DS-GA 1003: Machine Learning

Lecture 3: Regularization and Loss Functions

Slides adapted from material from David Rosenberg.

Logistics & Announcements

PS 1 due today! Due 11:59 PM ET, late deadline is Thursday, 11:59 PM ET.

PS 2 release. Due in two weeks, Tuesday, Feb. 17 11:59 PM ET.

Looks long! But most of the problems are review/exposition and subproblems are short.

Lab this week. Sam will be doing lab this week to lighten up the load for next lecture.

Lecture for Week 5 (02/17) is cancelled due to President's Day.

Lecture on Week 6 (02/24) will be remote and recorded. Sam out of town for conference :(

Math review videos. Stay tuned for several linear algebra review videos!

Outline

Model Complexity and Model Selection

Controlling Complexity with Regularization

ℓ_2 Regularization and Ridge Regression

ℓ_1 Regularization and Lasso Regression

Understanding Sparsity

Loss Functions: Regression

Loss Functions: Classification

Excess Risk

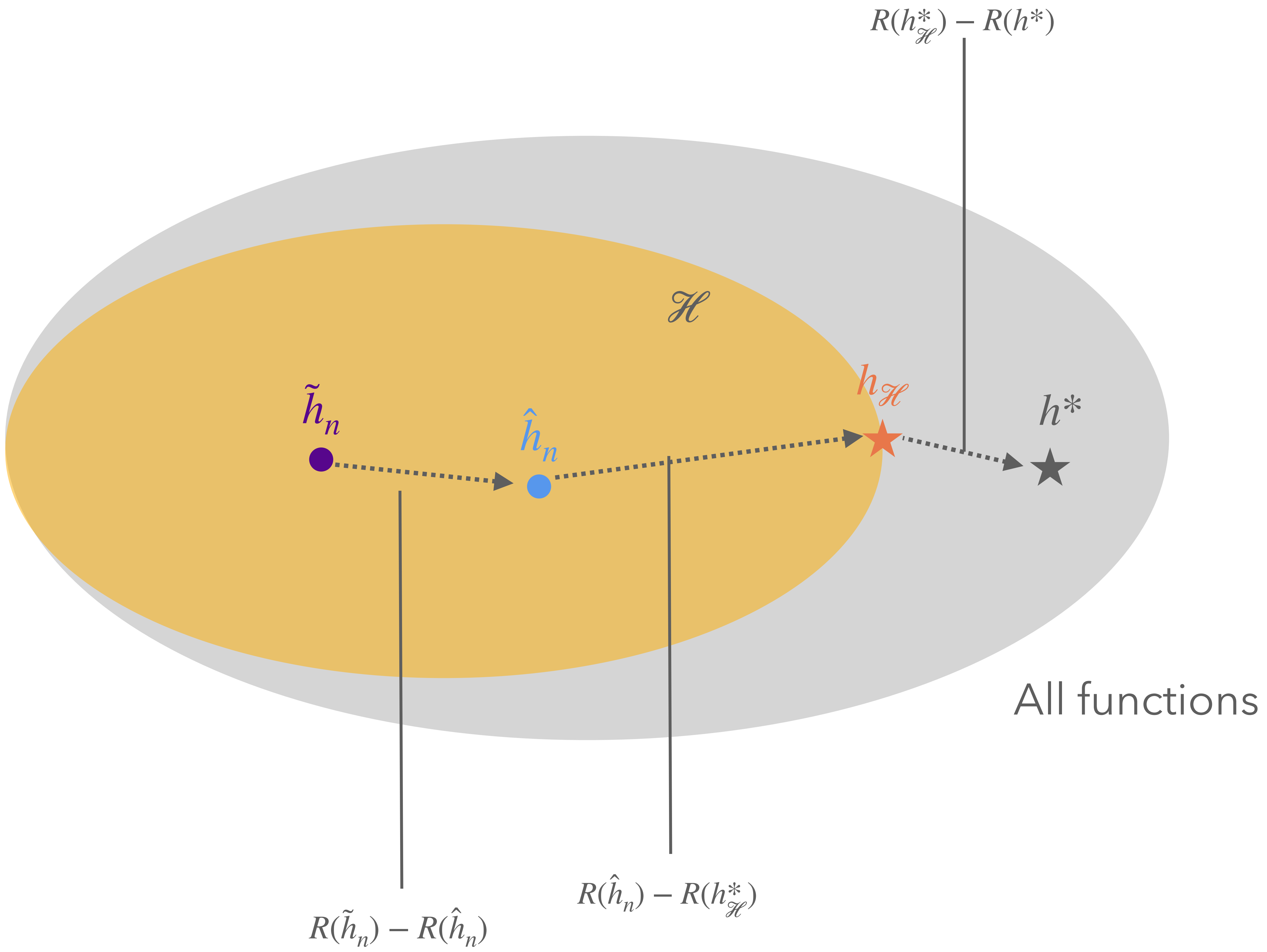
Full Decomposition

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

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

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

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


$R(h^*_{\mathscr{H}}) - R(h^*)$
$\mathscr{H}$
$\tilde{h}_n$
$\hat{h}_n$
$h_{\mathscr{H}}$
h^*
All functions
$R(\hat{h}_n) - R(h^*_{\mathscr{H}})$
$R(\tilde{h}_n) - R(\hat{h}_n)$

Estimation-Approximation Tradeoff

Recurring Theme

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

Estimation error: As $n \to \infty$, typically $R(\hat{h}_n) - R(h^*_{\mathscr{H}}) \to 0$.

Approximation error: Controlled by choosing a good hypothesis class $\mathscr{H}$.

Optimization error: Can we make this small using an efficient algorithm?

How does choosing the “size”/“complexity” of $\mathscr{H}$ affect estimation and approximation error?

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 outputs. 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

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 outputs.

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 $\tilde{h}_n$:

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

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

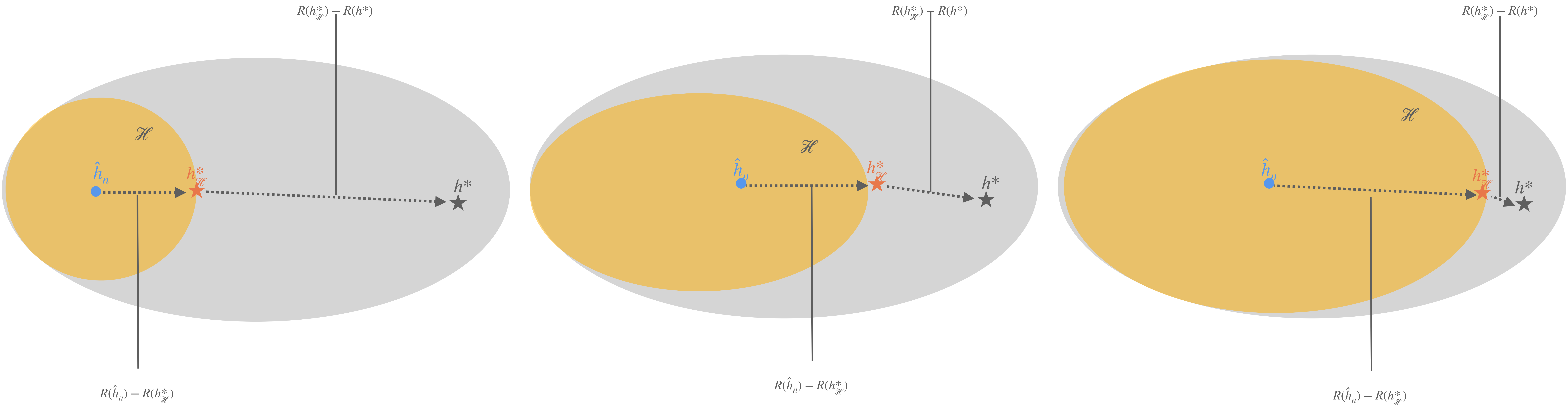
Optimization Generalization Representation

How do we vary the "size" of $\mathscr{H}$ to trade estimation error off with approximation error?

Excess Risk

Intuition: Size of $\mathscr{H}$

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

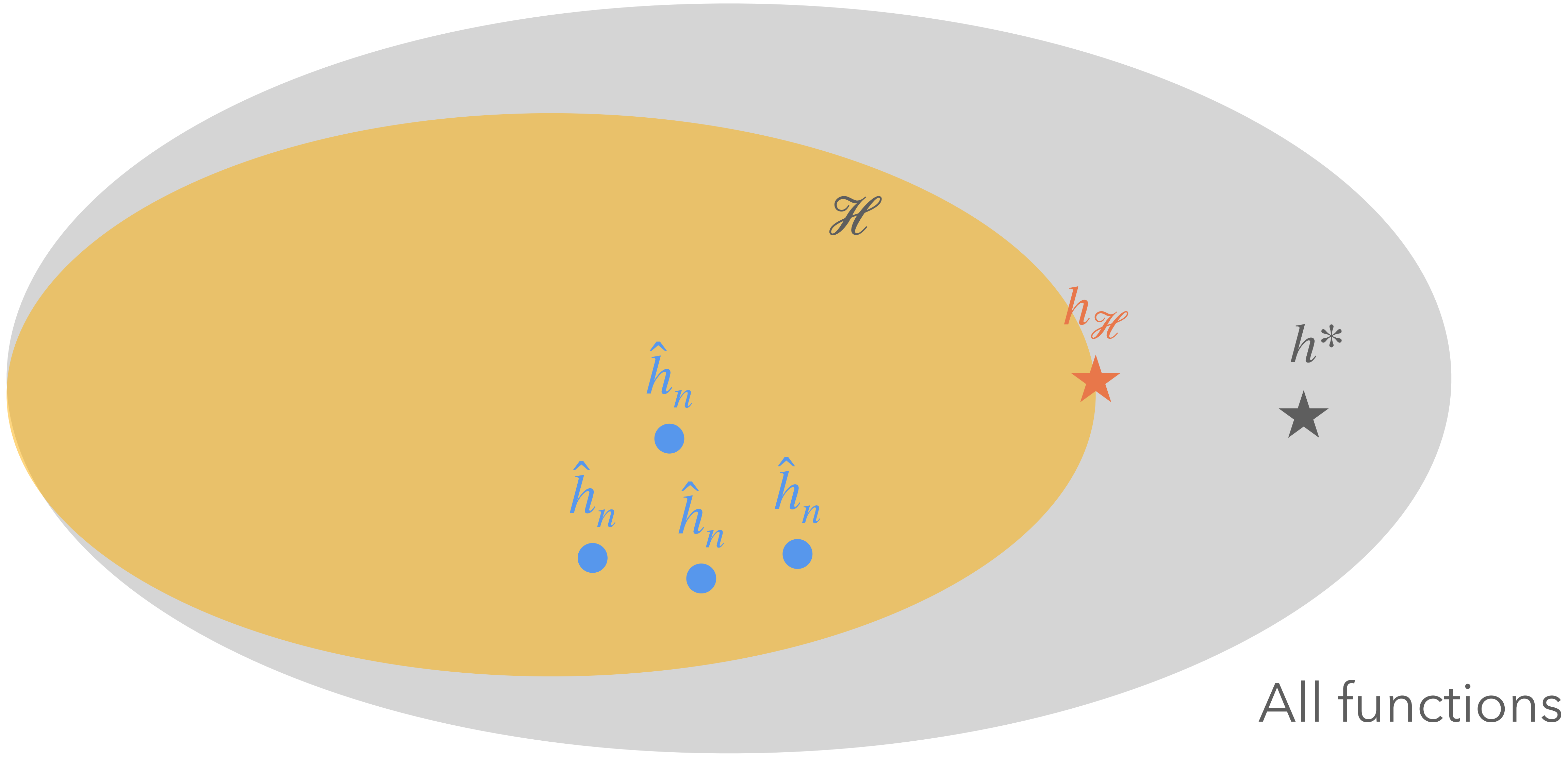

$R(h^*_{\mathscr{H}}) - R(h^*)$
$\mathscr{H}$
$\hat{h}_n$
$h^*_{\mathscr{H}}$
h^*
$R(\hat{h}_n) - R(h^*_{\mathscr{H}})$
$R(h^*_{\mathscr{H}}) - R(h^*)$
$\mathscr{H}$
$\hat{h}_n$
$h^*_{\mathscr{H}}$
h^*
$R(\hat{h}_n) - R(h^*_{\mathscr{H}})$
$R(h^*_{\mathscr{H}}) - R(h^*)$
$\mathscr{H}$
$\hat{h}_n$
$h^*_{\mathscr{H}}$
h^*
$R(\hat{h}_n) - R(h^*_{\mathscr{H}})$

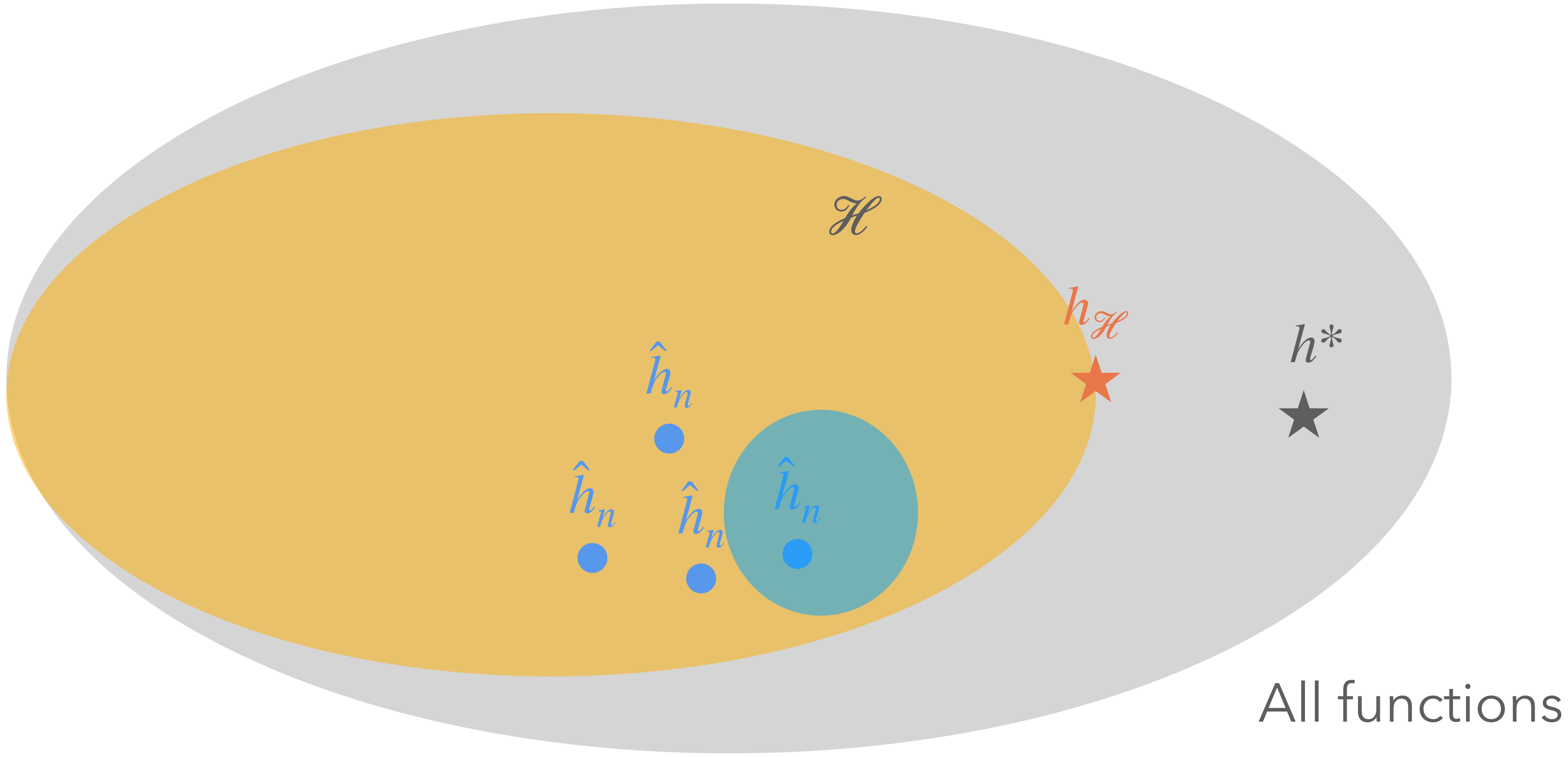
Complexity of $\mathscr{H}$

Trade-off

There can be an infinite number of ERMs!

Regularization: taking a problem with infinitely many solutions and biasing to a smaller ("less complex") subset of solutions.





Controlling Complexity

General Approach

1. Learn a sequence of models varying in complexity from the training data.

$$\mathscr{H}_1 \subset \mathscr{H}_2 \subset \ldots \subset \mathscr{H}_n \subset \mathscr{H}$$

Example: Polynomial Functions

$\mathscr{H}$ = { all polynomial functions}

$\mathscr{H}_d$ = { all polynomials of degree $\leq d$}

2. Select one of these models based on a score (e.g. validation error).

Controlling Complexity

Examples for Different Hypotheses

Number of variables/features.

Depth of a decision tree.

Degree of a polynomial.

How about for **linear** decision functions: $x \mapsto w_1x_1 + \ldots + w_dx_d$?

ℓ_0 complexity: number of non-zero coefficients.

ℓ_1 ("lasso") complexity: $\sum |w_i|$ for coefficients $w_1, \ldots, w_d$.

ℓ_2 ("ridge") complexity: $\sum w_i^2$ for coefficients $w_1, \ldots, w_d$.

Linear (Least Squares) Regression

Running Example

Input space: $\mathcal{X} = \mathbb{R}^d$

Hypothesis class is parametrized by $w \in \mathbb{R}^d$

Output space: $\mathcal{Y} = \mathbb{R}$ Action space: $\mathcal{A} = \mathcal{Y} = \mathbb{R}$

Loss Function: $\ell(\hat{y}, y) = (\hat{y} - y)^2$

Hypothesis Class: $\mathcal{H} = \{h : \mathbb{R}^d \to \mathbb{R} : h(x) = w^\top x, w \in \mathbb{R}^d\}$

Given dataset $D_n := \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$ we want to minimize the empirical risk:

$$\hat{R}_n(w) = \frac{1}{n}\sum_{i=1}^{n}(w^\top x^{(i)} - y^{(i)})^2 \text{ or } \hat{R}_n(w) = \frac{1}{n}\|Xw - y\|^2 \text{ with } X \in \mathbb{R}^{n\times d}, y \in \mathbb{R}^n.$$

Objective in scalar form

Objective in matrix-vector form

Polynomial Regression

1D Example

For a feature $x \in \mathbb{R}$, we can always transform $x \mapsto \phi(x)$ where

$$\phi(x) = \begin{pmatrix} 1 & x & x^2 & \dots & x^d \end{pmatrix}.$$

Then, fitting a linear model atop transformed features $(\phi(x_1), y_1), \dots, (\phi(x_n), y_n)$ is a polynomial:

$$w^\top \phi(x) = w_0 + w_1 x + w_2 x^2 + \dots + w_d x^d.$$

Polynomial Regression

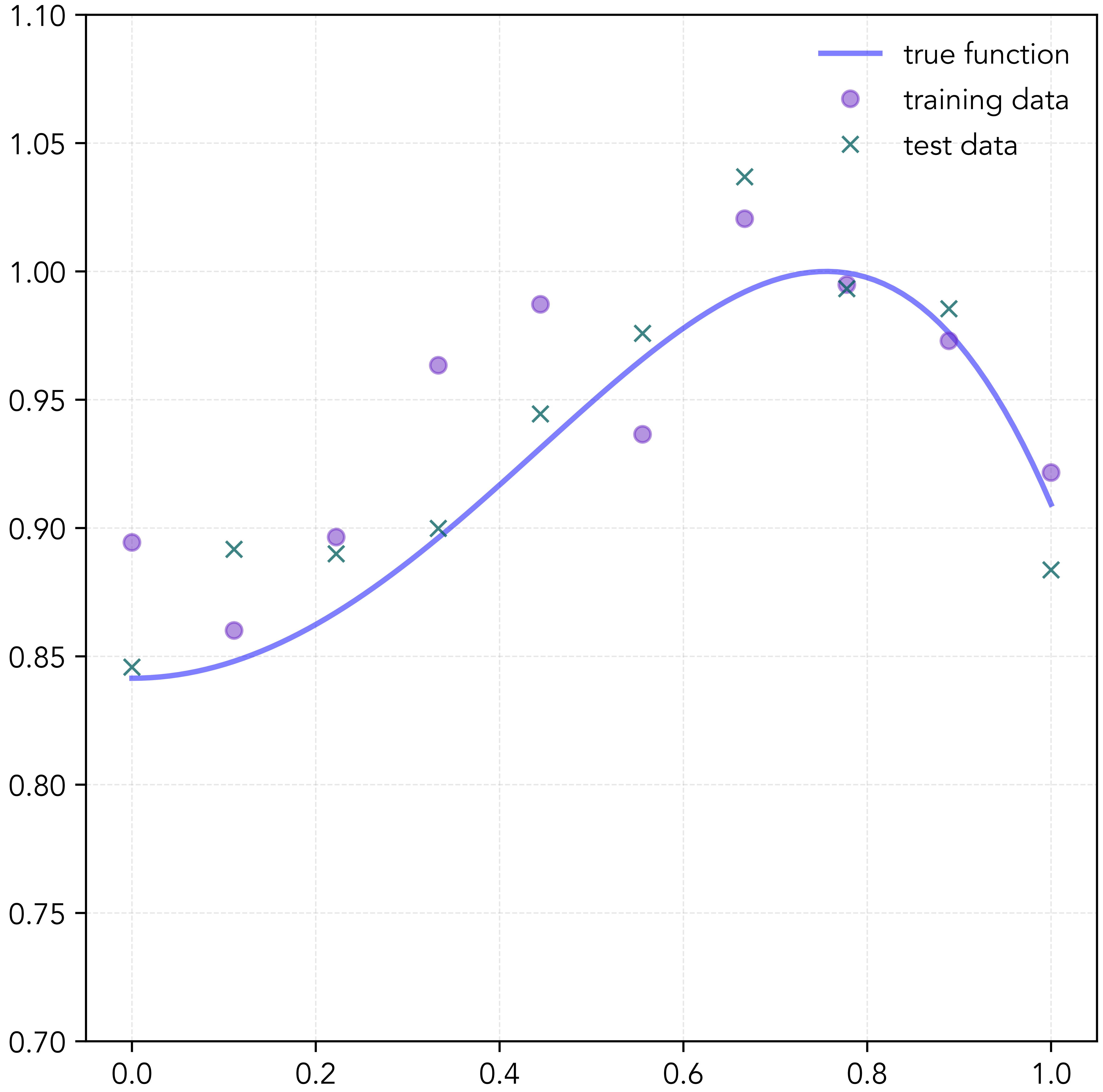
1D Example

For a feature $x \in \mathbb{R}$, we can always transform $x \mapsto \phi(x)$ where:

$$\phi(x) = \begin{pmatrix} 1 & x & x^2 & \dots & x^d \end{pmatrix}.$$

Then, fitting a linear model atop transformed features $(\phi(x_1), y_1), \dots, (\phi(x_n), y_n)$ is a polynomial:

$$w^\top \phi(x) = w_0 + w_1 x + w_2 x^2 + \dots + w_d x^d.$$



Polynomial Regression

1D Example: Degree 1

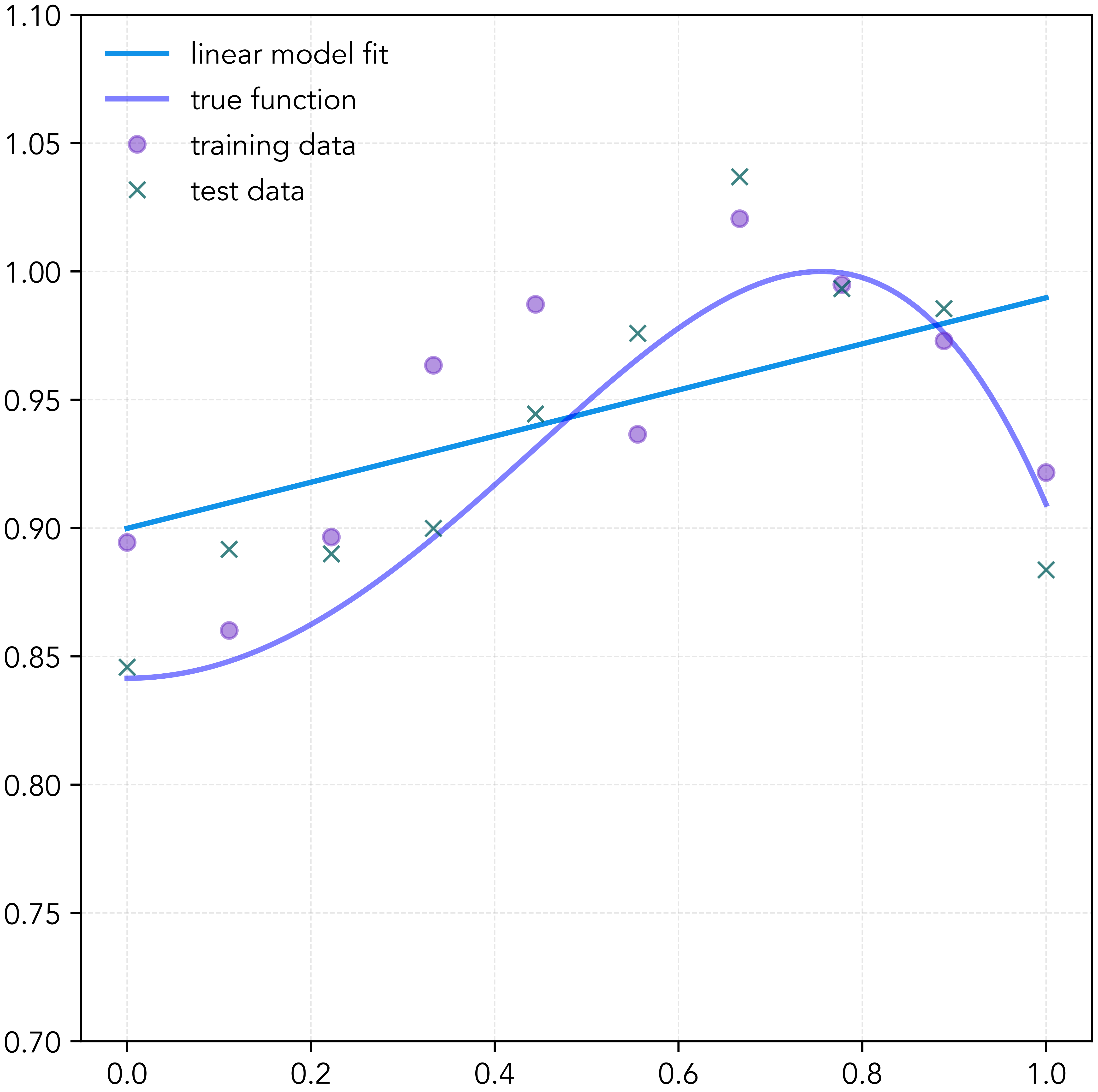
For a feature $x \in \mathbb{R}$, we can always transform $x \mapsto \phi(x)$ where:

$$\phi(x) = \begin{pmatrix} 1 & x & x^2 & \dots & x^d \end{pmatrix}.$$

$$w^\top \phi(x) = w_0 + w_1 x + w_2 x^2 + \dots + w_d x^d.$$

Fitting $d = 1$:

