UCLAAnderson
SCHOOL of MANAGEMENT

MGMTMFE 431:

Data Analytics and Machine Learning

Topic 5: Decision Trees. Boosting and Bagging

Spring 2019

Professor Lars A. Lochstoer

UCLAAnderson
SCHOOL of MANAGEMENT

Overview

a. Supervised vs. unsupervised machine learning

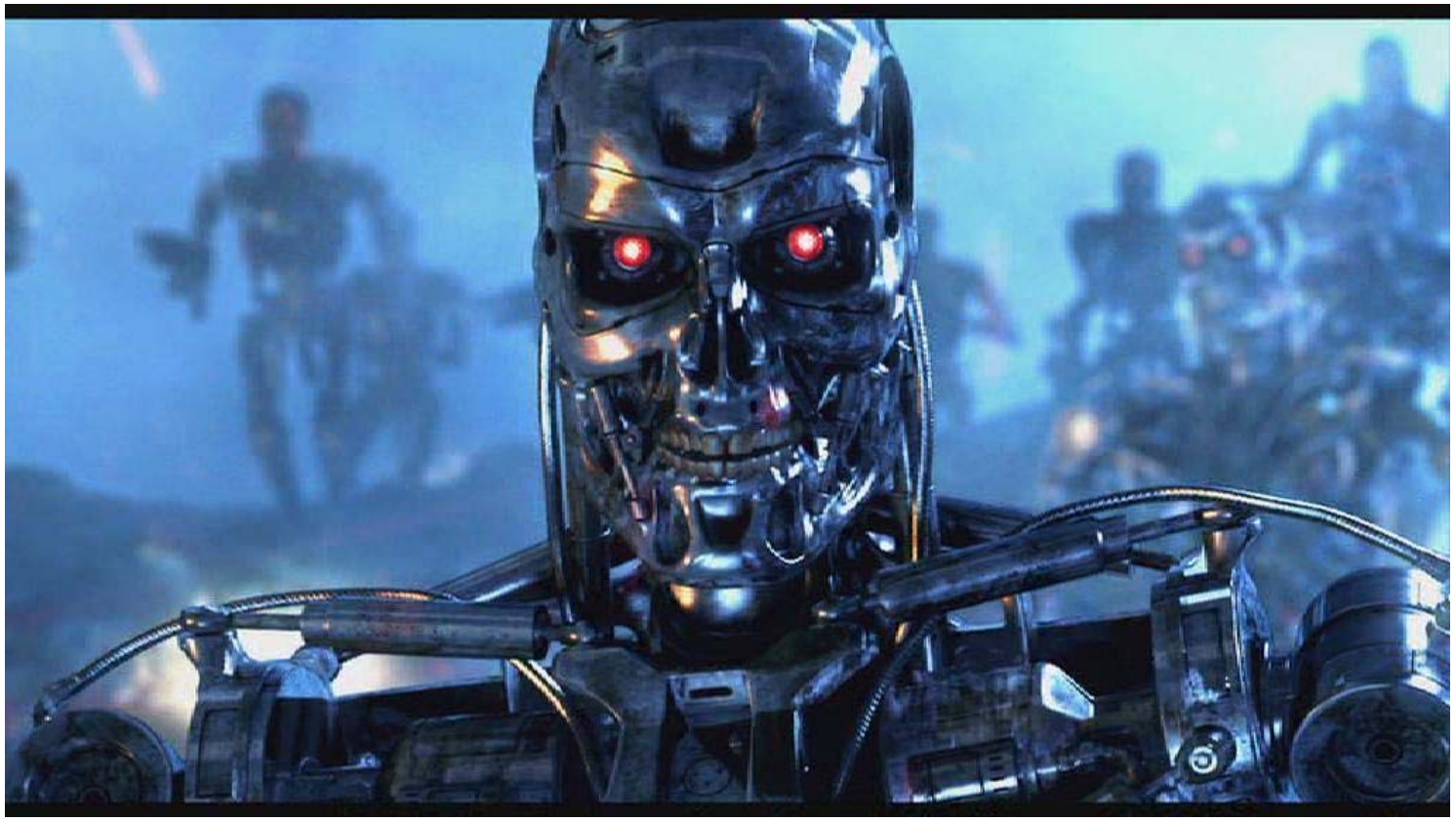
b. The bias-variance trade-off
   - High-dimensional data; model selection and regularization revisited

c. Supervised learning with flexible (non-linear) methods
   - Decision Trees
   - Bagging and Random Forest
   - Boosting and XGBoost

d. Next week's guest speaker

UCLAAnderson
SCHOOL of MANAGEMENT

a. What is Machine Learning?



UCLAAnderson
SCHOOL of MANAGEMENT

a. What is Machine Learning?

- Machine learning, statistical learning, data mining, data science
  - All about using (big) data to understand relationships in the data
  - We've been doing this all along: Linear (Panel) regressions, omitted variable bias, propensity scores, logistic regressions, shrinkage/regularization (Ridge, Lasso) for Model Selection
  - One could program a computer to automatically update a model based on data and using, e.g., cross-validation and the Lasso. This is machine learning.

- Overarching model: $Y = f(X) + irreducible\ noise$
  - Idea: let data tell us *f(X)*

a. Prediction

- Two common applications

1. Prediction

  - All we want is the best prediction of Y (e.g., expected returns on all stocks), as this is the relevant input in our subsequent decision problem (e.g., portfolio choice)
  - Two types of error: *reducible (model) error, irreducible error (unknowable)*
  - Let $\hat{Y} = \hat{f}(X)$. The mean-squared error is

$$E\left[\left(Y - \hat{Y}\right)^2\right] = E\left[\left(f(X) + \epsilon - \hat{f}(X)\right)^2\right]$$
$$= \underbrace{E\left[\left(f(X) - \hat{f}(X)\right)^2\right]}_{Reducible} + \underbrace{Var(\epsilon)}_{Irreducible}$$

- In prediction problems, we don't particularly care about the functional form of *f* or the *attributes* (*X*)
  - Main goal: minimize reducible error to get optimal predictor
  - Thus, flexible black box'ish methods like Random Forest may be appropriate

UCLAAnderson
SCHOOL of MANAGEMENT

a. Inference

- Two common applications (cont'd)

2. Inference

- Want to understand the way *Y* (e.g., profits) is affected by a change in *X* (e.g. advertising)
- Which predictors are associated with the response?
- What is the relationship between the response and each predictor?
- Here, functional form of *f* and identification are important issues
- Regression-based methods are convenient for this

UCLAAnderson
SCHOOL of MANAGEMENT

a. Supervised vs. unsupervised learning

Supervised Learning

- You have both the response variable (y) and the predictors (X) available
- This is what we have been doing so far in this course
- The learning is "supervised" because you have the right answer available (y)

Unsupervised Learning

- Here, we are not interested in prediction – we do not have *Y*
  - Note: no universally accepted mechanism for performing cross-validation, etc. Simply because the error is undefined when we don't have *Y*
- Goal is to discover interesting things about the measurements of the *p* "features" $X_1, X_2, X_3, \ldots, X_P$
  - Is there an informative way to visualize the data?
  - Can we discover subgroups among the variables or among the observations?
- Diverse set of ***unsupervised learning techniques*** available for answering such questions

UCLAAnderson
SCHOOL of MANAGEMENT

a. Examples of unsupervised learning

Principal Components Analysis

- You know this already
- Large set of correlated variables; can we find representative variables (factors) that explain most of the variation
- Useful for visualization: e.g., the Market factor (the 1st principal component) can be plotted against, say, macro events etc., as opposed to plotting return for each stock against these events.

