HSMMC Training Workshop 1
Advanced Linear Algebra and Linear Programming

HSMMC 訓練工作坊 1
進階線性代數及線性規劃

Prof. Gary Pui Tung CHOI 蔡沛彤教授

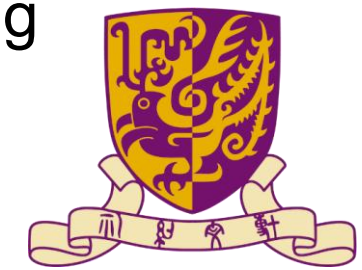
Dr. Jeff Chak Fu WONG 黃澤富博士

Dr. Lily Li PAN 潘莉博士

Department of Mathematics, The Chinese University of Hong Kong

香港中文大學數學系

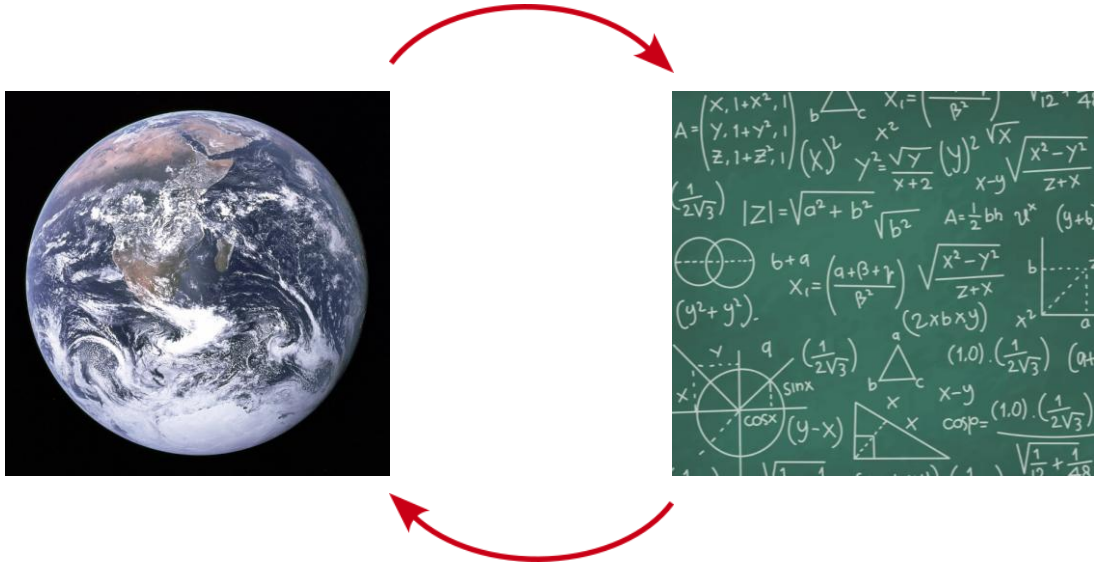
2026/05/02



Recall: Mathematical Modelling is … 數學建模是 ...

Understanding a real-world problem using mathematics

用數學了解現實生活問題



Mathematical Modelling Process
5 Steps of Mathematical Modelling
數學建模過程
數學建模 5 部曲

Real-world problem 現實世界問題

1. Analyzing the real-world Problem 分析現實世界問題
2. Formulating the Mathematical Model 建立數學模型

Mathematical model 數學模型

3. Solving the Mathematical Model 求解數學模型

Mathematical solution 數學模型的解

4. Interpreting the Mathematical Solution 解釋數學模型的解

Real-world Solution 現實世界的解

5. Evaluating the Mathematical Model 評估數學模型

Recall: What Mathematical Concepts may be involved in Mathematical Modelling? 數學建模會用到甚麼數學概念？

- Short answer: Everything is possible! 簡短答案：一切皆有可能！

- **Algebra 代數**
  - Solving equations 解方程
  - Modelling with functions 使用函數建模
- **Geometry 幾何**
  - Geometric measurements 幾何測量
  - Deductive geometry 演繹幾何
  - Coordinate geometry 座標幾何
  - Trigonometry 三角函數
- **Probability and statistics 概率與統計**
  - Randomness 隨機性
  - Statistical analysis 統計分析
- **More advanced techniques 更進階的技巧**
  - Calculus 微積分
  - Optimization 最優化

Mathematical and IT tools for modelling

- IT tools for math modelling
  https://www.math.cuhk.edu.hk/app/mathmodel/tool.html

- RShiny tools:
  - Linear Regression 線性迴歸
  - Nonlinear Regression for XY data XY數據非線性迴歸
  - Nonlinear Regression for time data 時間數據非線性迴歸
  - General Fitting for XY data XY數據一般擬合
  - Multi-Regression 多元線性迴歸
  - …