$$w^\top \phi(x) = w_0 + w_1 x$$



Polynomial Regression

1D Example: Degree 2

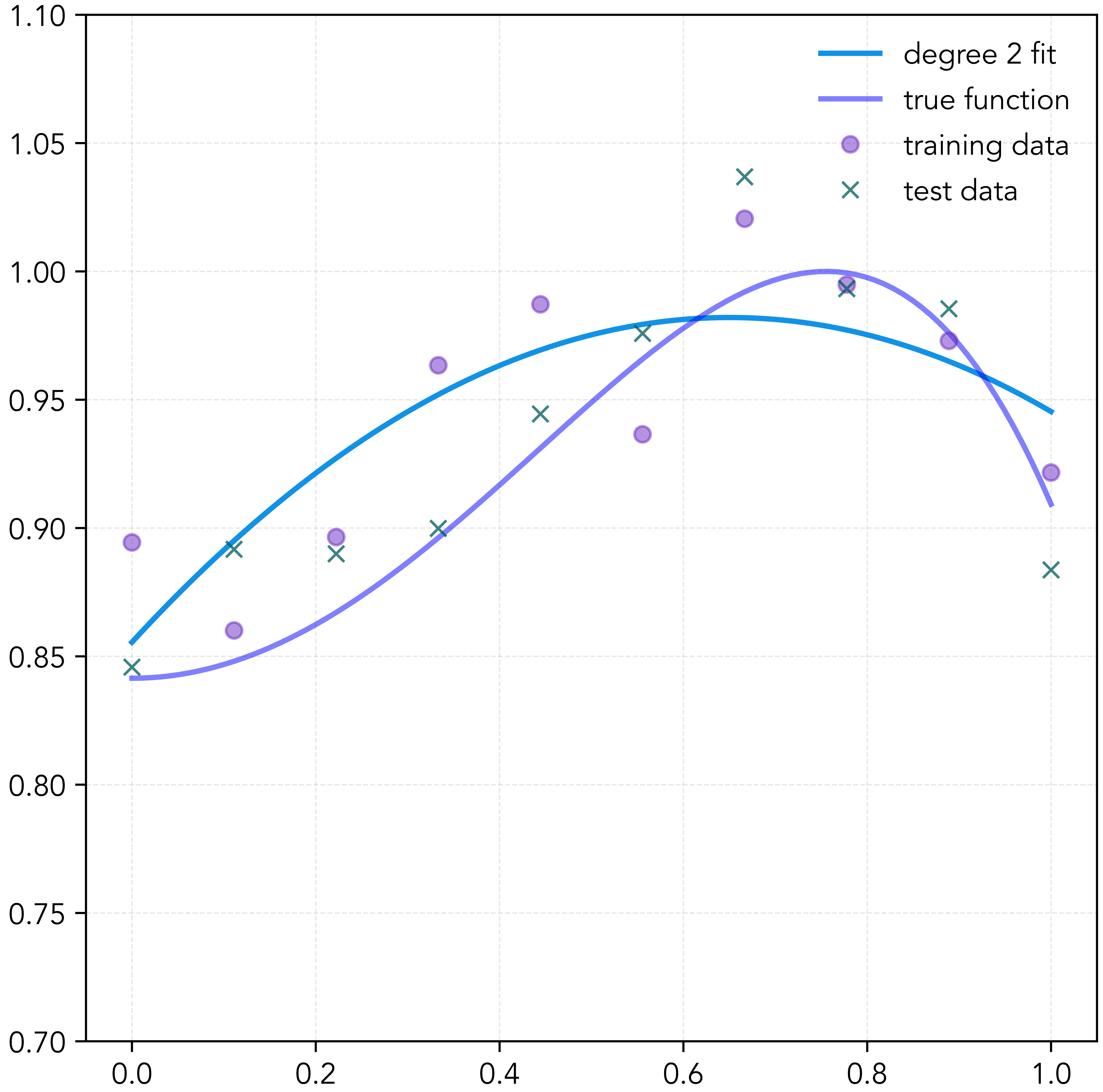
For a feature $x \in \mathbb{R}$, we can always transform $x \mapsto \phi(x)$ where:

$$\phi(x) = \begin{pmatrix} 1 & x & x^2 & \dots & x^d \end{pmatrix}.$$

$$w^\top \phi(x) = w_0 + w_1 x + w_2 x^2 + \ldots + w_d x^d.$$

Fitting $d = 2$:

$$w^\top \phi(x) = w_0 + w_1 x + w_2 x^2$$



Polynomial Regression

1D Example: Degree 3

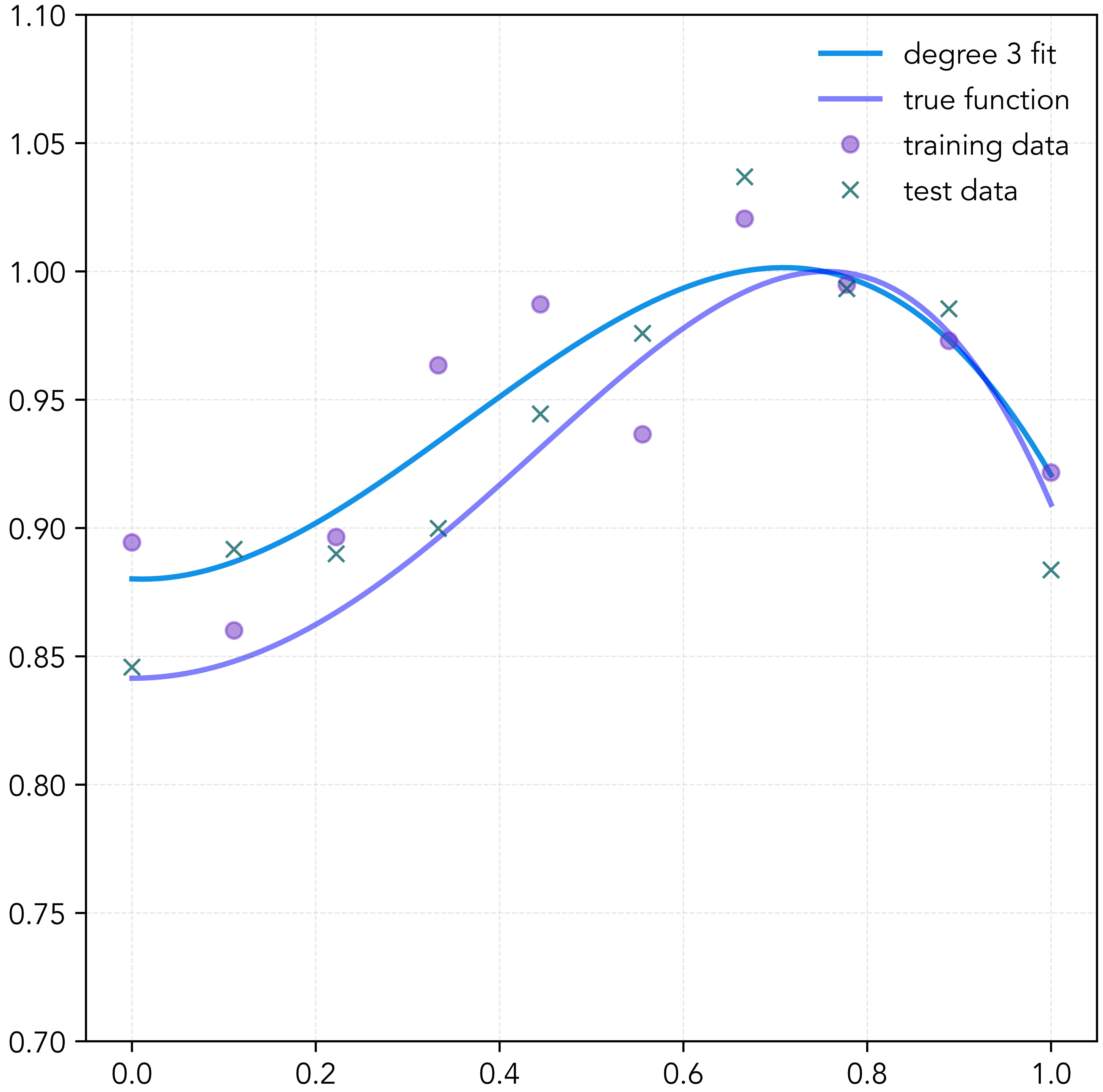
For a feature $x \in \mathbb{R}$, we can always transform $x \mapsto \phi(x)$ where:

$$\phi(x) = \begin{pmatrix} 1 & x & x^2 & \dots & x^d \end{pmatrix}.$$

$$w^\top \phi(x) = w_0 + w_1 x + w_2 x^2 + \ldots + w_d x^d.$$

Fitting $d = 3$:

$$w^\top \phi(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$



Polynomial Regression

1D Example: Degree 9

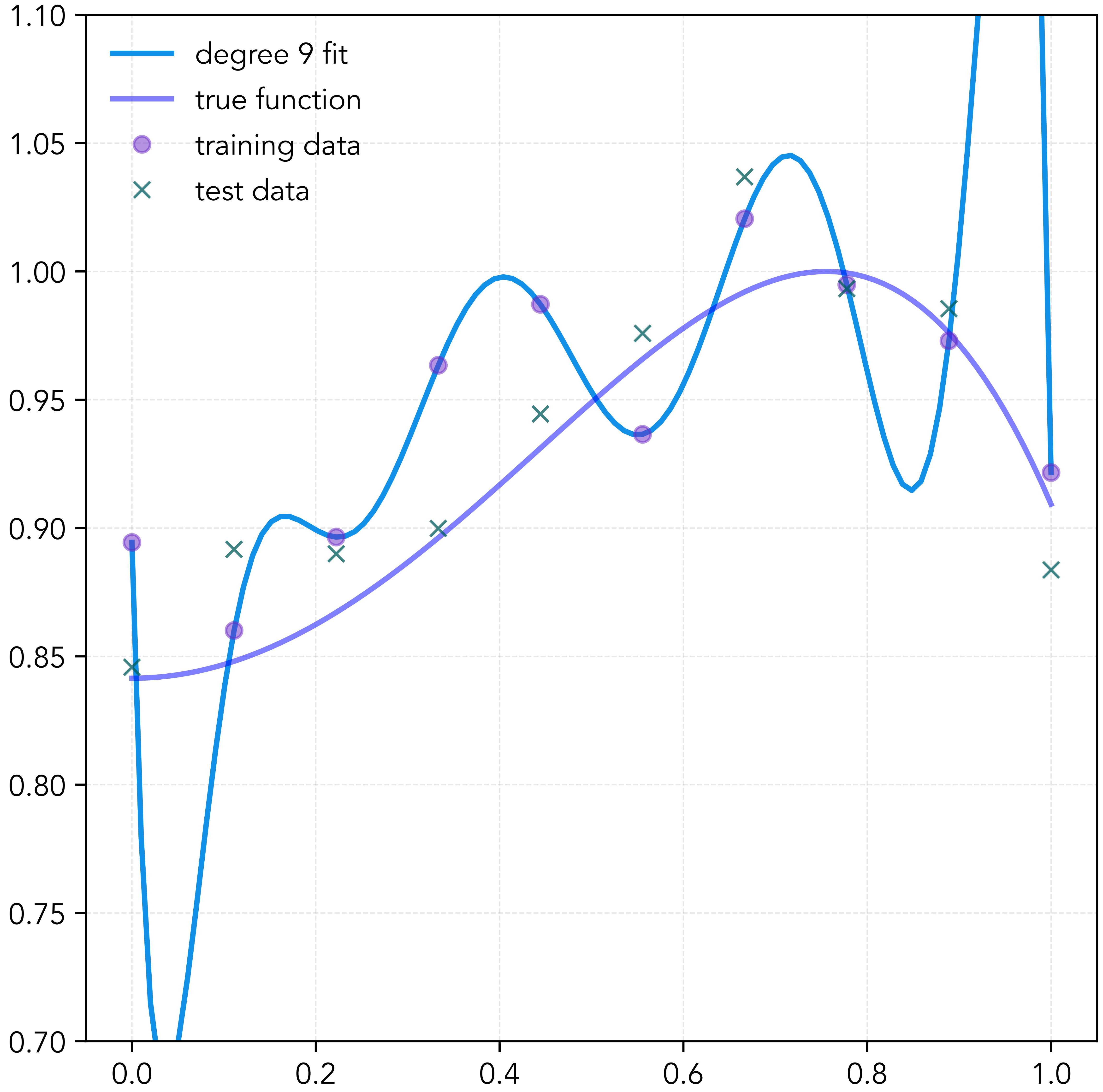
For a feature $x \in \mathbb{R}$, we can always transform $x \mapsto \phi(x)$ where:

$$\phi(x) = \begin{pmatrix} 1 & x & x^2 & \dots & x^d \end{pmatrix}.$$

$$w^\top \phi(x) = w_0 + w_1 x + w_2 x^2 + \dots + w_d x^d.$$

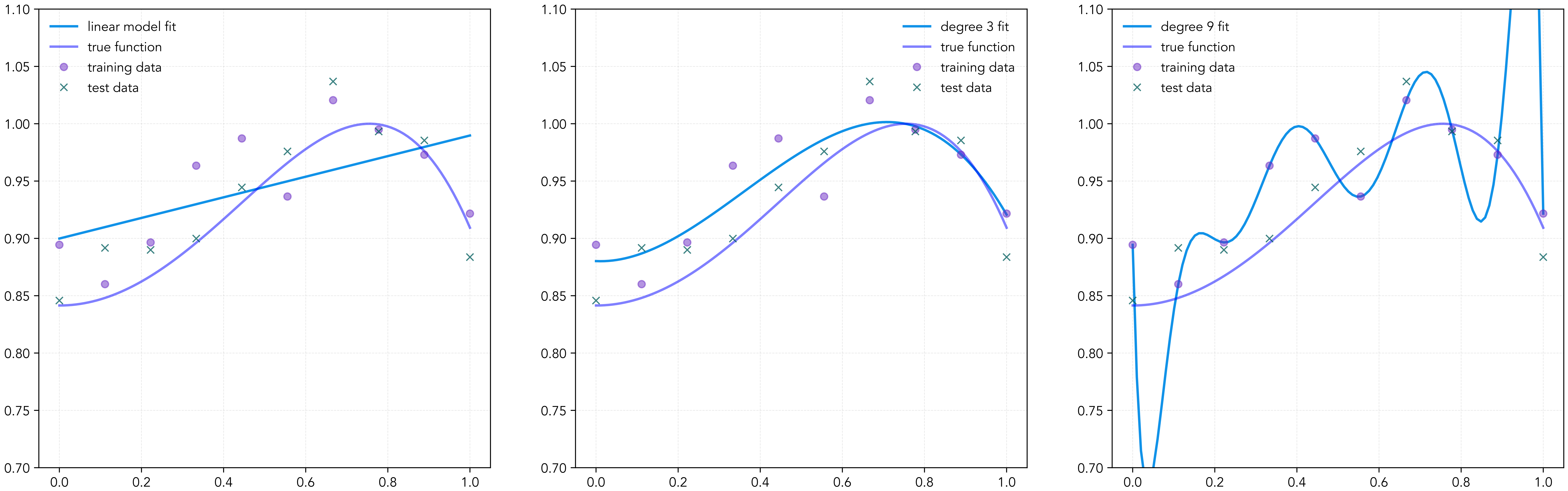
Fitting $d = 9$:

$$w^\top \phi(x) = w_0 + w_1 x + w_2 x^2 + \dots + w_{10} x^9$$



Polynomial Regression

Underfit, Just Right, Overfit



There can be an infinite number of ERMs!

Polynomial Regression

Model Selection

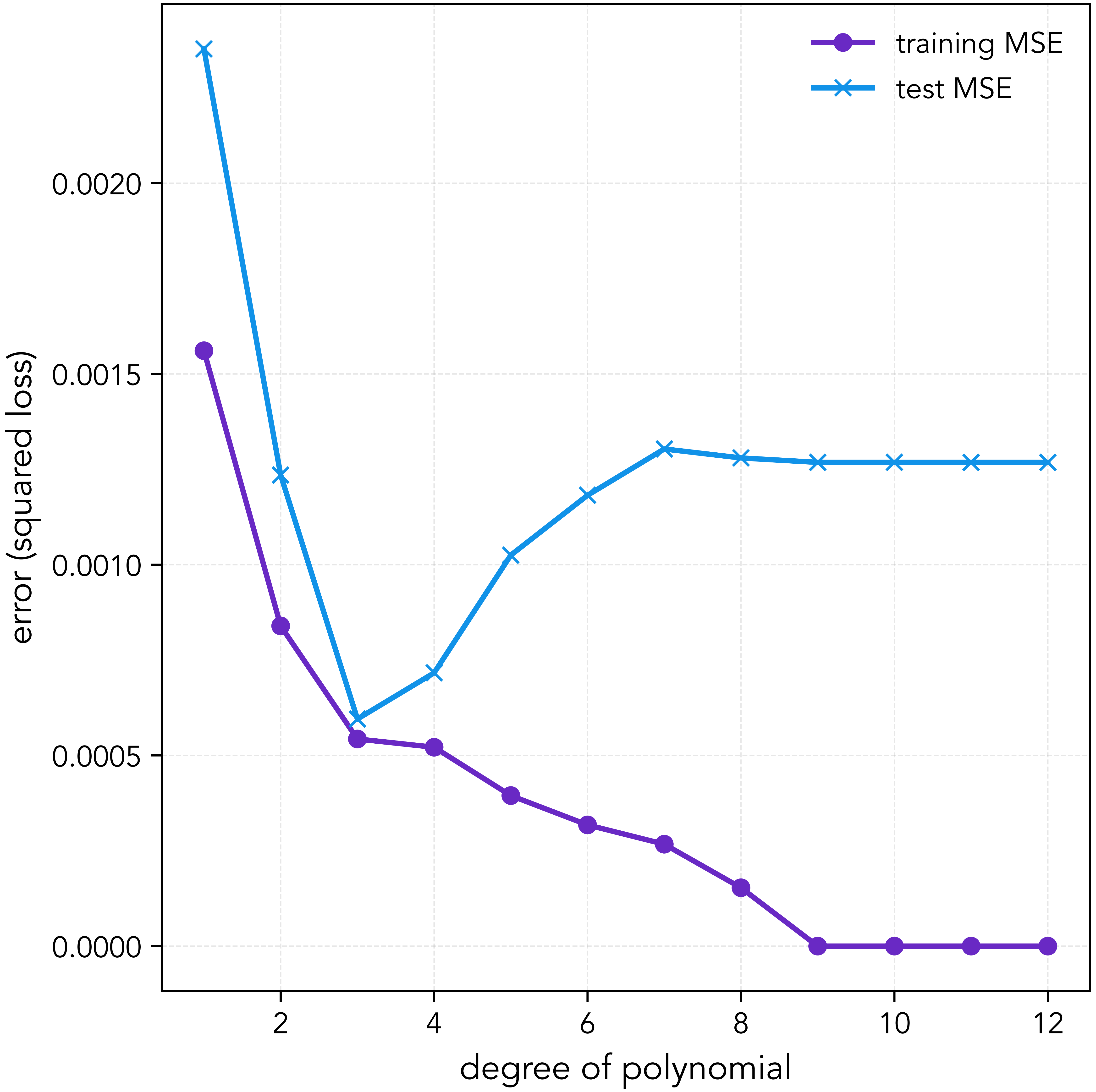
It doesn't take much to drive down empirical risk using polynomials:

$$\hat{R}_n(\hat{h}_n) \approx 0.$$

The more complex our model, the better our empirical risk during training.

Recall:

$$\hat{R}_n(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(x^{(i)}), y^{(i)})$$



Polynomial Regression

Model Selection

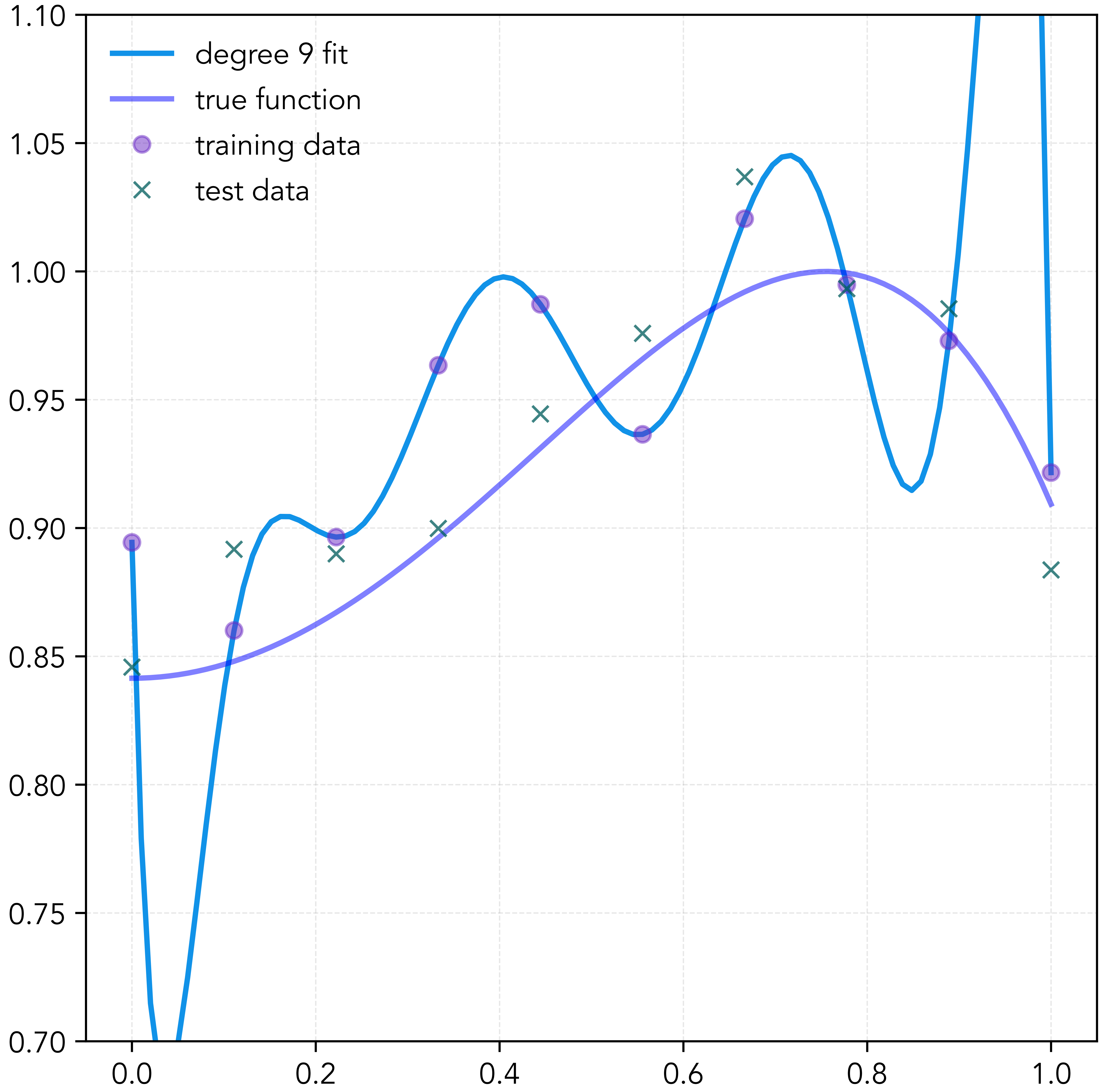
It doesn’t take much to drive down empirical risk using polynomials:

$$\hat{R}_n(\hat{h}_n) \approx 0.$$

The more complex our model, the better our empirical risk during training.

But this doesn’t mean we’ll necessarily do well on *new data*.

We call this overfitting.



Polynomial Regression

Model Selection

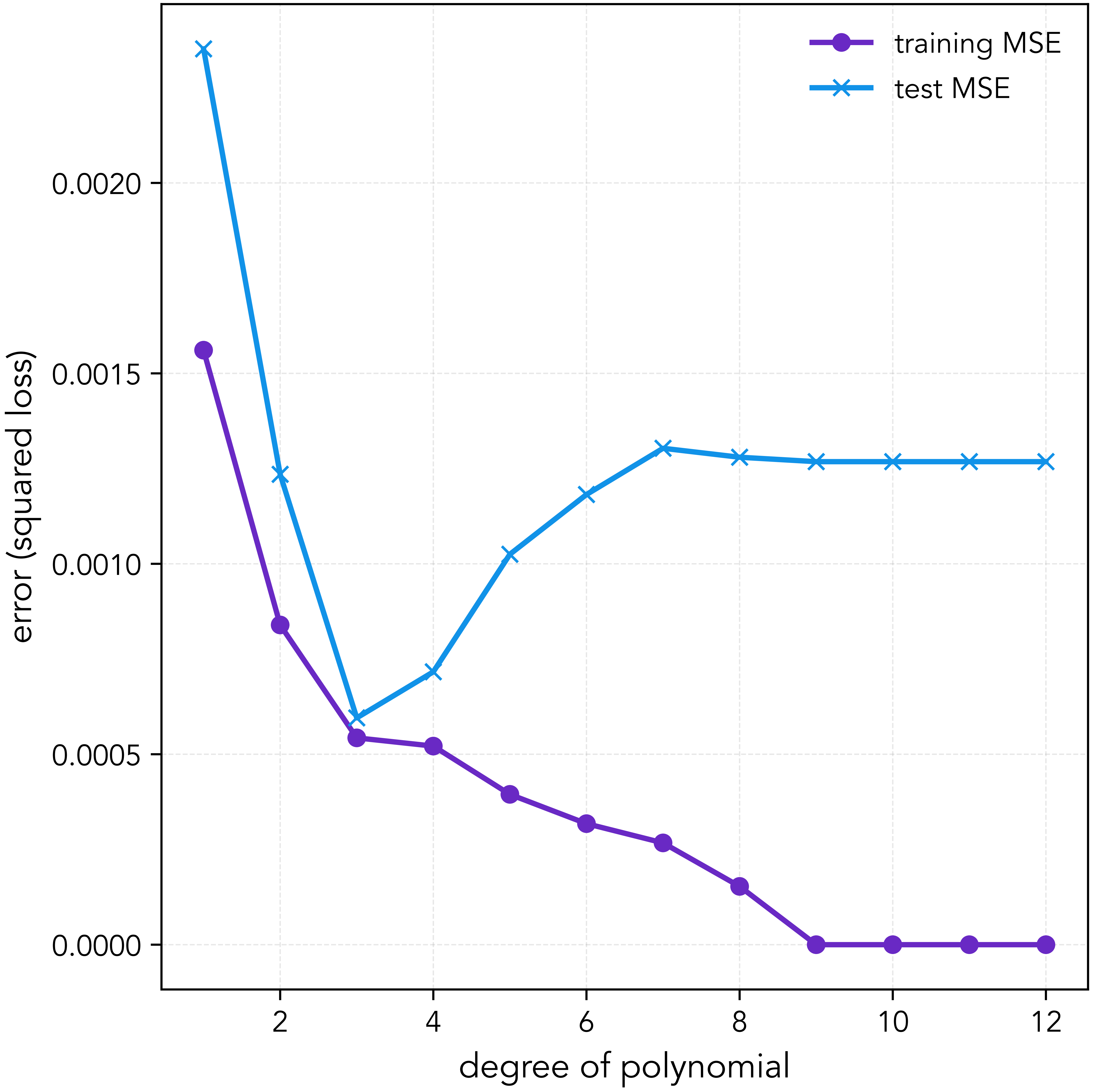
It doesn't take much to drive down empirical risk using polynomials:

$$\hat{R}_n(\hat{h}_n) \approx 0.$$

The more complex our model, the better our empirical risk during training.

But this doesn't mean we'll necessarily do well on *new data*.

We call this overfitting.



Polynomial Regression

Model Selection

In practice, we can directly test a model on "new data" (unseen in training).

