

$$\alpha_k = \arg \min_{\alpha} \|\alpha(\theta_k - \theta_{k-1}) - (g_k - g_{k-1})\|_2^2 = \frac{(\theta_k - \theta_{k-1})^\top (g_k - g_{k-1})}{(\theta_k - \theta_{k-1})^\top (\theta_k - \theta_{k-1})} \quad (13.82)$$

这叫作 **Barzilai–Borwein (BB) /谱步长** (Barzilai and Borwein 1988; Fletcher 2005; Raydan 1997)。该步长可与任意梯度法 (含近端法或非近端法) 搭配。它**不保证**目标函数单调下降, 但通常比标准线搜索**快得多**。(为保证收敛, 需要目标函数在一个长度为 $M + 1$ 的滑动窗口里**平均下降**。)

当我们把 **BB 步长** 与 **迭代软阈值** ($R(\theta) = \lambda \|\theta\|_1$) 以及**逐步减小 λ** 的**延续法**结合起来, 就得到一个求解 BPDN 的快速方法——**SpaRSA** (“sparse reconstruction by separable approximation”, Wright et al., 2009)。不过我们仍把它称为**迭代收缩与阈值算法**。伪代码见 Algorithm 12, Matlab 代码见 SpaRSA。

13.4.3.3 Nesterov 方法

更快的近端梯度法可以通过在**不是最新点**的位置展开二次近似来得到。具体地, 考虑如下更新:

$$\theta_{k+1} = \text{prox}_{t_k R}(\phi_k - t_k g_k) \quad (13.83)$$

$$g_k = \nabla L(\phi_k) \quad (13.84)$$

$$\phi_k = \theta_k + \frac{k-1}{k+2}(\theta_k - \theta_{k-1}). \quad (13.85)$$

这称为 **Nesterov 方法** (Nesterov 2004; Tseng 2008)。与前面一样, 设定 t_k 有多种方式; 通常使用**线搜索**。

当该方法与**迭代软阈值** ($R(\theta) = \lambda \|\theta\|_1$) 及**延续法** (逐步减小 λ) 结合时, 就得到一个用于 BPDN 的快速方法, 称为 **FISTA** (fast iterative shrinkage-thresholding algorithm, Beck & Teboulle 2009)。

通俗解释

- **近端梯度法 = 梯度一步 + 近端一步**

先按损失 L 做普通**梯度步** $u_k = \theta_k - t_k \nabla L(\theta_k)$, 再把 u_k 通过**近端算子**处理一次 (软阈值或投影), 得到 θ_{k+1} 。

- 如果 R 是 ℓ_1 , 近端就是**软阈值** \Rightarrow 把小系数推到 0;
- 如果 R 是集合约束, 近端就是**投影** \Rightarrow 把点拉回可行域 (图 13.11) ;
- 如果 $R = 0$, 就是普通梯度下降。

- 为什么要 BB 步长?

$\alpha_k = 1/t_k$ 想近似 Hessian 的“弯曲程度”。BB 步长用“**割线条件**”

$$\alpha(\theta_k - \theta_{k-1}) \approx g_k - g_{k-1}$$

来拟合一个合适的尺度，通常比保守的线搜索**大得多**，因此**收敛更快**（但可能偶尔上升），于是用“窗口平均下降”来保证总体收敛）。

- Nesterov 加速 (FISTA 的灵魂)

不在 θ_k 上做近似，而在一个**外推点** $\phi_k = \theta_k + \frac{k-1}{k+2}(\theta_k - \theta_{k-1})$ 上做梯度与近端，相当于加了**动量**，能显著提速。配合软阈值与延续法，就是FISTA：在实践里求解 Lasso/BPDN 通常非常快。

要点小结

- 近端梯度框架：

$$\theta_{k+1} = \text{prox}_{t_k R}(\theta_k - t_k \nabla L(\theta_k)).$$

统一地处理“**光滑损失 + 非光滑正则**”。

- 特例：

- $R = \lambda \|\theta\|_1 \rightarrow$ **迭代软阈值** (ISTA/SpaRSA)；
- $R = I_C \rightarrow$ **投影梯度** (把梯度步投影回可行集)；
- $R = 0 \rightarrow$ **普通梯度下降**。

- 步长选择：** BB/谱步长用割线条件估 Hessian 尺度，通常比线搜索更快；用**滑动窗口平均下降**确保收敛。
- 加速：** 在外推点 ϕ_k 做近端更新的 Nesterov/FISTA 显著提速；与**延续法** (λ 逐步减小) 搭配效果更佳。
- 直观：** 把“梯度步”视作**朝谷底走**，“近端步”视作**遵守正则/约束的纠偏** (软阈值/投影)；两步交替，既快又能得到**稀疏/可行**的解。

这一小节主要在讲什么？我不太懂

已思考 10s

这小节主要讲的是：**用“近端梯度法/投影梯度法”来高效求解带稀疏正则（如 L1）的优化问题**，并介绍两种提速技巧（BB 步长与 Nesterov 加速，FISTA）。

用大白话说

- 目标函数被写成 **光滑的损失** $L(\theta)$ + **可能不光滑的正则** $R(\theta)$ (比如 L 是平方损失, R 是 L_1 或“落在某个集合里”的约束)。
- **每一步怎么走:**
 1. 先按损失做一次**梯度步**: $u_k = \theta_k - t_k \nabla L(\theta_k)$ (往“下坡方向”走一点) ;
 2. 再对 u_k 做**近端操作** $\text{prox}_{t_k R}$:
 - 如果 $R = \lambda \|\theta\|_1$, 就是**软阈值** (小系数直接变 0, 其他被拉小) ;
 - 如果 $R = I_C$ 是集合指示函数, 就是把点**投影回可行集合** (图 13.11 所示) ;
 - 如果 $R = 0$, 就退化为**普通梯度下降**。
这就是**近端梯度法** (又叫 ISTA/迭代软阈值) 。

