Announcements

- Homework 1 due **Wednesday at 8pm**
- Quiz 1 released on Thursday (on Canvas)
- Office hours posted on Course Website (starting **today**!)
    - See announcement on Ed Discussion

Lecture 3: Linear Regression (Part 2)

CIS 4190/5190

Fall 2023

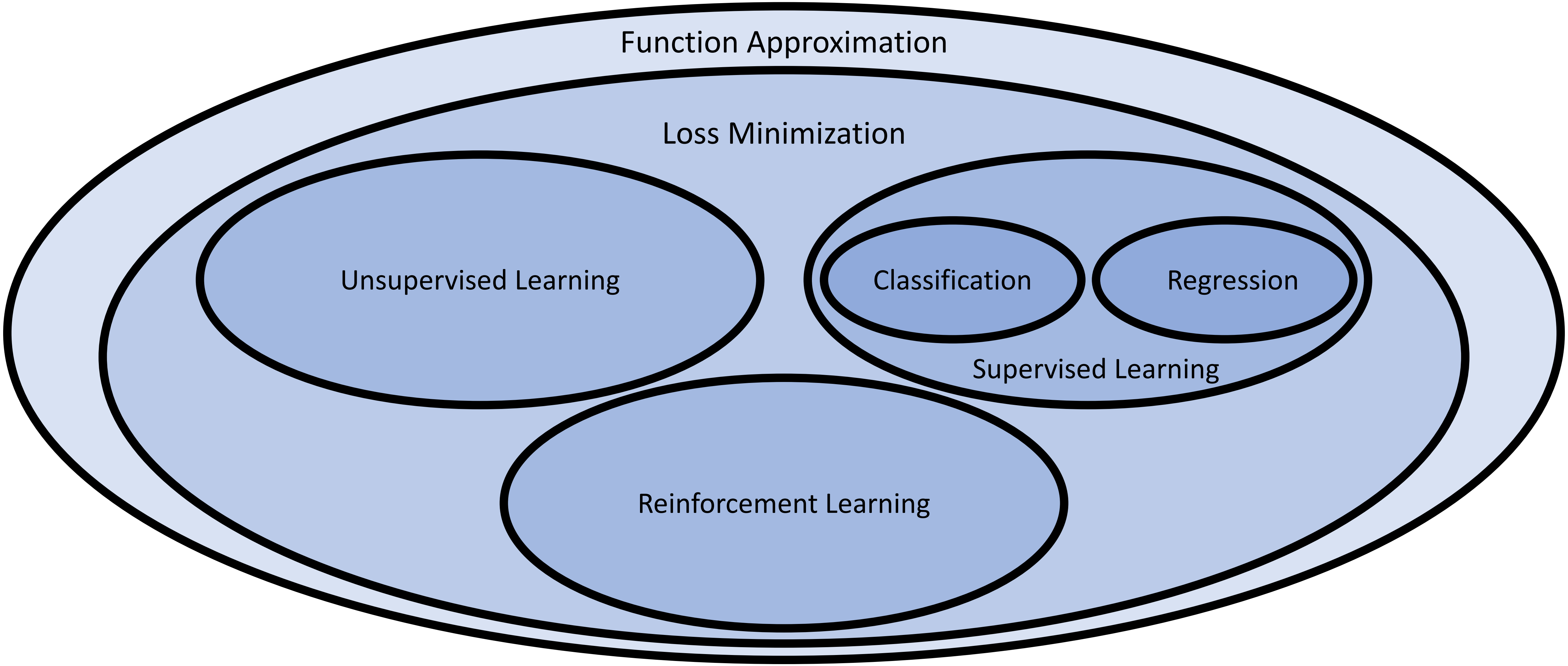
Recap: Linear Regression

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

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

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

Recap: Views of ML



Recap: Loss Minimization View of ML

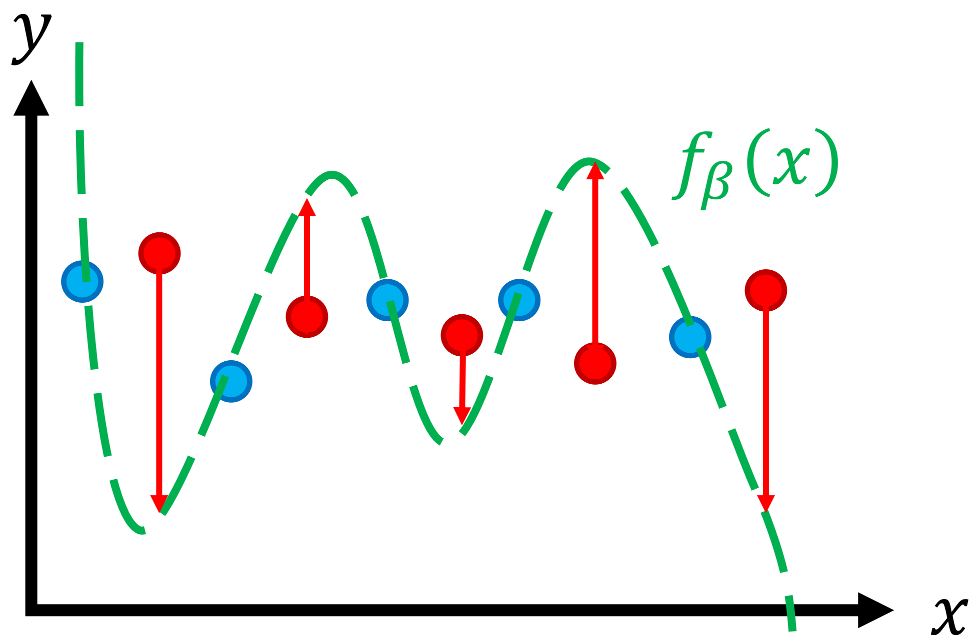
- **To design an ML algorithm:**
  - Choose model family $F = \{f_\beta\}_\beta$ (e.g., linear functions)
  - Choose loss function $L(\beta; Z)$ (e.g., MSE loss)

- **Resulting algorithm:**

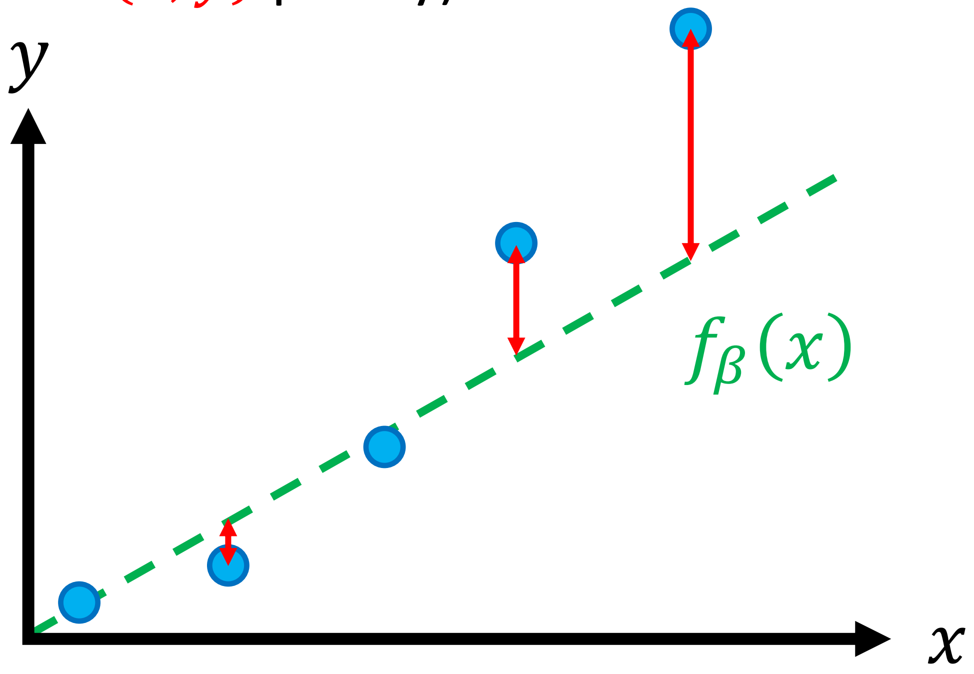
$$\hat{\beta}(Z) = \arg\min_{\beta} L(\beta; Z)$$

Recap: Overfitting vs. Underfitting

- **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)



Recap: Training/Test Split Algorithm

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

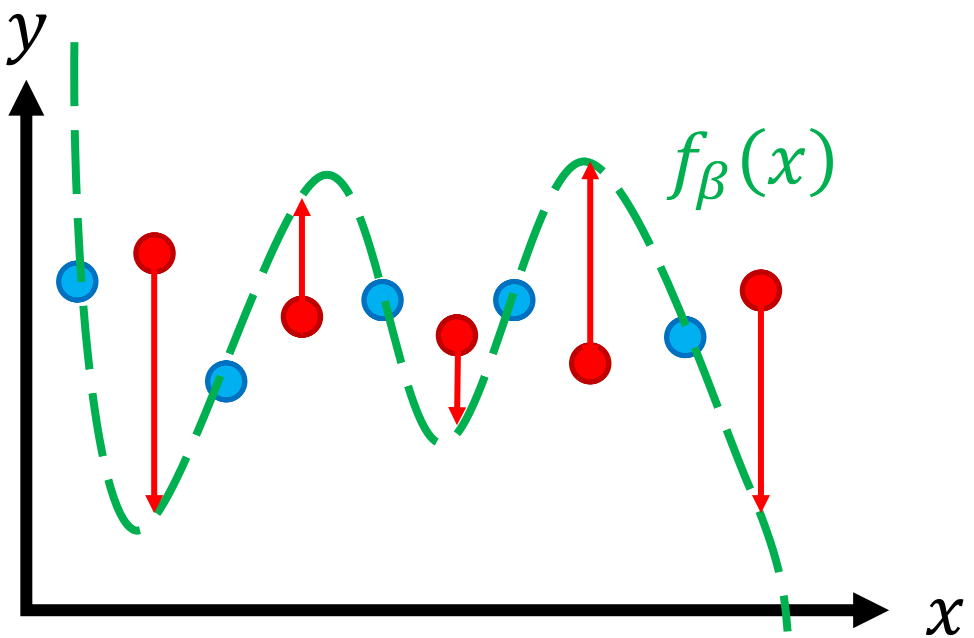
Training data Z_{train}	Test data Z_{test}

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

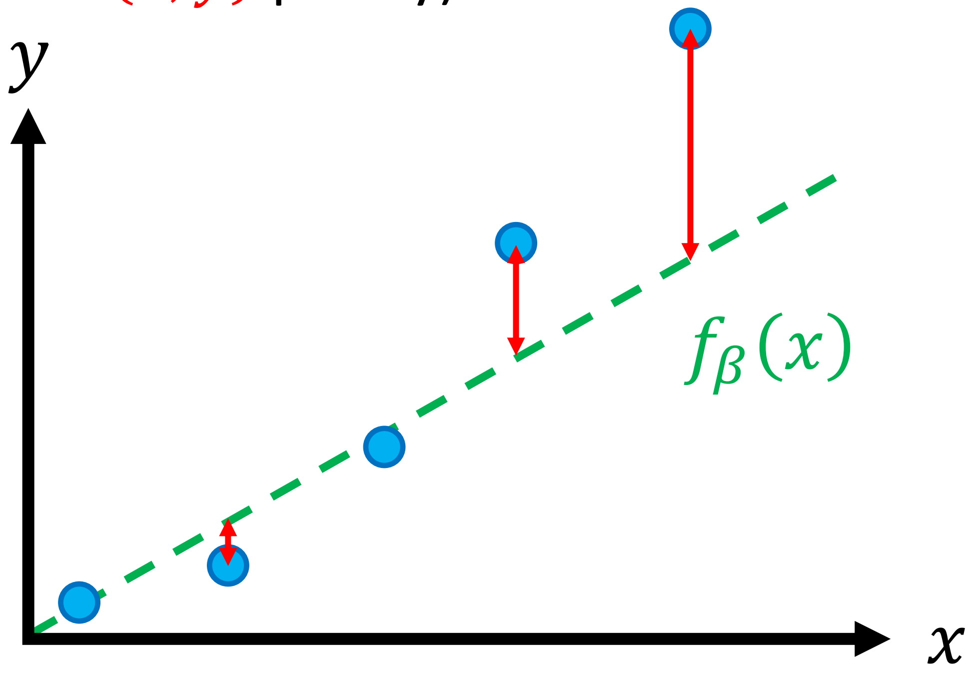
- **Step 3:** Evaluate
  - **Training loss:** $L_{\text{train}} = L\left(\hat{\beta}(Z_{\text{train}}); Z_{\text{train}}\right)$
  - **Test (or generalization) loss:** $L_{\text{test}} = L\left(\hat{\beta}(Z_{\text{train}}); Z_{\text{test}}\right)$

