Boosting

STAT 471

November 11, 2021

Where we are

- ✓ **Unit 1:** Intro to modern data mining
- ✓ **Unit 2:** Tuning predictive models
- ✓ **Unit 3:** Regression-based methods
- **Unit 4:** Tree-based methods
- **Unit 5:** Deep learning

Lecture 1: Growing decision trees

Lecture 2: Tree pruning and bagging

Lecture 3: Random forests

Lecture 4: Boosting

Lecture 5: Unit review and quiz in class

Homework 4 due the following Wednesday.

Looking back and looking ahead

Looking back:

- Decision trees: Great interpretability, but subpar prediction performance
- Bagging and random forests: Aggregating together the predictions of multiple decision trees to reduce variance and improve prediction performance

This lecture: we'll learn about boosting (AKA gradient boosting), another way of aggregating multiple decision trees to get excellent prediction performance.

- Random forests: Grow deep decision trees in parallel
- Boosting: Grow shallow decision trees sequentially

RF grow deep decision trees in parallel, while in boosting we grow shallow trees

Boosting: Learning sequentially

We are given some training data points (X_i, Y_i); suppose continuous response.

Consider a low-complexity weak learner $\hat{f}$, such as a shallow decision tree. We can boost the performance of the weak learner by applying it iteratively:

- First fit the weak learner to the training data to get $\hat{f}_1$ some decision tree
- Let $r_i \leftarrow Y_i - \hat{f}_1(X_i)$ be the residuals, portion of response left over to explain
- Now fit the weak learner to the residuals (X_i, r_i) to get $\hat{f}_2$
- Update the residuals via $r_i \leftarrow r_i - \hat{f}_2(X_i)$
- Repeat B times

The more trees you fit, the better you are going to fit your training data

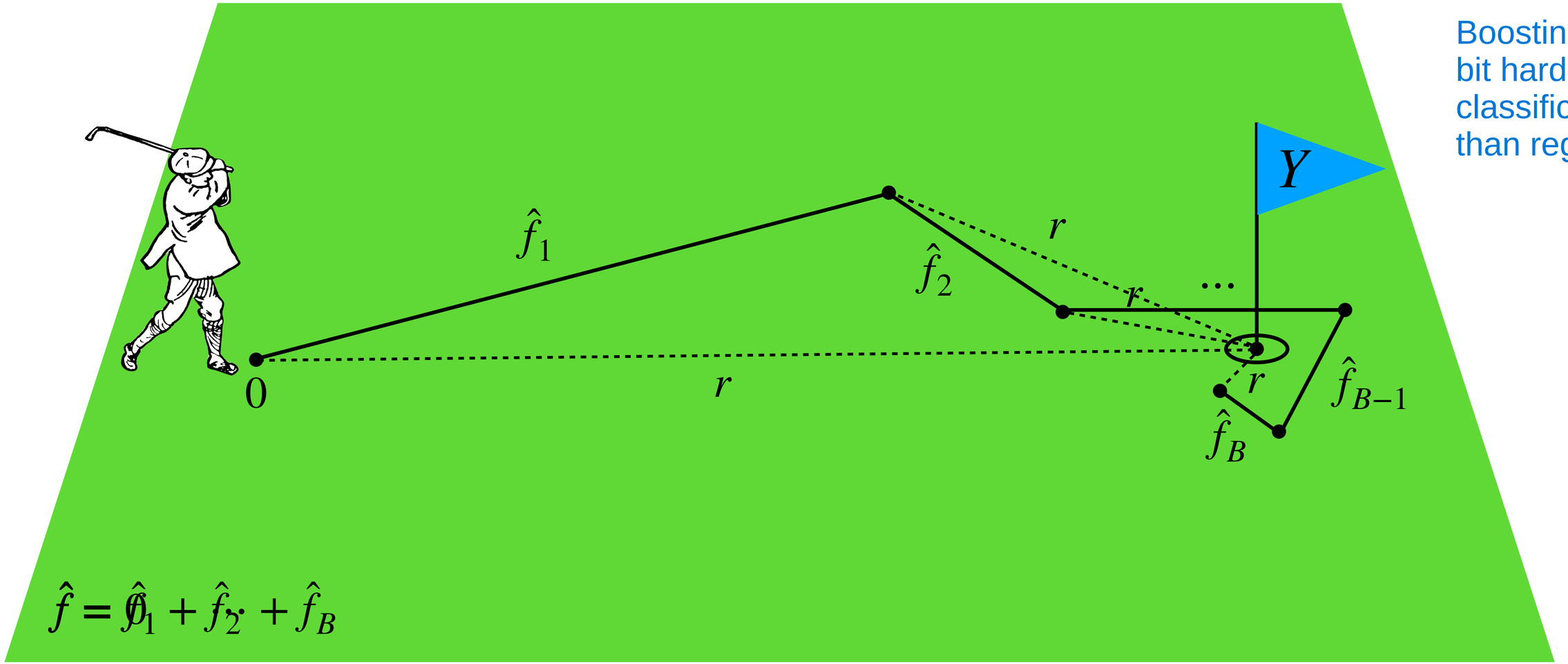
Final prediction rule: $\hat{f} = \hat{f}_1 + \cdots + \hat{f}_B$.

