

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

Regression 回归

从正交投影角度看线性回归、非线性回归

回归是一种统计方法，用于分析变量之间的关系，尤其是预测一个变量如何随着另一个变量的变化而变化。线性回归是指变量之间呈线性关系的回归模型，非线性回归则用于处理变量关系不是直线形式的复杂模型。本章将从正交投影角度看一元线性回归、二元线性回归、多元线性回归、多项式回归。

10.1 从投影到回归



本节你将掌握的核心技能：

- ▶ 一条直线拟合数据，表示变量之间的线性关系。
- ▶ 截距决定上下位置，斜率决定倾斜程度。
- ▶ 预测值向量是目标向量在解释变量张成空间的正交投影。
- ▶ 利用格拉姆矩阵来求解最优参数。
- ▶ 广义逆在超定方程组中的作用，并能用于回归系数求解。
- ▶ 预测值与残差都由投影矩阵表达，并彼此正交。
- ▶ 最小二乘原理：最小化残差平方和，并可转化为 L2 范数最小化问题。

这一节我们将矩阵乘法、正交投影、正交矩阵、正交补、格拉姆矩阵、秩、逆矩阵、广义逆、超定方程组等线性代数工具用在回归模型这个应用场景中。

什么是线性回归？

“回归”是指建立一个数学模型，用来描述**因变量** (dependent variable) 如何随着另一个或多个**自变量** (independent variable) 的变化而变化。

“线性”是指模型中的变量之间呈现一种线性关系，即因变量是自变量的加权和，即加法与乘法的组合。

线性回归的核心假设是：自变量对因变量的影响是“线性累加”的，而不是指数、乘积、或更复杂的非线性关系。这使得模型既易于解释，又便于计算和分析。

? 下一节将介绍多项式回归，一种常见的非线性回归模型。

本节中，**一元线性回归** (univariate linear regression) 是一种通过一条“直线”来描述两个变量之间关系的数学方法。

“一元线性回归”中的“元”，表示自变量的个数。所以，“一元”就是只有一个自变量参与建模，比如用身高预测体重，其中的身高就是“元”。

如果有两个或更多自变量，就是“二元”或“多元”回归。

想象我们有如图 1 所示的一堆散点。我们发现它们呈现某种“线性”关系，想画一条最合适的直线去“穿过”这些点，尽量贴近每个点，用来预测、解释或理解变量之间的关系。

! 注意，线性回归不代表因果关系。一元线性回归是一种常用的统计建模方法，用来描述一个自变量与一个因变量之间的线性关系。它通过拟合一条最优直线，使得预测值与实际观测值之间的误差最小。线性回归所揭示的只是变量之间的相关性，而非因果关系。即使模型显示自变量、因变量存在线性关联，也不能断定自变量的变化会导致因变量的变化，背后可能存在第三方因素或纯属巧合。

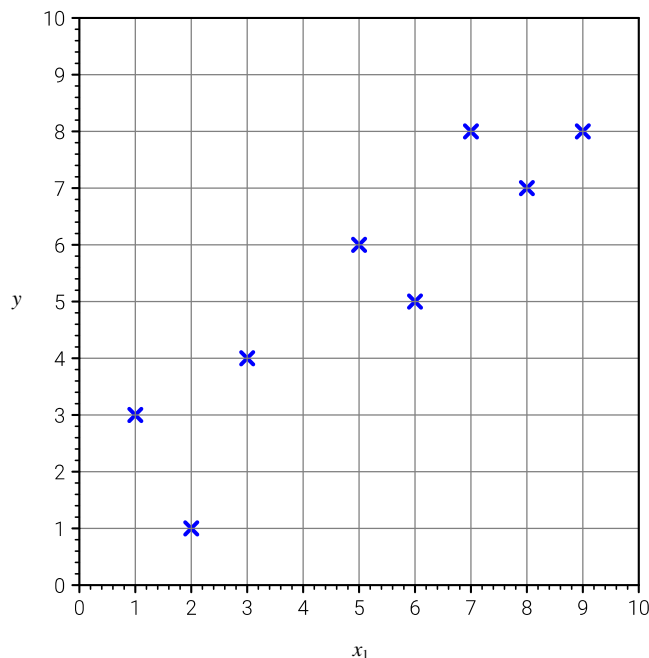


图 1. 呈现“线性”关系的散点数据

图 1 中 x_1 常被称作**自变量** (independent variable)、**解释变量** (explanatory variable) 或**回归元** (regressor)、**外生变量** (exogenous variables)、**预测变量** (predictor variables)。

y 常被称作**因变量** (dependent variable)、**被解释变量** (explained variable)、或**回归子** (regressand)、**内生变量** (endogenous variable)、**响应变量** (response variable) 等。

一条直线

图 2 平面上红色直线，可以写成

$$\hat{y} = b_0 + b_1 x_1 \quad (1)$$

这就是平面直线的**斜截式** (slope-intercept form)， b_0 为**截距** (intercept)， b_1 为**斜率** (slope)。

