Boosting

STAT 4710

November 7, 2023

Where we are

- ✓ **Unit 1:** R for data mining
- ✓ **Unit 2:** Prediction fundamentals
- ✓ **Unit 3:** Regression-based methods
- **Unit 4:** Tree-based methods
  - **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
- **Unit 5:** Deep learning

Looking back and looking ahead

Looking back and looking ahead

Looking back:

Looking back and looking ahead

Looking back:

- Decision trees: Great interpretability, but subpar prediction performance

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

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.

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

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

Boosting: Learning sequentially

Boosting: Learning sequentially

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

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:

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 $\hat{f}_1$

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 $\hat{f}_1$
- Let $r_i \leftarrow Y_i - \hat{f}_1(X_i)$ be the residuals, portion of response left over to explain

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 $\hat{f}_1$
- 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 $\hat{f}_2$

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 $\hat{f}_1$
- 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 $r_i \leftarrow r_i - \hat{f}_2(X_i)$

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 $\hat{f}_1$
- 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

Boosting: Learning sequentially

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

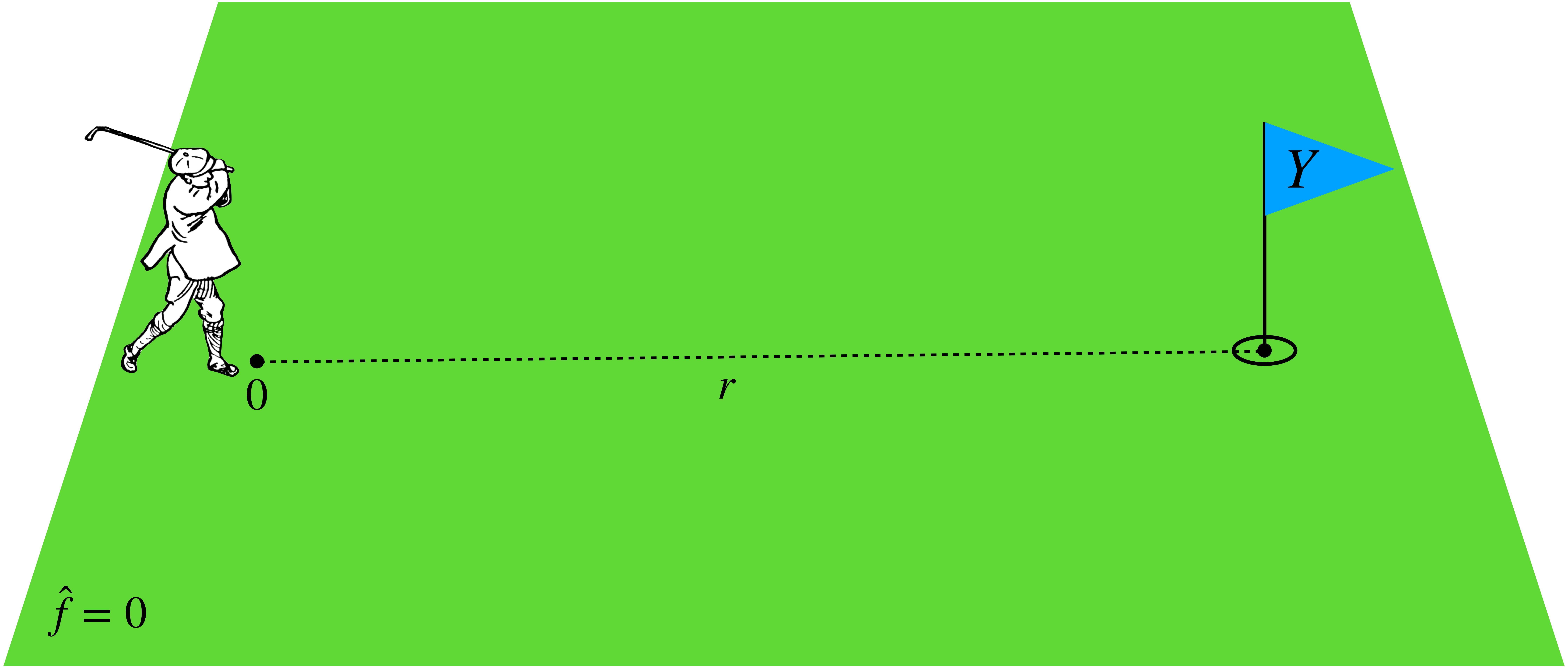
Consider a low-complexity weak learner $\hat{f}$, such as a shallow decision tree. We can boost the performance of the weak learner by applying it iteratively:

- First fit the weak learner to the training data to get $\hat{f}_1$
- 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

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

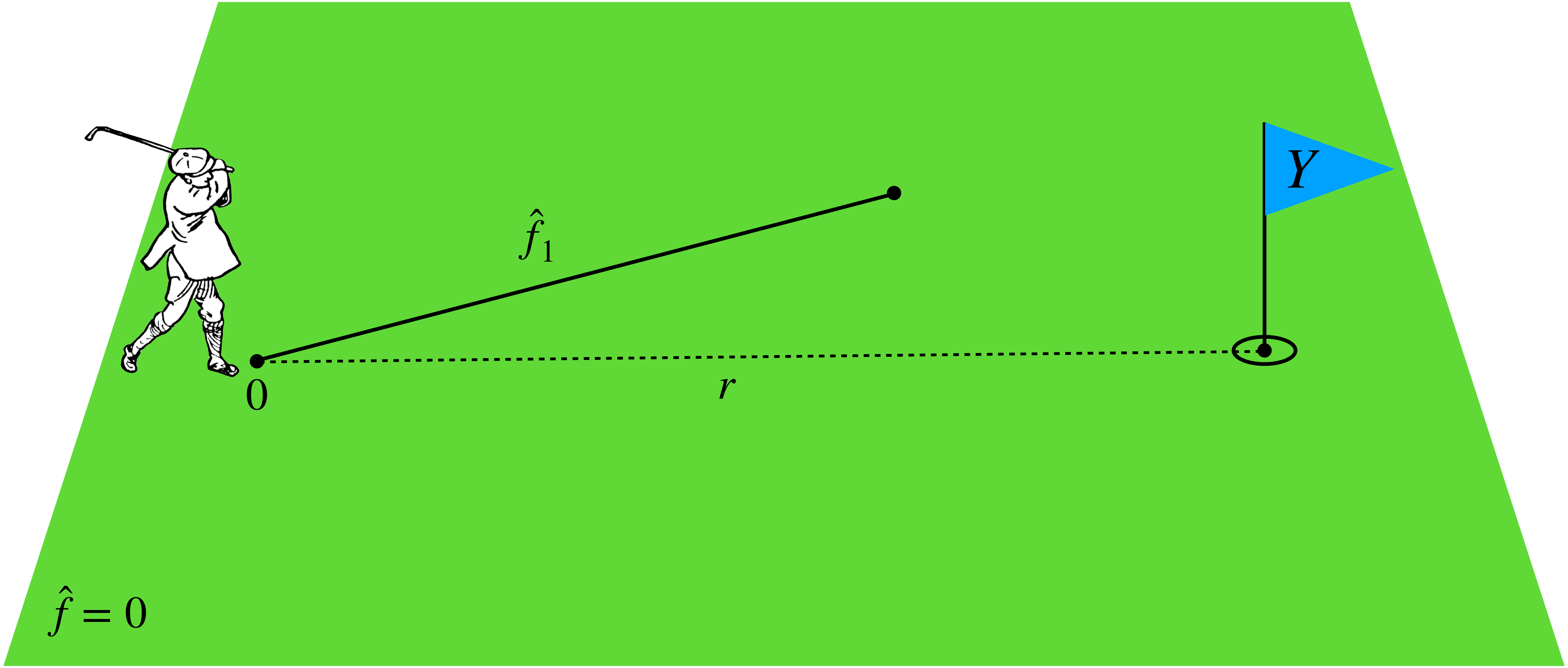
Boosting: An analogy with golf

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



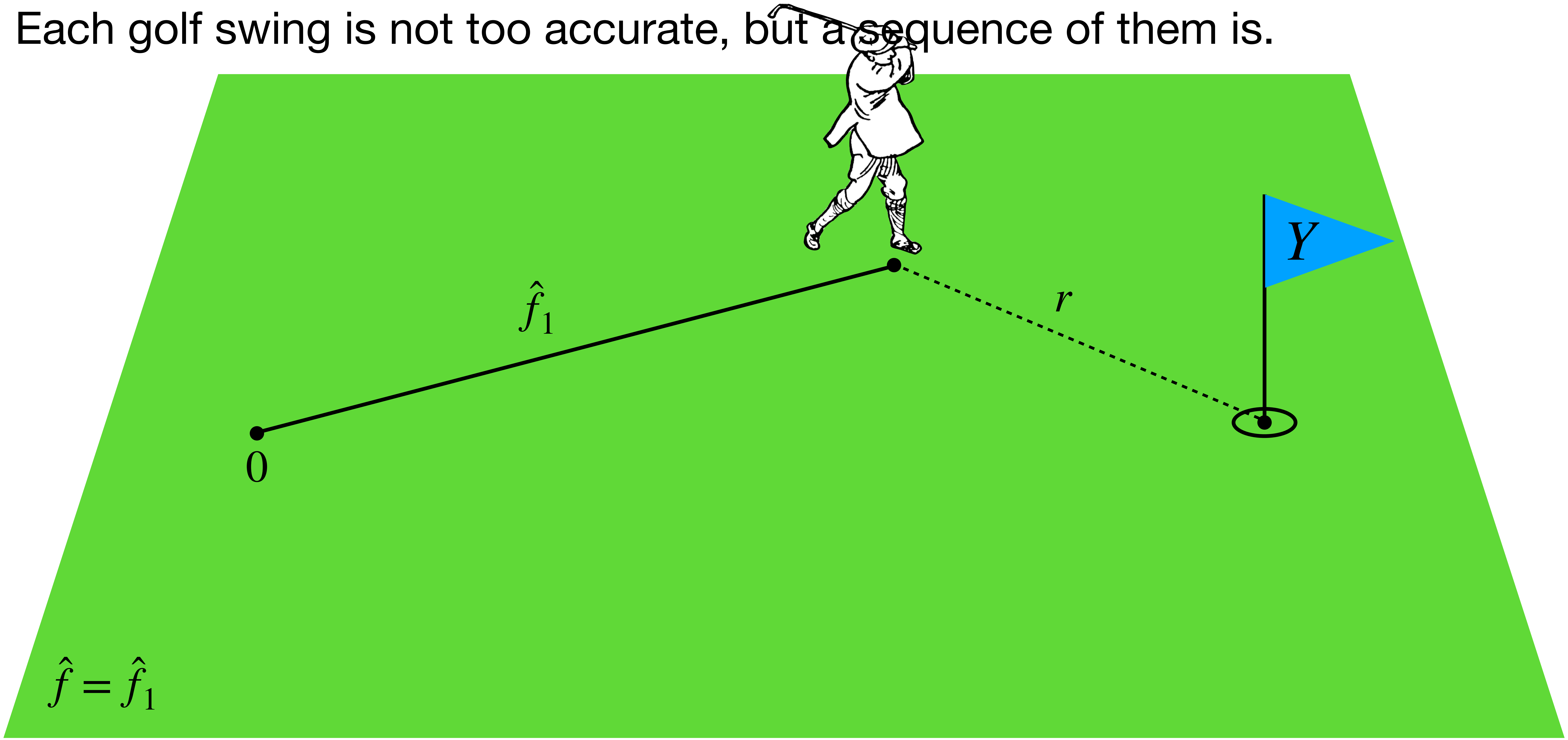
Boosting: An analogy with golf

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



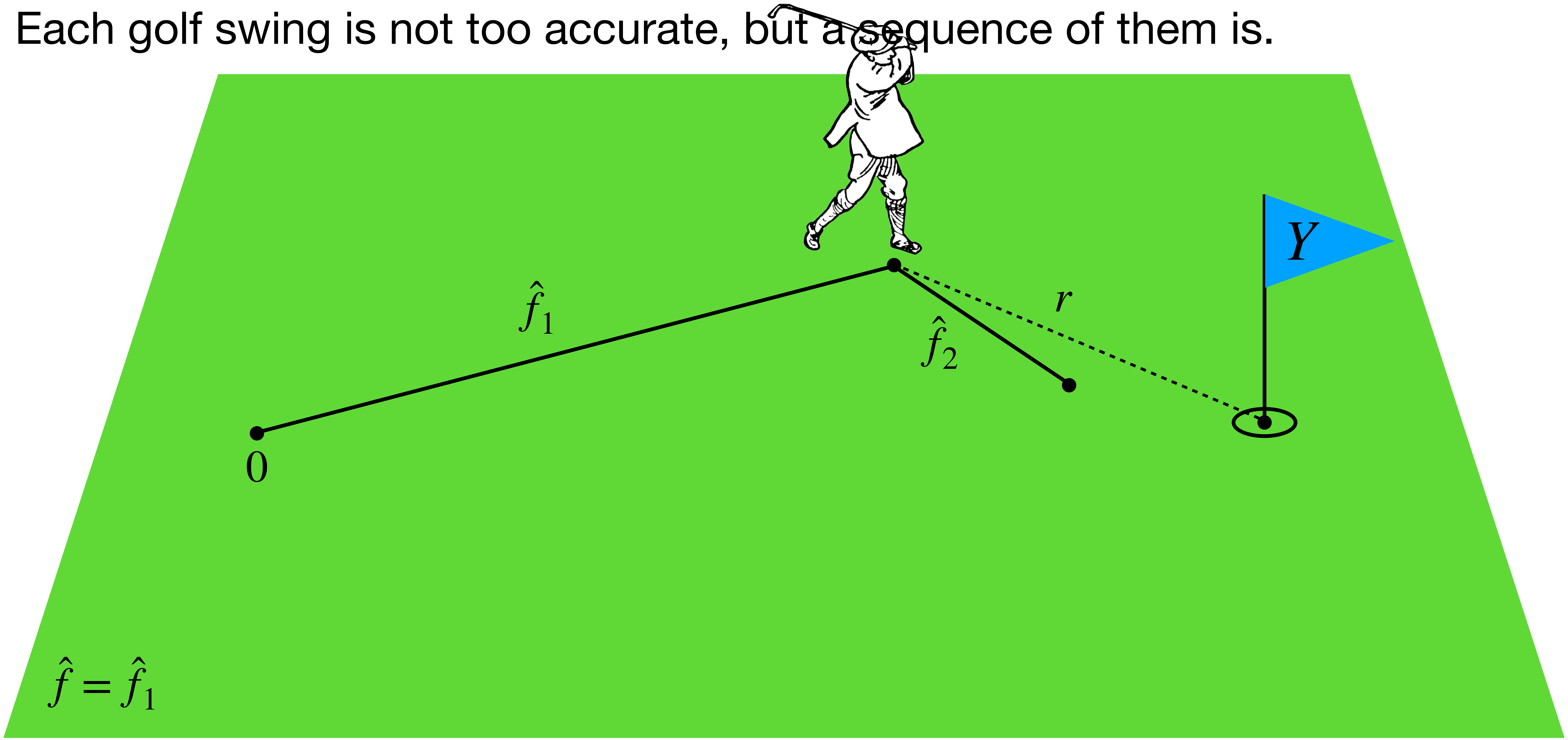
Boosting: An analogy with golf

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



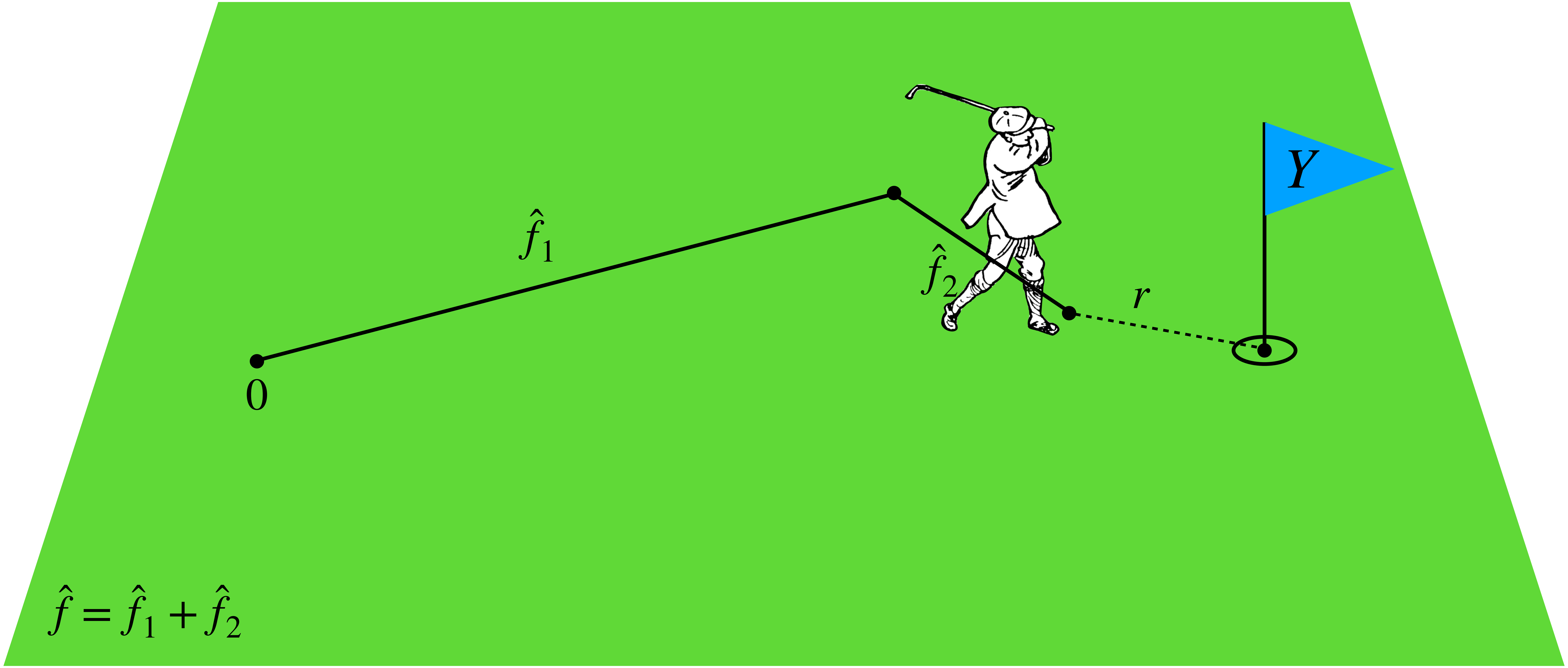
Boosting: An analogy with golf

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



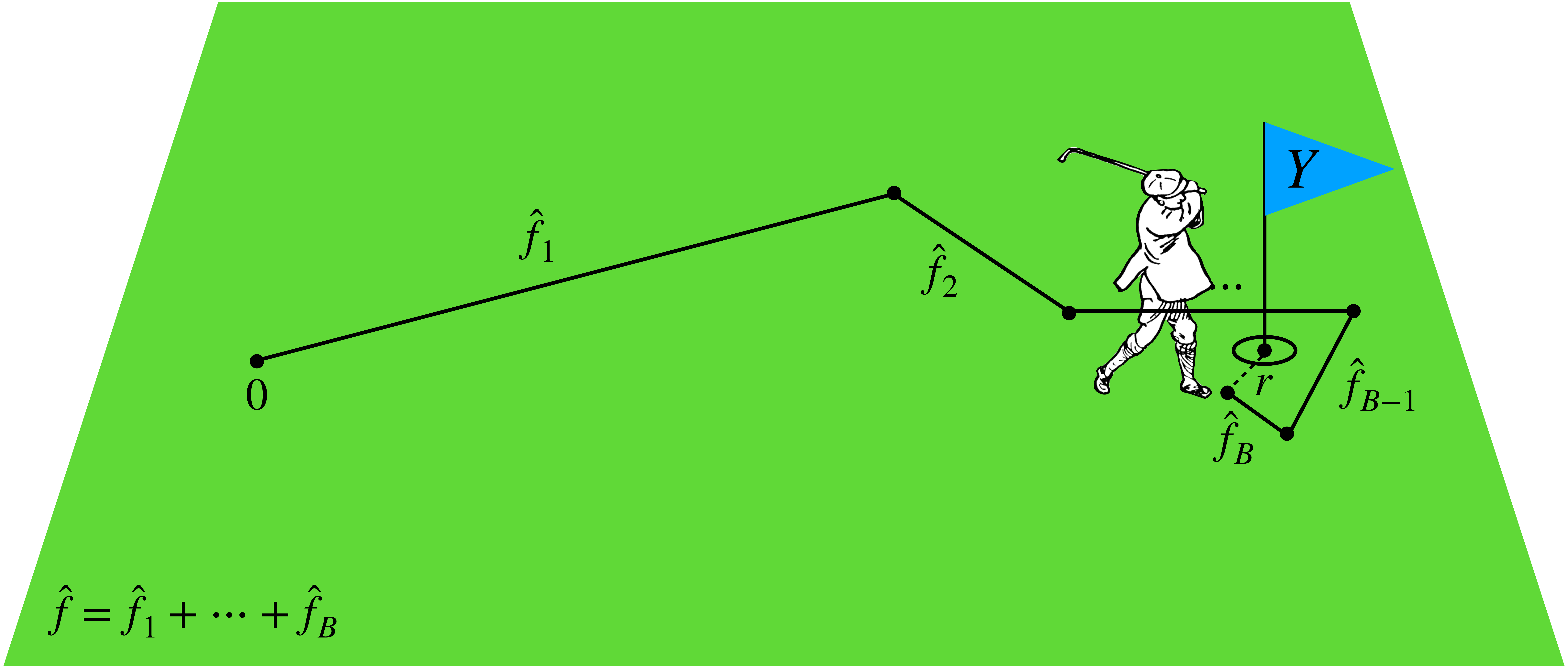
Boosting: An analogy with golf

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



Boosting: An analogy with golf

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

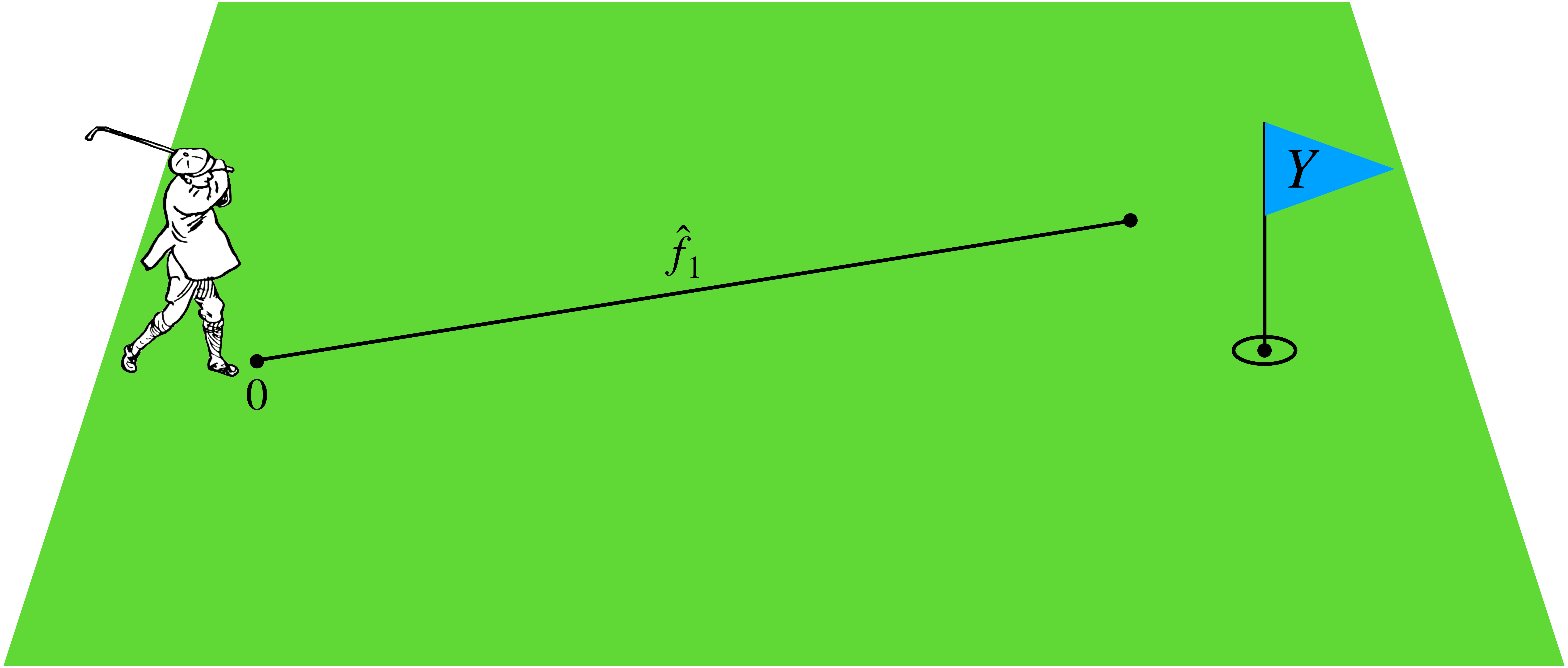


Comparison to random forests (driving range)