- More advanced tools:
  - Python, R, C/C++, MATLAB, Mathematica, …

Advanced linear algebra and linear programming

- Topics:
  - Vectors and matrices 向量和矩陣
  - Solving systems of equations 求解方程組
  - Least squares and weighted least squares 最小平方法和加權最小平方法
  - Regularized regression 正則化迴歸
  - Eigenvalues and eigenvectors 特徵值和特徵向量
  - Graphs and graph algorithms 圖論及圖算法
  - Linear programming 線性規劃
  - Integer programming整數規劃
  - Network flow problem網路流問題
- Goal:
  - Understand the principle of equation solving and regression methods
  - Explore other usages of linear algebra and linear programming in math modelling
  - Utilize IT tools (Python) to perform linear algebra and linear programming tasks

Why do we need vectors and matrices?

- A major task in linear algebra: **Solving systems of equations**

e.g.

$$\begin{cases} x + 3y = 7 \\ 2x + y = 4 \end{cases}$$

Applications:

- Prices of different products
- Mixture of chemicals
- etc.

Why do we need vectors and matrices?

- A major task in linear algebra: **Solving systems of equations**

e.g.

$$\begin{cases} x + 3y = 7 \\ 2x + y = 4 \end{cases}$$

How to solve it?

$$\begin{cases} x + 3y = 7 & (1) \\ 2x + y = 4 & (2) \end{cases}$$

$2 \times (1) - (2)$:

$$(2x + 6y) - (2x + y) = 14 - 4$$

$$5y = 10$$

$$y = 2$$

$$x = 7 - 3 \times 2 = 1$$

Why do we need vectors and matrices?

- A major task in linear algebra: **Solving systems of equations**

e.g.

$$\begin{cases} x + 2y + 3z = 6 \\ 2x - 3y + 2z = 14 \\ 3x + y - z = -2 \end{cases}$$

Why do we need vectors and matrices?

- A major task in linear algebra: **Solving systems of equations**

e.g.

$$\begin{cases} x + 2y + 3z = 6 \\ 2x - 3y + 2z = 14 \\ 3x + y - z = -2 \end{cases}$$

Eliminate the variables step by step (still manageable…)

$$\begin{cases} x + 2y + 3z = 6 \\ -7y - 4z = 2 \\ -5y - 10z = -20 \end{cases} \Rightarrow \begin{cases} x + 2y + 3z = 6 \\ y + 2z = 4 \\ 10z = 30 \end{cases} \Rightarrow \begin{cases} x = 1 \\ y = -2 \\ z = 3 \end{cases}$$

Why do we need vectors and matrices?

- More generally, we need some **better representations** and **better solving methods** for systems of linear equations!

$$\begin{cases} x + 3y = 7 \\ 2x + y = 4 \end{cases}$$

$$\begin{pmatrix} 1 & 3 \\ 2 & 1 \end{pmatrix} \qquad \begin{pmatrix} x \\ y \end{pmatrix}$$

Coefficients **Variables**

$$\begin{cases} x + 2y + 3z = 6 \\ 2x - 3y + 2z = 14 \\ 3x + y - z = -2 \end{cases}$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & -3 & 2 \\ 3 & 1 & -1 \end{pmatrix} \qquad \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Coefficients **Variables**

- This motivates us to consider **vectors and matrices**

Vector

- **Definition**

$$(a_1 \ a_2 \ \cdots \ a_n)$$

Row vector

$$\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$$

Column vector

Examples:

$$\begin{pmatrix} 1 & 3 \end{pmatrix}, \quad \begin{pmatrix} 2 & -1 & 0 & 10 & 7.5 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 3 \end{pmatrix}, \quad \begin{pmatrix} 3 \\ -1 \\ 4 \\ -1.59 \end{pmatrix}$$

- **Additions and scalar multiplications**

$$u + v = (a_1 + b_1, a_2 + b_2, \cdots, a_n + b_n)$$

$$cu = (ca_1, ca_2, \cdots, ca_n)$$

$$(3, -2, 0) + (-1, 1, 4) = (2, -1, 4)$$

$$-5(1, -2, 0) = (-5, 10, 0)$$

Matrix

- **Definition**

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Examples:

$$\begin{pmatrix} 1 & 3 \end{pmatrix}, \quad \begin{pmatrix} 2 & 1 \\ 3 & 8 \end{pmatrix}, \quad \begin{pmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \end{pmatrix}, \quad \begin{pmatrix} 2 & 7 \\ 1 & 8 \\ 2 & 8 \end{pmatrix}$$

