Indian Institute of Information Technology, Allahabad

Introduction to Machine Learning

Bias, Variance and Regularization

By

Dr. Shiv Ram Dubey
Associate Professor
Computer Vision and Biometrics Lab
Department of Information Technology
Indian Institute of Information Technology, Allahabad
Web: https://profile.iiita.ac.in/srdubey/

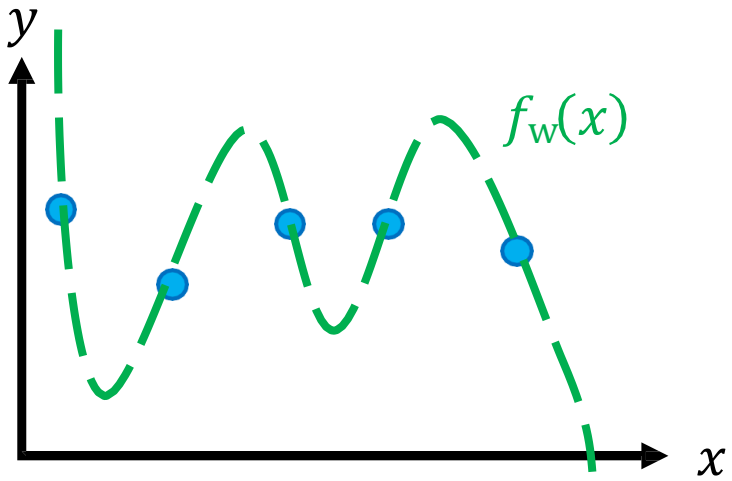
DISCLAIMER

The content (text, image, and graphics) used in this slide are adopted from many sources for academic purposes. Broadly, the sources have been given due credit appropriately. However, there is a chance of missing out some original primary sources. The authors of this material do not claim any copyright of such material.

Question

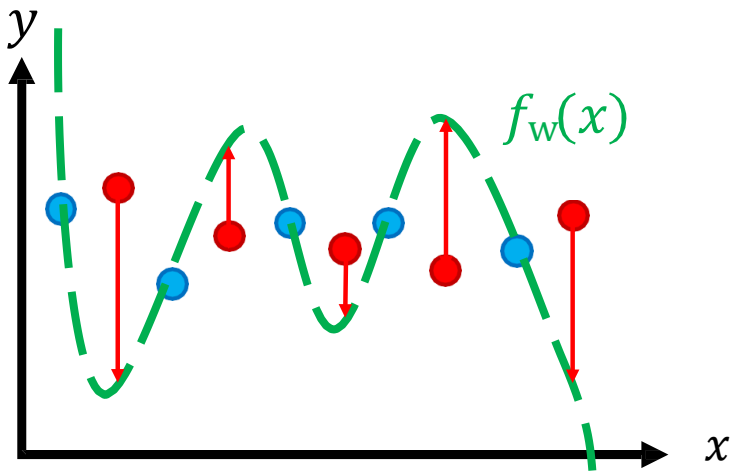
- Why not always throw in lots of features?
  - After all, more features => more expressive hypothesis space!
  - For example, if $\boldsymbol{\phi}(x) = [1, x_1, x_2, {x_1}^2, x_1x_2, {x_2}^2, \dots]$
  - Can fit any n points using an n-th degree polynomial

$$f(x) = \mathrm{w}_0 + \mathrm{w}_1x_1 + \mathrm{w}_2x_2 + \mathrm{w}_3{x_1}^2 + \mathrm{w}_4x_1x_2 + \mathrm{w}_5{x_2}^2 + \cdots$$



Prediction

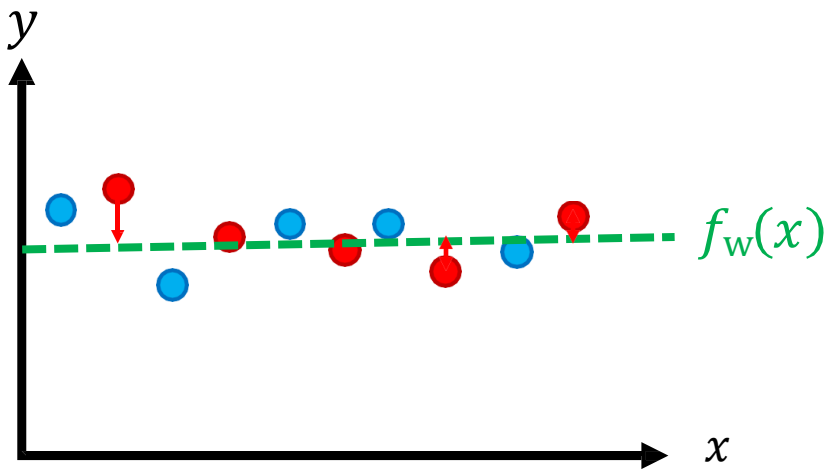
- **Issue:** The goal in machine learning is **prediction**
  - Given a **new** input x, predict the label $y = fw(x)$



The errors on new inputs is very large!

Prediction

- **Issue:** The goal in machine learning is **prediction**
  - Given a **new** input x, predict the label $y = fw(x)$



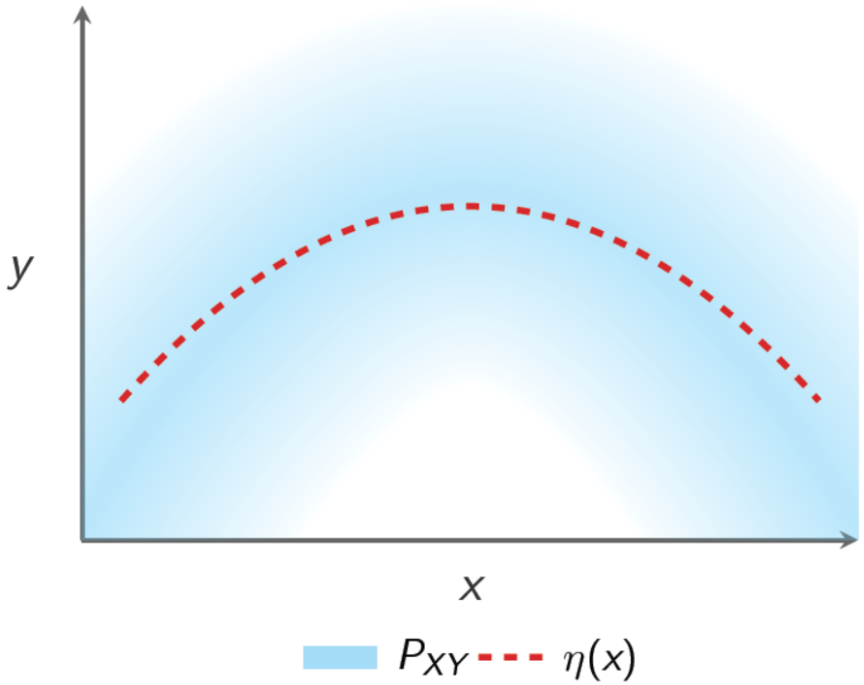
Vanilla linear regression actually works better!

Training vs. Test Data

- **Training data:** Examples $Z = \{(x, y)\}$ used to fit our model

- **Test data:** New inputs x whose labels y we want to predict

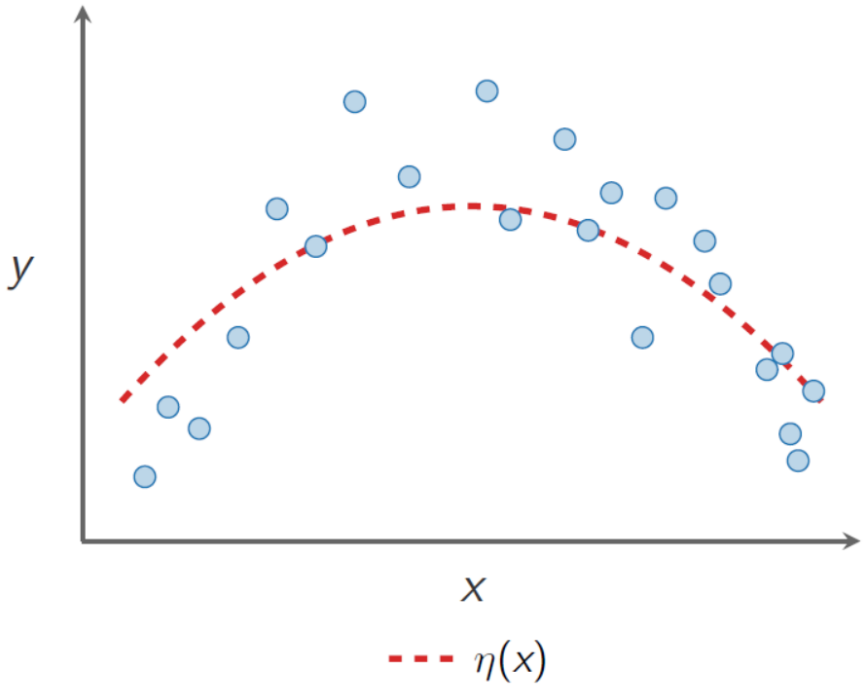
Statistical Learning



Goal: Minimize error $\mathbb{E}_{XY}[(Y - \eta(X))^2]$
Optimal predictor: $\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$

In practice, we can't find the optimal predictor $\eta(x)$ because we don't know P_{XY}.

Statistical Learning

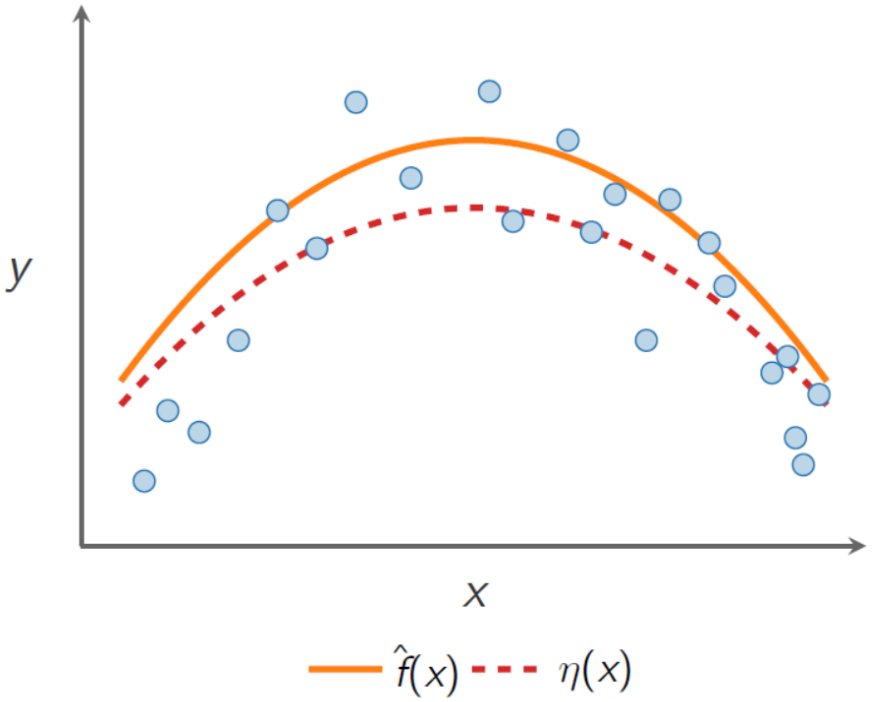


Goal: Minimize error $\mathbb{E}_{XY}[(Y - \eta(X))^2]$
Optimal predictor: $\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$

We only have samples

$$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY} \text{ for } i = 1, \ldots, n$$

Statistical Learning



Goal: Minimize error $\mathbb{E}_{XY}[(Y - \eta(X))^2]$
Optimal predictor: $\eta(x) = \mathbb{E}_{Y|X}[Y|X = x]$

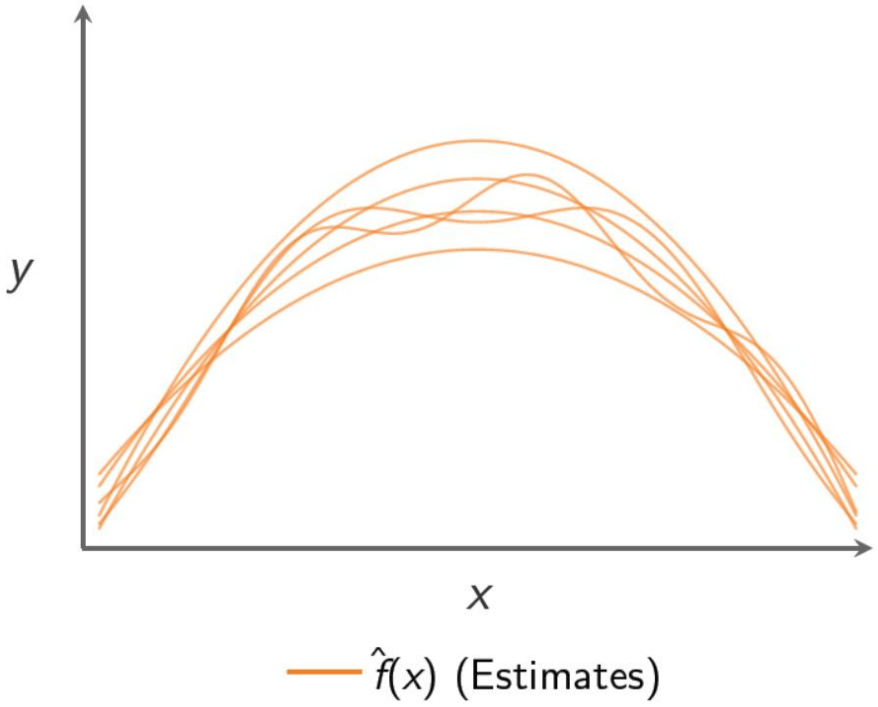
We only have samples

$$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY} \text{ for } i = 1, \ldots, n$$

and are restricted to a function class (e.g. linear), so we compute

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Statistical Learning



We only have samples

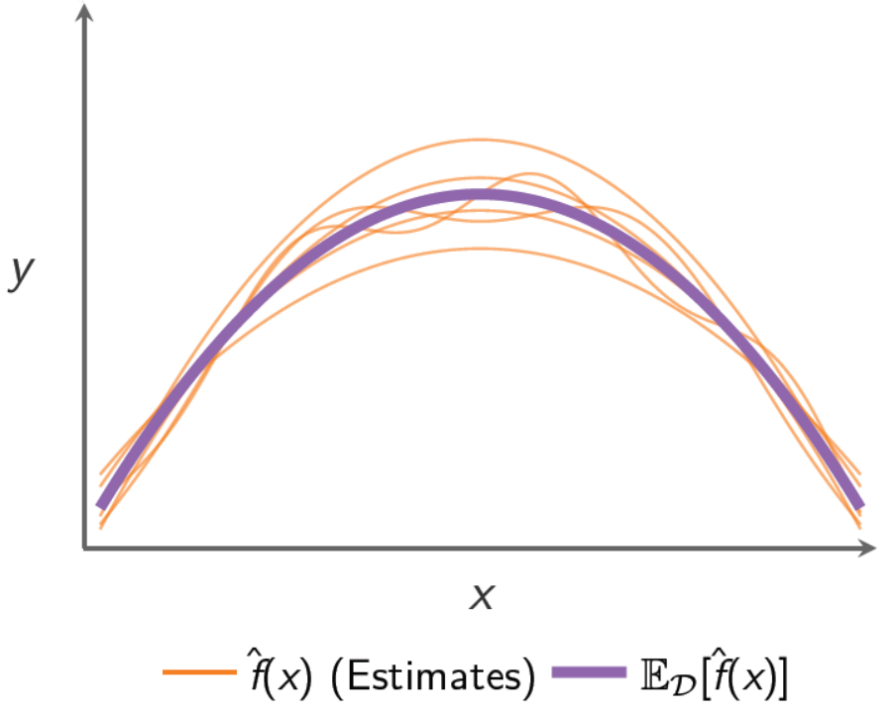
$$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY} \text{ for } i = 1, \ldots, n$$

and are restricted to a function class (e.g. linear), so we compute

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Each draw $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ results in a different $\hat{f}$

