BIG DATA and MACHINE LEARNING in Econometrics

Chapter 3: Tree-Based Methods and Random Forests

Anna Simoni[2]

[2]CREST, CNRS - ENSAE and École Polytechnique

Outline

Introduction

- Based on decision trees : the decision at the end of the process is a choice about how to classify/predict the observation.
- Popularized by the work of Breiman et al. (1984) and now known as CART = classification and regression trees.
- Idea : Partition the feature / covariables space into a set of rectangles and then fit a simple model (constant) in each one.
- So the estimated function is the average of outcomes falling in this rectangle.
- Recursive binary partitions, i.e. sequentially we choose variable and corresponding split which achieve best fit until some stopping criterion is reached.
- Used for Regression but also for classification (with different criteria).

Regression Trees. I

Our data consists of (x_i, y_i) for $i = 1, 2, \ldots, N$, with $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$. The process of building a regression tree consists of two steps (CART method) :

1. divide the predictor space (*i.e.* the set of possible values for $X = (X_1, X_2, \ldots, X_p)$) into M distinct and non-overlapping regions, $R_1, R_2, \ldots, R_M$.

2. For every observation that falls into the region R_m, make the same prediction, which is simply the mean of the y_i for the i's such that $x_i \in R_m$. That is :

$$\hat{f}(x) = \sum_{m=1}^{M} c_m I(x \in R_m). \tag{1}$$

If we adopt as our criterion minimization of the sum of squares $\sum_i (y_i - f(x_i))^2$, then the best c_m is the average of y_i in region R_m.

For instance, suppose that in Step 1 we obtain two regions, R1 and R2, and that $Ave(y_i|x_i \in R_1) = 10$, while $Ave(y_i|x_i \in R_2) = 20$. Then for a given observation $X = x$, if $x \in R_1$ we will predict a value of 10, and if $x \in R_2$ we will predict a value of 20.

Regression Trees. II

Now, let's discuss Step 1.

- Consider first the case $p = 2$:
  - we first split at $X_1 = t_1$.
  - Then the region $X_1 \leq t_1$ is split at $X_2 = t_2$ and the region $X_1 > t_1$ is split at $X_1 = t_3$.
  - Finally, the region $X_1 > t_3$ is split at $X_2 = t_4$.
  - The result of this process is a partition into the five regions $R_1, R_2, \ldots, R_5$.

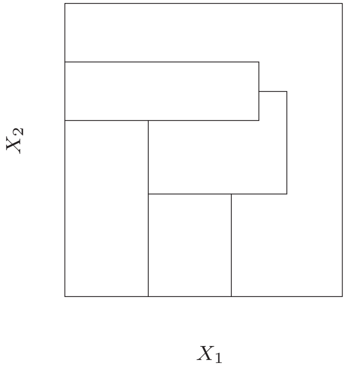
This same model can be represented by a binary tree :

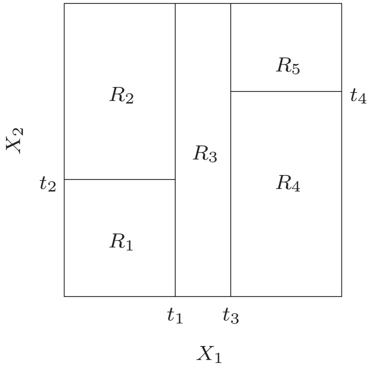
- the full dataset sits at the top of the tree ;
- observations satisfying the condition at each junction are assigned to the left branch, and the others to the right branch ;
- the terminal nodes or leaves of the tree correspond to the regions $R_1, R_2, \ldots, R_5$.

Regression Trees. III

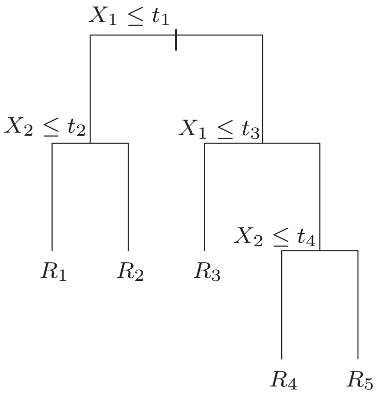
- A key advantage of the recursive binary tree is its interpretability : the feature space partition is fully described by a single tree.
- With more than 2 covariates, partitions like that in the figure are difficult to draw, but the binary tree representation works in the same way.
- This representation is important because it stratifies the population into strata (to take into account heterogeneity).