- **Additions and scalar multiplications**

$$(A+B)_{ij} = A_{ij} + B_{ij}$$

$$(cA)_{ij} = cA_{ij}$$

$$\begin{pmatrix} 2 & 0 & -1 \\ 1 & -3 & 4 \end{pmatrix} + \begin{pmatrix} -5 & -2 & 6 \\ 3 & 4 & -1 \end{pmatrix} = \begin{pmatrix} -3 & -2 & 5 \\ 4 & 1 & 3 \end{pmatrix}$$

$$-3 \begin{pmatrix} 1 & 0 & 2 \\ -3 & 2 & 3 \end{pmatrix} = \begin{pmatrix} -3 & 0 & -6 \\ 9 & -6 & -9 \end{pmatrix}$$

Matrix multiplication

- **We can perform matrix (or vector) multiplications**

  - A: $m \times n$ matrix
  - B: $n \times p$ matrix
  - AB (the product): $m \times p$ matrix with

$$(AB)_{ij} = A_{i1}B_{1j} + A_{i2}B_{2j} + \cdots + A_{in}B_{nj} = \sum_{k=1}^{n} A_{ik}B_{kj}$$

Example:

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 4 & -1 \end{pmatrix} \begin{pmatrix} 4 \\ 2 \\ 5 \end{pmatrix} = \begin{pmatrix} 1 \cdot 4 + 2 \cdot 2 + 1 \cdot 5 \\ 0 \cdot 4 + 4 \cdot 2 + (-1) \cdot 5 \end{pmatrix} = \begin{pmatrix} 13 \\ 3 \end{pmatrix}$$

- Just another way to represent the systems of equations!

$$\begin{pmatrix} 1 & 3 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x + 3y \\ 2x + y \end{pmatrix}$$

$$\begin{cases} x + 3y = 7 \\ 2x + y = 4 \end{cases} \Leftrightarrow \begin{pmatrix} 1 & 3 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 7 \\ 4 \end{pmatrix}$$

More on matrices

- **Identity matrix**

$$(I_n)_{ij} = 1 \text{ if } i = j$$

$$(I_n)_{ij} = 0 \text{ if } i \neq j$$

- $IA = A = AI$

$$I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- **Transpose**

$$(A^T)_{ij} = A_{ji}$$

- A: $m \times n$
- A^T: $n \times m$

$$\begin{pmatrix} 1 & -2 & 3 \\ 0 & 5 & -1 \end{pmatrix}^T = \begin{pmatrix} 1 & 0 \\ -2 & 5 \\ 3 & -1 \end{pmatrix}$$

Determinants

- **2×2 case:**

If $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, the determinant is

$$\det(A) = ad - bc$$

- Sometimes also denoted as $|A|$

For the matrices $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ and $B = \begin{pmatrix} 3 & 2 \\ 6 & 4 \end{pmatrix}$,

$$\det(A) = 1 \cdot 4 - 2 \cdot 3 = -2$$

$$\det(B) = 3 \cdot 4 - 2 \cdot 6 = 0$$

Determinants

- **General $n \times n$ case:** $$\det(A) = \sum_{j=1}^{n} (-1)^{1+j} A_{1j} \cdot \det(\tilde{A}_{1j})$$

Notation of $\tilde{A}_{ij}$:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

$$\tilde{A}_{11} = \begin{pmatrix} 5 & 6 \\ 8 & 9 \end{pmatrix}, \quad \tilde{A}_{13} = \begin{pmatrix} 4 & 5 \\ 7 & 8 \end{pmatrix},$$

$$\tilde{A}_{32} = \begin{pmatrix} 1 & 3 \\ 4 & 6 \end{pmatrix}$$

Let

$$A = \begin{pmatrix} 1 & 3 & -3 \\ -3 & -5 & 2 \\ -4 & 4 & -6 \end{pmatrix} \in M_{3\times 3}(\mathbb{R}).$$

Using cofactor expansion along the first row of A, we obtain

$$\begin{aligned} \det(A) &= (-1)^{1+1} A_{11} \cdot \det(\tilde{A}_{11}) + (-1)^{1+2} A_{12} \cdot \det(\tilde{A}_{12}) + (-1)^{1+3} A_{13} \cdot \det(\tilde{A}_{13}) \\ &= (-1)^2(1) \cdot \det \begin{pmatrix} -5 & 2 \\ 4 & -6 \end{pmatrix} + (-1)^3(3) \cdot \det \begin{pmatrix} -3 & 2 \\ -4 & -6 \end{pmatrix} + (-1)^4(-3) \cdot \det \begin{pmatrix} -3 & -5 \\ -4 & 4 \end{pmatrix} \\ &= 1(22) - 3(26) - 3(-32) \\ &= 40. \end{aligned}$$

Matrix inverse

- If $det(A) \neq 0$, then we can always find a matrix B such that

$$AB = BA = I$$

  - In this case, A is said to be invertible
  - B is called the matrix inverse of A (also denoted as A^{-1})

