Machine Learning in R

Lecture 6 — Boosting; Things to watch out for when doing supervised learning

Jean Feng – Epidemiology and Biostatistics

Last Lecture

- Decision Trees
- Ensembling
    - Bagging
    - Random Forests
- Variable Importance

Today's Lecture

- Boosting
  - AdaBoost
  - Forward Stagewise Additive Modeling
  - Gradient boosting
- Issues to think about
  - Confounders
  - Fairness

Today's Lecture

- **Boosting**
  - AdaBoost
  - Forward Stagewise Additive Modeling
  - Gradient boosting
- Issues to think about
  - Confounders
  - Fairness

The hypothesis boosting problem

- Suppose there is an efficient learning algorithm whose performance, on any dataset, is slightly better than random guessing.

“Weak learner”:
an algorithm that does slightly better than random

- Can these weak learners be combined into an efficient algorithm to achieve very good prediction accuracy?

“Strong learner”:
an algorithm that achieves arbitrarily good prediction accuracy as the sample size grows

- Can we “boost” a set of weak learners to create a single strong learner?

The answer is yes!

Boosting algorithms

- Boosting is a general-purpose method that builds an ensemble of “weak learners”.

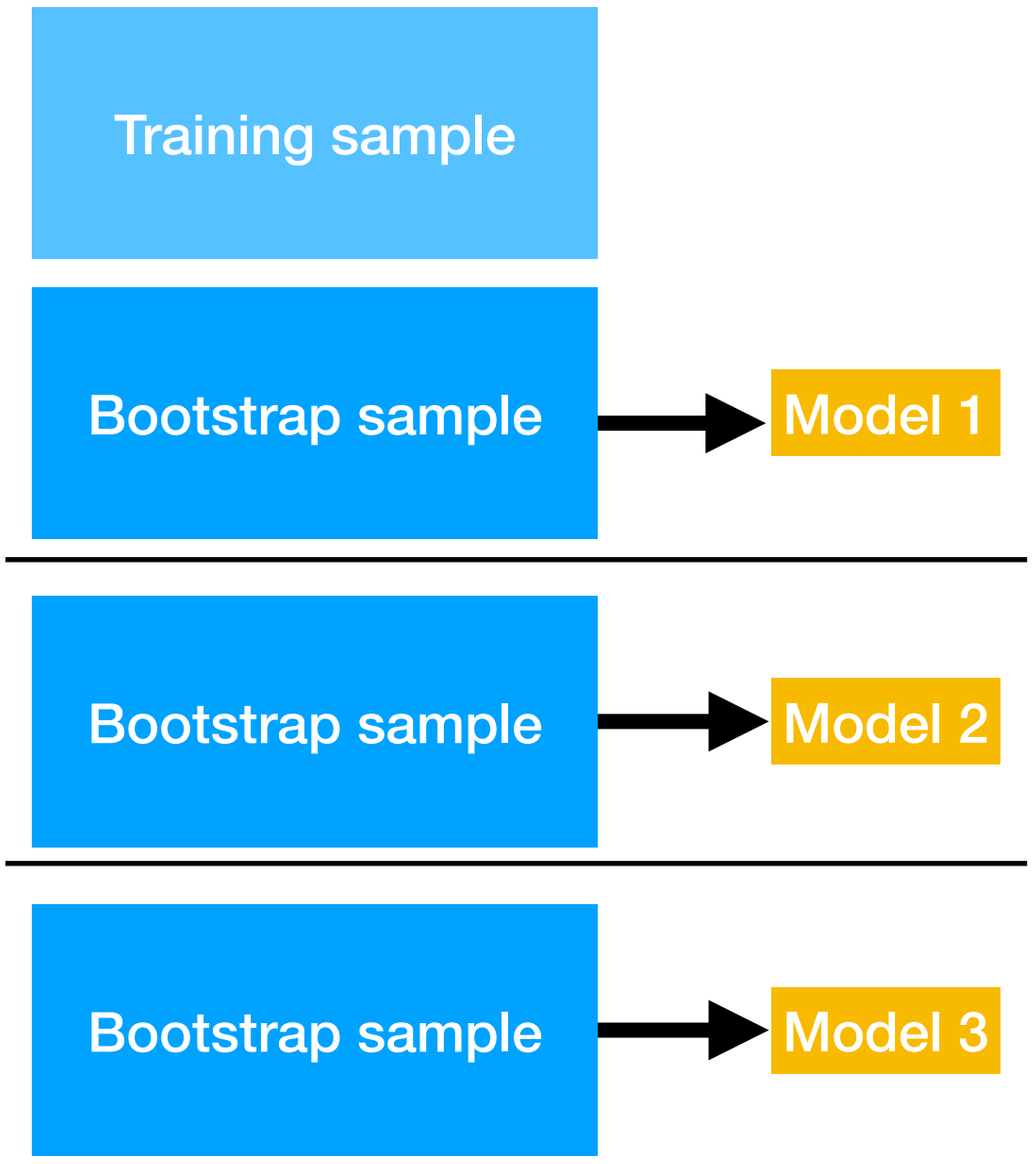
- Like bagging and random forests, boosting averages the predictions from each weak learner to make a prediction.

- Unlike bagging and random forests, boosting trains each weak learner ***sequentially***. Boosting optimizes how the weak learners are combined to maximize prediction accuracy.

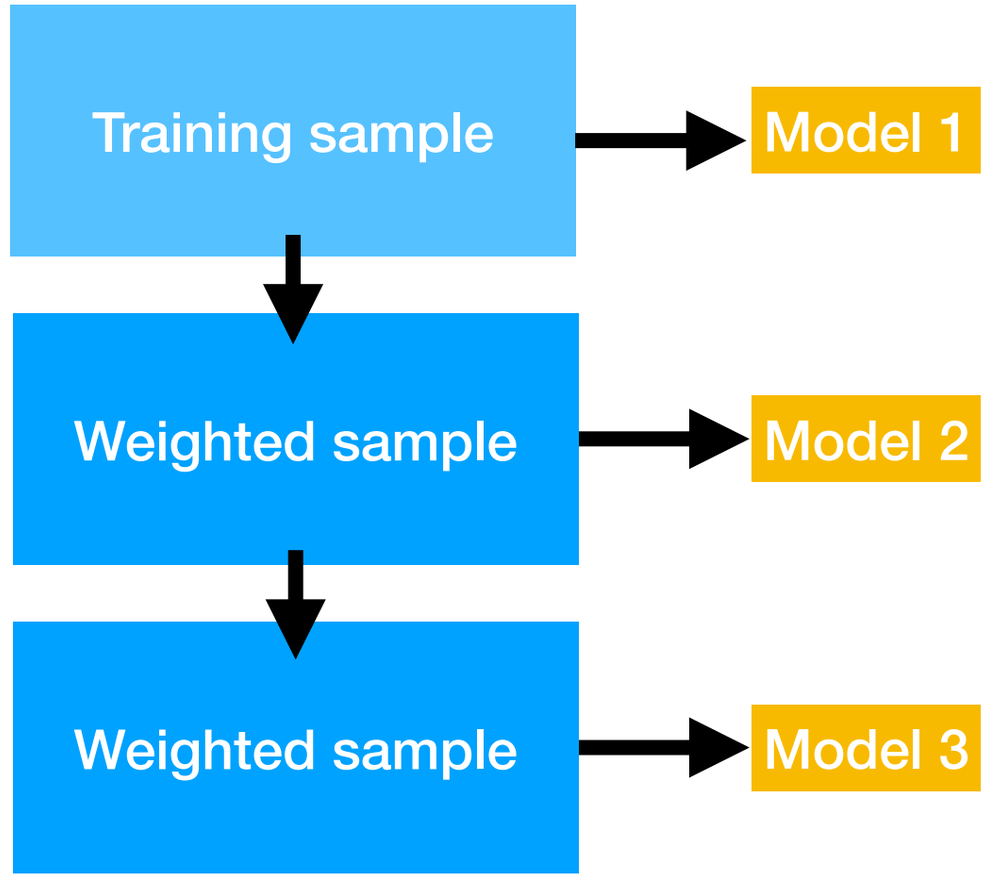
- Can be used for both regression and classification.

Ensemble methods

Bagging-type methods:
Models are built independently



Boosting:
Models build off one another



The many variants of Boosting

- **Adaptive Boosting (AdaBoost):** The first major boosting algorithm. Designed for classification.

- **Forward Stagewise Additive Modeling**: Generalization of AdaBoost

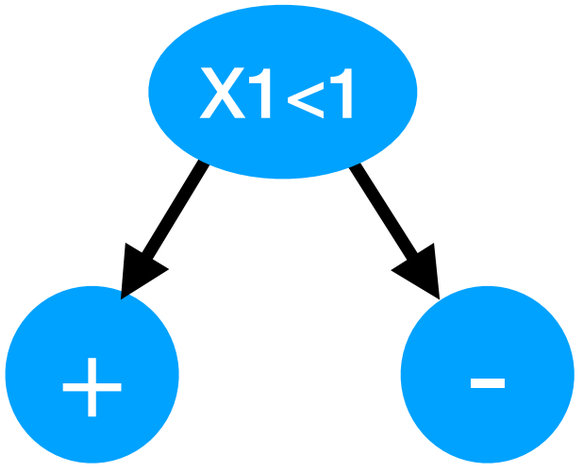
- **Gradient Boosting**: A gradient-based perspective of Boosting

Today's Lecture

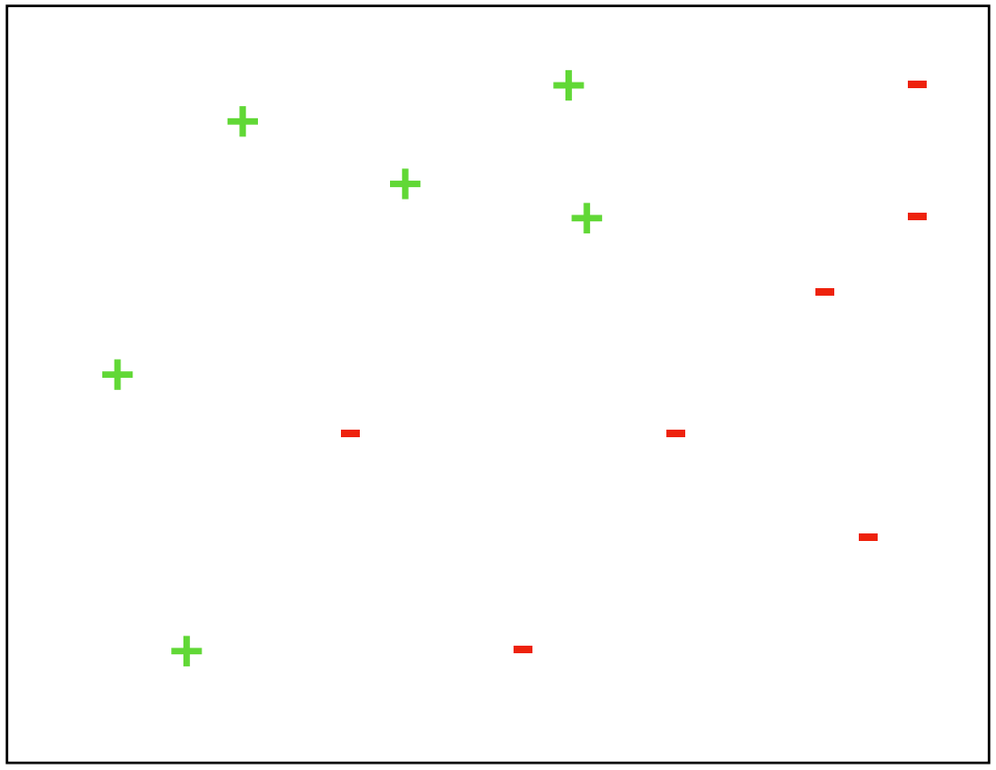
- Boosting
  - **AdaBoost**
  - Forward Stagewise Additive Modeling
  - Gradient boosting
- Issues to think about
  - Confounders
  - Fairness

AdaBoost: the basic idea

- Let's suppose our "weak learners" are decision stumps:

