Announcements

- **Reminder: Masks are required!**
- **Homework 1:** Due in **one week (next Wednesday at 8pm)!**
  - Requiring submission of Python file **in addition to iPython Notebook file** (see announcement on Ed Discussion for details)
- **Quiz 1 will be posted on canvas tonight:** Due in **one week!**
- **Waitlist**
  - Admitted to capacity
  - Only considering additional applications if students do not enroll or drop

Project: Goals

- Apply algorithms you learn in this class to a real-world dataset
- **Must go beyond simply applying an existing machine learning algorithm to an existing dataset**

Project: Goals

- **Data:** Collect a new dataset, augment an existing one, or modify data preprocessing to improve performance
- **Algorithm:** Modify an existing algorithm, by changing the neural network architecture, etc. to improve performance
- **Analysis:** Analyze sensitivity to hyperparameters, out-of-distribution inputs, etc.

Project: Grading

- **You will be graded on your understanding of ML covered in this class, and the quality and value of your novel contributions**
  - **Applying an existing algorithm to a standard dataset is not enough**

- We will share a link to past projects
- PapersWithCode.com or Kaggle.com can also be good starting points

Project: Logistics

- **Teams of 3 students**
  - Find teammates on your own
  - Email instructors **by Friday, 9/28** and we will do our best to help

- **Project milestones**
  - **Milestone 1 (2 pages, due 10/12):** Project proposal (with groups chosen)
  - **Milestone 2 (4 pages, due 11/9):** Preliminary results
  - **Milestone 3 (6 pages, due 12/7):** Final reports

Lecture 2: Linear Regression (Part 1)

CIS 4190/5190

Fall 2022

Recap: Types of Learning

- **Supervised learning**
  - **Input:** Examples of inputs and outputs
  - **Output:** Model that predicts unknown output given a new input
- **Unsupervised learning**
  - **Input:** Examples of some data (no “outputs”)
  - **Output:** Representation of structure in the data
- **Reinforcement learning**
  - **Input:** Sequence of interactions with an environment
  - **Output:** Policy that performs a desired task

Today

- Deep dive into **linear regression**
  - Basic example of a **supervised learning algorithm**

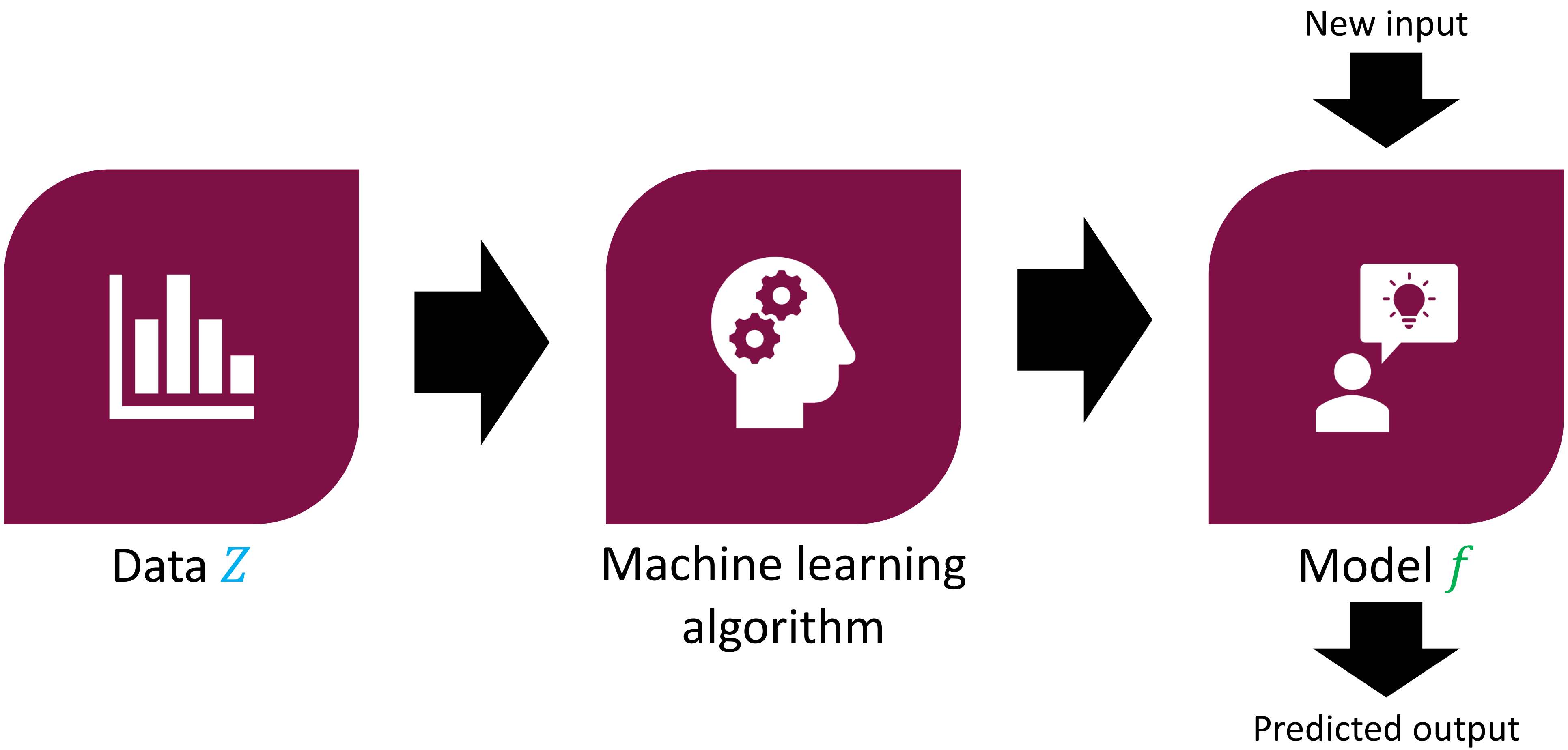
- Captures many fundamental machine learning concepts
  - Function approximation view of machine learning
  - Bias-variance tradeoff
  - Regularization
  - Training/validation/test split
  - Optimization and gradient descent

Agenda

- **Function approximation view of machine learning**
  - Modern strategy for designing machine learning algorithms
  - **By example:** Linear regression, a simple machine learning algorithm

- **Bias-variance tradeoff**
  - Fundamental challenge in machine learning
  - **By example:** Linear regression with feature maps

Machine Learning for Prediction



Question: What **model family** (a.k.a. **hypothesis class**) to consider?

Linear Functions

- Consider the space of linear functions $f_\beta(x)$ defined by

$$f_\beta(x) = \beta^\top x$$

Linear Functions

- Consider the space of linear functions $f_\beta(x)$ defined by

$$f_\beta(x) = \beta^\top x = \begin{bmatrix} \beta_1 & \cdots & \beta_d \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} = \beta_1 x_1 + \cdots + \beta_d x_d$$

- $x \in \mathbb{R}^d$ is called an **input** (a.k.a. **features** or **covariates**)
- $\beta \in \mathbb{R}^d$ is called the **parameters** (a.k.a. **parameter vector**)
- $y = f_\beta(x)$ is called the **label** (a.k.a. **output** or **response**)

Linear Regression Problem

- **Input:** Dataset $Z = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$
- **Output:** A linear function $f_\beta(x) = \beta^\top x$ such that $y_i \approx \beta^\top x_i$

- **Typical notation**
  - Use i to index examples (x_i, y_i) in data Z
  - Use j to index components x_j of $x \in \mathbb{R}^d$
  - x_{ij} is component j of input example i

- **Goal:** Estimate $\beta \in \mathbb{R}^d$

Linear Regression Problem

- **Input:** Data $Z = \{(x_1, y_1), \dots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$
- **Output:** A linear function $f_\beta(x) = \beta^\top x$ such that $y_i \approx \beta^\top x_i$

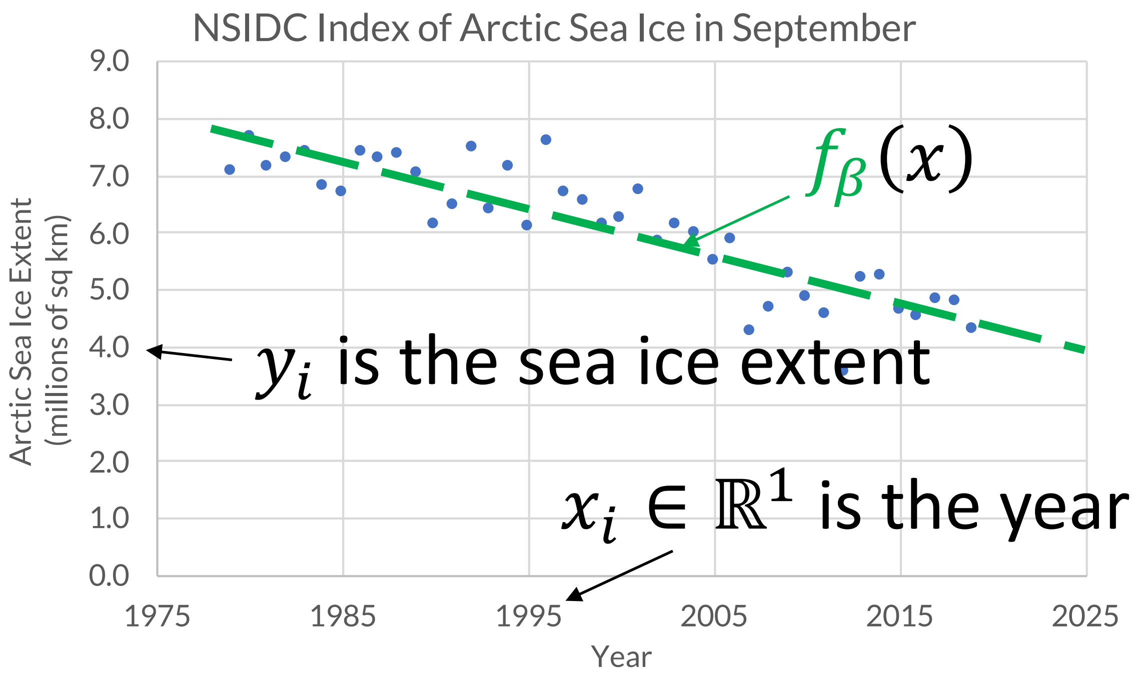


Image: https://www.flickr.com/photos/gsfc/5937599688/
Data from https://nsidc.org/arcticseaicenews/sea-ice-tools/

Linear Regression Problem

What does this mean?