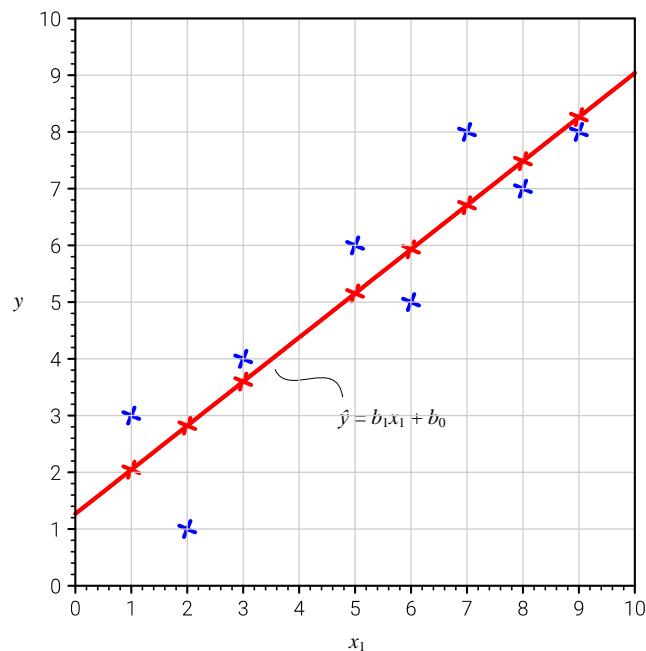


图 2. 一元线性回归

⚠ 注意，我们把 b_0 (截距) 写在前面， b_1 (斜率)，是为了配合本节后文的参数列向量的次序。

图 3 (a) 所示截距对直线的影响。截距 b_0 决定了直线和纵轴交点 (截距) 的位置。

当斜率 b_1 保持不变时，改变截距 b_0 会使整条直线上下平移，但不会改变其倾斜程度。

截距 b_0 为负数时，直线和纵轴的交点在原点之下；截距 b_0 为正数时，直线和纵轴的交点在原点上面。

图 3 (b) 所示为斜率 b_1 对直线的影响。斜率 b_1 决定了直线的倾斜程度。

当截距 b_0 保持不变时，改变斜率 b_1 会使直线以截距为支点旋转。

斜率绝对值越大，直线越陡峭。

斜率 b_1 为零时，直线为水平线。

斜率 b_1 为正数，直线从左下向右上倾斜。

斜率 b_1 为负时，直线从左上向右下倾斜。

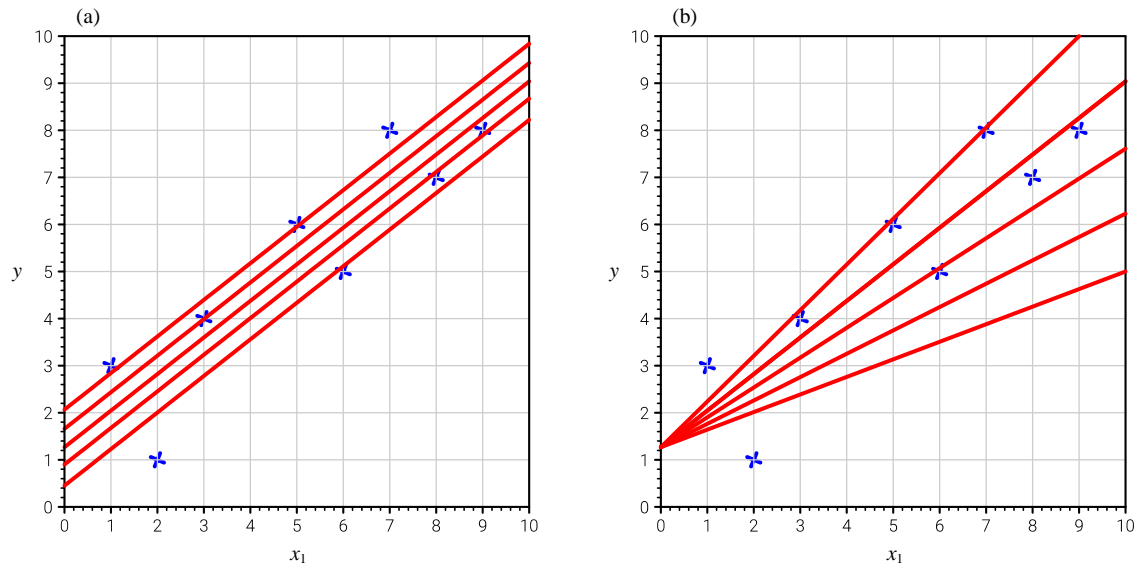


图 3. 截距、斜率对直线的影响

(1) 中给变量 y 带了个“帽子” \hat{y} ，是因为我们把 y 留给了真实值，就是散点纵轴值；而 \hat{y} 代表“估计”值、“预测”值，就是图 5 中红色线对应纵轴值。

真实的 y 、预测的 \hat{y} 之差就是 ε

$$\varepsilon = y - \hat{y} = y - \left(\underbrace{b_0 + b_1 x_1}_{\hat{y}} \right) \quad (2)$$

ε 叫做残差项 (residuals)，也叫误差项 (error term)、干扰项 (disturbance term) 或噪音项 (noise term)。

这样，我们便得到如图 4 所示的变量关系。

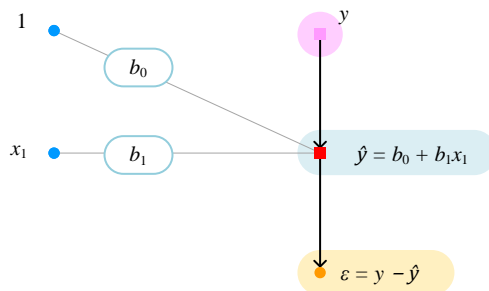


图 4. 一元线性回归变量关系

图 5 中，索引为 i 的散点 (黄色高亮) 横轴值为 $x_1^{(i)}$ ，纵轴值为 $y^{(i)}$ ；而 $x_1^{(i)}$ 对应红色斜线上的点为 $\hat{y}^{(i)}$ ，通过下式计算

$$\hat{y}^{(i)} = b_0 + b_1 x_1^{(i)} \quad (3)$$

显然， $y^{(i)}$ 和 $\hat{y}^{(i)}$ 不贴合！

两者在纵轴高度差就是残差

$$\varepsilon^{(i)} = y^{(i)} - \hat{y}^{(i)} = b_0 + b_1 x_1^{(i)} - \hat{y}^{(i)} \quad (4)$$

