Uncertainty Quantification in Machine Learning

Stefan Wager
Stanford University

DS@HEP
Fermilab, 9 May 2017

Motivating example

Suppose we are at a **weather station**, and want to estimate **air pressure** at a balloon 1 kilometer above us.

- We can conduct many **different kinds of measurements** at our weather station (but not at the balloon).
- We have access to **past (or simulated) data** from other weather stations that also includes air pressure measurements above the station.

The classical approach

Dust off the **barometric formula**. If P_0 and T_0 are pressure and temperature at ground level, ζ is the temperature gradient and $c > 0$ is a constant, then expected pressure at the balloon is:

$$\mu(h; P_0, T_0, \zeta) = P_0 \left(1 - \frac{\zeta h}{T}\right)^{c/\zeta}.$$

Given the above formula and training data, we can simply **estimate** $\hat{c}$ and then get **plug-in** pressure estimates.

The classical approach

Dust off the **barometric formula**. If P_0 and T_0 are pressure and temperature at ground level, ζ is the temperature gradient and $c > 0$ is a constant, then expected pressure at the balloon is:

$$\mu(h; P_0, T_0, \zeta) = P_0 \left(1 - \frac{\zeta h}{T}\right)^{c/\zeta}.$$

Given the above formula and training data, we can simply **estimate** $\hat{c}$ and then get **plug-in** pressure estimates.

1. What is **estimated air pressure** at h? $P_0(1 - \zeta h / T_0)^{\hat{c}/\zeta}$.

The classical approach

Dust off the **barometric formula**. If P_0 and T_0 are pressure and temperature at ground level, ζ is the temperature gradient and $c > 0$ is a constant, then expected pressure at the balloon is:

$$\mu(h; P_0, T_0, \zeta) = P_0 \left(1 - \frac{\zeta h}{T}\right)^{c/\zeta}.$$

Given the above formula and training data, we can simply **estimate** $\hat{c}$ and then get **plug-in** pressure estimates.

1. What is **estimated air pressure** at h? $P_0(1 - \zeta h / T_0)^{\hat{c}/\zeta}$.
2. What is the **uncertainty of this estimate** due to noise in the training data? Can be obtained with the **bootstrap**.
3. What about the estimated **pressure gradient** $\partial\mu/\partial h$? $-P_0(\hat{c}/T_0)(1 - \zeta h / T_0)^{\hat{c}/\zeta - 1}$.

The machine learning approach

Now, instead of just P_0, T_0 and ζ, we measure a large amount of **ground-level measurements** M; we get training data

$(M_i,\ h_i,\ Y_i) = (\text{ground-level data},\ \text{alt. of balloon},\ \text{press. at balloon})$

We then train a **black box** and get $\hat{\mu}(h;\ m) = \widehat{\mathbb{E}}\left[Y_i(h) \,\middle|\, M_i = m\right]$.
Have our questions become easier or harder to answer?

The machine learning approach

Now, instead of just P_0, T_0 and ζ, we measure a large amount of **ground-level measurements** M; we get training data

$(M_i,\ h_i,\ Y_i) = (\text{ground-level data},\ \text{alt. of balloon},\ \text{press. at balloon})$

We then train a **black box** and get $\hat{\mu}(h;\ m) = \widehat{\mathbb{E}}\left[Y_i(h) \,\middle|\, M_i = m\right]$.
Have our questions become easier or harder to answer?

1. What is **estimated air pressure** at h? $\hat{\mu}(h;\ M_i)$.
2. What is the **uncertainty of this estimate** due to noise in the training data? Could be obtained via the **bootstrap**, but computational burden + Monte Carlo errors are problems...
3. What about the estimated **pressure gradient** $\partial\mu/\partial h$? There is **no good reason** to expect $\hat{\mu}(h;\ M_i)$ to be differentiable...

The machine learning approach

Now, instead of just P_0, T_0 and ζ, we measure a large amount of **ground-level measurements** M; we get training data

$(M_i,\ h_i,\ Y_i) = (\text{ground-level data},\ \text{alt. of balloon},\ \text{press. at balloon})$

We then train a **black box** and get $\hat{\mu}(h;\ m) = \widehat{\mathbb{E}}\left[Y_i(h) \,\middle|\, M_i = m\right]$.
Have our questions become easier or harder to answer?

1. What is **estimated air pressure** at h? $\hat{\mu}(h;\ M_i)$.
2. What is the **uncertainty of this estimate** due to noise in the training data? Could be obtained via the **bootstrap**, but computational burden + Monte Carlo errors are problems...
3. What about the estimated **pressure gradient** $\partial\mu/\partial h$? There is **no good reason** to expect $\hat{\mu}(h;\ M_i)$ to be differentiable...

This talk is about how to approach 2 and 3.

