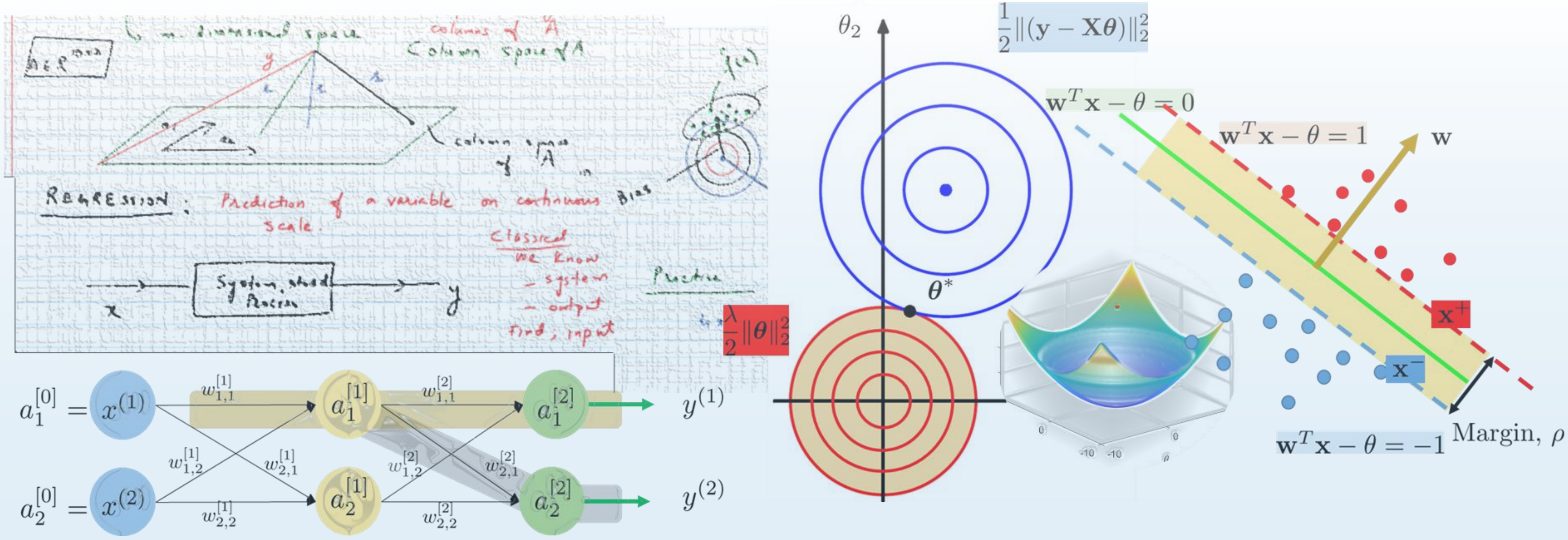


AI-501 Mathematics for AI

Regression, Linear, Polynomial, and Regularization

Zubair Khalid

School of Science and Engineering

LUMS
A Not-for-Profit University

https://www.zubairkhalid.org/ai501_2024.html

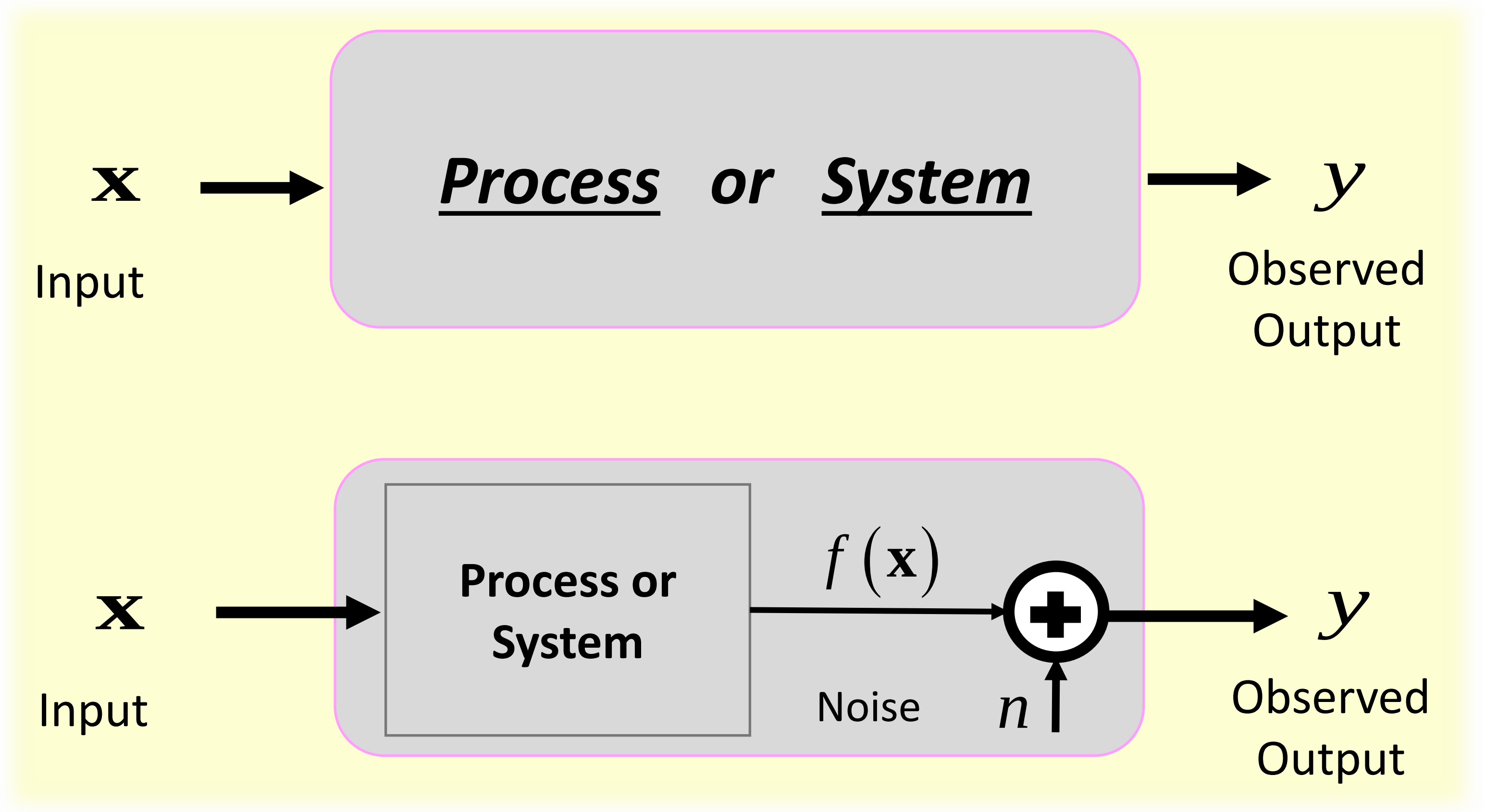
Outline

- Regression Set-up
- Linear Regression
- Polynomial Regression
- Underfitting/Overfitting
- Regularization

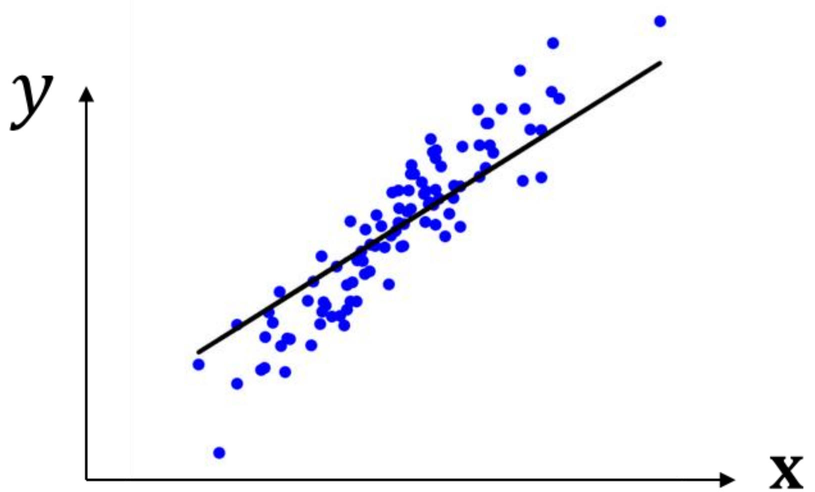
Regression

Regression: Quantitative Prediction on a continuous scale

- *Given a data sample, predict a numerical value*



Example: Linear relationship



Here, PROCESS or SYSTEM refers to any underlying physical or logical phenomenon which maps our input data to our observed and noisy output data.

Regression

Overview:

$\mathbf{x}$ → ***Process*** ***or*** ***System*** → y

Input

Observed Output

One variable regression: y is a scalar

Multi-variable regression: $\underline{\mathbf{y}}$ is a vector

Single feature regression: $\mathbf{x}$ is a scalar

Multiple feature regression: $\underline{\mathbf{x}}$ is a vector

We will cover

Regression

Examples:

Single Feature:

- Predict score in the course given the number of hours of effort per week.
- Establish the relationship between the monthly e-commerce sales and the advertising costs.