Boosting: An analogy with golf

Golf swing represents fitting to your weak learners/predictors

f1 and f2 are different trees with one split but two trees could different values and factors to split

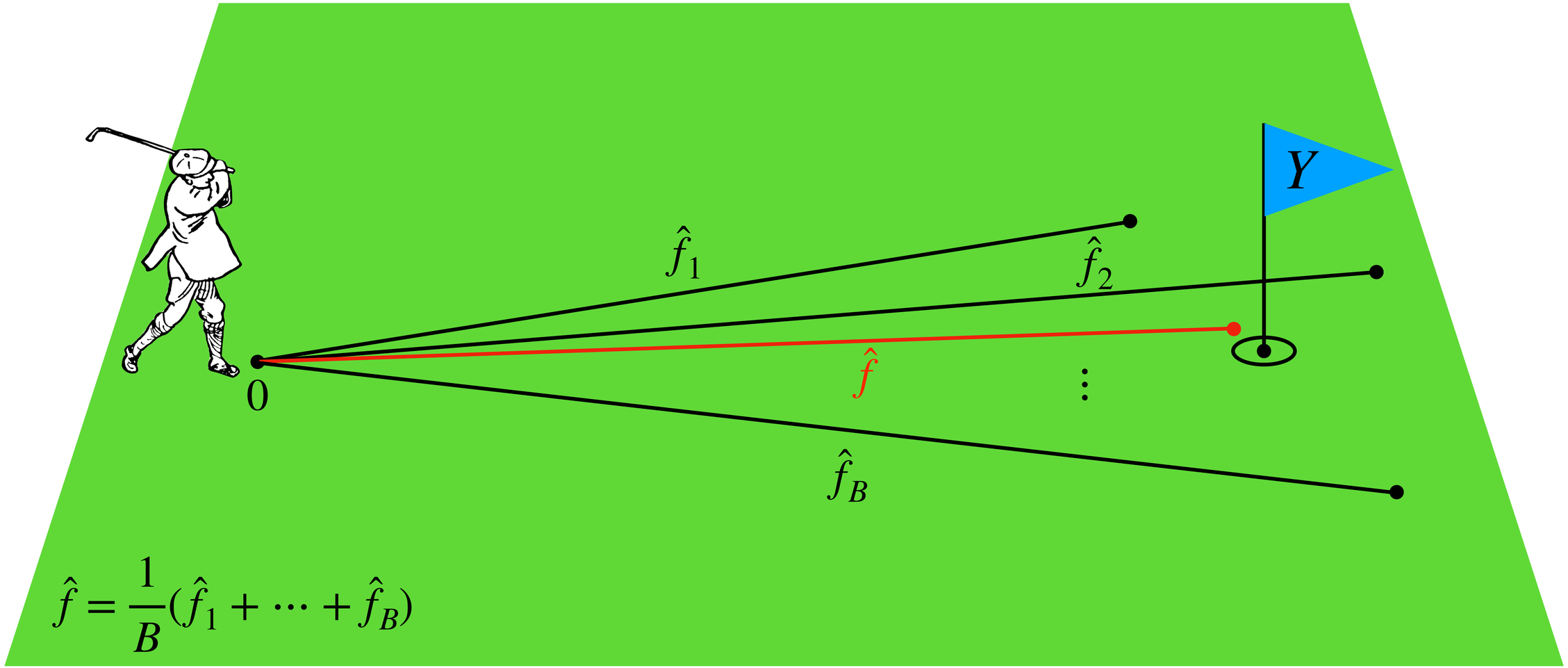
Each golf swing is not too accurate, but a sequence of them is.



