Big Data and Bayesian Nonparametrics

Matt Taddy, Chicago Booth

faculty.chicagobooth.edu/matt.taddy/research

Big Data

The sample sizes are enormous.

- We'll see 21 and 200 million today.
- Data can't fit in memory, or even storage, on a single machine.

The data are super weird.

- Internet transaction data distributions have a big spike at zero and spikes at other discrete values (e.g., 1 or $99).
- Big tails (e.g., $12 mil/month eBay user spend) that matter.
- The potential feature space is unmanageably large.
- We cannot write down or measure believable models.

Both 'Big' and 'Strange' beg for nonparametrics.

what are nonparametrics?

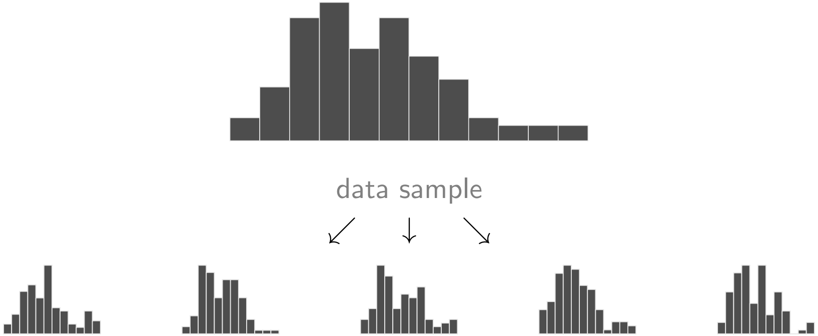
There are many flavors. One option is *distribution free NP:*

1: Find some statistic that is useful regardless of DGP.
(DGP = Data Generating Process)

2: Quantify uncertainty about this stat under minimal assumptions.

Practitioners apply the simple stat and feel happy that it will work.

Our sample of data points $\mathbf{z}_i = [\mathbf{x}_i, y_i]$ is as good a guess as any at what the population DGP looks like. We can explore uncertainty about this population by randomly re-weighting our sample.



data sample

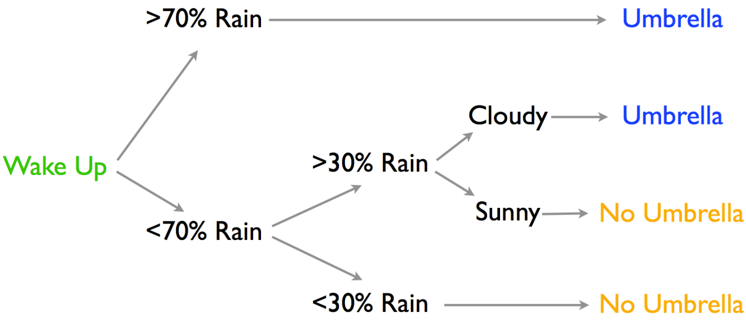
↙ ↓ ↘

randomly re-weighted samples

This is just the Bayesian bootstrap. (Rubin 1981)

Example: Decision Trees

Trees are great: nonlinearity, deep interactions, heteroskedasticity.



The ‘optimal’ decision tree is a statistic we care about (s.w.c.a).

CART: greedy growing with optimal splits

For any dataset, find the input variable and dimension so that you can use $\leq$ or $>$ on this x to minimize something like

$$n_{\text{left}}\text{var}(\mathbf{z}_k : x_{kj} \leq x) + n_{\text{right}}\text{var}(\mathbf{z}_k : x_{kj} > x)$$

You repeat this on each of the (left and right) child nodes until you get to leaves that only contain a small set of data.

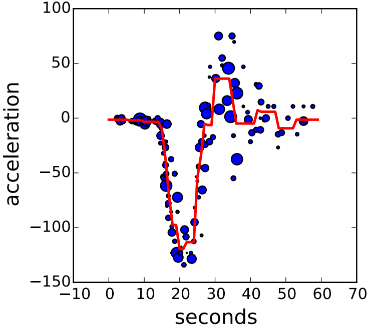
Quantifying uncertainty: do this, but for random reweightings of the data (i.e., so that those variances are now weighted sums).

Forests

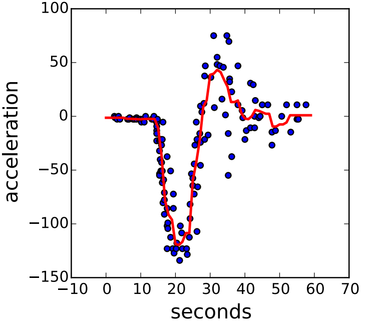
Each re-weighting yields a different tree. We are left with a forest.

Fact: the average prediction from this forest is better than the prediction from any single tree.

one tree



posterior mean



Random Forest ≈ Bayesian forest ≈ posterior over CART fits.

Trunk stability

Based on this model for forests as randomly weighted CART, we can derive a theory on the variance of trees.

