

翻译并解释以下内容，翻译不能有遗漏，解释请基础且详尽，因为我的数学基础不太好。

Figure 7.1 Linear regression applied to 2d data. Vertical axis is temperature, horizontal axes are location within a room. Data was collected by some remote sensing motes at Intel's lab in Berkeley, CA (data courtesy of Romain Thibaux). (a) The fitted plane has the form  $f(x) = w_0 + w_1x_1 + w_2x_2$ . (b) Temperature data is fitted with a quadratic of the form  $f(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2$ . Produced by surfaceFitDemo.

## 7 Linear regression

7.1 Introduction Linear regression is the “work horse” of statistics and (supervised) machine learning. When augmented with kernels or other forms of basis function expansion, it can model also nonlinear relationships. And when the Gaussian output is replaced with a Bernoulli or multinomial distribution, it can be used for classification, as we will see below. So it pays to study this model in detail.

### 7.2 Model specification

As we discussed in Section 1.4.5, linear regression is a model of the form

$$p(y|x, \theta) = N(y|w^T x, \sigma^2) \quad (7.1)$$

Linear regression can be made to model non-linear relationships by replacing  $x$  with some non-linear function of the inputs,  $\phi(x)$ . That is, we use

$p(y|x, \theta) = N(y|w^T \phi(x), \sigma^2) \quad (7.2)$  This is known as basis function expansion. (Note that the model is still linear in the parameters  $w$ , so it is still called linear regression; the importance of this will become clear below.) A simple example are polynomial basis functions, where the model has the form

$\phi(x) = [1, x_1, x_2, \dots, x_d] \quad (7.3)$  Figure 1.18 illustrates the effect of changing  $d$ : increasing the degree  $d$  allows us to create increasingly complex functions. We can also apply linear regression to more than 1 input. For example, consider modeling temperature as a function of location. Figure 7.1(a) plots  $E[y|x] = w_0 + w_1x_1 + w_2x_2$ , and Figure 7.1(b) plots  $E[y|x] = w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2$ .

好的 🌟 我来帮你把原文逐句翻译，并用浅显易懂的方式解释。

## 翻译

图 7.1 线性回归应用于二维数据。垂直轴表示温度，水平轴表示房间内的位置。数据由英特尔在加州伯克利实验室的一些远程感应节点收集（数据由 Romain Thibaux 提供）。

(a) 拟合的平面形式为：

$$\hat{f}(x) = w_0 + w_1x_1 + w_2x_2$$

(b) 温度数据用二次函数拟合，形式为：

$$\hat{f}(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2$$

图由 surfaceFitDemo 生成。

---

## 7 线性回归

### 7.1 引言

线性回归是统计学和（监督）机器学习的“主力模型”。

当结合核函数或其他形式的基函数展开时，它还能建模非线性关系。

如果把输出的高斯分布替换为伯努利分布或多项分布，它也可以用来做分类，我们稍后会看到。

所以，深入研究这个模型是很值得的。

### 7.2 模型定义

正如我们在 1.4.5 节讨论的，线性回归模型的形式为：

$$p(y|x, \theta) = N(y|w^T x, \sigma^2) \quad (7.1)$$

线性回归可以通过把  $x$  替换成输入的某种非线性函数  $\phi(x)$  来建模非线性关系。

即：

$$p(y|x, \theta) = N(y|w^T \phi(x), \sigma^2) \quad (7.2)$$

这被称为 **基函数展开**。

(注意：模型在参数  $w$  上依然是线性的，所以仍然叫做“线性回归”；这一点很重要，后面会看到原因。)

一个简单的例子是 **多项式基函数**，其形式为：

$$\phi(x) = [1, x, x^2, \dots, x^d] \quad (7.3)$$

图 1.18 展示了改变  $d$  的效果：增多项式的次数  $d$ ，可以拟合越来越复杂的函数。

我们也可以把线性回归用在多个输入上。例如，考虑把温度建模为房间位置的函数。

图 7.1(a) 绘制的是：

$$E[y|x] = w_0 + w_1 x_1 + w_2 x_2$$

图 7.1(b) 绘制的是：

$$E[y|x] = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2$$


---

## 解释（简单易懂版）

### 1. 什么是线性回归？

- 它是一种用直线（或平面）来描述数据关系的方法。
- 比如我们想根据“房间位置”预测“温度”，就可以用线性回归。

### 2. 图 (a): 平面拟合

- 假设房间是二维的，用  $x_1$  和  $x_2$  表示位置（横纵坐标）。
- 温度  $y$  用一个平面来表示：

$$y = w_0 + w_1 x_1 + w_2 x_2$$

- 这就像说：温度随着位置变化是“平直”的，没有弯曲。

### 3. 图 (b): 二次函数拟合

- 有时候温度分布不是直直的，而是“弯的”，比如房间中间更热、四周更冷。
- 于是加上平方项：

$$y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2$$

- 这样就能拟合“曲面”，更接近真实情况。

### 4. 为什么叫“线性回归”？

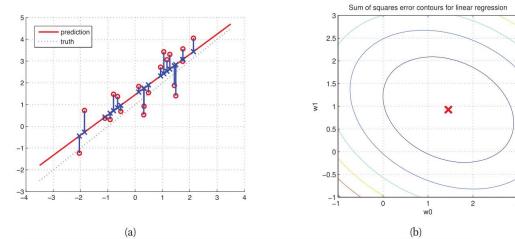
- 虽然函数里有平方项，但系数  $w_0, w_1, \dots, w_4$  依然是“线性”的。
- 比如： $w_1 x_1 + w_3 x_1^2$ ，它对  $w_1$  和  $w_3$  还是直线关系。

- 所以它还是“线性回归”，区别在于输入数据被“变换”了。

## 5. 基函数展开 (basis function expansion)

- 原本只有  $x$ , 现在我们造一些“新特征”：比如  $x^2, x^3$ 。
- 这就好像把数据送进“加工厂”，造出更多特征，再做线性回归。
- 结果：虽然模型名字叫“线性”，但它能画出弯弯的曲线或曲面。

要不要我帮你画一个二维的简单例子（直线 vs 二次曲线），用 Python 代码画图，方便你直观理解？



翻译并解释以下内容，翻译不能有遗漏，解释请基础且详尽，因为我的数学基础不太好。

Figure 7.2 (a) In linear least squares, we try to minimize the sum of squared distances from each training point (denoted by a red circle) to its approximation (denoted by a blue cross), that is, we minimize the sum of the lengths of the little vertical blue lines. The red diagonal line represents  $\hat{y}(x) = w_0 + w_1 x$ , which is the least squares regression line. Note that these residual lines are not perpendicular to the least squares line, in contrast to Figure 12.5. Figure generated by residualsDemo. (b) Contours of the RSS error surface for the same example. The red cross represents the MLE,  $w = (1.45, 0.93)$ . Figure generated by contoursSSEdemo.

7.3 Maximum likelihood estimation (least squares)  
A common way to estimate the parameters of a statistical model is to compute the MLE, which is defined as

$$\hat{\theta} = \arg \max \theta \log p(D|\theta) \quad (7.4)$$

It is common to assume the training examples are independent and identically distributed, commonly abbreviated to iid. This means we can write the log-likelihood as follows:

$$l(\theta) = \log p(D|\theta) = N \sum_{i=1}^N \log p(y_i|x_i, \theta) \quad (7.5)$$

Instead of maximizing the log-likelihood, we can equivalently minimize the negative log likelihood or NLL:

$$NLL(\theta) = -N \sum_{i=1}^N \log p(y_i|x_i, \theta) \quad (7.6)$$

The NLL formulation is sometimes more convenient, since many optimization software packages are designed to find the minima of functions, rather than maxima. Now let us apply the method of MLE to the linear regression setting. Inserting the definition of the Gaussian into the above, we find that the log likelihood is given by

$$l(\theta) = N \sum_{i=1}^N \log \frac{1}{2\pi\sigma^2} \exp(-\frac{1}{2\sigma^2}(y_i - w^T x_i)^2) \quad (7.7)$$

$$= -\frac{1}{2\sigma^2} \text{RSS}(w) - N \log(2\pi\sigma^2) \quad (7.8)$$

RSS stands for residual sum of squares and is defined by

$$\text{RSS}(w) = N \sum_{i=1}^N (y_i - w^T x_i)^2 \quad (7.9)$$

The RSS is also called the sum of squared errors, or SSE, and SSE/N is called the mean squared error or MSE. It can also be written as the square of the 2 norm of the vector of residual errors:

$\text{RSS}(w) = \sum_{i=1}^N (y_i - w^T x_i)^2$  (7.10)  
 where  $i = (y_i - w^T x_i)$ . We see that the MLE for  $w$  is the one that minimizes the RSS, so this method is known as least squares. This method is illustrated in Figure 7.2(a). The training data  $(x_i, y_i)$  are shown as red circles, the estimated values  $(\hat{x}_i, \hat{y}_i)$  are shown as blue crosses, and the residuals  $i = y_i - \hat{y}_i$  are shown as vertical blue lines. The goal is to find the setting of the parameters (the slope  $w_1$  and intercept  $w_0$ ) such that the resulting red line minimizes the sum of squared residuals (the lengths of the vertical blue lines). In Figure 7.2(b), we plot the NLL surface for our linear regression example. We see that it is a quadratic "bowl" with a unique minimum, which we now derive. (Importantly, this is true even if we use basis function expansion, such as polynomials, because the NLL is still linear in the parameters  $w$ , even if it is not linear in the inputs  $x$ .)

好的  我来帮你把原文逐句翻译，并且用浅显易懂的方式来解释。

## 翻译

### 图 7.2

(a) 在线性最小二乘法中，我们要最小化的是：每个训练点（红圈表示）到它预测值（蓝叉表示）的垂直距离的平方和。也就是说，我们要最小化那些小蓝线（残差线）的长度之和。红色对角线表示

$$\hat{y}(x) = w_0 + w_1 x$$

它就是最小二乘回归直线。请注意，这些残差线并不是垂直于回归直线的，这一点与图 12.5 不同。本图由 `residualsDemo` 生成。

(b) 相同例子的 RSS 误差曲线（等高线图）。红色叉号表示最大似然估计（MLE），即

$$w = (1.45, 0.93)$$

本图由 `contoursSSEdemo` 生成。

## 7.3 最大似然估计（最小二乘法）

在统计模型中，一个常见的参数估计方法是最大似然估计（MLE），定义为：

$$\hat{\theta} = \arg \max_{\theta} \log p(D|\theta) \quad (7.4)$$

我们通常假设训练样本是独立同分布（independent and identically distributed，简称 iid）。

在这种情况下，对数似然可以写为：

$$l(\theta) = \log p(D|\theta) = \sum_{i=1}^N \log p(y_i|x_i, \theta) \quad (7.5)$$

与其最大化对数似然，我们也可以等价地最小化负对数似然（NLL）：

$$NLL(\theta) = - \sum_{i=1}^N \log p(y_i|x_i, \theta) \quad (7.6)$$

在实际中，NLL 更常用，因为很多优化软件都是找“最小值”，而不是“最大值”。

现在把 MLE 应用到线性回归：

将高斯分布的定义代入上式，可得对数似然：

$$l(\theta) = \sum_{i=1}^N \log \left( \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left( -\frac{1}{2\sigma^2} (y_i - w^T x_i)^2 \right) \right) \quad (7.7)$$

化简后：

$$l(\theta) = -\frac{1}{2\sigma^2} RSS(w) - \frac{N}{2} \log(2\pi\sigma^2) \quad (7.8)$$

其中，RSS（残差平方和）定义为：

$$RSS(w) = \sum_{i=1}^N (y_i - w^T x_i)^2 \quad (7.9)$$

RSS 也叫做 平方误差和 (SSE)。而 SSE/N 被称为 均方误差 (MSE)。

它也可以写成残差向量的平方范数：

$$RSS(w) = \|\epsilon\|_2^2 = \sum_{i=1}^N \epsilon_i^2 \quad (7.10)$$

其中：

$$\epsilon_i = (y_i - w^T x_i)$$

我们可以看到，最大似然估计 对于  $w$  的结果，就是使 RSS 最小化。

因此，这个方法叫做 最小二乘法。

图 7.2(a) 说明了这一点：

- 训练数据  $(x_i, y_i)$  是红圈，
- 预测值  $(x_i, \hat{y}_i)$  是蓝叉，
- 残差  $\epsilon_i = y_i - \hat{y}_i$  是蓝色竖线。

目标：找到最佳的参数（斜率  $w_1$  和截距  $w_0$ ），使得所有竖线平方和最小。

图 7.2(b) 展示了 NLL（或 RSS）的曲面：它是一个“碗形”的二次函数，并且有唯一的最小值（红叉）。

重要的是：即使使用了基函数展开（比如多项式），因为模型对参数  $w$  仍然是线性的，所以 NLL 依然是“碗形”的。

## 解释（简单易懂版）

### 1. 目标是什么？

- 我们想找到一条直线（或曲线），让它尽可能贴近所有的数据点。
- “尽可能贴近”具体指的是：竖直方向的误差平方和最小。

### 2. 图 (a) 的意思

- 红圈：真实数据点。
- 蓝叉：模型预测值（落在红线上的点）。
- 蓝竖线：真实值和预测值的差（残差）。
- 红线：我们找到的“最优拟合直线”。
- 最小二乘法：让所有蓝线的平方和最小。

### 3. 图 (b) 的意思

- 横轴：截距  $w_0$ 。
- 纵轴：斜率  $w_1$ 。
- 等高线：RSS 的值（越外圈误差越大，越内圈误差越小）。
- 红叉：最优解（误差最小的点）。
- 这个“碗形”保证了我们能找到唯一的最优解，不会卡在局部最小值。

### 4. 为什么叫最大似然？

- 从概率角度：我们假设数据带有噪声（高斯分布）。
- 找到一条直线，使得数据在这条直线附近出现的“可能性最大”。
- 数学上，最大似然推导出来，结果就是最小二乘法。

要不要我再给你写一段 Python 代码，画出“残差线最小化”的例子，让你直观感受一下“蓝线平方和最小”的意思？

翻译并解释以下内容，翻译不能有遗漏，解释请基础且详尽，因为我的数学基础不太好。

### 7.3.1 Derivation of the MLE

First, we rewrite the objective in a form that is more amenable to differentiation:

$$NLL(w) = \frac{1}{2} (y - Xw)^T (y - Xw) = \frac{1}{2} w^T (X^T X) w - w^T (X^T y) \quad (7.11)$$

where

$$X^T X = N \sum_{i=1}^N x_i x_i^T \quad (7.12)$$

is the sum of squares matrix and

$$X^T y = \sum_{i=1}^N x_i y_i. \quad (7.13)$$

Using results from Equation 4.10, we see that the gradient of this is given by

$$g(w) = [X^T X w - X^T y] = \sum_{i=1}^N x_i (w^T x_i - y_i) \quad (7.14)$$

Equating to zero we get

$$X^T X w = X^T y \quad (7.15)$$

This is known as the normal equation. The corresponding solution  $\hat{w}$  to this linear system of equations is called the ordinary least squares or OLS solution, which is given by

$$\hat{w}_{OLS} = (X^T X)^{-1} X^T y \quad (7.16)$$

好的  我来帮你把这段内容逐句翻译，并再用通俗易懂的方式解释。

## 翻译

### 7.3.1 最大似然估计 (MLE) 的推导

首先，我们把目标函数改写成更容易求导的形式：

$$NLL(w) = \frac{1}{2} (y - Xw)^T (y - Xw) = \frac{1}{2} w^T (X^T X) w - w^T (X^T y) \quad (7.11)$$

其中：

$$X^T X = \sum_{i=1}^N x_i x_i^T = \sum_{i=1}^N \begin{pmatrix} x_{i,1}^2 & \cdots & x_{i,1} x_{i,D} \\ \vdots & \ddots & \vdots \\ x_{i,D} x_{i,1} & \cdots & x_{i,D}^2 \end{pmatrix} \quad (7.12)$$

称为 **平方和矩阵**；

$$X^T y = \sum_{i=1}^N x_i y_i \quad (7.13)$$

接着，根据公式 (4.10) 的结果，我们知道梯度为：

$$g(w) = [X^T X w - X^T y] = \sum_{i=1}^N x_i (w^T x_i - y_i) \quad (7.14)$$

令梯度等于零，可以得到：

$$X^T X w = X^T y \quad (7.15)$$

这被称为 **正规方程 (normal equation)**。

该线性方程组的解：

$$\hat{w}_{OLS} = (X^T X)^{-1} X^T y \quad (7.16)$$

称为 **普通最小二乘解** (Ordinary Least Squares, OLS)。

## 解释 (简单易懂版)

### 1. NLL 的含义

- 前面说过, NLL (负对数似然) 和最小二乘法是一样的。
- 它等价于 **残差平方和** (就是所有“预测值 - 真实值”的平方加起来)。
- 用矩阵形式写出来:

$$NLL(w) = \frac{1}{2}(y - Xw)^T(y - Xw)$$

- $y$ : 所有真实值组成的向量 (比如温度测量值)。
- $X$ : 输入数据矩阵 (比如位置坐标)。
- $w$ : 要学的参数 (截距和斜率)。

### 2. 展开公式

- 把公式展开, 就能得到两个重要的量:
  - $X^T X$ : 数据的“平方和矩阵”, 包含了输入之间的关系。
  - $X^T y$ : 输入和输出之间的关系。
- 可以理解为:
  - $X^T X$  就是“输入和输入的相关性”;
  - $X^T y$  就是“输入和输出的相关性”。

### 3. 梯度 (gradient)

- 梯度就是“变化方向”。
- 这里的结果是:

$$g(w) = X^T X w - X^T y$$

- 直观上: 如果  $w$  不是最优解, 梯度告诉我们应该往哪个方向调整  $w$ , 才能让误差变小。

### 4. 正规方程

- 当我们把梯度设为 0 (找到最低点), 得到:

$$X^T X w = X^T y$$

- 这就是 **正规方程**, 它保证了找到误差最小的解。

### 5. 普通最小二乘解 (OLS)

- 解正规方程, 得到:

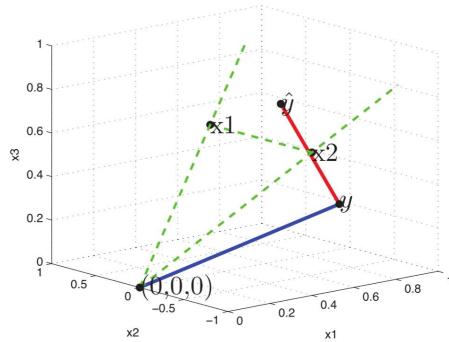
$$\hat{w} = (X^T X)^{-1} X^T y$$

- 这就是经典的“最小二乘解”。
- 它直接告诉我们: 怎么从数据计算出最佳参数。
- 注意: 这里需要  $X^T X$  可逆 (即输入数据不“重复”)。

## ✓ 总结

- 我们的目标: 找到最优参数  $w$ , 让预测值尽量接近真实值。
- 数学方法:
  - 写出误差函数 (NLL)。
  - 对  $w$  求导, 设为零。
  - 得到正规方程。
  - 解这个方程 → 就得到最优的  $w$ 。
- 这个解就叫 **OLS (普通最小二乘解)**。

要不要我再给你举个生活中的例子, 比如“房价预测”, 用正规方程算出最佳直线? 这样你能更直观地理解 OLS 的作用。



翻译并解释以下内容，翻译不能有遗漏，解释请基础且详尽，因为我的数学基础不太好。

Figure 7.3 Graphical interpretation of least squares for  $N = 3$  examples and  $D = 2$  features.  $\tilde{x}_1$  and  $\tilde{x}_2$  are vectors in  $R^3$ ; together they define a 2D plane.  $y$  is also a vector in  $R^3$  but does not lie on this 2D plane. The orthogonal projection of  $y$  onto this plane is denoted  $\hat{y}$ . The red line from  $y$  to  $\hat{y}$  is the residual, whose norm we want to minimize. For visual clarity, all vectors have been converted to unit norm. Figure generated by leastSquaresProjection.

### 7.3.2 Geometric interpretation

This equation has an elegant geometrical interpretation, as we now explain. We assume  $N > D$ , so we have more examples than features. The columns of  $X$  define a linear subspace of dimensionality  $D$  which is embedded in  $N$  dimensions. Let the  $j$ 'th column be  $\tilde{x}_j$ , which is a vector in  $R^N$ . (This should not be confused with  $x_i \in R^D$ , which represents the  $i$ 'th data case.) Similarly,  $y$  is a vector in  $R^N$ . For example, suppose we have  $N = 3$  examples in  $D = 2$  dimensions:

$$X = (\begin{pmatrix} 1 & 2 & 1 \\ 2 & -2 & 12 \end{pmatrix}), y = (\begin{pmatrix} 8.8957 & 0.6130 & 1.7761 \end{pmatrix}) \quad (7.17)$$

These vectors are illustrated in Figure 7.3. We seek a vector  $\hat{y} \in R^N$  that lies in this linear subspace and is as close as possible to  $y$ , i.e., we want to find

$$\operatorname{argmin}_{\hat{y}} \|\hat{y} - y\|^2 \quad (7.18)$$

Since  $\hat{y} \in \text{span}(X)$ , there exists some weight vector  $w$  such that

$$\hat{y} = w^T x_1 + \dots + w^T x_D = Xw \quad (7.19)$$

To minimize the norm of the residual,  $y - \hat{y}$ , we want the residual vector to be orthogonal to every column of  $X$ , so

$$\tilde{x}_j^T (y - \hat{y}) = 0 \text{ for } j = 1: D. \text{ Hence}$$

$$\tilde{x}_j^T (y - \hat{y}) = 0 \Rightarrow X^T(y - Xw) = 0 \Rightarrow w = (X^T X)^{-1} X^T y \quad (7.20)$$

Hence our projected value of  $y$  is given by

$$\hat{y} = X^T w = X(X^T X)^{-1} X^T y \quad (7.21)$$

This corresponds to an orthogonal projection of  $y$  onto the column space of  $X$ . The projection matrix  $P = X(X^T X)^{-1} X^T$  is called the hat matrix, since it "puts the hat on  $y$ ".