Statistical Learning



We only have samples

$$(x_i, y_i) \overset{i.i.d.}{\sim} P_{XY} \text{ for } i = 1, \ldots, n$$

and are restricted to a function class (e.g. linear), so we compute

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Each draw $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ results in a different $\hat{f}$

Prediction Error

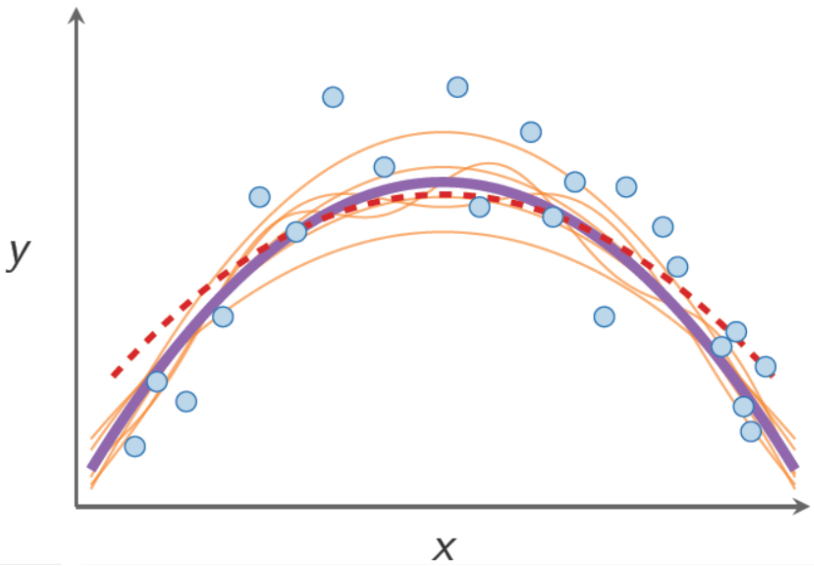
Total prediction error = $\underbrace{\text{bias}^2 + \text{variance}}_{\text{Learning error}}$ + irreducible error

- **Learning error** is because our models are imperfect. It is caused by either using too simple of a model or not enough data to learn the model accurately.
- **Irreducible error** is inherent to the problem, and cannot be reduced! It is caused by stochastic label noise.

- How the bias-variance trade-off (above) can help us design and select better models?

One-slide summary: bias-variance trade-off

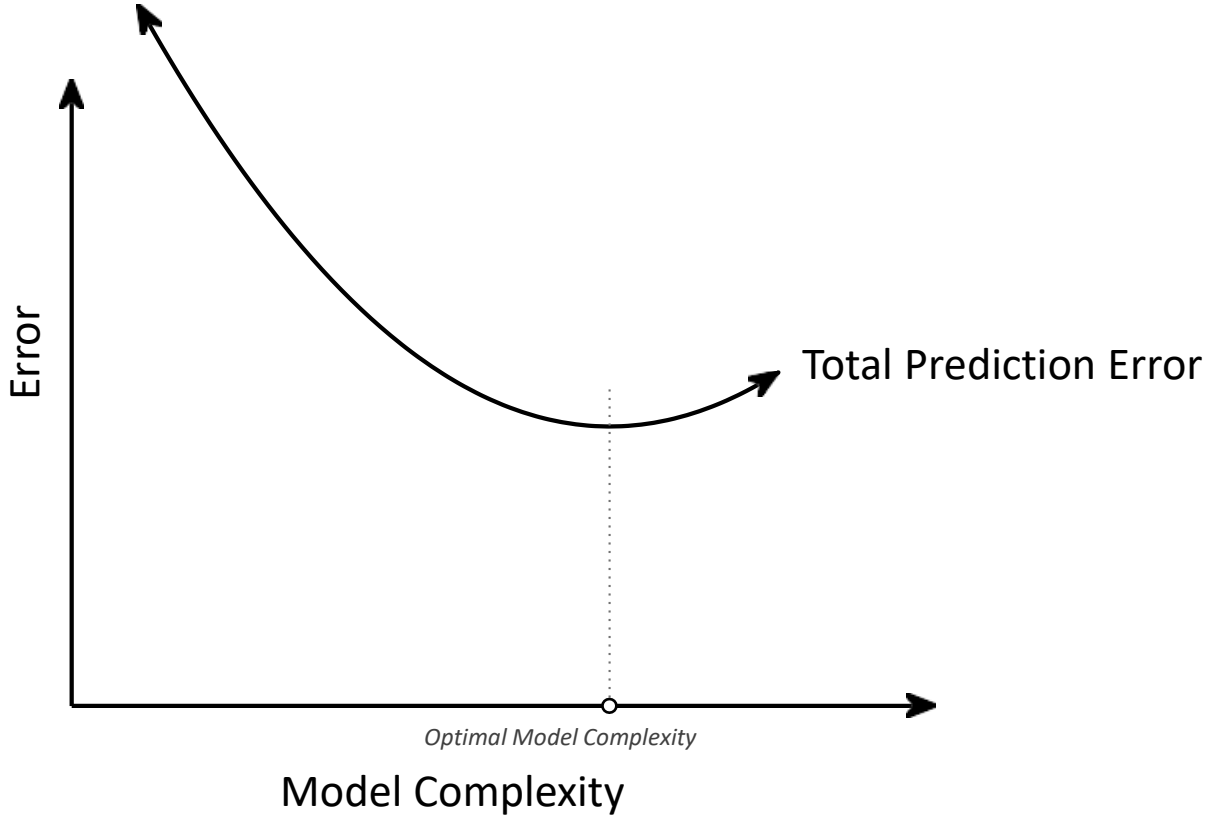
$$\underbrace{\mathbb{E}_{Y|X}[\mathbb{E}_{\mathcal{D}}[(Y-\hat{f}_{\mathcal{D}}(x))^2]|X=x]}_{\text{total prediction error}} = \underbrace{\mathbb{E}_{Y|X}[(Y-\eta(x))^2|X=x]}_{\text{irreducible error}} + \underbrace{\underbrace{(\eta(x)-\mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2}_{\text{bias}^2} + \underbrace{\mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)]-\hat{f}_{\mathcal{D}}(x))^2]}_{\text{variance}}}_{\text{learning error}}$$



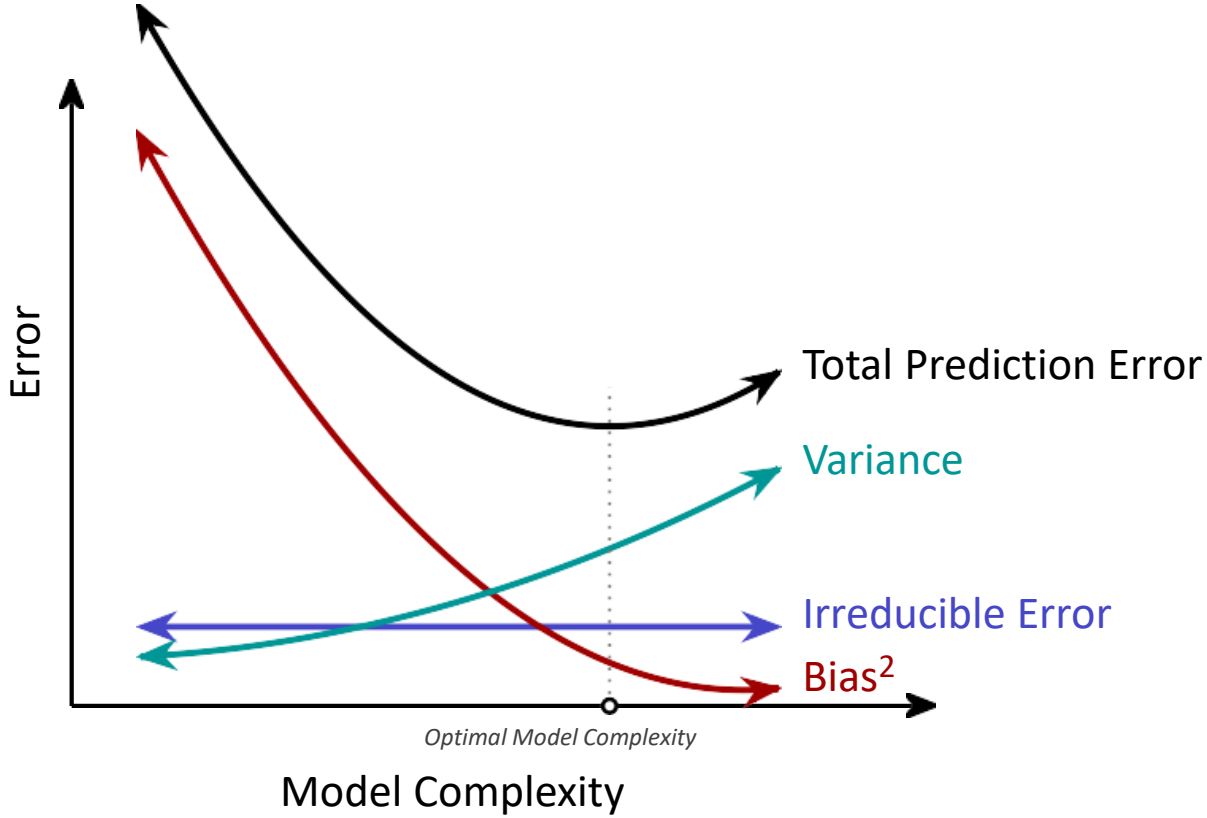
Total prediction error: samples vs. predictor $\hat{f}(x)$

1. **Irreducible error:** samples vs. optimal predictor $\eta(x)$. Cannot be reduced, inherent to the problem!
2. **Learning error:** optimal predictor $\eta(x)$ vs. prediction $\hat{f}(x)$. Breaks down into:
   - 2a. **Bias:** optimal predictor $\eta(x)$ vs. expected predictor $\mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)]$.
   - 2b. **Variance:** predictor $\hat{f}(x)$ vs. expected predictor $\mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)]$.

Bias-Variance Tradeoff

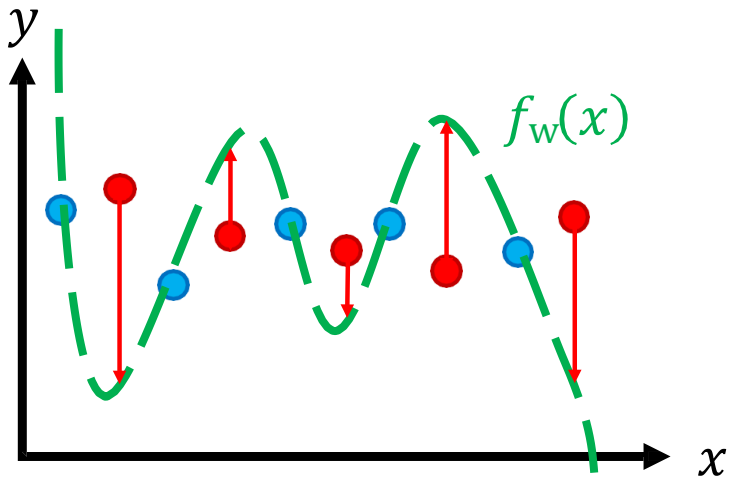


Bias-Variance Tradeoff



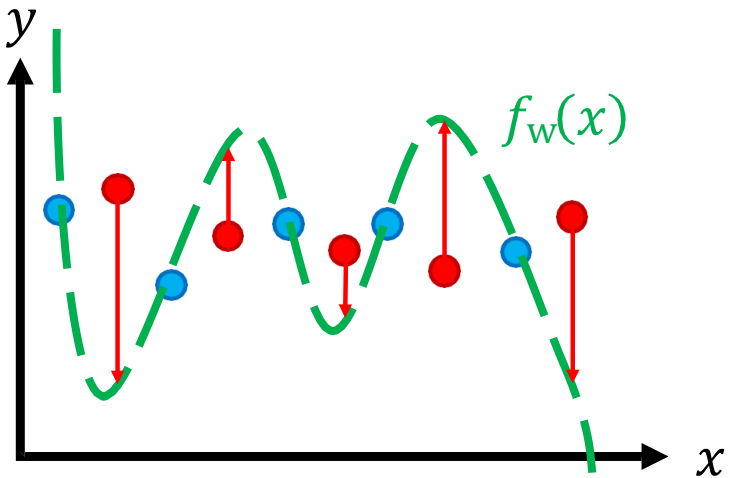
Overfitting vs. Underfitting

- **Overfitting**
  - Fit the **training data** Z well
  - Fit new **test data** (x, y) poorly
    - L_{train} is small
    - L_{test} is large

