

作者	生姜 DrGinger
脚本	生姜 DrGinger
视频	崔崔 CuiCui
开源学习资源	https://github.com/Visualize-ML
平台	https://www.youtube.com/@DrGinger_Jiang https://space.bilibili.com/3546865719052873 https://space.bilibili.com/513194466

10.2 多元回归



本节你将掌握的核心技能：

- ▶ 掌握二元与多元线性回归模型的代数结构。
- ▶ 几何视角理解回归平面与超平面如何表示变量间关系。
- ▶ 设计矩阵的结构，包含常数列与各个特征列。
- ▶ 正交投影角度理解多元回归。
- ▶ 最小二乘拟合的原理。
- ▶ 使用 QR 分解求解回归问题。

上一节，我们用正交投影解析了一元线性回归。

这一节，我们继续升维，将目光投向二元线性回归、多元线性回归！

⚠ 注意，非常建议大家平行阅读本节和上一节。

二元线性回归

有了一元线性回归，下面让我们升维，看看二元线性回归！

二元线性回归用于建立一个因变量 y 与两个自变量 (x_1, x_2) 之间的线性关系模型。

如图 1 所示，二元线性回归的表达式如下：

$$y = b_0 + b_1x_1 + b_2x_2 + \varepsilon \quad (1)$$

其中， b_0 为截距项， b_1 、 b_2 代表自变量系数， ε 为残差项。

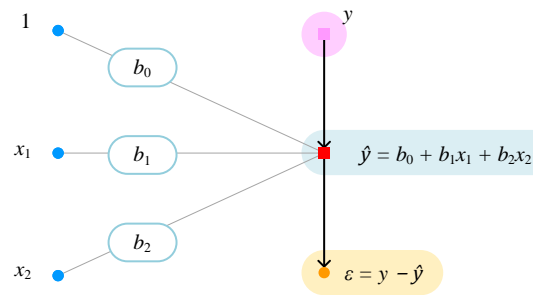


图 1. 二元线性回归变量关系

一个平面

几何角度来看，二元线性回归得到一个三维空间的平面 $\hat{y} = b_0 + b_1x_1 + b_2x_2$ 。

图 2 所示为二元线性回归中散点和回归平面的关系。

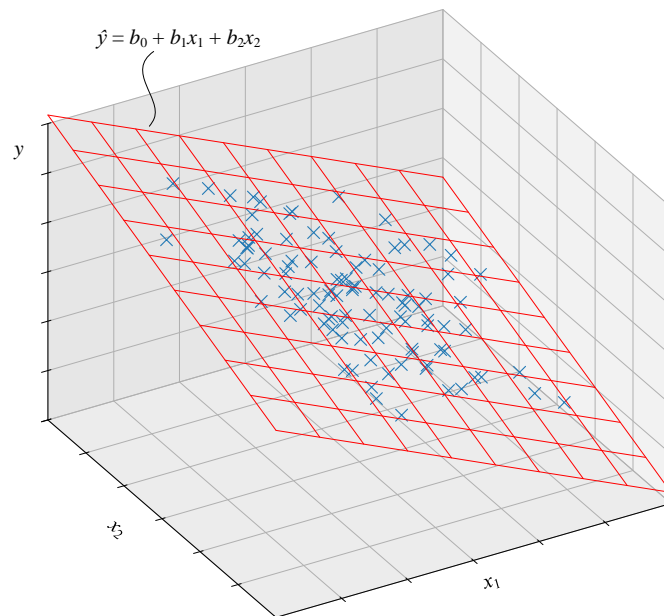
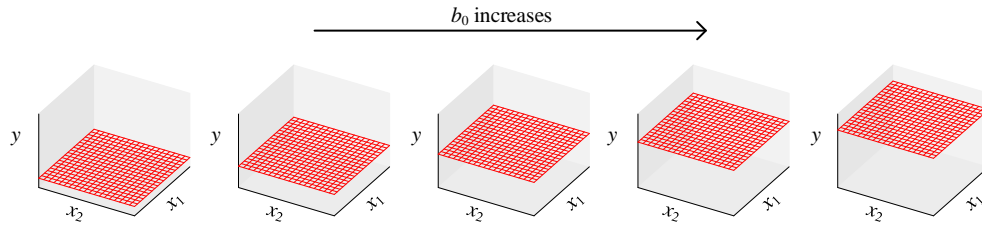