Recap: Training/Test Split Algorithm

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

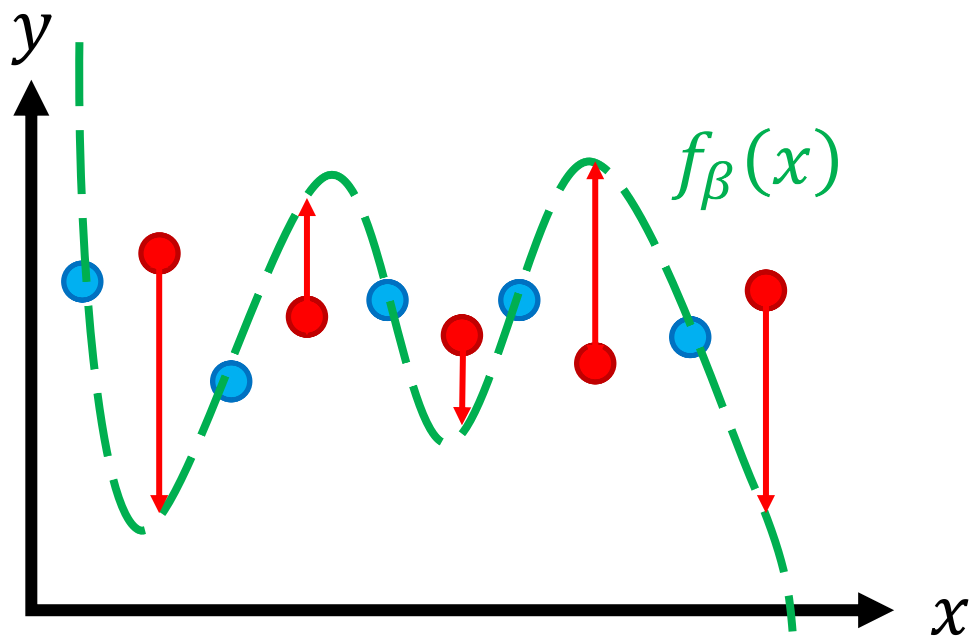


- **Underfitting**
  - Fit the **training data** Z poorly
  - (Necessarily fit new **test data** (x, y) poorly)

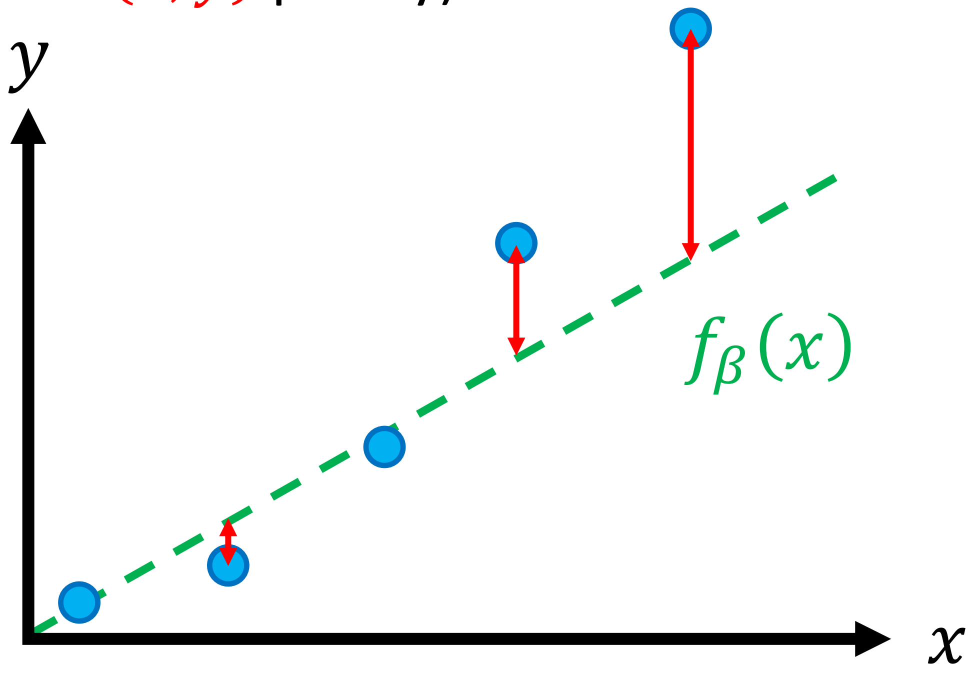


Recap: Training/Test Split Algorithm

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

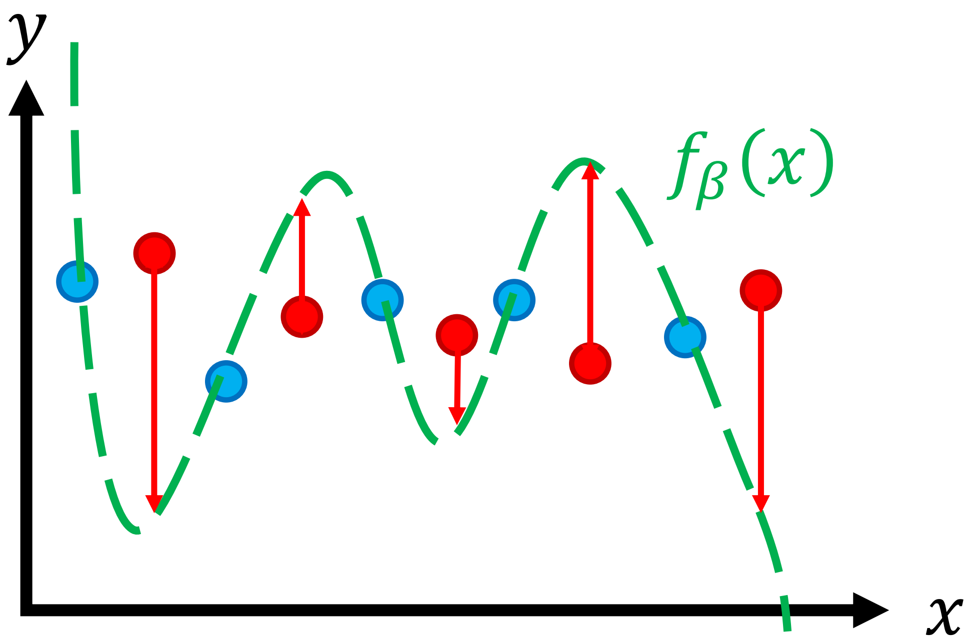


- **Underfitting**
  - Fit the **training data** Z poorly
  - (Necessarily fit new **test data** (x, y) poorly)

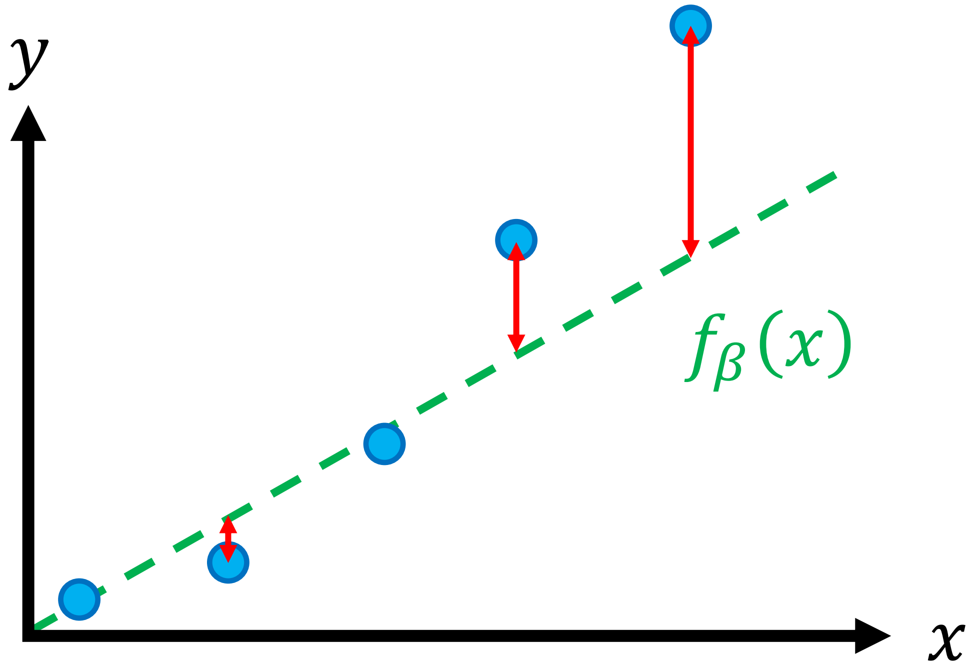


Recap: Training/Test Split Algorithm

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



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



How to Fix Underfitting/Overfitting?

- Choose the right model family!

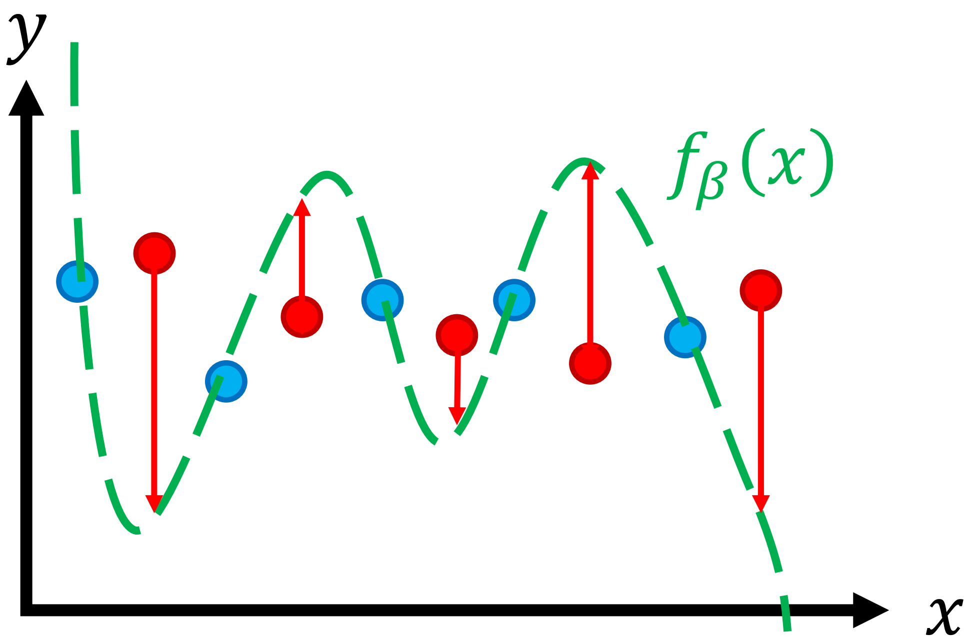
Role of Capacity

- **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**)

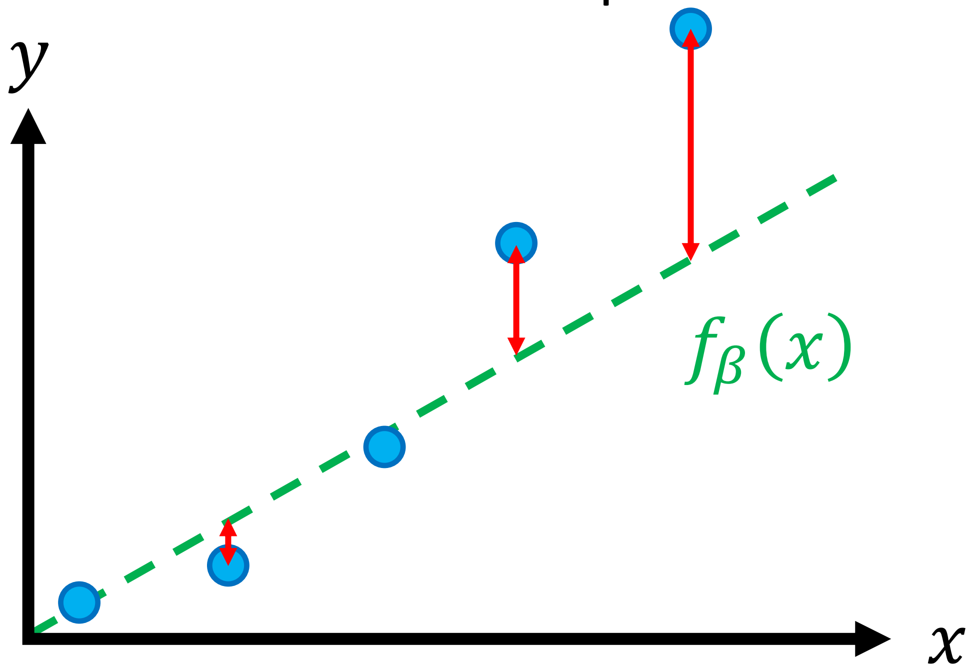
- For linear regression, capacity roughly corresponds to feature dimension d
  - I.e., number of features in $\phi(x)$

