Big Data and Bayesian Nonparametrics

Matt Taddy, Chicago Booth
Chun-Sheng Chen, Jun Yu, and Mitch Wyle, eBay Trust Science

Big Data

The sample sizes are enormous. 200+ million obs
The data are super weird. density spikes, obese tails

'Big' and 'Strange' beg for nonparametrics.

In usual BNP you *model* a complex generative process with flexible priors, then apply that model directly in prediction and inference.

$$\text{e.g.,} \quad y = f(\mathbf{x}) + \epsilon, \quad \text{or even just} \quad f(y|\mathbf{x})$$

However averaging over all of the nuisance parameters we introduce to be 'flexible' is a hard computational problem.

Can we do scalable BNP?

Frequentists are great at finding simple procedures (e.g. $[\mathbf{X}'\mathbf{X}]^{-1}\mathbf{X}'y$) and showing that they will 'work' regardless of the true DGP.

(DGP = Data Generating Process)

This is classical 'distribution free' nonparametrics.

1: Find some statistic that is useful regardless of DGP.

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

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

Can we Bayesians provide something like this?

A flexible model for the DGP

$$g(\mathbf{z}) = \frac{1}{|\boldsymbol{\theta}|}\sum_{l=1}^{L}\theta_l \mathbb{1}[\mathbf{z} = \boldsymbol{\zeta}_l], \quad \theta_l \overset{iid}{\sim} \mathrm{Exp}(a)$$

After observing $\mathbf{Z} = \{\mathbf{z}_1 \ldots \mathbf{z}_n\}$, posterior has $\theta_l \sim \mathrm{Exp}(a + \mathbb{1}_{\boldsymbol{\zeta}_l \in \mathbf{Z}})$.
(say every $\mathbf{z}_i = [\mathbf{x}_i, y_i]$ is unique).

$a \to 0$ leads to $\mathrm{p}(\theta_l = 0) = 1$ for $\boldsymbol{\zeta}_l \notin \mathbf{Z}$.

$$\Rightarrow g(\mathbf{z} \mid \mathbf{Z}) = \tfrac{1}{|\boldsymbol{\theta}|}\sum_{l=1}^{L}\theta_l \mathbb{1}[\mathbf{z} = \mathbf{z}_l], \quad \theta_i \sim \mathrm{Exp}(1)$$

This is just the Bayesian bootstrap.

Ferguson 1973, Rubin 1981

Example: **Ordinary Least Squares**

Population OLS is a posterior functional

$$\boldsymbol{\beta} = (\mathbf{X}'\boldsymbol{\Theta}\mathbf{X})^{-1}\mathbf{X}'\boldsymbol{\Theta}\mathbf{y}$$

where $\boldsymbol{\Theta} = \mathrm{diag}(\boldsymbol{\theta})$. This is a random variable. (sample via BB)

Posterior moments for a first-order approx

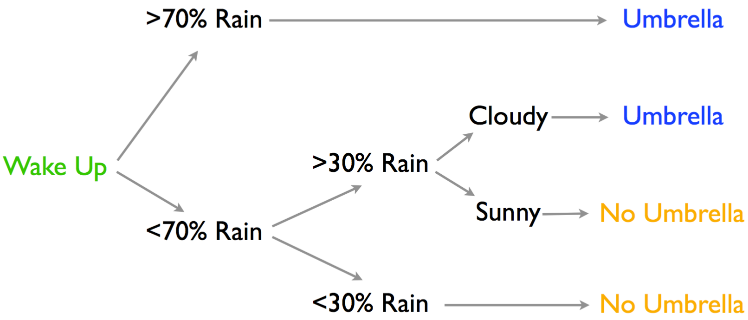
$$\tilde{\boldsymbol{\beta}} = [\mathbf{X}'\mathbf{X}]^{-1}\mathbf{X}'y + \nabla\beta\big|_{\boldsymbol{\theta}=\mathbf{1}}(\boldsymbol{\theta} - \mathbf{1})$$

e.g., $\mathrm{var}(\tilde{\boldsymbol{\beta}}) \approx (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathrm{diag}(\mathbf{e})^2\mathbf{X}'(\mathbf{X}'\mathbf{X})^{-1}$, where $e_i = y_i - \mathbf{x}_i'\hat{\boldsymbol{\beta}}$.

See Lancaster 2003 or Poirier 2011.

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

Given node $\{\mathbf{x}_i, y_i\}_{i=1}^n$ and DGP weights $\boldsymbol{\theta}$, find x to minimize

$$|\boldsymbol{\theta}|\sigma^2(x,\boldsymbol{\theta}) = \sum_{k\in\mathrm{left}(x)} \theta_k(y_k - \mu_{\mathrm{left}(x)})^2 + \sum_{k\in\mathrm{right}(x)} \theta_k(y_k - \mu_{\mathrm{right}(x)})^2$$

for a regression tree. Classification impurity can be Gini, etc.

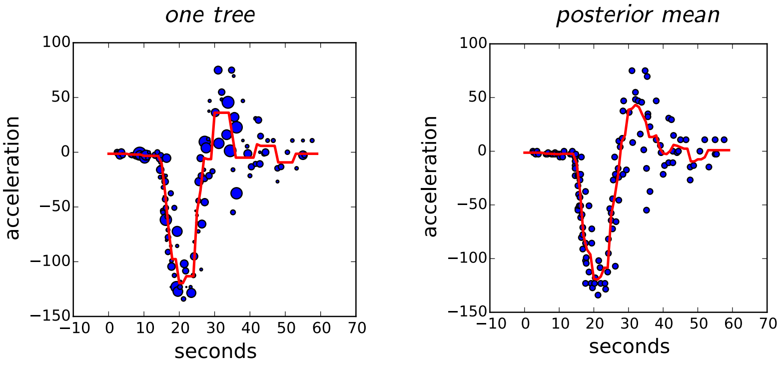
Population-CART might be a statistic we care about.