图 2. 二元线性回归对应的平面

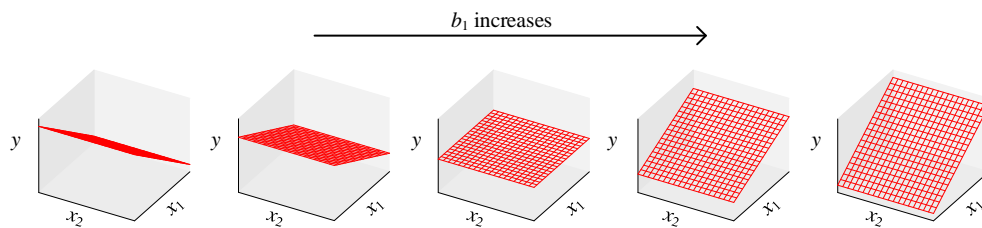
下面看看参数 b_0 、 b_1 、 b_2 如何影响 $\hat{y} = b_0 + b_1x_1 + b_2x_2$ 平面。

b_0 是截距项， b_0 控制这个平面的整体高度，也就是它在竖直方向上整体上移或下移，但不会改变平面的倾斜方向。

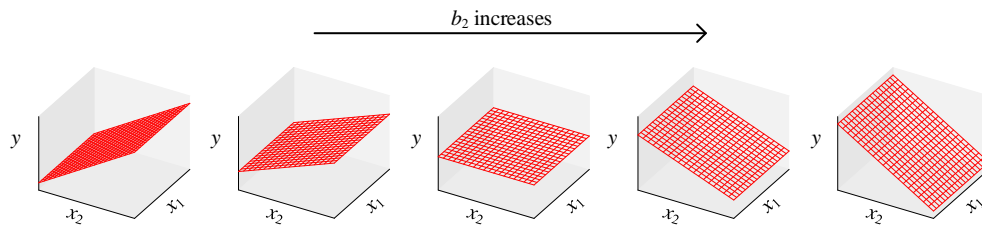
图 3. b_0 控制 $\hat{y} = b_0 + b_1x_1 + b_2x_2$ 平面的整体高度, $b_1 = 0, b_2 = 0$

b_1 、 b_2 是斜率系数。

b_1 绝对值控制 x_1 方向上的斜率, b_1 正负决定了在 x_1 增大时, 平面上升还是下降。

图 4. b_1 控制 $\hat{y} = b_0 + b_1x_1 + b_2x_2$ 平面沿 x_1 方向的斜率, $b_0 = 0, b_2 = 0$

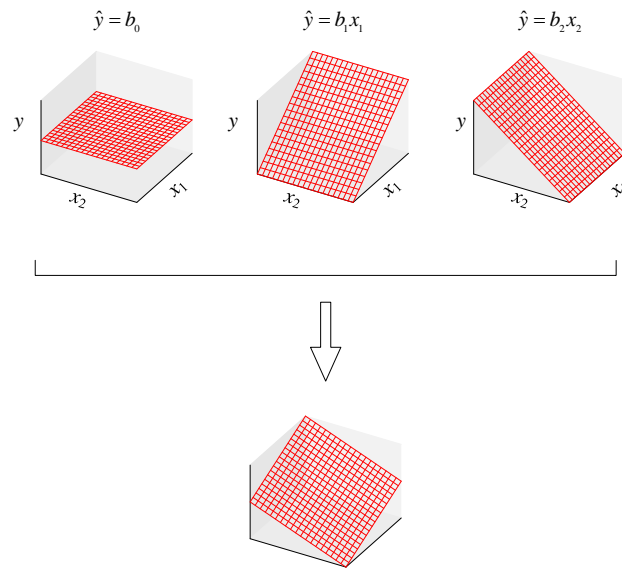
b_2 绝对值则控制平面在 x_2 方向上的斜率, b_2 正负决定了在 x_2 增大时, 平面是上升还是下降。