Bias-Variance Tradeoff

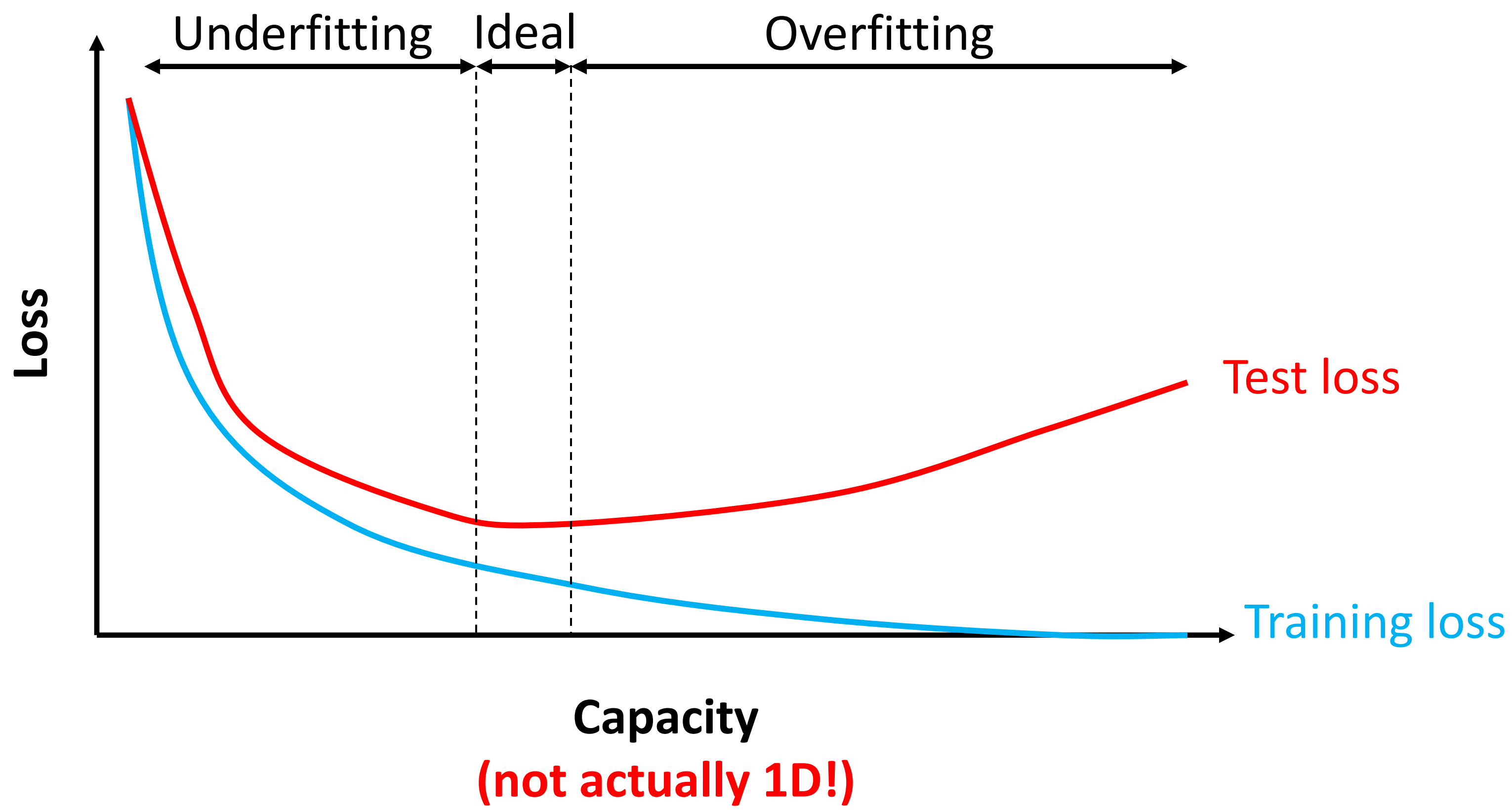
- **Overfitting (high variance)**
  - High capacity model capable of fitting complex data
  - Insufficient data to constrain it



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



Bias-Variance Tradeoff



Slide by Padhraic Smyth, UCIrvine

Bias-Variance Tradeoff

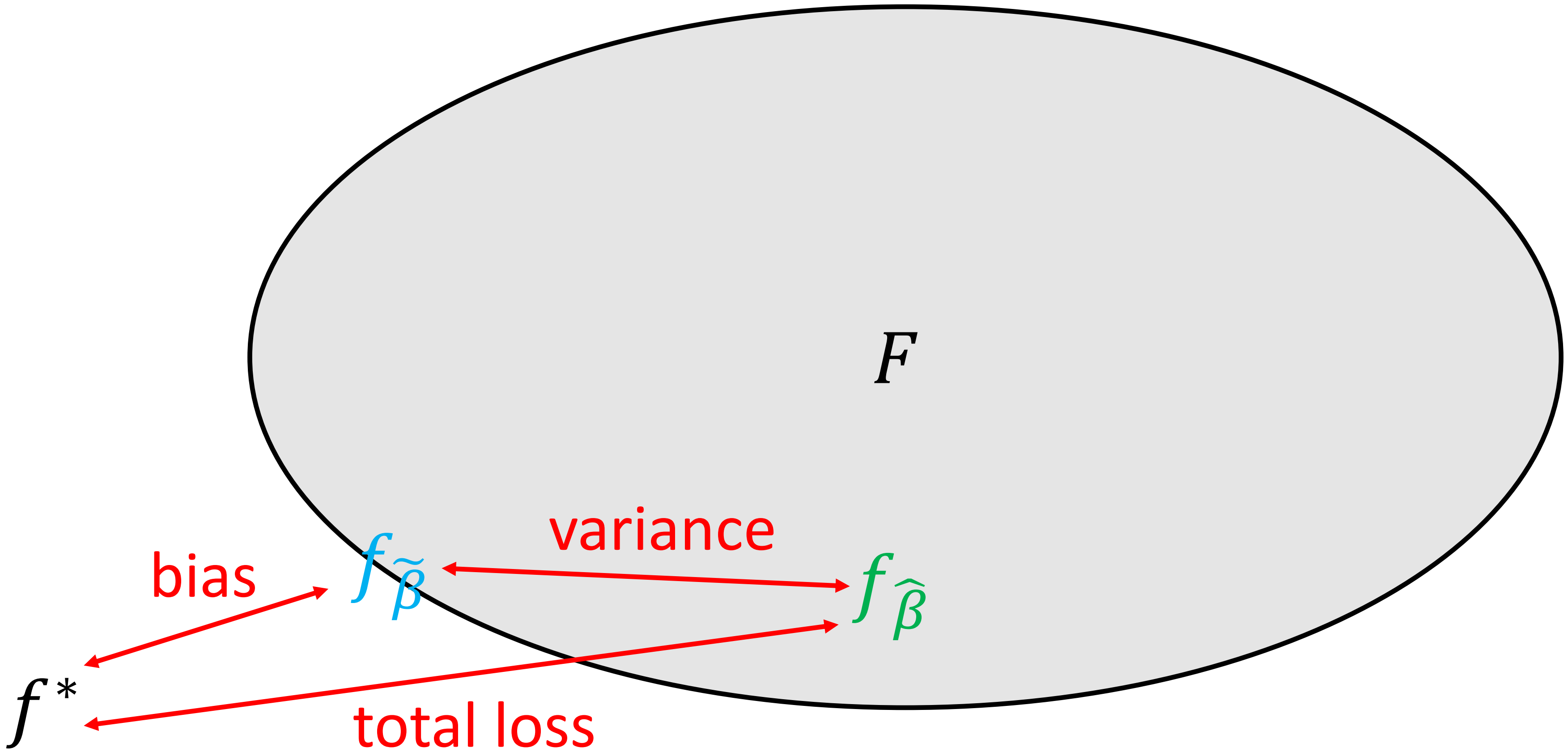
- For linear regression, increasing feature dimension d…
  - Tends to **increase capacity**
  - Tends to **decrease bias** but **increase variance**
- Need to construct ϕ to balance tradeoff between bias and variance
  - **Rule of thumb:** $n \approx d \log d$
  - Large fraction of data science work is data cleaning + feature engineering

Bias-Variance Tradeoff

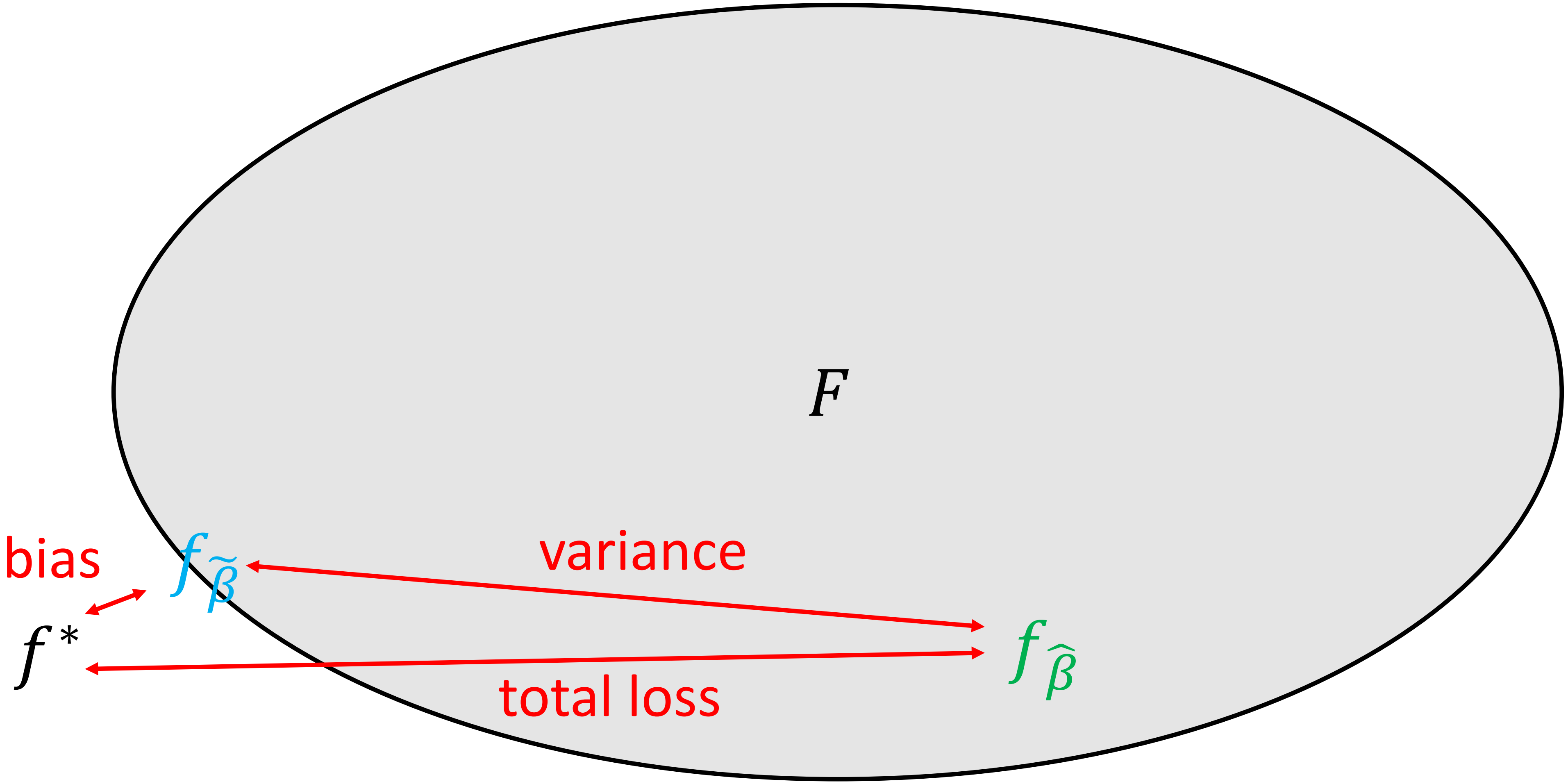
- Increasing number of examples n in the data…
  - Tends to **keep bias fixed** and **decrease variance**

- **General strategy**
  - **High bias:** Increase model capacity d
  - **High variance:** Increase data size n (i.e., gather more labeled data)