Multiple Feature:

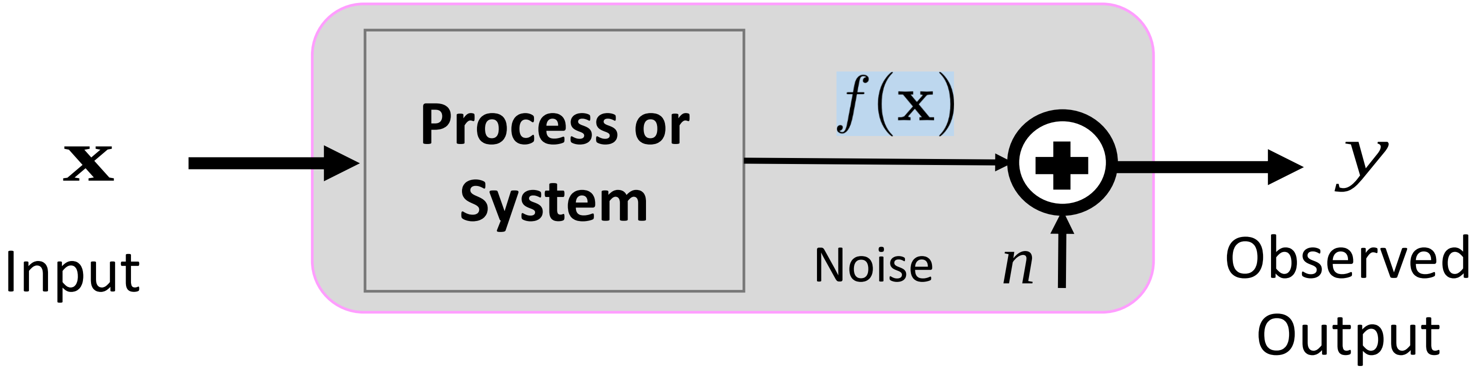
- Studying operational efficiency of machine given sensors (temperature, vibration) data.
- Predicting remaining useful life (RUL) of the battery from charging and discharging information.
- Estimate sales volume given population demographics, GDP indicators, climate data, etc.
- Predict crop yield using remote sensing (satellite images, gravity information).
- Dynamic Pricing or Surge Pricing by ride sharing applications (Uber).
- Rate the condition (fatigue or distraction) of the driver given the video.
- Rate the quality of driving given the data from sensors installed on car or driving patterns.

Regression

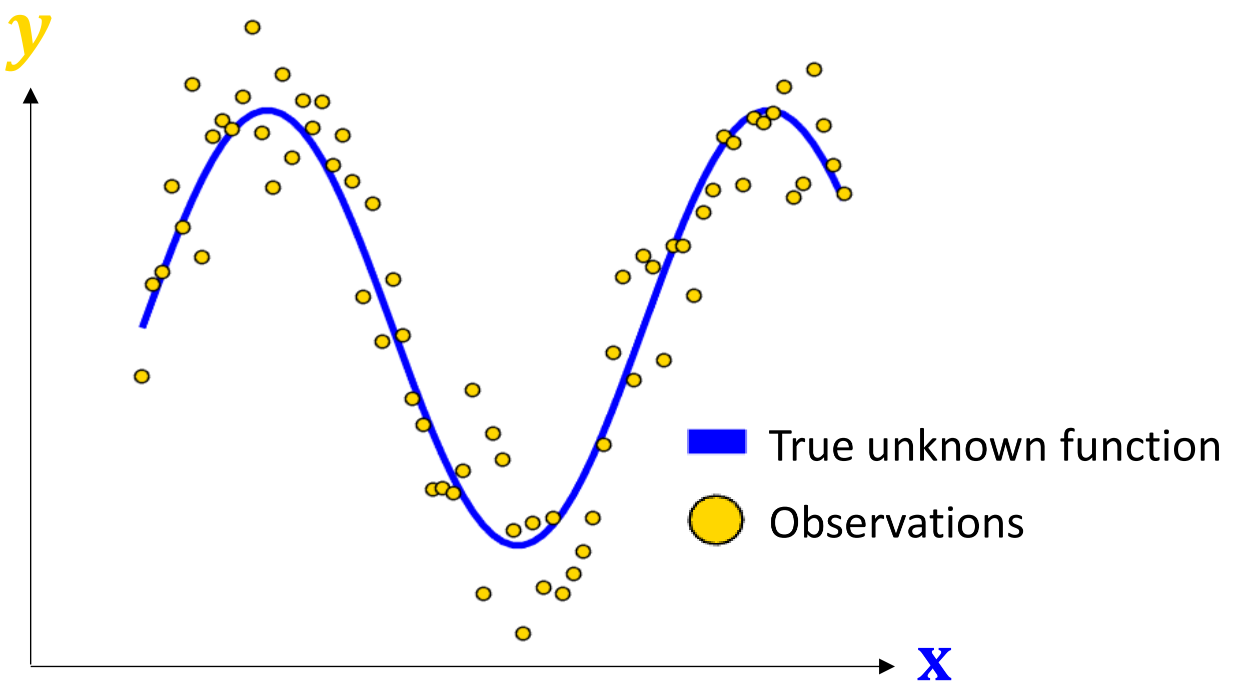
Model Formulation and Setup:

True Model:
We assume there is an inherent but **unknown** relationship between input and output.

$$y = f(\mathbf{x}) + n$$

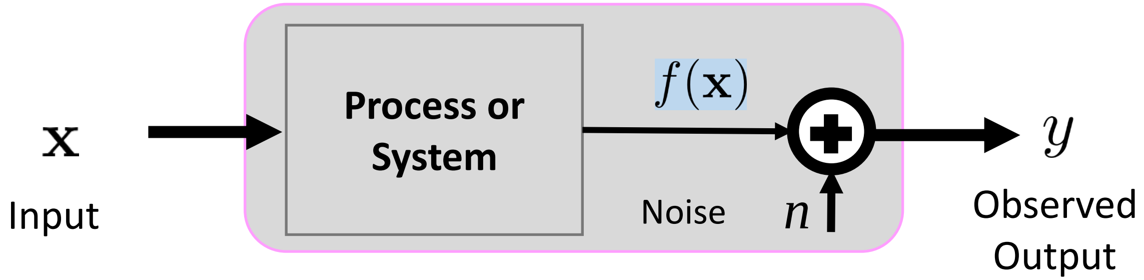


Goal:
Given *noisy observations*, we need to estimate ***the unknown functional relationship*** as accurately as possible.



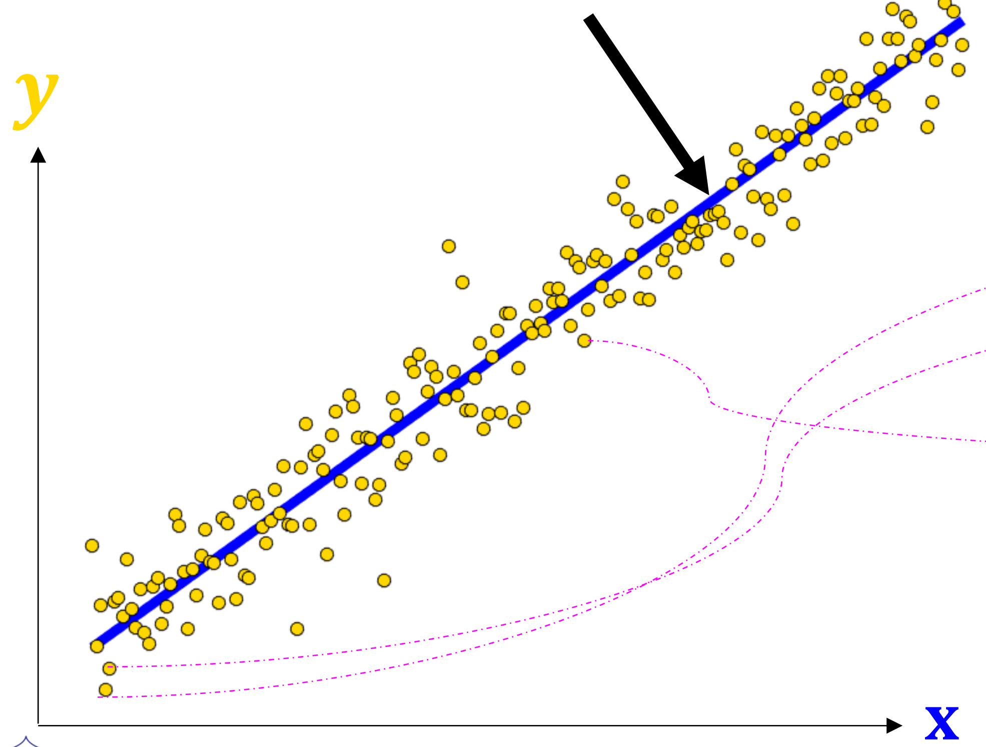
Regression

Model Formulation and Setup:



- Single Feature Regression, Example:

$f(x) = 4.2 + 2.4x$



Training Data

- First Data Sample: $\{\mathbf{x}(1), y(1)\}$
- Second Data Sample: $\{\mathbf{x}(2), y(2)\}$
- ...
- n-th Data Sample: $\{\mathbf{x}(n), y(n)\}$