也就是说

$$y^{(i)} = b_0 + b_1 x_1^{(i)} + \varepsilon^{(i)} \quad (5)$$

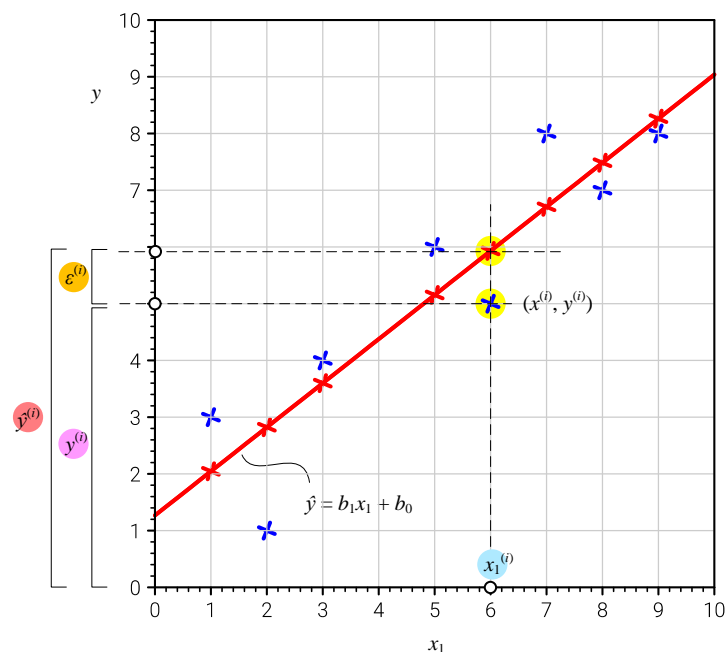


图 5. 一元线性回归中的真实值、估计值、误差

简单来说，我们就是想要找到合适的一条红色线，让图 6 中散点的所有残差 (橙色线段) 尽量小；这样红色线和散点更为贴合。

? 怎么“量化”尽量小？这是本节后文要讨论的话题。

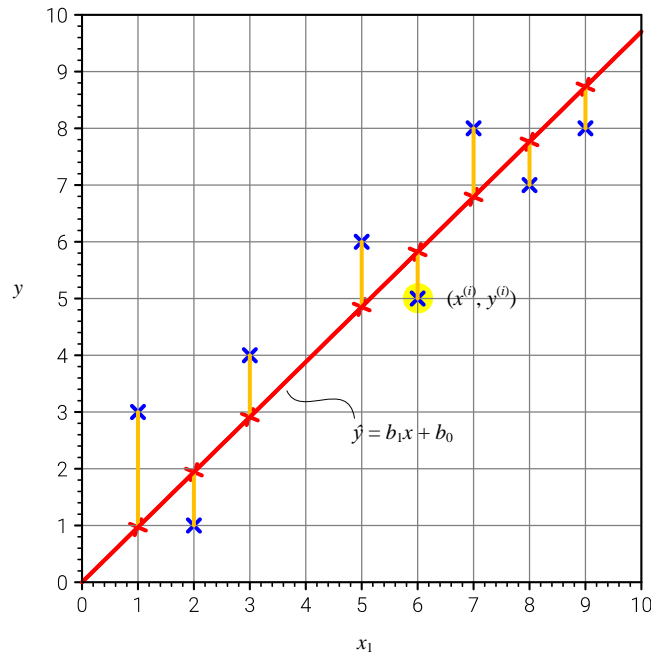


图 6. 一元线性回归中的所有散点的误差 (橙色线段)

写成向量

把散点的横坐标为列向量 \mathbf{x}_1 ，散点纵坐标写成列向量 \mathbf{y} ，红色线代表的估计值写成 $\hat{\mathbf{y}}$ ，残差列向量为 $\boldsymbol{\varepsilon}$ ，即

$$\mathbf{x}_1 = \begin{bmatrix} x_1^{(1)} \\ x_1^{(2)} \\ \vdots \\ x_1^{(i)} \\ \vdots \\ x_1^{(n)} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(i)} \\ \vdots \\ y^{(n)} \end{bmatrix}, \quad \hat{\mathbf{y}} = \begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \vdots \\ \hat{y}^{(i)} \\ \vdots \\ \hat{y}^{(n)} \end{bmatrix}, \quad \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon^{(1)} \\ \varepsilon^{(2)} \\ \vdots \\ \varepsilon^{(i)} \\ \vdots \\ \varepsilon^{(n)} \end{bmatrix} \quad (6)$$

\mathbf{x}_1 对应图 6 中红色 ×、蓝色 × 的横轴坐标。

\mathbf{y} 对应图 6 中蓝色 × 的纵轴坐标。

$\hat{\mathbf{y}}$ 对应图 6 中红色 × 的纵轴坐标。

$\boldsymbol{\varepsilon}$ 对应图 6 中橙色线段。

图 1 中的自变量数据列向量 \mathbf{x} 、因变量数据列向量 \mathbf{y} 具体数值为

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 3 \\ 1 \\ 4 \\ 6 \\ 5 \\ 8 \\ 7 \\ 8 \end{bmatrix} \quad (7)$$