Boosting is a bit harder for classification than regression

Comparison to random forests

Try to make each single swing as accurate as possible, then average.

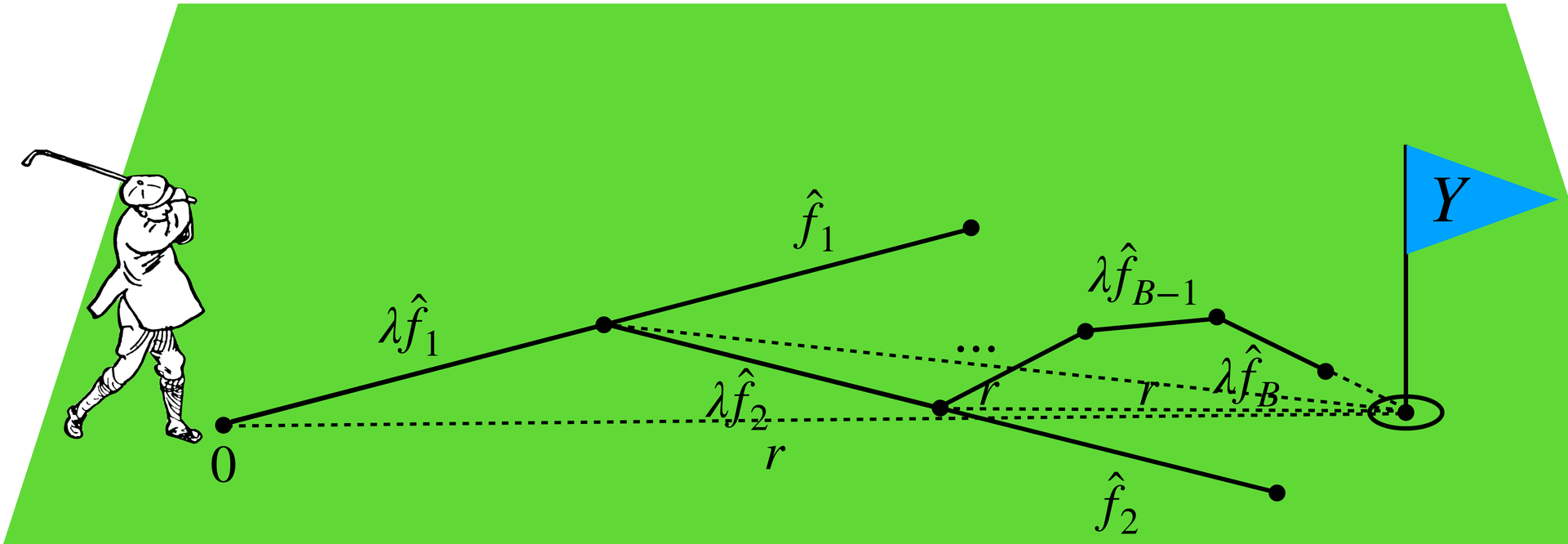


$$\hat{f} = \frac{1}{B}(\hat{f}_1 + \cdots + \hat{f}_B)$$

Boosting with shrinkage

Only travel a fraction of the distance

Let $\lambda \in (0,1]$ be a shrinkage parameter, e.g. 0.5. We only go λ of the way each time.