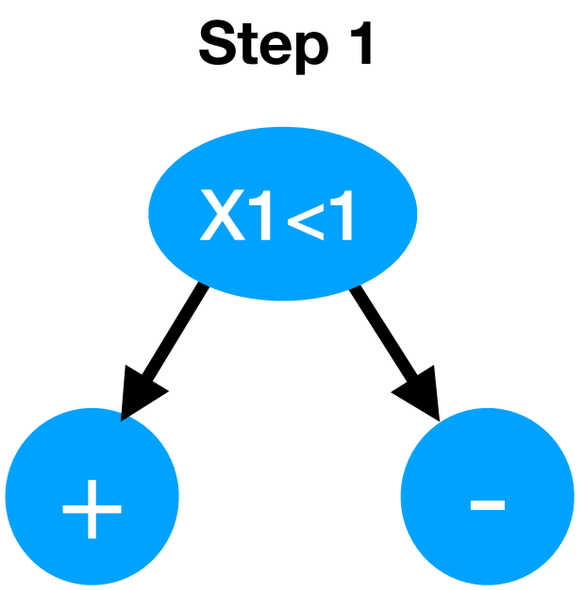


Ex: Decision stump

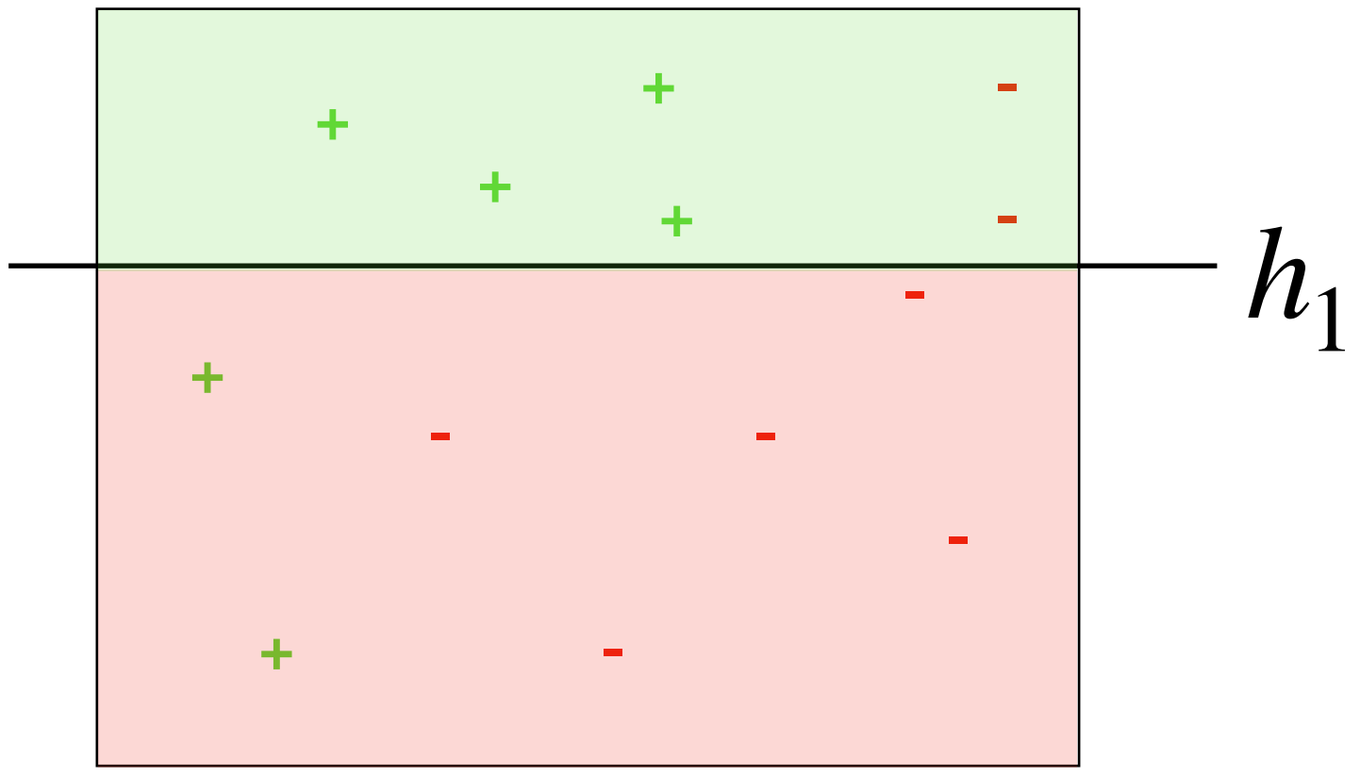


AdaBoost: the basic idea

- Let's suppose our "weak learners" are decision stumps:



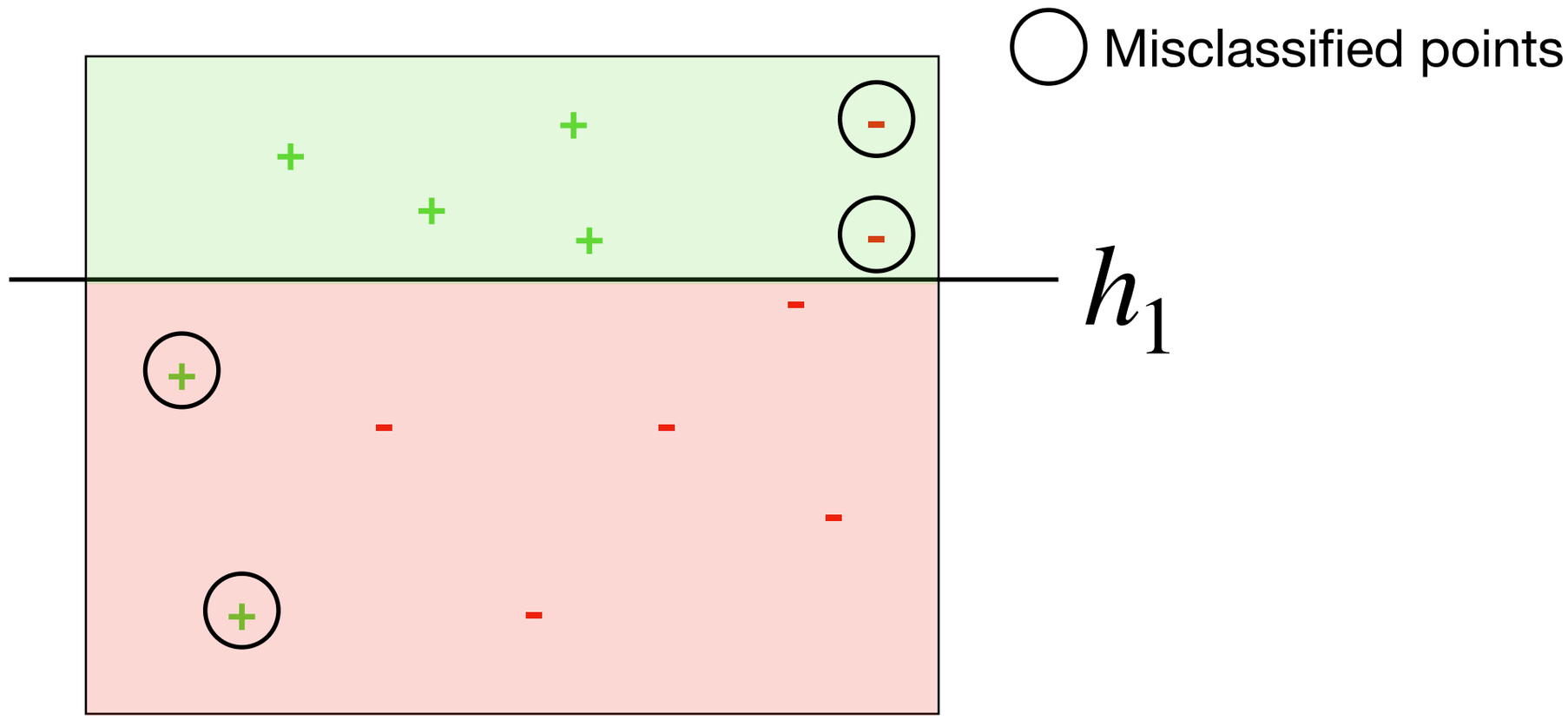
Ex: Decision stump



AdaBoost: the basic idea

- There are a lot of points that are misclassified. Let's try to fix this.

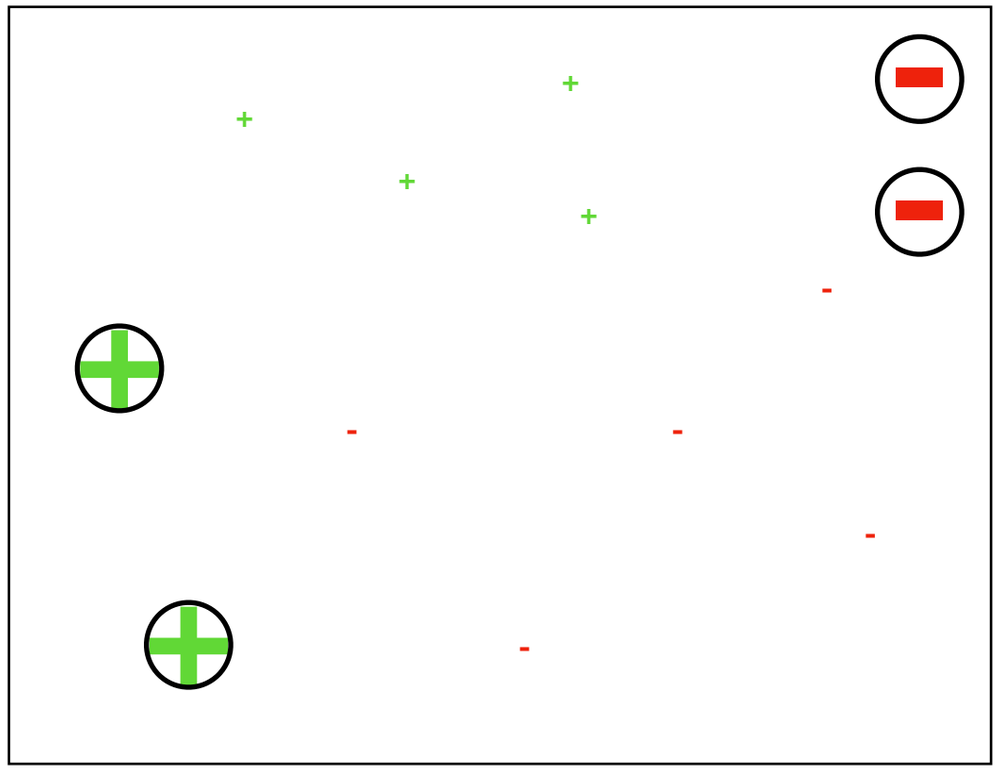
Step 1



AdaBoost: the basic idea

- Let's upweight misclassified points and downweight correctly classified points. Fit a new decision stump on this reweighed dataset.

