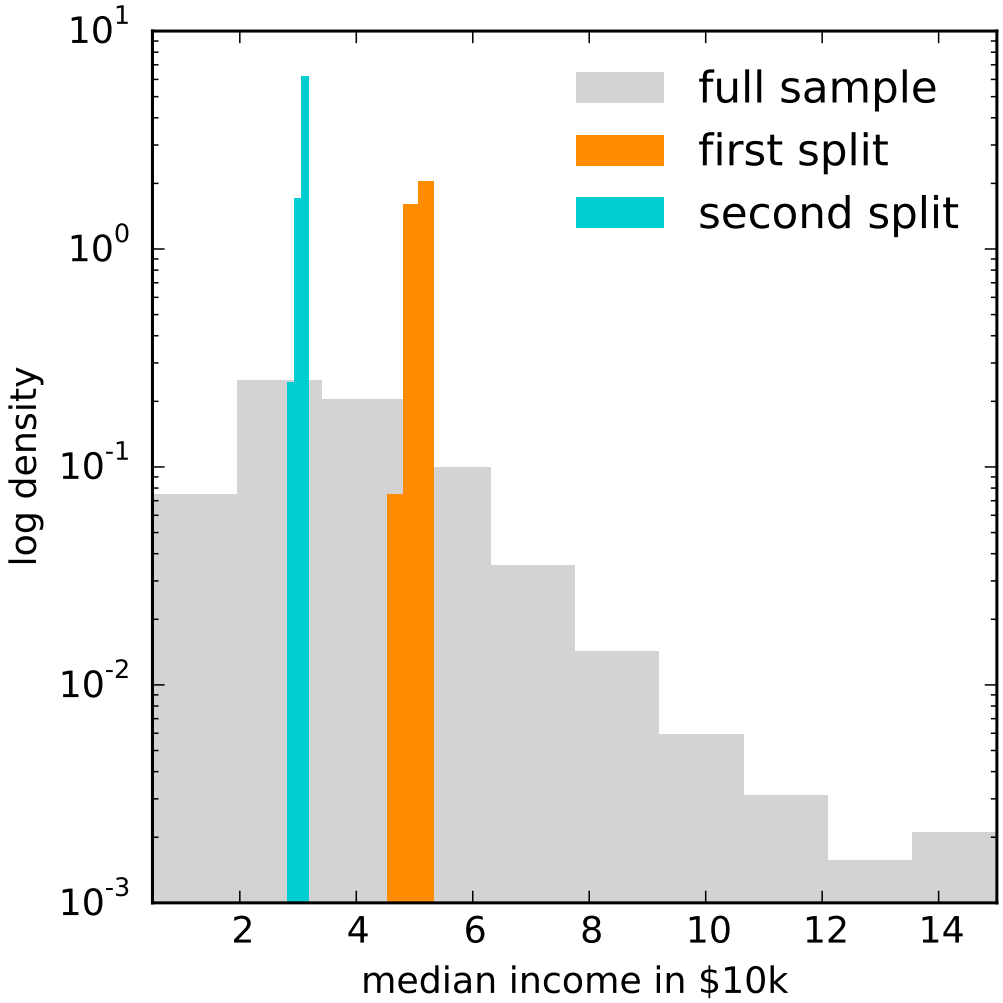
Empirical trunk stability

California Housing Data: 20k observations on median home prices in zip codes.



Above is the trunk you get setting min-leaf-size of 3500.

Empirical trunk stability cont'd



- 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.

Empirically and theoretically: trees are stable at the trunk.

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

Subsampling forests lead to a big drop in performance.