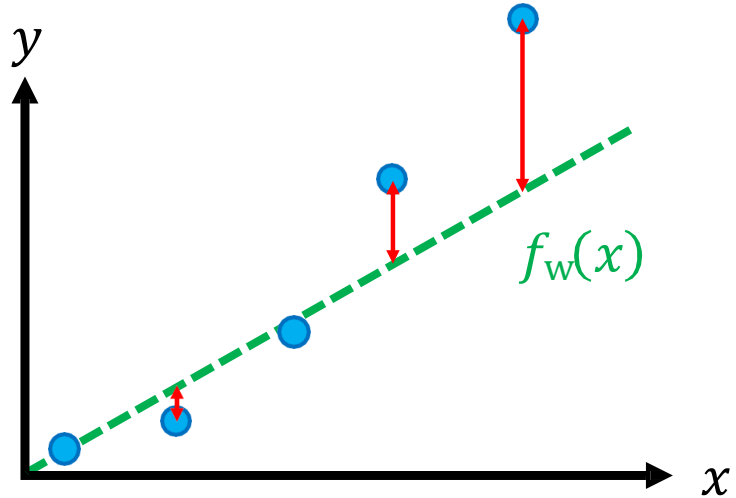


Overfitting vs. Underfitting

- **Overfitting**
  - Fit the **training data** Z well
  - Fit new **test data** (x, y) poorly
    - L_{train} is small
    - L_{test} is large



- **Underfitting**
  - Fit the **training data** Z poorly
  - (Also fit new **test data** (x, y) poorly)
    - L_{train} is large
    - L_{test} is large



Role of Capacity

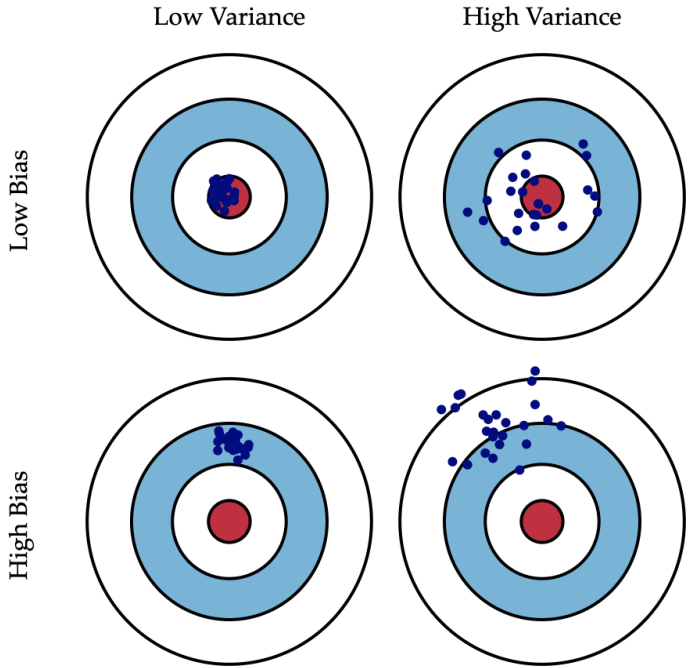
- **Capacity** of a model family captures "complexity" of data it can fit
  - Higher capacity → more likely to overfit (model family has high **variance**)
  - Lower capacity → more likely to underfit (model family has high **bias**)

- For linear regression, capacity roughly corresponds to feature dimension d
  - i.e., number of features in $\phi(x)$

Definitions: “Bias” and “Variance”

Imagine you draw k training datasets from the same probability distribution over data, and each time fit your model $\{f_w\}_{1:k}$ to it.

- “Variance”: how much do those fitted functions $\{f_w\}_{1:k}$ differ amongst each other, on average over the data distribution?
- “Bias” : how much does the average of all those fitted functions $\{f_w\}_{1:k}$ deviate from the “true” function over the data distribution?

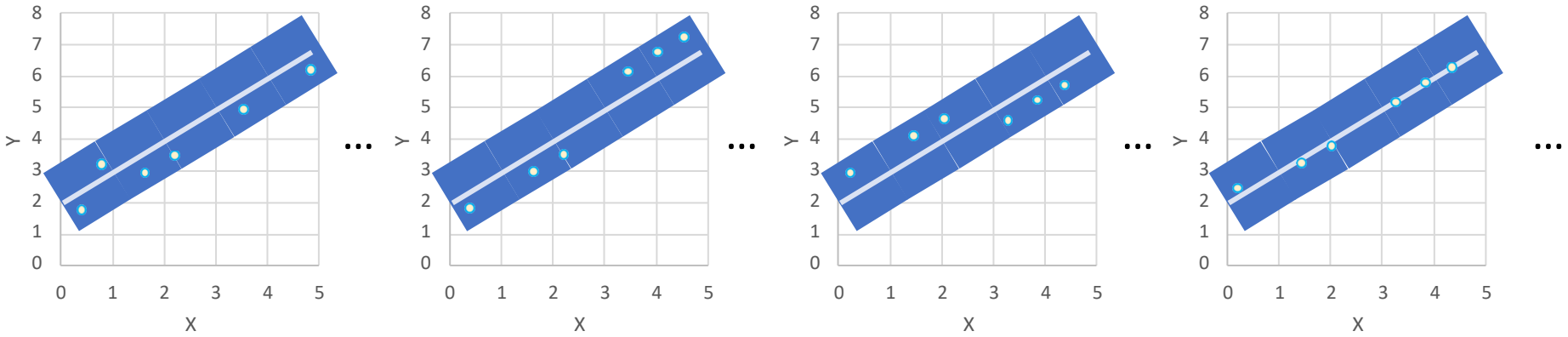


Scott Fortmann-Roe
http://scott.fortmann-roe.com/docs/BiasVariance.html

Drawing Multiple Training Datasets

Consider a linear "true function" $f^*(x) = x + 2$ that generates labels y_i for training data with uniform measurement noise in [-1, +1].

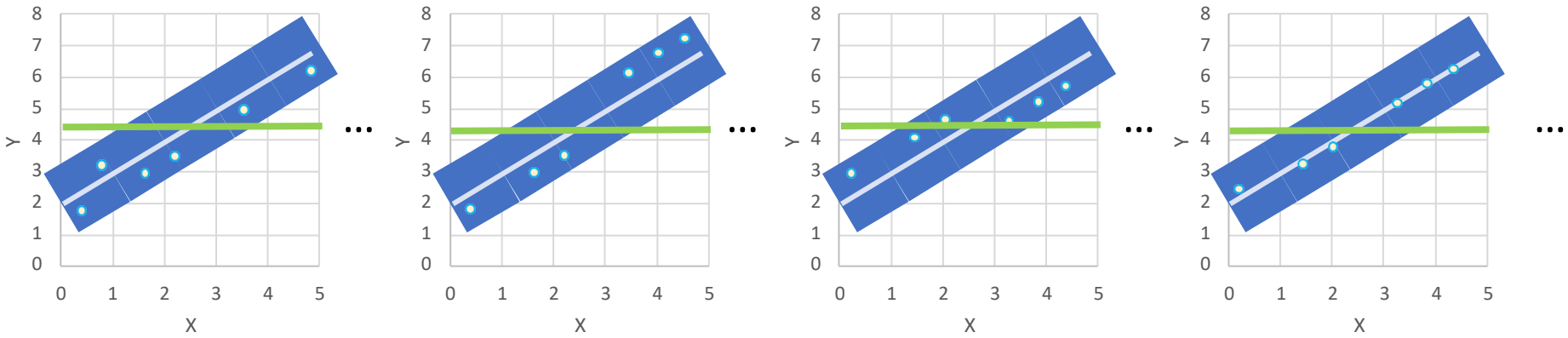
Let us draw $k \to \infty$ training sets of $n = 6$ samples each, drawn from $P(X, Y)$.



Different *Constant* Fits

What if the hypothesis class was the constant function class

$$f_{\mathrm{w}}(x) = \mathrm{w}_0$$



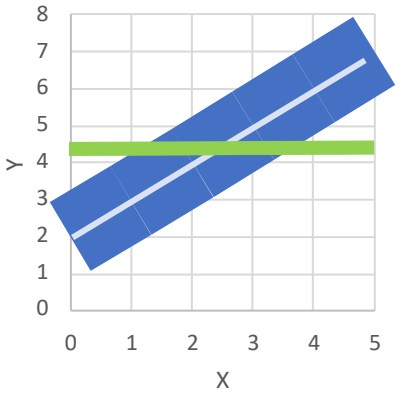
Different *Constant* Fits

What if the hypothesis class was the constant function class

$$f_w(x) = w_0$$

Almost identical fits

"low variance"



Average fit far from the true function

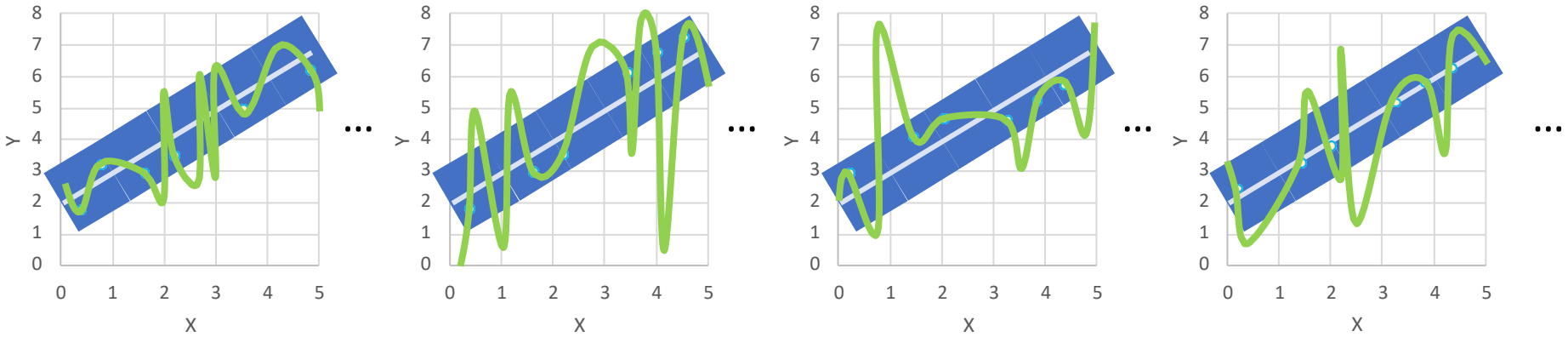
"high bias"

Theoretical result: Generalization MSE $\approx$ ``Bias'' + ``Variance''

Different *10th Degree Curve* Fits

What if the hypothesis class was instead a 10^{th} degree monomial

$$f_{\mathrm{w}}(x) = \mathrm{w}_0 + \mathrm{w}_1 x + \mathrm{w}_2 x^2 + \mathrm{w}_3 x^3 + \mathrm{w}_4 x^4 + \cdots \mathrm{w}_{10} x^{10}$$



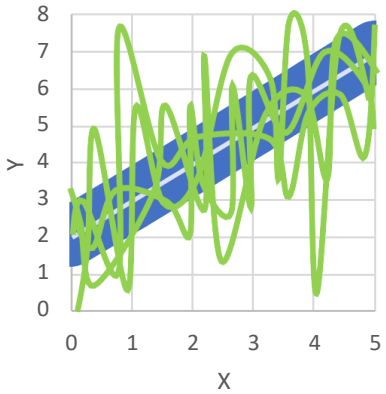
Different *10th Degree Curve* Fits

What if the hypothesis class was instead a 10^{th} degree monomial

$$f_w(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + \cdots w_{10} x^{10}$$

Very different fits

"high variance"



Average fit close to the true function

"low bias"

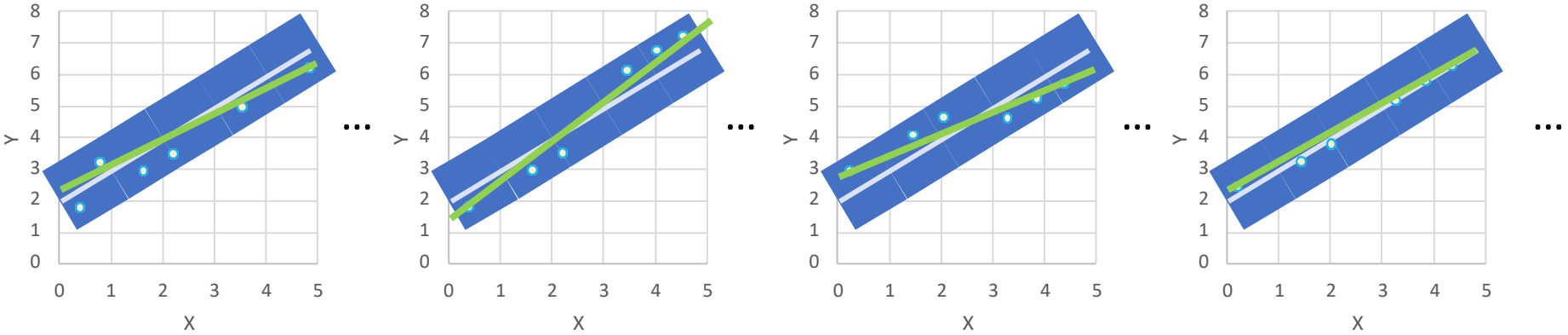
Theoretical result: Generalization MSE $\approx$ ``Bias'' + ``Variance''

Different *Linear* Fits

Say, our hypothesis class is a line:

$$f_{\mathrm{w}}(x) = \mathrm{w}_0 + \mathrm{w}_1 x_1$$

Fit by minimizing MSE with any optimizer. What would the resulting line look like?