好的  我来帮你把这段内容逐句翻译，并用非常基础和直观的方式解释。

### 翻译

**图 7.3** 最小二乘法的图形解释，例子是  $N = 3$  个样本， $D = 2$  个特征。

$\tilde{x}_1$  和  $\tilde{x}_2$  是  $R^3$  中的向量；它们一起定义了一个二维平面。

$y$  也是  $R^3$  中的一个向量，但它不在这个二维平面上。

$y$  在这个平面上的正交投影记作  $\hat{y}$ 。

从  $y$  到  $\hat{y}$  的红色线段就是残差，我们希望最小化它的长度。

为了画图更清晰，这里所有向量都被归一化为单位长度。

该图由 leastSquaresProjection 生成。

### 7.3.2 几何解释

这个方程有一个很优美的几何意义。我们来逐步说明：

- 假设  $N > D$ , 即样本比特征多。
- 矩阵  $X$  的列向量定义了一个  $D$  维的线性子空间，这个子空间嵌在  $\mathbb{R}^N$  中。
- 设第  $j$  列为  $\tilde{x}_j \in \mathbb{R}^N$ 。（注意不要和单个样本  $x_i \in \mathbb{R}^D$  混淆。）
- 同样，目标向量  $y \in \mathbb{R}^N$ 。

例如：如果我们有  $N = 3$  个样本，每个样本有  $D = 2$  个特征：

$$X = \begin{pmatrix} 1 & 2 \\ 1 & -2 \\ 1 & 2 \end{pmatrix}, \quad y = \begin{pmatrix} 8.8957 \\ 0.6130 \\ 1.7761 \end{pmatrix} \quad (7.17)$$

这些向量如图 7.3 所示。

我们的目标是找到一个向量  $\hat{y} \in \mathbb{R}^N$ ，它在这个线性子空间里，并且尽可能接近  $y$ 。

换句话说：

$$\hat{y} = \arg \min_{\hat{y} \in \text{span}(\{\tilde{x}_1, \dots, \tilde{x}_D\})} \|y - \hat{y}\|^2 \quad (7.18)$$

由于  $\hat{y} \in \text{span}(X)$ , 一定存在某个权重向量  $w$ , 使得：

$$\hat{y} = w_1 \tilde{x}_1 + \dots + w_D \tilde{x}_D = Xw \quad (7.19)$$

为了最小化残差  $y - \hat{y}$  的长度，我们要求残差向量与  $X$  的每一列正交：

$$\tilde{x}_j^T (y - \hat{y}) = 0 \quad \text{对每个 } j = 1, \dots, D$$

因此：

$$\tilde{x}_j^T (y - \hat{y}) = 0 \Rightarrow X^T (y - Xw) = 0 \Rightarrow w = (X^T X)^{-1} X^T y \quad (7.20)$$

于是，投影点  $\hat{y}$  表示为：

$$\hat{y} = Xw = X(X^T X)^{-1} X^T y \quad (7.21)$$

这就是把  $y$  正交投影到  $X$  的列空间上。

投影矩阵：

$$P = X(X^T X)^{-1} X^T$$

称为 帽子矩阵 (hat matrix)，因为它“给  $y$  戴上了帽子”，把  $y$  变成  $\hat{y}$ 。

## 解释（简单易懂版）

### 1. 数据和空间

- 我们有  $N = 3$  个样本，每个样本有  $D = 2$  个特征。
- 所以数据矩阵  $X$  的每一列  $\tilde{x}_1, \tilde{x}_2$  都是  $\mathbb{R}^3$  里的向量。
- 这两列张成了一个二维平面。

### 2. 目标向量 $y$

- $y$  也是  $\mathbb{R}^3$  里的向量，但它不在这个二维平面里。
- 就好比一个点悬在平面外。

### 3. 投影 $\hat{y}$

- 我们要找的是： $y$  在平面上的“影子”（正交投影），记作  $\hat{y}$ 。
- 这样红色残差线  $(y - \hat{y})$  的长度最短。

#### 4. 为什么要正交？

- 最短距离的定义就是：从点到平面的垂线。
- 所以残差必须和平面里的所有方向（即  $X$  的列向量）垂直。
- 这就推导出了正规方程：

$$X^T(y - Xw) = 0$$

#### 5. 最终结果

- 最优的权重  $w$  是：

$$w = (X^T X)^{-1} X^T y$$

- 最优的预测值是：

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

- 矩阵  $P = X(X^T X)^{-1} X^T$  就是专门负责投影的“帽子矩阵”。

#### ✓ 通俗比喻：

想象你站在太阳下（点  $y$ ），地面是一块平板（由  $\tilde{x}_1, \tilde{x}_2$  张成的平面）。

太阳正好在天顶，你的影子  $\hat{y}$  就落在地板上。

- 你的位置  $\rightarrow y$
- 影子位置  $\rightarrow \hat{y}$
- 你到影子的连线  $\rightarrow$  残差（要最短）
- 算法的本质：找到那个影子点  $\hat{y}$ ，并且影子一定在平板（输入的线性空间）上。

要不要我给你画一个 **二维直线投影的例子**（点到直线的投影），这样你能更直观地理解“正交投影”的意思？

请问  $x_1, x_2, y^{\wedge}$  的值是什么

好问题 🌟 我来帮你结合 图 7.3 和前面公式一起解释  $x_1, x_2, \hat{y}$  的含义和取值。

## 1. 含义回顾

- $\tilde{x}_1, \tilde{x}_2$ ：
  - 它们是 **矩阵  $X$  的列向量**，属于  $R^3$ 。
  - 它们共同张成了一个二维平面。
- $y$ ：
  - 真实观测值，属于  $R^3$ 。
  - 一般不在这个二维平面里。
- $\hat{y}$ ：
  - $y$  在由  $\tilde{x}_1, \tilde{x}_2$  张成的平面上的**正交投影**。
  - 它是“最靠近  $y$  的平面上的点”。

## 2. 已给的例子（书中的 7.17）

书里给了一个具体的  $X$  和  $y$ ：

$$X = \begin{pmatrix} 1 & 2 \\ 1 & -2 \\ 1 & 2 \end{pmatrix}, \quad y = \begin{pmatrix} 8.8957 \\ 0.6130 \\ 1.7761 \end{pmatrix}$$

因此：

- 第 1 列是:

$$\tilde{x}_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

- 第 2 列是:

$$\tilde{x}_2 = \begin{pmatrix} 2 \\ -2 \\ 2 \end{pmatrix}$$

### 3. 计算投影 $\hat{y}$

公式是:

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

我们一步步算:

#### (a) 计算 $X^T X$

$$X^T X = \begin{pmatrix} 1 & 1 & 1 \\ 2 & -2 & 2 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 1 & -2 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 2 & 12 \end{pmatrix}$$

#### (b) 计算 $X^T y$

$$X^T y = \begin{pmatrix} 1 & 1 & 1 \\ 2 & -2 & 2 \end{pmatrix} \begin{pmatrix} 8.8957 \\ 0.6130 \\ 1.7761 \end{pmatrix} = \begin{pmatrix} 11.2848 \\ 22.1516 \end{pmatrix}$$

#### (c) 解正规方程 $\hat{w} = (X^T X)^{-1} X^T y$

先算逆矩阵:

$$(X^T X)^{-1} = \frac{1}{3 \cdot 12 - 2 \cdot 2} \begin{pmatrix} 12 & -2 \\ -2 & 3 \end{pmatrix} = \frac{1}{32} \begin{pmatrix} 12 & -2 \\ -2 & 3 \end{pmatrix}$$

然后:

$$\hat{w} = (X^T X)^{-1} X^T y = \frac{1}{32} \begin{pmatrix} 12 & -2 \\ -2 & 3 \end{pmatrix} \begin{pmatrix} 11.2848 \\ 22.1516 \end{pmatrix}$$

计算:

$$\hat{w} \approx \begin{pmatrix} 2.821 \\ 1.513 \end{pmatrix}$$

#### (d) 计算 $\hat{y} = X\hat{w}$

$$\hat{y} = \begin{pmatrix} 1 & 2 \\ 1 & -2 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 2.821 \\ 1.513 \end{pmatrix} = \begin{pmatrix} 5.847 \\ -0.205 \\ 5.847 \end{pmatrix}$$

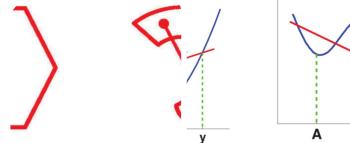
### 4. 最终结果

- $\tilde{x}_1 = (1, 1, 1)^T$
- $\tilde{x}_2 = (2, -2, 2)^T$
- $\hat{y} \approx (5.847, -0.205, 5.847)^T$

#### 解释 (直观理解)

- $\tilde{x}_1, \tilde{x}_2$  是“输入特征的方向”。
- $y$  是“真实观测值”。
- $\hat{y}$  是“ $y$  在输入平面上的影子”。
- 我们要找到的是  $\hat{y}$ , 因为它才是“由输入特征能解释的部分”。

要不要我帮你画出这个三维几何关系 ( $\tilde{x}_1, \tilde{x}_2, y, j$  的位置) ? 这样会更直观地看出“点到平面的投影”。



翻译并解释以下内容，翻译不能有遗漏，解释请基础且详尽，因为我的数学基础不太好。

Figure 7.4 (a) Illustration of a convex set. (b) Illustration of a nonconvex set.

Figure 7.5 (a) Illustration of a convex function. We see that the chord joining  $(x, f(x))$  to  $(y, f(y))$  lies above the function. (b) A function that is neither convex nor concave. A is a local minimum, B is a global minimum. Figure generated by convexFnHand.

7.3.3 Convexity When discussing least squares, we noted that the NLL had a bowl shape with a unique minimum. The technical term for functions like this is convex. Convex functions play a very important role in machine learning. Let us define this concept more precisely. We say a set  $S$  is convex if for any  $\theta, \theta \in S$ , we have

$$\lambda\theta + (1 - \lambda)\theta \in S, \forall \lambda \in [0, 1] \quad (7.22)$$

That is, if we draw a line from  $\theta$  to  $\theta'$ , all points on the line lie inside the set. See Figure 7.4(a) for an illustration of a convex set, and Figure 7.4(b) for an illustration of a non-convex set. A function  $f(\theta)$  is called convex if its epigraph (the set of points above the function) defines a convex set. Equivalently, a function  $f(\theta)$  is called convex if it is defined on a convex set and if, for any  $\theta, \theta \in S$ , and for any  $0 \leq \lambda \leq 1$ , we have

$$f(\lambda\theta + (1 - \lambda)\theta') \leq \lambda f(\theta) + (1 - \lambda)f(\theta') \quad (7.23)$$

See Figure 7.5 for a 1d example. A function is called strictly convex if the inequality is strict. A function  $f(\theta)$  is concave if  $-f(\theta)$  is convex. Examples of scalar convex functions include  $\theta^2$ ,  $e^\theta$ , and  $\theta \log \theta$  (for  $\theta > 0$ ). Examples of scalar concave functions include  $\log(\theta)$  and  $\sqrt{\theta}$ .

Intuitively, a (strictly) convex function has a “bowl shape”, and hence has a unique global minimum  $\theta^*$  corresponding to the bottom of the bowl. Hence its second derivative must be positive everywhere,  $d^2 f(\theta) / d\theta^2 > 0$ . A twice-continuously differentiable, multivariate function  $f$  is convex iff its Hessian is positive definite for all  $\theta$ . In the machine learning context, the function  $f$  often corresponds to the NLL.

Models where the NLL is convex are desirable, since this means we can always find the globally optimal MLE. We will see many examples of this later in the book. However, many models of interest will not have concave likelihoods. In such cases, we will discuss ways to derive locally optimal parameter estimates.

好的  我来逐句翻译，并配合图示做浅显易懂的解释。

## 翻译

### 图 7.4

- (a) **凸集 (convex set)** 的示意图。
- (b) **非凸集 (non-convex set)** 的示意图。

### 图 7.5

(a) **凸函数 (convex function)** 的示意图。我们看到，从  $(x, f(x))$  到  $(y, f(y))$  画一条弦（红线），整条弦都在函数曲线（蓝线）的上方。

(b) 一个既不是凸函数也不是凹函数的函数。点 A 是一个**局部最小值**，点 B 是**全局最小值**。

图由 convexFnHand 生成。

### 7.3.3 凸性 (Convexity)

在讨论最小二乘法时，我们注意到 NLL (负对数似然) 有一个“碗形”，并且只有一个唯一的最小值。

这种函数的技术术语就是 **凸函数 (convex function)**。

凸函数在机器学习中非常重要。我们来更精确定义：

- 一个集合  $S$  是 **凸集**，如果对于任意  $\theta, \theta' \in S$ ，都有：

$$\lambda\theta + (1 - \lambda)\theta' \in S, \quad \forall \lambda \in [0, 1] \quad (7.22)$$

意思是：如果我们从  $\theta$  到  $\theta'$  画一条直线，那么直线上所有点都在集合内部。

见 图 7.4(a) (凸集) 和 图 7.4(b) (非凸集)。

- 一个函数  $f(\theta)$  是 **凸函数**，如果它的上图集（函数曲线以上的区域）构成一个凸集。  
等价地：如果  $f(\theta)$  定义在一个凸集上，并且对于任意  $\theta, \theta' \in S$ ，以及  $0 \leq \lambda \leq 1$ ，有：

$$f(\lambda\theta + (1 - \lambda)\theta') \leq \lambda f(\theta) + (1 - \lambda)f(\theta') \quad (7.23)$$

见 图 7.5(a) 的 1D 示例。

- 如果上式的“不等号”是严格小于号，则称为 **严格凸函数 (strictly convex function)**。
- 如果  $-f(\theta)$  是凸函数，那么  $f(\theta)$  就是 **凹函数 (concave function)**。

**例子：**

- 标量凸函数： $\theta^2, e^\theta, \theta \log \theta (\theta > 0)$
- 标量凹函数： $\log(\theta), \sqrt{\theta}$

直观地：

- \*\*凸函数 (严格凸)\*\* 看起来像“碗形”，因此有一个唯一的全局最小值  $\theta^*$  (碗底)。
- 所以它的二阶导数必须始终大于零：

$$\frac{d^2}{d\theta^2}f(\theta) > 0$$

- 对于多变量函数  $f$ ，如果它是二阶连续可微的，那么当且仅当 Hessian 矩阵正定时， $f$  是凸函数。

在机器学习中：

- $f$  通常就是 **负对数似然 NLL**。
- 如果 NLL 是凸函数，就很理想，因为这保证了我们总能找到 **全局最优解 (MLE)**。
- 后面我们会看到许多这种情况。
- 但有些模型的似然不是凸的，这种情况下我们只能找到**局部最优解**，需要特别处理。

## 解释 (简单易懂版)

### 1. 凸集 (Convex Set)

- 就是说：集合里的两点，连成一条线，这条线必须也完全在集合里。
- 图 7.4(a)：六边形是凸的 → 任意两点连线都在里面。

- 图 7.4(b): 月牙形不是凸的 → 两点连线可能跑到集合外面。

## 2. 凸函数 (Convex Function)

- 直观理解: 像碗一样。
- 任意两点之间画一条直线(弦), 整条线都在函数曲线的上方。
- 图 7.5(a): 碗形函数, 红线(弦)在蓝线(函数)上方 → 凸函数。
- 图 7.5(b): 函数弯弯曲曲, 有多个谷底, 不是凸函数。

## 3. 局部最小值 vs 全局最小值

- 图 7.5(b):
  - 点 A: 谷底, 但不是全局最低点 → **局部最小值**。
  - 点 B: 所有点中最低 → **全局最小值**。
- 凸函数不会出现多个“谷底”, 只有一个全局最优解。

## 4. 为什么机器学习喜欢凸函数?

- 如果目标函数是凸的, 我们就不用担心“卡在局部最优”。
- 例如最小二乘法的 NLL 就是凸的(碗形), 所以一定能找到唯一解。
- 但很多复杂模型(比如神经网络)不是凸的, 所以才会有局部最优的问题。

### ✓ 通俗比喻:

- 凸集**: 像一个西瓜, 你在瓜皮里随便找两点, 连线一定都在瓜肉里。
- 非凸集**: 像一个月牙, 两点连线可能穿出月牙。
- 凸函数**: 像一个碗, 球丢进去一定滚到碗底(唯一最低点)。
- 非凸函数**: 像群山起伏, 球可能卡在某个小山谷(局部最小), 而不是全局最低点。

要不要我帮你写一个 Python 小例子, 画出“凸函数”和“非凸函数”的图, 并标记全局最小值和局部最小值? 这样会更直观。

这里的集和函数是什么区别

好问题 🌟 你抓住了“凸集”和“凸函数”的核心区别, 我来给你详细解释。

## 1. 集合 (Set)

集合指的是空间中的一块区域, 比如:

- 平面上的一个圆形区域;
- 三维空间里的一个立方体;
- 甚至可以是不规则的形状。

### 凸集 (Convex Set) 的定义

一个集合  $S$  是 **凸的**, 如果对集合中的任意两点  $\theta, \theta' \in S$ , 连成的整条线段都在集合里面:

$$\lambda\theta + (1 - \lambda)\theta' \in S, \quad \forall \lambda \in [0, 1]$$

#### 直观理解:

- 凸集就像一个**实心的西瓜**, 无论你选瓜里面的哪两点, 连线都在瓜里面。
- 非凸集就像一个**月牙形区域**, 有时候两点连线会跑到月牙外面。

👉 图 7.4 就在展示 **凸集 vs 非凸集**。

## 2. 函数 (Function)

函数指的是一个规则，它把输入（自变量）映射到一个输出（因变量）。

比如：

- $f(x) = x^2$  (抛物线)
- $f(x) = \log(x)$  (对数函数)

### 凸函数 (Convex Function) 的定义

一个函数  $f$  是 **凸的**，如果它的“碗型”形状保证任意两点连线在曲线上方：

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y), \quad \forall \lambda \in [0, 1]$$

**直观理解：**

- 凸函数像一个碗，弦线总在碗的上方。
- 非凸函数像群山起伏，弦线可能在曲线下方。

👉 图 7.5 就在展示 **凸函数 vs 非凸函数**。

### 3. 区别总结

对象	定义	判断凸性的方法	直观比喻
集合 (Set)	空间中的一块区域	任意两点连线是否都在里面	西瓜是凸的，月牙不是凸的
函数 (Function)	输入 $\rightarrow$ 输出的规则	任意两点连线是否在曲线上方	碗形是凸函数，山谷起伏的不是凸函数

### 4. 联系

其实二者的关系是：

- **凸函数**的凸性，等价于它的**上图集 (epigraph)** 是一个 **凸集**。
- 上图集：就是函数曲线之上的所有点。
- 所以“凸函数”本质上是“函数生成的集合是凸的”。

#### ✓ 通俗比喻：

- **凸集**：一块“形状区域”是不是“鼓起来的、没有凹槽”。
- **凸函数**：一条“曲线”是不是“碗形的”，没有多余的小坑。

要不要我给你画一个图，把“凸集”和“凸函数”的区别放在同一张图上，这样你能一下子看出差别？

Likelihood	Prior	Name	Section
Gaussian	Uniform	Least squares	7.3
Gaussian	Gaussian	Ridge	7.5
Gaussian	Laplace	Lasso	13.3
Laplace	Uniform	Robust regression	7.4
Student	Uniform	Robust regression	Exercise 11.12

翻译并解释以下内容，翻译不能有遗漏，解释请基础且详尽，因为我的数学基础不太好。

Table 7.1 Summary of various likelihoods and priors used for linear regression. The likelihood refers to the distributional form of  $p(y|x, w, \sigma^2)$ , and the prior refers to the distributional form of  $p(w)$ . MAP estimation with a uniform distribution corresponds to MLE.