Clustering

- Broad set of techniques for finding ***subgroups***
- Partition data into distinct groups; each group's observations are quite similar" to each other. This is the main difference relative to PCA, which tries to find a low-dimensional representation
- E.g., find ***market segments*** by identifying subgroups of people who might be more receptive to particular form of advertising or more likely to purchase a particular product

UCLAAnderson
SCHOOL of MANAGEMENT

Interlude

- In this lecture, we will introduce flexible, nonlinear supervised learning methods involving decision trees.

- But, before that we discuss the ***Bias vs. Variance Trade-Off***

UCLAAnderson
SCHOOL of MANAGEMENT

b. Bias-Variance Trade-Off

Why don't we just devise one ***best*** model?

- Depends on nature of data and our goal
- Some procedures work better in some case, some in other. But, why?

Fundamental trade-off (expectations are conditional on X):

$$E\left[\left(Y-\hat{f}(X)\right)^2\right]=Var\left(\hat{f}(X)\right)+\left[Bias\left(\hat{f}(X)\right)\right]^2+Var(\epsilon)$$

where $E\left[\left(Y-\hat{f}(X)\right)^2\right]$ is the expected MSE, where $\hat{f}(X)$ is the estimated model (which has prediction uncertainty due to standard errors), and where $Bias\left(\hat{f}(X)\right)=E\left[f(X)-\hat{f}(X)\right]$

UCLAAnderson
SCHOOL of MANAGEMENT

b. Bias-Variance Trade-Off (cont'd)

Fundamental trade-off from last slide (for convenience):

$$E\left[\left(Y-\hat{f}(X)\right)^2\right]=Var\left(\hat{f}(X)\right)+\left[Bias\left(\hat{f}(X)\right)\right]^2+Var(\epsilon)$$

Variance refers to the amount by which the function $\hat{f}$ would change if we estimated it using a different training data set (standard error)

- In general, more flexible methods have higher *variance*

Bias refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model

- E.g., a linear regression probably introduces bias as real-world not exactly linear
- Generally, more flexible methods result in less bias. For example, using the sample mean as the estimate of the unconditional mean is unbiased, but has a lot of variance (standard error squared)

UCLAAnderson
SCHOOL of MANAGEMENT

b. Bias-Variance Trade-Off and Regularization

Regularization (or Shrinkage) explicitly introduces ***bias*** *relative to OLS* by shrinking coefficients towards zero (or any other null hypothesis)

But, with cross-validation we can find the amount of regularization that gives the lowest "out-of-sample" MSE.

- Thus, we are reducing ***variance***

Regularizing too much introduces too much bias. For instance, setting all coefficients to zero makes variance zero, but (probably) is not optimal.

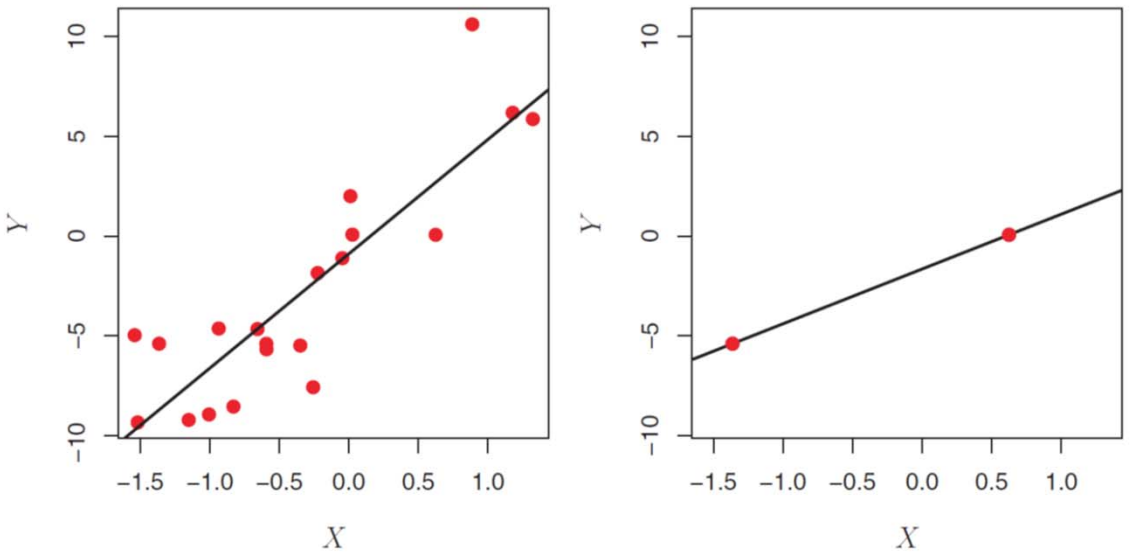
- Cross-validation is an attempt at finding the optimal trade-off between variance and bias

UCLAAnderson
SCHOOL of MANAGEMENT

b. Regularization and high-dimensional problems

With troves of new data available, prediction and inference problems often are ***high-dimensional***

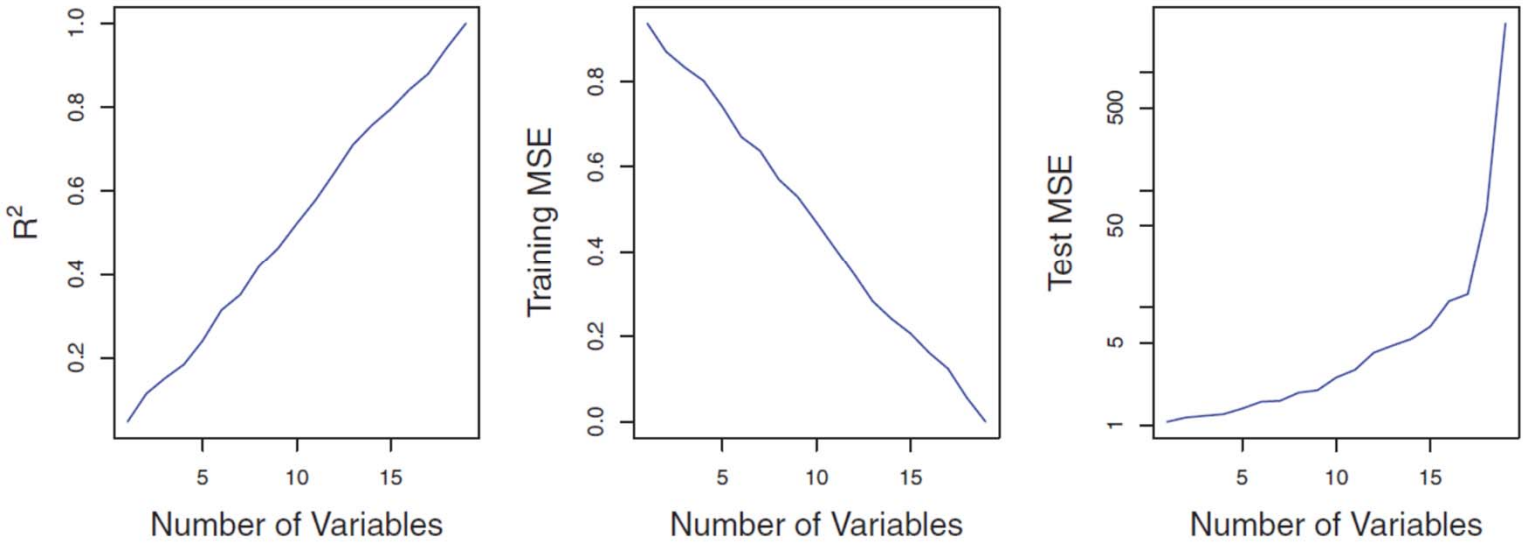
- Particularly extreme: number of predictors is larger than number of response observations (p > n)
- Regressions in this case will fit data perfectly, but solution is not unique + likely overfitting (e.g., see below for illustration)



UCLAAnderson
SCHOOL of MANAGEMENT

b. Regularization and high-dimensional problems

R2, training MSE both get 'better' with more variables, but out-of-sample (here, test MSE) goes down. Example has n = 20.