Step 2

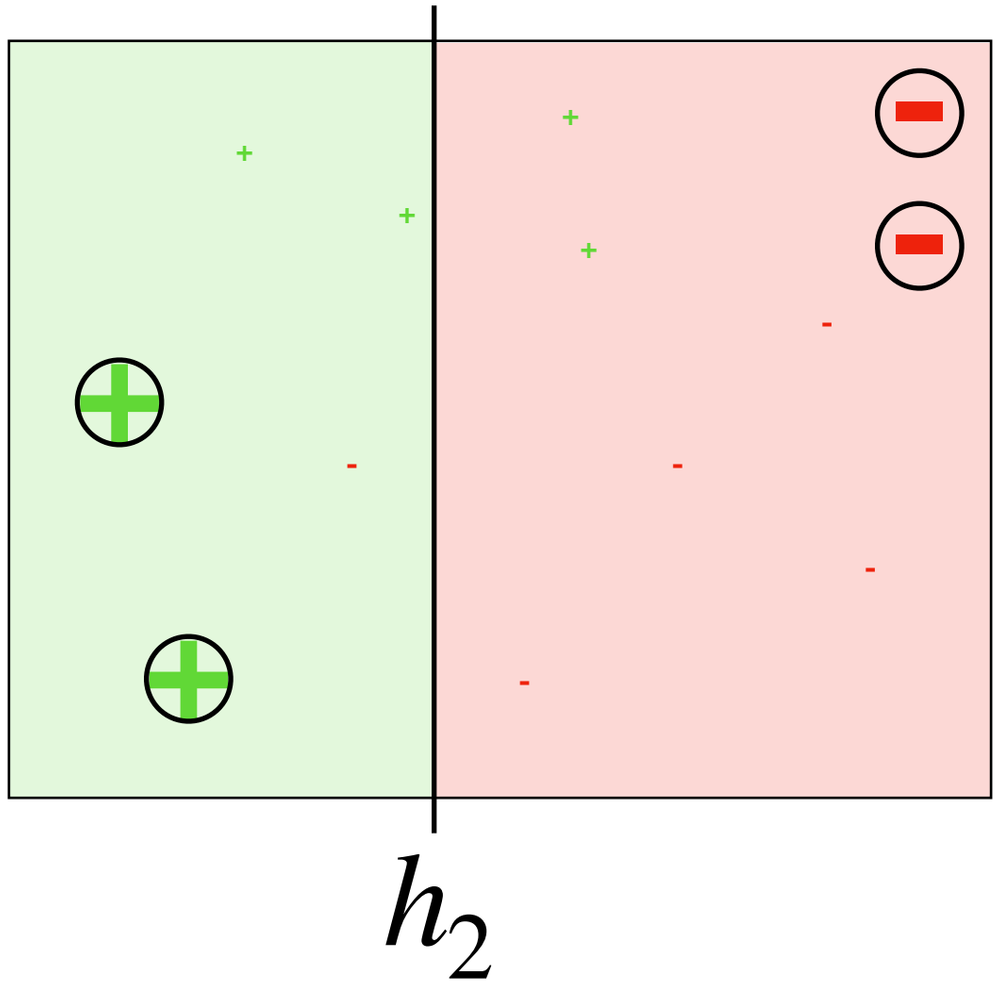


Misclassified points

AdaBoost: the basic idea

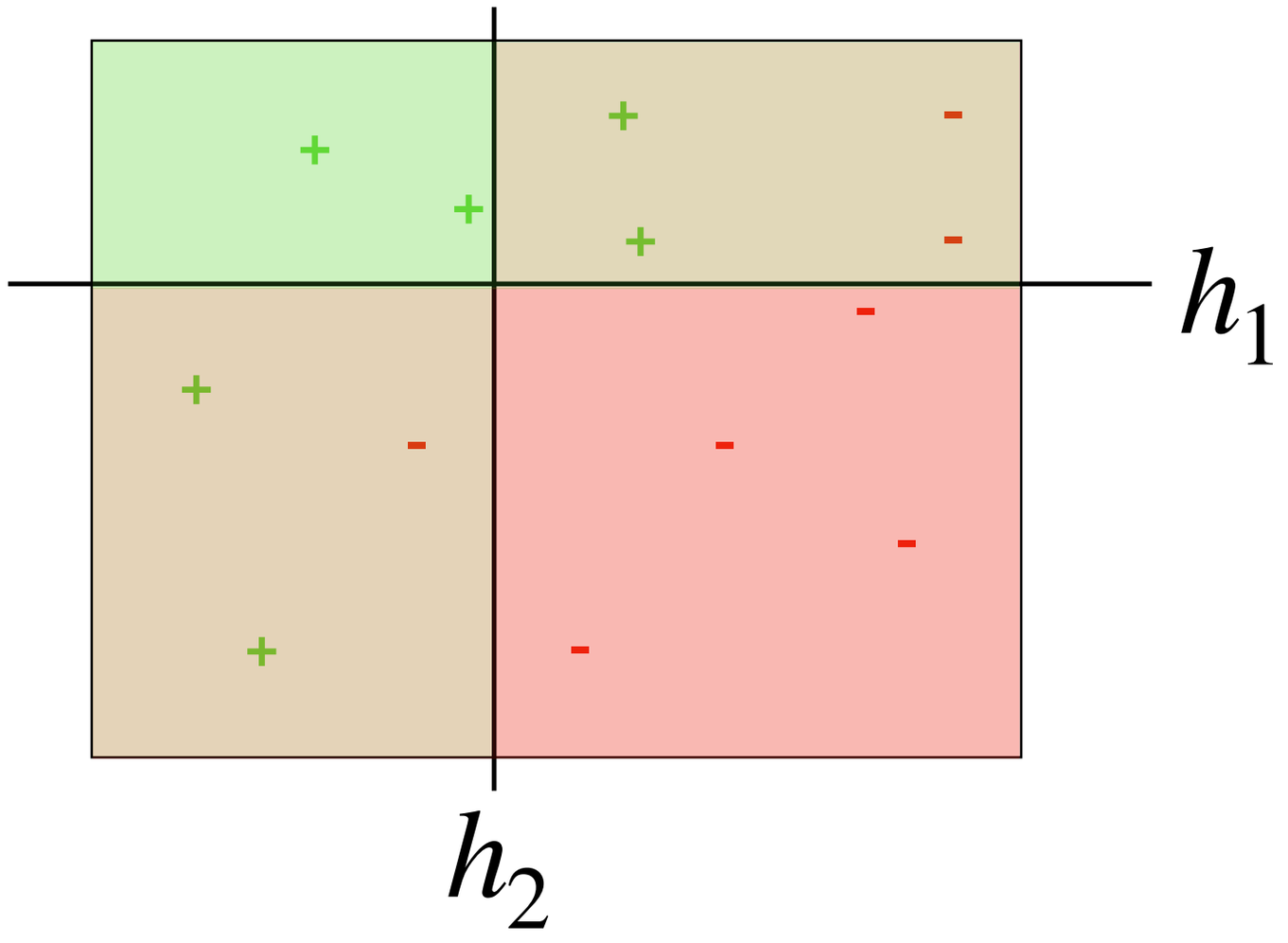
- Based on this reweighed dataset, we fit the following decision stump:

Step 2



AdaBoost: the basic idea

- The combined model looks like:



Moreover, we can take a weighted average of these two models:

$$f(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x)$$

Weights

AdaBoost

- Start with weighting all observations equally: $w_i = 1/n$ for $i = 1,2,...,n$.
- For b = 1,2,…B:
  - Fit a new classifier h_b for dataset with weights w_i.
  - Compute the weighted error of this classifier: $\text{err} = \dfrac{\sum_{i=1}^{n} w_i 1\{Y_i \neq h_b(X_i)\}}{\sum_{i=1}^{n} w_i}$
  - Set $\alpha_m = \log\left(\dfrac{1-\text{err}}{\text{err}}\right)$
  - Update weights $w_i \leftarrow w_i \exp\left(\alpha_m 1\{Y_i \neq h_b(X_i)\}\right)$.
- Final model: $f(x) = sign\left(\sum_{b=1}^{B} \alpha_b h_b(x)\right)$.

Fit a new model to fix up mistakes made by the previous models. h_b can be thought of as a set of basis functions, e.g. a set of decision stumps

Calculate the average performance of this weak learner

Decide the weight assigned to this weak learner.

Decrease the weight of observations correctly classified by h_b. Increase the weight of observations incorrectly classified by h_b.

Q: Which of these statements are true about the bias-variance tradeoff in AdaBoost, where the weak learners are decision trees?

- A: Increasing the number of iterations/models B will increase bias and decrease variance.

- B: Increasing the number of iterations/models B will decrease bias and increase variance.

- C: Using a tree with more levels will increase bias and decrease variance.

- D: Using a tree with more levels will decrease bias and increase variance.

Increasing the number of basis functions in linear regression tends to increase the chance of overfitting (higher variance, lower bias). Similar concept holds here.

Hyperparameters in boosting

- *Number of iterations/models B**:*** Unlike bagging and random forests, the model complexity and the risk of overfitting ***increases*** with *B in boosting.*

- *Complexity of the weak learners:* If you use weaker learners, the training error will decrease slower.
  - When the training error decreases slowly over time, we are able to have very fine control over the bias-variance trade-off.
  - The drawback of using weaker learners is that they increase computation time.

Today's Lecture

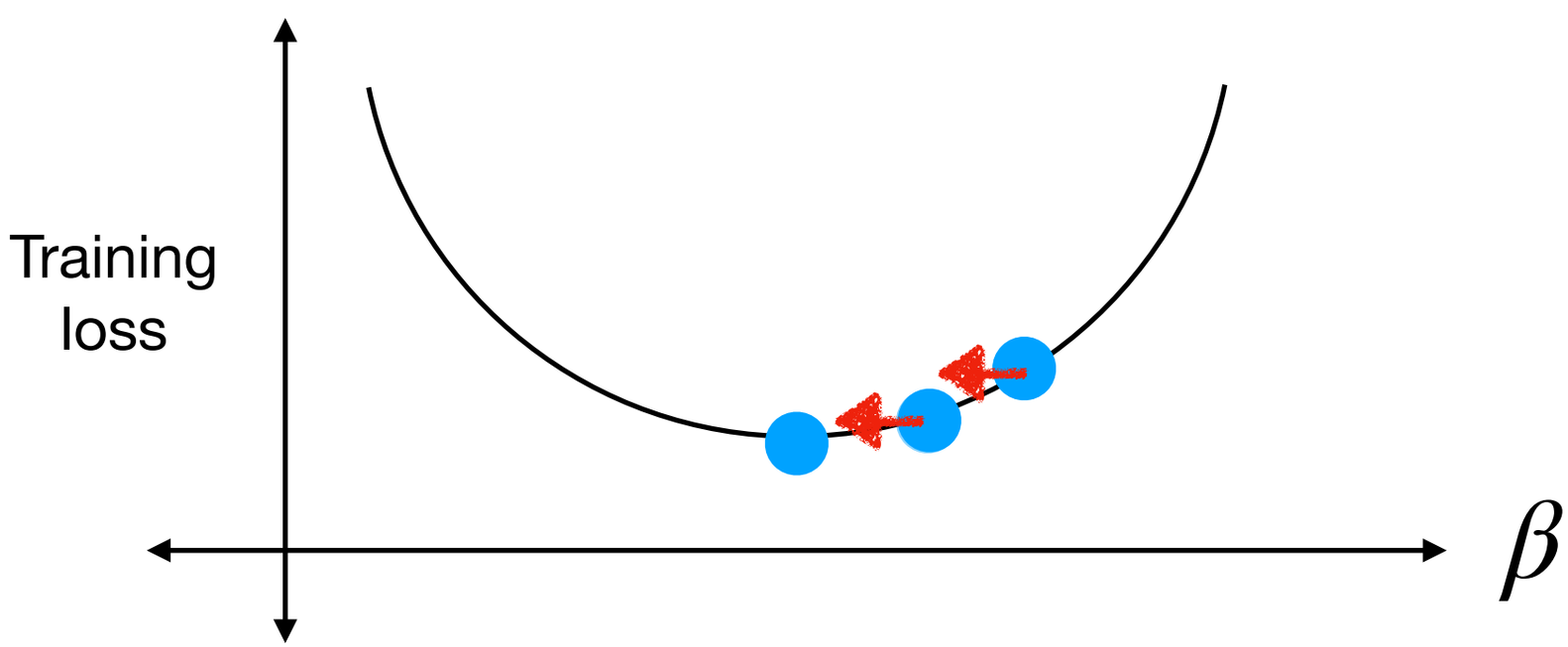
- Boosting
  - AdaBoost
  - **Forward Stagewise Additive Modeling**
  - Gradient boosting
- Issues to think about
  - Confounders
  - Fairness

Boosting as an optimization algorithm

- Recall: Many machine learning algorithms can be defined as the solution to an optimization problem of the form:

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, f_\beta(x_i)\right)$$

Loss function



Forward Stagewise Additive Modeling

An additive model has the form $f(x) = \sum_{b=1}^{B} \alpha_b h_b(x)$.

- For b = 1,2,…B:
  - Find a model h_b and weight α_b that minimize the training error

$$\frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, \underbrace{\sum_{j=1}^{b-1} \alpha_j h_j(x)}_{\text{Current ensemble}} + \underbrace{\alpha_b h_b(x)}_{\text{New weight \& model}}\right)$$

Solve a sequence of "mini" optimization problems to solve the "meta" optimization problem

- Final model: $f(x) = \sum_{b=1}^{B} \alpha_b h_b(x)$.

Different definitions of the loss ℓ will lead to different algorithms…

Forward Stagewise Additive Modeling

- Initialize the training data.
- For b = 1,2,…B:
  - Find a model h_b and weight α_b that minimize the training error

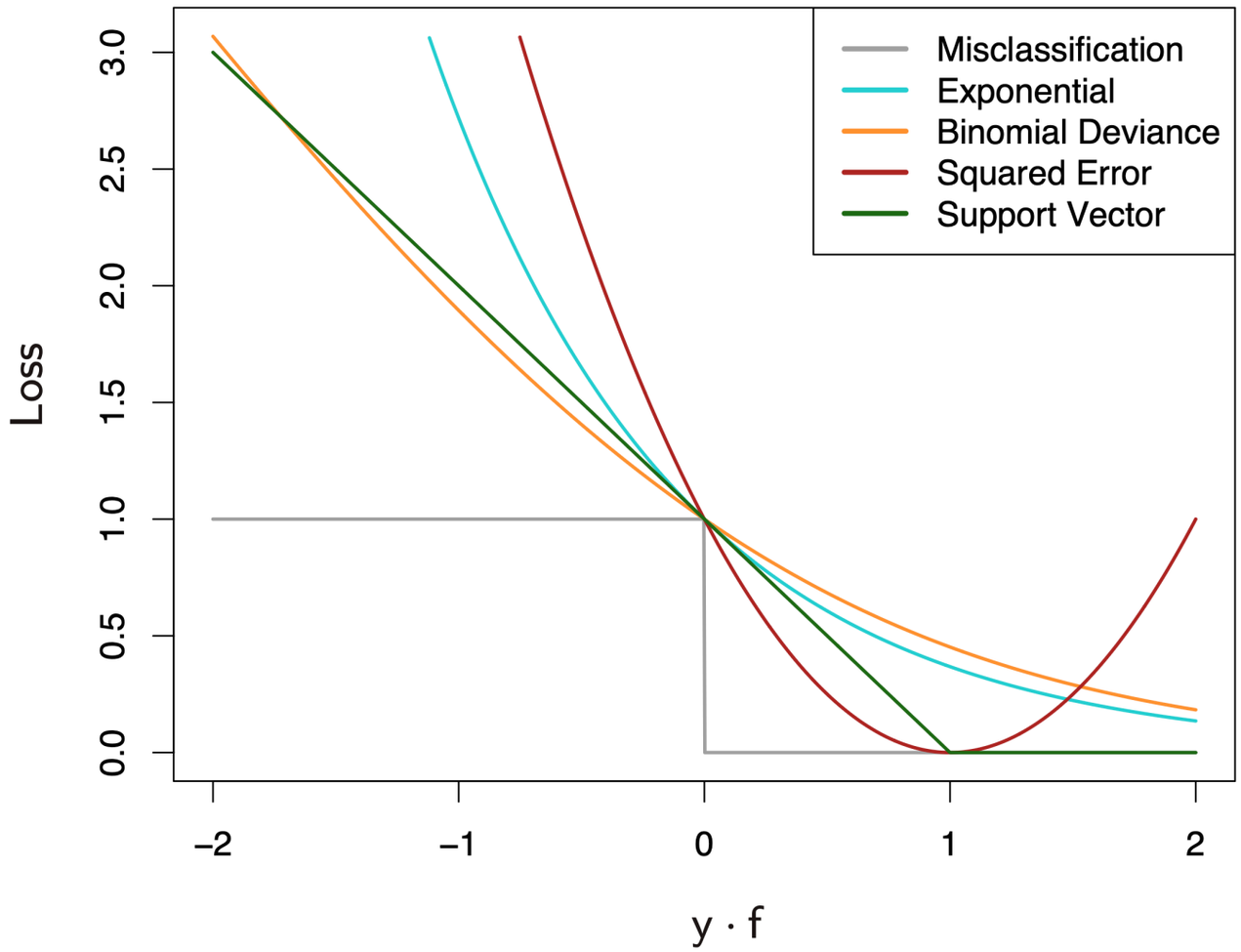
$$\frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, \sum_{j=1}^{b-1} \alpha_j h_j(x) + \alpha_b h_b(x)\right)$$

- Final model: $f(x) = \sum_{b=1}^{B} \alpha_b h_b(x)$.

Special case: AdaBoost

$$\ell(Y, f(X)) = \exp(-Yf(X)) \text{ for } Y \in \{-1, 1\}$$

AdaBoost minimizes the exponential loss.



