Linear Regression

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

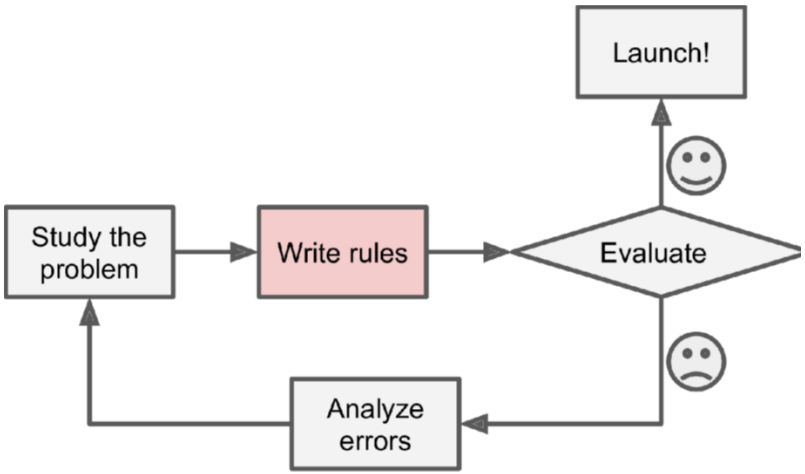
KAUST Academy
King Abdullah University of Science and Technology

July 30, 2025

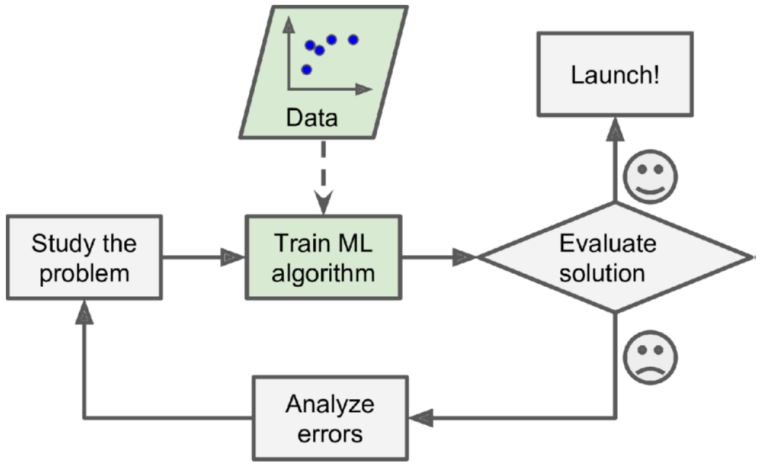
Outline

- Introduction to ML
- Linear Regression
- Optimization
- Linear Regression: Probabilistic Interpretation
- Bias-Variance Tradeoff
- Regularization

- **Machine Learning** is the science (and art) of programming computers so they can learn from data.



The traditional approach



The Machine Learning approach

Data Types

- **Tabular Data** (e.g., spreadsheets, databases)
  - Note: Columns are called **Features**. Rows are called **Samples**.
- **Time-Series Data** (e.g., stock prices, weather forecasts, IoT sensor data)
- **Text Data** (Natural Language Processing, e.g., emails, social media posts, documents)
- **Images and Videos** (Computer Vision, e.g., medical imaging, surveillance, facial recognition)
- **Audio Data** (Speech Recognition, Music Processing, e.g., voice commands, podcasts, sound classification)

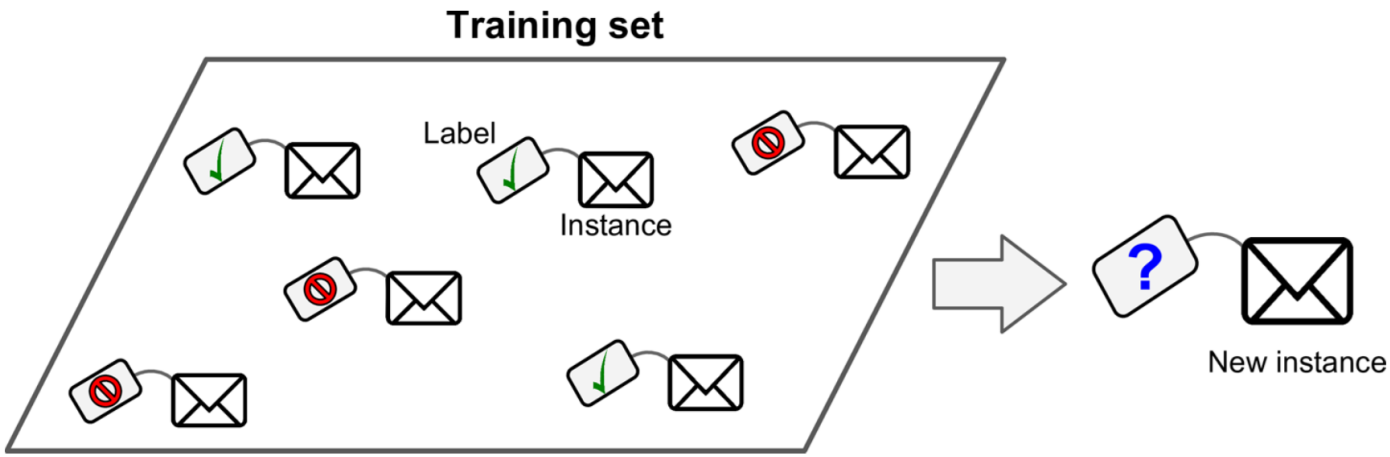
ML Algorithm Types

- **Supervised**: The algorithm learns from labeled data.
  - **Regression**: Predict continuous value (e.g. house prices).
  - **Classification**: Predict discrete value (e.g. spam/not-spam).
- **Unsupervised**: The algorithm works on unlabeled data. We are interested in things like:
  - **Clustering**: Grouping
  - **Dimensionality Reduction**: Reducing the Dimensions
  - **Anomaly Detection**: Detecting outliers
- **Reinforcement Learning**: involves learning to make decisions by interacting with an environment.
  - **Reward Signal**: The agent receives feedback in the form of rewards or penalties, guiding its learning.
  - **Policy**: A strategy the agent learns to decide actions based on the current state.

ML Algorithm Types (cont.)

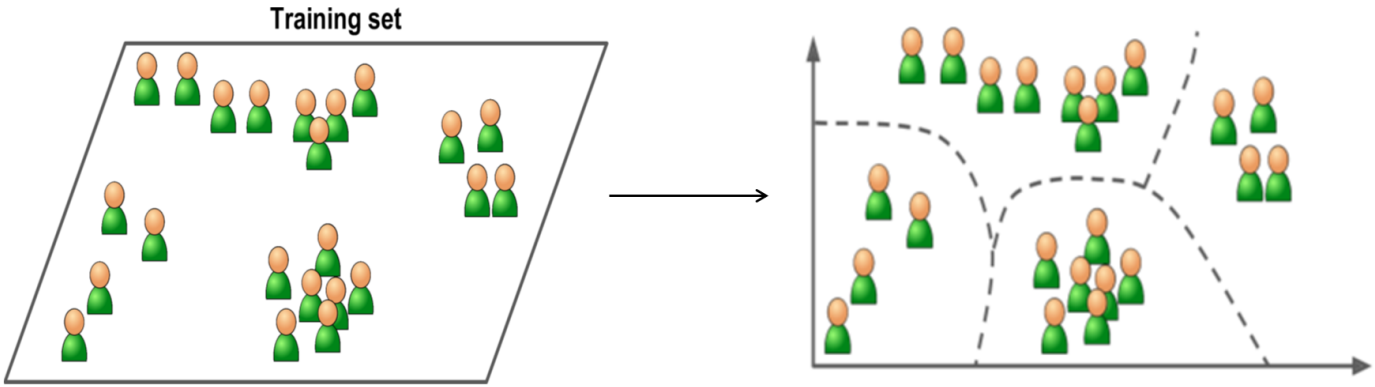
- **Value Function**: An estimate of the expected cumulative reward from a state or state-action pair.
- **Exploration vs Exploitation**: The agent balances exploring new actions to discover rewards and exploiting known actions to maximize them.
- Really popular in video games and robotics! (Also recently in LLMs, see RLHF)

Example: Spam Classification



An example of Supervised Learning: Spam Classification

Example: Clustering



An example of Unsupervised Learning: Clustering

How Does ML Work?

- Most of the ML systems consist of three main components:
- **Hypothesis (Model)**: The function that approximates the target.
  - E.g. Linear Regression, Logistic Regression, SVM, Decision Trees, NN, ...
- **Optimizer**: The mechanism for improving predictions of our model.
- **Loss Function**: The measure of how wrong the predictions are.

How Does ML Work?

- How are they related to each other? 🤔

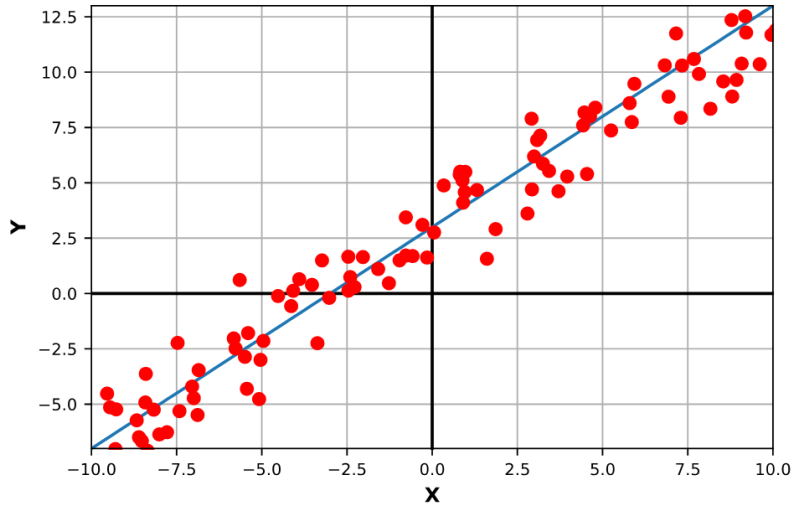
How Does ML Work?

- We firstly define our task (classification/regression) then choose an appropriate **model**.
- We will use an **optimization method** to minimize the **loss function**.
- Reached a minima?
  - Model is making the least possible number of mistakes.
  - **Model trained** 🎉.

Linear Regression: Motivation

- Linear Regression is "still" one of the more widely used ML/DL Algorithms
- Easy to understand and implement
- Efficient to Solve
- We will use Linear Regression to understand the concepts of:
  - Data
  - Models
  - Loss
  - Optimization

Simple Linear Regression



Model (*Linear*): $Y = mX + b$ $\theta = \{m, b\}$

- Y: Response Variable
- X: Covariate / Independent variable / Regressors
- m: slope
- b: bias

Simple Linear Regression

- **Hypothesis:** $\hat{y}_i = mx_i + b$
- **Input:** data (x_i, y_i), $i \in \{1, 2, \ldots, N\}$
  - (e.g., house size x and price y)
- **Goal:** learn values of variable (m, b)

Notation

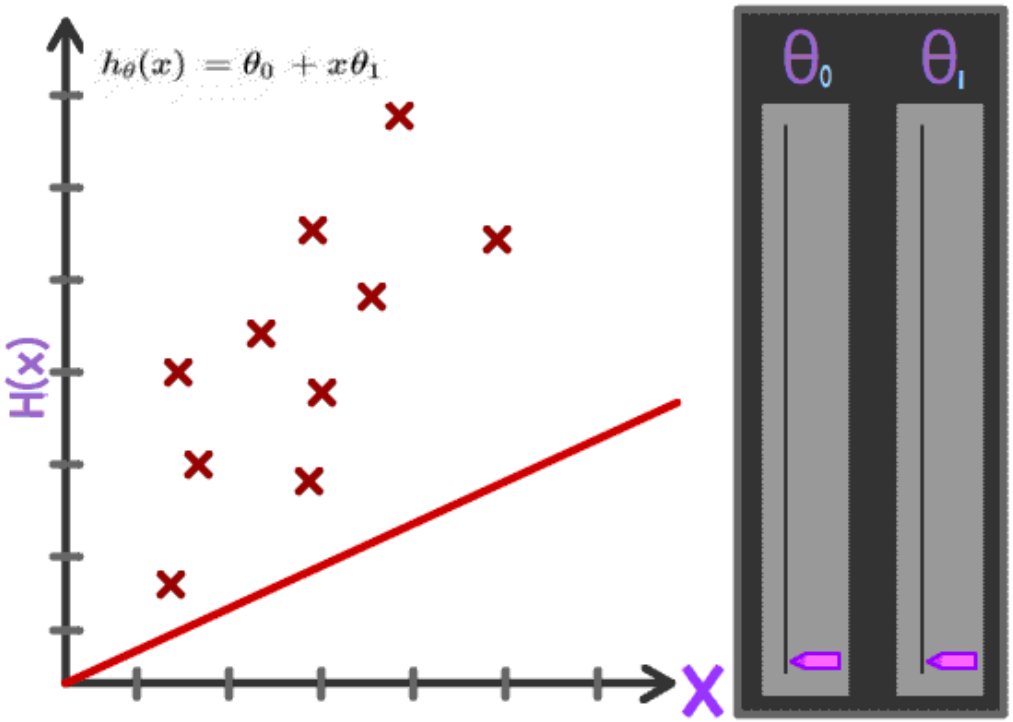
- Some clarification about the notation we will use for this course

$$X_i^{j,[k]}$$

- i is the index of the data, j is the feature number, and k is the power.

