Part 12: Trees, Boosting, and Bagging

Chris Conlon

April 27, 2020

Applied Econometrics II

Reading

- Chapters 9,10,15 of *Elements of Statistical Learning*

Trees

Decision Trees

Start with $y = f(x_i)$:

- Construct a tree by splitting the sample on an x_i.
  - Choose the split to maximize the criterion function.
  - Choose the x_i to maximize the criterion given the proposed split.
- Which x_i do we choose?
  - There are K possibilities.
  - But how do we know which order to split?
- Which split do we choose?
  - This is usually single dimensional optimization.
- Resulting problem is NP hard.

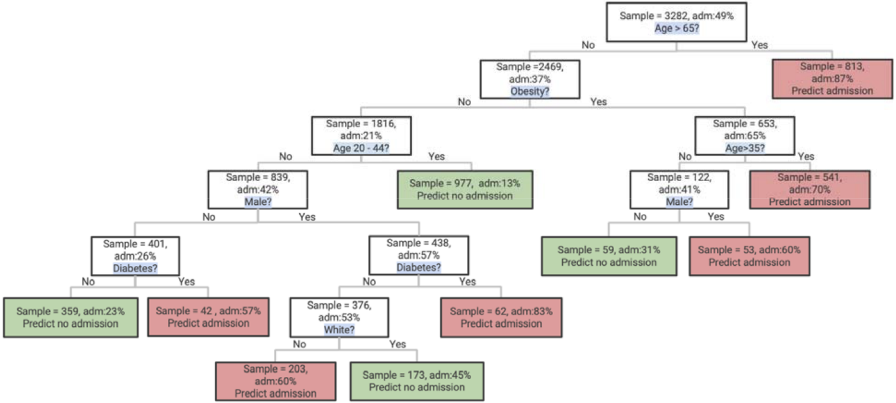
Decision Trees

What kind of tree?

- Classification Trees predict discrete outcomes.
- Regression Trees predict continuous outcomes.

Decision Trees: Covid Hospitalization (Petrilli et al 2020)

Figure2A: Maximum likelihood classification tree for hospitalization



Sample = 3282, adm:49%
Age > 65?

No

Yes

Sample =2469, adm:37%
Obesity?

Sample = 813, adm:87%
Predict admission

No

Yes

Sample = 1816, adm:21%
Age 20 - 44?

Sample = 653, adm:65%
Age>35?

No

Yes

Sample = 839, adm:42%
Male?

Sample = 977, adm:13%
Predict no admission

No

Yes

Sample = 122, adm:41%
Male?

Sample = 541, adm:70%
Predict admission

No

Yes

Sample = 401, adm:26%
Diabetes?

Sample = 438, adm:57%
Diabetes?

No

Yes

Sample = 59, adm:31%
Predict no admission

Sample = 53, adm:60%
Predict admission

No

Yes

Sample = 359, adm:23%
Predict no admission

Sample = 42 , adm:57%
Predict admission

No

Yes

Sample = 376, adm:53%
White?

Sample = 62, adm:83%
Predict admission

No

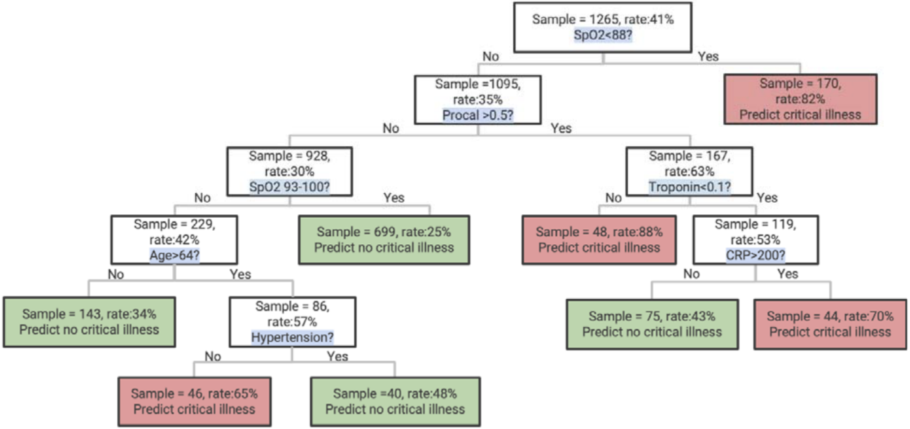
Yes

Sample = 203, adm:60%
Predict admission

Sample = 173, adm:45%
Predict no admission

Decision Trees: Covid Critical Illness

Figure 2B: Maximum likelihood classification tree for critical illness



How to choose splits?