Forward Stagewise Additive Modeling

- Initialize the training data.
- For b = 1,2,…B:
  - Find a model h_b and weight α_b that minimize the training error

$$\frac{1}{n}\sum_{i=1}^{n}\ell\left(y_i, \sum_{j=1}^{b-1}\alpha_j h_j(x) + \alpha_b h_b(x)\right)$$

- Final model: $f(x) = \sum_{b=1}^{B}\alpha_b h_b(x)$.

$\approx$

Special case: AdaBoost

$\ell(Y, f(X)) = \exp(-Yf(X))$ for $Y \in \{-1,1\}$

- Start with weighting all observations equally: $w_i = 1/n$ for $i = 1,2,...,n$.
- For b = 1,2,…B:
  - Fit a new classifier h_b for dataset with weights w_i.
  - Compute the weighted error of this classifier: $\text{err} = \frac{\sum_{i=1}^{n} w_i 1\{Y_i \neq h_b(X_i)\}}{\sum_{i=1}^{n} w_i}$
  - Set $\alpha_m = \log\left(\frac{1-\text{err}}{\text{err}}\right)$.
  - Update weights $w_i \leftarrow w_i \exp\left(\alpha_m 1\{Y_i \neq h_b(X_i)\}\right)$.
- Final model: $f(x) = sign\left(\sum_{b=1}^{B}\alpha_b h_b(x)\right)$.

Forward Stagewise Additive Modeling

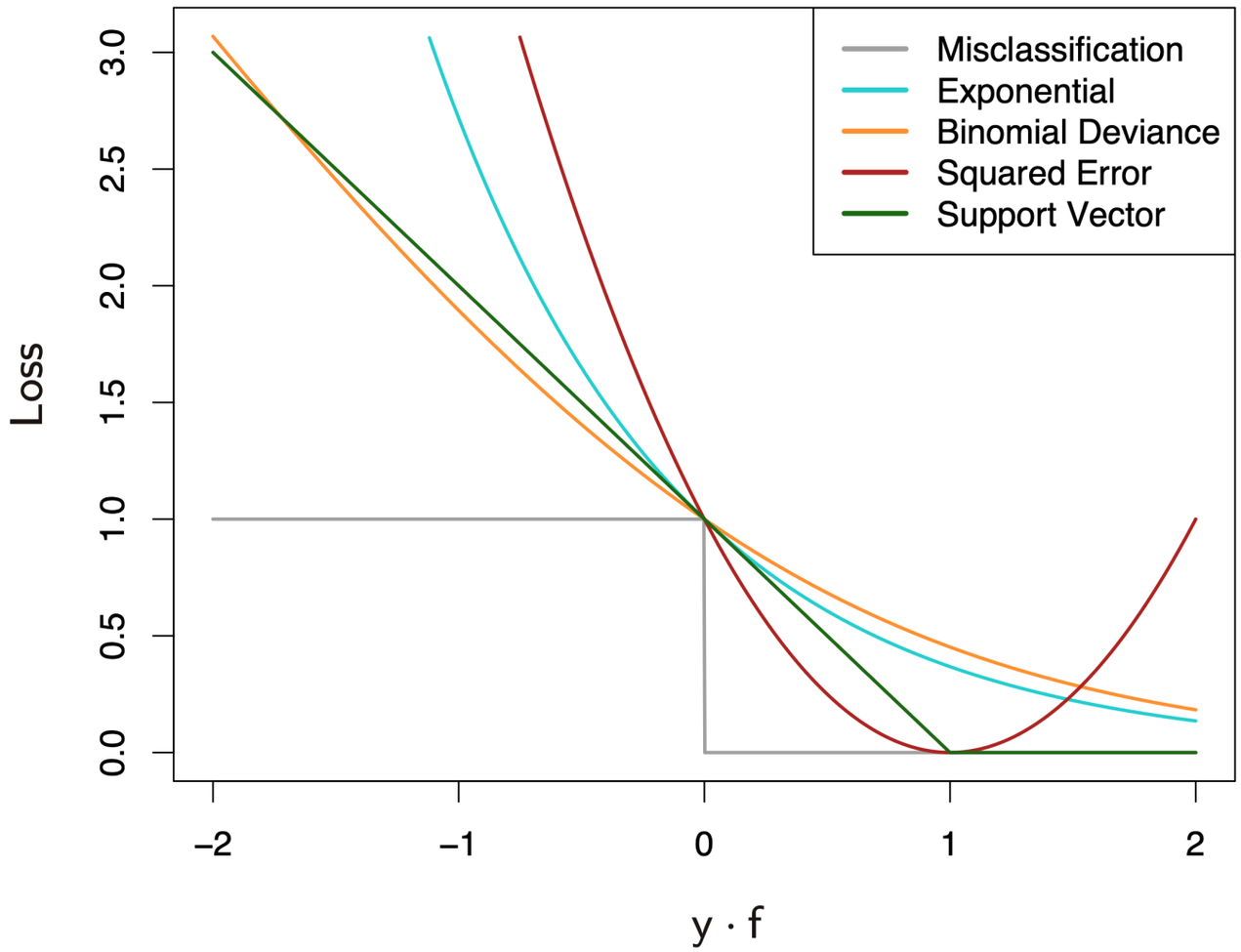
- Initialize the training data.
- For b = 1,2,…B:
  - Find a model h_b and weight α_b that minimize the training error

$$\frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, \sum_{j=1}^{b-1} \alpha_j h_j(x) + \alpha_b h_b(x)\right)$$

- Final model: $f(x) = \sum_{b=1}^{B} \alpha_b h_b(x)$.

Special case: Mean Squared Error

$$\ell(Y, f(X)) = (Y - f(X))^2$$



Forward Stagewise Additive Modeling

- Initialize the training data.
- For b = 1,2,…B:
  - Find a model h_b and weight α_b that minimize the training error $\frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, \sum_{j=1}^{b-1} \alpha_j h_j(x) + \alpha_b h_b(x)\right)$
- Final model: $f(x) = \sum_{b=1}^{B} \alpha_b h_b(x).$

Special case: Mean Squared Error

$$\ell(Y, f(X)) = (Y - f(X))^2$$

- Initialize the training data
- For b = 1,2,…B:
  - Construct a new dataset where we define a new outcome $\tilde{Y}_i$ to be the residual $Y_i - \sum_{j=1}^{b-1} \alpha_j h_j(X_i)$.
  - Fit the best model h_b and coefficient α_b to minimize the MSE on the newly constructed dataset.
- Final model: $f(x) = \sum_{b=1}^{B} \alpha_b h_b(x).$

Forward Stagewise Additive Modeling

- Initialize the training data.
- For b = 1,2,…B:
  - Find a model h_b and weight α_b that minimize the training error

$$\frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, \sum_{j=1}^{b-1} \alpha_j h_j(x) + \alpha_b h_b(x)\right)$$

- Final model: $f(x) = \sum_{b=1}^{B} \alpha_b h_b(x)$.

A general loss function

$$\ell(Y, f(X))$$

Minimizing this doesn't always boil down to reweighing the data or constructing a new dataset…

A more general approach is to take a ***gradient-based view***

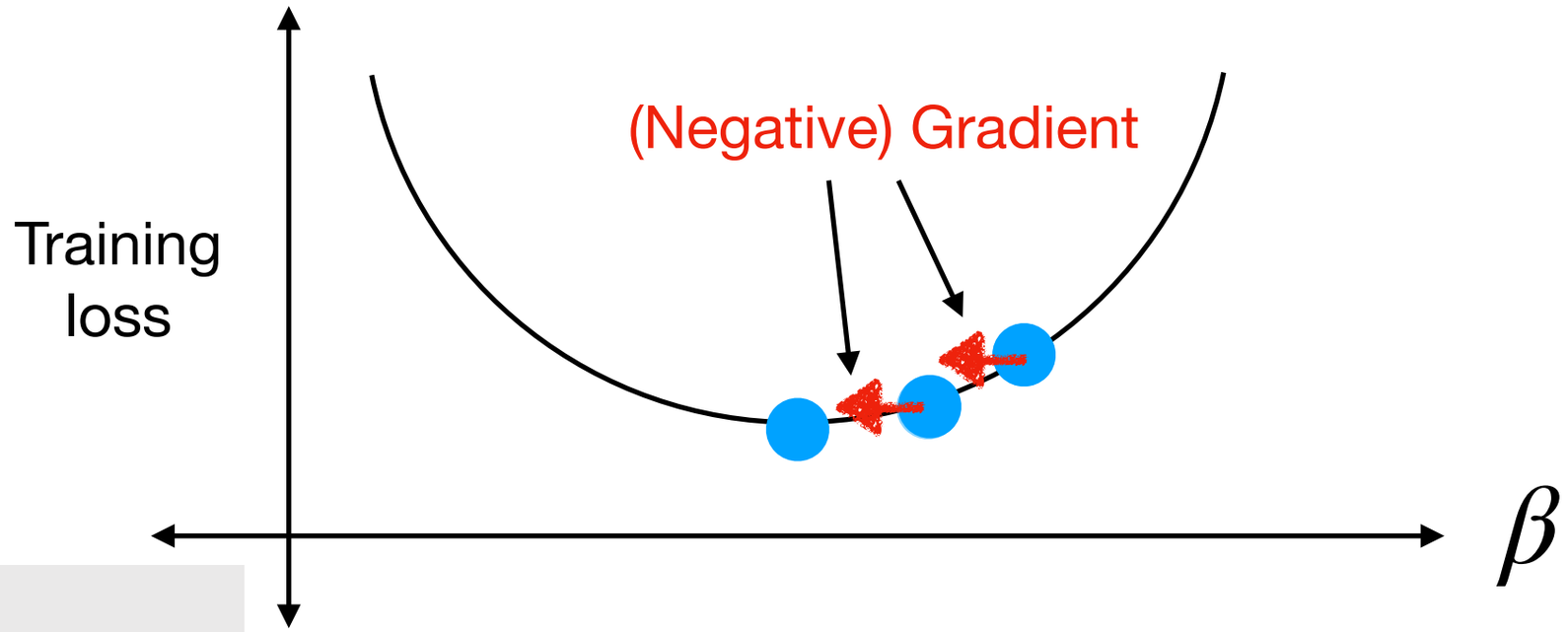
Today's Lecture

- Boosting
  - AdaBoost
  - Foreward Stagewise Additive Modeling
- **Gradient boosting**
- Issues to think about
  - Confounders
  - Fairness

A gradient-based approach

- The **gradient** of a function is the direction that increases the function's value the most. The **negative gradient** is the direction of "**steepest descent**" of the function's value.

- To minimize the training error, many machine learning algorithms compute the gradient and follow the direction of steepest descent.



Gradient update:

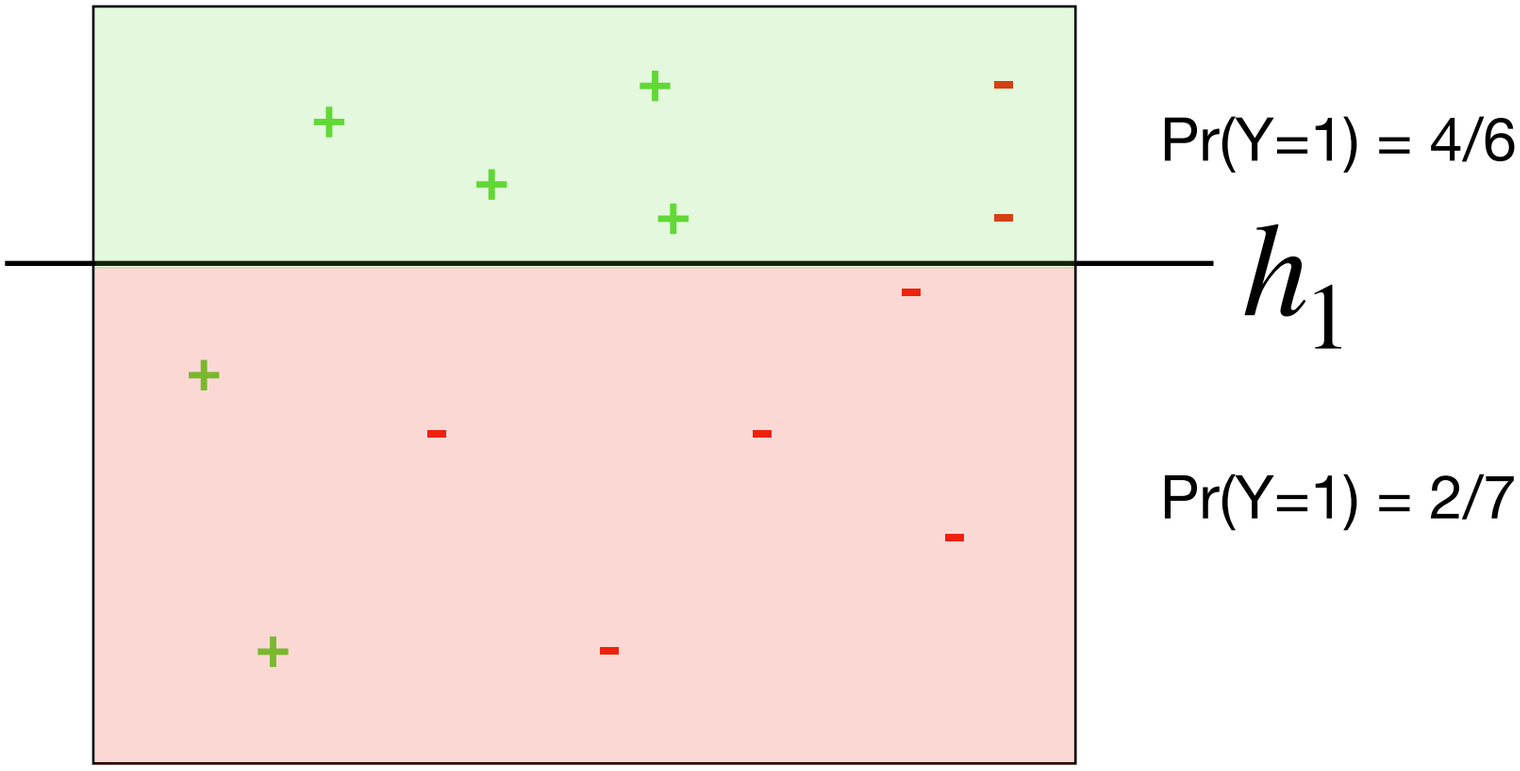
$$\beta \leftarrow \beta - \lambda \underbrace{\nabla_\beta \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f_\beta(x_i))}_{\text{Gradient}}$$

Step size

The step size λ determines how fast we update the model based on the gradient.

