DS-GA 1003: Machine Learning

Lecture 8: Boosting & “Classical ML” Review

Slides adapted from material from David Rosenberg.

Logistics & Announcements

PS 3 due today @ 11:59pm ET. Reminder that you have late days (and a free drop)!

Project proposals due today @ 11:59pm ET. Don't worry about the grading here, just focus on writing a clear proposal.

PS 4 released today. Shorter problem set, mostly coding.

Sam's last lecture today. Will be handing it over to Nick after today's lecture.

Thanks for your attention this semester!

Outline

Boosting Basics

Forward Stagewise Additive Modeling (FSAM)

Example: L^2 Boosting (Regression)

Example: AdaBoost (Classification)

Gradient Boosting

Boosting vs. Bagging

Ensemble Methods

Bagging. Reduce the variance of a **low bias, high variance** estimator by ensembling many hypotheses trained in parallel (on different datasets obtained through sampling).

$$\hat{h}_{avg}(x) = \text{combine}(\hat{h}_1(x), \ldots, \hat{h}_B(x))$$

Boosting. Reduce the risk of a **high bias** estimator by ensembling many estimators trained in sequence (without bootstrapping).

Like bagging, boosting is a general method that is popular with decision trees.

$$\hat{f}_{boost}(x) = \sum_{t=1}^{T} \nu_t \hat{h}_t(x).$$

Nonlinear Regression

Example

How should we fit this data?

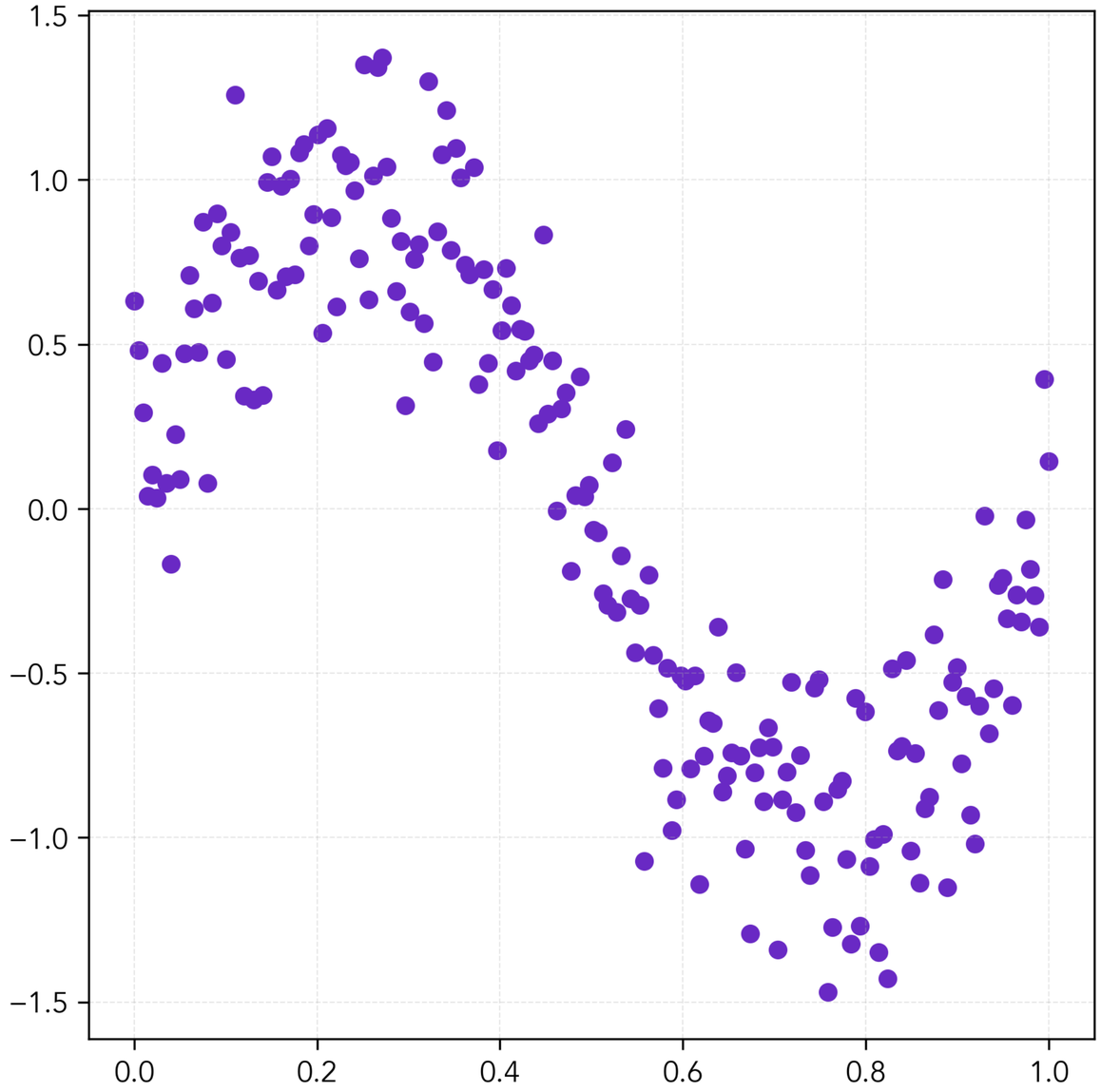
Polynomial regression.

Kernel regression.

Decision trees.

Another way to get non-linear models:

Adaptive basis function models.



Linear Model of Basis Functions

Hypothesis Class for Boosting

Goal. Fit a linear combination of transformations of the input:

$$f(x) = \sum_{t=1}^{T} \nu_t h_t(x), \text{ where } h_t\text{'s are called basis functions.}$$

$$h_1, \ldots, h_T \colon \mathcal{X} \to \mathbb{R}$$

Example. Polynomial regression where $h_t(x) = x^t$.

Can we use this model for classification?

If h_t's are fixed and known: can fit using standard methods for linear models.

Linear Model of Basis Functions

Hypothesis Class and Adaptive Basis

$$f(x) = \sum_{t=1}^{T} \nu_t h_t(x), \text{ where } h_t\text{'s are called basis functions.}$$

What if we *learn* the basis functions (hence, *adaptive*)?

Base hypothesis space $\mathscr{H}$ consisting of functions $h\colon \mathscr{X} \to \mathbb{R}$.

An adaptive basis function expansion over $\mathscr{H}$ is this ensemble where $\nu_t \in \mathbb{R}$ and $h_t \in \mathscr{H}$.

Combined hypothesis space:

$$\mathscr{F}_T := \left\{ \sum_{t=1}^{T} \nu_t h_t(x) : \nu_t \in \mathbb{R}, h_t \in \mathscr{H}, t = 1, \ldots, T \right\}$$

Using Adaptive Basis Model

Empirical Risk Minimization

$$\mathcal{F}_T := \left\{ \sum_{t=1}^{T} \nu_t h_t(x) : \nu_t \in \mathbb{R}, h_t \in \mathcal{H}, t = 1,\ldots,T \right\}$$

What's our learning objective?

$$\hat{f} \in \arg\min_{f \in \mathcal{F}_T} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x^{(i)}), y^{(i)}).$$

The objective function for this hypothesis class is:

$$F(\nu_1, \ldots, \nu_T, h_1, \ldots, h_T) = \frac{1}{n} \sum_{i=1}^{n} \ell \left(\sum_{t=1}^{T} \nu_t h_t(x^{(i)}), y^{(i)} \right)$$

Using Adaptive Basis Model

Optimizing the ERM?

$$F(\nu_1, \ldots, \nu_T, h_1, \ldots, h_T) = \frac{1}{n} \sum_{i=1}^{n} \ell \left(\sum_{t=1}^{T} \nu_t h_t(x^{(i)}), y^{(i)} \right)$$ seems hard to minimize.

Suppose that our hypothesis space is parameterized by $\Theta = \mathbb{R}^d$ (e.g. $w \in \mathbb{R}^d$ in linear models).

$$F(\nu_1, \ldots, \nu_T, \theta_1, \ldots, \theta_T) = \frac{1}{n} \sum_{i=1}^{n} \ell \left(\sum_{t=1}^{T} \nu_t h(x^{(i)}; \theta_t), y^{(i)} \right)$$

Can we optimize this using GD? For some hypothesis spaces and typical losses, yes!

$N(x) = \sum_{t=1}^{T} \nu_t \sigma(w_t^\top x)$ is just a two-layer neural network of width T.

Using Adaptive Basis Model

What if gradient-based methods don't apply?

$$F(\nu_1, \ldots, \nu_T, h_1, \ldots, h_T) = \frac{1}{n} \sum_{i=1}^{n} \ell \left(\sum_{t=1}^{T} \nu_t h_t(x^{(i)}), y^{(i)} \right)$$

What is $\mathscr{H}$ consists of decision trees?

Can't parameterize trees with $\Theta = \mathbb{R}^d$…

Even if we could, predictions do not change continuously with $\theta \in \Theta$, so not differentiable.

Solution: gradient boosting.

Solve the ERM problem as long as we can do regression with $\mathscr{H}$ (regression trees).

Outline

Boosting Basics

Forward Stagewise Additive Modeling (FSAM)

Example: L^2 Boosting (Regression)

Example: AdaBoost (Classification)

Gradient Boosting

Forward Stagewise Additive Modeling (FSAM)

Overview

Goal. Fit model $f(x) = \sum_{t=1}^{T} \nu_t h_t(x)$ given some loss function.

Approach. Greedily fit one function at a time without adjusting previous functions.

After $t-1$ stages, we have:

$$f_{t-1} = \sum_{s=1}^{t-1} \nu_s h_s \text{ (this is fixed).}$$

In the t'th round, we want to find $h_t \in \mathscr{H}$ (**step direction**) and $\nu_t > 0$ (**step size**) such that:

$f_t = f_{t-1} + \nu_t h_t$ that improves the objective.

FSAM with ERM

Algorithm

Let's plug in our objective function (ERM).

1. Initialize $f_0(x) = 0$.

2. For $t = 1$ to T:

   Compute: $(\nu_t, h_t) \in \underset{\nu \in \mathbb{R}, h \in \mathscr{H}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \ell\left(f_{t-1}(x^{(i)}) + \nu h(x^{(i)}), y^{(i)}\right).$

   Set $f_t = f_{t-1} + \nu_t h_t$.

3. Return f_T.

Outline

Boosting Basics

Forward Stagewise Additive Modeling (FSAM)

Example: L^2 Boosting (Regression)

Example: AdaBoost (Classification)

Gradient Boosting

ERM with Squared Loss

Main FSAM Step

$$\text{At step } t \in [T], \text{ find } (\nu_t, h_t) \in \operatorname*{arg\,min}_{\nu \in \mathbb{R}, h \in \mathscr{H}} \frac{1}{n} \sum_{i=1}^{n} \ell \left(f_{t-1}(x^{(i)}) + \nu h(x^{(i)}), y^{(i)} \right).$$

Suppose we care about the squared loss: $\ell(\hat{y}, y) = (\hat{y} - y)^2$. At each step, minimize:

$$J(v, h) = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - \left[f_{t-1}(x^{(i)} + \nu h(x^{(i)}) \right] \right)^2$$

If $\mathscr{H}$ is closed under rescaling, then don't need ν.

$$\implies J(h) = \frac{1}{n} \sum_{i=1}^{n} \left(\left[y^{(i)} - f_{t-1}(x^{(i)}) \right] - h(x^{(i)}) \right)^2$$

ERM with Squared Loss

Main FSAM Step

$$J(h) = \frac{1}{n} \sum_{i=1}^{n} \left(\left[y^{(i)} - f_{t-1}(x^{(i)}) \right] - h(x^{(i)}) \right)^2$$

This is just fitting the residuals with least-squares regression!

If we can do regression with the base hypothesis set $\mathscr{H}$, then we're all set!