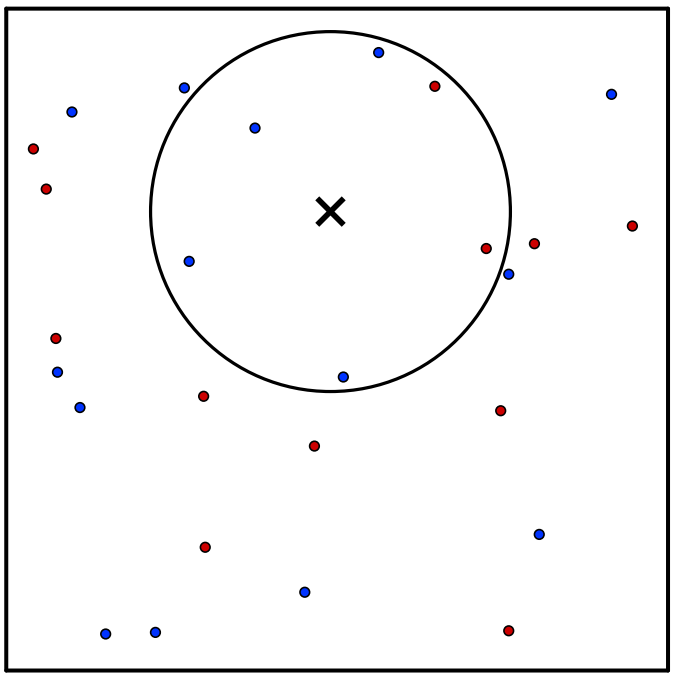
	Low-Noise Examples (a perfect system gets most examples right)	**High-noise Examples** (examples are only useful in aggregate)
Complex $f(\cdot)$	Computer vision, Natural language, ...	...
Simple $f(\cdot)$	Classical physics	Economics, Medicine, ...

	Low-Noise Examples (a perfect system gets most examples right)	**High-noise Examples** (examples are only useful in aggregate)
Complex $f(\cdot)$	Computer vision, Natural language, ...	...
Simple $f(\cdot)$	Classical physics	Economics, Medicine, ...

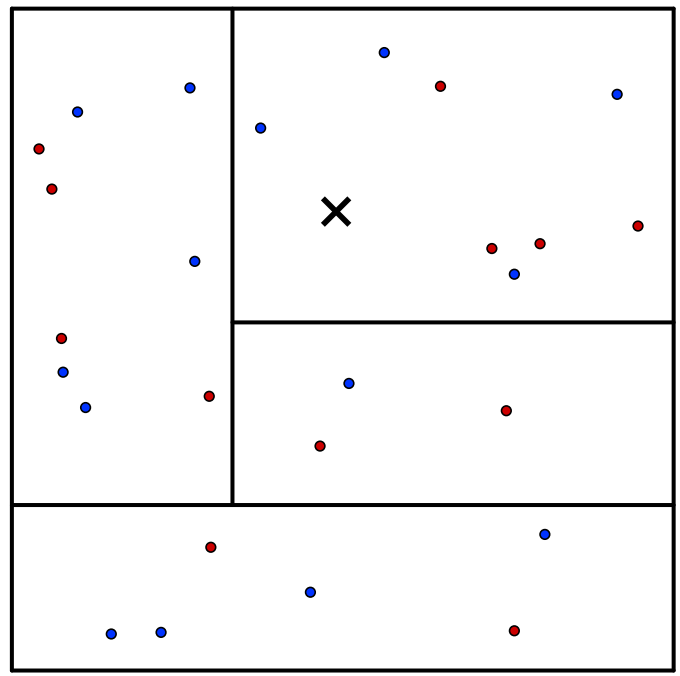
This talk is about how to approach 2 and 3 in the context of **random forests**.

Regression trees

Perhaps the simplest machine learning method is k-**nearest neighbors** regression. Letting X_i denote the pair (M_i, h_i), we set $\hat{\mu}(x) = \frac{1}{k} \sum_{\{i \in \mathcal{S}(x)\}} Y_i$, where $\mathcal{S}(x)$ consists of the k nearest neighbors to our test point x. Random forests are an attempt to improve on k-NN by making the neighborhood function **adaptive**.



k-NN neighborhood.



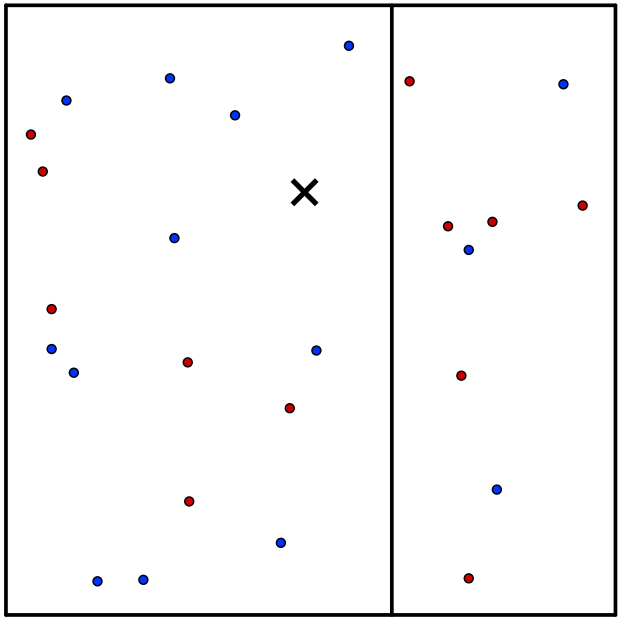
Tree-based neighborhood.

Regression trees

Trees recursively apply a **greedy splitting criterion**.