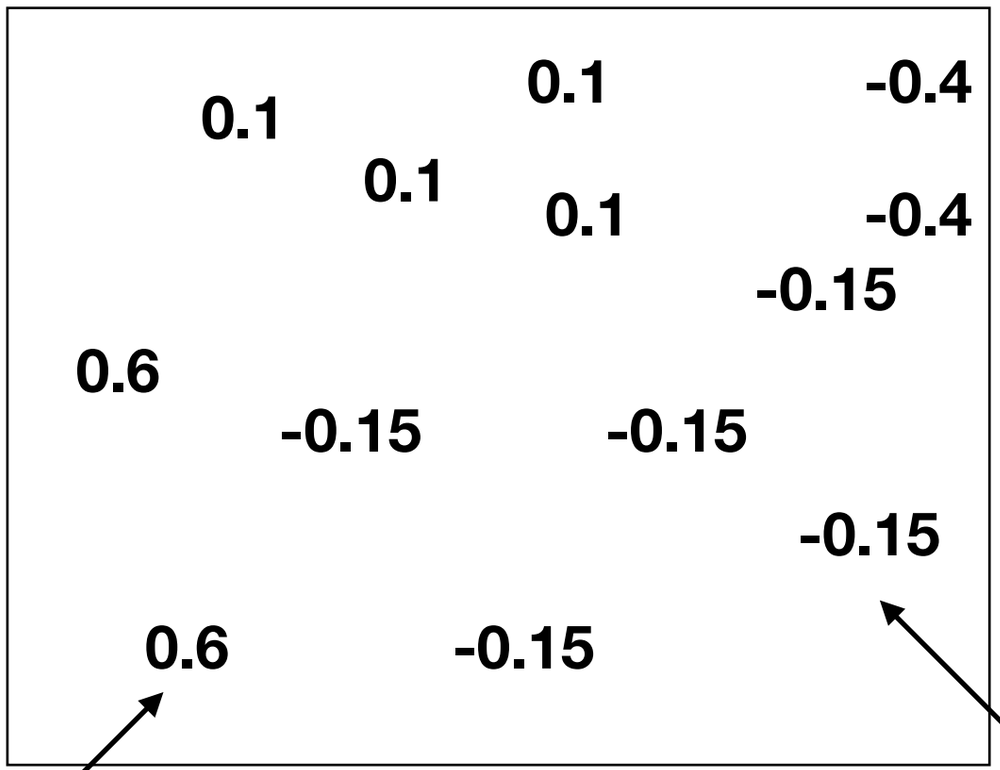
Gradient-boosting: the basic idea

- **Step 1**: Fit a “weak learner” (Here we use probability decision stumps)



Gradient-boosting: the basic idea

- **Step 2:** Compute the negative gradient $-\frac{\partial \frac{1}{n}\sum_{i=1}^{n}\ell(y_i, f(x_i))}{\partial f(x_i)}$ at observations $i = 1,2,...,n$ to determine how we can decrease the training error.

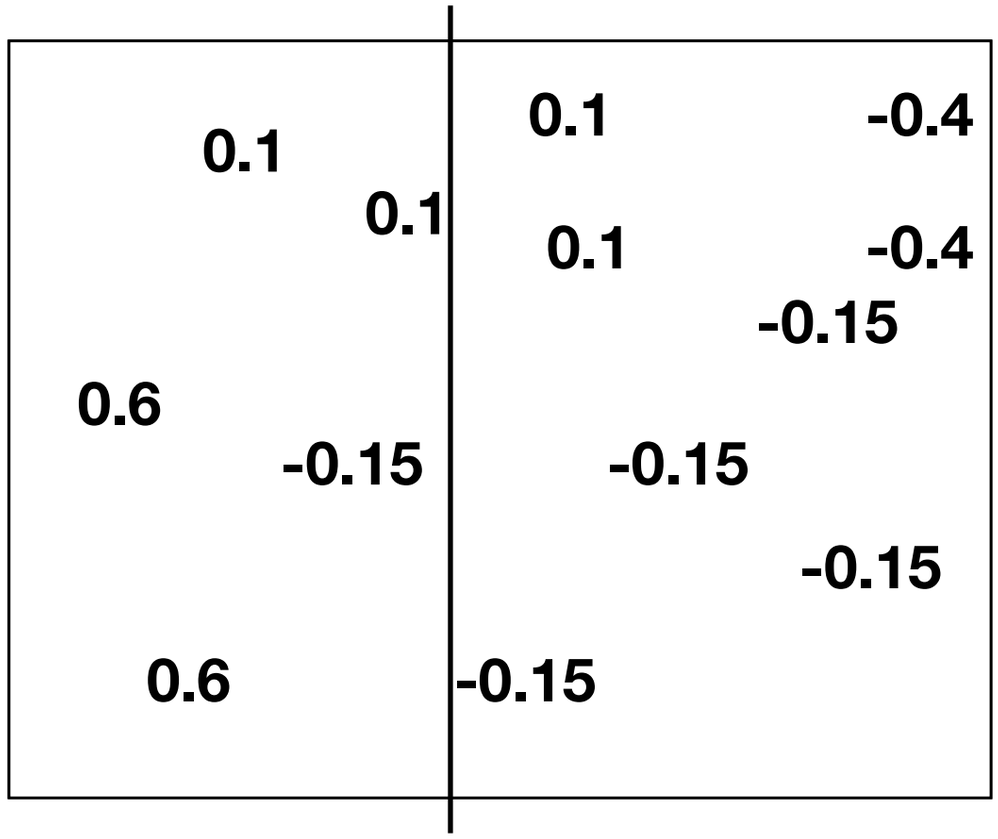


Example: We should increase the probability assigned to class 1 for this label by a lot.

Example: We should decrease the probability assigned to class 1 for this label by a little.

Gradient-boosting: the basic idea

- **Step 2B:** Fit a decision stump that best mimics the values of the gradient.



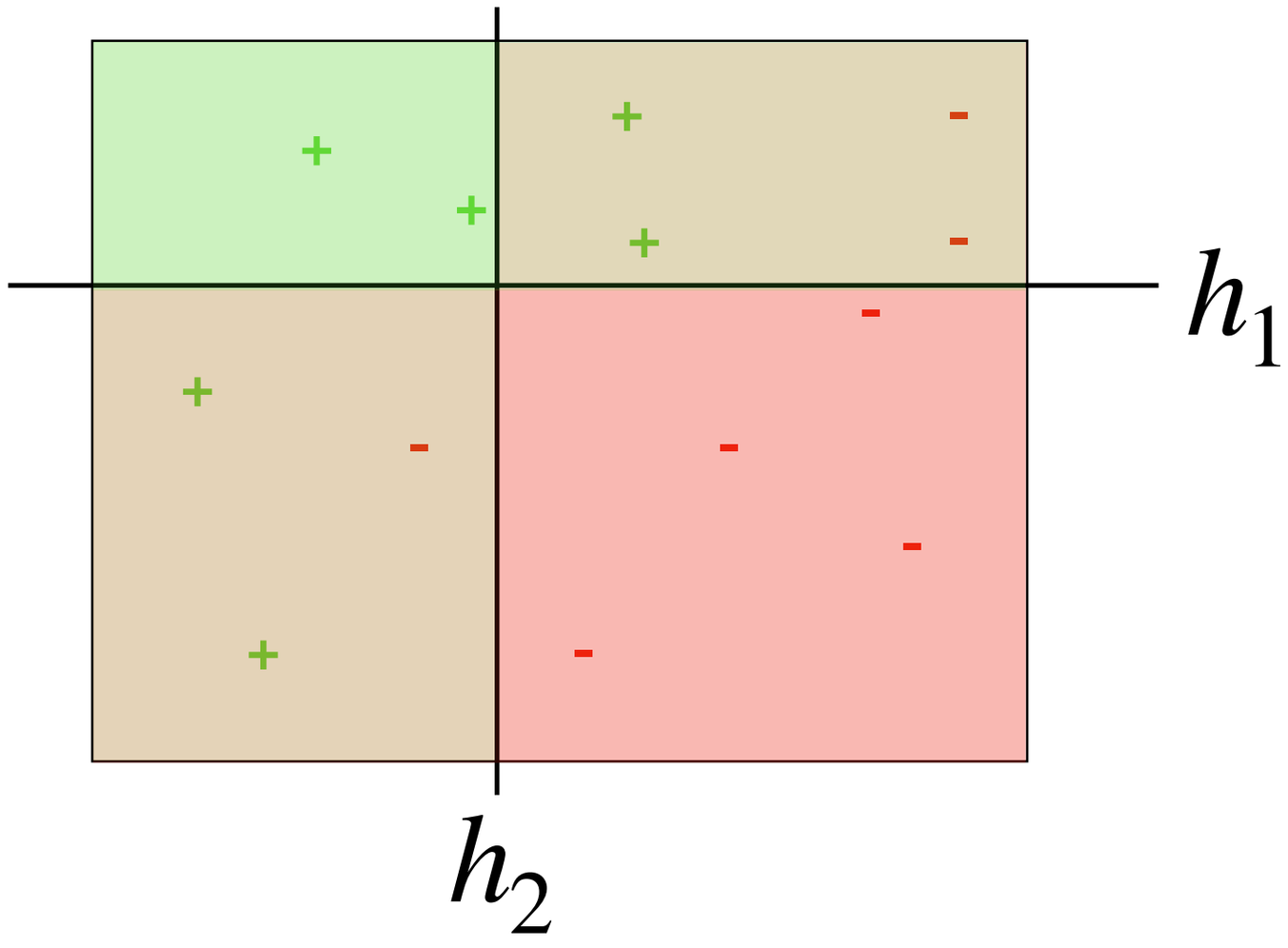
$h_2(x)$=Average value on the left side
(Increase the probability for class 1)

h_2

$h_2(x)$=Average value on the right side
(Decrease the probability for class 1)

Gradient-boosting: the basic idea

- **Step 3**: Combine the two models



$$f(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x)$$

Weights

Gradient boosting

- At each step, fit a model that best mimics the direction of the gradient.

- For b = 1,2,…B:
  - Calculate the negative gradient of the training loss with respect to the predicted outcomes.
  - Find a model h_b and weight α_b that best mimics the negative gradient.
  - Add this new model to the ensemble for step size λ:

$$f(x) = h_1(x) + \lambda \sum_{j=2}^{b} \alpha_j h_j(x)$$

Gradient boosted + Trees

- **Gradient boosted trees** are very popular off-the-shelf algorithms because of their high predictive accuracy.

- The tuning parameters you need to think about are:

  - **The number of splits *d* in each tree**: Controls the complexity of each tree. In practice, $d = 1$ often works well.

  - **The number of trees *B***: As B increases, the complexity of our boosted model increases.

  - **The shrinkage parameter λ**: λ controls the rate of learning (i.e. the step size), where a small value means that the model update at each iteration is small.