This is the subproblem at step $t \in [T]$ for solving the overall problem:

$$\text{Minimize } F(\nu_1, \ldots, \nu_T, h_1, \ldots, h_T) = \frac{1}{n} \sum_{i=1}^{n} \ell \left(\sum_{t=1}^{T} \nu_t h_t(x^{(i)}), y^{(i)} \right) \text{ w.r.t. } \nu_1, \ldots, \nu_T, h_1, \ldots, h_T.$$

Regression Stump

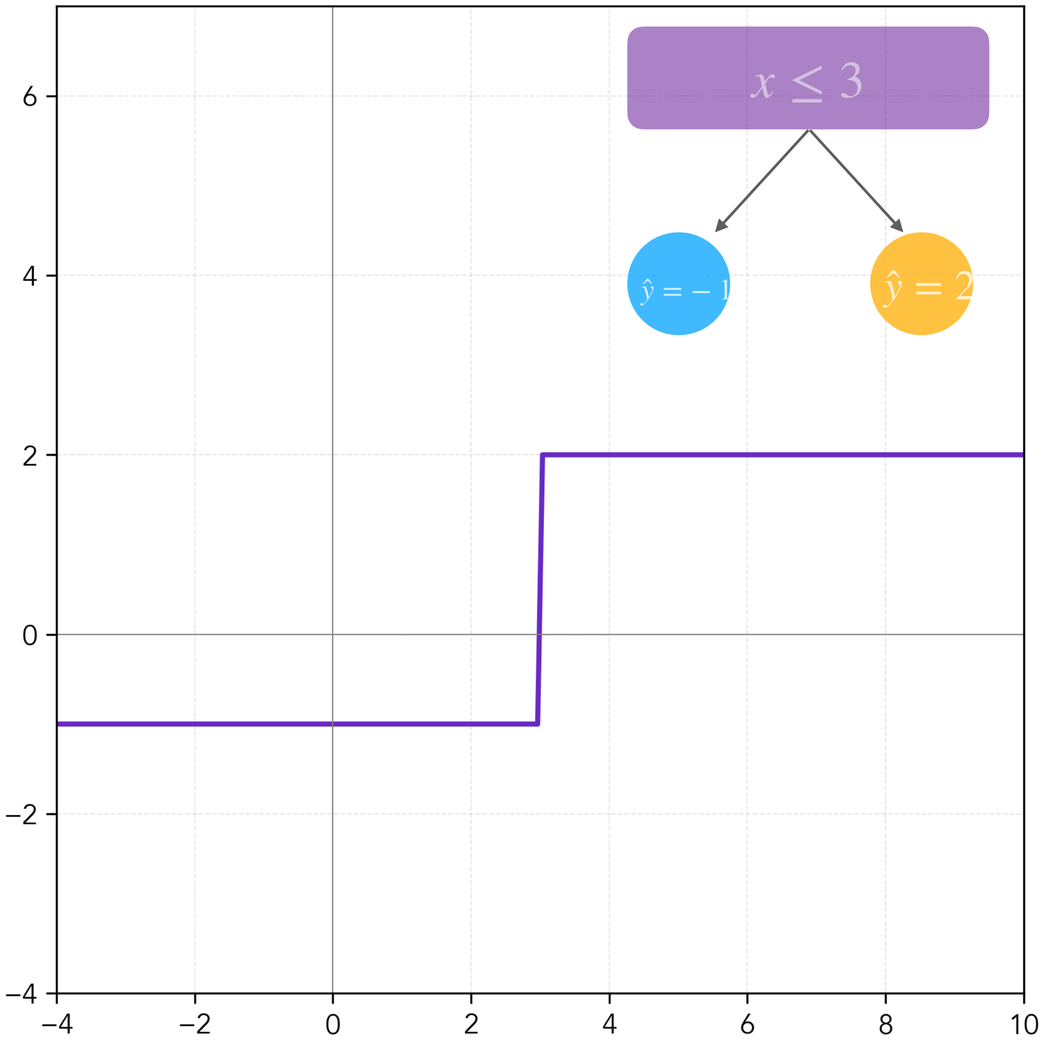
Simple $\mathscr{H}$ for Regression

A regression stump is a function of form:

$$h_{a,b,c}(x) = a\mathbf{1}\{x_j \leq c\} + b\mathbf{1}\{x_j > c\}$$

for a feature $j \in [d]$.

$$\mathscr{H}_{\text{stump}} := \left\{h_{a,b,c}(x) : a, b, c \in \mathbb{R}, j \in [d]\right\}$$



L^2 Boosting

Using $\mathscr{H}_{\text{stump}}$ for FSAM

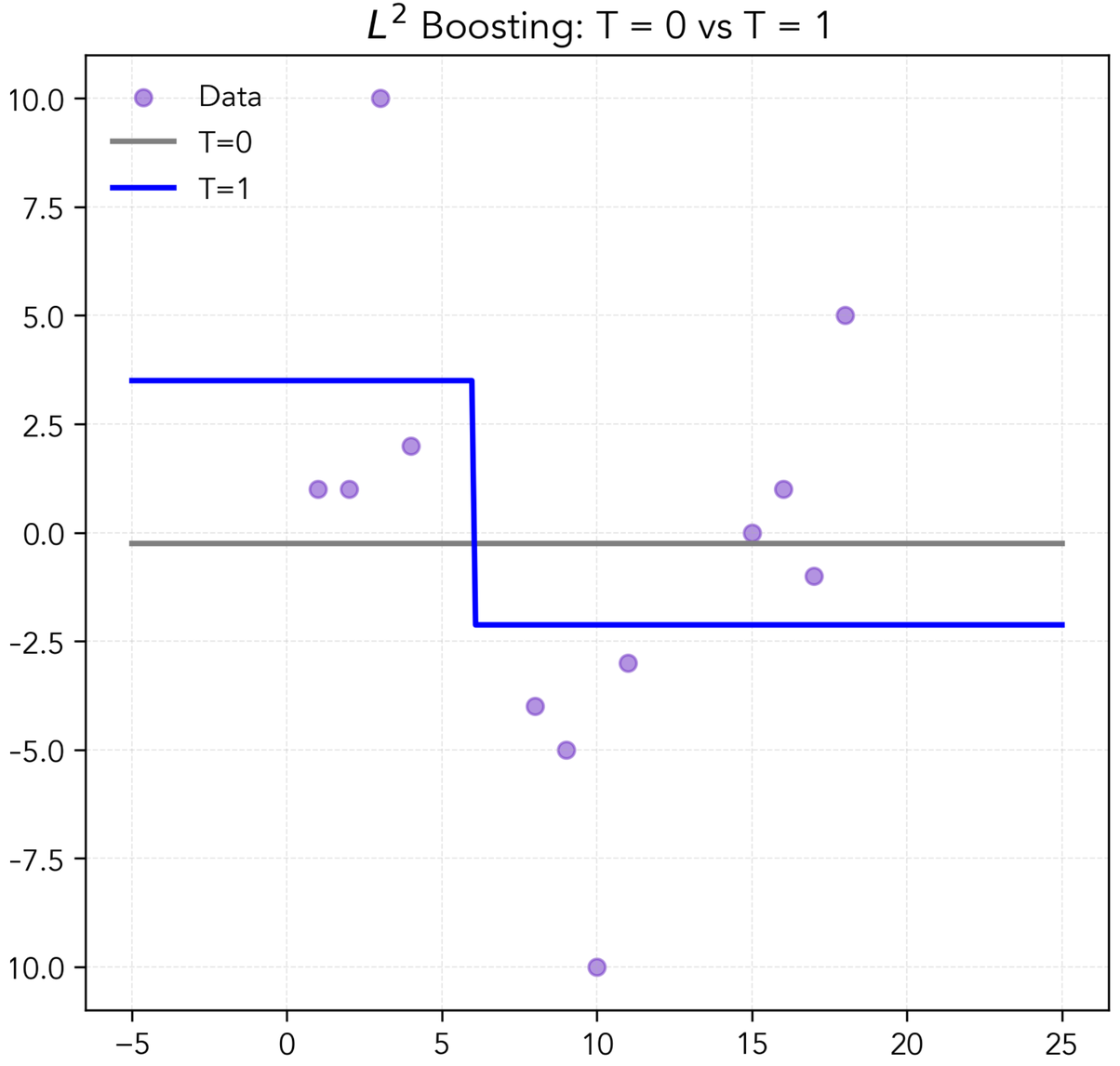
Overall goal:

$$\hat{f} \in \underset{\nu_1,\ldots,\nu_T,h_1,\ldots,h_T}{\arg\min} \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{t=1}^{T}\nu_t h_t(x^{(i)}) - y^{(i)}\right)^2$$

Using FSAM, at step $t \in 1,\ldots,T$, fit:

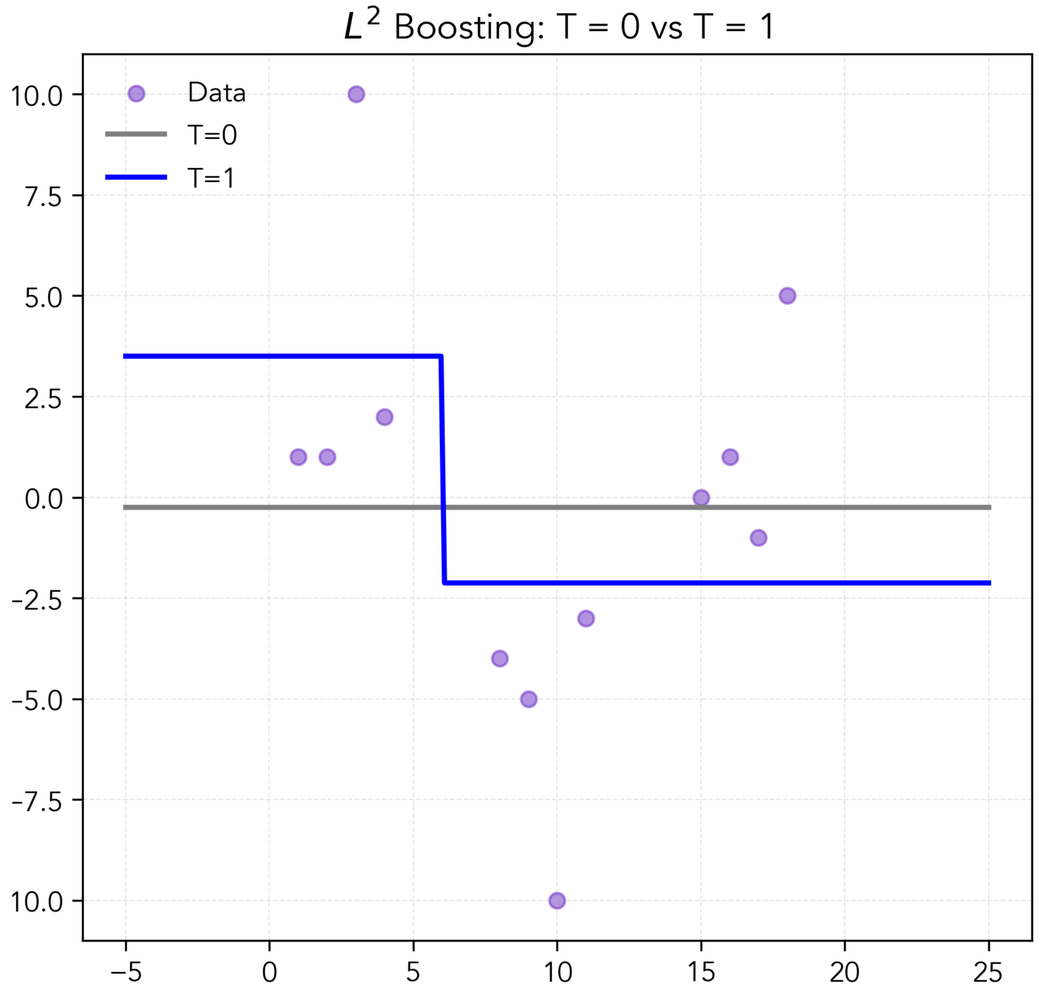
$$h_t \in \underset{h\in\mathscr{H}}{\arg\min} \frac{1}{n}\sum_{i=1}^{n}\left(\left[y^{(i)} - f_{t-1}(x^{(i)})\right] - h(x^{(i)})\right)^2$$