- **步长怎么定:**

用 Barzilai–Borwein (BB) /**谱步长**来近似曲率, 比保守的线搜索快, 但允许偶尔不单调下降; 用“滑动窗口的平均下降”可保证总体收敛。配合“逐步减小 λ 的延续法”, 得到求解 BPDN 的**迭代收缩与阈值算法** (如 SpaRSA) 。

- **再加速:**

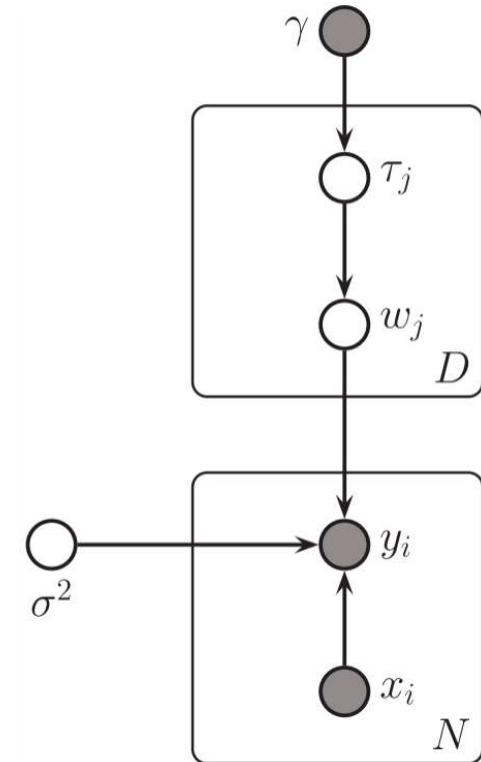
在一个**外推点** $\phi_k = \theta_k + \frac{k-1}{k+2}(\theta_k - \theta_{k-1})$ 做梯度与近端 (等于加了动量), 就是 Nesterov 加速; 与软阈值结合就是 FISTA, 比普通近端梯度更快。

- **适用场景:**

面向**超大规模**问题、只需一个或少量 λ 的解, 或需要容易扩展到其他正则/约束时, 近端/投影法很合适 (一次更新整个向量, 易并行) 。它和前面的**坐标下降** (一次只改一维) 与**主动集/同伦** (沿正则路径跳拐点) 做的是**同一类问题**, 但**方法不同**、更偏大规模与通用性。

要点小结

- **框架:** $\theta_{k+1} = \text{prox}_{t_k R}(\theta_k - t_k \nabla L(\theta_k))$ =梯度步 + 近端步。
- **近端算子:** $L_1 \rightarrow$ 软阈值; 集合约束 \rightarrow 投影; $R=0 \rightarrow$ 普通梯度。
- **图 13.11:** 梯度步可能跑出可行域, 投影把它拉回最近的可行点。
- **BB 步长:** 用割线条件估曲率, 通常比线搜索快; 与延续法结合形成 SpaRSA/ISTA。
- **Nesterov/FISTA:** 用外推点做更新, 显著提速 (常见于稀疏重建/Lasso) 。
- **与前法关系:** 目标一致; 坐标下降=逐坐标 prox, 主动集/同伦=沿路径热启动, 近端法=整向量 prox, **更适合大规模与多种正则/约束**。



翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

Figure 13.12 Representing lasso using a Gaussian scale mixture prior.

13.4.4 EM for lasso

In this section, we show how to solve the lasso problem using lasso. At first sight, this might seem odd, since there are no hidden variables. The key insight is that we can represent the Laplace distribution as a Gaussian scale mixture (GSM) (Andrews and Mallows 1974; West 1987) as follows:

$$\text{Lap}(w_j | 0, 1/\gamma) = \gamma^{-1} e^{-\gamma|w_j|} = N(w_j | 0, \tau_{2,j}) \text{Ga}(\tau_{2,j} | 1, \gamma^{-2}) d\tau_{2,j} \quad (13.86)$$

Thus the Laplace is a GSM where the mixing distribution on the variances is the exponential distribution, $\text{Expon}(\tau_{2,j} | \gamma^{-2}) = \text{Ga}(\tau_{2,j} | 1, \gamma^{-2})$. Using this decomposition, we can represent the lasso model as shown in Figure 13.12. The corresponding joint distribution has the form

$$p(y, w, \tau, \sigma^2 | X) = N(y | Xw, \sigma^2 I_N) N(w | 0, D\tau) \text{IG}(\sigma^2 | a\sigma, b\sigma) \prod_j \text{Ga}(\tau_{2,j} | 1, \gamma^{-2}) \quad (13.87)$$

where $D\tau = \text{diag}(\tau_{2,j})$, and where we have assumed for notational simplicity that X is standardized and that y is centered (so we can ignore the offset term μ). Expanding

out, we get

$$\begin{aligned} p(y, w, \tau, \sigma^2 | X) &\propto \sigma^{-N/2} \exp\left(-\frac{1}{2\sigma^2} \|y - Xw\|^2 - \frac{1}{2} |\mathbf{D}\tau|^2\right) \\ &\quad \exp\left(-\frac{1}{2} w^T \mathbf{D} \tau / (\sigma^2) - (a\sigma + 1)\right) \\ &\quad \exp(-b\sigma/\sigma^2) j \exp(-\gamma^2 \tau^2 / 2) \quad (13.88) \end{aligned}$$

Below we describe how to apply the EM algorithm to the model in Figure 13.12.5. In brief, in the E step we infer $\tau^2 j$ and σ^2 , and in the M step we estimate w . The resulting estimate w^* is the same as the lasso estimator. This approach was first proposed in (Figueiredo 2003) (see also (Griffin and Brown 2007; Caron and Doucet 2008; Ding and Harrison 2010) for some extensions).