- **Input:** Data $Z = \{(x_1, y_1), \dots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$
- **Output:** A linear function $f_\beta(x) = \beta^\top x$ such that $y_i \approx \beta^\top x_i$

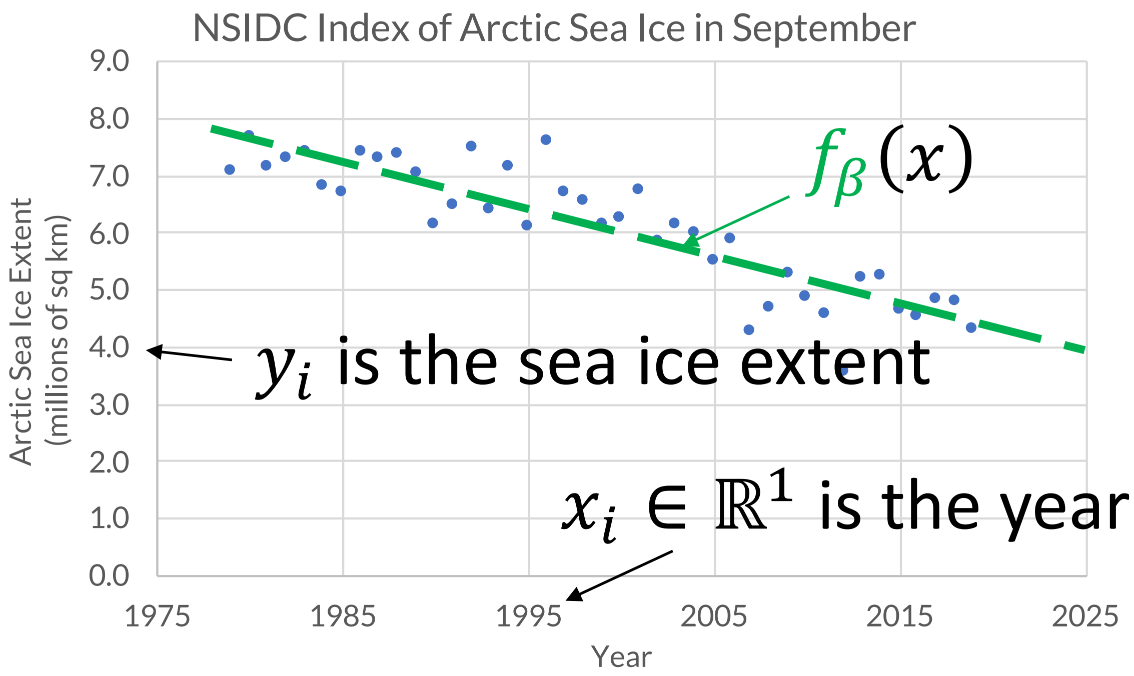


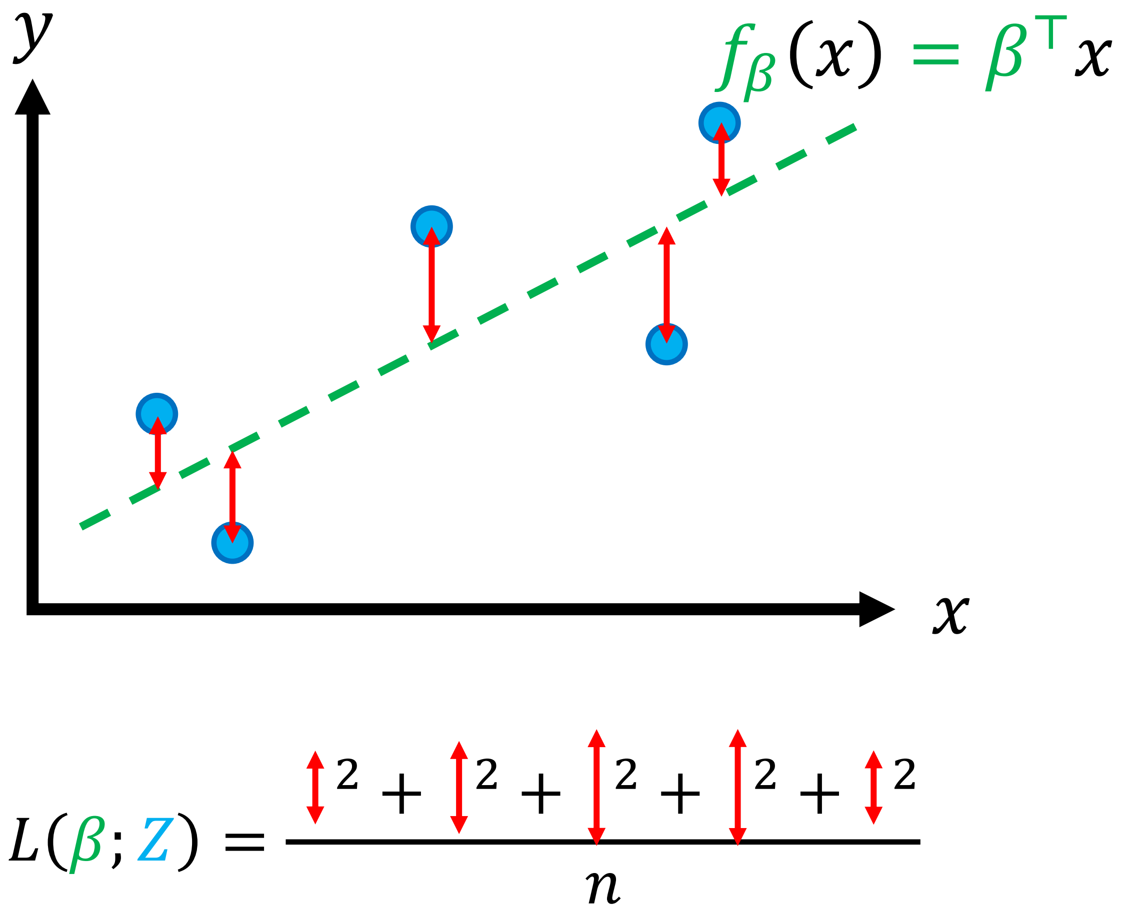
Image: https://www.flickr.com/photos/gsfc/5937599688/
Data from https://nsidc.org/arcticseaicenews/sea-ice-tools/

Choice of Loss Function

- $y_i \approx \beta^\top x_i$ if $(y_i - \beta^\top x_i)^2$ small
- **Mean squared error (MSE):**

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

- Computationally convenient and works well in practice



Linear Regression Problem

- **Input:** Data $Z = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$
- **Output:** A linear function $f_\beta(x) = \beta^\top x$ such that $y_i \approx \beta^\top x_i$

Linear Regression Problem

- **Input:** Data $Z = \{(x_1, y_1), \dots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$
- **Output:** A linear function $f_\beta(x) = \beta^\top x$ that minimizes the MSE:

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

Linear Regression Algorithm

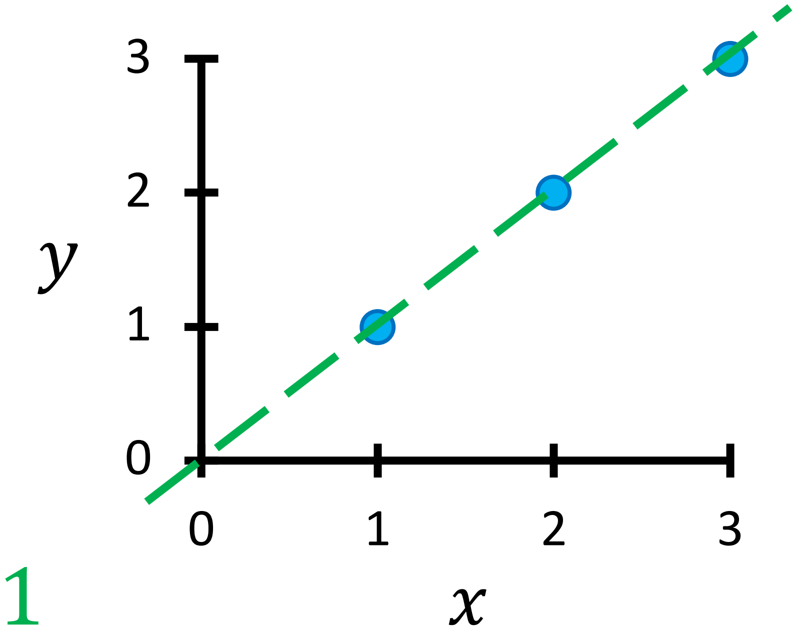
- **Input:** Dataset $Z = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
- Compute

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

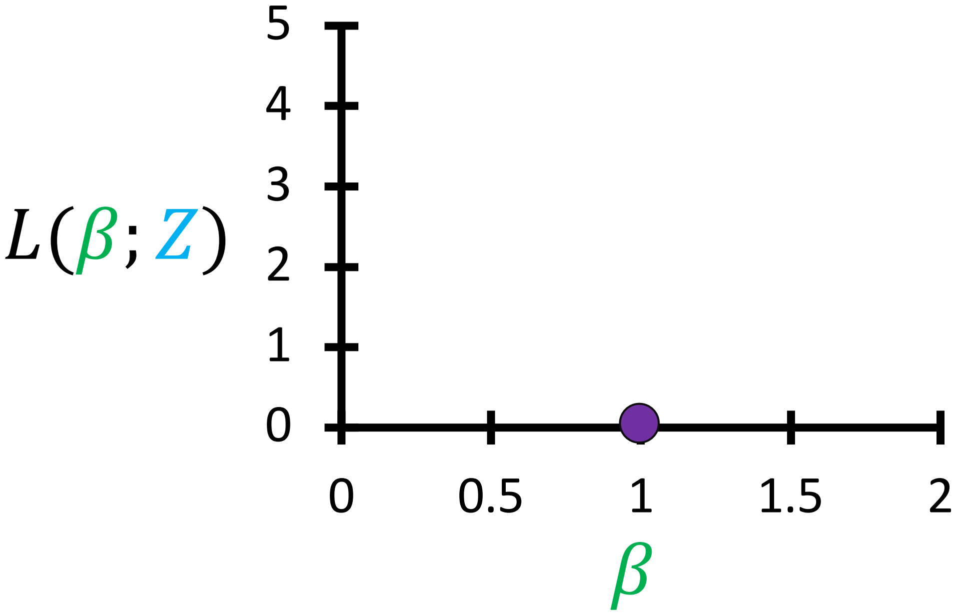
- **Output:** $f_{\hat{\beta}(Z)}(x) = \hat{\beta}(Z)^\top x$
- Discuss algorithm for computing the minimal β later