图 5. b_2 控制 $\hat{y} = b_0 + b_1x_1 + b_2x_2$ 平面沿 x_2 方向的斜率, $b_0 = 0, b_1 = 0$

因此, b_1 、 b_2 共同决定了这个平面的“倾斜方向”和“陡峭程度”, 而 b_0 决定它在空间中的垂直位置。

而二元线性回归就是找到合适的 b_0 、 b_1 、 b_2 , 让图 2 中平面尽可能贴合数据散点。

如图 1 所示, $\hat{y} = b_0 + b_1x_1 + b_2x_2$ 也可以看作三个平面的叠加。这三个平面内分别是: a) $\hat{y} = b_0$; b) $\hat{y} = b_1x_1$; c) $\hat{y} = b_2x_2$ 。

图 6. 三个平面叠加得到 $\hat{y} = b_0 + b_1x_1 + b_2x_2$

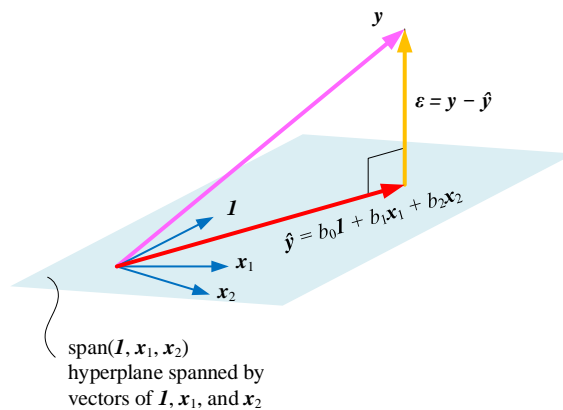
让我们同样用上一节介绍的正交投影视角来看二元线性回归。

正交投影

二元线性回归中，真实值列向量 y 可以写成

$$y = \hat{y} + \varepsilon = b_0\mathbf{I} + b_1x_1 + b_2x_2 + \varepsilon \quad (2)$$

如图 7 所示， \hat{y} 是 y 在 \mathbf{I} 、 x_1 、 x_2 张成的平面 $\text{span}(\mathbf{I}, x_1, x_2)$ 中的投影。

图 7. y 朝 $\text{span}(\mathbf{I}, x_1, x_2)$ 投影，二元线性回归

也就是说， \hat{y} 是 \mathbf{I} 、 x_1 、 x_2 三个向量的线性组合

$$\hat{y} = b_0\mathbf{I} + b_1x_1 + b_2x_2 \quad (3)$$

图 8 所示二元线性回归中设计矩阵 X 的形状。二元线性回归有两个变量，因此 X 有三个列向量。

⚠ 注意，这里默认 \mathbf{I} 、 \mathbf{x}_1 、 \mathbf{x}_2 线性无关；这样设计矩阵 \mathbf{X} 列满秩，格拉姆矩阵 $\mathbf{X}^T \mathbf{X}$ 满秩；因此， $\mathbf{X}^T \mathbf{X}$ 可逆。实践中，列向量可以线性相关。这种线性回归需要特别处理。

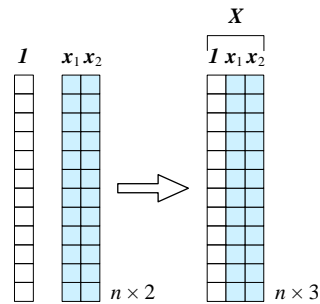


图 8. 设计矩阵 \mathbf{X} ，二元线性回归

把 (3) 写成矩阵乘法形式

$$\hat{\mathbf{y}} = \underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{x}_1 & \mathbf{x}_2 \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}}_{\mathbf{b}} = \mathbf{X}\mathbf{b} \quad (4)$$