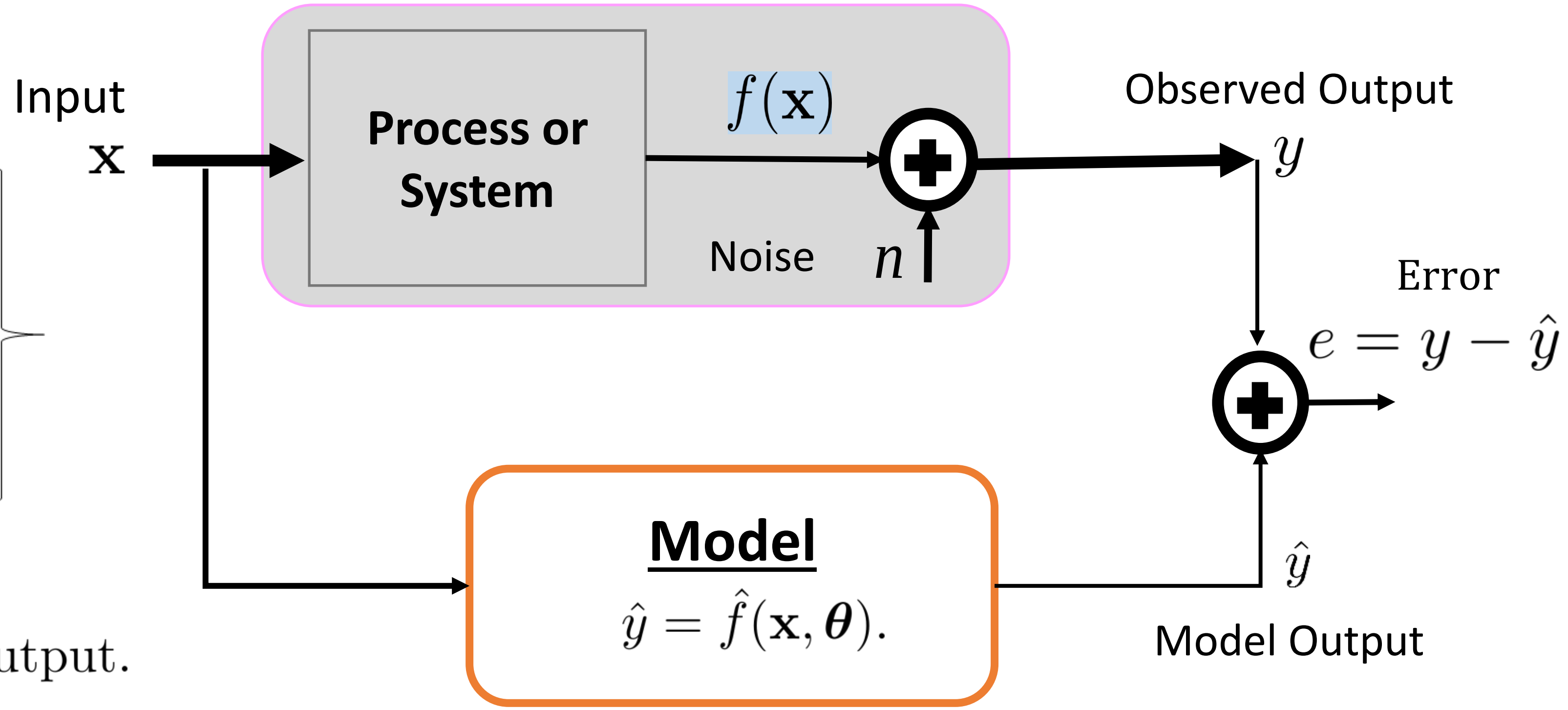
$$D = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\} \subseteq \mathcal{X}^d \times \mathcal{Y}$$

Regression

Model Formulation and Setup:

We have:

$$\left\{ \begin{array}{ll} \text{First Data Sample:} & \{\mathbf{x}(1), y(1)\} \\ \text{Second Data Sample:} & \{\mathbf{x}(2), y(2)\} \\ \dots & \\ n\text{-th Data Sample:} & \{\mathbf{x}(n), y(n)\} \end{array} \right\}$$



- For some input $\mathbf{x}$, $\hat{y}$ is our model output.
- Assume that our model is $\hat{f}(\mathbf{x}, \boldsymbol{\theta})$, characterized by the parameter(s) $\boldsymbol{\theta}$.
- Model $f(\mathbf{x}, \boldsymbol{\theta})$ has
  - A structure (e.g., linear, polynomial, inverse).
  - Paramaters in the vector $\boldsymbol{\theta} = [\theta_1, \theta_2, \dots, \theta_M]$.
- Our model error is $e = y - \hat{y}$.

Linear Regression

Overview:

- Second learning algorithm of the course

- Scalar output is a linear function of the inputs

- Different from KNN: Linear regression adopts a modular approach which we will use most of the times in the course.
  - Select a model
  - Defining a loss function
  - Formulate an optimization problem to find the model parameters such that a loss function is minimized.
  - Employ different techniques to solve optimization problem or minimize loss function.

Linear Regression

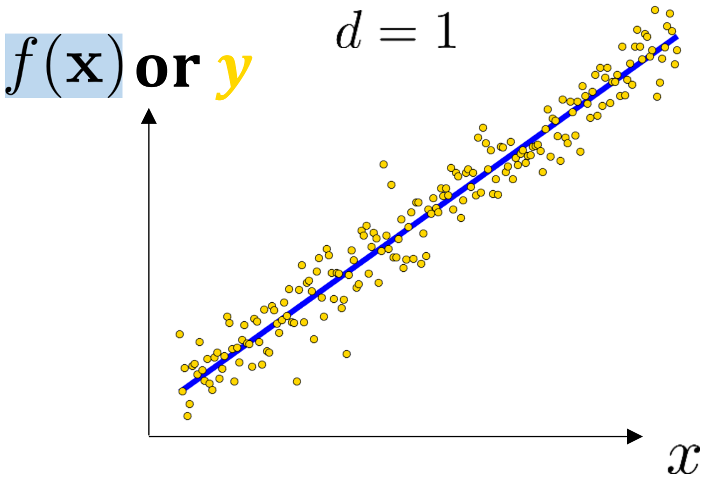
Model:

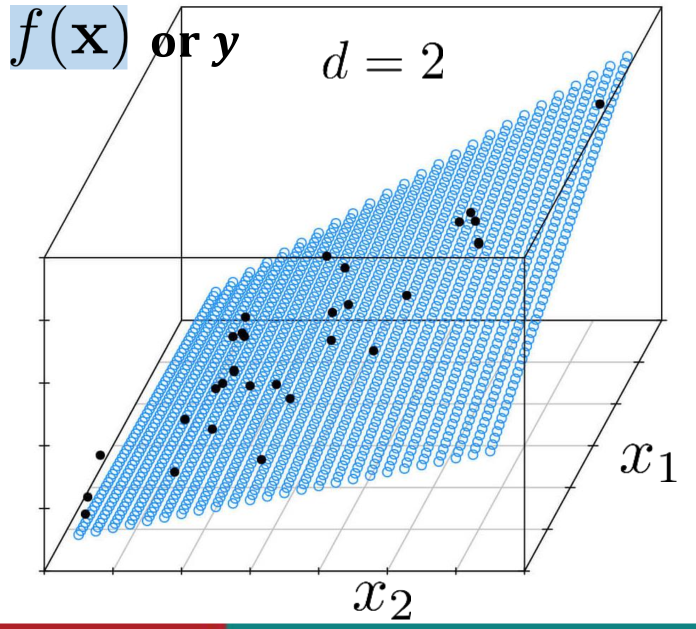
What is Linear?

We have $D = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\} \subseteq \mathcal{X}^d \times \mathcal{Y}$

Interpretation:

- $d = 1$ $\quad \hat{f}(\mathbf{x}, \boldsymbol{\theta}) = \theta_0 + \theta_1 x$ $\quad$ Line.
- $d = 2$ $\quad \hat{f}(\mathbf{x}, \boldsymbol{\theta}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$ $\quad$ Plane.
- d $\quad \hat{f}(\mathbf{x}, \boldsymbol{\theta}) = \theta_0 + \boldsymbol{\theta}^T \mathbf{x}$ $\quad$ Hyper-plane in $\mathbf{R}^{d+1}$





For different θ_0 and $\boldsymbol{\theta}$, we have different hyper-planes.

How do we find the 'best' line?

What do we mean by 'best'?

LUMS
A Not-for-Profit University

Linear Regression

Model:

We have $D = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\} \subseteq \mathcal{X}^d \times \mathcal{Y}$

Model is a linear function of the features, that is,

$$\hat{f}(\mathbf{x}, \boldsymbol{\theta}) = \theta_0 + \sum_{i=1}^{d} \theta_i x_i \quad = \theta_0 + \boldsymbol{\theta}^T \mathbf{x}$$