Slightly different fits

Different *Linear* Fits

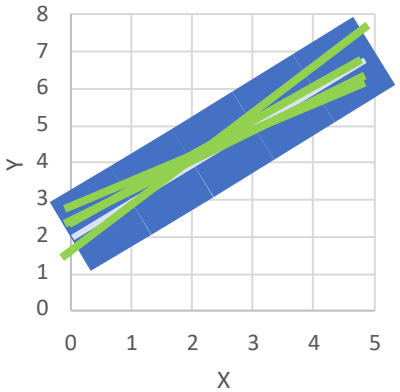
Say, our hypothesis class is a line:

$$f_w(x) = w_0 + w_1 x_1$$

Fit by minimizing MSE with any optimizer. What would the resulting line look like?

Quite similar fits

"low variance"



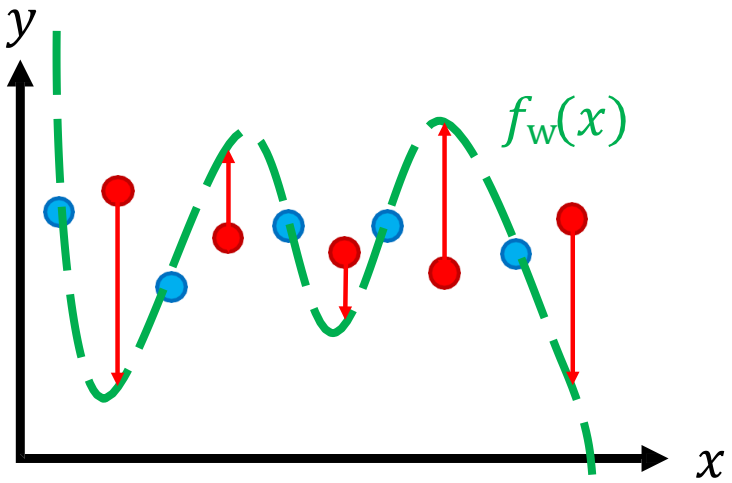
Average fit close to the true function

"low bias"

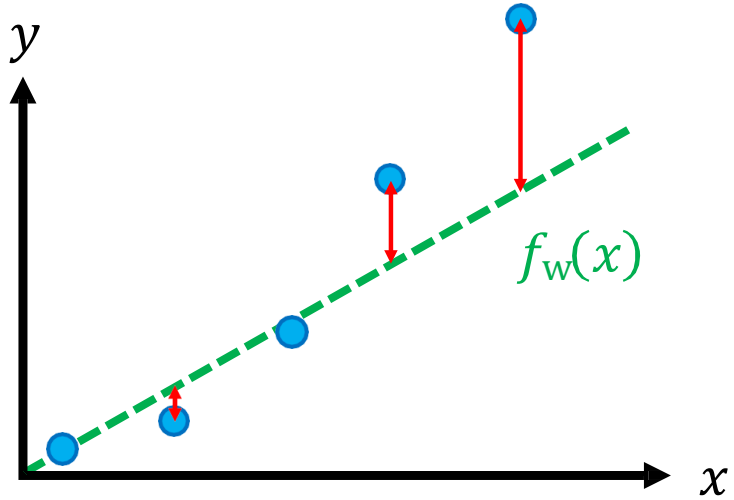
Theoretical result: Generalization MSE $\approx$ ``Bias'' + ``Variance''

Bias-Variance Tradeoff

- **Overfitting (high variance)**
  - High capacity model capable of fitting complex data
  - Insufficient data to constrain it



- **Underfitting (high bias)**
  - Low capacity model that can only fit simple data
  - Sufficient data but poor fit

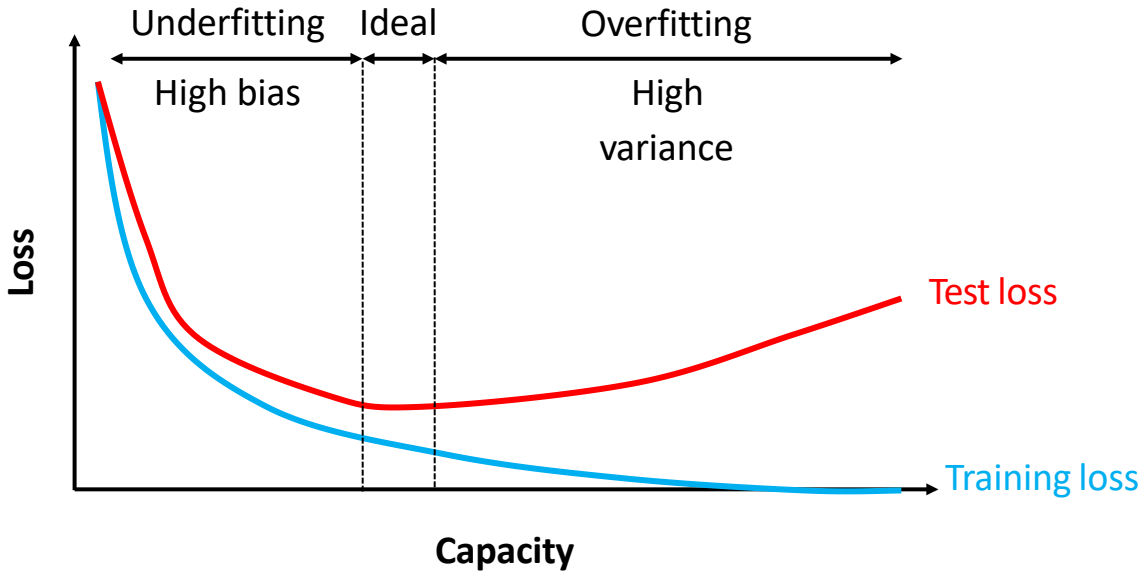


Bias-Variance Tradeoff

Under/Over -Fitting & Model Capacity

Expanding the hypothesis class usually leads to higher variance, lower bias.

(e.g. when adding new dimensions to the feature map)



Slide by Padhraic Smyth, UCIrvine

Combating Underfitting & Overfitting

How To Fix Underfitting/Overfitting?

Three main options:

- Choose the right model family (not too complex, not too simple)
- Improve the training dataset (i.e., collect more data)
- Choose the right loss function

Bias-Variance Tradeoff For Linear Regression

- For linear regression with feature maps, increasing feature dimension d'…
  - Tends to **increase capacity**
  - Tends to **decrease bias** but **increase variance**

- Need to construct basis function $\boldsymbol{\phi}$ to balance tradeoff between bias and variance
  - **Rule of thumb:** You will need $n \approx d' \log d'$ samples, if your $\boldsymbol{\phi}$ has dimension d'

- A large fraction of data science work is data cleaning + feature engineering.

How to Fix Underfitting/Overfitting?

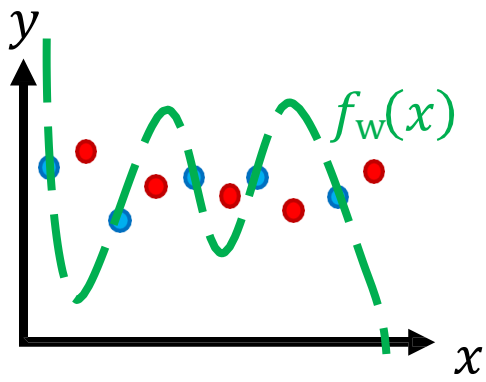
Three main options:

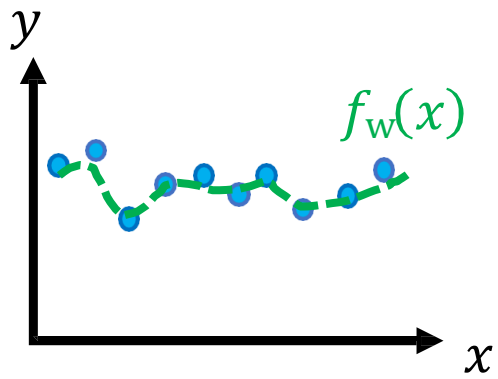
- Choose the right model family (not too complex, not too simple)
- Improve the training dataset (i.e., collect more data)
- Choose the right loss function

The Effect of Dataset Size

Increasing number of examples n in the data…

- Tends to **keep bias fixed** and **decrease variance**
- Tends to **decrease generalization MSE**





The Effect of Dataset Size

As dataset size grows:

- Generalization error ($\approx$ ``Bias'' + ``Variance'') is dominated by bias.
- To reduce error, we select high capacity, low bias models.

Larger datasets have room for expanded hypothesis classes.

How to Fix Underfitting/Overfitting?

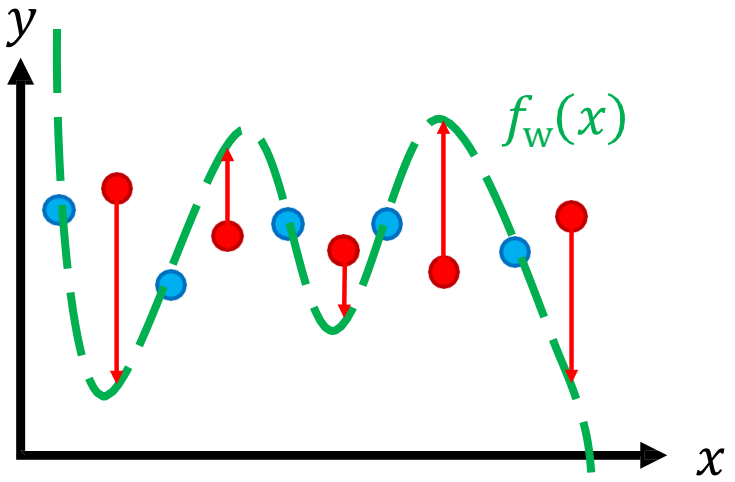
Three main options:

- Choose the right model family (not too complex, not too simple)
- Improve the training dataset (i.e., collect more data)
- Choose the right loss function

Regularization: Modifying the Loss Function

- **Intuition:** We *only* asked the ML algorithm to fit the training data as well as possible, so it produced overly complex fits → "Overfitting"

$$L(\mathrm{w}; Z) = \text{Train MSE}$$



- **Solution:** we will ask the model to produce a *"simple fit"* to the training data.

$$L(\mathrm{w}; Z) = \text{Train MSE} + \text{Fit complexity}$$

How to measure this?

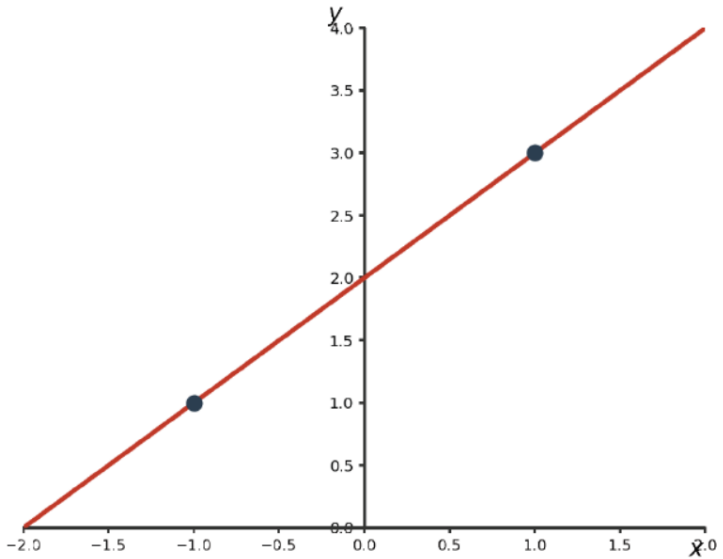
Recall: Mean Squared Error Loss

- Mean squared error loss for linear regression:

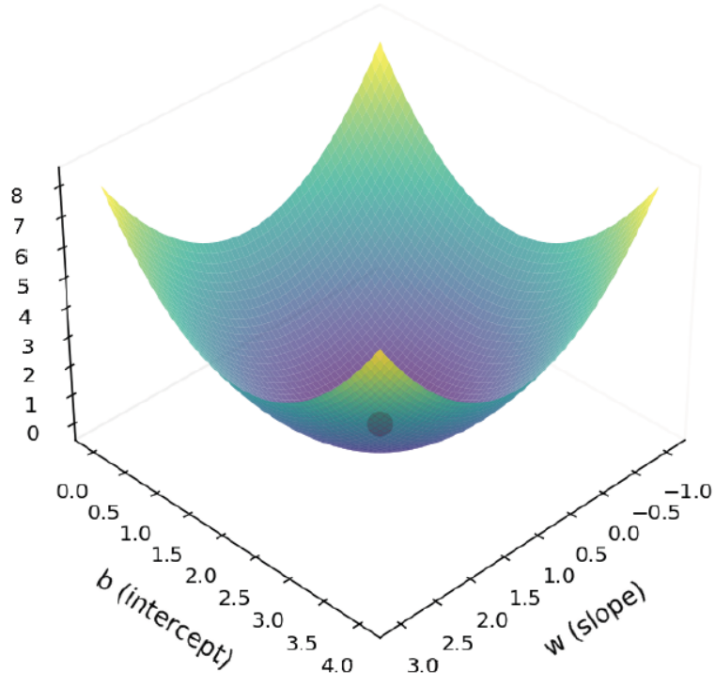
$$L(w\,;\,Z) = \frac{1}{n}\sum_{i=1}^{n}(yi - wTxi)^2$$