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

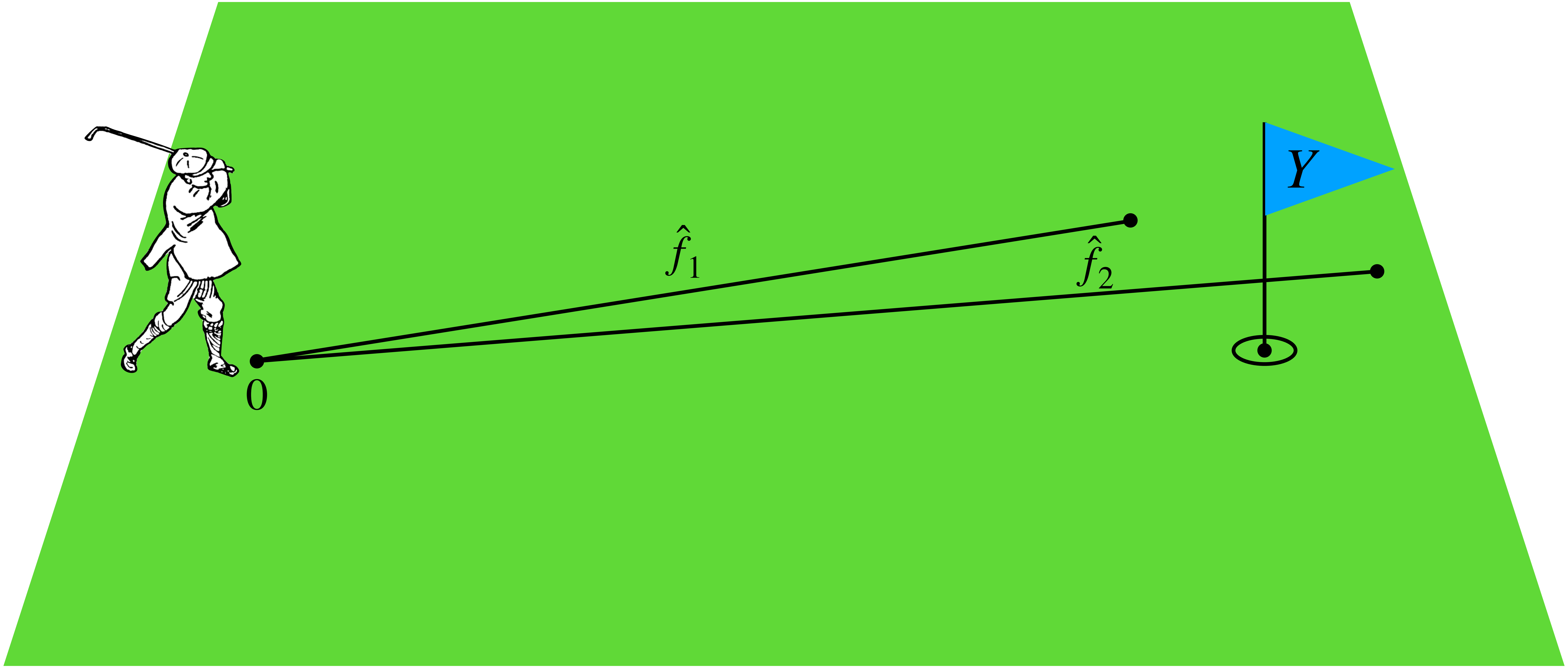
Comparison to random forests (driving range)

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



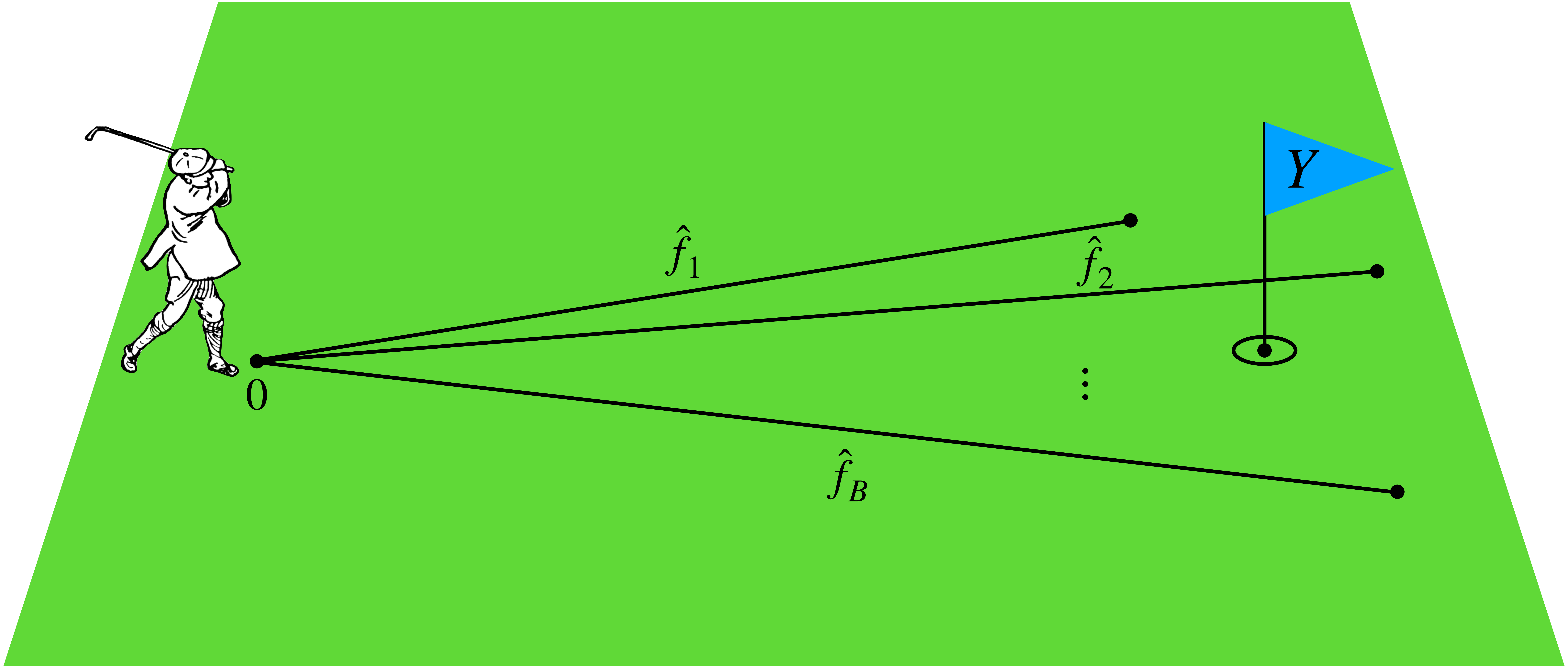
Comparison to random forests (driving range)

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



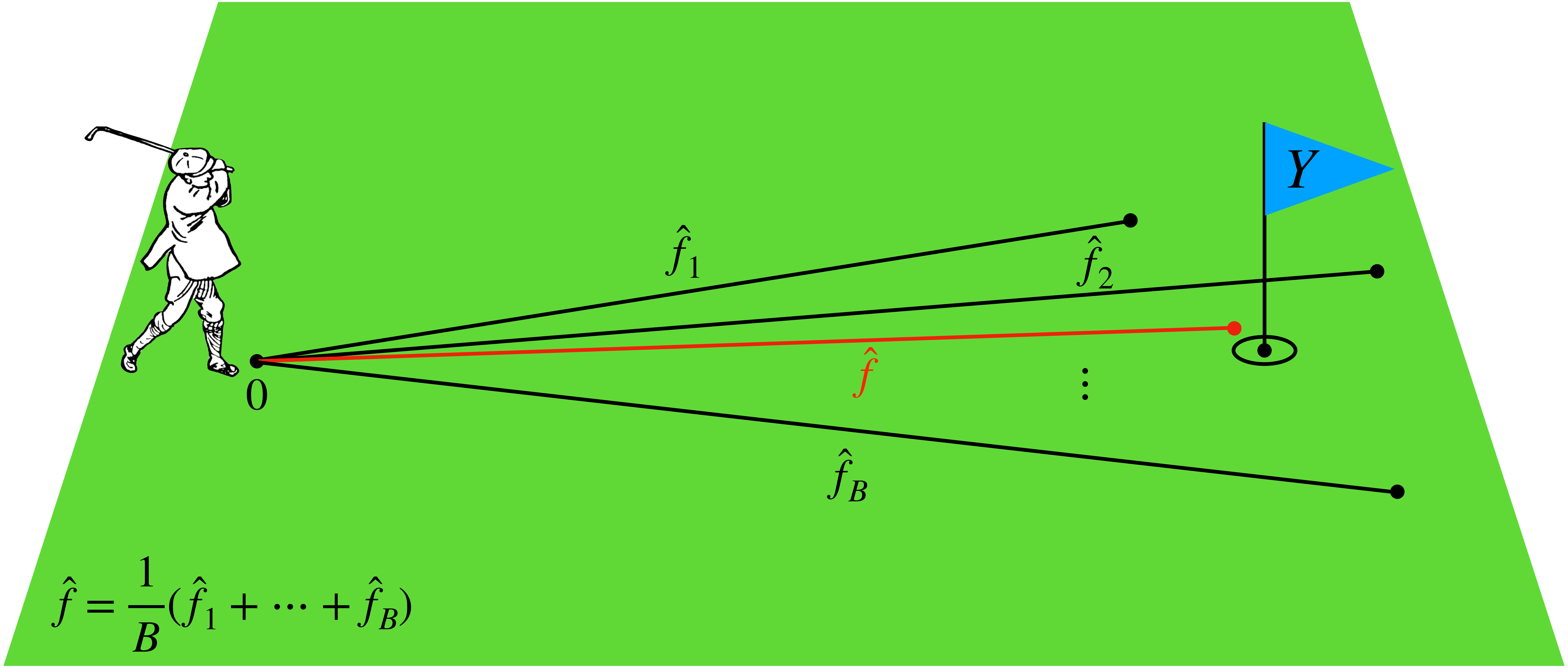
Comparison to random forests (driving range)

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



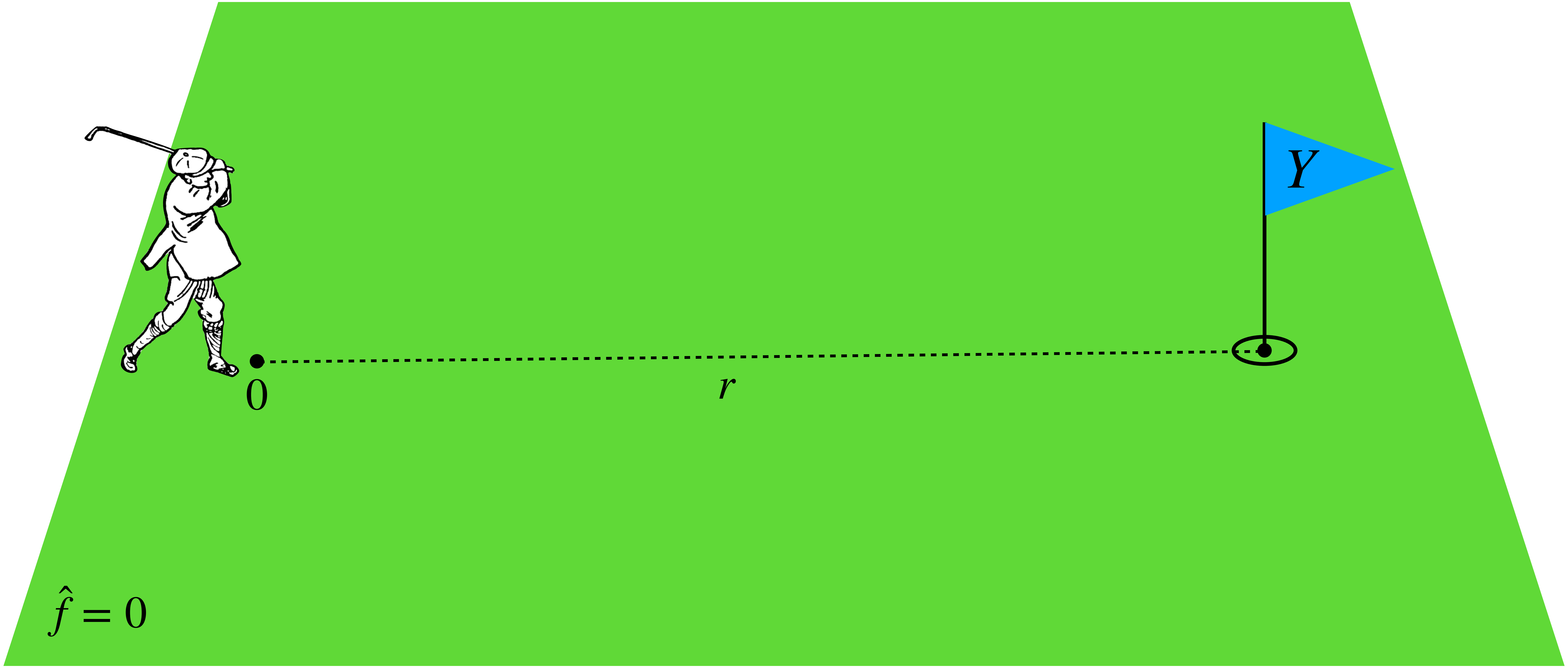
Comparison to random forests (driving range)