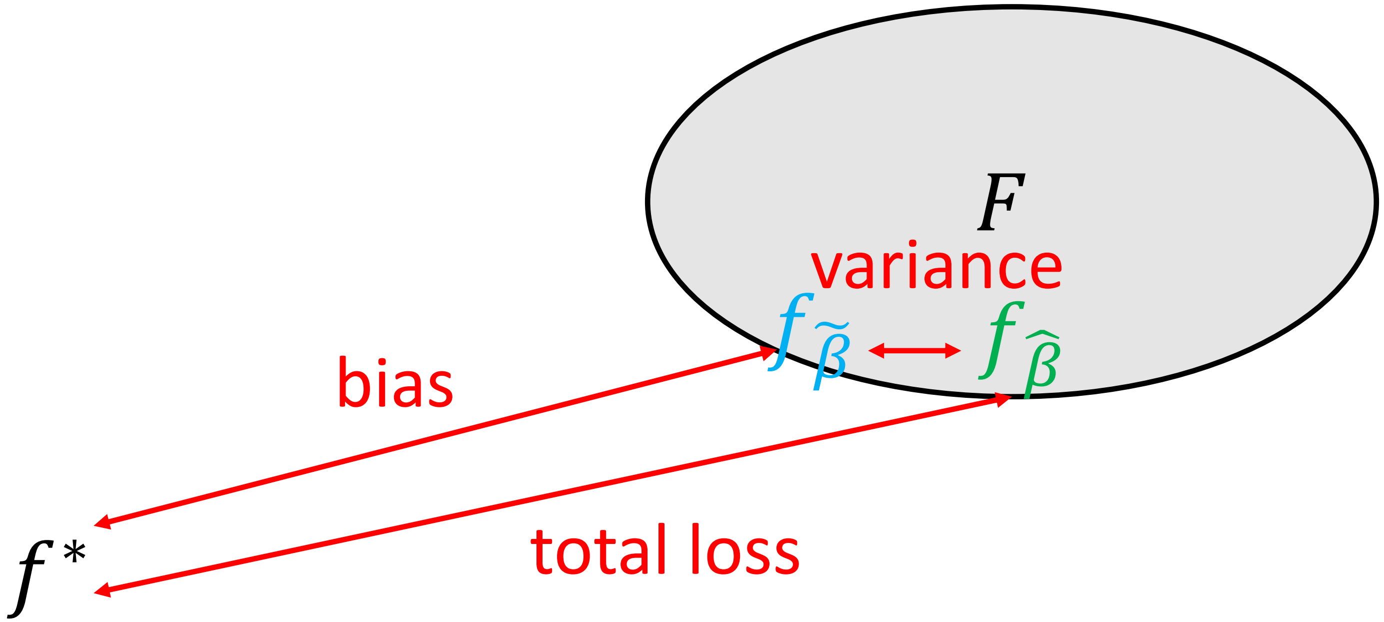
Bias-Variance Tradeoff



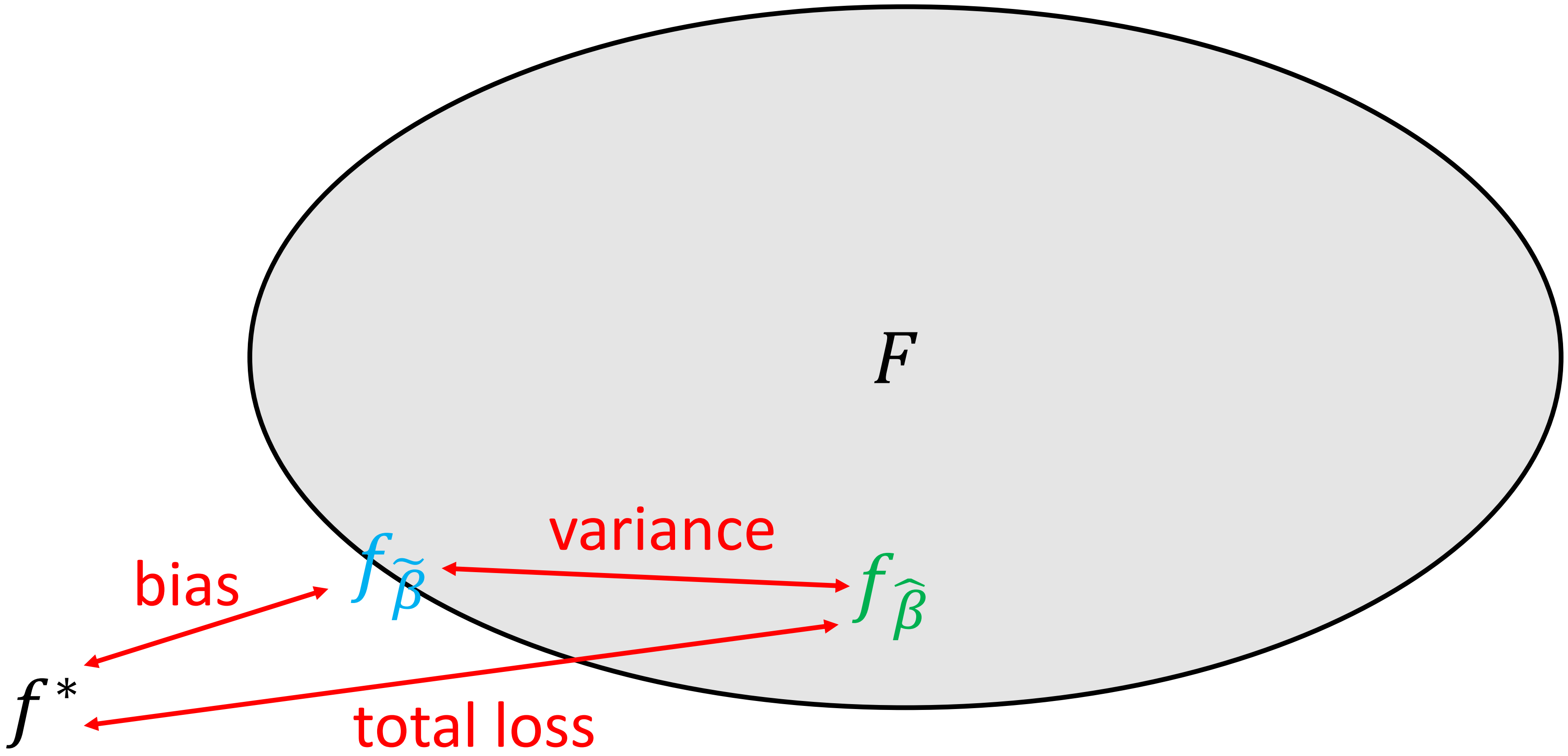
Bias-Variance Tradeoff (Overfitting)



Bias-Variance Tradeoff (Underfitting)



Bias-Variance Tradeoff (Ideal)



Agenda

- **Regularization**
  - Strategy to address bias-variance tradeoff
  - **By example:** Linear regression with L_2 regularization

- **Minimizing the MSE Loss**
  - Closed-form solution
  - Gradient descent

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:**

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

- $\lambda \in \mathbb{R}$ is a **hyperparameter** that must be tuned (satisfies $\lambda \geq 0$)

Intuition on $\boldsymbol{L_2}$ Regularization

- Equivalently the L_2 norm of β:

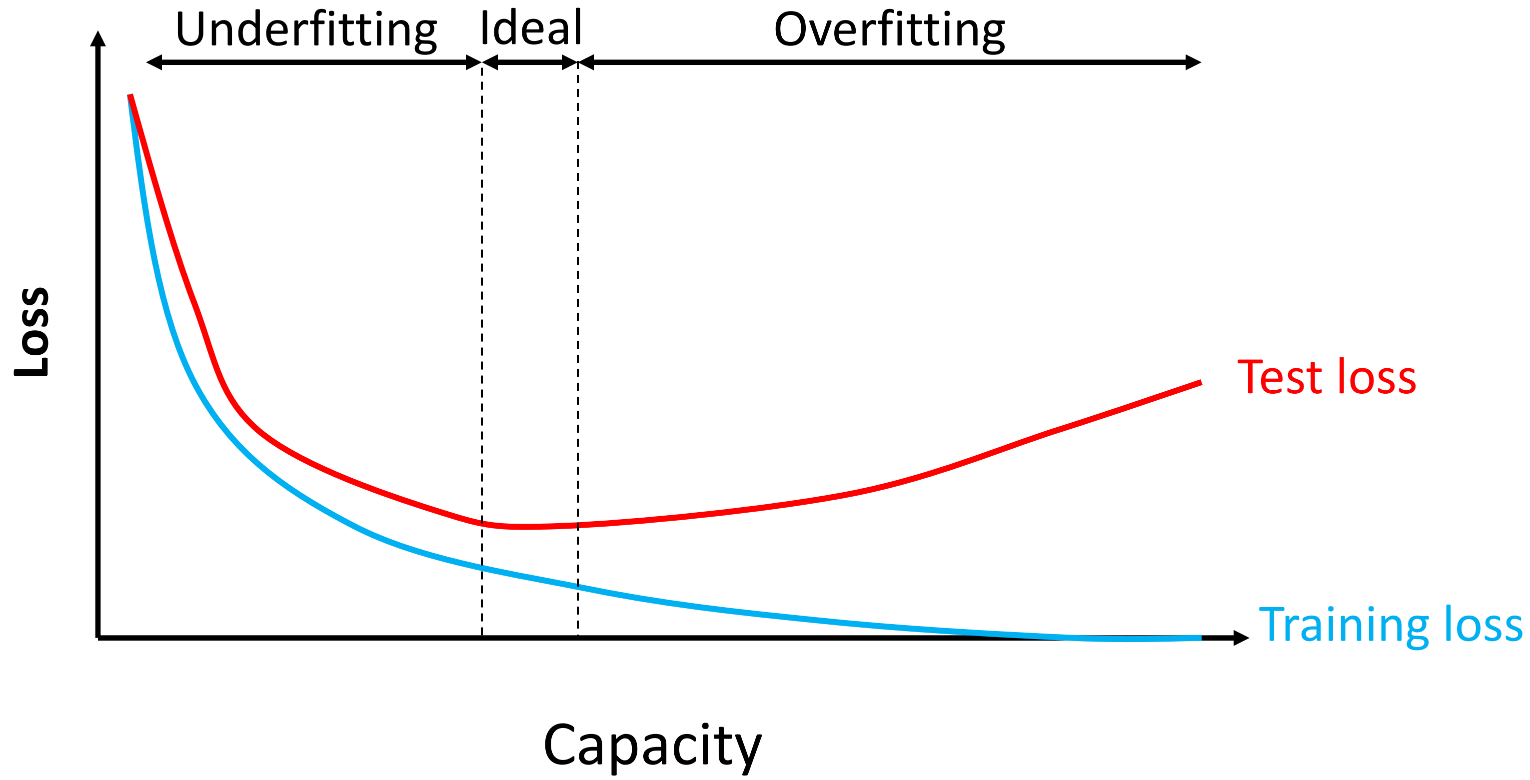
$$\sum_{j=1}^{d} \beta_j^2 = \|\beta\|_2^2 = \|\beta - 0\|_2^2$$

- I.e., “pulling” β to zero
  - “Pulls” more as λ becomes larger

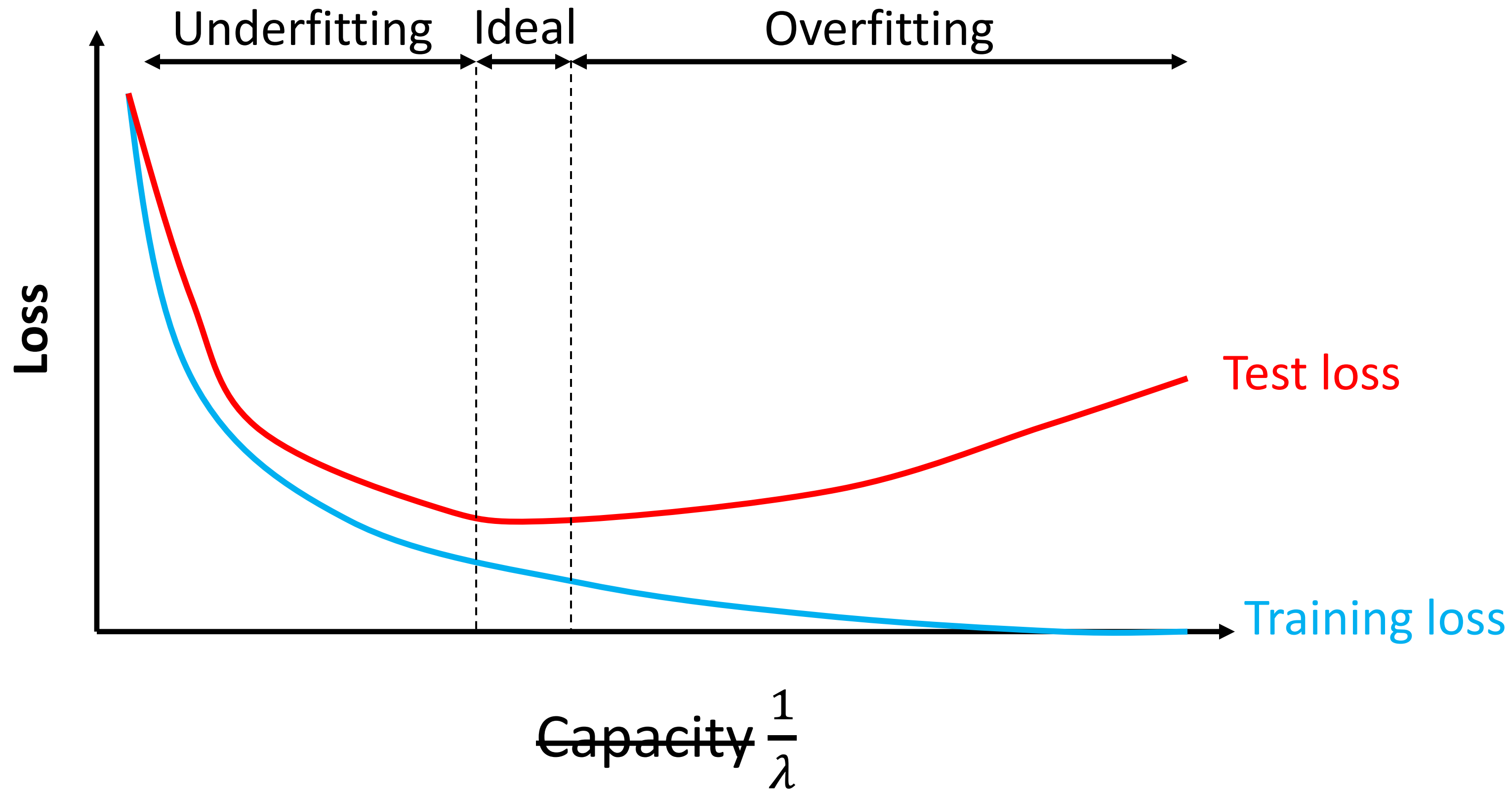
Intuition on $\boldsymbol{L_2}$ Regularization

- **Why does it help?**
  - Encourages “simple” functions
  - E.g., as $\lambda \to \infty$, obtain $\beta = 0$
  - Use λ to tune bias-variance tradeoff