UCLAAnderson
SCHOOL of MANAGEMENT

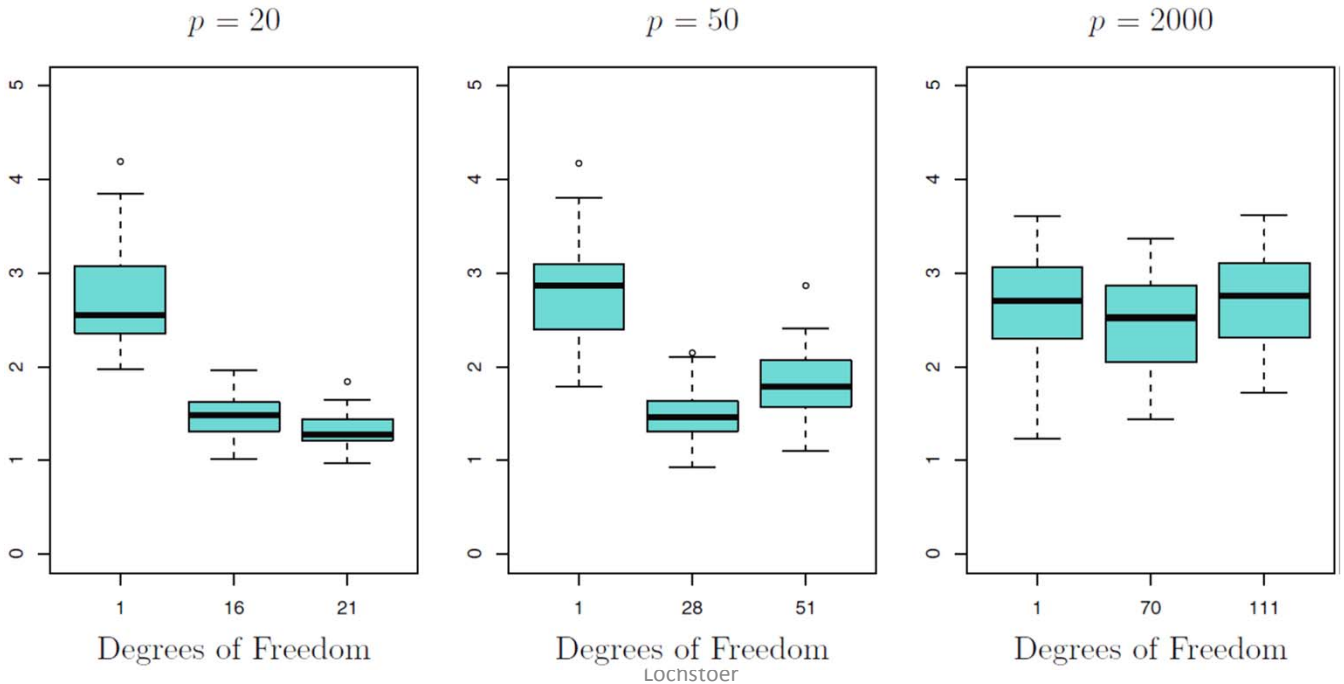
b. Regularization and high-dimensional problems

Regularization (e.g., using the LASSO) can help this problem, even when p > n.

- That said, as the below shows, just throwing in a bunch of variables is never a good idea. ***You should not use information you think a priori is noise to the extent possible.***

Below figure shows Test MSEs corresponding to a case with n = 100, with 20 variables that actually are related to response. Degrees of freedom is number of non-zero variables in the LASSO (a function of "lambda")

- Note how the p = 2000 >> n = 100 case always does badly: too much noise..!
- Note in the middle panel, that regularization does help in the case of 28 df



UCLAAnderson
SCHOOL of MANAGEMENT

c. Flexible non-linear models

As discussed, more flexible models typically reduces bias at the cost of higher variance

Next, we will look at Decision Trees

- Decision trees are intuitive and often easier to explain than regressions to non-technical people.

We first introduce decision trees, which are flexible but often has poor out-of-sample predictive power, in the simplest setting

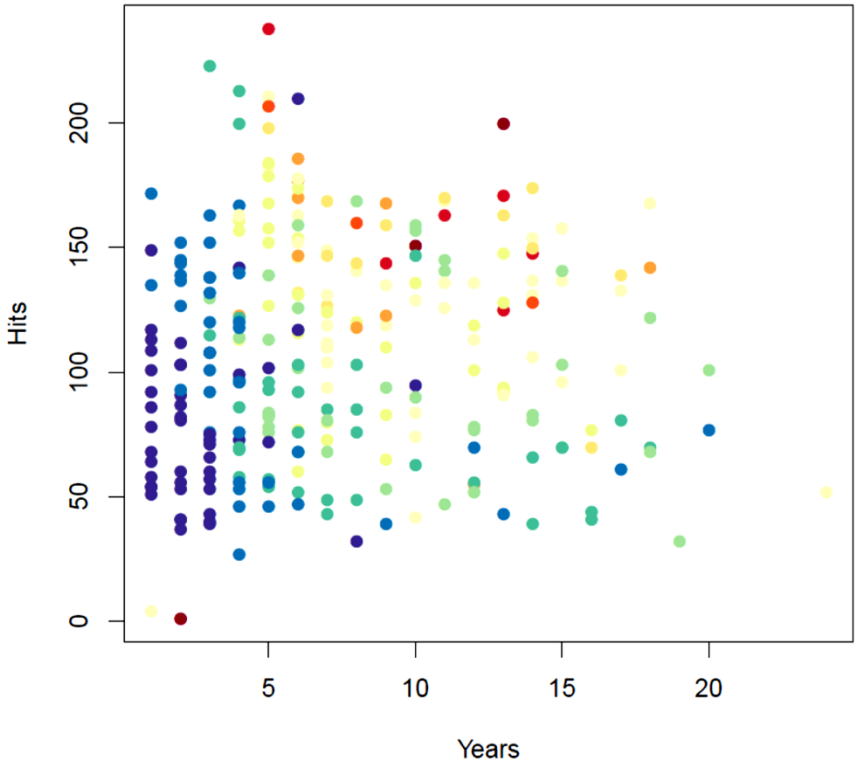
- Easier to understand
- Poor performance
  - Afterwards, we look at performance enhancing methods like ***boosting and bagging*** that strongly improves out-of-sample performance

UCLAAnderson
SCHOOL of MANAGEMENT

c. Decision trees: Example

A simple example from *Introduction to Statistical Learning*: Predicting baseball players' salaries. Data in plot below.