Intuition on Minimizing MSE Loss

- Consider $x \in \mathbb{R}$ and $\beta \in \mathbb{R}$

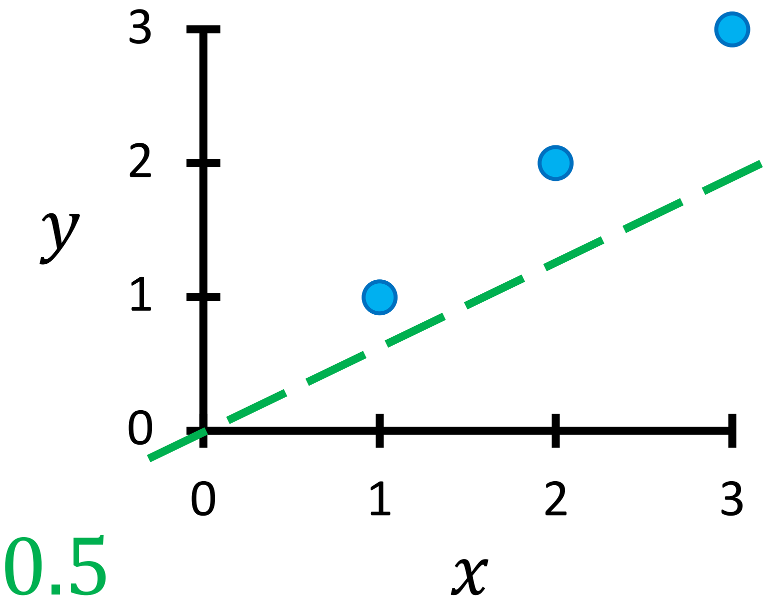


$\beta = 1$

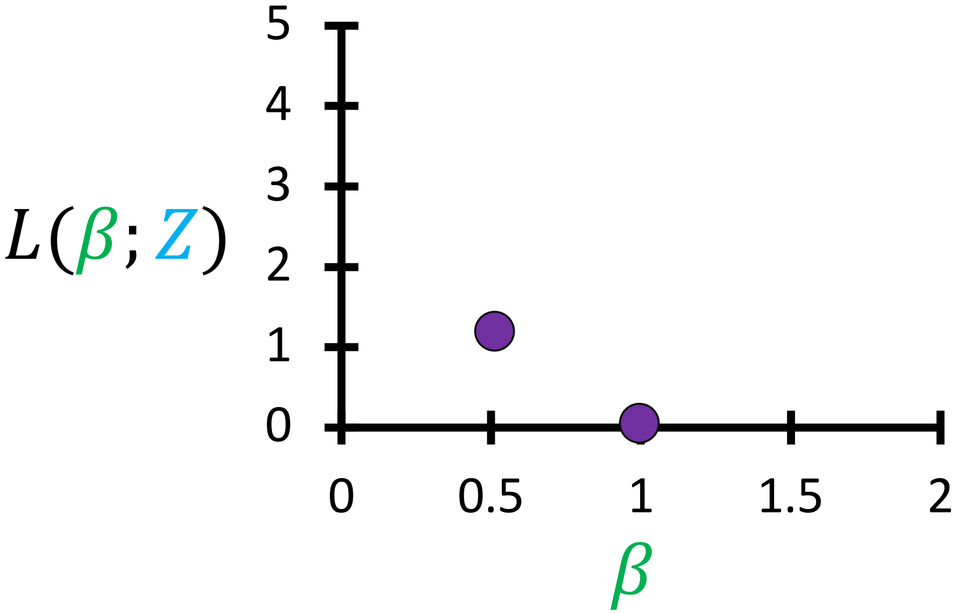


Intuition on Minimizing MSE Loss

- Consider $x \in \mathbb{R}$ and $\beta \in \mathbb{R}$

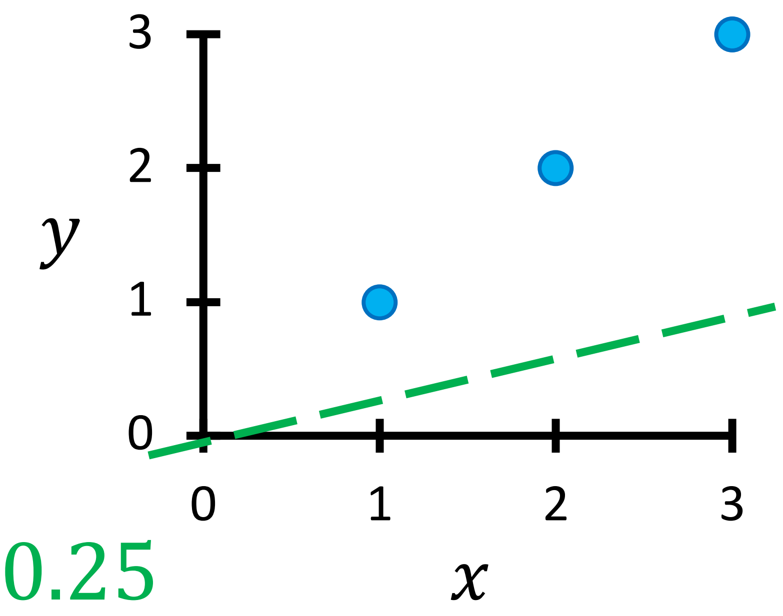


$\beta = 0.5$

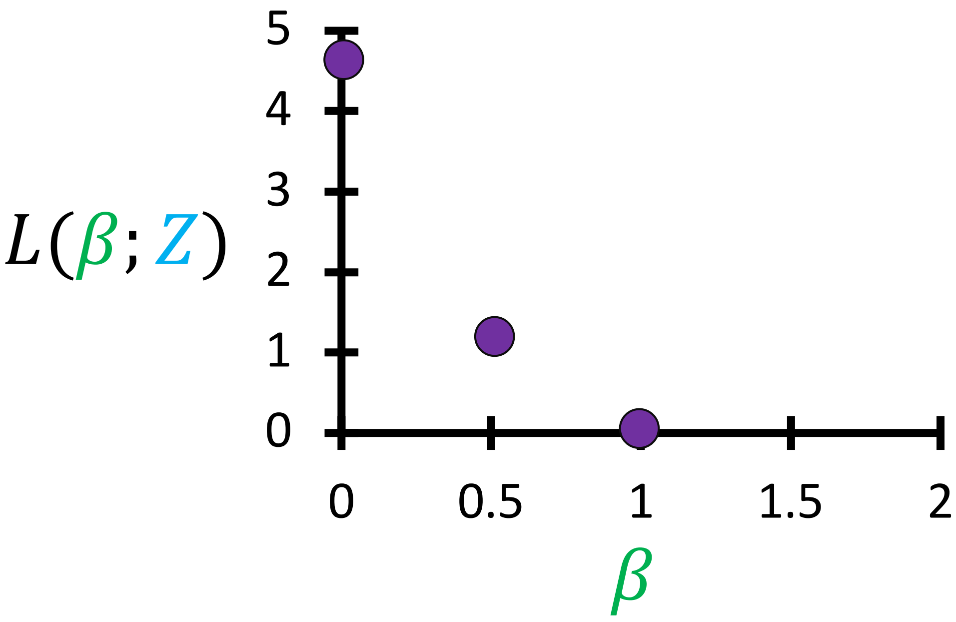


Intuition on Minimizing MSE Loss

- Consider $x \in \mathbb{R}$ and $\beta \in \mathbb{R}$

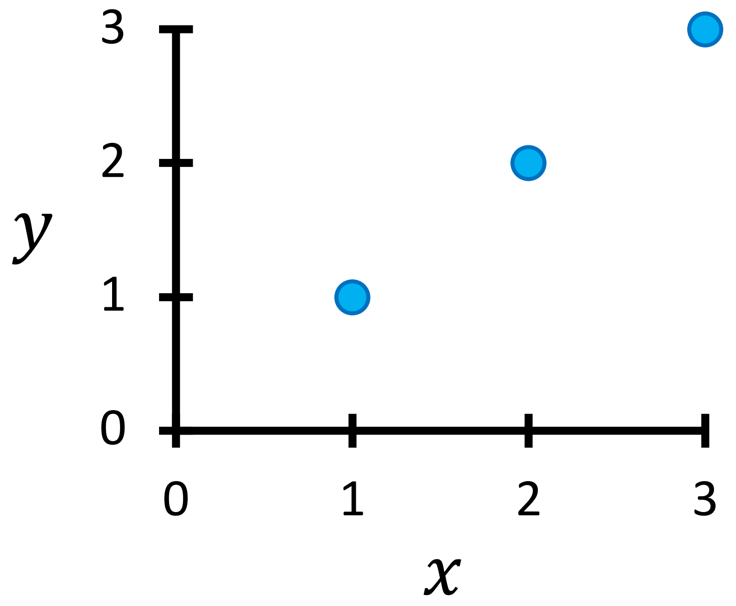


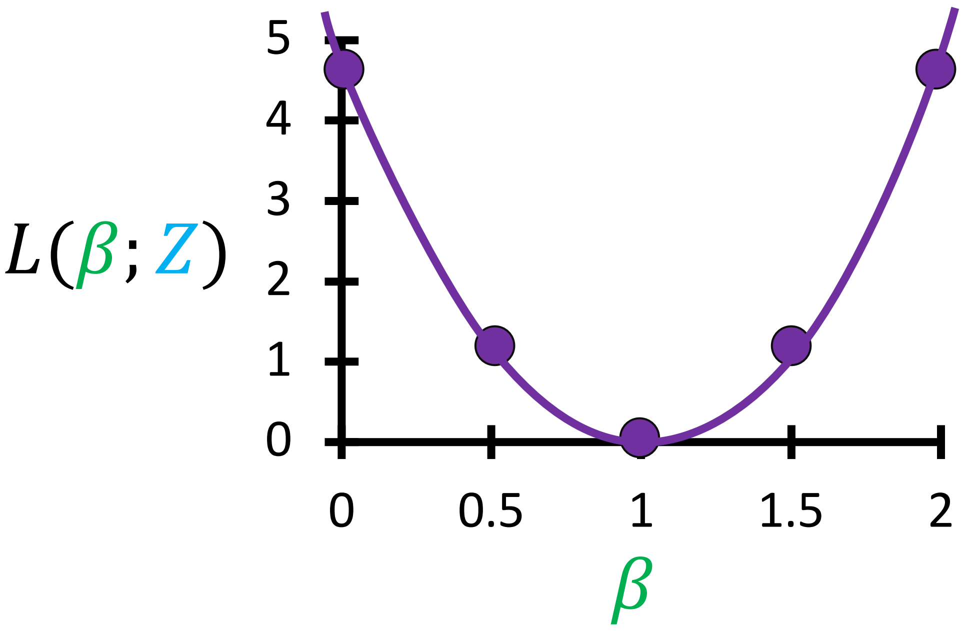
$\beta = 0.25$



Intuition on Minimizing MSE Loss

- Consider $x \in \mathbb{R}$ and $\beta \in \mathbb{R}$

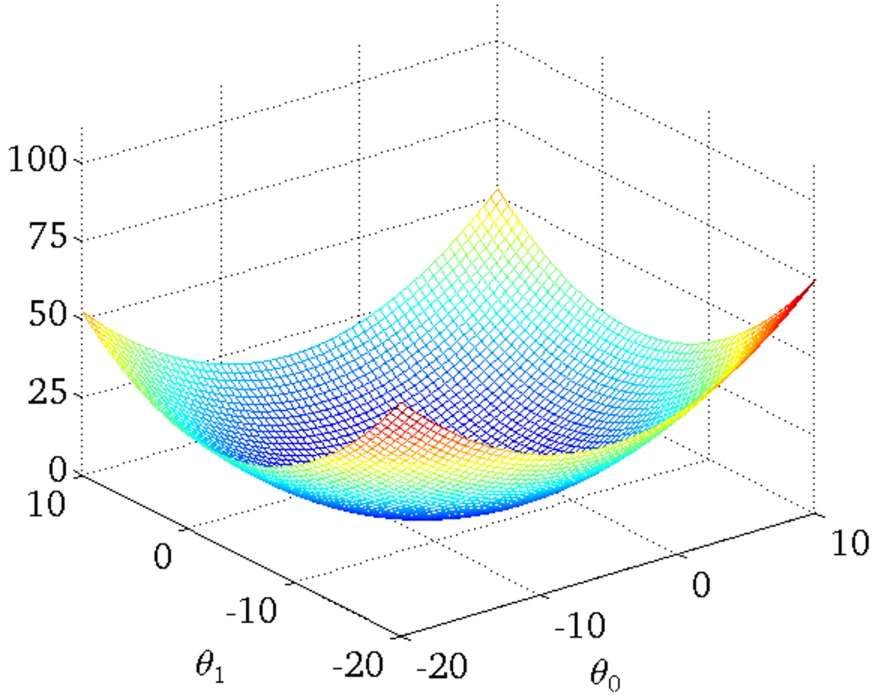


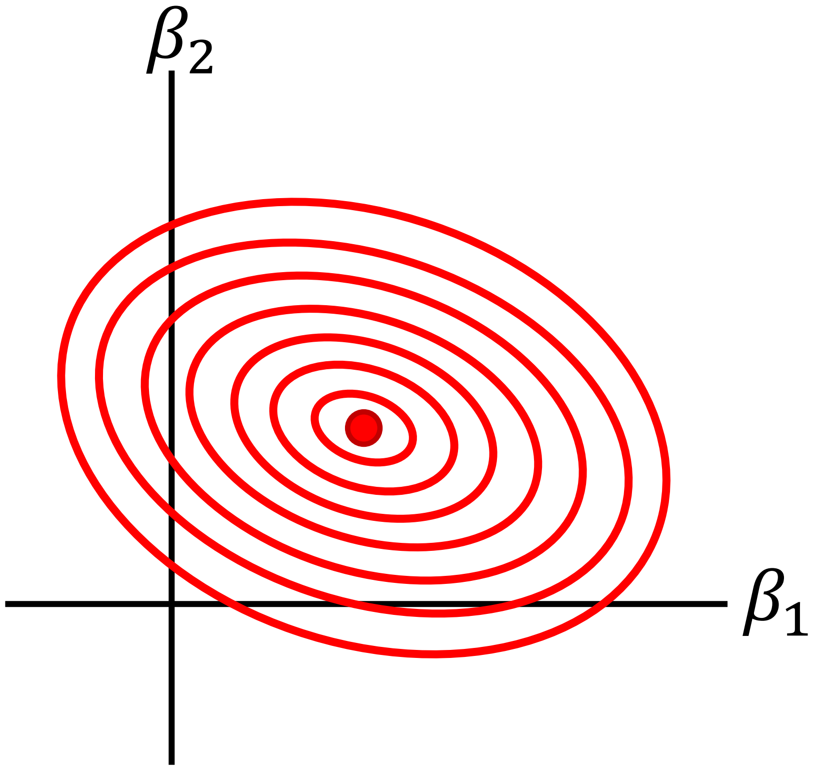


Intuition on Minimizing MSE Loss

- **Convex** (“bowl shaped”) in general

$L(\beta; Z)$





“Good” Mean Squared Error?