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



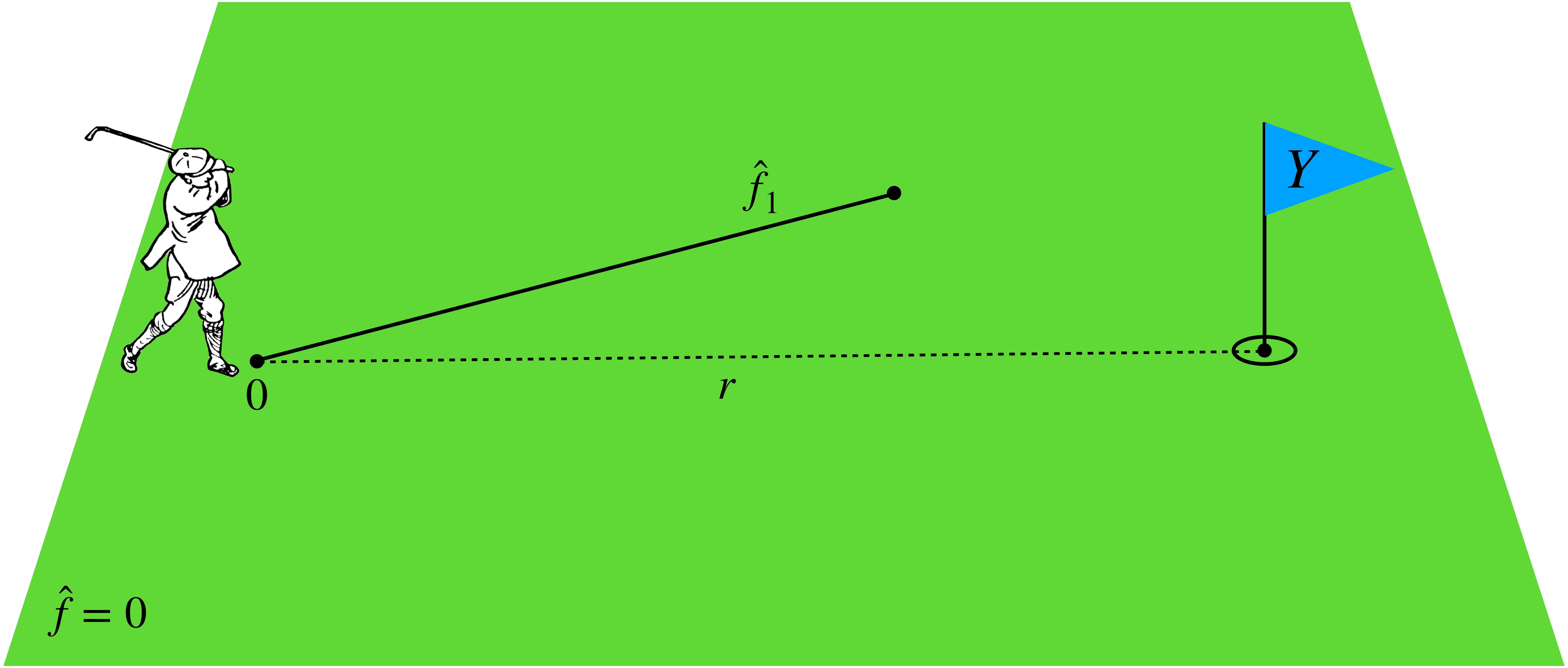
Boosting with shrinkage

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



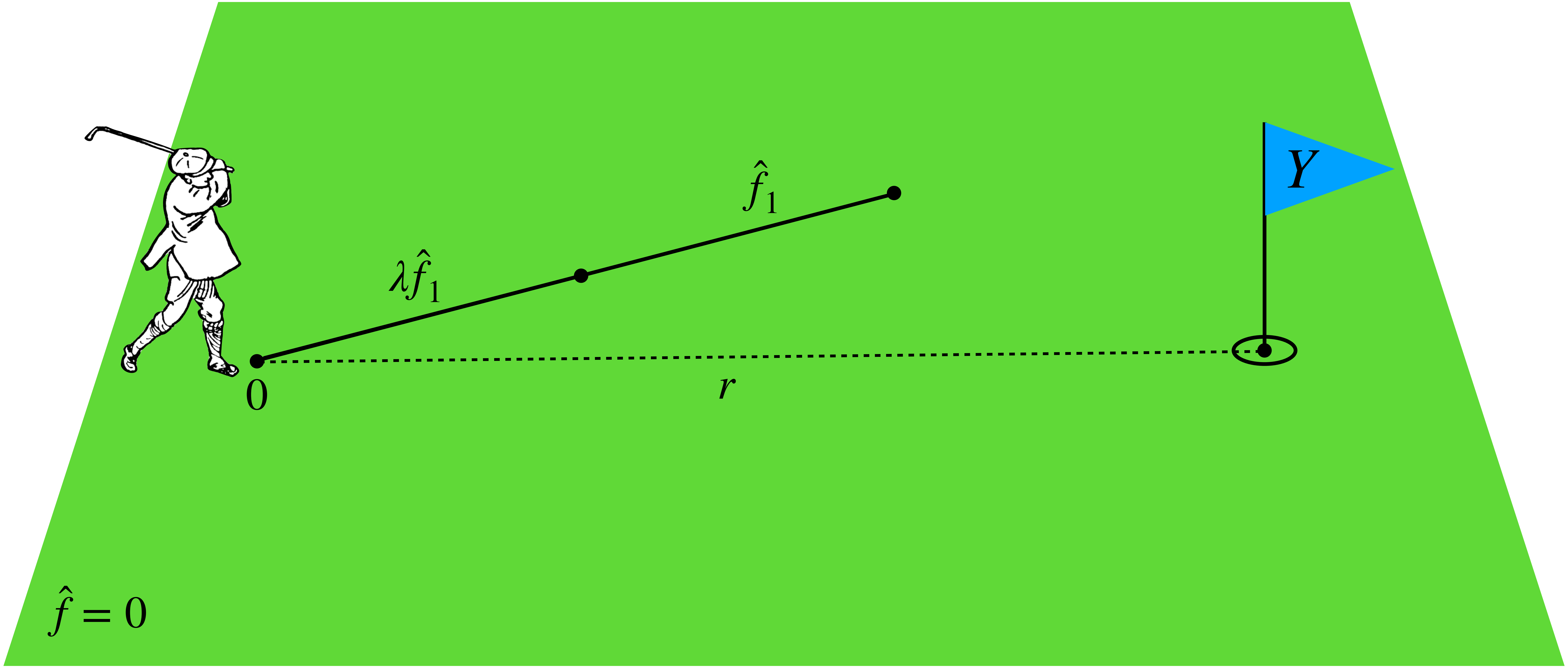
Boosting with shrinkage

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



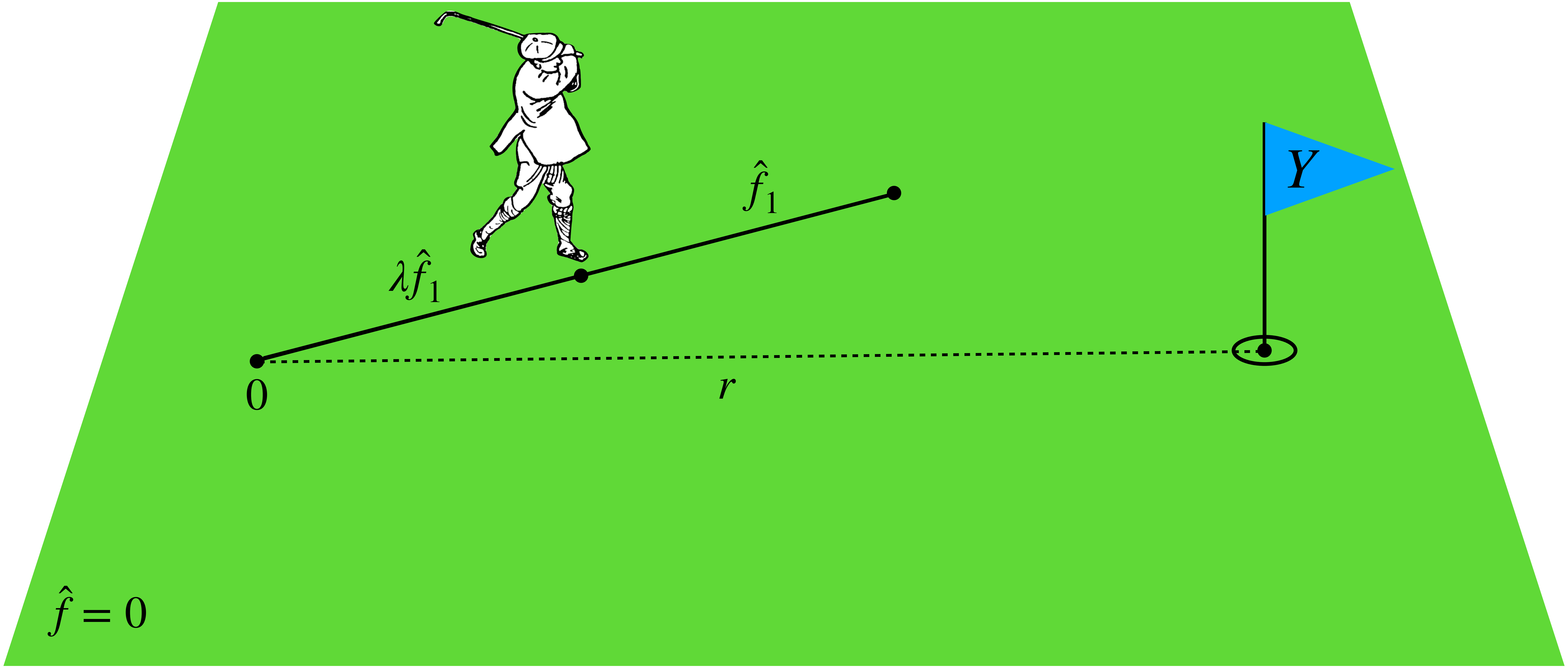
Boosting with shrinkage

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



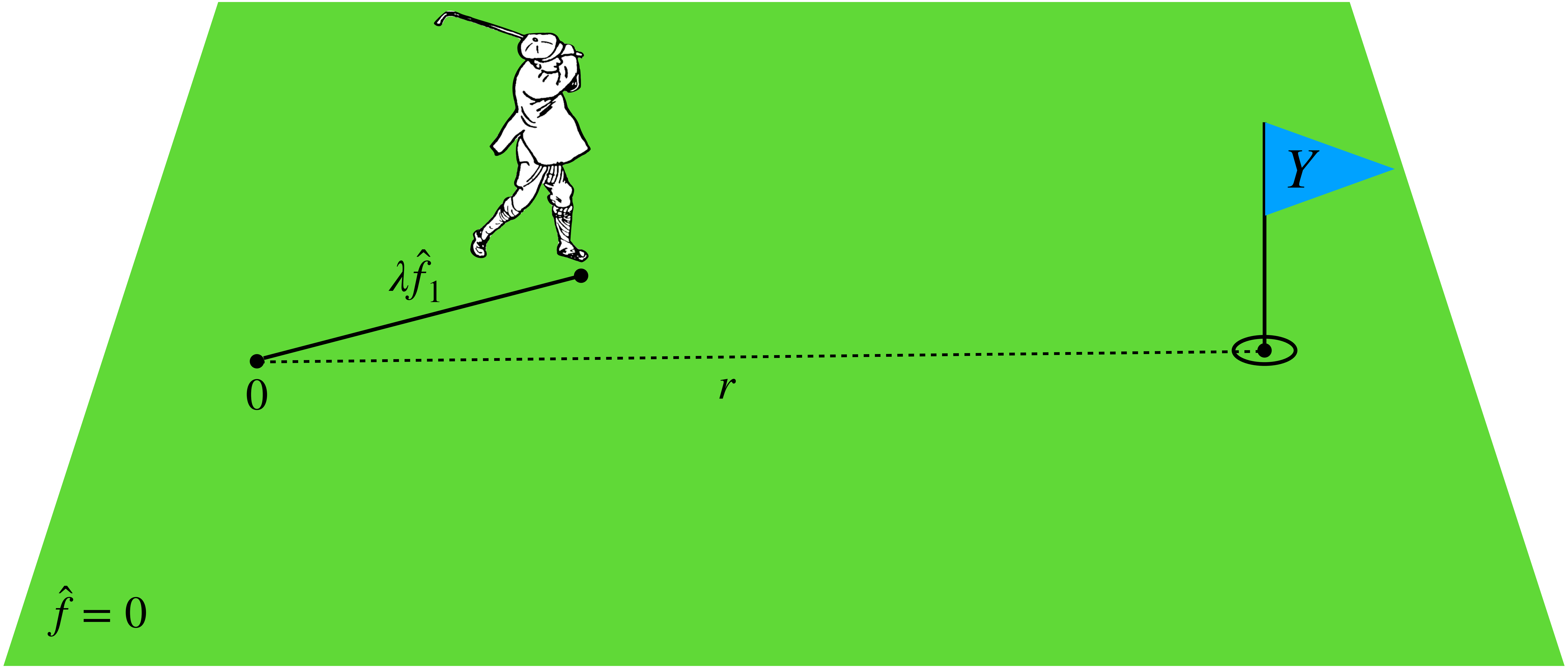
Boosting with shrinkage

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



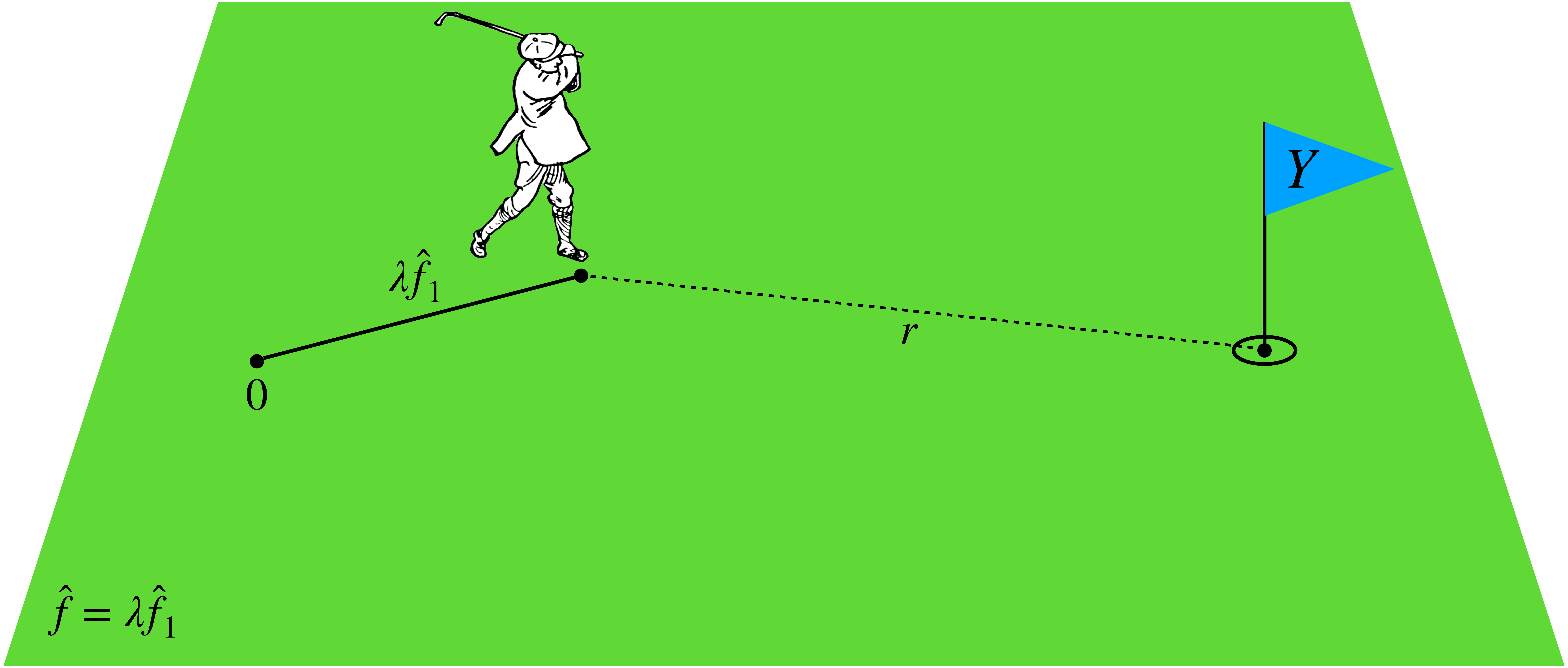
Boosting with shrinkage

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



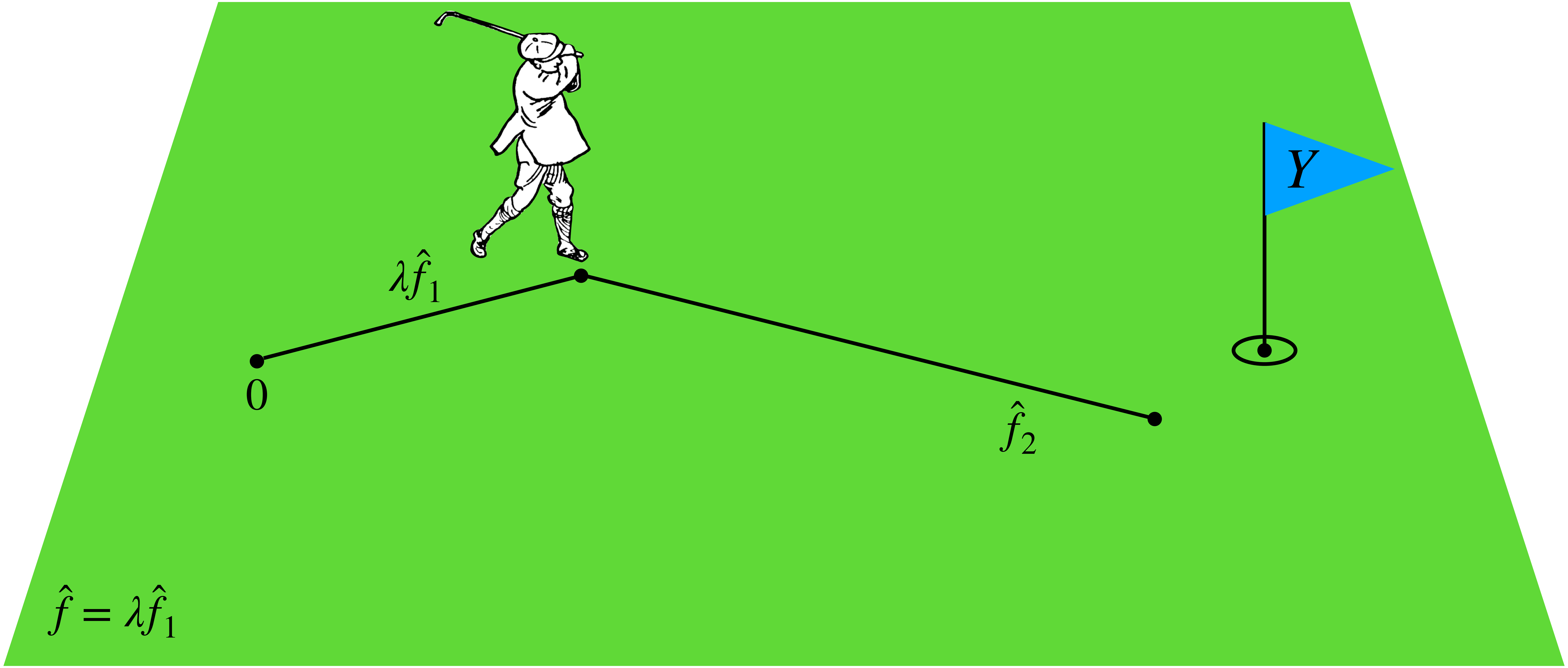
Boosting with shrinkage

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



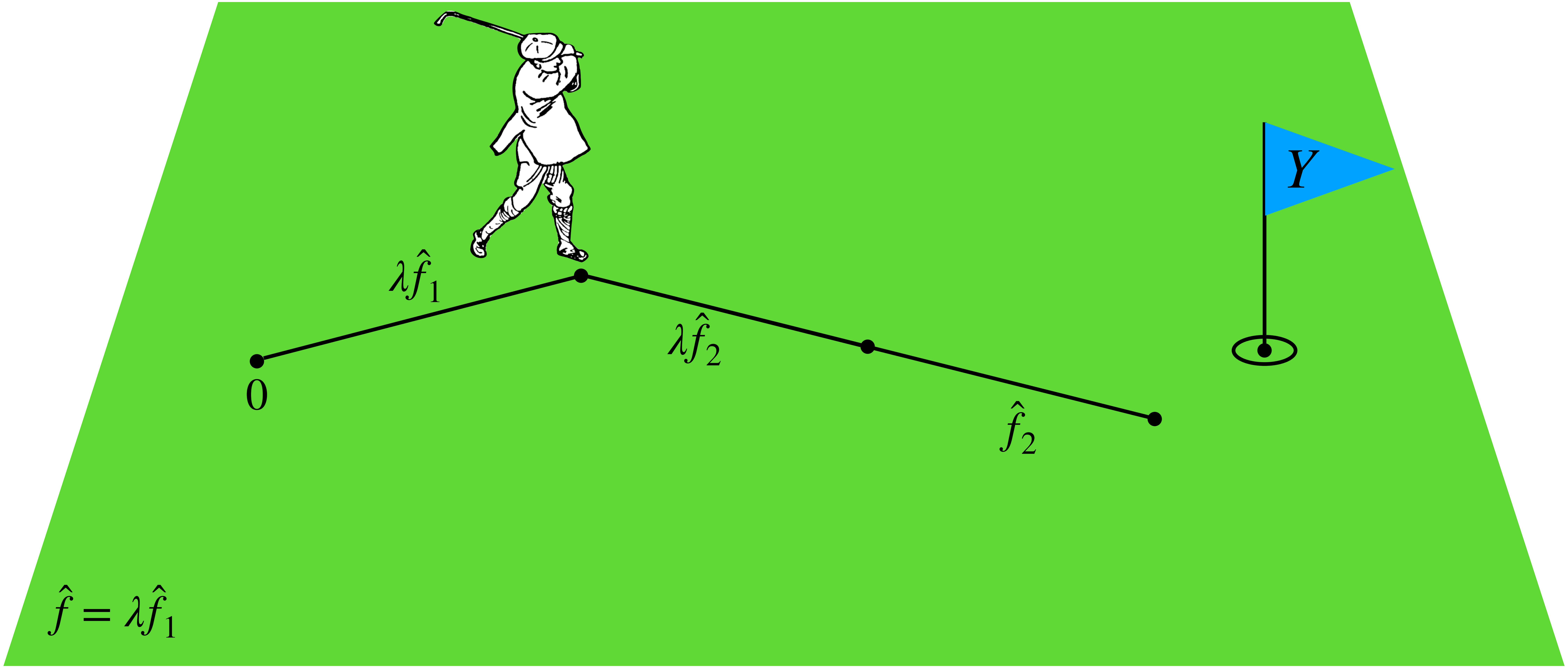
Boosting with shrinkage

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



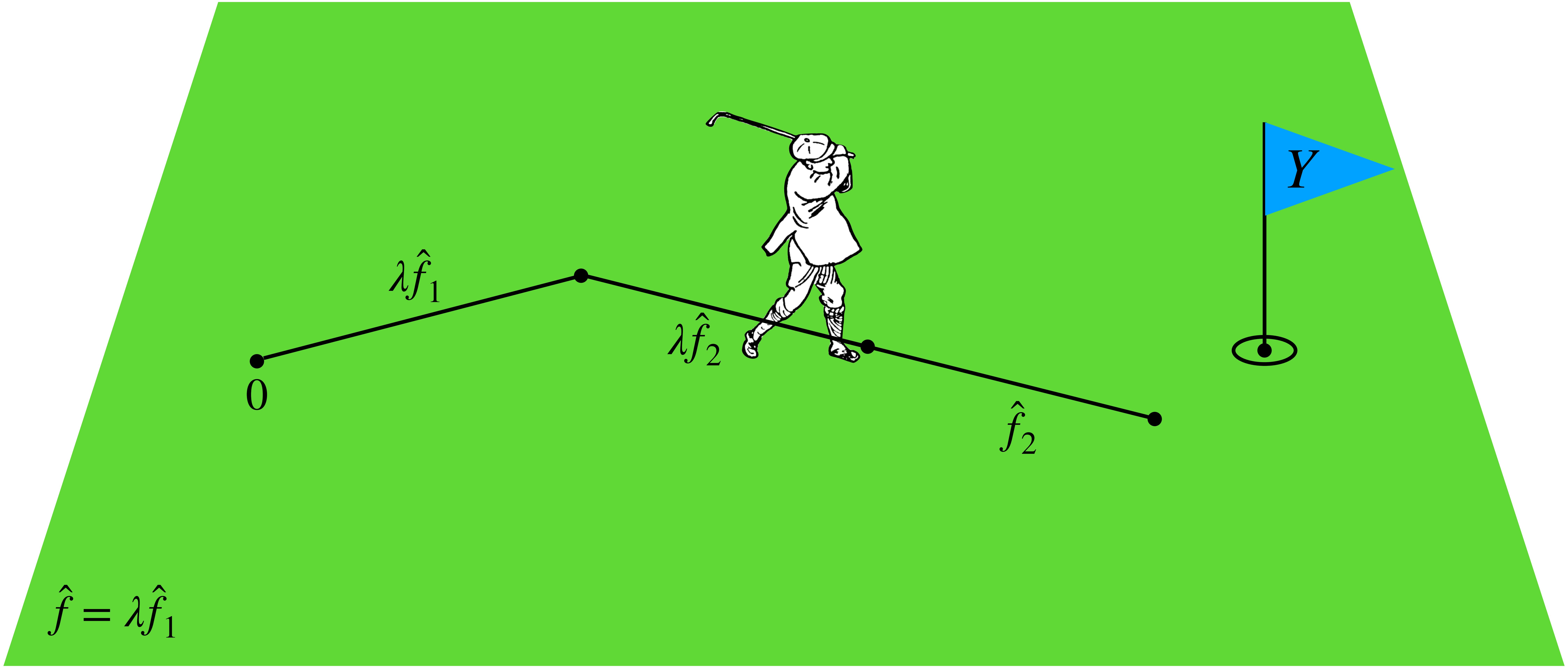
Boosting with shrinkage

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



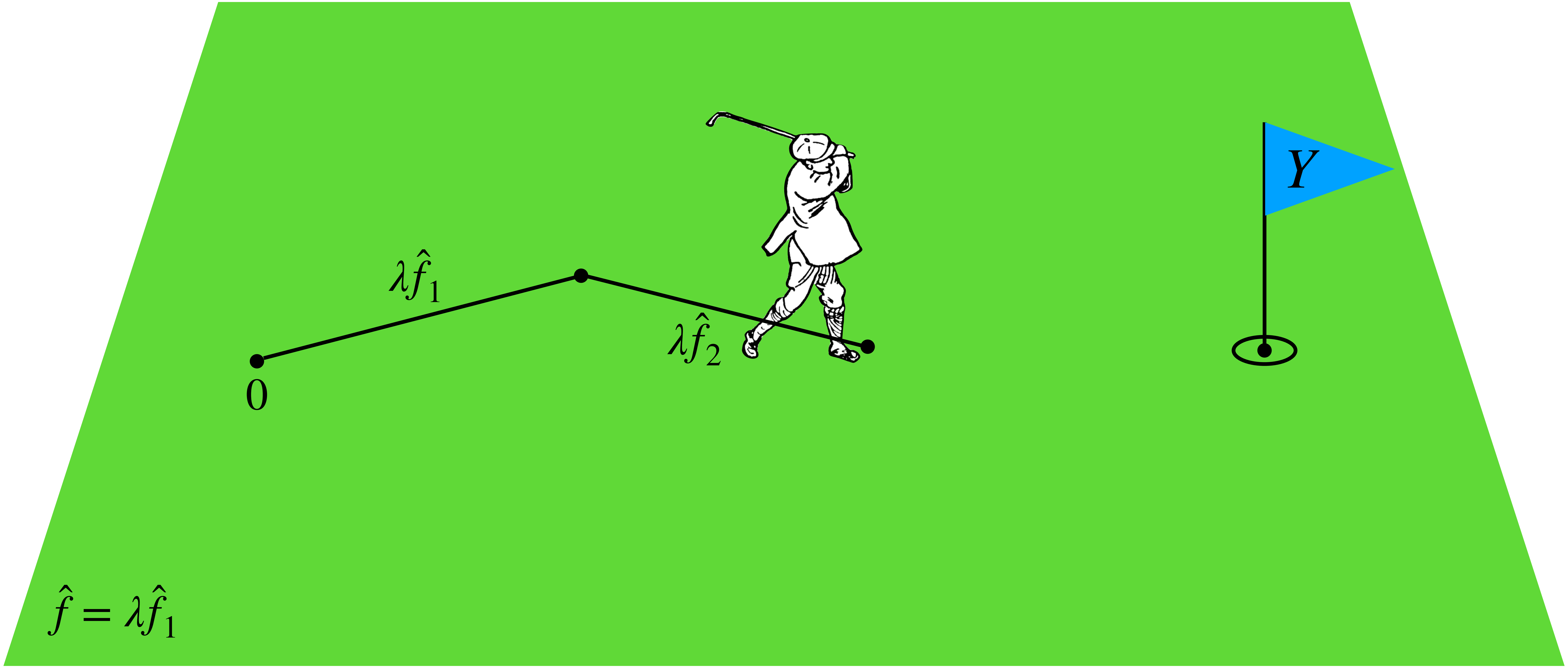
Boosting with shrinkage

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



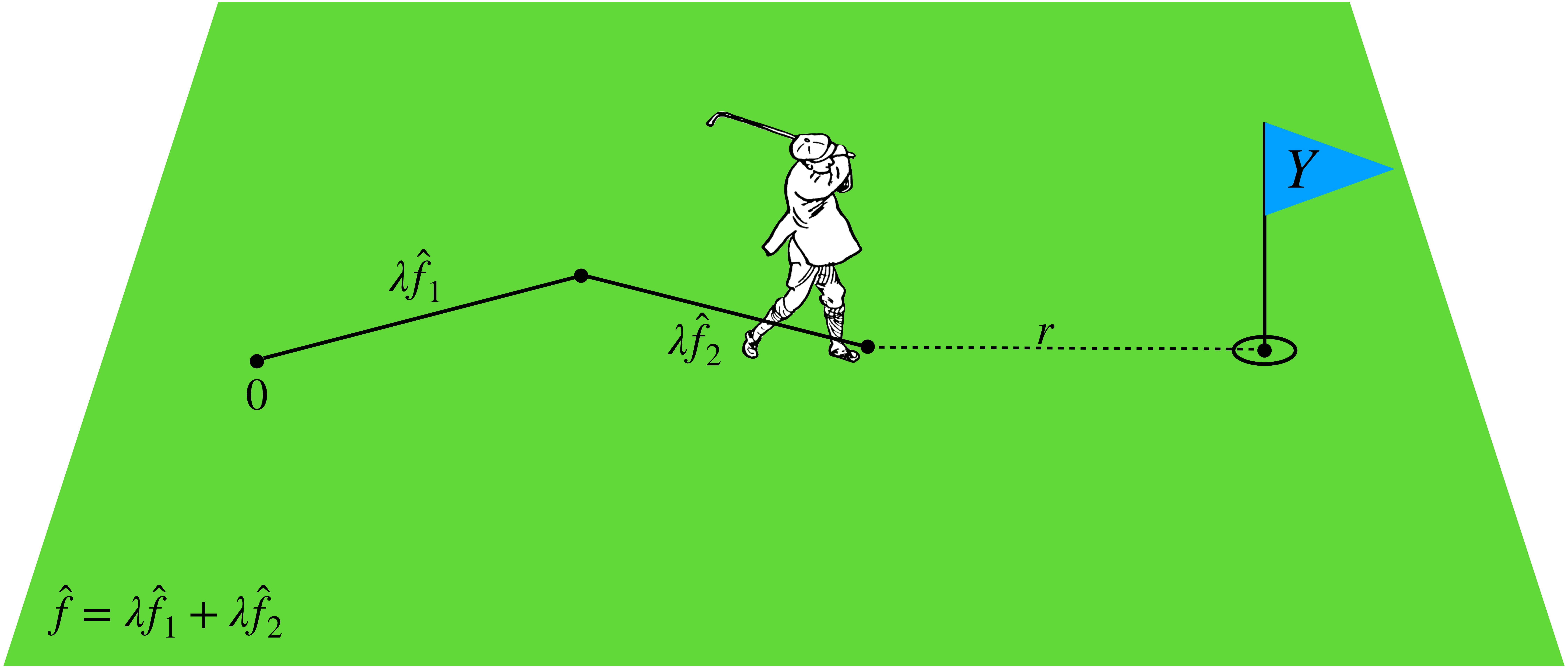
Boosting with shrinkage

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



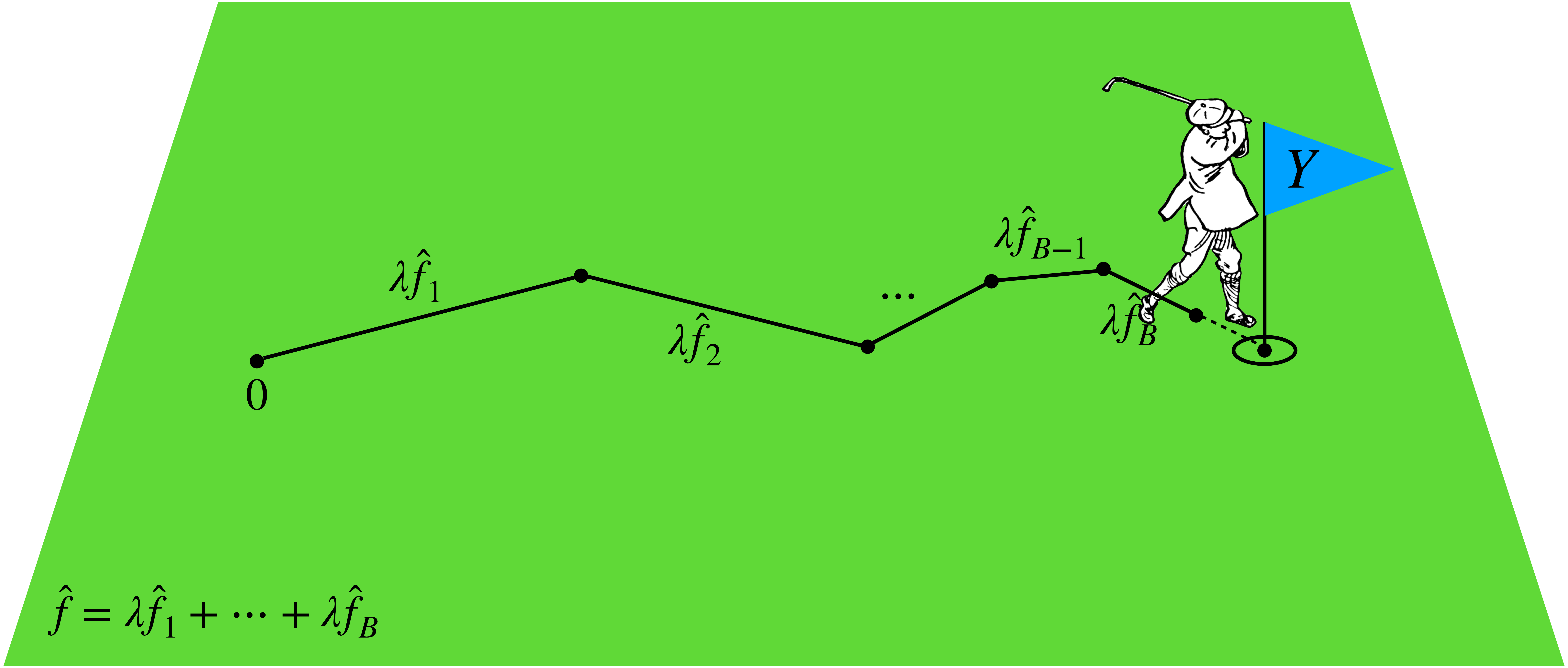
Boosting with shrinkage

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



Boosting with shrinkage