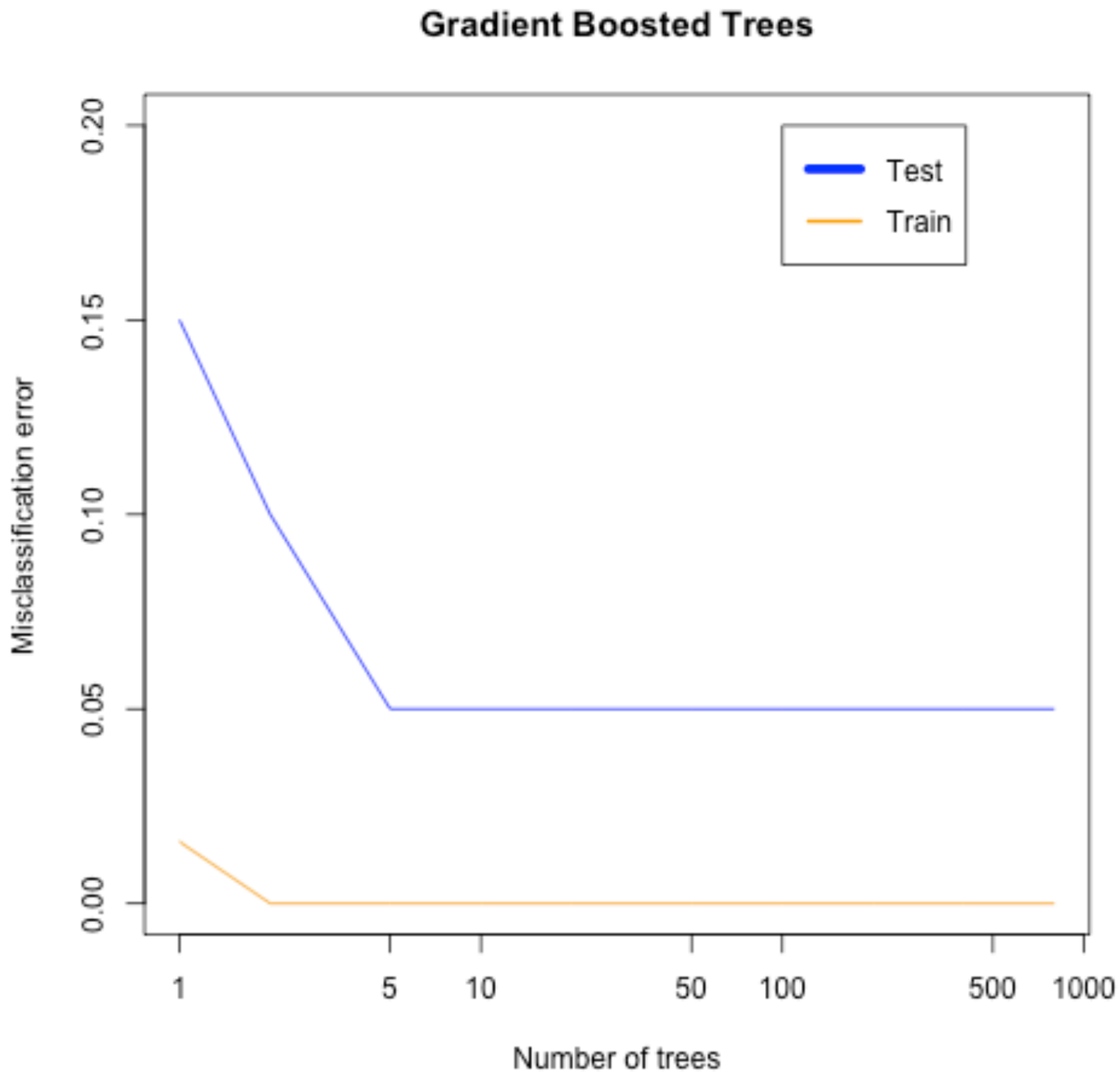
Example: Gene expression data

- Khan gene expression dataset from ISLR:
  - X = Gene expression measurements, p=2308
  - Y = Type of small round blue cell tumors, 4 classes
  - Training dataset size = 63
  - Test dataset size = 20

```
> library(xgboost)
> dtrain <- xgb.DMatrix(Khan$xtrain, label = Khan$ytrain)
> bst <- xgboost(
+   data=dtrain,
+   nrounds = 50,
+   max_depth=2,
+   num_class=num_class,
+   objective="multi:softprob")
```

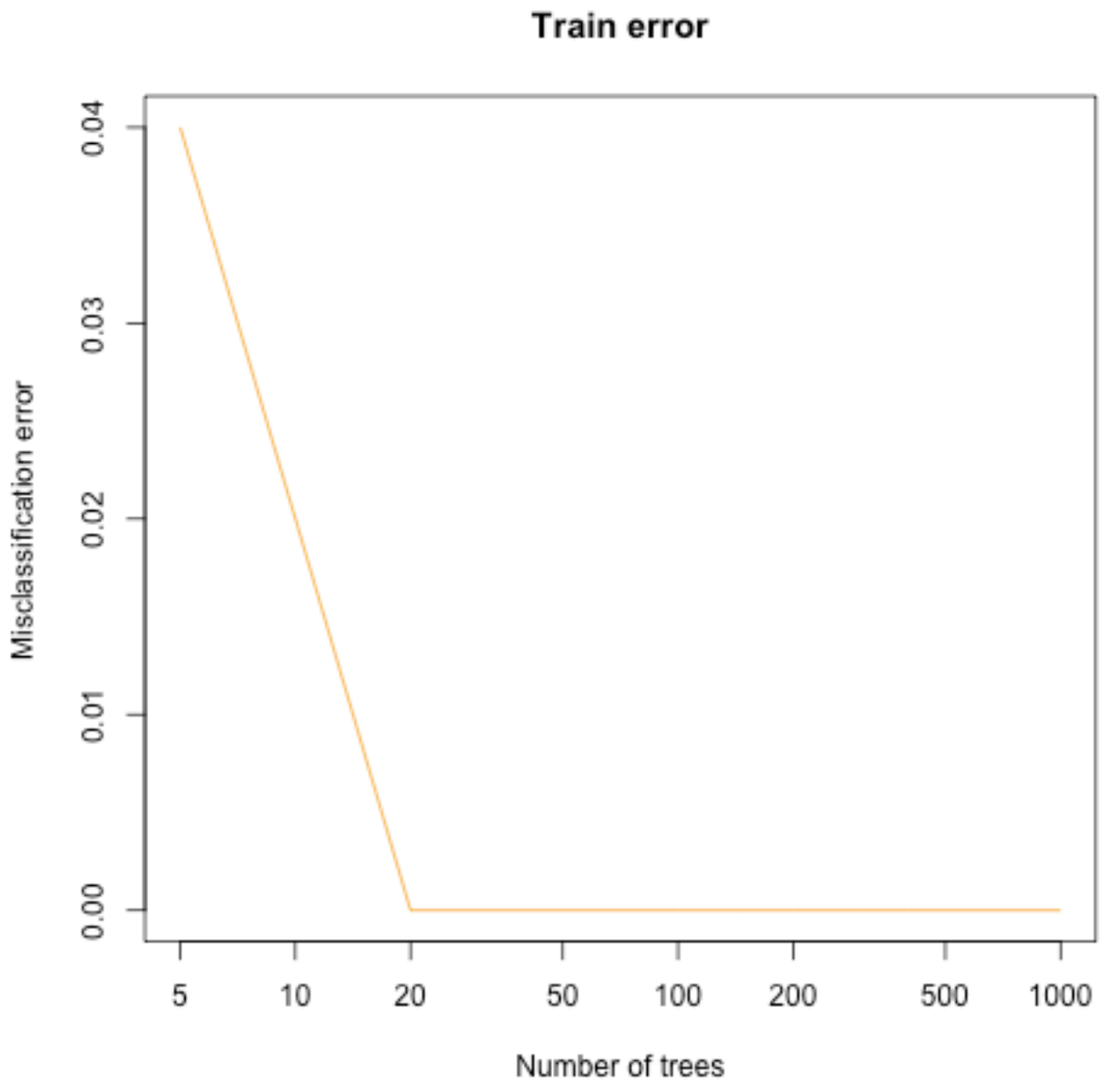
Example: Gene expression data

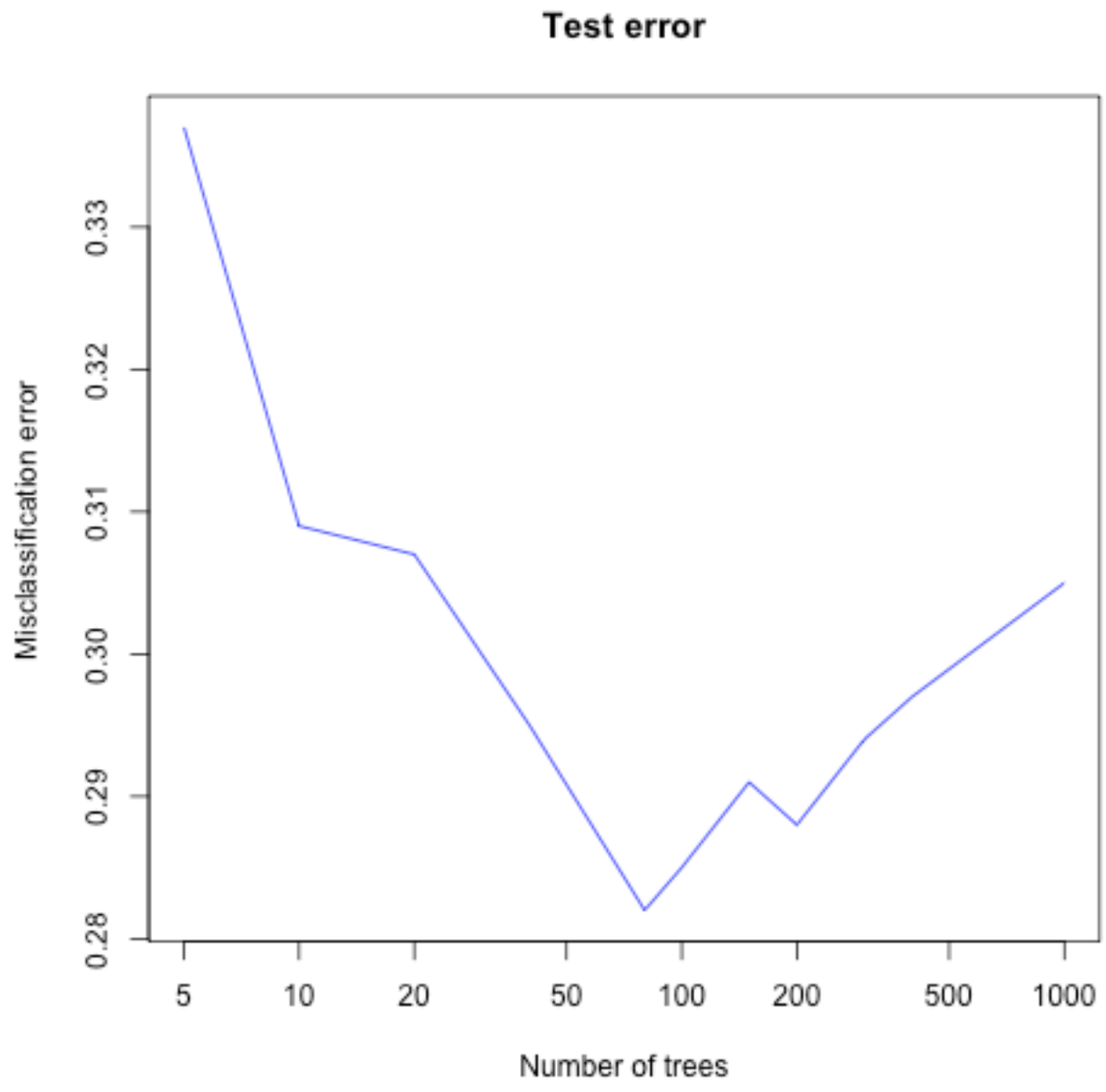
- Let's vary the number of trees *B*.



Example: Noisy binary classification problem

- Simulated data for a noisier binary classification problem.
- We see that boosting overfits in this problem and the number of trees needs to be tuned.



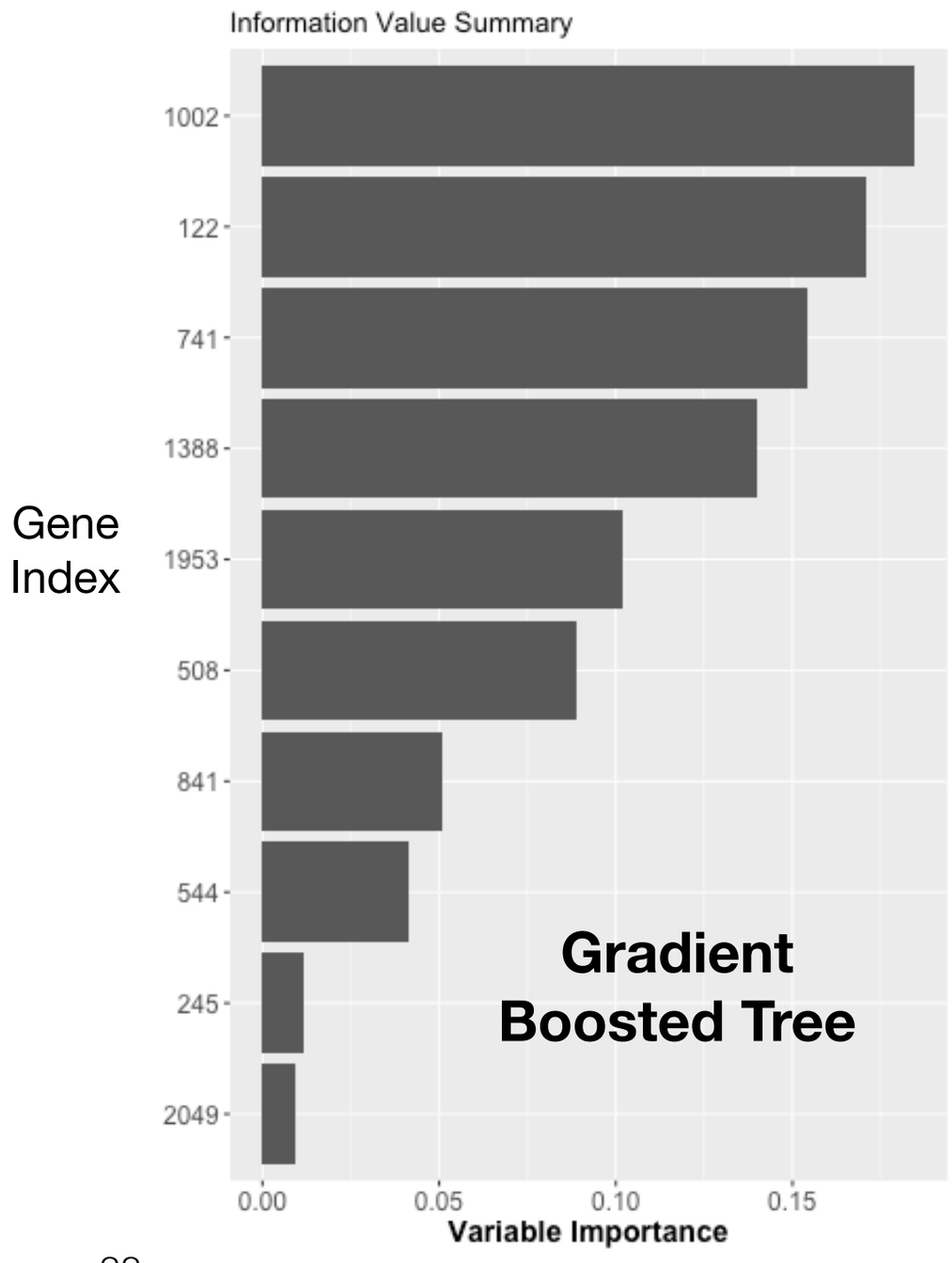


Example: Variable importance

```
> xgb.plot.importance(importance_matrix = head(xgb.importance(model=bst), n = 20))
```