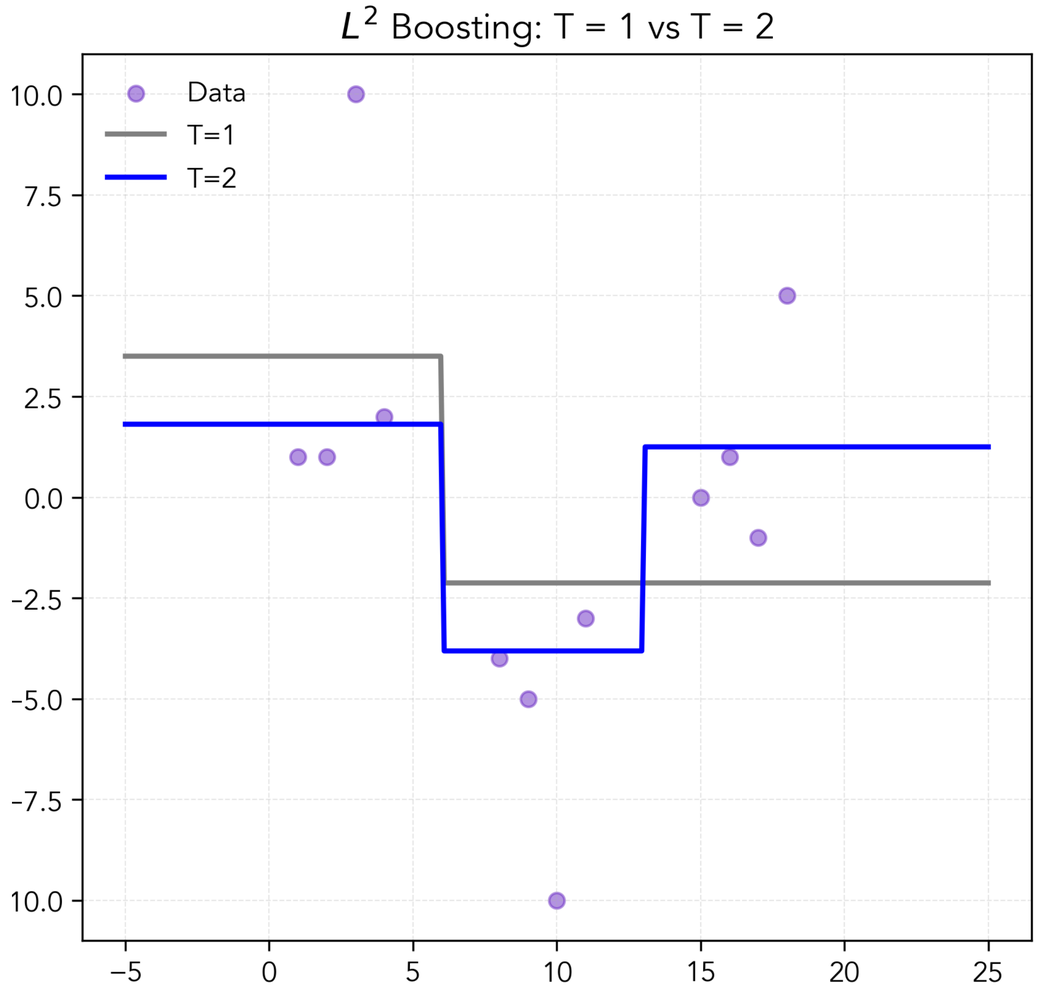
Choose $\mathscr{H}$ = regression stumps.



