Announcements

- **HW 0** due Wed 8 pm; **HW 1** (on linear regression) will be released that evening.
- Class currently full (181 enrolled, 39 approvals). Limited movement expected.
- **Edstem** to contact the course team, which is likely to have a fast response. But if you want to keep your message private to Tas:
  - **Always** email both instructors together.
  - Start subject line with **“[CIS 4190/5190 Fall 2024]”.**

Lecture 3: Linear Regression (Part 2)

CIS 4190/5190

Fall 2024

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 algorithms for computing the minimal β next lecture

Loss Minimization View of ML

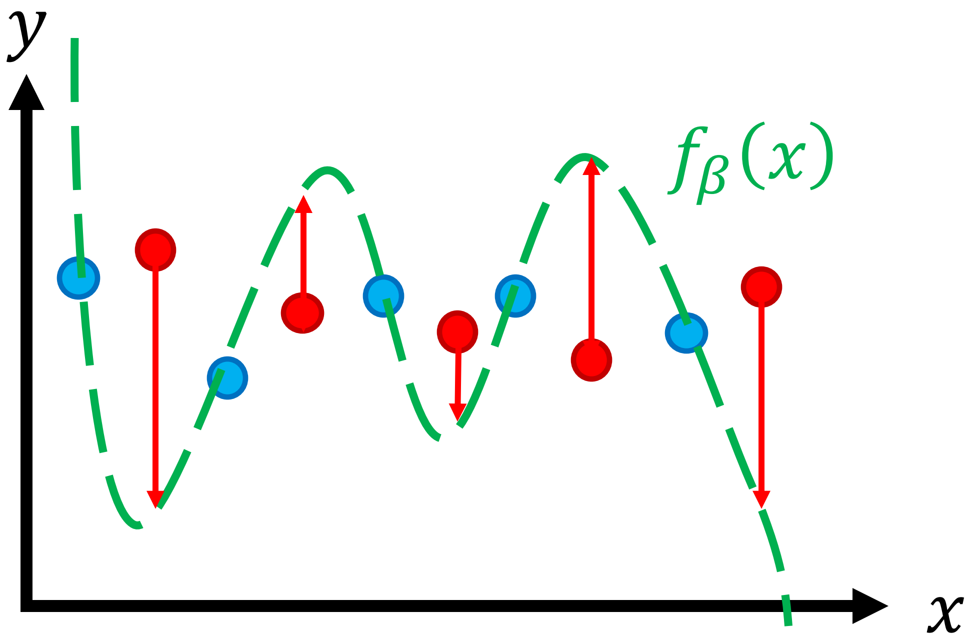
- **To design an ML algorithm:**
  - Choose model family $F = \{f_\beta\}_\beta$ (e.g., linear functions)
  - Choose loss function $L(\beta; Z)$ (e.g., MSE loss)

- **Resulting algorithm:**

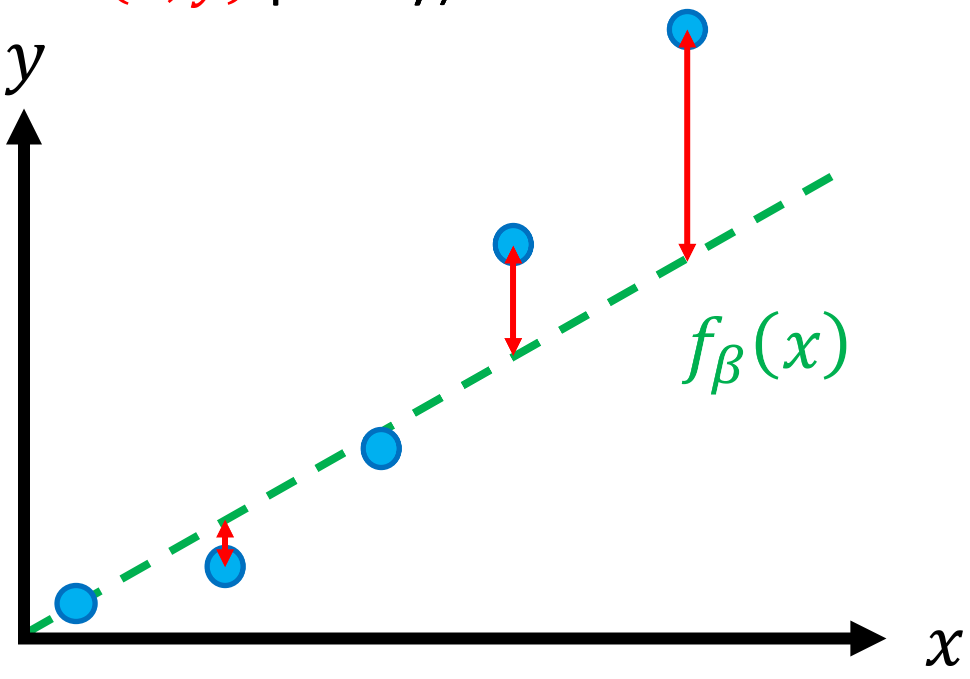
$$\hat{\beta}(Z) = \arg\min_{\beta} L(\beta; Z)$$

Recap: Overfitting vs. Underfitting

- **Overfitting**
  - Fit the **training data** Z well
  - Fit new **held out data** (x, y) poorly



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



Today’s Lecture

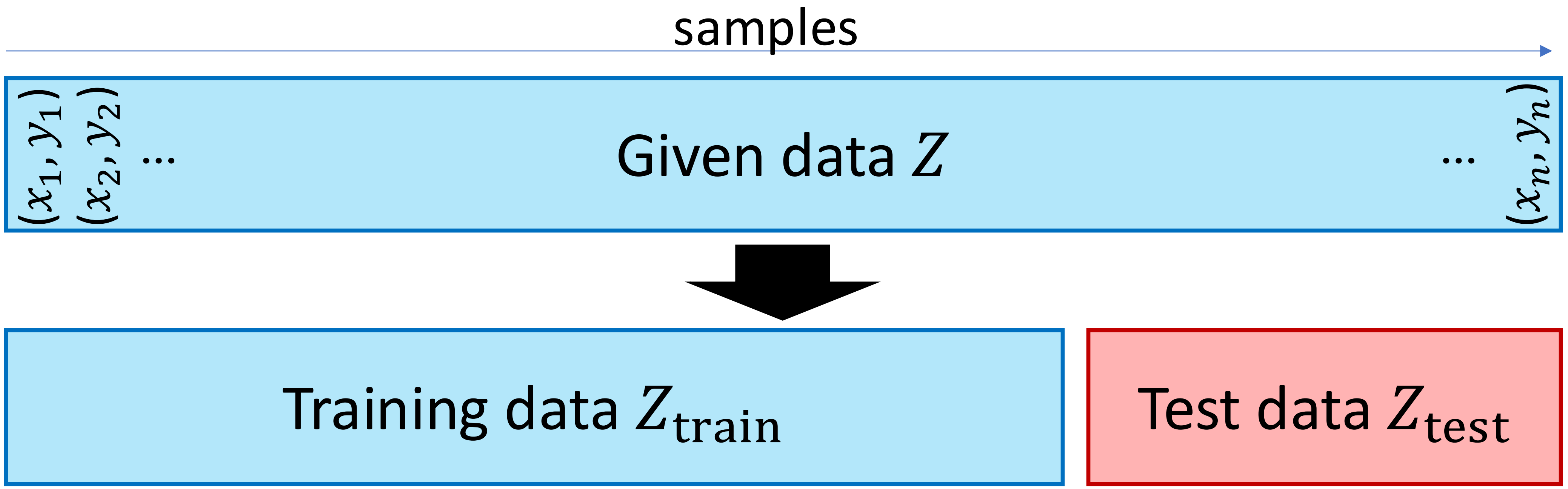
Assessing, Understanding, and Combating underfitting/overfitting:

- Bias and Variance of hypothesis classes
- Regularized linear regression
- Cross-Validation

Assessing Underfitting & 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 in ML

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

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

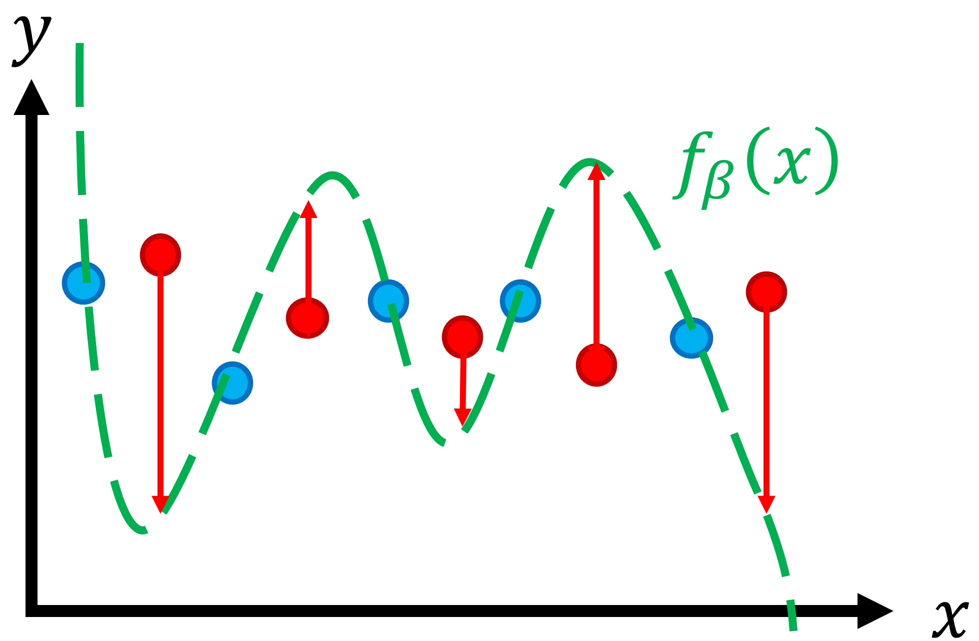
- **Step 2:** Run linear regression with Z_{train} to obtain $\hat{\beta}(Z_{\text{train}})$