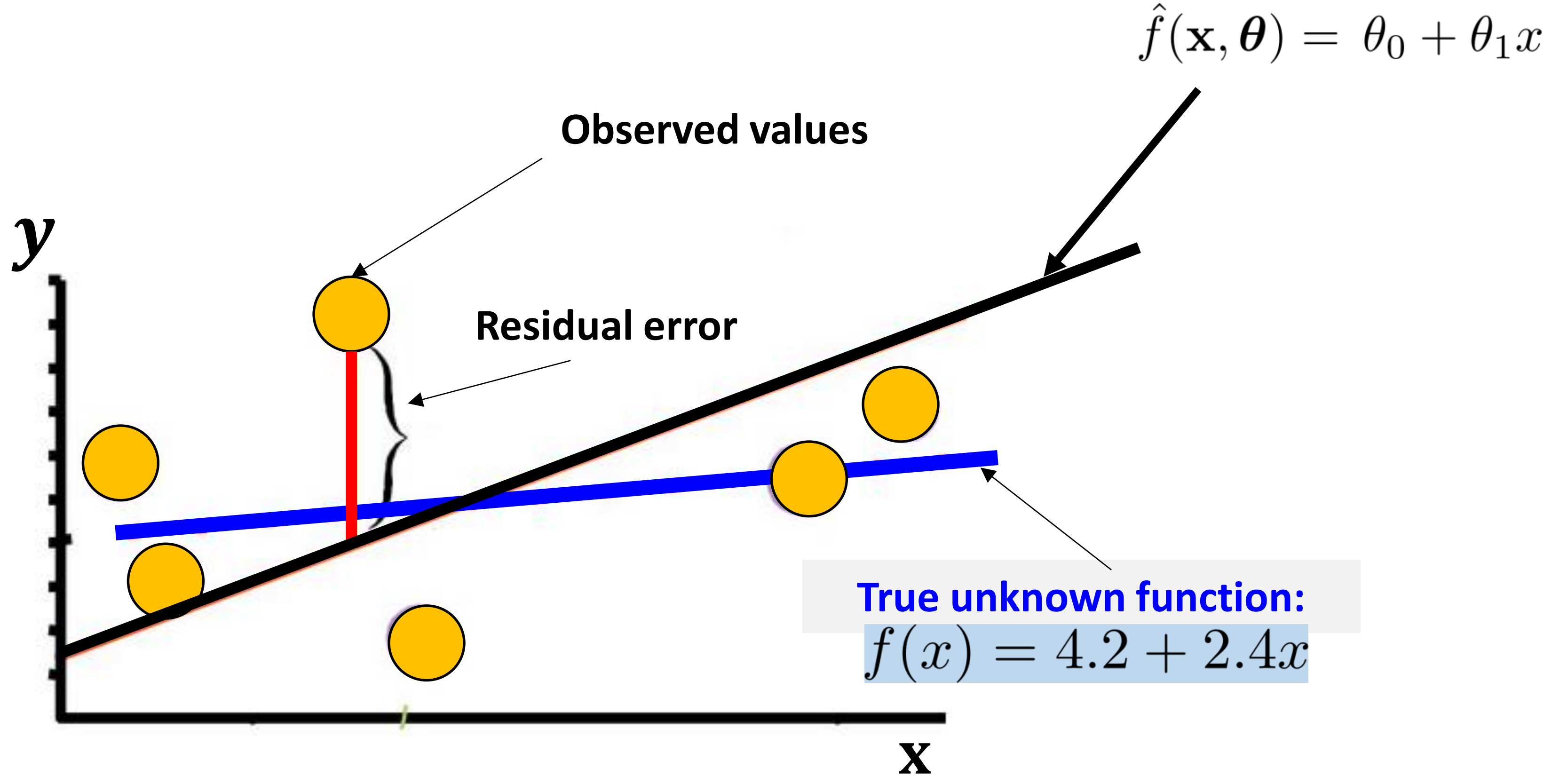
- Linear structure.
- Model Paramaters: θ_0 and $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_d]$.
    - θ_0 is bias or intercept.
    - $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_d]$ represents the weights or slope.
    - θ_i quantifies the contribution of i-th feature x_i.

Linear Regression

Define Loss Function:

- Loss function should be a function of model parameters.

- For input $\mathbf{x}$, our model error is $e = y - \hat{y} = y - \hat{f}(\mathbf{x}, \boldsymbol{\theta}) = y - \theta_0 - \boldsymbol{\theta}^T\mathbf{x}$.
- e is also termed as residual error as it is the difference between observed value and predicted value.
- $d = 1$



Linear Regression

Define Loss Function:

- For $D = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\} \subseteq \mathcal{X}^d \times \mathcal{Y}$, we have

$$e_i = y_i - \theta_0 - \boldsymbol{\theta}^T \mathbf{x_i}, \quad i = 1, 2, \ldots, n$$

- Using residual error, we can define different loss functions:

$$\mathcal{L}(\theta_0, \boldsymbol{\theta}) = \sum_{i=1}^{n} \left(y_i - \theta_0 - \boldsymbol{\theta}^T \mathbf{x_i}\right)^2 \qquad \text{Least-squared error (LSE)}$$

$$\mathcal{L}(\theta_0, \boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \theta_0 - \boldsymbol{\theta}^T \mathbf{x_i}\right)^2 \qquad \text{Mean-squared error (MSE)}$$

$$\mathcal{L}(\theta_0, \boldsymbol{\theta}) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(y_i - \theta_0 - \boldsymbol{\theta}^T \mathbf{x_i}\right)^2} \qquad \text{Root Mean-squared error (RMSE)}$$

– *One* minimizer for all loss functions.

Linear Regression

Define Loss Function:

- We minimize the following loss function: $\mathcal{L}(\theta_0, \boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \theta_0 - \boldsymbol{\theta}^T \mathbf{x_i}\right)^2$

- We have an **optimization problem**: find the parameters which minimize the loss function. We write optimization problem (with no constraints) as

$$\underset{\theta_0, \boldsymbol{\theta}}{\text{minimize}} \quad \mathcal{L}(\theta_0, \boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \theta_0 - \boldsymbol{\theta}^T \mathbf{x_i}\right)^2$$

Factor $\frac{1}{2}$ is added to make the formulation mathematically more convenient.

How to solve?

- Analytically: Determine a critical point that makes the derivtive (if it exists) equal to zero.

- Numerically: Solve optimization using some algorithm that iteratively takes us closer to the critical point minimizing objective function.

Linear Regression

Define Loss Function:

Reformulation:

$$\mathcal{L}(\theta_0, \boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \theta_0 - \boldsymbol{\theta}^T\mathbf{x_i}\right)^2 = \frac{1}{2}\mathbf{e}^T\mathbf{e}$$

Here $\mathbf{e} = [e_1, e_2, \ldots, e_n]^T$ (column vector) where

$$e_i = y_i - \theta_0 - \mathbf{x_i}^T\boldsymbol{\theta}, \quad i = 1, 2, \ldots, n$$

$$\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} - \theta_0 \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} - \begin{bmatrix} \mathbf{x_1}^T \\ \mathbf{x_2}^T \\ \vdots \\ \mathbf{x_n}^T \end{bmatrix}\boldsymbol{\theta} = \underset{\mathbf{y}}{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}} - \underbrace{\begin{bmatrix} 1 & \mathbf{x_1}^T \\ 1 & \mathbf{x_2}^T \\ \vdots & \\ 1 & \mathbf{x_n}^T \end{bmatrix}}_{\mathbf{X}} \underset{\mathbf{w}}{\begin{bmatrix} \theta_0 \\ \boldsymbol{\theta} \end{bmatrix}}$$

$$\Rightarrow \overset{\text{Residual error}}{\mathbf{e}} = \underset{\text{Observations}}{\mathbf{y}} - \underset{\text{Inputs}}{\mathbf{X}}\overset{\text{Model Parameters}}{\mathbf{w}}$$

Consequently:

$$\mathcal{L}(\theta_0, \boldsymbol{\theta}) = \mathcal{L}(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w})$$

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\|(\mathbf{y} - \mathbf{X}\mathbf{w})\|_2^2$$

Linear Regression

Solve Optimization Problem: (Analytical Solution employing Calculus)

$$\underset{\mathbf{w}}{\text{minimize}} \quad \mathcal{L}(\mathbf{w}) = \frac{1}{2}\|(\mathbf{y} - \mathbf{X}\mathbf{w})\|_2^2$$

- Very beautiful, elegant function we have here!

We first write the loss function as

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w})$$

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\left(\mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{w} - \mathbf{w}^T\mathbf{X}^T\mathbf{y} + \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w}\right)$$

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\left(\mathbf{y}^T\mathbf{y} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{y} + \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w}\right)$$

- To further solve this, let us quickly talk about the concept of a gradient of a function.

Linear Regression

Solve Optimization Problem: (Analytical Solution employing Calculus)

Gradient of a function: Overview

- For a function $f(\mathbf{x})$ that maps $\mathbf{x} \in \mathbf{R}^d$ to $\mathbf{R}$, we define a gradient (direec-tional derivative) with respect to $\mathbf{x}$ as
$$\nabla f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_d}\right]^T \in \mathbf{R}^d$$

- Interpretation: Quantifies the rate of change along different directions.

Examples:

- $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x} = \mathbf{x}^T\mathbf{a}$

  $\nabla f(\mathbf{x}) = \mathbf{a}$

- $f(\mathbf{x}) = \mathbf{x}^T\mathbf{x}$

  $\nabla f(\mathbf{x}) = 2\mathbf{x}$

- $f(\mathbf{x}) = \mathbf{x}^T\mathbf{P}\mathbf{x}$

  $\nabla f(\mathbf{x}) = 2\mathbf{P}\mathbf{x}$

Linear Regression

Solve Optimization Problem: (Analytical Solution employing Calculus)

We have a loss function: $$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\big(\mathbf{y}^T\mathbf{y} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{y} + \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w}\big)$$

- Take gradient with respect to $\mathbf{w}$ as

$$\nabla\mathcal{L}(\mathbf{w}) = \frac{1}{2}\big(-2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\mathbf{w}\big)$$

- Substituting it equal to zero yields

$$\mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{y}$$

$$\Rightarrow \mathbf{w} = \big(\mathbf{X}^T\mathbf{X}\big)^{-1}\mathbf{X}^T\mathbf{y}$$

- We have determined the weights for which LSE, MSE, RMSE or the norm of the residual is minimized.

- This solution is referred to as least-squared solution as it minimizes the squared error.

Linear Regression

So far and moving forward:

- We assumed that we know the structure of the model, that is, there is a linear relationship between inputs and output.
- Number of parameters = dimension of the feature space + 1 (bias parameter)
- Formulated loss function using residual error.
- Formulated optimization problem and obtain analytical solution.
- Linear regression is one of the models for which we can obtain an analytical solution.
- We will shortly learn an algorithm to solve optimization problem numerically.