(4) 对应图 9。

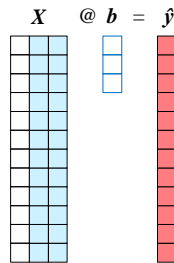


图 9. 矩阵运算得到 $\hat{\mathbf{y}}$ ，二元线性回归

图 10 展示如何计算二元线性回归中 \mathbf{b} 。

? 请大家把图 10 对应的矩阵运算写出来，并根据前文内容推导图 10 中矩阵乘法。

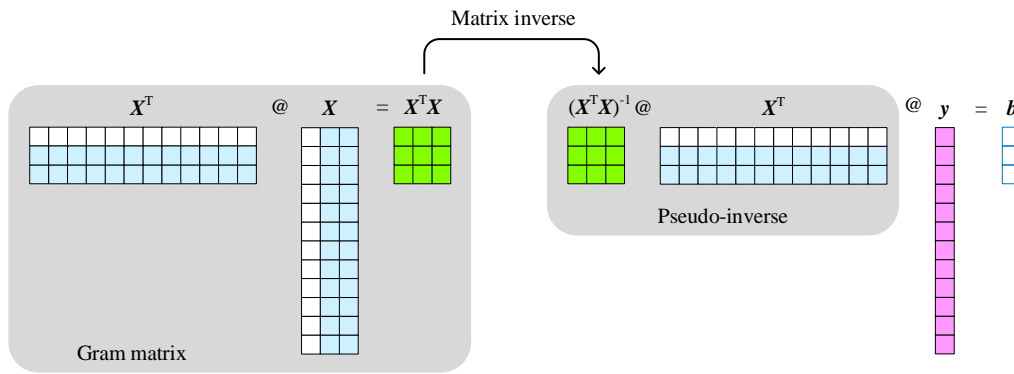
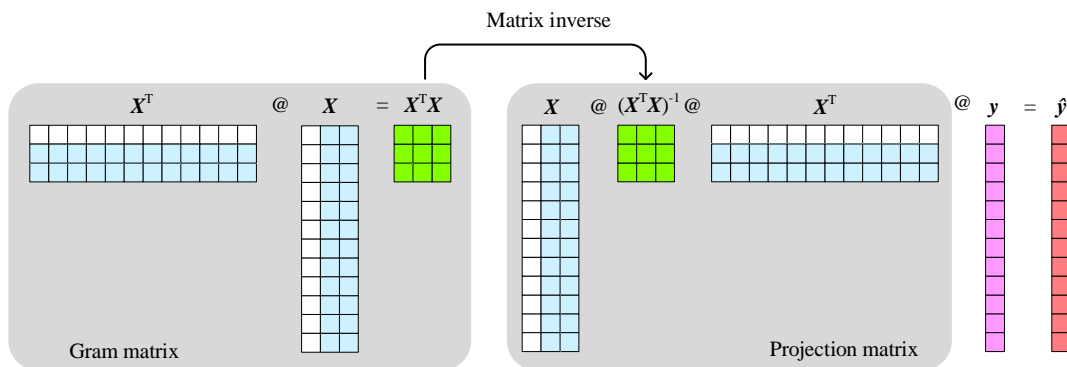
图 10. 计算 b ，二元线性回归

图 11 展示如何计算二元线性回归中 \hat{y} 。

? 请大家把图 11 对应的矩阵运算写出来。

图 11. 计算 \hat{y} ，二元线性回归

编程实现二元线性回归

代码 1 完成二元线性回归模型构造，并绘制图 2。

下面聊聊其中关键语句。

a 创建样本数据。其中，`np.random.seed(0)` 是为了让生成的随机数“可重复”，即每次运行这段代码都会产生相同的随机数据，方便调试和对比。`n_samples = 100` 设定生成 100 个样本点，这个变量就是样本数量。`np.random.randn(n_samples, 2)` 生成一个 100 行 2 列的随机数组，里面的数是标准正态分布的随机数。

标准正态分布 (standard normal distribution) 是均值为 0、标准差为 1 的正态分布 (normal distribution)。正态分布是一种对称的钟形概率分布，描述大量自然现象中数据围绕均值集中、两侧逐渐减少的规律。

➔ 正态分布，也叫高斯分布，是“数学不难”系列中重要的话题之一。我们将会在《概率统计不难》中讲解。

中最后一句生成因变量数据。具体来说， y 是 x_1 数据乘以 -3，加上 x_2 数据乘以 2，再加上常数 1，然后再加上一一点随机噪声 (也是 `np.random.randn()` 生成)，模拟现实中数据的不确定性。

b 构造“设计矩阵”，用于回归计算。我们把一列全是 1 的向量 (表示常数项) 和 x_1 、 x_2 这两列样本数据拼接在一起，形成一个 100 行 3 列的设计矩阵 X 。`np.ones_like(y)` 生成与 y 同样形状的全 1 列向量，`np.column_stack()` 用于按列拼接数组。