- 2×2 case: $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$

$$A^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

The inverse of $\begin{pmatrix} 5 & 7 \\ 2 & 3 \end{pmatrix}$ is $\begin{pmatrix} 3 & -7 \\ -2 & 5 \end{pmatrix}$

- General case: hard to calculate by hand

Doing basic matrix operations in Python

- We can use computers to help us with computations involving vectors and matrices

- See Python notebook

  https://colab.research.google.com/github/CUHKMathModel/Python/blob/main/Basic_Operation_of_Matrices.ipynb

Eigenvalues and eigenvectors

- (e-book Ch. 6) https://www.math.cuhk.edu.hk/~mathcal/MM/MMLA/

- Definition:
  Let A be an $n \times n$ matrix. A number λ is said to be an eigenvalue of A if there exists a nonzero vector x such that

$$Ax = \lambda x$$

  x is said to be an eigenvector of A associated with the eigenvalue λ

Eigenvalues and eigenvectors

- Finding eigenvalues and eigenvectors: Find the roots of the characteristic polynomial

$$\det(A - \lambda I) = 0$$

- Application to matrix diagonalization: We can have

$$A = PDP^{-1}$$

where

- D is a diagonal matrix containing the eigenvalues
- P is a matrix with columns being the corresponding eigenvectors
- Easy to calculate A^n

- Computation using Python: https://colab.research.google.com/github/CUHKMathModel/Python/blob/main/Eigenvalue_and_Eigenvector.ipynb

Vector and matrix norm

- **Vector norm:**
  - (Euclidean norm) $\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$
  - (p-norm) $\|\mathbf{x}\|_p = \left(|x_1|^p + |x_2|^p + \cdots + |x_n|^p\right)^{\frac{1}{p}}$

If $\mathbf{x} = (1, 3, 1)$, we have $\|\mathbf{x}\| = \sqrt{1^2 + 3^2 + 1^2} = \sqrt{11}$.

For the vector $\mathbf{x} = (2, -3)^T$, $\|\mathbf{x}\|_3 = \sqrt[3]{|2|^3 + |-3|^3} = \sqrt[3]{35}$.

- **Matrix norm:** $\|A\|_p = \max_{\|\mathbf{x}\|=1} \|A\mathbf{x}\|_p$
- Calculating vector and matrix norms in Python: See Python notebook https://colab.research.google.com/github/CUHKMathModel/Python/blob/main/Matrix_Norm.ipynb

(Recall) Evaluating the accuracy 準確性的評估方法

- **Coefficient of determination 決定係數 (R^2):** $R^2 = 1 - \frac{RSS}{TSS}$

  - RSS is the residual sum of squares RSS 是殘差平方和:

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = (y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + \cdots + (y_n - \hat{y}_n)^2$$

    with $\hat{y}_i$ being the predicted value based on the chosen model
    其中 $\hat{y}_i$ 是基於所選模型的預測值

  - TSS is the total sum of squares TSS 是總平方和:

$$TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2 = (y_1 - \bar{y})^2 + (y_2 - \bar{y})^2 + \cdots + (y_n - \bar{y})^2$$

    with $\bar{y} = \frac{1}{n}(\sum_{i=1}^{n} y_i) = \frac{y_1 + y_2 + \cdots + y_n}{n}$ being the mean of the given data
    其中$\bar{y} = \frac{1}{n}(\sum_{i=1}^{n} y_i) = \frac{y_1 + y_2 + \cdots + y_n}{n}$ 是已有數據的平均值

(Recall) Evaluating the accuracy 準確性的評估方法

- **Mean squared error 均方誤差 (MSE) :**

$$MSE = \frac{RSS}{n} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

  - Value can be from 0 to ∞ 數值可以為 0 到 ∞
  - Easy to compute 易於計算

- **Root mean squared error 均方根誤差 (RMSE):**

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

  - Value can be from 0 to ∞ 數值可以為 0 到 ∞
  - With the square root, the error is in the same units as the original data (hence more intuitive, but relatively hard to compute)
    加上平方根後，誤差的單位與原始數據相同（因此更直觀，但相對難以計算）

(Recall) Evaluating the accuracy 準確性的評估方法

- **Mean absolute error 平均絕對誤差 (MAE):**

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

  - Here, $|x|$ is the absolute value (絕對值) of x: $|x| = \begin{cases} x \text{ if } x \geq 0 \\ -x \text{ if } x < 0 \end{cases}$
  - Value can be from 0 to ∞ 數值可以為 0 到 ∞
  - The error is in the same units as the original data 誤差的單位與原始數據相同

- **Mean absolute percentage error 平均絕對百分比誤差 (MAPE)**

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\%$$

  - Value can be from 0% to 100%, hence giving an intuitive interpretation in terms of relative error 數值可以從 0% 到 100%，因此在相對誤差方面提供直觀的解釋
  - But small or close-to-zero values of some y_i may disproportionately affect the MAPE score 但某些 y_i 的值很小或接近零時，可能會不成比例地影響 MAPE 值