This yields that, for data at a given node,

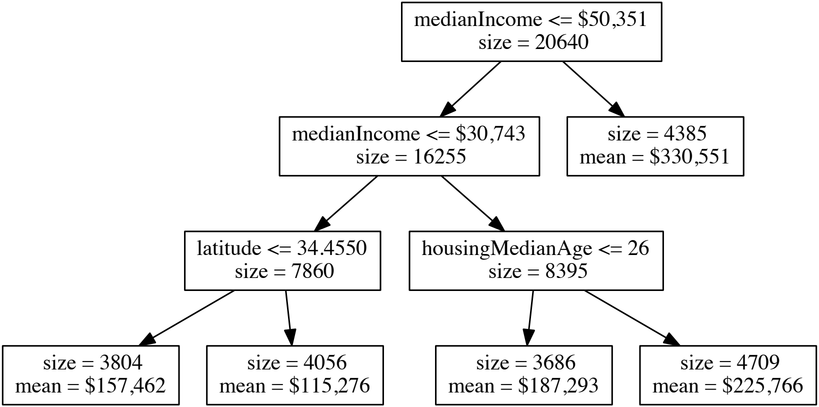
$$\mathrm{p}\,(\text{optimal split matches sample CART}) \gtrsim 1 - \frac{p}{\sqrt{n}} e^{-n},$$

with p split locations and n observations.

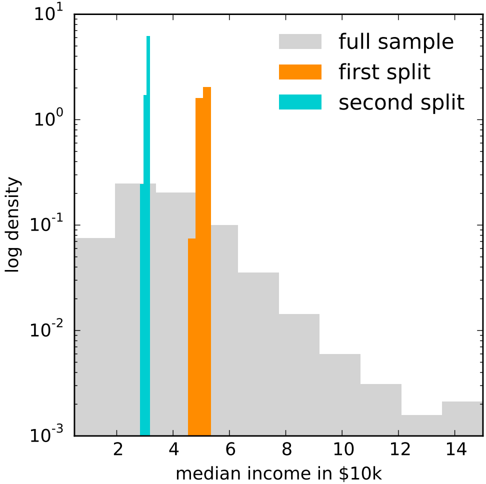
⇒ If we look at our *forest*, we expect that the trunks of the trees (the first few splits) will look the same across trees.

California Housing Data

20k observations on median home prices in zip codes.



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



- 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.

Random Forest bottlenecks

RFs (or BFs) are awesome, but when the data are too big to fit in memory or on a single machine they get extremely expensive. (e.g., Google PLANET, Panda 2009).

A common solution is a ‘sub-sampling forest’: instead of drawing with-replacement, or re-weighting, draw $m \ll n$ sub-samples.

This defeats the whole purpose of trees: these are rules that are designed to grow in complexity with the amount of data available. (that’s why we bother storing so much data!)

If you starve the individual trees of data you lose.

Data Distribution and Big Data

Distributed: independent computations on many datasets.

For truly Big Data we need distributed algorithms.

The trick is to find strategies for distribution so that local-machine calculations are as relevant as possible to the global inference question: only data that needs to be together is together.

Empirical Bayesian Forests (EBF)

RFs are expensive. Sub-sampling hurts bad.

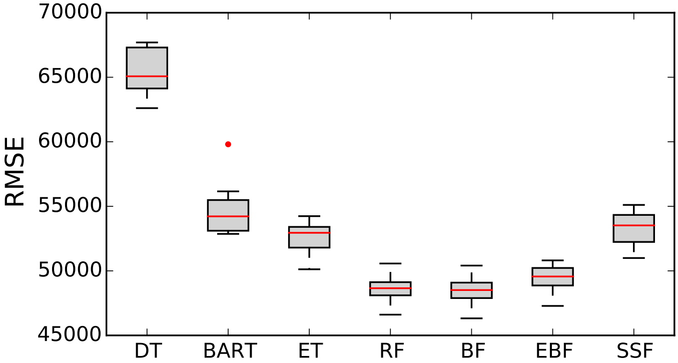
Instead:

- fit a single tree to a shallow trunk.
- Map data to each branch.
- Fit a full forest on the smaller branch datasets.

Since the trunks are all similar for each tree in a full forest, our EBF looks nearly the same at a fraction of computational cost.

It's an updated version of classical Empirical Bayes:
use plug-in estimates at high levels in a hierarchical model,
focus effort at full Bayesian learning for the the hard bits.

OOS predictive performance on California Housing



Here EBF and BF give nearly the same results. *SSF does not.*

EBFs crunch more data faster without hurting performance.

Catching Bad Buyer Experiences at eBay

BBE: ‘not as described’, delays, etc.

p(BBE) is a key input to search rankings, and elsewhere.

Big Data axiom: best way to improve prediction is more data.

(a.k.a. ‘data beats model’)

On 200 million transactions, EBF with 32 branches yields a 1.5% drop in misclassification over the SSF alternatives.

EBFs via Spark $\Rightarrow$ more data in less time.

Putting it into production requires some careful engineering, but this really is a very simple algorithm. Big gain, little pain.

Efficient Big Data analysis

To cut computation without hurting performance, we need to think about what portions of the ‘model’ are hard or easy to learn.

Once we figure this out, we can use a little bit of the data to learn the easy stuff and direct our full data at the hard stuff.

This is *the* future for Big Data.

thanks!