Solution Strategy for Solving the Problem

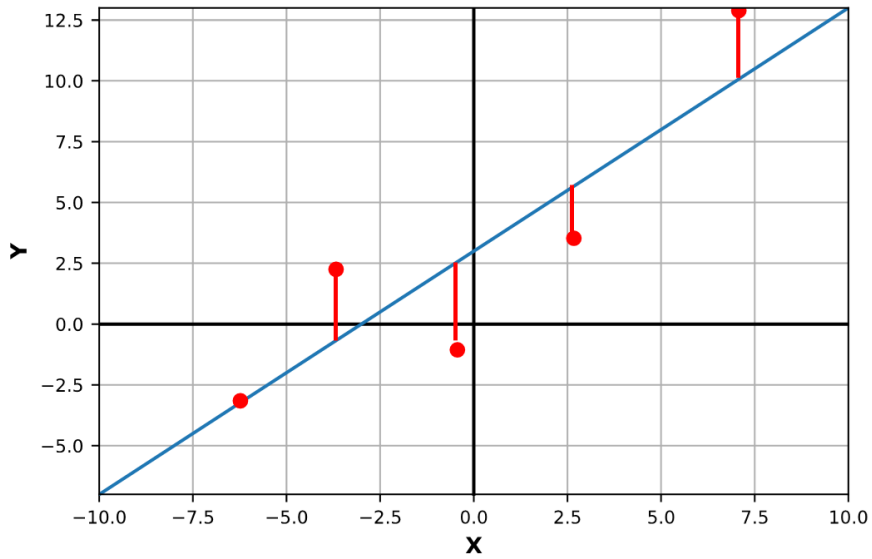
- There are countless possible lines.
- We want a line which is in some sense the "average line" that represents the data.
- Any ideas as to how we can do it?



Click the image to view the animated version.

Optimization

- To find the "best line," we should minimize the distances between our line's predictions and all the data points.
- How to define that mathematically?



Loss Function

- For one sample, this can be represented mathematically by:

$$(y_i - \hat{y}_i) \quad \text{(Error)}$$

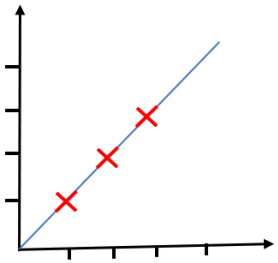
- But this could result in negative value if $\hat{y} > y$. Let's square it to remove the negative sign:

$$(y_i - \hat{y}_i)^2 \quad \text{(Squared Error)}$$

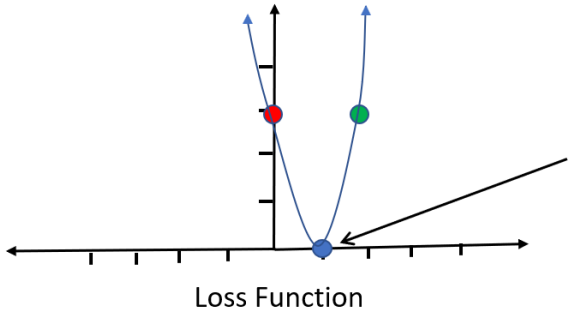
- But we have N samples, not only one. So, let's sum the errors and take the average:

$$\boxed{\text{Loss (MSE)} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \quad \text{(Mean Squared Error)}$$

Cost Function Example



Hypothesis



Loss Function

Notice: Lower is better.

$$J(m=0)=14$$

$$J(m=1)=0$$

$$J(m=2)=14$$

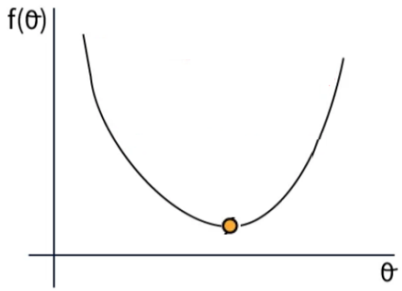
$$h(x)=mx \qquad J(m)=\sum_{i=1}^{3}(y_i-mx_i)^2$$

How to find minima of a function (Review):

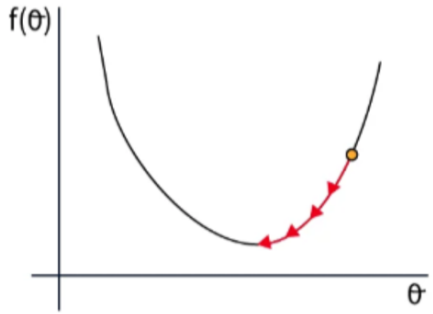
- There are two approaches to find the minima:
  - **Exact (Closed-form):** Directly calculates the solution mathematically by solving for $f'(x) = 0$.
    Important Note: can be used only with a very limited number of algorithms.
  - **Approximation (Iterative approach):** Gradually improves the solution step by step.
    Done by optimizers (e.g., Gradient Descent, ADAM, . . . etc).

How to find minima of a function (Review):

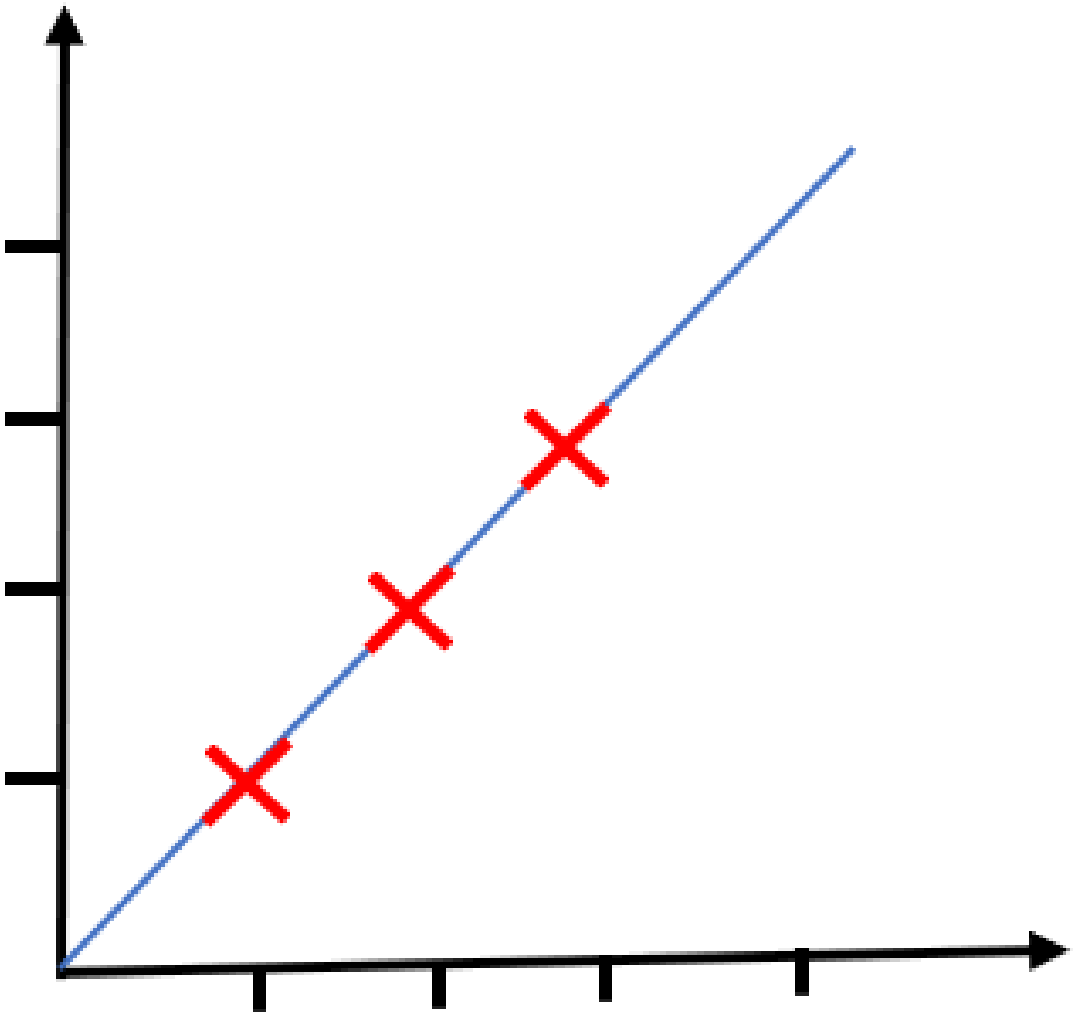
Closed-form:



Iterative:



- **Example:** $y = x^2$ (Solution: $x = 0$)
  - Closed-form Final Result: $x = 0$
  - Iterative Final Result: $x = 0.00001$ (close enough)



(Notice that $y = x$ for our 3 points.)

- Let's try to solve this using the closed-form here (Assume $\hat{y} = mx$):

$$J(m) = \sum_{i=1}^{3}(y_i - mx_i)^2 \quad \Rightarrow \quad J(m) = \sum_{i=1}^{3}(i - mi)^2$$

Review (cont.)

$$\frac{dJ(m)}{dm} = \frac{d}{dm}\sum_{i=1}^{3}(i - mi)^2 \quad \Rightarrow \quad \frac{dJ(m)}{dm} = \sum_{i=1}^{3}\frac{d}{dm}(i - mi)^2$$

$$\frac{dJ(m)}{dm} = \sum_{i=1}^{3} -2i(i - mi) \quad = \quad -2\sum_{i=1}^{3} i^2 + 2m\sum_{i=1}^{3} i^2 \quad = \quad 0 \quad \Rightarrow \quad m = 1$$

Hypothesis Function with 2 Variables

- Let's set up regression for a linear function in two variables:
- The hypothesis function is:

$$\hat{y}_i = mx_i + b$$

- Similar to the previous problem, our loss function is:

$$J = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- Let's calculate the partial derivatives of the loss function *w.r.t.* m, b

Gradient of the Loss Function

- We get the following expressions for the gradient of the cost function:

$$\frac{\partial J}{\partial m} = \frac{1}{N}\sum_{i=1}^{N} -2(y_i - \hat{y}_i)x_i$$

$$\frac{\partial J}{\partial b} = \frac{1}{N}\sum_{i=1}^{N} -2(y_i - \hat{y}_i)$$

Gradient of the Loss Function (cont.)