Computation in Python

- **Calculation of R^2 in Python:**

Basic commands (using the *Scikit-learn* library in Python):

```
>>> from sklearn.metrics import r2_score
>>> y_actual = [3, -0.5, 2, 7]
>>> y_predict = [2.5, 0.0, 2, 8] # by your model
>>> r2_score(y_actual, y_predict)
0.948...
```

More examples:

```
>>> y_actual = [1, 2, 3]
>>> y_predict = [1, 2, 3] # a perfect fit
>>> r2_score(y_actual, y_predict)
1.0
>>> y_actual = [1, 2, 3]
>>> y_predict = [2, 2, 2] # baseline model
>>> r2_score(y_actual, y_predict)
0.0
>>> y_actual = [1, 2, 3]
>>> y_predict = [3, 2, 1] # fits worse than baseline
>>> r2_score(y_actual, y_predict)
-3.0
```

- **Calculation of MSE/MAE/RMSE in Python:**

MSE:

```
>>> from sklearn.metrics import mean_squared_error
>>> y_actual = [3, -0.5, 2, 7]
>>> y_predict = [2.5, 0.0, 2, 8] # by your model
>>> mean_squared_error(y_actual, y_predict)
0.375
```

MAE:

```
>>> from sklearn.metrics import mean_absolute_error
>>> mean_absolute_error(y_actual, y_predict)
0.5
```

RMSE:

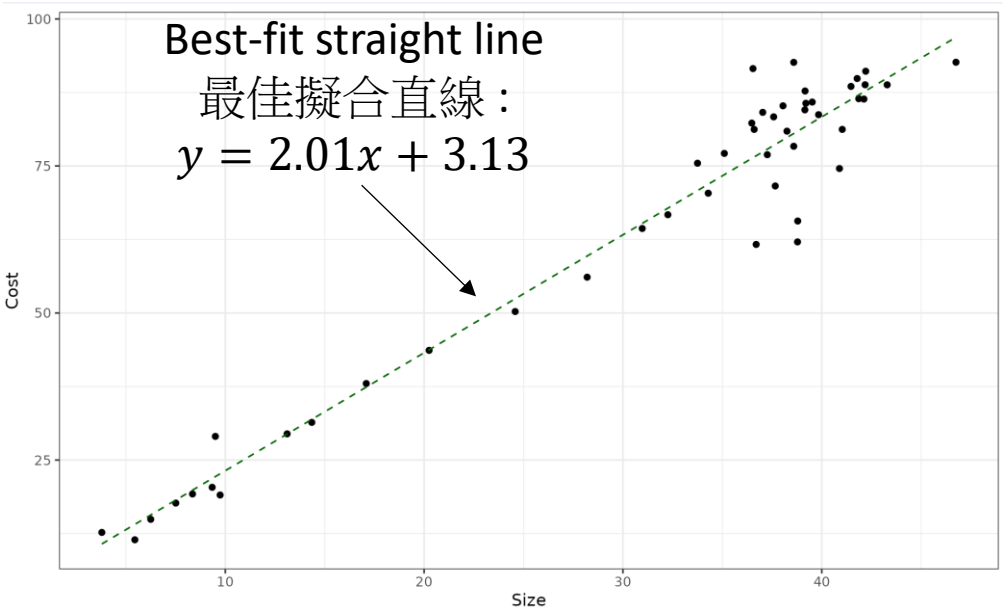
```
>>> from sklearn.metrics import
                              root_mean_squared_error
>>> root_mean_squared_error(y_actual, y_predict)
0.612...
```

Least-squares

- **Recall linear regression:**
  The "best-fit" straight line is defined as the line $y = a + bx$ that minimizes the residual sum of squares *RSS*, where
  最佳擬合直線定義為最小化殘差平方和 *RSS* 的直線 $y = a + bx$ ，其中

$$RSS = \left(y_1 - (a + bx_1)\right)^2 + \left(y_2 - (a + bx_2)\right)^2 + \left(y_3 - (a + bx_3)\right)^2 + \cdots + \left(y_n - (a + bx_n)\right)^2$$

  - $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ are the given data points 是給定的數據點



- Equivalently, we want to find the least-squares solution to the following matrix equation (i.e., minimizing $\left\|M\binom{a}{b} - y\right\|^2$):

$$\underbrace{\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}}_{M} \begin{bmatrix} a \\ b \end{bmatrix} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}}_{\mathbf{y}}$$