13.4.4.1 Why EM?

Before going into the details of EM, it is worthwhile asking why we are presenting this approach at all, given that there are a variety of other (often much faster) algorithms that directly solve the L1 MAP estimation problem (see linregFitL1Test for an empirical comparison). The reason is that the latent variable perspective brings several advantages, such as the following:

- It provides an easy way to derive an algorithm to find L1-regularized parameter estimates for a variety of other models, such as robust linear regression (Exercise 11.12) or probit regression (Exercise 13.9).
- It suggests trying other priors on the variances besides $Ga(\tau^2 j | 1, \gamma^2/2)$. We will consider various extensions below.
- It makes it clear how we can compute the full posterior, $p(w|D)$, rather than just a MAP estimate. This technique is known as the Bayesian lasso (Park and Casella 2008; Hans 2009).

13.4.4.2 The objective function

From Equation 13.88, the complete data penalized log likelihood is as follows (dropping terms that do not depend on w)

$$l_c(w) = -\frac{1}{2\sigma^2} \|y - Xw\|^2 - \frac{1}{2} w^T \Lambda w + \text{const} \quad (13.89)$$

where $\Lambda = \text{diag}(\tau^2 j)$ is the precision matrix for w .

13.4.4.3 The E step

The key is to compute $E[\tau^2 j | w]$. We can derive this directly (see Exercise 13.8). Alternatively, we can derive the full posterior, which is given by the following (Park and Casella 2008):

$$p(\tau^2 j | w, D) \sim \text{InverseGaussian}(\gamma^2 w^T j, \gamma^2) \quad (13.90)$$

(Note that the inverse Gaussian distribution is also known as the Wald distribution.) Hence

$$E 1 \tau^2 j | w_j = \gamma |w_j| \quad (13.91)$$

Let $\Lambda = \text{diag}(E 1/\tau^2 1, \dots, E 1/\tau^2 D)$ denote the result of this E step. We also need to infer σ^2 . It is easy to show that that the posterior is

$$p(\sigma^2 | D, w) = IG(a\sigma + (N)/2, b\sigma + 1/2(y - X^T w)^T(y - X^T w)) = IG(aN, bN) \quad (13.92)$$

Hence

$$E 1/\sigma^2 = aN/bN = \omega \quad (13.93)$$

13.4.4.4 The M step

The M step consists of computing

$$\hat{w} = \underset{w}{\operatorname{argmax}} \frac{1}{2} \omega \|y - Xw\|^2 - \frac{1}{2} w^T \Lambda w \quad (13.94)$$

This is just MAP estimation under a Gaussian prior:

$$\hat{w} = (\sigma^2 \Lambda + X^T X)^{-1} X^T y \quad (13.95)$$

However, since we expect many $w_j = 0$, we will have $\tau^2 j = 0$ for many j , making inverting Λ numerically unstable. Fortunately, we can use the SVD of X , given by $X = UDV^T$, as follows:

$$\hat{w} = \Psi V (V^T \Psi V + 1/\omega D - 2)^{-1} D^{-1} U^T y \quad (13.96)$$

where

$$\Psi = \Lambda^{-1} = \text{diag}(1/E 1/\tau^2 j) = \text{diag}(|w_j| \pi(w_j)) \quad (13.97)$$

13.4.4.5 Caveat

Since the lasso objective is convex, this method should always find the global optimum. Unfortunately, this sometimes does not happen, for numerical reasons. In particular, suppose that in the true solution, $w^* j = 0$.

Further, suppose that we set $\hat{w}_j = 0$ in an M step. In the following E step we infer that $\tau^2 j = 0$, so then we set $\hat{w}_j = 0$ again; thus we can never “undo” our mistake.

Fortunately, in practice, this situation seems to be rare.

See (Hunter and Li 2005) for further discussion.

已思考 16s

下面先完整翻译（不漏译），再用通俗解释讲清楚，最后总结要点。

图注翻译

图 13.12 用高斯尺度混合 (Gaussian scale mixture, GSM) 先验来表示 lasso。

译文

13.4.4 用 EM 求解 lasso

本节展示如何用 EM 来求解 lasso 问题。乍一看这有些奇怪，因为 lasso 并没有显式的隐变量。关键洞见是：**拉普拉斯分布可以写成“高斯在方差上的混合”** (GSM; Andrews and Mallows 1974; West 1987)：

$$\text{Lap}(w_j | 0, 1/\gamma) = \frac{\gamma}{2} e^{-\gamma|w_j|} = \int N(w_j | 0, \tau_j^2) \text{Ga}(\tau_j^2 | 1, \gamma^2/2) d\tau_j^2 \quad (13.86)$$

因此，拉普拉斯是一个 GSM，其中对方差的混合分布是指数分布 $\text{Expon}(\tau_j^2 | \gamma^2/2) = \text{Ga}(\tau_j^2 | 1, \gamma^2/2)$ 。利用这一分解，我们就能把 lasso 模型表示为图 13.12 的形式。其联合分布为

$$\begin{aligned} p(y, w, \tau, \sigma^2 | X) &= N(y | Xw, \sigma^2 I_N) N(w | 0, D_\tau) \\ &\times \text{IG}(\sigma^2 | a_\sigma, b_\sigma) \prod_j \text{Ga}(\tau_j^2 | 1, \gamma^2/2) \end{aligned} \quad (13.87)$$

其中 $D_\tau = \text{diag}(\tau_j^2)$ 。为简洁起见，我们假设 X 已标准化且 y 已中心化（因此可忽略截距 μ ）。将其展开：

$$\begin{aligned} p(y, w, \tau, \sigma^2 | X) &\propto (\sigma^2)^{-N/2} \exp\left(-\frac{1}{2\sigma^2} \|y - Xw\|_2^2\right) |D_\tau|^{-1/2} \\ &\times \exp\left(-\frac{1}{2} w^\top D_\tau^{-1} w\right) (\sigma^2)^{-(a_\sigma+1)} e^{-b_\sigma/\sigma^2} \prod_j e^{-(\gamma^2/2)\tau_j^2} \end{aligned} \quad (13.88)$$

下面说明如何把 EM 算法用于图 13.12 所示模型。简言之，**E 步推断 τ_j^2 与 σ^2** ，**M 步估计 w** 。最终得到的 \hat{w} 与 lasso 估计相同。该思路最早由 Figueiredo (2003) 提出（见 Griffin & Brown 2007; Caron & Doucet 2008; Ding & Harrison 2010 的扩展）。

13.4.4.1 为什么用 EM?