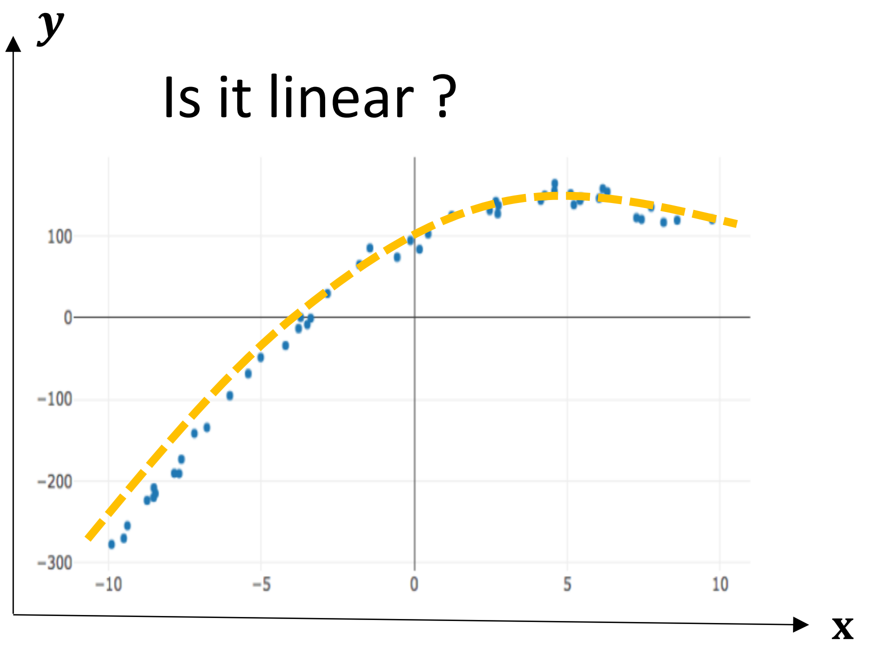
Outline

- Regression Set-up
- Linear Regression
- Polynomial Regression
- Underfitting/Overfitting
- Regularization

Polynomial Regression

Overview:

- *If the relationship between the inputs and output is not linear, we can use a polynomial to model the relationship.*
- *We will formulate the polynomial regression model for single feature regression problem.*
- *Polynomial Regression is often termed as* ***Non-linear Regression*** *or* ***Linear in Parameter Regression.***
- *We will also revisit the concept of 'over-fitting'.*



Polynomial Regression

Single Feature Regression:

Formulation:

- $d = 1$, input x is a scalar.
- Model is a polynomial function of the input, that is,

$$\boxed{\hat{f}(x, \boldsymbol{\theta}) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_M x^M = \sum_{i=0}^{M} \theta_i x^i}$$

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_M \end{bmatrix}$$

- M is the degree of plynomial; characterized by $M+1$ coefficients $\theta_0, \theta_1, \ldots, \theta_M$.

- M is the Hyper-Parameter of the model and determines the complexity of the model. For $M = 1$, we have a linear regression.

- We can use linear regression to find these coefficients by formulating the input x and its powers using a vector-valued function given by

$$\boldsymbol{g}(x) = \left[1,\ x,\ x^2,\ \cdots,\ x^M\right]^T$$

Polynomial Regression

Single Feature Regression:

Formulation:

- With this notation, we can formulate model as $\boxed{\hat{f}(x, \boldsymbol{\theta}) = \boldsymbol{g}(x)^T \boldsymbol{\theta}}$
- Note that the model is linear in terms of parameters due to which Polynomial Regression is termed as Linear in Parameter Regression.
- Note that $\boldsymbol{g}(x)$ can be any function of x. For example, we can have $\boldsymbol{g}(x) = \left[\frac{1}{x}, \sin(2\pi x), x^2, e^x \dots\right]^T$
- For n data points (input, output), we can define residual error in a similar way we computed for linear regression as follows:

$$\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}}_{\mathbf{y}} - \underbrace{\begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^M \\ 1 & x_2 & x_2^2 & \dots & x_2^M \\ \vdots & \vdots & \vdots & \vdots & \\ 1 & x_n & x_n^2 & \dots & x_n^M \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_M \end{bmatrix}}_{\boldsymbol{\theta}} \Rightarrow \mathbf{e} = \mathbf{y} - \mathbf{X}\boldsymbol{\theta}$$

$$\boldsymbol{\theta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

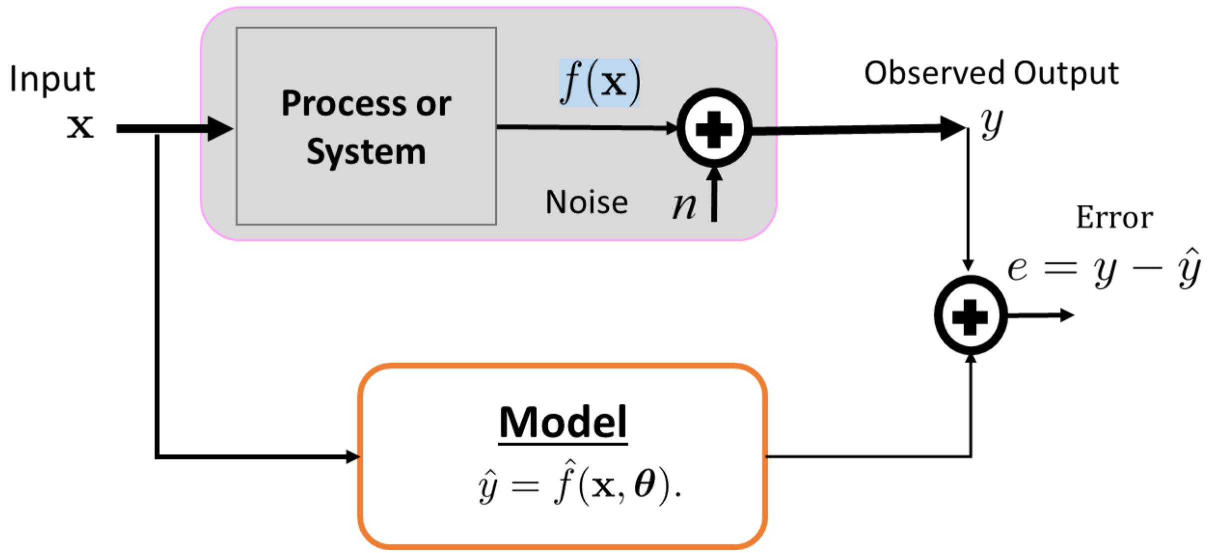
We have seen this before.
&
We are capable to solve this!

Polynomial Regression

Single Feature Regression:

Example (Ref: CB. Section 1.1):

- Model is a polynomial function of degree M.
- If M is not knwown, how do we choose it?

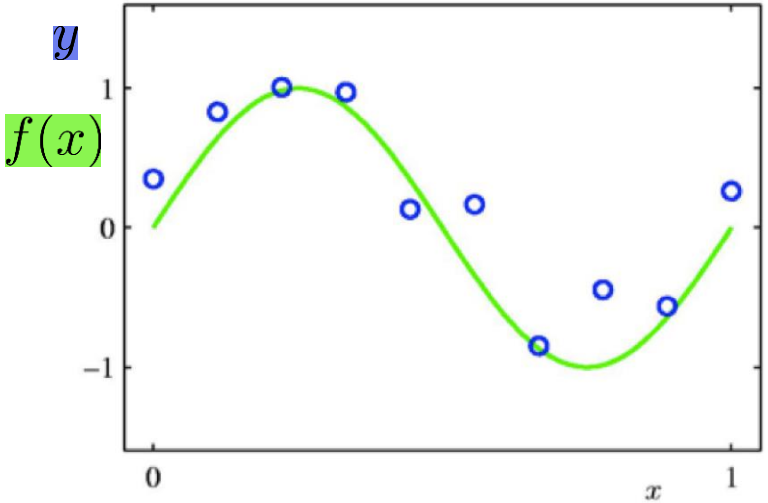


Process $f(x) = \sin(2\pi x)$

Observations $y = f(x) + n$

Model $\hat{f}(x, \boldsymbol{\theta}) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_i x^M$

- We take $n = 10$.