Decision Trees: Criteria #1: RSS

Residual sum of squares:

$$\min_f \sum_{i=1}^{N} (y - f(x_i))^2$$

- “Regression Trees”
- Seems sensible for continuous y not discrete y.
- This is not OLS because $E[y|x]$ is highly nonlinear.

Decision Trees: Criteria #2: Gini Impurity

For each side of each split calculate (smaller is better):

$$\sum_j \underbrace{p_j}_{\text{Correct}} \times \underbrace{(\sum_{k \neq j} p_k)}_{\text{Incorrect}} = \sum_j p_j(1 - p_j) = 1 - \sum_j p_j^2$$

- 1265 Patients with 41% Critical Rate → 0.2419.
- Split on $Sp02 < 88$
  - Left: 383 out of 1095 cases (35%) → Gini: 0.2275
  - Right: 140 out of 170 cases (82%) → Gini: 0.1476
- Weighted avg: $\underbrace{(1095/1265) * (0.2275)}_{0.1968} + \underbrace{(140/1265) * (0.1476)}_{0.0163} = 0.2131$
- Choose splits to minimze weighted average impurity.

Decision Trees: Criteria #3: Variance Reduction

Calculate pre-split variance: $\underbrace{\frac{1}{|S|^2}\sum_{i\in S}\frac{1}{2}(y_i-y_j)^2}_{\text{Pre split Var}}$ and calculate post-split variance:

$$\underbrace{\frac{1}{|S_t|^2}\sum_{i\in S_t}\sum_{j\in S_t}\frac{1}{2}(y_i-y_j)^2}_{\text{Post split TRUE}}+\underbrace{\frac{1}{|S_f|^2}\sum_{i\in S_f}\sum_{j\in S_f}\frac{1}{2}(y_i-y_j)^2}_{\text{Post split FALSE}}$$

- Maximize the reduction in variance / minimize post-split variance
- Note the connection, for discrete distribution $Var(y)=|S|\cdot p_j(1-p_j)$.

Decision Trees: Criteria #4: Information Gain/Entropy

Start with the Kullback-Leibler divergence.

- This measures the distance between two distributions $P(x), Q(x)$.
- If they are the same we get $\log(1) = 0$, otherwise expected log distance (where expectation over $p(x)$).

$$D_{\mathrm{KL}}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\log_b\left(\frac{P(x)}{Q(x)}\right), \quad D_{\mathrm{KL}}(P\|Q) = \int_{-\infty}^{\infty} p(x)\log_b\left(\frac{p(x)}{q(x)}\right)dx$$

- Should be ≥ 0 always.
- Can write as $D_{\mathrm{KL}}(P\|Q) = -\sum_{x\in\mathcal{X}} P(x)\log_b\left(\frac{Q(x)}{P(x)}\right)$

Decision Trees: Criteria #4: Information Gain/Entropy

Derive the mutual information and information gain and entropy $H(X)$:

$$\begin{aligned} I(X,Y) &\equiv KL(p(x,y)\|p(x)p(y)) \\ &= H(X) - H(X|Y) = H(Y) - H(Y|X) \end{aligned}$$

$$H(X) = -\sum_{j=1}^{J} p_j \log_b p_j$$

- Tells the divergence between $p(x,y)$ (joint distribution) and $p(x)p(y)$ (marginals).
- Mutual information is symmetric, KL divergence is not.
- Information Gain tells us how valuable how much divergence/entropy is reduced when observing $A = a$.