In the **regression case**, the CART (Breiman et al., 1984) is standard.

- Compute $\hat{y}$ by averaging data in left/right leaf.
- Split to minimize $\sum_i (y_i - \hat{y}_i)^2$.
- Equivalently, pick a split to **maximize the variance** $\widehat{\text{Var}}[\hat{y}_i]$.



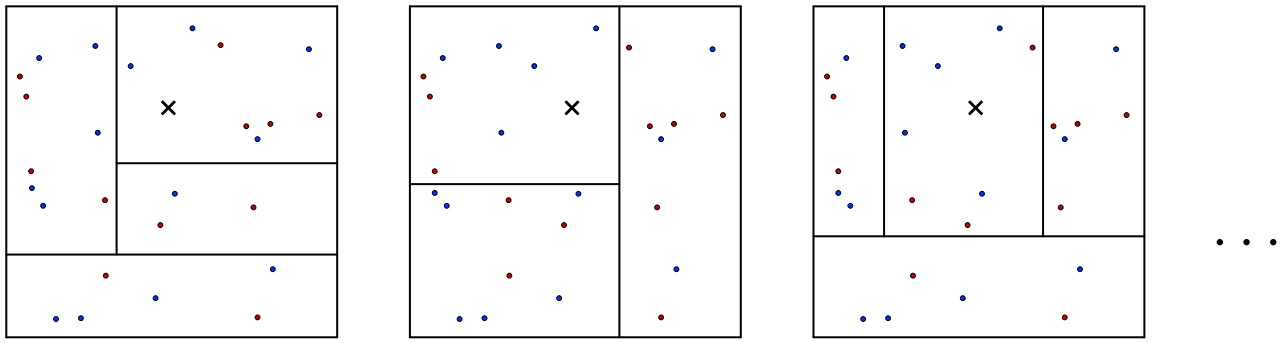
From trees to random forests (Breiman, 2001)

Suppose we have a training set $\{(X_i,\ Y_i)\}_{i=1}^n$, a test point x, and a tree predictor

$$\hat{\mu}\,(x) = T\,(x;\ \{(X_i,\ Y_i)\}_{i=1}^n)\,.$$

Random forest idea: build and average many different trees T^*:

$$\hat{\mu}\,(x) = \frac{1}{B}\sum_{b=1}^{B} T_b^*\,(x;\ \{(X_i,\ Y_i)\}_{i=1}^n)\,.$$



From trees to random forests (Breiman, 2001)

Suppose we have a training set $\{(X_i,\ Y_i)\}_{i=1}^n$, a test point x, and a tree predictor

$$\hat{\mu}\,(x) = T\,(x;\ \{(X_i,\ Y_i)\}_{i=1}^n)\,.$$

Random forest idea: build and average many different trees T^*:

$$\hat{\mu}\,(x) = \frac{1}{B}\sum_{b=1}^{B} T_b^*\,(x;\ \{(X_i,\ Y_i)\}_{i=1}^n)\,.$$

We turn T into T^* by:

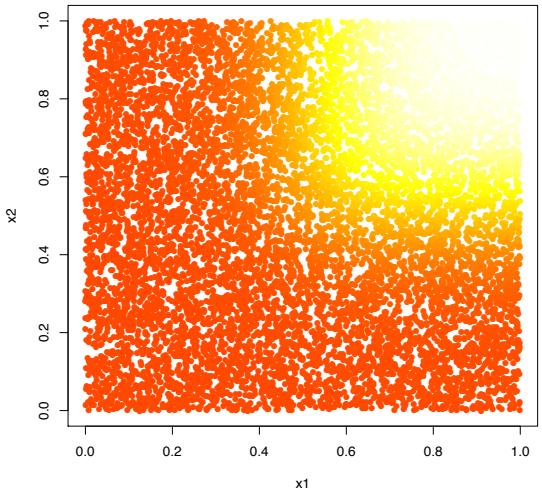
- Bagging / subsampling the training set (Breiman, 1996); this helps smooth over discontinuities (Bühlmann and Yu, 2002).
- Selecting the splitting variable at each step from m out of p randomly drawn features (Amit and Geman, 1997).

Random forests vs k-NN

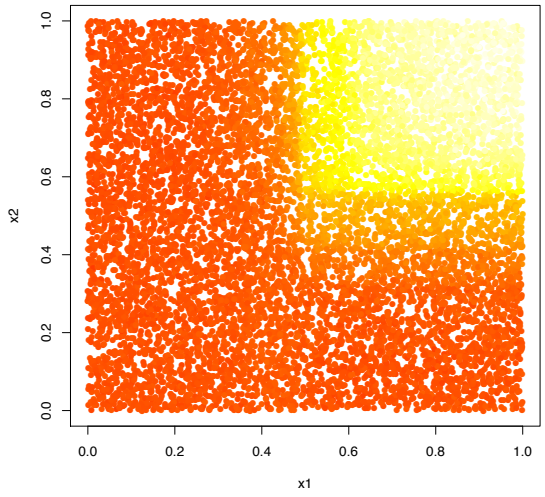
We have $n = 20$k observations whose features are distributed as $X \sim U([-1, 1]^p)$ with $p = 20$. All **the signal is concentrated along two features**.