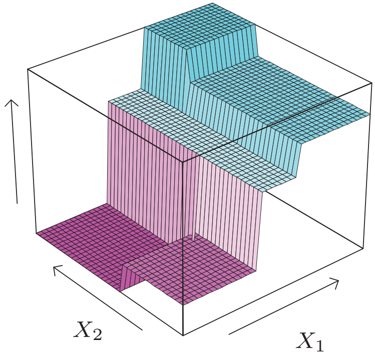
Regression Trees. IV





Regression Trees. V





Regression Trees. VI

Now, let's discuss Step 1 for a general p.

- Finding the best binary partition in terms of minimum sum of squares is generally computationally infeasible. So, we proceed with a different algorithm.

- Starting with all of the data, consider a splitting variable j and split point s, and define the pair of half-planes

$$R_1(j,s) = \{X | X_j \leq s\} \qquad \text{and} \qquad R_2(j,s) = \{X | X_j > s\}.$$

- Then we seek the splitting variable j and split point s that solve

$$\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right] \tag{2}$$

- For any choice j and s, the inner minimization is solved by

$$\hat{c}_1 = Ave(y_i | x_i \in R_1(j,s)) \qquad \text{and} \qquad \hat{c}_2 = Ave(y_i | x_i \in R_2(j,s)) \tag{3}$$

- Having found the best split, we partition the data into the two resulting regions and repeat the splitting process on each of the two regions.

- Then this process is repeated on all of the resulting regions.

Regression Trees. VII

How large should we grow the tree ?

- a very large tree might overfit the data, while a small tree might not capture the important structure.
- Tree size is a tuning parameter governing the model's complexity, and the optimal tree size should be adaptively chosen from the data.

Cost-complexity pruning to choose *tree size*. The ideas is the following :

- Define a subtree $T \subset T_0$ to be any tree that can be obtained by pruning T_0 (*i.e.* by collapsing any number of its internal (non-terminal) nodes).
- Index terminal nodes by m and the corresponding region by R_m.
- Let $|T|$ denote the number of terminal nodes in T.

Regression Trees. VIII

- Let

$$N_m = \sharp\{x_i \in R_m\}, \qquad \hat{c}_m = \frac{1}{N_m}\sum_{x_i \in R_m} y_i, \qquad Q_m(T) = \frac{1}{N_m}\sum_{x_i \in R_m} (y_i - \hat{c}_m)^2.$$

- Define the cost complexity criterion as :

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|.$$

- The idea is to find $\forall\alpha$ the subtree $T_\alpha \subset T_0$ to minimize $C_\alpha(T)$.
- The tuning parameter α governs the tradeoff between *tree size* and its *goodness of fit to the data* : large $\alpha \Rightarrow$ smaller trees T_α, and conversely for smaller α.
- Estimation of α is achieved by five- or tenfold cross-validation : choose the value α to minimize the cross-validated sum of squares. The final tree is T_α.

Regression Trees. IX

Algorithm 8.1 *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results for each value of α, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Regression Trees. X

Some Issues with the Trees :

- One major problem with trees is their high variance : a small change in the data can result in a very different series of splits, making interpretation precarious. This is mostly due to the hierarchical nature of the process. **Bagging** averages many trees to reduce this variance.

Classification Trees. I

If the target is a classification outcome taking values $1, 2, \ldots, K$, the only changes needed in the tree algorithm pertain to the criteria for splitting nodes.

- In a node m, representing a region R_m with N_m observations, let

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{i \in R_m} I(y_i = k)$$

  the proportion of class k observations in node m.

- We classify the observations in node m to class $k(m) = \arg\max_k \hat{p}_{mk}$, the majority class in node m.

- We can use three types of criteria to split nodes :

  Misclassification error : $\frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}$.

  Gini index : $\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$.

  Cross-entropy or deviance : $- \sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$.

Classification Trees. II

The Gini index can be interpreted in two interesting ways :

- Rather than classify observations to the majority class in the node, we could classify them to class k with probability $\hat{p}_{mk}$. Then the training error rate of this rule in the node is
$$\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} .$$

- Alternatively, if we code each observation as 1 for class k and 0 otherwise, the variance over the node of this $0-1$ response is $\hat{p}_{mk}(1-\hat{p}_{mk})$. Summing over classes k again gives the Gini index.