c 已经在上一节讲过，用来求解线性回归。这句对应图 10。

d 生成网格数据，用来绘制网格面。注意，我们在网格点数据矩阵前面加上一列 1，构造成跟训练数据一样的格式，也就是加上常数项，用来做预测。


e 用计算得到的列向量 b ，对网格点进行预测。这句对应图 11。

f 绘制训练样本数据的三维散点图。

g 用线框图的方式画出回归平面，显示预测值随 x_1 、 x_2 的变化。最后可视化的结果如图 2 所示。



请大家自学 `LA_09_07_02.ipynb`。代码使用的是 `Scikit-learn` 提供的 `LinearRegression` 工具，完成二元线性回归问题的求解。背后的数学原理与最小二乘法完全一致，结果也和 `LA_09_07_01.ipynb` 完全一致。请大家对比学习。

代码 1. 二元线性回归 |  `LA_09_07_01.ipynb`

```

## 初始化
import numpy as np
import matplotlib.pyplot as plt

## 创建数据
a np.random.seed(0)
  n_samples = 100
  x_12 = np.random.randn(n_samples, 2)
  y = -3*x_12[:,0] + 2*x_12[:,1]+1+0.25*np.random.randn(n_samples)

# 构造设计矩阵，增加全1列向量
b X = np.column_stack((np.ones_like(y), x_12))

## 创建线性回归模型并拟合数据
c b = np.linalg.inv(X.T @ X) @ X.T @ y

## 拟合
# 生成回归平面的数据点
d x1_grid, x2_grid = np.meshgrid(np.linspace(-3, 3, 10),
                                np.linspace(-3, 3, 10))
  X_grid = np.column_stack((x1_grid.flatten(), x2_grid.flatten()))
  X_grid = np.column_stack((np.ones_like(X_grid[:,0]), X_grid))

# 预测回归平面上的响应变量
e y_pred = X_grid @ b
  y_pred = y_pred.reshape(x1_grid.shape)

## 可视化
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
# 绘制三维样本散点
f ax.scatter(x_12[:,0], x_12[:,1], y, marker = 'x')

# 绘制回归平面
g ax.plot_wireframe(x1_grid, x2_grid, y_pred, colors = 'r')

ax.set_xlabel('$x_1$'); ax.set_ylabel('$x_2$')
ax.set_zlabel('$y$')
ax.set_xlim([-3,3]); ax.set_ylim([-3,3])
ax.set_zlim([-15,15])
ax.set_proj_type('ortho'); ax.view_init(azim=-120, elev=30)

```



多元线性回归

让我们继续“升维”，聊聊多元线性回归！

多元线性回归用于建立一个因变量与多个自变量之间的线性关系模型。

如图 12 所示，多元线性回归的表达式如下：

$$y = b_0 + b_1x_1 + b_2x_2 + \dots + b_Dx_D + \varepsilon \quad (5)$$

其中， b_0 为截距项， b_1, b_2, \dots, b_D 代表自变量系数， ε 为残差项， D 为自变量个数。

几何角度来看，多元线性回归得到一个超平面 (hyperplane)。超平面是三维空间中平面的一种推广。

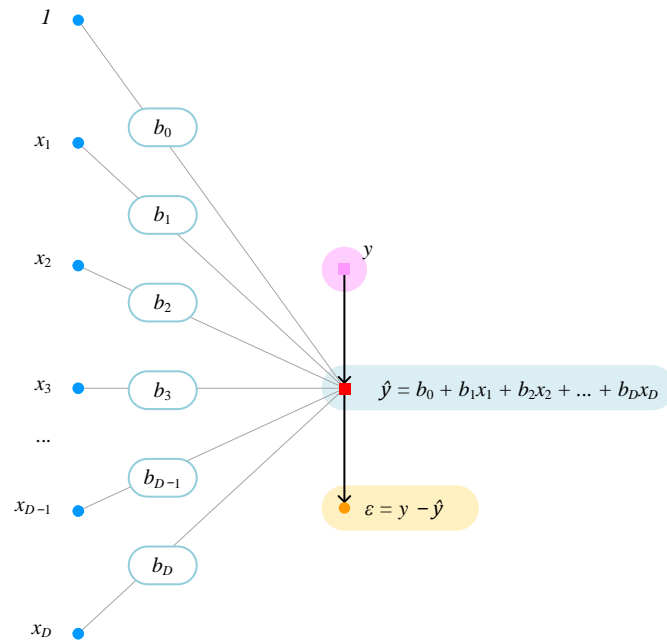


图 12. 多元线性回归变量关系

正交投影

从向量空间角度来看，如图 13 所示， I, x_1, x_2, \dots, x_D 张成 $\text{span}(I, x_1, x_2, \dots, x_D)$ 。

