Lecture 3: Linear Regression (Part 2)

CIS 4190/5190

Spring 2023

Administrivia

- Class currently full at 225 students (211 enrolled, 16 permits out) with about 180 students in the waitlist, many from 3 months ago. Limited movement expected. Unused permits will be revoked as announced earlier, unless you have informed us.
  - If you're on the waitlist, submitting HW1 early will increase priority, but no guarantees.
- HW1 due **Wednesday 8 p.m.**, and HW2 will be posted that evening, on linear regression.
- In most cases, you should use EdSTEM to contact the course team, where you are much more likely to receive a fast response. But if your message must be kept private from TAs:
  - **Always** email both instructors together.
  - Start subject line with **"[CIS 4190 / 5190 Spring 2023]"**.
- Canvas link (for recordings) is on the class webpage > files: https://canvas.upenn.edu/courses/1704503 (video recordings posted on day of lecture/next day)
- If you're on the waitlist and submitting HW1, submit directly to gradescope.
- If switching from CIS 4190 to CIS 5190, contact cis-undergrad-advising@seas.upenn.edu
  - More work, and possibly higher grade cutoffs for CIS 5190.
- TA office hours start today. No help for HW1, but you can ask for help with the course material.
- Recitations on Thursday at 5 p.m. on Python, Numpy, Pandas, Scikit-Learn. See EdSTEM post.

Recap: Loss Minimization View of ML

- **To design an ML algorithm:**
  - Choose model family $F = \{f_\beta\}_\beta$ (e.g., linear functions)
  - Choose loss function $L(\beta; Z)$ (e.g., MSE loss)

- **Resulting algorithm:**

$$\hat{\beta}(Z) = \arg\min_{\beta} L(\beta; Z)$$

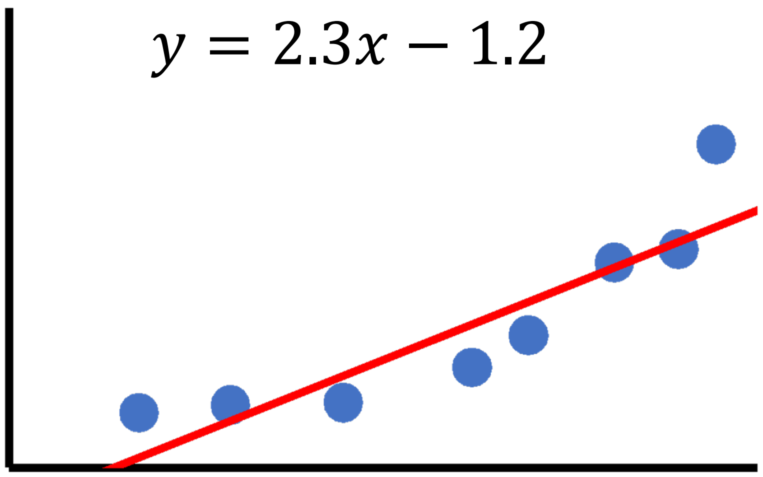
Recap: Linear Regression

- **Input:** Dataset $Z = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
- Compute

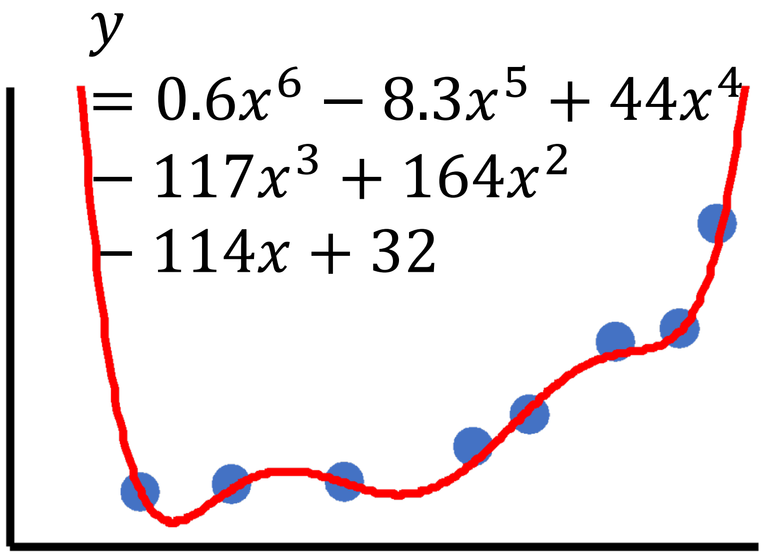
$$\hat{\beta}(Z) = \underset{\beta \in \mathbb{R}^d}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta^\top x_i)^2$$

- **Output:** $f_{\hat{\beta}(Z)}(x) = \hat{\beta}(Z)^\top x$
- Discuss algorithm for computing the minimal β later (next class)

Recall: Overfitting



Underfitting



Overfitting

Overfitting occurs when:

- The learned hypothesis fits the training set very well e.g. $\mathcal{L}(\boldsymbol{\theta}) \approx 0$, but fails to generalize to new examples

Today's Class

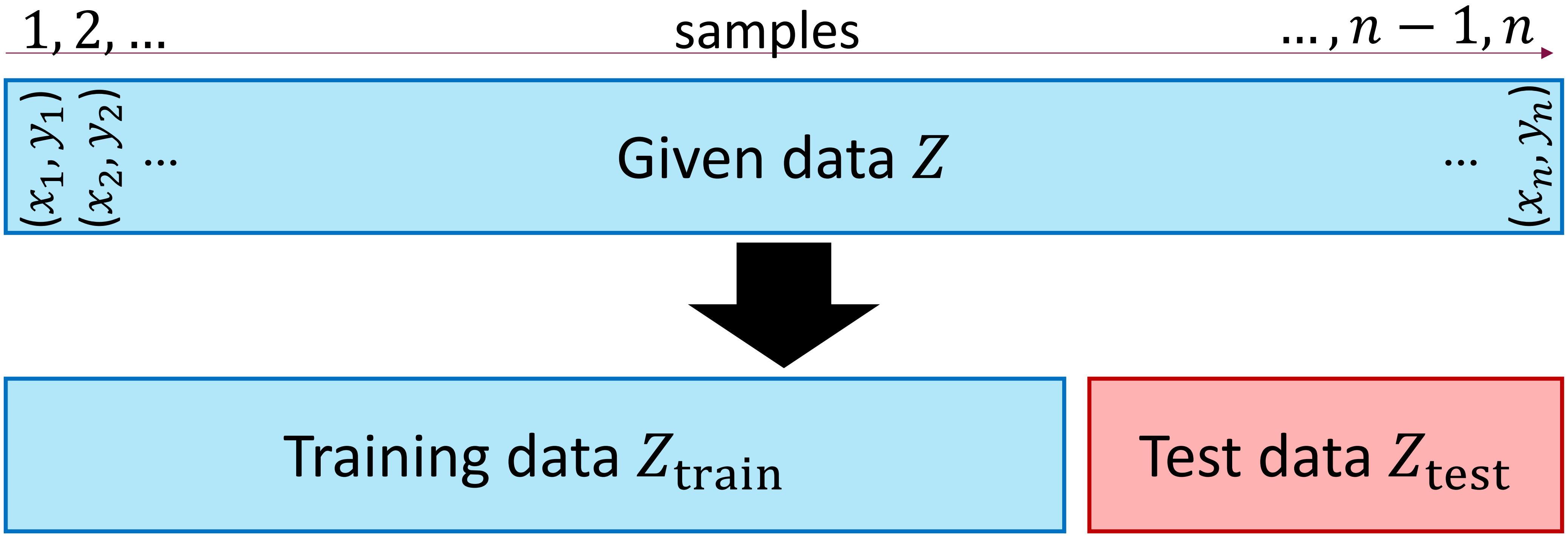
- Understanding, diagnosing, and combating overfitting:
  - Bias and Variance of hypothesis classes
  - Regularized linear regression
  - Cross-Validation

- Feature Selection and Preprocessing
  - Sparse linear regression

Assessing Overfitting

Training/Test Split

- **Issue:** How to detect overfitting vs. underfitting?
- **Solution:** Use **held-out test data** to estimate loss on new data
  - Typically, randomly shuffle data first



Training/Test Split Protocol for Machine Learning

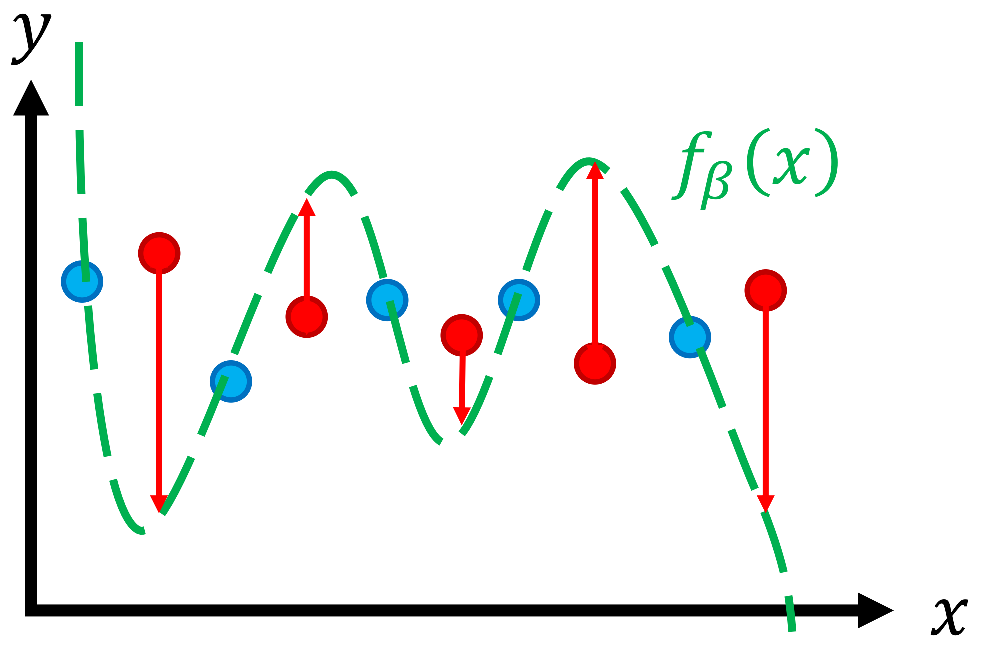
- **Step 1:** Split Z into Z_{train} and Z_{test}