- Need to compare to baseline!
  - Constant prediction
  - Handcrafted model
  - …

- **Later:** Training vs. test MSE

Alternative Loss Functions

- **Mean absolute error:** $\frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$

- **Mean relative error:** $\frac{1}{n}\sum_{i=1}^{n}\frac{|\widehat{y_i}-y_i|}{|y_i|}$

- $\boldsymbol{R^2}$ **score:** $1 - \frac{\text{MSE}}{\text{Variance}}$
  - “Coefficient of determination”
  - Higher is better, $R^2 = 1$ is perfect

Alternative Loss Functions

- **Pearson correlation:** $\frac{1}{n}\sum_{i=1}^{n}\frac{(\hat{y}_i-\hat{\mu})(y_i-\mu)}{\hat{\sigma}\sigma}$
  - Usually estimated from some sampled measurements of those variables, and denoted as R (related to R^2 on the last slide!)

- **Rank-order correlation:**
  - First rank the measurements of $\hat{y}_{\boldsymbol{i}}$ and y separately, then replace each value in y by its rank, and ditto for $\hat{y}$
  - Then measure the linear correlation between those ranks

Taking a Step Up…

Function Approximation View of ML

Data Z

Machine learning algorithm

Model f

ML algorithm outputs a model f that best "approximates" the given data Z

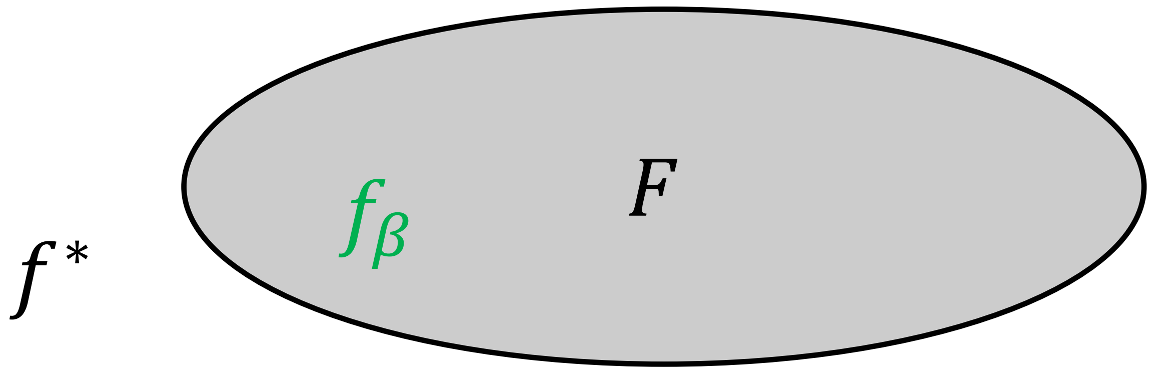
Function Approximation View of ML

- Framework for designing machine learning algorithms
- **Two design decisions**
  - What is the family of candidate models f? (E.g., linear functions)
  - How to define “approximating”? (E.g., MSE loss)

Aside: "True Function"

- **Input:** Dataset Z
  - Presume there is an unknown function f^* that **generates** Z

- **Goal:** Find an approximation $f_\beta \approx f^*$ in our model family $f_\beta \in F$
  - Typically, f^* not in our model family F



Function Approximation View of ML

- Framework for designing machine learning algorithms
- **Two design decisions**
  - What is the family of candidate models f? (E.g., linear functions)
  - How to define "approximating"? (E.g., MSE loss)
- How do we specialize to linear regression?

Function Approximation View of ML

Data Z

Machine learning algorithm

Model f

Loss Minimization

Data Z

Machine learning algorithm

Model f

Loss Minimization

Data Z

Machine learning algorithm

Model f_β

Parametric model family (i.e., $F = \{ f_\beta \mid \beta \in \mathbb{R}^d \}$)

Loss Minimization

Data Z

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

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

ML algorithm minimizes loss of parameters β over data Z

Loss Minimization for Supervised Learning

Data Z

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

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

Loss Minimization for Supervised Learning

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

Goal is for function to approximate **label** y given **input** x

Loss Minimization for Regression

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

Label is a real number $y_i \in \mathbb{R}$

Linear Regression

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

MSE loss

Model is a linear function $f_{\beta}(x) = \beta^{\top} x$

Linear Regression

General strategy

- Model family $F = \{f_\beta\}_\beta$
- Loss function $L(\beta; Z)$

Linear regression strategy

- Linear functions $F = \{f_\beta(x) = \beta^\top x\}$
- MSE $L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2$

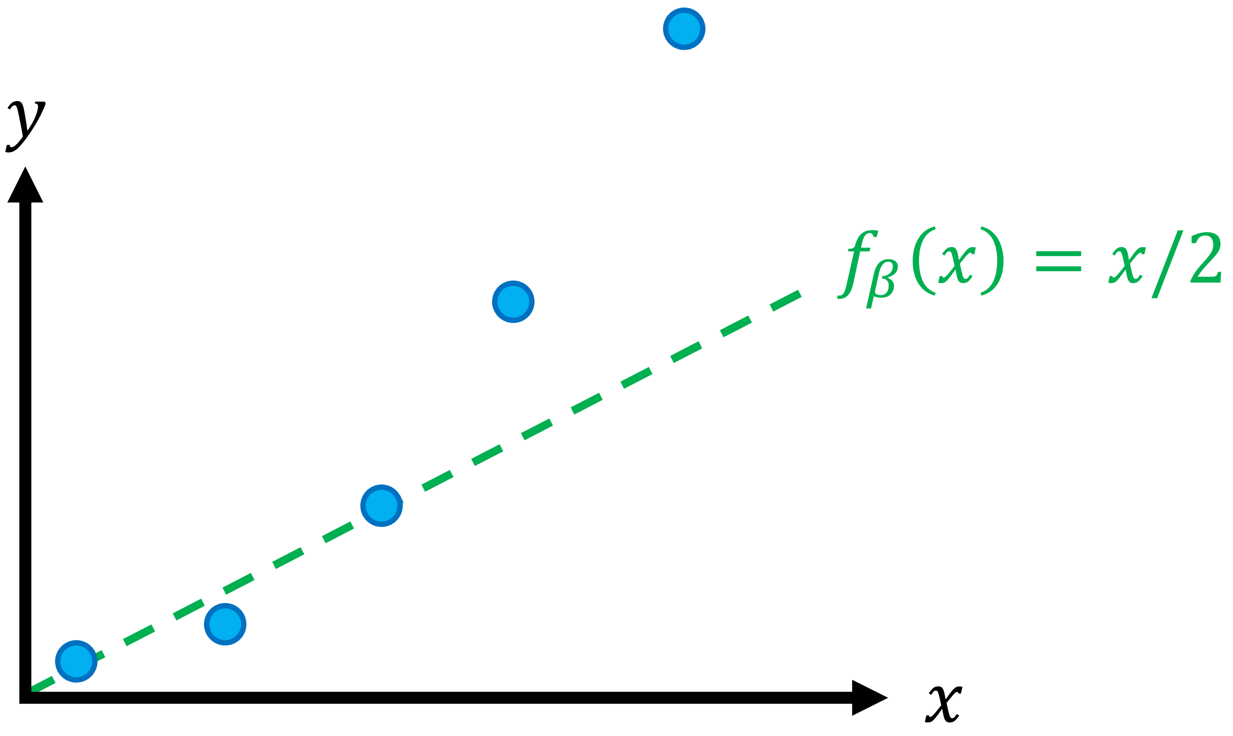
Linear regression algorithm

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

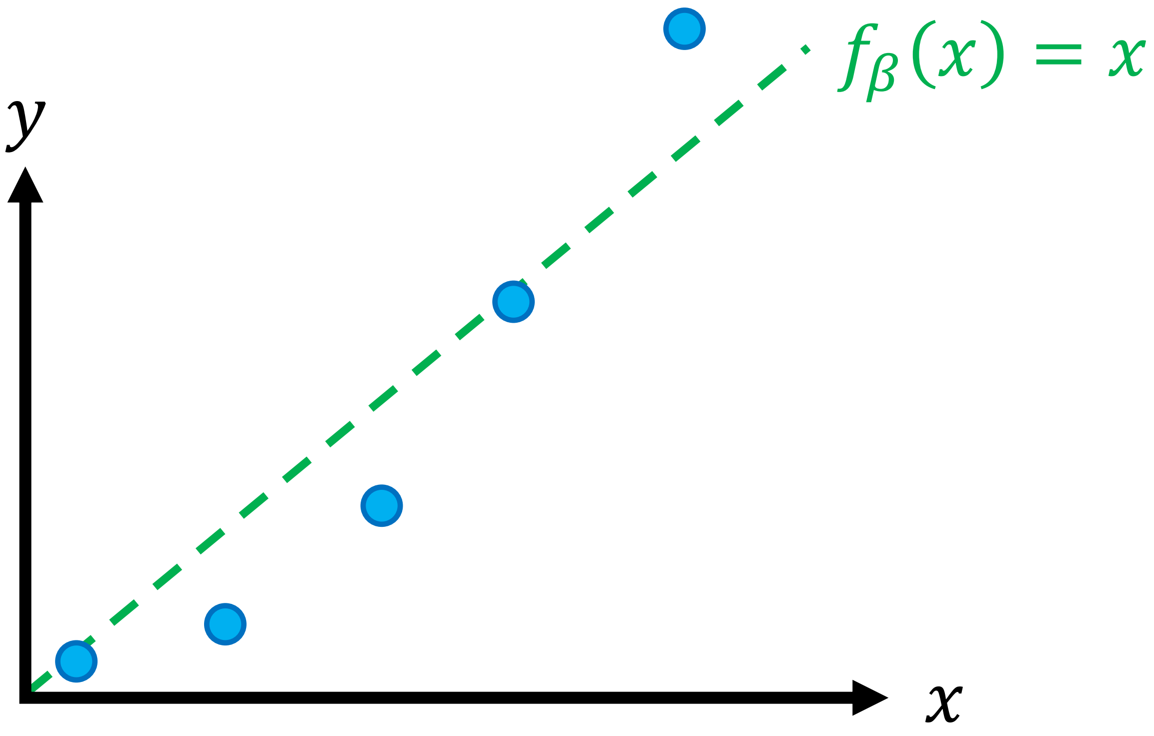
Agenda

- **Function approximation view of machine learning**
  - Modern strategy for designing machine learning algorithms
  - **By example:** Linear regression, a simple machine learning algorithm
- **Bias-variance tradeoff**
  - Fundamental challenge in machine learning
  - **By example:** Linear regression with feature maps

Example: Quadratic Function



Example: Quadratic Function



Can we get a better fit?