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



Boosting with shrinkage

Boosting with shrinkage

Inputs: weak learner $\hat{f}$, shrinkage parameter $\lambda \in (0,1]$, number of trees 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 $\hat{f}_1$

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 $r_i \leftarrow Y_i - \lambda \hat{f}_1(X_i)$

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 $\hat{f}_2$

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 $r_i \leftarrow r_i - \lambda \hat{f}_2(X_i)$

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

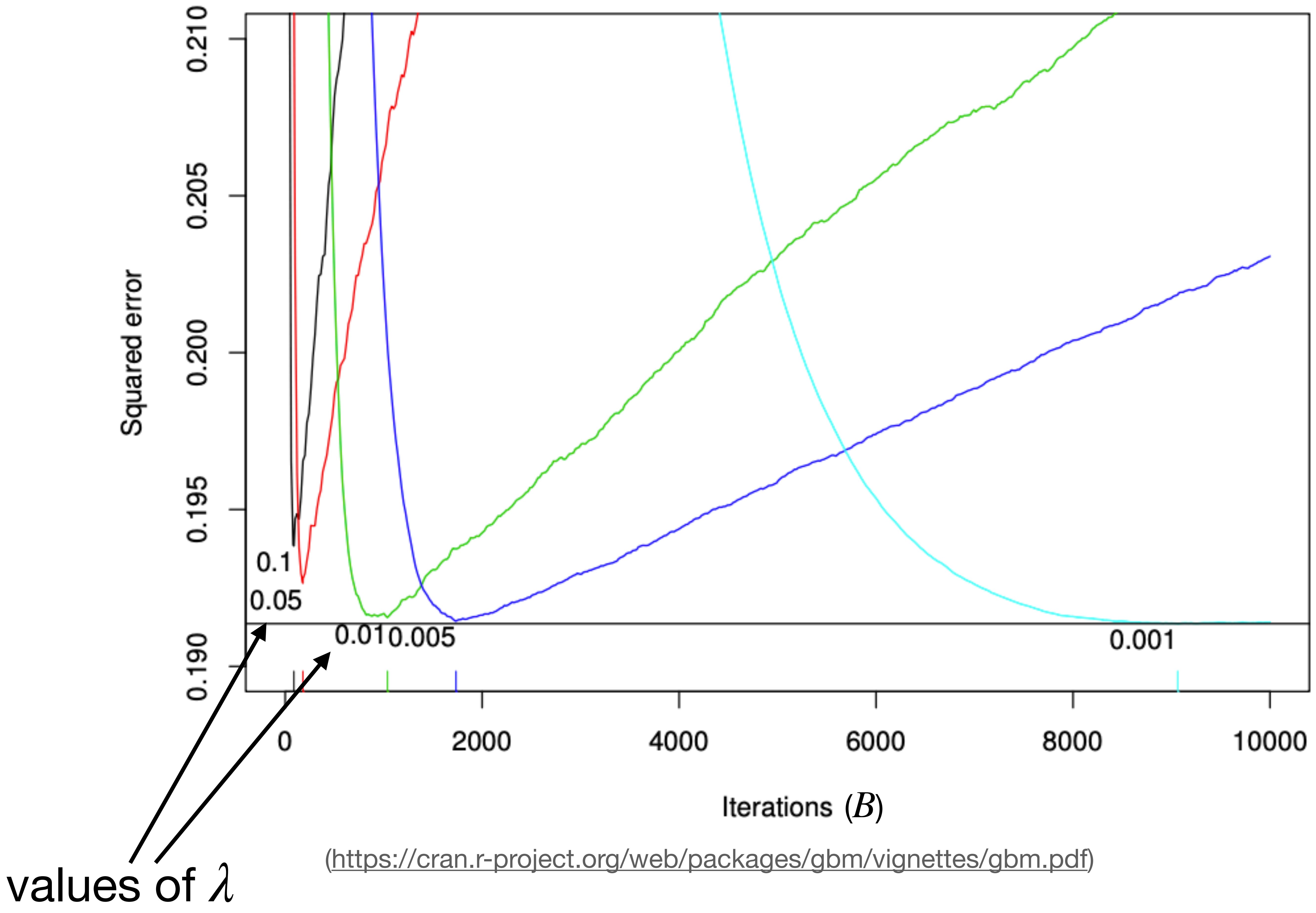
Boosting with shrinkage

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

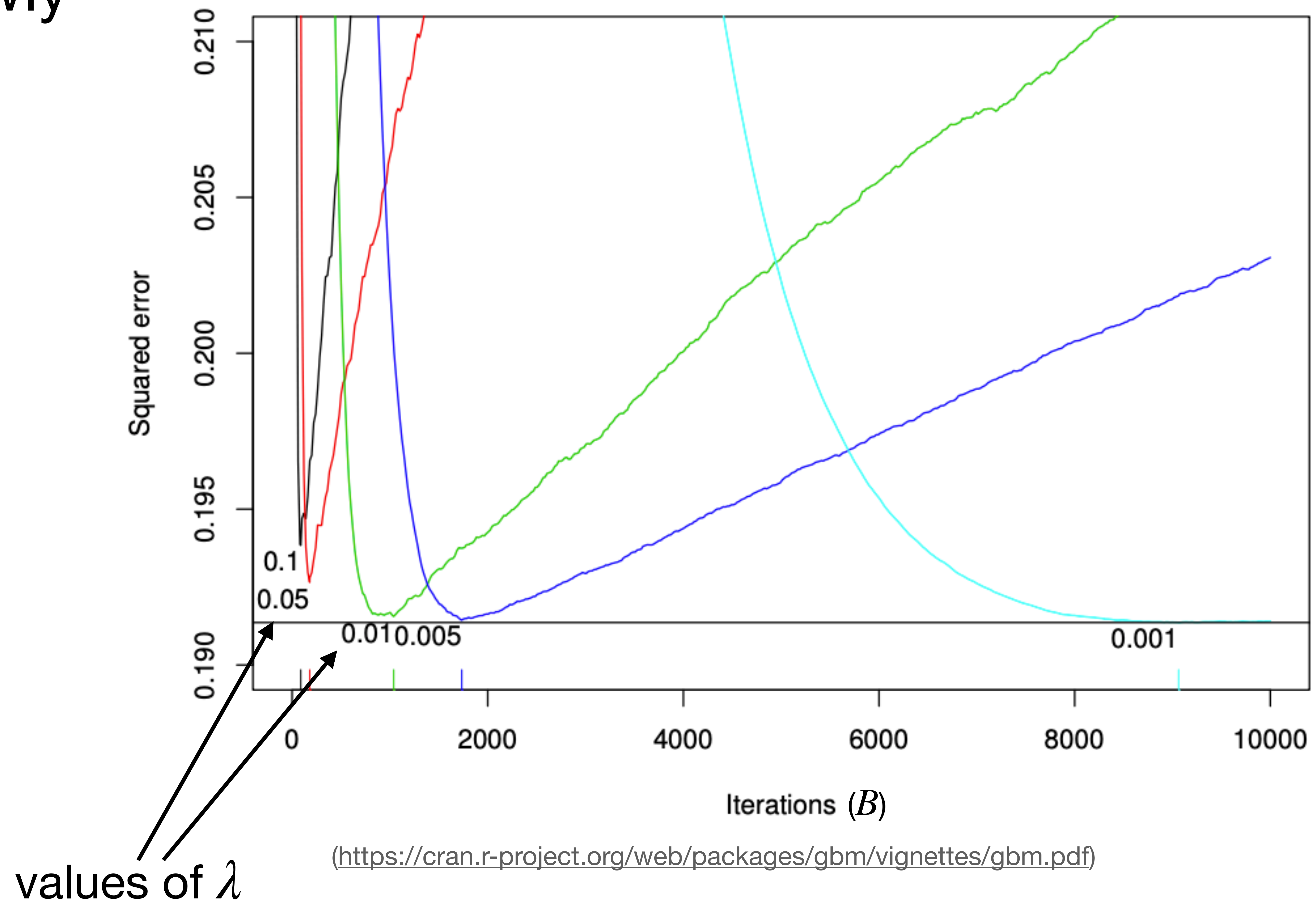
The parameters λ and B



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

The parameters λ and B

The parameter λ controls how slowly boosting learns.

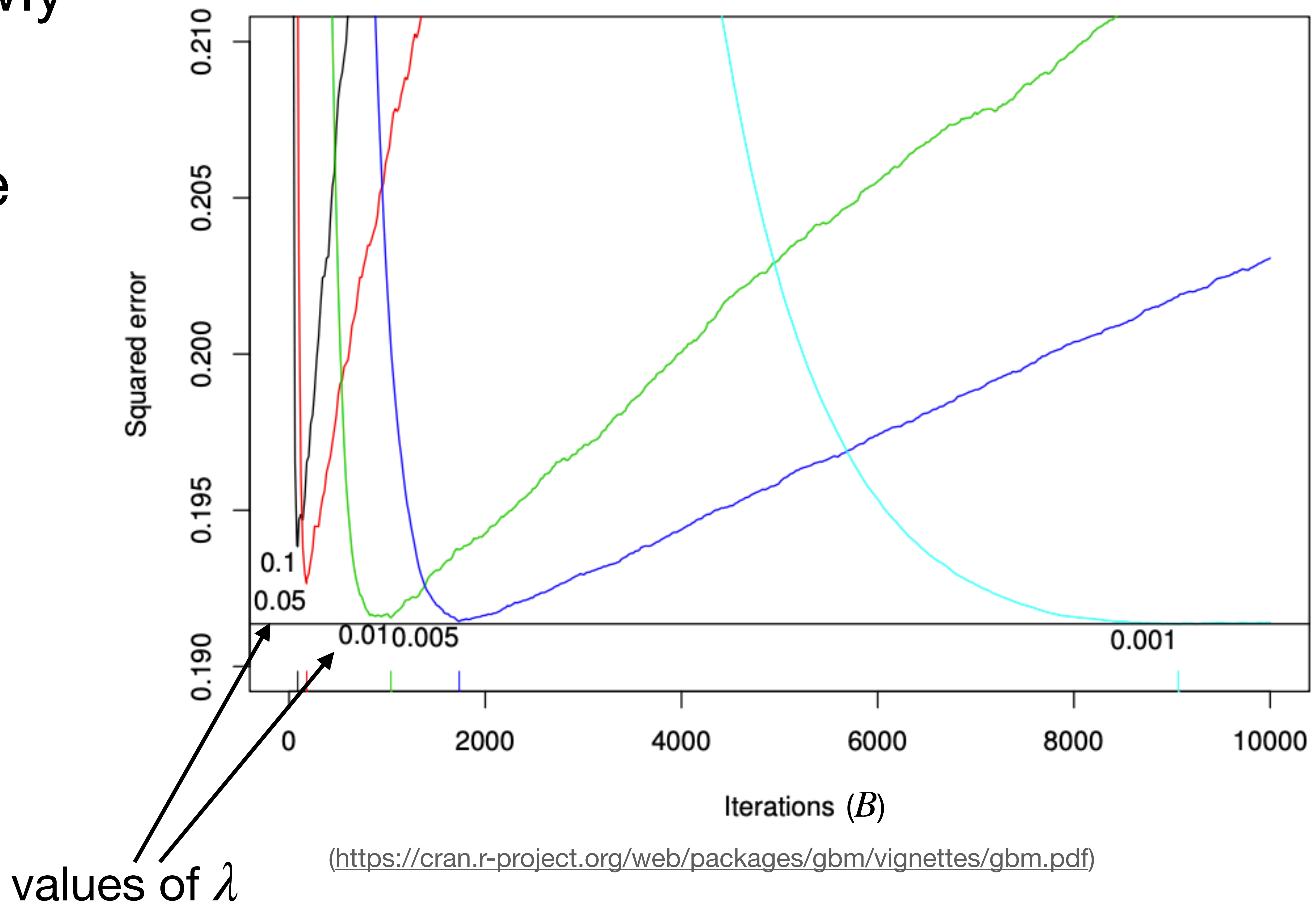


(https://cran.r-project.org/web/packages/gbm/vignettes/gbm.pdf)

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.



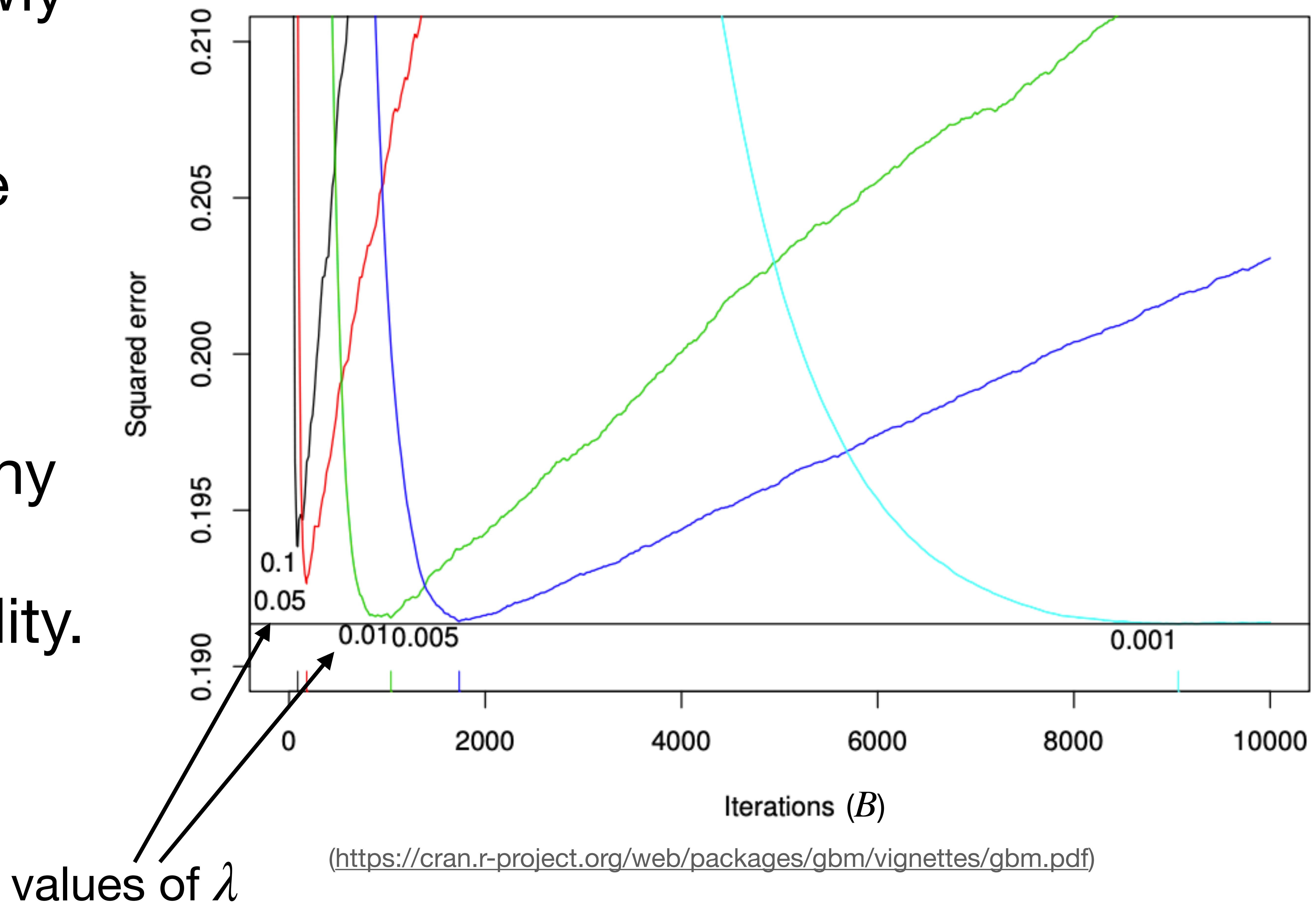
(https://cran.r-project.org/web/packages/gbm/vignettes/gbm.pdf)

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.



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

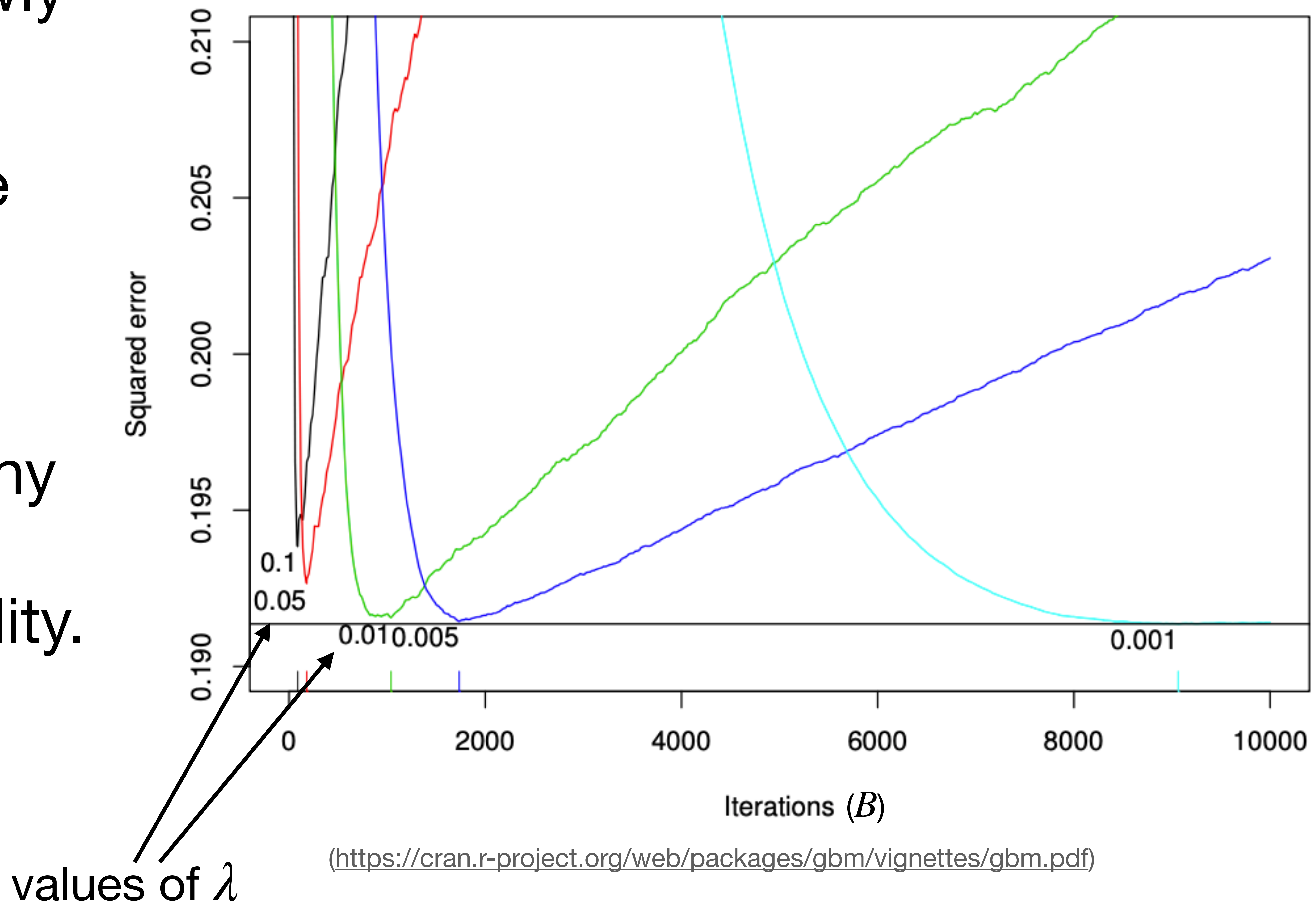
The parameters λ and B

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.