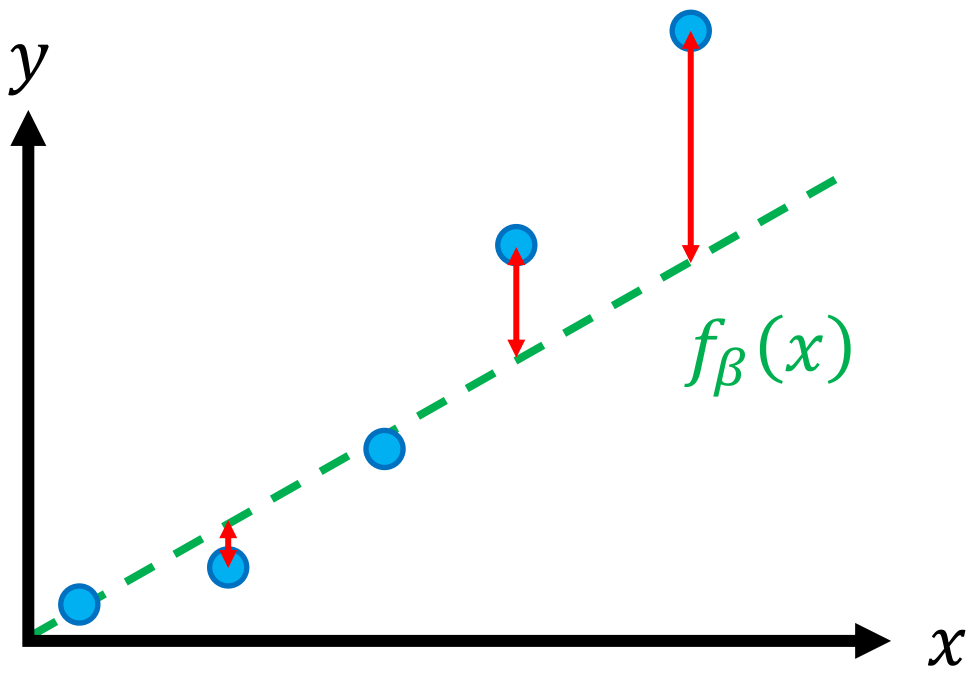
Training data Z_{train} | Test data Z_{test}

- **Step 2:** Run linear regression with Z_{train} to obtain $\hat{\beta}(Z_{\text{train}})$
- **Step 3:** Evaluate
  - **Training loss:** $L_{\text{train}} = L\big(\hat{\beta}(Z_{\text{train}}); Z_{\text{train}}\big)$
  - **Test (or generalization) loss:** $L_{\text{test}} = L\big(\hat{\beta}(Z_{\text{train}}); Z_{\text{test}}\big)$, (plus other performance metrics besides the loss function)

Training/Test Split Protocol for Machine Learning

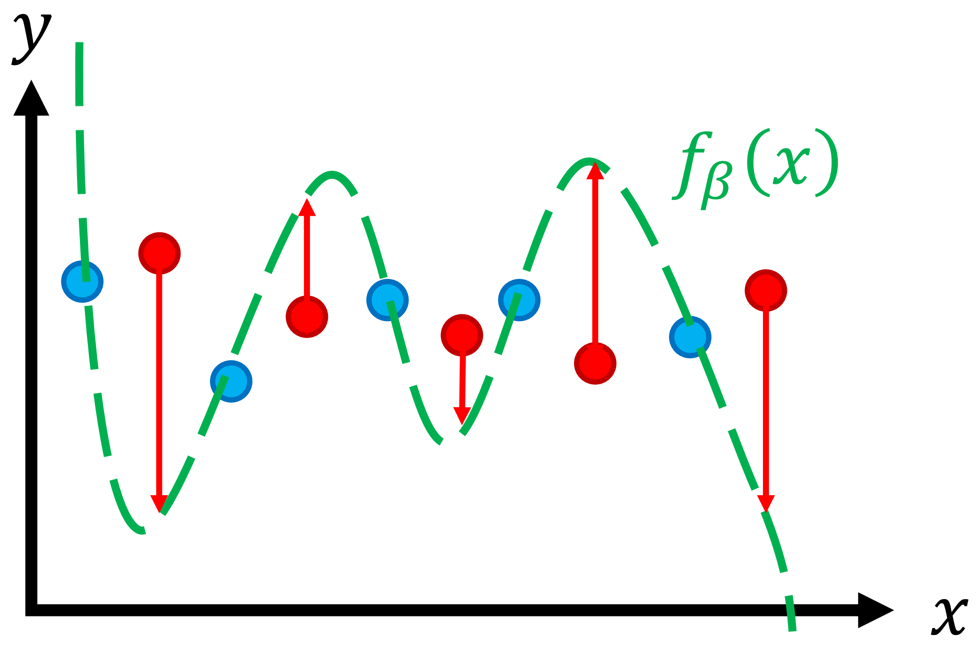
- **Overfitting**
  - Fit the **training data** Z well
  - Fit new **test data** (x, y) poorly
- **Underfitting**
  - Fit the **training data** Z poorly
  - (Necessarily fit new **test data** (x, y) poorly)



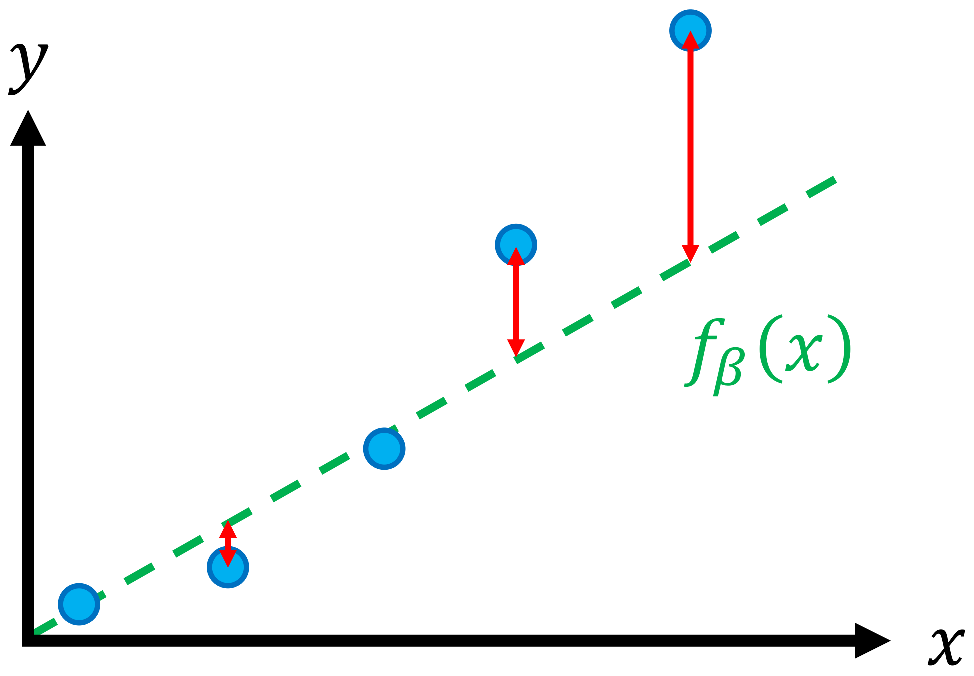


Training/Test Split Protocol for Machine Learning

- **Overfitting**
  - L_{train} is small
  - L_{test} is large

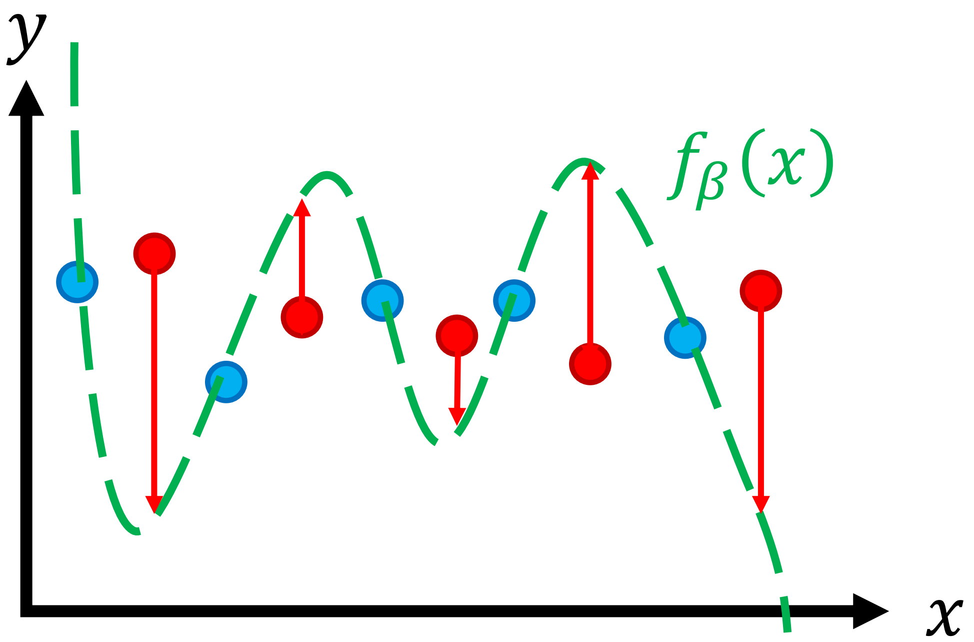


- **Underfitting**
  - Fit the **training data** Z poorly
  - (Necessarily fit new **test data** (x, y) poorly)

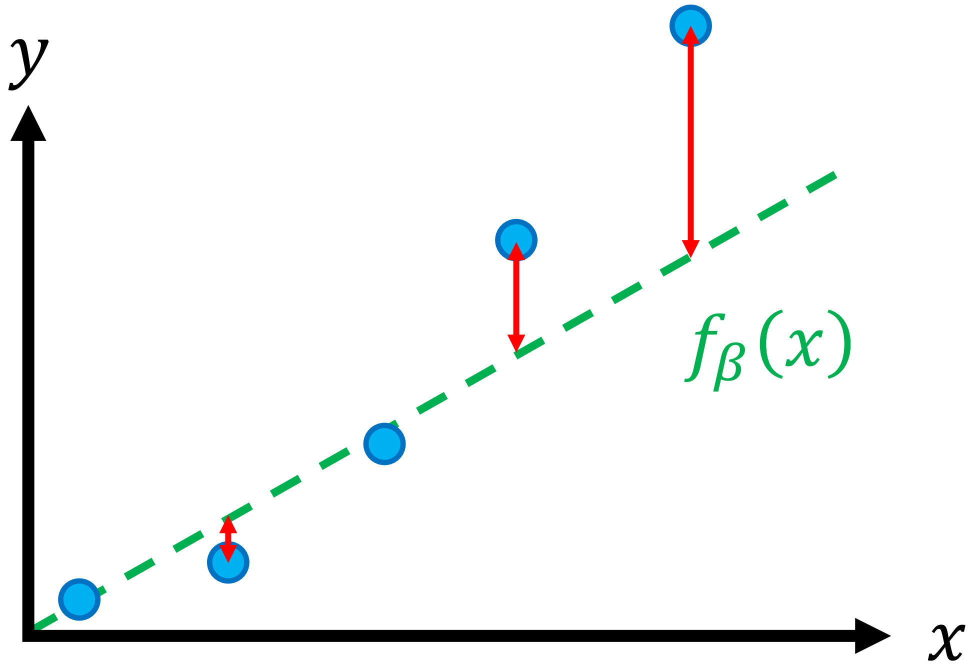


Training/Test Split Protocol for Machine Learning

- **Overfitting**
  - L_{train} is small
  - L_{test} is large



- **Underfitting**
  - L_{train} is large
  - L_{test} is large



“Independent and Identically Distributed”

- **The “IID” assumption**
  - “Test” data Z_{test} are drawn IID from same data distribution $P(x, y)$ as Z_{train}
  - IID = independent and identically distributed
  - This is a strong (but common) assumption!