Advantages and Disadvantages of Trees

Advantages and Disadvantages of Trees w.r.t. classical linear regression and classification :

+ Trees are very easy to explain, even easier than linear regression.
+ Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches.
+ Trees can be displayed graphically, and are easily interpreted.
+ Trees can easily handle qualitative predictors without the need to create dummy variables.
- Trees generally have poorer prediction accuracy than regression and classification approaches. Solution : to "prune" the tree by imposing a cost for complexity (= $\sharp$ of terminal nodes).

Bagging. I

Bagging or bootstrap aggregation is a technique for reducing the variance of an estimated prediction function. Consider the regression problem.

- Suppose we fit a model to our training data $Z = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ and obtain $\hat{f}(x)$ at input x.

- Bootstrap aggregation or bagging averages this prediction over a collection of bootstrap samples $\Rightarrow$ it reduces its variance.

- For each bootstrap sample Z^{*b}, $b = 1, 2, \ldots, B$, we fit our model, giving prediction $\hat{f}^{*b}(x)$. The bagging estimate is defined by

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x).$$

- Bagging works especially well for high-variance, low-bias procedures, such as trees.

Bagging. II

- Denote by $\hat{P}$ the empirical distribution putting equal probability $\frac{1}{N}$ on each of the data points (x_i, y_i). The *true bagging estimate* is defined by :

$$\mathbf{E}_{\hat{P}} \hat{f}^*(x),$$

  where $Z^* = \{(x_1^*, y_1^*), \ldots, (x_N^*, y_N^*)\}$ and each $(x_i^*, y_i^*) \sim \hat{P}$.

- Hence, $\hat{f}_{bag}(x)$ is a Monte Carlo estimate of the true bagging estimate, approaching it as $B \to \infty$.

- Bagging is useful with regression tree (since trees have low bias, if grown sufficiently deep, and are noisy), where $\hat{f}(x)$ denotes the tree's prediction at input vector x.

- Each bootstrap tree will typically involve different features than the original, and might have a different number of terminal nodes.

- The bagged estimate is the average prediction at x from these B trees.

- The bias of bagged trees is the same as that of the individual trees, and the only possible improvement is through variance reduction.

Bagging. III

For Classification :

- Suppose our tree produces a classifier $\hat{G}(x)$for a K-class response.
- It is useful to consider an underlying indicator-vector function $\hat{f}(x)$, with value a single one and $K-1$ 0s, such that $\hat{G}(x) = \arg\max_k \hat{f}(x)$.
- Then the bagged estimate $\hat{f}_{bag}(x)$ is a K-vector

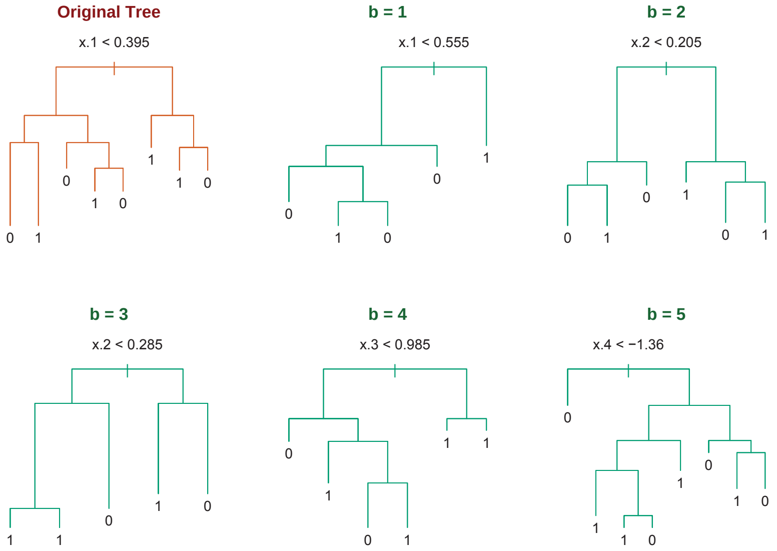
$$[p_1(x), p_2(x), \ldots, p_K(x)],$$

with $p_k(x)$ equal to the proportion of trees predicting class k at x.