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

Choice of weak learner

Usually, shallow trees are used as the weak learners.

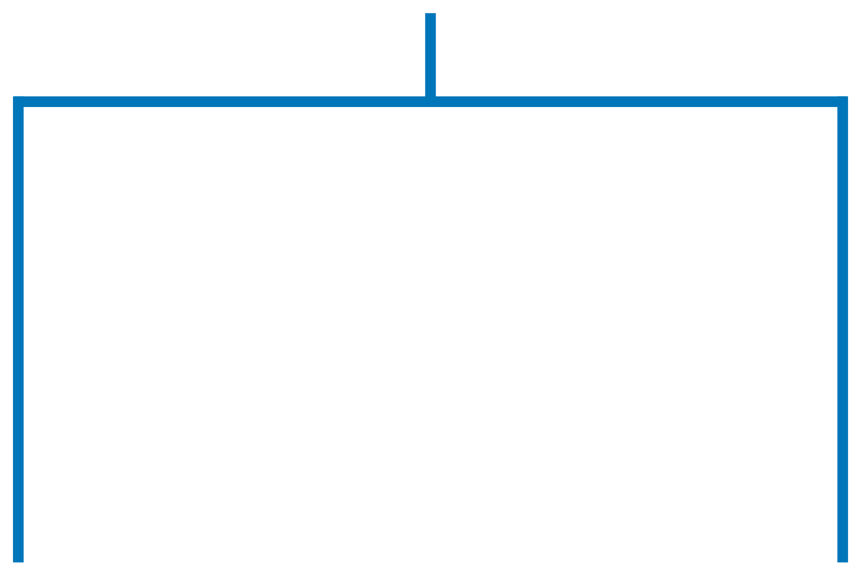
For boosting, tree size usually parameterized by interaction depth d, the maximum number of splits needed to get to a terminal node.

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.

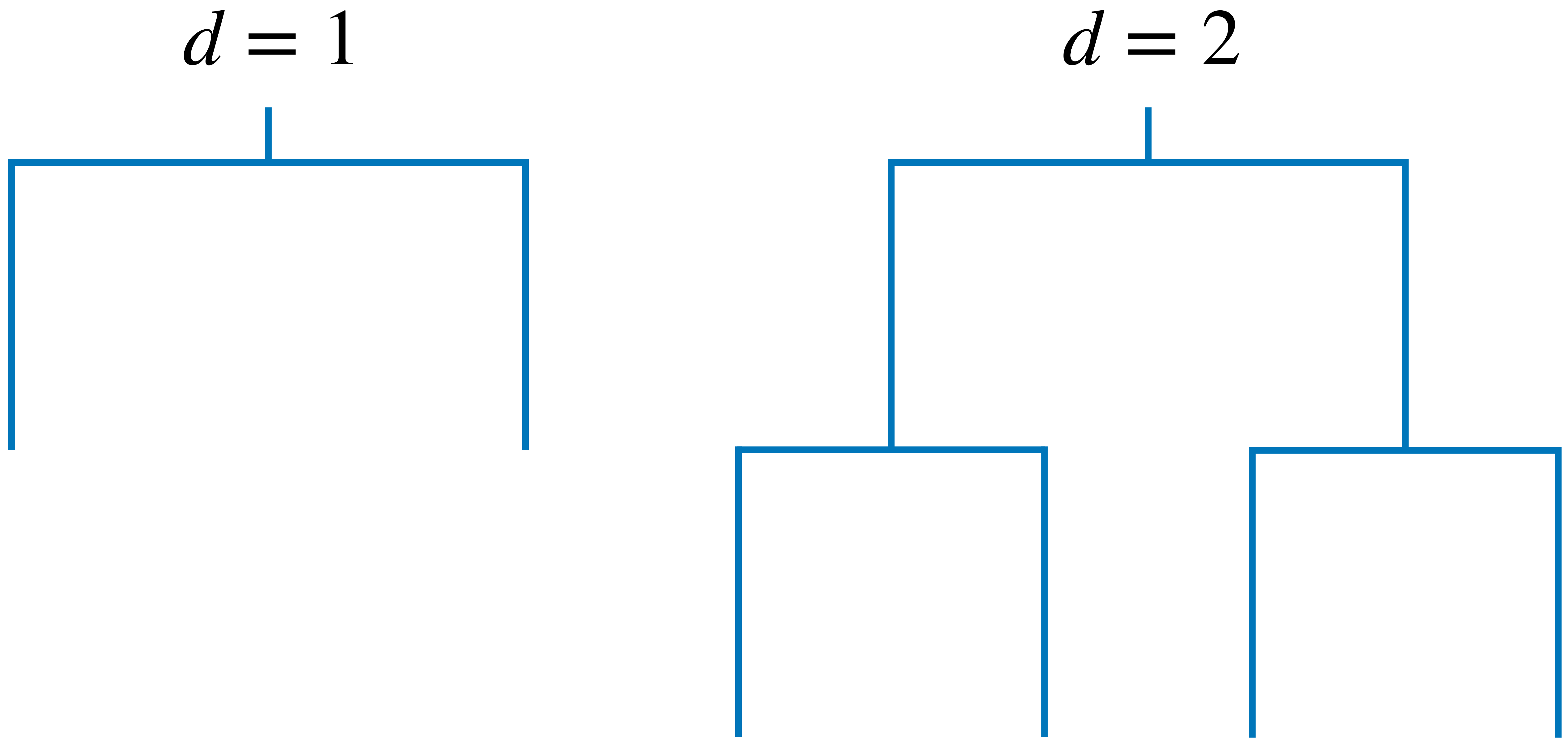
$d = 1$



Choice of weak learner

Usually, shallow trees are used as the weak learners.

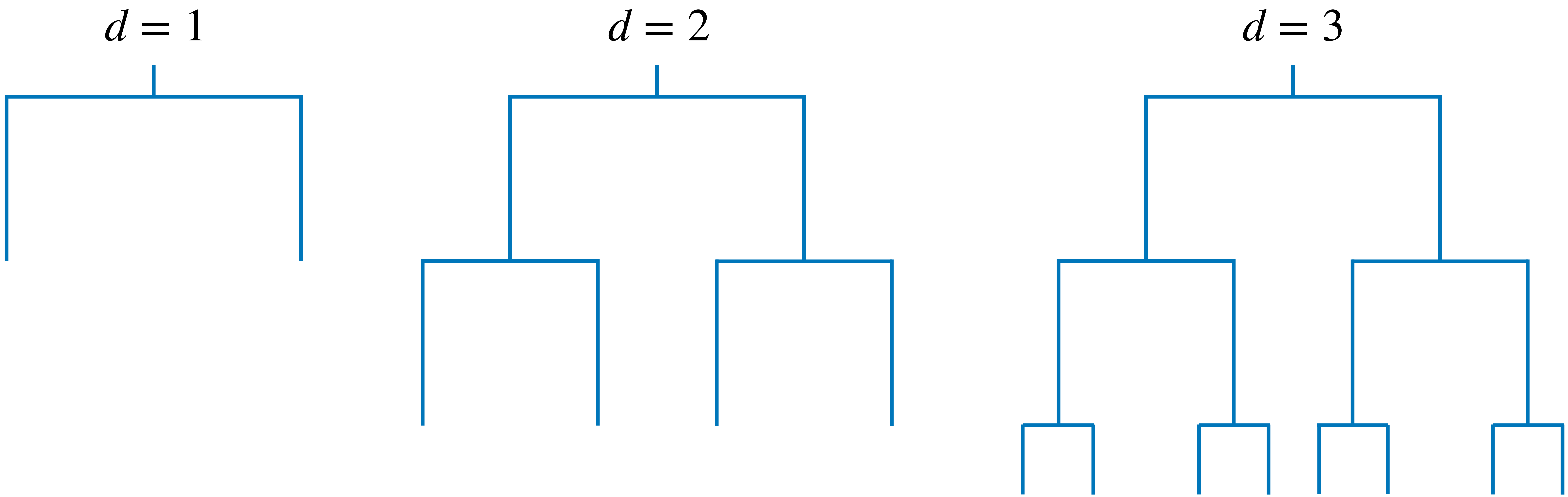
For boosting, tree size usually parameterized by interaction depth d, the maximum number of splits needed to get to a terminal node.



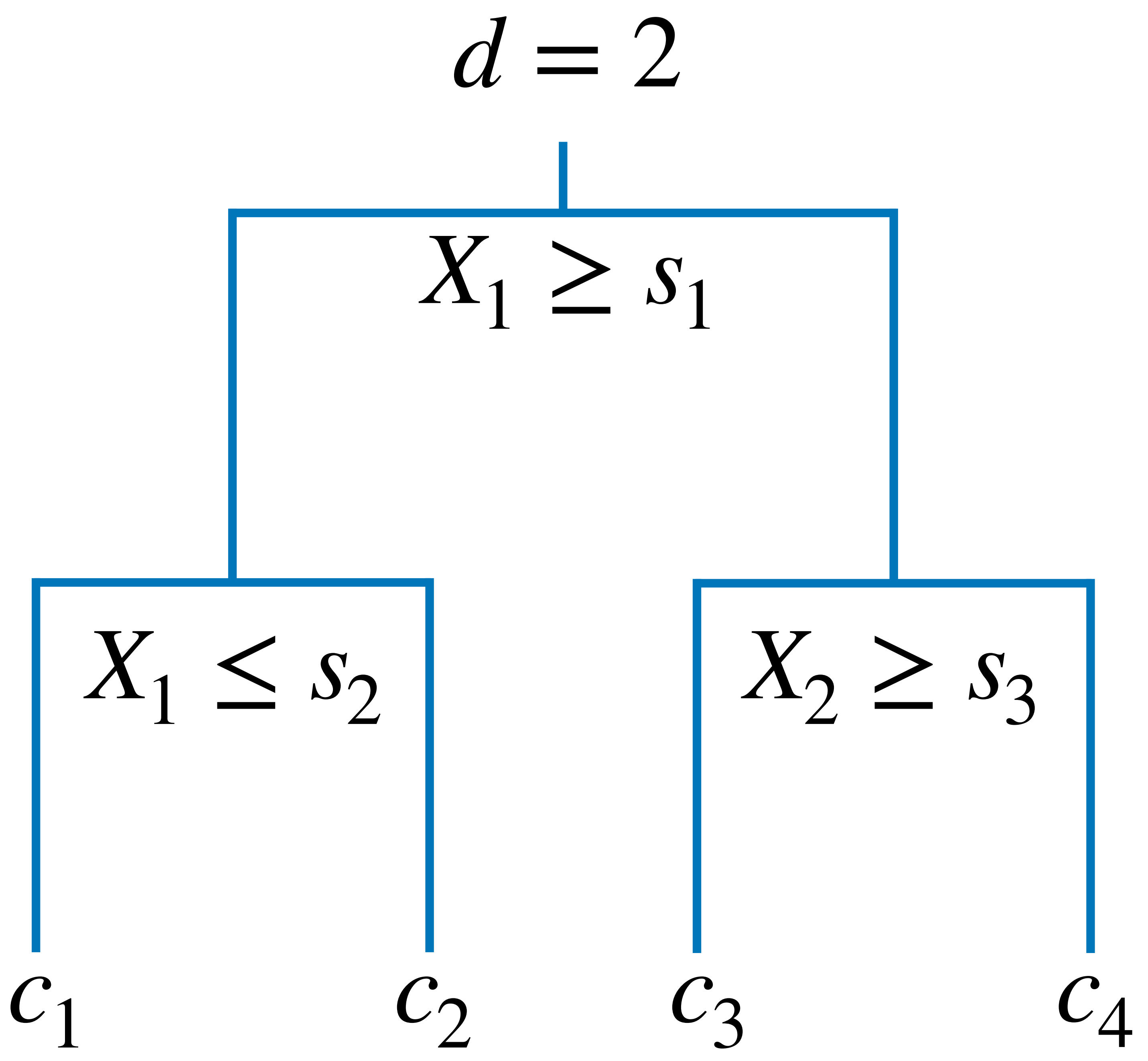
Choice of weak learner

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.

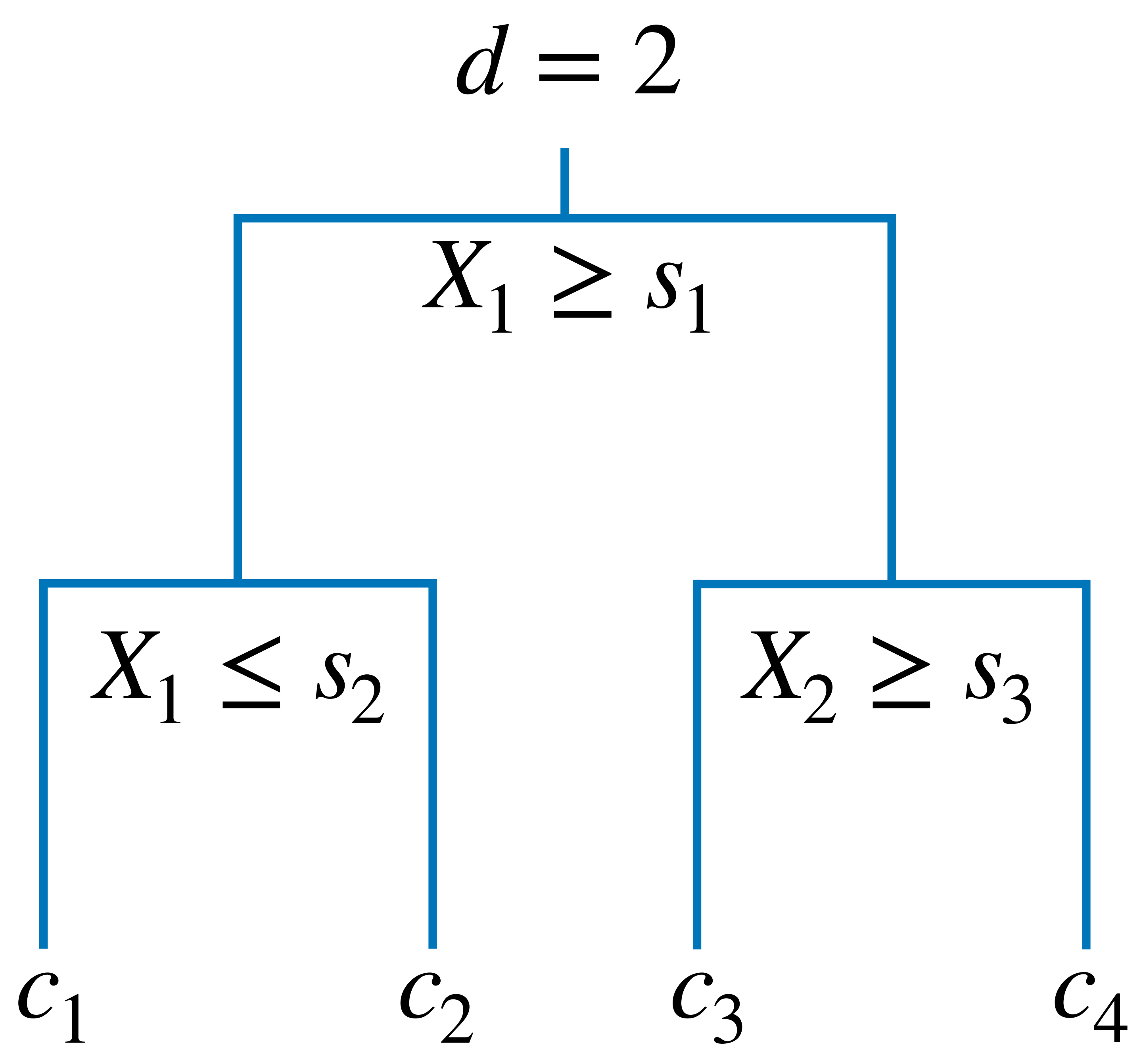


Interaction depth



Interaction depth

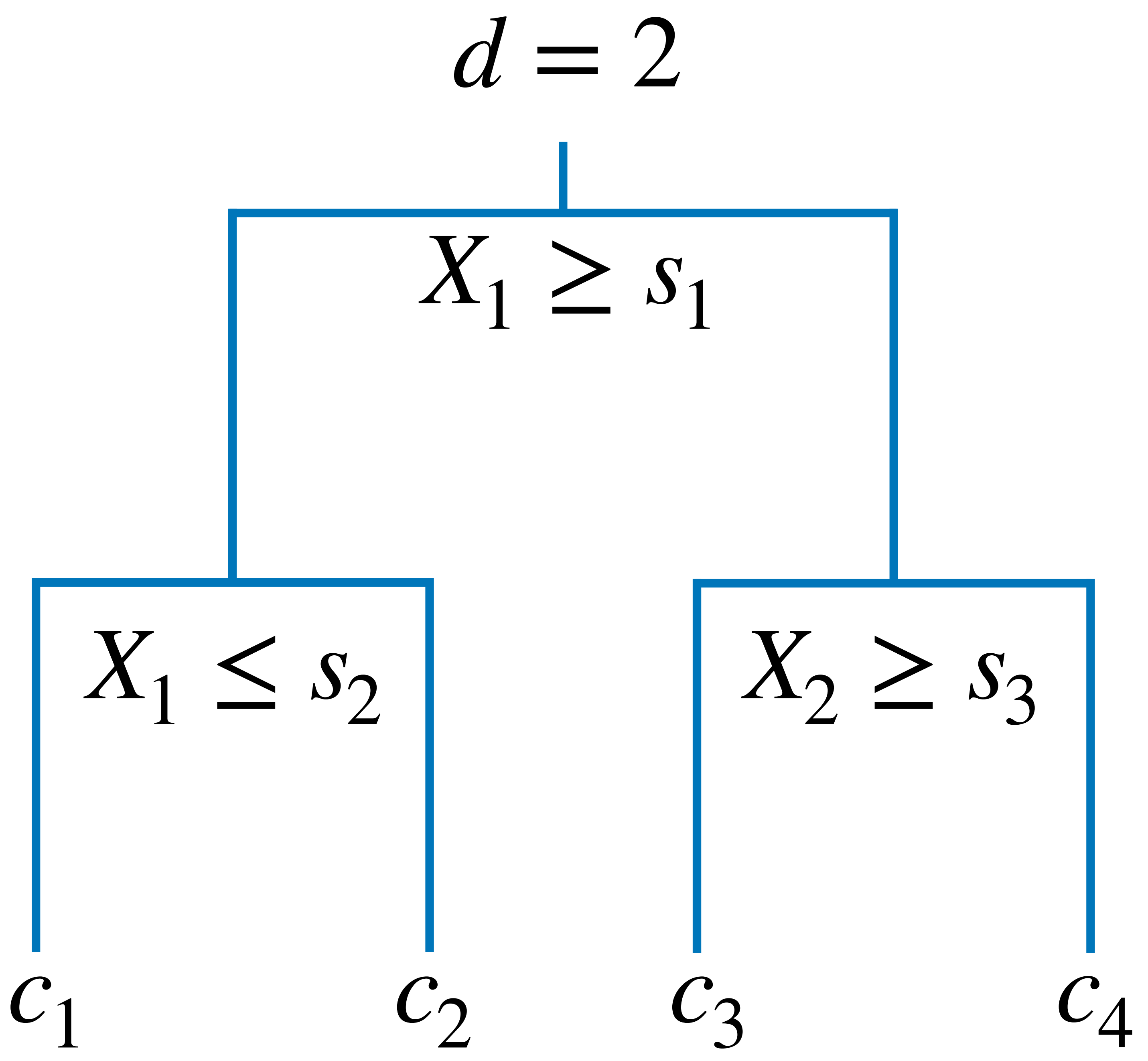
Consider a single tree, for example with $d = 2$.



Interaction depth

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

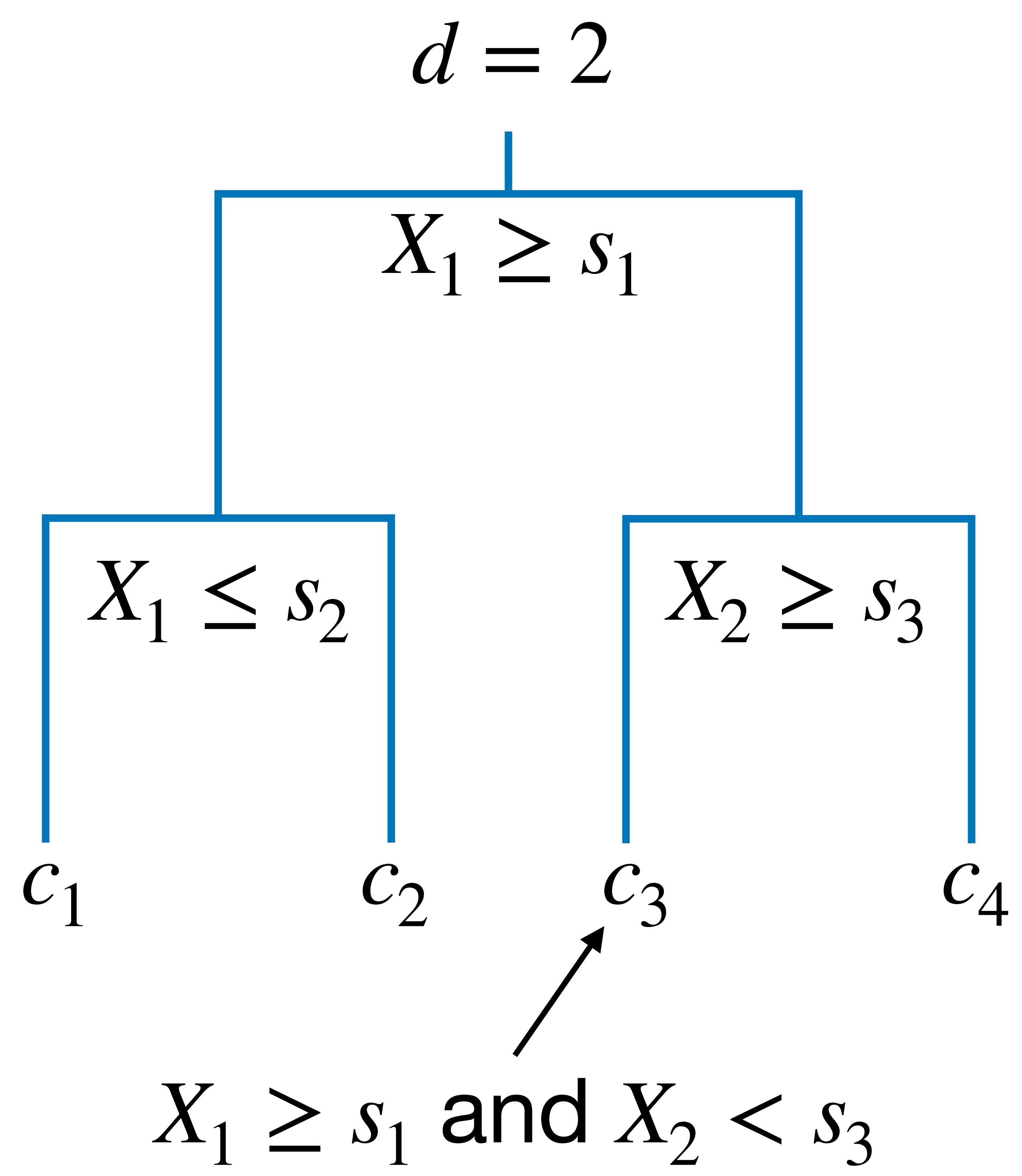
The prediction rule involves making decisions based on X_1 and X_2 together (two-way interactions).