- **Time series data**
  - Particularly important failure case since data distribution may shift over time
  - **Solution:** Split along time (e.g., data before 9/1/20 vs. data after 9/1/20)

Underfitting & Overfitting

↕️

Bias & Variance

Recall

Reminder: we don't yet know how to find the argmin. For the moment, we will continue to assume it will be found.

Data $Z = \{(x_i, y_i)\}_{i=1}^{n}$

$\hat{\beta}(Z) = \arg\min_\beta L(\beta; Z)$

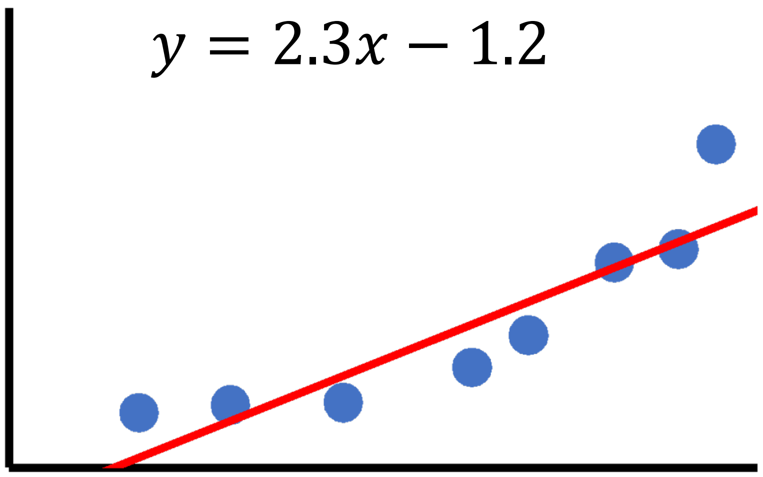
L encodes $y_i \approx f_\beta(x_i)$

Model $f_{\hat{\beta}(Z)}$

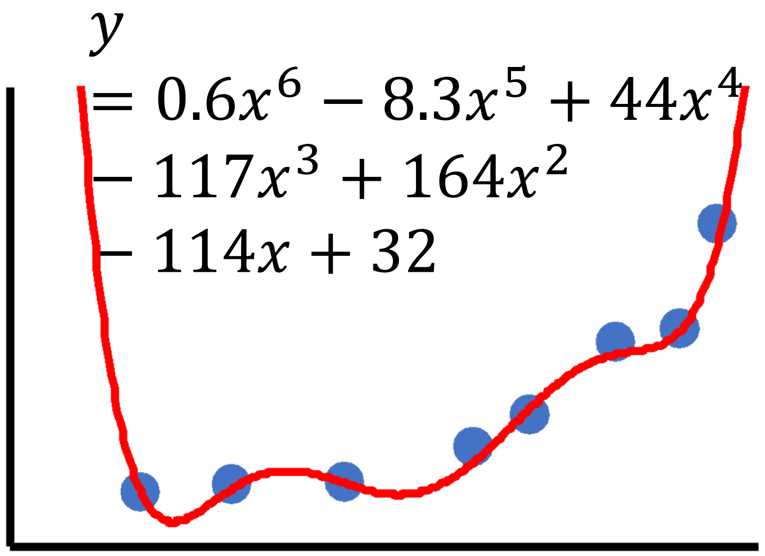
We are still thinking about two main design choices that influence this box:

Model class (linear regression, feature map etc.), and loss function (MSE)

Recall: Underfitting and Overfitting



Underfitting



Overfitting

We will understand these phenomena now through two properties of a model family, “bias”, and “variance”.

Language for thinking about the ways in which model families can be bad.

How to Fix Underfitting/Overfitting?

Three main options:

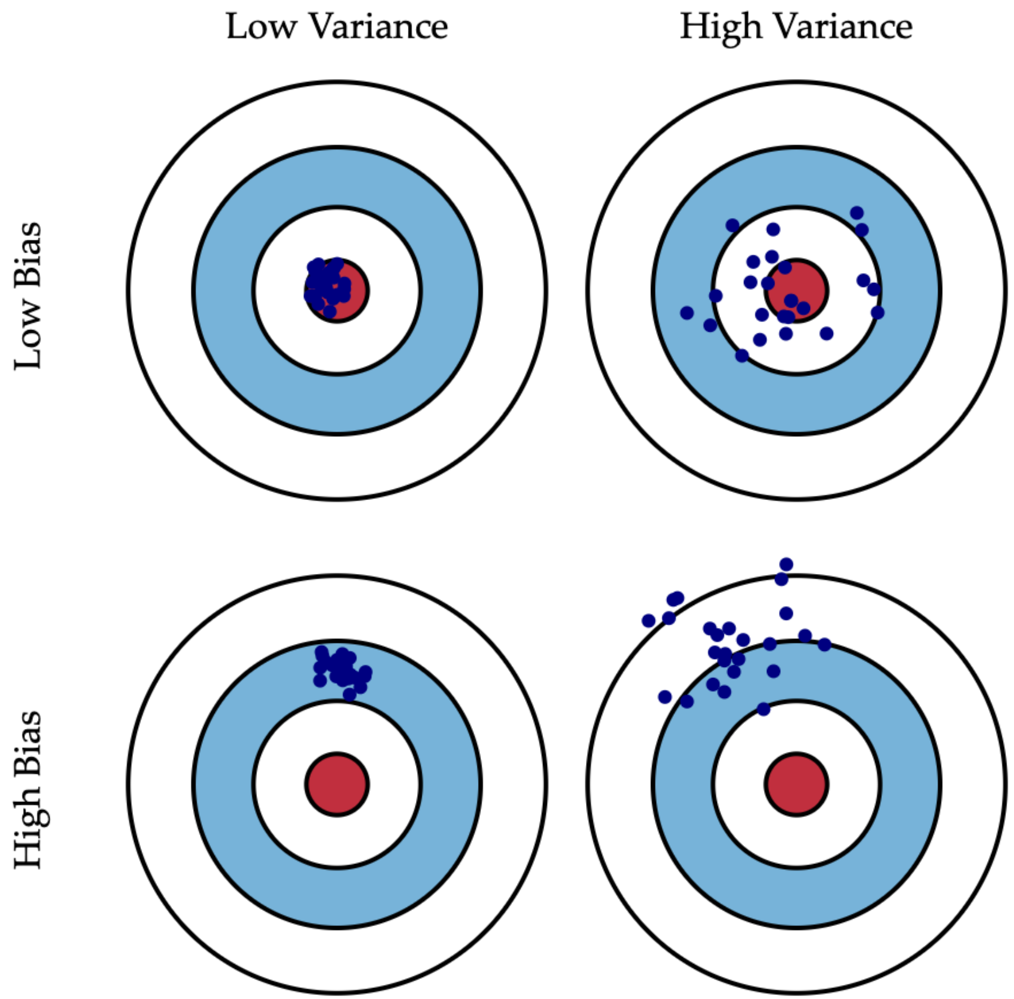
- Improve the training dataset
- Choose the right model family
- Choose the right loss function

We will explore these in some detail over the next few slides.

Definitions: “Bias” and “Variance”

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

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

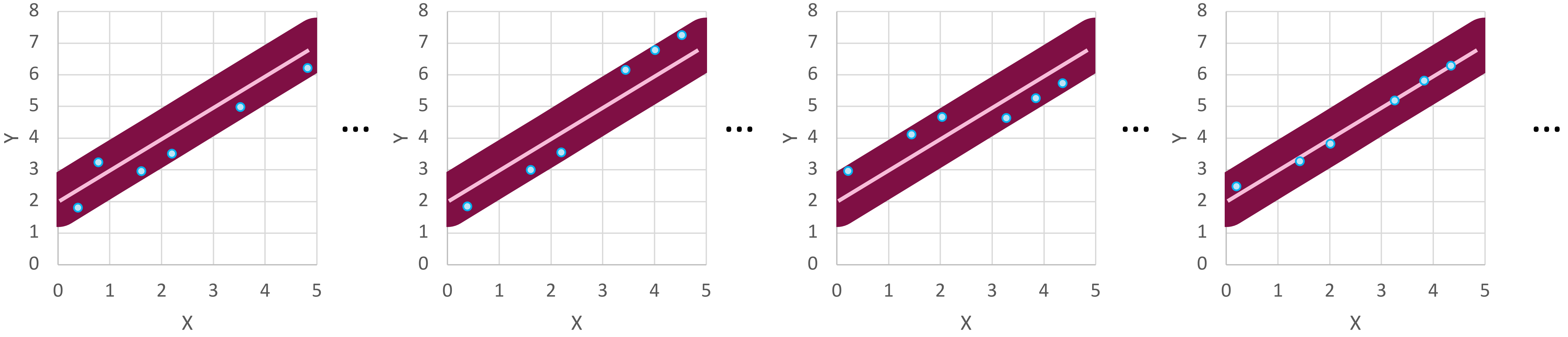


Scott Fortmann-Roe
http://scott.fortmann-roe.com/docs/BiasVariance.html

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].

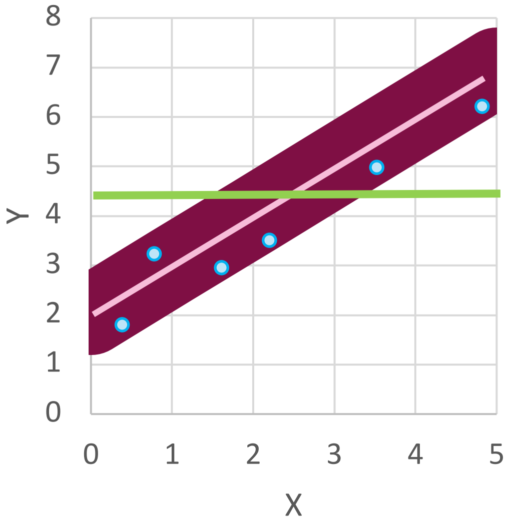
Let us draw $k \to \infty$ training sets of $n = 6$ samples each, drawn from $P(X, Y)$.