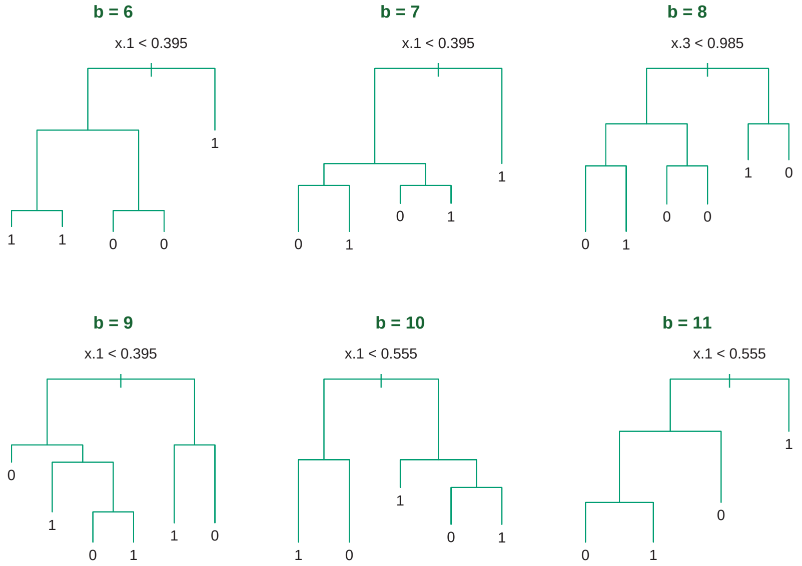
- The bagged classifier selects the class with the most ”votes” from the B trees :

$$\hat{G}_{bag}(x) = \arg\max_k \hat{f}_{bag}(x).$$

Bagging trees on simulated dataset



Bagging trees on simulated dataset



Outline

Random Forests. I

- Random forests (Breiman, 2001) is a substantial modification of Bagging that builds a large collection of de-correlated trees, and then averages them.
- The idea in random forests is to improve the variance reduction of bagging by reducing the correlation between the trees, without increasing the variance too much.
- Remark : an average of B *i.i.d.* random variables, each with variance σ^2, has variance $\frac{1}{B}\sigma^2$. If the variables are simply *i.d.* (identically distributed, but not necessarily independent) with positive pairwise correlation ρ, the variance of the average is

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

Random Forests. II

- The improvement of the variance is achieved through random selection of the input variables :
  - when growing a tree on a bootstrapped dataset :

    before each split, select $m \leq p$ of the covariates at random as candidates for splitting.

  - Denote Θ_b the characteristics of the b^{th} random forest tree : split variables, cutpoints at each node, and terminal-node values.
  - After B such trees $\{T(x;\Theta_b)\}_{b=1}^{B}$ are grown, the random forest (regression) predictor is

$$\hat{f}_{rf}^{B} = \frac{1}{B}\sum_{b=1}^{B} T(x;\Theta_b) \tag{4}$$

- Intuitively, reducing m will reduce the correlation between any pair of trees in the ensemble, and hence reduce the variance of the average.

Random Forests. III

Algorithm : Random forest

```
1  For $b = 1$ to $B$ :
    (a) Draw a bootstrap sample $Z^*$ of size $N$ from the training data.
    (b) Grow a random-forest tree $T_b$ to the bootstrapped data, by recursively
        repeating the following steps for each terminal node of the tree, until the
        minimum node size $n_{min}$ is reached.
          i. Select $m$ variables at random from the $p$ variables.
         ii. Pick the best variable/split-point among the $m$.
        iii. Split the node into two daughter nodes.
2  Output the ensemble of trees $\{T_b\}_{b=1}^{B}$.
```

To make a prediction at a new point x :

$$\hat{f}_{rf}^{B}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x).$$

Details of Random Forests. I

- The parameter m and the node size are tuning parameters. For regression, the default value for m is $\lfloor \frac{p}{3} \rfloor$ and the minimum node size is five. In practice the best values for these parameters will depend on the problem.
- Random forests can alternatively use out-of-bag (OOB) sample :

  For each observation $z_i = (x_i, y_i)$, construct its random forest predictor by averaging only those trees corresponding to bootstrap samples in which z_i did not appear.

Outline

Treatment Estimation. I

- n *i.i.d.* training examples : $X_i \in [0,1]^d$, $Y_i \in \mathbb{R}$ and a treatment indicator $W_i \in \{0,1\}$, $i = 1, \ldots, n$.

- Potential outcome framework (Neyman (1923), Rubin (1974)) :
  - $Y_i^{(1)}$ and $Y_i^{(0)}$ = response the i-th subject would have experienced with and without the treatment, respectively ;
  - treatment effect at x :

$$\tau(x) = \mathbf{E}\left[Y_i^{(1)} - Y_i^{(0)} | X_i = x\right] ; \qquad (5)$$

  - goal : to estimate $\tau(x)$;
  - main difficulty : we can only ever observe one of the two potential outcomes $Y_i^{(0)}$, $Y_i^{(1)}$ for a given training example, and so cannot directly train machine learning methods on differences of the form $Y_i^{(1)} - Y_i^{(0)}$.