在给出 EM 细节之前，值得先问：既然已有许多能**直接**求解 ℓ_1 的（而且通常更快的）算法，为什么还要讲这种方法（可在 `linregFitL1Test` 中作经验比较）？原因是**隐变量视角**带来若干好处，例如：

- 为多种模型推导 ℓ_1 **正则参数估计**提供了简洁途径，如**稳健线性回归**（练习 11.12）或**probit 回归**（练习 13.9）。
- 它提示我们尝试**除 $\text{Ga}(\tau_j^2 | 1, \gamma^2/2)$ 之外**的方差先验；下文将给出一些扩展。
- 它清楚地说明了如何计算**完整后验** $p(w | D)$ ，而不仅是 MAP。这一技术称为**贝叶斯 lasso** (Park & Casella 2008; Hans 2009)。

13.4.4.2 目标函数

由式 (13.88) 得到的**完全数据**带惩罚对数似然 (去除与 w 无关的项) 为

$$\ell_c(w) = -\frac{1}{2\sigma^2} \|y - Xw\|_2^2 - \frac{1}{2} w^\top \Lambda w + \text{const} \quad (13.89)$$

其中 $\Lambda = \text{diag}(1/\tau_j^2)$ 是 w 的精度矩阵。

13.4.4.3 E 步

关键是计算 $E[\frac{1}{\tau_j^2} | w_j, \cdot]$ 。可直接推导 (见练习 13.8)，也可以先写出其全后验 (Park & Casella 2008)：

$$p(\frac{1}{\tau_j^2} | w, D) = \text{InverseGaussian}(\frac{\gamma^2}{w_j^2}, \gamma^2) \quad (13.90)$$

(逆高斯分布又称 Wald 分布。) 因此

$$E[\frac{1}{\tau_j^2} | w_j] = \frac{\gamma}{|w_j|}. \quad (13.91)$$

令 $\Lambda = \text{diag}(E[1/\tau_1^2], \dots, E[1/\tau_D^2])$ 表示本次 E 步的结果。我们还需要推断 σ^2 。容易得到其后验：

$$p(\sigma^2 | D, w) = \text{IG}(a_\sigma + \frac{N}{2}, b_\sigma + \frac{1}{2}(y - X\hat{w})^\top(y - X\hat{w})) = \text{IG}(a_N, b_N) \quad (13.92)$$

因此

$$E[\frac{1}{\sigma^2}] = \frac{a_N}{b_N} = \omega. \quad (13.93)$$

13.4.4.4 M 步

M 步求解

$$\hat{w} = \arg \max_w (-\frac{1}{2}\omega \|y - Xw\|_2^2 - \frac{1}{2} w^\top \Lambda w). \quad (13.94)$$

这正是带高斯先验的 MAP：

$$\hat{w} = (\sigma^2 \Lambda + X^\top X)^{-1} X^\top y. \quad (13.95)$$

然而，我们预期很多 $w_j = 0$ ，对应会有许多 $\tau_j^2 = 0$ ，直接求 Λ^{-1} 可能**数值不稳定**。所幸可用 $X = UDV^\top$ 的 SVD 形式：

$$\hat{w} = \Psi V (V^\top \Psi V + \frac{1}{\omega} D^{-2})^{-1} D^{-1} U^\top y, \quad (13.96)$$

其中

$$\Psi = \Lambda^{-1} = \text{diag}\left(\frac{1}{E[1/\tau_j^2]}\right) = \text{diag}\left(\frac{|w_j|}{\gamma}\right). \quad (13.97)$$

13.4.4.5 警示

由于 lasso 的目标是凸的，理论上该方法应总能找到全局最优解。不幸的是，数值上偶尔会失败。尤其是：若真实解中 $w_j^{*} = 0$ ，并在某次 M 步我们碰巧把 $\hat{w}_j = 0$ ，下一次 E 步就会得到 $\tau_j^2 = 0$ ，从而下次 M 步又会使 $\hat{w}_j = 0$ ；这样就**永远无法“撤销”**这一步。所幸实践中这种情况很少见；详见 Hunter & Li (2005)。

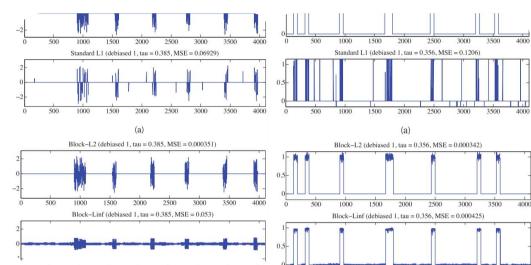
通俗解释

- **核心点：**把 Lasso 的拉普拉斯先验改写成“高斯 + 方差的指数先验”。这样就引入了隐变量 τ_j^2 （每个系数的方差），从而可以用 EM：
 - **E 步：**在当前 w 下，计算每个系数的“惩罚强度”

$$E[1/\tau_j^2 | w_j] = \gamma/|w_j|$$
 系数越小，惩罚越大 \Rightarrow 更倾向于把它推向 0（符合 L1 的“软阈值”味道）。
 同时更新 σ^2 的期望（或其倒数 ω ）。
 - **M 步：**在这些“惩罚强度”固定的情况下，解一个带对角二次惩罚的加权最小二乘（闭式解 13.95 或 SVD 版 13.96），得到新的 w 。
 - 循环往复，收敛后得到的 \hat{w} 与 Lasso 的解一致。
 - **为什么要这样做：**
 1. 这给了一个通用模板——换个似然或换个方差先验，就能推导其他模型（稳健回归、probit 等）的 ℓ_1 风格估计；
 2. 能自然地走向贝叶斯 Lasso：不仅要 MAP，还能近似/抽样整个后验分布 $p(w | D)$ ；
 3. 计算上是交替做“期望的权重”与“加权最小二乘”，实现朴素。
 - **小坑（零锁死）：**一旦某维在 M 步被设为 0，下一次 E 步会给该维无限大的惩罚（ $\tau_j^2 = 0 \Rightarrow 1/\tau_j^2 = +\infty$ ），从而之后都保持 0，难以翻身。实践里较少见，但要注意初始化或可加入微小保护项。
-