(CB) Pattern Recognition and Machine Learning, Christopher M. Bishop

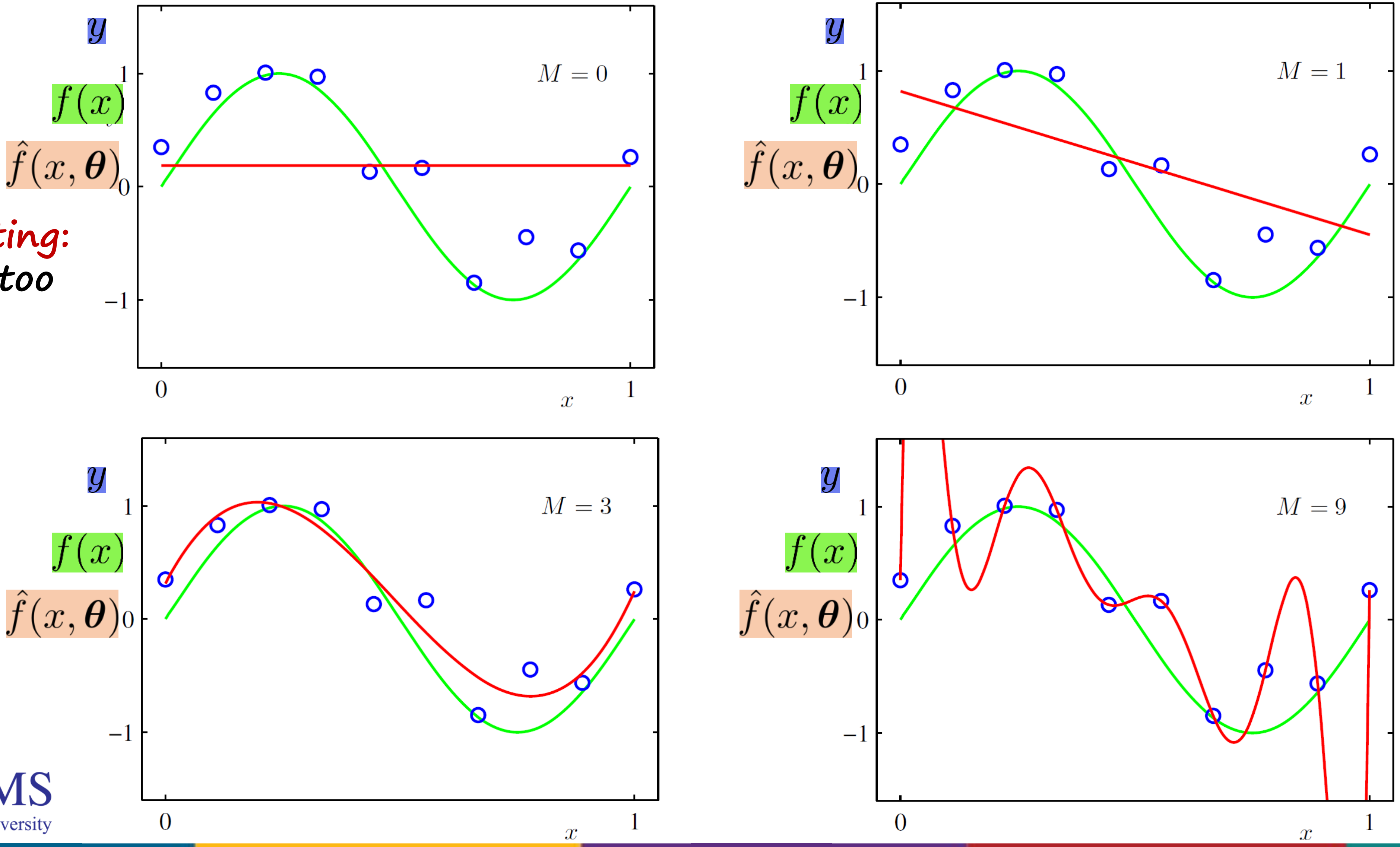
Polynomial Regression

Single Feature Regression: $f(x) = \sin(2\pi x)$

Example:

$$\hat{f}(x, \boldsymbol{\theta}) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_i x^M$$

Underfitting: Model is too simple



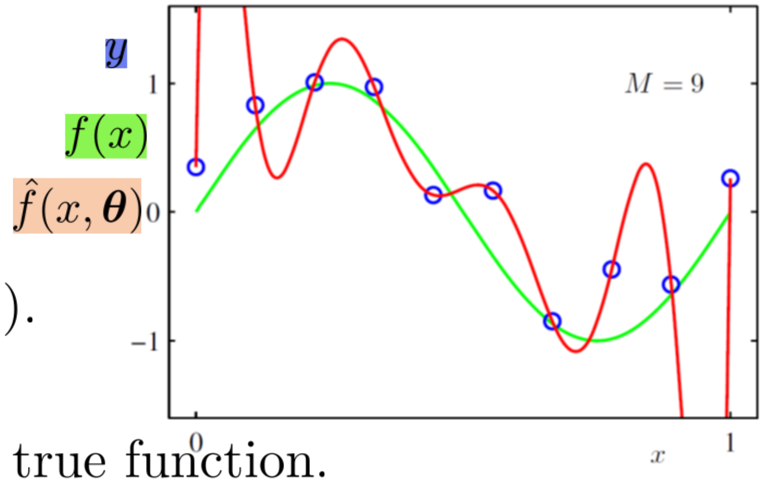
Overfitting: Model is too complex

Polynomial Regression

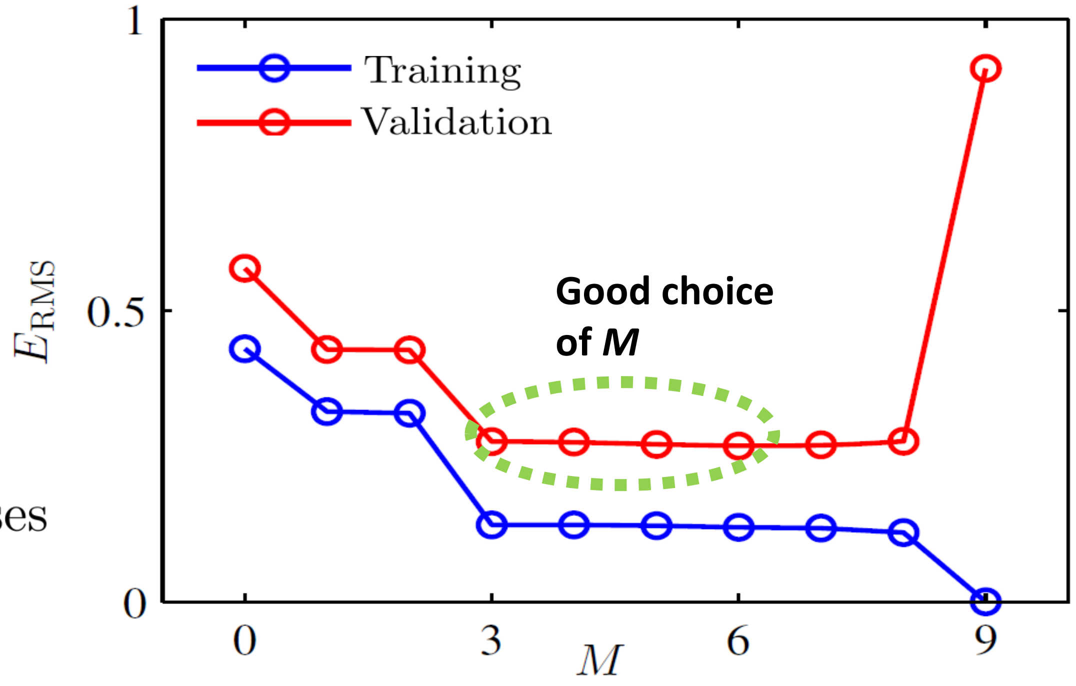
Single Feature Regression:

Example:

- What's happening with the increase in M? *Overfitting*
  - Model is fitting to the data, not the actual true function.
  - For $M = 9$, we have zero residual error, that is, $y = \hat{f}(x, \boldsymbol{\theta})$.
  - Is this a good solution?
  - No! The model is oscillating wildly and is not close to the true function.



- In this toy example, we had informtion about the true function and therefore we can conclude that $M = 9$, is not a good model to fit the data.
- How to choose model order M or How do we tell if a model is overfitting when we do not have knowledge about the true process/function?



Solution 1:

- Recall: Train-Validation Split. Overfitting causes poor generalization performance, that is, large error on the testing or validation data.

LUMS
A Not-for-Profit University

Polynomial Regression

Single Feature Regression:

Example:

- Let's pose another question!
- $M = 3$ degree polynomial is a special case of $M = 9$ degree polynomial.
    - Why $M = 9$ gives us poor performance?
- Coefficients magnitude increases with M.
- $M = 3$ solution cannot be recovered from $M = 9$ soltuion by setting the remaining weights equal to zero.
- 10 coefficients are tuned for 10 data-points when $M = 9$.

$\boldsymbol{\theta} =$	$M=0,$	$M=1,$	$M=3,$	$M=9$
θ_0	0.19	0.82	0.31	0.35
θ_1		-1.27	7.99	232.37
θ_2			-25.43	-5321.83
θ_3			17.37	48568.31
θ_4				-231639.30
θ_5				640042.26
θ_6				-1061800.52
θ_7				1042400.18
θ_8				-557682.99
θ_9				125201.43