- "*Years*" is years player has been in major leagues, "*hits*" is player's number of hits last season
- Salary is color-coded, yellow-red is high salary, purple-blue is low



From inspection of the figure, it seems like for players with only a few years in the Leagues, the number of hits last season is not that important

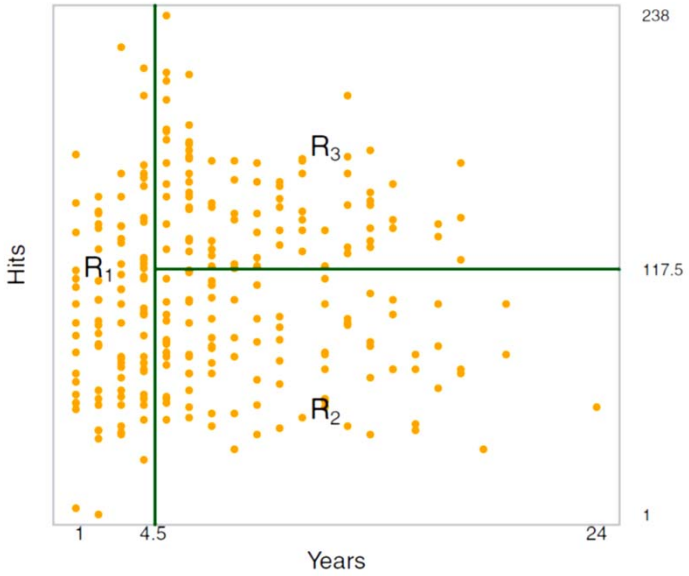
But, for players with more than, say, 5 years in the Leagues, the number of hits starts explaining the salary dispersion

UCLAAnderson
SCHOOL of MANAGEMENT

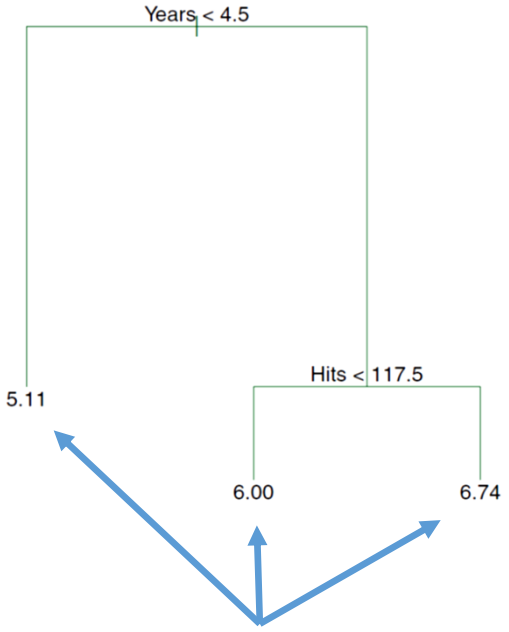
c. Decision trees: Example (cont'd)

A fitted tree with three "***boxes***"

- $R_1 = E_N[X|Years < 4.5]$, $R_2 = E_N[X|Years \geq 4.5,\ Hits < 117.5]$
- $R_3 = E_N[X|Years \geq 4.5,\ Hits \geq 117.5]$
- Notation: $E_N[Z|X \in A]$ sample average of *Z* for sample that satisfies $X \in A$



R_1, R_2, and R_3 are different splits of the data ("*boxes*")



Terminal nodes (also called "*leaves*") give average in each *box (log of salary/1000)*

UCLAAnderson
SCHOOL of MANAGEMENT

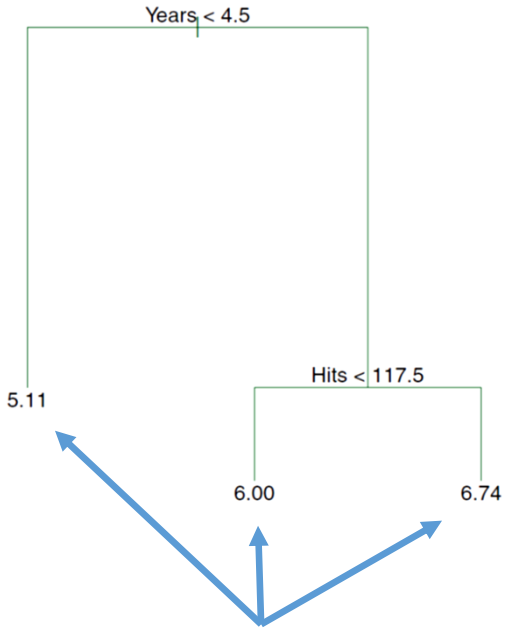
c. Decision tree advantage

A (simple) decision tree is easy to interpret

- Years played is arguably most important for salary
- Early on in player's career, number of hits does not matter much for wage
- Later in career, number of hits becomes for earned salary

Note the nonlinearity!

- Trees are arguably easier to explain to non-quants than regressions
- Nonlinearity simple here, in regression would need interaction term (even harder to explain)



Terminal nodes (also called "*leaves*") give average in each *box (log of salary/1000)*

UCLAAnderson
SCHOOL of MANAGEMENT

c. Decision trees: objective function

Let's first fix the number of ***terminal nodes*** or ***boxes*** or ***leaves*** to be ***J***

- Define the average of the observations in box R_j as $\hat{y}_{R_j}$
- The objective function is then to find the set of boxes that ***minimizes the sum of squared errors*** (SSE):

$$\min_{\{R_j\}_j} \sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j} \right)^2$$

Easy right?

- No. With J = 2 (two boxes) and N observations, there are order of *N* possible splits. With J = 3, there are order of N^2 possible splits, etc.
- Not at all obvious how to find the right partitions of the data that achieves the minimum if ***J*** is not small…
  - The complexity of the problem grows exponentially

UCLAAnderson
SCHOOL of MANAGEMENT

c. Decision trees: recursive binary splitting

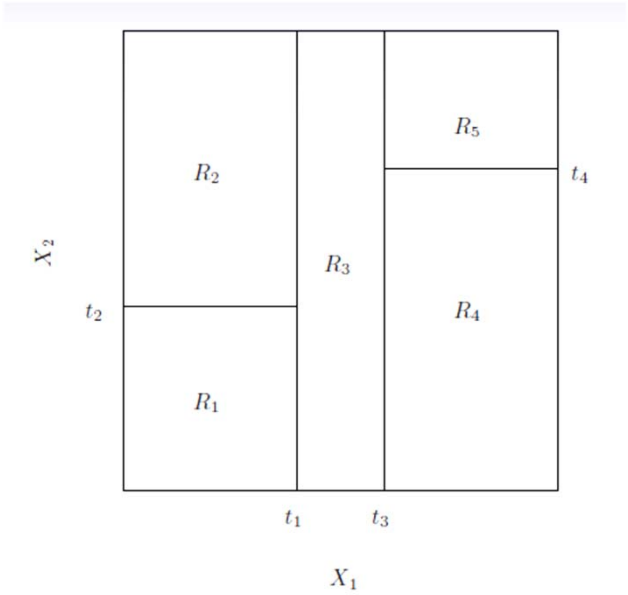
A ***top-down, greedy*** approach

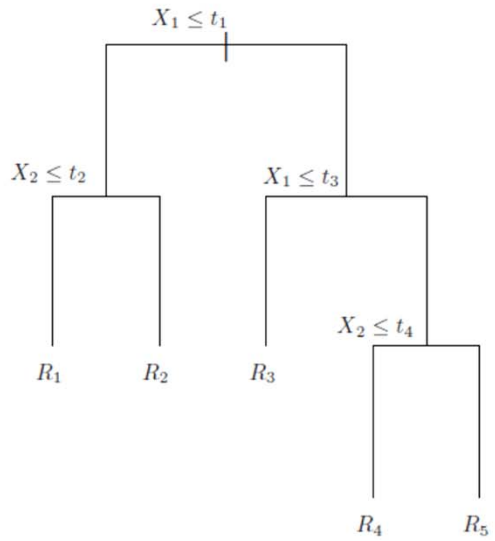
- Start with only one box (at the top), then add another, add another, etc.
- At each step the best split is made given that particular box, no looking ahead to find the globally optimal tree (this is the greedy part)