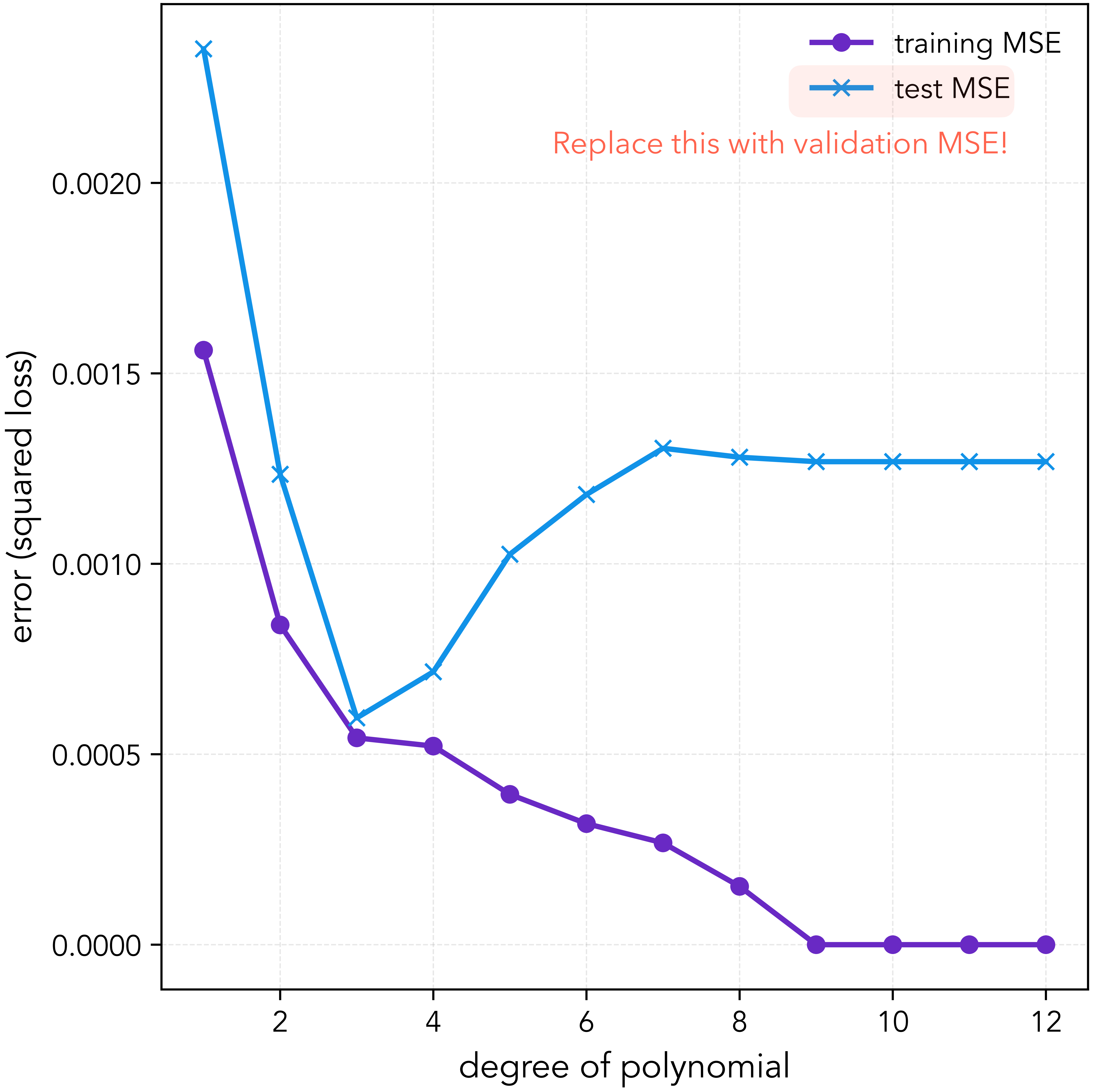
Model selection on a **validation set:**

1. Split training set D_n into **train set** D_n^1 and **validation set** D_n^2.

2. Train k models $f^{(1)}, \ldots, f^{(k)}$ of varying complexity using D_n^1.

3. Evaluate each of $f^{(1)}, \ldots, f^{(k)}$ on D_n^2.

4. Pick model with lowest validation loss.



Replace this with validation MSE!

Polynomial Regression

Model Selection

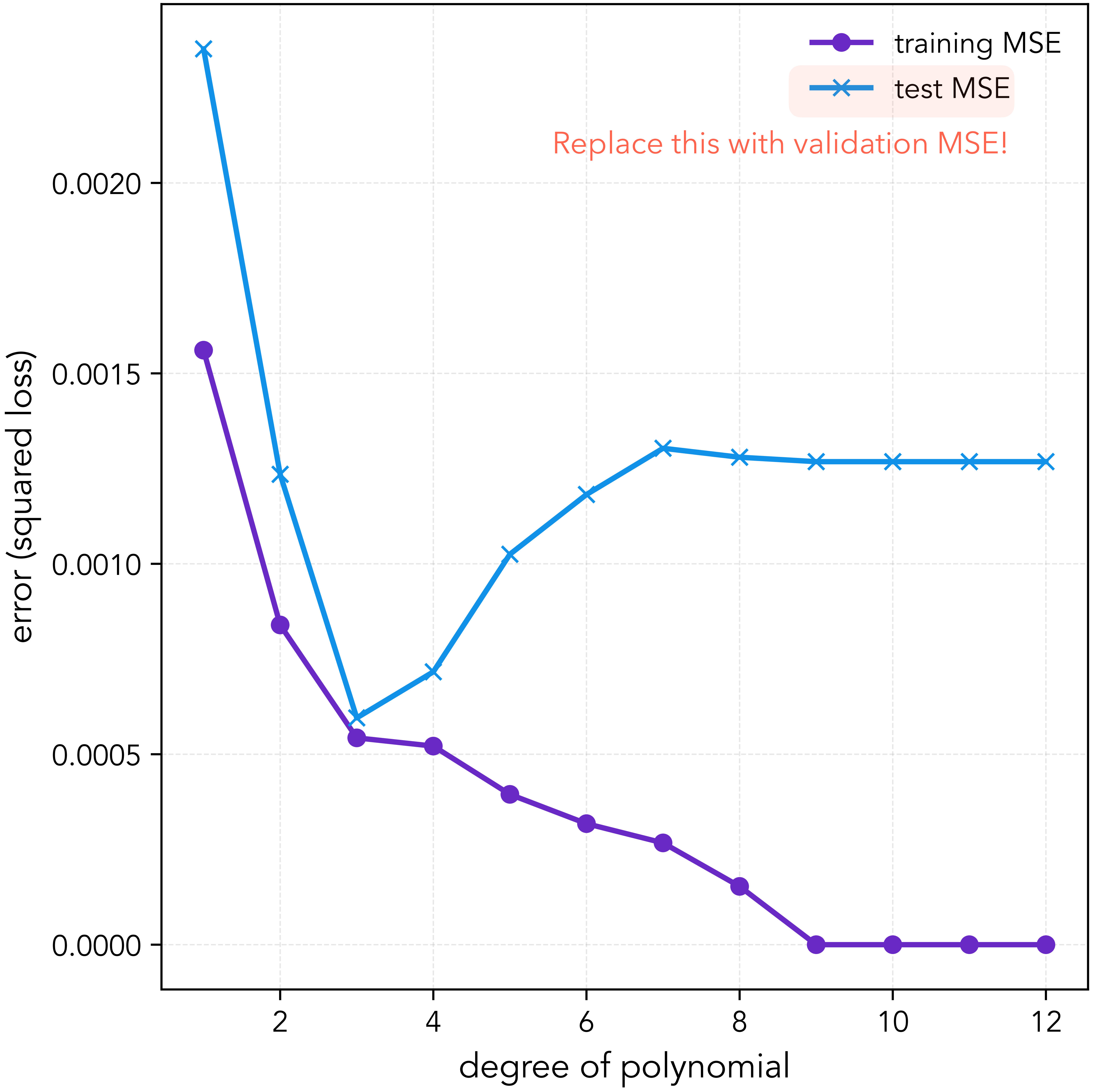
Split training set D_n into **train set** $D_n^1$and **validation set** D_n^2

$$D_n^1 = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$$

$$D_n^2 = \{(\tilde{x}^{(1)}, \tilde{y}^{(1)}), \ldots, (\tilde{x}^{(m)}, \tilde{y}^{(m)})\}$$

As long as $\hat{h}_n$ is chosen without looking at D_n^2, we have an estimate of **true risk**.

$$\frac{1}{n}\sum_{j=1}^{n} \ell(h(\tilde{x}^{(j)}), \tilde{y}^{(j)}) \approx \mathbb{E}[\ell(h(x), y)] = R(h)$$



Replace this with validation MSE!

Outline

Model Complexity and Model Selection

Controlling Complexity with Regularization

ℓ_2 Regularization and Ridge Regression

ℓ_1 Regularization and Lasso Regression

Understanding Sparsity

Loss Functions: Regression

Loss Functions: Classification

Controlling Complexity

Feature Selection in Linear Regression

ℓ_0 complexity: number of non-zero coefficients.

$$\mathscr{H}_1 \subset \mathscr{H}_2 \subset \ldots \subset \mathscr{H}_n \subset \mathscr{H}$$

Example: Linear Functions

$\mathscr{H}$ = { linear functions using all features}

$\mathscr{H}_d$ = { linear functions using fewer than d features }

Best subset selection: Choose subset of features that is best according to score (e.g. validation error). Example with two features: train models using $\{\}$, $\{x_1\}$, $\{x_2\}$ and $\{x_1, x_2\}$.

This is not an efficient search algorithm! 2^d subsets if total of d features.

Controlling Complexity

Feature Selection in Linear Regression

Best subset selection: Choose subset of features that is best according to score (e.g. validation error). Example with two features: train models using $\{\}$, $\{x_1\}$, $\{x_2\}$ and $\{x_1, x_2\}$.

Objective that balances number of feature with performance:

$$\text{score}(S) = \text{train\_loss}(S) + \lambda \left| S \right|$$

λ balances the training loss and the number of features used.

Adding an extra feature must be justified with λ improvement in training loss.

Larger λ: complex models penalized more heavily.

Complexity Penalty

Constrained ERM (Ivanov Regularization)

Goal: Balance the complexity of the hypothesis class $\mathscr{H}$ and its training loss.

For complexity measure $\Omega : \mathscr{H} \to [0,\infty)$ and fixed $r \geq 0$, the constrained ERM problem is:

$$\begin{aligned} \min_{h \in \mathscr{H}} \quad & \frac{1}{n} \sum_{i=1}^{n} \ell(h(x^{(i)}), y^{(i)}) \\ \text{s.t.} \quad & \Omega(f) \leq r \end{aligned}$$

Find r using performance on validation data.

Each r corresponds to different hypothesis classes. Could also write: $\min_{h \in \mathscr{H}_r} \frac{1}{n} \sum_{i=1}^{n} \ell(h(x^{(i)}), y^{(i)})$

Complexity Penalty

Penalized ERM (Tikhonov Regularization)

Goal: Balance the complexity of the hypothesis class $\mathscr{H}$ and its training loss.

For complexity measure $\Omega : \mathscr{H} \to [0,\infty)$ and fixed $r \geq 0$, the penalized ERM problem is:

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

Find λ using performance on validation data.

Penalized vs. Constrained Optimization

In General

Let $L : \mathscr{H} \to \mathbb{R}$ be *any* performance measure of h (e.g. empirical risk).

For many L and Ω, constrained and penalized regularization are "equivalent":

For any $r > 0$, $h_r^* \in \arg\min_{h \in \mathscr{H}} L(h)$ s.t. $\Omega(h) \leq r$ is in $\arg\min_{h \in \mathscr{H}} L(h) + \lambda\Omega(h)$ for some $\lambda > 0$.

For any $\lambda > 0$, $h_\lambda^* \in \arg\min_{h \in \mathscr{H}} L(h) + \lambda\Omega(h)$ is in $\arg\min_{h \in \mathscr{H}} L(h)$ s.t. $\Omega(h) \leq r$ for some $r > 0$.

In practice, both approaches are effective.

Penalized regularization convenient because it's an *unconstrained* optimization problem.

Complexity Penalty

Penalized ERM (Tikhonov Regularization)

Goal: Balance the complexity of the hypothesis class $\mathscr{H}$ and training loss.

For complexity measure $\Omega : \mathscr{H} \to [0,\infty)$ and fixed $r \geq 0$, the penalized ERM problem is:

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

Setting $\Omega(\cdot)$ as "number of features" is not differentiable and hard to optimize.

What other measures of complexity can we use?

Outline

Model Complexity and Model Selection

Controlling Complexity with Regularization

ℓ_2 Regularization and Ridge Regression

ℓ_1 Regularization and Lasso Regression

Understanding Sparsity

Loss Functions: Regression

Loss Functions: Classification

“Soft” Selection

Linear Regression

Input space: $\mathcal{X} = \mathbb{R}^d$

Output space: $\mathcal{Y} = \mathbb{R}$

Loss Function: $\ell(\hat{y}, y) = (\hat{y} - y)^2$

Hypothesis Class: $\mathcal{H} = \{h : \mathbb{R}^d \to \mathbb{R} : h(x) = w^\top x, w \in \mathbb{R}^d\}$

Imagine having a weight for each feature dimension.

In linear regression, model weights multiply each feature dimension.

If w_i is close to zero, then it means we aren’t using feature i.

Linear Regression

Running Example

Input space: $\mathcal{X} = \mathbb{R}^d$; Output space: $\mathcal{Y} = \mathbb{R}$; Loss Function: $\ell(\hat{y}, y) = (\hat{y} - y)^2$

Hypothesis Class: $\mathcal{H} = \{h : \mathbb{R}^d \to \mathbb{R} : h(x) = w^\top x, w \in \mathbb{R}^d\}$

Given dataset $D_n := \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$ we want to minimize the empirical risk:

$$\hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2$$

This often overfits, especially when $d \gg n$!

(e.g. in NLP one can have 1M features for 10K documents)

Ridge Regression

Constrained and Penalized ERM

The (penalized form) ridge regression solution with regularization parameter $\lambda \geq 0$ is

$$\hat{w} \in \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2 + \lambda \|w\|_2^2$$

where $\|w\|_2^2 = w_1^2 + \ldots + w_d^2$ is the squared ℓ_2-norm.

The (constrained form) ridge regression solution with regularization parameter $r^2 \geq 0$ is

$$\hat{w} \in \arg\min_{\|w\|_2^2 \leq r^2} \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2$$

Ridge Regression

Penalized ERM

The (penalized form) ridge regression solution with regularization parameter $\lambda \geq 0$ is

$$\hat{w} \in \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2 + \lambda \|w\|_2^2$$

where $\|w\|^2 = w_1^2 + \ldots + w_d^2$ is the squared ℓ_2-norm.

Equivalent to linear least squares regression with $\lambda = 0$.

ℓ_2 regularization can be used for other models too (e.g. neural networks).

Sensitivity to Inputs

Effect of Regularization

$h(x) = \hat{w}^\top x$ is Lipschitz continuous with Lipschitz constant $L = \|\hat{w}\|_2$: when moving from x to $x + \Delta$, h changes no more than $L\|\Delta\|$, because:

$$\begin{aligned} \left| h(x+\Delta) - h(x) \right| &= \left| \hat{w}^\top (x+\Delta) - \hat{w}^\top x \right| \\ &= \left| \hat{w}^\top \Delta \right| \leq \|\hat{w}\| \, \|\Delta\| \end{aligned}$$

Cauchy-Schwarz Inequality

So ℓ_2 regularization controls the maximum rate of change of h.

Other norms also provide a bound on L due to equivalence of norms:

For any p, $\exists C > 0$ such that $\|\hat{w}\|_2 \leq C\|\hat{w}\|_p$.

Linear vs. Ridge Regression

Analytical Comparison

Linear objective: $\frac{1}{2}\sum_{i=1}^{n}(w^\top x^{(i)} - y^{(i)})^2$

in matrix-vector form: $\frac{1}{2}\|Xw - y\|_2^2$

Gradient: $\nabla L(w) = X^\top(Xw - y)$

Closed-form solution:

$\hat{w} = (X^\top X)^{-1}X^\top y$ if X is full rank.

Ridge objective: $\frac{1}{2}\sum_{i=1}^{n}(w^\top x^{(i)} - y^{(i)})^2 + \frac{\lambda}{2}\|w\|_2^2$

in matrix-vector form: $\frac{1}{2}\|Xw - y\|_2^2 + \frac{\lambda}{2}\|w\|_2^2$

Gradient: $\nabla L(w) = X^\top(Xw - y) + \lambda w$

Closed-form solution:

$\hat{w} = (X^\top X + \lambda I)^{-1}X^\top y$

($X^\top X + \lambda I$ is always invertible)

Ridge Regression

Constrained and Penalized ERM

The (penalized form) ridge regression solution with regularization parameter $\lambda \geq 0$ is

$$\hat{w} \in \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2 + \lambda \|w\|_2^2$$

where $\|w\|^2 = w_1^2 + \ldots + w_d^2$ is the squared ℓ_2-norm.

The (constrained form) ridge regression solution with regularization parameter $r^2 \geq 0$ is

$$\boxed{\hat{w} \in \arg\min_{\|w\|_2^2 \leq r^2} \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2}$$

Linear vs. Ridge

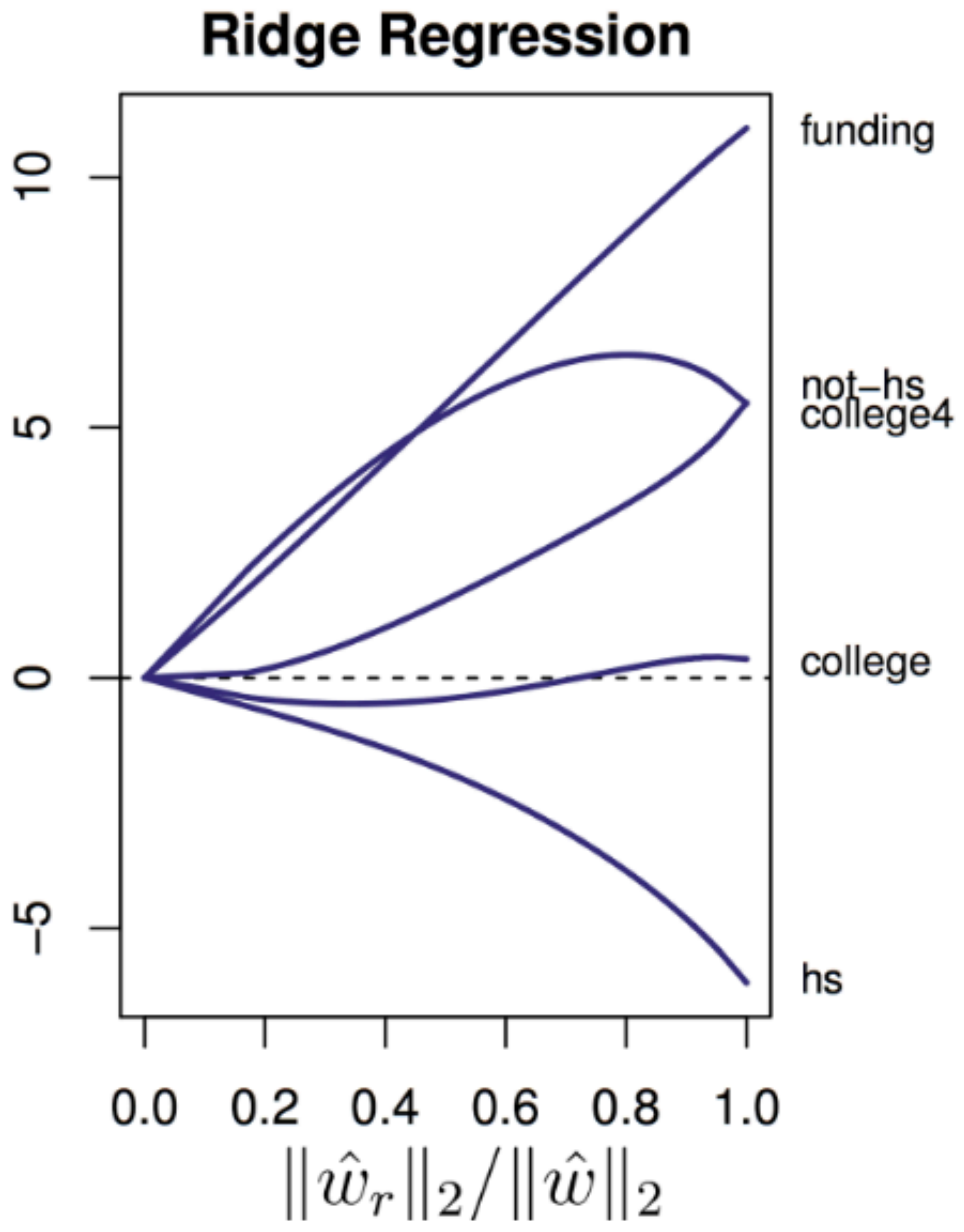
Regularization Path Comparison

$$\hat{w}_r \in \underset{\|w\|_2^2 \le r^2}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2$$

$$\hat{w} \in \underset{w \in \mathbb{R}^d}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2$$

For $r = 0$, $\|\hat{w}_r\|_2 / \|\hat{w}\|_2 = 0$.

For $r = \infty$, $\|\hat{w}_r\|_2 / \|\hat{w}\|_2 = 1$.



Modified from Hastie, Tibshirani, and Wainwright's *Statistical Learning with Sparsity*, Fig 2.1. About predicting crime in 50 US cities.

Outline

Model Complexity and Model Selection

Controlling Complexity with Regularization

ℓ_2 Regularization and Ridge Regression

ℓ_1 Regularization and Lasso Regression

Understanding Sparsity

Loss Functions: Regression

Loss Functions: Classification

Lasso Regression

Constrained and Penalized ERM

The (penalized form) lasso regression solution with regularization parameter $\lambda \geq 0$ is

$$\hat{w} \in \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2 + \lambda \|w\|_1$$

where $\|w\|_1 = \left|w_1\right| + \ldots + \left|w_d\right|$ is the ℓ_1-norm.

The (constrained form) lasso regression solution with regularization parameter $r \geq 0$ is

$$\hat{w} \in \arg\min_{\|w\|_1 \leq r} \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2$$

Linear vs. Ridge

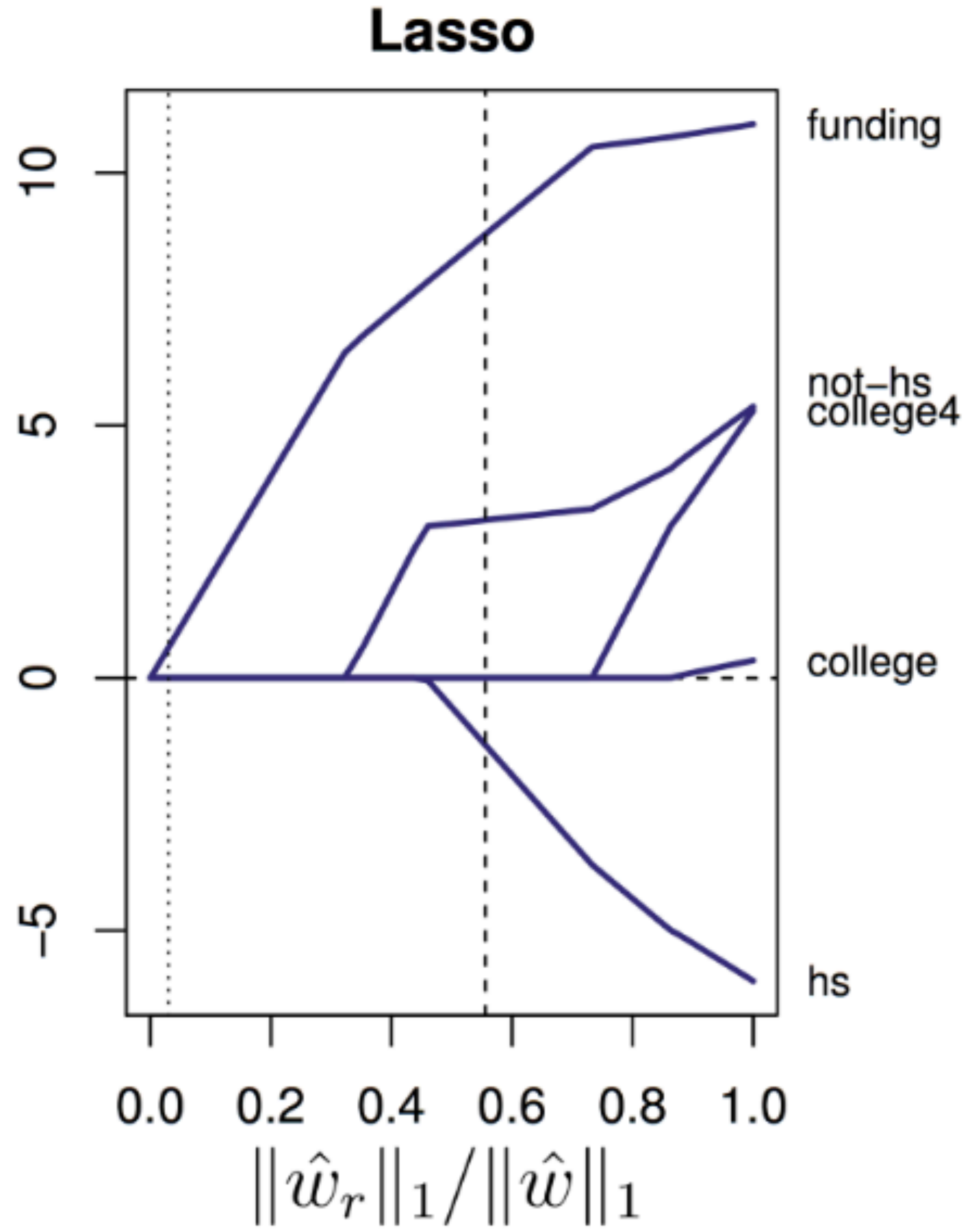
Regularization Path Comparison

$$\hat{w}_r \in \arg\min_{\|w\|_1 \le r} \frac{1}{n}\sum_{i=1}^{n}(w^\top x^{(i)} - y^{(i)})^2$$

$$\hat{w} \in \arg\min_{w \in \mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n}(w^\top x^{(i)} - y^{(i)})^2$$