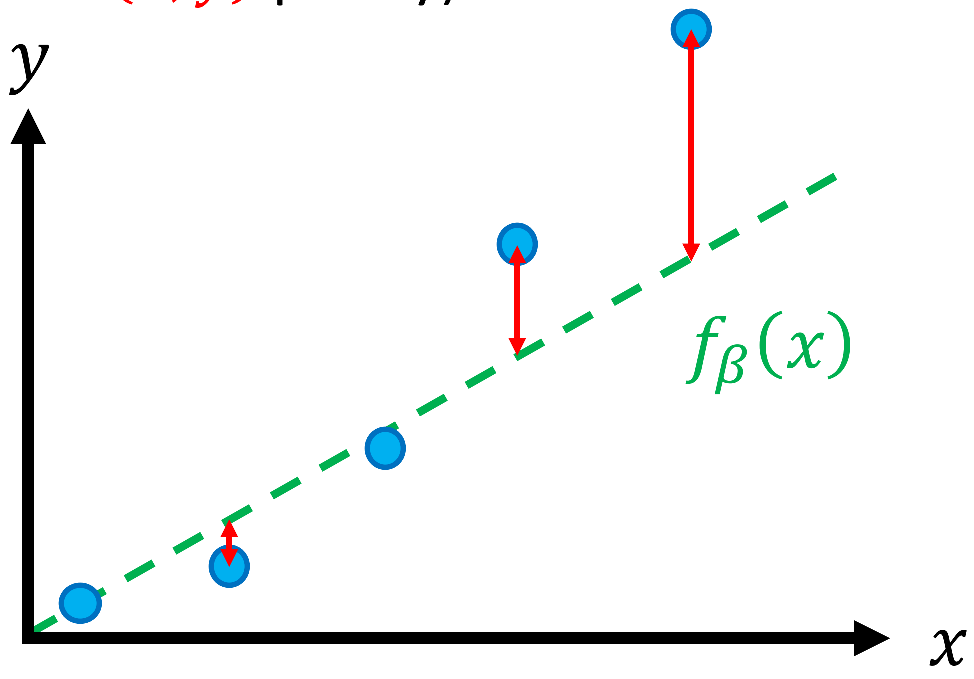
- **Step 3:** Evaluate
  - **Training loss:** $L_{\text{train}} = L\left(\hat{\beta}(Z_{\text{train}}); Z_{\text{train}}\right)$
  - **Test (or generalization) loss:** $L_{\text{test}} = L\left(\hat{\beta}(Z_{\text{train}}); Z_{\text{test}}\right)$, (plus other performance metrics besides the loss function)

Training/Test Split Protocol in ML

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

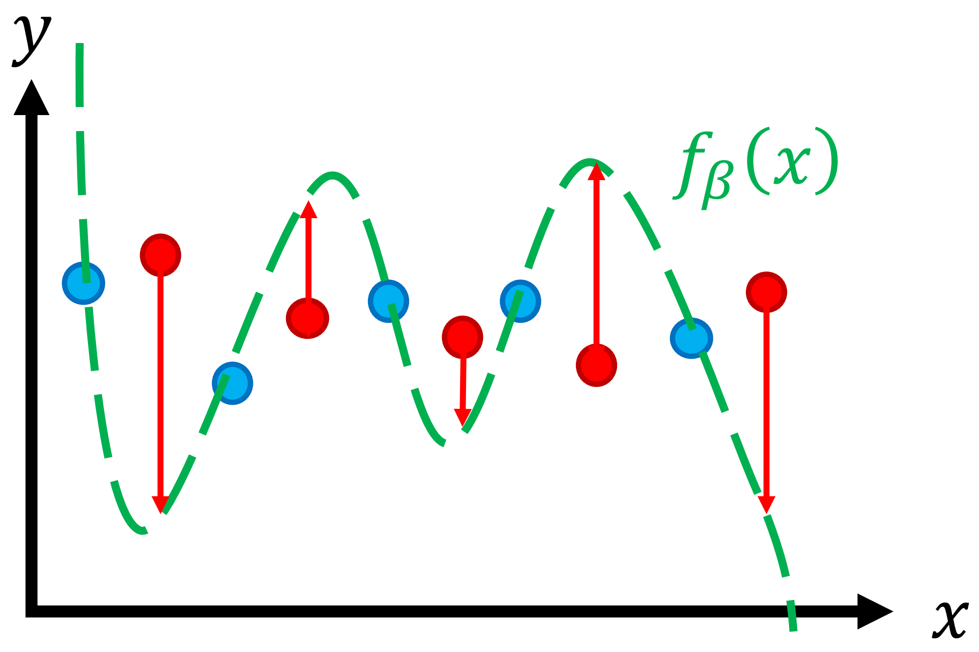
- **Underfitting**
  - Fit the **training data** Z poorly
  - (Necessarily fit new **test data** (x, y) poorly)



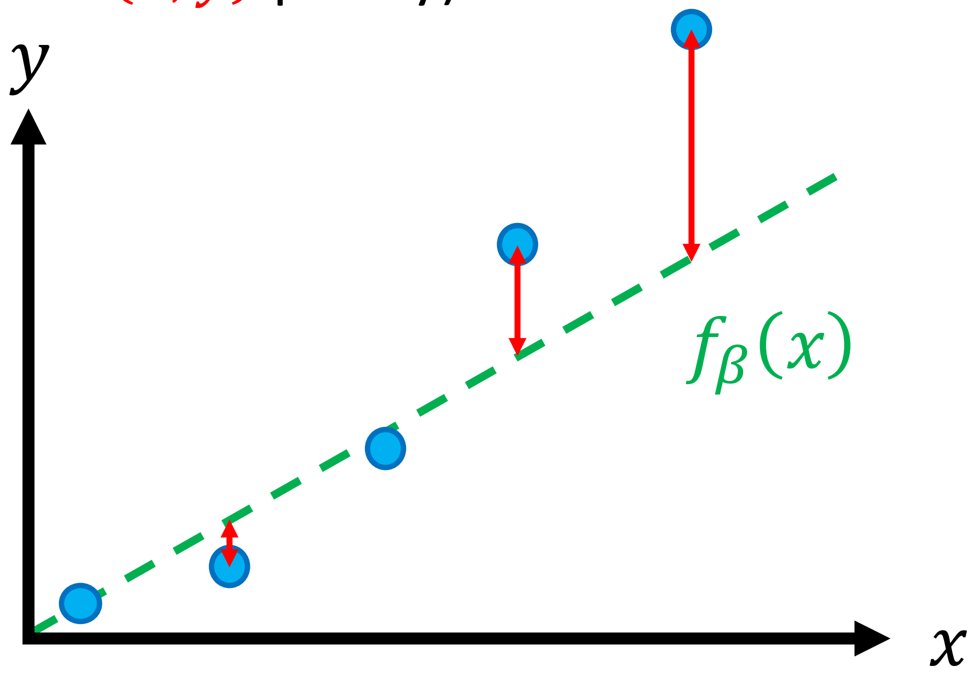


Training/Test Split Protocol in ML

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

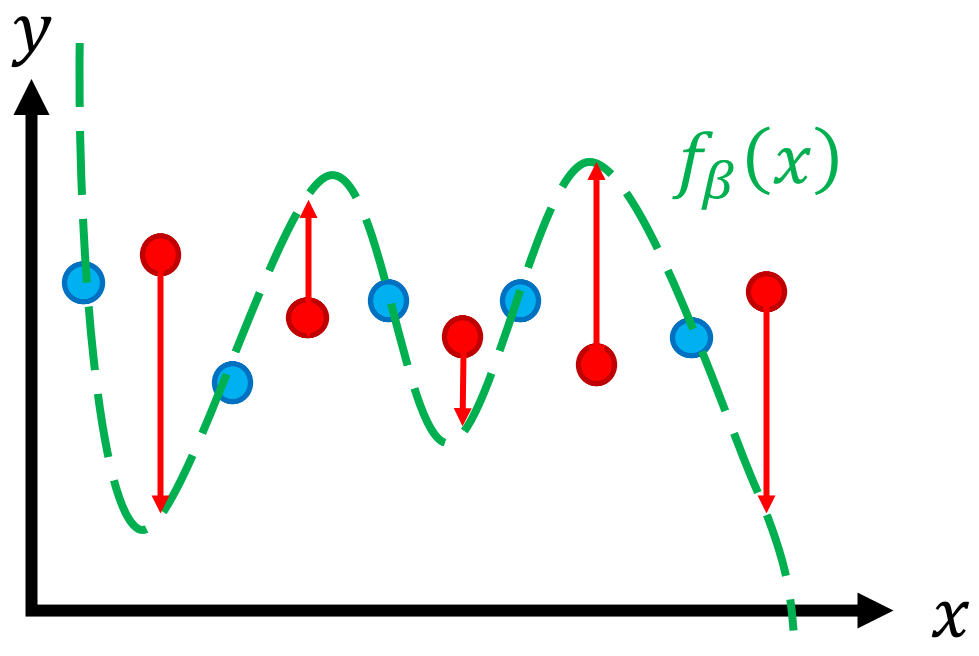


- **Underfitting**
  - Fit the **training data** Z poorly
  - (Necessarily fit new **test data** (x, y) poorly)