红色直线对应的向量加法

$$\begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \vdots \\ \hat{y}^{(i)} \\ \vdots \\ \hat{y}^{(n)} \end{bmatrix} = b_0 \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix} + b_1 \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(i)} \\ \vdots \\ x^{(n)} \end{bmatrix} \quad (8)$$

$\hat{\mathbf{y}} \qquad \mathbf{1} \qquad \mathbf{x}_1$

即

$$\hat{\mathbf{y}} = b_0 \mathbf{1} + b_1 \mathbf{x}_1 \quad (9)$$

其中， $\mathbf{1}$ 为和 \mathbf{x}_1 形状相同的全 $\mathbf{1}$ 列向量。

? 那么问题来了，如何找到合适的 b_0 、 b_1 ？这和本章前文讲解的正交投影又有什么关系？

正交投影找 b_0 、 b_1

秘密就藏在 (9) 这个式子中！

这种结构的式子，我们太熟悉了，这就是我们在本书前文反复提过的线性组合。

也就是说， $\hat{\mathbf{y}}$ 在 $\mathbf{1}$ 、 \mathbf{x}_1 张成的平面 $\text{span}(\mathbf{1}, \mathbf{x}_1)$ 中！

⚠ 注意，我们默认 $\mathbf{1}$ 、 \mathbf{x}_1 线性无关。

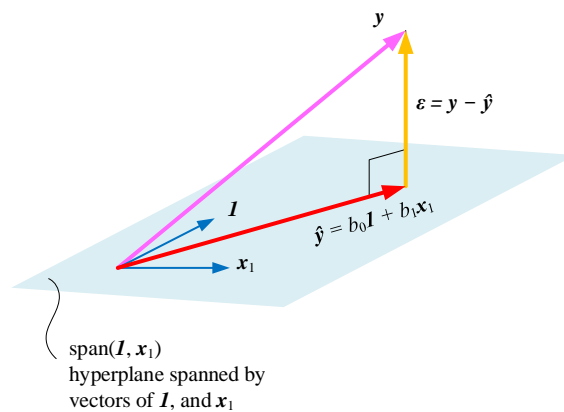


图 7. 向量 \mathbf{y} 向平面 $\text{span}(\mathbf{1}, \mathbf{x}_1)$ 投影，一元线性回归

如图 7 所示， $\hat{\mathbf{y}}$ 可以写成 $\mathbf{1}$ 、 \mathbf{x}_1 的线性组合，即

$$\hat{\mathbf{y}} = b_0 \mathbf{1} + b_1 \mathbf{x} \quad (10)$$

真实值列向量 \mathbf{y} 则在平面 $\text{span}(\mathbf{1}, \mathbf{x}_1)$ 之外！

y 向平面 $\text{span}(\mathbf{I}, \mathbf{x}_1)$ 正交投影得到 \hat{y} 。 \hat{y} 在 $\text{span}(\mathbf{I}, \mathbf{x}_1)$ 中， \hat{y} 中垂直 $\text{span}(\mathbf{I}, \mathbf{x}_1)$ 的分量为 ε 。

也就是说， y 正交分解成两部分，即 \hat{y} 、 ε 。

\hat{y} 、 ε 两者之和构造 y ，即

$$\mathbf{y} = \hat{\mathbf{y}} + \varepsilon = b_0 \mathbf{I} + b_1 \mathbf{x} + \varepsilon \quad (11)$$

整理上式，我们可以得到误差向量 ε 为

$$\varepsilon = \mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - (b_0 \mathbf{I} + b_1 \mathbf{x}) \quad (12)$$

误差向量 ε 垂直于平面 $\text{span}(\mathbf{I}, \mathbf{x}_1)$ 。这一点很重要，我们马上就会用到。

写成矩阵乘法形式

如图 8 所示，把 (10) 写成矩阵乘法形式

$$\hat{\mathbf{y}} = \underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{x} \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} b_0 \\ b_1 \end{bmatrix}}_{\mathbf{b}} = \mathbf{X}\mathbf{b} \quad (13)$$

\hat{y} 对应图 6 中红色 \times 的纵轴坐标。

$$\mathbf{X} @ \mathbf{b} = \hat{\mathbf{y}}$$

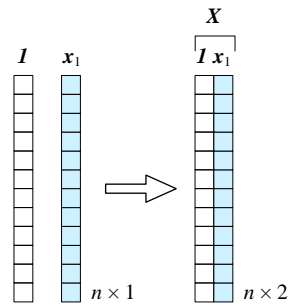
图 8. 矩阵运算得到 \hat{y} ，一元线性回归

我们管 (13) 这个 \mathbf{X} 叫做**设计矩阵** (design matrix)。图 9 所示为构造一元线性回归设计矩阵的方法。

设计矩阵通常在第一列添加全 1 列向量 \mathbf{I} 用于表示截距项，其余列为各自变量的取值，便于用矩阵方式表示回归模型。

一元线性回归，截距不为 0，设计矩阵 \mathbf{X} 左侧需要添加全 1 列向量 \mathbf{I} 。图 1 中这个一元线性回归对应的设计矩阵 \mathbf{X} 为