The essence of boosting is the concept of slow learning

$$\hat{f} = 0\hat{f}_1 + \lambda\hat{f}_2 + \lambda\hat{f}_B$$

Boosting with shrinkage

Inputs: weak learner $\hat{f}$, shrinkage parameter $\lambda \in (0,1]$, number of trees B

- Fit the weak learner to the training data to get $\hat{f}_1$
- Backtrack to $\lambda\hat{f}_1$ and compute residuals $r_i \leftarrow Y_i - \lambda\hat{f}_1(X_i)$
- Fit the weak learner to the residuals (X_i, r_i) to get $\hat{f}_2$
- Backtrack to $\lambda\hat{f}_2$ and update residuals $r_i \leftarrow r_i - \lambda\hat{f}_2(X_i)$
- Repeat B times

Final prediction rule: $\hat{f} = \lambda\hat{f}_1 + \cdots + \lambda\hat{f}_B$.

The parameters λ and B

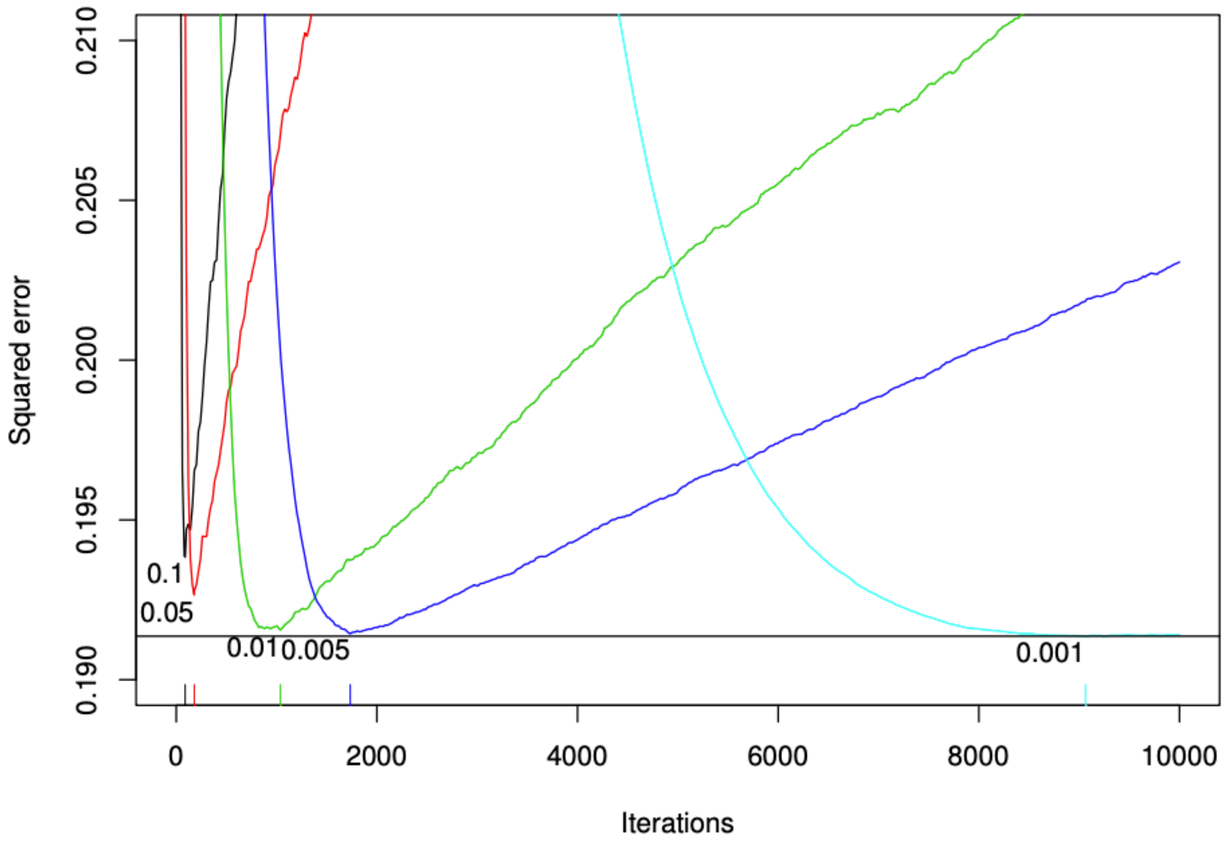
The parameter λ controls how slowly boosting learns.