Interaction depth

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

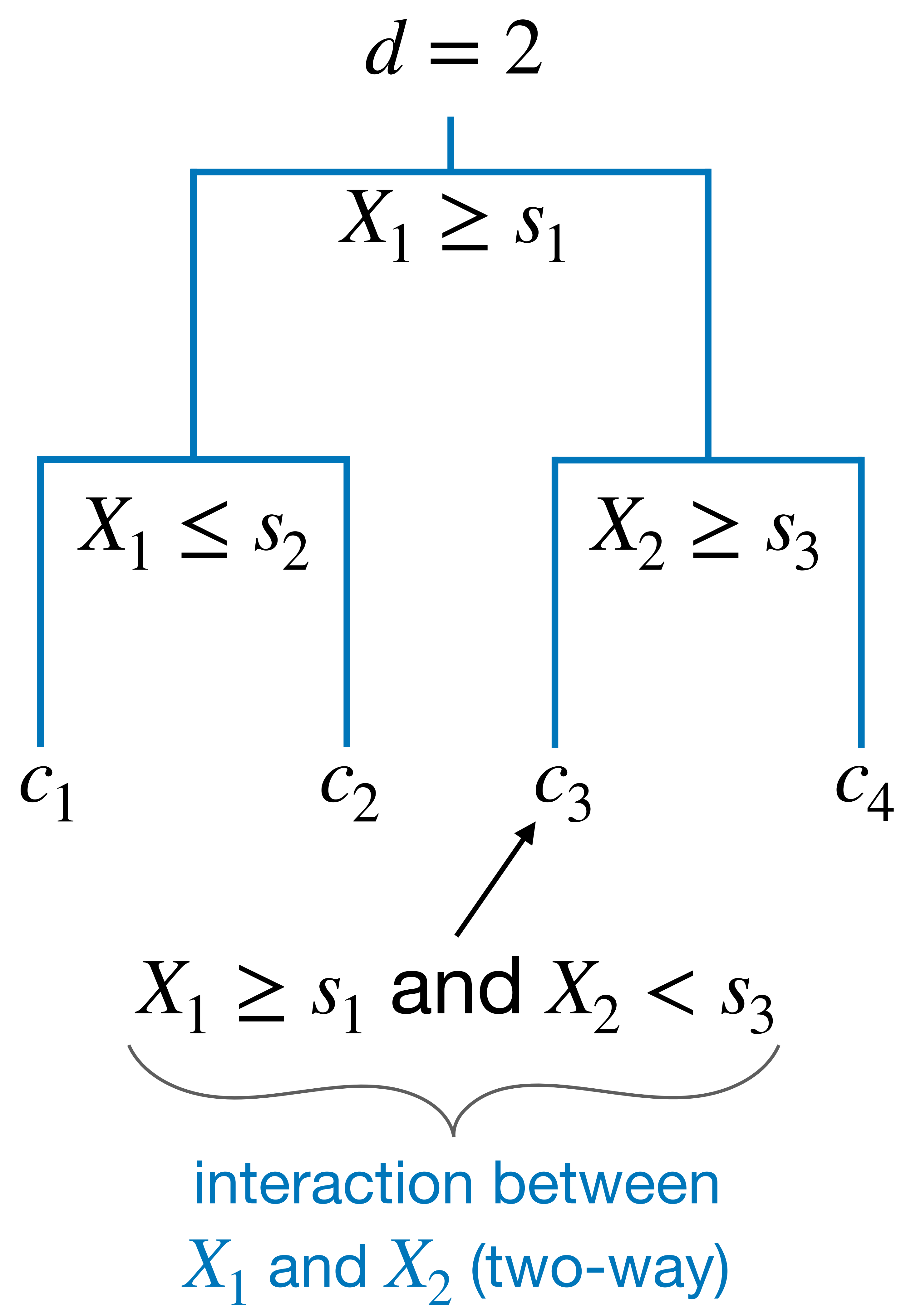
The prediction rule involves making decisions based on X_1 and X_2 together (two-way interactions).



Interaction depth

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

The prediction rule involves making decisions based on X_1 and X_2 together (two-way interactions).

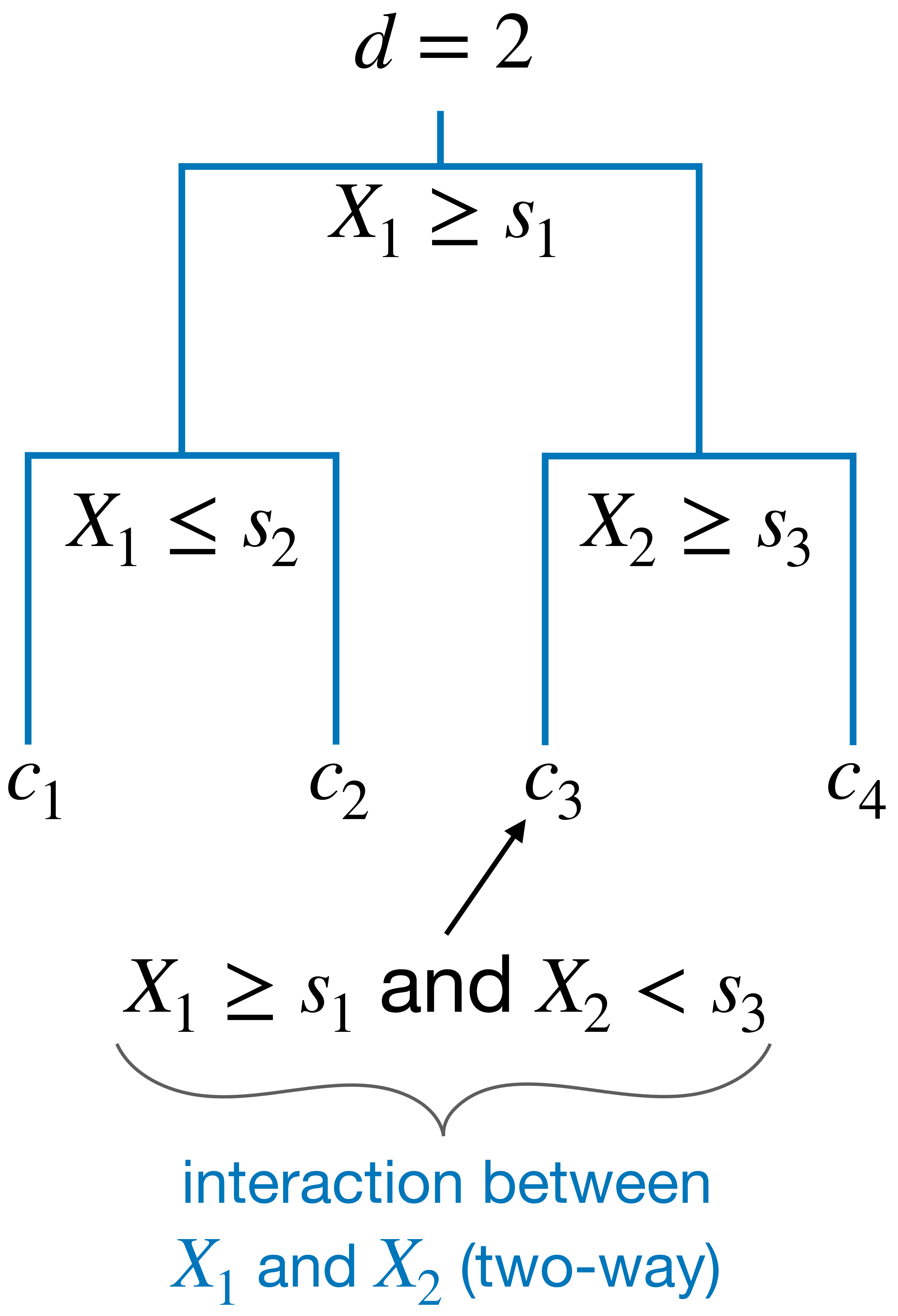


Interaction depth

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

The prediction rule involves making decisions based on X_1 and X_2 together (two-way interactions).

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



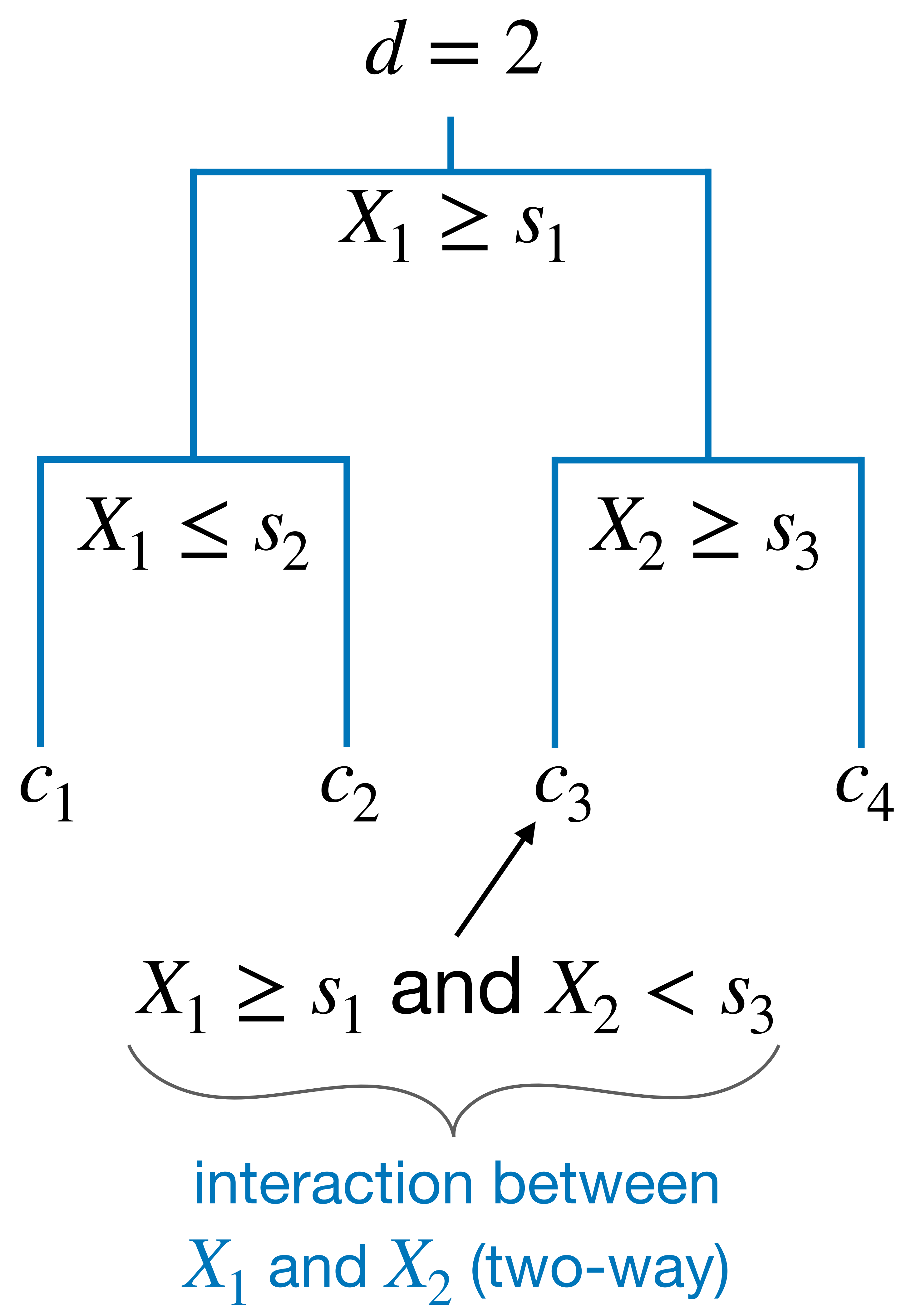
Interaction depth

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

The prediction rule involves making decisions based on X_1 and X_2 together (two-way interactions).

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.



Interaction depth

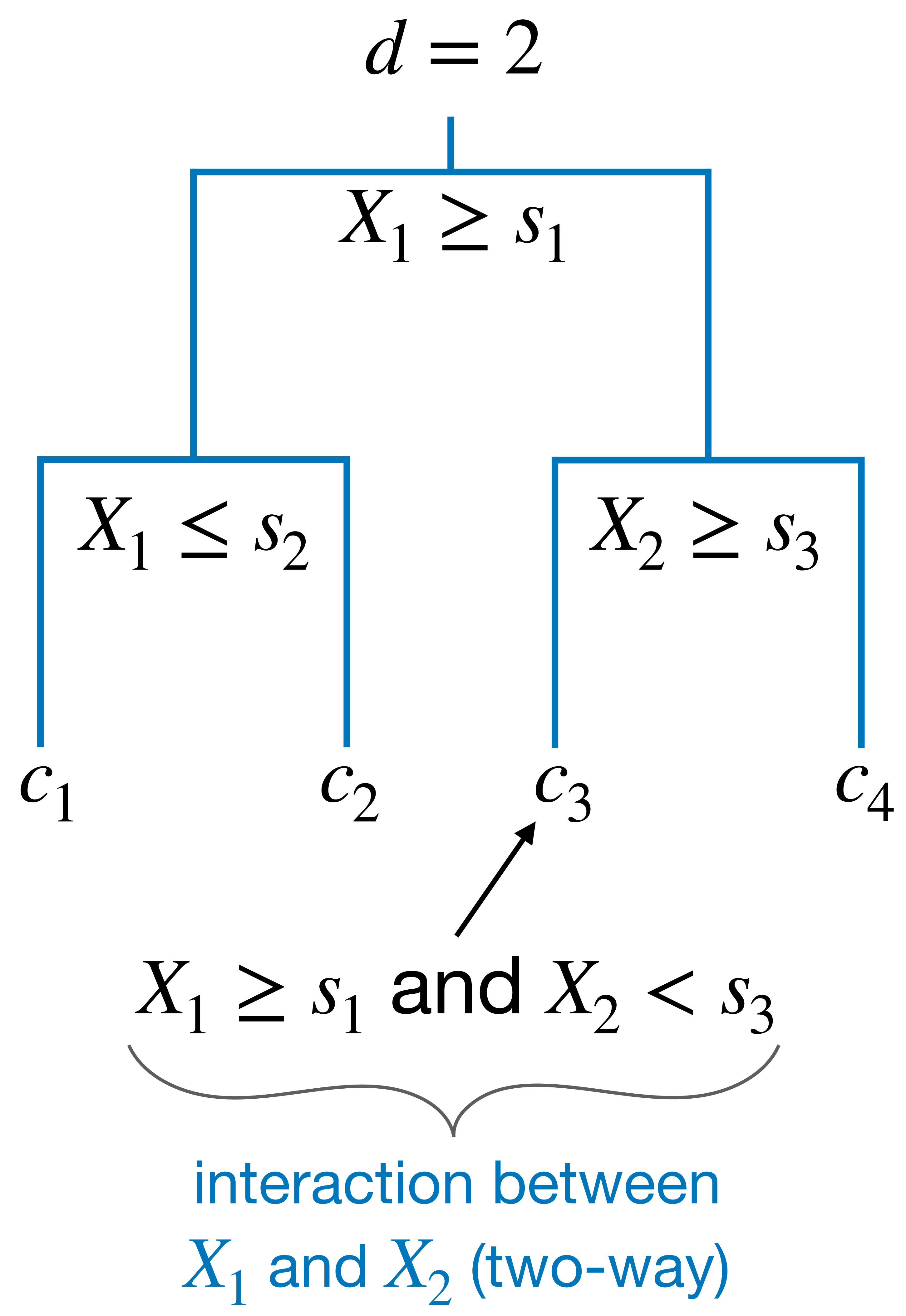
Consider a single tree, for example with $d = 2$.

The prediction rule involves making decisions based on X_1 and X_2 together (two-way interactions).

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.

$$d = 1: \quad \hat{f}(X_1, X_2, X_3) = \hat{g}_1(X_1) + \hat{g}_2(X_2) + \hat{g}_3(X_3)$$



Interaction depth

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

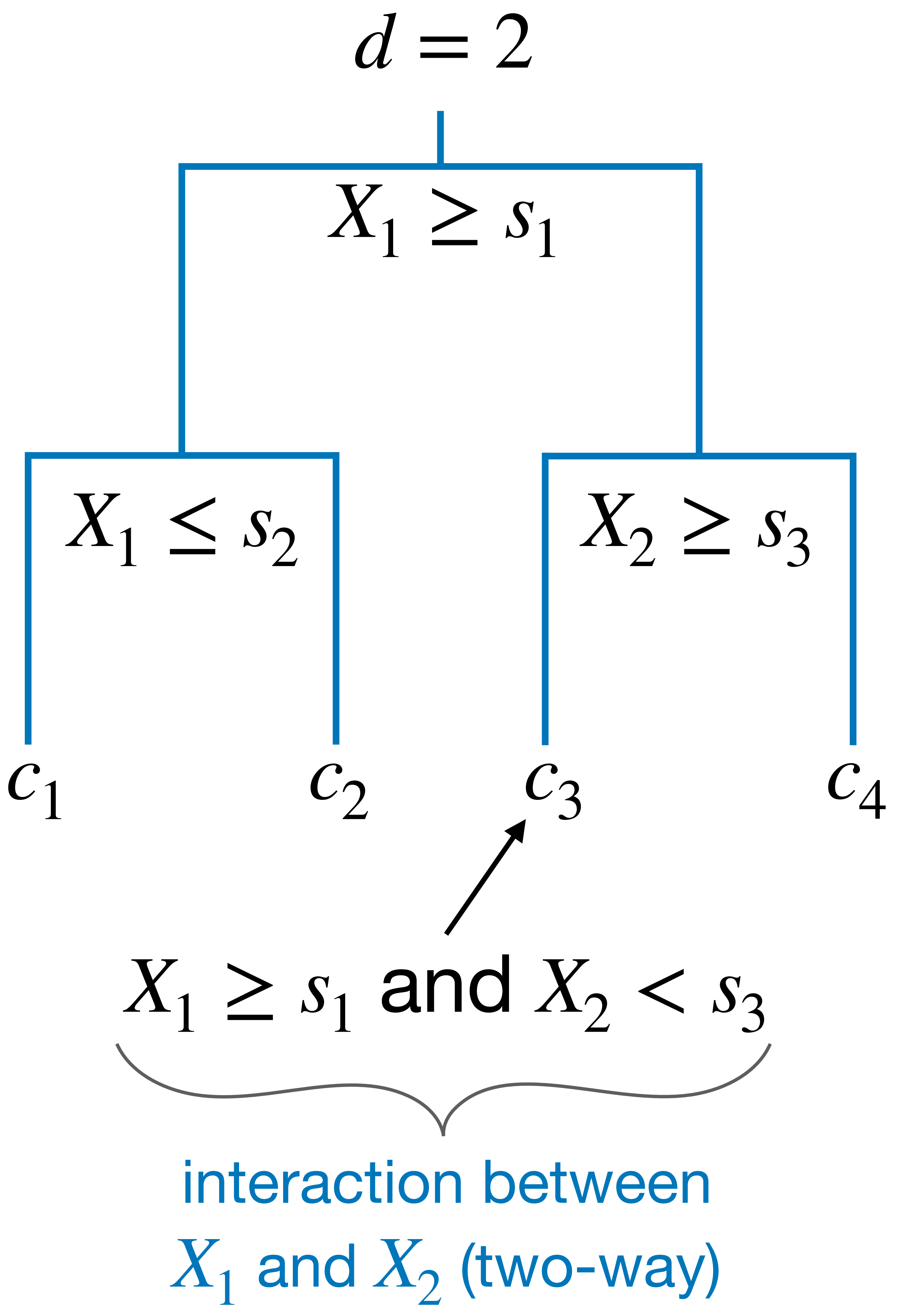
The prediction rule involves making decisions based on X_1 and X_2 together (two-way interactions).

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.

$$d = 1: \quad \hat{f}(X_1, X_2, X_3) = \hat{g}_1(X_1) + \hat{g}_2(X_2) + \hat{g}_3(X_3)$$

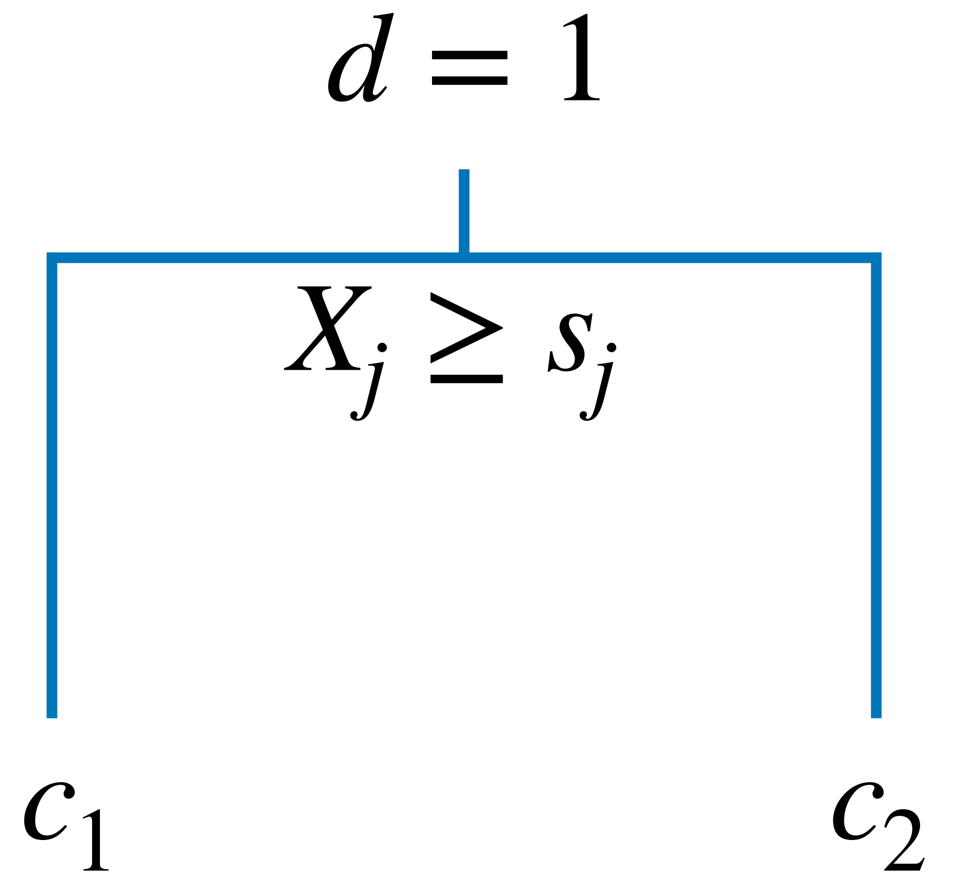
$$d = 2: \quad \hat{f}(X_1, X_2, X_3) = \hat{g}_{12}(X_1, X_2) + \hat{g}_{23}(X_2, X_3) + \hat{g}_{13}(X_1, X_3)$$



Special case: $d = 1$

Special case: $d = 1$

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



A “stump”

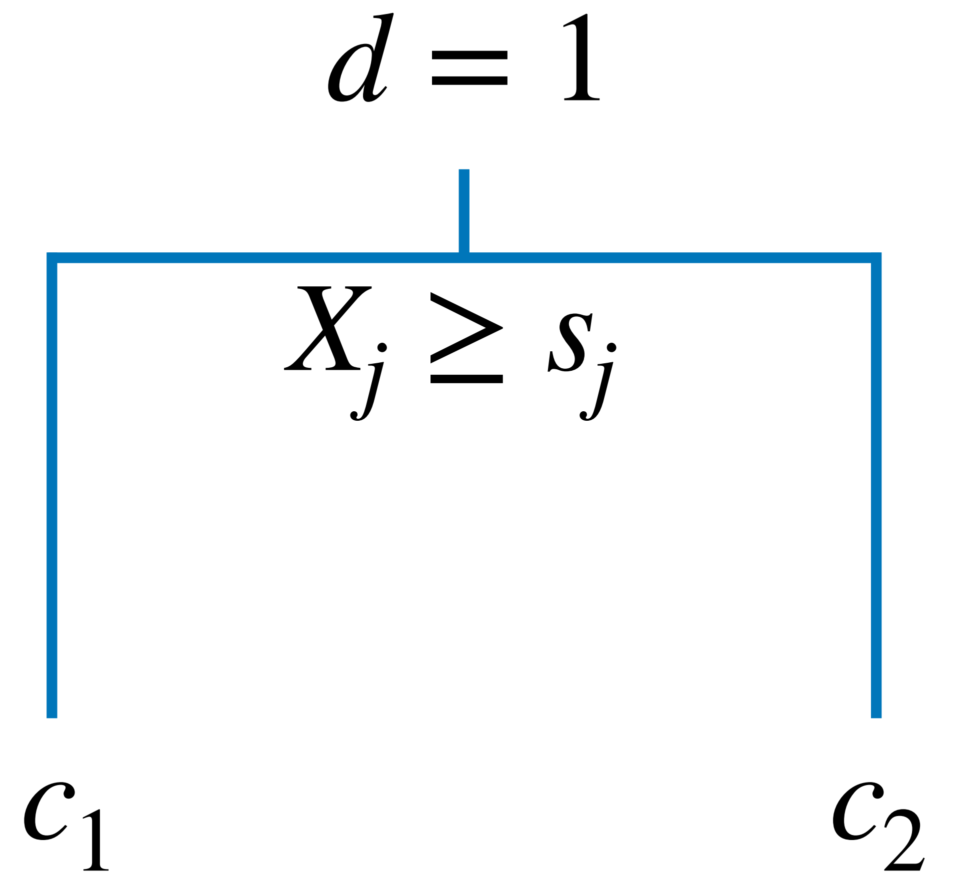
Special case: $d = 1$

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 $\hat{g}_j$.