Decision Trees: Criteria #4: Information Gain/Entropy

The same example:

- 1265 Patients with 41% Critical Rate → $-0.41 \log_2 0.41 - 0.59 \log_2 0.59 = 0.9765$
- Split on $Sp02 < 88$
  - Left: 383 out of 1095 cases (35%) → $-0.35 \log_2 0.35 - 0.65 \log_2 0.65 = 0.934$
  - Right: 140 out of 170 cases (82%) → $-0.82 \log_2 0.82 - 0.65 \log_2 0.18 = 0.68$
- Weighted avg: $\underbrace{(1095/1265) * (0.934)}_{0.866} + \underbrace{(140/1265) * (0.68)}_{0.134} = 0.900$

The goal is to maximize entropy reduction.

$$IG(T, a) = \underbrace{H(T)}_{0.9765} - \underbrace{H(T|a)}_{0.900} = 0.0765$$

What are trees actually doing?

Think about the RSS case : $\min_f \sum_{i=1}^{N}(y - f(x_i))^2$

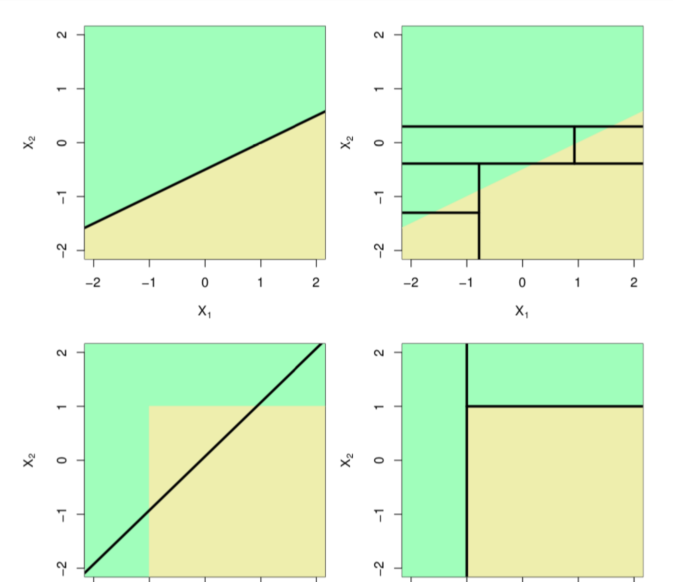
- This is nonparametric regression
- It is also a kernel model.
- Partition the data: $\{R_1, \ldots, R_M\}$ into leaves that are disjoint and span the space of X.
- $f(x) = \sum_{m=1}^{M} c_m 1\left(x \in R_m\right)$.
  - Where c_m is mean of y_i within leaf R_m.
- This is just a locally constant regression. But distance is not determined using bandwidth...
- Alternative: k-NN where k is the number of observations on same leaf.

What are trees actually doing?

When are trees the right tool?

- Because of multiple levels of splits: work best when true relationship is highly nonlinear.
- Who has a heart attack?
  - Lots of factors: high BP, overweight, age, family history, diabetes, etc.
  - Logit: these enter additively in the index and increase log odds proportionally.
  - Can interact multiple factors $highBP \times overweight$, etc.
- If the true model is highly interacted: trees will do well.
- In general trees have low bias but high variance
  - Small changes in data can lead to wildly different splits (and trees).

Linear vs. Nonlinear Relationships



How do splits work?

- Binary variables: Trivial
- Continuous Variables: Choose a split value s so that $x > s$ or $x \leq s$ optimizes your criteria.
    - This is a single dimensional search (Golden Section, etc.).
- Multiple Categories (A, B, C): this is the hard case
    - Order by the outcome variable $y_b > y_a > y_c$.
    - Then treat like the continuous case.

Growing your Tree

- Let $|T| = M$ denote the number of terminal nodes in T . We will use $|T|$ to measure the complexity of a tree. For any given complexity, we want the tree minimizing square error on training set.
- Finding the optimal binary tree of a given complexity is computationally intractable (NP hard).

Tree Algorithms

How do we build a tree?

