Adaptive Feature Selection in Random Forests

Christian Hilscher

University of Mannheim

Update on April 3, 2021

How to build a tree 1/3

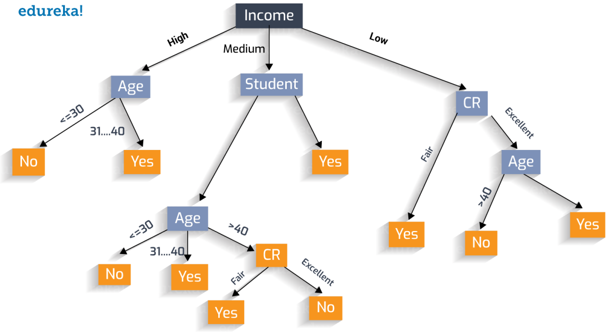

Figure: Example of a Decision Tree

Two questions at each node:

- Which variable to choose
- Which split-point to choose

How to build a tree - choosing the split-point 2/3

- Pick one variable (= feature, dimension) and sort the X
- Each node has a certain variance:

$$\hat{\Delta}(t) = \frac{1}{N(t)} \sum_{\mathbf{X}_i \in t} (Y_i - \bar{Y}_i)^2$$

- Choose split-point such that the two daughter nodes are homogeneous
- This yields the following *decrease in impurity*:

$$\hat{\Delta}(s; \mathbf{t}) = \hat{\Delta}(\mathbf{t}) - [\hat{P}(\mathbf{t}_L)\hat{\Delta}(\mathbf{t}_L) + \hat{P}(\mathbf{t}_R)\hat{\Delta}(\mathbf{t}_R)]$$

- Maximising $\hat{\Delta}(s; \mathbf{t})$ gives us split-point

How to build a tree - choosing the variable 3/3

- Given a variable j we get $\hat{\Delta}_j(s;\mathbf{t})$
- We then simply choose the variable with the highest j:

$$\arg\max_j \ \hat{\Delta}_j(s;\mathbf{t})$$

This continues until there are $k \geq 1$ observation in each node

Estimator is then average in a node:

$$\hat{Y}_t = \frac{\sum_i Y_i \mathbb{1}_{(X_i \in t)}}{\sum_i \mathbb{1}_{(X_i \in t)}}$$

Relevant concepts

Partial dependence function:

$$\bar{F}_j(x_j;t) = \mathbb{E}[Y|\mathbf{X} \in \mathbf{t}, X_j = j]$$

Mean-centered partial dependence function:

$$\begin{aligned}\bar{G}_j(x_j;t) &= \bar{F}_j(x_j;t) - \int \bar{F}_j(x_j;t)\mathbb{P}_{X_j|\mathbf{X}\in\mathbf{t}}(dx_j) \\ &= \mathbb{E}[Y|\mathbf{X} \in \mathbf{t}, X_j = j] - \mathbb{E}[Y|\mathbf{X} \in \mathbf{t}]\end{aligned}$$

Node balancedness:

$$\begin{aligned}\lambda_j(t) &= 4P_j(t_L)P_j(t_R) = 1 - |P_j(t_L) - P_j(t_R)|^2 \\ &= \frac{\Delta(j, s^*, t')}{|\bar{G}_j(x_j;t|^2 + \Delta(j, s^*, t')}\end{aligned}$$

Splitting properties of the CART algorithm

- $\lambda_j(t) = \frac{\Delta(j,s^*,t')}{|\bar{G}_j(x_j;t|^2+\Delta(j,s^*,t')}$
- Then, $\lim_{\Delta\to 0}\lambda = 0$ and $\lim_{\Delta\to \inf}\lambda = 1$
- Noisy variables ($\Delta \approx 0$) tend to be split on edges
- $==$ end-cut-preference (ecp)

Problems with splits on the edge:

- Very few observations in one of the two nodes
- Those nodes are not a good predictor

Elevator Pitch

- We know that $\arg\max_s \Delta$ leads to ecp
- Could instead maximise something else, for example

$$\Delta^\alpha = [4P(t_L)P(t_R)]^\alpha \Delta$$

- Since $\Delta^\alpha \leq \Delta$, this will lead to a lower decrease in impurity

Elevator Pitch

- We know that $\arg\max_s \Delta$ leads to ecp
- Could instead maximise something else, for example

$$\Delta^\alpha = [4P(t_L)P(t_R)]^\alpha \Delta$$

- Since $\Delta^\alpha \leq \Delta$, this will lead to a lower decrease in impurity

My idea:

- Our problem is essentially

$$\arg\max_j \ \arg\max_s \ \Delta_j^\alpha(s)$$

- Noisy variables are split at ends of feature space
- They get a low weight
- They will be chosen less often as optimal split dimension

Questions

Two approaches:

$$\text{a.)}\ \arg\max_j\ \arg\max_s\ [4P(t_L)P(t_R)]^\alpha \Delta_j(s)$$

$$\text{b.)}\ \arg\max_j\ [4P(t_L)P(t_R)]^\alpha \arg\max_s\ \Delta_j(s)$$

a.) weights incorporated into choosing the split-point $within$ a variable
b.) original CART with weights slapped on for penalizing ecp

Roadmap - next 14 days

- Show that noisy variables are less often chosen with b.) (should be fairly ok, no?)
- Think about whether one could show the same with a.) (is this even true?)
- upgrade DTRegressor to RFR
- Play around and get maybe a feeling on how stuff changes empirically