Or, in settings where greedy CART would do poorly (big p), a randomized splitting algorithm might be a better s.w.c.a.

Bayesian Forests: a posterior for CART trees

For $b = 1 \dots B$:

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



RF ≈ Bayesian forest ≈ posterior over CART fits.

Theoretical trunk stability

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

Consider the first-order approximation

$$\begin{aligned}\sigma^2(x,\boldsymbol{\theta}) \approx\ & \sigma^2(x,\mathbf{1}) + \nabla\sigma^2\big|_{\boldsymbol{\theta}=\mathbf{1}}(\boldsymbol{\theta}-\mathbf{1}) \\ & = \tfrac{1}{n}\textstyle\sum_i \theta_i \left[y_i - \bar{y}_i(x)\right]^2\end{aligned}$$

with $\bar{y}_i(x)$ the sample mean in i's node when splitting on x.

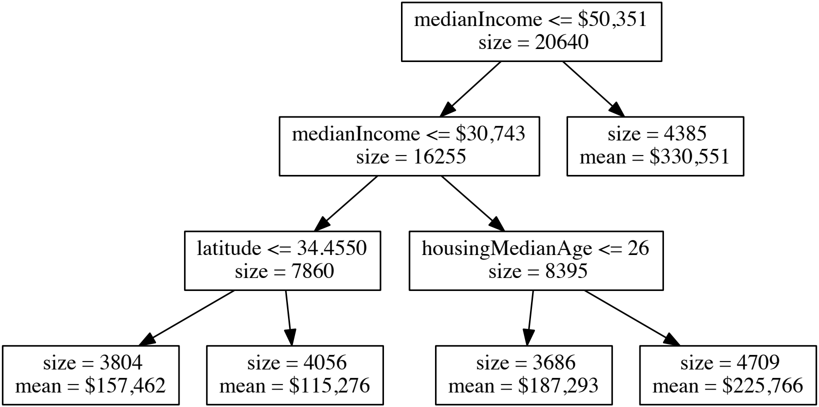
Based on this approx, we can say that for data at a given node,

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

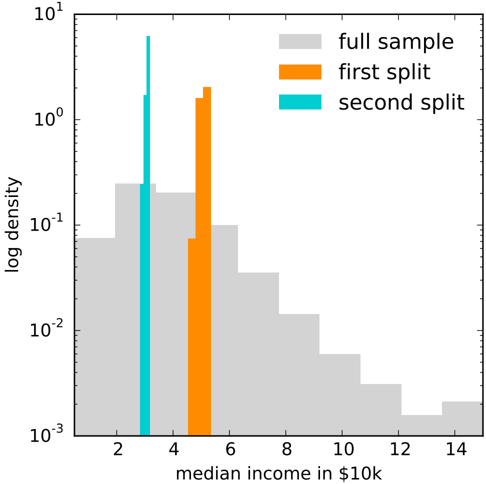
with p split locations and n observations.

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.

Empirical Bayesian Forests (EBF)

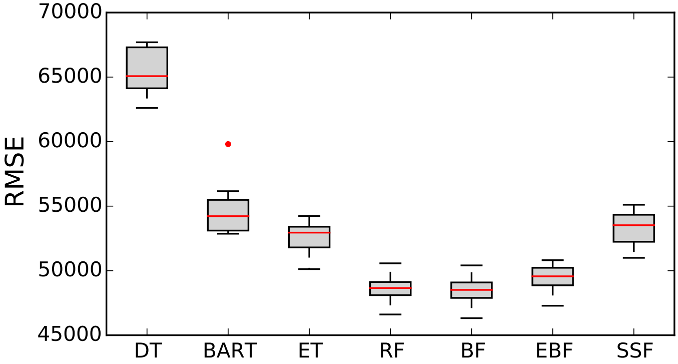
RFs are expensive. Sub-sampling hurts bad.

Instead:

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

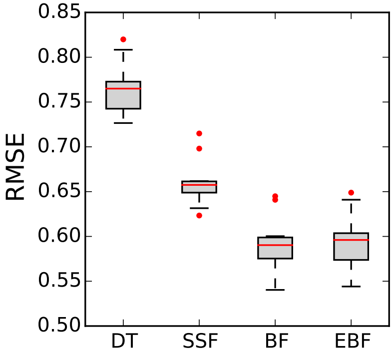
Empirical Bayes: fix plug-in estimates at high levels in a hierarchical model, focus effort at learning the hard bits.

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



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

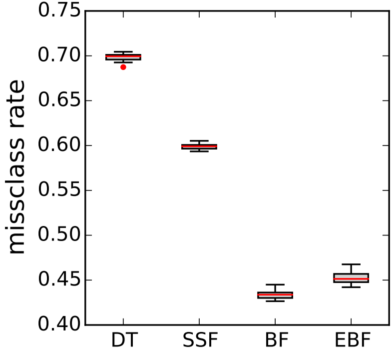
EBFs work all over the place



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

Predicting wine rating from chemical profile

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

or beer choice from demographics

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?

Catching Bad Buyer Experiences at eBay

BBE: ‘not as described’, delays, etc.

p(BBE) is an input to search rankings.

Best way to improve prediction is more data.

EBFs via Spark: more data in less time.

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

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

Talk to Chun-Sheng at the poster for some implementation detail.

Big Data and distribution free BNP

I think about BNP as a way to analyze (and improve) algorithms. Decouple action/prediction from the full generative process model.

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.

thanks!