For $r = 0$, $\|\hat{w}_r\|_1/\|\hat{w}\|_1 = 0.$

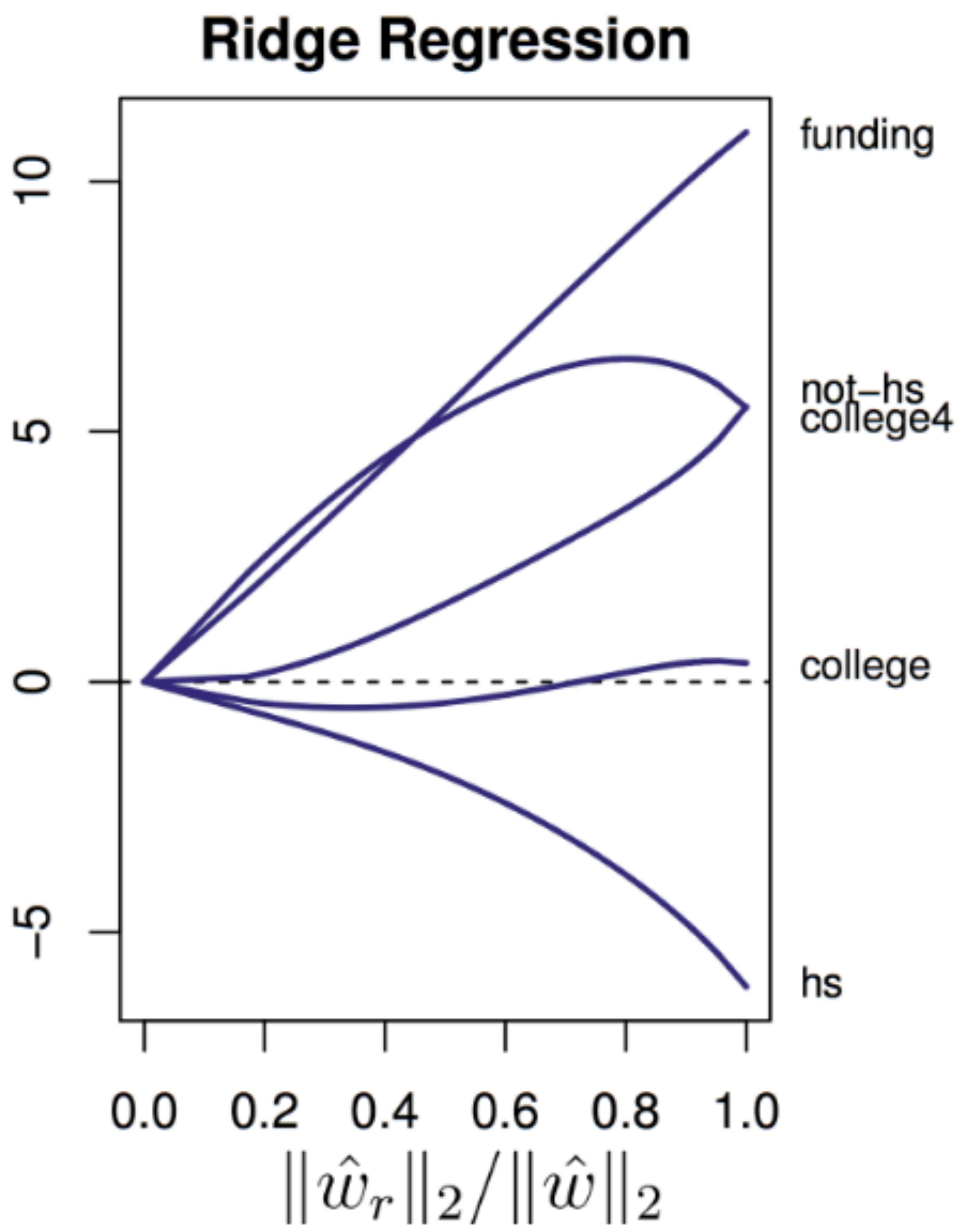
For $r = \infty$, $\|\hat{w}_r\|_1/\|\hat{w}\|_1 = 1.$

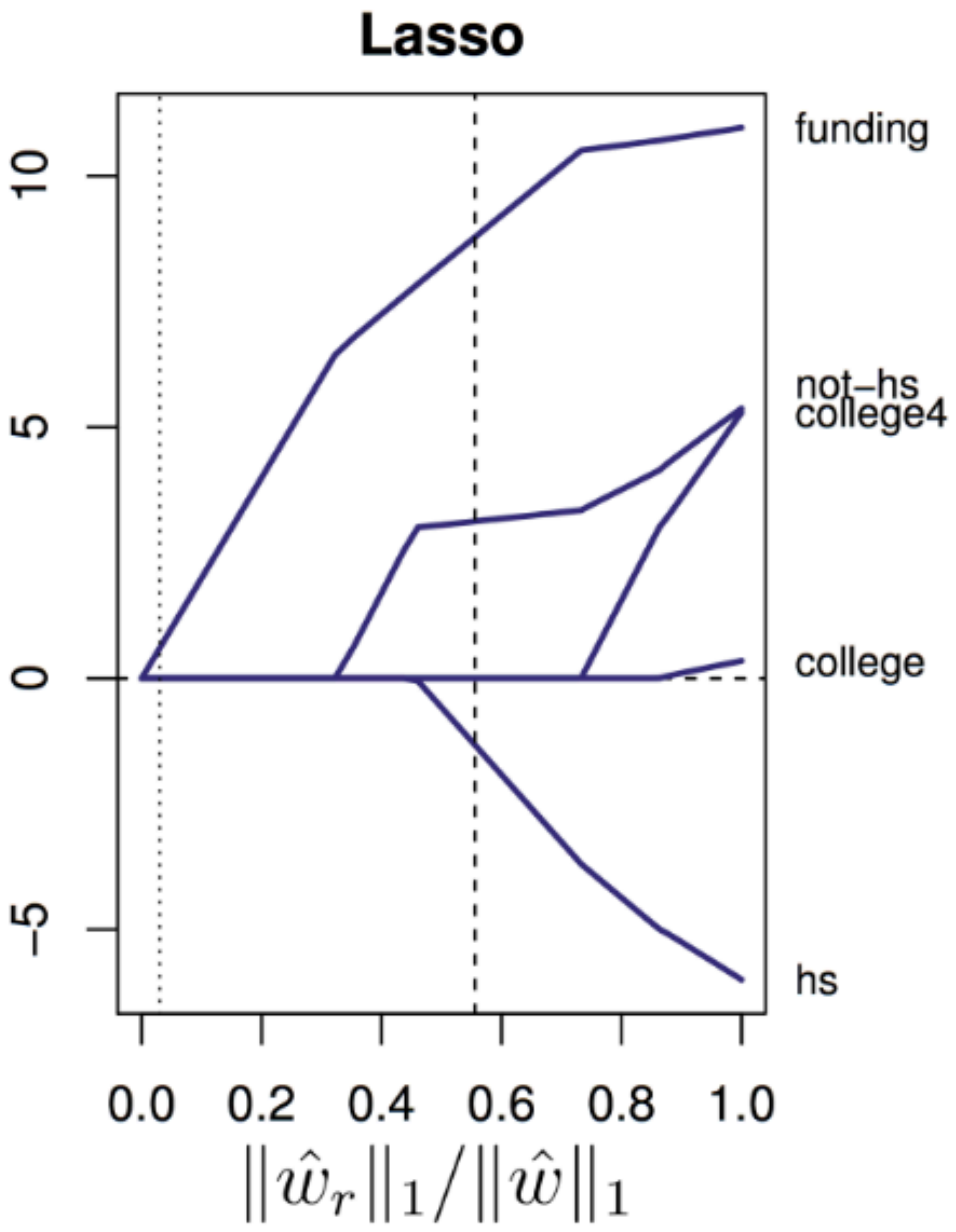


Modified from Hastie, Tibshirani, and Wainwright's *Statistical Learning with Sparsity*, Fig 2.1. About predicting crime in 50 US cities.

Lasso vs. Ridge

Regularization Path Comparison





Lasso Regression

Pros and Cons

Pros:

Output weights are *sparse* which can mean a more interpretable model.

More intuitive reduction in model complexity.

Cons:

No closed form solution because $\|w\|_1$ is not differentiable (unlike ridge regression).

Can solve Lasso with iterative methods, but generally not as quickly as ridge regression.

Outline

Model Complexity and Model Selection

Controlling Complexity with Regularization

ℓ_2 Regularization and Ridge Regression

ℓ_1 Regularization and Lasso Regression

Understanding Sparsity

Loss Functions: Regression

Loss Functions: Classification

Lasso Regression

Benefits of Sparsity

A **sparse** solution $\hat{w}$ is one in which many entries are 0. Why is this useful?

Faster to compute features; cheaper to measure or annotate them.

Less memory to store features (deployment on mobile device).

Interpretability: identifies the important features.

Prediction function may **generalize** better (model is less complex, i.e. $\mathscr{H}$ is “smaller”).

Parameter Space

Intuition

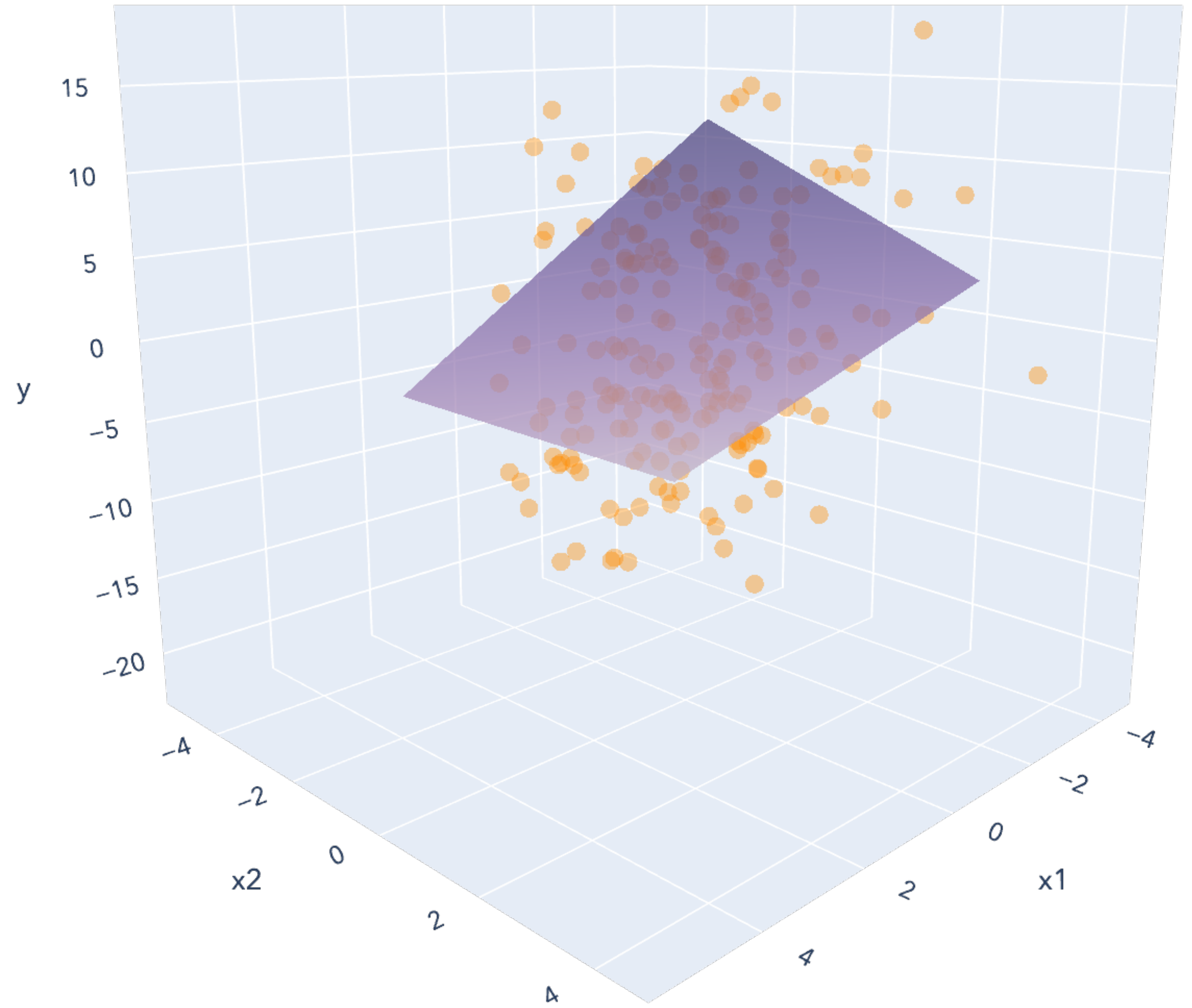
To visualize, suppose $\mathscr{X} = \mathbb{R}^2$.

$$\hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2$$

Geometrically,

$w^\top x = w_1 x_1 + w_2 x_2$ is a plane through the origin in $\mathbb{R}^3$.

$\hat{R}_n(w) : \mathbb{R}^2 \to \mathbb{R}$ is a loss surface in $\mathbb{R}^3$ for every possible plane.



Parameter Space

Intuition

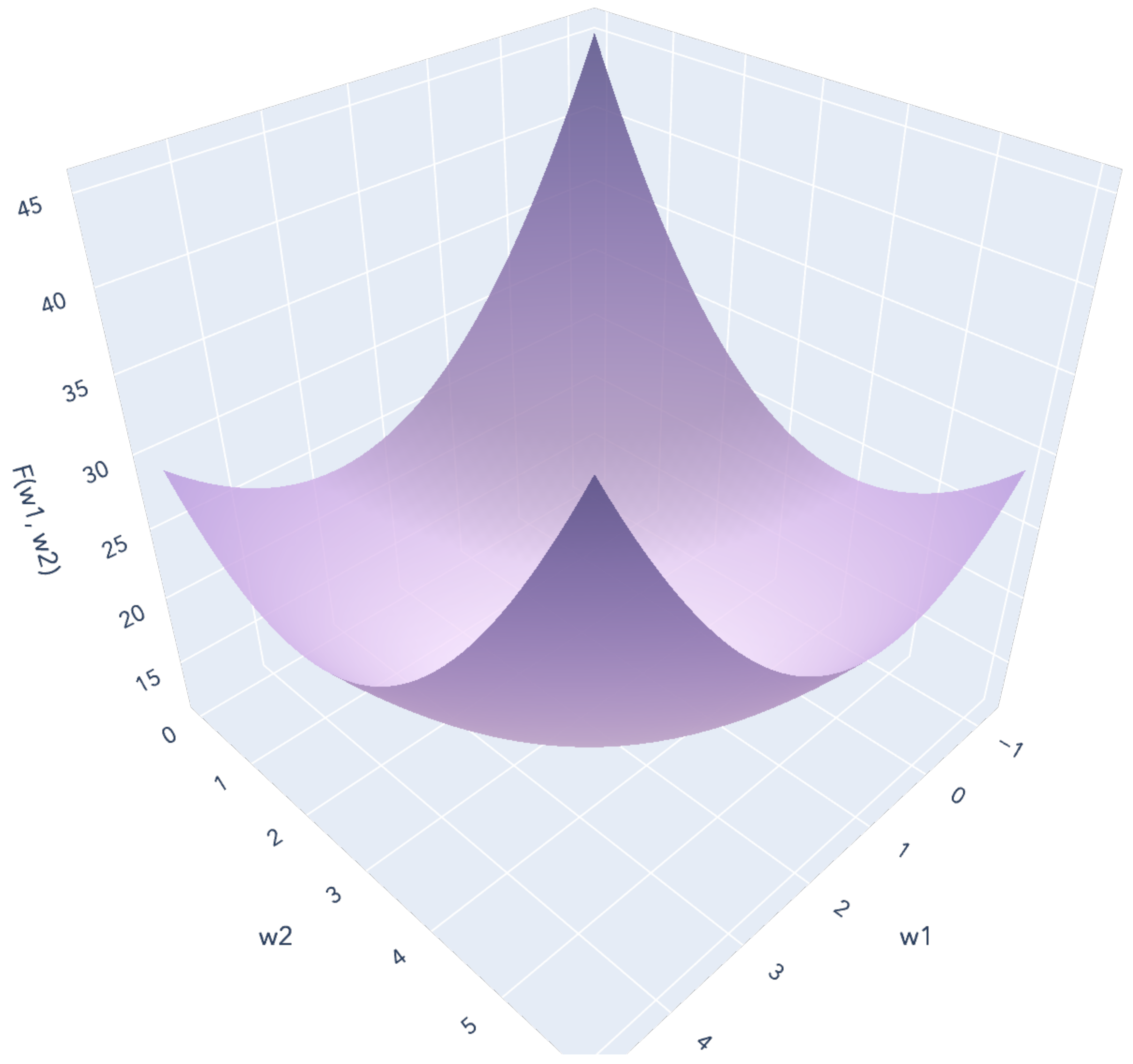
To visualize, suppose $\mathscr{X} = \mathbb{R}^2$.

$$\hat{R}_n(w) = \frac{1}{n}\sum_{i=1}^{n}(w^\top x^{(i)} - y^{(i)})^2$$

Geometrically,

$w^\top x = w_1x_1 + w_2x_2$ is a plane through the origin in $\mathbb{R}^3$.

$\hat{R}_n(w) : \mathbb{R}^2 \to \mathbb{R}$ is a loss surface in $\mathbb{R}^3$ for every possible plane.



Parameter Space

Intuition

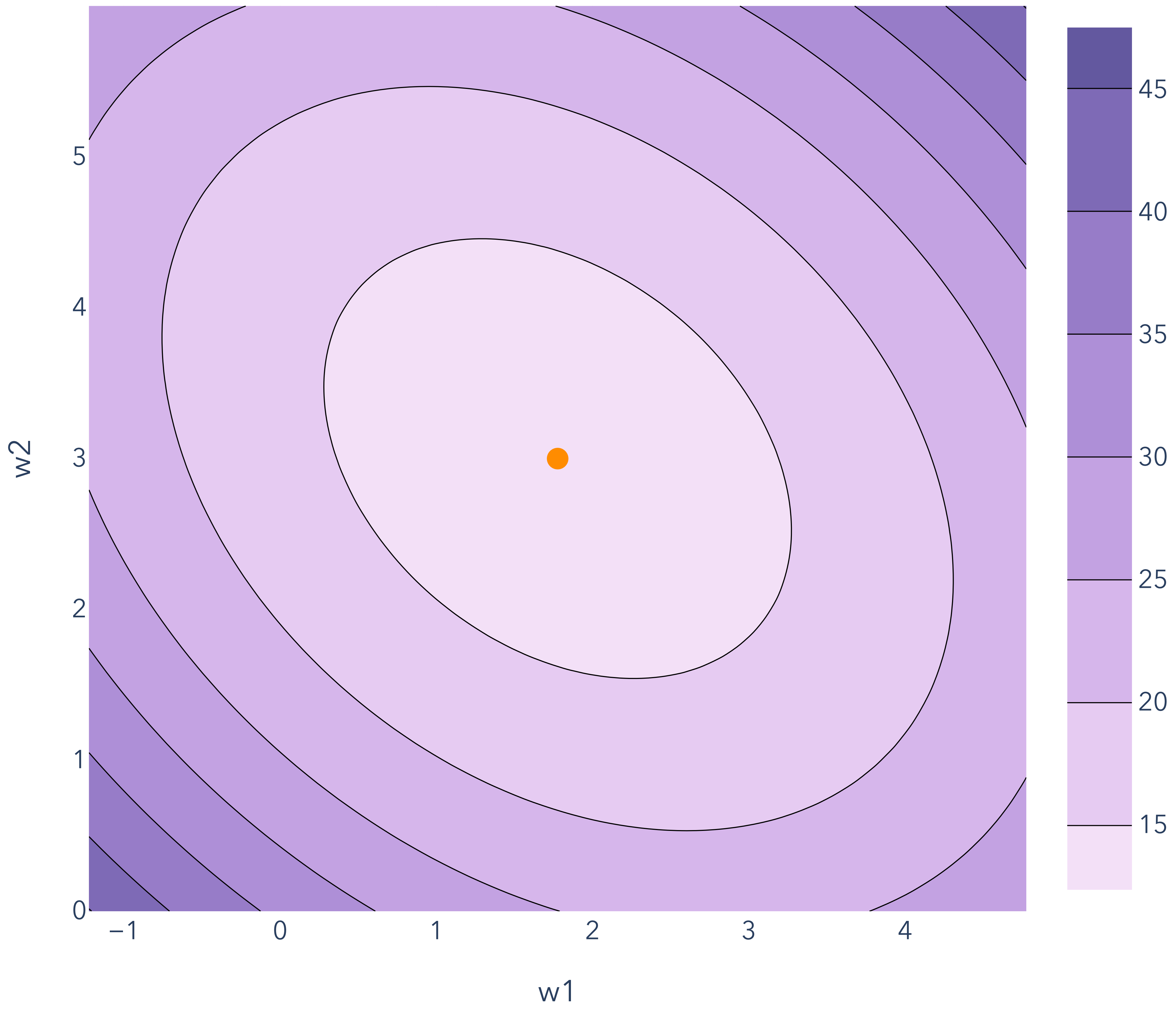
To visualize, suppose $\mathscr{X} = \mathbb{R}^2$.

$$\hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2$$

Geometrically,

$w^\top x = w_1 x_1 + w_2 x_2$ is a plane through the origin in $\mathbb{R}^3$.

$\hat{R}_n(w) : \mathbb{R}^2 \to \mathbb{R}$ is a loss surface in $\mathbb{R}^3$ for every possible plane.



ℓ_1 and ℓ_2 Constraints

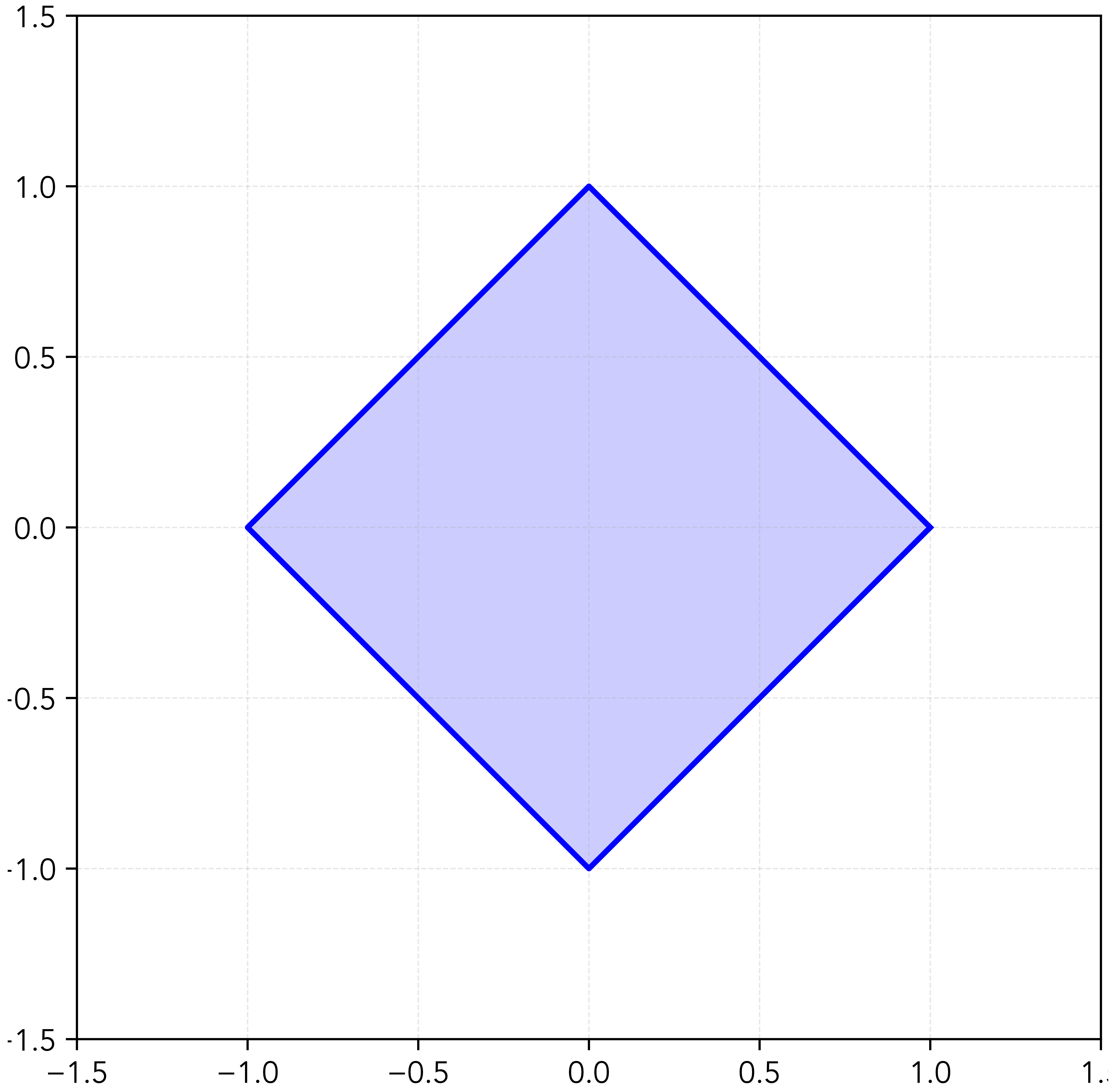
Intuition

For visualization, restrict to:

$$\mathscr{H} = \{h(x) = w_1x_1 + w_2x_2\}$$

Represent $\mathscr{H}$ by $\{(w_1, w_2) \in \mathbb{R}^2\}$.

Where are the **sparse** solutions?



ℓ_1 and ℓ_2 Constraints

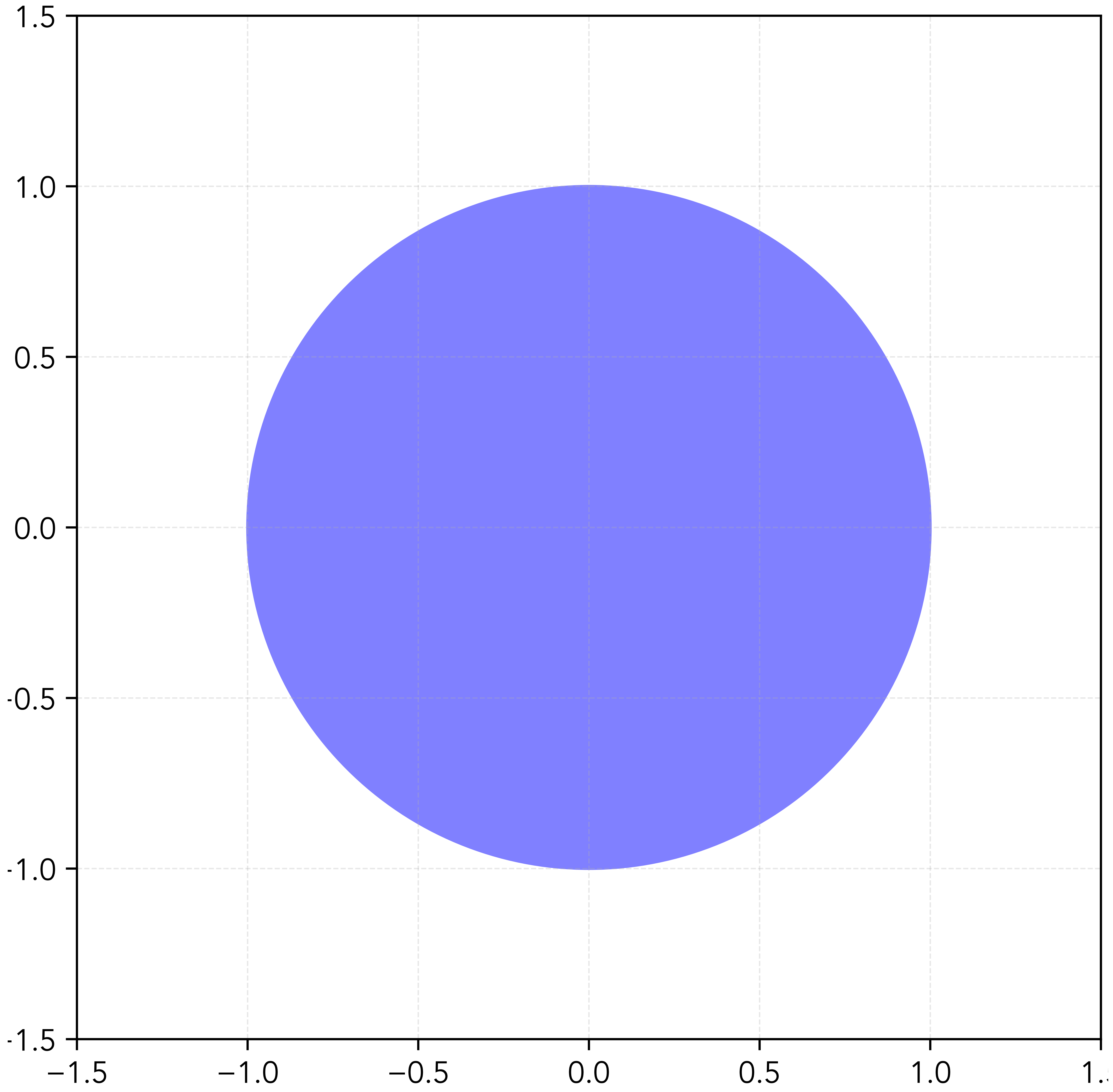
Intuition

For visualization, restrict to:

$$\mathcal{H} = \{h(x) = w_1x_1 + w_2x_2\}$$

Represent $\mathcal{H}$ by $\{(w_1, w_2) \in \mathbb{R}^2\}$.