Ordinary Least Squares (OLS) works when d ≤ n

Linear fit $(n = 2, d = 2)$



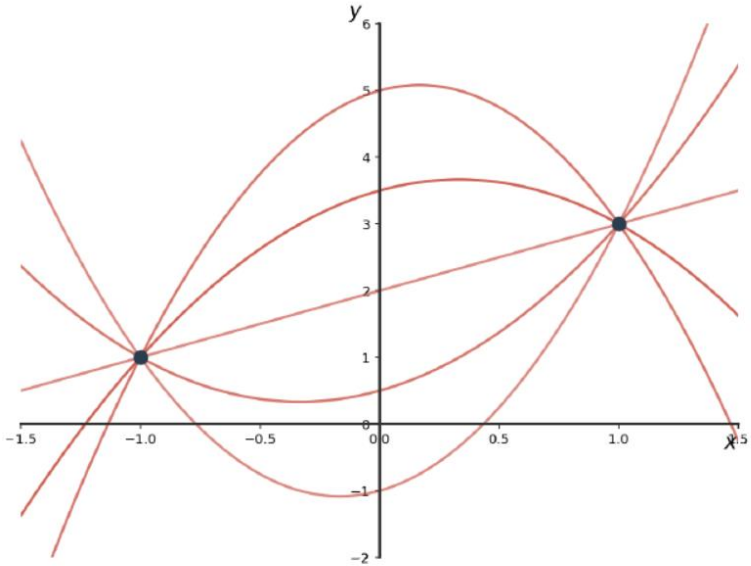
Loss curve



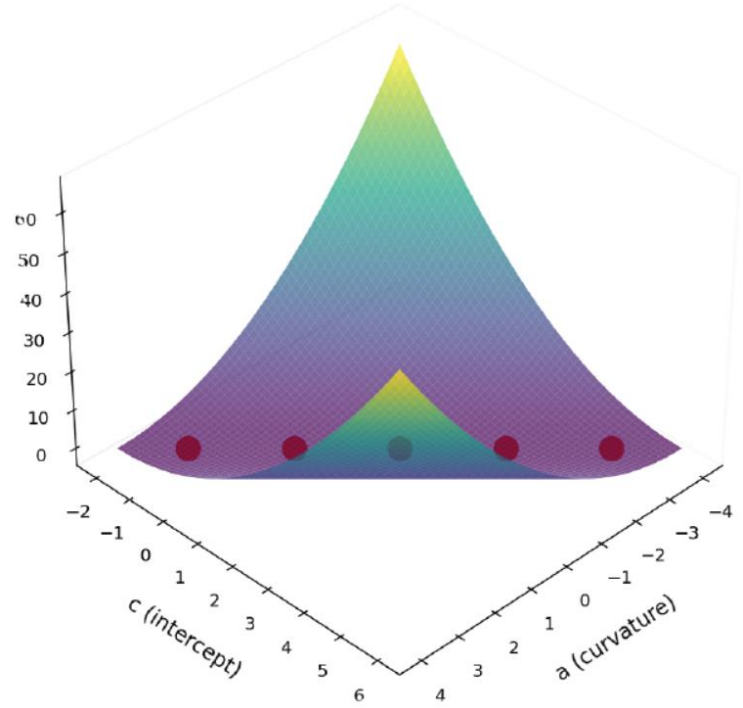
n = 2 and d = 2. Unique optimal solution!

Ordinary Least Squares (OLS) fails when d > n

Quadratic fits ($n = 2, d = 3$)



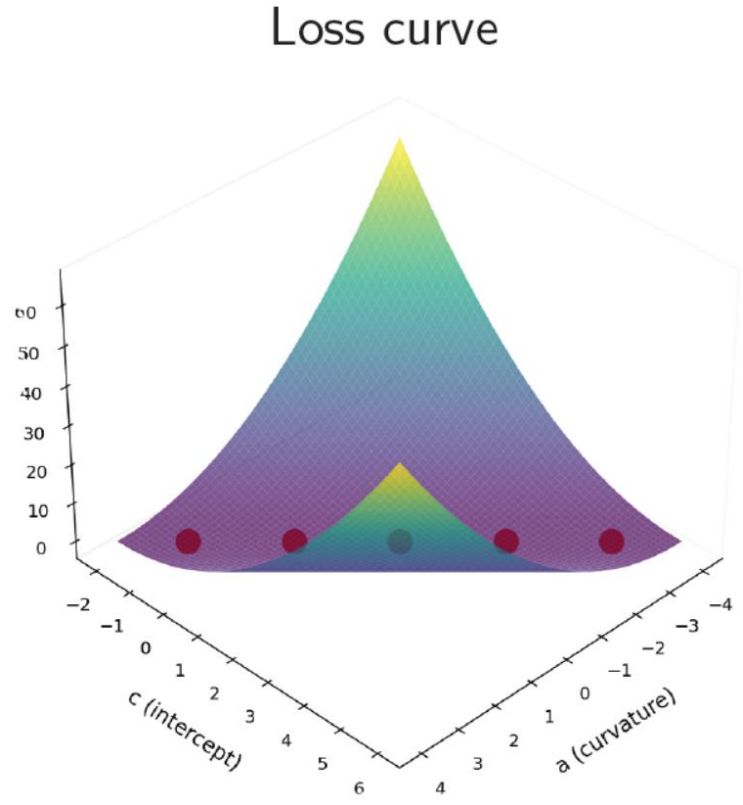
Loss curve



Objective function is flat in some directions – many optimal solutions!

Ordinary Least Squares (OLS) fails when d > n

- When $x \in \mathbb{R}^d$ and $d > n$ the objective function is flat in some directions
- Optimal solution is not unique and unstable due to lack of curvature
- Small changes in training data result in large changes in solution!
- Often the magnitudes of $\hat{w}$ are very large



Regularization

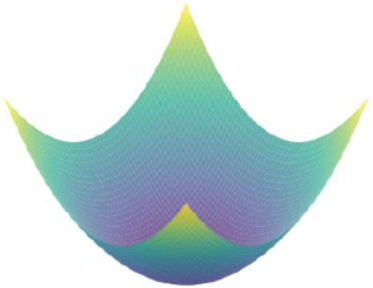
Regularization introduces a ‘complexity’ penalty to encourage ‘simpler’ solutions

- Introduces bias to reduce variance and improve stability (more on this later)
- Regularization is broadly useful
  - When $d > n$
  - Poorly conditioned problems (almost flat along some dimensions)
  - Training very large models

L1 and L2 Regularization Overview

Ridge regression adds L2 regularization

$$\hat{w}_{\text{ridge}} = \arg\min_{w} ||Y - Xw||_2^2 + \lambda||w||_2^2$$



LASSO adds L1 regularization

$$\hat{w}_{\text{LASSO}} = \arg\min_{w} ||Y - Xw||_2^2 + \lambda||w||_1$$

Linear Regression with L_2 Regularization

One measure of fit complexity

- **Original loss + regularization:**

$$L(w\,;\,Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - wTxi)^2 + \lambda\,.\,||w||_2^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}(y_i - wTxi)^2 + \lambda\sum_{j=1}^{d}{w_j}^2$$

- λ is a **hyperparameter** that must be tuned (satisfies $\lambda \geq 0$)

Intuition on L_2 Regularization

Why does it help?

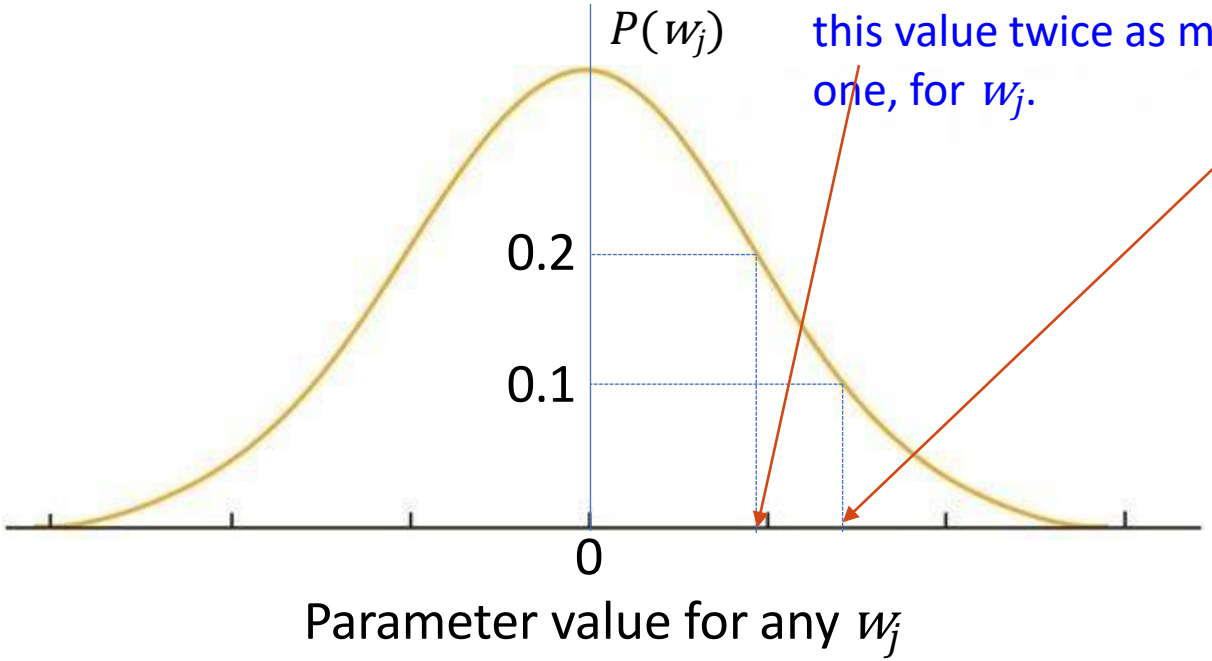
Intuition on L_2 Regularization

Why does it help?

- Encourages "simple" functions
  - This is what $\boldsymbol{L_2}$ regularization does: $\sum_{j=1}^{d} {w_j}^2 = ||w||_2^2 = ||w - 0||_2^2$
  - Pulls coefficients towards 0
  - As $\lambda \rightarrow \infty$, it forces w = 0

Intuition on L_2 Regularization: Gaussian Priors

L2 regularized linear regression amounts to preferring smaller weights according to a Gaussian pdf.

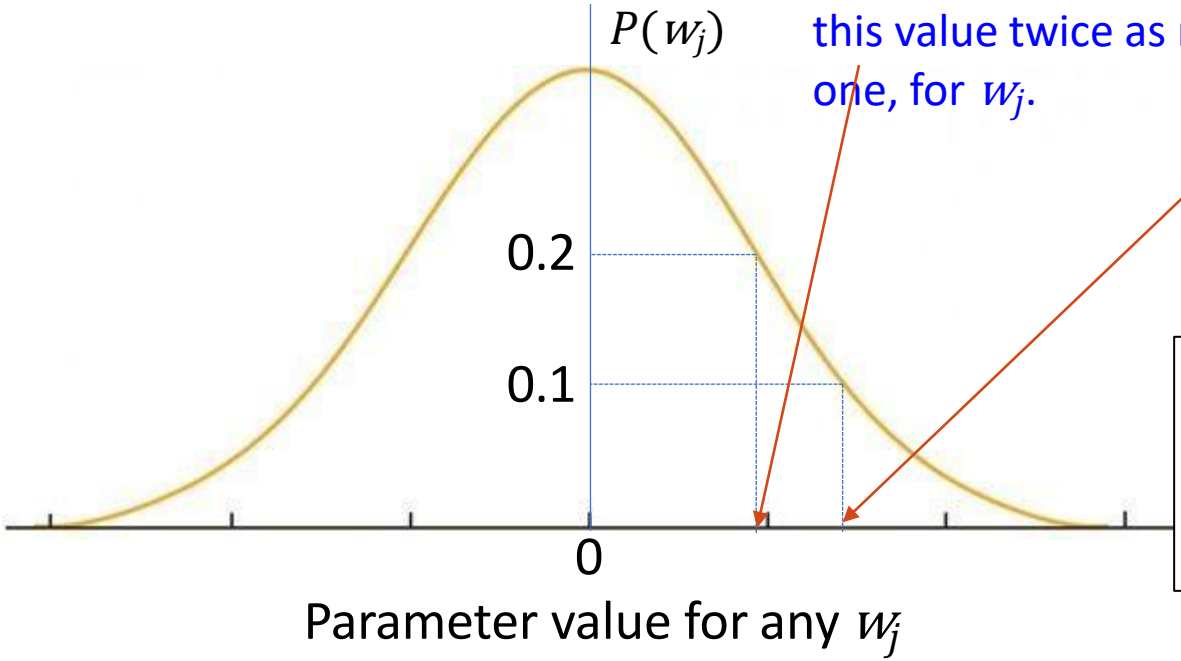


L2 regularization says: before looking at the data fit term, it likes this value twice as much as this one, for w_j.

So the larger value is only selected for the model if it is *much* better for the data fit term (MSE)

Intuition on L_2 Regularization: Gaussian Priors

L2 regularized linear regression amounts to preferring smaller weights according to a Gaussian pdf.



L2 regularization says: before looking at the data fit term, it likes this value twice as much as this one, for w_j.

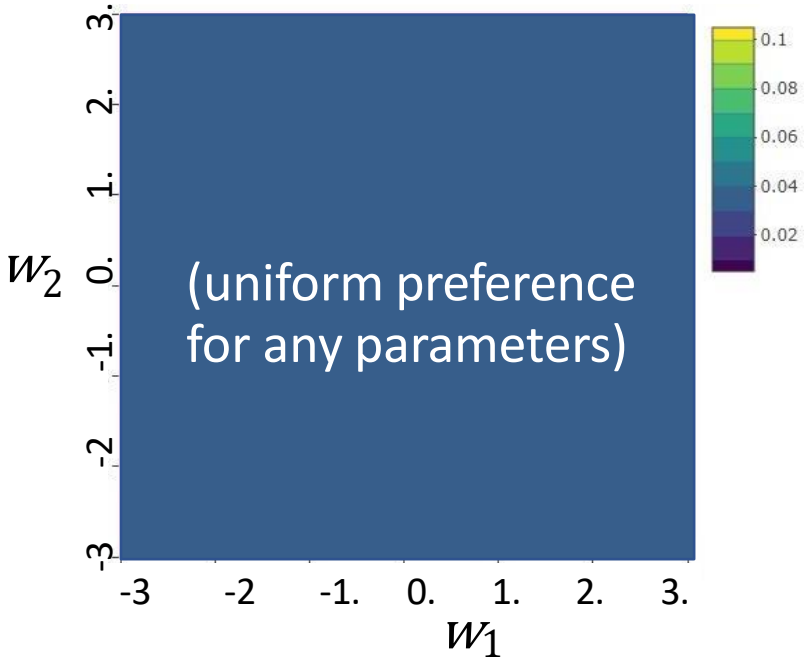
So the larger value is only selected for the model if it is *much* better for the data fit term (MSE)

Q: What happens to the shape of this plot if the value of λ increases?