- Because the problem is NP hard: no ideal way.
- Mostly heuristics.
- Greedy Algorithm: Choose best split first for best x go from there.
    - Need not be optimal...
- Avoiding over-fitting
    - How deep to make tree?
    - Minimum number of observations per branch?
    - Pruning?
    - Trees will always overfit if you let them!

Why Prune a Tree?

To avoid overfitting

- Any useful split will be made (eventually).
- We (may) end up with perfectly predicted outcomes $E[y_i|x_i]$. case we know for sure that $y_i = 1$?
- Remove leaves with too few elements.
- Usually use a hold-out sample (test set) and remove leaves if it increases OOS criteria.
- Other canned pruning algorithms will generate several candidate subtrees
    - Used out of sample critieria to pick the best subtree
- Can also do early stopping

Tree Algorithms

About criteria and heuristics.

- ID3 (Iterative Dichotomiser 3)
- C4.5 (successor of ID3)
- CART (Classification And Regression Tree)
- Chi-square automatic interaction detection (CHAID). Performs multi-level splits when computing classification trees.
- MARS: extends decision trees to handle numerical data better.
- Conditional Inference Trees. Statistics-based approach that uses non-parametric tests as splitting criteria, corrected for multiple testing to avoid overfitting. This approach results in unbiased predictor selection and does not require pruning.

Bagging

Bagging

What is Bagging? Bootstrap Aggregation.

- We re-sample a new dataset the size of our original dataset with replacement
- Before we re-ran our estimation to get $\hat{\theta}^b$ and took $Var(\hat{\theta}^{(1)}, \hat{\theta}^{(2)}, \ldots, \hat{\theta}^{(B)})$ and $E(\theta) = \frac{1}{B}\sum_{b=1}^{B} \hat{\theta}^b$.
- Since we are doing ML, let's bag $\hat{y}_i$.
- $E(y) = \frac{1}{B}\sum_{b=1}^{B} \hat{y}^b$ (we could also do this conditionally).

Bagging: What's the point?

Why would we want to do this?

- If we are worried about high-variance and overfitting bagging will reduce our variance.
- Each bootstrapped sample b is like an IID realization of our data.
- Averaging over samples can reduce the variance of $\hat{y}$ by $\frac{1}{B}$.

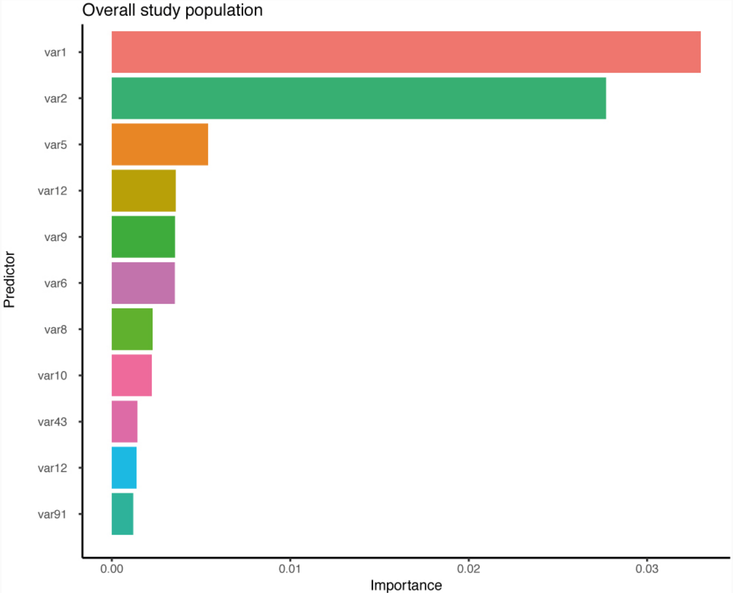
The problem Interpretability

- If my "model" is OLS I can report average coefficients.
- If my model is a tree, what do I even report averaged over 1000 bootstrapped simulations?