L^2 Boosting

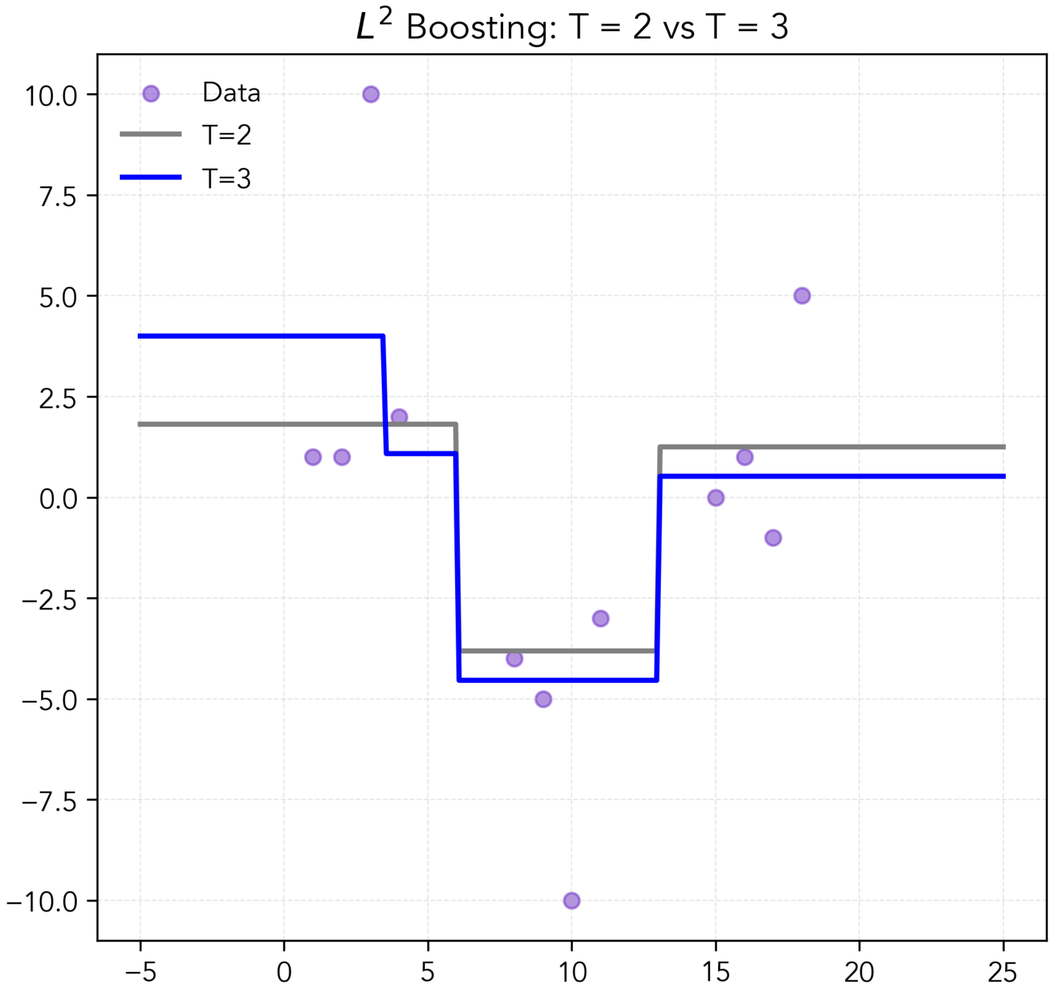
Example: Regression Stumps

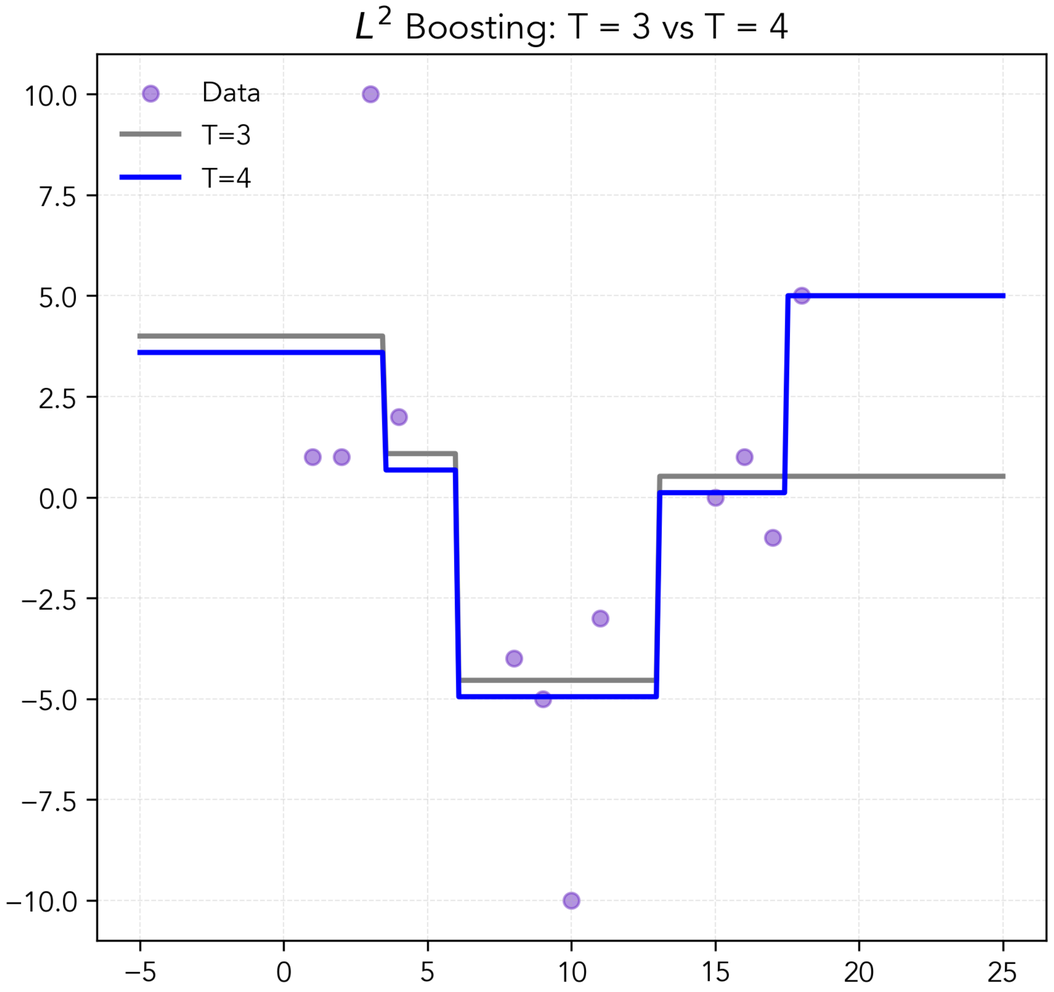




L^2 Boosting

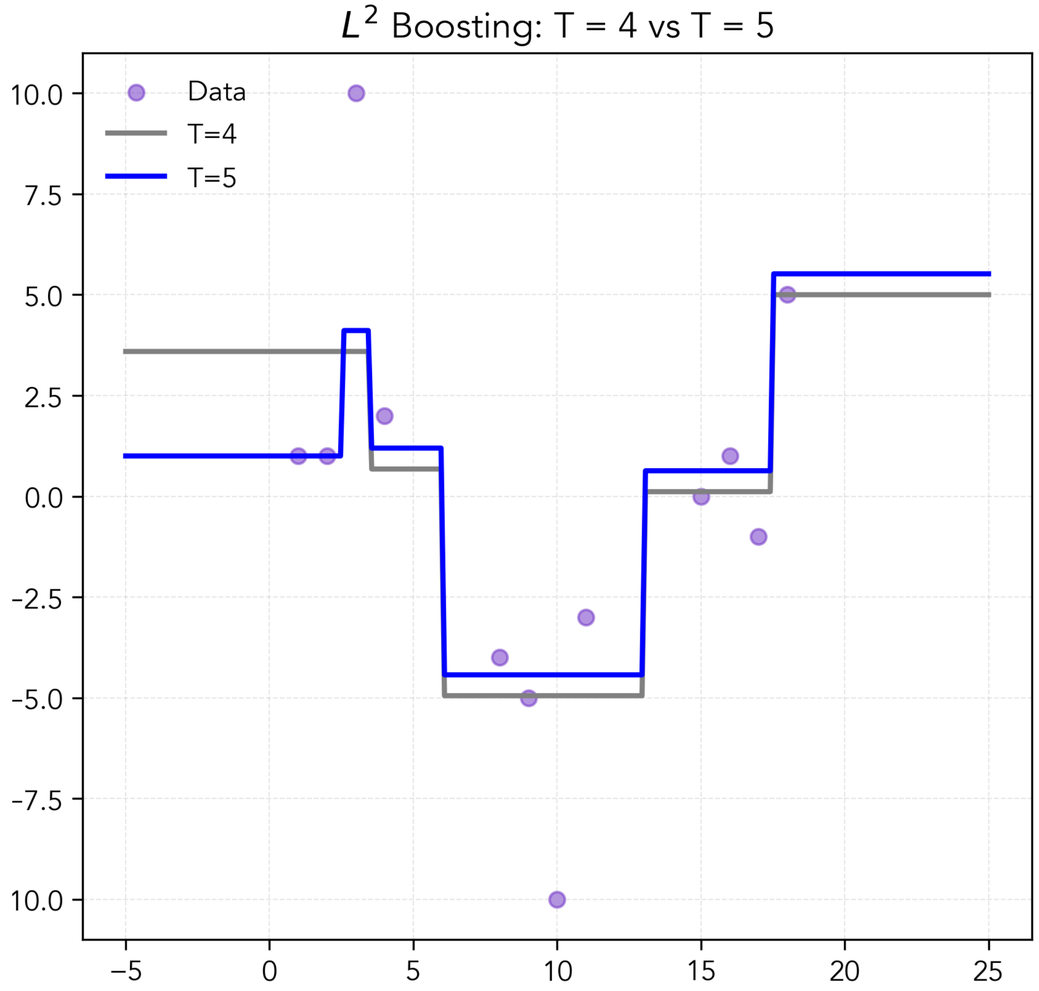
Example: Regression Stumps

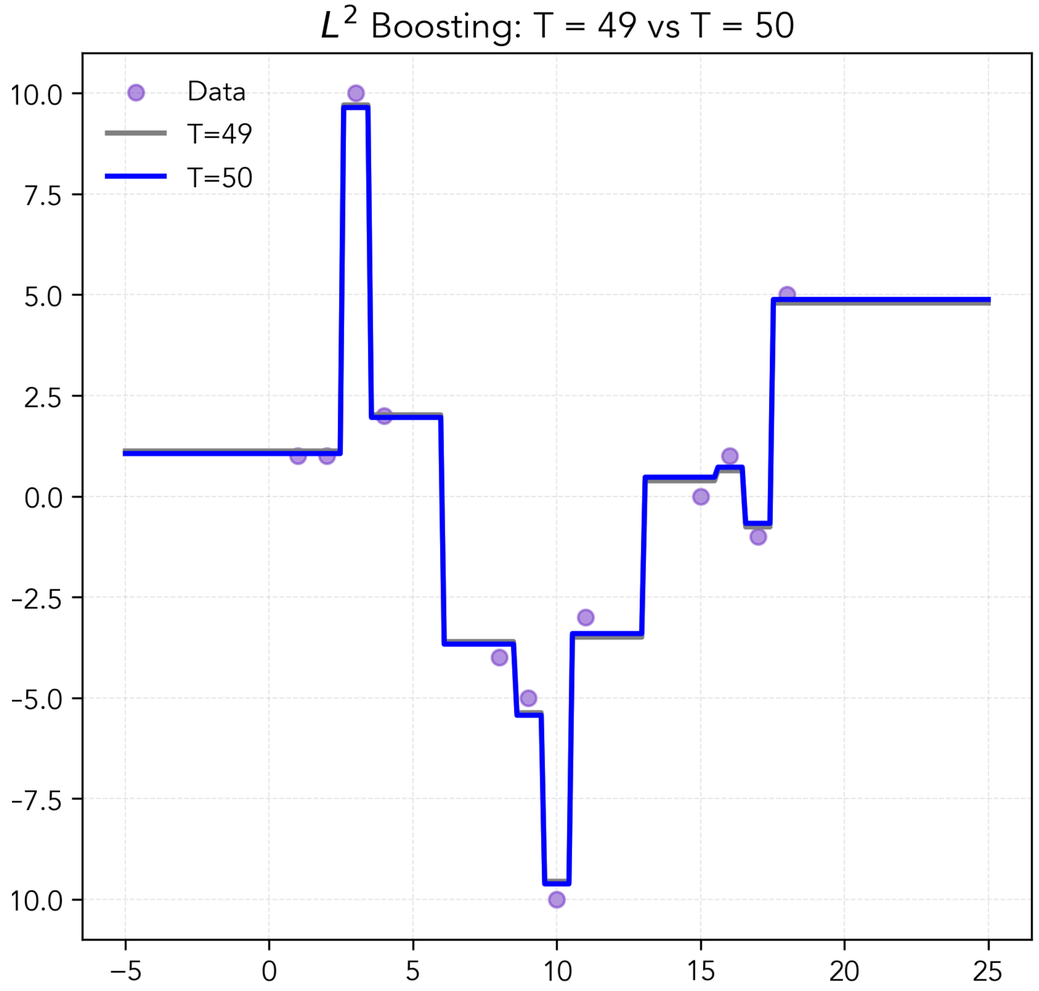




L^2 Boosting

Example: Regression Stumps





Outline

Boosting Basics

Forward Stagewise Additive Modeling (FSAM)

Example: L^2 Boosting (Regression)

Example: AdaBoost (Classification)

Gradient Boosting

Classification

Loss Functions

Outcome space: $\mathcal{Y} = \{-1, +1\}$.

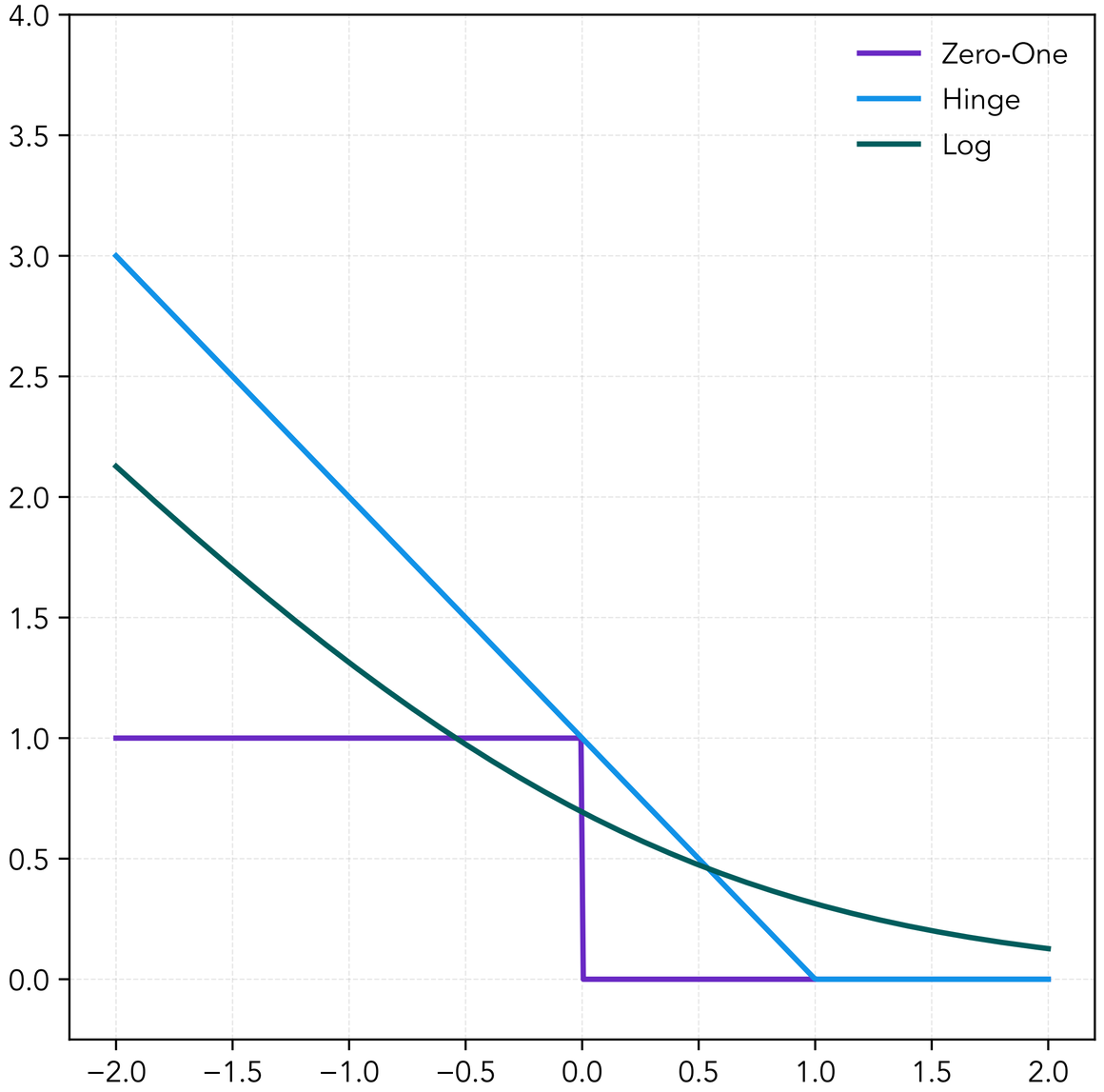
Score function: $f(x)\colon \mathcal{X} \to \mathbb{R}$.

Margin for example (x, y) is:

$$m = yf(x)$$

$m > 0 \iff$ classification is correct.

Larger m is "more confident."



Classification

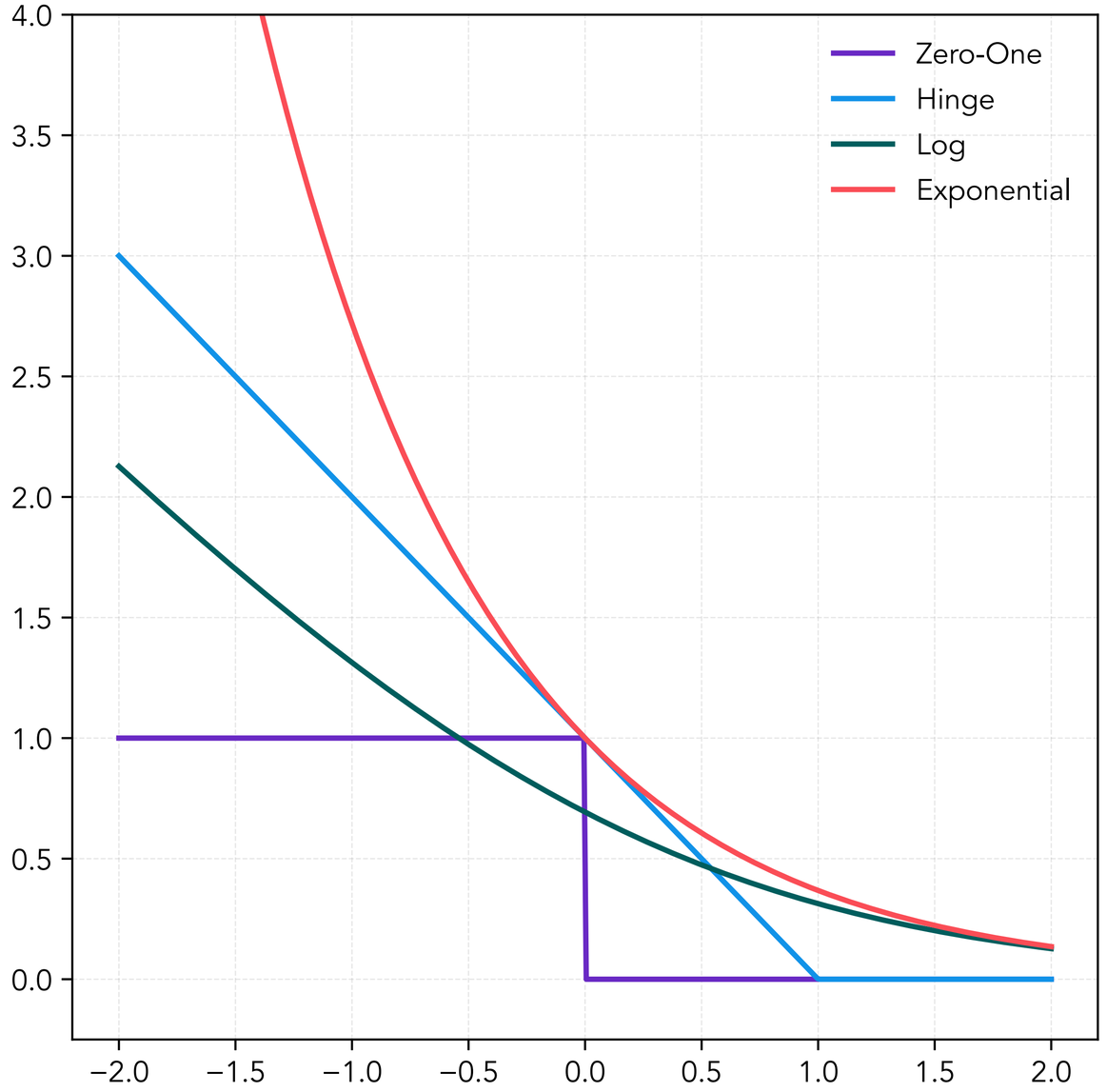
Exponential Loss

Margin for example (x, y) is:

$$m = yf(x)$$

The exponential loss is:

$$\ell(f(x), y) := \exp(-yf(x)).$$



Gradient Boosting for Classification

FSAM with Exponential Loss

Consider classification setting: $\mathcal{Y} = \{-1, +1\}$.

Take loss function to be exponential loss:

$$\ell(f(x), y) := \exp(-yf(x)).$$

Let $\mathcal{H}$ to be a base hypothesis class of classifiers $h\colon \mathcal{X} \to \{-1, +1\}$.

Then, FSAM reduces to a version of AdaBoost.