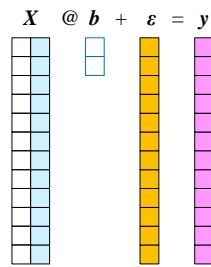
$$\mathbf{X} = [\mathbf{I} \quad \mathbf{x}_1] = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 5 \\ 1 & 6 \\ 1 & 7 \\ 1 & 8 \\ 1 & 9 \end{bmatrix} \quad (14)$$

图 9. 设计矩阵 X ，一元线性回归

如图 10 所示，真实值列向量 y 可以写成估计值向量 \hat{y} 、误差向量 ε 之和

$$y = \hat{y} + \varepsilon = Xb + \varepsilon \quad (15)$$

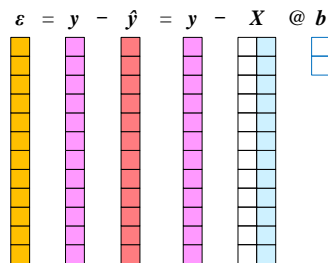
y 对应图 6 中蓝色 \times 的纵轴坐标。 ε 对应图 6 中橙色线段。

图 10. 矩阵运算得到 y ，一元线性回归

正交关系

如图 11 所示，误差向量 ε 为

$$\varepsilon = y - \hat{y} = y - Xb \quad (16)$$

图 11. 矩阵运算得到 ε ，一元线性回归

ε 垂直 X 的每一列，意味着，

$$\begin{cases} \mathbf{I}^T \boldsymbol{\varepsilon} = 0 \\ \mathbf{x}_1^T \boldsymbol{\varepsilon} = 0 \end{cases} \quad (17)$$

把上两式写成一个矩阵乘法

$$\begin{bmatrix} \mathbf{I}^T \\ \mathbf{x}_1^T \end{bmatrix} \boldsymbol{\varepsilon} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (18)$$

即

$$\mathbf{X}^T \boldsymbol{\varepsilon} = \mathbf{0} \quad (19)$$

这一点上一章讲过！

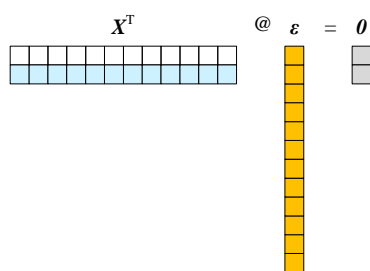


图 12. $\boldsymbol{\varepsilon}$ 垂直 \mathbf{X} 的每一列，一元线性回归

广义逆

把 (16) 带入 (19)

$$\mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{b}) = \mathbf{0} \quad (20)$$

整理得到等式

$$\mathbf{X}^T \mathbf{X} \mathbf{b} = \mathbf{X}^T \mathbf{y} \quad (21)$$

前文假设 \mathbf{I} 、 \mathbf{x}_1 线性无关，因此 $\mathbf{X} = [\mathbf{I}, \mathbf{x}_1]$ 列满秩。 \mathbf{X} 的格拉姆矩阵 $\mathbf{X}^T \mathbf{X}$ 满秩，因此可逆。

从而推导得到 \mathbf{b} 的解析式

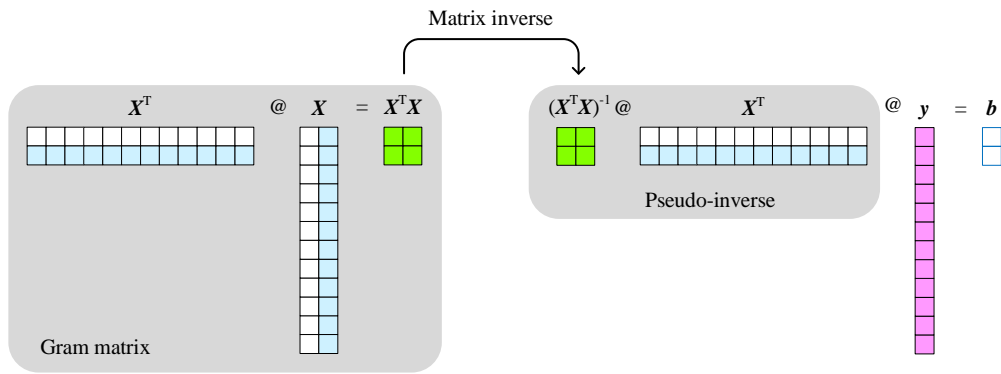
$$\mathbf{b} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (22)$$

图 13 所示为计算系数向量 \mathbf{b} 的流程。

? 那么在 $\text{span}(\mathbf{I}, \mathbf{x}_1)$ 空间中， \mathbf{b} 是什么？

根据 (10)， \mathbf{b} 是 \mathbf{y} 在 $\text{span}(\mathbf{I}, \mathbf{x}_1)$ 投影的坐标！ \mathbf{b} 就是标量投影！

此外，本书前文提过， $(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$ 叫**伪逆** (pseudo-inverse)，也叫**广义逆** (generalized inverse)。

图 13. 计算 \mathbf{b} ，一元线性回归

带入具体的数值，计算图 2 中红色直线的具体参数

$$\begin{aligned} \mathbf{b} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \begin{pmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 5 \\ 1 & 6 \\ 1 & 7 \\ 1 & 8 \\ 1 & 9 \end{bmatrix}^T & \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 5 \\ 1 & 6 \\ 1 & 7 \\ 1 & 8 \\ 1 & 9 \end{bmatrix}^{-1} & \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 5 \\ 1 & 6 \\ 1 & 7 \\ 1 & 8 \\ 1 & 9 \end{bmatrix}^T & \begin{bmatrix} 3 \\ 1 \\ 4 \\ 6 \\ 5 \\ 8 \\ 7 \\ 8 \end{bmatrix} \end{pmatrix} \\ &= \begin{bmatrix} 8 & 41 \\ 41 & 269 \end{bmatrix}^{-1} @ \begin{bmatrix} 42 \\ 261 \end{bmatrix} = \begin{bmatrix} 0.571 & -0.087 \\ -0.087 & 0.016 \end{bmatrix} @ \begin{bmatrix} 42 \\ 261 \end{bmatrix} = \begin{bmatrix} 1.267 \\ 0.777 \end{bmatrix} \end{aligned} \quad (23)$$