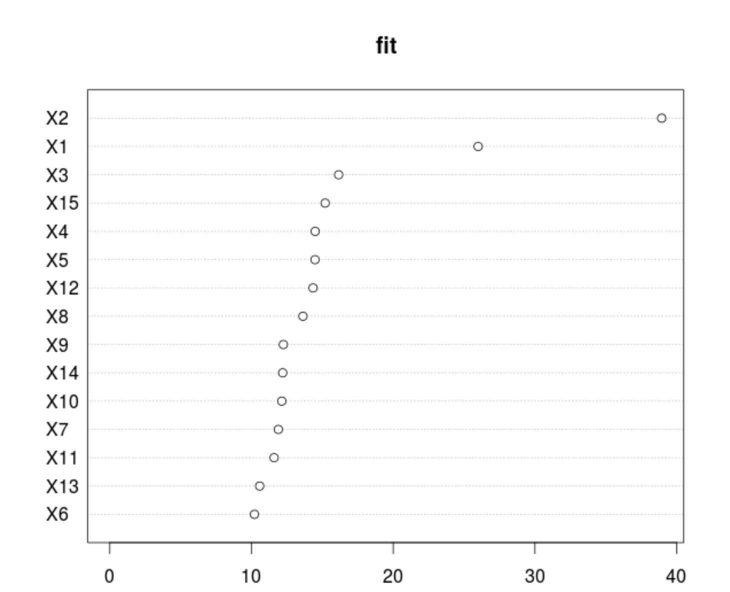
How to interpret variables?

- Pick a tree T^b calculate how much a particular variable x_1 increases the Gini Index, or decreases RSS, etc.
- Average over all trees (some trees will be zero if they don't include x_1).
- Report the average in relative terms for all x_k.

Variable Importance Plots



Variable Importance Plots



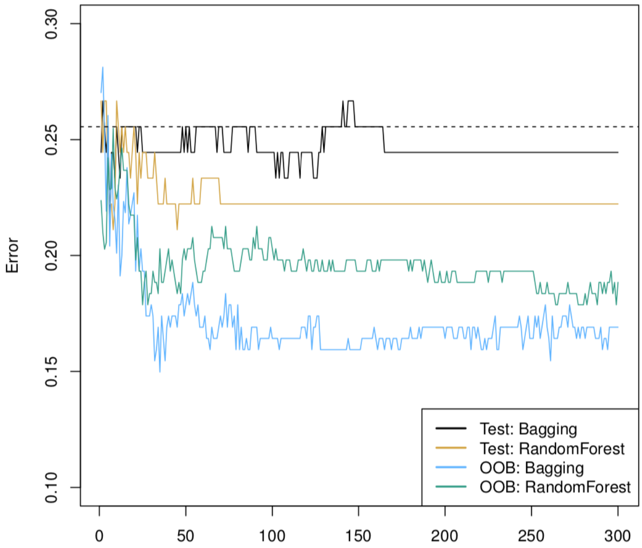
Validation: How do assess model fit?

Out of Bag Error Rate (OOB)

- Use cross validation
- Draw a bootstrap sample of size N.
- Split sample into two parts: training $\frac{2}{3}$, validation $\frac{1}{3}$
- Fit on the training sample
- Predict on the validation sample: $(y_i - \hat{y}_i^{OOB})^2$
- Report average or (mean-squared) error on the validation sample only.
- Gives us expected out of sample fit.

Is this the same as error rate on totally new data?

Bagging: OOB error



Why do Bagging do so poorly?

- Even though we bootstrap our sample many times → $(T^B, T^{B'})$ are highly correlated.
- Why does this happen?
  - We end up with the same X's and the same splits in each model.
- Adding more trees doesn't really improve forecast performance.
- We would like to add more trees but have them be less correlated with one another.

Solution: Random Forests

1. Draw a bootstrap sample b.
2. Fit a tree (usually a small one with limited depth)
3. At each node select a random subset of regressors $m < K$ from X.
4. Average the predictions of (hopefully) less correlated trees. (Or Majority Vote).

How to choose m? Usually $\sqrt{K}$.

Solution: Random Forests

- Even though we bootstrap our sample many times → $(T^{B}, T^{B'})$ are highly correlated.
- Adding more trees doesn't really improve forecast performance.
- We would like to add more trees but have them be less correlated with one another.