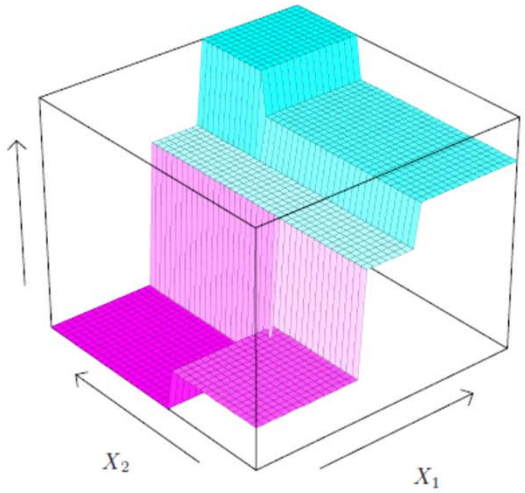
1. Select one of the predictors, X_j
2. Find the one split (creating two boxes out of all the data) that minimizes the sum of squares. Save the breakpoint $X_j = s$ for each j
3. Loop through all the predictors and choose the predictor that leads to the lowest SSE out of all the predictors. In our example, that was Years (and not Hits). This defines the first break point and the first two boxes.
4. Now, start over, for each predictor, find the split that minimizes the SSE. Note that this split can happen within any of the boxes already created. In our case, the best split, in terms of minimizing SSE was for the variable "Hits" in the Year > 4.5 region at the breakpoint Hits = 117.5.
5. Keep going, creating more boxes in the same way as given in 4., until a convergence criterion is met. For instance, one could require that no region contains more than, say, ten observations, or that the number of end nodes should be 30, etc.

UCLAAnderson
SCHOOL of MANAGEMENT

c. Decision trees: a 5-box example







5 breakpoints given by t_j, variables are X_1 and X_2

UCLAAnderson
SCHOOL of MANAGEMENT

c. Trees versus linear models

A linear regression assumes the model:

$$f(X) = \beta_0 + \sum_{j=1}^{P} X_j \beta_j$$

whereas a tree-structure assume a model of the form:

$$f(X) = \sum_{m=1}^{M} \beta_m \cdot 1_{(X \in R_m)}$$

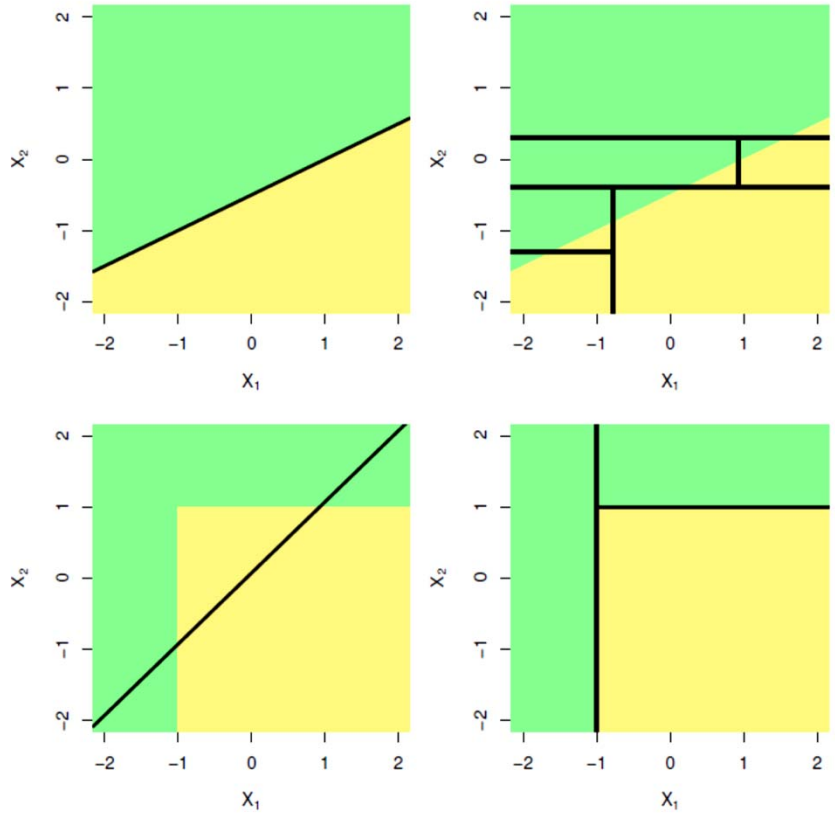
Note key differences

- Dummy variables allow for non-linear 'buckets' (ref: earlier examples)
- The definition of the dummy-variables is endogenous to the procedure
  - In contrast, in a regression setting variables are chosen in a first (often somewhat ad hoc) step, regression is then run in a second step

UCLAAnderson
SCHOOL of MANAGEMENT

c. Trees versus linear models

If true model is linear, trees will do worse than regression (obviously) and vice versa. Trees account better for nonlinearities.



The plot is of a categorization of a variable as green or yellow. Top row has a true linear boundary, bottom row true nonlinear boundary. Left column has linear model, right column has tree-based model.

UCLAAnderson
SCHOOL of MANAGEMENT

c. Bagging

Bootstrap aggregation (***bagging***)

- With *n* independent observations each with variance σ^2, the variance of the mean is σ^2/n.
- I.e., averaging reduces variance and can therefore improve performance
- Recall, typically with a flexible method bias is low but variance is high

But, we don't have *n* datasets, we only have the one we're working with

- Use ***bootstrap***, taking *B* repeated samples (typically with replacement) from the original dataset and fit *B* trees.
- Then ***average*** all the ***predictions***:

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

UCLAAnderson
SCHOOL of MANAGEMENT

c. Out-of-bag error estimation

- Trees that are repeatedly fit to subsets of observations using the bootstrapping technique, on average use only 2/3s of the data for each tree
- The other 1/3 of the data is “out of the bag” (abbreviated “OOB”) and can be used for cross-validation: how big is the error of the fitted model on this data?
- Average across all bootstraps to get the OOB mean squared error

UCLAAnderson
SCHOOL of MANAGEMENT

c. Random forests

- Initially, we motivated bagging by noting that with independent samples, the variance drops with 1/nrsamples.
- However, when bootstrapping from the same data, the samples are positively correlated as the ultimately come from the same data.
  - If samples are perfectly correlated, there is obviously no effect from bagging.
- Random Forests were suggested as a way to ***de-correlate the samples***.
- In particular, a random selection of *m* of the *p* predictors are chosen for each bootstrapped sample. This reduces correlation due to less overlap (e.g., two samples can have completely disjoint sets of predictors)
- Often, *m* is set to equal $\sqrt{p}$
- As with classic bagging, one in the end averages the prediction from all the trees. This is a state-of-the-art method with in practice good predictive properties, though the approach essentially leads to a black-box predictor

UCLAAnderson
SCHOOL of MANAGEMENT

c. Random forests: Trading model example

- Let's revisit the stock return and accounting data from Topic 2
  - Recall: annual next year stock return and (appropriately lagged) firm characteristics such as profitability, leverage, book-to-market ratio, etc.
  - Sample from 1981-2015 (stock returns; predictive variables one year lagged)

- We will let the computer create a trading model that using all the predictors and all the stock returns using