Learning more slowly tends to give better predictive performance, but requires more iterations B.

The parameter B controls how many iterative refinements are made, so larger B means more model flexibility.

Large enough B can lead to overfitting, unlike random forests.

It you give yourself time to learn slowly, your squared error is lower but you need more iterations to get there

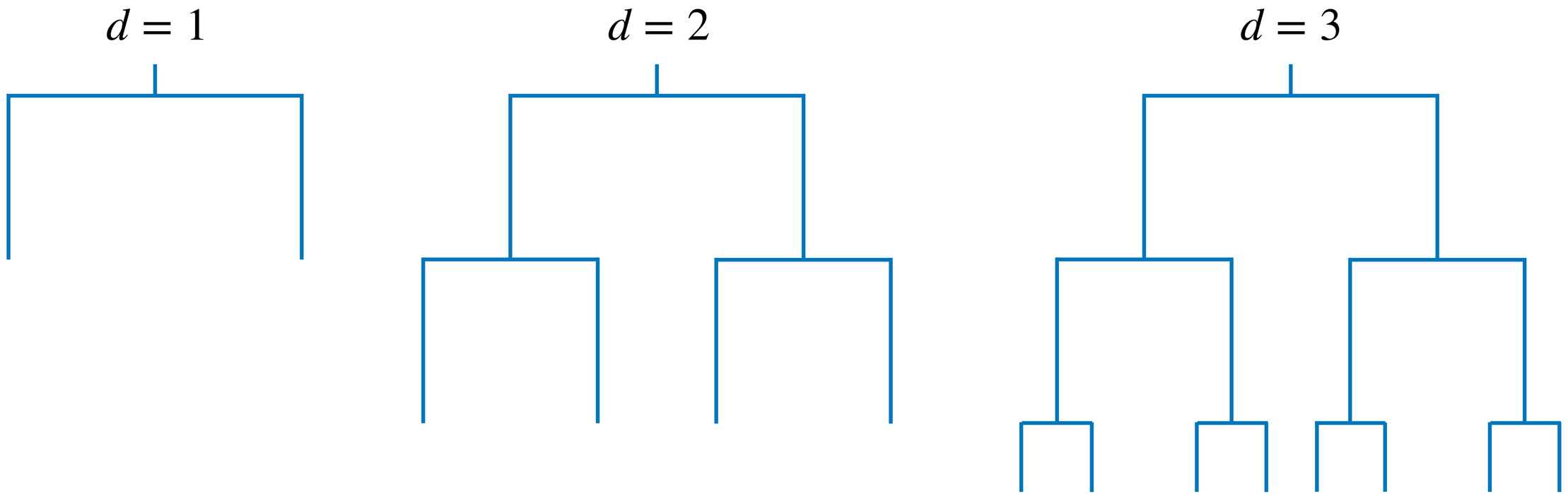


(https://cran.r-project.org/web/packages/gbm/vignettes/gbm.pdf)

Choice of weak learner

Usually, shallow trees are used as the weak learners.

For boosting, tree size usually parameterized by interaction depth d, the maximum number of splits needed to get to a terminal node.



What do interactions have to do with depth?