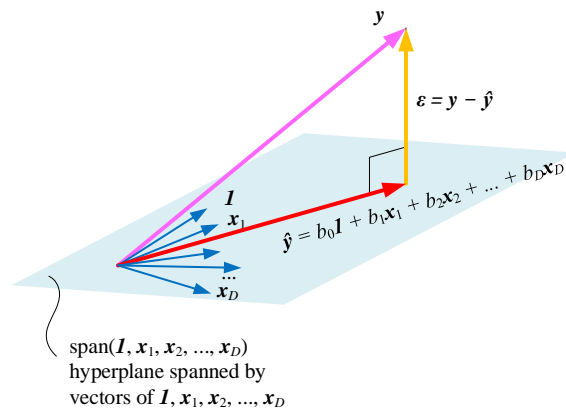


图 13. 向量 y 向超平面 $\text{span}(I, x_1, x_2, \dots, x_D)$ 投影，多元线性回归

向量 y 在 $\text{span}(I, x_1, x_2, \dots, x_D)$ 上投影结果为 \hat{y} 。

也就是说， \hat{y} 可以写成 I, x_1, x_2, \dots, x_D 的线性组合

$$\hat{y} = b_1 x_1 + b_2 x_2 + \dots + b_D x_D \quad (6)$$

误差 $y - \hat{y}$ 垂直 $\text{span}(I, x_1, x_2, \dots, x_D)$ ，也就是说 $y - \hat{y}$ 分别垂直于 x_1, x_2, \dots, x_D ，即：

$$\begin{cases} \mathbf{x}_1^T (\mathbf{y} - \hat{\mathbf{y}}) = 0 \\ \mathbf{x}_2^T (\mathbf{y} - \hat{\mathbf{y}}) = 0 \\ \vdots \\ \mathbf{x}_D^T (\mathbf{y} - \hat{\mathbf{y}}) = 0 \end{cases} \Rightarrow \begin{matrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_D^T \\ \mathbf{X}^T \end{matrix} (\mathbf{y} - \hat{\mathbf{y}}) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (7)$$

$\text{span}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_D)$ 和 $\text{span}(\boldsymbol{\varepsilon})$, 即 $\text{span}(\mathbf{y} - \hat{\mathbf{y}})$, 互为正交补。

图 14 所示为多元线性回归的设计矩阵 \mathbf{X} , 即

$$\mathbf{X}_{n \times (D+1)} = [\mathbf{I} \quad \mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_D] \quad (8)$$

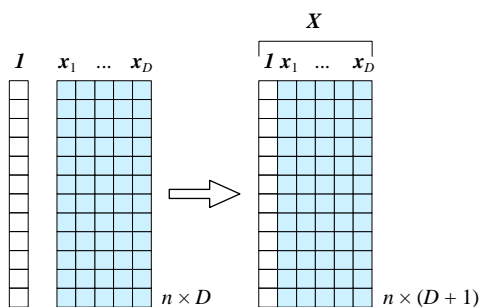


图 14. 设计矩阵 \mathbf{X} , 多元线性回归

写成矩阵乘法形式

预测值构成的列向量 $\hat{\mathbf{y}}$, 通过下式计算得到:

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{b} \quad (9)$$

以上矩阵乘法对应图 15。

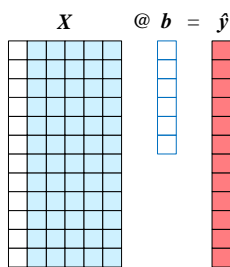
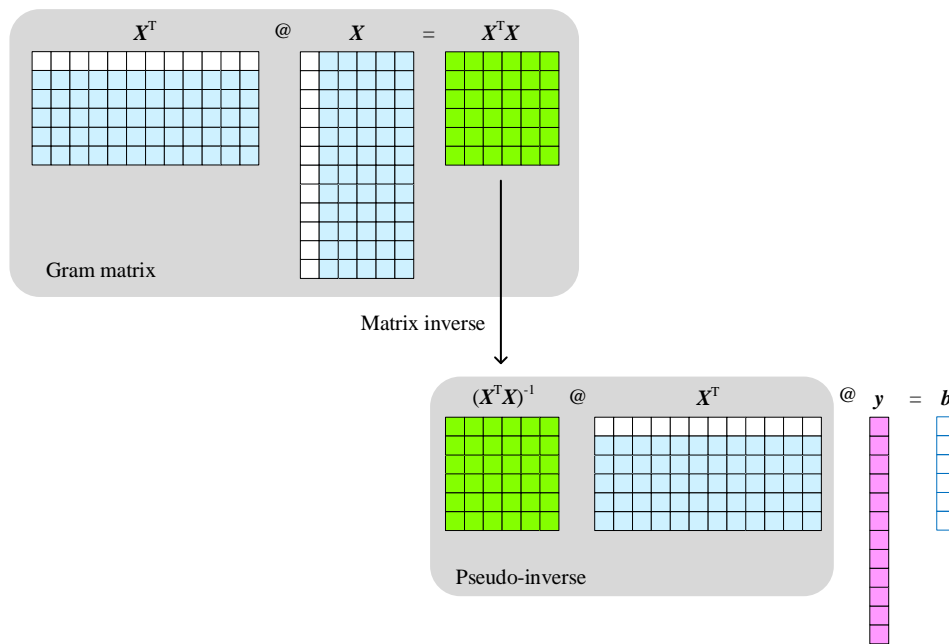