The plots below depict $\hat{\mu}(x)$ for 10k random test examples, projected into the 2 signal dimensions.

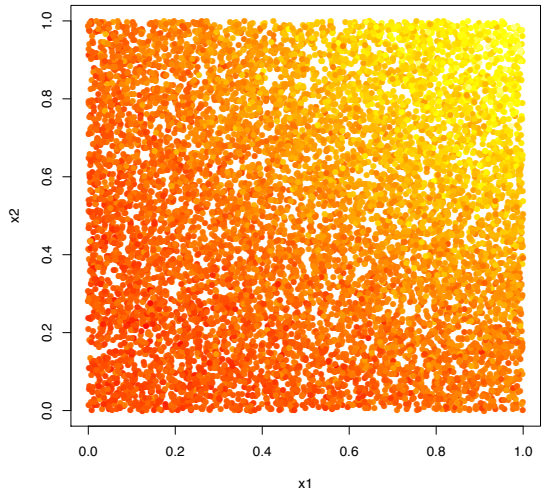
true function $\mu(x)$



random forest



non-adaptive



Uncertainty quantification for random forests

Recall that a random forest prediction is of the form

$$\hat{\mu}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b^*(x),$$

where each T_b^* is the prediction made by an individual tree. How can we quantify **sampling uncertainty**, i.e., the instability of $\hat{\mu}(x)$ due to randomness in the training data?

Uncertainty quantification for random forests

Recall that a random forest prediction is of the form

$$\hat{\mu}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b^*(x),$$

where each T_b^* is the prediction made by an individual tree. How can we quantify **sampling uncertainty**, i.e., the instability of $\hat{\mu}(x)$ due to randomness in the training data?

Idea 1: Look at the variance of the individual trees,

$$\frac{1}{B} \sum_{b=1}^{B} (T_b^*(x) - \hat{\mu}(x))^2.$$

This is **not what we want**: it estimates the sampling error of trees, not of the forest.

Uncertainty quantification for random forests

Recall that a random forest prediction is of the form

$$\hat{\mu}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b^*(x),$$

where each T_b^* is the prediction made by an individual tree. How can we quantify **sampling uncertainty**, i.e., the instability of $\hat{\mu}(x)$ due to randomness in the training data?

Idea 2: Bootstrap the whole thing. Requires **many trees**.

- For $r = 1, ..., R$ replications:
  - Draw a random half sample $\{(X_{i^*}, Y_{i^*})\}$.
  - For $b = 1, ..., B$ replications:
    - Draw a subsample of the half sample $\{(X_{i^{**}}, Y_{i^{**}})\}$.
    - Grow a tree $T_{rb}^{**}(x)$ on it.
  - Average the trees into a single forest $\hat{\mu}_r^*(x) = \frac{1}{B} \sum_{b=1}^{B} T_b^{**}(x)$.
- Consider the average of $(\hat{\mu}_r^*(x) - \hat{\mu}(x))^2$ over all boot. forests.

Bootstrapping little bags

Because a forest already involves **resampling** by construction, we might intuitively expect to be able to build bootstrap-like confidence intervals **"for free"**. For example:

- For $b = 1, ..., B$ replications:
  - Draw a **half sample** $\{(X_{i^*}, Y_{i^*})\}$.
  - Draw **2 further subsamples** $\{(X_{i^{**}}, Y_{i^{**}})\}$ form the half sample, and grow 2 trees $T^{**}_{b1}(X_i)$ using them.
- We then estimate: $\hat{\mu}(x) = \frac{1}{2B} \sum_{b=1}^{B} (T^{**}_{b1}(x) + T^{**}_{b2}(x))$.
  NB: As $B \to \infty$, this **matches the original** random forest.
- We can get at the sampling error of $\hat{\mu}(x)$ by examining the discrepancy of $T^{**}_{b1}(x)$ and $T^{**}_{b2}(x)$.

References: Sexton & Laake, 2009; Wager, Hastie & Efron, 2014; Mentch & Hooker, 2016; Athey, Tibshirani & Wager, 2017.

Bootstrapping little bags

Recall that we first forms **pairs of trees**, and set

$$\hat{\mu}(x) = \frac{1}{2B} \sum_{b=1}^{B} \left(T_{b1}^{**}(x) + T_{b2}^{**}(x) \right).$$

We then estimate the **sampling variance** $V(x)$ of $\hat{\mu}(x)$ as:

$$\widehat{V}_{between} = \frac{1}{B-1} \sum_{b=1}^{B} \left(\frac{T_{b1}^{**}(x) + T_{b2}^{**}(x)}{2} - \hat{\mu}(x) \right)^2$$

$$\widehat{V}_{within} = \frac{1}{B} \sum_{b=1}^{B} \frac{1}{2} \left(T_{b1}^{**}(x) - T_{b2}^{**}(x) \right)^2$$

$$\widehat{V} = \widehat{V}_{between} - \widehat{V}_{within} \,/\, 2.$$

$\widehat{V}_{between}$ is a bootstrap of **little forests** with two trees each; $\widehat{V}_{within}$ is used to substract out the extra noise due to **Monte Carlo error**.