Where are the **sparse** solutions?



Empirical Risk in $\mathbb{R}^2$

Visualization for Square Loss

In matrix form: $\hat{R}_n(w) = \dfrac{1}{n}\|Xw - y\|^2$.

Minimizer: $\hat{w} = (X^\top X)^{-1} X^\top y$

For any $w \in \mathbb{R}^d$, by "completing square":

$$\hat{R}_n(w) = \frac{1}{n}(w - \hat{w})^\top (X^\top X)(w - \hat{w}) + \hat{R}_n(\hat{w})$$

The w such that $\hat{R}_n(w)$ exceeds $\hat{R}_n(\hat{w})$ by $c > 0$ are ellipsoids centered at $\hat{w}$:

$$\left\{w : \hat{R}_n(w) = c + \hat{R}_n(\hat{w})\right\} = \{w : (w - \hat{w})^\top X^\top X (w - \hat{w}) = nc\}$$

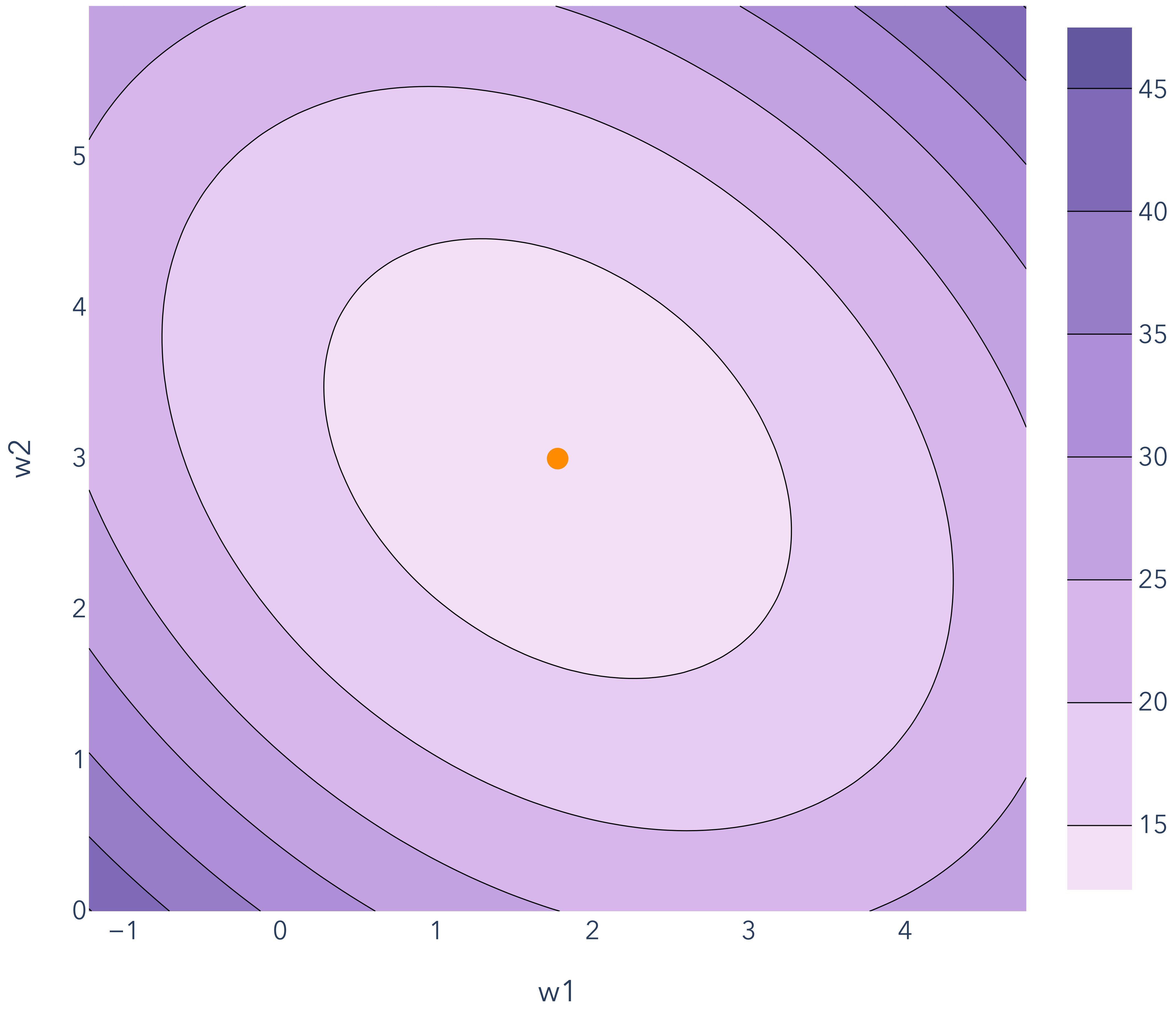
Empirical Risk in $\mathbb{R}^2$

Visualization for Square Loss

The w such that $\hat{R}_n(w)$ exceeds $\hat{R}_n(\hat{w})$ by $c > 0$ are ellipsoids centered at $\hat{w}$:

$$\left\{w : \hat{R}_n(w) = c + \hat{R}_n(\hat{w})\right\}$$

$$= \left\{w : (w - \hat{w})^\top X^\top X(w - \hat{w}) = nc\right\}$$

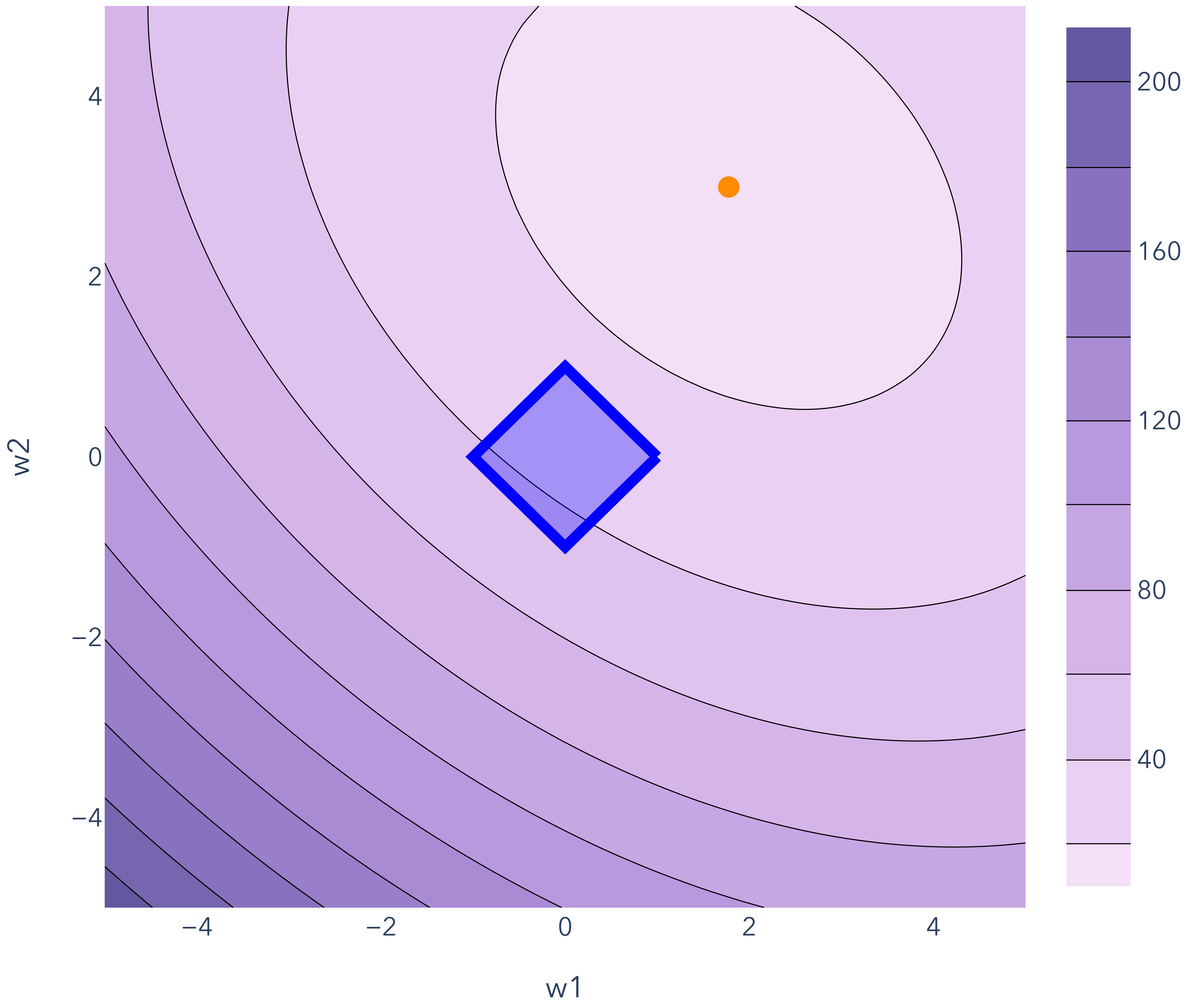


ℓ_1 Regularization

Visualization

$$\hat{w}_r \in \arg\min_{w \in \mathbb{R}^2} \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2$$

$$\text{subject to: } \left|w_1\right| + \left|w_2\right| \leq r$$

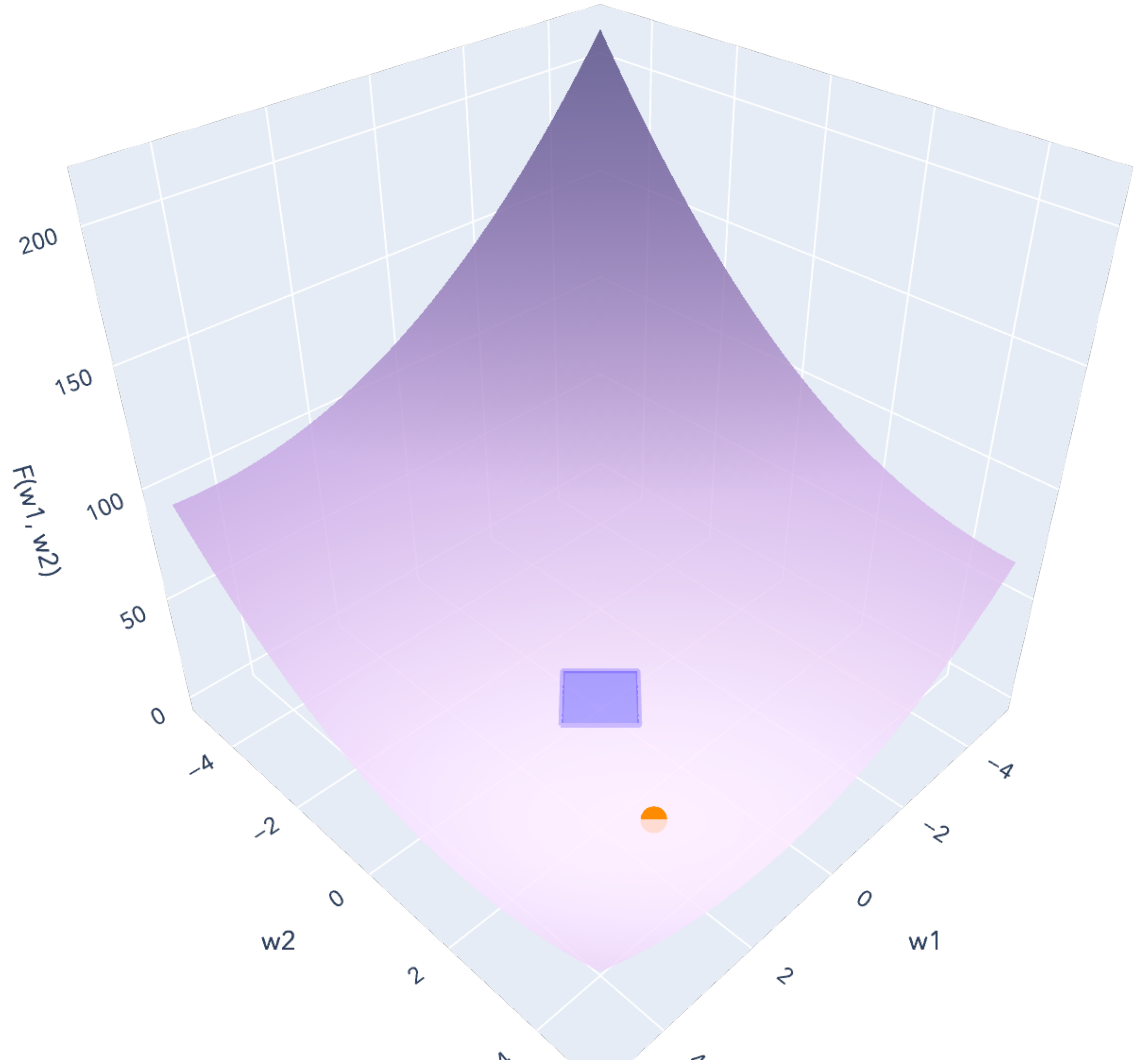


ℓ_1 Regularization

Visualization

$$\hat{w}_r \in \underset{w \in \mathbb{R}^2}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2$$

$$\text{subject to: } \left|w_1\right| + \left|w_2\right| \leq r$$

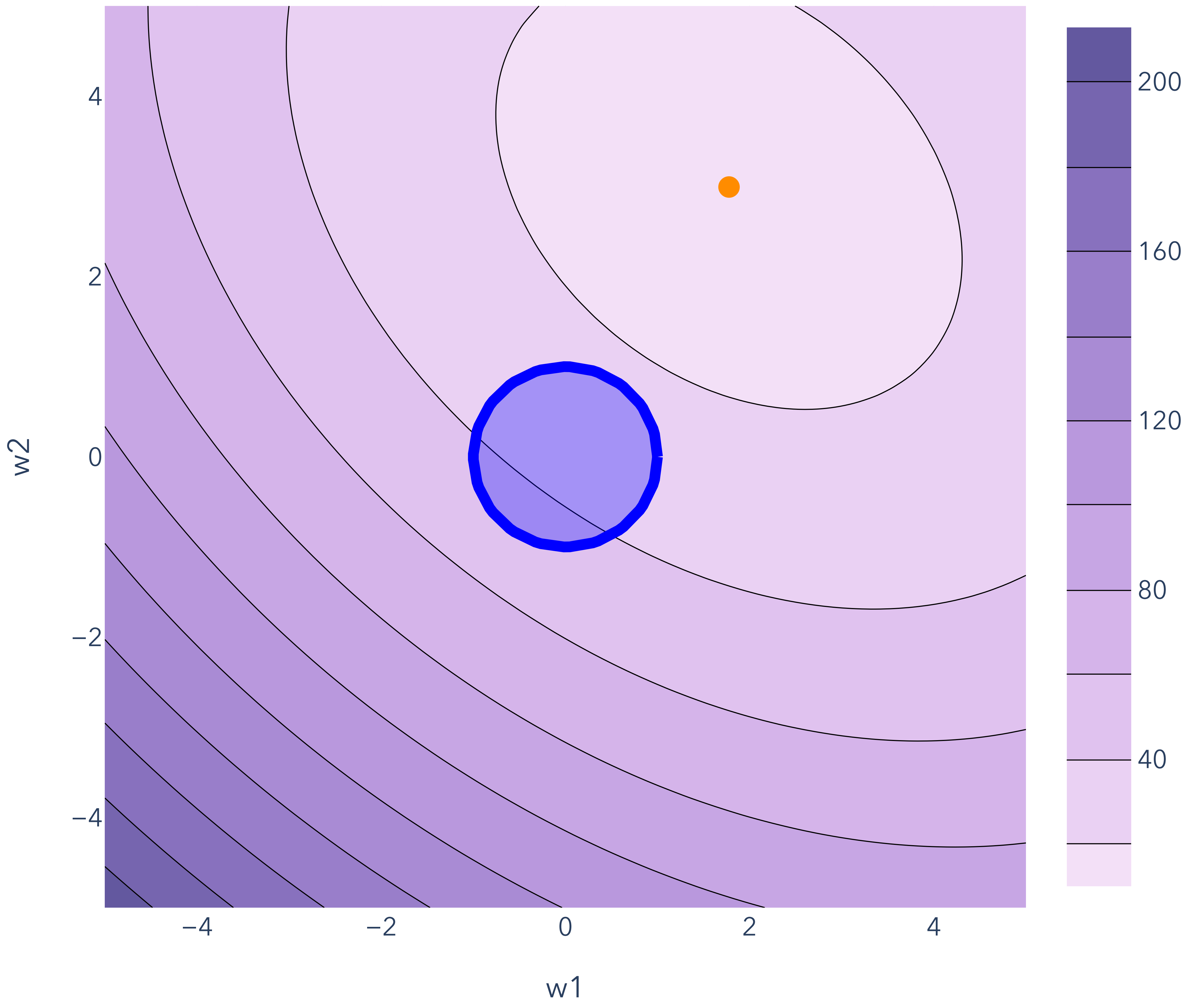


ℓ_2 Regularization

Visualization

$$\hat{w}_r \in \arg\min_{w \in \mathbb{R}^2} \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2$$

$$\text{subject to: } w_1^2 + w_2^2 \leq r$$

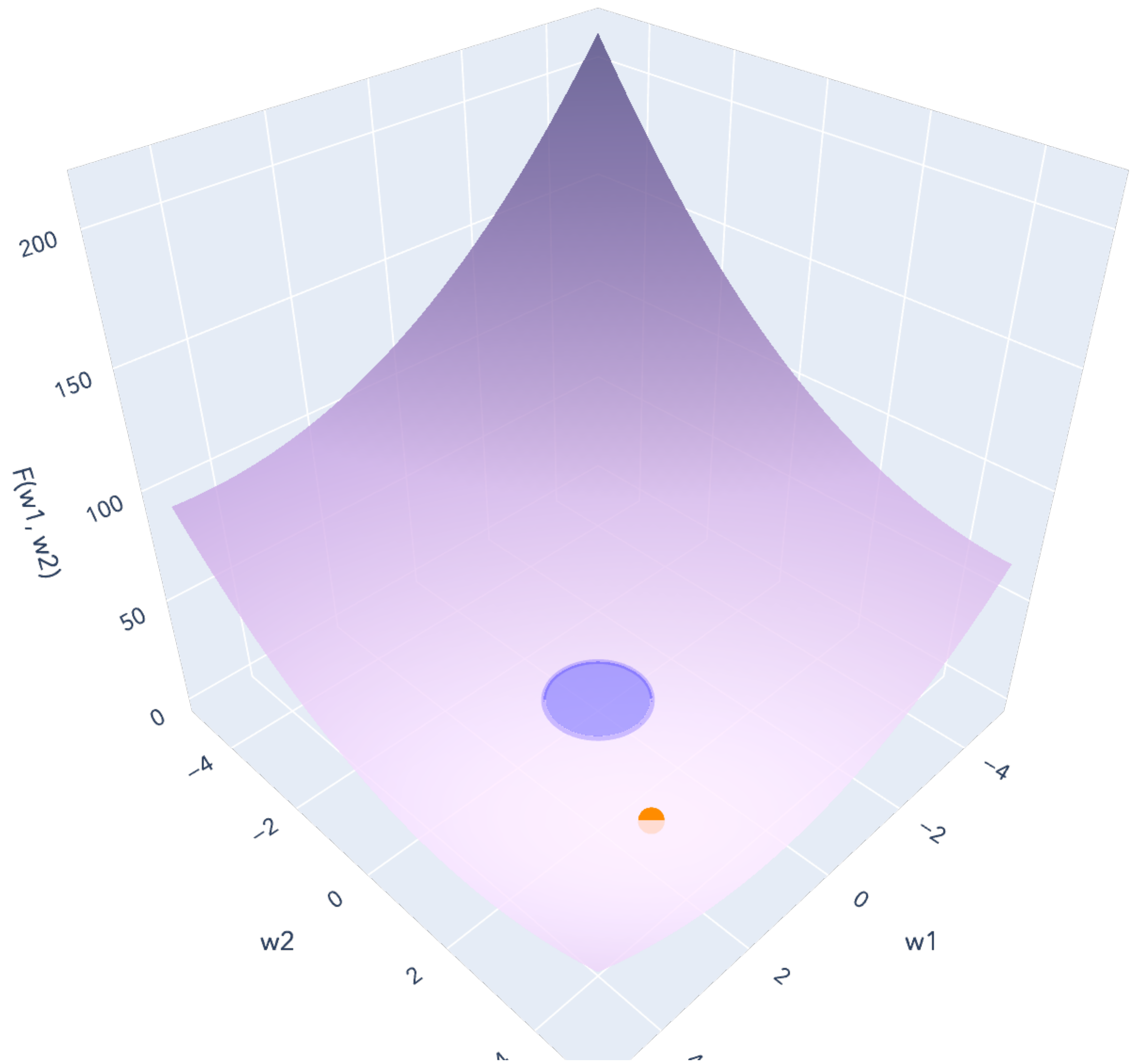


ℓ_2 Regularization

Visualization

$$\hat{w}_r \in \arg\min_{w \in \mathbb{R}^2} \frac{1}{n} \sum_{i=1}^{n} (w^\top x^{(i)} - y^{(i)})^2$$

subject to: $w_1^2 + w_2^2 \leq r$



Sparsity

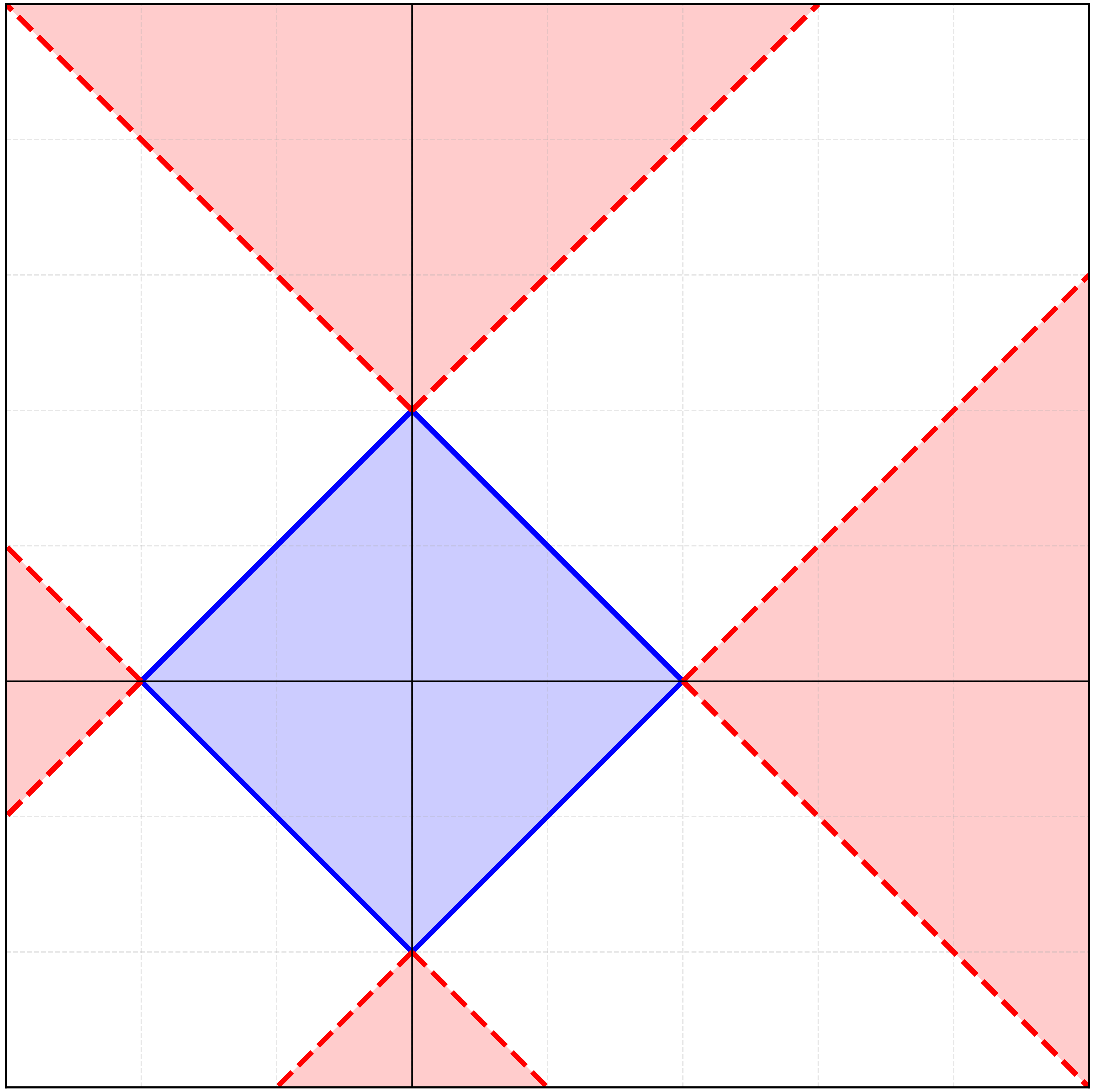
Geometric Intuition

Suppose $X^\top X = I$ (orthogonal features).

Then, contours are perfect circles.

The $\hat{w}$ in red regions are closest to corners in the ℓ_1 ball.

Geometric intuition: Projection onto diamond (ℓ_1 ball) encourages solutions at corners.



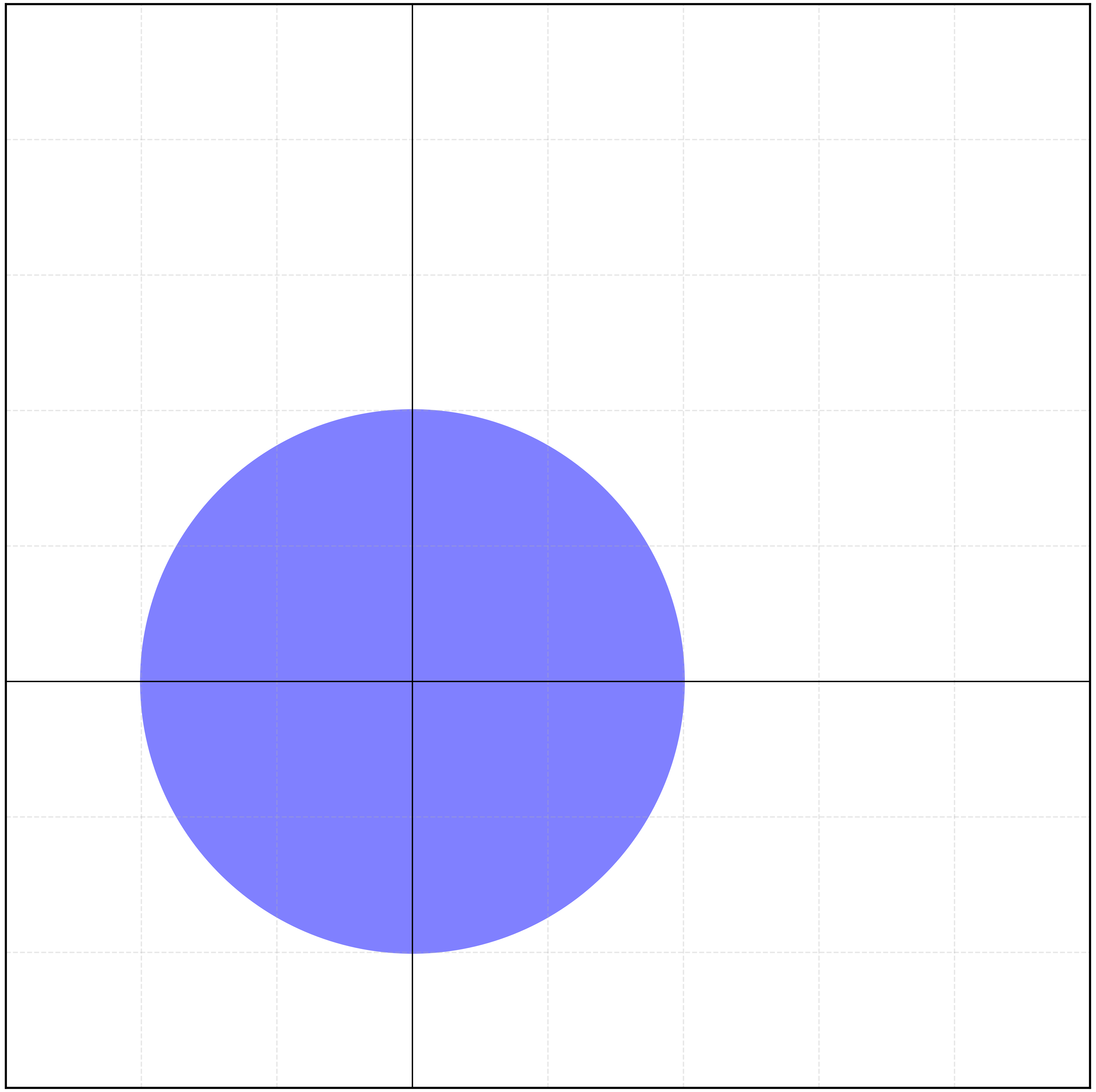
Sparsity

Geometric Intuition

Suppose $X^\top X = I$ (orthogonal features).