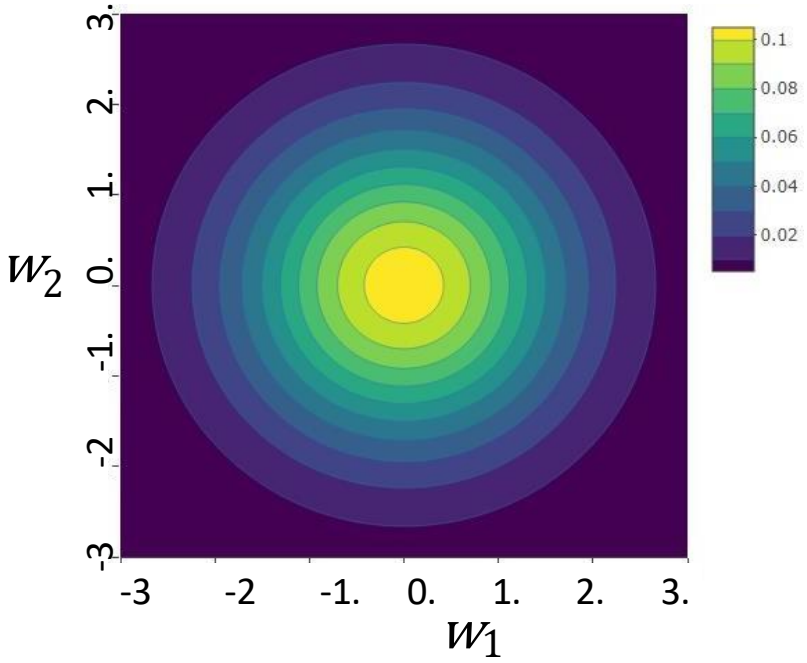
$$\frac{1}{n}\sum_{i=1}^{n}(y_i - wTxi)^2 + \lambda \,.\, ||w||_2^2$$

Intuition on L_2 Regularization: Gaussian Priors

Before regularization



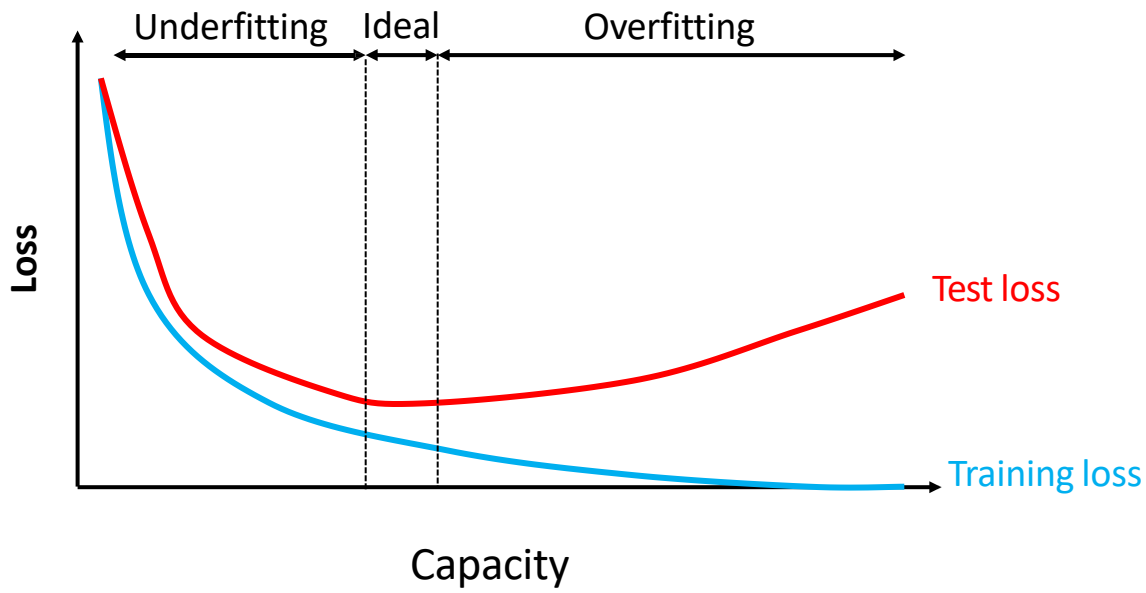
With L2 regularization



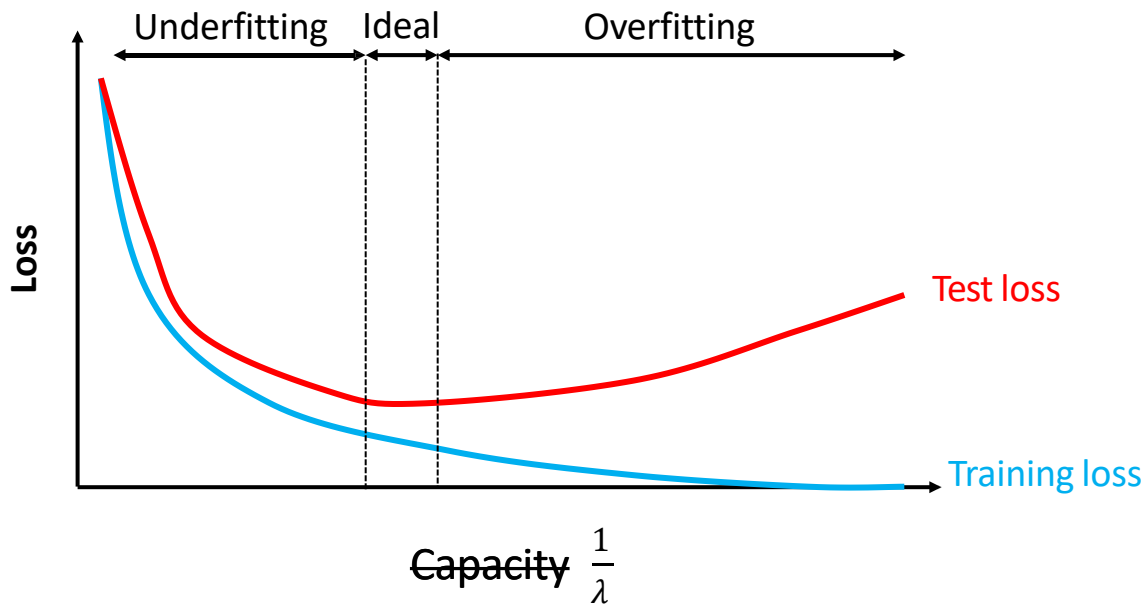
Intuition on L_2 Regularization

- Encourages “simple” functions
- Encouraging w_j’s to have small magnitude also induces a smaller-capacity hypothesis class.
- Use haperparameter λ to tune bias-variance tradeoff

Bias-Variance Tradeoff for Regularization



Bias-Variance Tradeoff for Regularization



General Regularization Strategy

- **Original loss + regularization:**

$$L_{\text{new}}(w; Z) = L(w; Z) + \lambda \cdot R(w)$$

  - Offers a way to express a preference for “simpler” functions in family
  - Typically, regularization is independent of data

Q: For the new parameters $w^*_{new} = \min_w L_{new}$, would their corresponding value of $L(\text{w}; Z)$ be smaller or larger than before regularization?

General Regularization Strategy

- **Original loss + regularization:**

$$L_{\text{new}}(w; Z) = L(w; Z) + \lambda \cdot R(w)$$

  - Offers a way to express a preference for "simpler" functions in family
  - Typically, regularization is independent of data

Q: For the new parameters $w^*_{new} = \min_w L_{new}$, would their corresponding value of $L(\mathrm{w}; Z)$ be smaller or larger than before regularization?

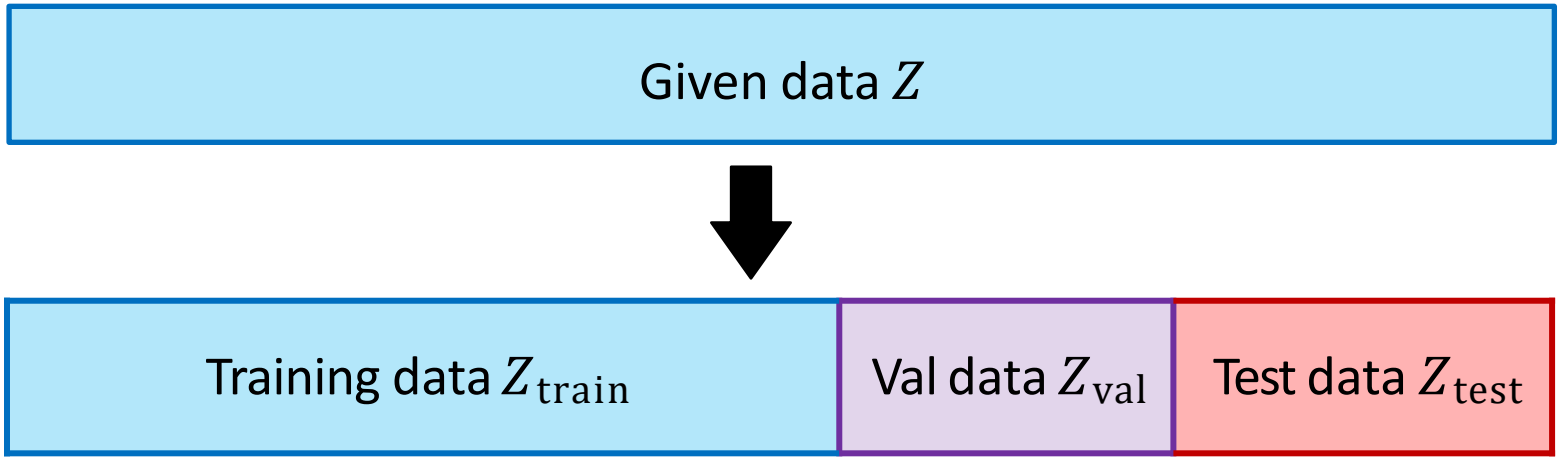
The new objective forces the parameters to balance minimizing the original loss and minimizing the penalty term (like L1 or L2 norm). Hence, to reduce model complexity, *W* moves away from the unregularized optimum, which strictly increases the loss.

Hyperparameter Tuning

- λ is a **hyperparameter** that must be tuned (satisfies $\lambda \geq 0$)
- **Naïve strategy:** Try a few different candidates λ_t and choose the one that minimizes the test loss
- **Problem:** We may overfit the test set!
  - Major problem if we have more hyperparameters
- **Solution:** A new subset of data just for selecting hyperparameters

Train/Val/Test Split for Model Selection

- **Goal:** Choose best hyperparameter λ
  - Can also compare different model families, feature maps, etc.
- **Solution:** Optimize λ on a **held-out validation data**
  - **Rule of thumb:** 60/20/20 split (usually shuffle before splitting)



L_2 Regularized Linear Regression: Ridge Regression

Ridge regression is a modified form of linear regression that adds an L2 penalty term to shrink coefficients, reducing model complexity and preventing overfitting.

- Linear regression with L_2 regularization minimizes the loss

$$L(w\ ;\ Z)\ = \frac{1}{n}\sum_{i=1}^{n}(y_i - w^T x_i)^2 + \lambda \sum_{j=1}^{d} {w_j}^2$$

$$= \frac{1}{n}||Y - Xw||_2^2 + \lambda||w||_2^2$$

  - Gradient is

$$\nabla_w L\ (w; Z) = -\frac{2}{n}X^\top Y + \frac{2}{n}X^\top X\, w + 2\lambda w$$

Strategy 1: Closed-Form Solution

- Gradient is

$$\nabla_{\mathrm{w}} L\left(\mathrm{w}; Z\right) = -\frac{2}{n} X^{\mathsf{T}} Y + \frac{2}{n} X^{\mathsf{T}} X w + 2\lambda w$$

- Setting $\nabla_{\mathrm{w}} L\left(\mathrm{w}; Z\right) = 0$, we have $\left(X^{\mathsf{T}} X + n\lambda I\right) w = X^{\mathsf{T}} Y$

- Always invertible if $\lambda > 0$, so we have

$$w(Z) = \left(X^{\mathsf{T}} X + n\lambda I\right)^{-1} X^{\mathsf{T}} Y$$

Strategy 2: Gradient Descent

- Gradient is

$$\nabla_{\mathrm{w}} L\left(\mathrm{w}; Z\right) = -\frac{2}{n} X^{\top} Y + \frac{2}{n} X^{\top} X w + 2\lambda w$$

- Same algorithm as vanilla linear regression (i.e., Ordinary Least Squares (OLS))
- **Intuition:** The extra term λw in the gradient is **weight decay** that encourages w to be small

What About L_1 Regularization? Lasso Regression

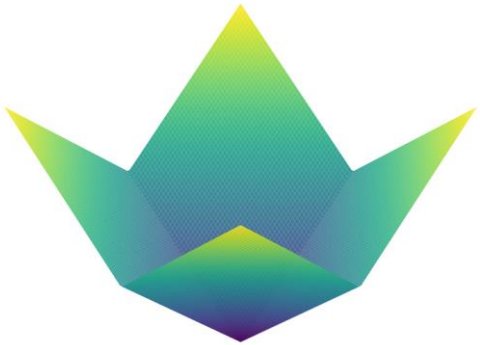
- Introduced by Rob Tibshirani in 1996 paper "Regression shrinkage and selection via the lasso"
- Author of highly-recommended stats books including "An Introduction to Statistical Learning"

What About L_1 Regularization? Lasso Regression

- Lasso regression adds an absolute value penalty (L1 regularization) to reduce overfitting and encourages sparsity through automatic feature selection.
- It shrinks the coefficients of less important variables down to exactly zero.

$$L(w\,;\,Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - wTxi)^2 + \lambda\sum_{j=1}^{d}|w_j|$$

- "Pointy bowl" (convex) → unique solution
- No closed form solution

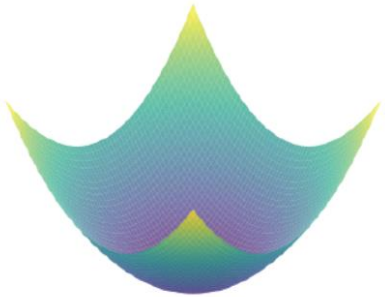


- Gradient descent still works!