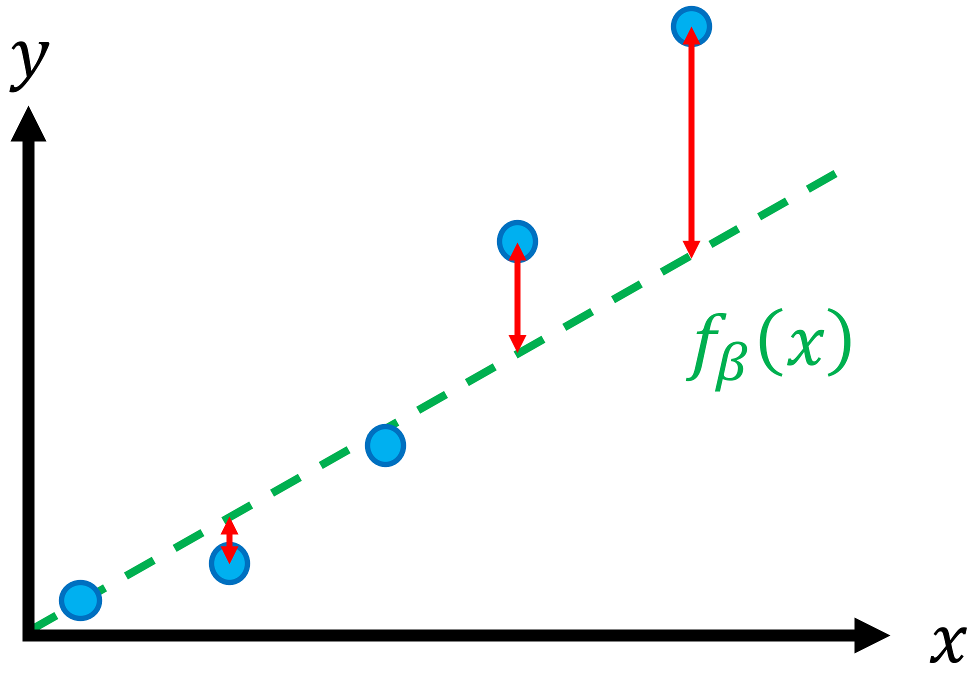


Training/Test Split Protocol in ML

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



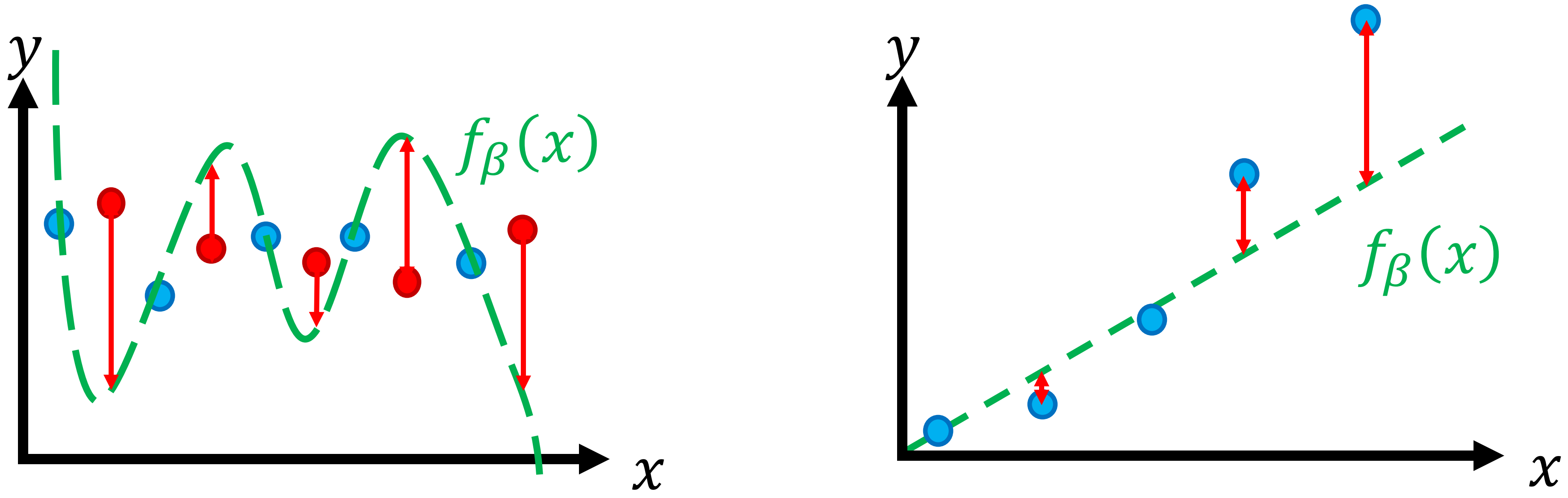
- **Underfitting**
  - L_{train} is large
  - L_{test} is large



Understanding Underfitting & Overfitting

With Bias & Variance

Underfitting/Overfitting <=> Bias/Variance



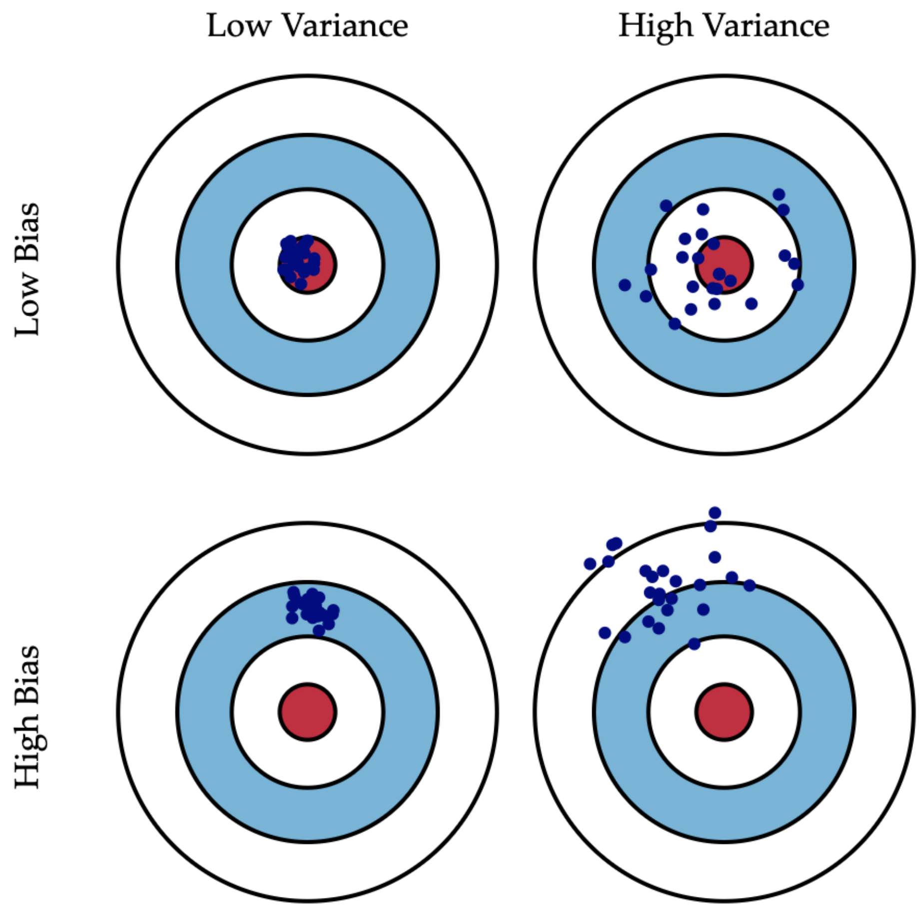
We will understand these phenomena now through two properties of a model family, “bias”, and “variance”.

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

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?

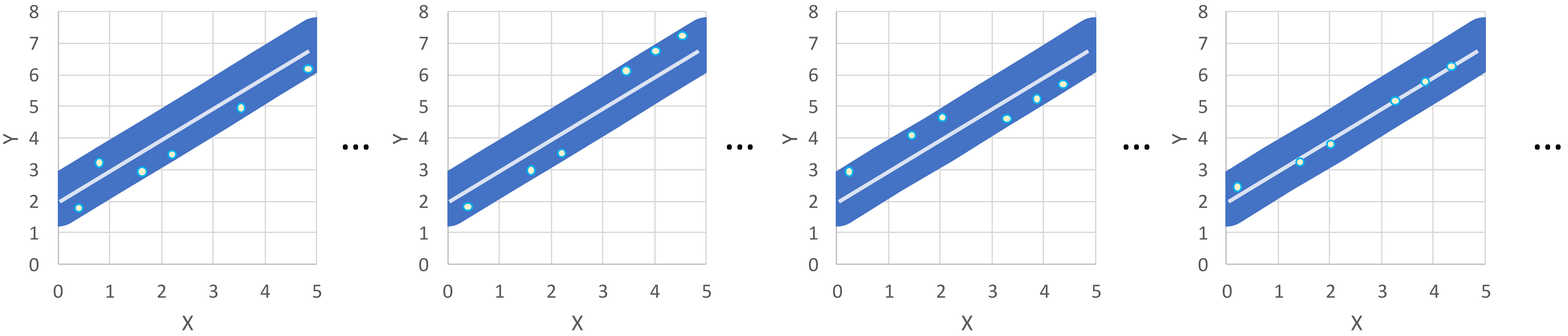


Scott Fortmann-Roe
http://scott.fortmann-roe.com/docs/BiasVariance.html

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].

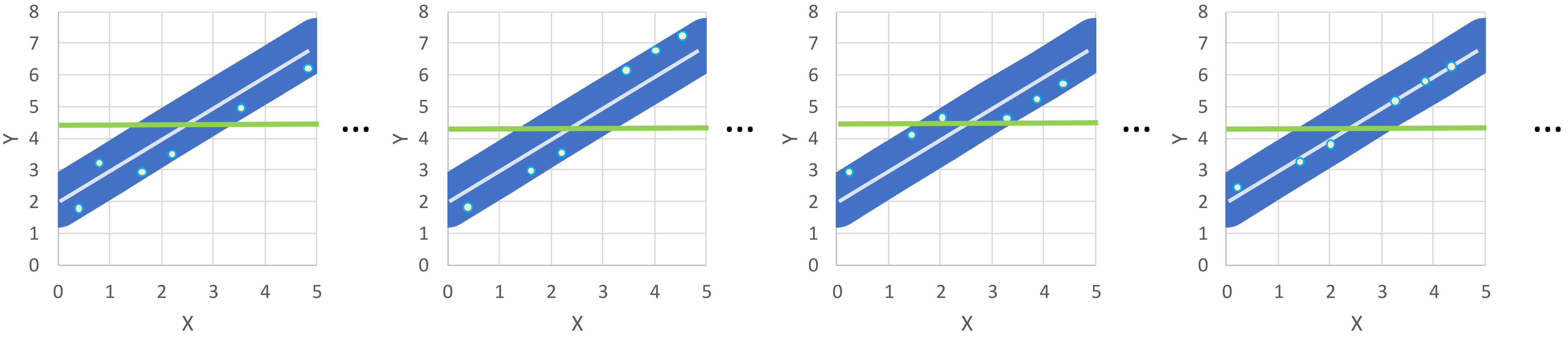
Let us draw $k \to \infty$ training sets of $n = 6$ samples each, drawn from $P(X, Y)$.