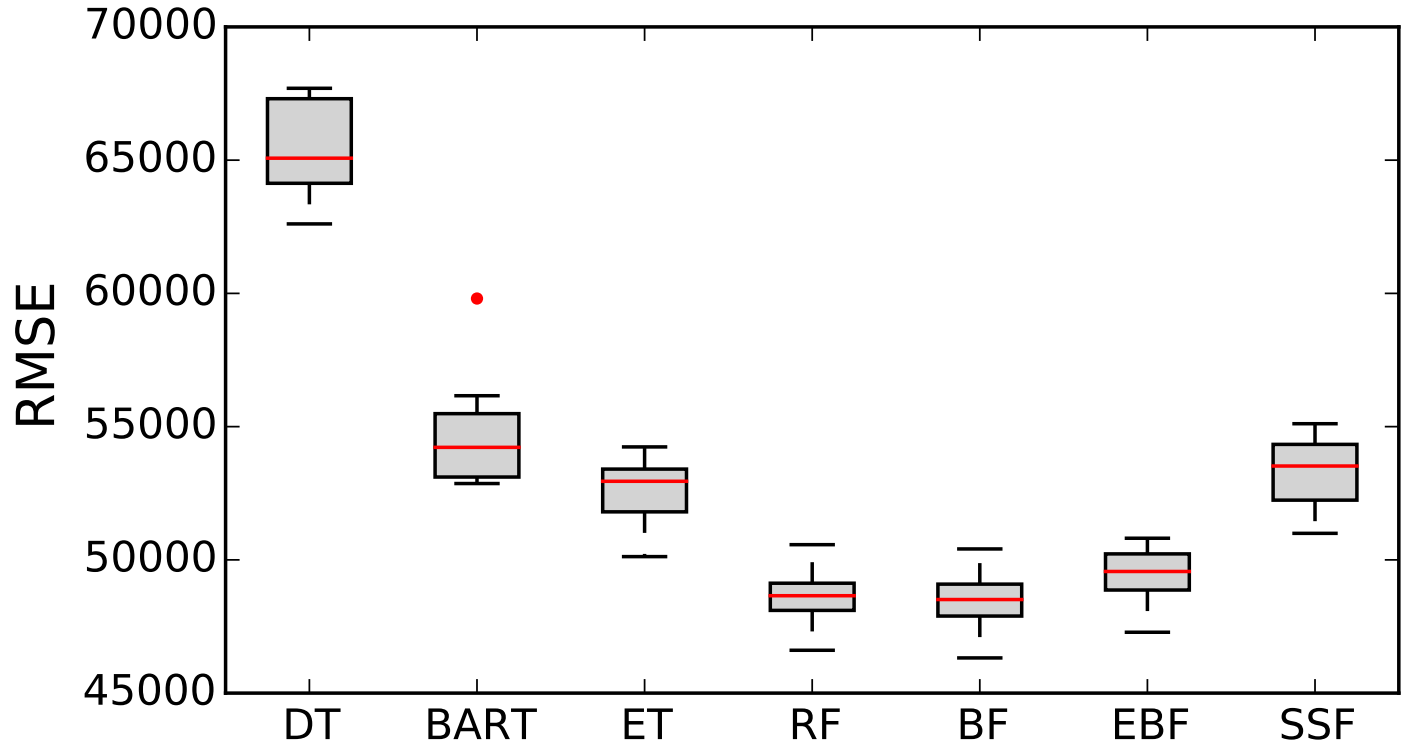
But wait: if the trunks are stable, can we just fit that once and then fit forests at each branch? Yes!

Empirical Bayesian Forests (EBF):

- fit a single tree to a shallow trunk.
- Use this as a mapper to direct full data to each branch.
- Fit a full forest on the smaller branch datasets.

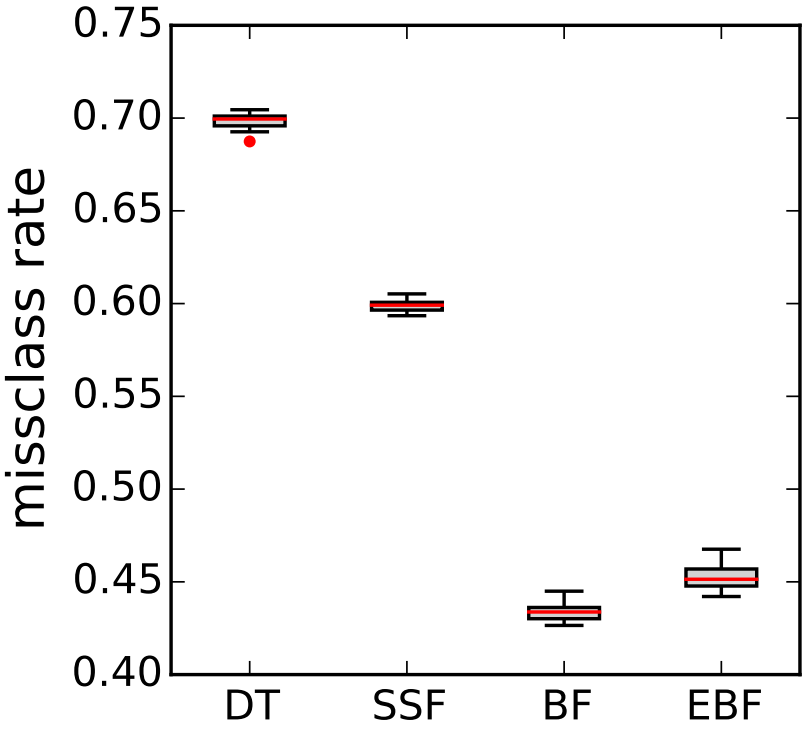
This is classic Empirical Bayes: fix higher levels in a *hierarchical model*, and direct your machinery+data at learning the hard bits.