Least-squares

- Note that finding the least-squares solution to Ax = b is equivalent to solving the normal equation $A^T Ax = A^T b$.
- Therefore, we have $\boxed{\begin{bmatrix} a \\ b \end{bmatrix} = (M^T M)^{-1} M^T \mathbf{y}}$ where $M = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$ and $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$.
- Similarly, for **multiple regression problem**,
  - The function has the form $y = f(x_1, x_2, \ldots, x_d) = a_0 + a_1 x_1 + a_2 x_2 + \cdots + a_d x_d$.
  - We want to minimize $RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - a_0 - a_1 x_{i,1} - a_2 x_{i,2} - \cdots - a_d x_{i,d})^2$

$$\begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & x_{n,2} & \cdots & x_{n,d} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$\boxed{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{bmatrix} = (M^T M)^{-1} M^T \mathbf{y}}$$

Weighted least-squares

- In the regression models we considered previously, we usually treat all data points "**equally**". What if we know that there are some data points that are more reliable or less reliable?

- In this case, we can make use of such additional information and consider weighted least-squares.

- Consider the following matrix formulation with a weight matrix

$$WM\mathbf{c} = W\mathbf{y}, \quad \text{where} \quad W = \begin{bmatrix} w_1 & & & \\ & w_2 & & \\ & & \ddots & \\ & & & w_n \end{bmatrix}$$

- This corresponds to the weighted residual sum of squares

$$\sum_{i=1}^{n} w_i^2 (y_i - \hat{y}_i)^2$$

Weighted least-squares

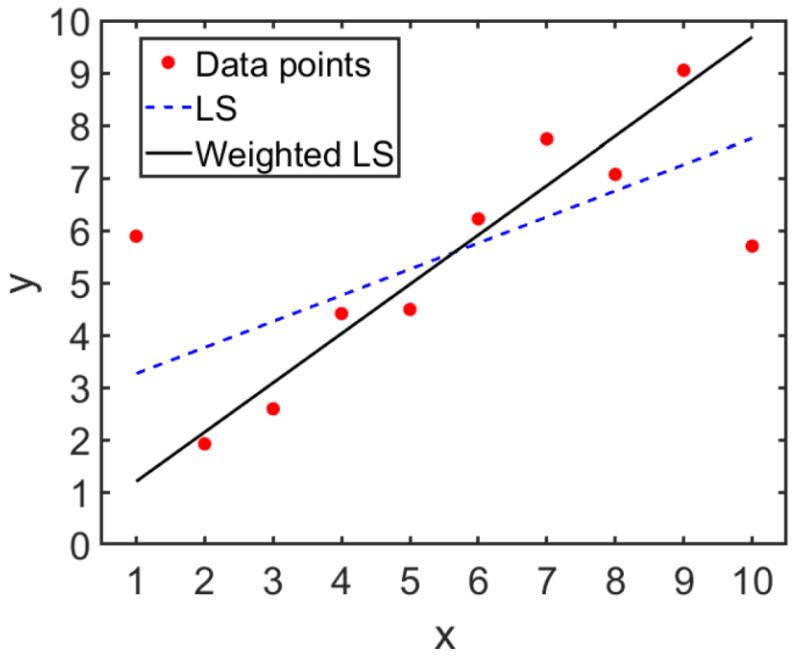
- The least-squares solution will be given by the weighted normal equations:

$$\mathbf{c} = \left((WM)^T(WM)\right)^{-1}(WM)^T(W\mathbf{y}) \quad = \boxed{(M^T W^2 M)^{-1} M^T W^2 \mathbf{y}}$$

- Example: Consider 10 data points:
$(x_i, y_i) = (1, 5.89), (2, 1.92), (3, 2.59), (4, 4.41), (5, 4.49),$
$(6, 6.22), (7, 7.74), (8, 7.07), (9, 9.05), (10, 5.7)$

Setting $w_i = 1$ for all points yields the usual linear regression result.

If we know that the first and last points are less reliable, we can set $w_1, w_{10} = 1/4$ and $w_2, \ldots, w_9 = 1$ and get a weighted least-squares result.



Least-squares and weighted least-squares

- For more details, see e-book Ch. 8.2, 9.4: https://www.math.cuhk.edu.hk/~mathcal/MM/MMLA/

- Computation: See Python notebook https://colab.research.google.com/github/CUHKMathModel/Python/blob/main/Ordinary_Least_Square.ipynb

- Some example commands for solving $Mv = y$ in the least-squares sense:
  - v = numpy. linalg .inv(M.T @ M) @ M.T @ y
  - v = numpy. linalg .lstsq(M,y)[0]

Regularized regression

- Problem: Sometimes the trained model may overfit the data.
  Regularization:
  - Try to prevent overfitting by adding penalty terms to the original cost function
  - Goal: Discourage complex models

- Consider an overdetermined system $A\mathbf{x} = \mathbf{b}$.
  Ordinary least squares (OLS) seeks to minimize RSS $= \|A\mathbf{x} - b\|^2$

- **Tikhonov regularization**:
  To give preference to a solution with some desirable properties, we can consider including a regularization term in this minimization:

$$\boxed{\|A\mathbf{x} - \mathbf{b}\|^2 + \|\Gamma\mathbf{x}\|^2}$$

  for some suitably chosen Tikhonov matrix Γ.