Different *Constant* Fits

What if the hypothesis class was the constant function class

$$f_{\boldsymbol{\beta}}(x) = \beta_0$$



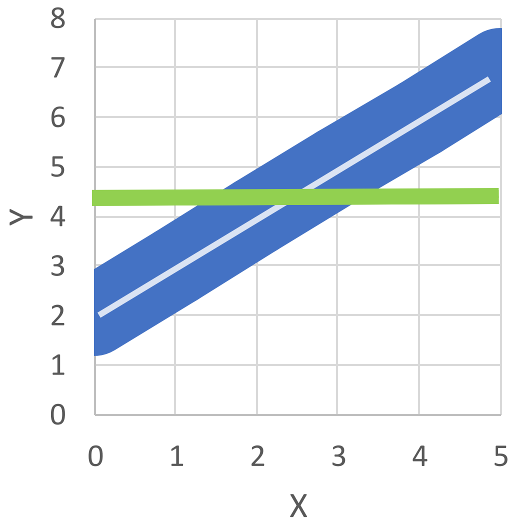
Different *Constant* Fits

What if the hypothesis class was the constant function class

$$f_{\boldsymbol{\beta}}(x) = \beta_0$$

Almost identical fits

"low variance"



Average fit far from the true function

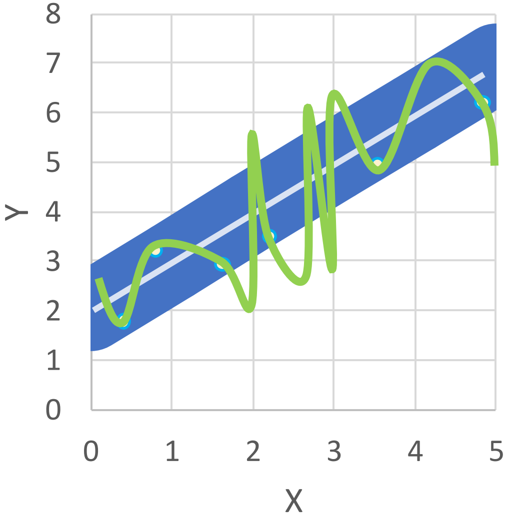
"high bias"

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

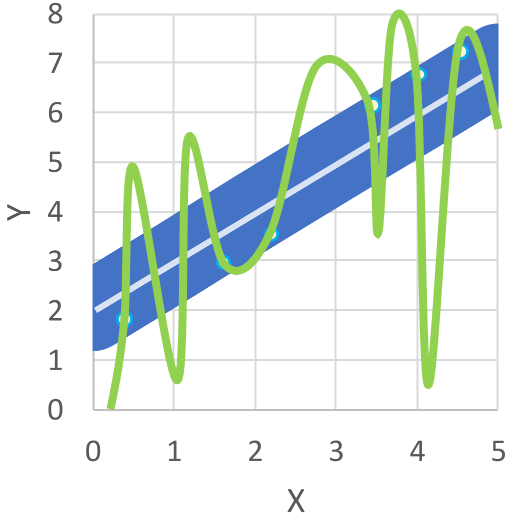
Different 10^{th} Degree Curve Fits

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

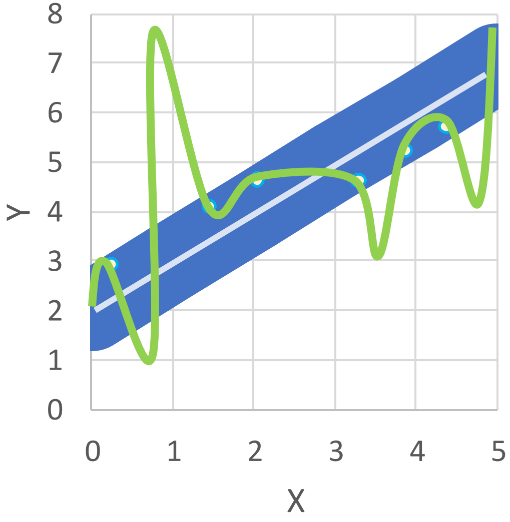
$$f_{\boldsymbol{\beta}}(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \cdots \beta_{10} x^{10}$$



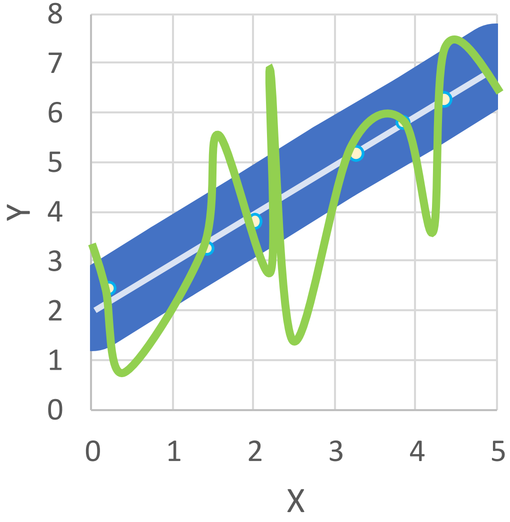
…



…



…



…

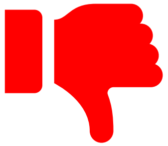
Different 10^{th} *Degree* Fits

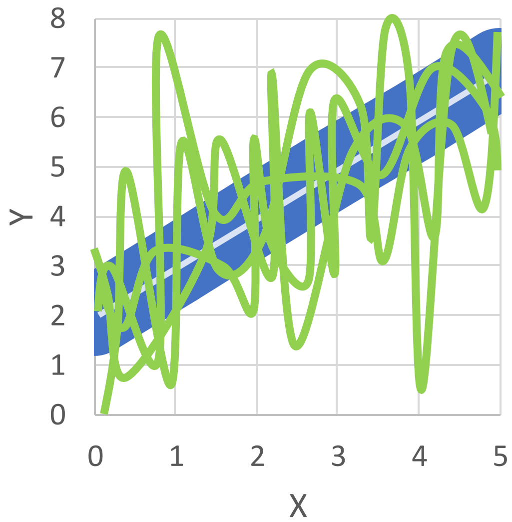
What if the hypothesis class was instead a 10^{th} degree monomial

$$f_{\boldsymbol{\beta}}(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \cdots \beta_{10} x^{10}$$

Very different fits

"high variance"





Average fit close to the true function

"low bias"

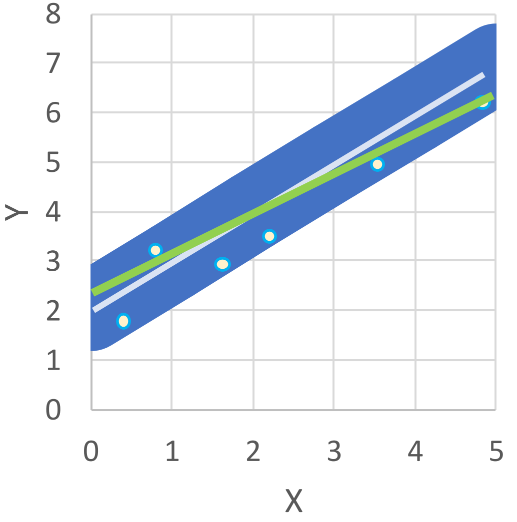
Theoretical result: Generalization MSE ≈ ``Bias'' + ``Variance''

Different *Linear* Fits

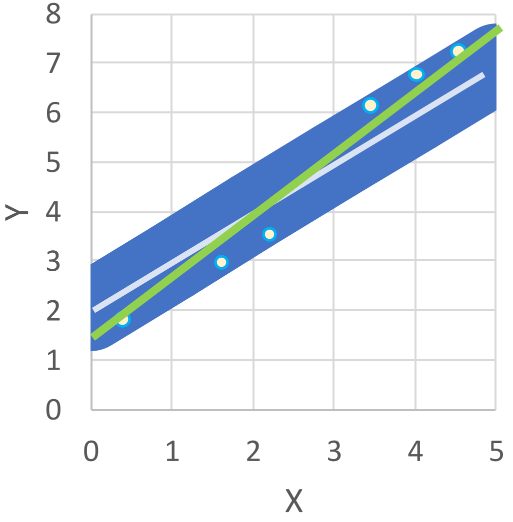
Say, our hypothesis class is a line:

$$f_{\boldsymbol{\beta}}(x) = \beta_0 + \beta_1 x_1$$

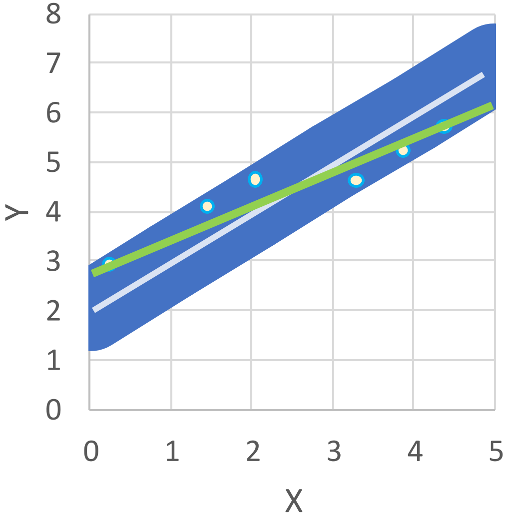
Fit by minimizing MSE with any optimizer. What would the resulting line look like?