- In general, we cannot estimate $\tau(x)$ simply from the observed data (X_i, Y_i, W_i) without further restrictions on the DGP $\Rightarrow$ assume unconfoundedness :

$$\left\{Y_i^{(0)}, Y_i^{(1)}\right\} \perp\!\!\!\perp W_i | X_i. \qquad (6)$$

Treatment Estimation. II

Under unconfoundedness assumption :

$$\tau(x) = \mathbf{E}\left[Y_i\left(\frac{W_i}{e(x)} - \frac{1-W_i}{1-e(x)}\right)|X_i = x\right], \qquad \text{where } e(x) = \mathbf{E}[W_i|X_i = x]$$

is the propensity of receiving treatment at x.

Under regularity assumptions, causal forests can use the unconfoundedness assumption (6) to achieve consistency without needing to explicitly estimate the propensity $e(x)$.

Causal trees and causal Forests. I

(Wager and Athey, 2017)

- Tree-based methods (*e.g.* regression trees and forests) seek to find training examples that are close to x where closeness is defined with respect to a decision tree, and the closest points to x are those that fall in the same leaf as it.
- Then, given a test point x, we evaluate the prediction $\hat{f}(x)$ by identifying the leaves R_j containing x and average the Y_i corresponding to the X_i belonging to theses leaves.
- This strategy is well-motivated if we believe that each leaf R_j containing x is small enough that the responses Y_i inside the leaf are roughly identically distributed. (For simplicity from now on denote by $R(x)$ the leaf containing x).
- In the context of **causal trees**, we analogously want to think of the leaves as small enough that the (Y_i, W_i) pairs corresponding to the indices i for which $i \in R(x)$ act as though they had come from a randomized experiment.
- Then, the treatment effect $\forall x \in R$ can be estimated as :

$$\hat{\tau}(x) = \frac{1}{|\{i : W_i = 1, X_i \in R\}|} \sum_{\{i:\, W_i=1, X_i \in R\}} Y_i - \frac{1}{|\{i : W_i = 0, X_i \in R\}|} \sum_{\{i:\, W_i=0, X_i \in R\}} Y_i . \tag{7}$$

Causal trees and causal Forests. II

- Given this procedure for generating a single causal tree, a causal forest generates an ensemble of B such trees, each of which outputs an estimate $\hat{\tau}_b(x)$.
- Then, the predictions are averaged :

$$\hat{\tau}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{\tau}_b(x).$$

- It is important that he individual causal trees in the forest are built using random subsamples of s training examples, where $s/n \ll 1$.
- This aggregation scheme helps reduce variance.
- For inference (under some conditions) :

$$\frac{\hat{\tau}(x) - \tau(x)}{\sqrt{Var(\hat{\tau}(x))}} \rightarrow^{d} \mathcal{N}(0,1),$$

where the variance can be estimated by

$$\hat{V}_{IJ}(x) = \frac{n-1}{n}\left(\frac{n}{n-s}\right)^{2}\sum_{i=1}^{n} Cov_{*}\left(\hat{\tau}_b^{*}(x), N_{ib}^{*}\right)^{2},$$

where $\hat{\tau}_b^{*}(x)$= treatment effect estimate given by the b-th tree and $N_{ib}^{*} \in \{0,1\}$ indicates whether the i-th training example was used for the b-th tree.

Causal trees and causal Forests. III

PROCEDURE 1 : **DOUBLE-SAMPLE TREES**.

- Split the available training data into 2 parts : one half for estimating the desired response inside each leaf, and another half for placing splits.

- Input : n training examples of the form (X_i, Y_i) for regression trees or (X_i, Y_i, W_i) for causal trees. A minimum leaf size k.

  1. Draw a random subsample of size s from $\{1, \ldots, n\}$ without replacement, and then divide it into two disjoint sets of size $|\mathcal{I}| = \lfloor \frac{s}{2} \rfloor$ and $|\mathcal{J}| = \lceil \frac{s}{2} \rceil$.
  2. Grow a tree via recursive partitioning. The splits are chosen using any data from the $\mathcal{J}$ sample and $X-$ or $W-$observations from the $\mathcal{I}$ sample, but without using $Y-$observations from the $\mathcal{I}$ sample.
  3. Estimate leaf-wise responses using only the $\mathcal{I}$-sample observations.