Evaluating Random Forests

How important is $X^{(j)}$? Look at the trees that randomly don't include it in m (or don't select it)

$$\widehat{\Delta}_j = \frac{1}{m} \sum_{i \in \mathcal{H}} \left(Y_i - \widehat{m}_{(-j)}\left(X_i\right)\right)^2 - \left(Y_i - \widehat{m}\left(X_i\right)\right)^2$$

Average this quantity across the entire training set:

$$\mathbb{E}\left[\widehat{\Delta}_j | \mathcal{T}\right] = \mathbb{E}\left[\left(Y - \widehat{m}_{(-j)}(X)\right)^2 - (Y - \widehat{m}(X))^2 | \mathcal{T}\right] \equiv \Delta_j$$

This is called Leave-One-Out-COvariates (LOCO).

Inference on Random Forests

- Technically possible but tricky
- Active area of research
- See Wager and Athey (2017).

What about Boosting?

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for $i = 1, \ldots, N$.
2. For $b = 1, \ldots, B$ iterate on:
   2.1 Fit a tree $\hat{f}^b$ with d splits to the response $r_1, \ldots, r_n$.
   2.2 Update the prediction to
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$
   2.3 Update the residual
   $$r_i \leftarrow r_i - \lambda \hat{f}^b\left(x_i\right)$$
3. Produce the final model by averaging
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

How does Boosting Work?

- At each bootstrap iteration we aren't fitting y_i
- We are fitting r_i the residual $y_i - \sum_{b'=1}^{b} \lambda \hat{f}^{b'}$.
- λ is usually small so that we update the model slowly.
- You can think about what we are doing as re-weighting our training data
  - Put more weight on the cases we fit the worst (large residuals).
- Usually set λ small $.01$
- Well known algorithm AdaBoost

Boosting $>$ Bagging $>$ Single Tree

AdaBoost: Freund & Schapire, 1996

1. Weight observations equally $w_i = \frac{1}{N}$. for all $i = 1, \ldots, N$.
2. From $b = 1, \ldots, B$ do the following:
    2.1 Fit a model $f^{(b)}(x_i)$ to training data using w_i.
    2.2 Compute $err_b = \frac{\sum_{i=1}^N w_i I[y_i \neq f^{(b)}(x_i)]}{\sum_{i=1}^N w_i}$
    2.3 Compute $\alpha_b = \log\left(\frac{1-err_b}{err_b}\right)$
    2.4 Update weights for $i = 1, \ldots, N$:

$$w_i \leftarrow w_i \cdot \exp[\alpha_b I(y_i \neq f^{(b)}(x_i))]$$

    renormalize so that $\sum w_i = 1$.

3. Compute $\hat{f}(x) = sign[\sum_b \alpha_b \hat{f}^{(b)}(x_i)]$.

Additive Models

- $\hat{f}(x) = \sum_b \alpha_b \hat{f}^{(b)}(x_i)$ is an additive model
- Lots of examples in the literature:
  - Basis Functions (ie: polynomials) $\sum_{k=1}^{K} \theta_k g_k(x)$.
  - GAMs: $f(x) = \sum_{k=1}^{K} f_k(x)$.
- Usually we fit these in one step (OLS, MLE, etc.)
- Here we fit α_b, θ_b in a stagewise manner.
- Process fits slowly but avoids overfitting problem.

What about Gradient Boosting?

Same as AdaBoost with some modifications

- Loss function $L(y_i, \gamma)$
- Choose γ to minimize $\min L(y_i, \gamma)$ and get $h_b(x)$.
- Update $r_i = \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]$ this is the gradient evaluated at bth iteration of $F_b(x_i)$.
- Update $F_b(x) = F_{b-1}(x) + \gamma_b h_b(x)$.

How do we do this in R?

- Sample Splitting, Bagging, etc. `caret`, `sample`
- Gradient Boosting. `gbm` (easy), `xgboost` (faster).
- Random forest `randomForest`.

Thanks!