Feature Maps

General strategy

- Model family $F = \{f_\beta\}_\beta$
- Loss function $L(\beta; Z)$

Linear regression with feature map

- Linear functions over a given **feature map** $\phi: X \to \mathbb{R}^d$

$$F = \{f_\beta(x) = \beta^\top \phi(x)\}$$

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

Quadratic Feature Map

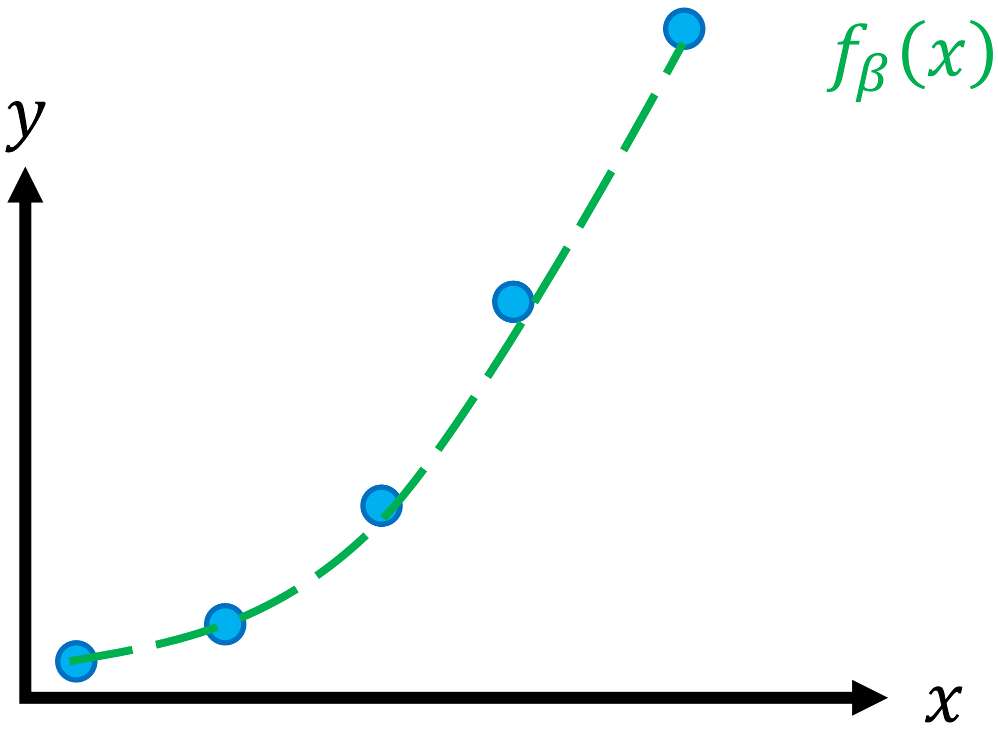
- Consider the feature map $\phi: \mathbb{R} \rightarrow \mathbb{R}^2$ given by

$$\phi(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$$

- Then, the model family is

$$f_\beta(x) = \beta_1 x + \beta_2 x^2$$

Quadratic Feature Map



$f_\beta(x) = 0x + 1x^2$

In our family for $\beta = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$!

Feature Maps

- Powerful strategy for encoding prior knowledge

- **Terminology**
  - x is the **input** and $\phi(x)$ are the **features**
  - Often used interchangeably

Examples of Feature Maps

- **Polynomial features**
  - $\phi(x) = \beta_1 + \beta_2 x_1 + \beta_3 x_2 + \beta_4 x_1^2 + \beta_5 x_1 x_2 + \beta_6 x_2^2 + \cdots$
  - Quadratic features are very common; capture "feature interactions"
  - Can use other nonlinearities (exponential, logarithm, square root, etc.)

- **Intercept term**
  - $\phi(x) = [1 \quad x_1 \quad \dots \quad x_d]^\top$
  - Almost always used; captures constant effect

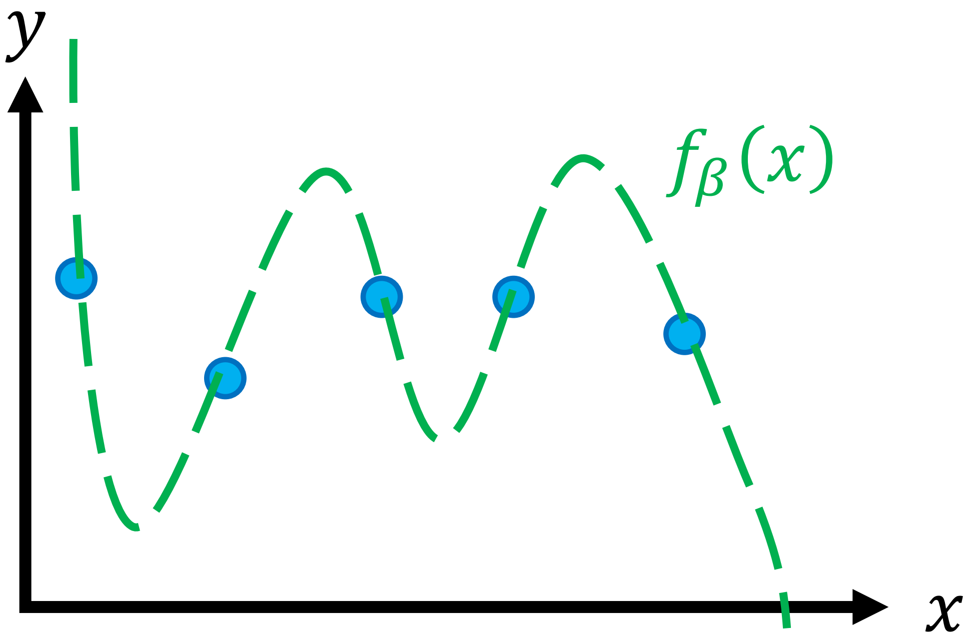
- **Encoding non-real inputs**
  - E.g., $x =$ "the food was good" and $y = 4$ stars
  - $\phi(x) = [1(\text{“good”} \in x) \quad 1(\text{“bad”} \in x) \quad \dots]^\top$

Algorithm

- Reduces to linear regression
- **Step 1:** Compute $\phi_i = \phi(x_i)$ for each x_i in Z
- **Step 2:** Run linear regression with $Z' = \{(\phi_1, y_1), \ldots, (\phi_n, y_n)\}$

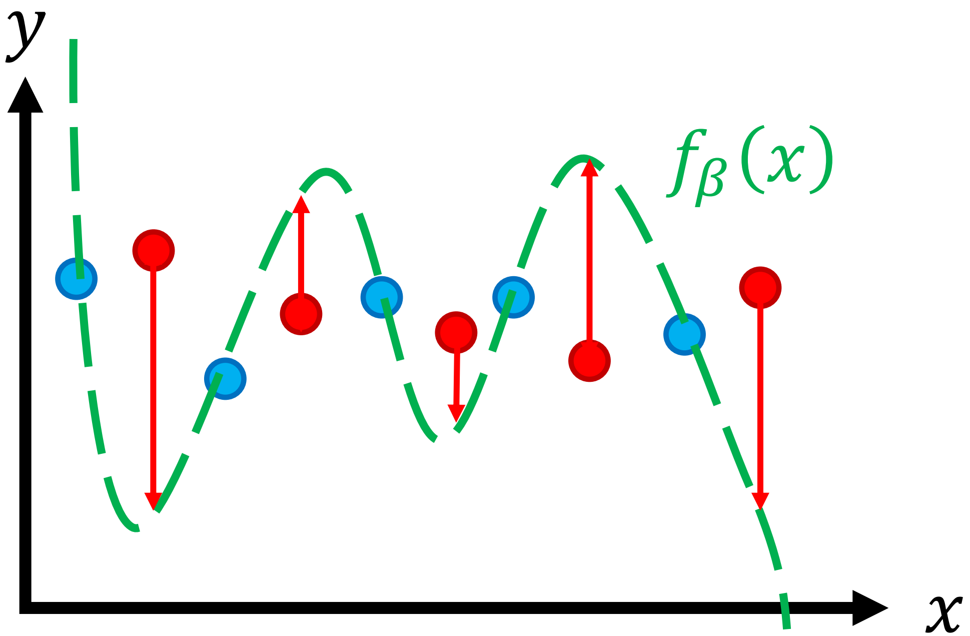
Question

- **Why not throw in lots of features?**
  - $\phi(x) = \beta_1 + \beta_2 x_1 + \beta_3 x_2 + \beta_4 x_1^2 + \beta_5 x_1 x_2 + \beta_6 x_2^2 + \cdots$
  - Can fit any n points using a polynomial of degree n



Prediction

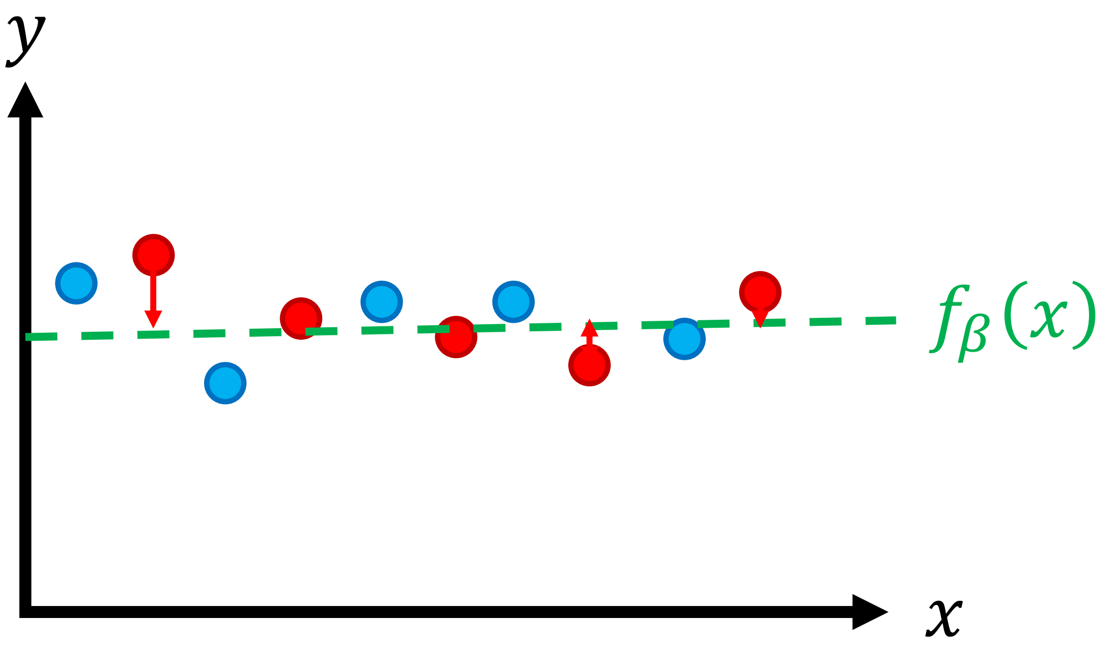
- **Issue:** The goal in machine learning is **prediction**
  - Given a **new** input x, predict the label $\hat{y} = f_\beta(x)$



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 $\hat{y} = f_\beta(x)$



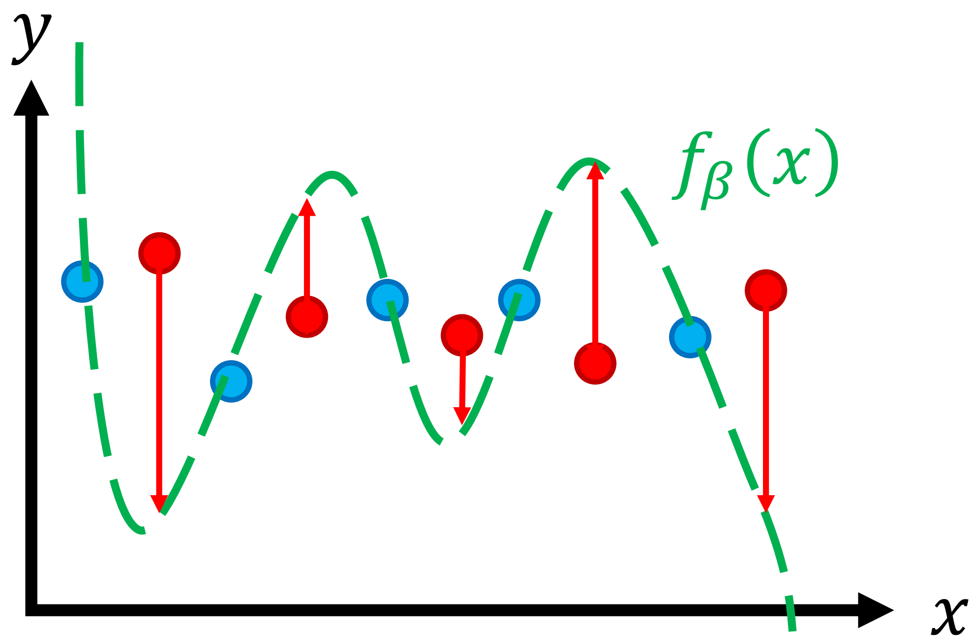
Vanilla linear regression actually works better!