Causal trees and causal Forests. IV

PROCEDURE 1 : **DOUBLE-SAMPLE TREES** (cont'd).

- Double-sample regression trees make predictions $\hat{f}(x)$ using (1) on the leaf containing x, only using the $\mathscr{I}$-sample observations. The splitting criteria is the standard for CART regression trees (minimizing MSE of predictions). Splits are restricted so that each leaf of the tree must contain $\geq k$ $\mathscr{I}$-sample observations.

- Double-sample causal trees are defined similarly, except that for prediction we estimate $\hat{\tau}(x)$ using (7) on the $\mathscr{I}$-sample. The splits of the tree are chosen by maximizing the variance of $\hat{\tau}(X_i)$ for $i \in \mathscr{J}$. Each leaf of the tree must contain $\geq k$ or more $\mathscr{I}$-sample observations of each treatment class.

Double-sample trees eliminates bias (and thus enable centered confidence intervals) and can improve upon standard random forests in terms of MSE.

Causal trees and causal Forests. V

PROCEDURE 2 : **PROPENSITY TREES**.

Propensity trees use only the treatment assignment indicator W_i to place splits, and save the responses Y_i for estimating τ.

Input : n training examples (X_i, Y_i, W_i), where X_i are features, Y_i is the response, and W_i is the treatment assignment. A minimum leaf size k.

1. Draw a random subsample $\mathcal{I} \in \{1, \ldots, n\}$ of size $|\mathcal{I}| = s$ (no replacement).
2. Train a classification tree using sample $\mathcal{I}$ where the outcome is the treatment assignment, i.e., on the (X_i, W_i) pairs with $i \in \mathcal{I}$. Each leaf of the tree must have k or more observations of each treatment class.
3. Estimate $\tau(x)$ using (7) on the leaf containing x.

In step 2, the splits are chosen by optimizing, *e.g.*, the Gini criterion.

Outline

Generalized Random Forests. I

Athey, Tibshirani and Wager (2017)

- Goal : to generalize Breiman's random forests, and to develop a forest-based method for estimating any quantity $\theta(x)$ identified via local moment conditions (data are $(O, X) \in \mathscr{O} \times \mathscr{X}$) :

$$\mathbf{E}[\psi_{\theta(x),\nu(x)}(O)|X = x] = 0 \tag{8}$$

  where $\psi_{\theta(x),\nu(x)}(\cdot)$ is a known function and $\nu(x)$ is a nuisance parameter.

- Examples : conditional means, conditional quantiles, etc.

- Several core elements of Breiman's forests (like recursive partitioning, subsampling, and random split selection) are preserved, but the idea that the final estimate is obtained by averaging estimates from each member of an ensemble is abandoned.

- A popular approach to estimating such functions $\theta(x)$ is to first define some kind of weights $\alpha_i(x)$ that measure the relevance of the i-th training example to fitting $\theta(\cdot)$ at x, and then fit the target of interest via an empirical version of the estimating equation :

$$\left(\hat{\theta}(x), \hat{\nu}(x)\right) \in \arg\min_{\theta,\nu} \left\| \sum_{i=1}^{n} \alpha_i(x) \psi_{\theta,\nu}(O_i) \right\|_2 . \tag{9}$$

Generalized Random Forests. II

- The generalized random forests is used to obtain such weights α_i by averaging neighborhoods implicitly produced by different trees.

- First, we grow a set of B trees indexed by $b = 1, \dots, B$ and, for each such tree, define $L_b(x)$ as the set of training examples falling in the same leaf as x.

- The weights $\alpha_i(x)$ capture the frequency with which the i-th training example falls into the same leaf as x :

$$\alpha_{bi}(x) = \frac{I(\{X_i \in L_b(x)\})}{|L_b(x)|}, \qquad \alpha_i = \frac{1}{B} \sum_{b=1}^{B} \alpha_{bi}(x). \tag{10}$$

- These weights sum to 1, and define the forest-based adaptive neighborhood of x.

- There are some subtleties in how the sets $L_b(x)$ are defined.

- The estimates $\hat{\theta}(x)$ produced by a generalized random forests are simply obtained by solving (9) with weights (10).