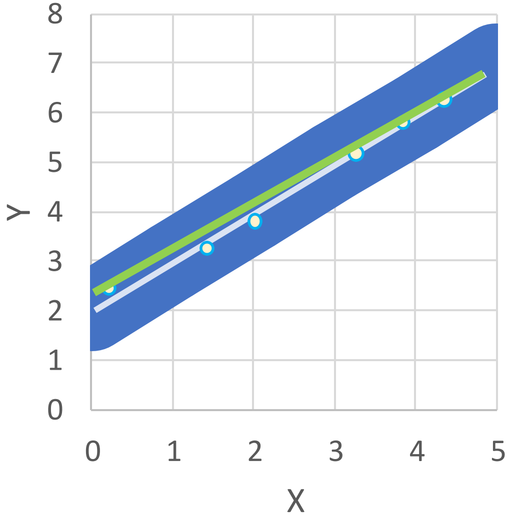
…



…



…



…

Slightly different fits

Different *Linear* Fits

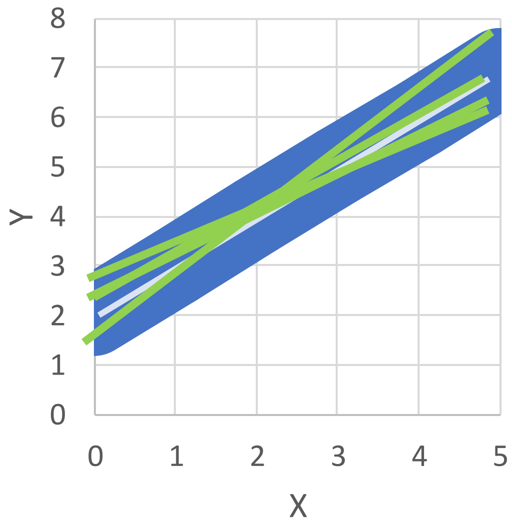
Say, our hypothesis class is a line:

$$f_{\boldsymbol{\beta}}(x) = \beta_0 + \beta_1 x_1$$

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

Quite similar fits

“low variance”



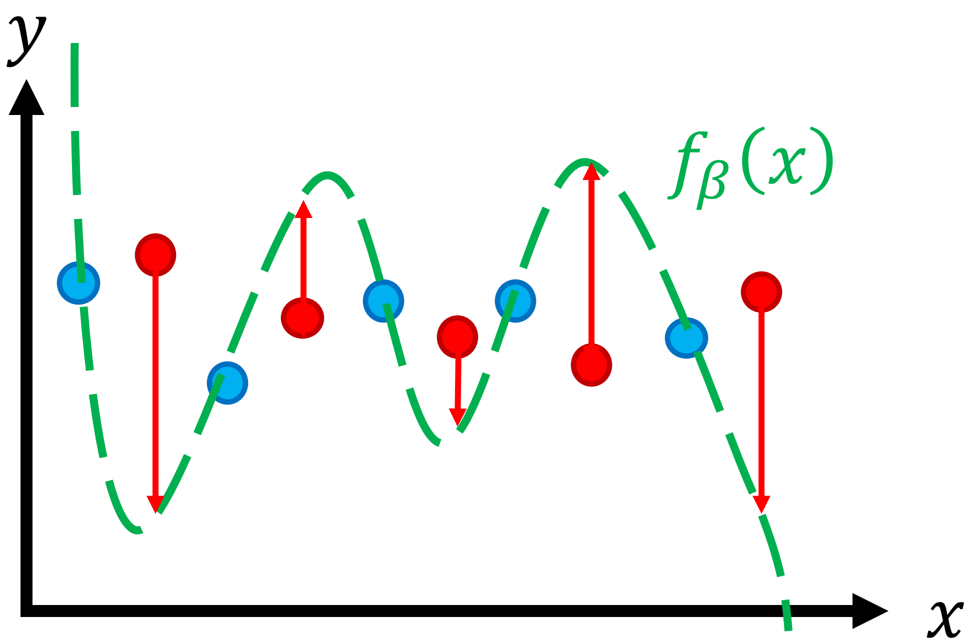
Average fit close to the true function

“low bias”

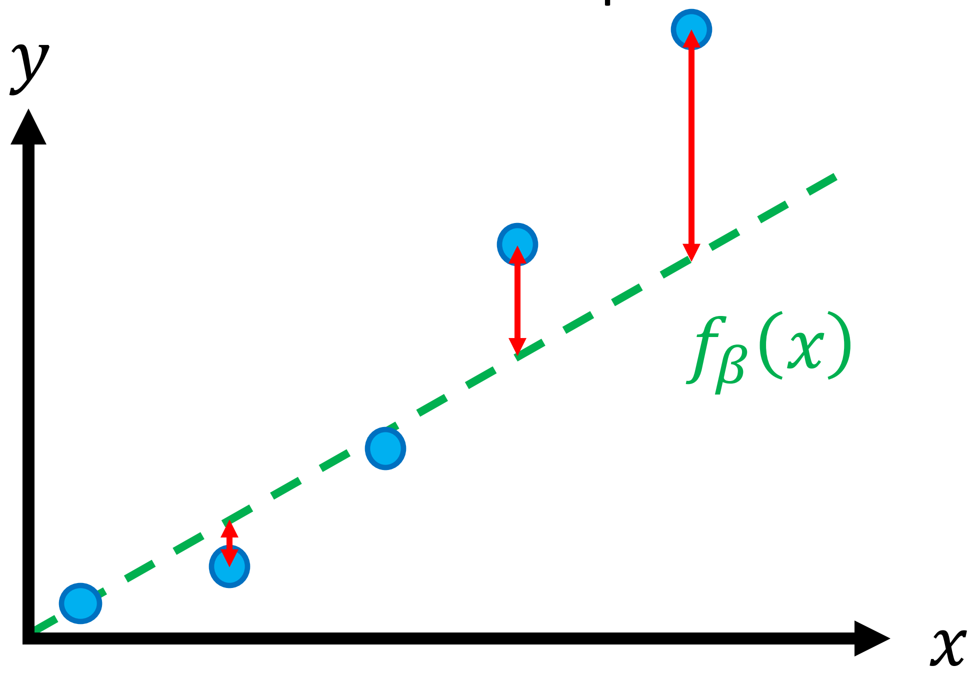
Theoretical result: Generalization MSE $\approx$ ``Bias'' + ``Variance''

Bias-Variance Tradeoff

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



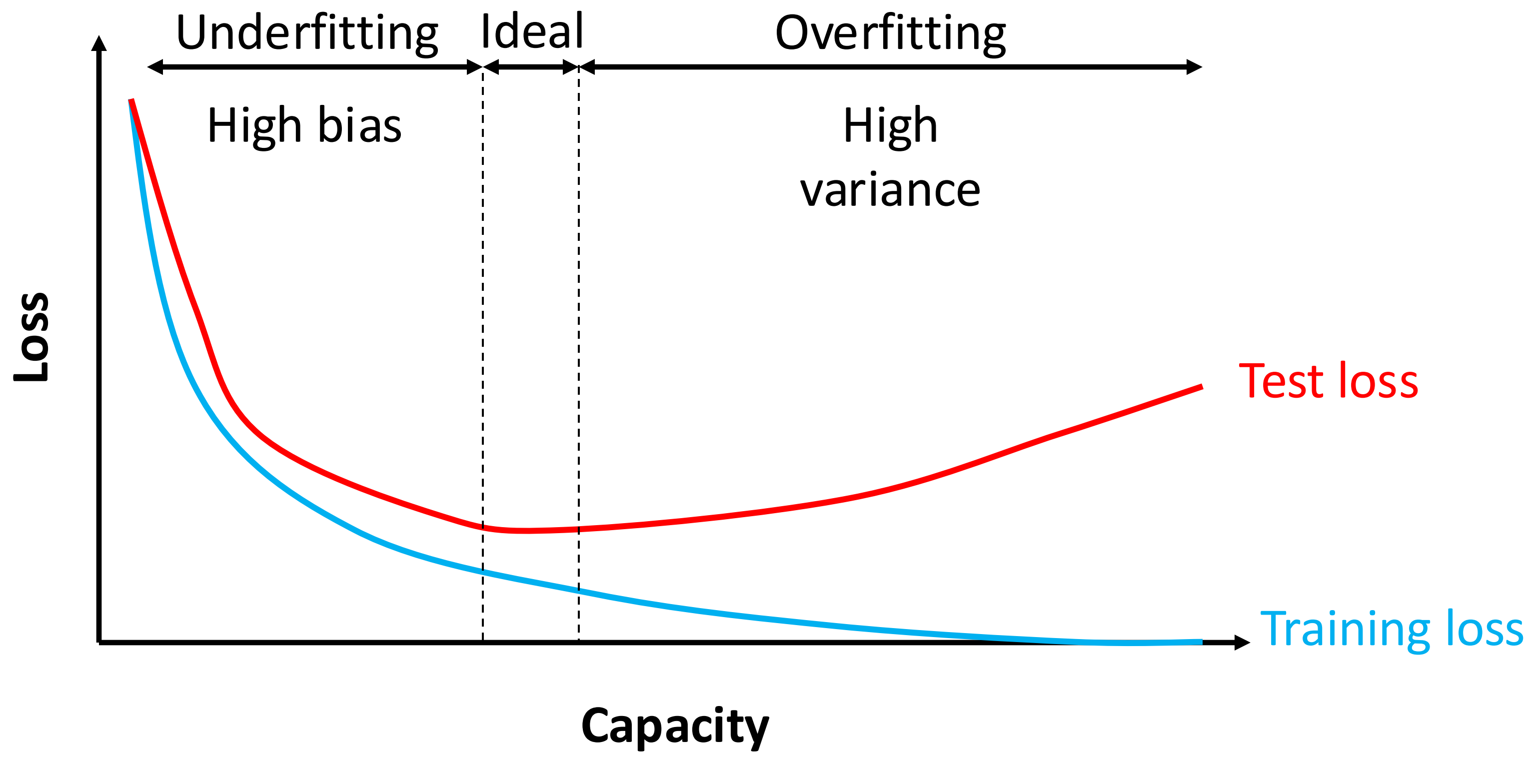
- **Underfitting (high bias)**
  - Low capacity model that can only fit simple data
  - Sufficient data but poor fit



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 $\boldsymbol{\phi}$ 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.

How to Fix Underfitting/Overfitting?