要点小结

- **把拉普拉斯写成 GSM:** $\text{Lap}(w) = \int N(w | 0, \tau^2) \text{Ga}(\tau^2 | 1, \gamma^2/2) d\tau^2$ 。
- 因此 lasso 可视作含隐变量 τ_j^2 的层次模型 (图 13.12)。
- **EM 程序:**
 - **E 步:** $E[1/\tau_j^2 | w_j] = \gamma/|w_j|$; σ^2 的后验为 IG, $\omega = E[1/\sigma^2] = a_N/b_N$ 。
 - **M 步:** 解 $\arg \max -\frac{1}{2}\omega \|y - Xw\|^2 - \frac{1}{2}w^\top \Lambda w$, 等价于**加权岭回归型闭式解**; 用 SVD 可避免数值不稳。
- **结果:** 收敛解 w 与标准 lasso **一致**; 该视角易推广、能做贝叶斯推断。
- **风险:** 可能出现“零锁死”，一般较少见，可靠初始化/数值技巧缓解。



翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

Figure 13.13 Illustration of group lasso where the original signal is piecewise Gaussian. Top left: original signal. Bottom left:: vanilla lasso estimate. Top right: group lasso estimate using a 2 norm on the blocks. Bottom right: group lasso estimate using an ∞ norm on the blocks.
Based on Figures 3-4 of (Wright et al. 2009). Figure generated by groupLassoDemo, based on code by Mario Figueiredo.

Figure 13.14 Same as Figure 13.13, except the original signal is piecewise constant.

13.5 |1 regularization: extensions

In this section, we discuss various extensions of “vanilla” |1 regularization.

13.5.1 Group Lasso

In standard |1 regularization, we assume that there is a 1:1 correspondence between parameters and variables, so that if $\hat{w}_j = 0$, we interpret this to mean that variable j is excluded. But in more complex models, there may be many parameters associated with a given variable. In

particular, we may have a vector of weights for each input, w_j . Here are some examples:

- Multinomial logistic regression Each feature is associated with C different weights, one per class.
- Linear regression with categorical inputs Each scalar input is one-hot encoded into a vector of length C .
- Multi-task learning In multi-task learning, we have multiple related prediction problems. For example, we might have C separate regression or binary classification problems. Thus each feature is associated with C different weights. We may want to use a feature for all of the tasks or none of the tasks, and thus select weights at the group level (Obozinski et al. 2007).

If we use an 1 regularizer of the form $\|w\| = j c |w_j|$, we may end up with some elements of w_j being zero and some not. To prevent this kind of situation, we partition the parameter vector into G groups. We now minimize the following objective

$$J(w) = NLL(w) + \sum_{g=1}^G \lambda_g \|w_g\|^2 \quad (13.98)$$

where

$$\|w_g\|^2 = \sum_{j \in g} w_j^2 \quad (13.99)$$

is the 2-norm of the group weight vector. If the NLL is least squares, this method is called group lasso (Yuan and Lin 2006). We often use a larger penalty for larger groups, by setting $\lambda_g = \lambda d_g$, where d_g is the number of elements in group g . For example, if we have groups {1, 2} and {3, 4, 5}, the objective becomes

$$J(w) = NLL(w) + \lambda \sqrt{2(2(w_1^2 + w_2^2)) + 3(2(w_3^2 + w_4^2 + w_5^2))} \quad (13.100)$$

Note that if we had used the square of the 2-norms, the model would become equivalent to ridge regression, since

$$\sum_{g=1}^G \|w_g\|^2 = \sum_{j \in g} w_j^2 = \|w\|^2 \quad (13.101)$$

By using the square root, we are penalizing the radius of a ball containing the group's weight vector: the only way for the radius to be small is if all elements are small. Thus the square root results in group sparsity. A variant of this technique replaces the 2-norm with the infinity-norm (Turlach et al. 2005; Zhao et al. 2005):

$$\|w_g\|_\infty = \max_{j \in g} |w_j| \quad (13.102)$$

It is clear that this will also result in group sparsity.

An illustration of the difference is shown in Figures 13.13 and 13.14. In both cases, we have a true signal w of size D

$=212 = 4096$, divided into 64 groups each of size 64. We randomly choose 8 groups of w and assign them non-zero values. In the first example, the values are drawn from a $N(0, 1)$. In the second example, the values are all set to 1. We then pick a random design matrix X of size $N \times D$, where $N = 210 = 1024$. Finally, we generate $y = Xw + \epsilon$, where $\epsilon \sim N(0, 10^{-4}I_N)$. Given this data, we estimate the support of w using 1 or group 1, and then estimate the non-zero values using least squares. We see that group lasso does a much better job than vanilla lasso, since it respects the known group structure. We also see that the ∞ norm has a tendency to make all the elements within a block to have similar magnitude. This is appropriate in the second example, but not the first. (The value of λ was the same in all examples, and was chosen by hand.)