Ridge regression

- Note that we can rewrite

$$\|A\mathbf{x} - \mathbf{b}\|^2 + \|\Gamma\mathbf{x}\|^2 = \|\widetilde{A}\mathbf{x} - \widetilde{\mathbf{b}}\|^2$$

where $\widetilde{A} = \begin{bmatrix} A \\ \Gamma \end{bmatrix}$ and $\widetilde{\mathbf{b}} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix}$. Therefore, the least-squares solution to the new problem is

$$\mathbf{x} = \left(\widetilde{A}^T\widetilde{A}\right)^{-1}\widetilde{A}^T\widetilde{\mathbf{b}} = \boxed{\left(A^TA + \Gamma^T\Gamma\right)^{-1}A^T\mathbf{b}}$$

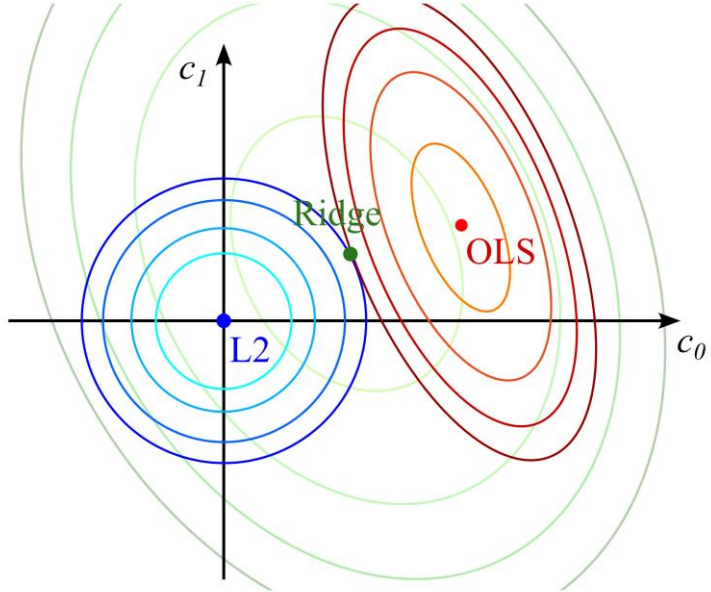
- **Ridge regression** (also known as L_2 **regularization**): $\boxed{\Gamma = \alpha I}$

In other words, the new cost function is $\|M\mathbf{c} - \mathbf{y}\|^2 + \alpha^2\|\mathbf{c}\|^2$

and the least-squares solution is $\boxed{\mathbf{c} = \left(M^TM + \alpha^2 I\right)^{-1}M^T\mathbf{y}}$

Ridge regression

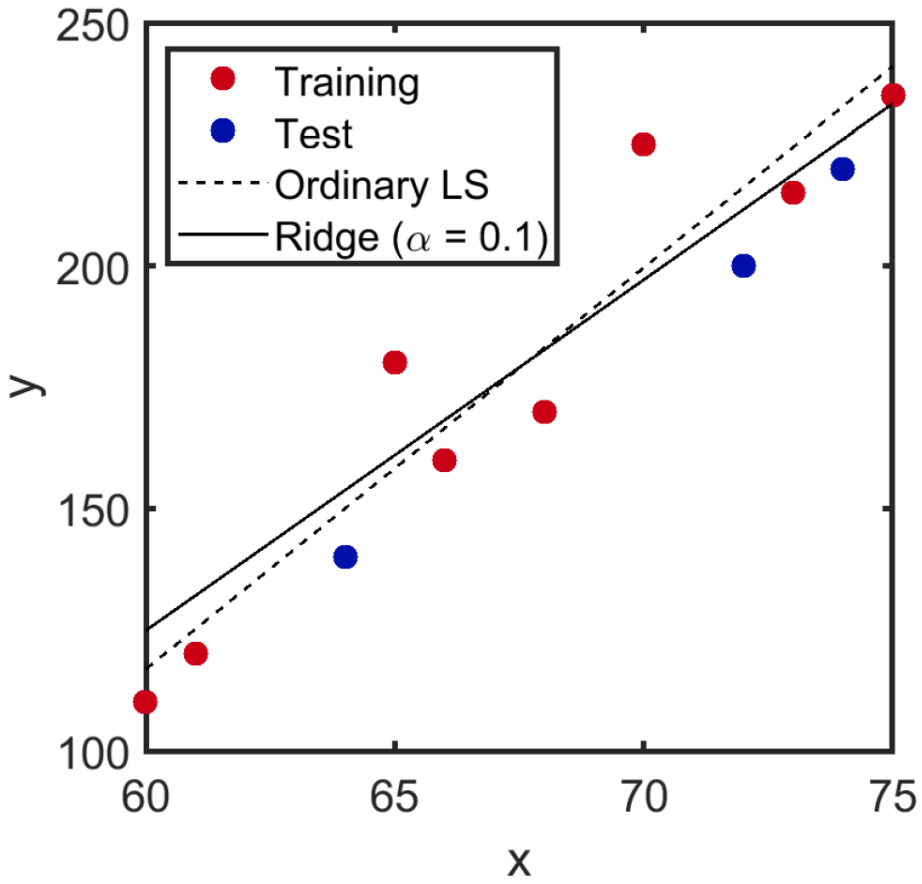
- Remarks:
  - Ridge regression gives preference to coefficients $\mathbf{c}$ with smaller norm because of the $+\alpha^2\|\mathbf{c}\|^2$ term
  - Hence, less complex model
  - Larger $\alpha \Rightarrow$ More regularized
  - Smaller $\alpha \Rightarrow$ Fit the training data more