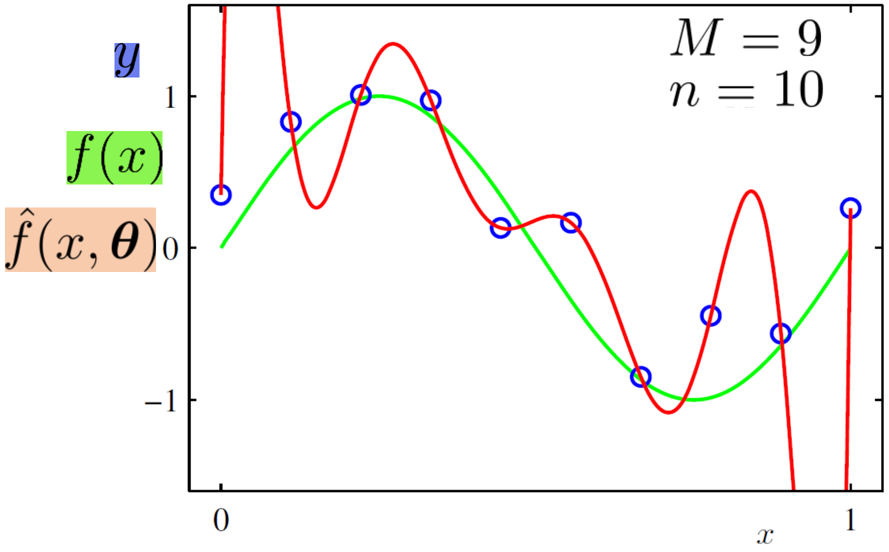
Polynomial Regression

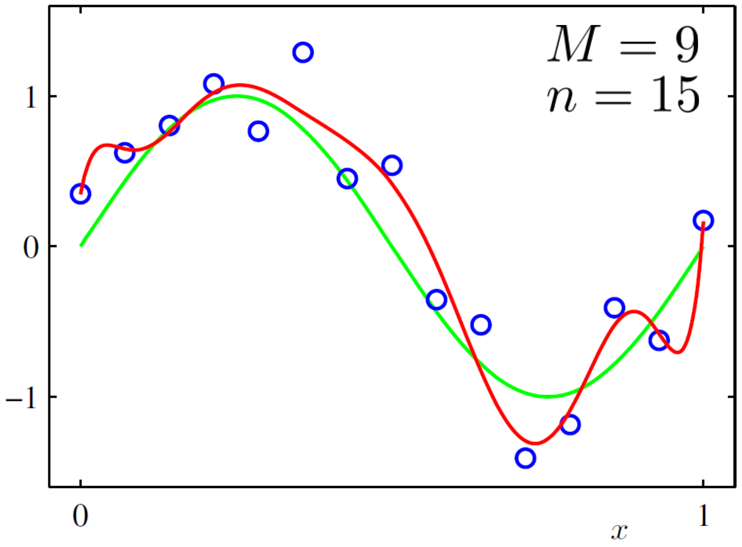
Single Feature Regression:

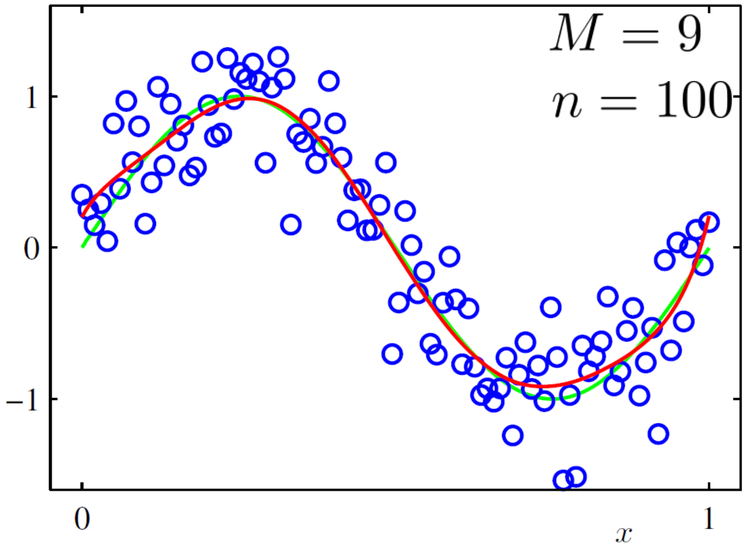
How to Handle Overfitting?

- The polynomial degree M is the hyper-parameter of our model, like we had k in kNN, and controls the complexity of the model.
- If we stick with M=3 model, this is the restriction on the number of parameters.
- We encounter overfitting for M=9 because we do not have sufficient data.

Solution 2: Take more data points to avoid over-fitting.







LUMS
A Not-for-Profit University

Solution 3: Regularization

Outline

- Regression Set-up
- Linear Regression
- Polynomial Regression
- Underfitting/Overfitting
- Regularization

Regularization

Regularization overview:

- The concept is broad but we will see in the context of linear regression or polynomial regression which we formulated as linear regression.
- Encourages the model coefficients to be small by adding a penalty term to the error.
- We had the loss function of the following form that we minimize to find the coefficients:

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\|(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})\|_2^2$$

See linear regression formulation.

- We add a 'penalty term', known as regularizer, in the loss function as

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \mathcal{L}_{\text{reg}}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \lambda\mathcal{R}(\boldsymbol{\theta})$$

Regularized Loss function

Regularizer

- $\lambda \geq 0$ maintains the trade-off between regularizer and the original loss function as it controls the relative importance of the regulrization term.

Regularization

L^2 Least-squares Regularization – Ridge Regression:

- Since we require to discourage the model coefficients from reaching large values; we can use the following simple regularizer:

$$\mathcal{R}(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\theta}\|_2^2$$

Known as L^2 or ℓ^2 penalty

- For this choice, regularized loss function becomes

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \mathcal{L}_{\text{reg}}(\boldsymbol{\theta}) = \frac{1}{2}\|(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2$$

- This regularization term maintains a trade-off between 'fit of the model to the data' and 'square of norm of the coefficients'.
  - If model is fitted poorly, the first term is large.
  - If coefficients have high values, the second term (penalty term) is large.

- Large λ penalizes coefficient values more.

Intuitive Interpretation: We want to minimize the error while keeping the norm of the coefficients bounded.

Regularization

L^2 Least-squares Regularization – Ridge Regression:

- Regularized loss function is still quadratic, and we can find closed form solution.

We have a loss function: $\mathcal{L}_{\text{reg}}(\boldsymbol{\theta}) = \frac{1}{2}\|(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2$

- Take gradient with respect to $\boldsymbol{\theta}$ as

$$\nabla\mathcal{L}_{\text{reg}}(\boldsymbol{\theta}) = \frac{1}{2}\big(- 2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} + 2\lambda\boldsymbol{\theta}\big)$$

- Substituting it equal to zero yields

$$\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} + \lambda\boldsymbol{\theta} = \mathbf{X}^T\mathbf{y} \;\Rightarrow\; \big(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\big)\boldsymbol{\theta} = \mathbf{X}^T\mathbf{y}$$

$$\Rightarrow \boldsymbol{\theta} = \big(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\big)^{-1}\mathbf{X}^T\mathbf{y}$$

- We have a solution of the ridge regression:

$$\boldsymbol{\theta}(\lambda) = \big(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\big)^{-1}\mathbf{X}^T\mathbf{y}$$

- $\lambda = 0$, we have non-regularized solution.
- $\lambda = \infty$, the solution is a zero vector.

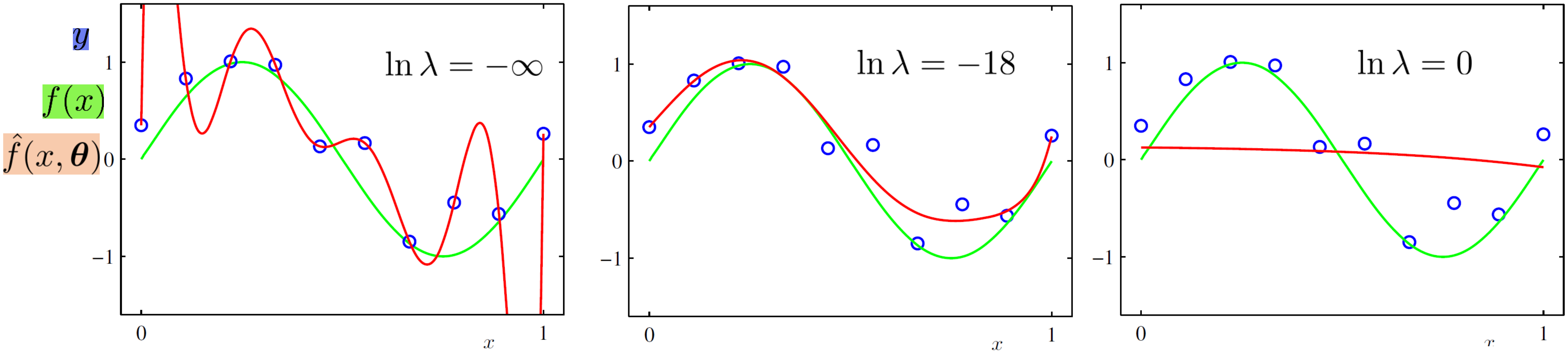
Regularization

L^2 Least-squares Regularization – Ridge Regression:

Example:

- Too small λ: no regularization.
- Too large λ: no weightage to the data.

- In practice, we use very small value of λ and therefore it is convenient to work with $\ln\lambda$ and compute it as $\lambda = e^{\ln\lambda}$.



No regularization

Too much regularization

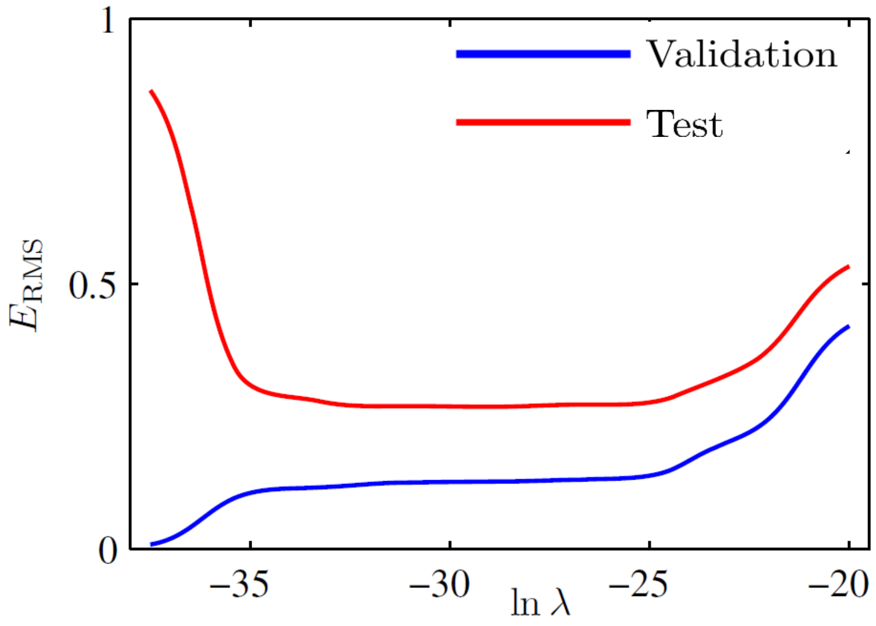
Regularization

L^2 Least-squares Regularization – Ridge Regression:

Example:

- λ restricts the coefficients from exploding as we have included the square of the norm of the coefficients in the loss function being minimized.