1. Linear regressions
2. Random Forests

UCLAAnderson
SCHOOL of MANAGEMENT

c. Reading and preparing the data

```
# Download data and set as data.table
StockRetAcct_DT <- as.data.table(read.dta("StockRetAcct_insample.dta"))

# remove any rows that contain missing data
StockRetAcct_DT <- StockRetAcct_DT[complete.cases(StockRetAcct_DT),]

# train model on data up until 2010
y_data_train <- as.vector(StockRetAcct_DT[year<2010,ExRet])
x_data_train <- as.matrix(StockRetAcct_DT[year<2010,list(lnIssue,lnMom,lnME,lnProf,
    + lnEP,lnInv,lnLever,lnROE,rv,lnBM,ff_ind,year)])

# test data is data from 2010 and on
y_data_test <- as.vector(StockRetAcct_DT[year>=2010,ExRet])
x_data_test <- as.matrix(StockRetAcct_DT[year>=list(lnIssue,lnMom,lnME,lnProf,
    + lnEP,lnInv,lnLever,lnROE,rv,lnBM,ff_ind,year)])
```

- Note the construction of a testing sample (2010 and on) for the panel of stock returns and characteristics
- Note using “complete.case()” to toss out any row that has any missing data

UCLAAnderson
SCHOOL of MANAGEMENT

c. Estimating Random Forest model

```
> rfLC <- randomForest(x_data_train, y_data_train, ntree = 500, maxnodes = 30)

> # check prediction fit in- and out-of-sample
> RF_pred_in_sample <- predict(rfLC, x_data_train)
> RF_test_in_sample <- lm(y_data_train~RF_pred_in_sample)
> summary(RF_test_in_sample)


Call:
lm(formula = y_data_train ~ RF_pred_in_sample)

Residuals:
    Min      1Q  Median      3Q     Max
-1.4741 -0.2482 -0.0464  0.1801  9.7590

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       -0.056838   0.002576  -22.07   <2e-16 ***
RF_pred_in_sample  1.750092   0.021831   80.16   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.442 on 50233 degrees of freedom
Multiple R-squared:  0.1134,   Adjusted R-squared:  0.1134
F-statistic:  6426 on 1 and 50233 DF,  p-value: < 2.2e-16
```

- Estimate Random Forest model with 500 trees (ntree) where each tree has a maximum of 30 terminal nodes (to avoid too much overfitting)
- Note high in-sample R2 (11.34%) due to trees being quite flexible…

UCLAAnderson
SCHOOL of MANAGEMENT

c. Random Forest out-of-sample

```
➢ RF_pred_out_of_sample <- predict(rfLC, x_data_test)
➢ RF_test_out_of_sample <- lm(y_data_test~RF_pred_out_of_sample)
➢ summary(RF_test_out_of_sample)
```

```
Call:
lm(formula = y_data_test ~ RF_pred_out_of_sample)

Residuals:
    Min      1Q  Median      3Q     Max
-1.2164 -0.2074 -0.0260  0.1705  4.0788

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)           0.14606    0.01126   12.98  < 2e-16 ***
RF_pred_out_of_sample 0.79269    0.18696    4.24 2.26e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3839 on 8141 degrees of freedom
Multiple R-squared:  0.002203, Adjusted R-squared:  0.002081
F-statistic: 17.98 on 1 and 8141 DF,  p-value: 2.261e-05
```

- Actual out-of-sample R2 is 0.2081%.
- Doesn’t seem high, but recall this is for individual stocks
- Coefficient is about 1, so looks ok in that sense

UCLAAnderson
SCHOOL of MANAGEMENT

c. Estimating panel linear regression model

```
> # compare to linear regression
> x_lm <- x_data_train
> LR_test_in_sample <- lm(y_data_train~x_lm)
> summary(LR_test_in_sample)

Call:
lm(formula = y_data_train ~ x_lm)

Residuals:
    Min      1Q  Median      3Q     Max
-1.2925 -0.2618 -0.0443  0.1917 10.1886

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.254e+00  5.995e-01   7.096 1.30e-12 ***
x_lmlnIssue  9.627e-03  9.962e-03   0.966 0.333876
x_lmlnMom   -5.726e-02  5.683e-03 -10.074  < 2e-16 ***
x_lmlnME    -1.357e-02  1.715e-03  -7.912 2.59e-15 ***
x_lmlnProf   9.633e-02  1.139e-02   8.454  < 2e-16 ***
x_lmlnEP    -7.834e-02  1.358e-02  -5.767 8.09e-09 ***
x_lmlnInv   -1.352e-01  9.438e-03 -14.324  < 2e-16 ***
x_lmlnLever  6.507e-05  3.157e-03   0.021 0.983558
x_lmlnROE    4.188e-02  1.680e-02   2.493 0.012684 *
x_lmrv      -3.588e-02  1.338e-02  -2.682 0.007311 **
x_lmlnBM     2.153e-02  3.073e-03   7.007 2.47e-12 ***
x_lmff_ind   2.428e-03  6.355e-04   3.820 0.000134 ***
x_lmyear    -1.995e-03  3.059e-04  -6.523 6.97e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4648 on 50222 degrees of freedom
Multiple R-squared:  0.01985,    Adjusted R-squared:  0.01961
```

- Note linear panel regression has much lower in-sample R2 (1.985% vs 12%)

UCLAAnderson
SCHOOL of MANAGEMENT

c. Linear regression model out-of-sample

```
> x_lm <- x_data_test
> LR_pred_out_of_sample <- predict(LR_test_in_sample, as.data.frame(x_lm))
> LR_test_out_of_sample <- lm(y_data_test~LR_pred_out_of_sample)
> summary(LR_test_out_of_sample)

Call:
lm(formula = y_data_test ~ LR_pred_out_of_sample)

Residuals:
    Min      1Q  Median      3Q     Max
-1.1823 -0.2082 -0.0259  0.1705  4.0449

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)           0.188938   0.004964  38.060   <2e-16 ***
LR_pred_out_of_sample 0.042496   0.083072   0.512    0.609
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3843 on 8141 degrees of freedom
Multiple R-squared:  3.214e-05,  Adjusted R-squared:  -9.069e-05
```

- Out-of-sample R2 is essentially zero.
- Coefficient on prediction should be one, but is close to zero both economically and statistically
- Indicates linear model does not uncover true stable relation between predictors (X) and future excess returns (Y) while nonlinear Random Forest does much better. Note that the actual prediction in Random Forest is so opaque (the average of 500 trees) that it’s hard to understand exactly *why* it performs better

UCLAAnderson
SCHOOL of MANAGEMENT

c. Simple trading models: Random Forest

- Given these predictions, let's (as in Topic 3) run the Fama-MacBeth regressions to get the Sharpe ratios and average excess returns on the out-of-sample trading strategies implied by our expected return signals
- Note RF in the below means we are using Random Forest predictions as signal
- Negative out of sample Sharpe ratio! Bummer…

```
> # Fama-MacBeth regressions using out-of-sample predicted values
> # Random Forest first
> port_ret = NULL
> row_counter_end = 0
> for (i in 2010:2014)
   + { + year_length <- StockRetAcct_DT[year==i,ExRet]
   + year_length <- length(year_length)
   + row_counter_start = row_counter_end + 1
   + row_counter_end = row_counter_end + year_length
   + x_temp <- RF_pred_out_of_sample[row_counter_start:row_counter_end]
   + y_temp <- y_data_test[row_counter_start:row_counter_end]
   + fit_yr <- lm(y_temp ~ x_temp)
   + temp <- coefficients(fit_yr)
   + port_ret = rbind(port_ret,temp[2])
+ }

> # recall, scale depends on magnitude of x-variable, so only fair to compare Sharpe ratios and t-stats
> fm_RF_output = list(SR_Return = mean(port_ret)/sqrt(var(port_ret)),
> + tstat_MeanRet = sqrt(1+2014-2010)*mean(port_ret)/sqrt(var(port_ret)))
> fm_RF_output
$SR_Return
           x_temp
x_temp -0.3599666

$tstat_MeanRet
           x_temp
x_temp -0.8049098
```

Try controlling for industry in the FMB regression as we did in Topic2

- Better results??