已思考 11s

下面先把原文完整翻译（不漏译），再用通俗解释说明，最后给要点小结。

译文

图 13.13 展示“组 lasso (group lasso) ”在分段高斯原始信号上的效果。左上：原始信号。左下：标准 L1 (普通 lasso) 的估计。右上：在各块上使用 2 范数的组 lasso 估计。右下：在各块上使用 ∞ 范数的组 lasso 估计。基于 (Wright et al., 2009) 的图 3–4。图由 `groupLassoDemo` 生成 (基础代码来自 Mario Figueiredo) 。

图 13.14 与图 13.13 相同，只是原始信号改为分段常数。

13.5 ℓ_1 正则化：扩展

本节讨论“普通” ℓ_1 正则化的多种扩展。

13.5.1 组 Lasso (Group Lasso)

在标准的 ℓ_1 正则化中，我们假设参数与变量一一对应，因此如果 $w_j = 0$ ，就把它理解为第 j 个变量被排除。但在更复杂的模型里，一个变量可能对应多个参数。特别地，我们可能对每个输入都有一个权重向量 w_j 。例如：

- **多项逻辑回归**：每个特征对应 C 个不同的权重（每类一个）。

- **含类别型输入的线性回归**: 每个标量输入被 one-hot 成长度为 C 的向量。
- **多任务学习**: 存在多个相关的预测问题 (例如 C 个回归或二分类问题)。因此每个特征对应 C 个权重。我们可能希望一个特征**要么在所有任务中都用，要么都不用**，因此在**组**层面进行选择 (Obozinski et al., 2007)。

若使用形如 $\|w\| = \sum_j \sum_c |w_{jc}|$ 的 ℓ_1 正则，可能会出现某个 $w_{j,:}$ 向量里**有的元素为零、有的不为零**的情况。为避免这种情形，我们把参数向量按先验结构划分为 G 个组。现在最小化下面的目标：

$$J(w) = \text{NLL}(w) + \sum_{g=1}^G \lambda_g \|w_g\|_2 \quad (13.98)$$

其中

$$\|w_g\|_2 = \sqrt{\sum_{j \in g} w_j^2} \quad (13.99)$$

是组权重向量的 2-范数。若 NLL 是最小二乘，这个方法称为**组 lasso** (Yuan & Lin, 2006)。我们常按组大小增大惩罚，设 $\lambda_g = \lambda \sqrt{d_g}$ ，其中 d_g 是第 g 组的元素数。例如，若分组是 {1, 2} 与 {3, 4, 5}，则目标变为

$$J(w) = \text{NLL}(w) + \lambda \frac{\sqrt{2}}{2} (w_1^2 + w_2^2)^{1/2} + \lambda \frac{\sqrt{3}}{2} (w_3^2 + w_4^2 + w_5^2)^{1/2} \quad (13.100)$$

注意：如果把 2-范数**平方**来惩罚，模型就等价为**岭回归**，因为

$$\sum_{g=1}^G \|w_g\|_2^2 = \sum_g \sum_{j \in g} w_j^2 = \|w\|_2^2. \quad (13.101)$$

使用**平方根等于惩罚一“球”的半径**：要让半径小，只能让组内**所有元素都小**；因此**平方根会产生组稀疏**。一种变体把 2-范数换成 ∞ 范数 (Turlach et al., 2005; Zhao et al., 2005)：

$$\|w_g\|_\infty = \max_{j \in g} |w_j|. \quad (13.102)$$

这也会导致组稀疏。

图 13.13 和 13.14 展示了二者差异。两种情况下，真实信号 w 的维度 $D = 2^{12} = 4096$ ，被分成 64 个、每个大小为 64 的组。我们随机选 8 个组令其非零。第一种例子中，组内值从 $N(0, 1)$ 采样；第二种例子中，组内值都设为 1。随后生成一个大小为 $N \times D$ 的随机设计矩阵 X ，其中 $N = 2^{10} = 1024$ 。最后生成 $y = Xw + \varepsilon$, $\varepsilon \sim N(0, 10^{-4} I_N)$ 。给定这些数据，我们用**L1** 或**组 L1** 来估计 w 的**支持集** (哪些位置非零)，然后对估计出的非零位

置再用最小二乘估计其数值。可以看到，组 lasso 比普通 lasso 好得多，因为它遵从了已知的**组结构**。还可以看到， ∞ 范数倾向把同一块里的元素压到**相近的幅度**。这在第二个例子（组内值相同）是合适的，而在第一个例子（组内是高斯、幅度差异大）则不合适。（四个小图中的 λ 取值相同，靠手动选择。）

通俗解释

- **为什么需要“组”版本？**

许多模型里“一个特征 = 多个参数”（多类、多任务、one-hot 等）。我们想做到**整组一起选/一起丢**，而不是一组里零零散散有的留有的删。普通 lasso（逐元素的 L1）会**碎片化**：同一组里有的被置零，有的没置零；组 lasso 通过**对组向量的长度**罚款，逼迫**整组同时变零** → 组稀疏。

- **怎么罚？**

换掉逐元素 $|w_{jc}|$ 的和，改为**每组的 2-范数之和**： $\sum_g \lambda_g \|w_g\|_2$ 。

- 用**平方根**（而不是平方）很关键：平方会退化为普通的 L2（岭），不会产生稀疏；平方根才会“要么整组为 0，要么保留”的效果。
- 组大小不同，常用 $\lambda_g = \lambda \sqrt{d_g}$ 做**公平惩罚**（组越大， λ_g 越大）。

- **2 范数 vs ∞ 范数**

两者都能让“整组为零/非零”。但

- **组-L2**：允许组内元素大小不同；
- **组-L ∞** ：更倾向把组内元素推向**相近幅度**（“齐刷刷”）。

所以：当真实组内值**差不多**（如分段常数）→ 组-L ∞ 更贴切；当组内值**差异大**（如分段高斯）→ 组-L2 更合适。

- **图里的实验在说明什么？**

在已知“非零按组成片出现”的前提下：

- 普通 lasso 会**漏检/误检**（零散的尖刺很多，MSE 大）；
 - 组 lasso 能**稳稳找回整段非零**（MSE 小）；
 - 用 L ∞ 会把每段内部的振幅拉得相似——这在“分段常数”时几乎完美，在“分段高斯”时会显得过度平滑。
-

要点小结

- **组 lasso 目标：** $J(w) = \text{NLL}(w) + \sum_g \lambda_g \|w_g\|_2$ （或 $\|\cdot\|_\infty$ 变体）。