Main FSAM step: $J(h) = \frac{1}{n} \sum_{i=1}^{n} \exp\left(-y^{(i)}(f_{t-1}(x^{(i)} + \nu h(x^{(i)}))\right)$

…can re-derive as a weighted classification problem: $J(h) = \sum_{i=1}^{n} w_t^{(i)} \mathbf{1}\{y^{(i)} \neq h(x^{(i)})\}$

Exponential Loss

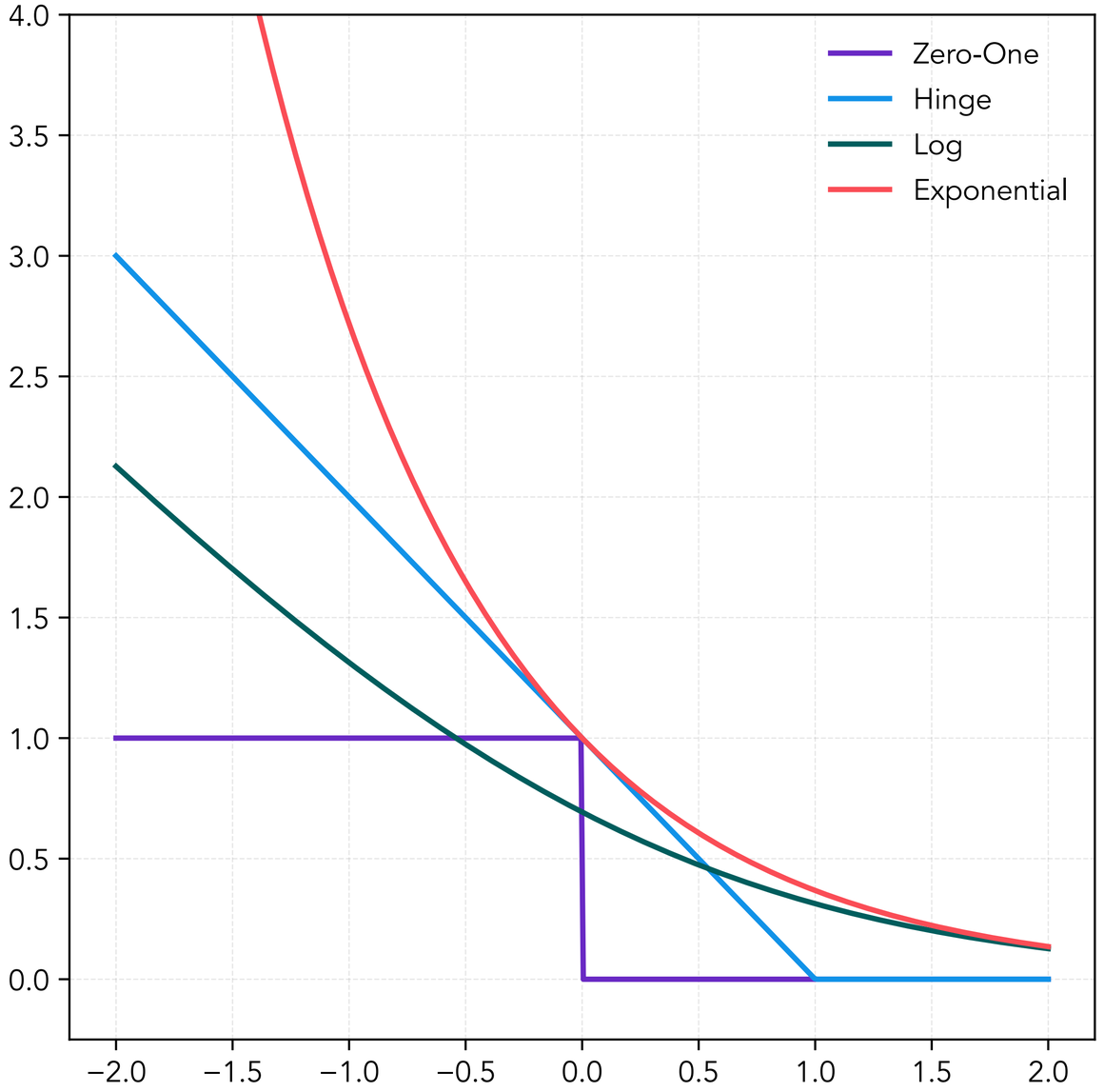
Robustness of AdaBoost

When Bayes error rate is high (e.g. $\Pr(f^*(x) \neq y) = 0.25$)…

> e.g. when there is some intrinsic noise/randomness in the label.
>
> e.g. training examples with same input, different classifications.

Exponential loss puts high penalty on misclassified examples.

Empirically, AdaBoost has degraded performance in high Bayes error settings.



Outline

Boosting Basics

Forward Stagewise Additive Modeling (FSAM)

Example: L^2 Boosting (Regression)

Example: AdaBoost (Classification)

Gradient Boosting

FSAM Step

As iterative optimization

$$\text{FSAM step: } (\nu_t, h_t) \in \underset{\nu \in \mathbb{R}, h \in \mathscr{H}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \ell \left(f_{t-1}(x^{(i)}) + \nu h(x^{(i)}), y^{(i)} \right)$$

Hard part: finding the best **step direction** h.

Look for the locally best step direction (like in gradient descent).

What does this look like for squared loss?

Gradient Boosting

Main Idea: "Functional" Gradient Descent

Take a step back, our ERM objective is, as a function of $f \in \mathcal{F}_T$ is:

$$\hat{R}(f) := \sum_{i=1}^{n} \ell(f(x^{(i)}), y^{(i)}).$$

In some sense, want to take the gradient "with respect to f."

Objective $\hat{R}$ only depends on f at the n training points.

Define the vector: $\mathbf{f} = (f(x^{(1)}), \ldots, f(x^{(n)})) \in \mathbb{R}^n$.

Objective is now: $\hat{R}(\mathbf{f}) = \sum_{i=1}^{n} \ell(\mathbf{f}_i, y^{(i)})$ (a function $\mathbb{R}^n \to \mathbb{R}$).

Gradient Boosting

Main Idea: "Functional" Gradient Descent

Define the vector: $\mathbf{f} = (f(x^{(1)}), \ldots, f(x^{(n)})) \in \mathbb{R}^n$.

Objective is now: $\hat{R}(\mathbf{f}) = \sum_{i=1}^{n} \ell(\mathbf{f}_i, y^{(i)})$ (a function $\mathbb{R}^n \to \mathbb{R}$).

The negative gradient step direction at $\mathbf{f}$ is:

$$-\mathbf{g} = -\nabla_{\mathbf{f}} \hat{R}(\mathbf{f}) = -\left(\partial_{\mathbf{f}_1} \ell(\mathbf{f}_1, y^{(1)}), \ldots, \partial_{\mathbf{f}_n} \ell(\mathbf{f}_n, y^{(n)})\right)$$, which we can calculate.

The $-\mathbf{g} \in \mathbb{R}^n$ is the direction we want to change the n predictions on the training data.

Gradient Boosting

Unconstrained Step in Function Space

$$-\mathbf{g} = -\nabla_{\mathbf{f}} \hat{R}(\mathbf{f}) = -\left(\partial_{\mathbf{f}_1} \ell(\mathbf{f}_1, y^{(1)}), \ldots, \partial_{\mathbf{f}_n} \ell(\mathbf{f}_n, y^{(n)})\right)$$

We can consider taking "steps in the function space."

$$\mathbf{f}^{(t)} \leftarrow \mathbf{f}^{(t-1)} - \nu_t \nabla_{\mathbf{f}} \hat{R}(\mathbf{f})$$

Eventually, we have "function" $\mathbf{f}^{(T)} = \sum_{t=0}^{T} \mathbf{f}^{(t)}$.

Issue: $\mathbf{f}^{(T)}$ is defined only at training points!

Gradient Boosting

Example: Squared Loss

$$-\mathbf{g} = -\nabla_{\mathbf{f}} \hat{R}(\mathbf{f}) = -\left(\partial_{\mathbf{f}_1} \ell(\mathbf{f}_1, y^{(1)}), \ldots, \partial_{\mathbf{f}_n} \ell(\mathbf{f}_n, y^{(n)})\right)$$

These are called the pseudo-residuals in the context of gradient boosting.

Example. Taking the derivative "with respect to" $f(x^{(i)})$ for squared loss.

$$\hat{R}(f) = \sum_{i=1}^{n} (f(x^{(i)}) - y^{(i)})^2$$

$$\frac{\partial}{\partial f(x^{(i)})} \hat{R}(f) = -2(y^{(i)} - f(x^{(i)})) \implies -\mathbf{g} = 2(y - \mathbf{f}) \text{ where } y \in \mathbb{R}^n \text{ and } \mathbf{f} \in \mathbb{R}^n.$$

Gradient Boosting

"Projection" Step onto $\mathscr{H}$

$$-\mathbf{g} = -\nabla_{\mathbf{f}} \hat{R}(\mathbf{f}) = -\left(\partial_{\mathbf{f}_1} \ell(\mathbf{f}_1, y^{(1)}), \ldots, \partial_{\mathbf{f}_n} \ell(\mathbf{f}_n, y^{(n)})\right)$$

These are called the pseudo-residuals in the context of gradient boosting.

Issue: $\mathbf{f}$ is defined only at training points!

The "trick" in gradient boosting is to find closest base hypothesis $h \in \mathscr{H}$:

$$\min_{h \in \mathscr{H}} \sum_{i=1}^{n} (-\mathbf{g}_i - h(x^{(i)}))^2$$ so now h is defined everywhere!

This is a least-squares regression problem over the hypothesis space $\mathscr{H}$!

Gradient Boosting

Unconstrained Step in Function Space

$$-\mathbf{g} = -\nabla_{\mathbf{f}} \hat{R}(\mathbf{f}) = -\left(\partial_{\mathbf{f}_1} \ell(\mathbf{f}_1, y^{(1)}), \ldots, \partial_{\mathbf{f}_n} \ell(\mathbf{f}_n, y^{(n)})\right)$$

We can consider taking "steps in the function space."

$$\mathbf{f}^{(t)} \leftarrow \mathbf{f}^{(t-1)} - \nu_t \mathbf{g}^{(t-1)}$$

$$\min_{h \in \mathcal{H}} \sum_{i=1}^{n} (-\mathbf{g}_i - h(x^{(i)}))^2$$

$$f_t(x) = f_{t-1}(x) + \nu_t h_t(x)$$

Gradient Boosting

Step Size

$$\mathbf{f}^{(t)} \leftarrow \mathbf{f}^{(t-1)} - \nu_t \mathbf{g}^{(t-1)} \text{ and approximate with } \min_{h \in \mathcal{H}} \sum_{i=1}^{n} (-\mathbf{g}_i - h(x^{(i)}))^2$$

$$f_t(x) = f_{t-1}(x) + \nu_t h_t(x)$$

Finally, we need to choose a stepsize.

Option 1 (line search): $\nu_t = \arg\min_{\nu > 0} \sum_{i=1}^{n} \ell(f_{t-1}(x^{(i)}) + \nu h_t(x^{(i)}), y^{(i)})$

Option 2 (fixed stepsize): choose $\nu_t \in (0,1)$ or optimize as a hyperparameter.

Gradient Boosting

Example: Squared Loss

With squared loss $\implies -\mathbf{g} = 2(y - \mathbf{f})$ where $y \in \mathbb{R}^n$ and $\mathbf{f} \in \mathbb{R}^n$.

1. Initialize $f_0(x) = 0$.

2. At round $t = 1,2,3,\ldots,T$:

Solve the regression: $h_t = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} (\mathbf{g}_i - h(x^{(i)}))^2 = \sum_{i=1}^{n} (2(y^{(i)} - f_{t-1}(x^{(i)})) - h(x^{(i)}))^2$

Update: $f_t(x) = f_{t-1}(x) + \nu_t h_t(x)$.

3. **Return**: $f_T(x) = \sum_{t=0}^{T} \nu_t h_t(x)$.

“Classical ML” Recap

Supervised Learning

Excess Risk Formalization

1. Collect training dataset, a collection of labeled input-output pairs.

2. Decide on the template of the hypothesis mapping that will map inputs to actions. — Representation

3. A learning algorithm takes the labeled training data as input and outputs a hypothesis. — Optimization

4. The hypothesis predicts on new, unseen data which we hope it does well on, under a notion of loss. — Generalization

We receive $\tilde{h}_n$ from an algorithm.