- Simplifying the above expressions, we get:

$$\frac{\partial J}{\partial m} = \frac{-2}{N}\sum_{i=1}^{N} y_i x_i + \frac{2m}{N}\sum_{i=1}^{N} x_i^2 + \frac{2b}{N}\sum_{i=1}^{N} x_i$$

$$\frac{\partial J}{\partial b} = \frac{-2}{N}\sum_{i=1}^{N} y_i + \frac{2m}{N}\sum_{i=1}^{N} x_i + \frac{2b}{N}\sum_{i=1}^{N} 1$$

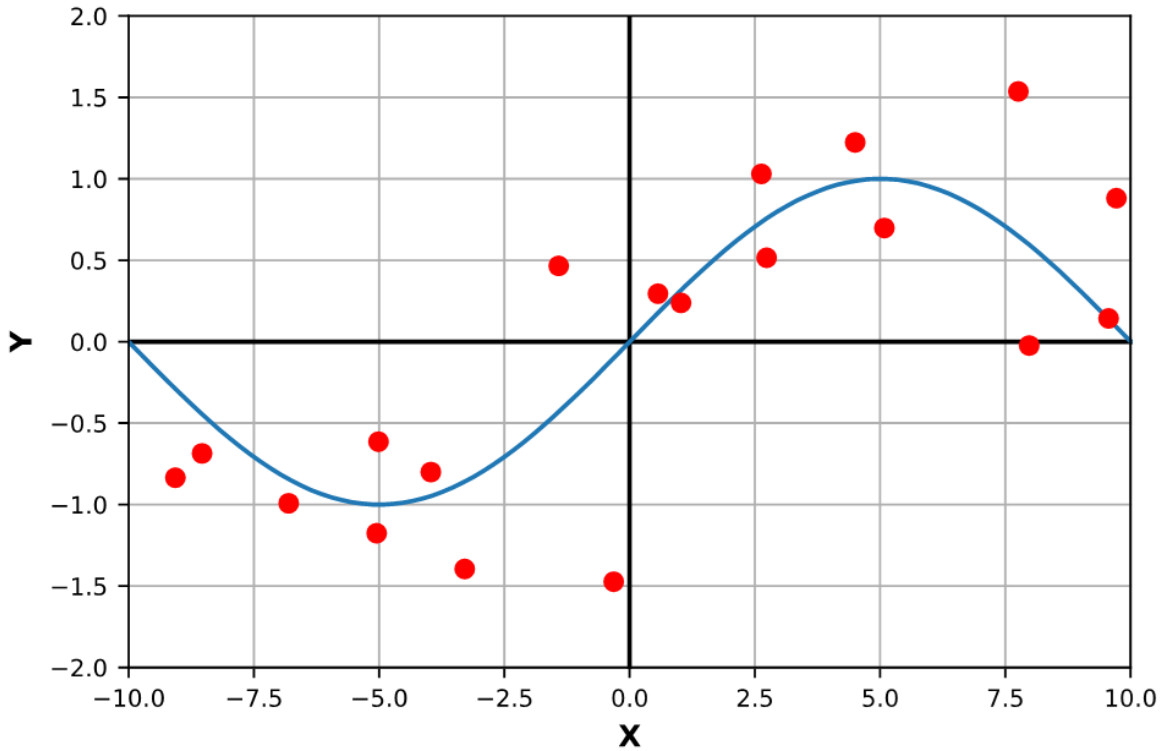
Gradient of the Loss Function (cont.)

- Setting the gradient equal to 0 and solving for m and b, we get:

$$\begin{bmatrix} \frac{\sum_i x_i^2}{N} & \frac{\sum_i x_i}{N} \\ \frac{\sum_i x_i}{N} & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} \frac{\sum_i x_i y_i}{N} \\ \frac{\sum_i y_i}{N} \end{bmatrix}$$

Fitting Non-linear Data

- What if y is a non-linear function of x?
- Will this approach still work?



Transforming the Feature Space (Feature Engineering)

- We can transform features x_i:

$$x_i = \left(x_i^1, x_i^2, x_i^3, \ldots, x_i^m\right)$$

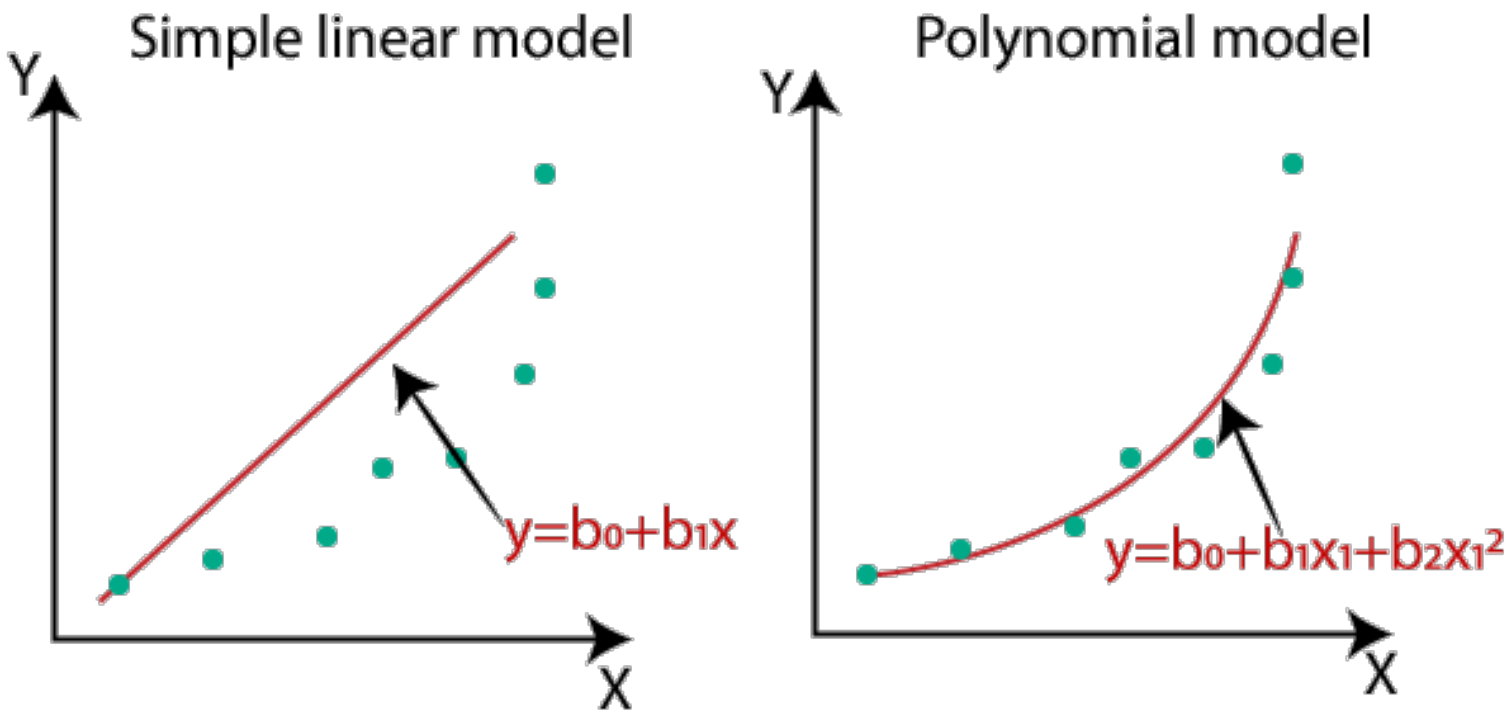
- We will apply some non-linear transformation ϕ:

$$\phi : \mathbb{R}^m \rightarrow \mathbb{R}^M$$

- For example, Polynomial transformation:

$$\phi(x_i) = \left\{1, x_i^1, x_i^{1,[2]}, \ldots, x_i^{1,[k]}, x_i^2, x_i^{2,[2]}, \ldots, x_i^{2,[k]}, \ldots, x_i^m, x_i^{m,[2]}, \ldots, x_i^{m,[k]}\right\}$$

Transforming the Feature Space (Feature Engineering)



Transforming the Feature Space (Feature Engineering)

- **Example:** Assume you have:

$$x_i^1 : \text{Length} \qquad x_i^2 : \text{Width}$$

- You can add $x_i^3 : \text{Area} = x_i^1 \cdot x_i^2$ to the dataset.
- **Other types:**
  - Cosine, splines, radial basis functions, etc.
  - Encoding (Label encoding, One-hot, . . .)
  - Domain-related features (e.g., financial measures)
  - Time-related features (Day, month, year, . . .)
  - Group-level features (e.g., average income per household, total sales per region, median age per team). Often called "Aggregation features".

Gradient of the loss function

- Let's get back to the gradients...

Issues with the Approach

- Assume we have 100 variables instead of 2.
- Calculating gradients like this can quickly become tedious.
- **Notice:** Each term on either side of the expression can be written as a dot product of two vectors (maybe we can calculate it more efficiently)?

- Let's explore if we can do something better through **vectorization** (Writing equations as matrices).

Vectorization

- To truly appreciate the power of vectorization, let's make the problem a little more complex. The hypothesis function is now

$$\hat{y}_i = w_0 + w_1 x_i^1 + w_2 x_i^2 + \cdots + w_M x_i^M$$

- Where w_j $(j = 0, 1, \ldots, M)$ are the unknown weights of the data, and x_i^j is the jth feature of the ith input.
- Next, we denote the discrepancy between y_i and $\hat{y}_i$ as ϵ_i

$$y_i = \hat{y}_i + \epsilon_i$$

- Now let's collect the above equation for all N datapoints:

$$
\begin{aligned}
y_1 &= \hat{y}_1 + \epsilon_1 \\
y_2 &= \hat{y}_2 + \epsilon_2 \\
&\vdots \\
y_N &= \hat{y}_N + \epsilon_N
\end{aligned}
$$

- Replacing the values of $\hat{y}$, we get:

$$
\begin{aligned}
y_1 &= w_0 + w_1 x_1^1 + w_2 x_1^2 + \cdots + w_M x_1^M + \epsilon_1 \\
y_2 &= w_0 + w_1 x_2^1 + w_2 x_2^2 + \cdots + w_M x_2^M + \epsilon_2 \\
&\vdots \\
y_N &= w_0 + w_1 x_N^1 + w_2 x_N^2 + \cdots + w_M x_N^M + \epsilon_N
\end{aligned}
$$

Vectorization

- Collecting the equations in matrix form:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} 1 & x_1^1 & x_1^2 & \cdots & x_1^M \\ 1 & x_2^1 & x_2^2 & \cdots & x_2^M \\ 1 & x_3^1 & x_3^2 & \cdots & x_3^M \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N^1 & x_N^2 & \cdots & x_N^M \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_N \end{bmatrix}$$

- Notice the rows of the matrix on the right are data samples:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} \cdots & \mathbf{x}_1 & \cdots \\ \cdots & \mathbf{x}_2 & \cdots \\ \cdots & \mathbf{x}_3 & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \mathbf{x}_N & \cdots \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_N \end{bmatrix}$$

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$$

- Let's formalize some notations:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} \cdots & \mathbf{x}_1 & \cdots \\ \cdots & \mathbf{x}_2 & \cdots \\ \cdots & \mathbf{x}_3 & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \mathbf{x}_N & \cdots \end{bmatrix} \quad \boldsymbol{\theta} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{bmatrix} \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_N \end{bmatrix}$$

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\epsilon}$$

Cost function for the Vectorized form

- Notice that we are using the MSE cost function:

$$J = \frac{1}{N}\sum_{i}(y_i - \hat{y}_i)^2$$

- Using the definition of epsilon we can write the above as:

$$J = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(\epsilon_i)^2$$

- Using the definition of dot product the above can be written as:

$$J = \frac{1}{N}\sum_{i=1}^{N}(\epsilon_i)^2 = \frac{1}{N}\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}$$

- We get:

$$\frac{\partial J}{\partial \boldsymbol{\theta}} = \mathbf{X}^T 2(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})$$

- Setting it equal to zero we can solve for $\boldsymbol{\theta}$:

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

Closed-form solution for Linear Regression

Perspectives on ML Algorithms

- We can look at ML algorithms from two perspectives:
  - **Loss Minimization** Problem (like what we did).
  - **Probability Maximization** Problem (using Maximum Likelihood Estimation).
- For linear regression, under particular assumptions, these two approaches yield equivalent solutions.