Bayesian Prior Interpretation

Ridge regression adds L2 regularization

$$\hat{w}_{\text{ridge}} = \arg\min_w ||Y - Xw||_2^2 + \lambda||w||_2^2$$



$$\hat{w}_{\text{ridge}} = \arg\max_w P(Y|X, w)\, P(w)$$

where $P(w) = \mathcal{N}(0, \frac{\sigma^2}{\lambda} I)$

LASSO adds L1 regularization

$$\hat{w}_{\text{LASSO}} = \arg\min_w ||Y - Xw||_2^2 + \lambda||w||_1$$

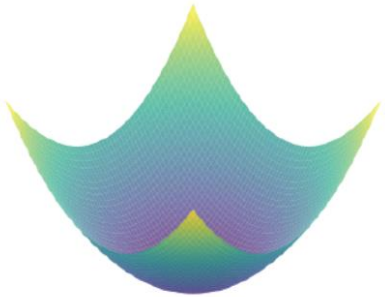
$$\hat{w}_{\text{LASSO}} = \arg\max_w P(Y|X, w)\, P(w)$$

where $P(w) = \text{Laplace}(0, \frac{\sigma^2}{\lambda})$

Bayesian Prior Interpretation

Ridge regression adds L2 regularization

$$\hat{w}_{\text{ridge}} = \arg\min_{w} ||Y - Xw||_2^2 + \lambda ||w||_2^2$$



$$\hat{w}_{\text{ridge}} = \arg\max_{w} P(Y|X, w)\, P(w)$$

where $P(w) = \mathcal{N}(0, \frac{\sigma^2}{\lambda} I)$

Intuition:

- $\lambda \to \infty$, variance of $P(w)$ decreases and distribution becomes a highly "concentrated" prior at $\vec{0}$.
- $\lambda \to 0$, $P(w)$ becomes a "flat" uninformative prior.

Constrained Optimization Interpretation

Penalized least squares has an equivalent constrained version

So far, we've been looking at the regularized optimization

$$\hat{w}_r = \arg\min_w ||Y - Xw||_2^2 + \lambda r(w)$$

where $r(w) = ||w||_2^2$ for ridge regression or $r(w) = ||w||_1$ for LASSO.

Equivalent constrained optimization

The constrained optimization problem is

$$\hat{w}_c = \arg\min_w ||Y - Xw||_2^2$$

$$\text{subject to } r(w) \leq \mu$$

Theorem: For any $\lambda \geq 0$, there exists a $\mu \geq 0$ such that $\hat{w}_c = \hat{w}_r$.

In other words, there are pairs of (λ, μ) with equivalent solutions $\hat{w}_r = \hat{w}_c$

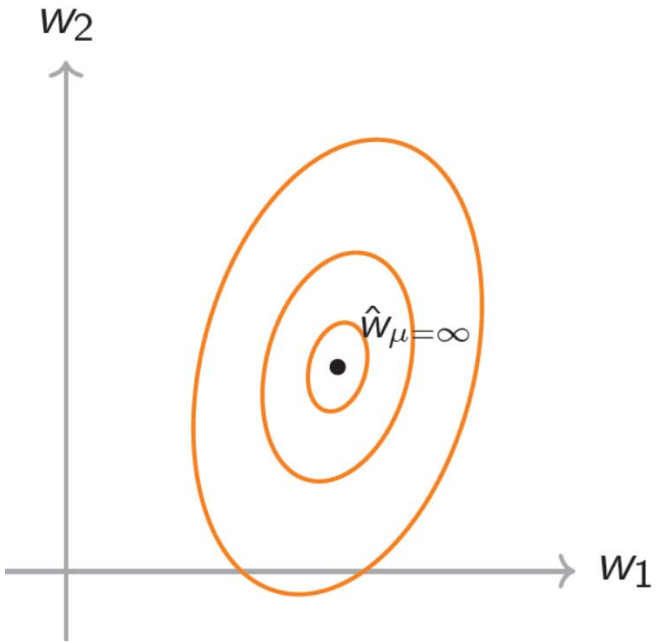
Level Sets Definition

Level sets

The **level set** $L_c(f)$ of a function $f : \mathbb{R}^d \to \mathcal{R}$ is defined as the set of points with the same function value c

$$L_c(f) = \{w \in \mathbb{R}^d : f(w) = c\}$$

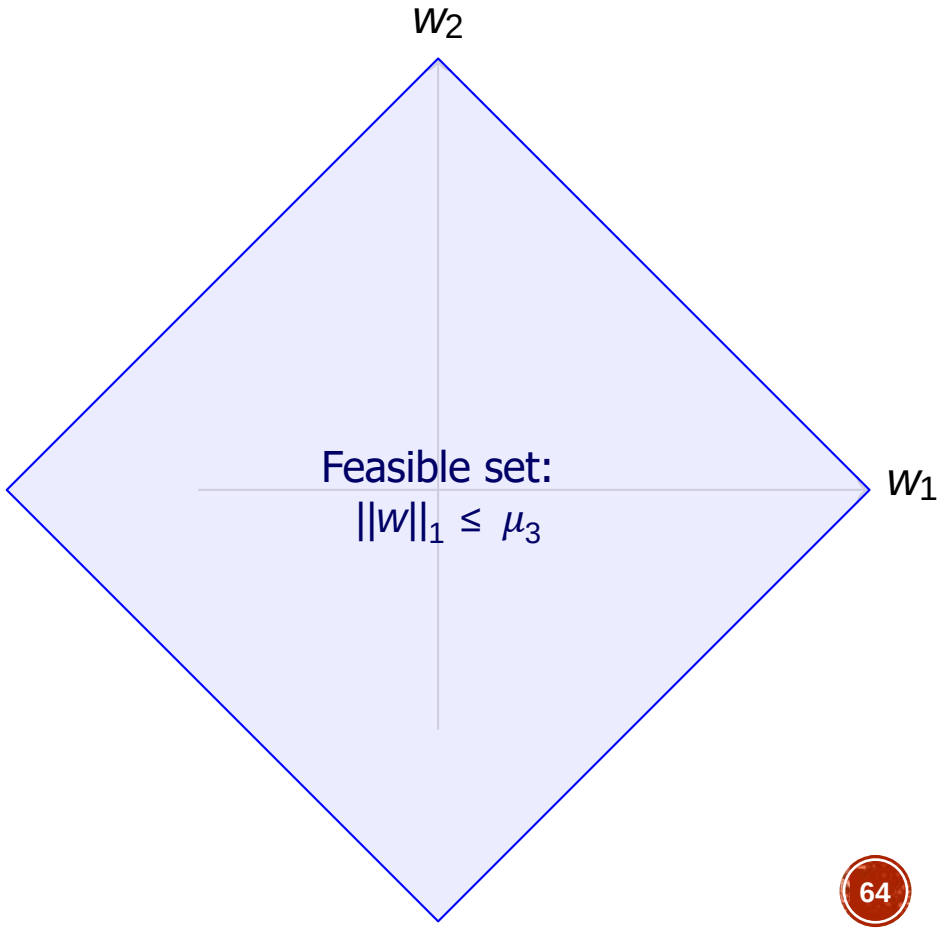
- Level set of a quadratic function (e.g. OLS) is an oval
- Center of the oval is $\hat{w}_{OLS} = \hat{w}_{\mu=\infty}$



LASSO give sparse solutions

The LASSO **feasible set** is a high dimensional diamond known as the L_1 ball

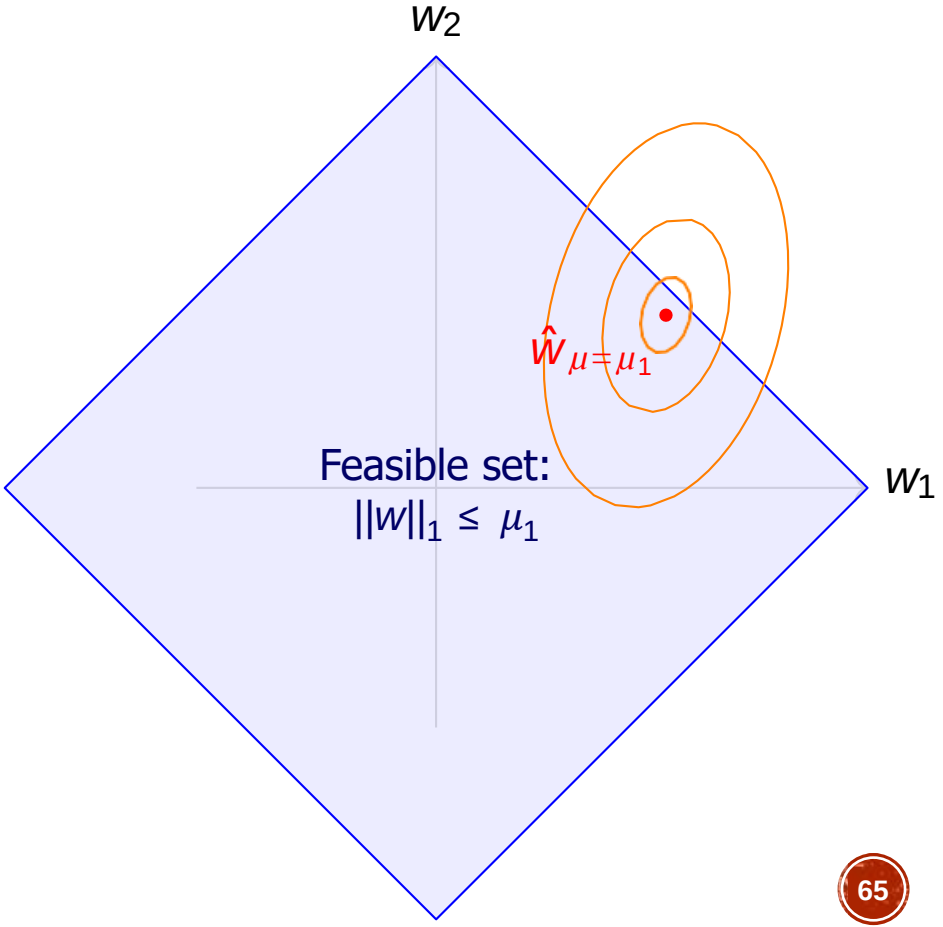
$$\{w \in \mathbb{R}^d : ||w||_1 \leq \mu\}$$



LASSO give sparse solutions

For large μ such that $||\hat{w}_{OLS}||_1 \leq \mu$

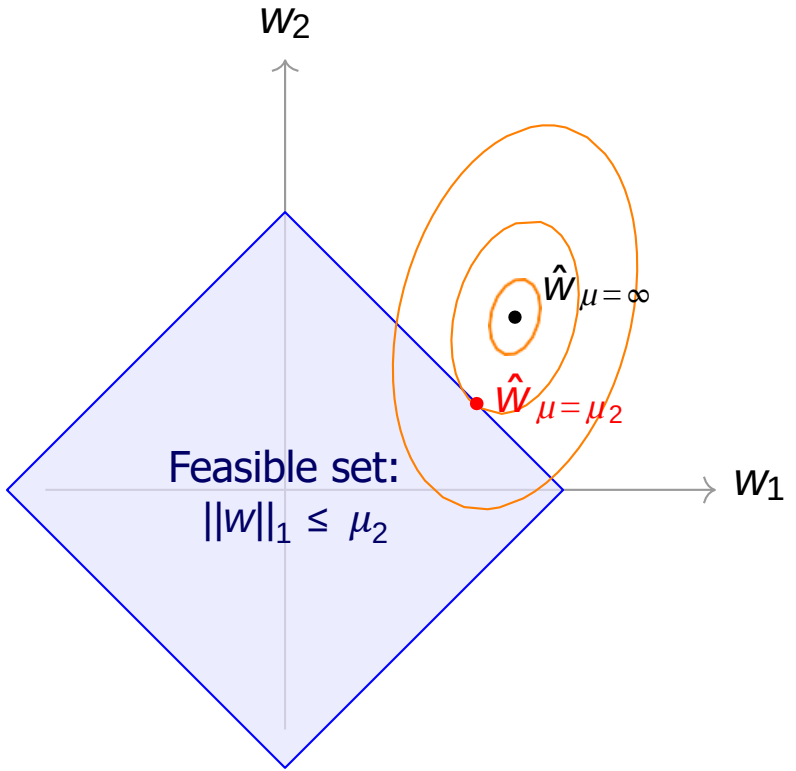
- Feasible set includes the un-regularized optimal solution
- Thus, optimal solution is the same as un-regularized solution!