Benchmark Dataset: California housing data



- Here EBF and BF give nearly the same results.
- Since the trunks are all the same for each tree in a full forest, EBF looks nearly the same at a fraction of computational cost.

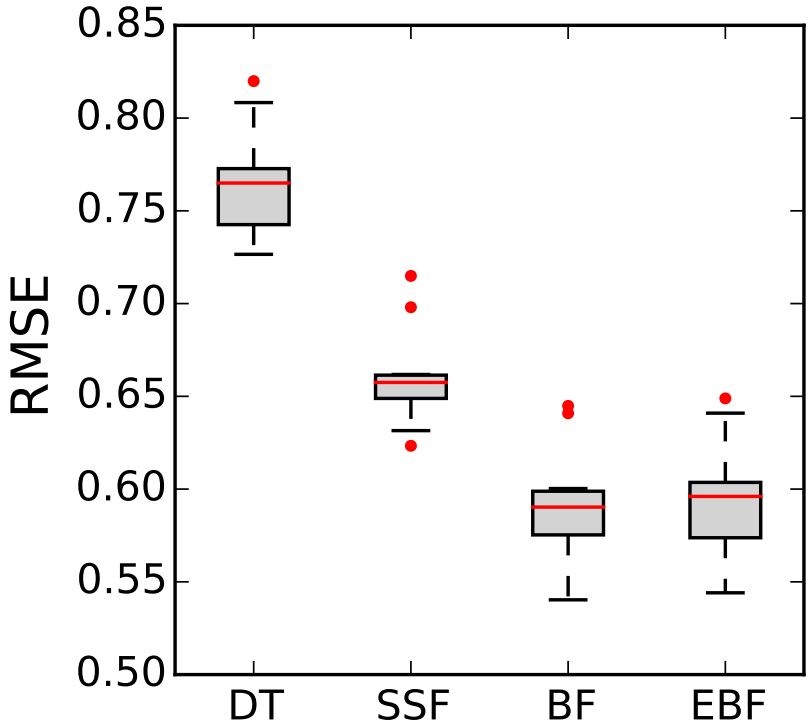
Benchmark Dataset: Beer data



$\overline{\text{MCR}}$	% 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

Benchmark Dataset: Wine data



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

Predicting wine rating from chemical profile

Choosing the trunk depth

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

How big is each machine? Make that your branch size.

	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?

A key point: we do not think that EBFs are better than forests fit to all the data. But EBFs allow you to fit to much more data in less time without hurting performance too much.

Big Data axiom: more data beats fancy model.

EBFs at eBay: Predicting Defects

Defects include various complaints from eBay buyers regarding a transaction: item not received, significantly not as described, negative feedback for seller, shipping delays, others

The defect predictor model learns from sellers' historical behaviors and predicts the likelihood of a seller's causing defects on eBay. The Defect model's prediction scores for each seller are fed into search rankings which demotes (moves down) results from sellers more likely to cause defects.

The best way to improve predictions is more data. With many millions of daily transactions, there is almost no limit to the size of our data.

EBFs at eBay: Predicting Defects cont'd

Full random forest training takes too long on a full data sample (even using distributed tree algorithms).

Subsampling led to a noticeable and big drop in performance.

So: EBFs!

- trunk can be fit in distributed training using Spark `MLLib`.
- trunk of "leaf trees" acts as a sorting function to map observations
  to separate locations corresponding to each forest.
- Forests are then fit on a single "slot" (machine) and finally sifted together

EBFs at eBay: Predicting Defects cont'd

For 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 accepted ICML paper can be found at

`http://icml.cc/2015/?page_id=710`

Thanks!

Q & A