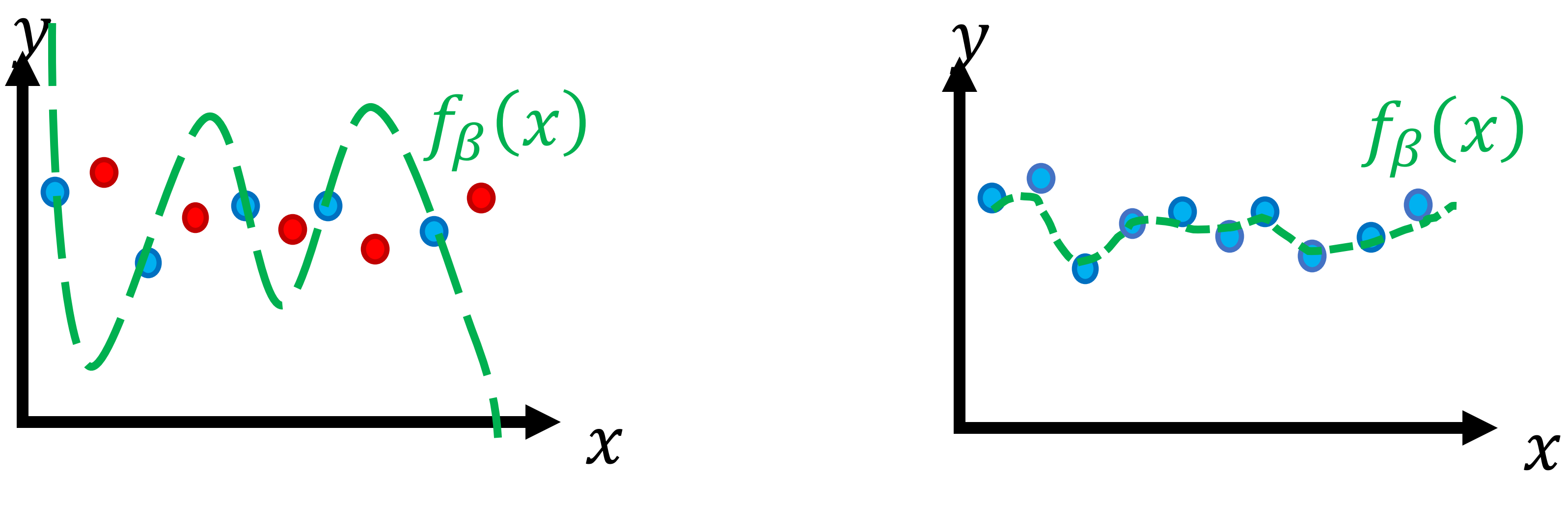
Three main options:

- 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 ($\approx$ ``$\text{Bias}''$ $+$ ``$\text{Variance}''$) 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

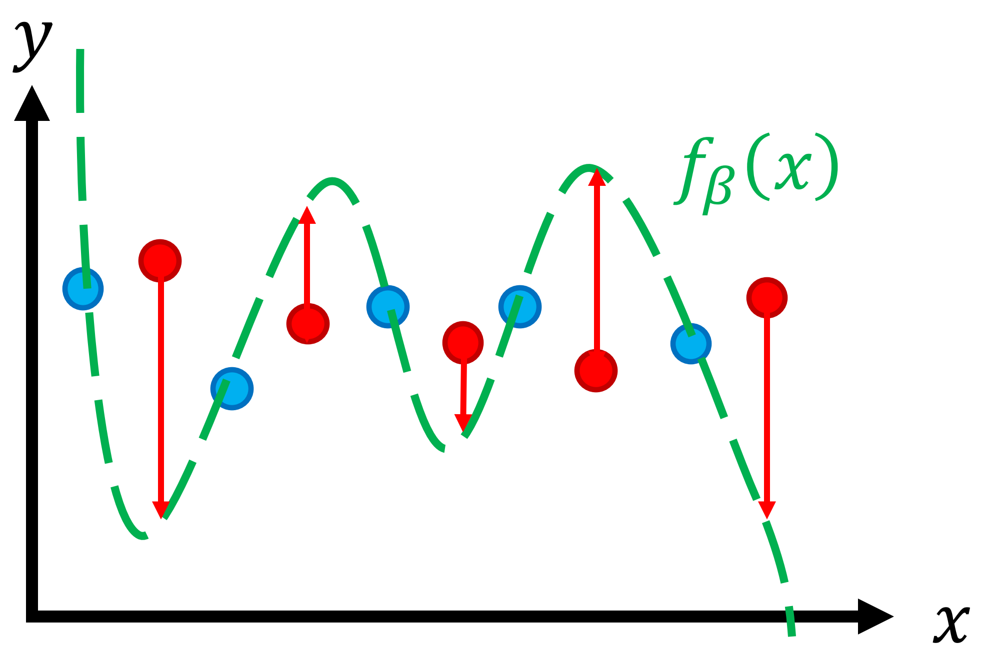
- **Intuition:** We *only* asked the ML algorithm to fit the training data as well as possible, so it produced overly complex fits → "Overfitting"

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

- **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

One measure of fit complexity

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

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

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

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

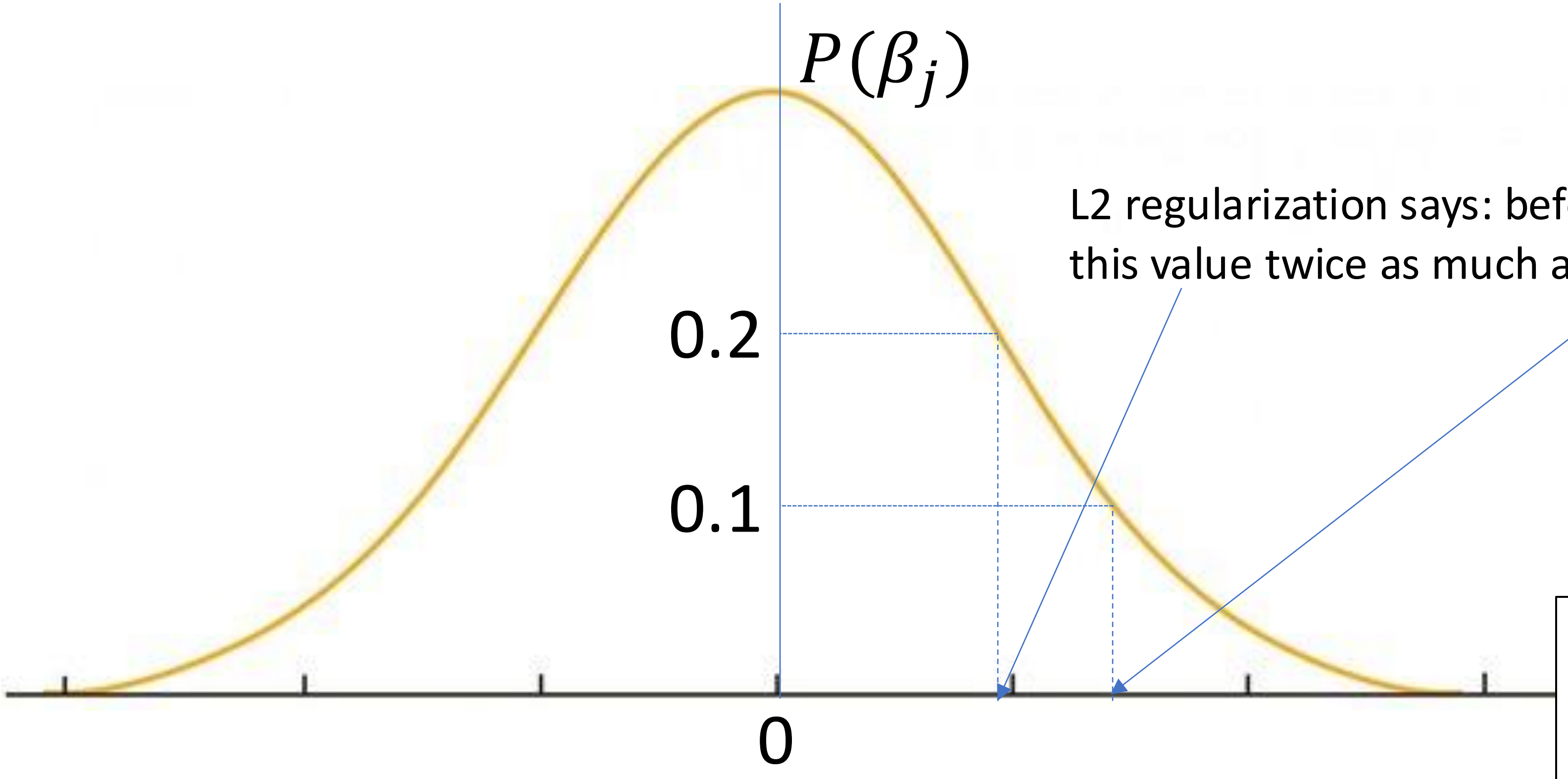
Intuition on L_2 Regularization

Why does it help?

- Encourages "simple" functions
  - 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.