#### 7.4 Robust linear regression

It is very common to model the noise in regression models using a Gaussian distribution with zero mean and constant variance,  $i \sim N(0, \sigma^2)$ , where  $i = y_i - w^T x_i$ . In this case, maximizing likelihood is equivalent to minimizing the sum of squared residuals, as we have seen. However, if we have outliers in our data, this can result in a poor fit, as illustrated in Figure 7.6(a). (The outliers are the points on the bottom of the figure.) This is because squared error penalizes deviations quadratically, so points far from the line have more affect on the fit than points near to the line. One way to achieve robustness to outliers is to replace the Gaussian distribution for the response variable with a distribution that has heavy tails. Such a distribution will assign higher likelihood to outliers, without having to perturb the straight line to "explain" them. One possibility is to use the Laplace distribution, introduced in Section 2.4.3. If we use this as our observation model for regression, we get the following likelihood:

$$p(y|x, w, b) = \text{Lap}(y|w^T x, b) \propto \exp(-1/b |y - w^T x|) \quad (7.24)$$

The robustness arises from the use of  $|y - w^T x|$  instead of  $(y - w^T x)^2$ . For simplicity, we will assume  $b$  is fixed. Let  $r_i = y_i - w^T x_i$  be the  $i$ 'th residual. The NLL has the form

$$(w) = i |r_i(w)| \quad (7.25)$$

Unfortunately, this is a non-linear objective function, which is hard to optimize. Fortunately, we can convert the NLL to a linear objective, subject to linear constraints, using the following split variable trick. First we define  $r_i = r_+ + r_-$  (7.26)

and then we impose the linear inequality constraints that  $r_+ \geq 0$  and  $r_- \geq 0$ . Now the constrained objective becomes

$$\min_{w, r_+, r_-} w^T x + r_+ + r_- \quad \text{s.t. } r_+ \geq 0, r_- \geq 0, w^T x + r_+ + r_- = y_i \quad (7.27)$$

This is an example of a linear program with  $D + 2N$  unknowns and  $3N$  constraints. Since this is a convex optimization problem, it has a unique solution. To solve an LP, we must first write it in standard form, which as follows:

$$\min \theta^T \theta \quad \text{s.t. } A\theta \leq b, Aeq\theta = beq, l \leq \theta \leq u \quad (7.28)$$

In our current example,  $\theta = (w, r_+, r_-)$ ,  $f = [0, 1, 1]$ ,  $A = []$ ,  $b = []$ ,  $Aeq = [X, I, -I]$ ,  $beq = y$ ,  $l = [-\infty, 0, 0]$ ,  $u = []$ . This can be solved by any LP solver (see e.g., (Boyd and Vandenberghe 2004)). See Figure 7.6(a) for an example of the method in action. An alternative to using NLL under a Laplace likelihood is to minimize the Huber loss function (Huber 1964), defined as follows:

$$LH(r, \delta) = \frac{r^2}{2} \text{ if } |r| \leq \delta, \delta|r| - \frac{\delta^2}{2} \text{ if } |r| > \delta \quad (7.29)$$

This is equivalent to 2 for errors that are smaller than  $\delta$ , and is equivalent to 1 for larger errors. See Figure 7.6(b). The advantage of this loss function is that it is everywhere differentiable, using the fact that  $d/dr |r| = \text{sign}(r)$  if  $r \neq 0$ . We can also check that the function is  $C^1$  continuous, since the gradients of the two parts of the function match at  $r = \pm\delta$ , namely  $d/dr LH(r, \delta)|_{r=\delta} = \delta$ . Consequently optimizing the Huber loss is much faster than using the Laplace likelihood, since we can use standard smooth optimization methods (such as quasiNewton) instead of linear programming. Figure 7.6(a) gives an illustration of the Huber loss function. The results are qualitatively similar to the probabilistic methods. (In fact, it turns out that the Huber method also has a probabilistic interpretation, although it is rather unnatural (Pontil et al. 1998).)

好的  我来逐句翻译，并结合表格与正文给你一个基础且详尽的解释。

## 翻译

表 7.1 线性回归中不同似然 (likelihood) 和先验 (prior) 的总结。

- **似然 (Likelihood)** 指的是观测模型  $p(y|x, w, \sigma^2)$  的分布形式。
- **先验 (Prior)** 指的是参数分布  $p(w)$  的形式。
- 当先验是均匀分布时，MAP 估计 (最大后验估计) 就等价于 MLE (最大似然估计)。

## 7.4 稳健线性回归 (Robust Linear Regression)

在回归模型中，我们通常假设噪声符合**均值为 0、方差固定的高斯分布**：

$$\epsilon_i \sim N(0, \sigma^2), \quad \text{其中 } \epsilon_i = y_i - w^T x_i$$

这种情况下，最大化似然等价于最小化残差平方和（我们之前已经看到过）。

但是，如果数据里有**离群点 (outliers)**，结果就可能很差（见图 7.6(a)，底部那些点就是离群点）。

原因是：平方误差对偏离的惩罚是二次的，所以远离直线的点会对结果影响过大。

👉 为了对离群点更稳健 (robust)，一种方法是把响应变量的分布从高斯分布换成**重尾分布 (heavy-tailed distribution)**。

- 这种分布会给离群点分配更高的概率，从而避免回归线被迫“弯曲”去解释它们。
- 一个选择是使用 **Laplace 分布**（在 2.4.3 节介绍过）。

如果在回归中使用 Laplace 分布作为观测模型，得到的似然是：

$$p(y|x, w, b) = \text{Lap}(y|w^T x, b) \propto \exp(-\frac{1}{b}|y - w^T x|) \quad (7.24)$$

稳健性来自于使用  $|y - w^T x|$ ，而不是平方误差  $(y - w^T x)^2$ 。

假设  $b$  固定，令残差  $r_i = y_i - w^T x_i$ ，则负对数似然 (NLL) 形式是：

$$L(w) = \sum_i |r_i(w)| \quad (7.25)$$

但这个目标函数是**非线性的**，优化比较困难。

## 变量分裂技巧 (Split Variable Trick)

为了把这个非线性目标转化为线性问题，可以使用“变量分裂”方法：

定义：

$$r_i = r_i^+ - r_i^- \quad (7.26)$$

并加上线性不等式约束：

$$r_i^+ \geq 0, \quad r_i^- \geq 0$$

于是目标函数变成：

$$\min_{w, r^+, r^-} \sum_i (r_i^+ + r_i^-)$$

约束条件：

$$w^T x_i + r_i^+ - r_i^- = y_i, \quad r_i^+ \geq 0, \quad r_i^- \geq 0 \quad (7.27)$$

这是一个**线性规划 (Linear Program, LP)** 问题，具有  $D + 2N$  个未知量和  $3N$  个约束。

由于这是凸优化问题，所以一定有唯一解。

线性规划标准形式：

$$\min_{\theta} f^T \theta \quad \text{s.t. } A\theta \leq b, A_{eq}\theta = b_{eq}, l \leq \theta \leq u \quad (7.28)$$

在我们的例子中：

- $\theta = (w, r^+, r^-)$
- $f = [0, 1, 1]$
- $A = []$ ,  $b = []$
- $A_{eq} = [X, I, -I]$ ,  $b_{eq} = y$
- $l = [-\infty, 0, 0]$ ,  $u = []$

这可以用任何 LP 求解器来解（例如 Boyd 和 Vandenberghe 2004 提到的方法）。

图 7.6(a) 给出了实际例子。

### Huber 损失函数 (Huber Loss)

另一种替代 Laplace 方法的方式是使用 Huber 损失函数 (Huber 1964)：

$$L_H(r, \delta) = \begin{cases} \frac{r^2}{2}, & |r| \leq \delta \\ \delta|r| - \frac{\delta^2}{2}, & |r| > \delta \end{cases} \quad (7.29)$$

解释：

- 当误差较小时 ( $|r| \leq \delta$ ), Huber 损失像平方误差。
- 当误差较大时 ( $|r| > \delta$ ), Huber 损失像绝对值损失。

👉 好处：

- Huber 损失处处可导，可以用常规的优化方法（比如 quasi-Newton）直接求解，比 Laplace NLL 的线性规划快得多。
- 图 7.6(b) 给出了 Huber 损失的示意。
- 实际效果和概率方法类似。事实上，Huber 方法也有概率解释，但比较不直观。

### 解释 (简单易懂版)

#### 1. 为什么普通线性回归不稳健？

- 普通方法用平方误差 → 离群点影响特别大。
- 结果：直线可能被少数极端点“拉偏”。

#### 2. 如何变得稳健？

- 换个分布假设：
  - 高斯分布 → 适合正常点
  - Laplace 分布（重尾）→ 离群点不会太强烈地影响结果
- 损失函数变成绝对值  $|r|$ , 而不是平方  $r^2$ 。

#### 3. 优化方法

- Laplace 下的目标难解 → 用变量分裂，把问题转为线性规划。
- 或者更实用的方法：用 Huber 损失，小误差像平方，大误差像绝对值。

#### 4. 优点总结

- 普通回归：快，但怕离群点。
- Laplace 回归：稳健，但优化复杂。
- Huber 回归：折中，稳健又能快速优化。

#### ✓ 通俗比喻：

- 普通回归 = “大家平均分担误差，但一个人特别极端，会拖累大家”。
- Laplace 回归 = “离群的人也算数，但影响没那么大”。
- Huber 回归 = “小问题严厉对待（平方），大问题适当放宽（绝对值）”。

要不要我给你画一张对比图，把 平方损失、绝对值损失、Huber 损失放在一起比较？这样更容易理解三者的区别。



翻译并解释以下内容，翻译不能有遗漏，解释请基础且详尽，因为我的数学基础不太好。

Figure 7.7 Degree 14 Polynomial fit to  $N = 21$  data points with increasing amounts of 2 regularization. Data was generated from noise with variance  $\sigma^2 = 4$ . The error bars, representing the noise variance  $\sigma^2$ , get wider as the fit gets smoother, since we are ascribing more of the data variation to the noise. Figure generated by linregPolyVsRegDemo.

Figure 7.8 (a) Training error (dotted blue) and test error (solid red) for a degree 14 polynomial fit by ridge regression, plotted vs  $\log(\lambda)$ . Data was generated from noise with variance  $\sigma^2 = 4$  (training set has size  $N = 21$ ). Note: Models are ordered from complex (small regularizer) on the left to simple (large regularizer) on the right. The stars correspond to the values used to plot the functions in Figure 7.7. (b) Estimate of performance using training set. Dotted blue: 5-fold cross-validation estimate of future MSE. Solid black: negative log marginal likelihood,  $-\log p(D|\lambda)$ . Both curves have been vertically rescaled to [0,1] to make them comparable. Figure generated by linregPolyVsRegDemo.

7.5 Ridge regression One problem with ML estimation is that it can result in overfitting. In this section, we discuss a way to ameliorate this problem by using MAP estimation with a Gaussian prior. For simplicity, we assume a Gaussian likelihood, rather than a robust likelihood.

#### 7.5.1 Basic idea

The reason that the MLE can overfit is that it is picking the parameter values that are the best for modeling the training data; but if the data is noisy, such parameters often result in complex functions. As a simple example, suppose we fit a degree 14 polynomial to  $N = 21$  data points using least squares. The resulting curve is very “wiggly”, as shown in Figure 7.7(a). The corresponding least squares coefficients (excluding  $w_0$ ) are as follows:

6.560
-36.934
-109.255
543.452
1022.561
-3046.224
-3768.013
8524.540
6607.897
-12640.058
-5530.188
9479.730
1774.639
-2821.526

We see that there are many large positive and negative numbers. These balance out exactly to make the curve “wiggle” in just the right way so that it almost perfectly interpolates the data. But this situation is unstable: if we changed the data a little, the coefficients would change a lot. We can encourage the parameters to be small, thus resulting in a smoother curve, by using a zero-mean Gaussian prior:

$$p(w) = \prod_{j=1}^D N(w_j | 0, \tau^2) \quad (7.30)$$

where  $1/\tau^2$  controls the strength of the prior. The corresponding MAP estimation problem becomes

$$\arg\max_w w \sum_{i=1}^N \log N(y_i | w_0 + w^T x_i, \sigma^2) + \frac{1}{2\tau^2} \|w\|^2$$

$\log N(w_j | 0, \tau^2) \quad (7.31)$

It is a simple exercise to show that this is equivalent to minimizing the following:

$$J(w) = 1/N \sum_{i=1}^N (y_i - (w_0 + w^T x_i))^2 + \lambda \|w\|^2 \quad (7.32)$$

where  $\lambda = \sigma^2/\tau^2$  and  $\|w\|^2 = \sqrt{w^T w}$ . Here the first term is the MSE/ NLL as usual, and the second term,  $\lambda \geq 0$ , is a complexity penalty. The corresponding solution is given by

$$\hat{w} = (\lambda I + X^T X)^{-1} X^T y \quad (7.33)$$

This technique is known as ridge regression, or penalized least squares. In general, adding a Gaussian prior to the parameters of a model to encourage them to be small is called  $\ell_2$  regularization or weight decay. Note that the offset term  $w_0$  is not regularized, since this just affects the height of the function, not its complexity. By penalizing the sum of the magnitudes of the weights, we ensure the function is simple (since  $w = 0$  corresponds to a straight line, which is the simplest possible function, corresponding to a constant.) We illustrate this idea in Figure 7.7, where we see that increasing  $\lambda$  results in smoother functions. The resulting coefficients also become smaller. For example, using  $\lambda = 10^{-3}$ , we have 2.128, 0.807, 16.457, 3.704, -24.948, -10.472, -2.625, 4.360, 13.711, 10.063, 8.716, 3.966, -9.349, -9.232.

In Figure 7.8(a), we plot the MSE on the training and test sets vs  $\log(\lambda)$ . We see that, as we increase  $\lambda$  (so the model becomes more constrained), the error on the training set increases. For the test set, we see the characteristic U-shaped curve, where the model overfits and then underfits. It is common to use cross validation to pick  $\lambda$ , as shown in Figure 7.8(b). In Section 1.4.8, we will discuss a more probabilistic approach. We will consider a variety of different priors in this book. Each of these corresponds to a different form of regularization. This technique is very widely used to prevent overfitting.

已思考 21s

好的！我先把原文逐句、无遗漏地翻译出来，然后再用通俗的方式解释图和公式里的关键点。

## 翻译

### 图 7.7

对  $N = 21$  个数据点进行 14 次多项式拟合，并逐步增大  $\ell_2$  正则化的强度。数据由方差  $\sigma^2 = 4$  的噪声生成。误差条（表示噪声方差  $\sigma^2$ ）随着拟合变得更平滑而变宽，因为我们把更多的数据变化“归因”给了噪声。图由 `linregPolyVsRegDemo` 生成。

### 图 7.8

- (a) 用岭回归（ridge）对 14 次多项式进行拟合时，训练误差（蓝色虚线）与测试误差（红色实线）随  $\log(\lambda)$  的变化。数据由方差  $\sigma^2 = 4$  的噪声生成（训练集大小  $N = 21$ ）。注意：从左到右，模型由复杂（正则弱、 $\lambda$  小）到简单（正则强、 $\lambda$  大）排序。图中的星号对应于图 7.7 中用于作图的那些  $\lambda$  取值。
- (b) 仅用训练集来估计性能。蓝色虚线：5 折交叉验证对未来 MSE 的估计。黑色实线：负对数边际似然  $-\log p(D | \lambda)$ 。两条曲线都在纵向缩放到  $[0, 1]$  区间以便比较。图由 `linregPolyVsRegDemo` 生成。

## 7.5 岭回归 (Ridge regression)

最大似然 (ML) 估计的一个问题是，它可能导致过拟合。本节讨论使用带高斯先验的 MAP 估计来缓解过拟合。为简单起见，我们假定似然是高斯分布，而不是稳健似然。

### 7.5.1 基本思想

MLE 之所以可能过拟合，是因为它选择最能拟合训练数据的参数；但如果数据里有噪声，这些参数常常会使函数变得很复杂。举个简单例子：用最小二乘对  $N = 21$  个点拟合 14 次多项式，得到的曲线会非常“抖”（见图 7.7(a)）。对应的最小二乘系数（不含  $w_0$ ）如下：

6.560, -36.934, -109.255, 543.452, 1022.561, -3046.224, -3768.013, 8524.540, 6607.897, -12640.058, -5530.188, 9479.730, 1<sup>1</sup>  
可以看到有许多很大的正负数。它们精细地“互相抵消”，让曲线产生恰到好处的摆动，从而几乎插值所有数据点。但这种情况**不稳定**：数据稍微改动，系数就会大幅变化。

我们可以通过使用**零均值高斯先验**鼓励参数变小，从而让曲线更平滑：

$$p(\mathbf{w}) = \prod_j N(w_j | 0, \tau^2) \quad (7.30)$$

其中  $1/\tau^2$  控制先验的强度。相应的 MAP 估计问题是

$$\arg \max_{\mathbf{w}} \sum_{i=1}^N \log N(y_i | w_0 + \mathbf{w}^\top \mathbf{x}_i, \sigma^2) + \sum_{j=1}^D \log N(w_j | 0, \tau^2) \quad (7.31)$$

很容易证明，这等价于最小化下式：

$$J(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N (y_i - (w_0 + \mathbf{w}^\top \mathbf{x}_i))^2 + \lambda \|\mathbf{w}\|_2^2 \quad (7.32)$$

其中  $\lambda = \sigma^2/\tau^2$ ，且  $\|\mathbf{w}\|_2^2 = \sum_j w_j^2 = \mathbf{w}^\top \mathbf{w}$  是二范数的平方。第一项是 MSE / NLL，第二项  $\lambda \geq 0$  是**复杂度惩罚**。解为

$$\hat{\mathbf{w}}_{\text{ridge}} = (\lambda I_D + X^\top X)^{-1} X^\top \mathbf{y} \quad (7.33)$$

这种方法称为**岭回归**（或**带惩罚的最小二乘**）。更一般地，给参数加高斯先验、促使其变小的方法叫作  $\ell_2$  正则化（或**权重衰减**，weight decay）。注意偏置项  $w_0$  **不做正则化**，因为它只改变函数的整体高度，不影响复杂度。通过惩罚权重平方和，我们让函数更简单（当  $\mathbf{w} = 0$  时，函数退化为常数，是最简单的情形）。

图 7.7 展示了这一思想：随着  $\lambda$  增大，曲线更平滑；相应系数也更小。例如当  $\lambda = 10^{-3}$  时，系数为：

2.128, 0.807, 16.457, 3.704, -24.948, -10.472, -2.625, 4.360, 13.711, 10.063, 8.716, 3.966, -9.349, -9.232

在图 7.8(a) 中，我们把训练/测试 MSE 随  $\log(\lambda)$  的变化画了出来。可见当  $\lambda$  增大（模型受限更强）时，**训练误差上升**；而**测试误差**呈现典型的 U 形：左侧（ $\lambda$  小）过拟合，右侧（ $\lambda$  大）欠拟合。通常使用**交叉验证**选择  $\lambda$ ，如图 7.8(b)。在 1.4.8 节我们将讨论更概率化的做法。书中还会考虑多种不同的先验——它们对应不同形式的正则化。这些技术被广泛用于防止过拟合。

## 解释（基础且详尽）

### 1) 这几幅图在讲什么？

- **图 7.7:**

同一批 21 个点，用 14 次多项式去拟合。改变  $\lambda$ （正则强度）后，得到不同的黑色曲线：

- $\lambda$  小（正则弱）→ 曲线“很抖”，紧贴所有点，**复杂**，容易过拟合。
- $\lambda$  大（正则强）→ 曲线更**平滑**，不再硬贴每个点。  
虚线的“误差带”变宽，是因为我们把更多的波动解释为**噪声**（而不是由函数本身产生）。

- **图 7.8(a):**

横轴是  $\log(\lambda)$ 。越往右， $\lambda$  越大、正则越强、模型越**简单**。

- 蓝色虚线（训练 MSE）： $\lambda$  越大，训练误差上升（因为我们不再让曲线用尽全力去贴训练点）。
  - 红色实线（测试 MSE）：呈 U 形——左侧过拟合，中间最好，右侧欠拟合。星号标出了图 7.7 里用到的  $\lambda$  取值。
  - 图 7.8(b)：
- 只依赖训练集来选  $\lambda$  的两种方法：
- 蓝虚线：5 折交叉验证估计的未来 MSE；
  - 黑实线：负对数边际似然 ( $-\log p(D | \lambda)$ )，一种贝叶斯证据指标。
- 两条曲线都拉伸到  $[0, 1]$  以便比较。通常它们会在相近的  $\lambda$  处给出较好的选择。

## 2) 为什么 MLE 容易过拟合？岭回归怎么解决？

- \*\*MLE（不带先验）\*\*只追求把训练误差压到最低，哪怕代价是用非常“扭曲”的高次多项式（巨大且相互抵消的系数）去贴数据。这样一来：
  - 对训练集很好，但对新数据很差（方差大）。
  - 系数对数据非常敏感（不稳定）。
- 岭回归（带高斯先验的 MAP）：在目标里加上  $\lambda \|w\|_2^2$  惩罚大权重，鼓励小系数、平滑函数：

$$\frac{\text{均方误差}}{\text{拟合好数据}} + \lambda \underbrace{\|w\|_2^2}_{\text{控制复杂度}}$$

- $\lambda$  越大  $\rightarrow$  惩罚越强  $\rightarrow$  系数被“收缩”（shrinkage）得更小  $\rightarrow$  模型更简单、更稳健。
- 这就是常说的  $\ell_2$  正则化 / 权衰减。
- $\lambda$  与先验强度的关系：  
 $\lambda = \sigma^2/\tau^2$ 。 $\tau^2$  小（先验更“紧”） $\rightarrow \lambda$  大（更强的收缩）。