Bootstrapping little bags

We then estimate the **sampling variance** $V(x)$ of $\hat{\mu}(x)$ as:

$$\widehat{V}_{between} = \frac{1}{B-1} \sum_{b=1}^{B} \left(\frac{T_{b1}^{**}(x) + T_{b2}^{**}(x)}{2} - \hat{\mu}(x) \right)^2$$

$$\widehat{V}_{within} = \frac{1}{B} \sum_{b=1}^{B} \left(T_{b1}^{**}(x) - T_{b2}^{**}(x) \right)^2 \Big/ 2$$

$$\widehat{V} = \widehat{V}_{between} - \widehat{V}_{within} \Big/ 2.$$

In the $B \to \infty$ limit, this is equivalent to the regular bootstrap.

- For finite B, this is **unbiased** for its $B \to \infty$ limit.
- However, for finite B, nothing prevents this variance estimate from being **negative**.
- **Recommendation:** Model $\widehat{V}_{between}$ as a χ^2-random variable with mean $V + \widehat{V}_{within}/2$; give V a **uniform prior** over $\mathbb{R}_+$.

Application: General Social Survey

The General Social Survey is an extensive survey, collected since 1972, that seeks to measure demographics, political views, social attitudes, etc. of the U.S. population.

Of particular interest to us is a **randomized experiment**, for which we have data between 1986 and 2010.

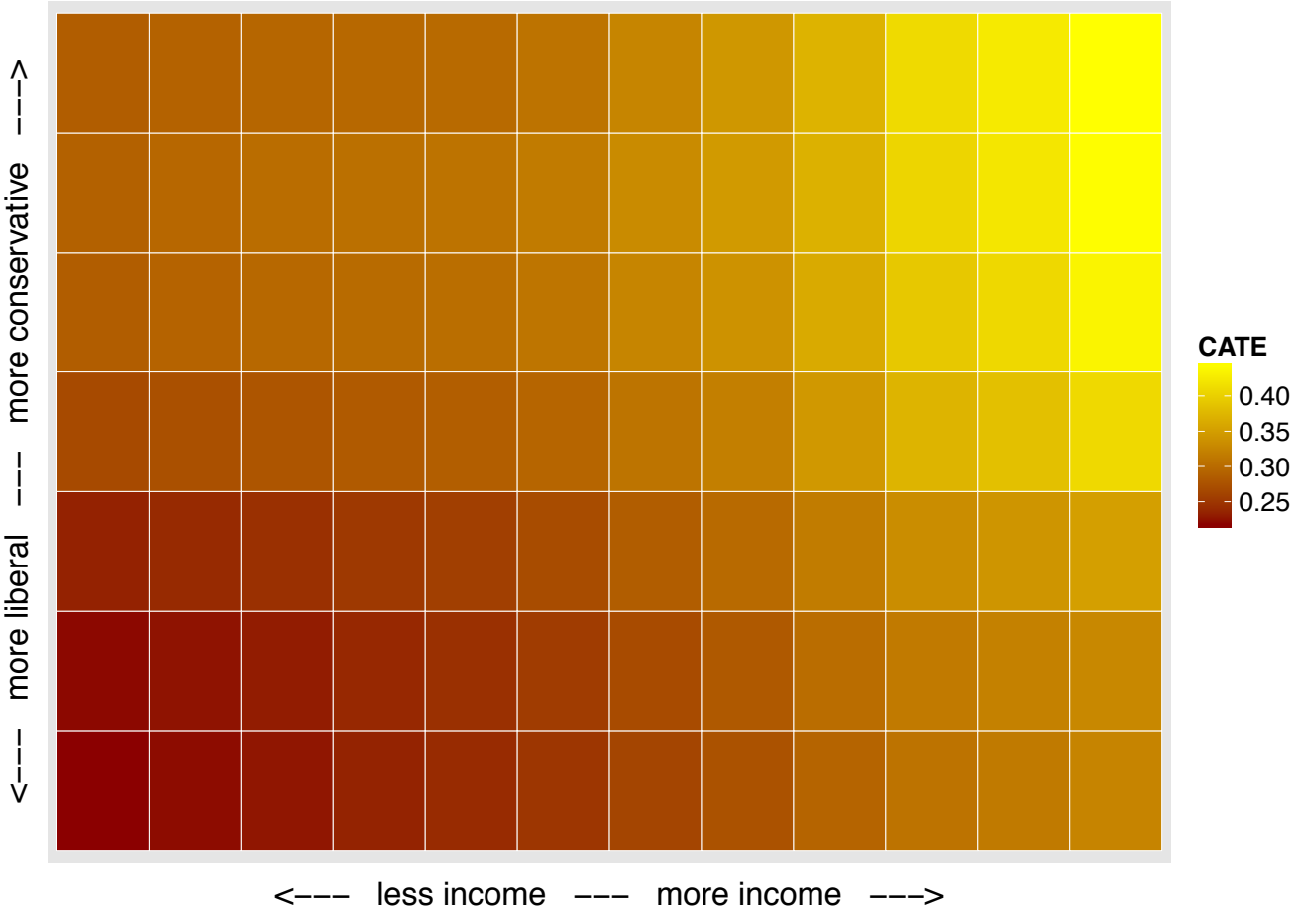
- **Question A:** Are we spending too much, too little, or about the right amount on **welfare**?
- **Question B:** Are we spending too much, too little, or about the right amount on **assistance to the poor**?