Excess risk of $\tilde{h}_n$:

$$R(\tilde{h}_n) - R(h^*) =$$

$$\underbrace{R(\tilde{h}_n) - R(\hat{h}_n)}_{\text{opt. error}} + \underbrace{R(\hat{h}_n) - R(h^*_{\mathscr{H}})}_{\text{est. error}} + \underbrace{R(h^*_{\mathscr{H}}) - R(h^*)}_{\text{approx. error}}$$

Optimization | Generalization | Representation

Excess Risk

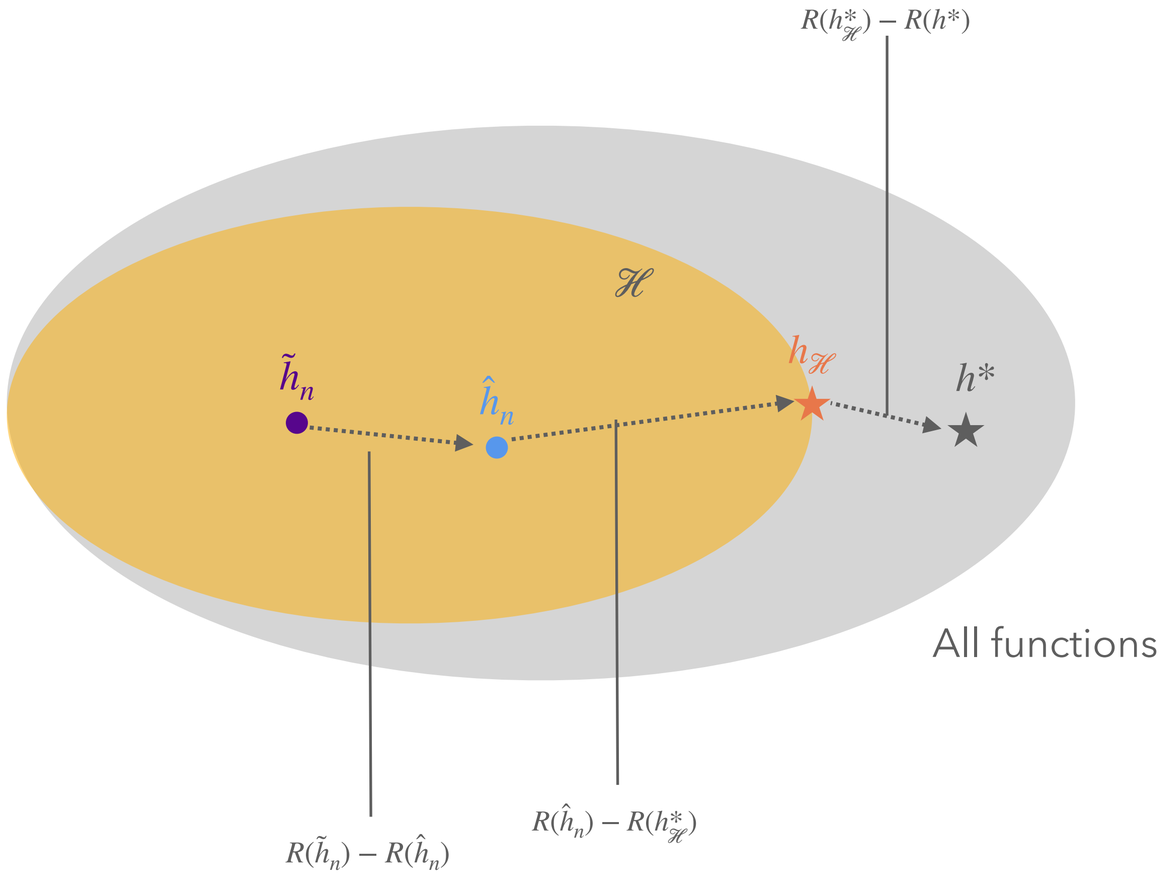
Full Decomposition

We receive $\tilde{h}_n$ from an algorithm.

Excess risk of $\tilde{h}_n$:

$$R(\tilde{h}_n) - R(h^*) =$$

$$\underbrace{R(\tilde{h}_n) - R(\hat{h}_n)}_{\text{opt. error}} + \underbrace{R(\hat{h}_n) - R(h^*_{\mathscr{H}})}_{\text{est. error}} + \underbrace{R(h^*_{\mathscr{H}}) - R(h^*)}_{\text{approx. error}}$$



Three Main Questions

Representation, Optimization, and Generalization

$$R(\tilde{h}_n) - R(h^*) =$$

$$\underbrace{R(\tilde{h}_n) - R(\hat{h}_n)}_{\text{opt. error}} + \underbrace{R(\hat{h}_n) - R(h^*_{\mathscr{H}})}_{\text{est. error}} + \underbrace{R(h^*_{\mathscr{H}}) - R(h^*)}_{\text{approx. error}}$$

Optimization Generalization Representation

Representation: *Which hypothesis class $\mathscr{H}$ best models the relationship of $\mathscr{X}$ to $\mathscr{A}$?*

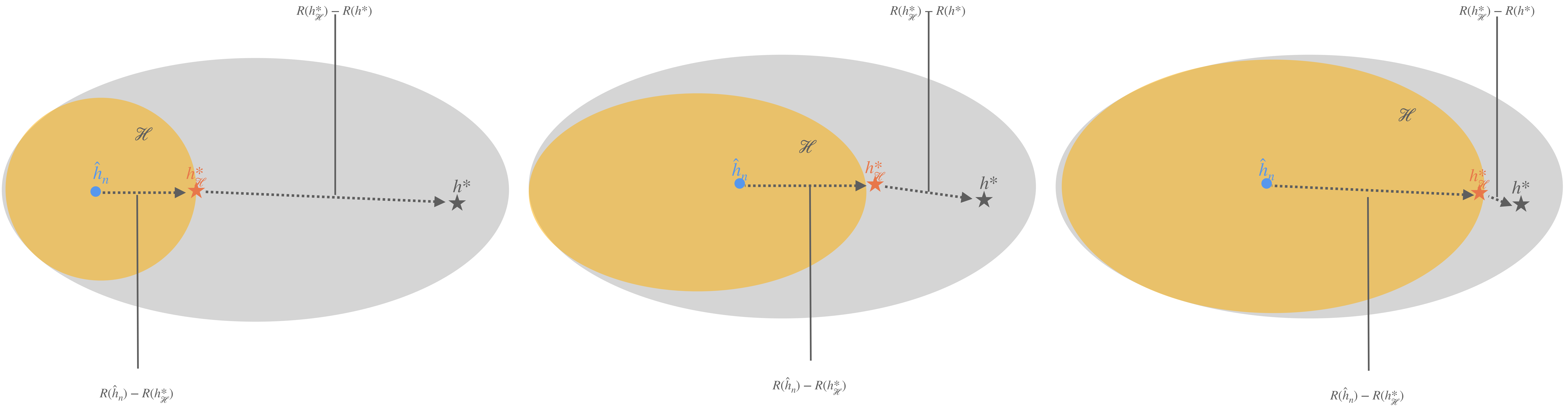
Generalization: *How well can we extrapolate from training data to new, unseen data?*

Optimization: *How can we efficiently and accurately solve the ERM optimization problem?*

Excess Risk

Balanced with regularization

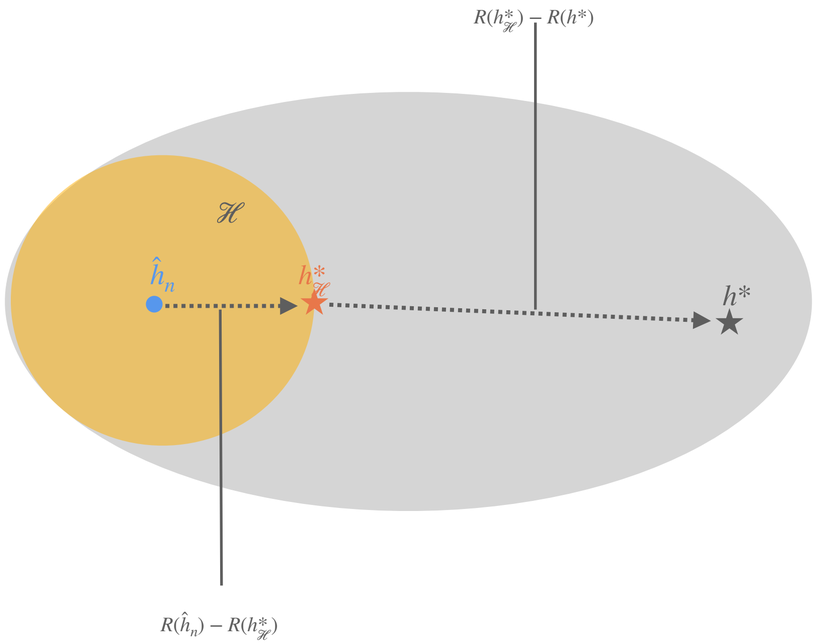
$$R(\hat{h}_n) - R(h^*) = \underbrace{R(\hat{h}_n) - R(h^*_{\mathcal{H}})}_{\text{est. error}} + \underbrace{R(h^*_{\mathcal{H}}) - R(h^*)}_{\text{approx. error}}$$

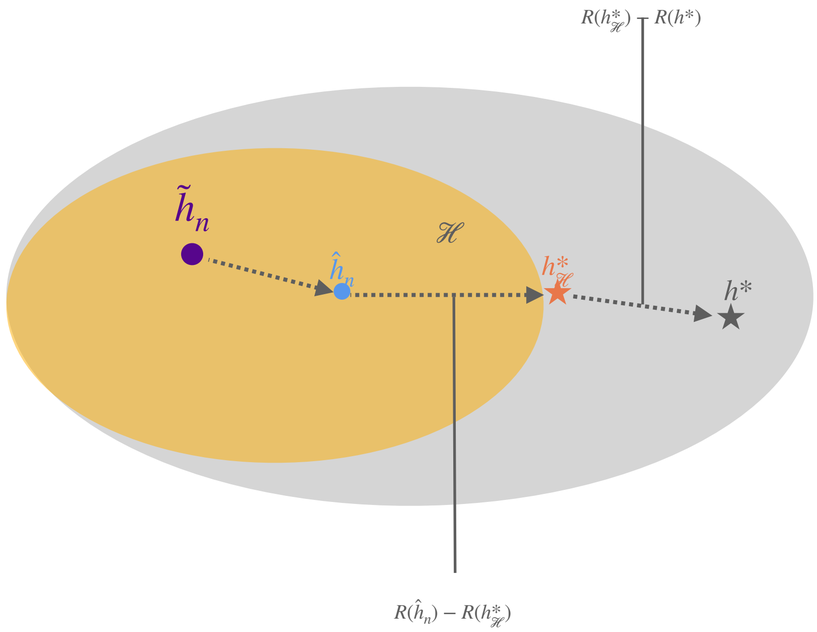


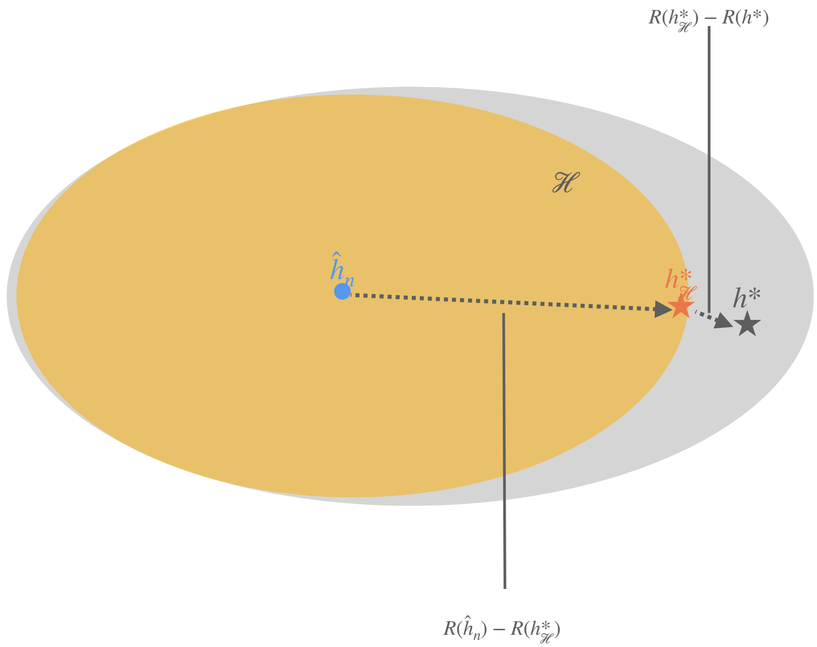
Optimization Error

Handled with gradient descent

$$\underbrace{R(\tilde{h}_n) - R(\hat{h}_n)}_{\text{opt. error}}$$







Learning as ERM

$\mathscr{H}$ = linear models

$$\min_{h \in \mathscr{H}} \sum_{i=1}^{n} \ell(h(x^{(i)}), y^{(i)})$$

Using $\mathscr{H} = \{w^\top x : w \in \mathbb{R}^d\}$ (the class of linear models)…

$\mathscr{Y} = \mathbb{R}$, $\ell(\hat{y}, y) = (\hat{y} - y)^2 \Longrightarrow$ **linear regression**.

$\mathscr{Y} = \mathbb{R}$, $\ell(\hat{y}, y) = (\hat{y} - y)^2$ + regularizer $\Longrightarrow$ **ridge regression**, **lasso regression**.

$\mathscr{Y} = \{-1, +1\}$, $\ell(\hat{y}, y) = \max\{0, 1 - \hat{y}y\}$ + regularizer $\Longrightarrow$ (soft-margin) **SVM**

$\mathscr{Y} = \{-1, +1\}$, $\ell(\hat{y}, y) = \log(1 + e^{-\hat{y}y}) \Longrightarrow$ **logistic regression**

Learning as ERM

$\mathscr{H}$ = decision trees

$$\min_{h \in \mathscr{H}} \sum_{i=1}^{n} \ell(h(x^{(i)}), y^{(i)})$$

Using $\mathscr{H} = \{\text{decision trees of depth d}\}$ (the class of decision trees)…

Solve the ERM with a more heuristic optimization problem (greedy procedure).

Works for $\mathscr{Y} = \mathbb{R}$ (**regression trees**) or $\mathscr{Y} = \{-1, +1\}$ (**classification trees**).