LASSO give sparse solutions

As μ decreases (equivalent to increasing regularization λ)

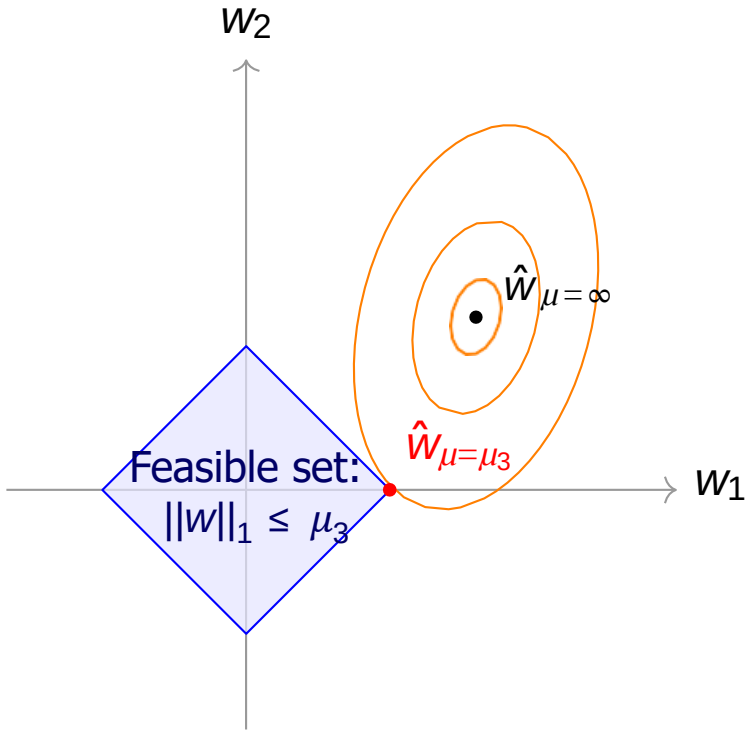
- Feasible set shrinks
- Solution $\hat{w}_{\mu=\mu_2}$ begins to change



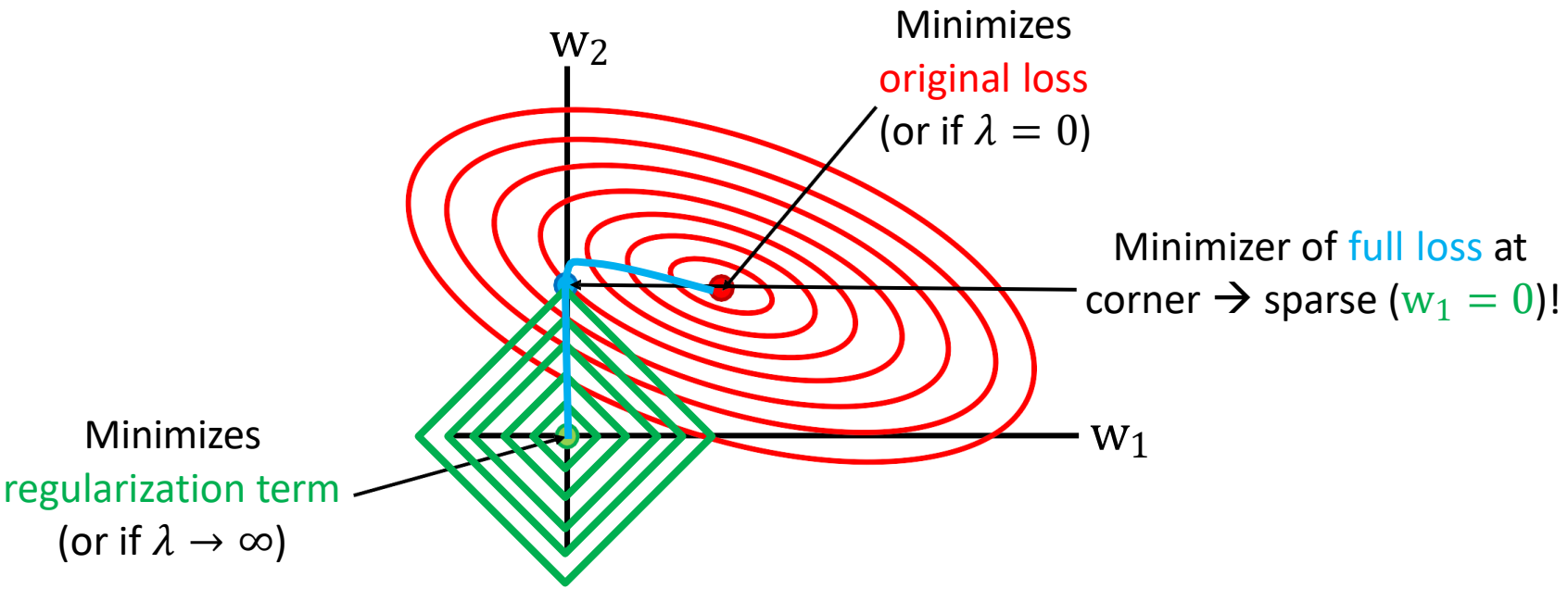
LASSO give sparse solutions

For small enough μ

- "Pointy" L_1-ball pushes less critical dimensions to zero
- Solution $\hat{w}_{\mu=\mu_3}$ becomes **sparse**



L_1 Regularization

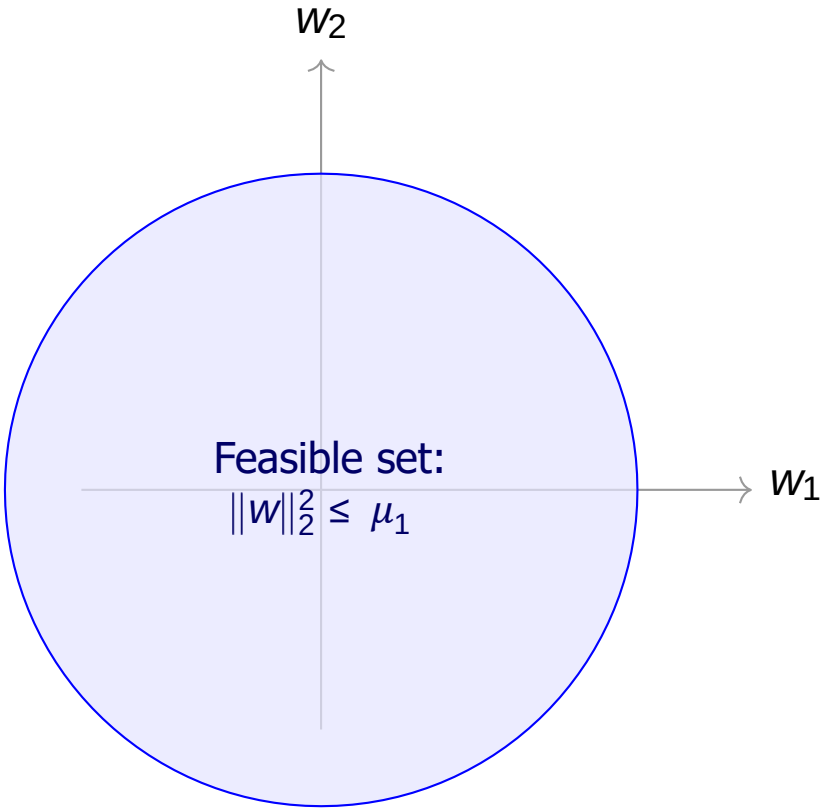


$$L(w\,;\,Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - wTxi)^2 + \lambda\sum_{j=1}^{d}|w_j|$$

Ridge regression does not give sparse solutions

The ridge regression feasible set is a high dimensional ball known as the **L_2 ball**

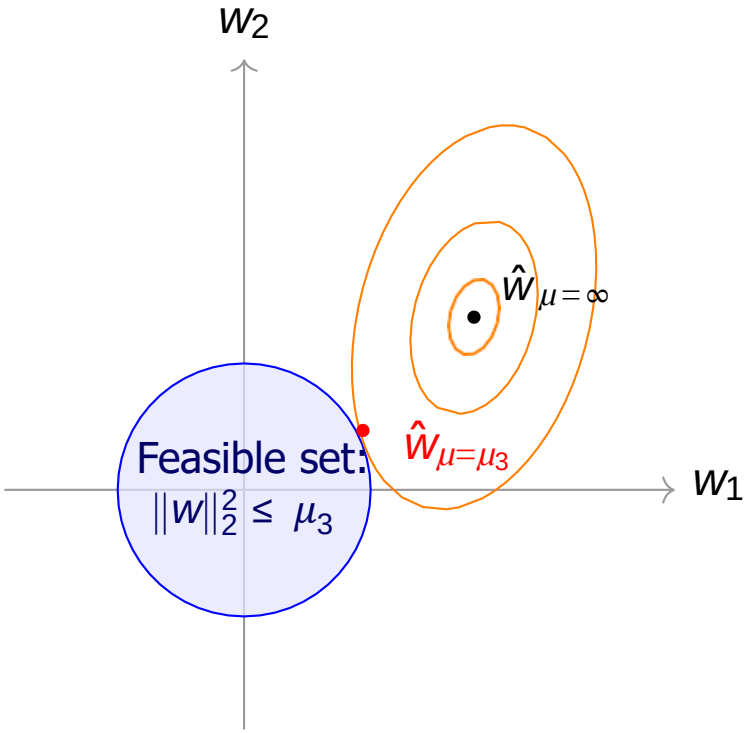
$$\{w \in \mathbb{R}^d : ||w||_2^2 \leq \mu\}$$



Ridge regression does not give sparse solutions

For small μ

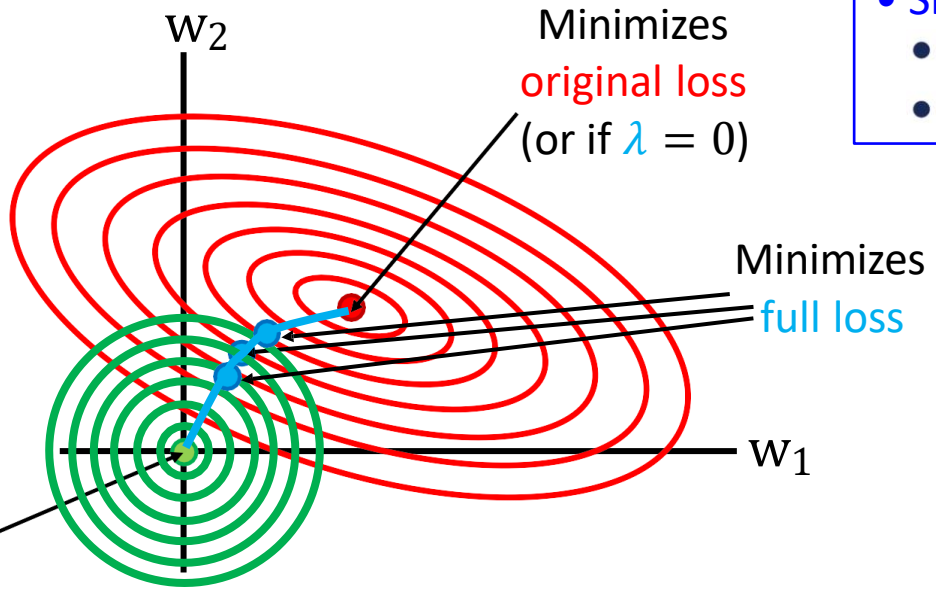
- "Smooth" L_2-ball does not encourage sparsity



L_2 Regularization

If $\lambda > 0$, always full rank and invertible!

- Shrinkage properties:
  - As $\lambda \to 0$, $\hat{w}_{\text{ridge}} \to \hat{w}_{\text{OLS}}$
  - As $\lambda \to \infty$, $\hat{w}_{\text{ridge}} \to \vec{0}$



Minimizes original loss (or if $\lambda = 0$)

Minimizes full loss

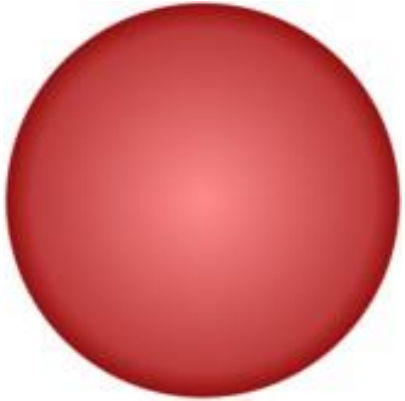
Minimizes regularization term (or if $\lambda \to \infty$)

- Tradeoff depends on choice of λ

$$L(w\,;\,Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - wTxi)^2 + \lambda\sum_{j=1}^{d} w_j{}^2$$

L1 and L2 ball in 3 dimensions

L_1 ball in 3 dimensions



L_2 ball in 3 dimensions

L_p Norm Generalizes L_1 and L_2

The L_p norm is defined as

$$||w||_p \triangleq \left(\sum_i |w_i|^p \right)^{1/p}$$

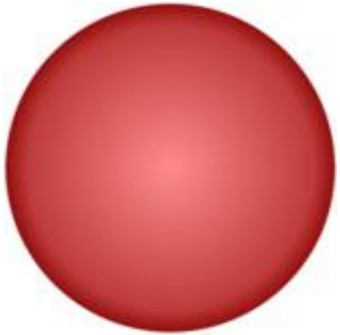
This generalizes the L_1 and L_2 norm

- $p = 1 \quad \rightarrow \quad L_1$ norm
- $p = 2 \quad \rightarrow \quad L_2$ norm

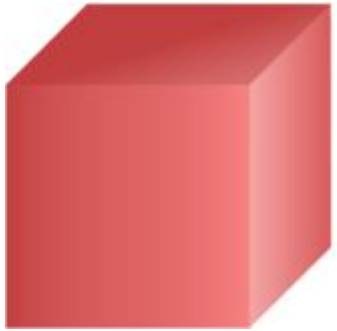
L_p Norm Generalizes L_1 and L_2

$p = 0.5$

$p = 1$

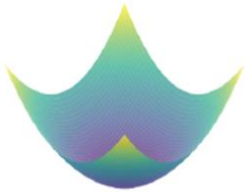


$p = 2$



$p = \infty$

Ridge Regression vs LASSO

	Ridge regression	LASSO
Objective	$\min_w \|\|Y - Xw\|\|_2^2 + \lambda\|\|w\|\|_2^2$	$\min_w \|\|Y - Xw\|\|_2^2 + \lambda\|\|w\|\|_1$
Regularizer shape		
Solution	Dense	Sparse
Prior $P(w)$	Normal	Laplace
Constraint set	L2-ball	L1-ball

Ridge Regression vs LASSO

- **Ridge**

$$\text{minimize}_w \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \lambda \|w\|_2^2$$

  - Very fast:
    - Closed form solution if used with linear models
    - Even with other loss functions, optimization is fast for squared ℓ_2 regularization, because $\|w\|_2^2$ is **convex and smooth**

- **Lasso**

$$\text{minimize}_w \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \lambda \|w\|_1$$

  - Slower than Ridge:
    - Requires iterative optimization algorithm like sub-gradient descent
    - In particular, it is slower because $\|w\|_1$ is **convex but non-smooth**

Acknowledgement

Most of the slides are based on the ML course of:

- Matt Barnes and Matt Golub, University of Washington
- Mingmin Zhao and Jiatao Gu, University of Pennsylvania

THANK YOU