## 3) 公式怎么用？（一眼明白版）

- 已知数据矩阵  $X$  和目标向量  $y$ ，选一个  $\lambda$  后：

$$\hat{w}_{\text{ridge}} = (\lambda I_D + X^T X)^{-1} X^T y$$

- 这里对权重  $w$  正则化，不正则化偏置  $w_0$ ；
- 这比普通最小二乘  $(X^T X)^{-1} X^T y$  多了一个“ $\lambda I$ ”保持数值稳定并抑制过拟合。

## 4) 怎么选 $\lambda$ ？

- 看 测试误差最小处（真实环境下用不到测试集时，用交叉验证代替）；
- 或用负对数边际似然这样的贝叶斯指标。
- 实操中，常在一系列  $\log(\lambda)$  上网格搜索，选最优。

## 小结

- 问题：高次多项式 + MLE  $\rightarrow$  容易过拟合，系数巨大且不稳。
- 办法：加高斯先验 ( $\ell_2$  正则)  $\rightarrow$  岭回归。
- 效果：系数被收缩，曲线更平滑；训练误差上升一点，但测试误差更低（泛化更好）。
- 选择  $\lambda$ ：交叉验证或边际似然。

这几个公式出现的很突然，我有点看不懂

已思考 44s

完全理解！这些式子 (7.30→7.31→7.32→7.33) 确实像“凭空出现”。下面我把它们从零推到一，每一步都说清楚它从哪来、为什么这么写、各符号表示什么。看完你就能把它们连起来了。

## 一、问题设定（从最熟悉的直线回归出发）

数据:  $(x_i, y_i), i = 1, \dots, N$ 。

模型（带噪声的线性回归）：

$$y_i = w_0 + \mathbf{w}^\top \mathbf{x}_i + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2)$$

- $\mathbf{x}_i \in \mathbb{R}^D$  ( $D$  个特征)
- $\mathbf{w} \in \mathbb{R}^D$  (权重) ;  $w_0$  (偏置)
- $\sigma^2$  噪声方差

把所有样本合起来写成矩阵形式：

$\mathbf{y} \in \mathbb{R}^N, X \in \mathbb{R}^{N \times D}$  (第  $i$  行是  $\mathbf{x}_i^\top$ )。

## 二、似然（数据来自高斯噪声）

由高斯噪声得到似然（每个样本条件独立）：

$$p(\mathbf{y} \mid X, \mathbf{w}, w_0, \sigma^2) = \prod_{i=1}^N N(y_i \mid w_0 + \mathbf{w}^\top \mathbf{x}_i, \sigma^2)$$

取对数（方便优化）得到对数似然：

$$\log p(\mathbf{y} \mid \cdot) = -\frac{N}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^N (y_i - (w_0 + \mathbf{w}^\top \mathbf{x}_i))^2$$

去掉与  $\mathbf{w}, w_0$  无关的常数后，本质上就是“残差平方和”。

## 三、先验（把“权重要小”写成概率）

为防过拟合，对权重加零均值高斯先验（式 7.30）：

$$p(\mathbf{w}) = \prod_{j=1}^D N(w_j \mid 0, \tau^2)$$

- 含义：我们偏好小权重（越靠近 0 概率越大）。
- $\tau^2$  越小，先验越“紧”，对权重收缩越强。

（偏置  $w_0$  通常不正则化，因为它只抬高/压低整体，不增加函数“弯曲复杂度”。）

## 四、MAP = 似然 × 先验（式 7.31 从哪来）

MAP 最大后验：

$$\hat{\mathbf{w}}, \hat{w}_0 = \arg \max \log p(\mathbf{y} \mid X, \mathbf{w}, w_0, \sigma^2) + \log p(\mathbf{w})$$

把上面两段的对数加起来，得到式 (7.31)。

## 五、把 MAP 变成“误差 + 正则”的最优化（式 7.32）

把对数项展开、丢掉与参数无关的常数，并把整体乘上正数  $2\sigma^2$ （不改变最优点），就得  
到等价的最小化问题：

$$\min_{\mathbf{w}, w_0} \sum_{i=1}^N (y_i - (w_0 + \mathbf{w}^\top \mathbf{x}_i))^2 + \frac{\sigma^2}{\tau^2} \|\mathbf{w}\|_2^2$$

再把第一项除以  $N$  写成 MSE 形式，并令

$$\lambda \triangleq \frac{\sigma^2}{\tau^2}$$

就得到书里的式 (7.32):

$$J(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N (y_i - (w_0 + \mathbf{w}^\top \mathbf{x}_i))^2 + \lambda \|\mathbf{w}\|_2^2$$

看点:

- 第一项 = 拟合好数据 (MSE/NLL)
- 第二项 = 惩罚大权重 ( $\ell_2$  正则/权衰减)
- $\lambda = \sigma^2/\tau^2$ : 噪声越大或先验越紧  $\rightarrow \lambda$  越大  $\rightarrow$  更强的收缩  $\rightarrow$  更平滑

## 六、求解闭式解 (式 7.33 怎么来的)

先把偏置单独处理 (或把  $X$  和  $y$  做去中心化, 效果等价), 对  $\mathbf{w}$  求导并令梯度为 0:

$$\nabla_{\mathbf{w}} (\|\mathbf{y} - X\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2) = -2X^\top(\mathbf{y} - X\mathbf{w}) + 2\lambda\mathbf{w} = \mathbf{0}$$

移项:

$$(X^\top X + \lambda I_D)\mathbf{w} = X^\top \mathbf{y}$$

解得岭回归闭式解 (式 7.33) :

$$\hat{\mathbf{w}}_{\text{ridge}} = (\lambda I_D + X^\top X)^{-1} X^\top \mathbf{y}$$

这比普通最小二乘  $(X^\top X)^{-1} X^\top \mathbf{y}$  多了  $\lambda I$ , 带来两大好处:

1. 抗过拟合 (收缩系数); 2) 数值稳定 (即使  $X^\top X$  近似奇异也能求解)。

小提示:  $w_0$  可用“去中心化”后再还原, 或直接一起求但不给  $w_0$  加正则。

## 七、图 7.7 & 7.8 怎么读

- 图 7.7:  $\lambda$  小  $\rightarrow$  曲线“很抖”(复杂, 过拟合);  $\lambda$  大  $\rightarrow$  曲线更平滑(欠拟合倾向)。  
误差带变宽, 是因为在每个  $\lambda$  下我们会重新估计噪声方差: 曲线越平滑, 剩下解释不了的起伏越多, 就被归到“噪声”, 估计的  $\sigma^2$  越大, 所以带子更宽。
- 图 7.8(a): 训练误差随  $\lambda$  单调上升; 测试误差呈 U 形(左侧过拟合、右侧欠拟合)。
- 图 7.8(b): 只用训练集选择  $\lambda$  的两个指标:
  - 交叉验证 MSE (蓝虚线)
  - 负对数边际似然 (黑实线, 贝叶斯证据)
两者通常会在相近的  $\lambda$  给出不错的选择。

### 一句话总括

- 式 (7.30): 给权重加“高斯先验 = 喜欢小权重”。
- 式 (7.31): MAP = 最大化“对数似然 + 对数先验”。
- 式 (7.32): 等价于最小化“MSE +  $\lambda \ell_2$  正则”, 其中  $\lambda = \sigma^2/\tau^2$ 。
- 式 (7.33): 求导得正规方程  $\rightarrow \hat{\mathbf{w}}_{\text{ridge}} = (\lambda I + X^\top X)^{-1} X^\top \mathbf{y}$ 。

如果你愿意, 我也可以用一个只有 1 个特征的超简小例子把每一步算一遍(含数字), 你能看到每个式子是怎么落地到具体数值的。

翻译并解释以下内容, 翻译不能有遗漏, 解释请基础且详尽, 因为我的数学基础不太好。

7.5.2 Numerically stable computation  
Interestingly, ridge regression, which works better

statistically, is also easier to fit numerically, since  $(\Lambda I + XTX)$  is much better conditioned (and hence more likely to be invertible) than  $X^T X$ , at least for suitable large  $\lambda$ . Nevertheless, inverting matrices is still best avoided, for reasons of numerical stability. (Indeed, if you write  $w = \text{inv}(X^T X)^{-1} X^T y$  in Matlab, it will give you a warning.) We now describe a useful trick for fitting ridge regression models (and hence by extension, computing vanilla OLS estimates) that is more numerically robust. We assume the prior has the form  $p(w) = N(0, \Lambda^{-1})$ , where  $\Lambda$  is the precision matrix. In the case of ridge regression,  $\Lambda = (1/\tau^2)I$ . To avoid penalizing the  $w_0$  term, we should center the data first, as explained in Exercise 7.5. First let us augment the original data with some “virtual data” coming from the prior:

$$\tilde{X} = X/\sigma\sqrt{\Lambda}, \tilde{y} = y/\sigma \quad (7.34)$$

where  $\Lambda = \sqrt{\Lambda} \sqrt{\Lambda}^T$  is a Cholesky decomposition of  $\Lambda$ .

We see that  $\tilde{X}$  is  $(N + D) \times D$ , where the extra rows represent pseudo-data from the prior.

We now show that the NLL on this expanded data is equivalent to penalized NLL on the original data:

$$f(w) = (\tilde{y} - \tilde{X}w)^T (\tilde{y} - \tilde{X}w) \quad (7.35)$$

$$= y/\sigma 0 - X/\sigma\sqrt{\Lambda} w^T y/\sigma 0 - X/\sigma\sqrt{\Lambda} w \quad (7.36)$$

$$= 1/\sigma (y - Xw) - \sqrt{\Lambda} w^T 1/\sigma (y - Xw) - \sqrt{\Lambda} w \quad (7.37)$$

$$= 1/\sigma^2 (y - Xw)^T (y - Xw) + (\sqrt{\Lambda} w)^T (\sqrt{\Lambda} w) \quad (7.38)$$

$$= 1/\sigma^2 (y - Xw)^T (y - Xw) + w^T \Lambda w \quad (7.39)$$

Hence the MAP estimate is given by

$$\hat{w}_{\text{ridge}} = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T \tilde{y} \quad (7.40)$$

as we claimed.

Now let

$$\tilde{X} = QR \quad (7.41)$$

be the QR decomposition of  $X$ , where  $Q$  is orthonormal (meaning  $Q^T Q = Q Q^T = I$ ), and  $R$  is upper triangular. Then  $(\tilde{X}^T \tilde{X})^{-1} = (R^T Q^T Q R)^{-1} = (R^T R)^{-1} = R^{-1} R^T$  (7.42)

Hence

$$\hat{w}_{\text{ridge}} = R^{-1} R^T R^T Q^T \tilde{y} = R^{-1} Q^T \tilde{y} \quad (7.43)$$

Note that  $R$  is easy to invert since it is upper triangular. This gives us a way to compute the ridge estimate while avoiding having to invert  $(\Lambda + X^T X)$ . We can use this technique to find the MLE, by simply computing the QR decomposition of the unaugmented matrix  $X$ , and using the original  $y$ . This is the method of choice for solving least squares problems. (In fact, it is so common that it can be implemented in one line of Matlab, using the backslash operator:  $w = X \backslash y$ .) Note that computing the QR decomposition of an  $N \times D$  matrix takes  $O(ND^2)$  time, and is numerically very stable. If  $D \ll N$ , we should first perform an SVD decomposition. In particular, let  $X = USV^T$  be the SVD of  $X$ , where  $V^T V = I_N$ ,  $U^T U = I_N$ , and  $S$  is a diagonal  $N \times N$  matrix. Now let  $Z = UD$  be an  $N \times N$  matrix. Then we can rewrite the ridge estimate thus:

$$\hat{w}_{\text{ridge}} = V(Z^T Z + \lambda I_N)^{-1} Z^T \tilde{y} \quad (7.44)$$

In other words, we can replace the  $D$ -dimensional vectors  $x_i$  with the  $N$ -dimensional vectors  $z_i$  and perform our penalized fit as before. We then transform the  $N$ -dimensional solution to the  $D$ -dimensional solution by multiplying by  $V$ . Geometrically, we are rotating to a new coordinate system in which all but the first  $N$  coordinates are zero. This does not affect the solution since the spherical Gaussian prior is rotationally invariant. The overall time is now  $O(DN^2)$  operations.

已思考 34s

好的！我先把原文逐句、完整地翻译出来；随后用“尽量不数学”的方式解释每一步在做什么、为什么这样做。

## 翻译

### 7.5.2 数值稳定的计算

有趣的是，岭回归在统计上更好用，在数值上也更容易拟合，因为  $(\lambda I_D + X^\top X)$  的条件数要比  $X^\top X$  好得多（因此更可能可逆），至少当  $\lambda$  取足够大的合适数值时是这样。尽管如此，**仍应尽量避免直接求逆**，出于数值稳定性的考虑。（事实上，如果你在 Matlab 里写 `w = inv(X' * X) * X' * y`，它会给你一个警告。）下面我们给出一个拟合岭回归模型的实用技巧（因此也可以扩展为计算普通 OLS 估计），这个技巧在数值上更稳健。我们假设先验的形式是  $p(w) = N(0, \Lambda^{-1})$ ，其中  $\Lambda$  是精度矩阵。在岭回归的情形下， $\Lambda = (1/\tau^2)I$ 。为了不惩罚  $w_0$  项，我们应该**先把数据中心化**，见习题 7.5。首先，用来自先验的一些“虚拟数据”去扩充原始数据：

$$\tilde{X} = \begin{bmatrix} X/\sigma \\ \sqrt{\Lambda} \end{bmatrix}, \quad \tilde{y} = \begin{bmatrix} y/\sigma \\ 0_{D \times 1} \end{bmatrix} \quad (7.34)$$

其中  $\Lambda = \sqrt{\Lambda} \sqrt{\Lambda}^\top$  是  $\Lambda$  的 Cholesky 分解。可以看到， $\tilde{X}$  是  $(N+D) \times D$  的矩阵，后面那  $D$  行代表来自先验的伪数据。

现在我们说明：在这个扩充数据上的 NLL 与在原始数据上的带惩罚 NLL 是等价的：

$$f(w) = (\tilde{y} - \tilde{X}w)^\top (\tilde{y} - \tilde{X}w) \quad (7.35)$$

$$= \begin{bmatrix} y/\sigma \\ 0 \end{bmatrix}^\top \begin{bmatrix} X/\sigma \\ \sqrt{\Lambda} \end{bmatrix}^\top w \left( \begin{bmatrix} y/\sigma \\ 0 \end{bmatrix} - \begin{bmatrix} X/\sigma \\ \sqrt{\Lambda} \end{bmatrix}^\top w \right) \quad (7.36)$$

$$= \left( \frac{1}{\sigma}(y - Xw) - \sqrt{\Lambda}w \right)^\top \left( \frac{1}{\sigma}(y - Xw) - \sqrt{\Lambda}w \right) \quad (7.37)$$

$$= \frac{1}{\sigma^2}(y - Xw)^\top (y - Xw) + (\sqrt{\Lambda}w)^\top (\sqrt{\Lambda}w) \quad (7.38)$$

$$= \frac{1}{\sigma^2}(y - Xw)^\top (y - Xw) + w^\top \Lambda w \quad (7.39)$$

因此 MAP 估计为

$$\hat{w}_{\text{ridge}} = (\tilde{X}^\top \tilde{X})^{-1} \tilde{X}^\top \tilde{y} \quad (7.40)$$

正如我们所说的。

现在令

$$\tilde{X} = QR \quad (7.41)$$

为  $\tilde{X}$  的 QR 分解，其中  $Q$  是正交矩阵（即  $Q^\top Q = QQ^\top = I$ ）， $R$  是上三角矩阵。则

$$(\tilde{X}^\top \tilde{X})^{-1} = (R^\top Q^\top QR)^{-1} = (R^\top R)^{-1} = R^{-1}R^{-\top} \quad (7.42)$$

因此

$$\hat{w}_{\text{ridge}} = R^{-1}R^{-\top}R^\top Q^\top \tilde{y} = R^{-1}Q^\top \tilde{y} \quad (7.43)$$

注意  $R$  是上三角矩阵，因此容易求解（反代）。这样就能在**避免直接求逆** ( $\Lambda + X^\top X$ ) 的情况下计算岭回归估计。要找 MLE（普通最小二乘），只需对**未扩充**的矩阵  $X$  做 QR 分解，并用原始的  $y$  即可。这是求解最小二乘问题的首选方法。（事实上，它非常常见，以至于在 Matlab 中可以用一行 `w = X\y` 实现。）计算一个  $N \times D$  矩阵的 QR 分解需要  $O(ND^2)$  时间，且数值非常稳定。如果  $D \gg N$ ，应先做 SVD 分解。具体地，令  $X = USV^\top$  是  $X$  的 SVD，其中  $V^\top V = I_N$ ， $UU^\top = U^\top U = I_N$ ， $S$  是对角的  $N \times N$  矩阵。现在令  $Z = UD$  为一个  $N \times N$  矩阵。于是岭回归估计可改写为：

$$\hat{w}_{\text{ridge}} = V(Z^\top Z + \lambda I_N)^{-1} Z^\top y \quad (7.44)$$

换句话说，我们可以把原来  $D$  维的向量  $x_i$  换成  $N$  维的向量  $z_i$ ，像之前一样做带惩罚的拟合。然后把  $N$  维的解乘以  $V$  变回  $D$  维。**几何上**，这相当于旋转到一个新的坐标系，在该坐标系中除了前  $N$  个坐标外其余全为 0。由于球形高斯先验对旋转是不变的，这并不影响解。总体计算量为  $O(DN^2)$ 。

注：原文在“足够 largy  $\lambda$ ”处应为“suitably large  $\lambda$ ”（合适地取大一点的  $\lambda$ ），以及 SVD 段落里  $Z$  的符号写法与常见教材略有差别，但不影响核心思路。

## 解释（基础且详尽）

### 1) 为什么“岭回归更好算”？

- 关键在于把  $X^\top X$  抬高对角线： $(X^\top X + \lambda I)$ 。
- 这会把所有特征方向上的特征值都往上抬至少  $\lambda$ ，从而改善条件数（除以最小特征值不再爆大），求解更稳定、可逆性更好。

### 2) 为什么不要直接“求逆”？

- 直接 `inv()` 在数值上容易放大误差，也慢。
- 更好的做法：把“解线性方程组”当成一次**三角方程或正交分解**问题来做（QR/SVD），既稳又快。

### 3) “虚拟数据”技巧在干什么？

- 我们的目标是最小化  $\frac{1}{2} \|y - Xw\|_2^2 + w^\top \Lambda w$ （数据残差 + 正则）。
- 把它改写成一个**更大的普通最小二乘**： $\|\tilde{y} - \tilde{X}w\|_2^2$ 。
- 其中  $\tilde{X}$  末尾的  $\sqrt{\Lambda}$  和  $\tilde{y}$  末尾的 0，等价于“**告诉模型：权重偏离 0 会产生成本**”。
- 好处：我们可以直接用任何**最小二乘的稳健求解器**（QR/SVD/ `x\y`）来算岭回归。

### 4) 为什么 QR 很稳、怎么用？

- QR 把矩阵分成“**正交旋转**  $Q$ ”和“**上三角**  $R$ ”。
- 正交矩阵不改变长度（不会放大/缩小误差）；
- 解  $Rw = Q^\top \tilde{y}$  只需**反代**，稳定又高效。
- 复杂度  $O(ND^2)$ （典型  $N \geq D$  时）。

### 5) 何时用 SVD？公式 (7.44) 意味着什么？

- 当**特征远多于样本** ( $D \gg N$ ) 时，SVD 更高效更稳。
- 紧凑 SVD： $X = USV^\top$ （只保留前  $N$  个奇异值/向量）。
- 岭解可写成

$$\hat{w} = V \operatorname{diag}\left(\frac{s_i}{s_i^2 + \lambda}\right) U^\top y \quad (\text{常见等价形式}) ,$$

这说明：

- 在每个奇异向量方向上，OLS 的系数  $\frac{1}{s_i}$  被**缩小**为  $\frac{s_i}{s_i^2 + \lambda}$ ；
- 小奇异值方向（不稳定、噪声大）被强烈压缩，稳定性和泛化都更好。
- 复杂度约  $O(DN^2)$ （典型  $N \ll D$  时）。

### 6) 实操小结

- 先中心化**特征与目标（避免惩罚  $w_0$ ）。
- 选一种**稳健求解**：
  - 常规：QR on  $\tilde{X}$ ，算  $R^{-1}Q^\top \tilde{y}$ ；
  - 高维少样本：SVD on  $X$ ，用奇异值公式。
- 不要 `inv()`；在 Matlab/NumPy 里用**方程求解器**（`x\y`、`np.linalg.lstsq`）。

这样，式 (7.34)–(7.44) 就从“为何这么构造”到“怎样高效稳定地求解”串起来了。需要的话，我可以用一组小数据给你演示 QR 与 SVD 两种解法算出来是一样的。



翻译并解释以下内容，翻译不能有遗漏，解释请基础且详尽，因为我的数学基础不太好。

Figure 7.9 Geometry of ridge regression. The likelihood is shown as an ellipse, and the prior is shown as a circle centered on the origin. Based on Figure 3.15 of (Bishop 2006b). Figure generated by geomRidge

### 7.5.3 Connection with PCA

In this section, we discuss an interesting connection between ridge regression and PCA (Section 12.2), which gives further insight into why ridge regression works well. Our discussion is based on (Hastie et al. 2009, p66). Let  $X = USVT$  be the SVD of  $X$ . From Equation 7.44, we have

$$\hat{w}_{\text{ridge}} = V(S^2 + \lambda I)^{-1} S U T y \quad (7.45)$$

Hence the ridge predictions on the training set are given by

$$\begin{aligned} \hat{y} &= X \hat{w}_{\text{ridge}} = USVT(S^2 + \lambda I)^{-1} S U T y \quad (7.46) \\ &= U D^{-1} U^T y = D_{j=1}^m u_j \tilde{S} j u_j^T y \quad (7.47) \end{aligned}$$

where

$$\tilde{S} j = [S(S^2 + \lambda I)^{-1} S] j = \sigma_j^2 j \quad (7.48)$$

and  $\sigma_j$  are the singular values of  $X$ . Hence

$$\hat{y} = X \hat{w}_{\text{ridge}} = D_{j=1}^m u_j \sigma_j^2 j + \lambda u_j^T y \quad (7.49)$$

In contrast, the least squares prediction is

$$\begin{aligned} \hat{y} &= X \hat{w}_{\text{ls}} = (USVT)(VS^{-1}U^T y) = U U^T y = D_{j=1}^m u_j u_j^T y \quad (7.50) \end{aligned}$$

If  $\sigma_j^2 j$  is small compared to  $\lambda$ , then direction  $u_j$  will not have much effect on the prediction. In view of this, we define the effective number of degrees of freedom of the model as follows:

$$\text{dof}(\lambda) = D_{j=1}^m \sigma_j^2 j + \lambda \quad (7.51)$$

When  $\lambda = 0$ ,  $\text{dof}(\lambda) = D$ , and as  $\lambda \rightarrow \infty$ ,  $\text{dof}(\lambda) \rightarrow 0$ . Let us try to understand why this behavior is desirable. In Section 7.6, we show that  $\text{cov}[w|D] = \sigma^2(X^T X)^{-1}$ , if we use a uniform prior for  $w$ . Thus the directions in which we are most uncertain about  $w$  are determined by the eigenvectors of this matrix with the smallest eigenvalues, as shown in Figure 4.1. Furthermore, in Section 12.2.3, we show that the squared singular values  $\sigma_j^2 j$  are equal to the eigenvalues of  $X^T X$ . Hence small singular values  $\sigma_j$  correspond to directions with high posterior variance. It is these directions which ridge shrinks the most.