Learning as ERM

$\mathscr{F}$ = ensemble methods with base $\mathscr{H}$

$$\min_{f \in \mathscr{F}} \sum_{i=1}^{n} \ell(f(x^{(i)}), y^{(i)})$$

Using $\mathscr{H} = \{\text{decision trees of depth d}\}$ (the class of decision trees) as a base class.

Let $\mathscr{F} = \left\{\text{combine}(h_1(x), \ldots, h_B(x)) : h_1, \ldots, h_B \in \mathscr{H}\right\}$ (bagging)…

Let $\mathscr{F} = \left\{ \sum_{t=1}^{T} \nu_t h_t(x) : \nu_1, \ldots, \nu_T \in \mathbb{R}, h_1, \ldots, h_T \in \mathscr{H} \right\}$ (boosting)…

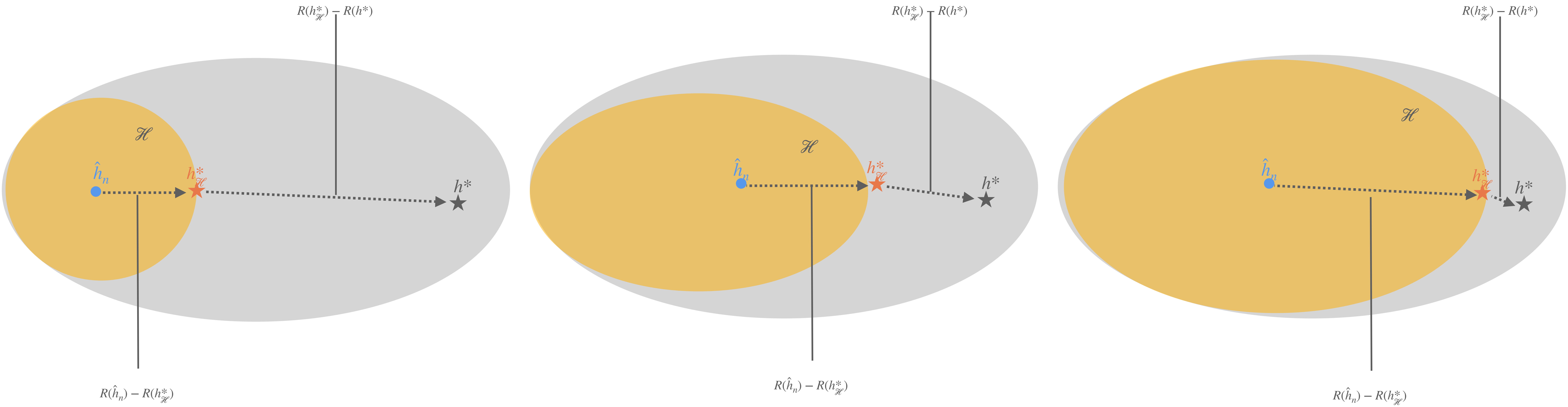
Both methods use ERM over $\mathscr{H}$ as a subroutine.

Excess Risk

Contextualizing "Bias vs. Variance"

Related to "variance" Related to "bias"

$$R(\hat{h}_n) - R(h^*) = \underbrace{R(\hat{h}_n) - R(h^*_{\mathscr{H}})}_{\text{est. error}} + \underbrace{R(h^*_{\mathscr{H}}) - R(h^*)}_{\text{approx. error}}$$



Bias vs. Variance

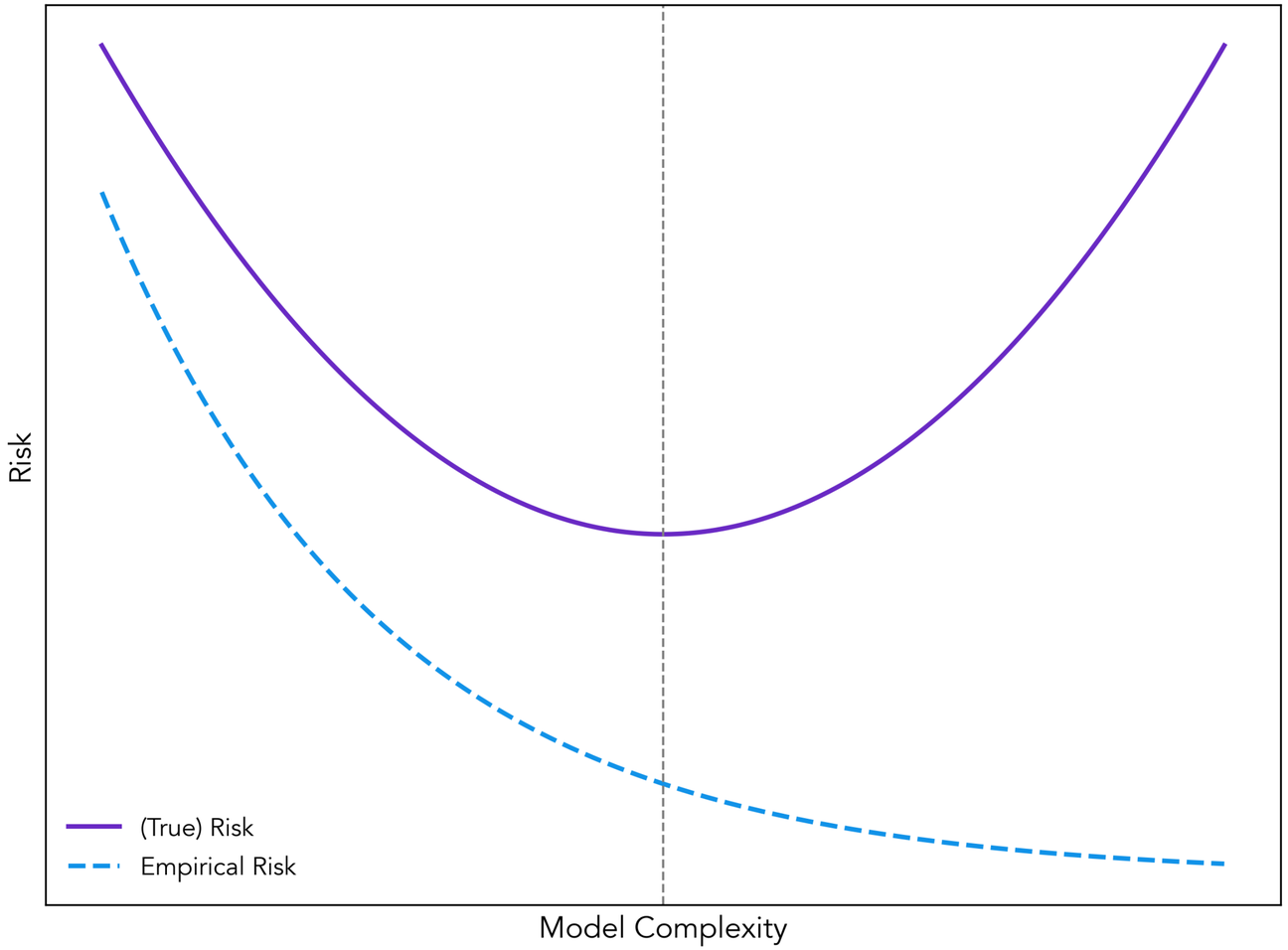
Classical Trade-off

The bias is error incurred from **assumptions** in the learning algorithm.

> Underfitting, less complex an $\mathscr{H}$.
>
> In this class: high bias ≈ high **approximation error**.

Variance is error incurred from sensitivity to fluctuations in the training data.

> Overfitting, more complex an $\mathscr{H}$.
>
> In this class: high variance ≈ high **estimation error**.



Bias vs. Variance

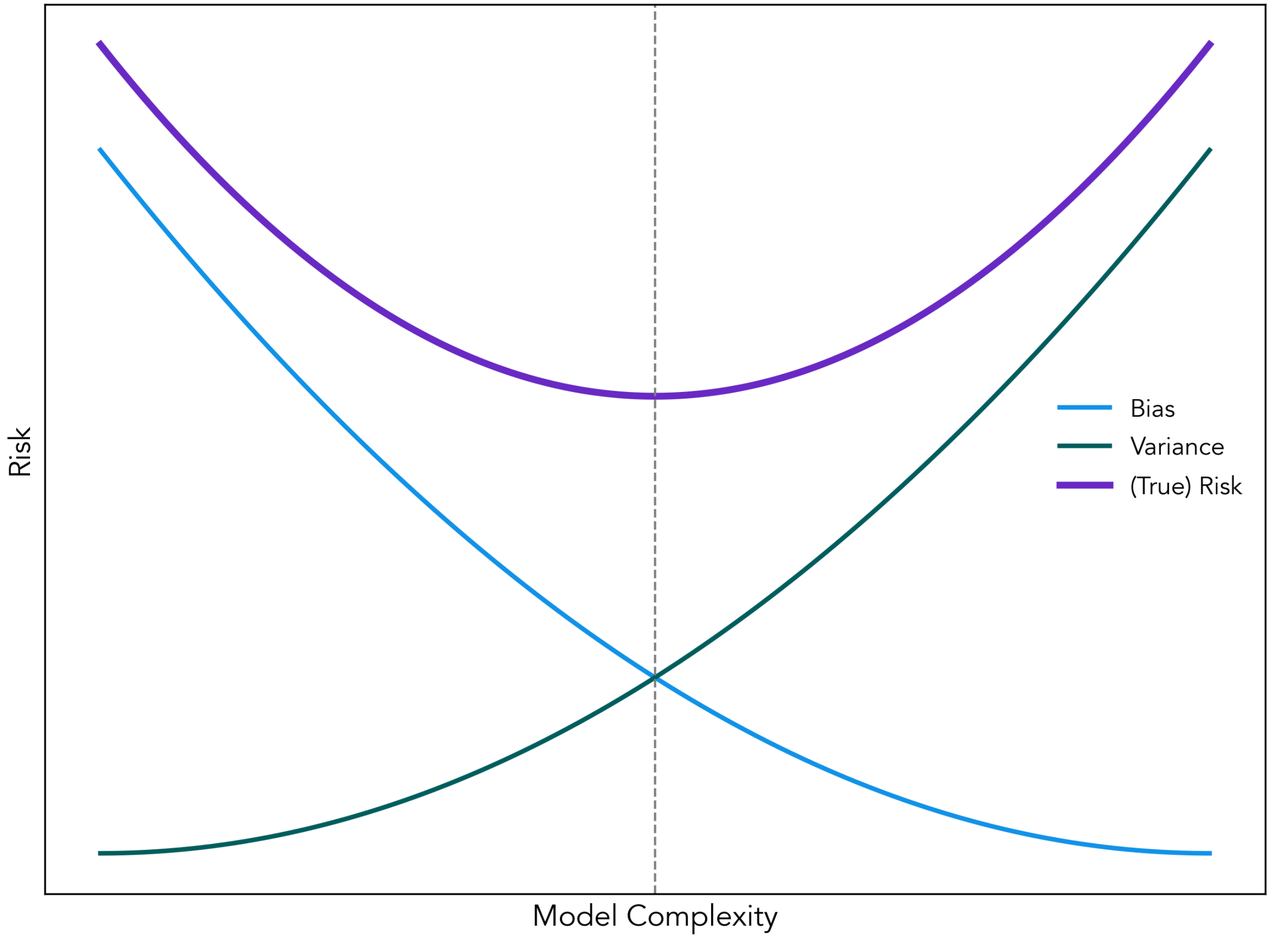
Sources of Bias

The bias is error incurred from **assumptions** in the learning algorithm.

> Underfitting, less complex an $\mathscr{H}$.
>
> In this class: high bias ≈ high **approximation error**.