A “stump”

Special case: $d = 1$

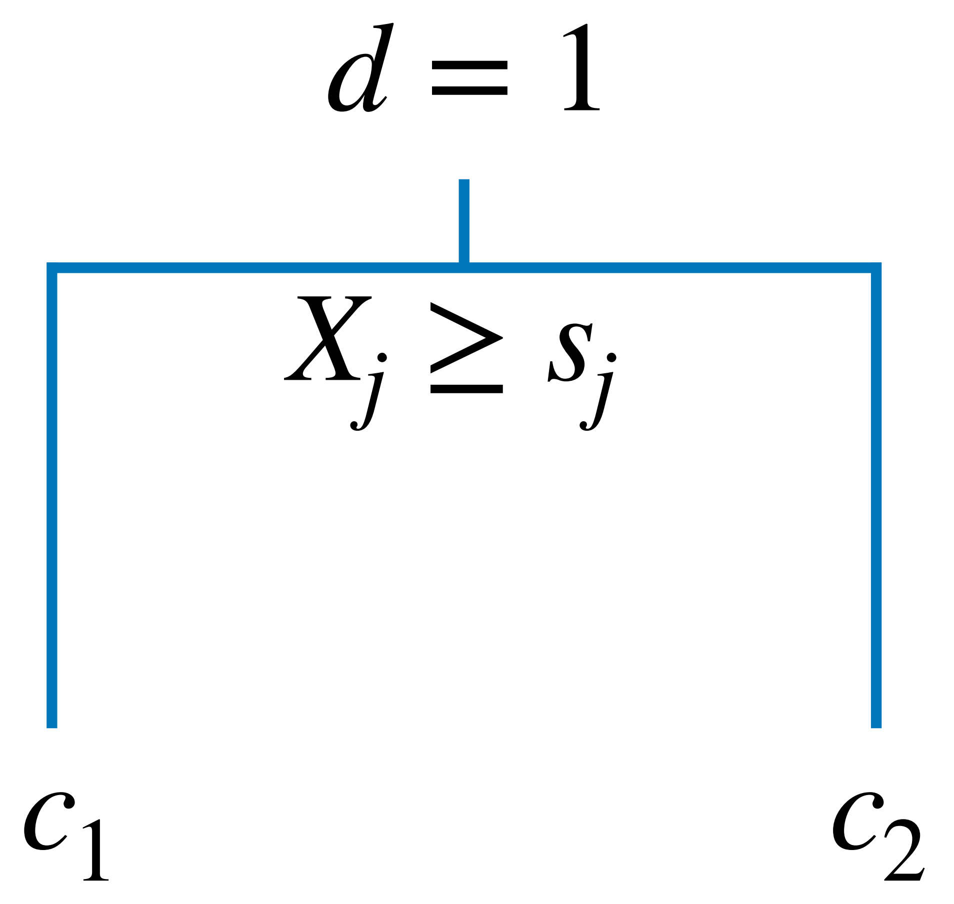
When $d = 1$, each tree has only one split; such trees are called stumps.

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

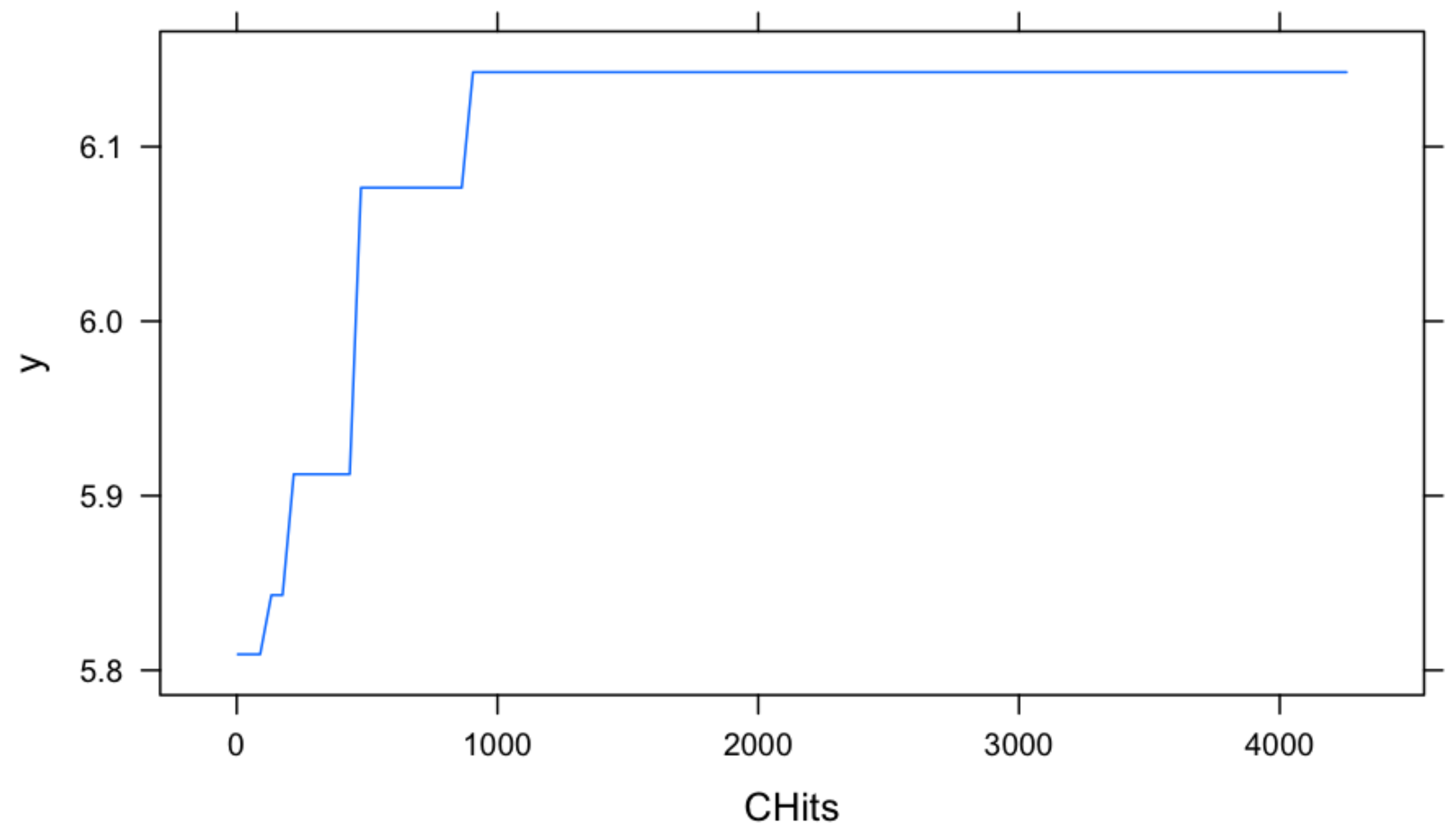
$$\hat{f}(X) = \hat{g}_1(X_1) + \hat{g}_2(X_2) + \cdots + \hat{g}_p(X_p)$$

for some coordinate functions $\hat{g}_j$.

The coordinate functions can be easily plotted and interpreted.



A "stump"



Derivation of coordinate functions

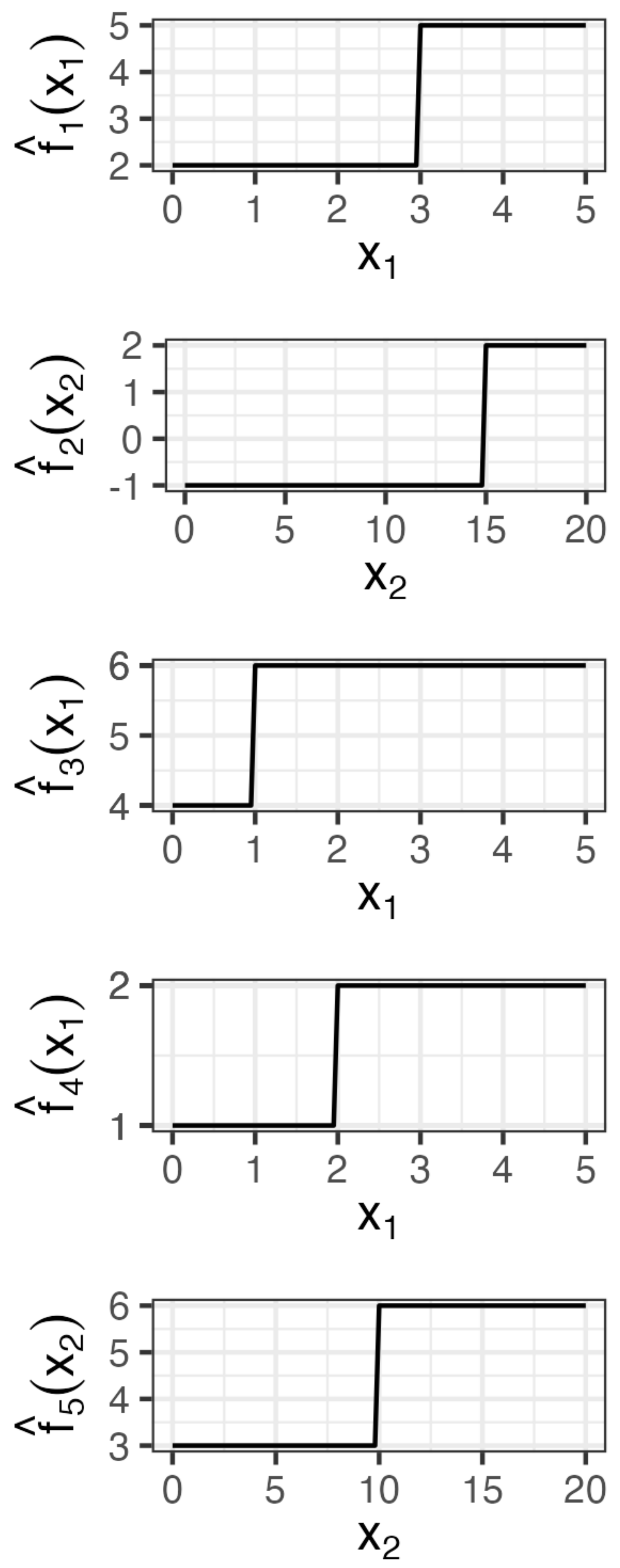
Get coordinate functions by grouping stumps splitting on the same variable:

$$\hat{f}(X_1, X_2) = \hat{f}_1(X_1) + \hat{f}_2(X_2) + \hat{f}_3(X_1) + \hat{f}_4(X_1) + \hat{f}_5(X_2)$$

Derivation of coordinate functions

Get coordinate functions by grouping stumps splitting on the same variable:

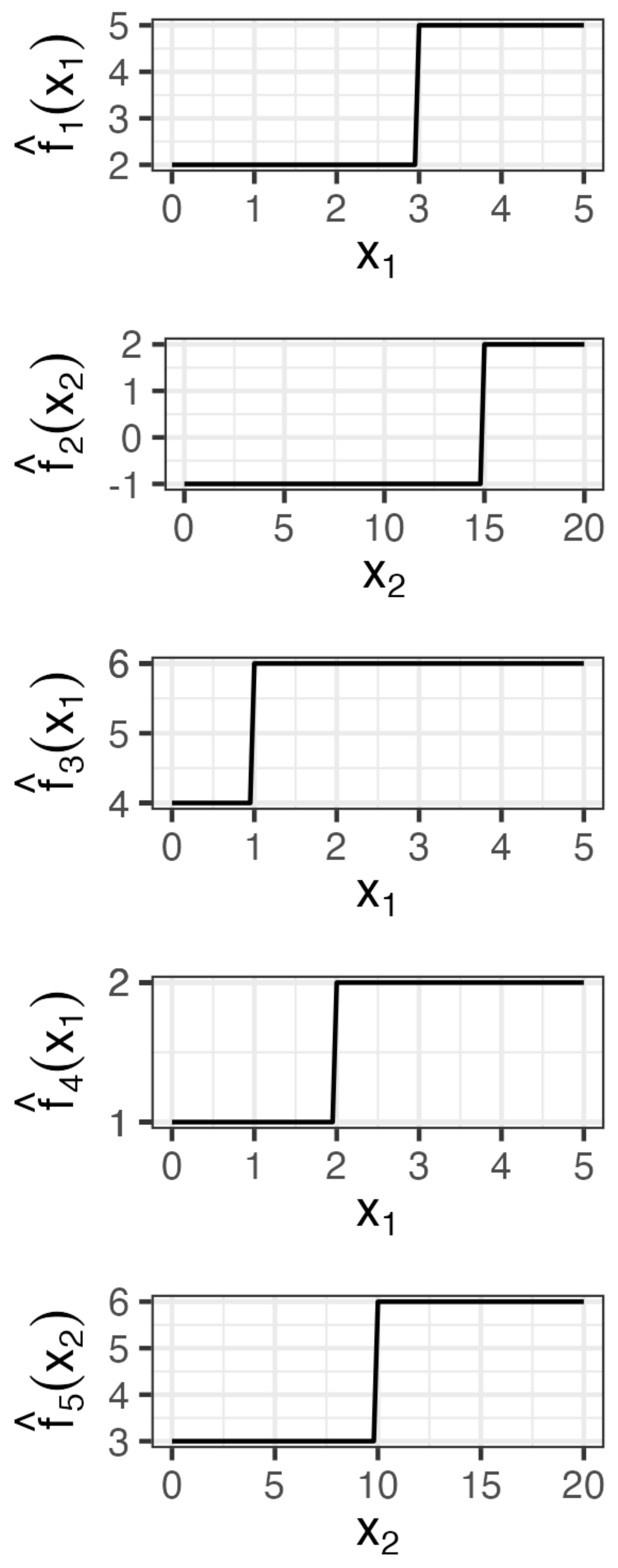
$$\hat{f}(X_1, X_2) = \hat{f}_1(X_1) + \hat{f}_2(X_2) + \hat{f}_3(X_1) + \hat{f}_4(X_1) + \hat{f}_5(X_2)$$



Derivation of coordinate functions

Get coordinate functions by grouping stumps splitting on the same variable:

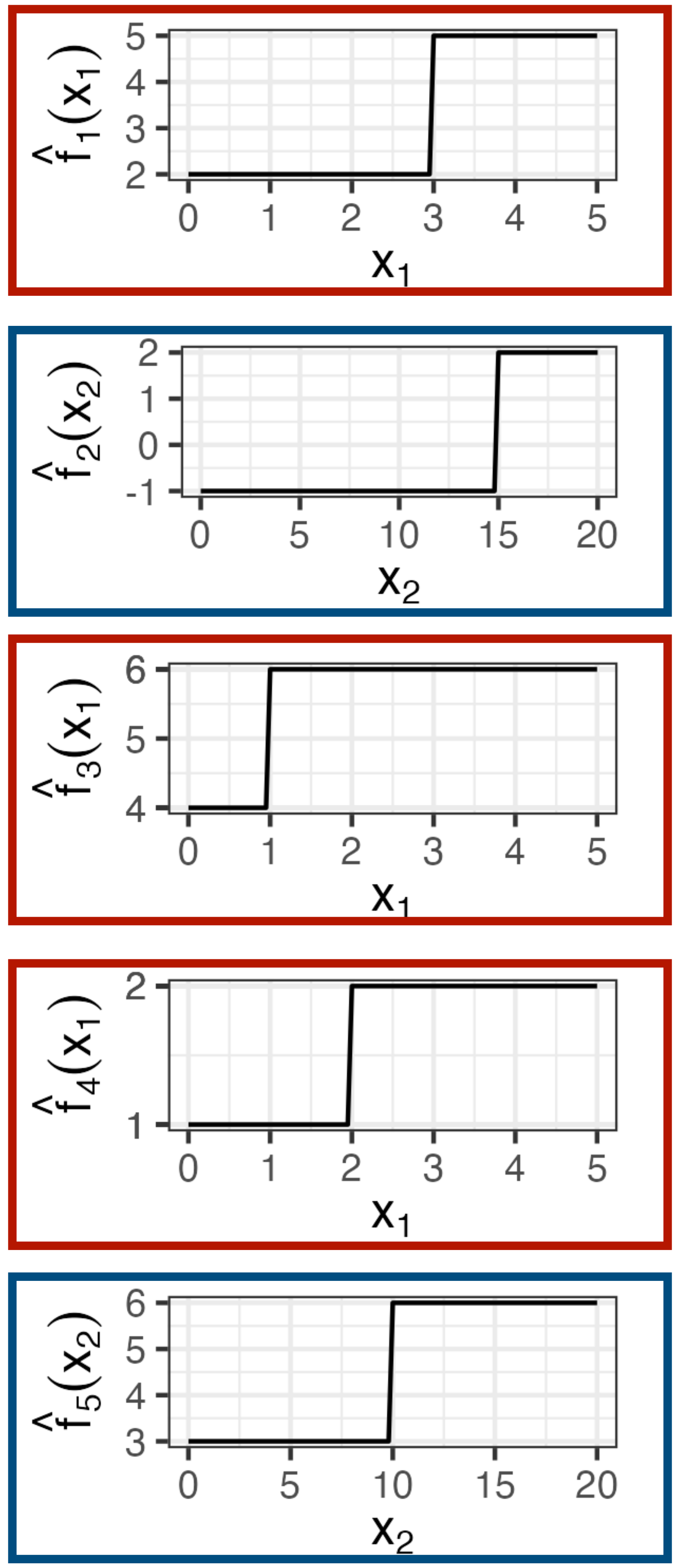
$$\hat{f}(X_1, X_2) = \hat{f}_1(X_1) + \hat{f}_2(X_2) + \hat{f}_3(X_1) + \hat{f}_4(X_1) + \hat{f}_5(X_2)$$
$$= \left(\hat{f}_1(X_1) + \hat{f}_3(X_1) + \hat{f}_4(X_1)\right) + \left(\hat{f}_2(X_2) + \hat{f}_5(X_2)\right)$$



Derivation of coordinate functions

Get coordinate functions by grouping stumps splitting on the same variable:

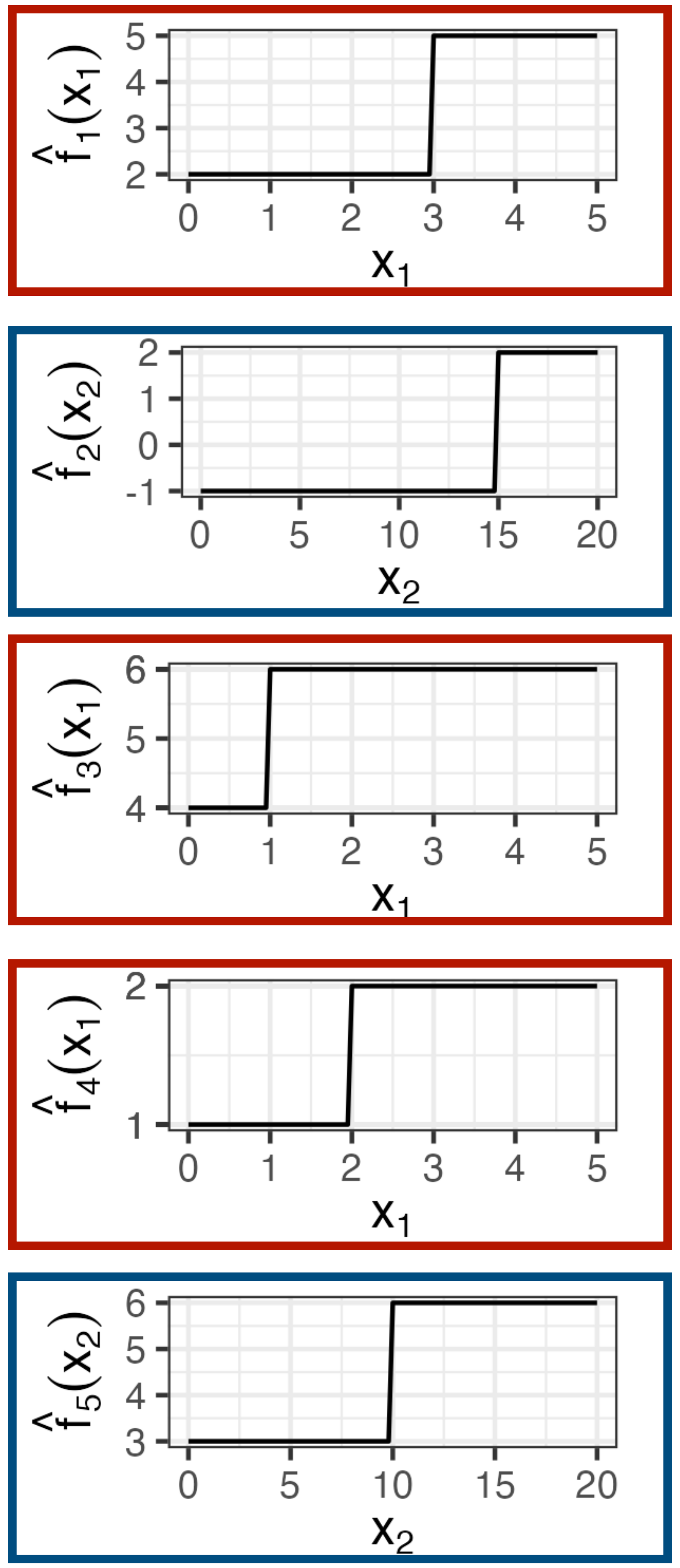
$$
\begin{aligned}
\hat{f}(X_1, X_2) &= \hat{f}_1(X_1) + \hat{f}_2(X_2) + \hat{f}_3(X_1) + \hat{f}_4(X_1) + \hat{f}_5(X_2) \\
&= \left(\hat{f}_1(X_1) + \hat{f}_3(X_1) + \hat{f}_4(X_1)\right) + \left(\hat{f}_2(X_2) + \hat{f}_5(X_2)\right)
\end{aligned}
$$