Probabilistic Interpretation of Linear Regression and MLE

- We can also look at the probabilistic interpretation of Linear Regression.
- Keeping everything else same as the previous formulation

$$y_i = \mathbf{x}_i^T \boldsymbol{\theta} + \epsilon_i$$

- Now assume that $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, then

$$y_i \mid \mathbf{x}_i \sim \mathcal{N}(\mathbf{x}_i^T \boldsymbol{\theta}, \sigma^2)$$

- We can write the conditional distribution as:

$$\mathbb{P}(y_i \mid \mathbf{x}_i) \sim \mathcal{N}(\mathbf{x}_i^T \boldsymbol{\theta}, \sigma^2)$$

Probabilistic Interpretation of LR

- Let's assume that all data point in the dataset are i.i.d. (independent identically distributed). Then we have:

$$\mathbb{P}(\mathcal{D}) = \prod_{i=1}^{N} \mathbb{P}(\mathbf{x}_i, y_i)$$

- Using Bayes Theorem we can write:

$$\prod_{i=1}^{N} \mathbb{P}(\mathbf{x}_i, y_i) = \prod_{i=1}^{N} \mathbb{P}(\mathbf{x}_i)\mathbb{P}(y_i \mid \mathbf{x}_i)$$

- In simple words, given the dataset, we want to find the values of the unknown parameters which maximize the probability of the dataset.
- Using the definition of the conditional distribution we have:

$$\mathbb{P}(y_i \mid \mathbf{x}_i) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-(y_i - \mathbf{x}_i^T\boldsymbol{\theta})^2\right)$$

- Using the definition we get:

$$\prod_{i=1}^{N} \mathbb{P}(\mathbf{x}_i, y_i) = \prod_{i=1}^{N} \mathbb{P}(\mathbf{x}_i) \prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-(y_i - \mathbf{x}_i^T\boldsymbol{\theta})^2\right)$$

- Let's try to maximize:

$$\prod_{i=1}^{N} \mathbb{P}(\mathbf{x}_i, y_i) = \prod_{i=1}^{N} \mathbb{P}(\mathbf{x}_i) \prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-(y_i - \mathbf{x}_i^T \boldsymbol{\theta})^2\right)$$

- Note that:

$$\arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{N} \mathbb{P}(\mathbf{x}_i, y_i) = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{N} \exp\left(-(y_i - \mathbf{x}_i^T \boldsymbol{\theta})^2\right)$$

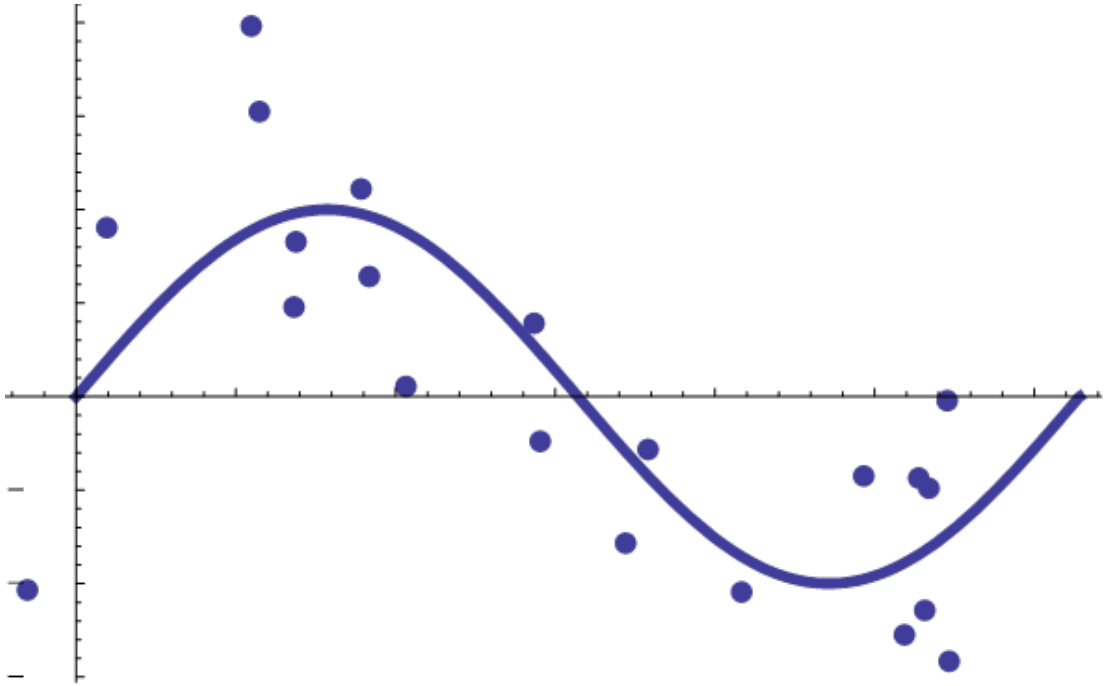
- Furthermore, since the right-hand side of the above equation is monotonic in $\boldsymbol{\theta}$, the arg max will not change if we take the logarithm of the expression.

$$\arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{N} \exp\left(-(y_i - \mathbf{x}_i^T \boldsymbol{\theta})^2\right) = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \left(-(y_i - \mathbf{x}_i^T \boldsymbol{\theta})^2\right)$$

- Notice that the right-hand side is minimizing the MSE.
- Hence, the solution of minimizing the MSE is equivalent to Maximum Likelihood Estimation for linear regression.

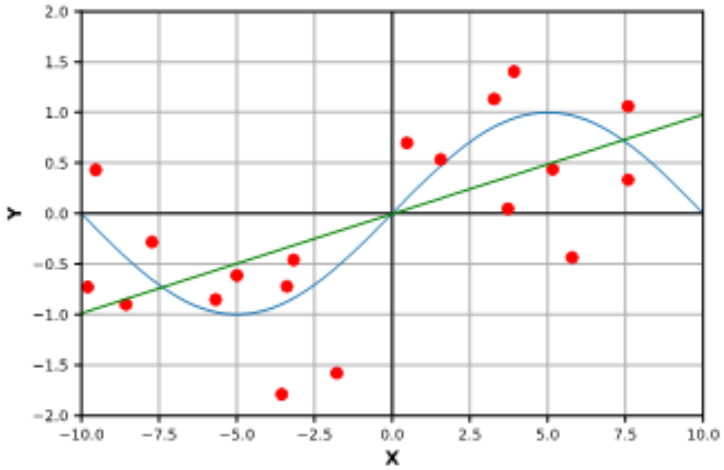
Bias and Variance

- What if Y has a non-linear response?

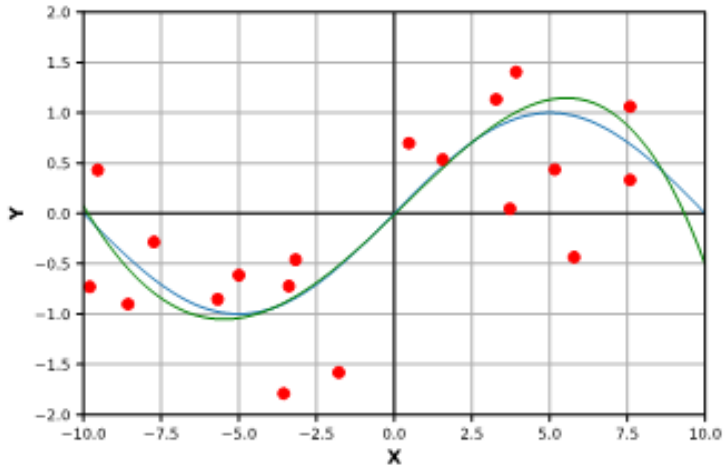


- Can we still use a linear model?

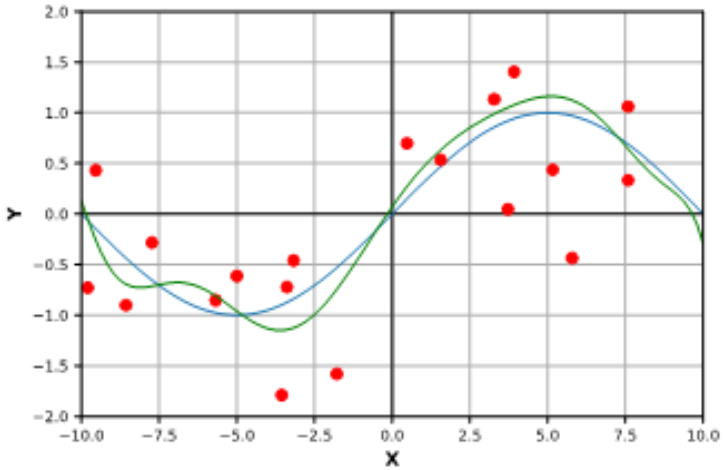
What is Bias and Variance?