Examples where we've seen this:

> *Purely linear models (if relationship nonlinear).*
>
> *Choosing small set of features.*
>
> *Strong regularization.*
>
> *Shallow decision trees.*



Bias vs. Variance

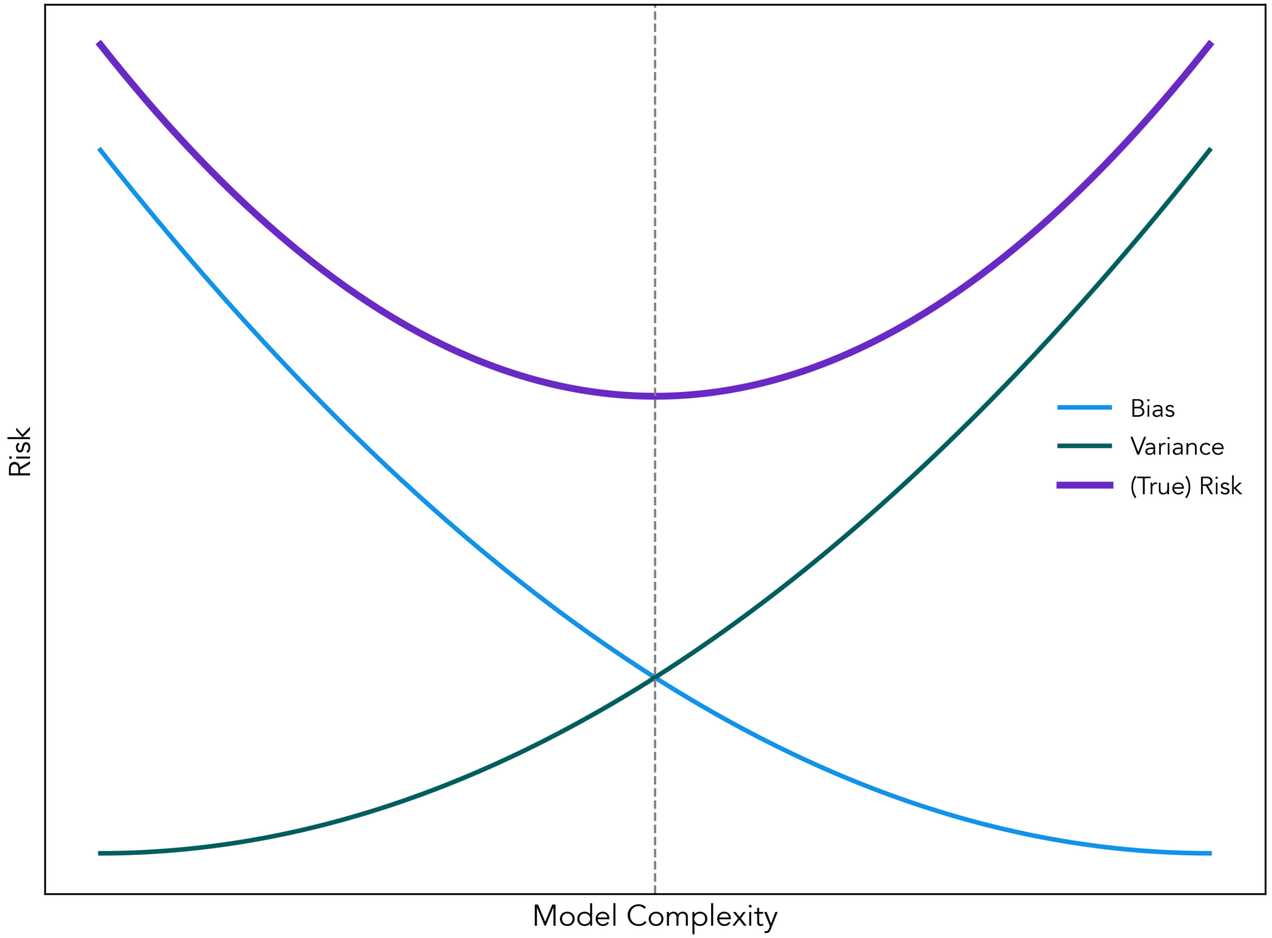
Sources of Variance

Variance is error incurred from sensitivity to fluctuations in the training data.

> Overfitting, more complex an $\mathscr{H}$.
>
> In this class: high variance ≈ high **estimation error**.

Examples where we've seen this:

> *High-degree polynomial regression.*
>
> *Choosing large set of expressive features.*
>
> *Weak regularization.*
>
> *Deep decision trees.*



Bias vs. Variance

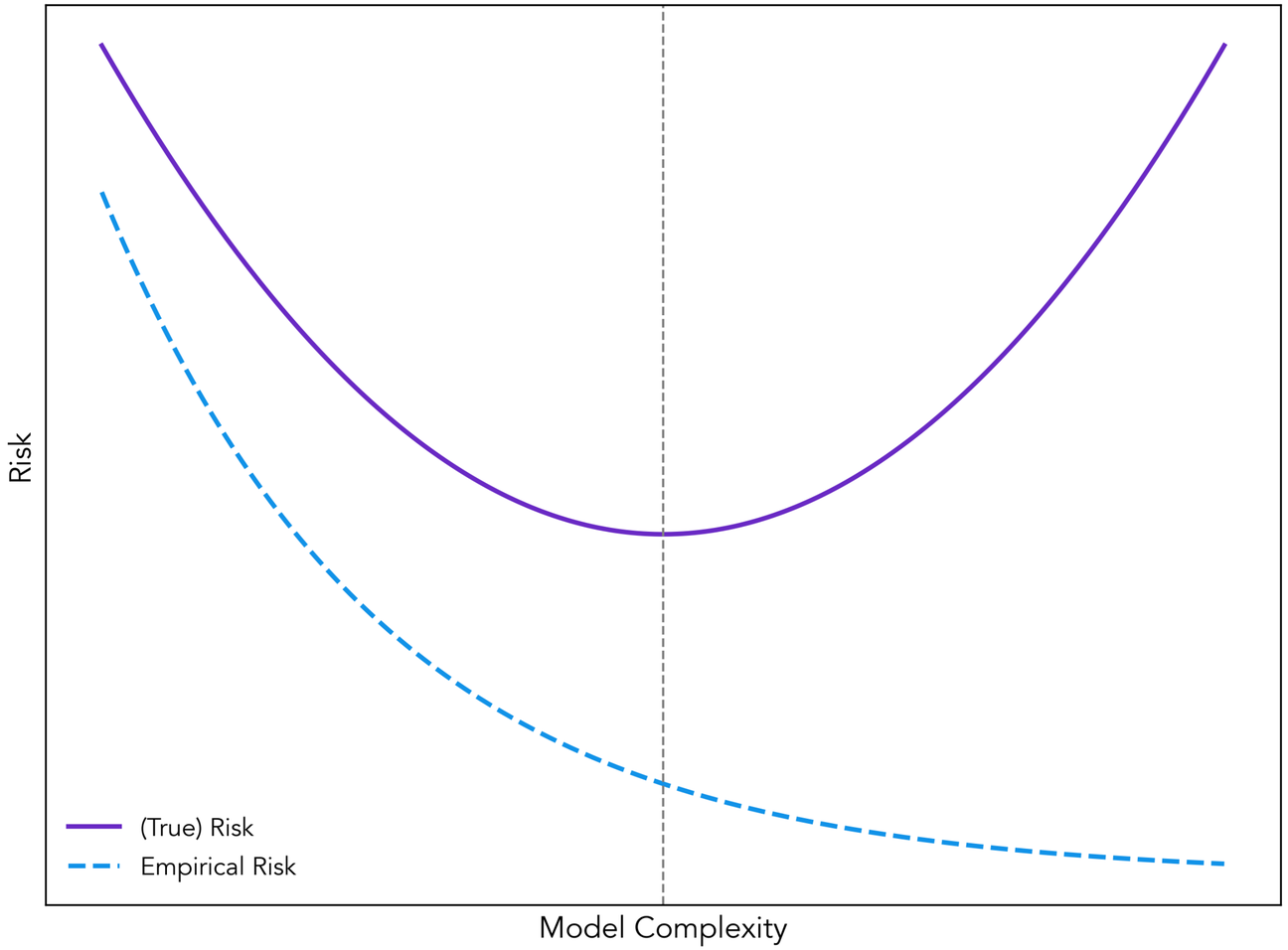
Classical Trade-off

The bias is error incurred from **assumptions** in the learning algorithm.

Variance is error incurred from sensitivity to fluctuations in the training data.

These observations were "classical" wisdom in machine learning…

In the modern era, there is often no "trade-off"!



Bias vs. Variance

Modern View: Double Descent

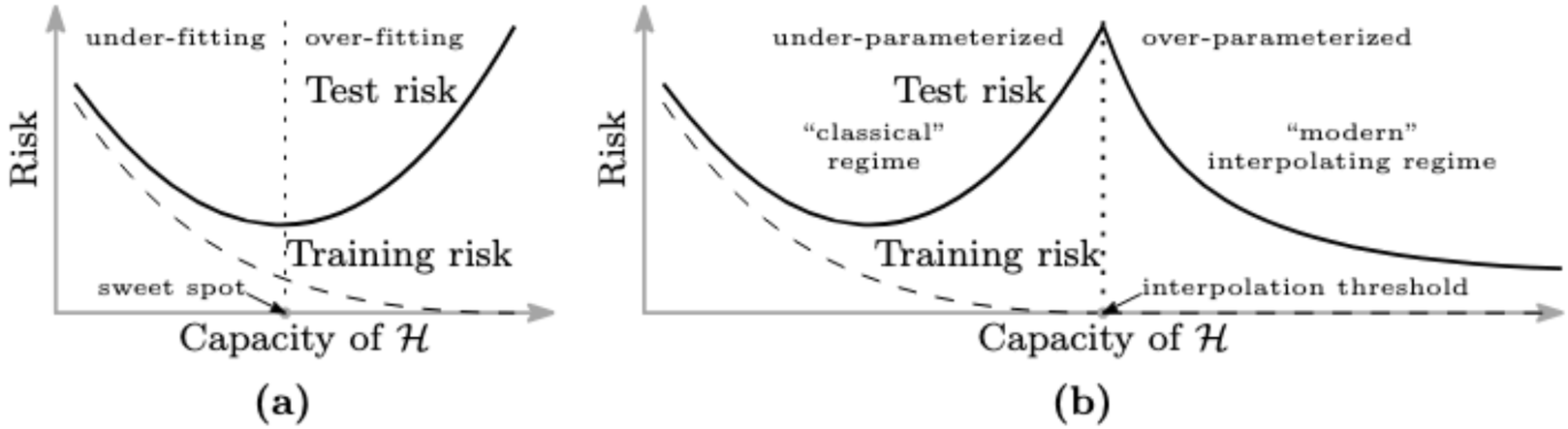


Figure 1: **Curves for training risk (dashed line) and test risk (solid line).** (**a**) The classical *U-shaped risk curve* arising from the bias-variance trade-off. (**b**) The *double descent risk curve*, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high capacity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

Figure from *Reconciling modern machine learning practice and the bias-variance trade-off* (Belkin, Hsu, Ma, Mandal 2019).

Bias vs. Variance

Modern View: Double Descent

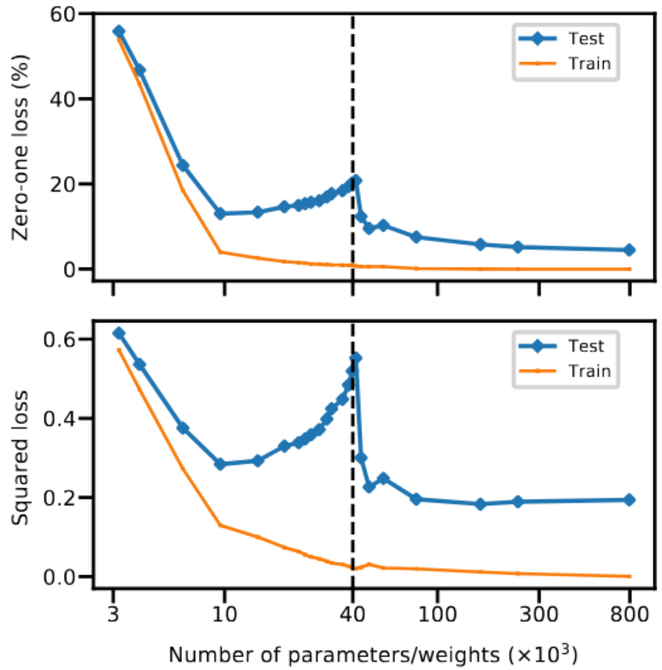


Figure 4: **Double descent risk curve for fully connected neural network on MNIST.** Training and test risks of network with a single layer of H hidden units, learned on a subset of MNIST ($n = 4 \cdot 10^3$, $d = 784$, $K = 10$ classes). The number of parameters is $(d+1) \cdot H + (H+1) \cdot K$. The interpolation threshold (black dotted line) is observed at $n \cdot K$.

Figure from *Reconciling modern machine learning practice and the bias-variance trade-off* (Belkin, Hsu, Ma, Mandal 2019).

Some takeaways from “classical ML”

Handing over to Nick now — thank you!

Learning as optimization. Almost all foundational methods in “classical ML” drop out of ERM atop a well-defined loss function (and hypothesis class).

Linear regression. Ridge regression “is just” ERM with squared loss + ℓ_1/ℓ_2 regularizer.

Linear classification. SVM “is just” ERM with ℓ_2-regularization + hinge loss.

Linear classification. Logistic regression “is just” ERM with logistic loss (solved with GD).

Gradient descent as workhorse algorithm. Can throw GD at most differentiable problems.

Many classical methods were *convex* — guarantees for optimization.

Regularization for balancing estimation error. $\mathscr{H}$ “too big” can lead to overfitting; $\mathscr{H}$ “too small” lead to underfitting.

Linear methods meant features were important. Feature engineering, kernels, linear separability.