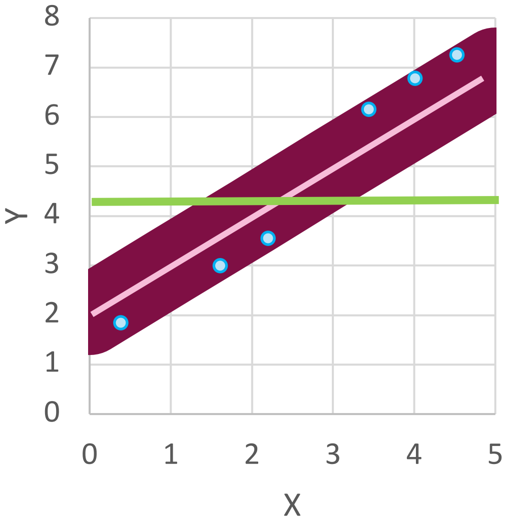


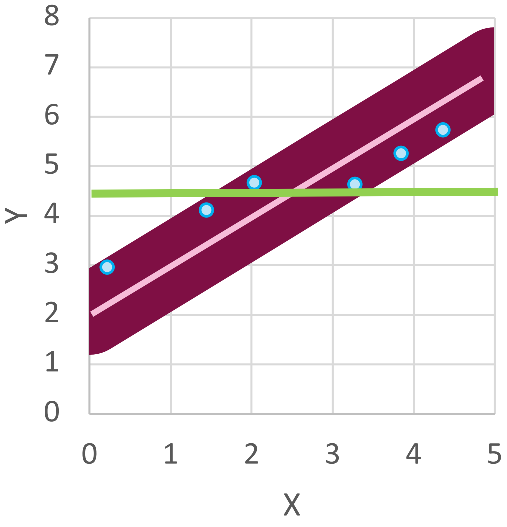
Different *Constant* Fits

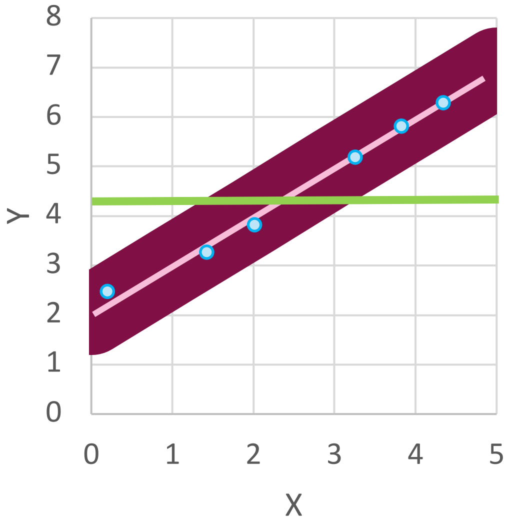
What if the hypothesis class was the constant function class

$$f_{\boldsymbol{\theta}}(x) = \theta_0$$









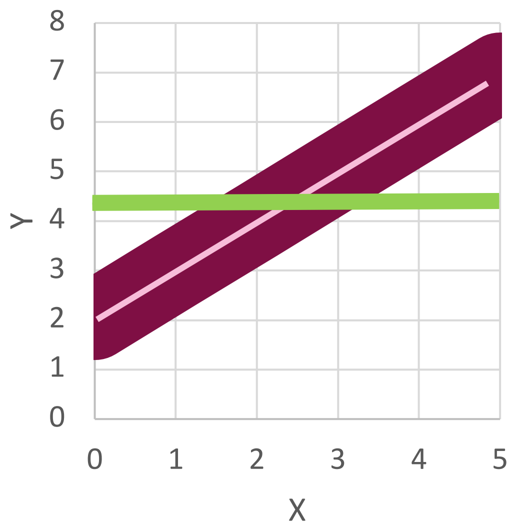
Different *Constant* Fits

What if the hypothesis class was the constant function class

$$f_{\boldsymbol{\theta}}(x) = \theta_0$$

Almost identical fits

"low variance"



Average fit far from the true function

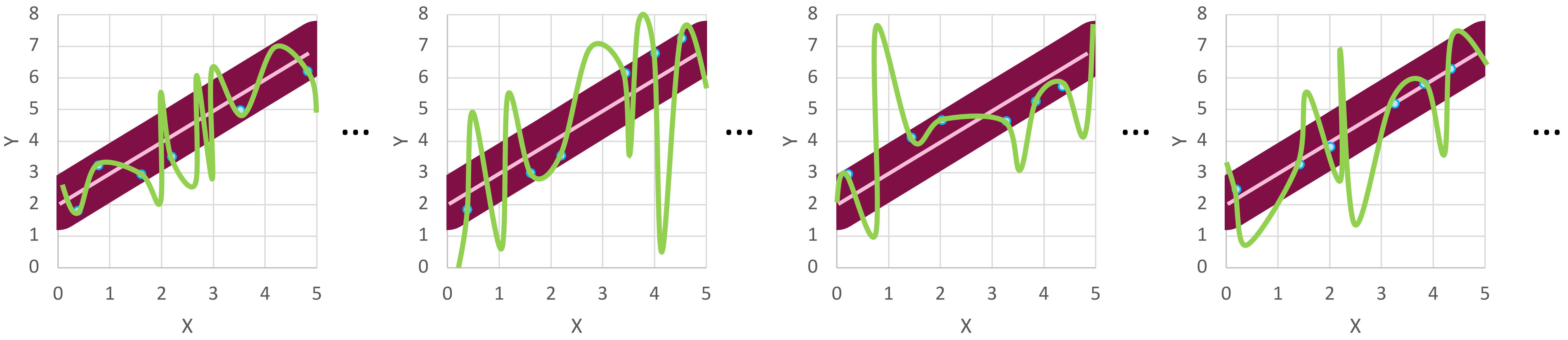
"high bias"

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

Different 10th Degree Curve Fits

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

$$f_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \cdots \theta_{10} x^{10}$$



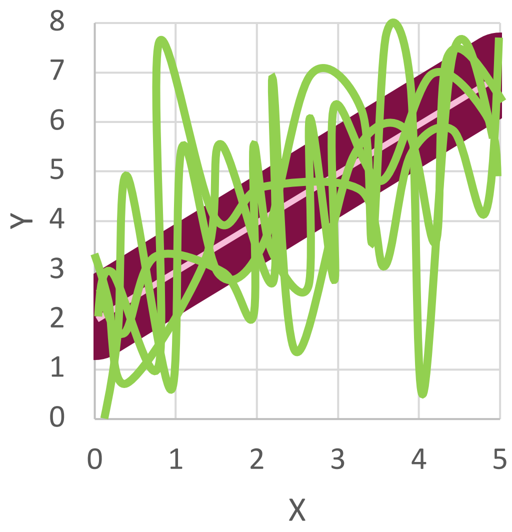
Different *10^th Degree* Fits

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

$$f_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \cdots \theta_{10} x^{10}$$

Very different fits

"high variance"



Average fit close to the true function

"low bias"

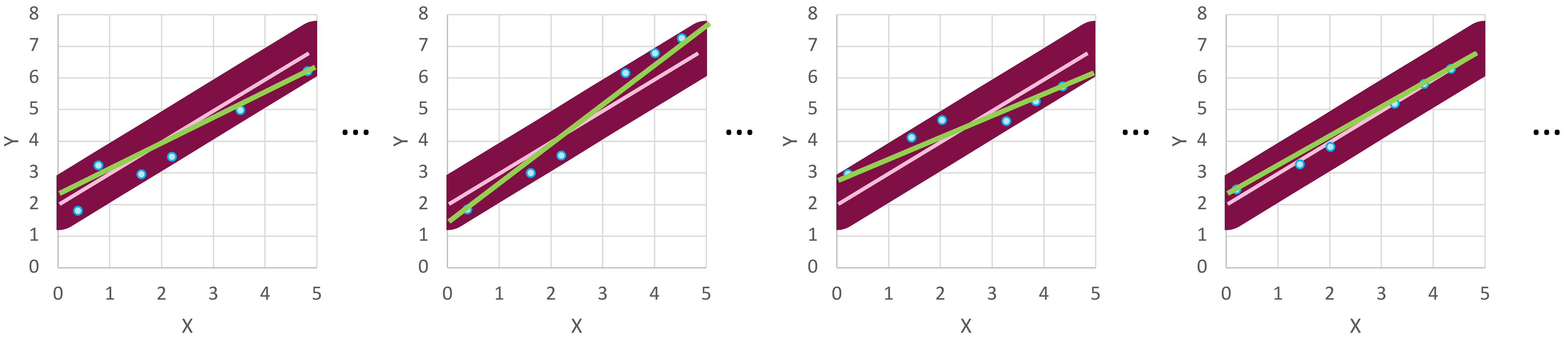
Theoretical result: Generalization MSE $\approx$ ``Bias'' + ``Variance''

Different *Linear* Fits

Say, our hypothesis class is a line:

$$f_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x_1$$

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



Slightly different fits

Different *Linear* Fits

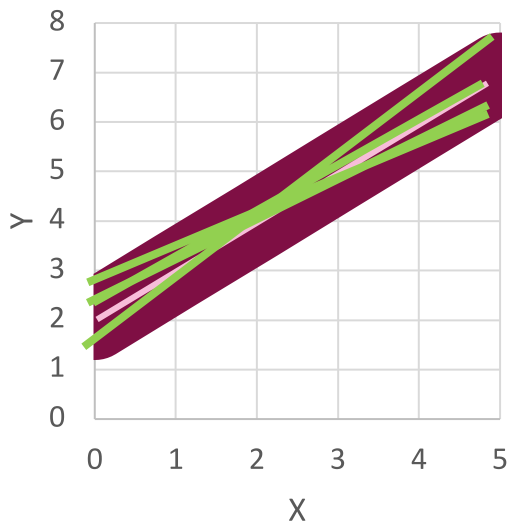
Say, our hypothesis class is a line:

$$f_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x_1$$

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

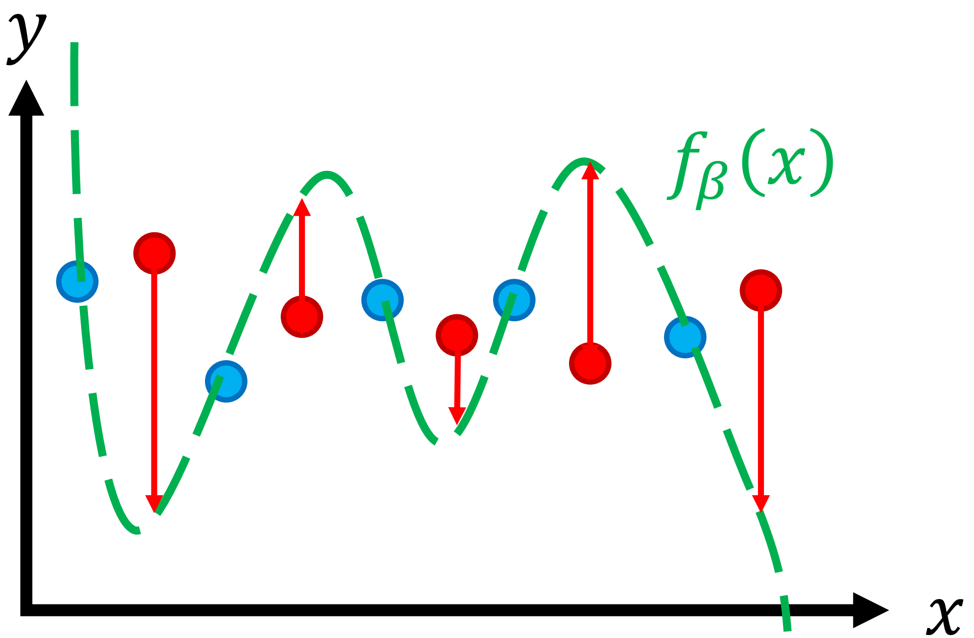
"low bias"