Bias-Variance Tradeoff for Regularization



Bias-Variance Tradeoff for Regularization



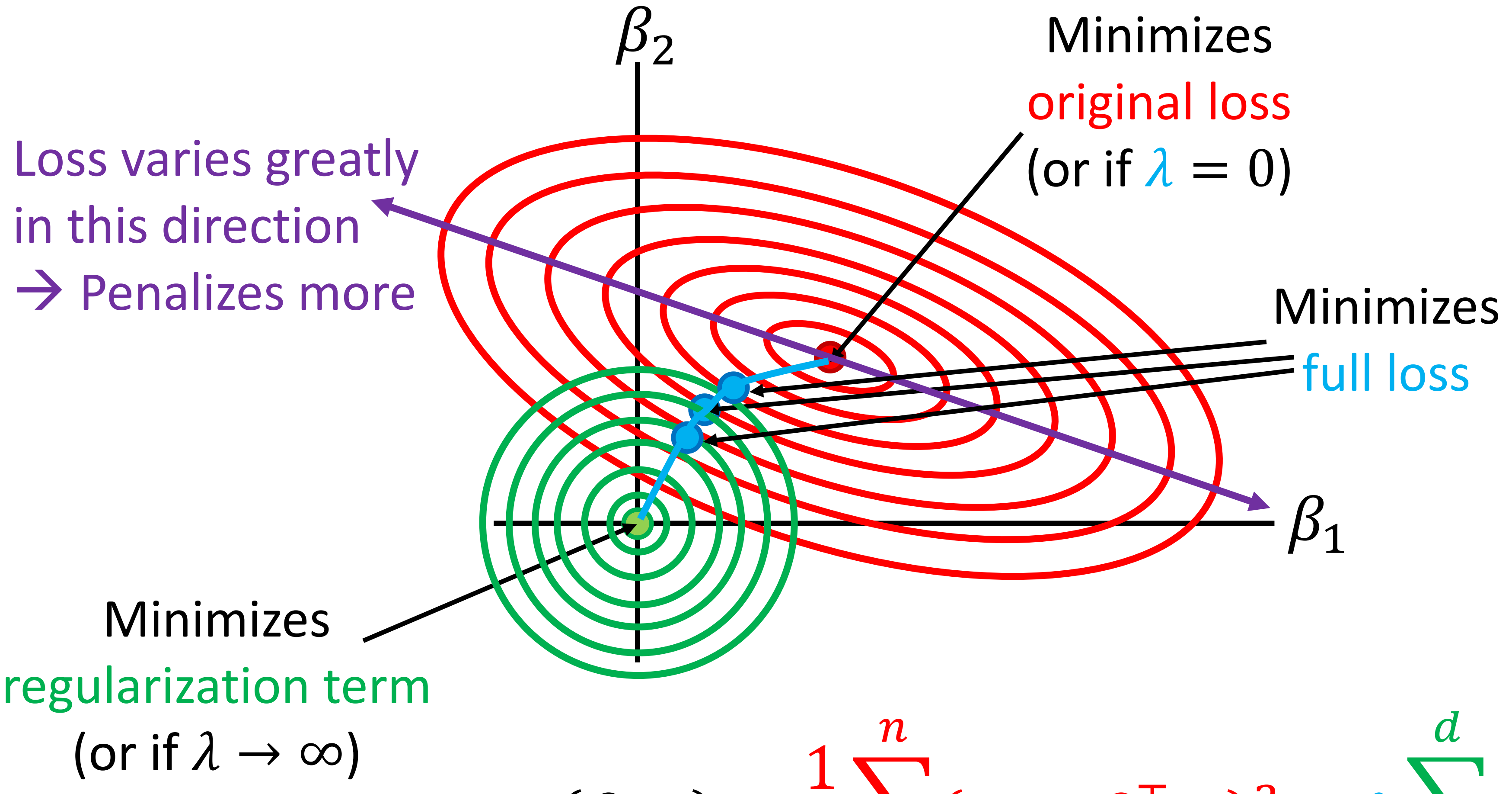
Intuition on $\boldsymbol{L_2}$ Regularization

- **More precisely:** Restricts directions of β with little variation in data
  - Little variation in data → highly varying loss

- **Example:**
  - Suppose that $x_{ij} = 0.36$ for all training examples x_i
  - Then, we cannot learn what would happen if $x_j = 1.29$ (for a new input x)
  - I.e., hard to estimate β_j

- How does L_2 regularization help?

Intuition on $\boldsymbol{L_2}$ Regularization



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

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

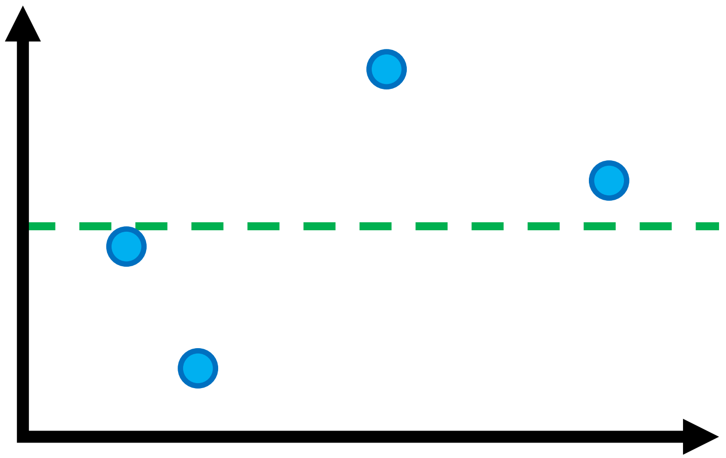
Aside: Regularization and Intercept Term

- If using intercept term ($\phi(x) = [1 \quad x_1 \quad \cdots \quad x_d]^\top$), no penalty on 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 $\hat{\beta}_1(Z) = \mathrm{mean}(\{y_i\}_{i=1}^n)$)



Aside: Feature Standardization

- **Unregularized linear regression is invariant to feature scaling**
  - Suppose we scale $x_{ij} \leftarrow 2x_{ij}$ for all examples x_i
  - Without regularization, simply use $\beta_j \leftarrow \beta_j/2$ to obtain equivalent solution
  - In particular, $\frac{\beta_j}{2} \cdot 2x_{ij} = \beta_j \cdot x_{ij}$
- Not true for regularized regression!
  - Penalty $\left(\beta_j/2\right)^2$ is scaled by 1/4 (not cancelled out!)

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

Aside: Feature Standardization

- **Unregularized linear regression is invariant to feature scaling**
  - Suppose we scale $x_{ij} \leftarrow 2x_{ij}$ for all examples x_i
  - Without regularization, simply use $\beta_j \leftarrow \beta_j/2$ to obtain equivalent solution
  - In particular, $\frac{\beta_j}{2} \cdot 2x_{ij} = \beta_j \cdot x_{ij}$
- Not true for regularized regression!
  - Penalty $(\beta_j/2)^2$ is scaled by 1/4 (not cancelled out!)

$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda(\beta_2^2 + \cdots + \beta_j^2 + \cdots + \beta_d^2)$$

Feature Standardization

- **Unregularized linear regression is invariant to feature scaling**
  - Suppose we scale $x_{ij} \leftarrow 2x_{ij}$ for all examples x_i
  - Without regularization, simply use $\beta_j \leftarrow \beta_j/2$ to obtain equivalent solution
  - In particular, $\sum_{j=1}^{d} \frac{\beta_j}{2} \cdot 2x_{ij} = \sum_{j=1}^{d} \beta_j \cdot x_{ij}$

- Not true for regularized regression!
  - Penalty $\left(\beta_j/2\right)^2$ is scaled by 1/4 (not cancelled out!)

$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda\left(\beta_2^2 + \cdots + \frac{\beta_j^2}{4} + \cdots + \beta_d^2\right)$$

Feature Standardization

- **Solution:** Rescale features to zero mean and unit variance

$$x_{i,j} \leftarrow \frac{x_{i,j} - \mu_j}{\sigma_j} \qquad \mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j} \qquad \sigma_j = \frac{1}{N}\sum_{i=1}^{N}\left(x_{i,j} - \mu_j\right)^2$$

  - **Note:** When using intercept term, do not rescale $x_1 = 1$
  - Can be sensitive to outliers (fix by dropping outliers)

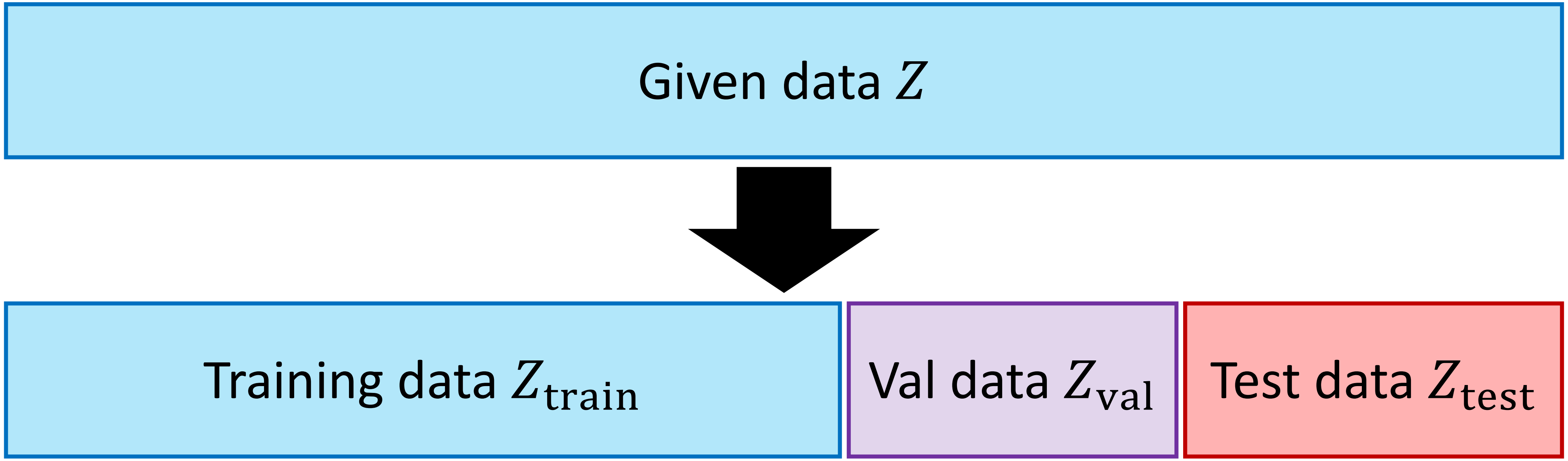
- **Must use same transformation during training and for prediction**
  - Compute μ_j and σ_j on training data and use on test data

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

Training/Val/Test Split

- **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



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 $t \in \{1, \ldots, h\}$:
  - **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 $L_{\text{val}}^t = L\left(\hat{\beta}(Z_{\text{train}}, \lambda_t); Z_{\text{val}}\right)$

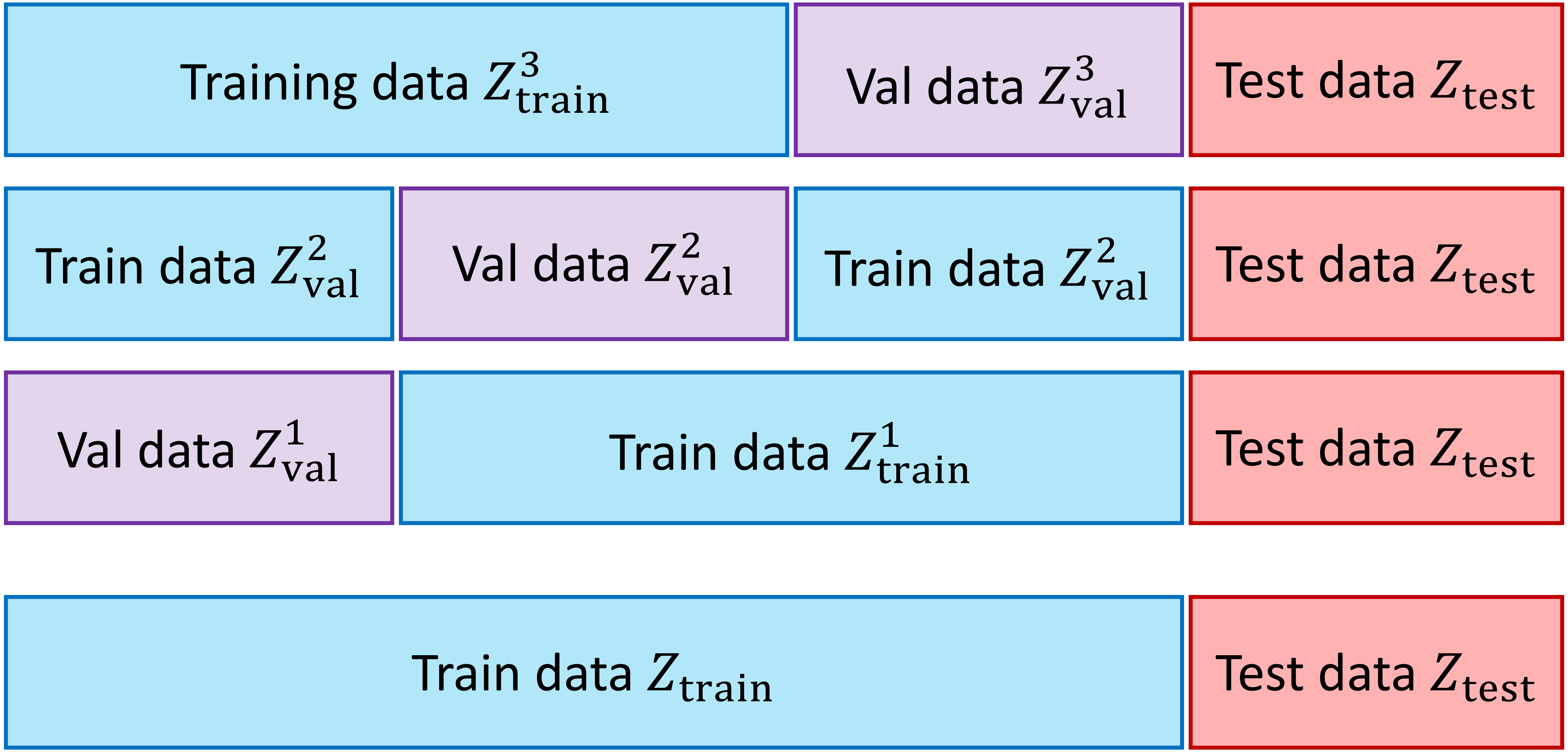
- **Step 3:** Use best λ_t
  - Choose $t' = \arg\min_t L_{\text{val}}^t$ with lowest validation loss
  - Re-run linear regression with Z_{train} and $\lambda_{t'}$ to obtain $\hat{\beta}(Z_{\text{train}}, \lambda_{t'})$