Parameter value for any β_j

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

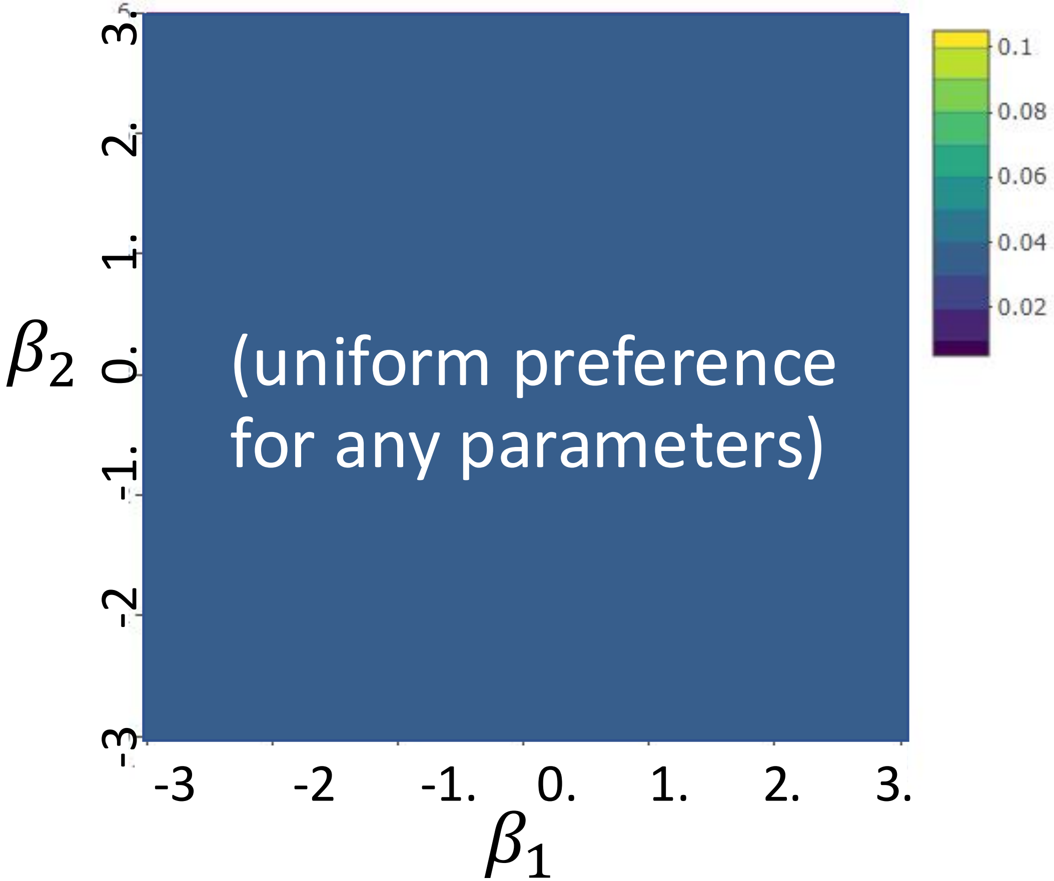
So the larger value is only selected for the model if it is *much* better for the data fit term (MSE)

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

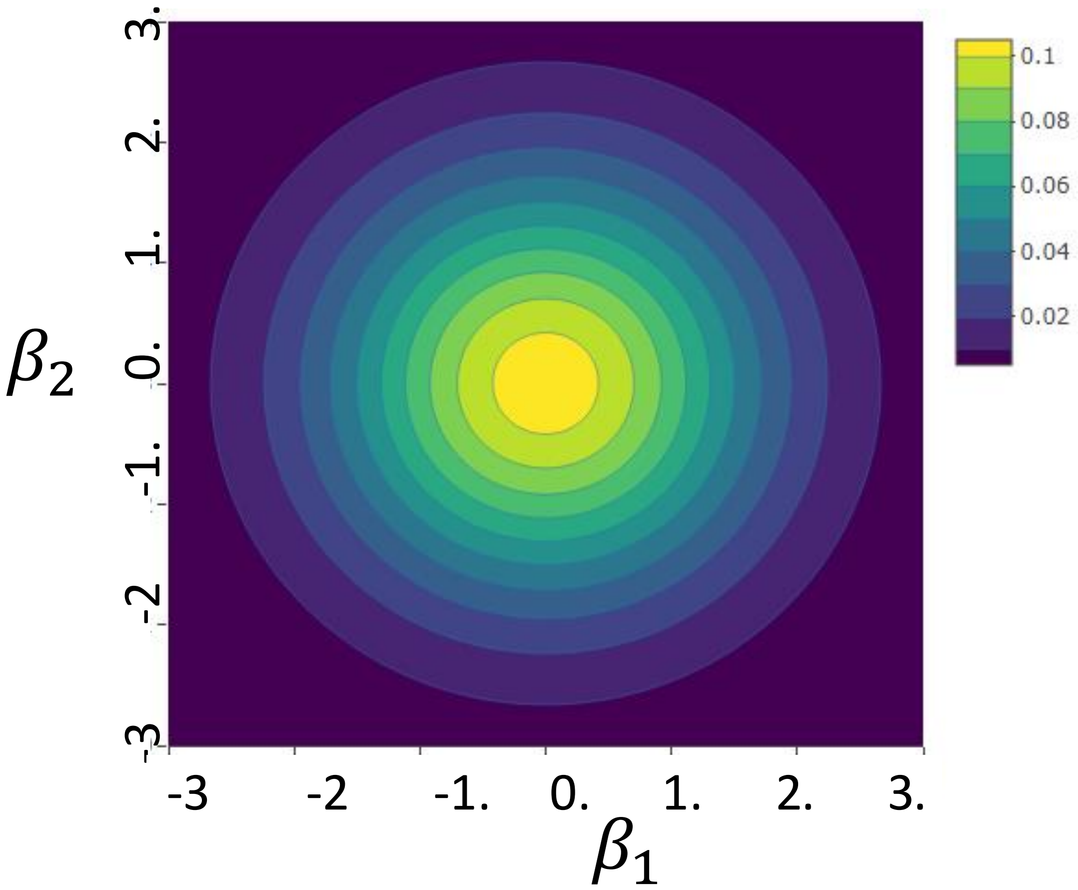
$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda \cdot \|\beta\|_2^2$$