This is the type of model complexity you want to have

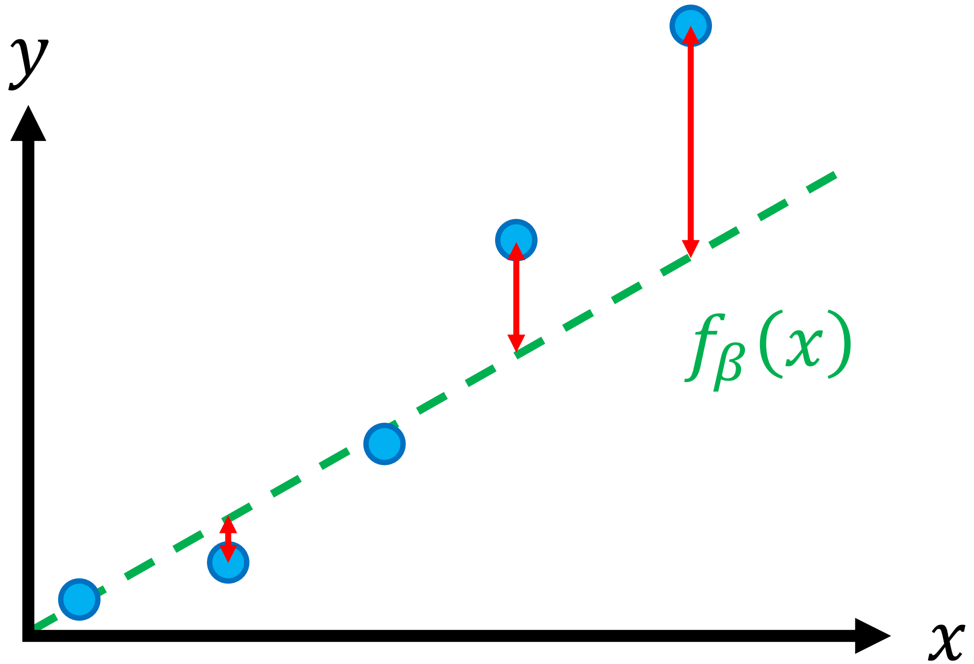
Theoretical result: Generalization MSE $\approx$ ``Bias'' + ``Variance''

Bias-Variance Tradeoff

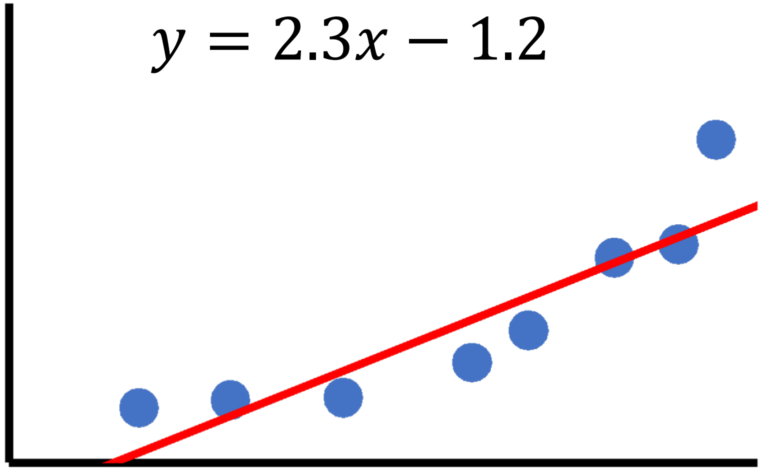
- **Overfitting (high variance)**
  - High capacity model capable of fitting complex data
  - Insufficient data to constrain it



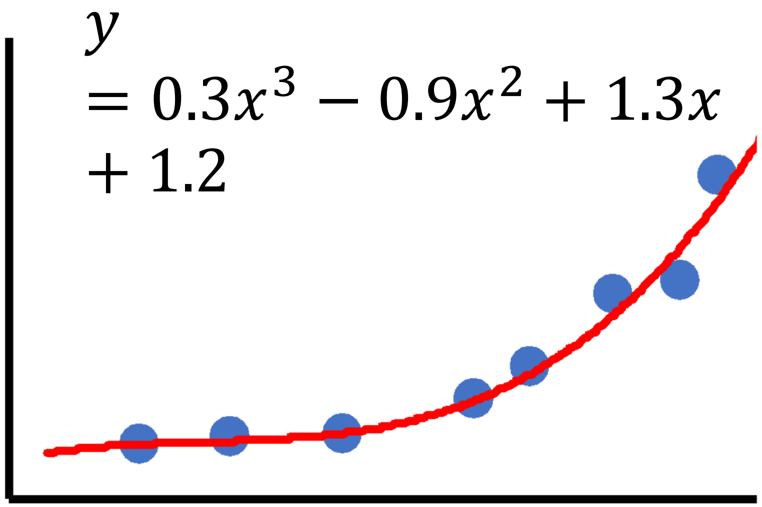
- **Underfitting (high bias)**
  - Low capacity model that can only fit simple data
  - Sufficient data but poor fit



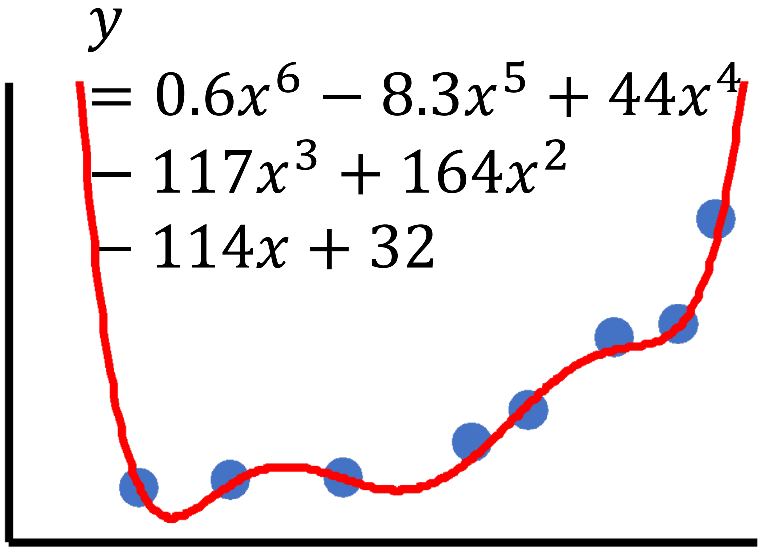
Underfitting & Overfitting, Bias & Variance



Underfitting



Correct fit



Overfitting

This hypothesis space has high bias

The data decides which hypothesis class choices are good/bad.

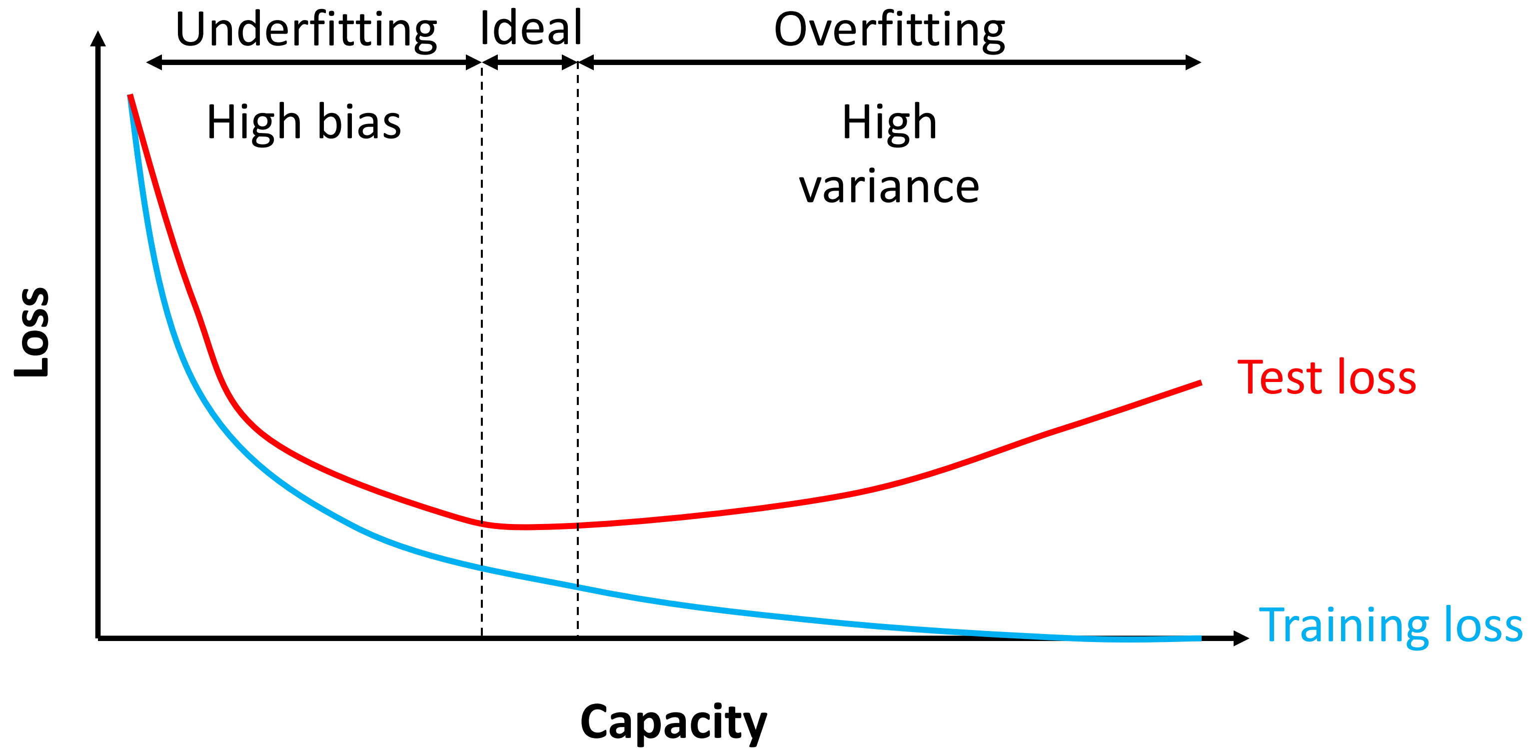
(Train/Test Split Protocol)

This hypothesis space has high variance

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

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 $\boldsymbol{\phi}$ 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. We will see some common rules of thumb for feature engineering soon.

The Effect of Dataset Size

Recall, we said:

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

Q: What if we had drawn larger training sets? Would it impact:

- Bias?

  Usually no. As $k \to \infty$, the average of fits $\{f_\beta\}_{1:k}$ to k training sets of finite size n (for any n) approaches the fit to an $n' \to \infty$-sized set.