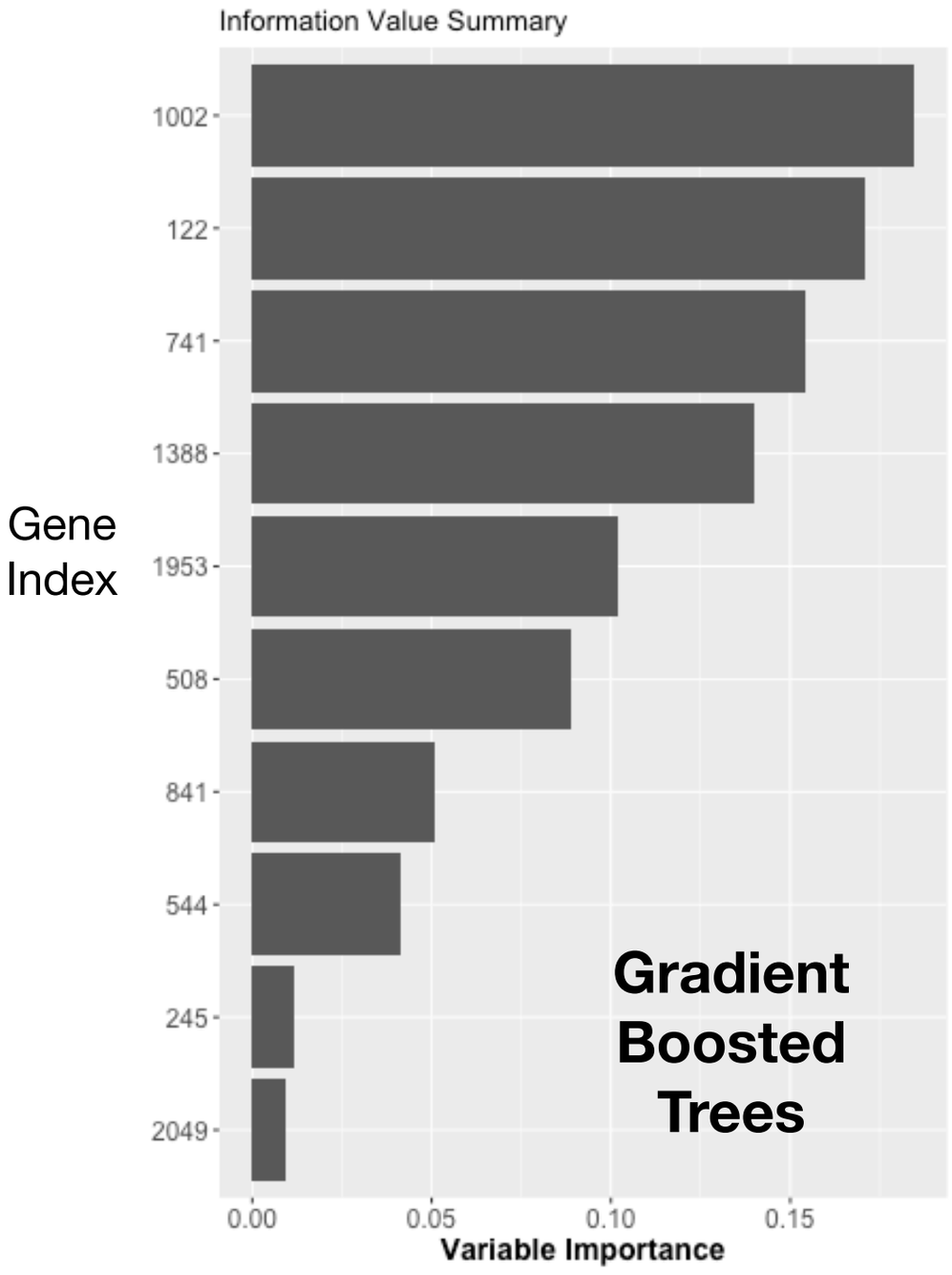
- Let's now try to interpret the gradient boosted tree we fit to the Khan gene expression dataset.

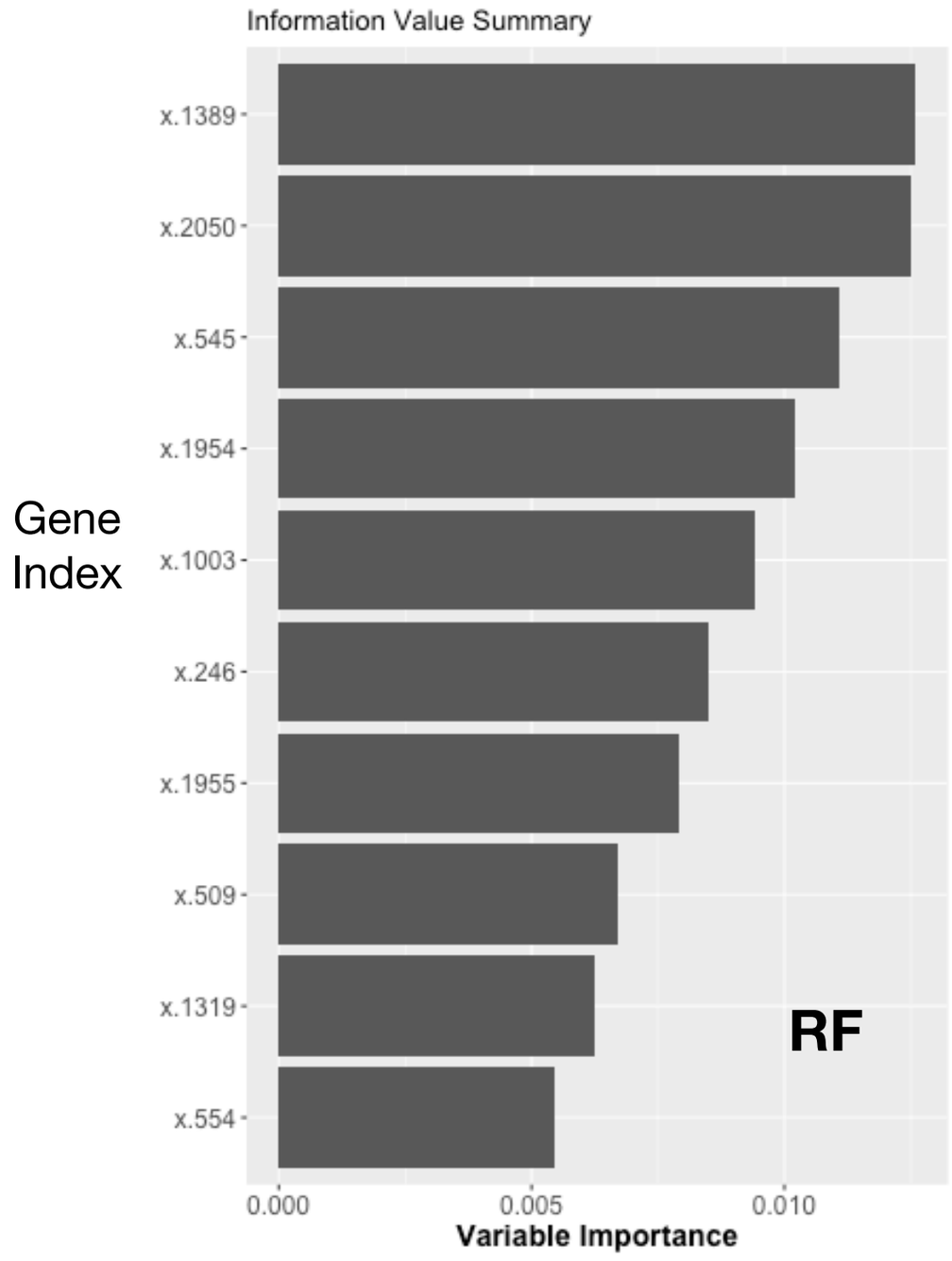
- Here we show the top 10 most important genes according to the variable importance function from xgboost.



Example: Variable importance

- These variable importance estimates are model specific, so be careful when you interpret them!





Variable importance in gradient boosted trees

- Consider this example: Suppose there are two highly predictive variables X_1, X_2 that are also highly correlated. Then…
    - Gradient boosted trees will just choose one at random and ignore the other variable.
    - Random forests will choose both of them with similar probabilities.

Today's Lecture

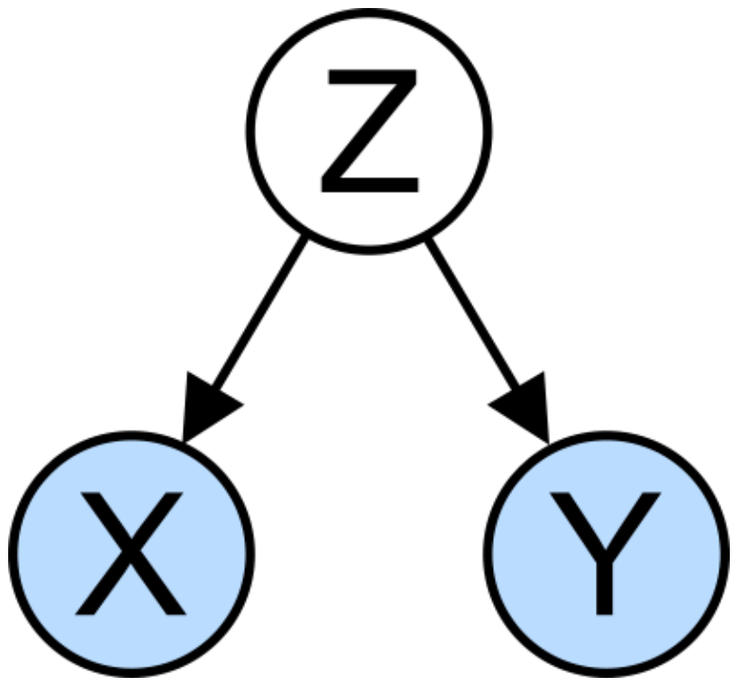
- Boosting
  - AdaBoost
  - Forward Stagewise Additive Modeling
  - Gradient boosting
- **Issues to think about**
  - Confounders
  - Fairness

Confounders and spurious associations

- A confounding variable **Z** is associated with both an explanatory variable **X** and the outcome of interest **Y**.

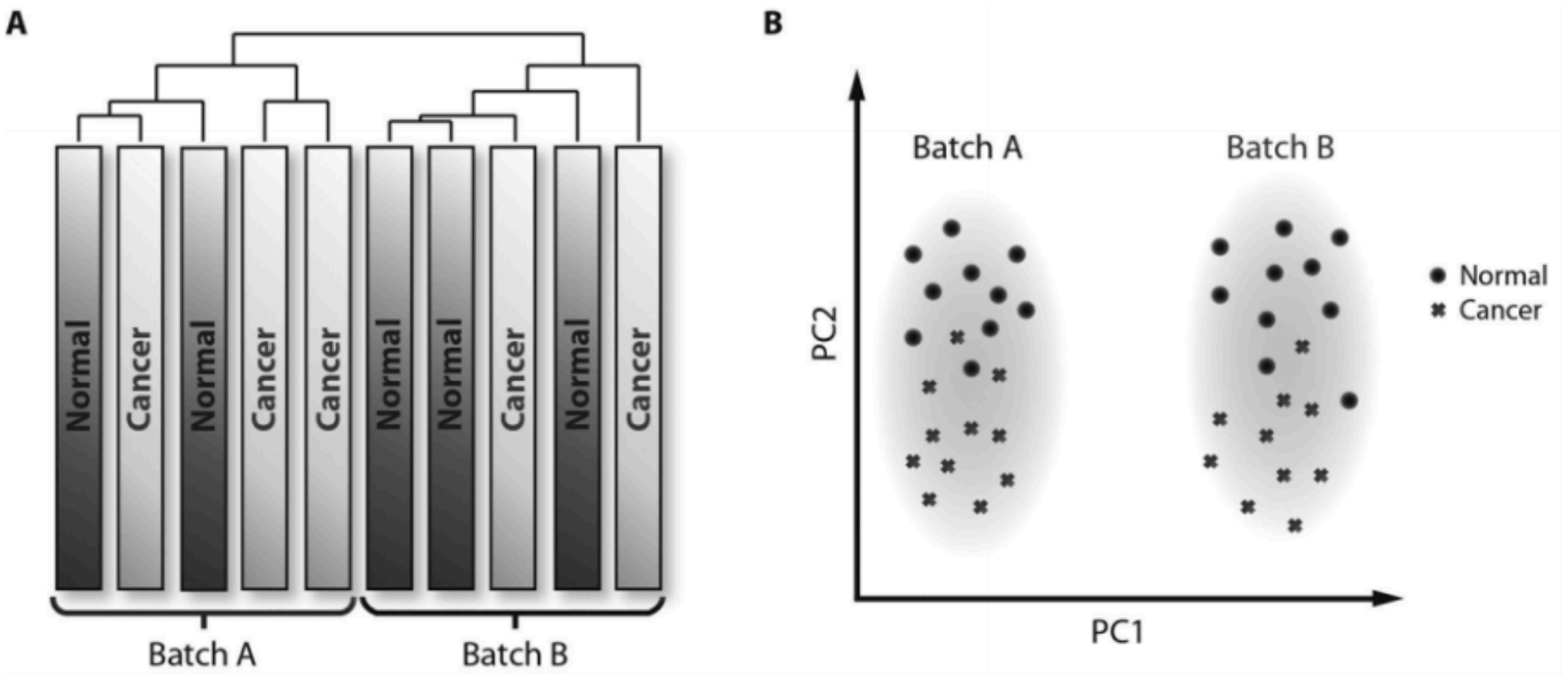
- Machine learning algorithms can rely on confounders to achieve high prediction accuracy for a given dataset. In fact, the prediction accuracies can often be overly optimistic.