Intuition on L_2 Regularization: Gaussian Priors

Before regularization



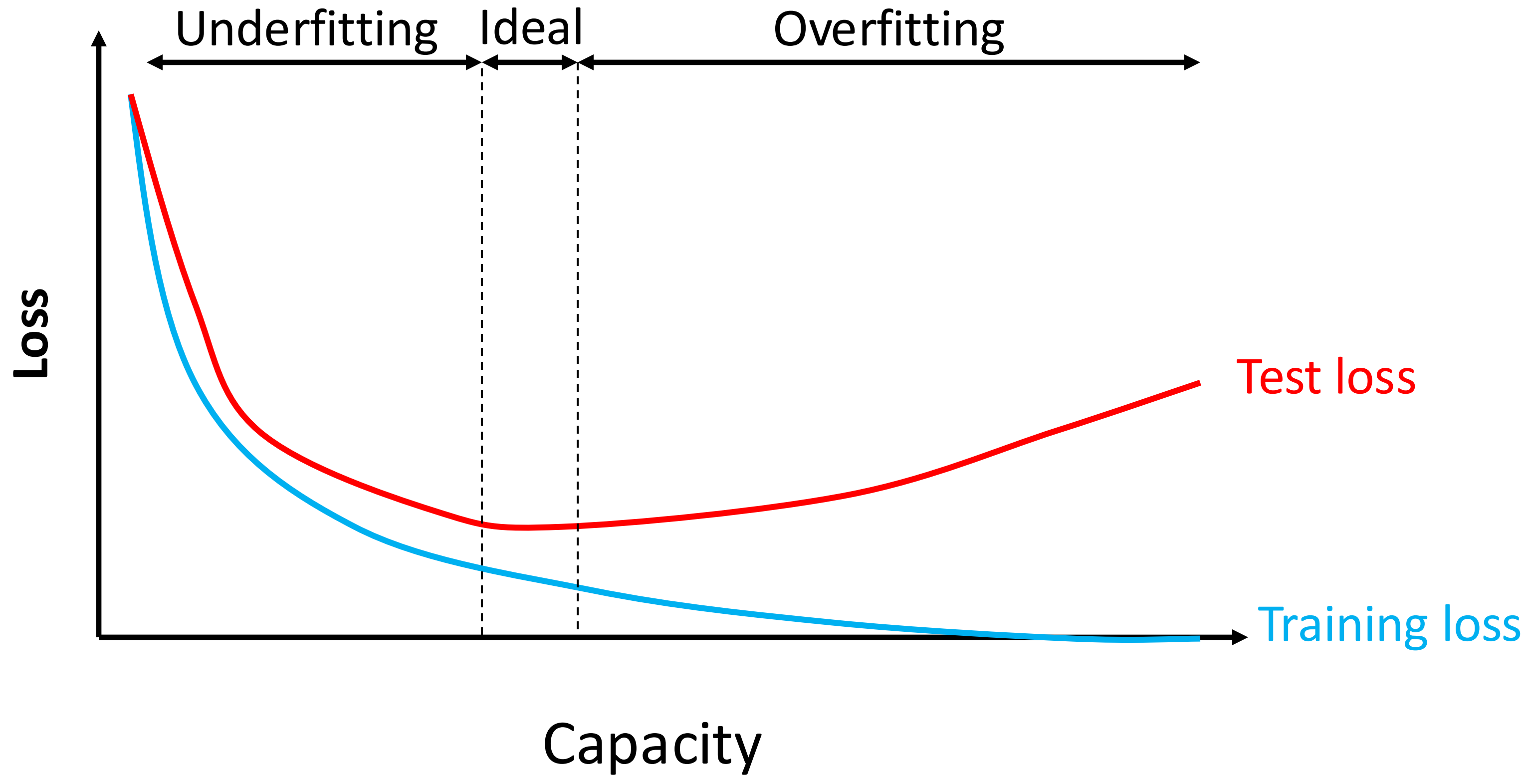
With L2 regularization



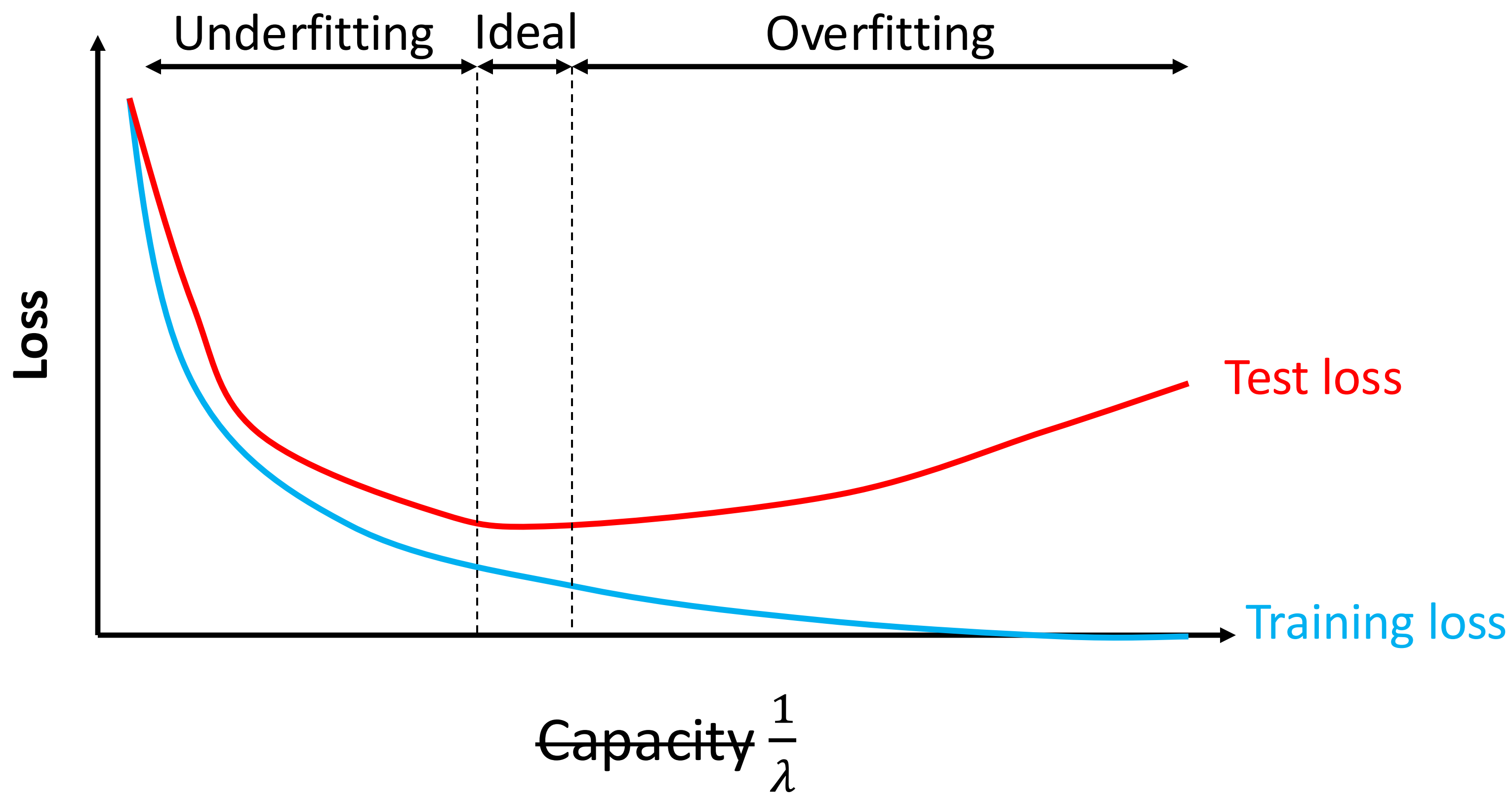
Intuition on L_2 Regularization

- Encourages “simple” functions
- Encouraging β_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:**

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

  - 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?

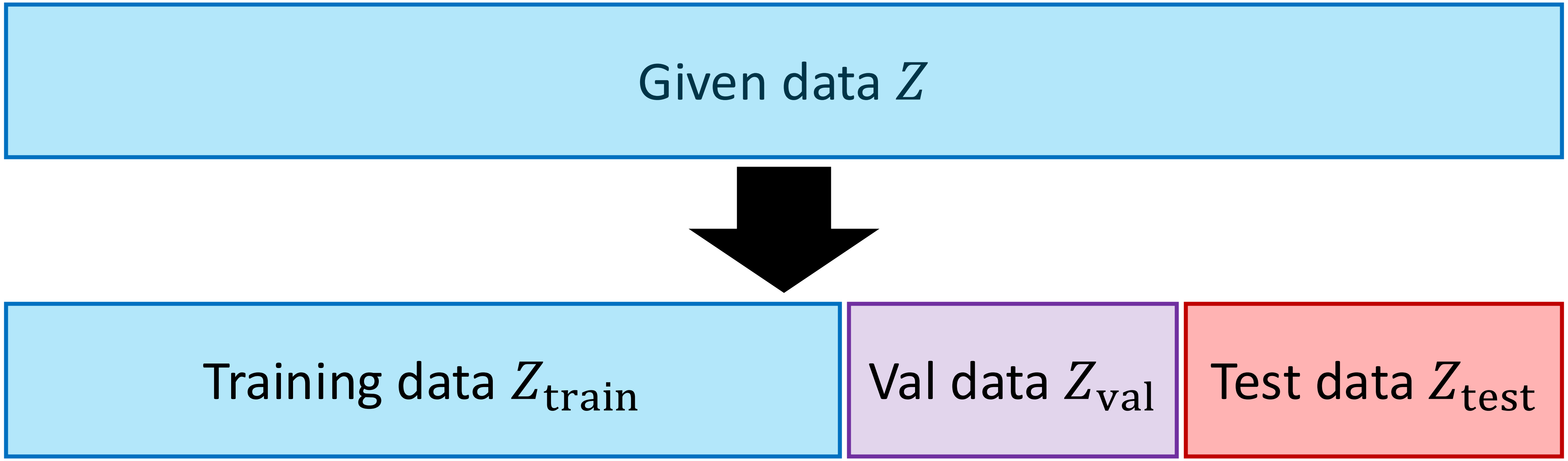
Hyperparameter Tuning & Model Selection

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)



Basic Cross Validation Algorithm

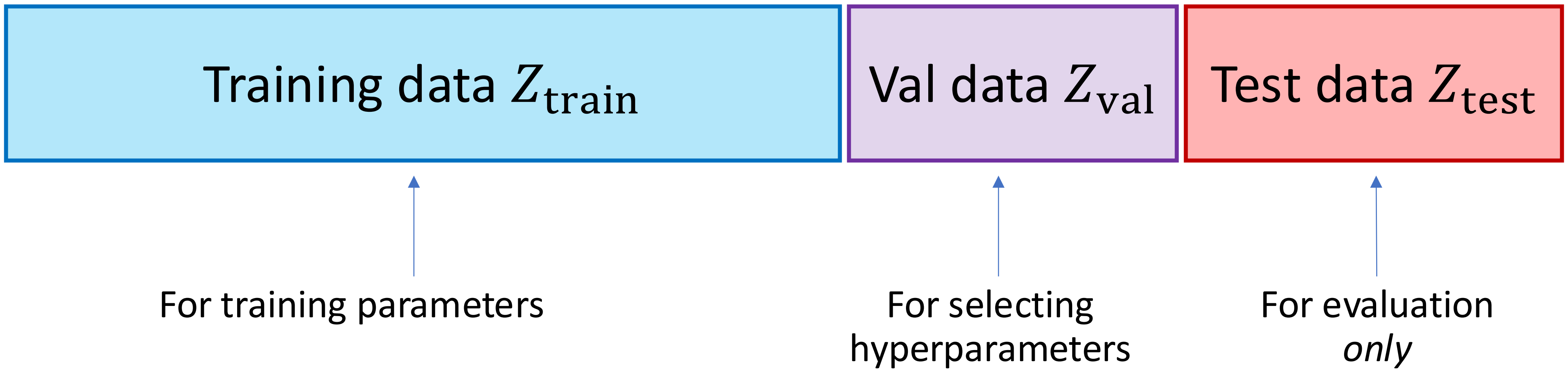
- **Step 1:** Split Z into Z_{train}, Z_{val}, and Z_{test}

Training data Z_{train}	Val data Z_{val}	Test data Z_{test}

- **Step 2:** For $t \in \{1, \dots, h\}$ hyperparameter choices:
  - **Step 2a:** Run linear regression with Z_{train} and λ_t to obtain $\hat{\beta}(Z_{\text{train}}, \lambda_t)$
  - **Step 2b:** Evaluate validation loss $L_{\text{val}}^t = L\left(\hat{\beta}(Z_{\text{train}}, \lambda_t); Z_{\text{val}}\right)$

- **Step 3:** Use best λ_t
  - Choose $t' = \arg\min_t L_{\text{val}}^t$ with lowest validation loss
  - Re-run linear regression with Z_{train} and $\lambda_{t'}$ to obtain $\hat{\beta}(Z_{\text{train}}, \lambda_{t'})$

Cross Validation Hygiene



- The moment that test data is used for hyperparameter selection or to iterate on ML design choices, it should be treated as “contaminated”.
- Remember: Performance on contaminated test data is an overly ***optimistic*** estimate of the “true” test performance.

Alternative Cross-Validation Algorithms

- If Z is small, then splitting it can reduce performance
  - Can use $Z_{\text{train}} \cup Z_{\text{val}}$ in Step 3
- **Alternative more thorough CV strategy:** "k-fold" cross-validation
  - Split Z into Z_{train} and Z_{test}
  - Split Z_{train} into k disjoint sets Z_{val}^{s}, and let $Z_{\text{train}}^{s} = \bigcup_{s' \neq s} Z_{\text{val}}^{s'}$
  - Use λ' that works best on average across $s \in \{1, \ldots, k\}$ with Z_{train}^{s}
  - Chooses better λ' than above strategy

Example: $k = 3$-Fold Cross Validation

Training data Z^3_{train}		Val data Z^3_{val}	Test data Z_{test}
Train data Z^2_{val}	Val data Z^2_{val}	Train data Z^2_{val}	Test data Z_{test}
Val data Z^1_{val}	Train data Z^1_{train}		Test data Z_{test}
Train data Z_{train}			Test data Z_{test}

Compute vs. accuracy tradeoff: As $k \to N$, model selection becomes more accurate, but algorithm becomes more computationally expensive

k-Fold Cross-Validation

- **Compute vs. accuracy tradeoff**
  - As $k \to N$, the model becomes more accurate
  - But algorithm becomes more computationally expensive

Note: What Exactly Are “Hyperparameters”?

- Cross-Validation is a general, systematic trial-and-error procedure for selecting hyperparameters.
- Other hyperparameters too, not *just* the regularization λ.
- “Hyperparameters” are ML system properties / design choices that are not directly set in the optimization problem.

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

- Examples of other hyperparameters you could set with cross-validation:
  - choice of feature maps in linear regression.
  - data selection and other preprocessing procedures (coming up soon).
  - linear regression versus another ML algorithm, altogether.

Today’s Lecture

Assessing, Understanding, and Combating underfitting/overfitting:

- Bias and Variance of hypothesis classes
- Regularized linear regression
- Cross-Validation

Next Lecture

- How to find $\hat{\beta}(Z) = \arg\min_{\beta} L(\beta; Z)$