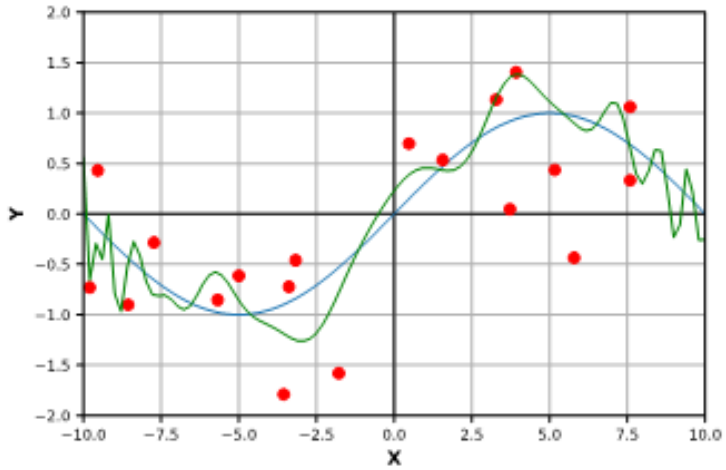
$\{1, x\}$



$\{1, x, x^{[2]}, x^{[3]}, x^{[4]}\}$

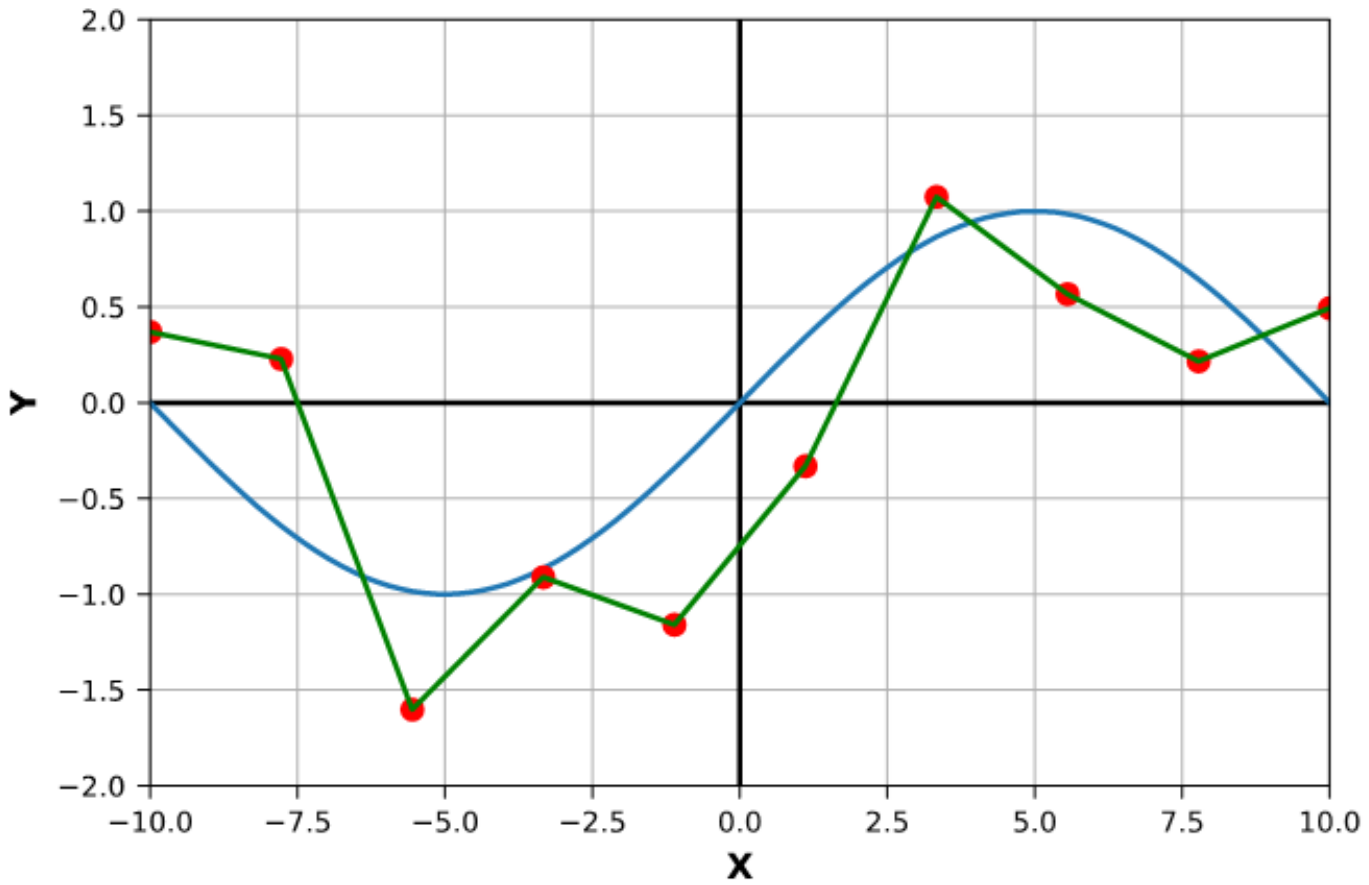


$\{1, x, x^{[2]}, \ldots, x^{[10]}\}$



$\{1, x, x^{[2]}, \ldots, x^{[100]}\}$

Real Bad Overfit?



Bias-Variance Tradeoff

- So far, we have minimized the error (loss) with respect to **training data**
  - Low training error does not imply good expected performance: **over-fitting**
- We would like to reason about the **expected loss (Prediction Risk)** over:
  - Training Data: $\{(y_1, x_1), \ldots, (y_n, x_n)\}$
  - Test point: (y_*, x_*)
- We will decompose the expected loss into:

$$\mathbb{E}_{D,(y_*,x_*)}\left[(y_* - f(x_* \mid D))^2\right] = \text{Noise} + \text{Bias}^2 + \text{Variance}$$

Bias Variance Plot

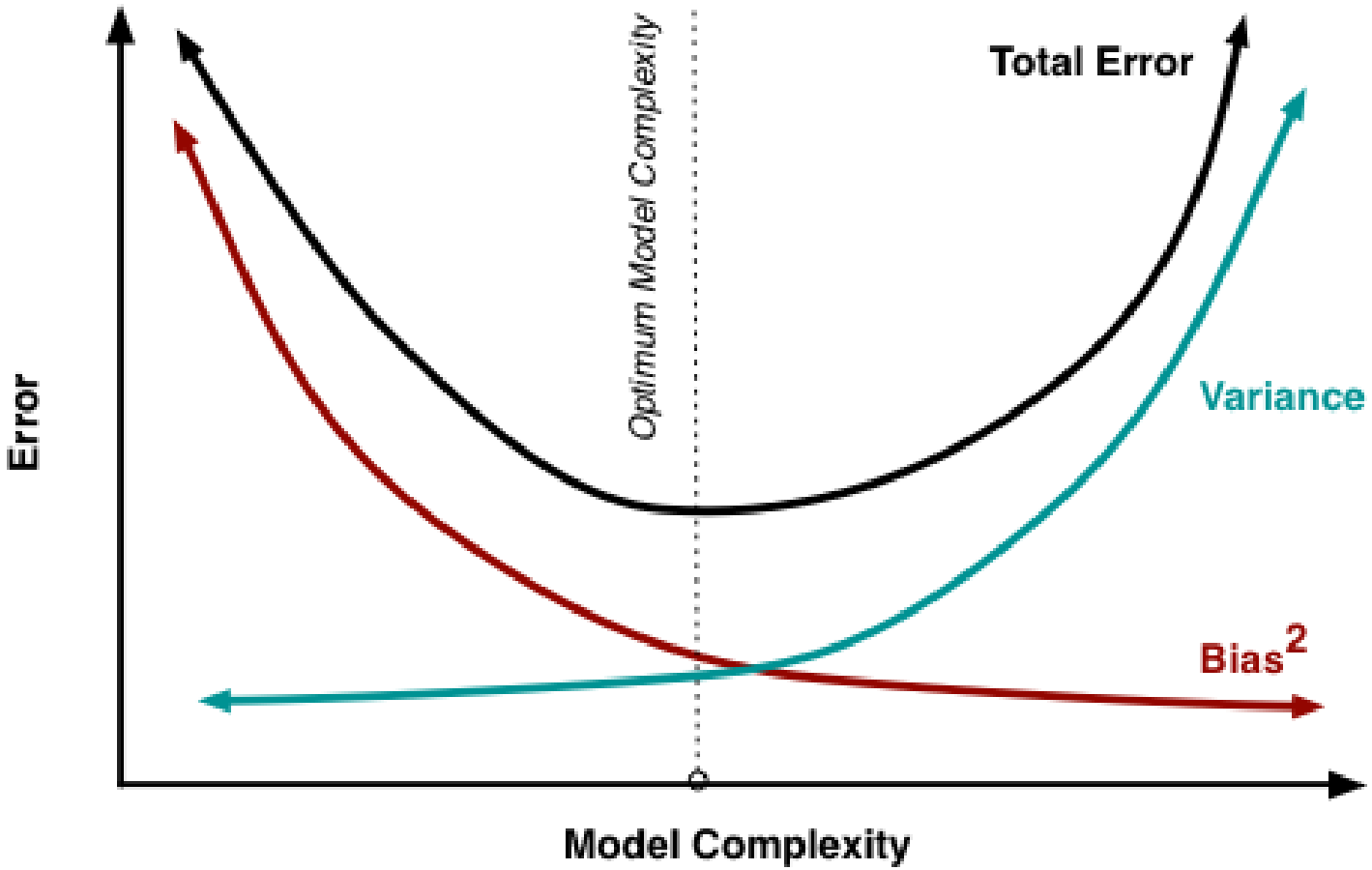


Image from http://scott.fortmann-roe.com/docs/BiasVariance.html

R-squared (R^2 or r^2) — Coefficient of Determination

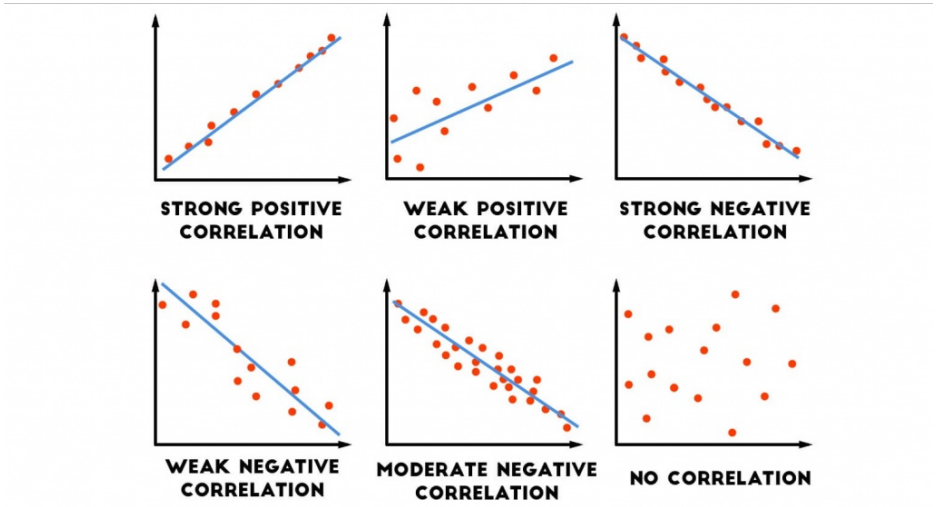
- **A statistical metric** used to measure how well the independent variables explain the variability in the dependent variable.
- It provides a measure of the goodness-of-fit (**performance**) for a regression model:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

Where:

- $SS_{residual} = \sum(y_i - \hat{y}_i)^2$: sum of squared residuals (difference between observed and predicted values).
- $SS_{total} = \sum(y_i - \bar{y})^2$: total variation in the data around the mean.

Visualisation of R^2



Interpretation:

- $R^2 = 1$: The model perfectly explains the data.
- $R^2 = 0$: The model explains none of the variability (as good as guessing the mean).
- $R^2 < 0$: The model performs worse than a simple horizontal line at the mean of the target variable.

Properties of R^2

Range:

- $0 \leq R^2 \leq 1$
  - $R^2 = 1$: Perfect model; predictions perfectly match the observations.
  - $R^2 = 0$: Model does no better than the mean of the dependent variable.

Interpretability:

- R^2 indicates the proportion of the variance in the dependent variable that is predictable from the independent variables.

Limitations:

- R^2 increases with the addition of independent variables, even if they don't improve model prediction significantly.
- Does not account for overfitting or the complexity of the model.

Negative Values:

Properties of R^2 (cont.)

- In rare cases, R^2 can be negative when the model fits the data worse than a horizontal line representing the mean of the dependent variable.

Adjusted R^2:

- Adjusted R^2 penalizes for the addition of non-significant predictors and is often a better metric for comparing models.
- Here, n is the number of data points and p is the number of predictors:

$$R^2_{\text{adj}} = 1 - \frac{(1 - R^2)(n - 1)}{n - p - 1}$$

R-squared vs. Cost Function

R-squared vs. Cost Function:

- **Cost Function:** A mathematical function used to optimize a model during training by minimizing the prediction error (e.g., Mean Squared Error or MSE).
  - Example: Gradient Descent minimizes MSE for linear regression.
- **R-squared:** A metric to evaluate how well the model fits the data **after training**. It is not used during model optimization.

Limitations of R^2:

1. **Doesn't Indicate Causation:** A high R^2 doesn't mean the predictors cause the dependent variable.
2. **Sensitive to Overfitting:** A complex model might have a high R^2 but poor generalization.
3. **Adjusted R^2:** For models with many predictors, use adjusted R^2, which accounts for the number of predictors to avoid overestimating performance.