Generalized Random Forests. III

- The approach here is to mimic the algorithm of Breiman (2001) as closely as possible, while tailoring the splitting scheme to focus on heterogeneity in the target functional $\theta(x)$.
- The search for good splits proceeds as follows : we seek splits that immediately improve the quality of the tree fit as much as possible.
- Every split starts with a parent node $P \subset \mathscr{X}$; given a sample of data $\mathscr{J}$, we define $\left(\hat{\theta}_P, \hat{\nu}_P\right)(\mathscr{J})$ as the solution to the estimating equation, as follows :

$$\left(\hat{\theta}_P, \hat{\nu}_P\right)(\mathscr{J}) \in \arg\min_{\theta,\nu} \left\| \sum_{\{i \in \mathscr{J} : X_i \in P\}} \psi_{\theta,\nu}(O_i) \right\|_2 . \tag{11}$$

- The idea is to divide P into two children $C_1, C_2 \in \mathscr{X}$ such that the following error is minimized

$$err(C_1, C_2) = \sum_{j=1,2} \mathbb{P}_{X \in P}[X \in C_j] \mathbb{E}_{X \in C_j} \left[\left(\hat{\theta}_{C_j}(\mathscr{J}) - \theta(X) \right)^2 \right]$$

where the $\hat{\theta}_{C_j}(\mathscr{J})$ are fit over the child nodes C_j in analogy to (11) and X is a random evaluation point.

Generalized Random Forests. IV

Proposition 1

Suppose that some regularity assumptions hold and that the parent node P has a radius smaller than r for some value $r > 0$. Fix a training sample $\mathcal{J}^{tr}$. We write $n_P = |\{i \in \mathcal{J}^{tr} : X_i \in P\}|$ for the number of observations in the parent and n_{C_j} for the number of observations in each child, and define

$$\Delta(C_1, C_2) := \frac{n_{C_1} n_{C_2}}{n_P^2} \left(\hat{\theta}_{C_1}(\mathcal{J}^{tr}) - \hat{\theta}_{C_2}(\mathcal{J}^{tr}) \right)^2$$

where $\hat{\theta}_{C_1}(\mathcal{J}^{tr})$ and $\hat{\theta}_{C_2}(\mathcal{J}^{tr})$ are solutions to the estimating equation computed in the children, following (11). Then, treating the children C_1 and C_2 as well as the counts n_{C_1} and n_{C_2} as constant, and assuming that $n_{C_1}, n_{C_2} \gg r^{-2}$, we have

$$err(C_1, C_2) = K(P) - \mathbb{E}[\Delta(C_1, C_2)] + o(r^2)$$

where $K(P)$ is a deterministic term that does not depend on how the parent is split, and the o-term incorporates terms that depend on sampling variance.

Generalized Random Forests. V

- Hence, we consider splits that make the above Δ-criterion large.

- We can think of this Δ-criterion as favoring splits that increase the heterogeneity of the in-sample θ-estimates as fast as possible.

- Including this error term in the splitting criterion may stabilize the construction of the tree, and further it can prevent the splitting criterion from favoring splits that make the model difficult to estimate, for example, splits where there is not sufficient variation in the data to estimate the model parameters within the resulting child leaves.

- However, from a computational perspective, optimizing the criterion $\Delta(C_1, C_2)$ over all possible splits while explicitly solving for $\hat{\theta}_{C_1}$ and $\hat{\theta}_{C_2}$ in each candidate child using an analogue to (11) may be quite expensive.

Generalized Random Forests. VI

- To avoid this issue, we instead optimize an approximate criterion $\widetilde{\Delta}(C_1, C_2)$ built using gradient-based approximations for $\hat{\theta}_{C_1}$ and $\hat{\theta}_{C_2}$.

- For each child C, we use $\widetilde{\theta}_C \approx \hat{\theta}_C$ as follows :
  - first compute A_P as any consistent estimate for the gradient of the expectation of the ψ-function (*i.e.* $\nabla \mathbb{E}[\psi_{\hat{\theta}_P, \hat{\nu}_P}(O) | X = x]$)
  - then, set

$$\widetilde{\theta}_C = \hat{\theta}_C - \frac{1}{|\{i : X_i \in C\}|} \sum_{i: X_i \in C} \xi^T A_P^{-1} \psi_{\hat{\theta}_P, \hat{\nu}_P}(O_i) \qquad (12)$$

    where $\hat{\theta}_P$ and $\hat{\nu}_P$ are obtained by solving (11) once in the parent node, and ξ is a vector that picks out the θ-coordinate from the (θ, ν) vector.
  - When the ψ-function itself is continuously differentiable, we use

$$A_P = \frac{1}{|\{i : X_i \in P\}|} \sum_{i: X_i \in P} \nabla \psi_{\hat{\theta}_P, \hat{\nu}_P}(O_i). \qquad (13)$$