Alternative Cross-Validation Algorithms

- If Z is small, then splitting it can reduce performance
  - **Solution:** Can use $Z_{\text{train}} \cup Z_{\text{val}}$ in Step 3

- **Alternative solution:** k-fold cross-validation (e.g., $k = 3$)
  - 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 $s \in \{1, \dots, k\}$ with Z_{train}
  - Chooses better λ' than above strategy

Example: 3-Fold Cross Validation



k-Fold Cross-Validation

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

General Regularization Strategy

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

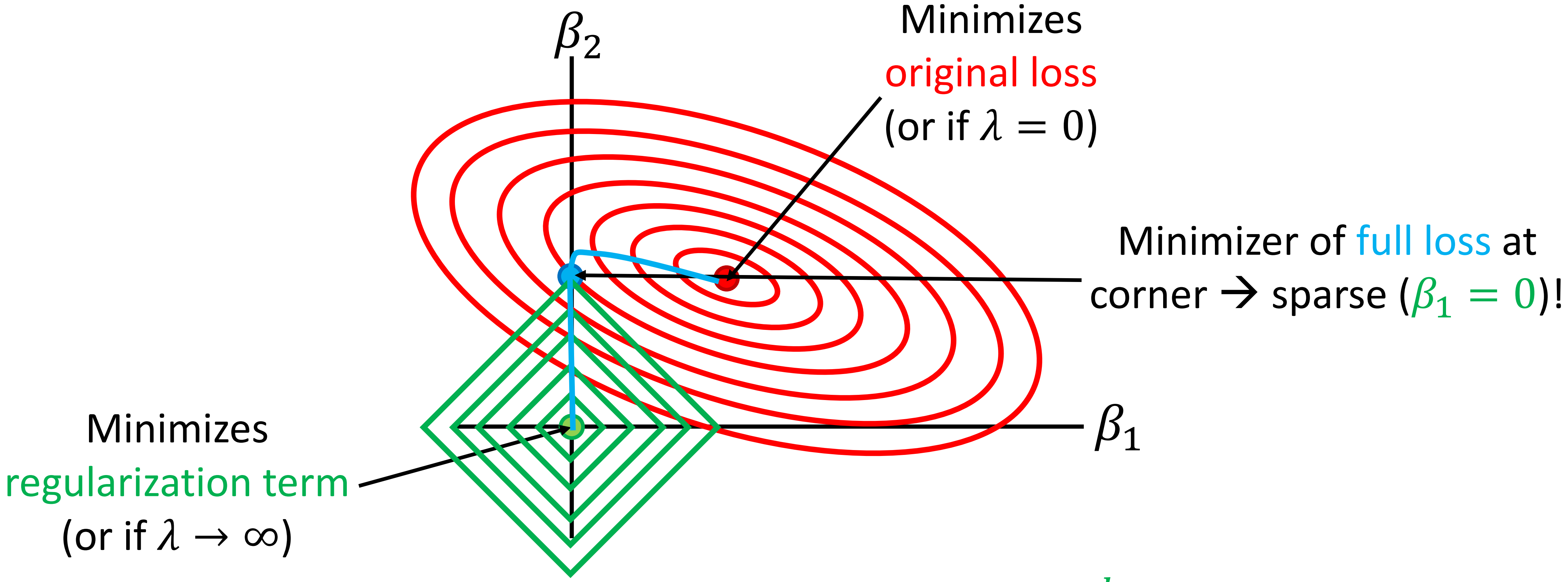
$$L_{\text{new}}(\beta; Z) = L(\beta; Z) + \lambda \cdot R(\beta)$$

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

L_1 Regularization

- **Sparsity:** Can we minimize $\|\beta\|_0 = \left|\{ j \mid \beta_j \neq 0 \}\right|$?
  - That is, the number of nonzero components of β
  - Improves interpretability (automatic **feature selection**!)
  - Also serves as a **strong** regularizer ($n \sim s \log d$, where $s = \|\beta\|_0$)
- **Challenge:** $\|\beta\|_0$ is not differentiable, making it hard to optimize
- **Solution**
  - We can instead use an L_1 norm as the regularizer!
  - Still harder to optimize than L_2 norm, but at least it is convex

Intuition on $\boldsymbol{L_1}$ Regularization



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

$\boldsymbol{L_1}$ Regularization for Feature Selection

- **Step 1:** Construct a lot of features and add to feature map
- **Step 2:** Use L_1 regularized regression to "select" subset of features
  - I.e., coefficient $\beta_j \neq 0$ → feature j is selected)
- **Optional:** Remove unselected features from the feature map and run vanilla linear regression (a.k.a. ordinary least squares)

Housing Dataset

- Sales of residential property in Ames, Iowa from 2006 to 2010
  - **Examples:** 1,022
  - **Features:** 79 total (real-valued + categorical), some are missing!
  - **Label**: Sales price

MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	...	MoSold	YrSold	SaleType	SaleCondition	SalePrice
20	RL	80.0	10400	Pave	NaN	Reg	...	5	2008	WD	Normal	174000
180	RM	35.0	3675	Pave	NaN	Reg	...	5	2006	WD	Normal	145000
60	FV	72.0	8640	Pave	NaN	Reg	...	6	2010	Con	Normal	215200
20	RL	84.0	11670	Pave	NaN	IR1	...	3	2007	WD	Normal	320000
60	RL	43.0	10667	Pave	NaN	IR2	...	4	2009	ConLw	Normal	212000
80	RL	82.0	9020	Pave	NaN	Reg	...	6	2008	WD	Normal	168500
60	RL	70.0	11218	Pave	NaN	Reg	...	5	2010	WD	Normal	189000
80	RL	85.0	13825	Pave	NaN	Reg	...	12	2008	WD	Normal	140000
60	RL	NaN	13031	Pave	NaN	IR2	...	7	2006	WD	Normal	187500

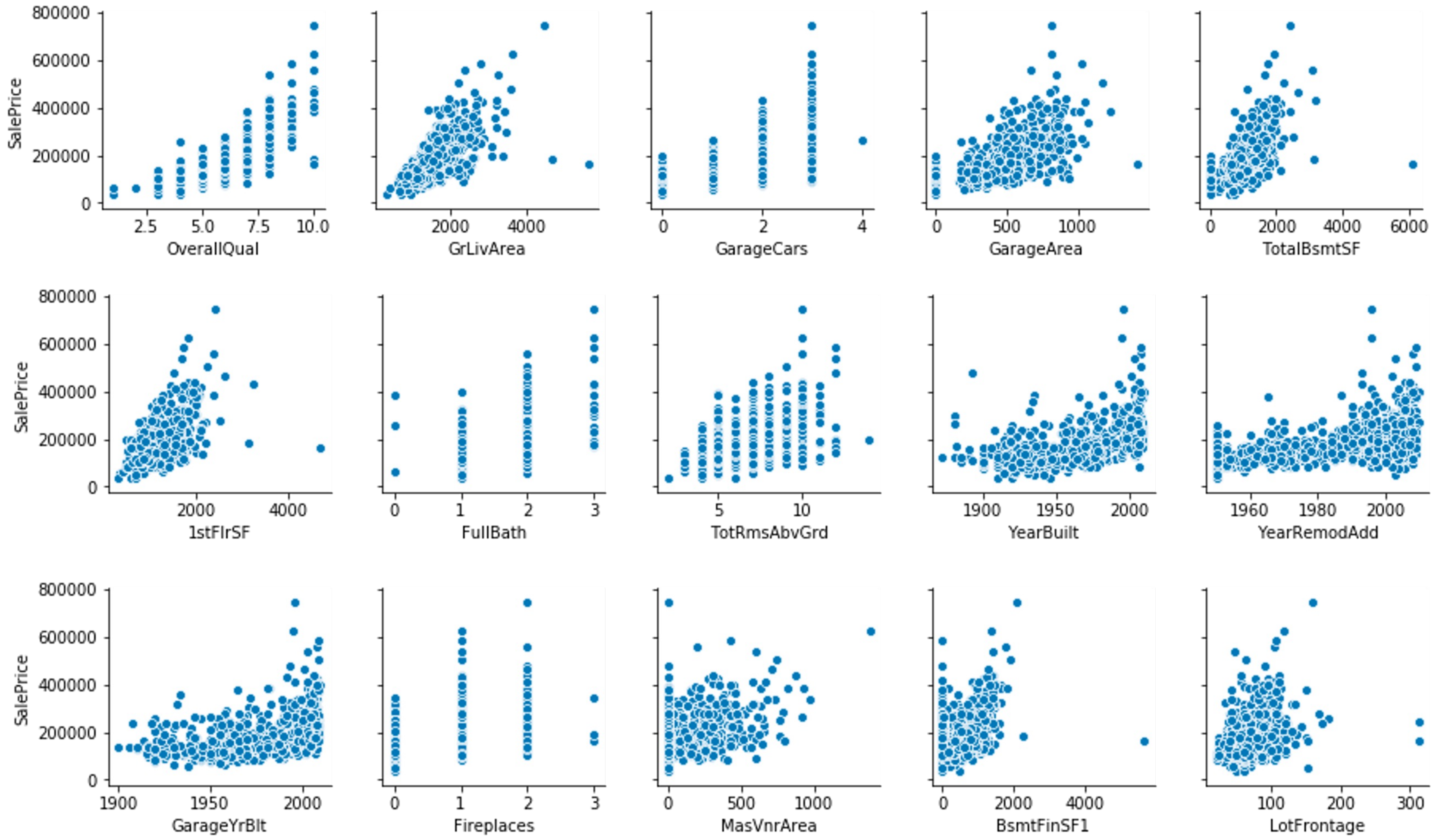
Data from: De Cock. Journal of Statistics Education 19(3), 2011

Summary Statistics

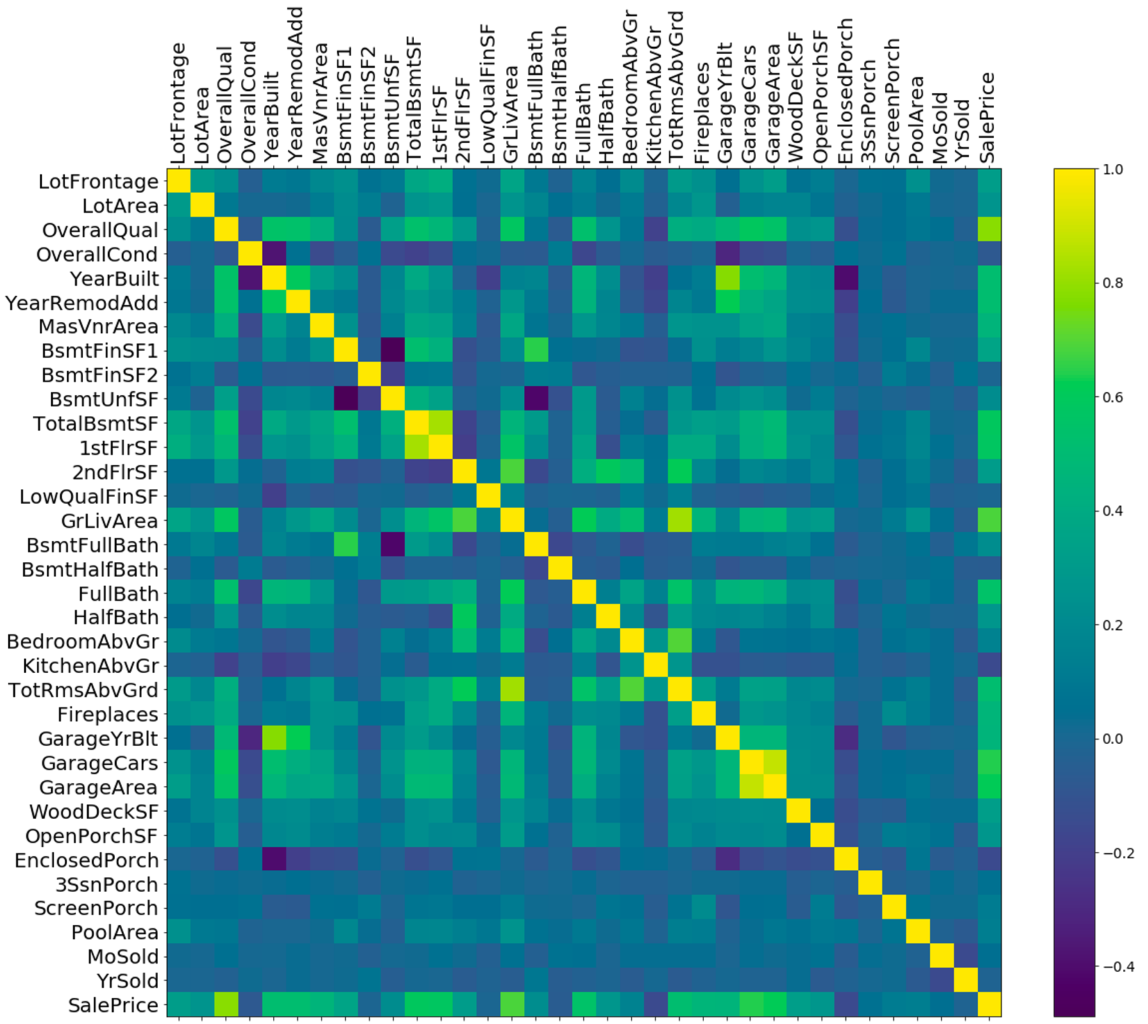
- dataframe.describe()