UCLAAnderson
SCHOOL of MANAGEMENT

c. Simple trading models: Linear regression

- Now, run the trading strategy for the panel linear regression case
- Note LR in the below means we are using linear reg predictions as signal
- About the same (poor) results as for Random Forest in this case

```
> # Fama-MacBeth regressions using out-of-sample predicted values
> # Random Forest first
> port_ret = NULL
> row_counter_end = 0
> for (i in 2010:2014)
   + { + year_length <- StockRetAcct_DT[year==i,ExRet]
   + year_length <- length(year_length)
   + row_counter_start = row_counter_end + 1
   + row_counter_end = row_counter_end + year_length
   + x_temp <- LR_pred_out_of_sample[row_counter_start:row_counter_end]
   + y_temp <- y_data_test[row_counter_start:row_counter_end]
   + fit_yr <- lm(y_temp ~ x_temp)
   + temp <- coefficients(fit_yr)
   + port_ret = rbind(port_ret,temp[2])
+ }

> # recall, scale depends on magnitude of x-variable, so only fair to compare Sharpe ratios and t-stats
> fm_RF_output = list(SR_Return = mean(port_ret)/sqrt(var(port_ret)),
> + tstat_MeanRet = sqrt(1+2014-2010)*mean(port_ret)/sqrt(var(port_ret)))
> fm_RF_output
                $SR_Return
                           x_temp
                x_temp -0.3584438

                $tstat_MeanRet
                           x_temp
                x_temp -0.8015047
```

UCLAAnderson
SCHOOL of MANAGEMENT

c. Boosting

Boost: "to increase or improve"

- Quite different from bagging
  - Bagging involved overfitting of each individual tree, but then averaging to get rid of noise in samples with low correlation
  - Boosting fits smaller trees and instead learns slowly by sequentially adding small trees fit to the prediction errors of the existing 'ensemble' of trees.
  - Thus, each tree added depends on the trees already grown
- Sometimes referred to as an '*ensemble method.*' It creates a strong predictor based on many weak predictors.

Three tuning parameters:

1. The number of trees, ***B*** (or γ). If very large can over-fit, use cross-validation to find this variable
2. Shrinkage parameter, λ, determines the rate at which we are learning (adding new trees). Small λ can mean large *B* is needed to fit data well.
3. The number of splits in each tree, ***d*** – *interaction depth*. If $d = 1$ we are fitting an additive model (no interactions); each tree is just a single split (a stump!)

c. Boosting: General Algorithm

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.

2. For $b = 1, 2, \ldots, B$, repeat:

   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).

   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$$

   (c) Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$$

This is ***gradient boosting***. You are improving the model by minimizing residuals one step (tree) at a time (like gradient optimization methods)

- *The loss function can be SSE, but need not..!*

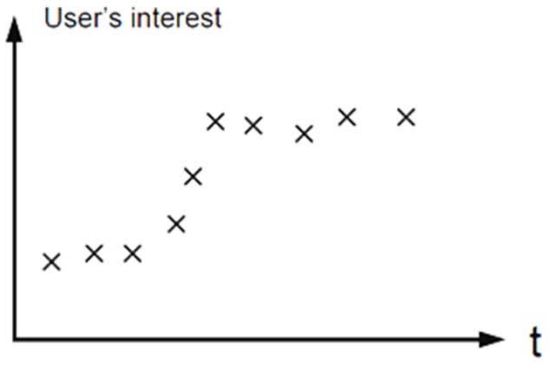
UCLAAnderson
SCHOOL of MANAGEMENT

c. XGBoost

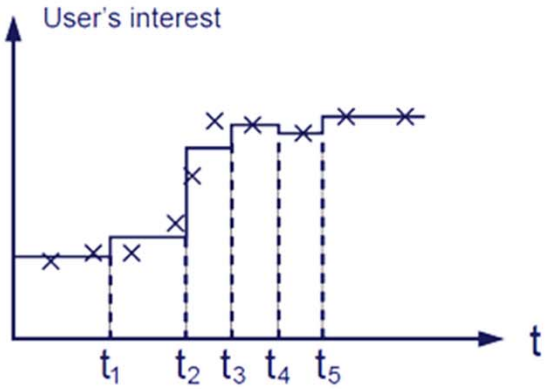
- There are several different boosting algorithms (e.g., AdaBoost)
- Recent research as well as 'word on the street' has shown that XGBoost seems to be the best performer
- XGBoost: e**X**treme **G**radient **B**oosting
  - High speed, high accuracy
  - Supports gradient boosting, stochastic gradient boosting, and regularized gradient boosting
  - The regularization part, which controls over-fitting, is what stands out the most and is the main source of outperformance
- Let's consider regularization with trees
  - Next slide

c. Regularization with trees

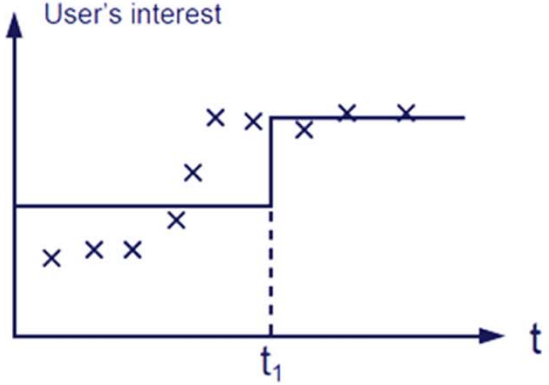
- Denote $\Omega(f)$ as the ***complexity*** of tree
  - We will define this on next slide, but for now just think of this intuitively
- Denote $L(f)$ as the ***prediction error loss function***
  - E.g., variance of prediction errors; again think of it intuitively



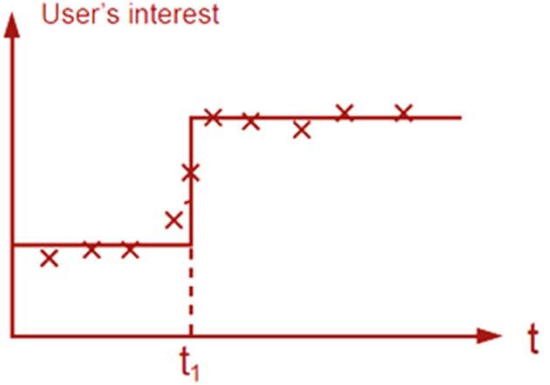
Observed user's interest on topic k against time t



☒ Too many splits, $\Omega(f)$ is high



☒ Wrong split point, $L(f)$ is high



☑ Good balance of $\Omega(f)$ and $L(f)$

UCLAAnderson
SCHOOL of MANAGEMENT

c. XGBoost: General objective function

- At a high level, think of the objective function over a set of decisions/parameters Θ as:

$$\min Obj(\Theta) = \min\big(L(\Theta) + \Omega(\Theta)\big)$$

- Here $L(\Theta)$ is the loss function and $\Omega(\Theta)$ is the regularization term
- A common loss function for the tree is

$$L = \sum_i (y_i - \hat{y}_i)^2$$

- A regularization term defines tree complexity as:

$$\Omega(f) = \gamma N + \frac{1}{2}\lambda \sum_{n=1}^{N} \beta_n^2$$

- Can also have a Lasso type constraint, or elastic net even

c. XGBoost: Tree complexity

- Repeated from last slide:

$$\Omega(f) = \gamma N + \frac{1}{2}\lambda \sum_{n=1}^{N} \beta_n^2$$

- *N* is the number of leaves (boxes, terminal nodes), *f* is the prediction function:

$$f = \sum_{n=1}^{N} \beta_n \cdot 1_{(X \in R_n)}$$

- Thus, there are two regularization parameters: γ and λ.
- Notice this looks a lot like ridge regularization (L2 regularization), where there is an additional penalty for the size of the tree *(N)*
  - *For proper math background, see XGBoost White Paper in CCLE*