也就是说， $b_0 = 1.267$ ， $b_1 = 0.777$ 。

⚠ 注意，本节计算结果仅保留三位小数。

这样，比例函数的解析式为

$$\hat{y} = b_0 + b_1 x_1 = 1.267 + 0.777 x_1 \quad (24)$$

这样我们便的都图 2 中红色直线的解析式。

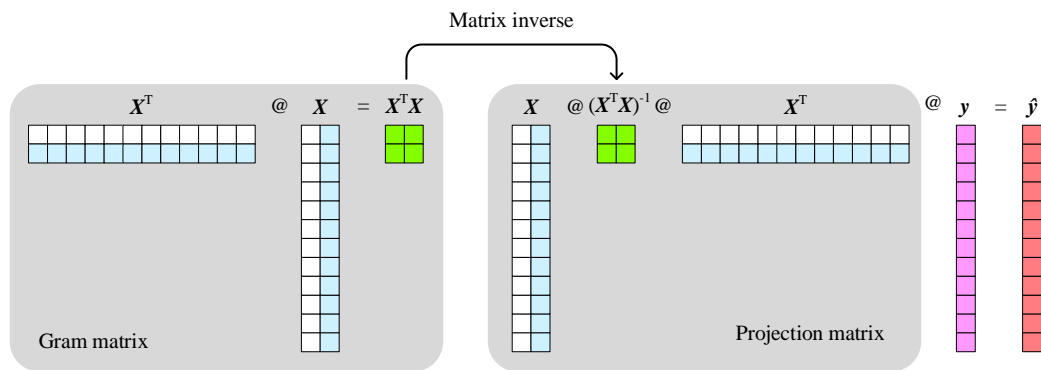
投影矩阵

(22) 代入 (13)，可以得到：

$$\hat{\mathbf{y}} = \mathbf{X} \underbrace{(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T}_{\mathbf{b}} \mathbf{y} \quad (25)$$

图 14 所示为计算系数向量 $\hat{\mathbf{y}}$ 的矩阵乘法。

❓ 请大家试着分析 (25) 中 $\mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$ 的作用是什么？

图 14. 计算 \hat{y} , 一元线性回归

计算预测值向量

$$\hat{y} = Xb = \begin{bmatrix} 1 & x_1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = b_0 + b_1 x_1 = 1.267 + 0.777 x_1 = 1.267 + 0.777 \times \begin{bmatrix} 1 \\ 2 \\ 3 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{bmatrix} = \begin{bmatrix} 2.044 \\ 2.821 \\ 3.598 \\ 5.152 \\ 5.929 \\ 6.707 \\ 7.484 \\ 8.261 \end{bmatrix} \quad (26)$$

\hat{y} 对应图 6 中红色 \times 的纵轴坐标。显然，所有的红色 \times 都在红色回归直线上。

最小二乘拟合

前文提到找到合适的 b_0 、 b_1 让残差列向量为 ε 尽量小。

回顾本书前文向量范数相关内容，向量范数的作用就是计算向量的“大小”；而最常见的范数就是 L^2 范数。残差列向量为 ε 的 L^2 范数为

$$\|\varepsilon\| = \|y - Xb\| \quad (27)$$

带入具体值，残差列向量为 ε 为

$$\varepsilon = y - \hat{y} = \begin{bmatrix} 0.955 \\ -1.821 \\ 0.401 \\ 0.847 \\ -0.929 \\ 1.292 \\ -0.484 \\ -0.261 \end{bmatrix} \quad (28)$$

这个列向量中每个值对应图 6 中的橙色竖直线段。

为了方便运算，我们把 ε 的 L^2 范数平方，得到

$$\|\boldsymbol{\varepsilon}\|_2^2 = \|\mathbf{y} - \mathbf{X}\mathbf{b}\|_2^2 \quad (29)$$

上式很容易写成矩阵乘法

$$\|\boldsymbol{\varepsilon}\|_2^2 = \boldsymbol{\varepsilon}^T \boldsymbol{\varepsilon} = (\mathbf{y} - \mathbf{X}\mathbf{b})^T (\mathbf{y} - \mathbf{X}\mathbf{b}) \quad (30)$$

即

$$\|\boldsymbol{\varepsilon}\|_2^2 = \boldsymbol{\varepsilon}^T \boldsymbol{\varepsilon} = \boldsymbol{\varepsilon} \cdot \boldsymbol{\varepsilon} = \langle \boldsymbol{\varepsilon}, \boldsymbol{\varepsilon} \rangle = \begin{bmatrix} \varepsilon^{(1)} \\ \varepsilon^{(2)} \\ \vdots \\ \varepsilon^{(i)} \\ \vdots \\ \varepsilon^{(n)} \end{bmatrix} \cdot \begin{bmatrix} \varepsilon^{(1)} \\ \varepsilon^{(2)} \\ \vdots \\ \varepsilon^{(i)} \\ \vdots \\ \varepsilon^{(n)} \end{bmatrix} = (\varepsilon^{(1)})^2 + (\varepsilon^{(2)})^2 + \cdots + (\varepsilon^{(i)})^2 + \cdots + (\varepsilon^{(n)})^2 \quad (31)$$