- How to choose a suitable regularization parameter α?
  A possible method is to use cross-validation:
  - Consider a certain range of α values
  - Perform cross-validation and compute the cross-validation error for each α
  - Select the value that yields the best average performance

Ridge regression

- Example: Consider
  - Training set: $(60, 110)$, $(61, 120)$, $(65, 180)$, $(66, 160)$, $(68, 170)$, $(70, 225)$, $(73, 215)$, $(75, 235)$
  - Test set: $(64, 140)$, $(72, 200)$, $(74, 220)$
- Ordinary least-squares linear model:
  - $y = -379.75 + 8.28x$
  - MSE on test set $= 174.68$
- Linear model with ridge regression ($\alpha = 0.1$):
  - $y = -309.24 + 7.23x$
  - MSE on test set $= 120.43$

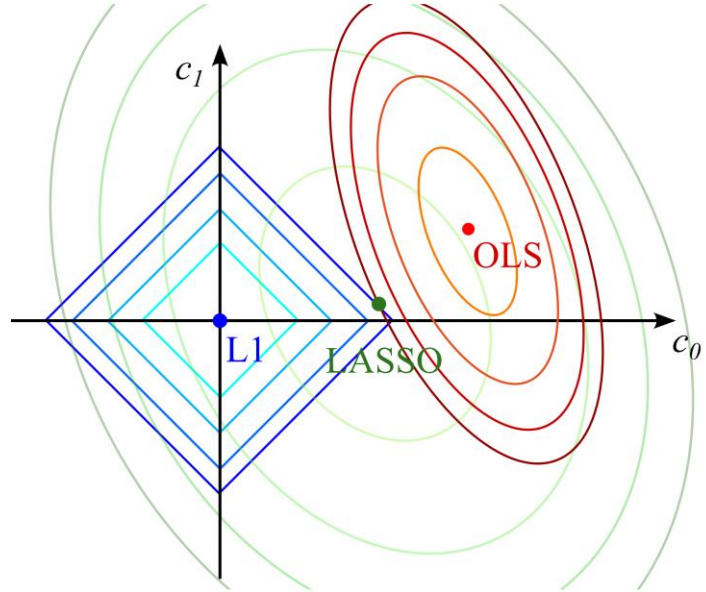


LASSO (L_1 regularization)

- **LASSO (Least Absolute Shrinkage and Selection Operator)** (also known as $\boldsymbol{L_1}$ **regularization**):
  - Similar to ridge regression
  - Replace the regularization term $\|\cdot\|^2$ with L_1-norm: $\|\cdot\|_1$
- In other words, the new cost function is

$$\|M\mathbf{c} - \mathbf{y}\|^2 + \alpha^2\|\mathbf{c}\|_1$$

- Features:
  - More difficult to optimize; no simple closed form using the matrix inverse
  - More evenly distributed magnitudes
  - Biases toward sparser solutions (simpler to understand and more efficient to use)!



Computation of regularized regression

- Ridge regression in Python:

```
>>> from sklearn.linear_model import Ridge
>>> ridgeReg = Ridge(alpha=10) # Can be other values
>>> ridgeReg.fit(X_train,y_train)
```

- LASSO regression in Python:

```
>>> from sklearn.linear_model import Lasso
>>> lasso = Lasso(alpha=10) # Can be other values
>>> lasso.fit(X_train,y_train)
```

Mathematical Modelling @ CUHK Mathematics:
https://www.math.cuhk.edu.hk/app/mathmodel

Contact:
CUHK Mathematical Modelling Project Team
mathmodel@math.cuhk.edu.hk

Thank you!