  Often convenient to think of bias as the lowest achievable error corresponding to the "best model" within the hypothesis space.

- Variance?

  Yes, as the dataset size grows, different i.i.d. datasets would induce similar fits $\Rightarrow$ lower variance.

  Convenient to think of variance as average error w.r.t. the "best model".

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.

Regularization

How to Fix Underfitting/Overfitting?

Recall, three main options:

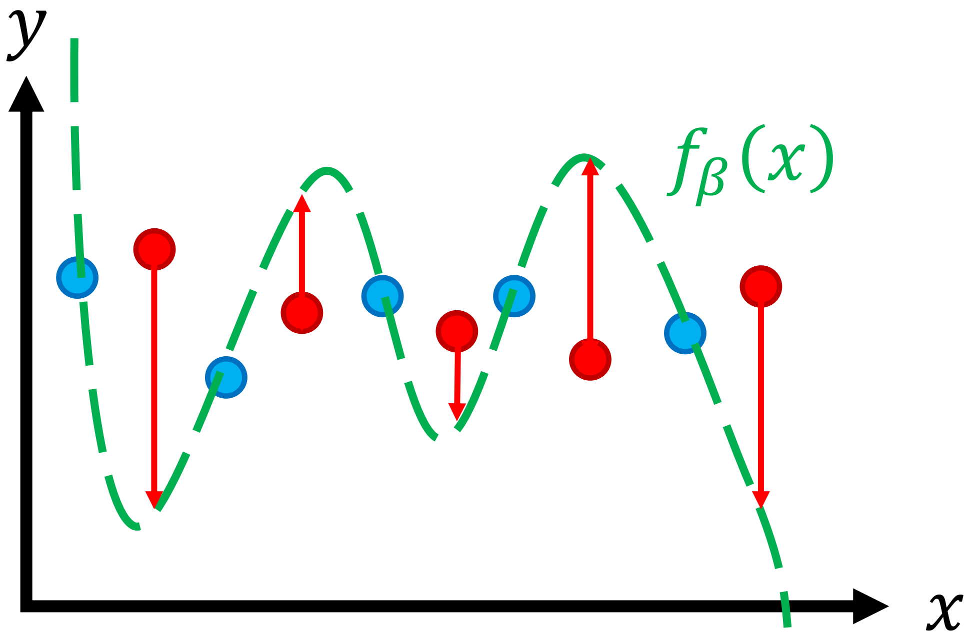
- Improve the training dataset (collect more data)
- Choose the right model family (not too complex, not too simple)
- Choose the right loss function

We will explore this third option now.

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"

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



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

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

How to measure this?

Recall: Mean Squared Error Loss

- Mean squared error loss for linear regression:

$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2$$

Linear Regression with L_2 Regularization

- **Original loss** + **regularization**:

One measure of fit complexity

$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda \cdot \|\beta\|_2^2$$
$$= \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda \sum_{j=1}^{d} \beta_j^2$$

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

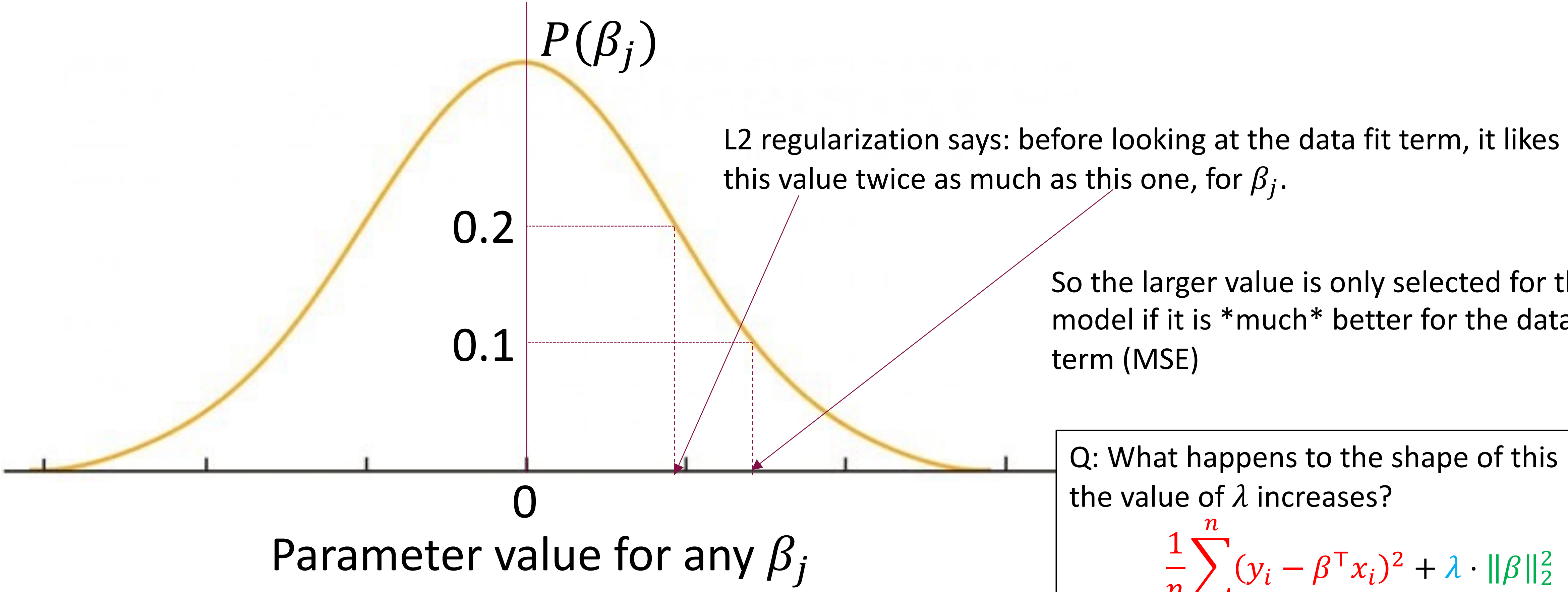
Intuition on $\boldsymbol{L_2}$ Regularization

- A thought experiment.
  - Consider a feature map with $d = 10$ features $[x_1, \ldots, x_d]$
  - Suppose during linear regression that we forced $\beta_j = 0$ for all $j > 5$
  - This is exactly equal to choosing a smaller-capacity hypothesis class, induced by the smaller feature map with $d = 5$

- Thus, forcing β_j's to be 0 induces a smaller-capacity hypothesis class.
- The soft version of this: encouraging β_j's to have small magnitude also induces a smaller-capacity hypothesis class.
  - This is what $\boldsymbol{L_2}$ regularization does: $\sum_{j=1}^{d} \beta_j^2 = \|\beta\|_2^2 = \|\beta - 0\|_2^2$
  - Pulls coefficients towards 0
  - As $\lambda \to \infty$, it forces $\beta = 0$

Intuition on L_2 Regularization: Gaussian Priors

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



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

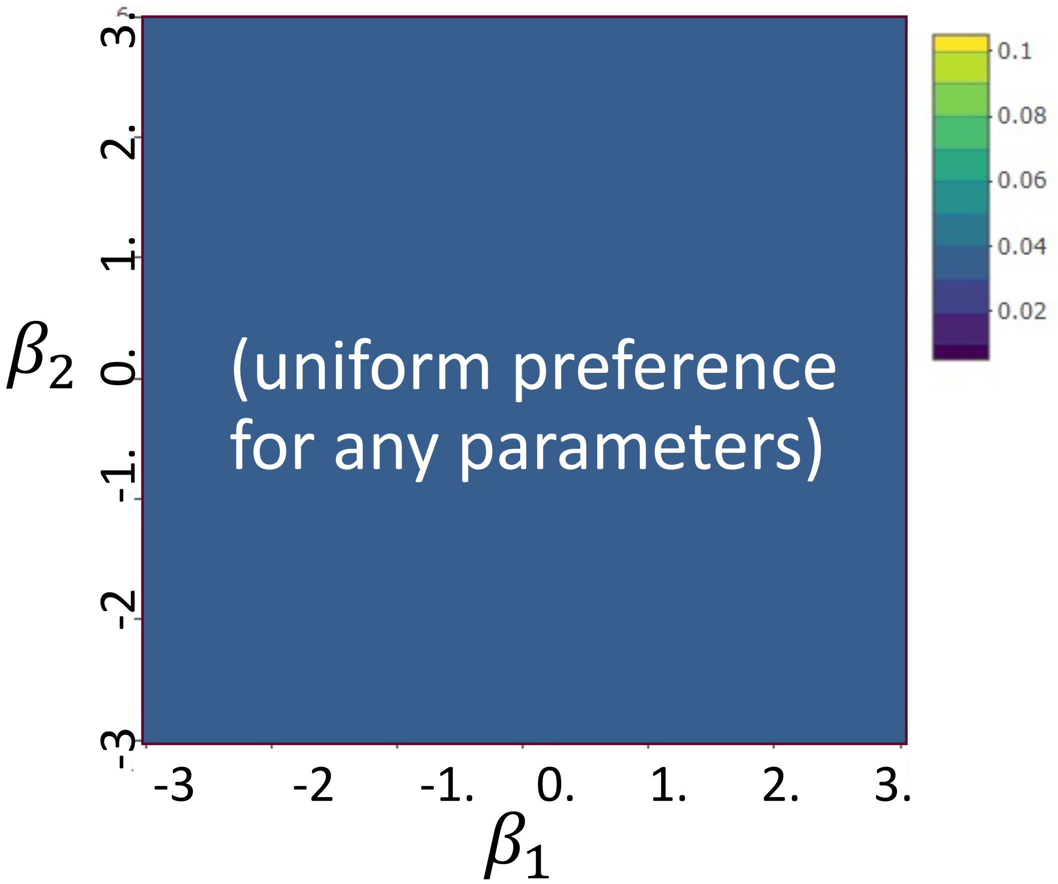
Q: What happens to the shape of this plot if the value of λ increases?