Treatment effect: how much less likely are people to answer **too much** to question B than to question A.

- We want to understand how the treatment effect depends on **covariates**: political views, income, age, hours worked, ...

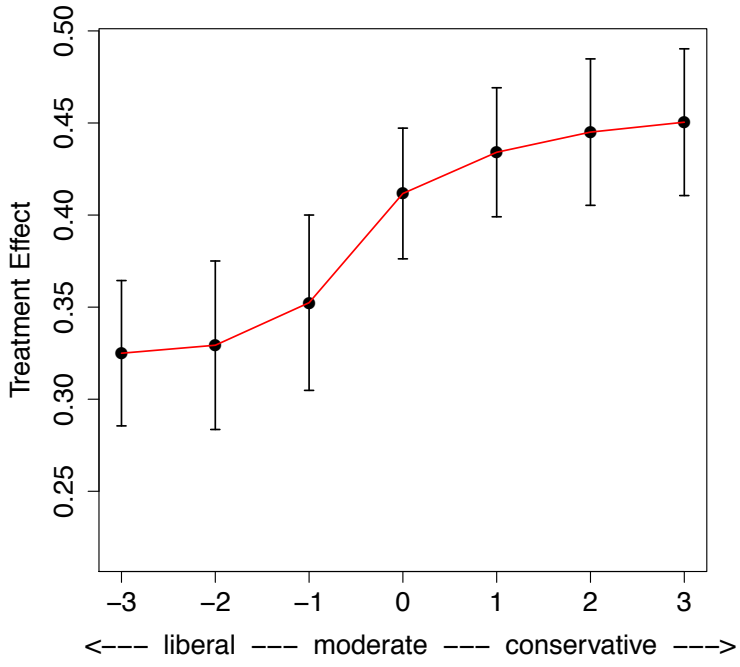
Application: General Social Survey

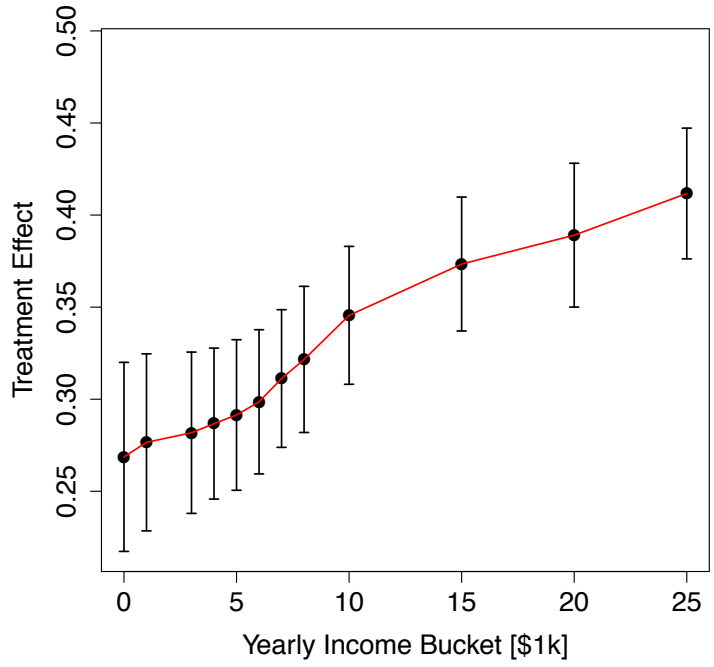
A causal forest analysis uncovers **strong treatment heterogeneity** ($n = 28,686$, $p = 12$).



Application: General Social Survey

We want to understand the difference in attitudes about **welfare** vs. **assistance to the poor**. The treatment effect measures how much likelier a respondent is to say we spend too much money on the former vs. the latter.





NB: All other covariates are set to their median values.

Back to the beginning

We have a collection of training data

$(M_i,\ h_i,\ Y_i) = (\text{ground-level data, alt. of balloon, press. at balloon})$

We then train a **black box** and get $\hat{\mu}(h;\ m) = \widehat{\mathbb{E}}\left[Y_i(h) \,\middle|\, M_i = m\right]$.
We're interested in the following questions:

1. What is **estimated air pressure** at h? $\hat{\mu}(h;\ M_i)$.
2. What is the **uncertainty of this estimate** due to noise in the training data? Leverage **internal randomness** of the black box to emulate the bootstrap.
3. What about the **pressure gradient** $\Delta(h;\ m) = \partial\mu(h;\ m)/\partial h$?