$\boldsymbol{\theta} =$	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
θ_0	0.35	0.35	0.13
θ_1	232.37	4.74	-0.05
θ_2	-5321.83	-0.77	-0.06
θ_3	48568.31	-31.97	-0.05
θ_4	-231639.30	-3.89	-0.03
θ_5	640042.26	55.28	-0.02
θ_6	-1061800.52	41.32	-0.01
θ_7	1042400.18	-45.95	-0.00
θ_8	-557682.99	-91.53	0.00
θ_9	125201.43	72.68	0.01



- λ is a hypermater of the model and we learn it in practice using the validation data.

Regularization

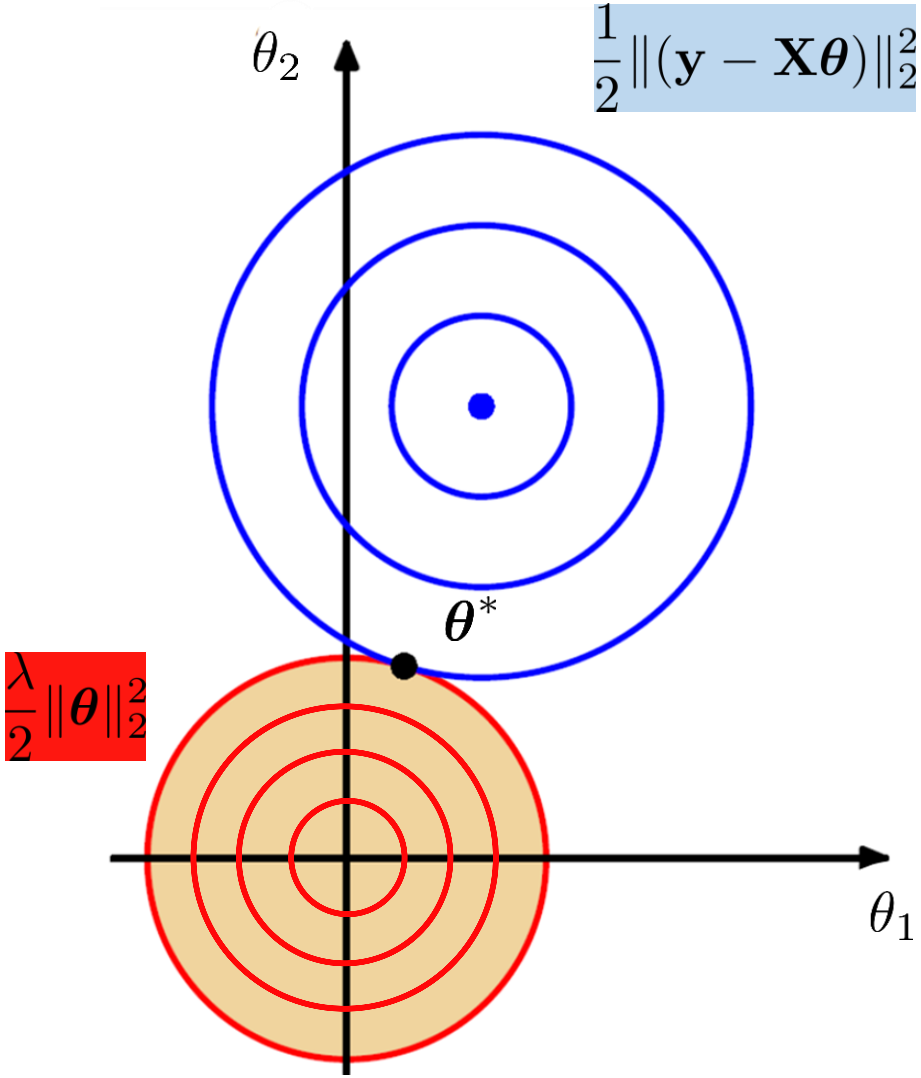
L^2 Least-squares Regularization – Ridge Regression:

Graphical Visualization:

$\boldsymbol{\theta} = [\theta_1, \theta_2]$, we assume we have two coefficients: θ_1 and θ_2.

We have a loss function: $\mathcal{L}_{\text{reg}}(\boldsymbol{\theta}) = \frac{1}{2}\|(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2$

- Good value of λ helps us in avoiding overfitting.
- Irrelevant features get small but non-zero value in the regularized solution.
- Ideally, we would like to assign zero weight to the irrelevant features.



Regularization

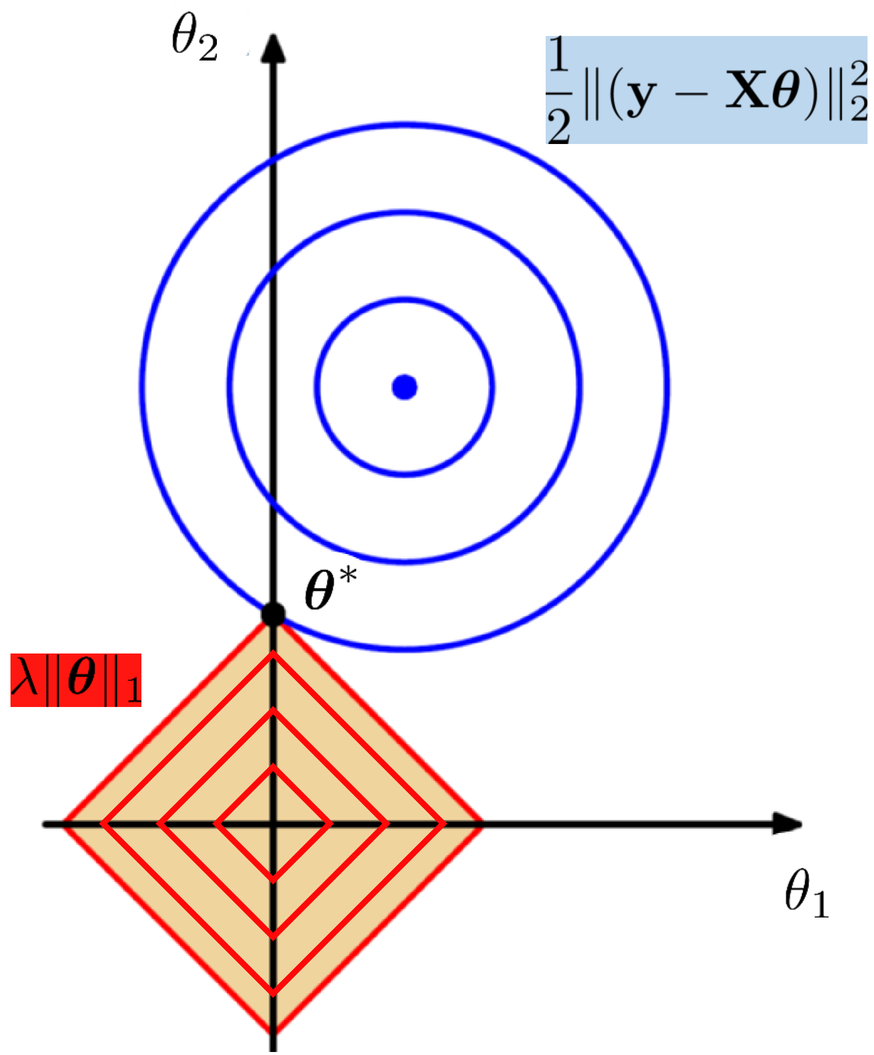
L^1 Least-squares Regularization – Lasso Regression

- Use L^1 or ℓ^1 penalty instead, that is, $\mathcal{R}(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_1 = \sum_i |x_i|$
- For this choice, regularized loss function becomes

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \mathcal{L}_{\text{reg}}(\boldsymbol{\theta}) = \frac{1}{2}\|(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})\|_2^2 + \lambda\|\boldsymbol{\theta}\|_1$$

- This regularization is referred to as least absolute shrinkage and selection operator (Lasso).
- The intersection is at the corners of the diamond.
    - Lasso regression gives us sparse solution.

Graphical Visualization:



Regularization

Elastic Net Regression, L^1 vs L^2

- Ridge: Error + λ times (sum of squares of coefficients)
- Lasso: Error + λ times (sum of absolute values of the coefficients)
- Lasso optimization: computationally expensive than ridge regression.
- Due to the corners included in the solution, regularized solution will have some weights qual to zero.
  - Solution is sparse in general, and is therefore biased.
- **Elastic Net Regression:** Hybrid version; both L_1 and L_2 penalties.

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \mathcal{L}_{\text{reg}}(\boldsymbol{\theta}) = \frac{1}{2}\|(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})\|_2^2 + \lambda_1\|\boldsymbol{\theta}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\theta}\|_2^2$$

- Ridge and Lasso are special cases of elastic net regression.
- Combines the strength of both but require tuning of hyperparameters λ_1 and λ_2 using validation data.