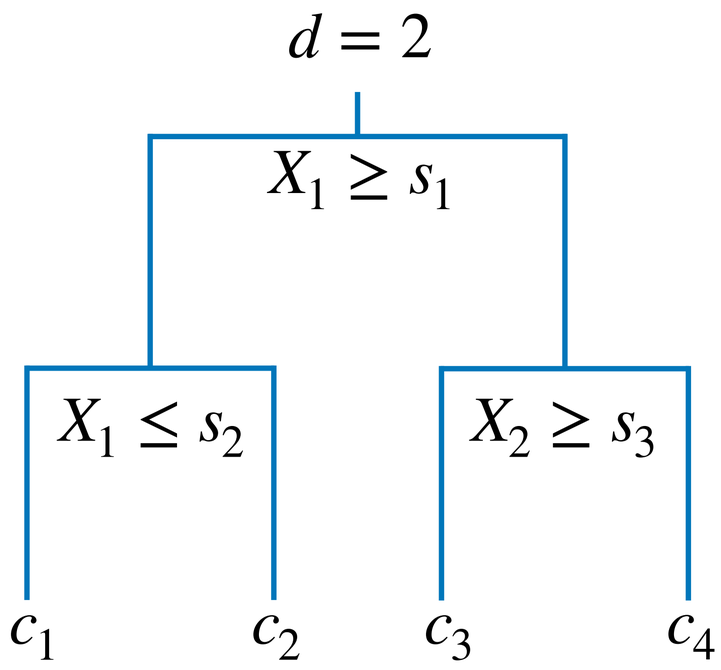
$$\hat{f}(X) = c_1 I(X_1 \geq s_1) I(X_1 \leq s_2) + c_2 I(X_1 \geq s_1) I(X_1 > s_2) + c_3 I(X_1 < s_1) I(X_2 \geq s_3) + c_4 I(X_1 < s_1) I(X_2 < s_3)$$

Consider a single tree, for example with $d = 2$.

Writing out the prediction rule (above), we see we get interaction terms up to degree two in the tree.

For $d = 3$, we can get three-way interactions, etc.

Since the boosting prediction is the sum of tree predictions, if all trees have e.g. $d \leq 2$ the entire forest contains interactions of at most two features.



maximum number of interactions equal to d

Special case: $d = 1$

Stumps do not agree or disagree because they are trying to predict different things

Boosting models prediction is the sum of B things and you could each of these capital B things and can see which ones are a function of just x1 or just x2

This formula is only for d = 1

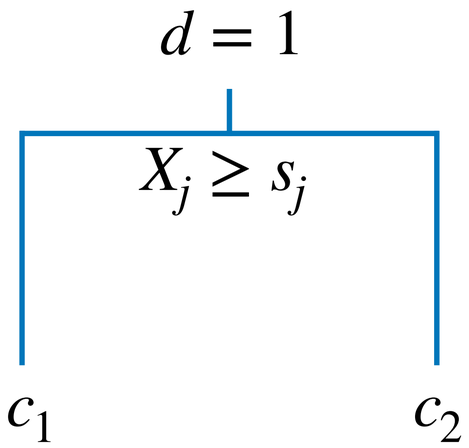
When $d = 1$, each tree has only one split; such trees are called stumps.

Since each tree involves only one feature, the entire boosted model can be viewed as an additive model:

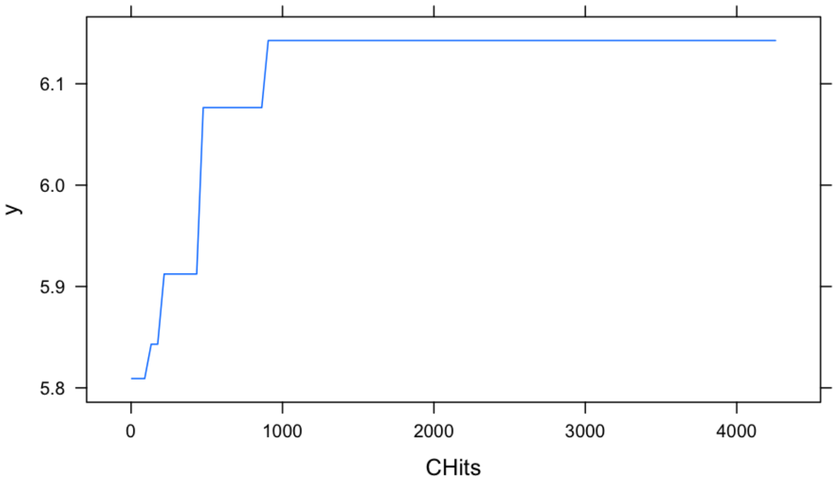
$$\hat{f}(X) = \hat{g}_1(X_1) + \hat{g}_2(X_2) + \cdots + \hat{g}_p(X_p)$$

for some coordinate functions $\hat{g}_j$. The coordinate functions can be easily plotted and interpreted.