Then, contours are perfect circles.

Geometric intuition: Projection onto sphere (ℓ_2 ball) encourages solutions equally.



ℓ_q Regularization

Geometric Intuition

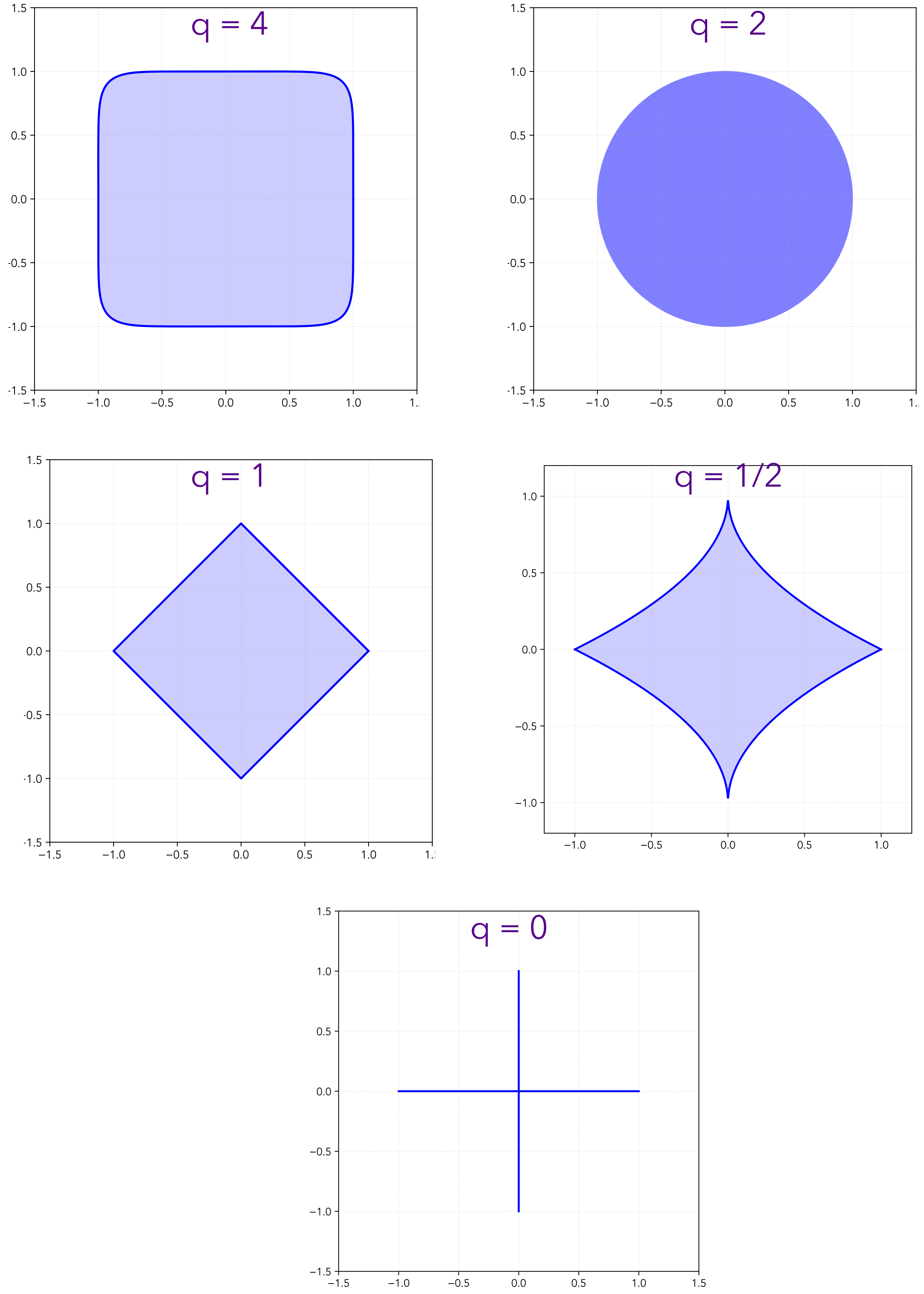
Generalize to ℓ_q:

$$(\|w\|_q)^q = \left|w_1\right|^q + \left|w_2\right|^q$$

Note: $\|w\|_q$ is only a norm for $q \geq 1$ but not for $q \in (0,1)$.

When $q < 1$, the ℓ_q constraint is non-convex (so hard to optimize).

ℓ_0 defined as number of non-zero weights, i.e. subset selection.

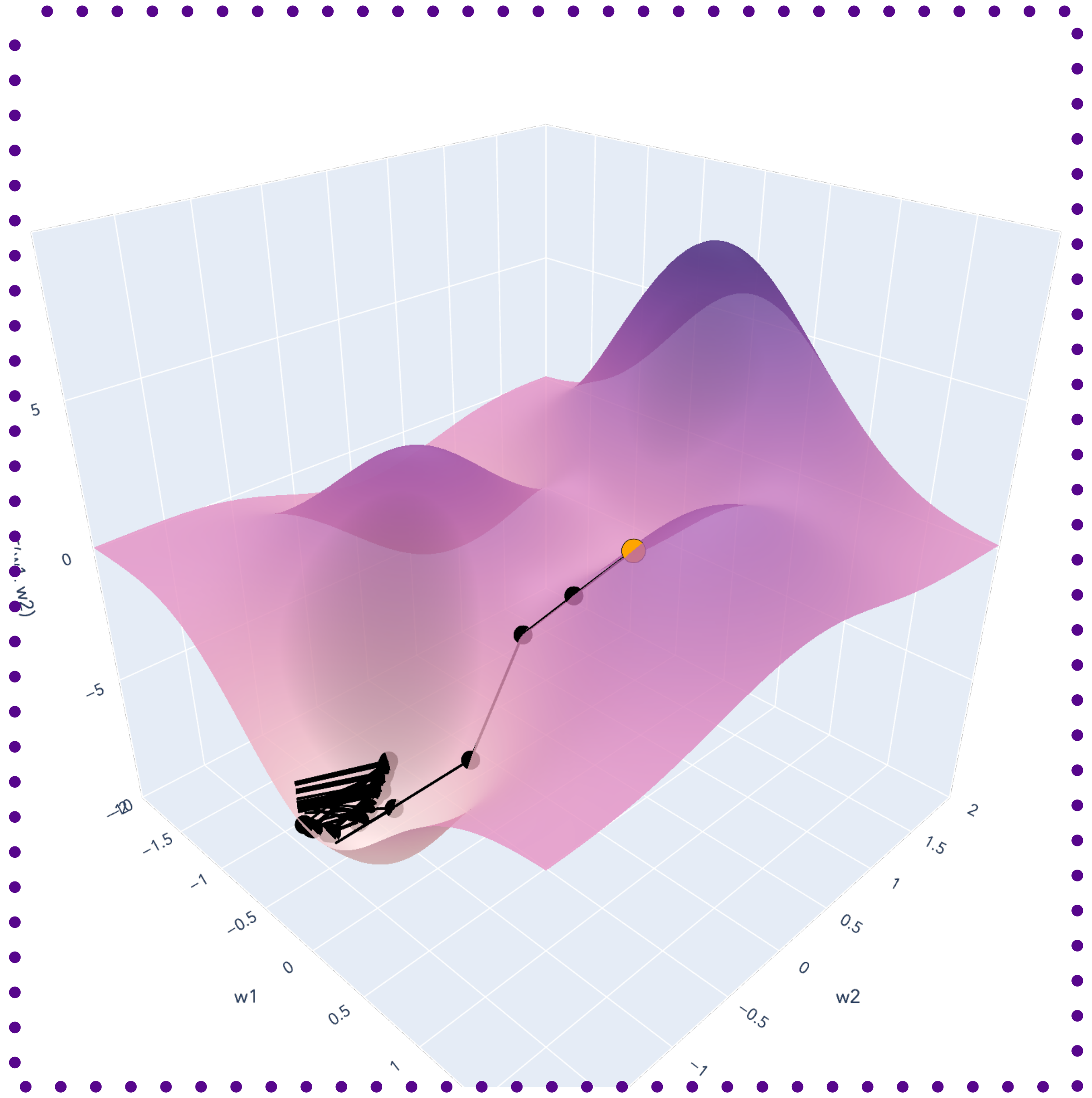


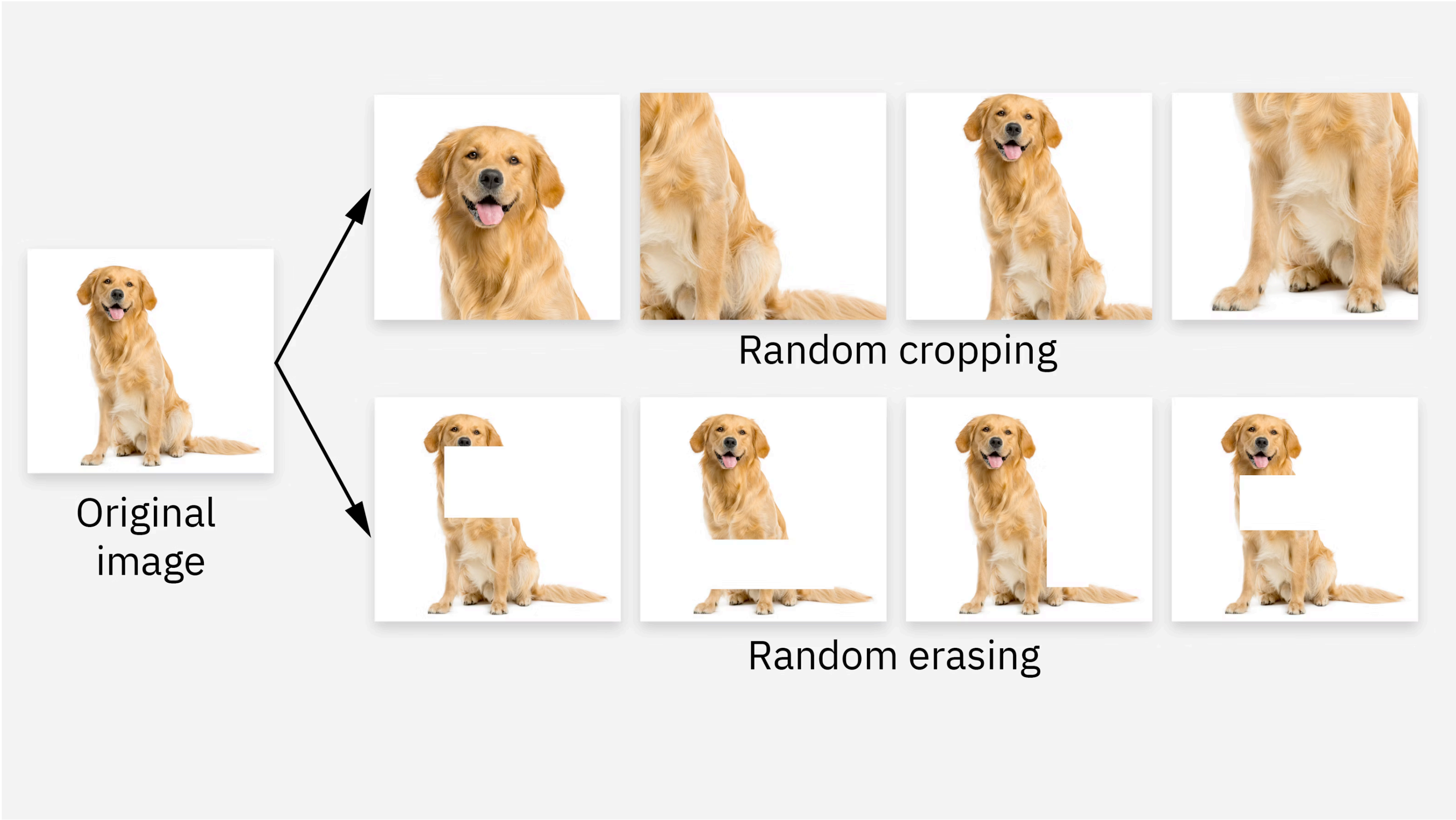
Other Forms of Regularization

Implicit Regularization, Weight Decay, etc.

In general, regularization is a term that describes ways to “bias” a problem with infinitely many solutions to a smaller subset of solutions.

> **Implicit regularization.** Properties of the optimization algorithm lead to “simple” solutions.
>
> **Data augmentation.** Randomly modify training data in by an operation, usually used in deep learning (e.g. randomly cropping images).





Outline

Model Complexity and Model Selection

Controlling Complexity with Regularization

ℓ_2 Regularization and Ridge Regression

ℓ_1 Regularization and Lasso Regression

Understanding Sparsity

Loss Functions: Regression

Loss Functions: Classification

Regression

Problem Instance

Input space: $\mathscr{X} = \mathbb{R}^d$

Action space: $\mathscr{A} = \mathbb{R}$

Outcome space: $\mathscr{Y} = \mathbb{R}$

$\hat{y}$ is the predicted value (the **action**).

y is the observed value (the **outcome**).

Distance Based Loss

Definition

In general, loss functions take the form:

$$(\hat{y}, y) \mapsto \ell(\hat{y}, y) \in \mathbb{R}$$

Regression losses typically depend on the residual $r = y - \hat{y}$.

A loss function is distance-based if:

1. It only depends on the residual: $\ell(\hat{y}, y) = \psi(y - \hat{y})$ for some $\psi : \mathbb{R} \to \mathbb{R}$
2. It is zero when the residual is zero: $\psi(0) = 0$.

Loss Functions

Examples

$$r = y - \hat{y}$$

Square (ℓ_2) loss: $\ell(r) = r^2$.

Absolute (ℓ_1) loss: $\ell(r) = |r|$.

Outliers typically have large residuals.

Square loss more affected by outliers than absolute loss.

y	$\hat{y}$	$y - \hat{y}$	$(y - \hat{y})^2$
1	0	1	1
5	0	5	25
10	0	10	100
50	0	50	2500

Loss Functions

Examples

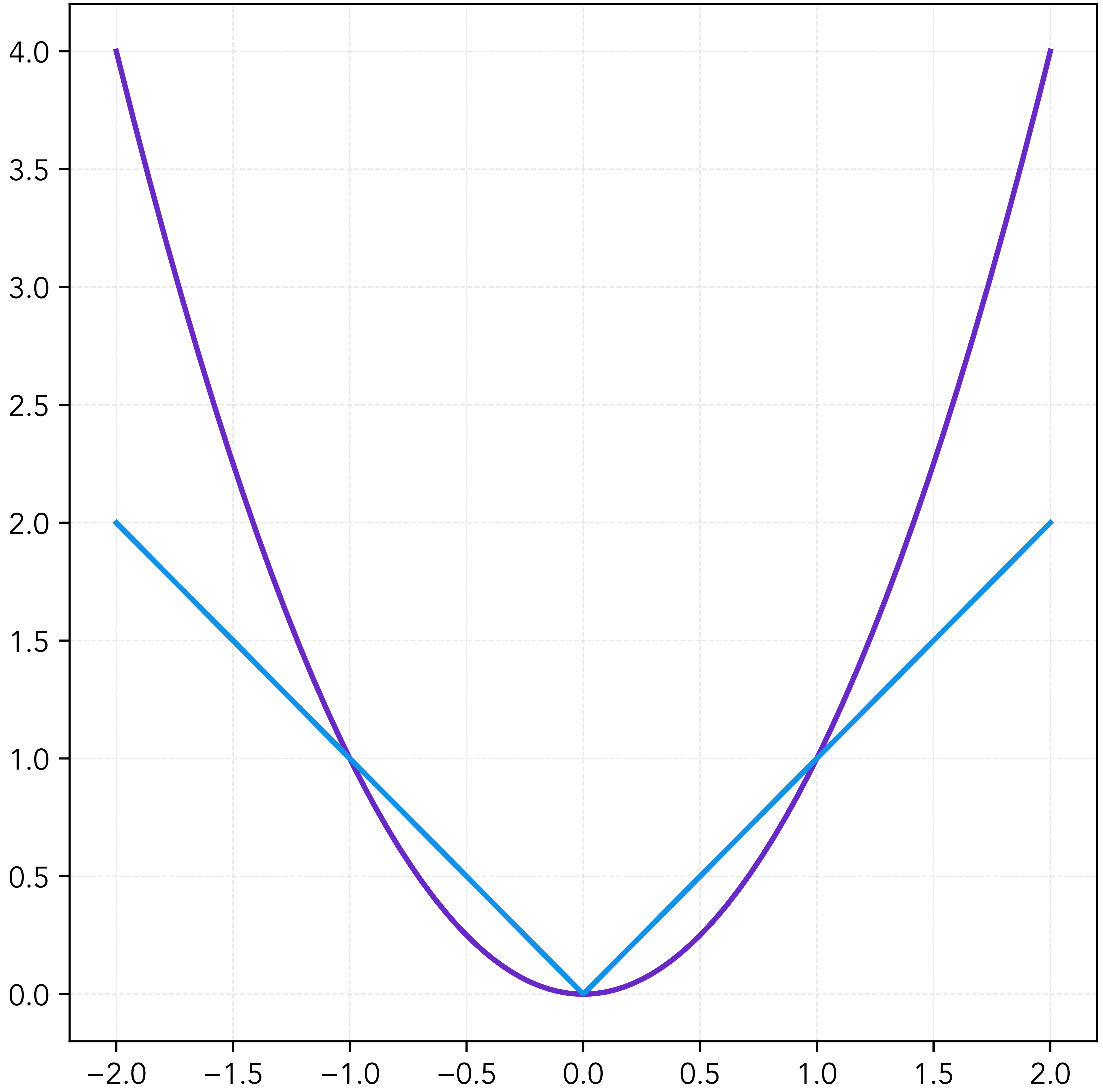
$$r = y - \hat{y}$$

Square (ℓ_2) loss: $\ell(r) = r^2$.

Absolute loss: $\ell(r) = |r|$.

Outliers typically have large residuals.

Square loss more affected by outliers than absolute loss.



Loss Functions

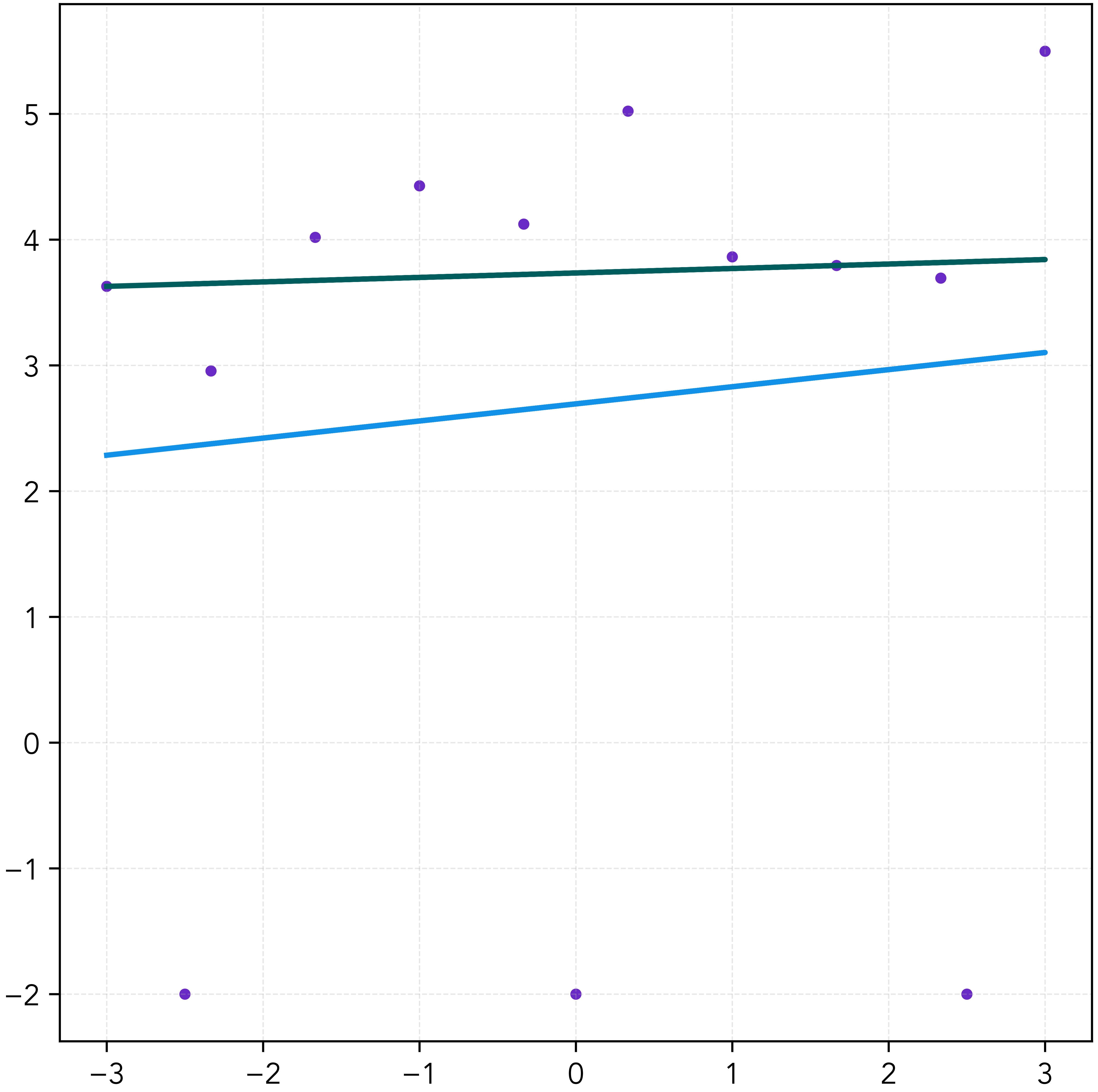
Robustness

$$r = y - \hat{y}$$

Square (ℓ_2) loss: $\ell(r) = r^2$.

Absolute loss: $\ell(r) = |r|$.

Robustness refers to how affected a learning algorithm is by outliers.



Loss Functions

Robustness

Square loss: $\ell(r) = r^2$

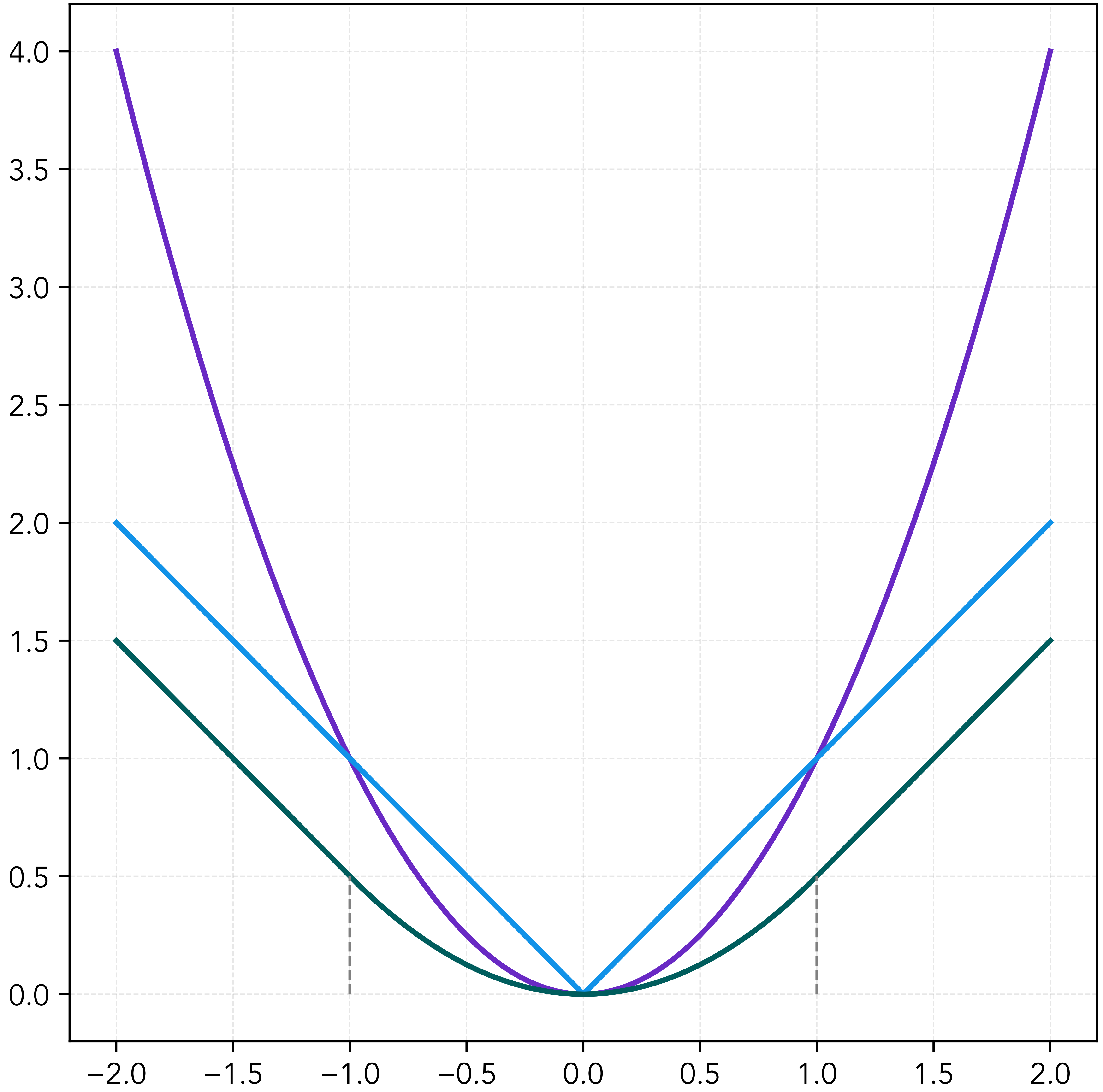
(not robust)

Absolute loss: $\ell(r) = |r|$

(not differentiable)

Huber loss: Quadratic for $|r| \leq \delta$ and linear for $|r| > \delta$

(robust and differentiable)



Outline

Model Complexity and Model Selection

Controlling Complexity with Regularization

ℓ_2 Regularization and Ridge Regression

ℓ_1 Regularization and Lasso Regression

Understanding Sparsity

Loss Functions: Regression

Loss Functions: Classification

Classification

Problem Instance

Input space: $\mathcal{X} = \mathbb{R}^d$

Action space: $\mathcal{A} = \{-1,1\}$

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

We've already seen the zero-one loss for $f : \mathcal{X} \to \{-1,1\}$:

$$\ell(f(x), y) = \mathbf{1}\{f(x) \neq y\}$$

But let's allow real-valued predictions $f : \mathcal{X} \to \mathbb{R}$.