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda \cdot \|\beta\|_2^2$$

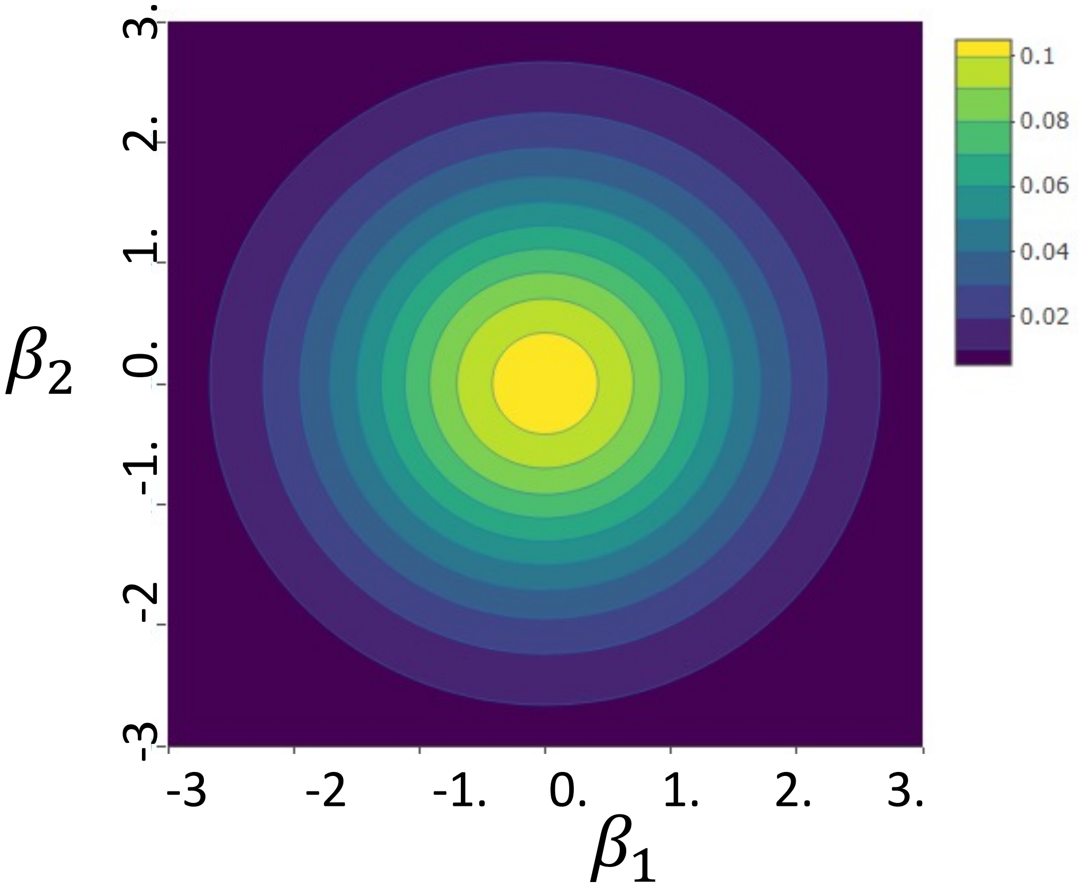
Intuition on L_2 Regularization: Gaussian Priors

- Extending the gaussian pdf over each individual weight β_j to the full parameter vector β, the hypothesis space is constrained.

Before regularization



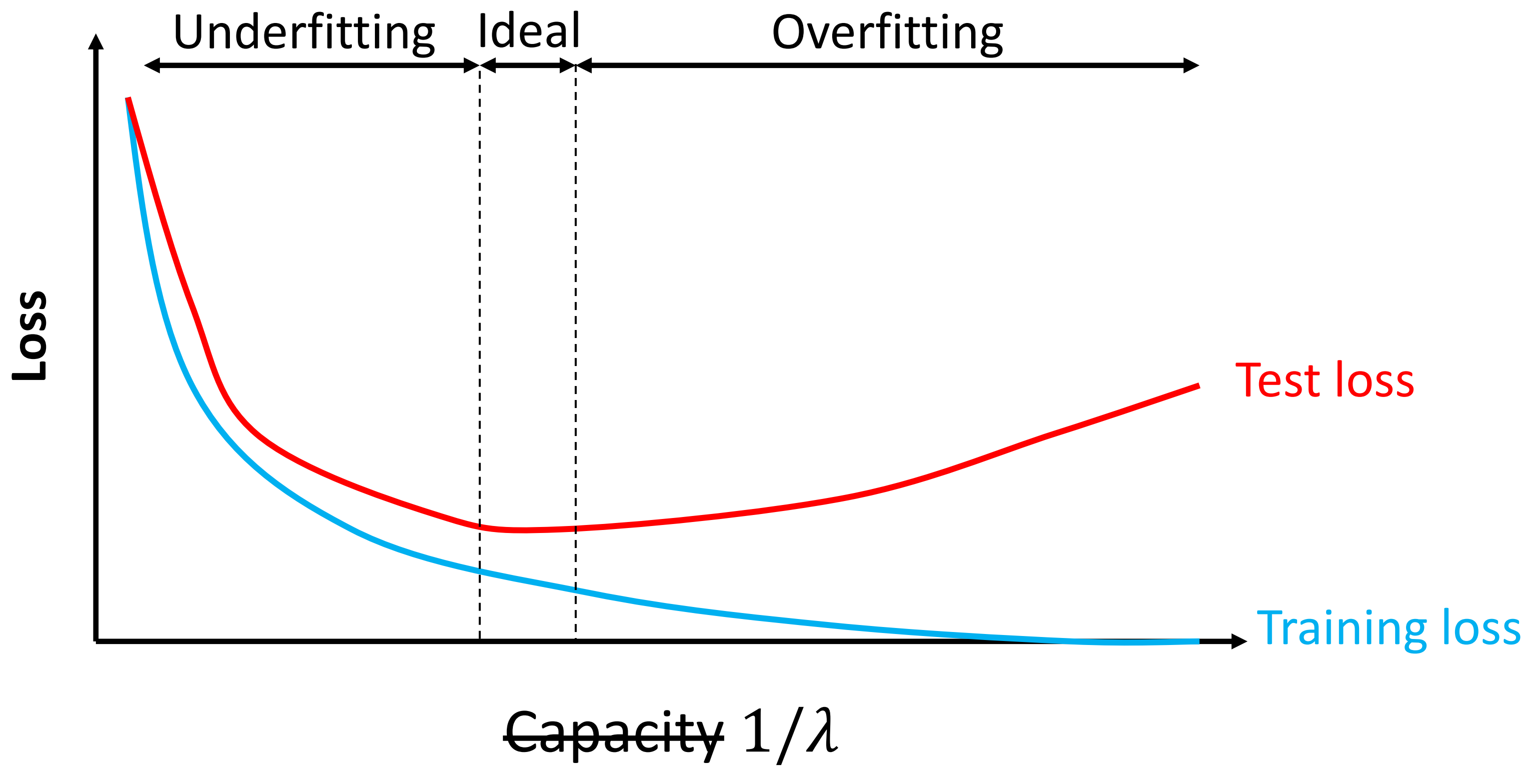
With L2 regularization



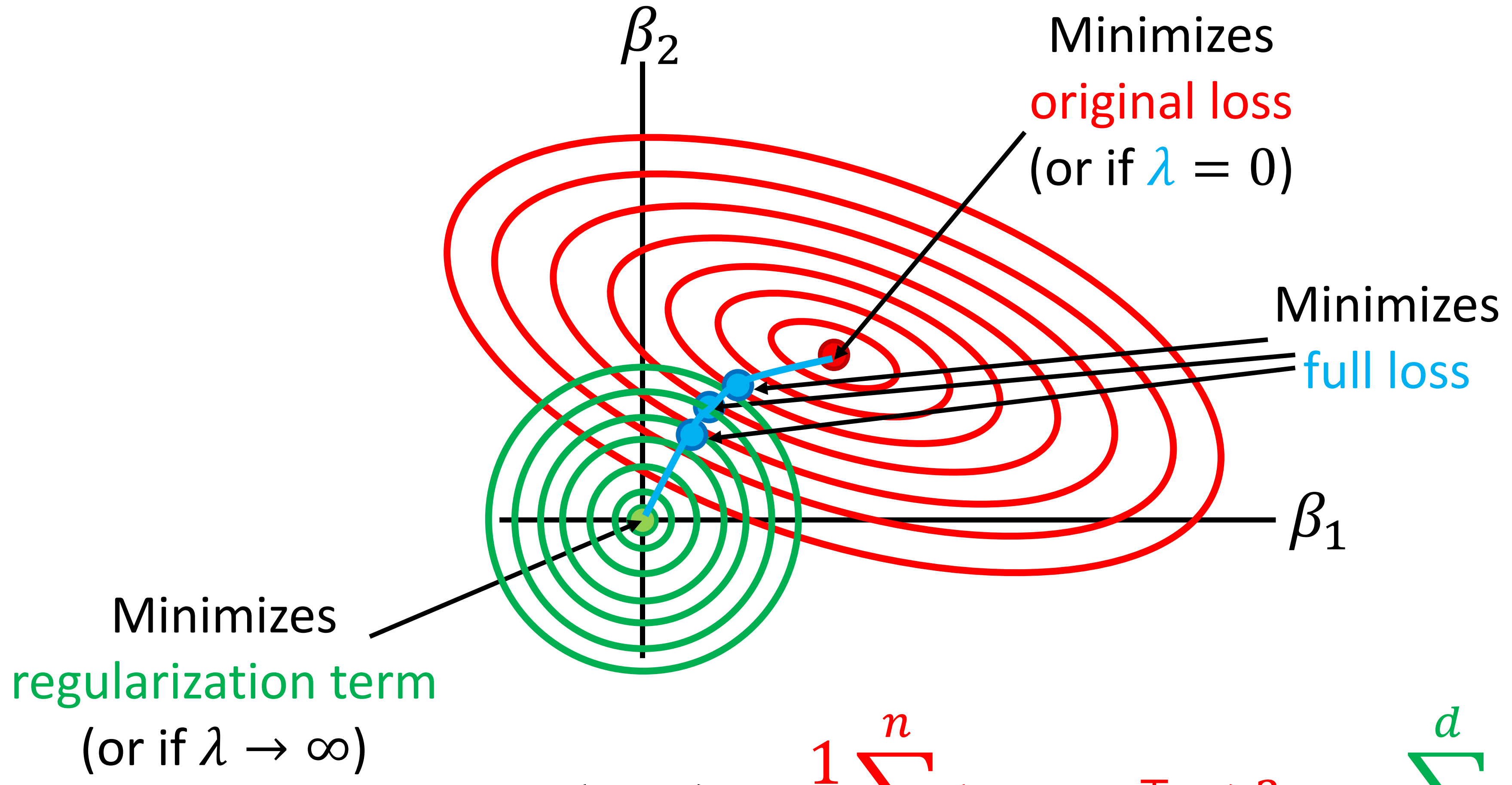
Intuition on L_2 Regularization

- **Why does it help?**
  - Encourages "simple" functions
  - E.g., Use λ to tune bias-variance tradeoff.

Q: How would you set λ to get higher bias / lower variance?