Training vs. Test Data

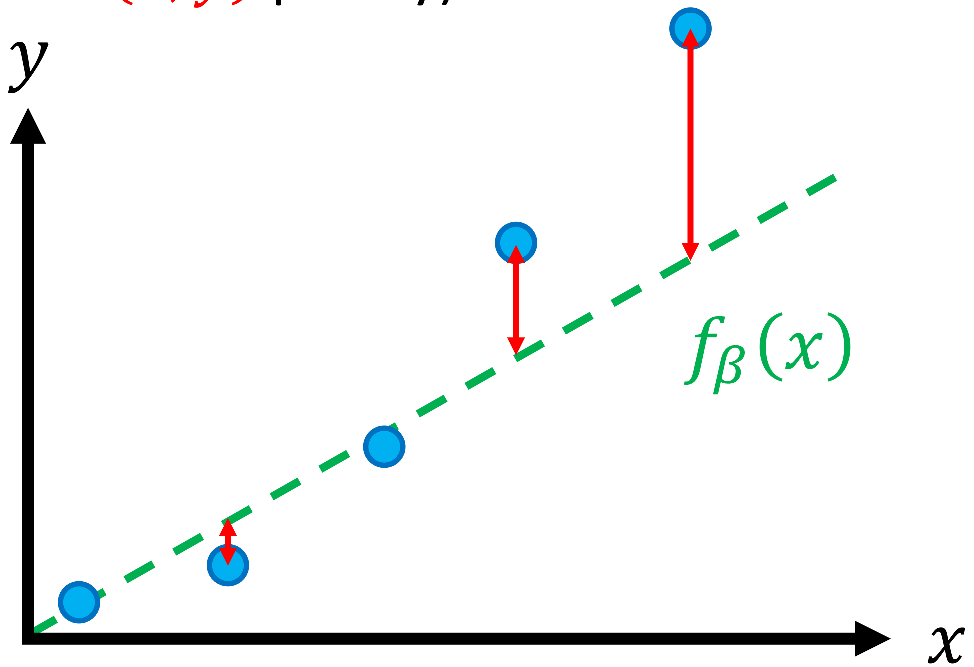
- **Training data:** Examples $Z = \{(x, y)\}$ used to fit our model
- **Test data:** New inputs x whose labels y we want to predict

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)



Aside: Why Does Overfitting Happen?

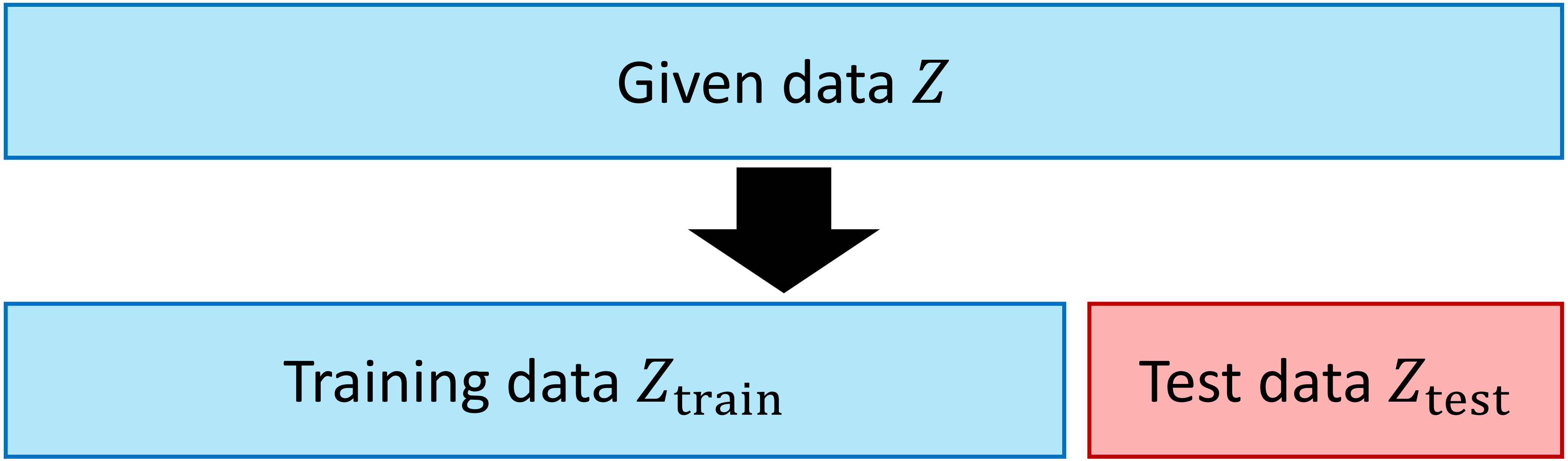
- Overfitting typically due to fitting noise in the data

- **Noise in labels $\boldsymbol{y}_i$**
  - True data generating process is more complex than we can capture
  - May depend on unobserved features

- **Noise in features $\boldsymbol{x}_i$**
  - Measurement error in the feature values
  - Errors due to preprocessing
  - Some features might be irrelevant to the decision function

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 Algorithm

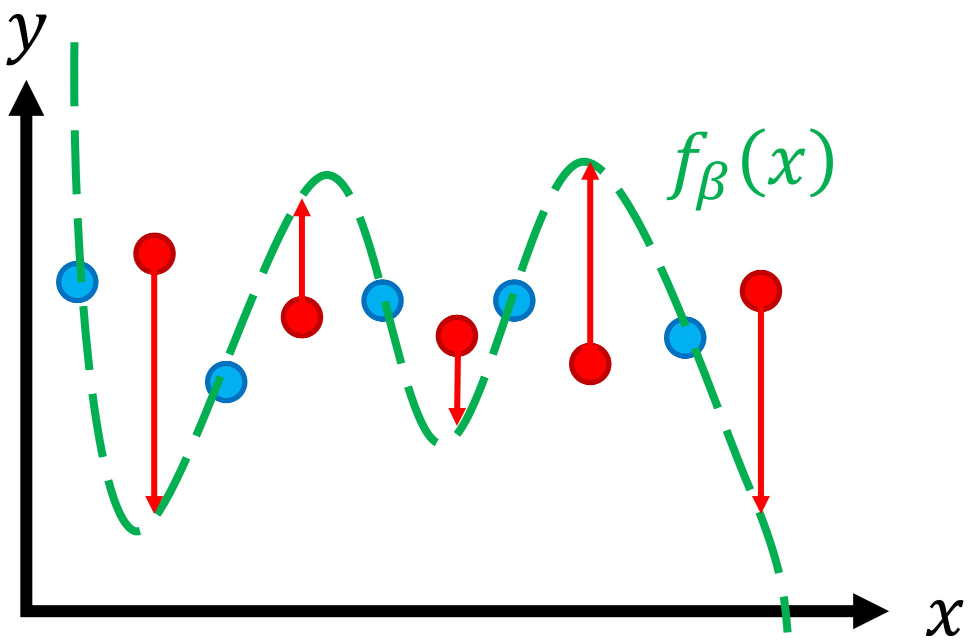
- **Step 1:** Split Z into Z_{train} and Z_{test}

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

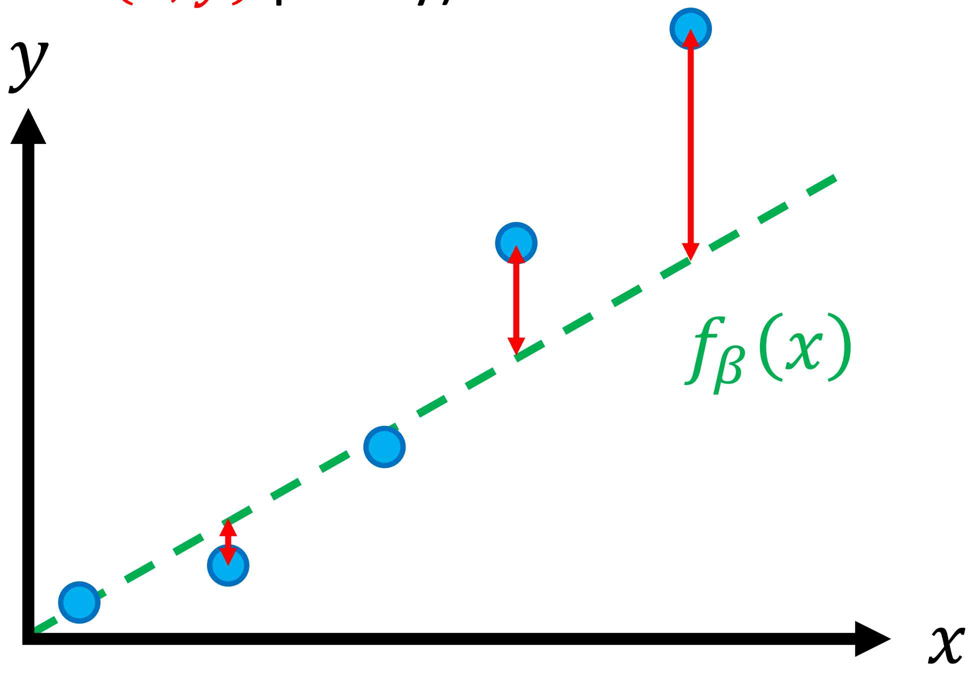
- **Step 2:** Run linear regression with Z_{train} to obtain $\hat{\beta}(Z_{\text{train}})$
- **Step 3:** Evaluate
  - **Training loss:** $L_{\text{train}} = L(\hat{\beta}(Z_{\text{train}}); Z_{\text{train}})$
  - **Test (or generalization) loss:** $L_{\text{test}} = L(\hat{\beta}(Z_{\text{train}}); Z_{\text{test}})$

Training/Test Split Algorithm

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

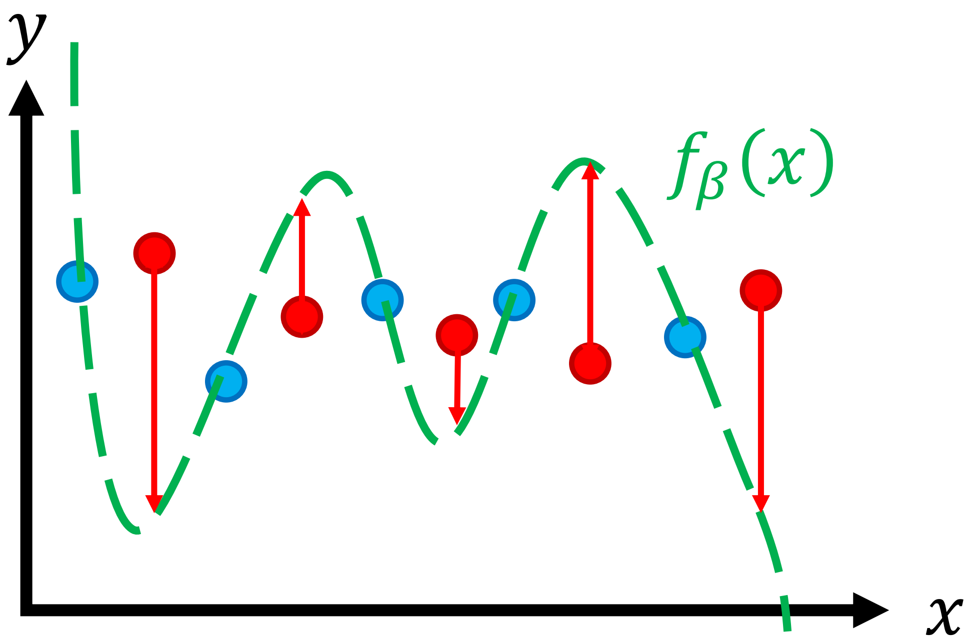


- **Underfitting**
  - Fit the **training data** Z well
  - (Necessarily fit new **test data** (x, y) poorly)