Derivation of coordinate functions

Get coordinate functions by grouping stumps splitting on the same variable:

$$
\begin{aligned}
\hat{f}(X_1, X_2) &= \hat{f}_1(X_1) + \hat{f}_2(X_2) + \hat{f}_3(X_1) + \hat{f}_4(X_1) + \hat{f}_5(X_2) \\
&= \left(\hat{f}_1(X_1) + \hat{f}_3(X_1) + \hat{f}_4(X_1)\right) + \left(\hat{f}_2(X_2) + \hat{f}_5(X_2)\right) \\
&= \hat{g}_1(X_1) + \hat{g}_2(X_2)
\end{aligned}
$$



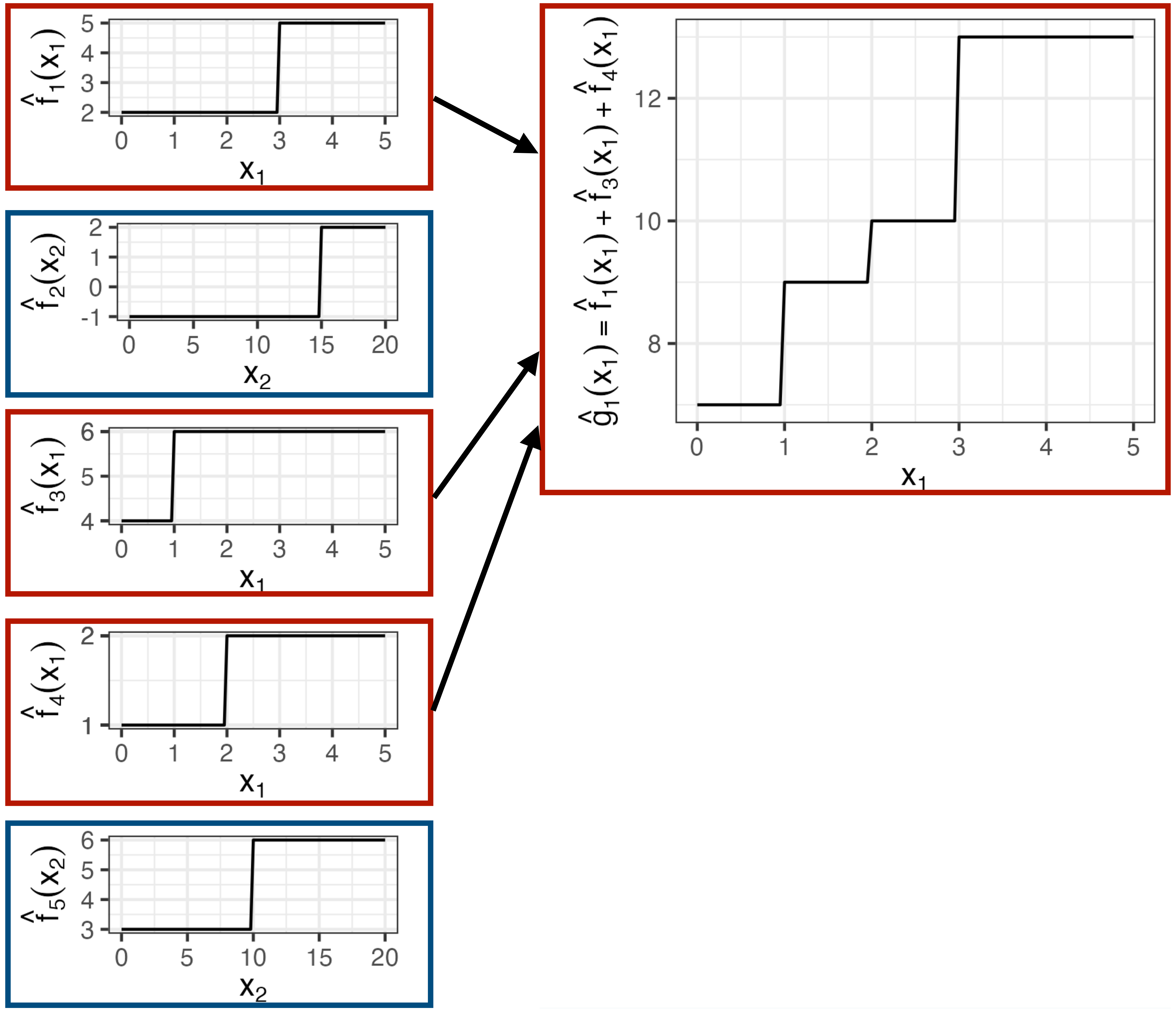
Derivation of coordinate functions

Get coordinate functions by grouping stumps splitting on the same variable:

$$\hat{f}(X_1, X_2) = \hat{f}_1(X_1) + \hat{f}_2(X_2) + \hat{f}_3(X_1) + \hat{f}_4(X_1) + \hat{f}_5(X_2)$$

$$= \left(\hat{f}_1(X_1) + \hat{f}_3(X_1) + \hat{f}_4(X_1)\right) + \left(\hat{f}_2(X_2) + \hat{f}_5(X_2)\right)$$

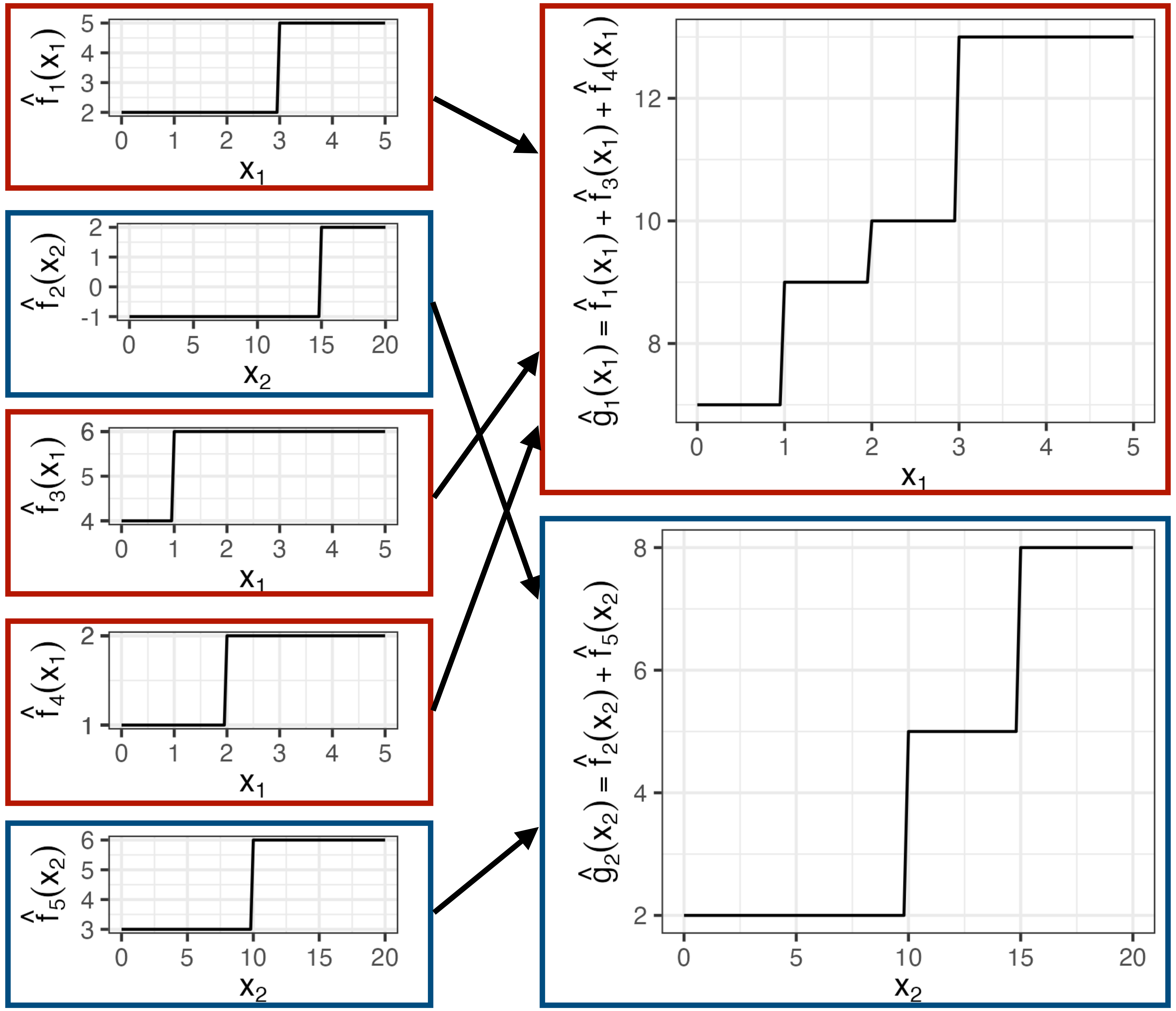
$$= \hat{g}_1(X_1) + \hat{g}_2(X_2)$$



Derivation of coordinate functions

Get coordinate functions by grouping stumps splitting on the same variable:

$$\hat{f}(X_1, X_2) = \hat{f}_1(X_1) + \hat{f}_2(X_2) + \hat{f}_3(X_1) + \hat{f}_4(X_1) + \hat{f}_5(X_2)$$
$$= \left(\hat{f}_1(X_1) + \hat{f}_3(X_1) + \hat{f}_4(X_1)\right) + \left(\hat{f}_2(X_2) + \hat{f}_5(X_2)\right)$$
$$= \hat{g}_1(X_1) + \hat{g}_2(X_2)$$



Stochastic gradient boosting

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.

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.

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.

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.

Tuning gradient boosting

Tuning gradient boosting

Parameters:

Tuning gradient boosting

Parameters:

- Shrinkage parameter λ: Choose some small number, e.g. $\lambda = 0.1$.

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.

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.

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

Model interpretation

Variable importance measure:

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

Model interpretation

Variable importance measure:

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

Partial dependence plots:

Model interpretation

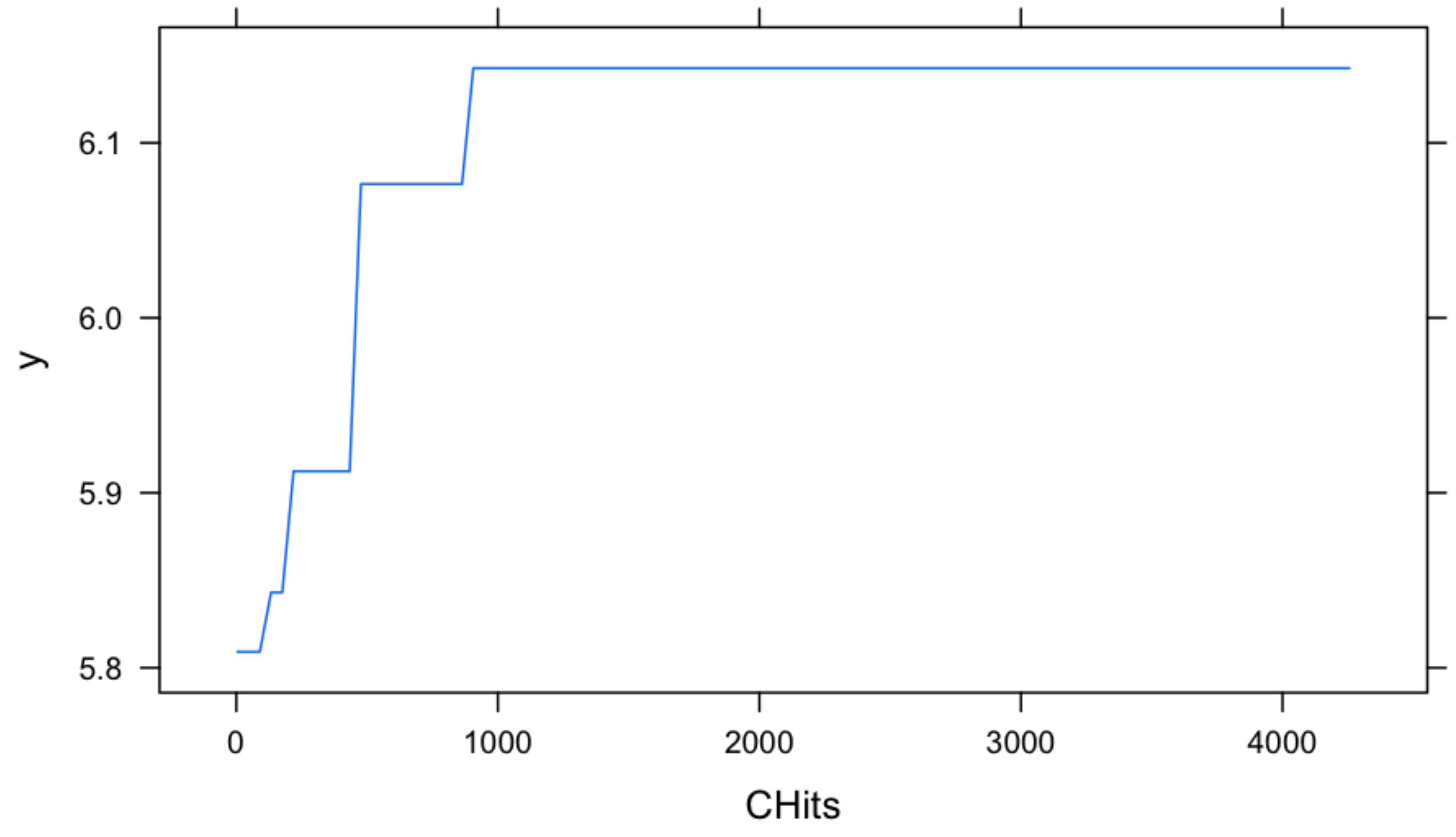
Variable importance measure:

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

Partial dependence plots:

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

$$\hat{f}(X) = \hat{g}_1(X_1) + \cdots + \hat{g}_B(X_B).$$



Model interpretation

Variable importance measure:

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

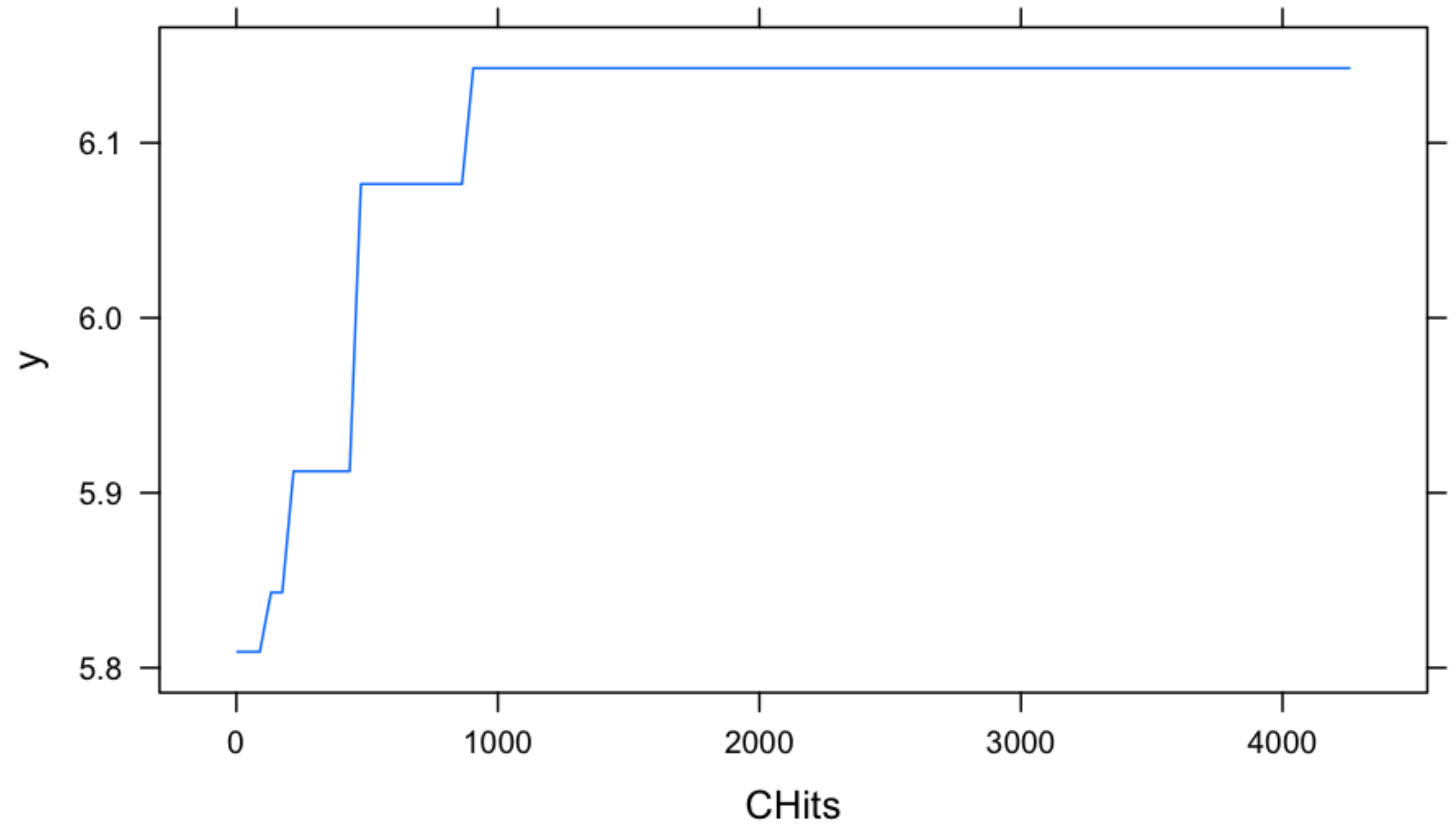
Partial dependence plots:

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

$$\hat{f}(X) = \hat{g}_1(X_1) + \cdots + \hat{g}_B(X_B).$$

For $d > 1$, can define a generalization of $\hat{g}_j$ such that

$$\hat{f}(X) \approx \hat{g}_1(X_1) + \cdots + \hat{g}_B(X_B).$$



Model interpretation

Variable importance measure:

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

Partial dependence plots:

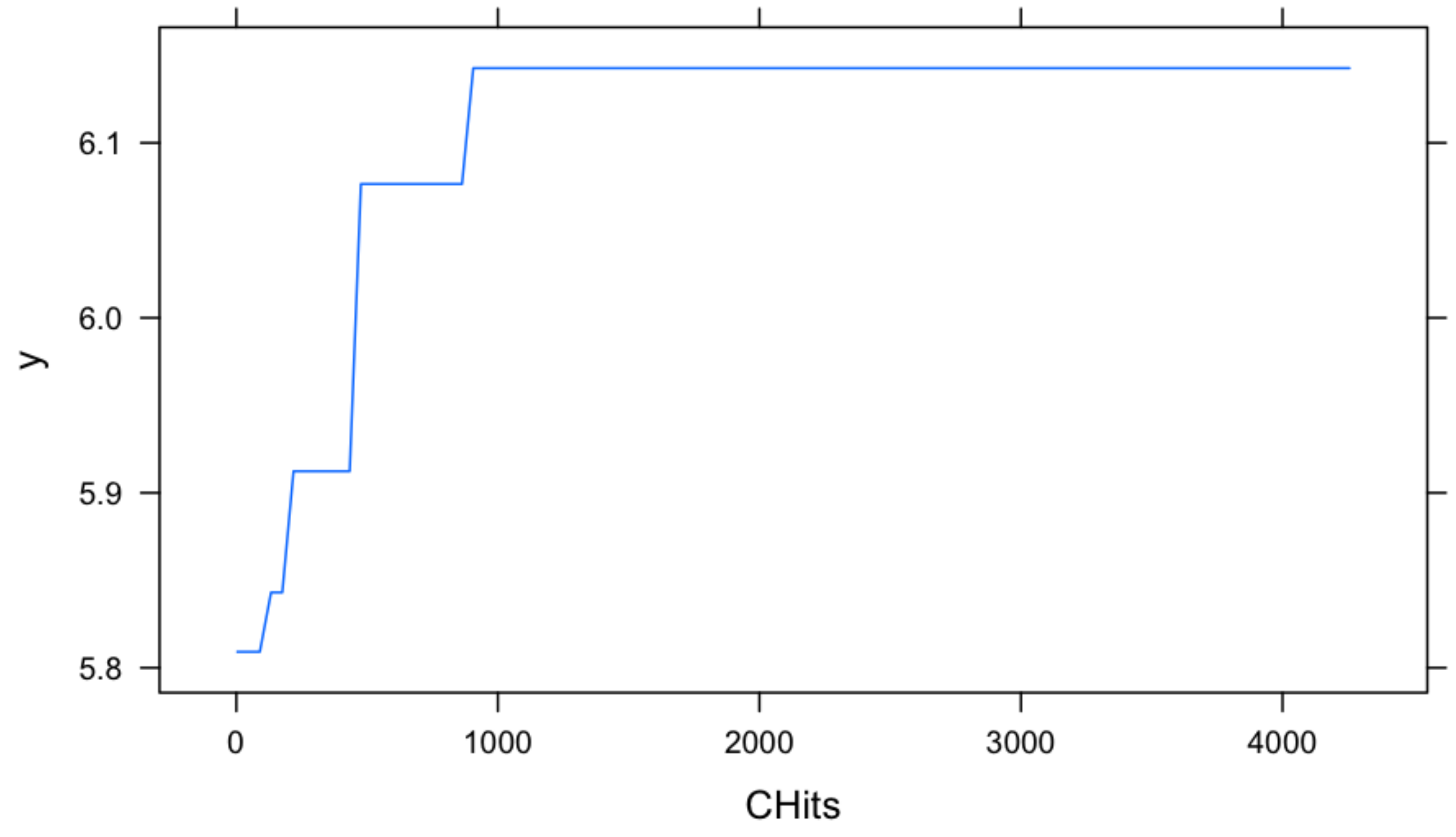
For $d = 1$, the coordinate functions $\widehat{g}_j$ show exactly how $\hat{f}$ depends on X_j, i.e.

$$\hat{f}(X) = \widehat{g}_1(X_1) + \cdots + \widehat{g}_B(X_B).$$

For $d > 1$, can define a generalization of $\widehat{g}_j$ such that

$$\hat{f}(X) \approx \widehat{g}_1(X_1) + \cdots + \widehat{g}_B(X_B).$$

The larger d is, the worse the approximation.



Boosting for classification

Boosting for classification

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

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.

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.

Implementation of boosting for classification is beyond the scope of the class, but the same intuitions from this lecture carry over to boosting for classification.

Summary

Summary

- Like random forests, boosting aggregates the results of many decision trees to build a predictive model with state-of-the-art performance.

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.

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.

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.

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.