- We must carefully consider possible confounding in our datasets. Overlooking these can lead to models that fail to generalize in other settings.



An example from omics

- In omics experiments, the measured variables often have a strong association with non-biological factors (e.g. inter-machine or inter-lab or inter-operator variability, or time of day), which are often referred to as “batch effects.”



An example from omics

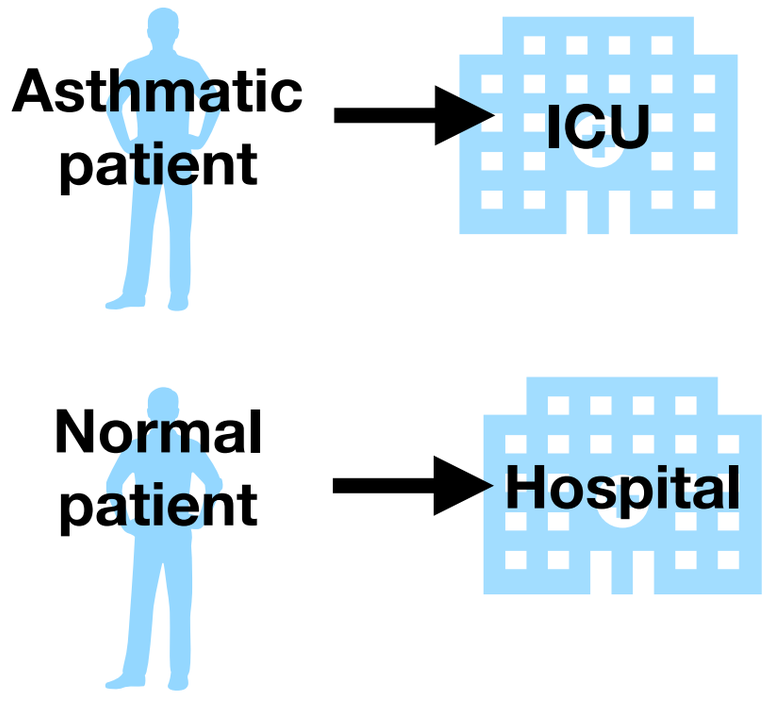
- Petricoin et. al. (2002) reported that their predictive model could use proteomic patterns to discriminate between healthy patients and those with ovarian cancer with 100% sensitivity and 95% specificity!

- However, when independent researchers looked at the publicly released data, they found:

- inadvertent changes in protocol mid-experiment
- difference in processing between tumor and normal samples.

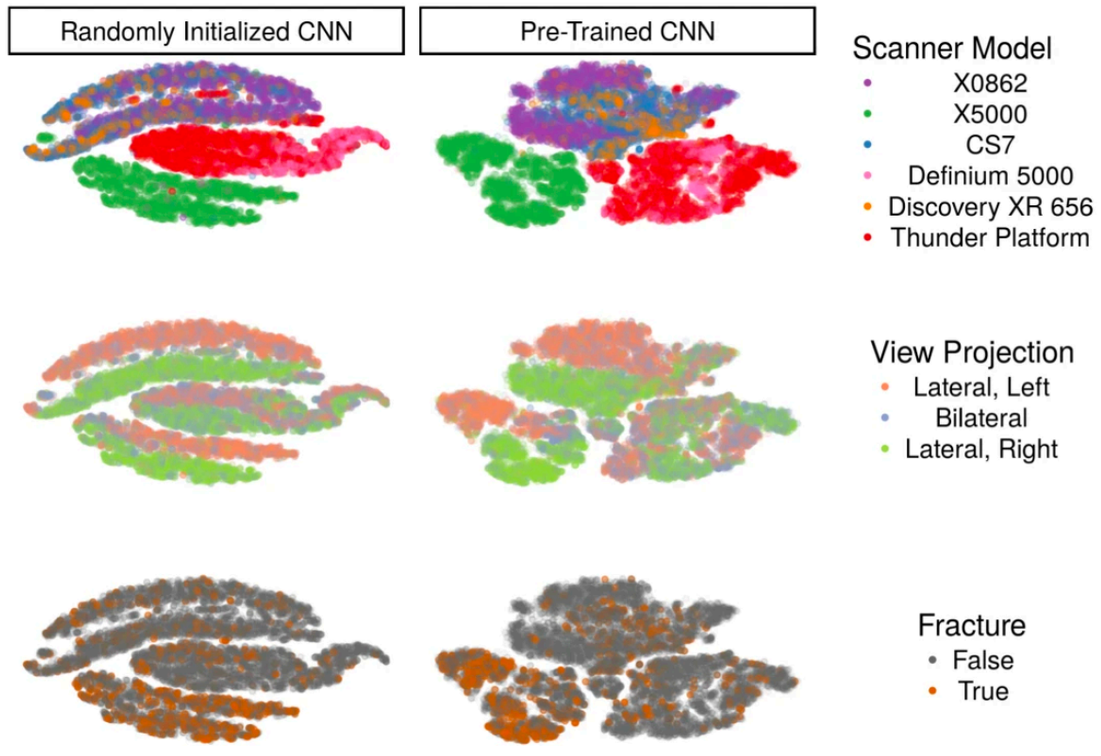
Examples from EHR data

Caruana 2015: ML predicted
that asthmatics are less
likely to die from pneumonia



Examples from imaging

Badgeley 2019: ML made predictions by
detecting the scanner model from the radiograph



Confounders and distributional shifts

- Machine learning algorithms simply try to maximize prediction accuracy for the setting from which the data was collected. If future data is going to be collected in exactly the same manner, the prediction accuracy of your ML algorithm will remain stable.

- However, this is often not the case. We often want our model to be usable by other hospitals and labs.

- ML algorithms cannot determine which associations will generalize to other settings and which ones don’t (e.g. confounders and spurious associations). In the absence of additional guidance, ML algorithms will learn to depend on whichever associations are the strongest.

Improving data collection

- Good experimental design is the best strategy for reducing spurious associations in the data and avoiding faulty conclusions.

    - Randomize sample run times: e.g. don’t run cases first and controls second.

    - Document how the data was collected (date, machine, cell culture medium, ...)

Improving the analysis

- If you know what confounders commonly exist in your data setting, apply methods that adjust for this known source of confounding.

- Train on data from different institutions, rather than using a single data set. Validate results on independent data sets from a different institution.

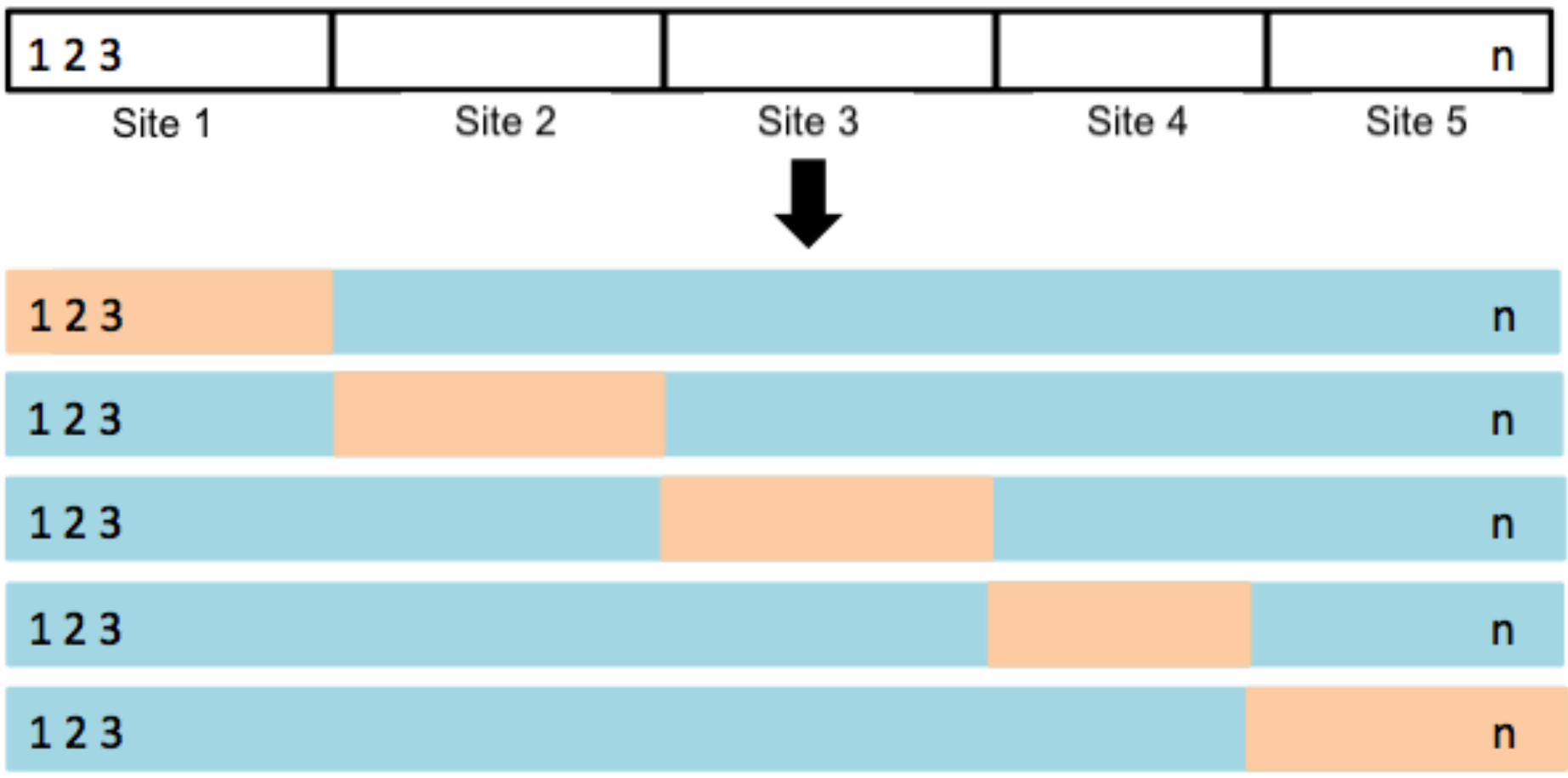
- Use an interpretable model and/or understand how the model came to its final conclusion. Check that the model explanations make sense (and fix them if they don’t).

- Build in causal reasoning to your prediction model.

- Make sure your data analysis is reproducible. Clearly document your code that contains every step, from data pre-processing to the final analysis.

Cross-validation when combining datasets

- When training models on datasets across different institutions, remember to do site-wise cross-validation!



Today's Lecture

- Boosting
  - AdaBoost
  - Forward Stagewise Additive Modeling
  - Gradient boosting
- Issues to think about
  - Confounders
  - **Fairness**

ML Fairness

RESEARCH ARTICLE

ECONOMICS

Dissecting racial bias in an algorithm used to manage the health of populations

Ziad Obermeyer[1,2]*, Brian Powers[3], Christine Vogeli[4], Sendhil Mullainathan[5]*†

"We show that a widely used algorithm … exhibits significant racial bias: At a given risk score, Black patients are considerably sicker than White patients, as evidenced by signs of uncontrolled illnesses… The bias arises because the algorithm predicts health care costs rather than illness, but unequal access to care means that we spend less money caring for Black patients than for White patients. Thus, despite health care cost appearing to be an effective proxy for health by some measures of predictive accuracy, large racial biases arise. We suggest that the choice of convenient, seemingly effective proxies for ground truth can be an important source of algorithmic bias in many contexts."

ML fairness

Google Photos Tags Two African-Americans As Gorillas Through Facial Recognition Software

The Best Algorithms Struggle to Recognize Black Faces Equally

US government tests find even top-performing facial recognition systems misidentify blacks at rates five to 10 times higher than they do whites.

- Another well-known example is the Google photo-tagging incident in 2015.
- “Why facial recognition systems perform differently for darker skin tones is unclear. Buolamwini told Congress that many datasets used by companies to test or train facial analysis systems are not properly representative. The easiest place to gather huge collections of faces is from the web, where content skews white, male, and western. Three face-image collections most widely cited in academic studies are 81 percent or more people with lighter skin, according to an IBM review."

Common causes of unfairness

- **Bias Encoded in Data**: Often the training data that is available encodes human biases.

- **Minimizing Average Error Fits Majority Populations**: The relationships between the variables and outcomes can differ across subpopulations. Unfortunately, most ML algorithms aim to minimize the average error, which may be done at the expense of minority groups.

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \ell \left(y_i, f_\beta(x_i) \right)$$

Many ML algorithms aim to minimize the average loss, which down-weights errors made on minority groups.

Chouldechova and Roth. "The Frontiers of Fairness in Machine Learning" 2018.

ML fairness is still a young field

- There are *many* proposals for how to quantify the (un)fairness of an ML algorithms. In general, it is impossible for all of these fairness criteria to hold simultaneously. (Chouldechova 2017)

- There are one-time fairness questions and then there are questions about the dynamics induced by ML algorithms. For example, it is possible for algorithms that were initially fair to become unfair.

- There is also lot of research on how to correct for biases in datasets, which typically requires understanding how the data was generated in the first place.

Chouldechova and Roth. “The Frontiers of Fairness in Machine Learning” 2018.

Today's Lecture

- Boosting
  - AdaBoost
  - Forward Stagewise Additive Modeling
  - Gradient boosting
- Issues to think about
  - Confounders
  - Fairness

Final project