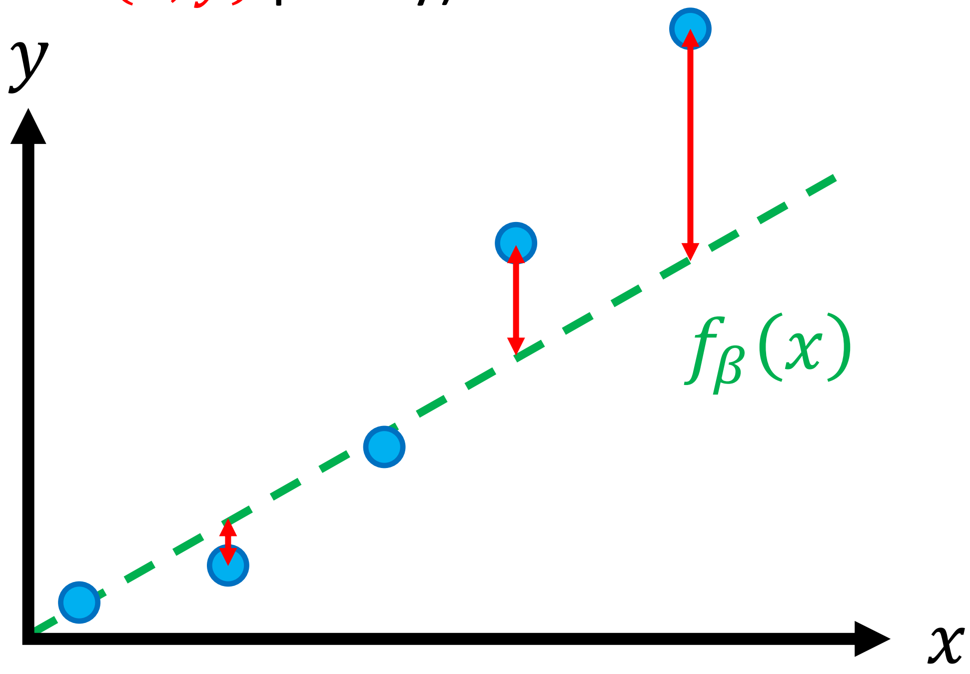


Training/Test Split Algorithm

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

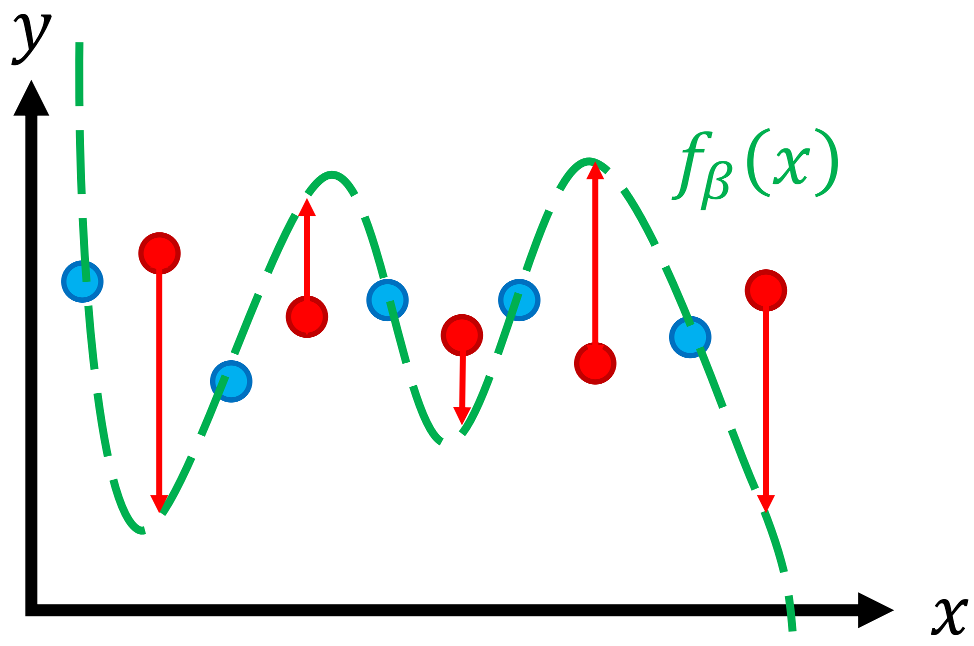


- **Underfitting**
  - Fit the **training data** Z well
  - (Necessarily fit new **test data** (x, y) poorly)

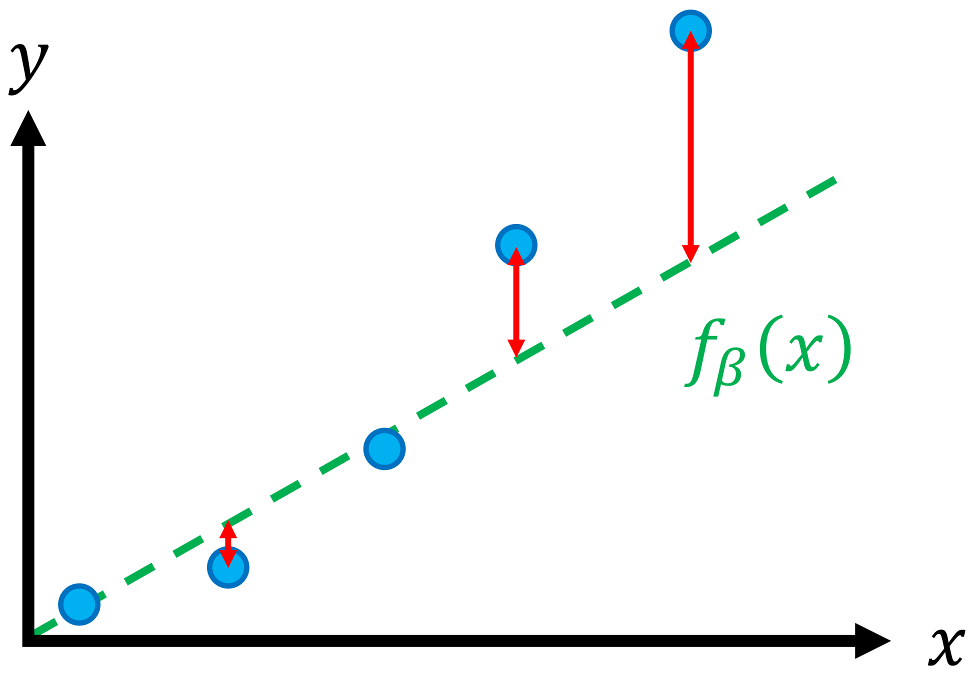


Training/Test Split Algorithm

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



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



Aside: IID Assumption

- **Underlying IID assumption**
  - Future data are drawn IID from same data distribution $P(x, y)$ as Z_{test}
  - 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)

How to Fix Underfitting/Overfitting?

- Choose the right model family!

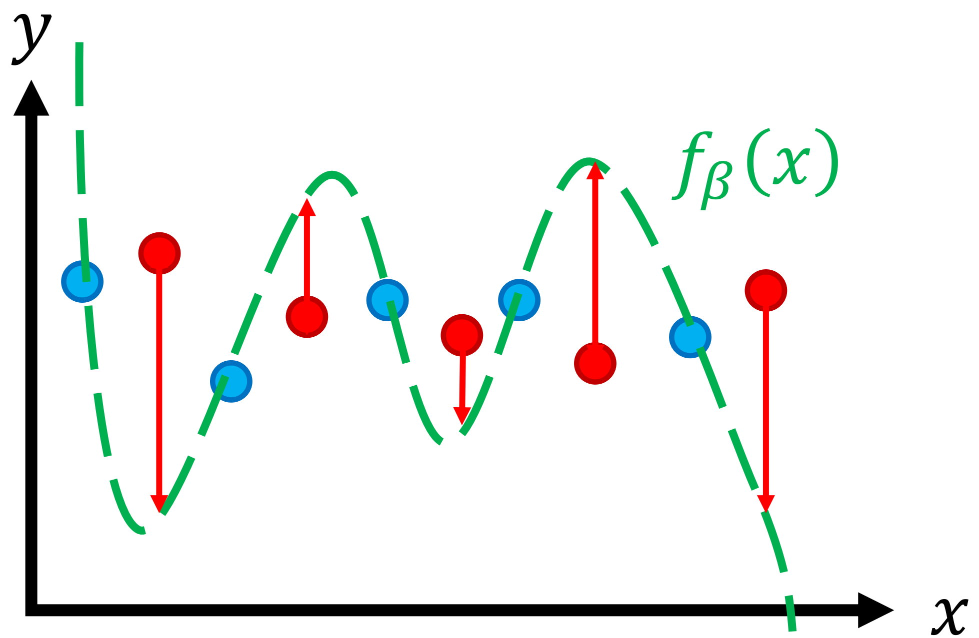
Role of Capacity

- **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**)

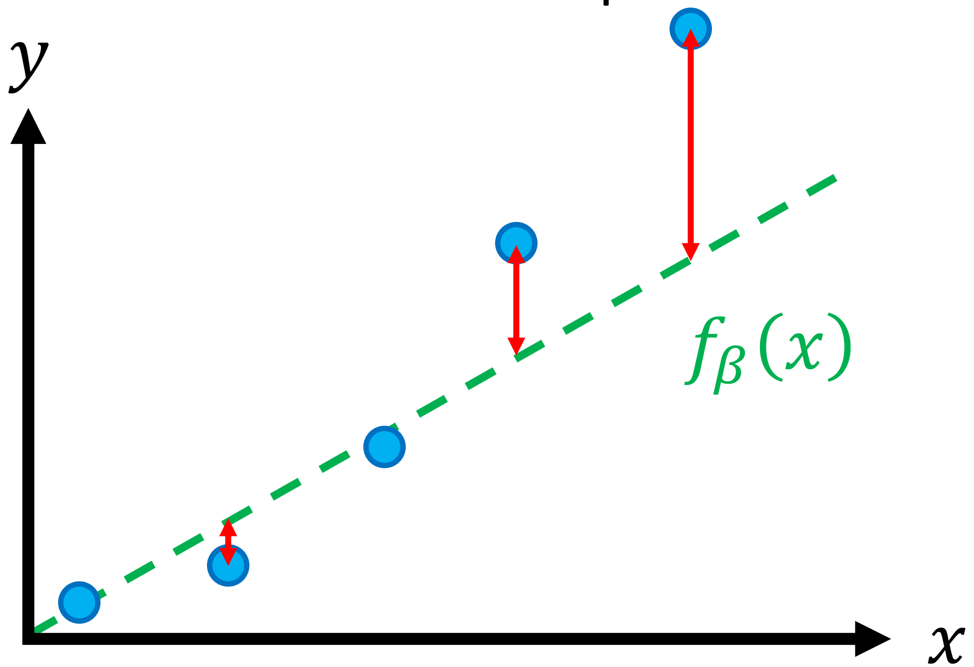
- For linear regression, capacity corresponds to feature dimension d
  - I.e., number of features in $\phi(x)$