这便是**最小二乘拟合** (ordinary least squares, OLS)。简单来说，最小二乘拟合是一种数学方法，用于在拟合模型时最小化预测值与真实值之间误差的平方和。

带入具体值， $\boldsymbol{\varepsilon}$ 的 L^2 范数平方为

$$\|\boldsymbol{\varepsilon}\|_2^2 = \boldsymbol{\varepsilon}^T \boldsymbol{\varepsilon} = \begin{bmatrix} 0.955 \\ -1.821 \\ 0.401 \\ 0.847 \\ -0.929 \\ 1.292 \\ -0.484 \\ -0.261 \end{bmatrix}^T \begin{bmatrix} 0.955 \\ -1.821 \\ 0.401 \\ 0.847 \\ -0.929 \\ 1.292 \\ -0.484 \\ -0.261 \end{bmatrix} = 7.949 \quad (32)$$

如图 15 所示，图中的正方形的边长便是残差。上式的标量值为图中正方形面积之和。

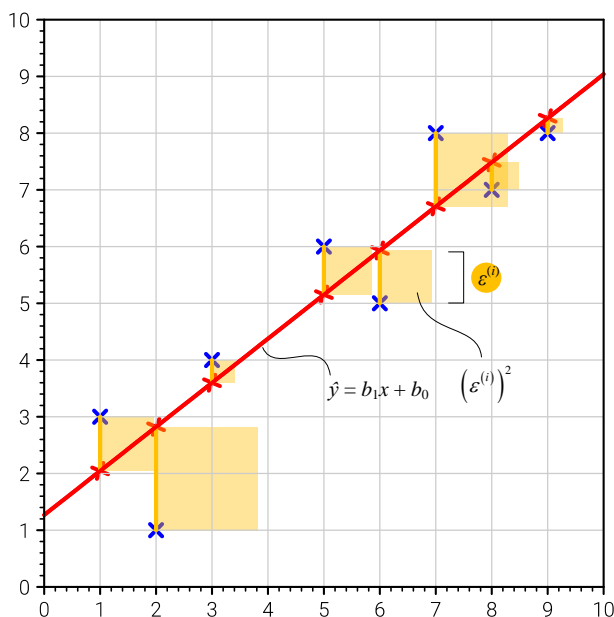




图 15. 残差平方之和对应图中正方形面积之和

 怎么求解 (30) 的最小值？这需要用到优化工具，这是“数学不难”中《高等数学不难》要讨论的话题。

获得一条红色线并不是线性回归分析的终点，实际上只是一个开始。我们还要借助各种概率统计工具来判断这条线的可靠性，比如评估模型是否显著、变量是否显著、残差是否满足正态分布和同方差性等假设。同时，我们还需要估计参数的不确定性，计算置信区间，检验模型的拟合优度，甚至探讨变量之间可能存在的多重共线性。只有在充分理解这些统计特征后，我们才能真正对线性模型的解释能力、预测能力以及其在实际问题中的应用前景做出科学判断。

 线性回归分析是“数学不难”中《概率统计不难》要讨论的话题。

编程实现

代码 1 展示如何用 Python 编程完成一元线性回归求解。

a 创建了两个列向量 x 和 y ， x 是输入值，也就是“自变量”， y 是对应的输出值，也就是“因变量”。`.reshape(-1,1)` 的意思是把原本的一维数组变成二维的“列向量”，因为后面要做矩阵运算，它必须是二维形式。

b 构造了一个“设计矩阵” X 。`np.ones_like(x)` 创建了和 x 一样大小的全是 1 的列向量，作为常数项（也叫截距项）的那一列；`np.column_stack` 是把这列全 1 和 x 拼在一起，拼出的 X 就是我们要用来自回归的输入矩阵，它每一行对应一个样本，每列对应一个特征，第 1 列是常数 1，第 2 列是原始的 x 值。


c 是最关键的一句：计算线性回归的参数向量 b ，也就是回归线的截距和斜率。这用的是最小二乘法的标准公式，整体含义是：用一个矩阵公式，把输入 X 和输出 y 之间的关系“拟合”出来，得到一个最合理的直线。


d 计算在 x_array 上预测出来的 y 值，也就是回归直线在新数据点上的位置，用来画出平滑的拟合曲线。

e 在训练数据上做预测，也就是我们用训练数据 x 输入，得到的预测值 y_pred ，目的是和真实值 y 对比，看拟合效果好不好。

f 计算预测误差，也叫残差。我们把真实的 y 减去预测值 y_pred ，结果 $error$ 就是每个数据点的“误差向量”，反映预测值和实际值之间的偏差。

g 计算误差向量的平方和，叫做 L2 范数的平方，也叫“残差平方和”。它是评估拟合好坏的指标，越小表示回归线越贴近数据点。

 `LA_Ch09_06_01.ipynb` 还有一段可视化代码，代码很简单，请大家自行学习。

代码 1. 一元线性回归 |  `LA_10_01_01.ipynb`

```

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

## 数据
a x = np.array([1,2,3,5,6,7,8,9]).reshape(-1,1) # 自变量 x
y = np.array([3,1,4,6,5,8,7,8]).reshape(-1,1) # 因变量 y

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

## 计算参数
c b = np.linalg.inv(X.T @ X) @ X.T @ y

# 格拉姆矩阵
X.T @ X

# 格拉姆矩阵的逆
np.linalg.inv(X.T @ X)

## 计算预测值
x_array = np.linspace(0,10)
X_array = np.column_stack((np.ones_like(x_array), x_array))

d y_array_pred = X_array @ b
e y_pred = X @ b

## 误差向量
f error = y - y_pred

# 误差向量的L2范数的平方
g error.T @ error