Intuition on L_2 Regularization



- At this point, the gradients are **equal** (with opposite sign)
- Tradeoff depends on choice of λ

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

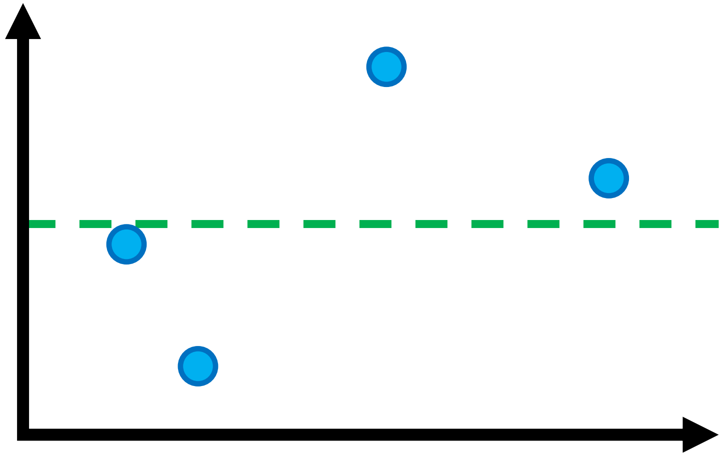
Regularization and Intercept Term / “Bias” Parameter

- **Common convention:** if using intercept term ($\phi(x) = [1 \quad x_1 \quad \cdots \quad x_d]^\top$), no penalty on the “bias parameter” β_1:

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

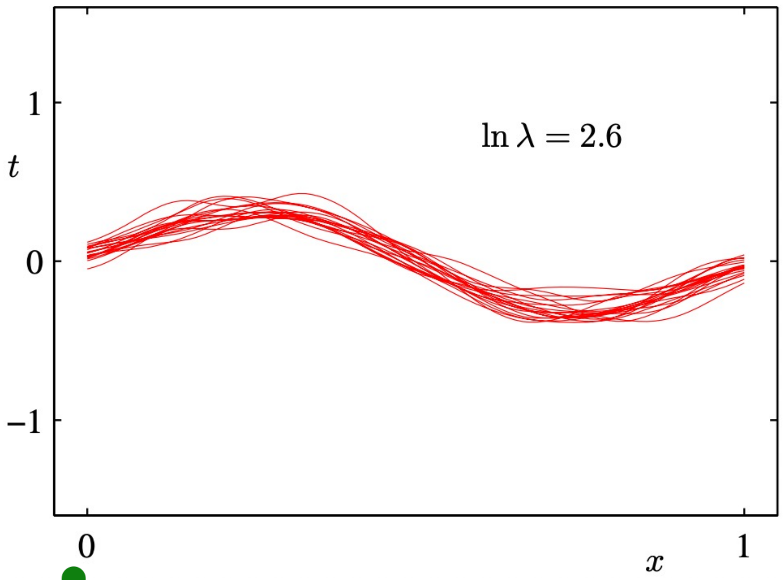
Sum from $j = 2$

- As $\lambda \to \infty$, we have $\beta_2 = \cdots = \beta_d = 0$
  - I.e., only fit β_1 (which yields $\hat{\beta}_1(Z) = \mathrm{mean}(\{y_i\}_{i=1}^n)$)

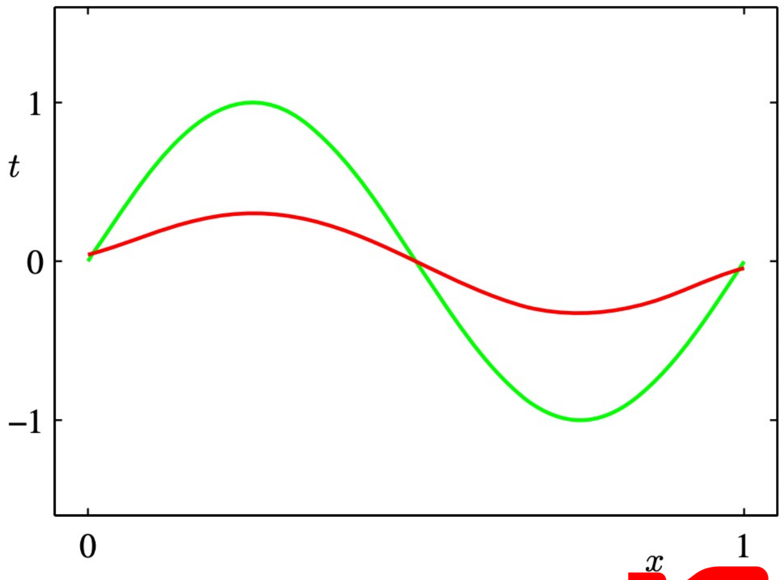


Effect of λ on Bias and Variance

Fits to various datasets sampled from an underlying sinusoid



Mean fit compared to the "true function"



Illustrates variance

Illustrates bias

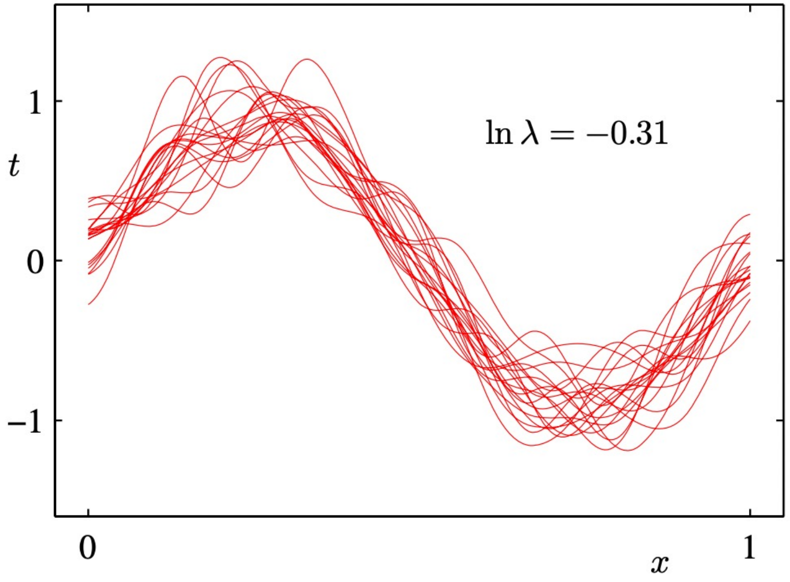
With λ too high, the algorithm finds overly simple solutions.

*dataset size n set to 25, and feature map dimensionality d' is 25 (gaussian feature map)

Bishop, Pattern Recognition and Machine Learning

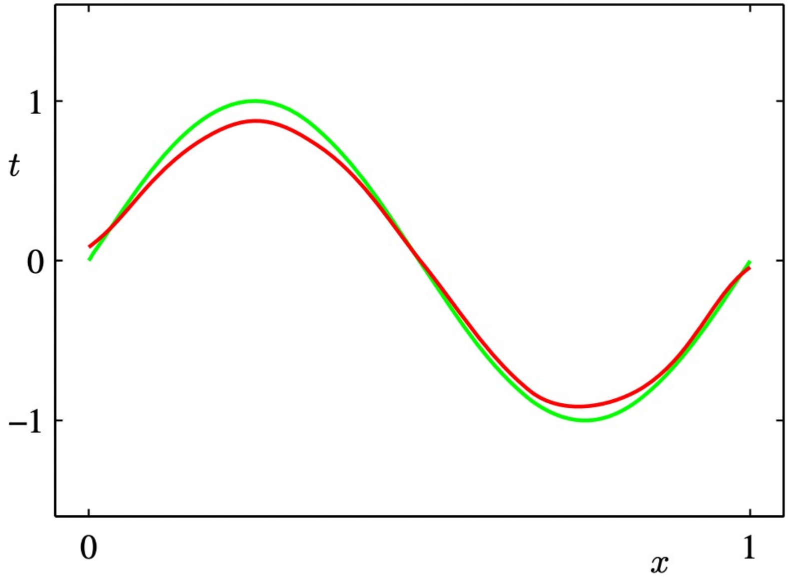
Effect of λ on Bias and Variance

Fits to various datasets sampled from an underlying sinusoid



Illustrates variance

Mean fit compared to the "true function"



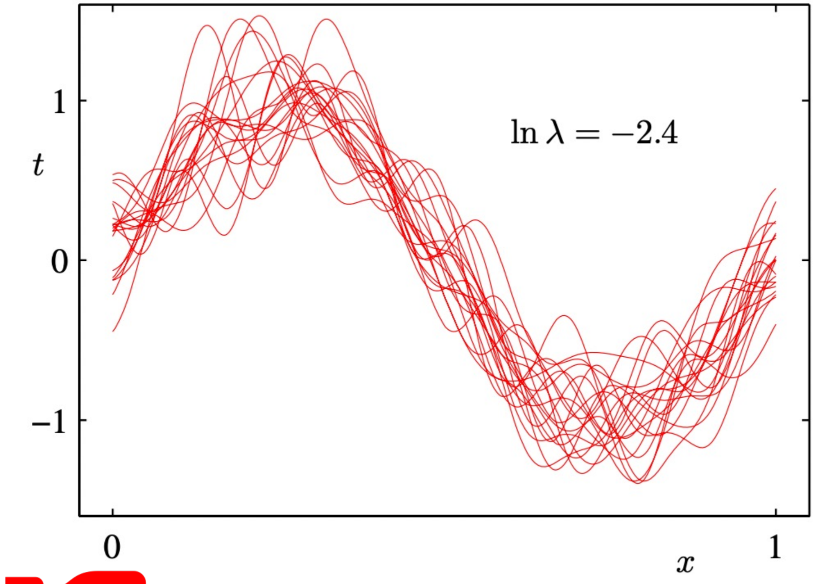
Illustrates bias

*dataset size n set to 25, and feature map dimensionality d' is 25 (gaussian feature map)

Bishop, Pattern Recognition and Machine Learning

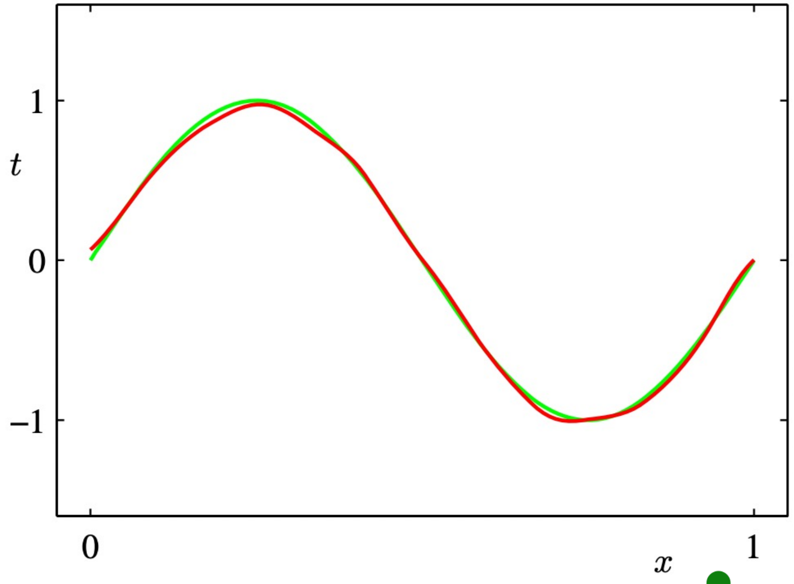
Effect of λ on Bias and Variance

Fits to various datasets sampled from an underlying sinusoid



Illustrates variance

Mean fit compared to the "true function"



Illustrates bias

With λ too low, the algorithm finds overly complex solutions.

*dataset size n set to 25, and feature map dimensionality d' is 25 (gaussian feature map)

Bishop, Pattern Recognition and Machine Learning

General Regularization Strategy

- **Original loss** + **regularization**:

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

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

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