This process is illustrated in Figure 7.9. The horizontal  $w_1$  parameter is not-well determined by the data (has high posterior variance), but the vertical  $w_2$  parameter is well-determined. Hence  $w_{\text{map}} 2$  is close to  $\hat{w}_{\text{MLE}} 2$ , but  $w_{\text{map}} 1$  is shifted strongly towards the prior mean, which is 0. (Compare to Figure 4.14(c), which illustrated sensor fusion with sensors of different reliabilities.) In this way, ill-determined parameters are reduced in size towards 0. This is called shrinkage. There is a related, but different, technique called principal components regression. The

idea is this: first use PCA to reduce the dimensionality to K dimensions, and then use these low dimensional features as input to regression. However, this technique does not work as well as ridge in terms of predictive accuracy (Hastie et al. 2001, p70). The reason is that in PC regression, only the first K (derived) dimensions are retained, and the remaining D – K dimensions are entirely ignored. By contrast, ridge regression uses a “soft” weighting of all the dimensions.

已思考 22s

好的！我把原文逐句翻译，然后用直观、基础的方式解释里面的式子与图意。

## 翻译

图 7.9 岭回归的几何图像。红色椭圆表示似然，绿色圆表示先验（以原点为中心）。改编自 (Bishop, 2006b) 的图 3.15。本图由 geomRidge 生成。

### 7.5.3 与 PCA 的联系

本节讨论岭回归与 PCA（第 12.2 节）之间的一个有趣联系，这能进一步解释为什么岭回归工作得很好。我们的讨论基于 (Hastie 等, 2009, 第 66 页)。设  $X = USV^\top$  是  $X$  的 SVD。由式 (7.44) 可得

$$\hat{\mathbf{w}}_{\text{ridge}} = V(S^2 + \lambda I)^{-1} S U^\top \mathbf{y} \quad (7.45)$$

因此，在训练集上的岭回归预测为

$$\hat{\mathbf{y}} = X \hat{\mathbf{w}}_{\text{ridge}} = USV^\top V(S^2 + \lambda I)^{-1} S U^\top \mathbf{y} \quad (7.46)$$

$$= U \tilde{S} U^\top \mathbf{y} = \sum_{j=1}^D \mathbf{u}_j \tilde{S}_{jj} \mathbf{u}_j^\top \mathbf{y} \quad (7.47)$$

其中

$$\tilde{S}_{jj} = [S(S^2 + \lambda I)^{-1} S]_{jj} = \frac{\sigma_j^2}{\sigma_j^2 + \lambda} \quad (7.48)$$

并且  $\sigma_j$  是  $X$  的奇异值。因此

$$\hat{\mathbf{y}} = X \hat{\mathbf{w}}_{\text{ridge}} = \sum_{j=1}^D \mathbf{u}_j \frac{\sigma_j^2}{\sigma_j^2 + \lambda} \mathbf{u}_j^\top \mathbf{y} \quad (7.49)$$

与之对比，最小二乘 (OLS) 的预测是

$$\hat{\mathbf{y}} = X \hat{\mathbf{w}}_{\text{ls}} = (USV^\top)(V S^{-1} U^\top \mathbf{y}) = U U^\top \mathbf{y} = \sum_{j=1}^D \mathbf{u}_j \mathbf{u}_j^\top \mathbf{y} \quad (7.50)$$

如果  $\sigma_j^2$  相比  $\lambda$  很小，则方向  $\mathbf{u}_j$  对预测的影响很小。基于此，我们把模型的**有效自由度**定义为

$$\text{dof}(\lambda) = \sum_{j=1}^D \frac{\sigma_j^2}{\sigma_j^2 + \lambda} \quad (7.51)$$

当  $\lambda = 0$  时， $\text{dof}(\lambda) = D$ ；当  $\lambda \rightarrow \infty$  时， $\text{dof}(\lambda) \rightarrow 0$ 。

让我们理解为什么这种行为是理想的。在第 7.6 节，我们将证明：若对  $\mathbf{w}$  使用均匀先验，则

$$\text{cov}[\mathbf{w} \mid D] = \sigma^2(X^\top X)^{-1}.$$

因此，对  $\mathbf{w}$  最不确定的方向由该矩阵**最小特征值**对应的特征向量决定（参见图 4.1）。此外，在 12.2.3 节我们会看到， $\sigma_j^2$  就是  $X^\top X$  的特征值。所以，较小的奇异值  $\sigma_j$  对应**后验方差很大的方向**；而岭回归正是在这些方向上**收缩得最厉害**。

这一过程如图 7.9 所示。水平方向的参数  $w_1$  被数据确定得不好（后验方差大），而竖直方向的  $w_2$  被数据确定得很好。因此  $\hat{w}_2^{\text{map}}$  接近  $\hat{w}_2^{\text{mle}}$ ，但  $\hat{w}_1^{\text{map}}$  被强烈拉向先验均值（即 0）。（可与图 4.14(c) 比较，后者展示了不同可靠性传感器的融合。）通过这种方式，**确定性差的参数会被压向 0，这称为收缩 (shrinkage)**。

一个相关但不同的技术叫**主成分回归** (PCR)：先用 PCA 把维度降到  $K$ ，再用这些低维特征做回归。但它在预测精度上通常不如岭回归 (Hastie 等, 2001, 第 70 页)。原因是 PCR 只保留前  $K$  个（变换后的）维度，剩下的  $D - K$  维完全丢弃；而岭回归对所有维度做“软权重”处理。

## 解释 (基础且详尽)

### 1) 图 7.9 在表达什么？

- 红色椭圆 = 似然的等高线：给定数据后，哪些  $(w_1, w_2)$  更可能。椭圆在某方向拉得很长，说明那个方向数据**约束弱/不确定性大**。
- 绿色圆 = 先验：我们事先相信参数应该靠近 0（圆心在原点，圆形说明各方向同等偏好）。
- MLE (只看数据) 是红椭圆的中心；MAP (数据 + 先验) 是折中点：在不确定性的大的方向 (椭圆长轴) 被更强地拉回原点，这就是“收缩”。

### 2) SVD 分解和几个向量是谁？

- $X = USV^T$ ：
  - $U \in \mathbb{R}^{N \times D}$ ：列向量  $u_j$  是训练样本空间中的“主方向”（左奇异向量）；
  - $V \in \mathbb{R}^{D \times D}$ ：列向量  $v_j$  是特征空间中的主方向（右奇异向量）；
  - $S = \text{diag}(\sigma_1, \dots, \sigma_D)$ ：奇异值（非负）。 $\sigma_j^2$  即  $X^T X$  的特征值。

### 3) 岭回归的“软投影”公式 (7.45–7.49) 说明了什么？

- OLS 的预测：把  $y$  投到所有  $u_j$  上，然后完整保留每个方向（权重 = 1）。
- 岭回归的预测：同样先投影，但每个方向都乘上一个**缩放因子**

$$\alpha_j = \frac{\sigma_j^2}{\sigma_j^2 + \lambda} \in [0, 1]$$

- $\sigma_j^2$  大（数据在该方向信息足） $\rightarrow \alpha_j \approx 1$ （几乎不衰减）；
- $\sigma_j^2$  小（数据信息弱/噪声大） $\rightarrow \alpha_j \ll 1$ （被强烈压缩）。

这就是“**在不靠谱的方向上少相信数据，多相信先验**”。

### 4) 有效自由度 $\text{dof}(\lambda)$ 是啥？

- 线性回归的“帽子矩阵” $H_\lambda = X(X^T X + \lambda I)^{-1} X^T = U \text{diag}(\alpha_j) U^T$ 。
- $\text{dof}(\lambda) = \text{trace}(H_\lambda) = \sum_j \alpha_j$ 。它度量“模型有效用了多少维度”：
  - $\lambda = 0$ （无收缩） $\rightarrow$ 每个  $\alpha_j = 1 \rightarrow \text{dof} = D$ 。
  - $\lambda \rightarrow \infty \rightarrow$ 每个  $\alpha_j \rightarrow 0 \rightarrow \text{dof} \rightarrow 0$ 。
- 这正是“从复杂到简单”的平滑控制旋钮。

### 5) 为什么岭回归优于主成分回归 (PCR)？

- PCR：只保留前  $K$  个主成分，硬截断，其余信息全扔掉；
- 岭：对所有方向做**软权重** ( $\alpha_j$  介于 0 和 1)，既抑制不可靠方向，又保留一点可能有用的信息 $\rightarrow$ 预测往往更好。

## 一句话总结

- 岭回归 = OLS + 高斯先验  $\Rightarrow$  在 SVD 坐标系里，对每个主方向乘上  $\sigma_j^2 / (\sigma_j^2 + \lambda)$  的**收缩因子**；
- 小奇异值（数据不确定方向）被强烈收缩  $\rightarrow$  防过拟合、数值更稳；
- 有效自由度 = 这些收缩因子的总和，随  $\lambda$  从  $D$  逐步降到 0；

- 与 PCA 的关系：岭是“软”版本，PCR 是“硬”截断，岭通常更准。



翻译并解释以下内容，翻译不能有遗漏，解释请基础且详尽，因为我的数学基础不太好。

Figure 7.10 MSE on training and test sets vs size of training set, for data generated from a degree 2 polynomial with Gaussian noise of variance  $\sigma^2 = 4$ . We fit polynomial models of varying degree to this data. (a) Degree 1. (b) Degree 2. (c) Degree 10. (d) Degree 25. Note that for small training set sizes, the test error of the degree 25 polynomial is higher than that of the degree 2 polynomial, due to overfitting, but this difference vanishes once we have enough data. Note also that the degree 1 polynomial is too simple and has high test error even given large amounts of training data. Figure generated by linregPolyVsN.

#### 7.5.4 Regularization effects of big data

Regularization is the most common way to avoid overfitting. However, another effective approach — which is not always available — is to use lots of data. It should be intuitively obvious that the more training data we have, the better we will be able to learn.<sup>2</sup> So we expect the test set error to decrease to some plateau as N increases. This is illustrated in Figure 7.10, where we plot the mean squared error incurred on the test set achieved by polynomial regression models of different degrees vs N (a plot of error vs training set size is known as a learning curve). The level of the plateau for the test error consists of two terms: an irreducible component that all models incur, due to the intrinsic variability of the generating process (this is called the noise floor); and a component that depends on the discrepancy between the generating process (the “truth”) and the model: this is called structural error. In Figure 7.10, the truth is a degree 2 polynomial, and we try fitting polynomials of degrees 1, 2 and 25 to this data. Call the 3 models M1, M2 and M25. We see that the structural error for models M2 and M25 is zero, since both are able to capture the true generating process. However, the structural error for M1 is substantial, which is evident from the fact that the plateau occurs high above the noise floor.

For any model that is expressive enough to capture the truth (i.e., one with small structural error), the test error will go to the noise floor as  $N \rightarrow \infty$ . However, it will typically go to zero faster for simpler models, since there are fewer parameters to estimate. In particular, for finite training sets, there will be some discrepancy between the parameters that we estimate and the best parameters that we could estimate given the particular model class. This is

called approximation error, and goes to zero as  $N \rightarrow \infty$ , but it goes to zero faster for simpler models. This is illustrated in Figure 7.10. See also Exercise 7.1. In domains with lots of data, simple methods can work surprisingly well (Halevy et al. 2009). However, there are still reasons to study more sophisticated learning methods, because there will always be problems for which we have little data. For example, even in such a data-rich domain as web search, as soon as we want to start personalizing the results, the amount of data available for any given user starts to look small again (relative to the complexity of the problem).

In such cases, we may want to learn multiple related models at the same time, which is known as multi-task learning. This will allow us to “borrow statistical strength” from tasks with lots of data and to share it with tasks with little data. We will discuss ways to do later in the book.

已思考 5s

好的！我把原文逐句完整翻译，再用尽量通俗的方式解释图和概念。

## 翻译

**图 7.10** 训练集和测试集上的 MSE 随训练集大小变化的曲线。数据由一个二次多项式 (degree 2) 加方差为  $\sigma^2 = 4$  的高斯噪声生成。我们用不同次数的多项式去拟合这些数据。(a) 一次多项式。(b) 二次多项式。(c) 十次多项式。(d) 25 次多项式。请注意：当训练集很小时，25 次多项式的测试误差比 2 次多项式更高，这是过拟合造成的；但一旦数据足够多，这个差异会消失。还要注意：一次多项式太简单了，即使给了大量训练数据，它的测试误差仍然很高。图由 `linregPolyVsN` 生成。

### 7.5.4 大数据的“正则化效应”

避免过拟合最常见的方法是正则化。不过，另一种（并非总能做到的）有效方法是使用大量数据。直觉上我们都能明白：训练数据越多，学习得越好。因此我们期望：当  $N$  增大时，测试误差会下降并趋于某个平台值。这在图 7.10 中得到了展示：我们把不同阶数的多项式回归在测试集上的均方误差随  $N$  的变化画成了曲线（这种“误差 vs 训练集大小”的图叫学习曲线）。

测试误差的平台高度由两部分组成：

- 一部分是不可约误差，由生成过程本身的内在波动造成（称为噪声下限 noise floor）；
- 另一部分取决于真实生成过程（“真相”）与模型之间的不匹配，称为结构性误差。

在图 7.10 中，真相是二次多项式；我们尝试用 1 次、2 次和 25 次多项式去拟合。把三种模型分别记作  $M_1, M_2, M_{25}$ 。可以看到：模型  $M_2$  与  $M_{25}$  都能表达真实生成过程，所以它们的结构性误差为 0；而  $M_1$  的结构性误差很大，这从它的测试误差平台明显高于噪声下限就能看出来。

对于任何足够表达真相的模型（即结构性误差小的模型），当  $N \rightarrow \infty$  时，测试误差都会靠近噪声下限。不过，一般来说，简单模型到达这个下限会更快，因为需要估计的参数更少。更具体地说，对于有限训练集，模型得到的参数与“该模型族中最优参数”之间会有差距，这叫做近似误差；当  $N \rightarrow \infty$  时它会消失，但在简单模型里消失得更快。图 7.10 就说明了这一点，另见习题 7.1。

在数据非常多的领域，简单方法可能出人意料地好用 (Halevy 等, 2009)。但仍有研究更复杂方法的理由：总会有数据稀缺的问题。例如，即使是在数据极其丰富的网页搜索中，一旦我们想做个性化，针对某个用户的数据量相对问题的复杂度就又变小了。

在这些情况下，我们可能希望同时学习多个相关的模型，这叫多任务学习。这样可以从数据多的任务“借统计力量”，共享给数据少的任务。后文我们会讨论具体做法。

## 解释（基础且详尽）

### 1) 图该怎么看？

- 横轴：训练集大小  $N$ 。
- 纵轴：MSE（均方误差），分别画训练误差（蓝虚线）和测试误差（红实线）。
- 黑色水平线：噪声下限 ( $\sigma^2 = 4$  的量级；注意具体单位由实验设定决定)。

四张子图对应拟合的多项式阶数：

- (a) degree 1 (欠拟合)：模型太简单，哪怕  $N$  很大，测试误差也降不到噪声下限——结构性误差大。
- (b) degree 2 (刚好)：与真相同阶，训练越多，测试误差越接近噪声下限。
- (c) degree 10 (偏复杂)、(d) degree 25 (很复杂)：
  - 当  $N$  小时，测试误差很大（尤其是 25 次）：过拟合，因为参数多、方差大；
  - 随着  $N$  增加，过拟合被数据“冲淡”，测试误差下降并接近噪声下限。
  - 当数据足够多时，复杂模型与合适模型的测试误差差异几乎消失。

关键词：

- **过拟合（高方差）**：模型太灵活、参数多，少量数据不足以把参数估准 → 测试误差高；
- **欠拟合（高偏差/结构性误差）**：模型太简单，表达不了真相 → 无论数据多少，测试误差都高；
- **噪声下限**：数据里不可避免的噪声导致的误差下界，任何模型都无法再低。

### 2) 为什么“更多数据”像“正则化”？

- 正则化（如岭回归）通过**压缩参数**来降低方差；
- 而**增大  $N$** 会让参数估计更稳定，方差也随之按  $\sim 1/N$  的速度下降；
- 二者都能减轻过拟合，因此“**大数据自带正则化效果**”。

### 3) 三种“误差”别混淆

- **噪声（不可约误差）**：来自随机噪声，怎么做都无法消除——决定平台的最低高度；
- **结构性误差**（模型与真相不匹配造成的“偏差”）：模型太简单时出现， $N$  再大也消不掉；
- **近似误差**（有限样本导致参数没估准）： $N$  增大时会下降，**简单模型下降更快**。

### 4) 实战建议

- 数据很少：优先**简单模型 + 正则化**（防过拟合）。
- 数据很多：可以尝试更复杂的模型，但仍要**监控测试误差/学习曲线**。
- 个性化、小样本、多任务：考虑**共享信息**（多任务学习、迁移学习）。

### 一句话总结

- 小数据 + 复杂模型  $\Rightarrow$  容易过拟合；
- 数据增加会把测试误差推向**噪声下限**，复杂模型与合适模型的差异会缩小；
- 但过于简单的模型有**结构性误差**，再多数据也救不了。
- 因此：**正则化**和**多数据**都是抑制过拟合的有效手段，能用就两者都用，并用**学习曲线**检查效果。



翻译并解释以下内容，翻译不能有遗漏，解释请基础且详尽，因为我的数学基础不太好。

Figure 7.11 Sequential Bayesian updating of a linear regression model  $p(y|x) = N(y|w_0x_0 + w_1x_1, \sigma^2)$ . Row 0 represents the prior, row 1 represents the first data point  $(x_1, y_1)$ , row 2 represents the second data point  $(x_2, y_2)$ , row 3 represents the 20th data point  $(x_{20}, y_{20})$ . Left column: likelihood function for current data point. Middle column: posterior given data so far,  $p(w|x_1:n, y_1:n)$  (so the first line is the prior). Right column: samples from the current prior/posterior predictive distribution. The white cross in columns 1 and 2 represents the true parameter value; we see that the mode of the posterior rapidly (after 20 samples) converges to this point. The blue circles in column 3 are the observed data points. Based on Figure 3.7 of (Bishop 2006a). Figure generated by bayesLinRegDemo2d.

### 7.6 Bayesian linear regression

Although ridge regression is a useful way to compute a point estimate, sometimes we want to compute the full posterior over  $w$  and  $\sigma^2$ . For simplicity, we will initially assume the noise variance  $\sigma^2$  is known, so we focus on computing  $p(w|D, \sigma^2)$ . Then in Section 7.6.3 we consider the general case, where we compute  $p(w, \sigma^2|D)$ . We assume throughout a Gaussian likelihood model. Performing Bayesian inference with a robust likelihood is also possible, but requires more advanced techniques (see Exercise 24.5).