```



请大家自学 LA_10_01_02.ipynb。这段代码使用的是 Scikit-learn 提供的 LinearRegression 工具，自动完成了模型训练与预测步骤，背后的数学原理与最小二乘法完全一致，结果也和 LA_10_01_01.ipynb 完全一致。

比例函数：不含截距项

本节最后让我们看一个特殊的一元线性回归模型——过原点的一元线性模型。如果线性回归模型采用比例函数，即截距 b_0 为 0，这样只需要估算 b_1 ，得到的直线过原点。

由于我们要用比例函数拟合数据，因此截距 b_0 设为 0；这样，设计矩阵 $X = x_1$ ， X 左侧不需要添加全 1 列向量 $\mathbf{1}$ 。这样很容易计算得到参数 b_1

$$b_1 = (\mathbf{x}_1^T \mathbf{x}_1)^{-1} \mathbf{x}_1^T \mathbf{y} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{pmatrix}^T @ \begin{pmatrix} 1 \\ 2 \\ 3 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{pmatrix}^{-1} @ \begin{pmatrix} 1 \\ 2 \\ 3 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{pmatrix}^T @ \begin{pmatrix} 3 \\ 1 \\ 4 \\ 6 \\ 5 \\ 8 \\ 7 \\ 8 \end{pmatrix} = 269^{-1} \times 261 = 0.970 \quad (33)$$

这样，过原点线性回归模型解析式为

$$\hat{y} = b_1 x_1 = 0.970 x_1 \quad (34)$$

上式对应图 16 中红色直线。

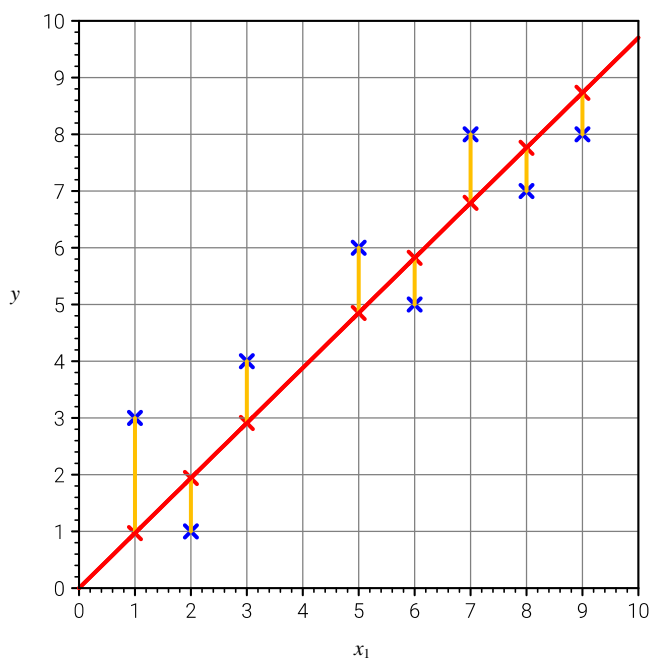


图 16. 比例函数模型

计算预测值向量

$$\hat{\mathbf{y}} = b_1 \mathbf{x}_1 = 0.970 \times \begin{bmatrix} 1 \\ 2 \\ 3 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{bmatrix} = \begin{bmatrix} 0.970 \\ 1.941 \\ 2.911 \\ 4.851 \\ 5.822 \\ 6.792 \\ 7.762 \\ 8.732 \end{bmatrix} \quad (35)$$

计算误差向量

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

$$\boldsymbol{\varepsilon} = \mathbf{y} - \hat{\mathbf{y}} = \begin{bmatrix} 2.029 \\ -0.940 \\ 1.089 \\ 1.148 \\ -0.821 \\ 1.208 \\ -0.762 \\ -0.732 \end{bmatrix} \quad (36)$$

带入具体值计算误差平方之和

$$\|\boldsymbol{\varepsilon}\|_2^2 = \boldsymbol{\varepsilon}^T \boldsymbol{\varepsilon} = \begin{bmatrix} 2.029 \\ -0.940 \\ 1.089 \\ 1.148 \\ -0.821 \\ 1.208 \\ -0.762 \\ -0.732 \end{bmatrix}^T \begin{bmatrix} 2.029 \\ -0.940 \\ 1.089 \\ 1.148 \\ -0.821 \\ 1.208 \\ -0.762 \\ -0.732 \end{bmatrix} = 10.762 \quad (37)$$



请大家自学 LA_10_01_03.ipynb。这段代码求解以上截距为 0（比例函数）的一元线性模型。



请大家用 DeepSeek/ChatGPT 等工具完成本节如下习题。

Q1. 请手解，并编写代码求解如下超定方程组。

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 4 \\ 3 \end{bmatrix}$$

Q2. 以下代码生成自变量、因变量样本数据，请大家补全代码，构造一元回归模型。

```
import numpy as np
np.random.seed(0)
x = np.random.rand(50) * 10
true_slope = 0.8
true_intercept = 2
y = true_slope * x + true_intercept + noise
```