UCLAAnderson
SCHOOL of MANAGEMENT

c. XGBoost: A worked example

- Let's go back to the usual return forecasting problem
- This time, we will see how XGBoost performs, compared to Random Forest and the standard Linear Panel model

```
➢ params <- list(booster = "gbtree", objective = "reg:linear", eta = 0.3, gamma = 0, max_depth = 20)
➢ # XGBoost likes to use xgb.Dmatrix
➢ xgb_train <- xgb.DMatrix(data = x_data_train, label = y_data_train)
➢ # use xgb.cv to find the best nround (number of trees) for this model.
➢ xgbcv <- xgb.cv(params = params, data = xgb_train, nfold = 10, nrounds = 100, showsd = T, print_every_n = 10)
```

- We set gamma = 0, don't regularize as we are performing cross-validation using xgb.cv to find B (or lambda, or nrounds). (But, you are free to try higher values of gamma)
- nrounds gives the maximal number of trees (we will find this using cv)
- eta is how much weight you give to each new tree in prediction, vs old prediction
  - Gets at speed of learning, lambda in the constraint given earlier
- Max_depth is the max number of splits per tree one allows
  - This is where interactions and other nonlinearities come into play

UCLAAnderson
SCHOOL of MANAGEMENT

c. XGBoost: A worked example

- Get the best nround from cross-validation procedure

```
cv_nrounds = which.min(xgbcv$evaluation_log$test_rmse_mean)

# with optimal nrounds in hand, run the prediction for out of sample
xgb_optb <- xgboost(params = params, data = xgb_train, nround = cv_nrounds)
```

- `# define true out of sample data`
- `xgb_test <- xgb.DMatrix(data = x_data_test, label = y_data_test)`
- `# get predicted values in the true out of sample period`
- `xgb_pred <- predict(xgb_optb, xgb_test)`
- `xgb_test_out_of_sample <- lm(y_data_test~xgb_pred) > summary(xgb_test_out_of_sample)`

```
Call:
lm(formula = y_data_test ~ xgb_pred)

Residuals:
    Min      1Q  Median      3Q     Max
-1.2077 -0.2081 -0.0244  0.1694  4.0568

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.165754   0.008163  20.305  < 2e-16 ***
xgb_pred    0.101658   0.028917   3.516 0.000441 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.384 on 8141 degrees of freedom
Multiple R-squared:  0.001516, Adjusted R-squared:  0.001393
F-statistic: 12.36 on 1 and 8141 DF,  p-value: 0.0004413
```

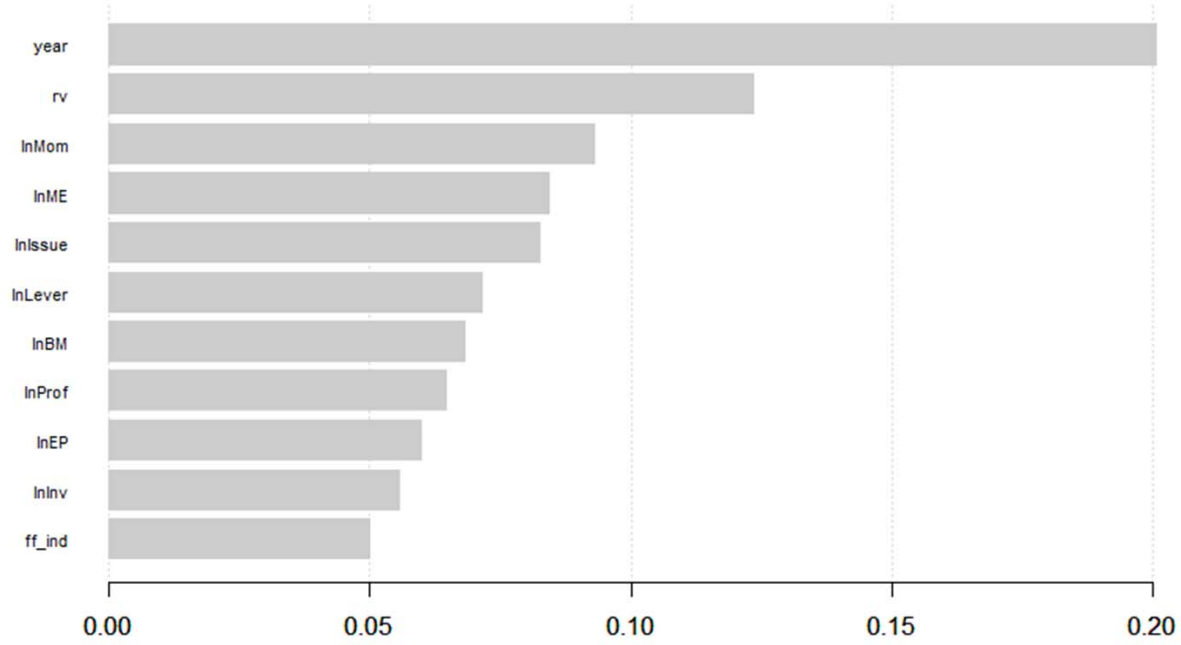
- R2 is only 0.15%, but again this is on individual stocks, not the portfolio

UCLAAnderson
SCHOOL of MANAGEMENT

c. XGBoost: Importance of features

- Use xgb.importance to see which features are most important in terms of "gain"
  - Average improvement in prediction accuracy arising from including feature in a tree

```
# plot importance of features in order to understand model
boost_out <- xgb.importance(feature_names = colnames(x_data_test), model = xgb_optb)
xgb.plot.importance(importance_matrix = boost_out[1:11])
```



- Year comes in as important: i.e., there is a time-trend in data…

UCLAAnderson
SCHOOL of MANAGEMENT

c. XGBoost: Portfolio performance

- Run the usual FMB regressions to get portfolio performance based on this sample
- Use the true out of sample 2010 – 2014 sample

```
$SR_Return
          x_temp
x_temp 0.1794339

$tstat_MeanRet
          x_temp
x_temp 0.4012264
```

- Thus, XGBoost in the case beats Random Forest in terms of out of sample performance, though it is not statistically significant
- Also, do we trust results that in large part are due to a time-trend ('year' consistently shows up as important)?
  - Theory says probably not
  - Data says yes
- In general, you would want to run a gridsearch on min_depth, gamma, eta, and nrounds to get the best cross-validation results.

UCLAAnderson
SCHOOL of MANAGEMENT

c. XGBoost references

- There are many variants of XGBoost
- I have posted a number of resources on CCLE

UCLAAnderson
SCHOOL of MANAGEMENT

c. In sum…

- Decision trees are flexible (high variance, low bias) data-mining models
- Bagging and boosting, in particular Random Forest and XGBoost, make predictions much more accurate
- Downside is opaque nature of averaging across many trees. It can be quite hard to understand why something works. But then, why should we believe it works in the future?
- That said, we constructed an arguably profitable automated out-of-sample trading strategies based on XGBoost that outperformed a trading strategy based on a large panel linear regression model
  - Perhaps that's all that matters? Depends on application.