There is no good reason to expect $\hat{\mu}(h;\ M_i)$ to be **differentiable**, so the following is usually a **bad idea**:

$$\widehat{\Delta}(h;\ m) = \varepsilon^{-1}\left(\hat{\mu}(h + \varepsilon/2;\ m) - \hat{\mu}(h - \varepsilon/2;\ m)\right).$$

Random forests as local estimation

For regression, natural to write a forest as an **average of trees**:

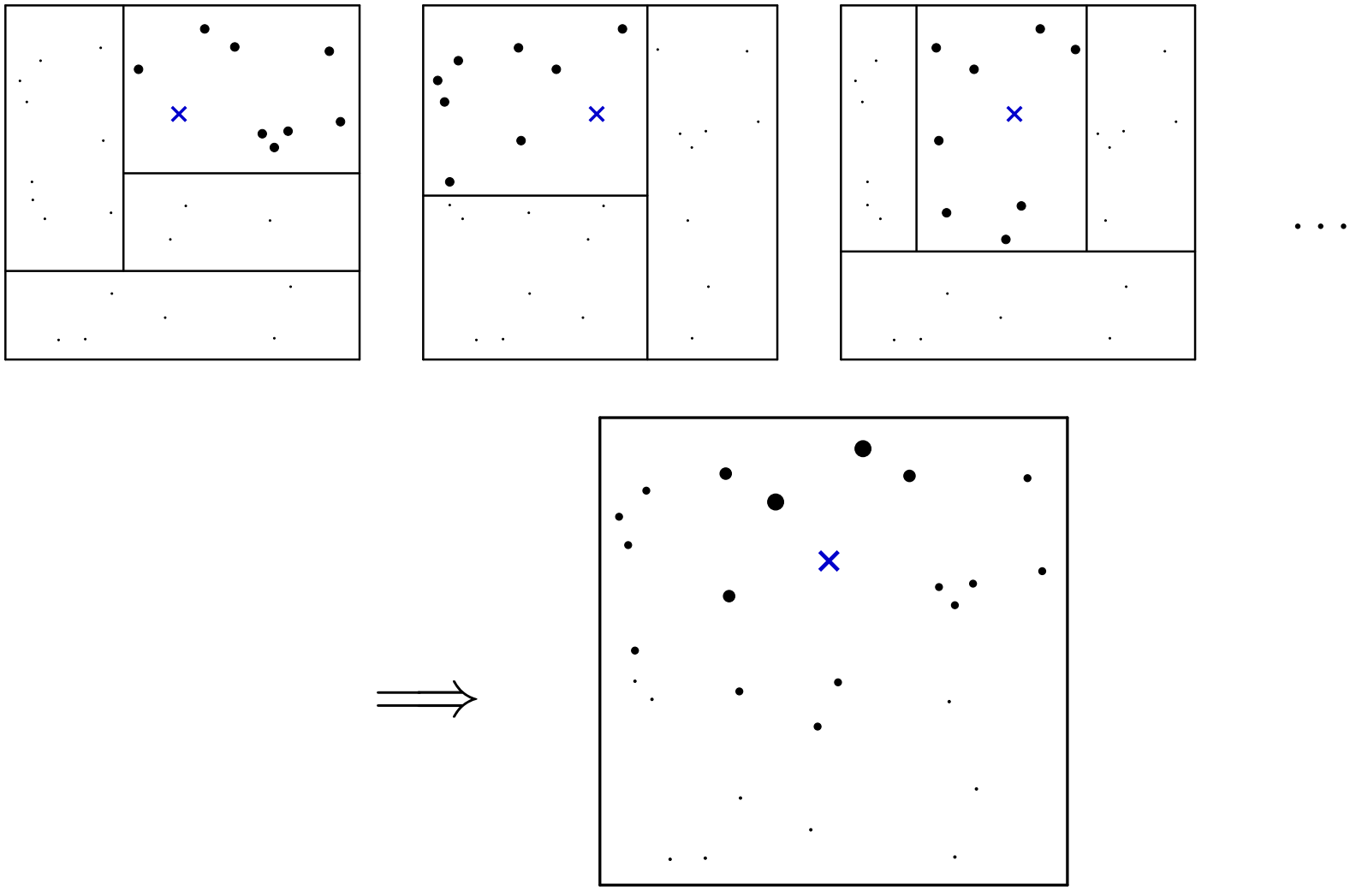
$$\hat{\mu}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b^*(x; \{(X_i, Y_i)\}_{i=1}^n).$$

A helpful alternative perspective is to view forests as weighting:

$$\hat{\mu}(x) = \frac{1}{B} \sum_{b=1}^{B} \sum_{i=1}^{n} Y_i \frac{1(Y_i \in L_b(x))}{|L_b(x)|} = \sum_{i=1}^{n} Y_i \underbrace{\frac{1}{B} \sum_{b=1}^{B} \frac{1(Y_i \in L_b(x))}{|L_b(x)|}}_{\alpha_i(x)}.$$

In other words, we understand random forests as a **data-adaptive "kernel"** with weights $\alpha_i(x)$.

The random forest kernel



Forests induce a kernel via **averaging tree-based neighborhoods**.
This idea was used by Meinshausen (2006) for quantile regression.