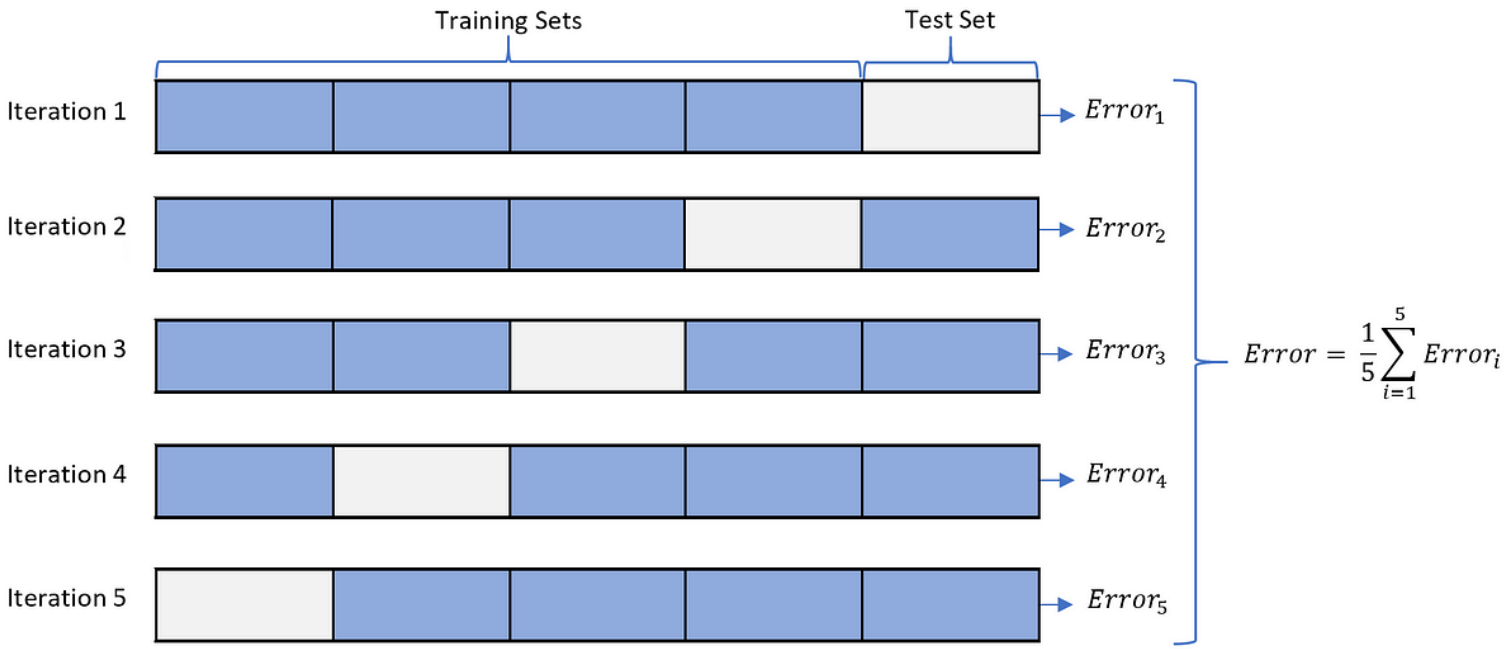
Data Split

- To ensure your model doesn't overfit to the training data, you should have another subset called **testing data**.
- You will evaluate your model against this **subset**, and based on its **metric score (e.g., accuracy)** you will decide if it's overfitting or not.
- But how should I split my data?

Data Split

- **Hold-out set:**
  - A portion of the dataset set aside and not used during training.
  - E.g. 80% for training and 20% for testing.
- **Issues:**
  - Imagine you have these labels: [1, 1, 2, 2, 2, 3, 3, 3, 3, 3] and you took last 30% as test: [3,3,3]. You didn't include 1 and 2 in test!
  - **Solution:** Always shuffle before split: [3, 2, 3, 1, 3, 2, 3, 1, 3, 2] $\Rightarrow$ test: [1,3,2]
  - My dataset is small. Taking 20% as test would not be representative!
  - **Solution:** Use `KFold`.

- **K-Fold Cross Validation (CV):**
  - Split data into k parts (folds)
  - Train on $k-1$ folds, test on the remaining fold
  - Repeat k times and average the scores



Regularization

Slides from: https://harvard-iacs.github.io/2019-CS109A/lectures/lecture8/presentation/Lecture8a_Regularization.pptx

Regularization: An Overview

- Regularization involves modifying the loss function L to improve model generalization.
- A regularization term is added to penalize certain properties of the model parameters:

$$L_{\text{reg}}(\beta) = L(\beta) + \lambda R(\beta)$$

- λ is a scalar that controls the strength of the regularization.
- The function $R(\beta)$ encodes the properties to penalize (e.g., large coefficients).
- Minimizing L_{reg} leads to model parameters with more desirable characteristics.

LASSO Regression

- LASSO discourages extreme values in model parameters by penalizing their magnitudes.
- We use MSE as the base loss function.
- The regularized loss function is:

$$L_{\text{LASSO}}(\beta) = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \beta^\top x_i\right|^2 + \lambda\sum_{j=1}^{J}|\beta_j|$$

- The term $\sum_{j=1}^{J}|\beta_j|$ is the ℓ_1 norm of the vector β:

$$\sum_{j=1}^{J}|\beta_j| = \|\beta\|_1$$

Ridge Regression

- Regularization term penalizes the **squares** of the parameter magnitudes.
- The regularized loss function becomes:

$$L_{\text{Ridge}}(\beta) = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \beta^\top x_i\right|^2 + \lambda\sum_{j=1}^{J}\beta_j^2$$

- Note that:

$$\sum_{j=1}^{J}\beta_j^2 = \|\beta\|_2^2$$

which is the square of the ℓ_2-norm of β.

Choosing λ

- In both ridge and LASSO regression, the larger the regularization parameter λ, the more we penalize large values in β.
- If λ is close to zero:
  - The penalty is minimal.
  - We recover the MSE, i.e., ridge and LASSO become ordinary regression.
- If λ is sufficiently large:
  - The MSE term becomes negligible.
  - The regularization forces β_{ridge} and β_{LASSO} close to zero.
- To avoid ad-hoc selection, λ should be chosen via cross-validation.

- **Ridge Regression Solution:**

$$\beta = (X^T X + \lambda I)^{-1} X^T Y$$

- **LASSO Regression:**
  - No closed-form analytical solution.
  - The L_1 norm is not differentiable at 0.
  - We use the concept of *subdifferential* or *subgradient* to derive manageable expressions.

Regularization Parameter with a Validation Set

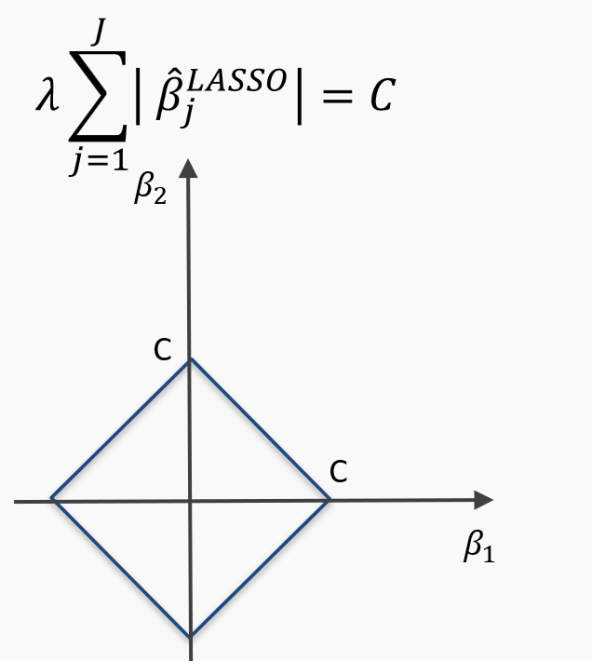
The solution of the Ridge/Lasso regression involves three steps:

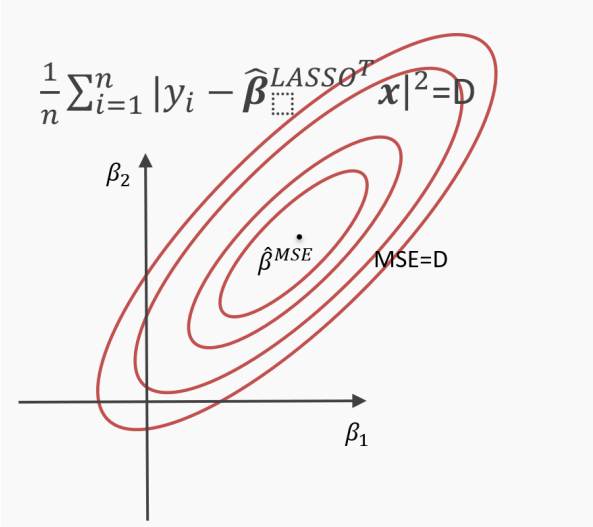
- Select λ
- Find the minimum of the ridge/Lasso regression loss function (using the ridge formula), and record the **MSE on the validation/test set**.
- Find the λ that gives the smallest MSE

The Geometry of Regularization (LASSO)

$$L_{\text{LASSO}}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \boldsymbol{\beta}^T\mathbf{x}\right|^2 + \lambda\sum_{j=1}^{J}|\beta_j|$$

$$\hat{\boldsymbol{\beta}}^{\text{LASSO}} = \arg\min L_{\text{LASSO}}(\boldsymbol{\beta})$$





The Geometry of Regularization (LASSO)

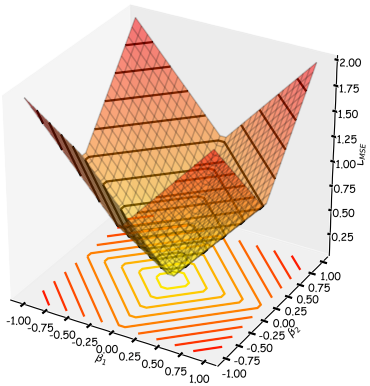
- Regularized loss function:

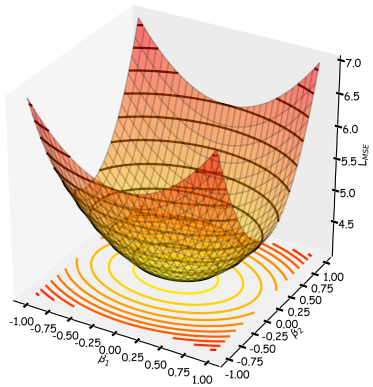
$$L_{\text{LASSO}}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \boldsymbol{\beta}^T\mathbf{x}\right|^2 + \lambda\sum_{j=1}^{J}|\beta_j|$$

- LASSO solution:

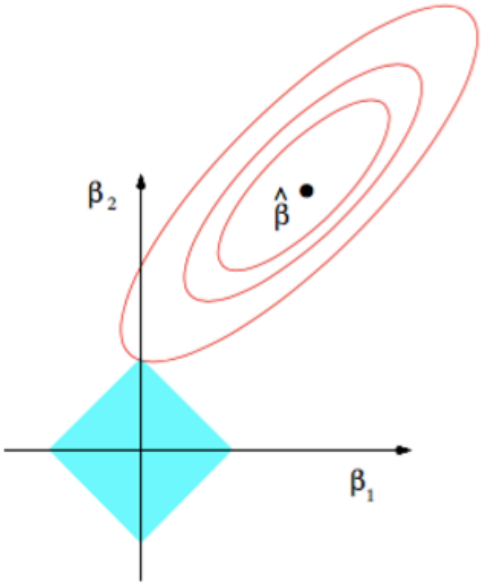
$$\hat{\boldsymbol{\beta}}^{\text{LASSO}} = \arg\min L_{\text{LASSO}}(\boldsymbol{\beta})$$

- $L_1 = \lambda\sum_{j=1}^{J}\left|\hat{\beta}_j^{\text{LASSO}}\right|$
- $L_{\text{MSE}}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \boldsymbol{\beta}^T\mathbf{x}\right|^2$