7.6.1 Computing the posterior In linear regression, the likelihood is given by

$$p(y|X, w, \mu, \sigma^2) = N(y|\mu + Xw, \sigma^2 I_N) \quad (7.52)$$

$$\propto \exp(-\frac{1}{2\sigma^2} (y - \mu)^T (y - \mu)) \quad (7.53)$$

where  $\mu$  is an offset term. If the inputs are centered, so  $x_{ij} = 0$  for each  $j$ , the mean of the output is equally likely to be positive or negative. So let us put an improper prior on  $\mu$  of the form  $p(\mu) \propto 1$ , and then integrate it out to get

$$p(y|X, w, \sigma^2) \propto \exp(-\frac{1}{2\sigma^2} \|y - y_1\|^2) \quad (7.54)$$

where  $y_1 = \frac{1}{N} \sum_{i=1}^N y_i$  is the empirical mean of the output. For notational simplicity, we shall assume the output has been centered, and write  $y$  for  $y - y_1$ . The conjugate prior to the above Gaussian likelihood is also a Gaussian, which we will denote by  $p(w) = N(w|w_0, V_0)$ .

Using Bayes rule for Gaussians, Equation 4.125, the posterior is given by

$$p(w|X, y, \sigma^2) \propto N(w|w_0, V_0) N(y|Xw, \sigma^2 I_N) = N(w|w_N, V_N) \quad (7.55)$$

$$w_N = V_N^{-1} w_0 + V_N^{-1} X^T y \quad (7.56)$$

$$V_N^{-1} = V_0^{-1} + X^T X \quad (7.57)$$

$$V_N = \sigma^2 (V_0^{-1} + X^T X)^{-1} \quad (7.58)$$

If  $w_0 = 0$  and  $V_0 = \tau^2 I$ , then the posterior mean reduces

to the ridge estimate, if we define  $\lambda = \sigma^2 \tau^2$ . This is because the mean and mode of a Gaussian are the same. To gain insight into the posterior distribution (and not just its mode), let us consider a 1D example:

$$y(x, w) = w_0 + w_1 x + (7.59)$$

where the “true” parameters are  $w_0 = -0.3$  and  $w_1 = 0.5$ . In Figure 7.11 we plot the prior, the likelihood, the posterior, and some samples from the posterior predictive. In particular, the right hand column plots the function  $y(x, w(s))$  where  $x$  ranges over  $[-1, 1]$ , and  $w(s) \sim N(w|wN, VN)$  is a sample from the parameter posterior. Initially, when we sample from the prior (first row), our predictions are “all over the place”, since our prior is uniform. After we see one data point (second row), our posterior becomes constrained by the corresponding likelihood, and our predictions pass close to the observed data. However, we see that the posterior has a ridge-like shape, reflecting the fact that there are many possible solutions, with different slopes/intercepts. This makes sense since we cannot uniquely infer two parameters from one observation. After we see two data points (third row), the posterior becomes much narrower, and our predictions all have similar slopes and intercepts. After we observe 20 data points (last row), the posterior is essentially a delta function centered on the true value, indicated by a white cross. (The estimate converges to the truth since the data was generated from this model, and because Bayes is a consistent estimator; see Section 6.4.1 for discussion of this point.)

好的！我先把原文逐句完整翻译，再用基础而详尽的方式解释图、式子和它们之间的关系。

## 翻译

**图 7.11** 线性回归模型  $p(y | x) = N(y | w_0 x_0 + w_1 x_1, \sigma^2)$  的贝叶斯顺序更新示意。第 0 行表示先验；第 1 行表示第一个数据点  $(x_1, y_1)$ ；第 2 行表示第二个数据点  $(x_2, y_2)$ ；第 3 行表示第 20 个数据点  $(x_{20}, y_{20})$ 。

左列：当前数据点的似然函数。

中列：到目前为止给定数据的后验  $p(w | x_{1:n}, y_{1:n})$ （因此第一行就是先验）。

右列：来自当前先验/后验的预测分布的样本。第 1、2 列中的白色“十字”是参数的真值；可以看到后验的众数很快（20 个样本后）收敛到该点。第 3 列中的蓝色圆点是观测到的数据点。该图基于 (Bishop 2006a) 的图 3.7。图由 bayesLinRegDemo2d 生成。

## 7.6 贝叶斯线性回归

虽然岭回归能给出一个点估计，但有时我们希望计算  $w$  与  $\sigma^2$  的完整后验。为简单起见，先假设噪声方差  $\sigma^2$  已知，因此专注于  $p(w | D, \sigma^2)$  的计算。然后在第 7.6.3 节考虑一般情形，即计算  $p(w, \sigma^2 | D)$ 。我们始终假设高斯似然。用稳健似然也可以做贝叶斯推断，但需要更高阶技巧（见习题 24.5）。

### 7.6.1 计算后验

在线性回归中，似然为

$$p(y | X, w, \mu, \sigma^2) = N(y | \mu + Xw, \sigma^2 I_N) \quad (7.52)$$

$$\propto \exp\left(-\frac{1}{2\sigma^2}(y - \mu \mathbf{1}_N - Xw)^T(y - \mu \mathbf{1}_N - Xw)\right) \quad (7.53)$$

其中  $\mu$  是偏移项。若输入已中心化，即对每个特征  $j$  都有  $\sum_i x_{ij} = 0$ ，输出均值为正或为负是等可能的。因此对  $\mu$  采用不适当先验  $p(\mu) \propto 1$ ，并对其积分消去，得到

$$p(y | X, w, \sigma^2) \propto \exp\left(-\frac{1}{2\sigma^2} \|y - \bar{y}\mathbf{1}_N - Xw\|_2^2\right) \quad (7.54)$$

其中  $\bar{y} = \frac{1}{N} \sum_{i=1}^N y_i$  是输出的样本均值。为书写简便，假设我们已把输出做了中心化，用  $y$  表示  $y - \bar{y}\mathbf{1}_N$ 。

与上述高斯似然共轭的先验也是高斯，记为  $p(w) = N(w | w_0, V_0)$ 。利用高斯的贝叶斯法则（式 4.125），后验为

$$p(w | X, y, \sigma^2) \propto N(w | w_0, V_0) N(y | Xw, \sigma^2 I_N) = N(w | w_N, V_N) \quad (7.55)$$

$$w_N = V_0^{-1} w_0 + \frac{1}{\sigma^2} X^\top y \quad (7.56)$$

$$V_N^{-1} = V_0^{-1} + \frac{1}{\sigma^2} X^\top X \quad (7.57)$$

$$V_N = \sigma^2 (\sigma^2 V_0^{-1} + X^\top X)^{-1} \quad (7.58)$$

若  $w_0 = 0, V_0 = \tau^2 I$ ，把  $\lambda = \sigma^2/\tau^2$  代入可知：后验均值就等于岭回归解；因为对高斯来说，均值=众数。

为更直观地理解后验（不仅仅是它的众数），看一个一维示例：

$$y(x, w) = w_0 + w_1 x + \varepsilon \quad (7.59)$$

其中“真值”  $w_0 = -0.3, w_1 = 0.5$ 。图 7.11 画出了先验、似然、后验以及后验预测的样本。右列具体画的是：取  $x \in [-1, 1]$ ，从参数后验  $w^{(s)} \sim N(w_N, V_N)$  抽样，绘制  $y(x, w^{(s)})$ 。最初（第 0 行），从先验采样的预测“遍地飞”，因为先验很宽松；看到一个数据点后（第 1 行），后验被该点的似然约束，预测线都会经过（或接近）该点。但后验呈“脊形”，说明仅靠一个点，满足的直线有很多（斜率/截距不唯一）。看到两个点后（第 2 行），后验明显变窄，预测线的斜率与截距都接近；观察到 20 个点后（第 3 行），后验几乎变成以真值为中心的“尖峰”（近似  $\delta$  分布）。这是因为数据按该模型生成，且贝叶斯估计是一致的（见 6.4.1 节）。

## 解释（基础且详尽）

### 1) 三件事：先验、似然、后验

- 先验  $p(w)$ : 我们在看到数据前对参数的信念（这里取高斯  $N(w_0, V_0)$ ）。
- 似然  $p(y | X, w, \sigma^2)$ : 给定参数  $w$  后，数据  $y$  出现的概率（这里是高斯噪声  $\rightarrow$  二次型）。
- 后验  $p(w | X, y, \sigma^2)$ : 综合“先验  $\times$  似然”得到的更新后的信念（仍是高斯，式 7.55–7.58）。

这叫“共轭”：高斯  $\times$  高斯  $\rightarrow$  高斯，公式可直接写出。

### 2) 为什么要“中心化”并把 $\mu$ 积分掉？

- 如果特征已中心化，输出的平均水平 ( $\mu$ ) 没有偏好。
- 给  $\mu$  一个“均匀先验”并积分掉，等价于直接把  $y$  减去样本均值  $\bar{y}$  再做回归（式 7.54）。
- 好处：简化推导；并且不把“水平高度”与权重  $w$  的不确定性混在一起。

### 3) 岭回归为什么会有“从贝叶斯里‘蹦出来’”？

- 取先验  $w \sim N(0, \tau^2 I)$ ，后验均值

$$w_N = (X^\top X + \lambda I)^{-1} X^\top y, \lambda = \sigma^2/\tau^2$$

这正是岭回归。

- 直观：高斯先验“偏好小权重”，等价于  $\ell_2$  正则化——把不确定方向的参数收缩向 0。

### 4) 如何阅读图 7.11？

- 左列 (似然)**：每来一个数据点，就得到一条“可能的  $w$  组合”的带状区域（满足该点的直线）。
- 中列 (后验)**：先验与所有已见似然的结合。看到的点越多，后验越窄，定位越准；白十字是真值，后验峰值逐步靠近它。
- 右列 (预测)**：从当前后验抽参数、画出对应的直线；样本越多，这些直线越相似、越贴近蓝色观测点。

## 5) 公式 7.56–7.58 的直觉

- $V_N^{-1} = V_0^{-1} + \frac{1}{\sigma^2} X^\top X$ :  
“精度 = 先验精度 + 数据提供的精度”。数据越多或噪声越小 ( $\sigma^2$  越小)，后验越确定（方差更小）。
- $w_N = V_N(V_0^{-1}w_0 + \frac{1}{\sigma^2}X^\top y)$ :  
后验均值是“在先验均值与最小二乘解之间取权重的折中”。

### 小结

- 线性回归 + 高斯噪声 + 高斯先验  $\Rightarrow$  后验仍是高斯（可显式写出均值与协方差）。
- 当先验是  $w \sim N(0, \tau^2 I)$  时，后验均值就是岭回归解。
- 顺序更新：每来一个数据点，后验就从“宽”到“窄”，预测从“到处乱飞”到“稳定贴合”，最终收敛到真值（在模型正确时）。

如果你愿意，我可以用一组小数据把式 (7.56)–(7.58) 代入计算，展示一次从“先验  $\rightarrow$  后验”的具体数字变化，效果会更直观。

↳ 虽然岭回归能给出一个点估计，但有时我们希望计算  $w$  与  $\sigma^2$  的完整后验。

请解释这个

已思考 25s

明白 ~ 你指的是这句：

“虽然岭回归能给出一个点估计，但有时我们希望计算  $w$  和  $\sigma^2$  的完整后验。”

这里有两个关键词：**点估计 vs 完整后验**。我把它拆开讲。

### 1) 我们在估计什么？

- $w$ ：回归的参数（斜率们/权重量向量）。
- $\sigma^2$ ：噪声方差（数据里每次观测的随机波动强度）。

### 2) 点估计是什么（岭回归给的）？

- 岭回归在给定数据后，只输出一组“最好的”参数：  
 $\hat{w}_{\text{ridge}} = (X^\top X + \lambda I)^{-1} X^\top y$  (忽略偏置的写法)。
- 这是一个点（单一答案），没告诉我们“有多不确定”。

### 3) “完整后验”是什么意思？

- 后验分布  $p(w, \sigma^2 | D)$ ：看到数据  $D$  之后，所有可能的  $(w, \sigma^2)$  组合各自有多可能。
- 它不是一个点，而是一个分布，能量化不确定性。
- 有了它，我们可以：
  - 给每个参数做置信（贝叶斯叫可信）区间；
  - 对新点  $x_*$  预测时，用

$$p(y_* | x_*, D) = \iint N(y_* | x_*^\top w, \sigma^2) p(w, \sigma^2 | D) dw d\sigma^2$$

得到预测分布/预测区间（同时考虑了数据噪声和参数不确定性）。

#### 4) 怎么得到“完整后验”？（高斯情形很友好）

- 采用高斯似然（线性回归常见假设）。
- 选一个先验：常用的共轭先验是

$$w \mid \sigma^2 \sim N(w_0, \sigma^2 V_0), \quad \sigma^2 \sim \text{Inverse-Gamma}(a_0, b_0)$$

- 用贝叶斯规则更新后得到封闭形式（不用数值积分）：

$$w \mid \sigma^2, D \sim N(w_N, \sigma^2 V_N), \quad \sigma^2 \mid D \sim \text{Inverse-Gamma}(a_N, b_N)$$

其中（把  $y$  视为已中心化）

$$V_N = (V_0^{-1} + X^\top X)^{-1}, \quad w_N = V_N (V_0^{-1} w_0 + X^\top y),$$

$$a_N = a_0 + \frac{N}{2}, \quad b_N = b_0 + \frac{1}{2}(y^\top y + w_0^\top V_0^{-1} w_0 - w_N^\top V_N^{-1} w_N).$$

- 特例：若  $w_0 = 0, V_0 = \tau^2 I$  且把  $\lambda = \sigma^2/\tau^2$ ，则后验的众数/均值恰好等于岭回归解。也就是说：岭回归 = 贝叶斯线性回归的 MAP 点，而“完整后验”比它多了不确定性信息。

#### 5) 什么时候要“完整后验”？

- 数据少/噪声大**：点估计不可靠，需要不确定性条带来保护。
- 要给区间/风险控制**：如 A/B 测试、医疗、金融决策。
- 顺序更新**：新数据到来就把后验当先验继续更新；图 7.11 展示了这种逐步收敛。
- 模型比较**：可以用边际似然（证据）做模型选择，而不仅是交叉验证。

小更正：你引文里“www”应为  $w$ （参数向量）的小排版错误。

#### 一句话版

- 岭回归给一个“最好”的参数点；
- 贝叶斯线性回归给“参数和噪声的整条可能性分布”，因此能给可信区间和预测区间，对小数据或高风险决策更靠谱。



翻译并解释以下内容，翻译不能有遗漏，解释请基础且详尽，因为我的数学基础不太好。

Figure 7.12 (a) Plug-in approximation to predictive density (we plug in the MLE of the parameters). (b) Posterior predictive density, obtained by integrating out the parameters. Black curve is posterior mean, error bars are 2 standard deviations of the posterior predictive density. (c) 10 samples from the plugin approximation to posterior predictive. (d) 10 samples from the posterior predictive. Figure generated by linregPostPredDemo.

7.6.2 Computing the posterior predictive  
It's tough to make predictions, especially about the future.

— Yogi Berra

In machine learning, we often care more about predictions than about interpreting the parameters. Using Equation 4.126, we can easily show that the posterior predictive distribution at a test point  $x$  is also Gaussian:  
 $p(y|x, D, \sigma^2) = N(y|x^T w, \sigma^2) N(w|w_N, V_N) dw$  (7.60)  
 $= N(y|w^T Nx, \sigma^2 N(x))$  (7.61)  
 $\sigma^2 N(x) = \sigma^2 + x^T V_N x$  (7.62)

The variance in this prediction,  $\sigma^2 N(x)$ , depends on two terms: the variance of the observation noise,  $\sigma^2$ , and the variance in the parameters,  $V_N$ . The latter translates into variance about observations in a way which depends on how close  $x$  is to the training data  $D$ . This is illustrated in Figure 7.12, where we see that the error bars get larger as we move away from the training points, representing increased uncertainty. This is important for applications such as active learning, where we want to model what we don't know as well as what we do. By contrast, the plugin approximation has constant sized error bars, since  
 $p(y|x, D, \sigma^2) \approx N(y|x^T w, \sigma^2) \delta^w(w) dw = p(y|x, \hat{w}, \sigma^2)$  (7.63)

See Figure 7.12(a).

已思考 21s

好的！我先把原文完整翻译，然后再做基础而详尽的解释，帮你把图和公式都串起来。

## 翻译

图 7.12

- (a) 插件 (plug-in) 近似的预测密度 (把参数的 MLE 直接“代入” ) 。
- (b) 后验预测密度，通过把参数积分掉得到。黑色曲线是后验均值，误差条是后验预测密度的 2 倍标准差。
- (c) 来自“插件近似到后验预测”的 10 条函数样本。
- (d) 来自“后验预测”的 10 条函数样本。

图由 linregPostPredDemo 生成。

### 7.6.2 计算后验预测

“预测很难，尤其是预测未来。”——Yogi Berra

在机器学习中，我们常常更关心预测而不是解释参数。利用公式 4.126，我们很容易得到：在测试点  $x$  处的后验预测分布也是高斯分布：

$$p(y | x, D, \sigma^2) = \int N(y | x^T w, \sigma^2) N(w | w_N, V_N) dw \quad (7.60)$$

$$= N(y | w_N^T x, \sigma_N^2(x)) \quad (7.61)$$

$$\sigma_N^2(x) = \sigma^2 + x^T V_N x \quad (7.62)$$

这个预测方差  $\sigma_N^2(x)$  由两部分组成：

- 观测噪声的方差  $\sigma^2$ ；
- 参数的不确定性  $V_N$  经由  $x$  映射到观测的不确定性  $x^T V_N x$ 。

第二部分与测试点  $x$  离训练数据的远近有关。这在图 7.12 中有所体现：当远离训练点时，误差条变大，表示不确定性增加。这对诸如主动学习之类的应用很重要，因为我们既要刻画“知道的”，也要刻画“不知道的”。

与此相对，“插件近似”的误差条是常数宽度，因为

$$p(y | x, D, \sigma^2) \approx \int N(y | x^T w, \sigma^2) \delta_w(w) dw = p(y | x, \hat{w}, \sigma^2) \quad (7.63)$$

见图 7.12(a)。

## 解释 (基础且详尽)

### 1) “插件近似” vs “后验预测”

- **插件近似 (plug-in)** : 先用某个点估计 (常用 MLE 或 MAP) 得到  $\hat{w}$ , 预测就用  $N(y | x^\top \hat{w}, \sigma^2)$ 。  
只考虑观测噪声, 不考虑“参数还不确定”的那部分 → 误差条恒定 (与  $x$  无关)。
- **后验预测 (full Bayesian predictive)** : 把“参数的不确定性”也算进去:

$$\underbrace{\sigma^2}_{\text{数据内在噪声}} + \underbrace{x^\top V_N x}_{\text{参数不确定性传到输出}}$$

离训练点远时,  $x^\top V_N x$  往往大 → 误差条变宽, 更符合直觉。

直观: 插件近似像“只给你一条最佳曲线 + 固定噪声”; 后验预测像“给你一族可能的曲线 + 噪声”, 所以离已知数据越远, 不确定性越大。

### 2) 图 7.12 四幅图怎么读?

- (a) **插件近似**: 黑曲线  $x^\top \hat{w}$ ; 蓝色误差条宽度恒定 (只反映  $\sigma$ )。
- (b) **后验预测**: 黑曲线  $x^\top w_N$ ; 蓝色误差条随  $x$  改变, 离数据点时更宽。
- (c) “从插件近似采样的函数”: 由于参数被固定为  $\hat{w}$ , 采样出的曲线几乎重合 (只有小幅噪声抖动)。
- (d) “从后验预测采样的函数”: 先从  $w \sim N(w_N, V_N)$  采样, 再画  $y(x) = x^\top w$  (常不加观测噪声以便看清曲线差异)。曲线间差异明显, 尤其在缺少数据的区域。

### 3) 公式里的量是谁?

- $w_N, V_N$ : 上一小节算出的**参数后验**的均值与协方差;
- $\sigma^2$ : 观测噪声方差 (可知或也可在 7.6.3 联合推断);
- $\sigma_N^2(x)$ : 预测时的**总方差** (= 噪声 + 模型不确定性)。

### 4) 为什么这很重要?

- **风险意识**: 不仅给点预测, 还能给**预测区间**;
- **主动学习/采样**: 去那些误差条大的地方采新点, 信息增益更高;
- **小数据场景**: 参数不确定性很大, 只用插件近似会**过度自信**, 后验预测更可靠。

### 一句话总结

- 插件近似:  $y | x \approx N(x^\top \hat{w}, \sigma^2) \rightarrow$  误差条固定。
- 完整贝叶斯:  $y | x \sim N(x^\top w_N, \sigma^2 + x^\top V_N x) \rightarrow$  误差条随  $x$  变化, 离数据远更宽。
- 图 (a)-(d) 正在对比“只信一条最优曲线”与“考虑参数不确定性的一簇曲线”的差异。

翻译并解释以下内容, 翻译不能有遗漏, 解释请基础且详尽, 因为我的数学基础不太好。

#### 7.6.3 Bayesian inference when $\sigma^2$ is unknown

In this section, we apply the results in Section 4.6.3 to the problem of computing  $p(w, \sigma^2 | D)$  for a linear regression model. This generalizes the results from Section 7.6.1 where we assumed  $\sigma^2$  was known. In the case where we use an uninformative prior, we will see some interesting connections to frequentist statistics.