Generalized Random Forests. VII

- Algorithmically, this recursive partitioning scheme reduces to alternatively applying the following two steps :
    1. First, in a labeling step, we compute $\hat{\theta}_P$, $\hat{\nu}_P$ and A_P^{-1} on the parent data as in (11), and use them to get pseudo-outcomes
    $$\rho_i = -\xi^T A_P^{-1} \psi_{\hat{\theta}_P, \hat{\nu}_P}(O_i) \in \mathbb{R}. \tag{14}$$
    2. Next, in a regression step, we run a standard CART regression split on the pseudo-outcomes ρ_i. Specifically, we split P into two children C_1 and C_2 such as to maximize the criterion
    $$\widetilde{\Delta}(C_1, C_2) = \sum_{j=1}^{2} \frac{-1}{|\{i : X_i \in C_j\}|} \left(\sum_{i:X_i \in C_j} \rho_i \right)^2 . \tag{15}$$
    3. Finally, once we have executed the regression step, we relabel the observations in each child by solving the estimating equation, and continue on recursively.

Generalized Random Forests. VIII

Algorithm 1 Generalized random forest with honesty and subsampling

Note: All tuning parameters, such as the total number of trees B and the sub-sampling s rate used in Subsample, are taken as pre-specified. This function is implemented in the package `grf` for `R` and `C++`.

```
1: procedure GeneralizedRandomForest(set of examples $\mathcal{S}$, test point $x$)
2:     weight vector $\alpha \leftarrow$ Zeros($|\mathcal{S}|$)
3:     for $b = 1$ to total number of trees $B$ do
4:         set of examples $\mathcal{I} \leftarrow$ Subsample($\mathcal{S}$, $s$)
5:         sets of examples $\mathcal{J}_1$, $\mathcal{J}_2 \leftarrow$ SplitSample($\mathcal{I}$)
6:         tree $\mathcal{T} \leftarrow$ GradientTree($\mathcal{J}_1$)        ▷ Grows a tree by recursive partitioning,
                                                            alternating the steps (14) and (15).
7:         $\mathcal{N} \leftarrow$Neighbors($x$, $\mathcal{T}$, $\mathcal{J}_2$)          ▷ Returns those elements of $\mathcal{J}_2$ that fall
                                                            into the same leaf as $x$ in the tree $\mathcal{T}$.
8:         for all example $e \in \mathcal{N}$ do
9:             $\alpha[e]$ += $1/|\mathcal{N}|$
10:    output $\hat{\theta}(x)$, the solution to (5) with weights $\alpha/B$
```

The function call Zeros creates a vector of zeros of length $|\mathcal{S}|$; Subsample draws a subsample of size s from $\mathcal{S}$ without replacement; and SplitSample randomly divides a set into two evenly-sized, non-overlapping halves.
The final step (5) can be solved using any numerical estimator. Our implementation `grf` provides an explicit plug-in point where a user can write a solver for (5) that is appropriate for their ψ-function of interest.

Outline

R Commands for Trees. I

The tree library is used to construct classification and regression trees.

```
> library (tree)
```

First use classification trees to analyze the Carseats data set. Sales is a continuous variable, and so we begin by recoding it as a binary variable. We use the ifelse() function to create a variable, called High, which takes on a value of Yes if the Sales variable exceeds 8, and No otherwise.

```
> library (ISLR)
> attach (Carseats )
> High=ifelse (Sales <=8," No"," Yes ")
```

We use the data.frame() function to merge High with the rest of the Carseats data.

R Commands for Trees. II

Now use the `tree()` function to fit a classification tree in order to predict `High` using all variables but `Sales`

```
> tree.carseats =tree(High~.-Sales ,Carseats )
```

The function `cv.tree()` performs cross-validation in order to determine the optimal level of tree complexity ; *cost complexity pruning* is used in order to select a sequence of trees for consideration.

The `cv.tree()` function reports the number of terminal nodes of each tree considered (`size`) as well as the corresponding error rate and the value of the cost-complexity parameter used (`k`, which corresponds to α in $C_\alpha(T)$).

R Commands for Bagging and Forests. I

Use the `randomForest` package in `R`.

Recall that bagging is simply a special case of a random forest with $m = p$. Therefore, the `randomForest()` function can be used to perform both random forests and bagging.