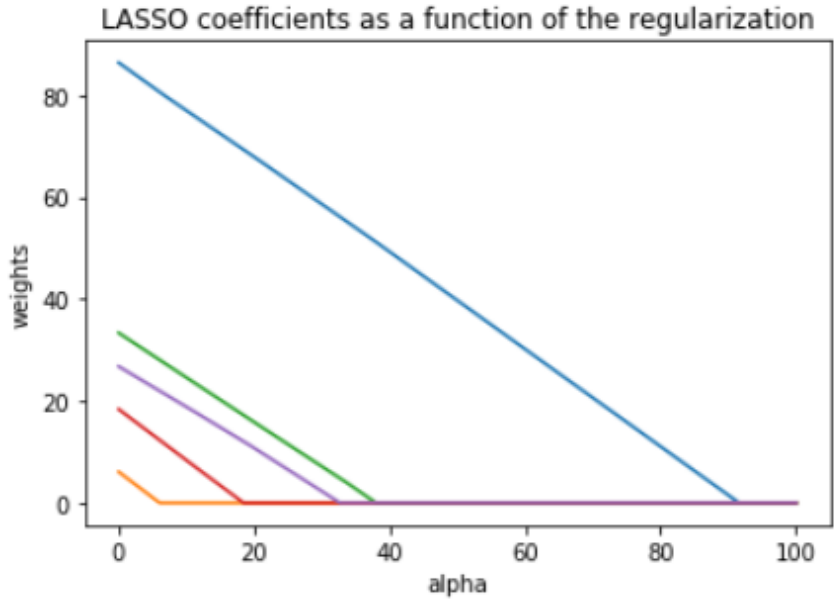




LASSO visualized

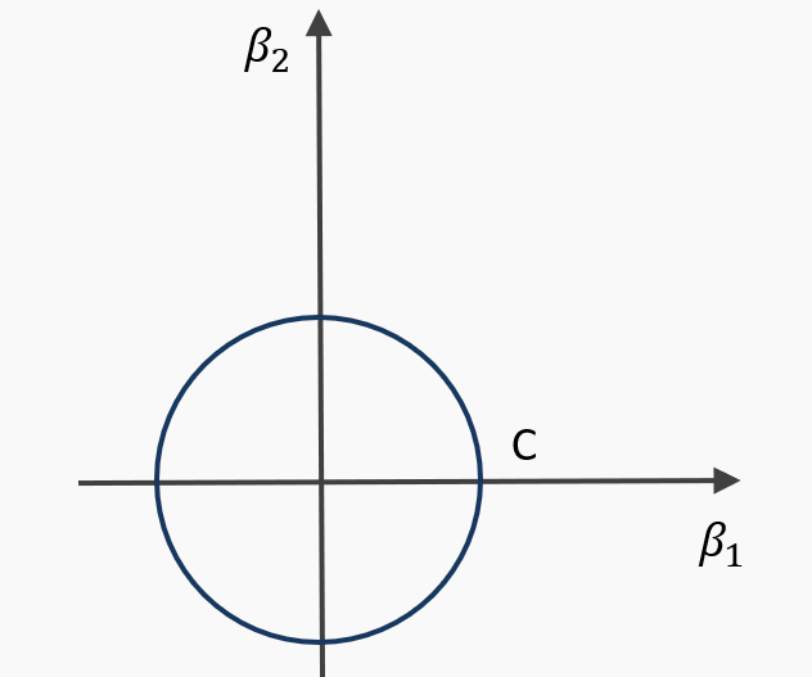


- The LASSO estimator tends to zero out parameters.
- The OLS loss can easily intersect with the constraint on one of the axes.

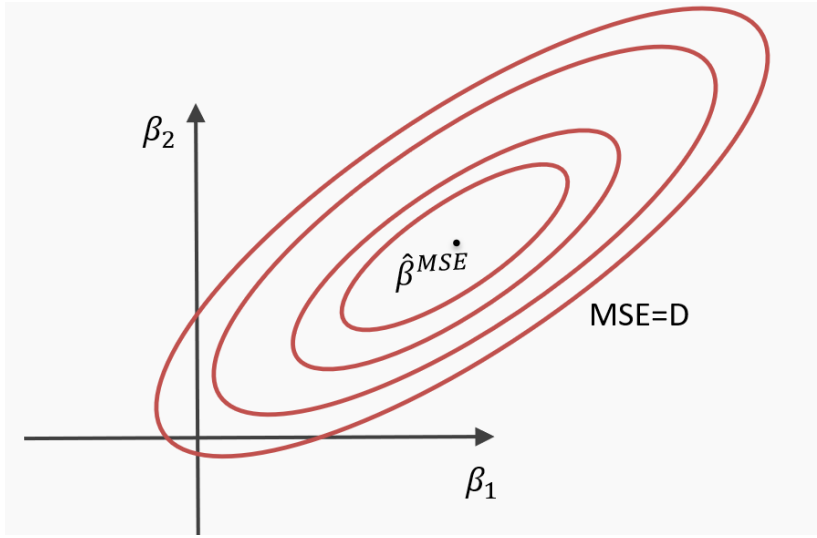


- The values of the coefficients decrease as λ increases.
- Coefficients are nullified quickly.

The Geometry of Regularization (Ridge)

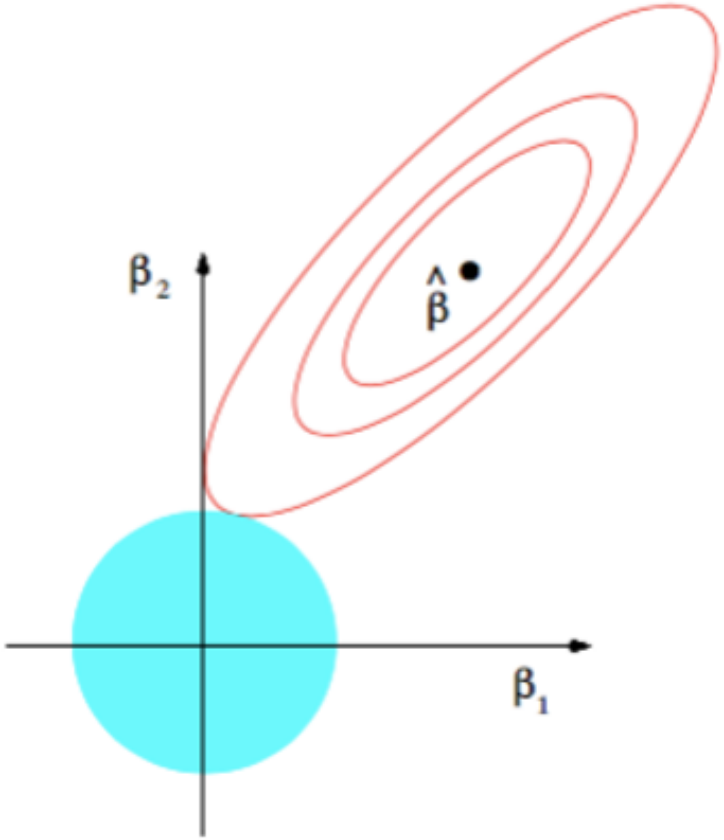


- $L_{Ridge}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \boldsymbol{\beta}^T\boldsymbol{x}\right|^2 + \lambda\sum_{j=1}^{J}(\beta_j)^2$
- $\hat{\boldsymbol{\beta}}^{Ridge} = \arg\min L_{Ridge}(\boldsymbol{\beta})$
- $\lambda\sum_{j=1}^{J}\left|\hat{\beta}_j^{Ridge}\right|^2 = C$

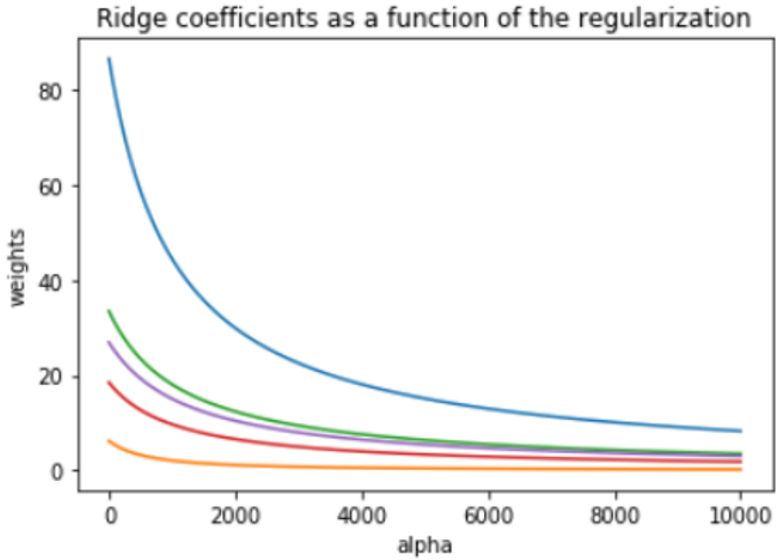


- $\frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{\boldsymbol{\beta}}^{Ridge\ T}\boldsymbol{x}\right|^2 = D$
- The ridge constraint forms a circle in 2D.
- The solution lies at the point where the MSE contour touches the constraint boundary.

Ridge visualized

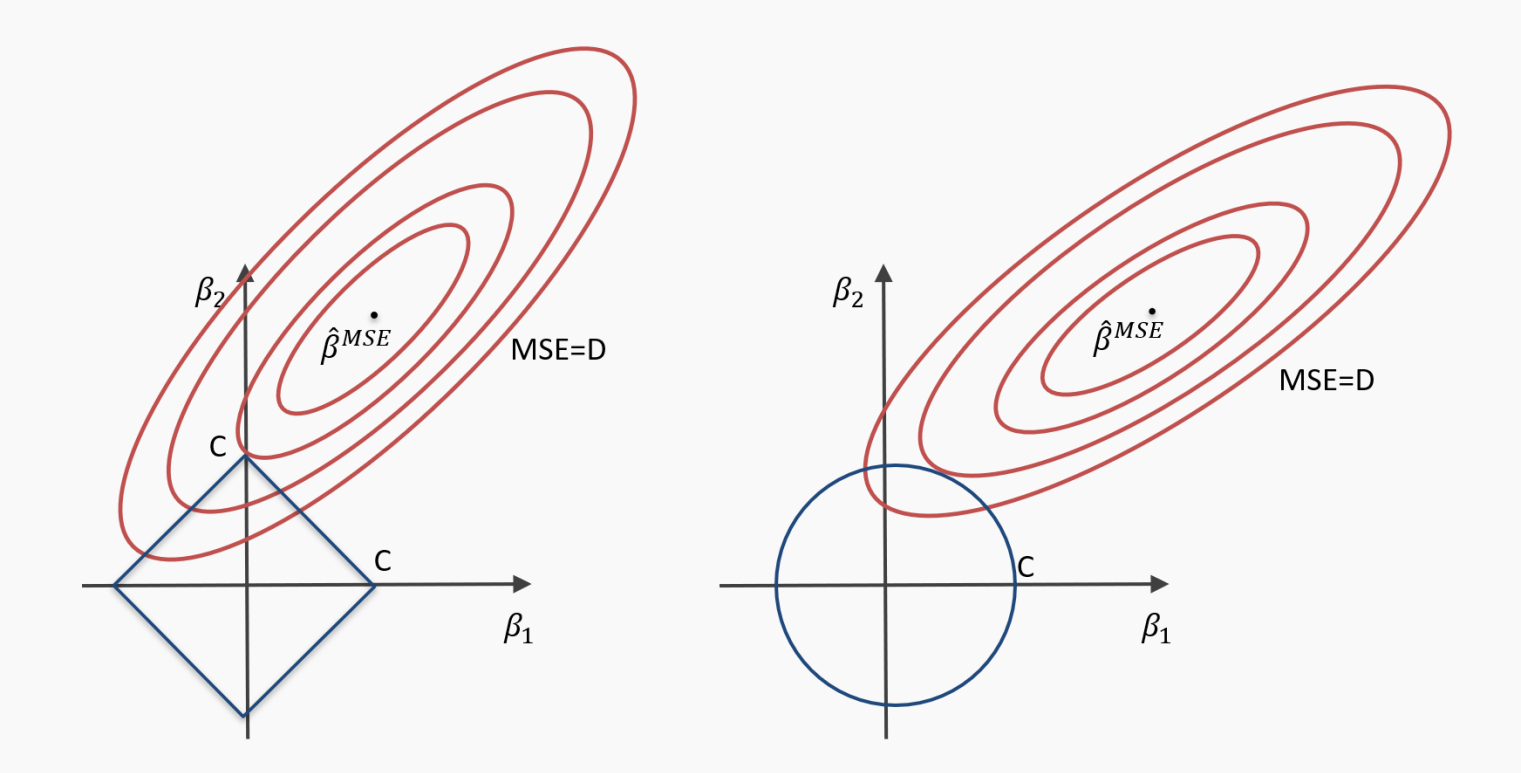


- The ridge estimator is where the constraint and the loss intersect.