Example: Estimating pressure gradients

We can use this kernel-based idea to derive a **forest-based algorithm** for **local derivative learning**:

- Use the ground-level measurements M_i to build a forest, yielding **weights** $\alpha_i(m)$ (don't use height h_i).
  - We can use these weights to to make predictions at m, $\hat{\mu}(m) = \sum_{i=1}^n \alpha_i(m) Y_i$; useful, but not yet what we want...
- Instead, given a neighborhood function α_i, use a **white-box** derivative estimator:

$$\widehat{\Delta}(h;\, m) = \frac{\sum_{i=1}^n \alpha_i(m) \left(1 - \lambda\,(h_i - h)^2\right)_+ (Y_i - \hat{\mu}(m))\,(h_i - h)}{\sum_{i=1}^n \alpha_i(m) \left(1 - \lambda\,(h_i - h)^2\right)_+ (h_i - h)^2}.$$

This is just **local linear regression** using a **forest-based weighting** function; $\lambda^{-1/2}$ is a height bandwidth.

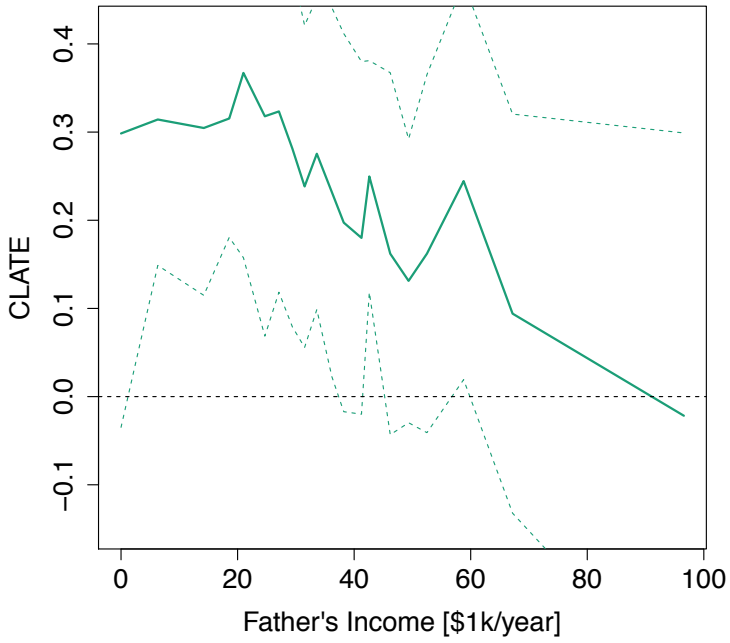
Application: Family size

Angrist and Evans (1998) study the **effect of family size** on women's **labor market** outcomes, using a sample of $n = 334,535$ married mothers with at least 2 children (1980 census data). Understanding heterogeneity can guide policy.

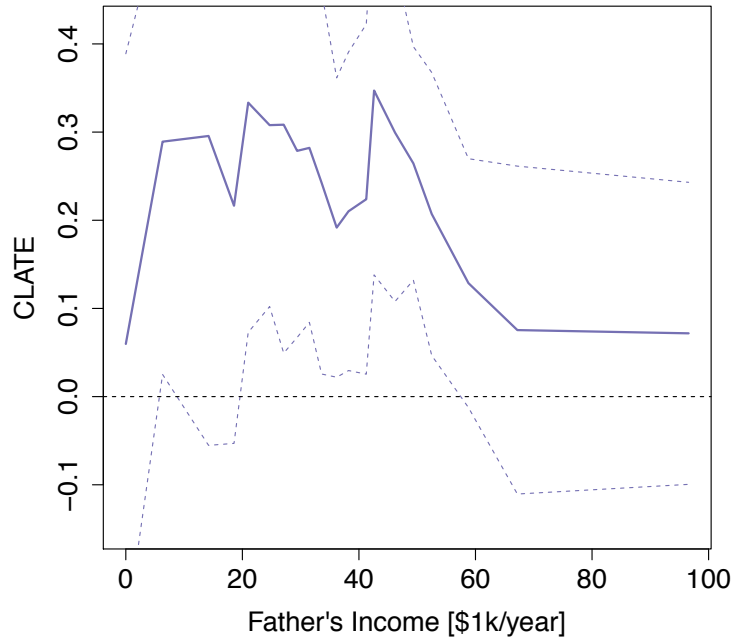
- **Outcome:** Mother's labor market participation, income.
- **Treatment:** Mother had more than two kids.
- **Instrument:** First two kids were of same sex.
- First stage effect of instrument on treatment: 0.016
- Reduced form effect of having two same sex children on mother's probability of work: 0.0021
- Local average treatment effect estimates on the **probability of labor market participation**, $\tau \in (0.14 \pm 0.054)$.

The instrument has a **weak effect** relative to other predictors; a raw machine learning method would simply ignore it.

Application: Family size



Mother 18 years old at 1st birth



Mother 22 years old at 1st birth

We extend this analysis by fitting **heterogeneity** on several covariates, including the mothers **age** at the birth of her first child, her age at census time, her years of **education** and her **race** (black, hispanic, other), as well as the **fathers income**.

- The display also shows 95% **confidence intervals**.

Closing thoughts: Using machine learning for science

Traditional tools for **uncertainty quantification** are unwieldy with machine learning methods: They are computationally impractical, and can confuse sampling error with Monte Carlo error.

- However, we can exploit **embedded randomness** in the algorithm to emulate the bootstrap, often at very low cost.

Pure black-box methods may not accurately answer **domain-specific** questions.

- However, it is often possible to **white-box** the parts of the problem we care about, and **black-box** the rest.

Thanks!