7.6.3.1 Conjugate prior As usual, the likelihood has the form

$$p(y | X, w, \sigma^2) = N(y | Xw, \sigma^2 I) \quad (7.64)$$

By analogy to Section 4.6.3, one can show that the natural

conjugate prior has the following form:

$$p(w, \sigma^2) = NIG(w, \sigma^2 | w_0, V_0, a_0, b_0) \quad (7.65)$$

$$= N(w | w_0, \sigma^2 V_0) IG(\sigma^2 | a_0, b_0) \quad (7.66)$$

$$= b_0 \Gamma(2\pi D/2 | V_0) \Gamma(a_0) (\sigma^2)^{-(a_0 + (D/2) + 1)} \quad (7.67)$$

$$\times \exp(-(w - w_0)^T V_0^{-1} (w - w_0) + 2b_0 \sigma^2) \quad (7.68)$$

With this prior and likelihood, one can show that the posterior has the following form:

$$p(w, \sigma^2 | D) = NIG(w, \sigma^2 | w_N, V_N, a_N, b_N) \quad (7.69)$$

$$w_N = VN(V^{-1} w_0 + X^T y) \quad (7.70)$$

$$V_N = (V^{-1} + X^T X)^{-1} \quad (7.71)$$

$$a_N = a_0 + n/2 \quad (7.72)$$

$$b_N = b_0 + 1/2 w^T V^{-1} w_0 + y^T y - w^T V_N^{-1} w_N \quad (7.73)$$

The expressions for  $w_N$  and  $V_N$  are similar to the case where  $\sigma^2$  is known. The expression for  $a_N$  is also intuitive, since it just updates the counts. The expression for  $b_N$  can be interpreted as follows: it is the prior sum of squares,  $b_0$ , plus the empirical sum of squares,  $y^T y$ , plus a term due to the error in the prior on  $w$ . The posterior marginals are as follows:

$$p(\sigma^2 | D) = IG(a_N, b_N) \quad (7.74)$$

$$p(w | D) = T(w_N, b_N, a_N, V_N, 2a_N) \quad (7.75)$$

We give a worked example of using these equations in Section 7.6.3.3. By analogy to Section 4.6.3.6, the posterior predictive distribution is a Student T distribution. In particular, given  $m$  new test inputs  $X'$ , we have

$$p(\tilde{y} | X', D) = T(\tilde{y} | X' w_N, b_N, a_N, (I_m + X' V_N X'^T), 2a_N) \quad (7.76)$$

The predictive variance has two components:  $(b_N/a_N)I_m$  due to the measurement noise, and  $(b_N/a_N)X' V_N X'^T$  due to the uncertainty in  $w$ . This latter term varies depending on how close the test inputs are to the training data. It is common to set  $a_0 = b_0 = 0$ , corresponding to an uninformative prior for  $\sigma^2$ , and to set  $w_0 = 0$  and  $V_0 = g(X^T X)^{-1}$  for any positive value  $g$ . This is called Zellner's g-prior (Zellner 1986). Here  $g$  plays a role analogous to  $1/\lambda$  in ridge regression. However, the prior covariance is proportional to  $(X^T X)^{-1}$  rather than  $I$ . This ensures that the posterior is invariant to scaling of the inputs (Minka 2000b). See also Exercise 7.10. We will see below that if we use an uninformative prior, the posterior precision given  $N$  measurements is  $V^{-1} N = X^T X$ . The unit information prior is defined to contain as much information as one sample (Kass and Wasserman 1995). To create a unit information prior for linear regression, we need to use  $V^{-1} 0 = 1/N X^T X$ , which is equivalent to the g-prior with  $g = N$ .

### 7.6.3.2 Uninformative prior

An uninformative prior can be obtained by considering the uninformative limit of the conjugate g-prior, which corresponds to setting  $g = \infty$ . This is equivalent to an improper NIG prior with  $w_0 = 0$ ,  $V_0 = \infty I$ ,  $a_0 = 0$  and  $b_0 = 0$ , which gives  $p(w, \sigma^2) \propto \sigma^{-(D+2)}$ . Alternatively, we can start with the semi-conjugate prior  $p(w, \sigma^2) = p(w)p(\sigma^2)$ , and take each term to its uninformative limit individually, which gives  $p(w, \sigma^2) \propto \sigma^{-2}$ . This is equivalent to an improper NIG prior with  $w_0 = 0$ ,  $V = \infty I$ ,  $a_0 = -D/2$  and  $b_0 = 0$ . The corresponding posterior is given by

$$p(w, \sigma^2 | D) = NIG(w, \sigma^2 | w_N, V_N, a_N, b_N) \quad (7.77)$$

$$w_N = \hat{w} \text{ mle } = (X^T X)^{-1} X^T y \quad (7.78)$$

$$V_N = (X^T X)^{-1} \quad (7.79)$$

$$a_N = N - D/2 \quad (7.80)$$

$$b_N = s^2/2 \quad (7.81)$$

$$s^2 = (y - X^T \hat{w})^T (y - X^T \hat{w}) \quad (7.82)$$

The marginal distribution of the weights is given by

$$p(\mathbf{w} | D) = T(\mathbf{w} | \hat{\mathbf{w}}, s^2 N - D C, N - D) \quad (7.83)$$

where  $C = (X^\top X)^{-1}$  and  $\hat{\mathbf{w}}$  is the MLE. We discuss the implications of these equations below.

已思考 20s

好的！我把原文逐句完整翻译，然后给出基础且详尽的解释，把式子和直觉都串起来。

## 翻译

### 7.6.3 当 $\sigma^2$ 未知时的贝叶斯推断

本节把第 4.6.3 节的结果应用到线性回归里，来计算  $p(\mathbf{w}, \sigma^2 | D)$ 。这推广了第 7.6.1 节（那里假设  $\sigma^2$  已知）的结果。在使用**不含信息的先验**时，我们会看到它与**频率学派统计**之间一些有趣的联系。

#### 7.6.3.1 共轭先验

照常，似然为

$$p(\mathbf{y} | X, \mathbf{w}, \sigma^2) = N(\mathbf{y} | X\mathbf{w}, \sigma^2 I_N) \quad (7.64)$$

类比第 4.6.3 节，可以得到其**自然的共轭先验**为

$$p(\mathbf{w}, \sigma^2) = NIG(\mathbf{w}, \sigma^2 | \mathbf{w}_0, V_0, a_0, b_0) \quad (7.65)$$

$$= N(\mathbf{w} | \mathbf{w}_0, \sigma^2 V_0) IG(\sigma^2 | a_0, b_0) \quad (7.66)$$

$$= \frac{b_0^{a_0}}{(2\pi)^{D/2} |V_0|^{1/2} \Gamma(a_0)} (\sigma^2)^{-(a_0+D/2+1)} \quad (7.67)$$

$$\times \exp\left(-\frac{(\mathbf{w} - \mathbf{w}_0)^\top V_0^{-1}(\mathbf{w} - \mathbf{w}_0) + 2b_0}{2\sigma^2}\right) \quad (7.68)$$

在这个先验与似然下，后验为

$$p(\mathbf{w}, \sigma^2 | D) = NIG(\mathbf{w}, \sigma^2 | \mathbf{w}_N, V_N, a_N, b_N) \quad (7.69)$$

$$\mathbf{w}_N = V_N(V_0^{-1}\mathbf{w}_0 + X^\top \mathbf{y}) \quad (7.70)$$

$$V_N = (V_0^{-1} + X^\top X)^{-1} \quad (7.71)$$

$$a_N = a_0 + \frac{n}{2} \quad (7.72)$$

$$b_N = b_0 + \frac{1}{2}(\mathbf{w}_0^\top V_0^{-1} \mathbf{w}_0 + \mathbf{y}^\top \mathbf{y} - \mathbf{w}_N^\top V_N^{-1} \mathbf{w}_N) \quad (7.73)$$

其中  $\mathbf{w}_N$ 、 $V_N$  与“ $\sigma^2$  已知”的情形类似； $a_N$  的表达式也直观（只是更新“计数”）。对  $b_N$  的解释是：它等于**先验的平方和**  $b_0$ ，加上**经验平方和**  $\mathbf{y}^\top \mathbf{y}$ ，再加上一个由  $w$  的先验误差引起的项。边缘分布为

$$p(\sigma^2 | D) = IG(a_N, b_N) \quad (7.74)$$

$$p(\mathbf{w} | D) = T(\mathbf{w} | \mathbf{w}_N, \frac{b_N}{a_N} V_N, 2a_N) \quad (7.75)$$

第 7.6.3.3 节给出了使用这些公式的一个算例。类比第 4.6.3.6 节，**后验预测分布**是 Student-t 分布。具体地，给定  $m$  个新的测试输入  $\tilde{X}$ ，有

$$p(\tilde{\mathbf{y}} | \tilde{X}, D) = T(\tilde{\mathbf{y}} | \tilde{X}\mathbf{w}_N, \frac{b_N}{a_N}(I_m + \tilde{X}V_N\tilde{X}^\top), 2a_N) \quad (7.76)$$

预测方差包含两部分： $(b_N/a_N)I_m$  来自测量噪声， $(b_N/a_N)\tilde{X}V_N\tilde{X}^\top$  来自  $w$  的不确定性；后者会随测试点距离训练数据的远近而变化。常见的做法是取  $a_0 = b_0 = 0$ （对应  $\sigma^2$  的不含信息先验），且取  $\mathbf{w}_0 = 0$ 、 $V_0 = g(X^\top X)^{-1}$  ( $g > 0$  任意)。这叫 Zellner 的 g-先验。此处  $g$  的作用类似于岭回归中的  $1/\lambda$ 。不过，这里的先验协方差与  $(X^\top X)^{-1}$  成正比而不是与  $I$  成正比，这保证了**对输入缩放的不变性**。此外，如果使用不含信息先验， $N$  个观测后的后验精度为  $V_N^{-1} = X^\top X$ 。**单位信息先验**被定义为“包含相当于一个样本的信息量”；在线性回归中取  $V_0^{-1} = \frac{1}{N}X^\top X$ ，等价于  $g = N$  的 g-先验。

### 7.6.3.2 不含信息先验

把 g-先验取到无信息极限  $g = \infty$  可得到不含信息先验。这等价于不适当的 NIG 先验：

$\mathbf{w}_0 = 0, V_0 = \infty I, a_0 = 0, b_0 = 0$ , 从而  $p(\mathbf{w}, \sigma^2) \propto \sigma^{-(D+2)}$ 。

或者, 从半共轭先验  $p(\mathbf{w}, \sigma^2) = p(\mathbf{w})p(\sigma^2)$  出发, 分别把每项取到不含信息极限, 得到  $p(\mathbf{w}, \sigma^2) \propto \sigma^{-2}$ 。这等价于不适当的 NIG 先验:  $\mathbf{w}_0 = 0, V = \infty I, a_0 = -D/2, b_0 = 0$ 。  
对应的后验为

$$p(\mathbf{w}, \sigma^2 | D) = \text{NIG}(\mathbf{w}, \sigma^2 | \mathbf{w}_N, V_N, a_N, b_N) \quad (7.77)$$

$$\mathbf{w}_N = \hat{\mathbf{w}}_{\text{mle}} = (X^\top X)^{-1} X^\top \mathbf{y} \quad (7.78)$$

$$V_N = (X^\top X)^{-1} \quad (7.79)$$

$$a_N = \frac{N - D}{2} \quad (7.80)$$

$$b_N = \frac{s^2}{2} \quad (7.81)$$

$$s^2 = (\mathbf{y} - X\hat{\mathbf{w}}_{\text{mle}})^\top (\mathbf{y} - X\hat{\mathbf{w}}_{\text{mle}}) \quad (7.82)$$

权重的边缘分布为

$$p(\mathbf{w} | D) = T(\mathbf{w} | \hat{\mathbf{w}}, \frac{s^2}{N - D} C, N - D), \quad C = (X^\top X)^{-1} \quad (7.83)$$

其中  $\hat{\mathbf{w}}$  为 MLE。我们在下文讨论这些式子的含义。

## 解释 (基础且详尽)

### 1) 我们在估计什么? 为何要 NIG

- 模型:  $\mathbf{y} = X\mathbf{w} + \varepsilon, \varepsilon \sim N(0, \sigma^2 I)$ 。
- 既要估  $\mathbf{w}$ , 也要估  $\sigma^2$ 。选高斯似然 + 共轭先验 NIG (正态-逆伽马) :

$$\mathbf{w} | \sigma^2 \sim N(\mathbf{w}_0, \sigma^2 V_0), \quad \sigma^2 \sim IG(a_0, b_0).$$

这样后验仍是 NIG, 有封闭式更新 (7.69–7.73), 不用数值积分。

### 这些更新的直觉

- $V_N^{-1} = V_0^{-1} + X^\top X$ : 精度 = 先验精度 + 数据精度; 数据越多/信息越强, 协方差越小。
- $\mathbf{w}_N = V_N(V_0^{-1}\mathbf{w}_0 + X^\top \mathbf{y})$ : 后验均值是“在先验均值与 OLS 信息之间加权折中”。
- $a_N = a_0 + N/2$ : 自由度/计数的累加。
- $b_N$ : “平方和”累加: 先验的 + 数据的 + 纠正 (把先验与后验的一致性考虑进去)。

### 2) 后验边缘与后验预测

- $\sigma^2 | D \sim IG(a_N, b_N)$  (把不确定性留在分布上)。
- $\mathbf{w} | D \sim \text{多元 t}(\mathbf{w}_N, \frac{b_N}{a_N} V_N, 2a_N)$ 。t 分布比高斯更“厚尾”, 反映小样本时的不确定性更大。
- 新点  $\tilde{X}$  的后验预测也是 t 分布 (7.76) :
  - 方差 = 噪声  $(b_N/a_N)I$  + 参数不确定性  $(b_N/a_N)\tilde{X}V_N\tilde{X}^\top$ 。
  - 离训练数据远的  $\tilde{X}$  会让第二项变大 → 预测区间变宽 (合理的“谦虚”)。

### 3) g-先验与岭回归、尺度不变性

- 取  $V_0 = g(X^\top X)^{-1}$  (Zellner g-先验)
  - $g$  与岭回归里的  $1/\lambda$  类似:  $g$  大 → 先验更宽松, 收缩更弱。
  - 因为协方差  $\propto (X^\top X)^{-1}$ , 对输入的线性缩放不敏感 (不变性更好)。
- 单位信息先验: 让先验的“信息量≈一个样本”; 取  $V_0^{-1} = \frac{1}{N}X^\top X$  (等价  $g = N$ )。

### 4) 不含信息先验与频率学派的联系

- 把  $g$ -先验取极限  $g \rightarrow \infty$ , 得到“无信息”先验 (可能是不适当的, 但后验是适当的, 前提  $N > D$ ) 。
- 这时:
  - $\mathbf{w}_N = \hat{\mathbf{w}}_{\text{mle}}$  (后验中心 = OLS/MLE) ;
  - $V_N = (X^\top X)^{-1}$ ;
  - $\mathbf{w} | D$  为多元 t, 位置在 MLE、尺度为  $\frac{s^2}{N-D}(X^\top X)^{-1}$ 、自由度  $N - D$ 。
- 这正是经典 OLS 的置信区间/ t 检验的形式:
  - $s^2 = \text{RSS}$  (残差平方和) ;
  - $\frac{s^2}{N-D}$  是残差方差估计;
  - 由此导出的区间与频率学派的“t 区间”一致。
- 结论: 用不含信息先验的贝叶斯线性回归, 数值上就还原了经典 OLS 的区间估计与检验。

## 5) 小提示与注意

- 需要  $N > D$  保证  $X^\top X$  可逆、后验适当。
- 小样本时 t 分布的重尾能更谨慎地反映不确定性; 样本大时  $2a_N = N - D$  很大, t 近似高斯。
- 选择  $g$  (或一般先验超参数) 可用经验贝叶斯、交叉验证或边际似然比较。

## 一句话总结

- 未知  $\sigma^2$  时, 用 NIG (正态-逆伽马) 共轭: 得到封闭式后验与 t-型后验预测。
- 取不含信息先验时: 后验中心是 OLS, 权重边缘是多元 t (与经典 t 区间一致) ;
- $g$ -先验把“收缩强度”与数据几何对齐 (尺度不变),  $g$  的作用像岭回归的  $1/\lambda$ 。



翻译并解释以下内容, 翻译不能有遗漏, 解释请基础且详尽, 因为我的数学基础不太好。

Table 7.2 Posterior mean, standard deviation and credible intervals for a linear regression model with an uninformative prior fit to the caterpillar data. Produced by linregBayesCaterpillar.

7.6.3.3 An example where Bayesian and frequentist inference coincide \* The use of a (semi-conjugate) uninformative prior is interesting because the resulting posterior turns out to be equivalent to the results from frequentist statistics (see also Section 4.6.3.9). In particular, from Equation 7.83 we have  
 $p(w_j | D) = T(w_j | \hat{w}_j, C_{jj} s^2 / (N - D), N - D)$  (7.84)  
 This is equivalent to the sampling distribution of the MLE which is given by the following (see e.g., (Rice 1995, p542), (Casella and Berger 2002, p554)):  
 $w_j - \hat{w}_j | s_j \sim t_{N - D}$  (7.85)  
 where

$s_j = s_2 C_{jj} N - D$  (7.86)

is the standard error of the estimated parameter. (See Section 6.2 for a discussion of sampling distributions.) Consequently, the frequentist confidence interval and the Bayesian marginal credible interval for the parameters are the same in this case. As a worked example of this, consider the caterpillar dataset from (Marin and Robert 2007). (The details of what the data mean don't matter for our present purposes.) We can compute the posterior mean and standard deviation, and the 95% credible intervals (CI) for the regression coefficients using Equation 7.84. The results are shown in Table 7.2. It is easy to check that these 95% credible intervals are identical to the 95% confidence intervals computed using standard frequentist methods (see linregBayesCaterpillar for the code). We can also use these marginal posteriors to compute if the coefficients are "significantly" different from 0. An informal way to do this (without using decision theory) is to check if its 95% CI excludes 0. From Table 7.2, we see that the CIs for coefficients 0, 1, 2, 4, 5 are all significant by this measure, so we put a little star by them. It is easy to check that these results are the same as those produced by standard frequentist software packages which compute p-values at the 5% level. Although the correspondence between the Bayesian and frequentist results might seem appealing to some readers, recall from Section 6.6 that frequentist inference is riddled with pathologies. Also, note that the MLE does not even exist when  $N < D$ , so standard frequentist inference theory breaks down in this setting. Bayesian inference theory still works, although it requires the use of proper priors. (See (Maruyama and George 2008) for one extension of the g-prior to the case where  $D > N$ .)