- The values of the coefficients decrease as lambda increases, but they are not nullified.

The Geometry of Regularization



Ridge regularization with only **validation**: step by step

1. Split data into $\{\{X, Y\}_{\text{train}}, \{X, Y\}_{\text{validation}}, \{X, Y\}_{\text{test}}\}$
2. For λ in $\{\lambda_{\text{min}}, \ldots, \lambda_{\text{max}}\}$:
   2.1 Determine the β that minimizes the L_{ridge}:

$$\hat{\beta}_{\text{Ridge}}(\lambda) = (X^T X + \lambda I)^{-1} X^T Y, \quad \text{using the train data}$$

   2.2 Record $L_{\text{MSE}}(\lambda)$ using validation data.
3. Select the λ that minimizes the loss on the validation data:

$$\lambda_{\text{ridge}} = \arg\min_{\lambda} L_{\text{MSE}}(\lambda)$$

4. Refit the model using both **train and validation** data:

$$\{X, Y\}_{\text{train}}, \{X, Y\}_{\text{validation}} \Rightarrow \hat{\beta}_{\text{ridge}}(\lambda_{\text{ridge}})$$

5. Report MSE or R^2 on $\{X, Y\}_{\text{test}}$ given the $\hat{\beta}_{\text{ridge}}(\lambda_{\text{ridge}})$

Lasso regularization with **validation only**: step by step

1. Split data into $\{\{X, Y\}_{\text{train}}, \{X, Y\}_{\text{validation}}, \{X, Y\}_{\text{test}}\}$
2. For λ in $\{\lambda_{\text{min}}, \ldots, \lambda_{\text{max}}\}$:
   - 2.1 Determine the β that minimizes the L_{lasso},

     $$\hat{\beta}_{\text{lasso}}(\lambda),$$

     using the train data. **This is done using a solver**.
   - 2.2 Record $L_{\text{MSE}}(\lambda)$ using validation data.
3. Select the λ that minimizes the loss on the validation data:

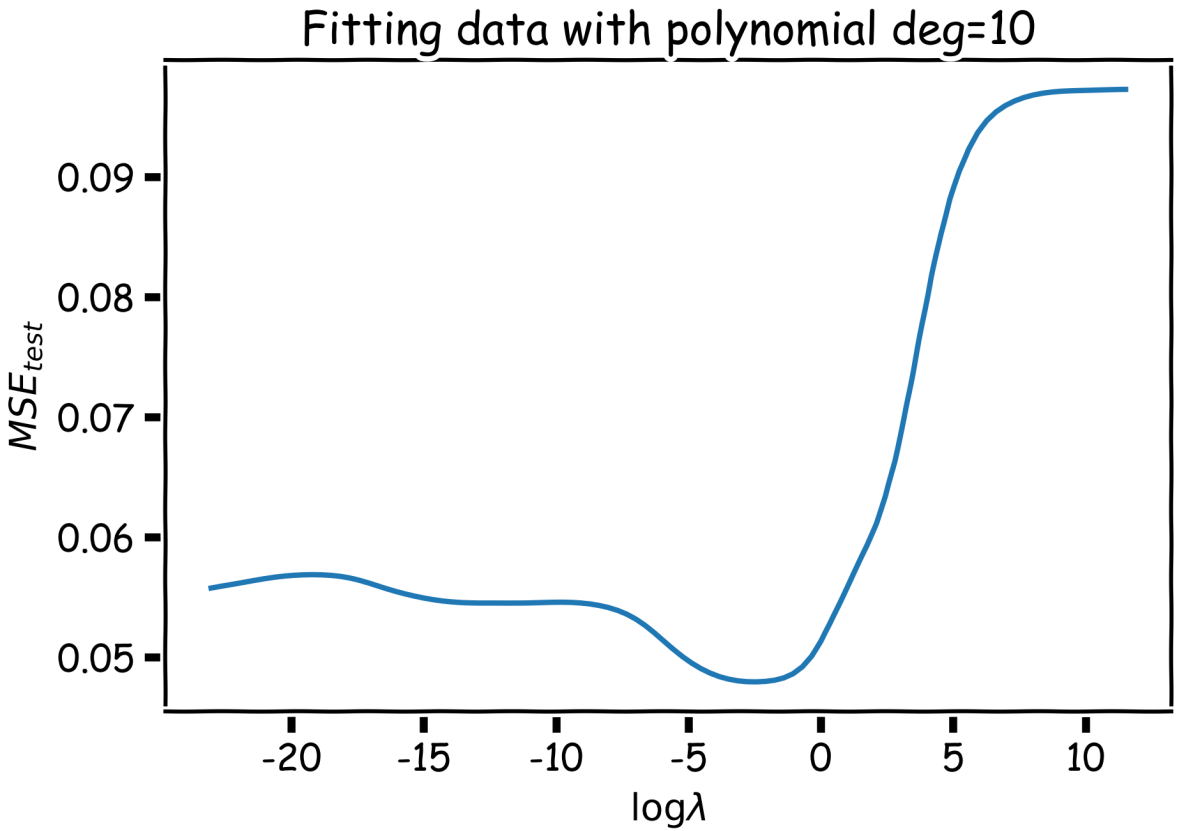
   $$\lambda_{\text{lasso}} = \arg\min_{\lambda} L_{\text{MSE}}(\lambda)$$

4. Refit the model using both **train** and **validation** data:

   $$\{X, Y\}_{\text{train}}, \{X, Y\}_{\text{validation}} \Rightarrow \hat{\beta}_{\text{lasso}}(\lambda_{\text{lasso}})$$

5. Report MSE or R^2 on $\{X, Y\}_{\text{test}}$ given the $\hat{\beta}_{\text{lasso}}(\lambda_{\text{lasso}})$

Ridge regularization with **validation only**: step by step



Ridge regularization with **CV**: step by step

1. Remove $\{X, Y\}_{\text{test}}$ from data.
2. Split the rest of the data into K folds: $\left\{\{X, Y\}^{-k}_{\text{train}}, \{X, Y\}^{k}_{\text{val}}\right\}$
3. For $k \in \{1, \ldots, K\}$:

   3.1 For $\lambda \in \{\lambda_0, \ldots, \lambda_n\}$:

   - Determine the β that minimizes the L_{ridge}:

     $$\hat{\beta}_{\text{ridge}}(\lambda, k) = (X^\top X + \lambda I)^{-1} X^\top Y$$

     using the training data $\{X, Y\}^{k}_{\text{train}}$.
   - Record $L_{\text{MSE}}(\lambda, k)$ using the validation data $\{X, Y\}^{k}_{\text{val}}$.

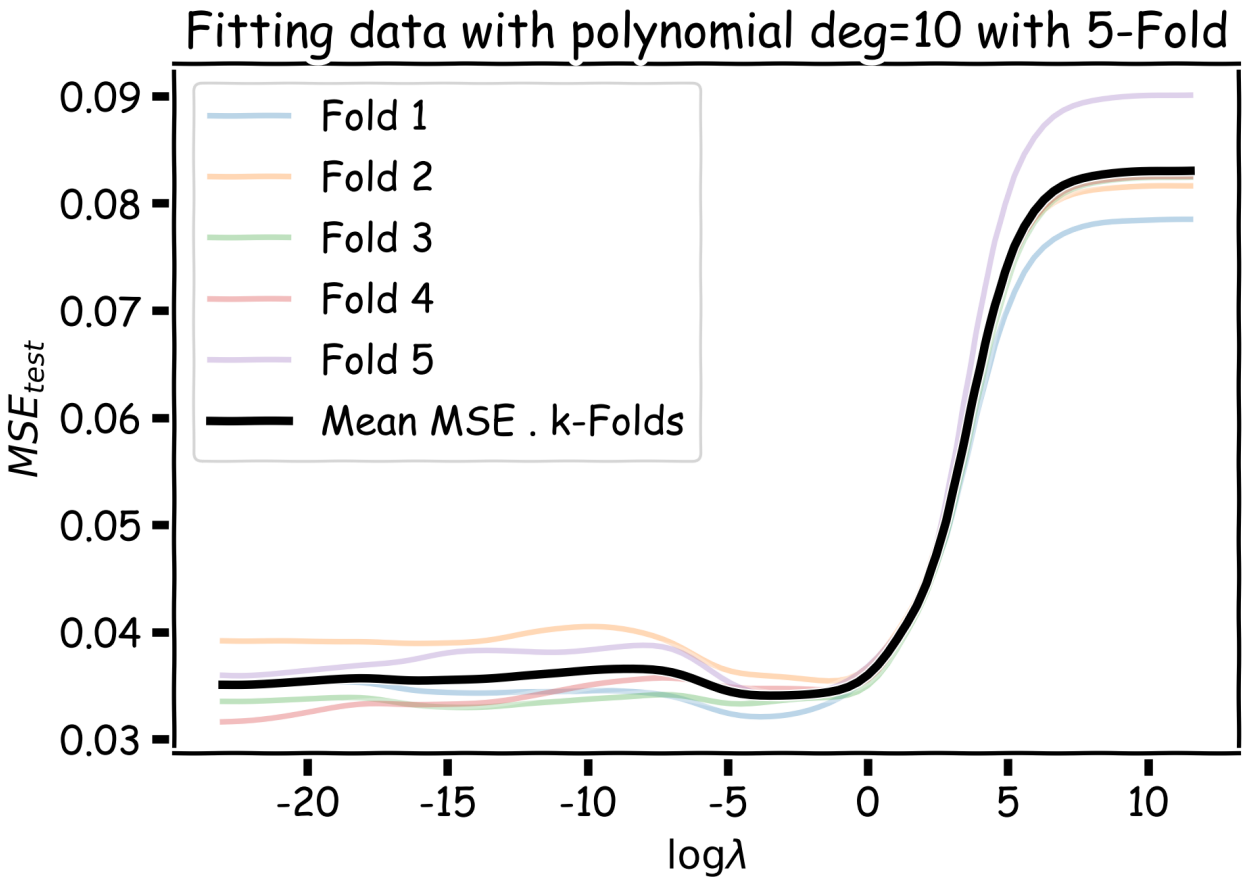
   At this point, we have a 2D matrix: rows correspond to different k, and columns to different λ values.
4. Average the $L_{\text{MSE}}(\lambda, k)$ for each λ:

$$\bar{L}_{\text{MSE}}(\lambda) = \frac{1}{K} \sum_{k=1}^{K} L_{\text{MSE}}(\lambda, k)$$

Ridge regularization with **CV**: step by step (cont.)

5. Find the λ that minimizes $\bar{L}_{\text{MSE}}(\lambda)$: $\lambda_{\text{ridge}} = \arg\min_{\lambda} \bar{L}_{\text{MSE}}(\lambda)$
6. Refit the model using the full training data $\{\{X, Y\}_{\text{train}}, \{X, Y\}_{\text{val}}\} \Rightarrow \hat{\beta}_{\text{ridge}}(\lambda_{\text{ridge}})$
7. Report MSE or R^2 on $\{X, Y\}_{\text{test}}$ using $\hat{\beta}_{\text{ridge}}(\lambda_{\text{ridge}})$

Ridge regularization with **CV** only: step by step



Validation error MSE_{val} across λ values (log scale) using 5-fold cross-validation. Each colored line corresponds to one fold. The thick black curve shows the mean validation error across all folds.

Variable Selection as Regularization

- Since **LASSO** regression tends to produce zero estimates for a number of model parameters – we say that LASSO solutions are **sparse** – we consider LASSO to be a method for variable selection.
- Many prefer using LASSO for variable selection (as well as for suppressing extreme parameter values) rather than stepwise selection, as LASSO avoids the statistical problems that arise in stepwise selection.
- **Question:** What are the pros and cons of the two approaches?