- 作用：**实现**组层面的稀疏**（整组选/整组丢），适用于多类、多任务、one-hot 等“一个特征→多个参数”的情形。
- 为何不用** $\|w_g\|_2^2$ ：平方会变成岭回归（不稀疏）；平方根才会产生组稀疏。
- 组大小校正**：常用 $\lambda_g = \lambda \sqrt{d_g}$ 。
- L2 vs L ∞ (组内) :**
 - 组-L2：组内幅度可不同，适合“组内高斯/差异大”；
 - 组-L ∞ ：组内幅度趋同，适合“组内常数/差异小”。
- 经验现象：**已知存在组结构时，组 lasso 比普通 lasso **更准确、更稳定**；选好范数能匹配数据的真实结构。

这一小节主要在讲什么？我不太懂

已思考 5s

这小节讲的是**组 Lasso (group lasso)**：当“一个特征对应多个参数”时，怎么一次性**整组选/整组丢**，而不是像普通 Lasso 那样把同一组里的参数剪得零零散散。

为什么需要“组”的稀疏？

很多模型里，一个输入特征会变成**一串参数**，例如：

- 多分类逻辑回归：每个特征对每个类别各有一个权重（共 C 个）。
- 类别型变量做 one-hot：一个原始变量展开成 C 个虚拟变量。
- 多任务学习：同一特征在 C 个任务上各有一个权重。

这时我们常希望：**要么这组参数全用，要么全不用**。普通 L1（逐元素惩罚 $|w_{jcl}|$ ）会导致一组里有的为 0、有的不为 0，决策不干净。

怎么做？

把参数按先验结构分成 G 组 w_g ，最小化

$$J(w) = \text{损失} + \sum_{g=1}^G \lambda_g \|w_g\|_2$$

- 这里对**组向量的 2 范数求和**（注意是**平方根**，不是平方），会产生**组稀疏**：要让半径小，组内每个分量都得小，整组就容易变成 0。
- 组大小不同时，常用 $\lambda_g = \lambda \sqrt{d_g}$ (d_g 为组内元素数) 做公平惩罚。

- 还有一个变体：把组内的 $\|\cdot\|_2$ 换成 $\|\cdot\|_\infty$ （取组内最大绝对值）。它同样鼓励组稀疏，并且倾向把组内数值拉得差不多。

注意：如果用的是 $\sum_g \|w_g\|_2^2$ （平方），就退化成岭回归，不会稀疏。

图 13.13 & 13.14 想说明什么？

- 人为构造一个长度 $D = 4096$ 的真权重 w ，分成 64 组（每组 64 个），只有 8 组是非零。再随机取设计矩阵 X ($N = 1024$)，加很小噪声生成 y 。
- 普通 L1 (vanilla lasso)**：找回来的非零位置很零散，误检/漏检多，均方误差 (MSE) 较大。
- 组 Lasso (组-L2)**：基本能找回整块非零组，MSE 很小。
- 组-L ∞** ：会把同一块里数值拉得更一致。当真实组内是常数（图 13.14）时更贴切；若组内是高斯起伏（图 13.13），反而显得过度平滑。

什么时候用？

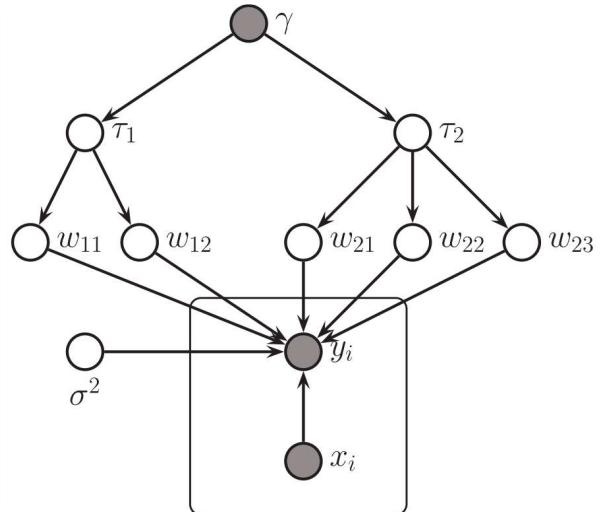
- 多类/多任务/one-hot 等“一个特征 \rightarrow 多个参数”的场景；
- 已知/怀疑存在成组出现的相关性或先验结构；
- 想要解释性强：整组选/整组丢更好讲清楚。

小对比

- 普通 Lasso (逐元素 L1)**：产生元素级稀疏，组内可能参差不齐；
- 组 Lasso ($\sum \|w_g\|_2$)**：产生组级稀疏，整组 0 或整组非 0；
- 组-L ∞** ：同样组稀疏，且组内幅度更趋同；
- 岭回归**： $\|w\|_2^2$ 惩罚，不稀疏，只做平滑收缩。

要点小结

- 组 Lasso 通过 $\sum_g \lambda_g \|w_g\|_2$ （或 $\|\cdot\|_\infty$ ）实现整组级别的特征选择。
- 用平方根 ($\|\cdot\|_2$) 而不是平方，才能产生稀疏；平方会变岭回归。
- 组大小用 $\lambda_g = \lambda \sqrt{d_g}$ 校正更公平。
- 组-L2 适合组内幅度差异大的情形；组-L ∞ 适合组内“应当差不多”的情形（如分段常数）。
- 实验图展示：当真实结构是“按组非零”时，组 Lasso 明显优于普通 L1。



翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

Figure 13.15 Graphical model for group lasso with 2 groups, the first has size G1 = 2, the second has size G2 = 3.

13.5.1.1 GSM interpretation of group lasso

Group lasso is equivalent to MAP estimation using the following prior

$$p(w|\gamma, \sigma^2) \propto \exp - \gamma \sum_g \|w_g\|^2 \quad (13.103)$$

Now one can show (Exercise 13.10) that this prior can be written as a GSM, as follows:

$$w_g | \sigma^2, \tau_g \sim N(0, \sigma^2 \tau_g / d_g) \quad (13.104)$$

$$\tau_g | \gamma \sim Ga(d_g + 1/2, \gamma^2) \quad (13.105)$$

where d_g is the size of group g . So we see that there is one variance term per group, each of which comes from a Gamma prior, whose shape parameter depends on the group size, and whose rate parameter is controlled by γ . Figure 13.15 gives an example, where we have 2 groups, one of size 2 and one of size 3.