Why does whole boosting model look something like this? In each of steps find one variable to split on, and might do multiple times which will be aggregated together.



A “stump”



Stochastic gradient boosting

Same as gradient boosting, except at each iteration, sample only a fraction π of the training observations (with replacement) and train only on those.

Subsampling empirically demonstrated to improve boosting performance.

Subsampling increases variance of individual trees but de-correlates them; benefit of the latter tends to outweigh the former.

A subsampling fraction of $\pi = 0.5$ tends to work well in most cases.

Boosting has two parameters b and d

Proabably doing to fix pi and lambda

Tuning gradient boosting

Parameters:

- Shrinkage parameter λ: Choose some small number, e.g. $\lambda = 0.1$.
- Number of trees B: Choose via cross-validation or using a validation set.
- Interaction depth d: $1 \leq d \leq 5$ usually works well. Try a few values and tune by cross-validation or validation set.
- Subsampling fraction π: Leave at the default of 0.5.

Model interpretation

Hard to interpret in general. However, like RF, we can define purity
notion of variable importance

Variable importance measure:

The purity-based notion of variable importance is also applicable to boosting.

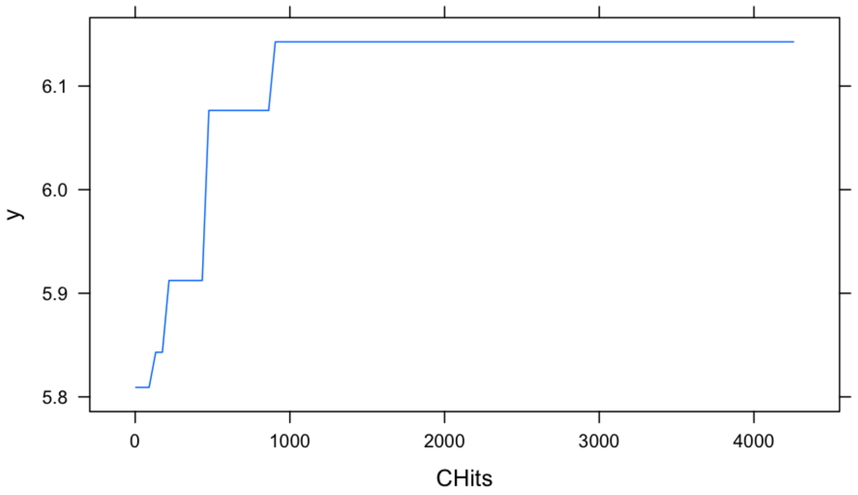
Partial dependence plots:

makes it more interpretable

For $d = 1$, the coordinate functions $\hat{g}_j$ show exactly how $\hat{f}$ depends on X_j.

For general d, can define a generalization of $\hat{g}_j$ via

$$\hat{g}_j(X_j) = \frac{1}{n} \sum_{i=1}^{n} \hat{f}(X_{i,-j}, X_j).$$



d >1 then can define generalization

The larger d is, the worse the approximation because if you have a large d then whatever prediction you are getting is not solely decided by x

Boosting for classification

Boosting is applicable—but a little harder— in the classification context.

The key issue is that there is not an obvious notion of residual in classification.

The exact implementation of boosting for classification is beyond the scope of the class, but in short the idea is to translate classification to a kind of regression problem based on the continuous response $\mathbb{P}[Y = 1]$.

All the same intuitions from this lecture carry over to boosting for classification.

Summary

- Like random forests, boosting aggregates the results of many decision trees to build a predictive model with state-of-the-art performance.
- Unlike random forests, boosting builds the trees sequentially rather than in parallel, using shallow trees rather than deep trees.
- Boosting works best when paired with shrinkage to further slow learning.
- Unlike random forests, the number of trees B controls the complexity of the fit, and therefore must be tuned via cross-validation.
- Purity-based variable importance as well as partial dependence plots help interpret boosting models.