Classification

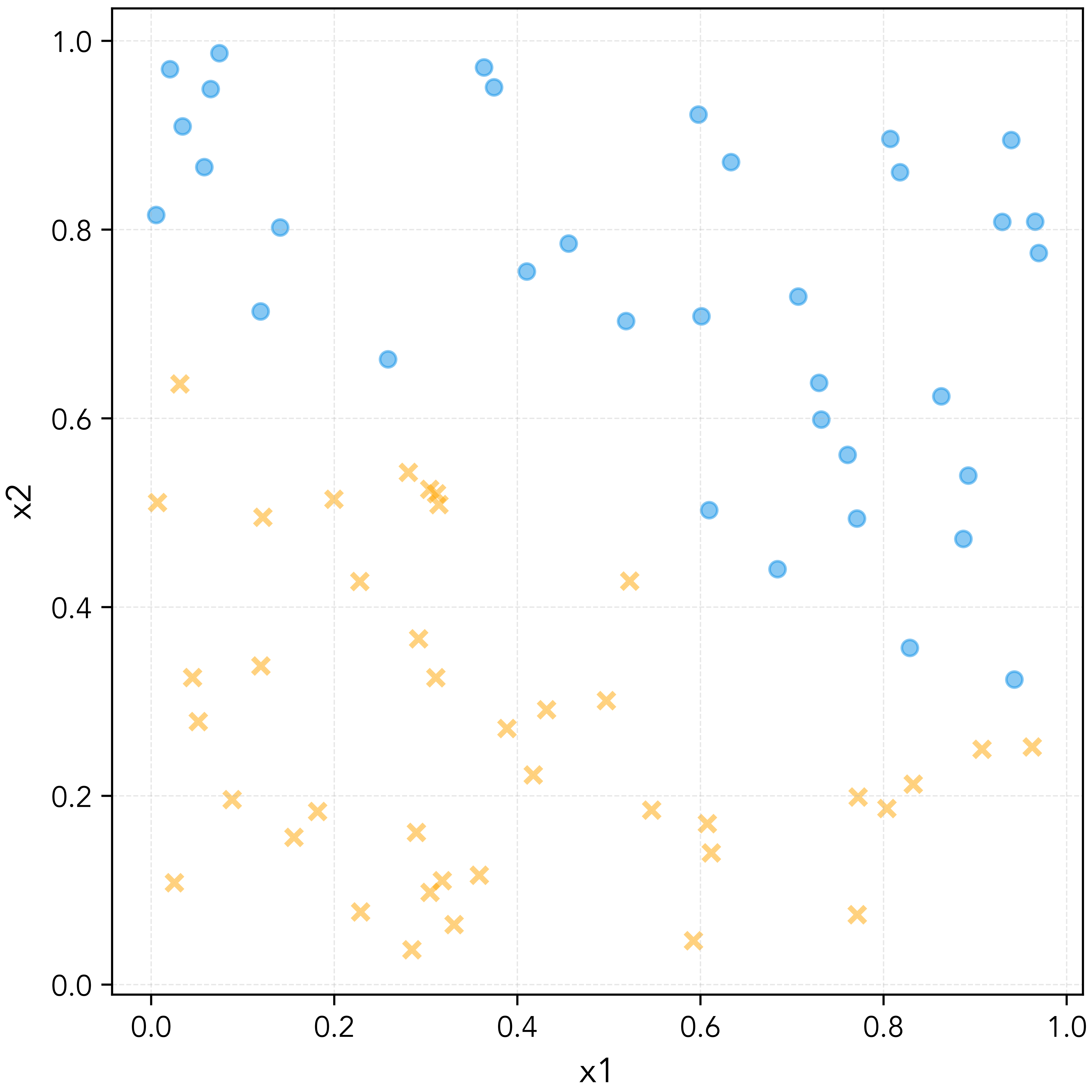
Geometric Picture

Input space: $\mathcal{X} = \mathbb{R}^d$

Action space: $\mathcal{A} = \{-1,1\}$

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

Geometrically: find a **decision boundary** between the classes.



Classification

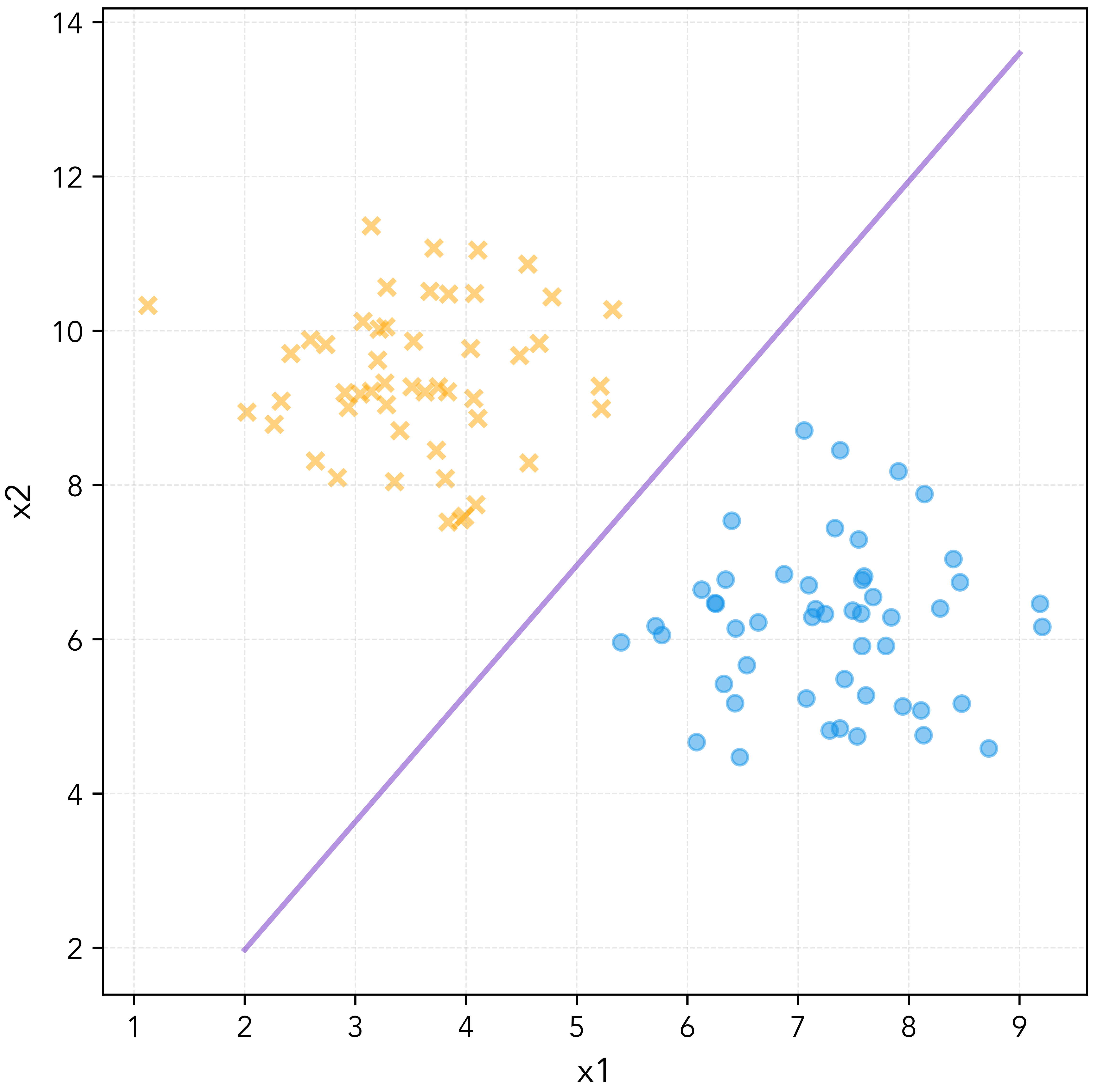
Geometric Picture

Input space: $\mathscr{X} = \mathbb{R}^d$

Action space: $\mathscr{A} = \{-1,1\}$

Outcome space: $\mathscr{Y} = \{-1,1\}$

We will focus on methods that induce **linear decision boundaries.**



Classification

Geometric Picture

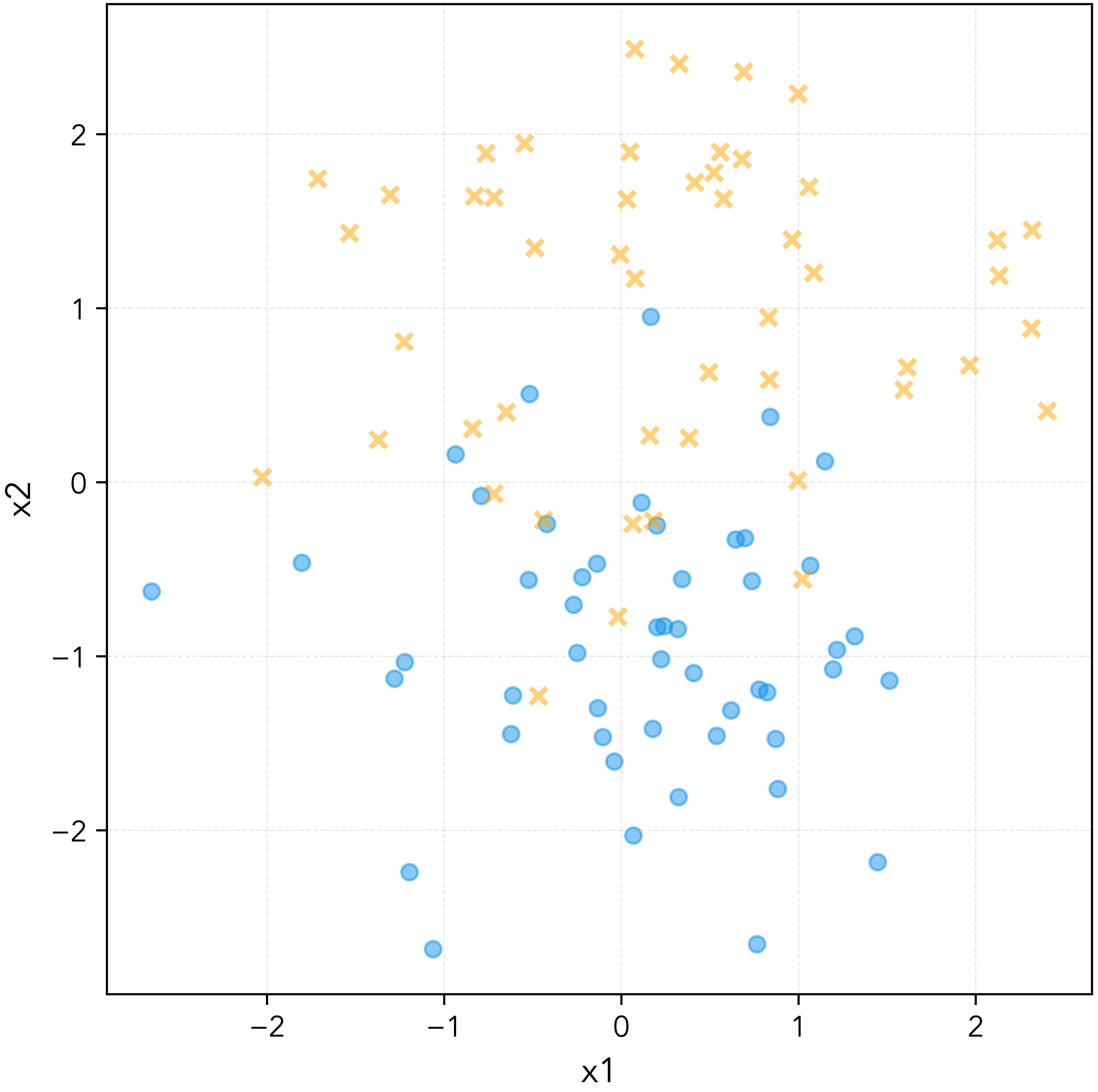
Input space: $\mathscr{X} = \mathbb{R}^d$

Action space: $\mathscr{A} = \{-1,1\}$

Outcome space: $\mathscr{Y} = \{-1,1\}$

We will focus on methods that induce **linear decision boundaries.**

Most problems are *not* linearly separable (i.e. there exists a hyperplane separating the $y = -1$ and $y = 1$ points).



Classification

Problem Instance

Input space: $\mathscr{X} = \mathbb{R}^d$

Action space: $\mathscr{A} = \mathbb{R}$

Outcome space: $\mathscr{Y} = \{-1,1\}$

But let's allow real-valued predictions $f : \mathscr{X} \to \mathbb{R}$.

$f(x) > 0 \implies$ Predict 1

$f(x) < 0 \implies$ Predict -1

Classification

Problem Instance

Input space: $\mathscr{X} = \mathbb{R}^d$

Action space: $\mathscr{A} = \mathbb{R}$

Outcome space: $\mathscr{Y} = \{-1,1\}$

For a **linear** function $f(x) = w^\top x$:

$$w^\top x > 0 \implies \text{Predict } 1$$

$$w^\top x < 0 \implies \text{Predict } -1$$

Classification

Score Function

Outcome space: $\mathcal{Y} = \{-1, 1\}$ Action space: $\mathcal{A} = \mathbb{R}$

For a real-valued prediction function $f : \mathcal{X} \to \mathbb{R}$, the value $f(x)$ is called the score for input x.

In this context, we can call f a **score function**.

The magnitude of the score can be interpreted as **confidence** in our prediction.

Margin

Definition

The margin for a predicted score $\hat{y}$ and the true class $y \in \{-1, 1\}$ is $y\hat{y}$.

With a score function $f : \mathcal{X} \to \mathbb{R}$, the margin is $yf(x)$.

If y and $\hat{y}$ are the same sign, prediction is **correct** and margin is **positive**.

If y and $\hat{y}$ have different sign, prediction is **incorrect** and margin is **negative**.

We want to find f that **maximizes** the margin.

Many classification losses only depend on the margin (margin-based losses).

Classification Losses

Zero-One Loss

$$h(x) = \text{sign}(f(x)) := \begin{cases} 1 & \text{if } f(x) \geq 0 \\ -1 & \text{if } f(x) < 0 \end{cases}$$

The zero-one loss for $h : \mathcal{X} \to \{-1,1\}$ is $\ell(h(x), y) = \mathbf{1}\{h(x) \neq y\}$.

We can rewrite this in terms of the margin and score function as

$$\ell(f(x), y) := \mathbf{1}\{yf(x) \leq 0\}.$$

The empirical risk for zero-one loss, given dataset D_n:

$$\hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{y^{(i)} f(x^{(i)}) \leq 0\}$$

Classification Losses

Zero-One Loss

The empirical risk for zero-one loss, given dataset D_n:

$$\hat{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{y^{(i)} f(x^{(i)}) \leq 0\}$$

Non-convex, non-differentiable, and discontinuous.

Optimization problem is NP-hard (computationally infeasible).

Classification Losses

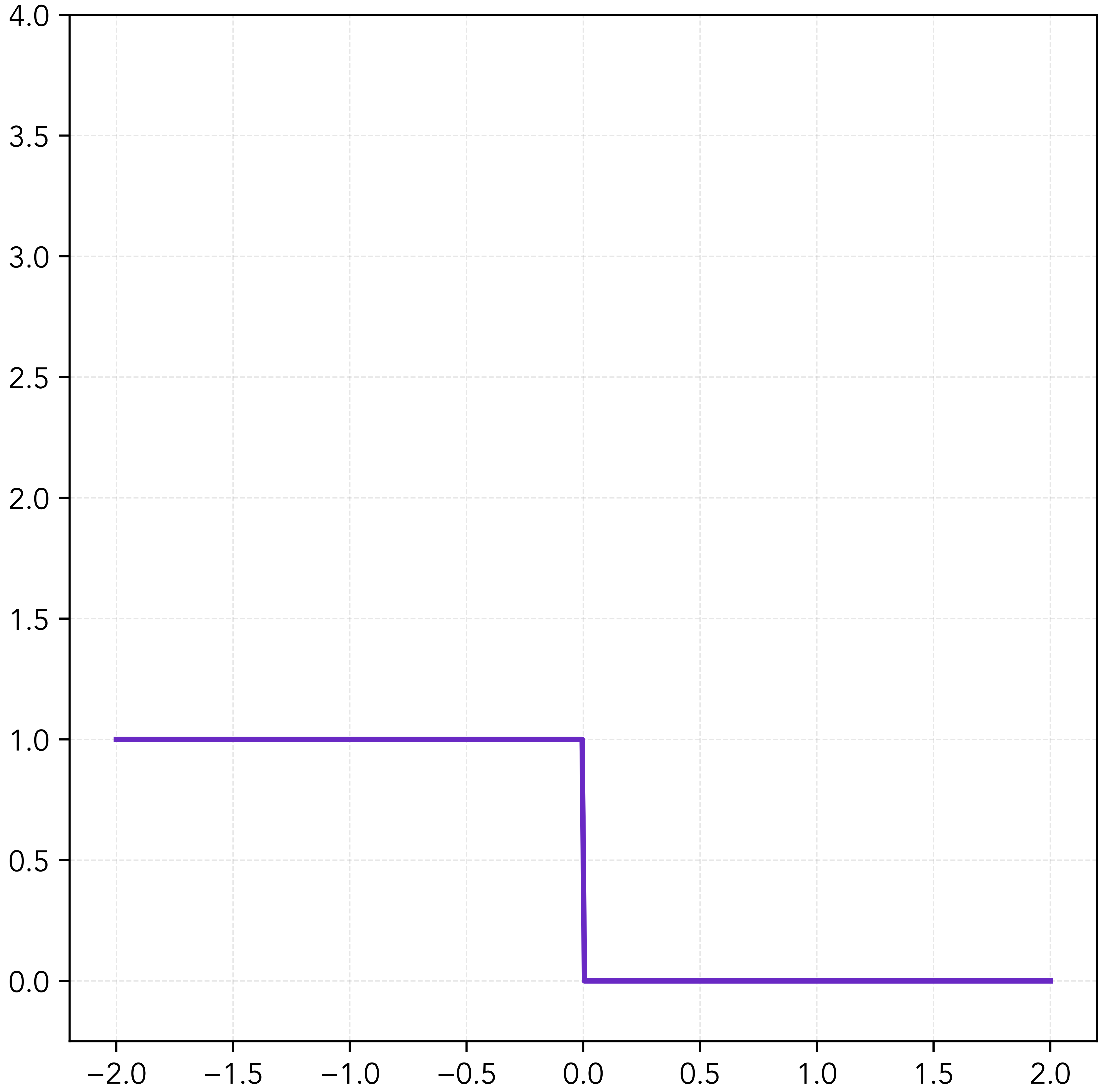
Zero-One Loss

Margin: $m = \hat{y}y$

Zero-one loss: $\ell_{0-1}(m) := \mathbf{1}\{m \leq 0\}$

x-axis is margin:

$m > 0 \iff$ classification is **correct**.



Classification Losses

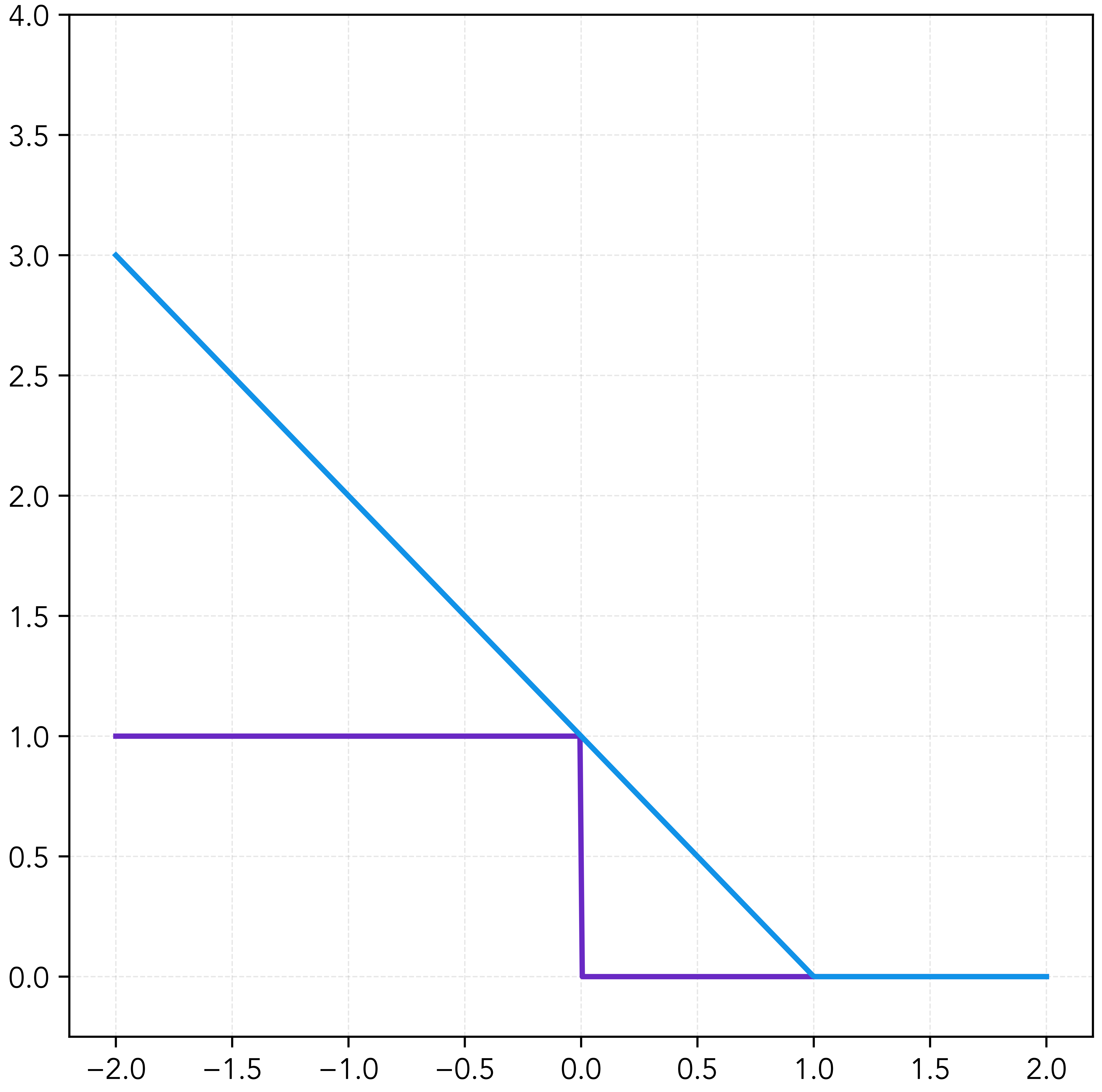
Hinge Loss

Margin: $m = \hat{y}y$

Hinge loss: $\ell_{\text{hinge}}(m) := \max(1 - m, 0)$

Hinge loss is **convex, upper bound** on zero-one loss.

Not differentiable at $m = 1$.



Hinge Loss

(Soft-Margin) Support Vector Machine

Hypothesis class: $\mathscr{H} = \{h_w(x) = w^\top x : w \in \mathbb{R}^d\}$

Loss: $\ell_{\text{hinge}}(m) = \max(1 - m, 0)$ (hinge loss)

Regularizer: ℓ_2

Empirical risk minimization:

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \max(1 - y^{(i)} h_w(x^{(i)}), 0) + \lambda \|w\|_2^2$$

Classification Losses

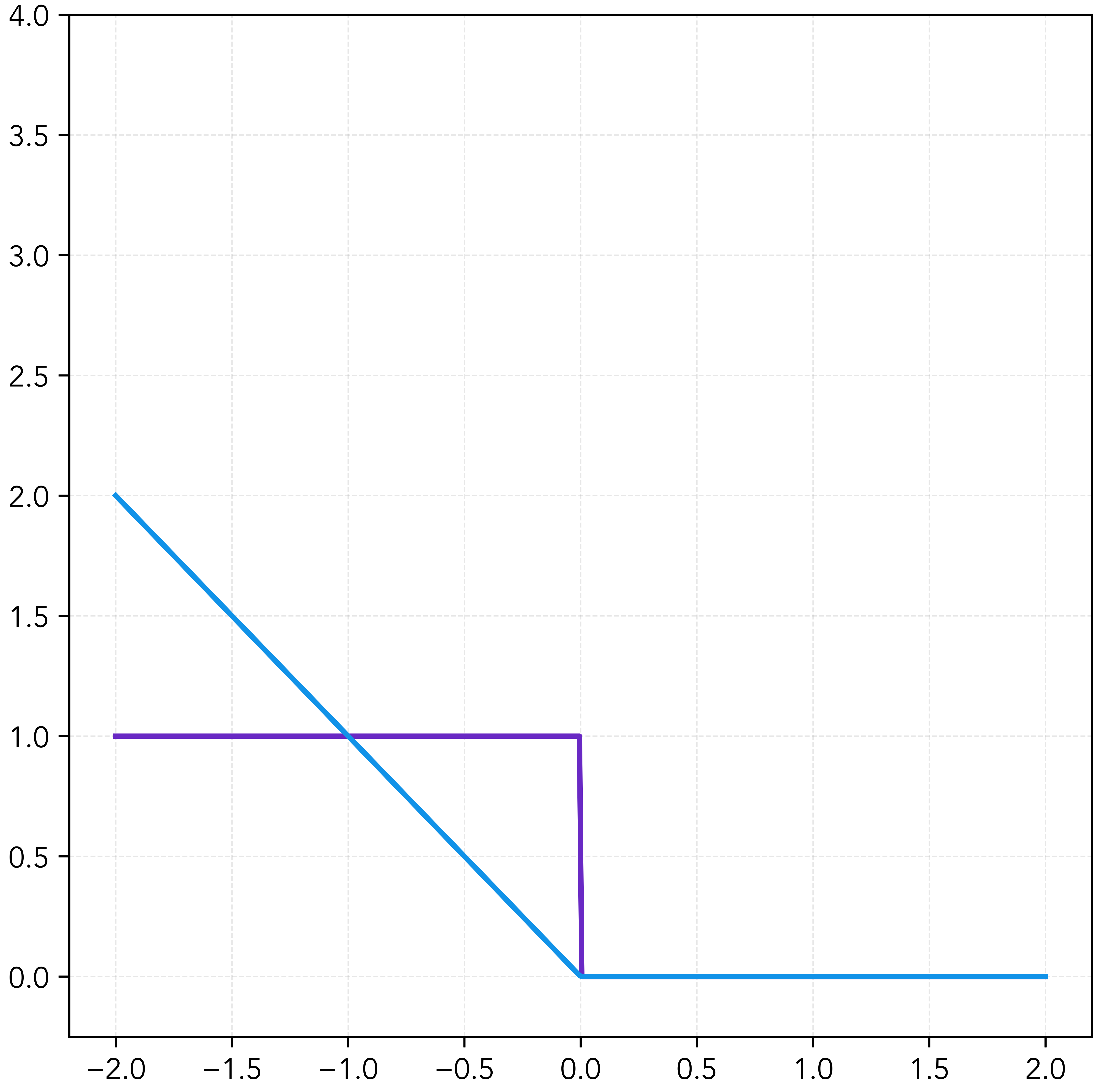
Perceptron Loss

Margin: $m = \hat{y}y$

Perceptron loss: $\ell_{\text{perc}}(m) := \max(-m, 0)$

Hinge loss, with "hinge at zero."