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	...	SalePrice
count	1022.000000	1022.000000	832.000000	1022.000000	1022.000000	1022.000000	1022.000000	1022.000000	1019.000000		1022.000000
mean	732.338552	57.059687	70.375000	10745.437378	6.128180	5.564579	1970.995108	1984.757339	105.261040		181312.692759
std	425.860402	42.669715	25.533607	11329.753423	1.371391	1.110557	30.748816	20.747109	172.707705		77617.461005
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.000000	...	34900.000000
25%	367.500000	20.000000	59.000000	7564.250000	5.000000	5.000000	1953.000000	1966.000000	0.000000		130000.000000
50%	735.500000	50.000000	70.000000	9600.000000	6.000000	5.000000	1972.000000	1994.000000	0.000000		165000.000000
75%	1100.500000	70.000000	80.000000	11692.500000	7.000000	6.000000	2001.000000	2004.000000	170.000000		215000.000000
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1378.000000		745000.000000

Features Most Correlated with Label



Feature Correlation Matrix



Missing Values

- Possible ways to handle missing values
  - **Numerical:** Impute with mean
  - **Categorical:** Impute with mode

Feature	% Missing Values
PoolQC	99.5108
MiscFeature	96.0861
Alley	93.5421
Fence	80.2348
FireplaceQu	47.6517
LotFrontage	18.5910
GarageCond	05.2838
GarageType	05.2838
GarageYrBlt	05.2838
GarageFinish	05.2838
GarageQual	05.2838
BsmtFinType1	02.5440
…	

Other Preprocessing

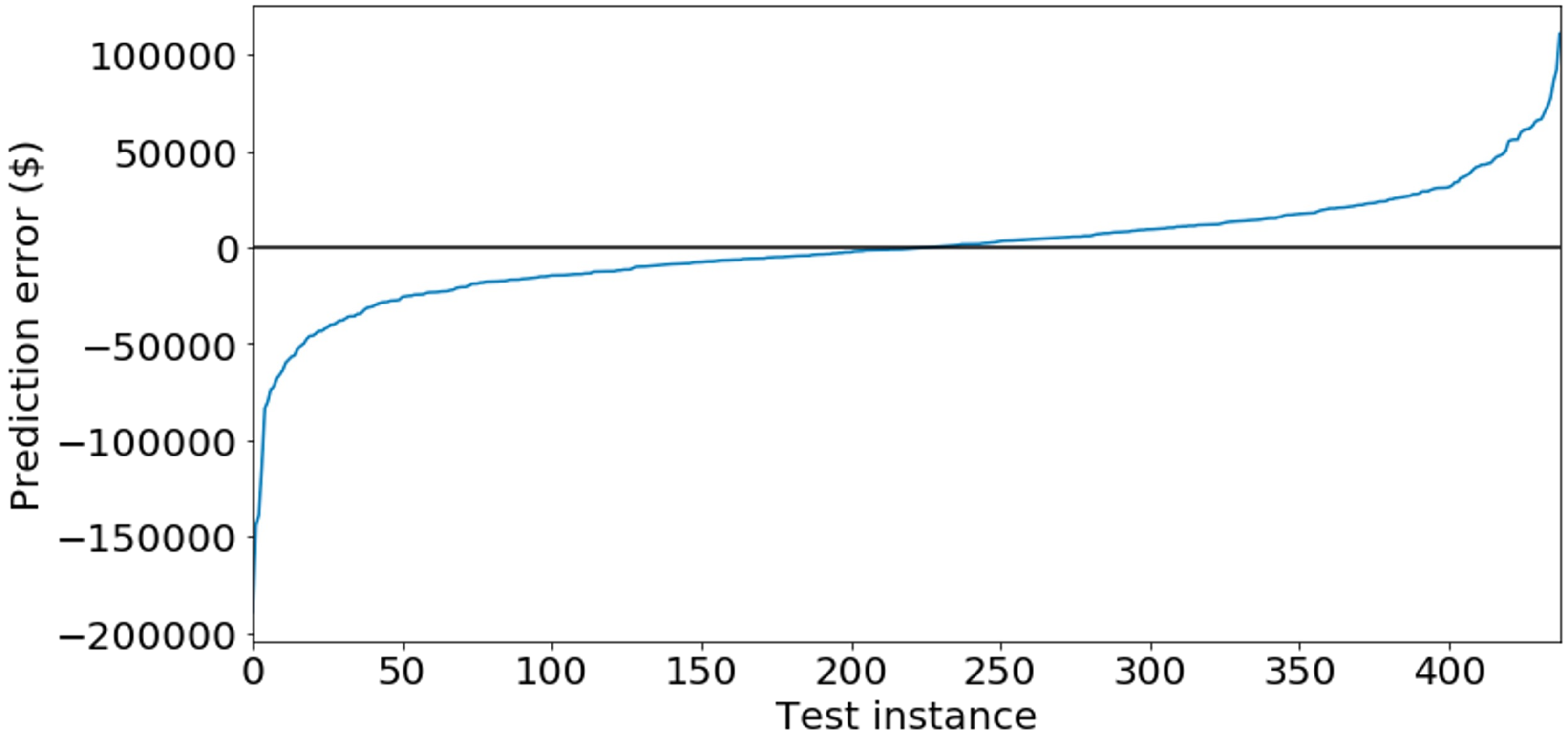
- **Categorical:** Featurize using one-hot encoding
- **Ordinal**
  - Convert to integer (e.g., low, medium, high → 1, 2, 3)
  - Does not fully capture relationships (try different featurizations!)

HouseStyle	FullBath	RoofMatl	BsmtCond	KitchenQual
1Story	2	CompShg	TA	TA
SLvl	1	CompShg	TA	TA
2Story	2	CompShg	TA	Gd
1Story	2	CompShg	Gd	Ex
2Story	2	CompShg	TA	Gd
SLvl	1	WdShngl	TA	TA
2Story	2	CompShg	TA	Gd
SLvl	1	CompShg	TA	TA
2Story	2	CompShg	TA	TA
2Story	2	CompShg	TA	Gd

HouseStyle	FullBath	RoofMatl	BsmtCond	KitchenQual
1Story	2	CompShg	3	3
SLvl	1	CompShg	3	3
2Story	2	CompShg	3	4
1Story	2	CompShg	4	5
2Story	2	CompShg	3	4
SLvl	1	WdShngl	3	3
2Story	2	CompShg	3	4
SLvl	1	CompShg	3	3
2Story	2	CompShg	3	3
2Story	2	CompShg	3	4

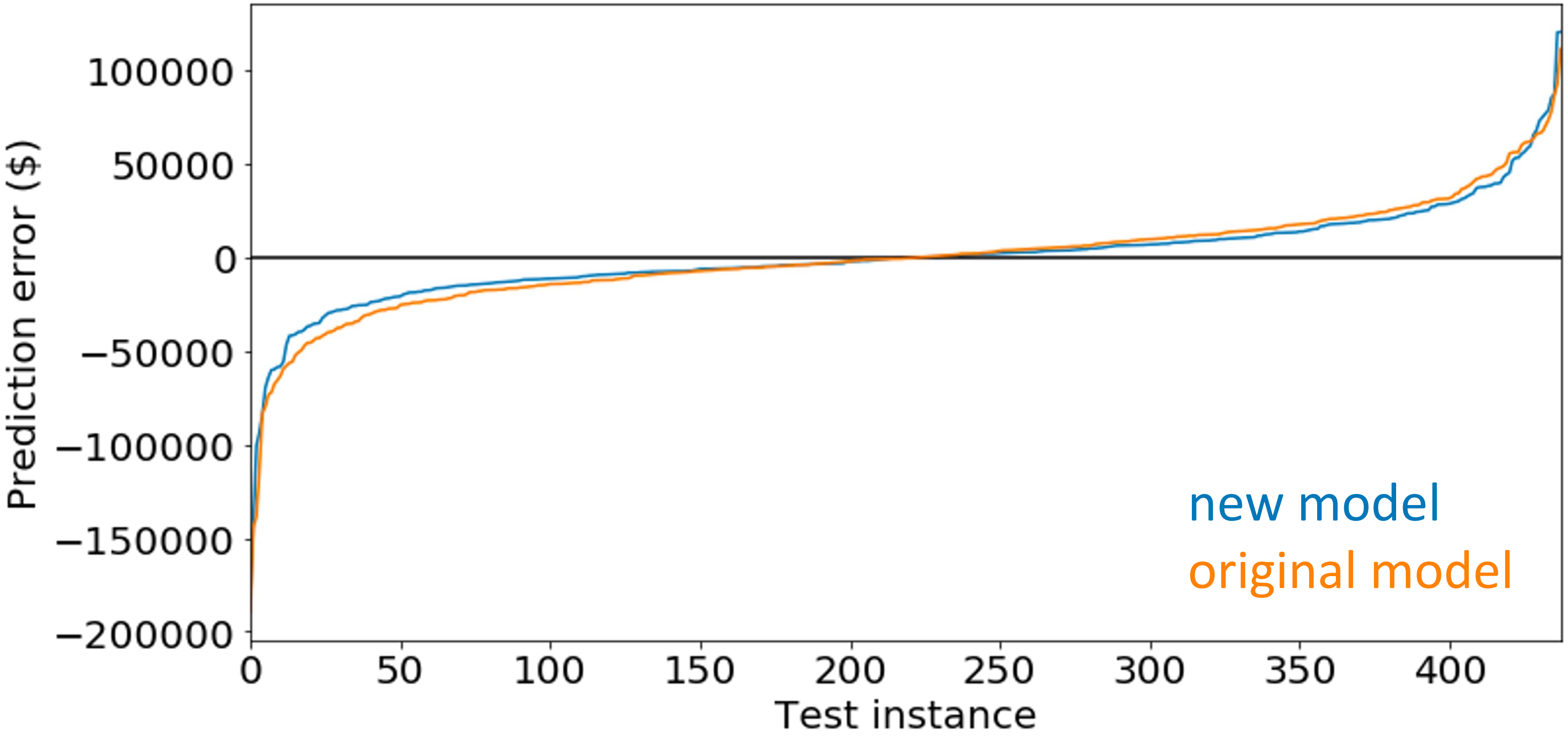
Evaluation

- 438 test examples, **preprocessed same as training data**
- Sorted by prediction error



Regularization

- Quadratic features, feature standardization, L_2 regularization



Agenda

- **Regularization**
  - Strategy to address bias-variance tradeoff
  - **By example:** Linear regression with L_2 regularization

- **Minimizing the MSE Loss**
  - Closed-form solution
  - Stochastic gradient descent

Minimizing the MSE Loss

- Recall that linear regression minimizes the loss

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

- **Closed-form solution:** Compute using matrix operations
- **Optimization-based solution:** Search over candidate β

Vectorizing Linear Regression

Vectorizing Linear Regression

$$\begin{bmatrix} f_\beta(x_1) \\ \vdots \\ f_\beta(x_n) \end{bmatrix}$$

Vectorizing Linear Regression

$$\begin{bmatrix} f_\beta(x_1) \\ \vdots \\ f_\beta(x_n) \end{bmatrix} = \begin{bmatrix} \beta^\top x_1 \\ \vdots \\ \beta^\top x_n \end{bmatrix}$$

Vectorizing Linear Regression

$$\begin{bmatrix} f_\beta(x_1) \\ \vdots \\ f_\beta(x_n) \end{bmatrix} = \begin{bmatrix} \beta^\top x_1 \\ \vdots \\ \beta^\top x_n \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{d} \beta_j x_{1,j} \\ \vdots \\ \sum_{j=1}^{d} \beta_j x_{n,j} \end{bmatrix}$$

Vectorizing Linear Regression

$$\begin{bmatrix} f_\beta(x_1) \\ \vdots \\ f_\beta(x_n) \end{bmatrix} = \begin{bmatrix} \beta^\top x_1 \\ \vdots \\ \beta^\top x_n \end{bmatrix} = \begin{bmatrix} \boxed{\sum_{j=1}^{d} \beta_j x_{1,j}} \\ \vdots \\ \sum_{j=1}^{d} \beta_j x_{n,j} \end{bmatrix} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,d} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_d \end{bmatrix}$$