已思考 33s

好的！我先逐句翻译，再给出基础且详尽的解释，帮你把表格、公式 (7.84)–(7.86) 和“贝叶斯=频率学派”的对应关系串起来。

## 翻译

表 7.2

对毛毛虫数据 (caterpillar data) 在不含信息先验下拟合线性回归模型后的后验均值、标准差和可信区间 (95% CI)。由 `linregBayesCaterpillar` 生成。

### 7.6.3.3 一个贝叶斯与频率学派结论一致的例子

(半共轭的) 不含信息先验很有趣，因为由此得到的后验与频率学派统计的结果等价 (参见 4.6.3.9 节)。特别地，由式 (7.83) 可得

$$p(w_j | D) = T(w_j | \hat{w}_j, \frac{C_{jj} s^2}{N-D}, N-D) \quad (7.84)$$

这与 MLE 的抽样分布是等价的 (见 Rice 1995, p542; Casella & Berger 2002, p554)：

$$\frac{w_j - \hat{w}_j}{s_j} \sim t_{N-D} \quad (7.85)$$

其中

$$s_j = \sqrt{\frac{s^2 C_{jj}}{N-D}} \quad (7.86)$$

是参数估计的标准误。 (关于抽样分布, 见 6.2 节。)

因此, 在这个设定下, 频率学派的置信区间与贝叶斯边缘可信区间是相同的。作为例子, 考虑 (Marin & Robert, 2007) 的毛毛虫数据。 (数据含义细节与此无关。) 利用式 (7.84) 我们可以计算回归系数的后验均值与标准差, 以及 95% 可信区间。结果见表 7.2。很容易核对: 这些 95% 可信区间与按标准频率学派方法计算的 95% 置信区间完全一致 (代码见 `linregBayesCaterpillar`)。我们也可用这些边缘后验判断系数是否与 0 “显著不同”: 一种不正式但常用的做法 (不诉诸决策理论) 是检查其 95% 区间是否不包含 0。从表 7.2 可见, 系数 w0、w1、w2、w4、w5 的区间都不含 0, 因此在“sig”列标上星号 \*。容易验证, 这与标准频率学派软件按 5% 显著性水平计算得到的 p 值结论一致。

尽管这种“贝叶斯与频率学派一致”的对应关系对一些读者很有吸引力, 但回想 6.6 节: 频率学派推断有很多病理问题。此外, 当  $N < D$  时, MLE 甚至不存在, 标准的频率学派理论在该情形失效; 而贝叶斯推断仍然可行, 但需要使用适当的先验。 (当  $D > N$  时 g-先验的一种扩展见 Maruyama & George, 2008。)

## 解释 (基础且详尽)

### 1) 表 7.2 每一列什么意思?

- $w_j$ : 第  $j$  个系数 ( $w_0$  是截距)。
- $E[w_j | D]$ : 后验均值 (在不含信息先验下, 它等于 OLS/MLE 点估计  $\hat{w}_j$ )。
- $\sqrt{\text{var}[w_j | D]}$ : 后验标准差, 刻画该系数的不确定性。
- 95% CI: 该系数的 95% 可信区间 (本例=频率学派 95% 置信区间)。
- sig: 显著性标记。区间不含 0 → 打 \*, 表示“显著不为 0”。

从表中可读出:  $w_0 \approx 11.0$  且显著为正;  $w_1 \approx -0.004$  (虽很小但区间不含 0);  $w_2 \approx -0.054$  显著为负;  $w_4 \approx -1.294$  显著为负;  $w_5 \approx 0.232$  显著为正; 而  $w_3, w_6, w_7, w_8, w_9, w_{10}$  的区间跨过 0 → 不显著。

### 2) 为什么“贝叶斯区间 = 频率学派区间”?

- 在“不含信息先验”下 (上一节已推导),  $w | D$  的边缘分布是多元 t; 对单个系数  $w_j$ :

$$w_j \sim T(\hat{w}_j, \frac{s^2 C_{jj}}{N-D}, N-D)$$

这恰与频率学派中

$$\frac{\hat{w}_j - w_j}{\sqrt{\theta^2 C_{jj}}} \sim t_{N-D}, \quad \theta^2 = \frac{\text{RSS}}{N-D}$$

的形式对应 (注意这里  $s^2 = \text{RSS}$ , 所以  $s_j = \sqrt{s^2 C_{jj}/(N-D)} = \sqrt{\theta^2 C_{jj}}$ )。

- 因此基于 t 分布构造的 95% 区间在两套框架下完全一致。

### 3) 如何用这些数做“显著性”判断?

- 频率学派: 计算  $t = \hat{w}_j/s_j$ , 看  $|t|$  是否超过临界值 (约 1.96 对应大样本、双侧 5%)。
- 贝叶斯 (此处不含信息先验): 看 95% 可信区间是否含 0; 不含 0 ↔ 频率学派 p 值  $< 0.05$ 。
- 表 7.2 的星号就是这么来的。

### 4) 注意与局限

- “显著”≠“重要” (效应大小仍要结合业务尺度看);
- 多重比较会增加假阳性 (这里只是演示, 不做校正);
- 当  $N < D$ : OLS/MLE 不存在, 频率学派路子断; 贝叶斯仍可用, 但需合适的 (proper) 先验来约束问题。

## 一句话总结

- 在不含信息先验下，贝叶斯线性回归的系数后验是 t 分布，其 95% 可信区间与 OLS 的 95% 置信区间完全一致；
- 表 7.2 就是把每个  $w_j$  的后验均值、标准差与 95% 区间列出来，并用“是否跨 0”打星，作为显著性指示。



翻译并解释以下内容，翻译不能有遗漏，解释请基础且详尽，因为我的数学基础不太好。

Figure 7.13 (a) Estimate of test MSE produced by 5-fold cross-validation vs  $\log(\lambda)$ . The smallest value is indicated by the vertical line. Note the vertical scale is in log units.  
 (c) Log marginal likelihood vs  $\log(\alpha)$ . The largest value is indicated by the vertical line. Figure generated by linregPolyVsRegDemo.

7.6.4 EB for linear regression (evidence procedure)  
 So far, we have assumed the prior is known. In this section, we describe an empirical Bayes procedure for picking the hyper-parameters. More precisely, we choose  $\eta = (\alpha, \lambda)$  to maximize the marginal likelihood, where  $\lambda = 1/\sigma^2$  be the precision of the observation noise and  $\alpha$  is the precision of the prior,  $p(w) = N(w|0, \alpha^{-1}I)$ . This is known as the evidence procedure (MacKay 1995b). See Section 13.7.4 for the algorithmic details. The evidence procedure provides an alternative to using cross validation. For example, in Figure 7.13(b), we plot the log marginal likelihood for different values of  $\alpha$ , as well as the maximum value found by the optimizer. We see that, in this example, we get the same result as 5-CV, shown in Figure 7.13(a). (We kept  $\lambda = 1/\sigma^2$  fixed in both methods, to make them comparable.) The principle practical advantage of the evidence procedure over CV will become apparent in Section 13.7, where we generalize the prior by allowing a different  $\alpha_j$  for every feature. This can be used to perform feature selection, using a technique known as automatic relevancy determination or ARD. By contrast, it would not be possible to use CV to tune D different hyper-parameters. The evidence procedure is also useful when comparing different kinds of models, since it provides a good approximation to the evidence:  

$$p(D|m) = p(D|w, m)p(w|m, \eta)p(\eta|m)dwd\eta \quad (7.87)$$
  

$$\approx \max \eta p(D|w, m)p(w|m, \eta)p(\eta|m)dw \quad (7.88)$$

It is important to (at least approximately) integrate over  $\eta$  rather than setting it arbitrarily, for reasons discussed in Section 5.3.2.5. Indeed, this is the method we used to evaluate the marginal likelihood for the polynomial regression models in Figures 5.7 and 5.8. For a “more Bayesian” approach, in which we model our uncertainty about  $\eta$  rather than computing point estimates, see Section 21.5.2.

好的！我先把原文完整翻译，然后给出通俗、逐点的解释。

## 翻译

图 7.13

- (a) 用 5 折交叉验证得到的测试 MSE 估计，横轴是  $\log(\lambda)$ 。竖线标出最小值。注意纵轴是 对数刻度。
- (b) 对数边际似然随  $\log(\alpha)$  的变化。竖线标出最大值。图由 `linregPolyVsRegDemo` 生成。

### 7.6.4 线性回归的经验贝叶斯 (evidence 方法)

到目前为止，我们都假设先验已知。本节介绍一种用经验贝叶斯 (Empirical Bayes, EB) 选择超参数的做法。更准确地说，我们选择  $\eta = (\alpha, \lambda)$  来最大化边际似然，其中  $\lambda = 1/\sigma^2$  是 观测噪声的精度 (方差的倒数)， $\alpha$  是先验的精度，先验取

$$p(\mathbf{w}) = N(\mathbf{w} \mid \mathbf{0}, \alpha^{-1} I).$$

这称为 evidence 方法 (MacKay, 1995)。算法细节见 13.7.4 节。

evidence 方法可作为交叉验证的替代。例如，在图 7.13(b) 中，我们画出了不同  $\alpha$  下的对数边际似然以及优化器找到的最大值。可以看到在这个例子里，我们得到的结果与 5 折交叉验证 (图 7.13(a)) 一致。（两种方法里都把  $\lambda = 1/\sigma^2$  固定，以便可比。）

evidence 方法相对 CV 的一个实际优势将在 13.7 节更明显：我们把先验推广为对每个特征 使用不同的  $\alpha_j$ 。这可用于特征选择，技术名为 ARD (自动相关性确定)。相反，CV 很难 同时调  $D$  个超参数。

在比较不同模型时，evidence 方法也很有用，因为它给出了对模型证据

$$p(D \mid m) = \iint p(D \mid \mathbf{w}, m) p(\mathbf{w} \mid m, \eta) p(\eta \mid m) d\mathbf{w} d\eta \quad (7.87)$$

的良好近似：

$$p(D \mid m) \approx \max_{\eta} \int p(D \mid \mathbf{w}, m) p(\mathbf{w} \mid m, \eta) p(\eta \mid m) d\mathbf{w}. \quad (7.88)$$

按 5.3.2.5 节讨论的原因，至少近似地对  $\eta$  做积分很重要，而不是随意把它固定。事实上，我们在图 5.7 与 5.8 的多项式回归里就是用这种方法计算边际似然的。若想要更“贝叶斯”的做法（对  $\eta$  也建模而不是取点估计），见 21.5.2 节。

## 解释 (基础且详尽)

### 1) 这两个超参数是谁？

- $\lambda = 1/\sigma^2$ : 噪声精度 (方差越小，精度越大)；控制“数据有多干净”。
- $\alpha$ : 先验精度；先验  $\mathbf{w} \sim N(0, \alpha^{-1} I)$ 。 $\alpha$  大  $\Rightarrow$  先验更紧  $\Rightarrow$  系数被更强地收缩 (更强的  $\ell_2$  正则)。

### 2) evidence (边际似然) 在做什么？

- 它衡量：在给定超参数  $\eta$  时，模型把数据  $D$  解释得有多好：

$$p(D \mid \eta) = \int p(D \mid \mathbf{w}, \lambda) p(\mathbf{w} \mid \alpha) d\mathbf{w}$$

——把参数  $\mathbf{w}$  积分掉，仅剩对  $\eta$  的函数。

- 最大化  $p(D \mid \eta)$  (或其对数) 来选  $\eta$ ，这就是经验贝叶斯 / type-II 最大似然。

直觉：evidence 自带“奥卡姆剃刀”。一方面希望拟合好 (后验均值把残差压小)，另一方面惩罚过于灵活的模型 ( $\log|C|$  这类项抑制复杂度)，因此能在“拟合度”和“复杂度”之间自动折中。

### 3) 如何读图 7.13？

- (a) 交叉验证：横轴  $\log \lambda$ ，纵轴 (对数刻度) 是 5 折 CV 得到的测试 MSE 估计；竖线 给出最小值处的  $\lambda$ 。

- (b) evidence: 横轴  $\log \alpha$ , 纵轴是**对数边际似然**; 坚线给出最大值处的  $\alpha$ 。
- 这个例子里, 两种方法选出的正则强度导致**相近的最佳模型** (性能一致)。

注: 为便于比较, 作者固定了其中一个超参数 (文中说固定  $\lambda$ ), 只用另一种方法来调另一个超参数。

#### 4) 为什么 evidence 比 CV 有时更合适?

- 效率: CV 要多次切分/多次训练; evidence 在一次训练中就能算 (有闭式表达式)。
- 多超参数可扩展: ARD 用  $\alpha_1, \dots, \alpha_D$  每维一个精度——CV 很难同时调  $D$  个; 而 EB 可通过最大化  $p(D | \alpha_1, \dots, \alpha_D, \lambda)$  来完成。
- 模型比较: 近似  $p(D | m)$  (式 7.87–7.88), 可直接比较不同模型 (不同基函数、不同结构), 不必依赖外部验证集。

#### 5) 实操步骤 (简版)

1. 选好特征/基函数  $X$ , 设先验  $w \sim N(0, \alpha^{-1} I)$ 。
2. 给定候选  $\alpha, \lambda$ , 计算对数边际似然
 
$$\log p(D | \alpha, \lambda) = -\frac{1}{2}(\log |C| + y^T C^{-1} y + N \log 2\pi), \quad C = \lambda^{-1} I + \alpha^{-1} X X^T.$$
3. 用数值优化找最优  $\alpha, \lambda$ 。
4. 代入得到的  $(\alpha, \lambda)$  计算后验  $w_N, V_N$  并做预测。
5. 若用 ARD, 把  $\alpha$  替换为  $\alpha_j$  (每个特征一个), 优化得到的  $\alpha_j$  很大时, 对应  $w_j$  被强收缩到 0  $\Rightarrow$  自动特征选择。

#### 6) 与“更贝叶斯”的版本

- EB 把  $\eta$  当**点估计** (用最大化替代积分, 式 7.88)。
- 更纯粹的做法是给  $\eta$  也放一个先验  $p(\eta)$ , 然后**对  $\eta$  积分** (见 21.5.2)。小数据时这样更保守, 但计算更重。

#### 小结

- 交叉验证和 \*\*evidence (经验贝叶斯) \*\*都能选正则强度; 本例里两者给出相同的最佳复杂度。
- evidence 的优势: 不需切分数据、能同时处理**很多超参数** (如 ARD), 还能用于**模型比较**。
- 但要记得: 真正“全贝叶斯”应对  $\eta$  也做积分; EB 是一种高效的近似。

基于本对话里我贴过的整章原文, 输出: ①知识点总结  
(分点+关键词+用途) ②子章节关联 (依赖/对比/递进+关系句, 附Mermaid可选) ③本章总结。

已思考 14s

好的! 基于你前面贴的整章原文 (第7章: 线性回归) 与配图, 这里给出三部分输出。

#### ① 知识点总结 (分点 + 关键词 + 用途)

- **线性回归模型**
  - **关键词:**  $y | x \sim N(w^T x, \sigma^2)$ , 基函数扩展  $\phi(x)$  (多项式等)
  - **用途:** 建模输入到连续输出的关系; 通过基函数可表达非线性。
- **最小二乘 = 极大似然 (MLE)**
  - **关键词:** RSS/SSE, MSE, NLL;  $\hat{w} = \arg \min \sum_i (y_i - w^T x_i)^2$
  - **用途:** 参数估计的基线方法; 误差解释与可视化 (残差线)。
- **正规方程与 OLS 闭式解**

- **关键词:**  $X^\top Xw = X^\top y$ ,  $\hat{w}_{OLS} = (X^\top X)^{-1}X^\top y$
- **用途:** 小规模问题直接解；与数值分解 (QR/SVD) 对比。
- **几何解释 & 帽子矩阵**
  - **关键词:** 投影到列空间;  $\hat{y} = X\hat{w} = PXy$ ,  $P = X(X^\top X)^{-1}X^\top$
  - **用途:** 理解“预测=正交投影”；诊断影响点与杠杆值 (leverages) 。
- **凸性**
  - **关键词:** NLL 二次“碗形”，唯一极小；集合/函数的凸性定义
  - **用途:** 保证全局最优；选择可安全优化的模型/损失。
- **鲁棒回归**
  - **关键词:** Laplace 似然 ( $|r|$  损失, 线性规划), Huber 损失 ( $\ell_2 \Leftrightarrow \ell_1$  混合)
  - **用途:** 对异常点不敏感；工程上常用 Huber (可用光滑优化) 。
- **岭回归 ( $\ell_2$  正则)**
  - **关键词:** MAP + 高斯先验；目标  $MSE + \lambda \|w\|_2^2$ ;  $\hat{w} = (\lambda I + X^\top X)^{-1}X^\top y$
  - **用途:** 抑制过拟合、提升数值稳定性；权重收缩、曲线更平滑。
- **数值稳定计算**
  - **关键词:** 避免 `inv()` ; QR:  $\tilde{X} = QR \Rightarrow \hat{w} = R^{-1}Q^\top \tilde{y}$ ; SVD:  $\hat{w} = V(Z^\top Z + \lambda I)^{-1}Z^\top y$
  - **用途:** 大幅提升稳定性；高维小样本 ( $D \gg N$ ) 优先用 SVD。
- **与 PCA 的联系 & 有效自由度**
  - **关键词:** 收缩因子  $\sigma_j^2 / (\sigma_j^2 + \lambda)$ ;  $dof(\lambda) = \sum_j \sigma_j^2 / (\sigma_j^2 + \lambda)$
  - **用途:** 理解为何岭更好 (对“小奇异值方向”强收缩) ；度量模型复杂度。
- **大数据的“正则化效应”**
  - **关键词:** 学习曲线；噪声下限 vs 结构性误差 vs 近似误差
  - **用途:** 判断“用更多数据”能否替代强正则；选择模型复杂度。
- **贝叶斯线性回归 ( $\sigma^2$  已知)**
  - **关键词:** 先验  $w \sim N(w_0, V_0)$ ; 后验  $w | D \sim N(w_N, V_N)$
  - **用途:** 获得不确定性 (方差/区间) ；顺序更新；MAP=岭回归。
- **后验预测**
  - **关键词:**  $y | x, D \sim N(w_N^\top x, \sigma^2 + x^\top V_N x)$ ; 插件近似 vs 全贝叶斯
  - **用途:** 给出随位置变化的预测区间；主动学习/风险评估。
- **$\sigma^2$  未知: NIG 共轭**
  - **关键词:**  $w, \sigma^2 \sim \text{Normal-Inverse-Gamma}$ ;  $w | D$  为 t 分布；预测为 t 分布
  - **用途:** 同时量化噪声与参数不确定性；小样本更保守。
- **不含信息先验 & 与频率学派等价**
  - **关键词:**  $\hat{w}_{MLE}$ 、 $(X^\top X)^{-1}$ 、t 区间；表7.2
  - **用途:** 贝叶斯可信区间=频率学派置信区间 (在该设定下) 。
- **经验贝叶斯 (Evidence/Type-II ML)**
  - **关键词:** 最大化边际似然  $p(D | \alpha, \lambda)$ ; 与 CV 对比；ARD ( $\alpha_j$  每维一个)
  - **用途:** 自动设超参；可扩展到多超参 (特征选择)，便于模型比较。

## ② 子章节关联 (依赖 / 对比 / 递进)

### 关系句 (文字版)

- 7.1→7.2: 从应用动机过渡到**模型定义** (高斯似然 + 基函数) 。
- 7.2→7.3: 用 MLE 推导到**最小二乘 (RSS)**，得到**OLS 闭式解** 与 **RSS 几何图**。
- 7.3→7.3.2/7.3.3: 一边给**几何/投影理解** (帽子矩阵)，一边给**凸性保证** (唯一最优) 。
- 7.3 → 7.4: **对比**: 平方损失易被异常点影响 ⇒ 用 Laplace/Huber 做鲁棒替代。

- 7.3 → 7.5: 从 MLE 过拟合问题递进到 MAP+高斯先验=岭回归；展示学习曲线与收缩。
- 7.5.2: 在 7.5 基础上数值化：QR/SVD/“伪数据”技巧，避免矩阵求逆。
- 7.5.3: 把岭放到 SVD/PCA 坐标下解释“为何有效”（小奇异值方向强收缩），引入有效自由度；与 PCR 对比（硬截断 vs 软收缩）。
- 7.5.4: 与正则化并列的另一条路：多数据可降低方差、逼近噪声下限。
- 7.6: 从点估计升级到贝叶斯全后验（已知/未知  $\sigma^2$ ），并给出后验预测（位置相关的不确定性）。
- 7.6.3.3: 在无信息先验下，贝叶斯区间 = 频率学派 t 区间（等价性）。
- 7.6.4: 对比 CV 与 Evidence/EB；为后续 ARD 与模型比较铺路。

可选 Mermaid (简图)

```
flowchart TD
    A[7.1 动机] --> B[7.2 模型定义/基函数]
    B --> C[7.3 MLE=最小二乘/OLS]
    C --> D[7.3.2 几何投影/帽子矩阵]
    D --> E[7.3.3 凸性=唯一最优]
    E --> F[7.4 鲁棒回归(Laplace/Huber)]
    F --> G[7.5 岭回归=MAP(高斯先验)]
    G --> H[7.5.2 数值稳定(QR/SVD)]
    H --> I[7.5.3 与PCA/有效自由度/收缩]
    I --> J[7.5.4 大数据的正则化效应]
    J --> K[7.6 贝叶斯线性回归]
    K --> L[7.6.2 后验预测]
    L --> M[7.6.3  $\sigma^2$  未知:NIG与t分布]
    M --> N[7.6.3.3 贝叶斯=频率学派]
    N --> O[7.6.4 经验贝叶斯/Evidence/ARD]
```

### ③ 本章总结 (提纲挈领)

- **核心主线：**从最基础的线性—高斯模型出发，用 MLE 获得 最小二乘；再用 MAP(高斯先验) 得到 岭回归 以防过拟合；随后进入 贝叶斯视角，给出全后验与后验预测（能表达不确定性），并讨论  $\sigma^2$  未知、无信息先验与频率学派等价；最后用 经验贝叶斯最大化边际似然来自动调超参，为 ARD 特征选择与模型比较打基础。
- **三层理解：**
  1. **代数：**正规方程、闭式解、NIG 共轭更新；
  2. **几何：**投影、帽子矩阵、SVD 收缩、有效自由度；
  3. **统计：**过拟合 vs 正则、鲁棒损失、学习曲线、可信区间与 t 区间、后验预测的不确定性分解。
- **实务要点：**
  - 小数据/噪声大：优先 岭/鲁棒损失，并报告区间（贝叶斯或 t）；
  - 计算上：别求逆，用 QR/SVD；
  - 调参：小规模可 CV，多超参/需比较模型时用 Evidence/EB；
  - 解释性：用 SVD/PCA 视角看收缩；用 学习曲线 判断是否“多数据即可”；
  - 异常值：尝试 Huber 或 Laplace；
  - 不确定性：尽量给 后验预测区间 ( $\sigma^2 + x^\top V_N x$  或 t 版)。

一句话：线性回归不仅是“拟一条线/面”，而是一整套从点估计到全后验、从数值稳定到超参自动化、从鲁棒性到不确定性的系统方法论；把这些积木组合起来，基本可以覆盖绝大多数回归场景的建模与落地。