图 15. 矩阵运算得到 $\hat{\mathbf{y}}$, 多元线性回归

广义逆

如图 16 所示, 如果 $\mathbf{X}^T \mathbf{X}$ 可逆, \mathbf{b} 的解为

$$\mathbf{b} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (10)$$

图 16. 计算 b ，多元线性回归

超定方程组

实际上求解 b 就是求解如下超定方程组

$$Xb = y \quad (11)$$

本书前文提过，超定方程组是指方程个数多于未知数个数的方程组。

(11) 等式左右乘 X^T 得到

$$X^T X b = X^T y \quad (12)$$

而 $X^T X$ 便是 X 的格拉姆矩阵。

只有格拉姆矩阵 $X^T X$ 满秩 (X 对应列满秩)， $X^T X$ 才可逆。(12) 左右乘 $X^T X$ 的逆，我们便得到(10)。

这是从超定方程组视角来求解 b 。

QR 分解

对设计矩阵 X 进行完全型 QR 分解

$$X = QR \quad (13)$$

Q 为正交矩阵，是个方阵。把上式写成分块矩阵乘法

$$X = [Q_1 \quad Q_2] \begin{bmatrix} R_1 \\ O \end{bmatrix} \quad (14)$$

整理之后，我们便得到经济型 QR 分解

$$X = Q_1 R_1 \quad (15)$$

矩阵乘法可以写成

$$\hat{y} = Q_1 R_1 b \quad (16)$$

上式左右乘半正交矩阵 Q_1^T

$$Q_1^T \hat{y} = Q_1^T Q_1 R_1 b = R_1 b \quad (17)$$

由于 R_1 是上三角矩阵，计算逆更方便——这便是 QR 分解在求解方程组的优势所在。

投影矩阵

如图 17 所示，拟合值向量 \hat{y} 为：

$$\hat{y} = Xb = X(X^T X)^{-1} X^T y \quad (18)$$

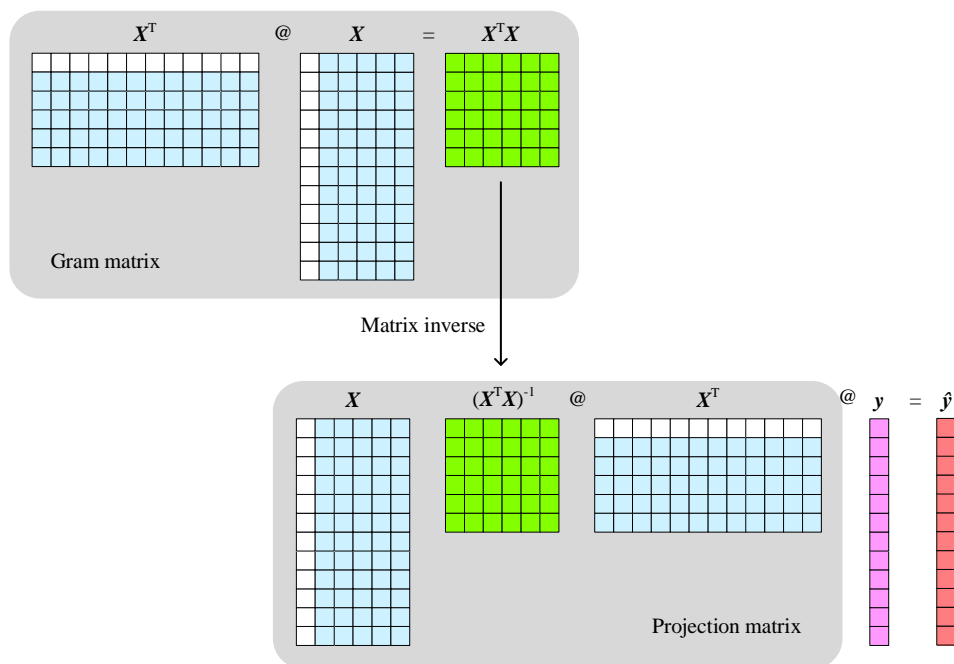


图 17. 计算 \hat{y} ，多元线性回归

回到图 13，我们发现 (18) 中 $X(X^T X)^{-1} X^T$ 就是 y 向 X 列向量张成平面的投影矩阵 (projection matrix)。

上式就是 y 在 $\text{span}(I, x_1, x_1, \dots, x_D)$ 的向量投影！

本书前文提过， $X(X^T X)^{-1} X^T$ 常被称作**帽子矩阵** (hat matrix)。

根据 (18)，残差列向量 $\varepsilon = y - \hat{y}$ 可以写成

$$\varepsilon = y - X(X^T X)^{-1} X^T y = (I - X(X^T X)^{-1} X^T) y \quad (19)$$

回到图 13，我们发现 $\text{span}(\boldsymbol{\varepsilon})$ 是 $\text{span}(\boldsymbol{I}, \boldsymbol{x}_1, \boldsymbol{x}_1, \dots, \boldsymbol{x}_D)$ 的正交补！

换个角度来看， $\boldsymbol{\varepsilon}$ 就是 $\text{span}(\boldsymbol{I}, \boldsymbol{x}_1, \boldsymbol{x}_1, \dots, \boldsymbol{x}_D)$ 平面的法向量。

这意味着，(19) 中 $(\boldsymbol{I} - \boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top)$ 也是一个投影矩阵。这个投影矩阵将 \boldsymbol{y} 投影到“残差”空间，得到残差向量 $\boldsymbol{\varepsilon}$ 。

将 (18)、(19) 相加

$$\hat{\boldsymbol{y}} + \boldsymbol{\varepsilon} = \underbrace{\boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top}_{\text{Projection matrix}} \boldsymbol{y} + \underbrace{(\boldsymbol{I} - \boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top)}_{\text{Projection matrix}} \boldsymbol{y} = \boldsymbol{I} \boldsymbol{y} = \boldsymbol{y} \quad (20)$$

反过来看上式，向量 \boldsymbol{y} 正交分解为：估计值向量 $\hat{\boldsymbol{y}}$ 、误差向量 $\boldsymbol{\varepsilon}$ 。



请大家用 DeepSeek/ChatGPT 等工具完成本节如下习题。

Q1. 以下代码产生多元回归自变量、因变量数据，请大家用 Python 编程计算 \boldsymbol{b} 、 $\hat{\boldsymbol{y}}$ 、 $\boldsymbol{\varepsilon}$ 。

```
## 初始化
import numpy as np

## 创建数据
np.random.seed(0)
n_samples = 100

# 自变量数据
X_vecs = np.random.randn(n_samples, 6)

# 因变量数据
y = X_vecs @ np.array([[-3], [2], [-2], [3], [1], [-1]])
y = y + 1 + 0.25*np.random.randn(n_samples)
```

Q2. 用 `sklearn.linear_model` 模块中的 `LinearRegression` 函数求解 Q1 中的多元线性回归，并比较结果。