Vectorizing Linear Regression

$$\begin{bmatrix} f_\beta(x_1) \\ \vdots \\ f_\beta(x_n) \end{bmatrix} = \begin{bmatrix} \beta^\top x_1 \\ \vdots \\ \beta^\top x_n \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{d} \beta_j x_{1,j} \\ \vdots \\ \boxed{\sum_{j=1}^{d} \beta_j x_{n,j}} \end{bmatrix} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,d} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_d \end{bmatrix}$$

Vectorizing Linear Regression

$$\begin{bmatrix} f_\beta(x_1) \\ \vdots \\ f_\beta(x_n) \end{bmatrix} = \begin{bmatrix} \beta^\top x_1 \\ \vdots \\ \beta^\top x_n \end{bmatrix} = \begin{bmatrix} \displaystyle\sum_{j=1}^{d} \beta_j x_{1,j} \\ \vdots \\ \displaystyle\sum_{j=1}^{d} \beta_j x_{n,j} \end{bmatrix} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,d} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_d \end{bmatrix} = X\beta$$

Vectorizing Linear Regression

$$\begin{bmatrix} f_\beta(x_1) \\ \vdots \\ f_\beta(x_n) \end{bmatrix} = \begin{bmatrix} \beta^\top x_1 \\ \vdots \\ \beta^\top x_n \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{d} \beta_j x_{1,j} \\ \vdots \\ \sum_{j=1}^{d} \beta_j x_{n,j} \end{bmatrix} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,d} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_d \end{bmatrix} = X\beta$$

$$\begin{matrix} \begin{bmatrix} f_\beta(x_1) \\ \vdots \\ f_\beta(x_n) \end{bmatrix} \\ \wr\wr \\ \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \end{matrix}$$

Vectorizing Linear Regression

$$\begin{bmatrix} f_\beta(x_1) \\ \vdots \\ f_\beta(x_n) \end{bmatrix} = \begin{bmatrix} \beta^\top x_1 \\ \vdots \\ \beta^\top x_n \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{d} \beta_j x_{1,j} \\ \vdots \\ \sum_{j=1}^{d} \beta_j x_{n,j} \end{bmatrix} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,d} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_d \end{bmatrix} = X\beta$$

$$\wr\wr$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = Y$$

Summary: $Y \approx X\beta$

Vectorizing Linear Regression

$$Y \approx X\beta$$

$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,d} \end{bmatrix} \qquad \beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_d \end{bmatrix}$$

Vectorizing Mean Squared Error

Vectorizing Mean Squared Error

$$L(\beta; Z)$$

Vectorizing Mean Squared Error

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

Vectorizing Mean Squared Error

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \begin{bmatrix} f_\beta(x_1) \\ \vdots \\ f_\beta(x_n) \end{bmatrix}$$

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

$$\|z\|_2^2 = \sum_{i=1}^{n} z_i^2$$

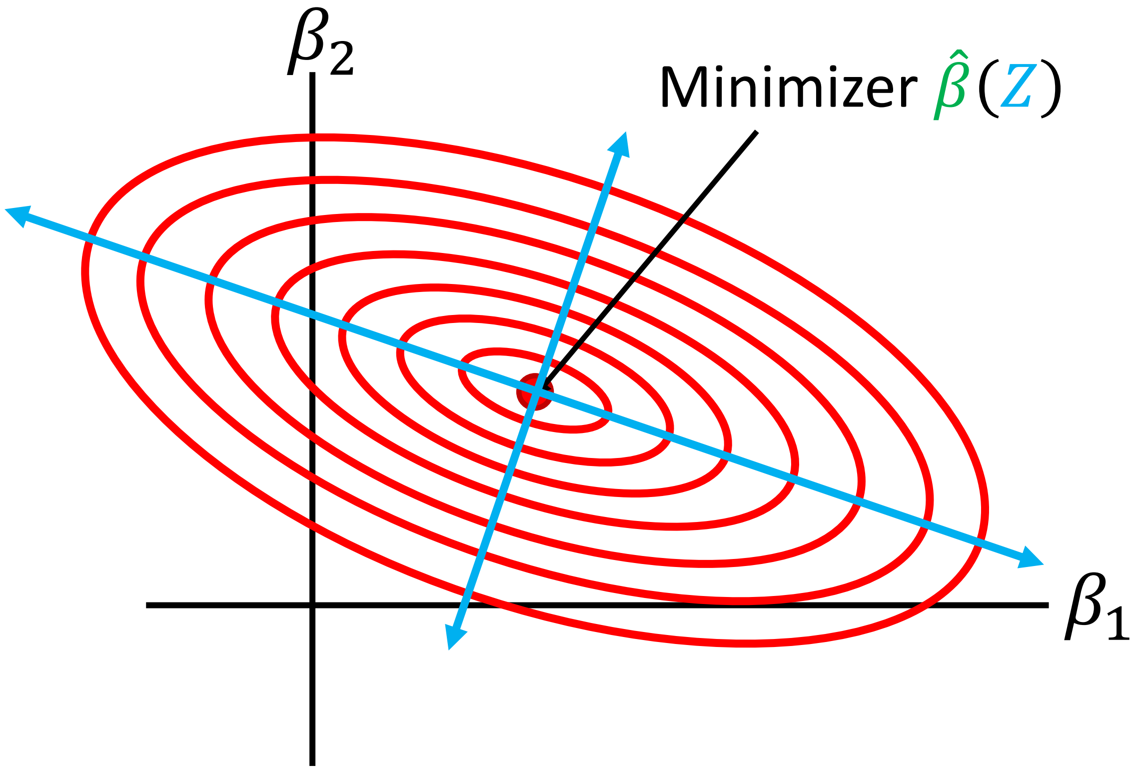
Intuition on Vectorized Linear Regression

- Rewriting the vectorized loss:

$$
\begin{aligned}
n \cdot L(\beta; Z) = \|Y - X\beta\|_2^2 &= \|Y\|_2^2 - 2Y^\top X\beta + \|X\beta\|_2^2 \\
&= \|Y\|_2^2 - 2Y^\top X\beta + \beta^\top (X^\top X)\beta
\end{aligned}
$$

- Quadratic function of β with leading "coefficient" $X^\top X$
  - In one dimension, "width" of parabola $ax^2 + bx + c$ is a^{-1}
  - In multiple dimensions, "width" along direction v_i is λ_i^{-1}, where v_i is an eigenvector of $X^\top X$ with eigenvalue λ_i

Intuition on Vectorized Linear Regression



Directions/magnitudes are given by eigenvectors/eigenvalues of $X^\top X$

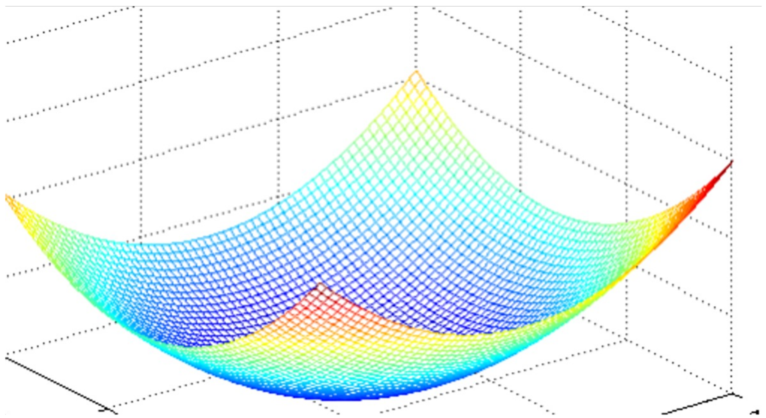
Strategy 1: Closed-Form Solution

- Recall that linear regression minimizes the loss

$$L(\beta; Z) = \frac{1}{n} \|Y - X\beta\|_2^2$$

- Minimum solution has gradient equal to zero:

$$\nabla_\beta L\big(\hat{\beta}(Z); Z\big) = 0$$



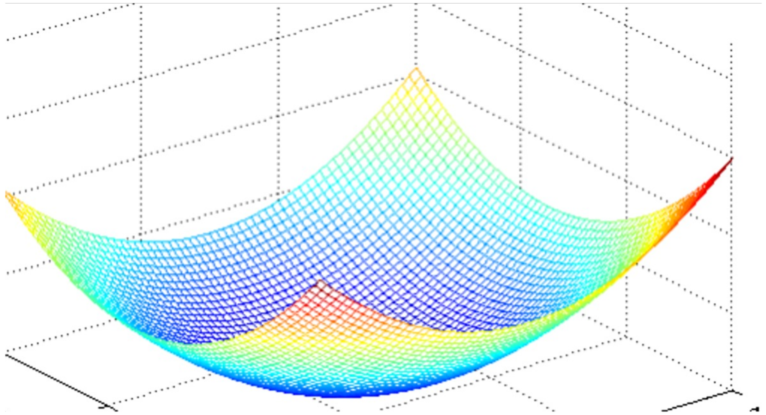
Strategy 1: Closed-Form Solution

- Recall that linear regression minimizes the loss

$$L(\beta; Z) = \frac{1}{n}\|Y - X\beta\|_2^2$$

- Minimum solution has gradient equal to zero:

$$\nabla_\beta L(\hat{\beta}; Z) = 0$$



Strategy 1: Closed-Form Solution

- The gradient is

$$\nabla_{\beta} L(\beta; Z)$$

Strategy 1: Closed-Form Solution

- The gradient is

$$\nabla_{\beta} L(\beta; Z) = \nabla_{\beta} \frac{1}{n} \|Y - X\beta\|_2^2$$

Strategy 1: Closed-Form Solution

- The gradient is

$$
\begin{aligned}
\nabla_\beta L(\beta; Z) = \nabla_\beta \frac{1}{n} \|Y - X\beta\|_2^2 &= \nabla_\beta \frac{1}{n} (Y - X\beta)^\top (Y - X\beta) \\
&= \frac{2}{n} \left[\nabla_\beta (Y - X\beta)^\top\right] (Y - X\beta) \\
&= -\frac{2}{n} X^\top (Y - X\beta) \\
&= -\frac{2}{n} X^\top Y + \frac{2}{n} X^\top X \beta
\end{aligned}
$$

Aside: Intuition on Computing Gradients

- **Warning:** Intuitive but **easy to make mistakes**
- The loss is

$$
\begin{aligned}
L(\beta + d\beta; Z) &= \frac{1}{n}\|Y - X(\beta + d\beta)\|_2^2 \\
&= \frac{1}{n}\|(Y - X\beta) - Xd\beta\|_2^2 \\
&= \frac{1}{n}\|Y - X\beta\|_2^2 - \frac{2}{n}(Y - X\beta)^\top X d\beta + \frac{1}{n}\|Xd\beta\|_2^2 \\
&= L(\beta; Z) \underbrace{- \frac{2}{n}(Y - X\beta)^\top X}_{= \nabla_\beta L(\beta; Z)^\top} d\beta + O(\|d\beta\|_2^2)
\end{aligned}
$$

Coefficient of $d\beta$ term

Intuition on the Gradient

- By linearity of the gradient, we have

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

- The gradient for a single term is

$$\nabla_\beta (y_i - \beta^\top x_i)^2 = 2(y_i - \beta^\top x_i)x_i$$

- I.e., the current error $y_i - \beta^\top x_i$ times the feature x_i

Strategy 1: Closed-Form Solution

- The gradient is

$$\nabla_\beta L(\beta; Z) = \nabla_\beta \frac{1}{n} \|Y - X\beta\|_2^2 = -\frac{2}{n} X^\top Y + \frac{2}{n} X^\top X\beta$$

- Setting $\nabla_\beta L(\hat{\beta}; Z) = 0$, we have $X^\top X\hat{\beta} = X^\top Y$

Strategy 1: Closed-Form Solution

- Setting $\nabla_{\beta} L(\hat{\beta}; Z) = 0$, we have $X^{\top} X \hat{\beta} = X^{\top} Y$
- Assuming $X^{\top} X$ is invertible, we have

$$\hat{\beta}(Z) = (X^{\top} X)^{-1} X^{\top} Y$$

Note on Invertibility

- Closed-form solution only **unique** if $X^\top X$ is invertible
  - Otherwise, **multiple solutions exist** to $X^\top X\hat{\beta} = X^\top Y$
  - **Intuition:** Underconstrained system of linear equations

- **Example:**

$$\begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

  - In this case, any $\hat{\beta}_2 = 2 - \hat{\beta}_1$ is a solution

When Can this Happen?

- **Case 1**
  - Fewer data examples than feature dimension (i.e., $n < d$)
  - **Solution:** Remove features so $d \leq n$
  - **Solution:** Collect more data until $d \leq n$

- **Case 2:** Some feature is a linear combination of the others
  - **Special case (duplicated feature):** For some j and j', $x_{i,j} = x_{i,j'}$ for all i
  - **Solution:** Remove linearly dependent features
  - **Solution:** Use L_2 regularization

Shortcomings of Closed-Form Solution

- Computing $\hat{\beta}(Z) = (X^\top X)^{-1} X^\top Y$ can be challenging

- **Computing $(\boldsymbol{X}^\top \boldsymbol{X})^{-1}$ is $\boldsymbol{O}(\boldsymbol{d}^3)$**
  - $d = 10^4$ features → $O(10^{12})$
  - Even storing $X^\top X$ requires a lot of memory

- **Numerical accuracy issues due to “ill-conditioning”**
  - $X^\top X$ is “barely” invertible
  - Then, $(X^\top X)^{-1}$ has large variance along some dimension
  - Regularization helps (more on this later)