Bias-Variance Tradeoff

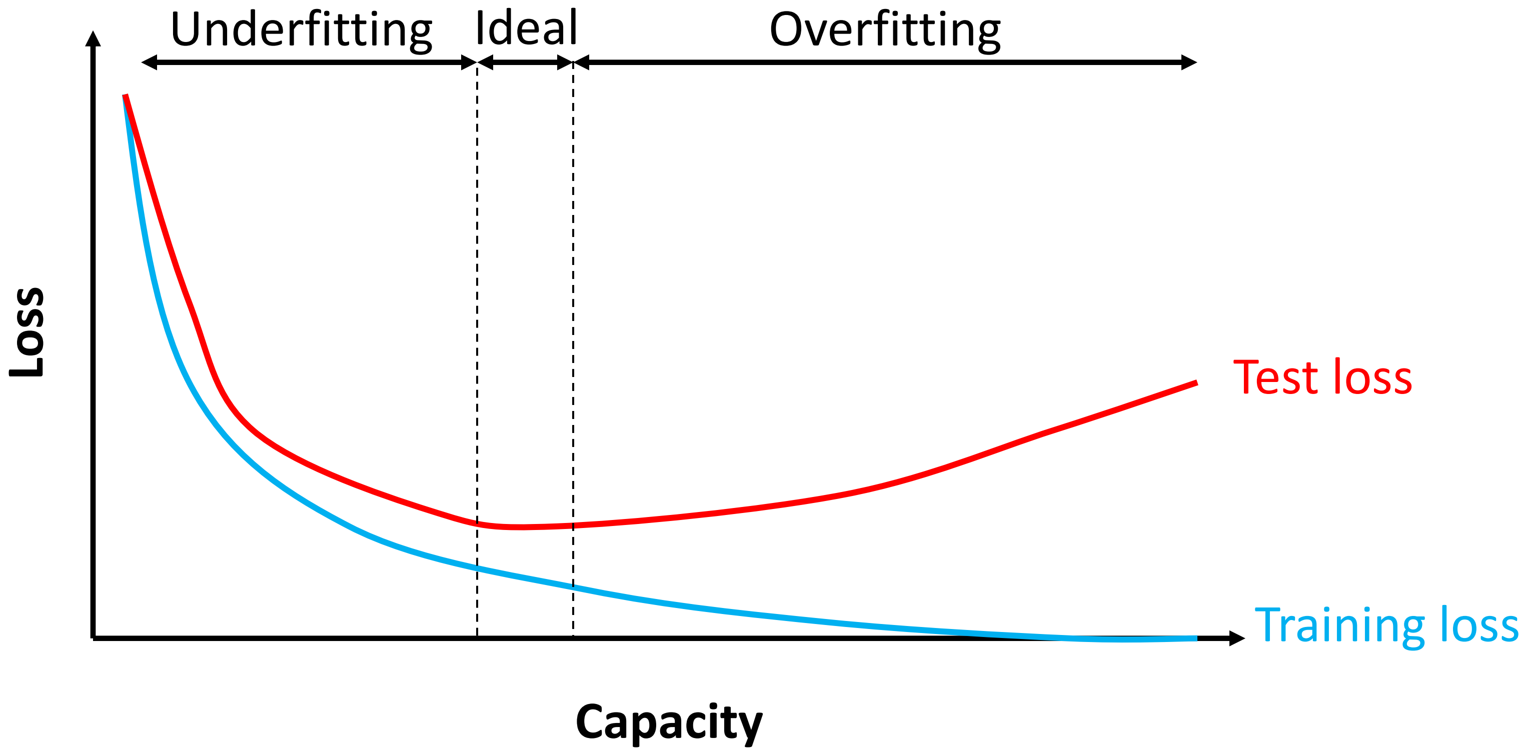
- **Overfitting (high variance)**
  - High capacity model capable of fitting complex data
  - Insufficient data to constrain it



- **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 **increase bias** and **decrease variance**

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

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

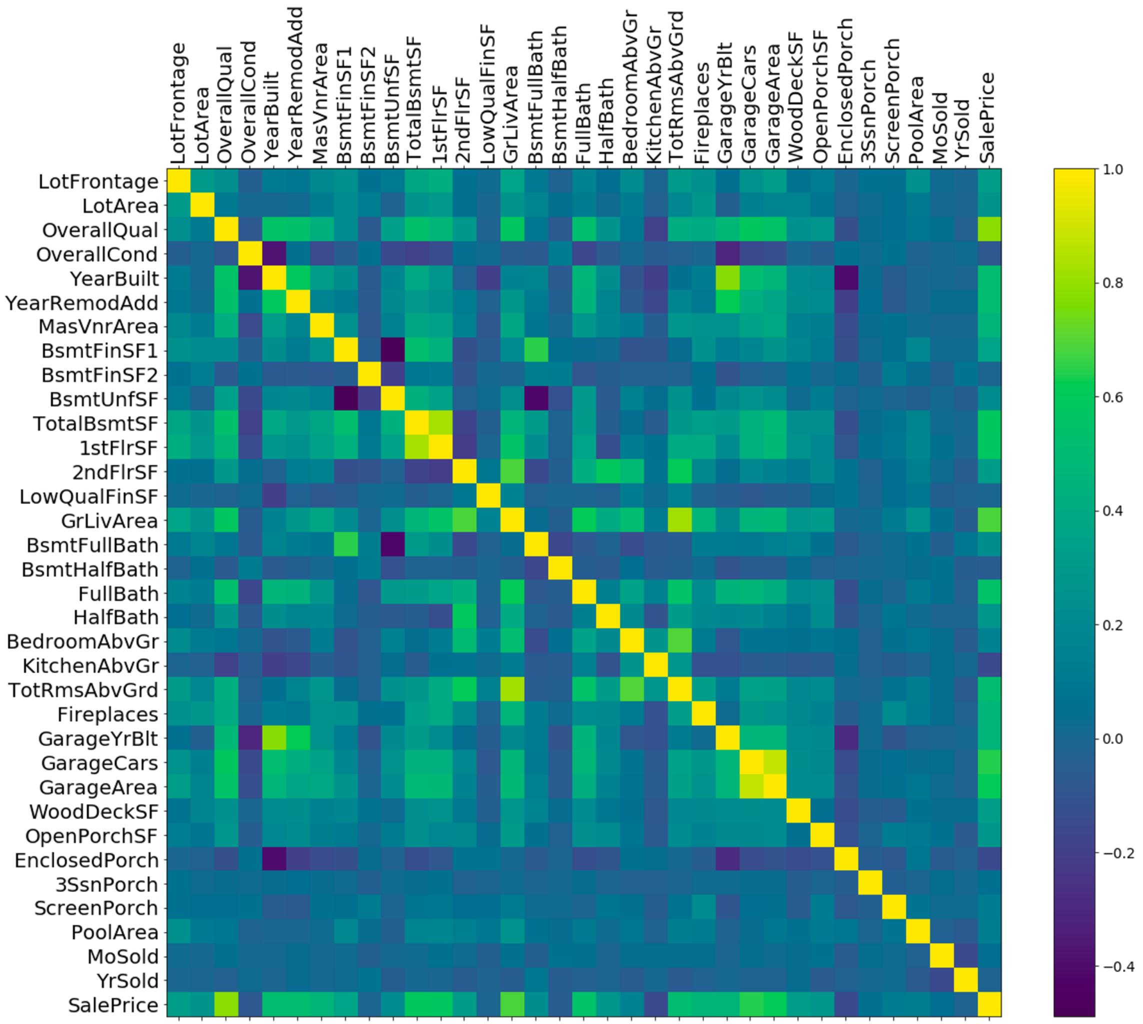
Data from: De Cock. Journal of Statistics Education 19(3), 2011

Housing Dataset

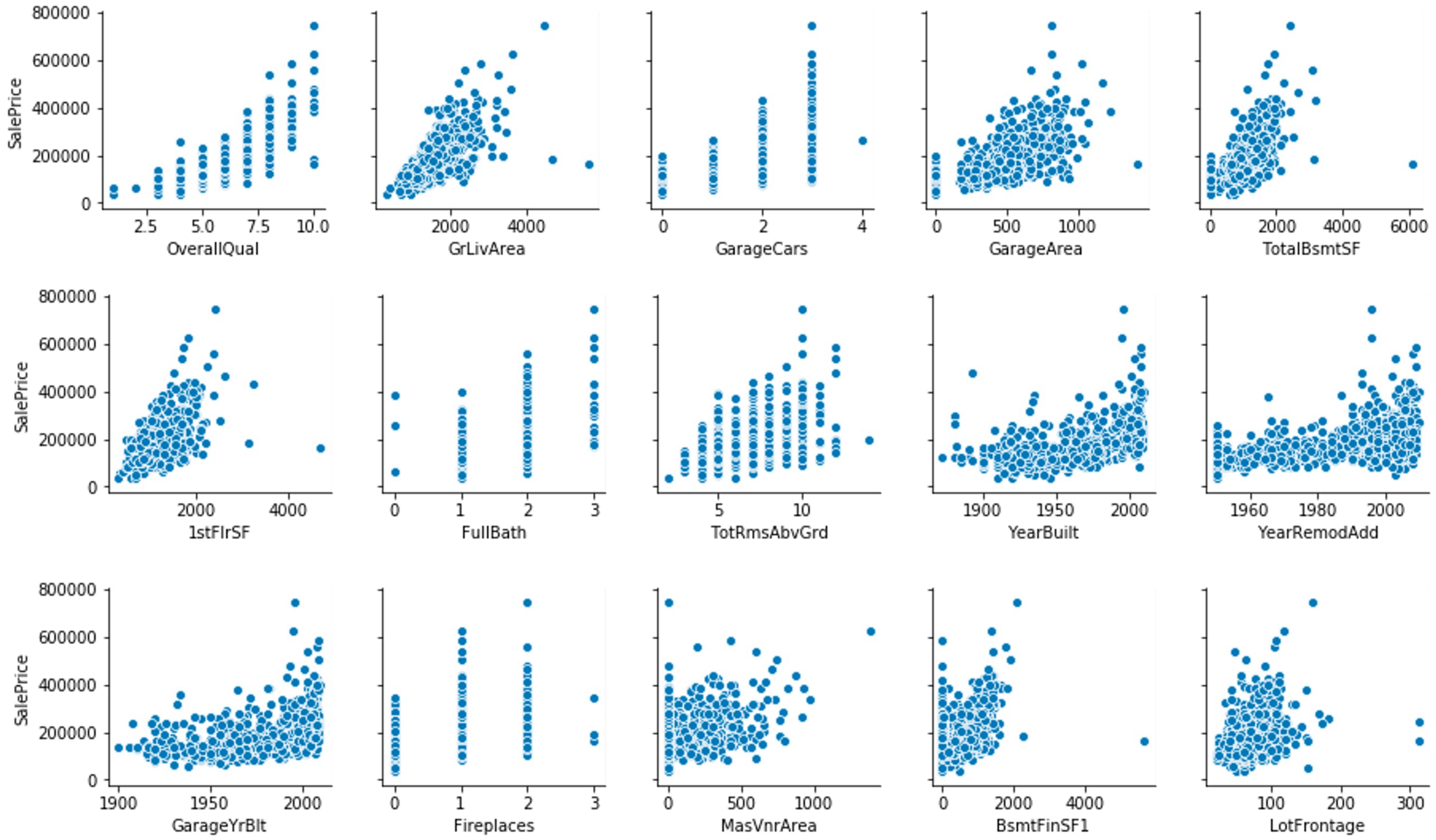
- dataframe.describe()

	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

Feature Correlation Matrix



Features Most Correlated with Label



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