Not an upper bound on zero-one loss, but it is **convex.**



Perceptron Loss

Perceptron Algorithm

Hypothesis class: $\mathscr{H} = \{h_w(x) = w^\top x : w \in \mathbb{R}^d\}$

“SGD” on the perceptron loss $\ell_{\text{perc}}(m) = \max(-m, 0)$ (perceptron loss) is equivalent to:

```
Initialize $w \leftarrow 0$.
While there exists $(x^{(i)}, y^{(i)})$ that is misclassified:
    For $(x^{(i)}, y^{(i)}) \in D_n$:
        If $y^{(i)} w^\top x^{(i)} < 0$ (wrong prediction):
            Update $w \leftarrow w + y^{(i)} x^{(i)}$.
```

Classification Losses

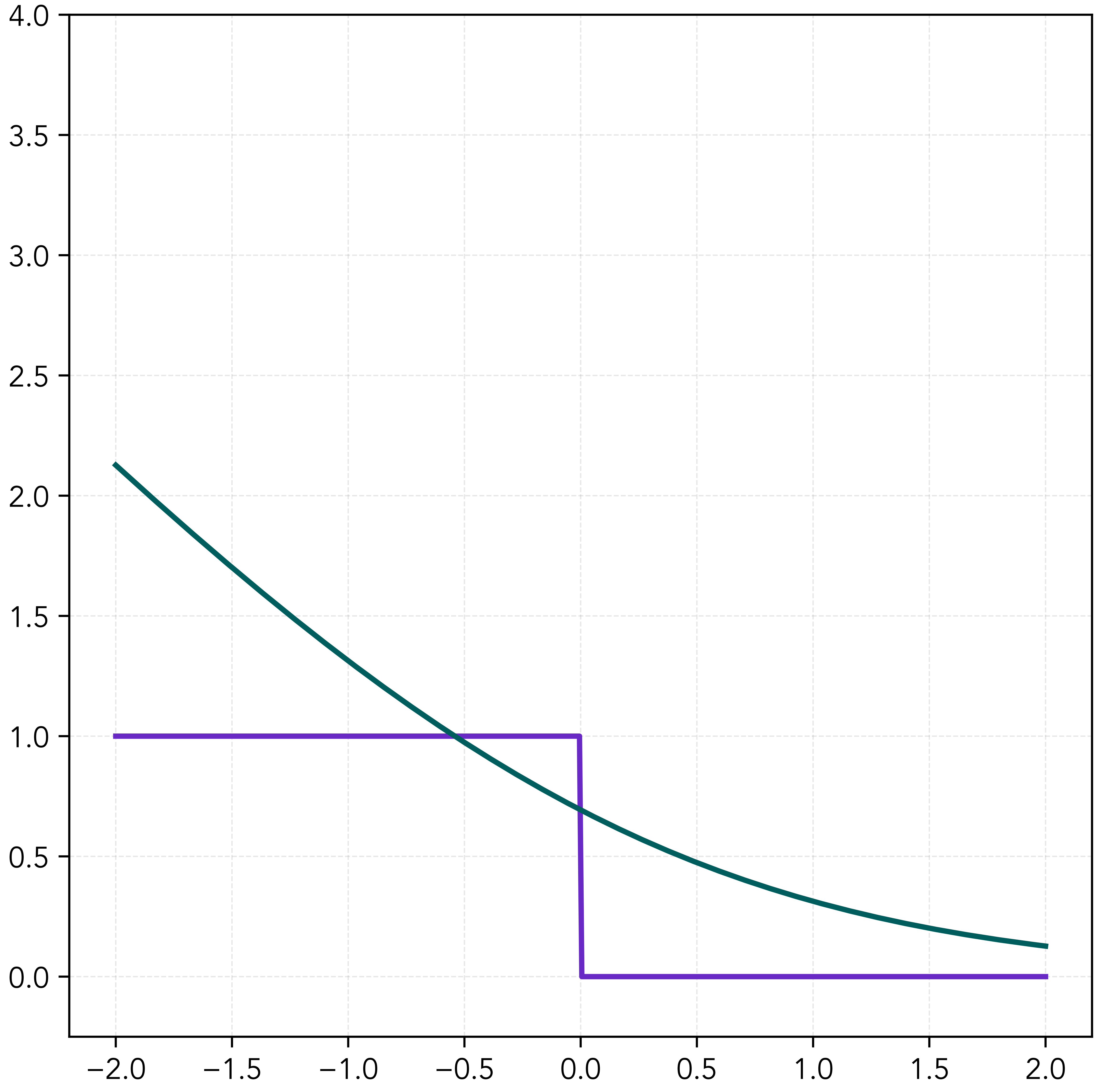
Logistic Loss

Margin: $m = \hat{y}y$

Logistic/Log loss: $\ell_{\log}(m) := \log(1 + e^{-m})$

Logistic loss is **differentiable**.

Always rewards more margin (loss never 0).



Logistic Loss

Logistic Regression

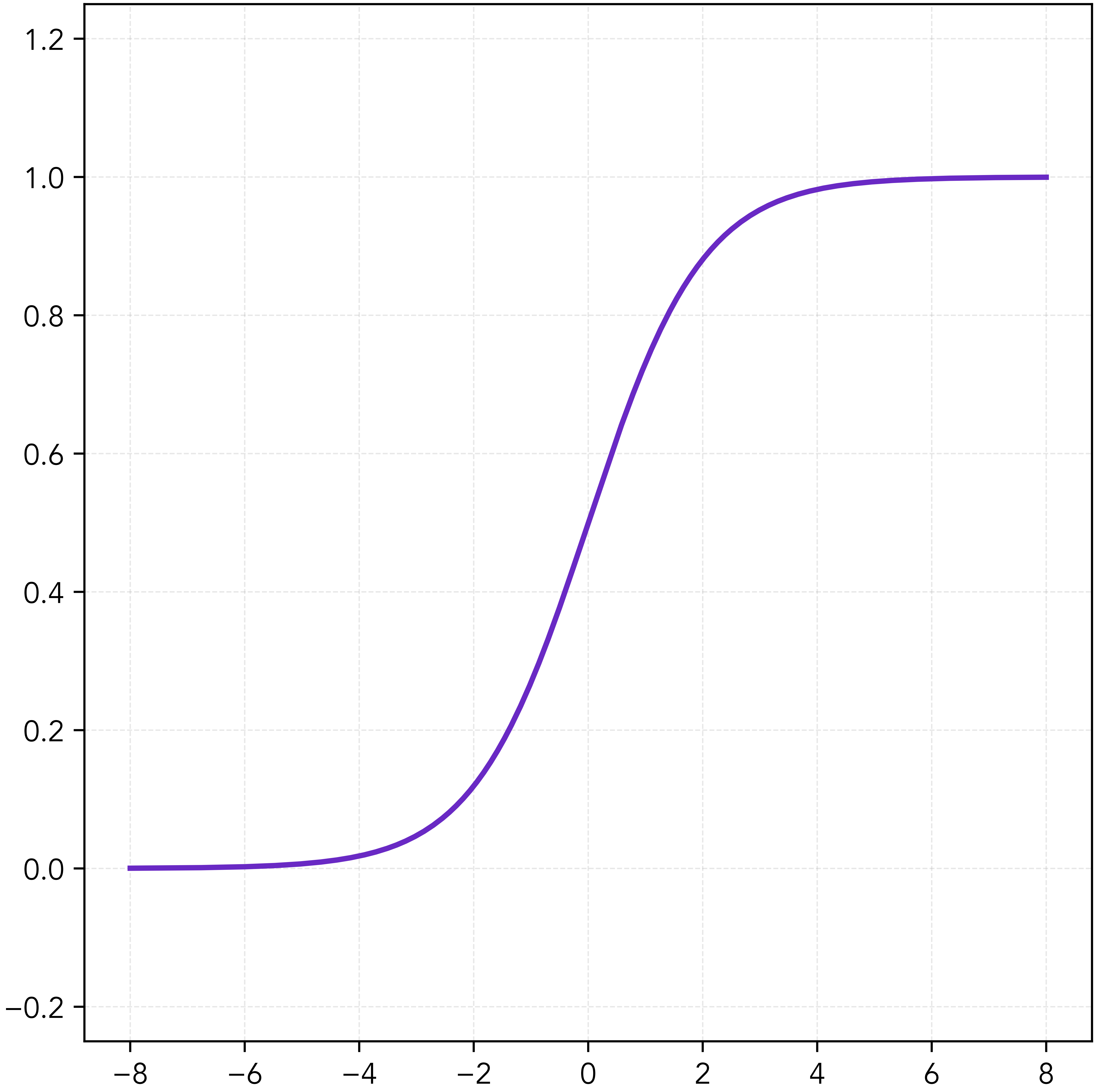
Suppose we want some $h : \mathbb{R}^d \to [0,1]$ (to be interpreted as *probability* of -1 or 1).

The sigmoid function $\phi : \mathbb{R} \to [0,1]$:

$$\phi(z) := \frac{1}{1 + \exp(-z)}$$

Useful property:

$$1 - \phi(z) = \phi(-z).$$



Logistic Loss

Logistic Regression

$$\phi(z) := \frac{1}{1 + \exp(-z)}$$

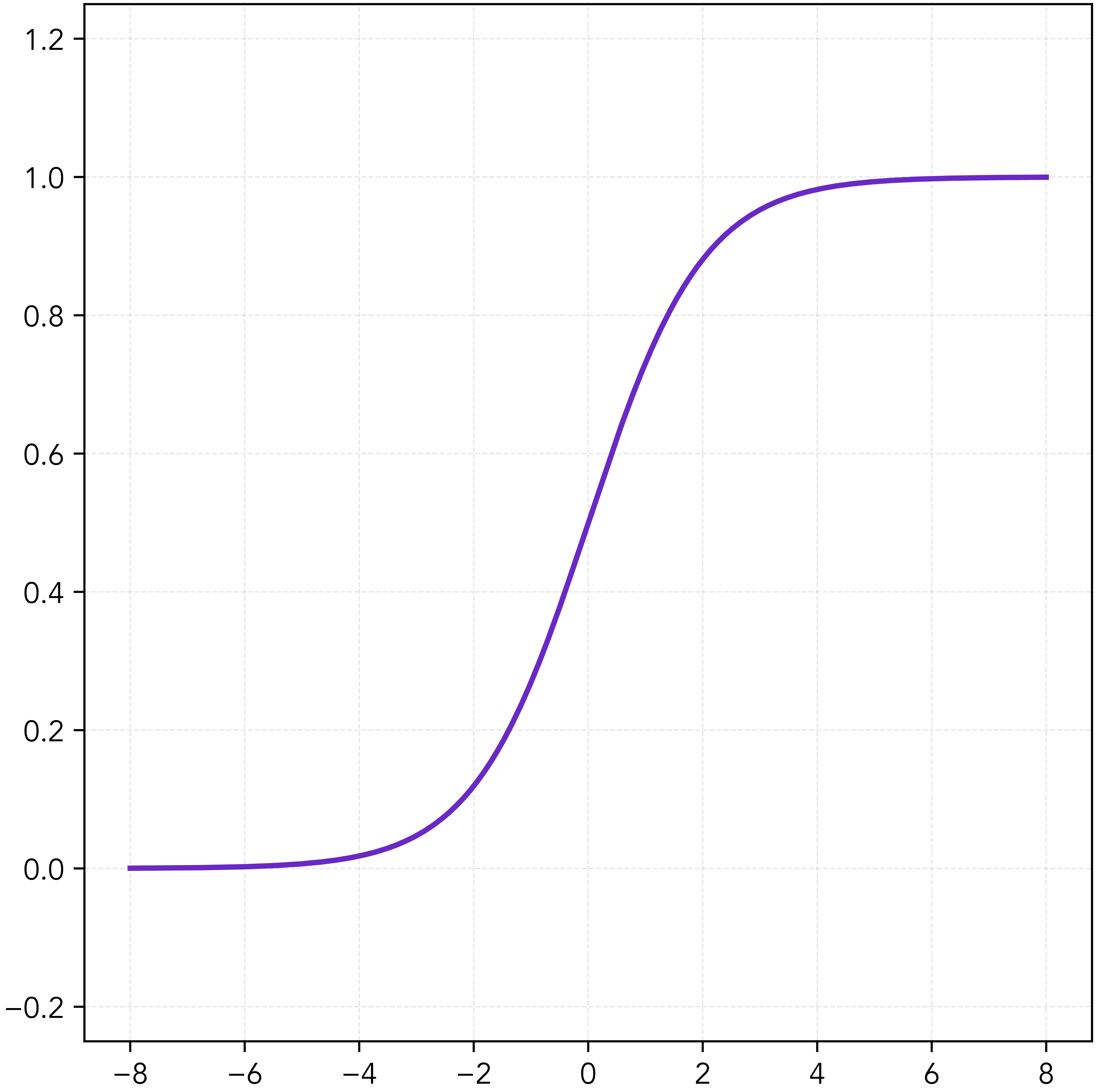
Compose sigmoid with linear functions:

$$\mathscr{F}_{\text{sig}} := \{x \mapsto \phi(w^\top x) : w \in \mathbb{R}^d\}$$

If $w^\top x \gg 0$, $\phi(w^\top x)$ is close to 1.

If $w^\top x \ll 0$, $\phi(w^\top x)$ is close to 0.

If $w^\top x \approx 0$, $\phi(w^\top x)$ is close to $1/2$.



Logistic Loss

Logistic Regression

$$\phi(z) := \frac{1}{1 + \exp(-z)} \text{ and hypothesis class } \mathcal{F}_{\text{sig}} := \{x \mapsto \phi(w^\top x) : w \in \mathbb{R}^d\}$$

What's a reasonable loss function?

If $y = 1$, we want $\phi(w^\top x)$ large (probability of predicting 1).

If $y = -1$, we want $\phi(w^\top x)$ small (probability of predicting -1) $\Longrightarrow 1 - \phi(w^\top x)$ large.

Important property of sigmoid: $1 - \phi(z) = \phi(-z)$.

If $y = -1$, we want $1 - \phi(w^\top x) = \phi(-w^\top x)$ large.

Logistic Loss

Logistic Regression

What's a reasonable loss function?

If $y = 1$, we want $\phi(w^\top x)$ large (probability of predicting 1).

If $y = -1$, we want $\phi(-w^\top x)$ large (probability of predicting -1).

Summary. For $y \in \{-1,1\}$, we want $\phi(yw^\top x)$ large $\Longrightarrow$ Smaller loss for larger $\phi(yw^\top x)$.

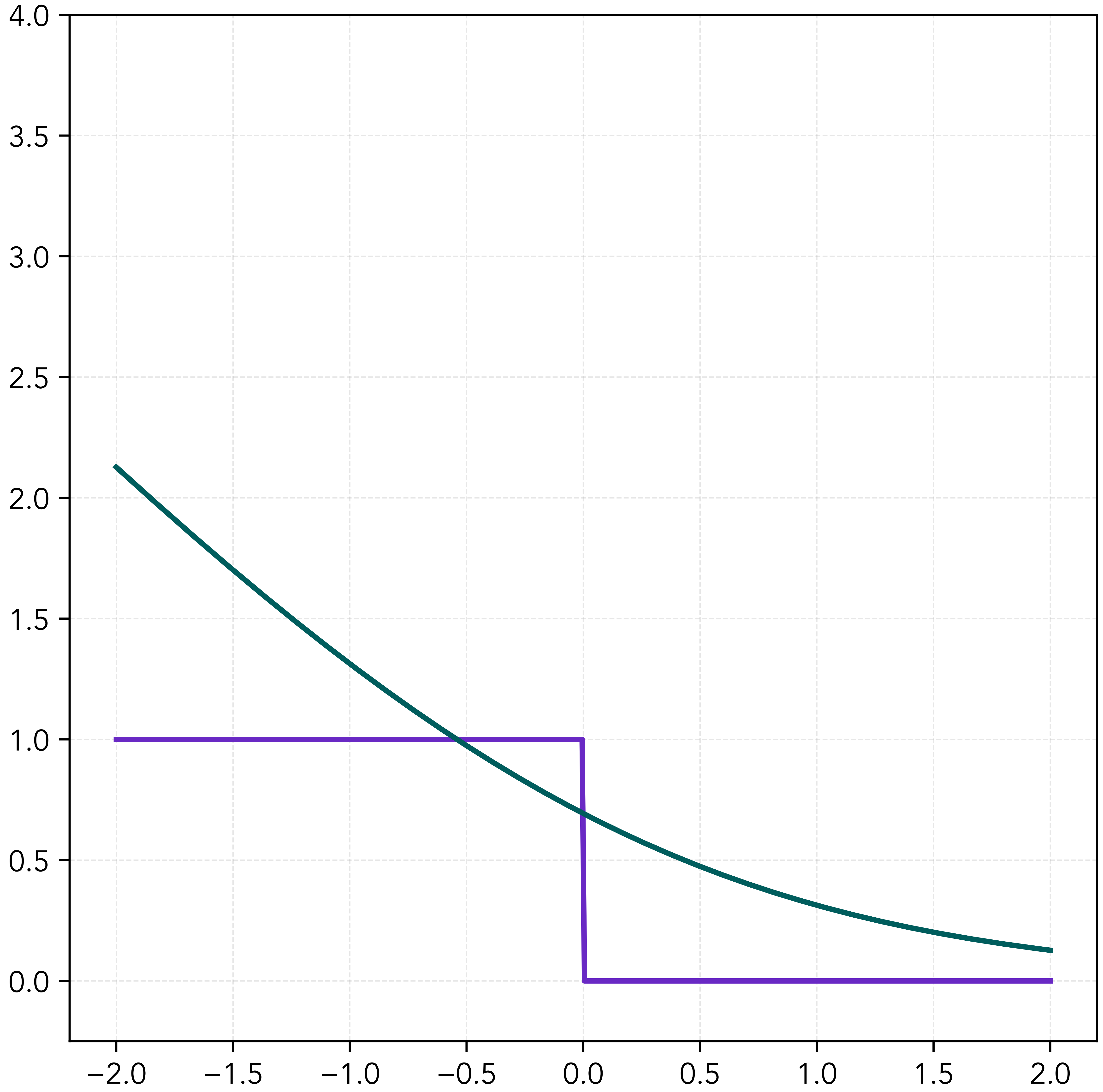
$$-\log(\phi(yw^\top x)) = -\log\left(\frac{1}{1+\exp(-yw^\top x)}\right) = \log(1+\exp(-yw^\top x))$$

Classification Losses

Logistic Loss

Margin: $m = \hat{y}y$

Logistic/Log loss: $\ell_{\log}(m) := \log(1 + e^{-m})$



Logistic loss is **differentiable**.

Always rewards more margin (loss never 0).

Logistic Loss

Logistic Regression

Hypothesis class: $\mathscr{H} = \{h_w(x) = w^\top x : w \in \mathbb{R}^d\}$

Loss: $\ell_{\log}(m) := \log(1 + e^{-m})$ (logistic loss)

Empirical risk minimization:

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y^{(i)} w^\top x^{(i)}))$$

Minimizing this objective is known as logistic regression (a linear **classification** method).

Square Loss

Square loss for classification?

Recall the square loss: $\ell(f(x), y) = (f(x) - y)^2$.

For $y \in \{-1,1\}$, we have $y^2 = 1$, so we can write this in terms of the margin:

$$\ell(f(x), y) = (f(x) - y)^2 = f^2(x)y^2 - 2f(x)y + 1 = (1 - f(x)y)^2 = (1 - m)^2$$

Classification Losses

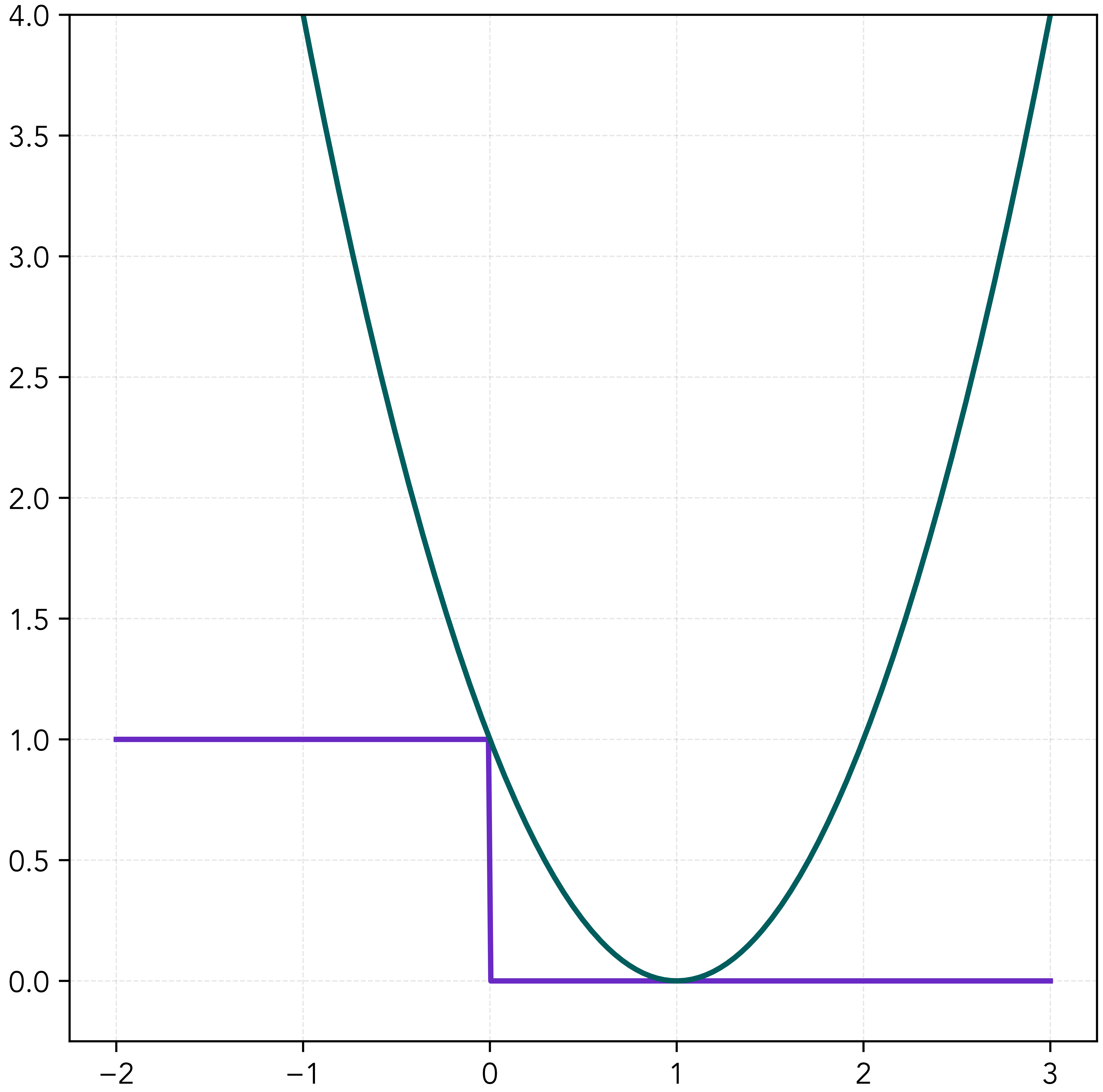
Square Loss

Margin: $m = \hat{y}y$

Square loss: $\ell_{\text{square}}(m) := (1 - m)^2$

Convex and differentiable.

Heavily penalizes outliers (e.g. mislabeled examples).



Classification Losses

Convexity

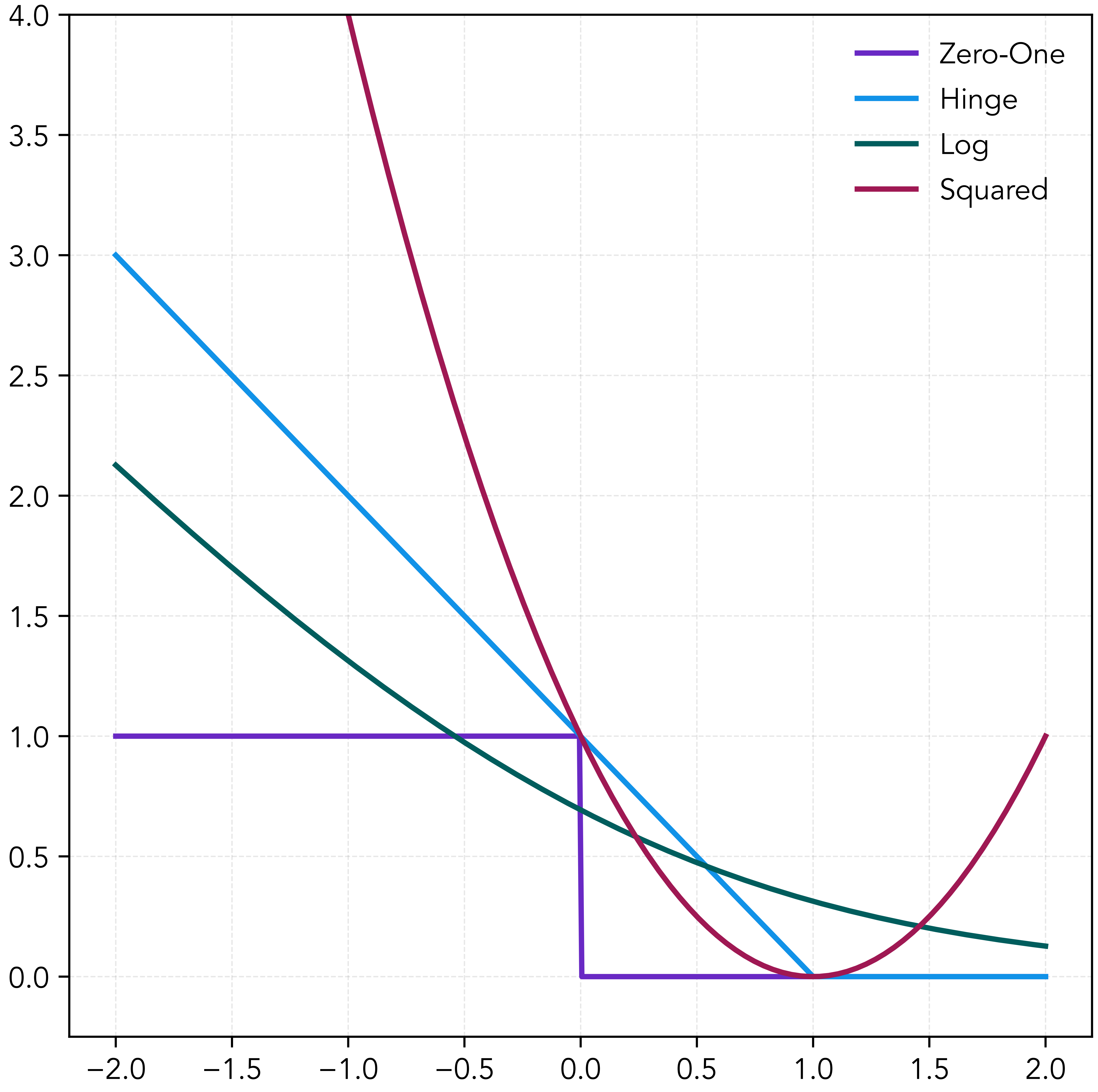
All of these losses have a property in common: **convexity**.

$$\ell_{\text{hinge}}(m) := \max(1 - m, 0)$$

$$\ell_{\text{perc}}(m) := \max(-m, 0)$$

$$\ell_{\log}(m) := \log(1 + e^{-m})$$

$$\ell_{\text{square}}(m) := (1 - m)^2$$



Outline

Model Complexity and Model Selection

Controlling Complexity with Regularization

ℓ_2 Regularization and Ridge Regression

ℓ_1 Regularization and Lasso Regression

Understanding Sparsity

Loss Functions: Regression

Loss Functions: Classification