This picture also makes it clearer why there should be a grouping effect. Suppose $w_{1,1}$ is small; then $\tau_{2,1}$ will be estimated to be small, which will force $w_{1,2}$ to be small. Conversely, suppose $w_{1,1}$ is large; then $\tau_{2,1}$ will be estimated to be large, which will allow $w_{1,2}$ to become large as well.

13.5.1.2 Algorithms for group lasso

There are a variety of algorithms for group lasso. Here we briefly mention two. The first approach is based on

proximal gradient descent, discussed in Section 13.4.3.

Since the regularizer is separable, $R(w) = g ||wg||_p$, the proximal operator decomposes into G separate operators of the form

$$\text{prox}_R(b) = \operatorname{argmin}_{z \in RDg} \|z - b\|_2^2 + \lambda \|z\|_p \quad (13.106)$$

where $b = \theta kg - tkgkg$. If $p = 2$, one can show (Combettes and Wajs 2005) that this can be implemented as follows

$$\text{prox}_R(b) = b - \text{proj}_{\lambda C}(b) \quad (13.107)$$

where $C = \{z : \|z\|_2 \leq 1\}$ is the 2 ball. Using Equation 13.74, if $\|b\|_2 < \lambda$, we have

$$\text{prox}_R(b) = b - b = 0 \quad (13.108)$$

otherwise we have

$$\text{prox}_R(b) = b - \lambda b \|b\|_2 = b \|b\|_2 - \lambda \|b\|_2 \quad (13.109)$$

We can combine these into a vectorial soft-threshold function as follows (Wright et al. 2009):

$$\text{prox}_R(b) = b \max(\|b\|_2 - \lambda, 0) \max(\|b\|_2 - \lambda, 0) + \lambda \quad (13.110)$$

If $p = \infty$, we use $C = \{z : \|z\|_1 \leq 1\}$, which is the 1 ball. We can project onto this in $O(dg)$ time using an algorithm described in (Duchi et al. 2008).

Another approach is to modify the EM algorithm. The method is almost the same as for vanilla lasso. If we define $\tau_{2,j} = \tau_{2,g(j)}$, where $g(j)$ is the group to which dimension j belongs, we can use the same full conditionals for σ^2 and w as before. The only changes are as follows:

- We must modify the full conditional for the weight precisions, which are estimated based on a shared set of weights:

$$1 \tau_{2,g} | \gamma, w, \sigma^2, y, X \sim \text{InverseGaussian}(\gamma, \sigma^2 \|wg\|_2^2, \gamma^2) \quad (13.111)$$

where $\|wg\|_2^2 = j \in g w_j^2$. For the E step, we can use $E 1 \tau_{2,g} = \gamma \sigma \|wg\|_2$ (13.112)

- We must modify the full conditional for the tuning parameter, which is now only estimated based on G values of $\tau_{2,g}$:

$$p(\gamma^2 | \tau) = \text{Ga}(\alpha\gamma + G/2, \beta\gamma + 1/2 G \tau_{2,g}) \quad (13.113)$$

已思考 22s

下面先把原文完整翻译（含公式），再用通俗解释说明，最后给要点小结。

译文

图 13.15 具有两个组 (第一组大小 $G_1 = 2$, 第二组大小 $G_2 = 3$) 的组 lasso 图模型。

13.5.1.1 组 lasso 的 GSM (高斯尺度混合) 解释

组 lasso 等价于在如下先验下做 MAP 估计:

$$p(w \mid \gamma, \sigma^2) \propto \exp\left(-\frac{\gamma}{\sigma} \sum_{g=1}^G \|w_g\|_2\right) \quad (13.103)$$

可以证明 (练习 13.10) 这个先验可以写成一个 GSM:

$$w_g \mid \sigma^2, \tau_g^2 \sim N(0, \sigma^2 \tau_g^2 I_{d_g}) \quad (13.104)$$

$$\tau_g^2 \mid \gamma \sim Ga\left(\frac{d_g + 1}{2}, \frac{\gamma^2}{2}\right) \quad (13.105)$$

其中 d_g 是第 g 组的大小。也就是说每个组都有一个方差参数 τ_g^2 , 它来自一个 Gamma 先验; 其形状参数依赖组大小, 速率由 γ 控制。图 13.15 给出一个例子: 我们有两个组, 大小分别为 2 和 3。

这个图也更清楚地说明了为什么会产生**组效应**: 如果 $w_{1,1}$ 很小, 那么会把 τ_1^2 估得很小, 从而也会把同组的 $w_{1,2}$ 压小; 反过来, 如果 $w_{1,1}$ 很大, 那么 τ_1^2 会被估得很大, 从而允许 $w_{1,2}$ 也变大。

13.5.1.2 组 lasso 的算法

组 lasso 有多种算法。这里简要提两类。

(一) 近端梯度法 (见 13.4.3)

正则项可分拆: $R(w) = \sum_g \|w_g\|_p$ 。于是近端算子可分解为 G 个小问题:

$$\text{prox}_R(b) = \arg \min_{z \in \mathbb{R}^{d_g}} \|z - b\|_2^2 + \lambda \|z\|_p \quad (13.106)$$

其中 $b = \theta_k^{(g)} - t_k g_k^{(g)}$ 。

- 当 $p = 2$ 时 (组- L_2) , 可得 (Combettes & Wajs, 2005) :

$$\text{prox}_R(b) = b - \text{proj}_{\lambda C}(b), \quad C = \{z : \|z\|_2 \leq 1\} \quad (13.107)$$

利用式 (13.74):

$$\|b\|_2 < \lambda \Rightarrow \text{prox}_R(b) = b - b = 0 \quad (13.108)$$

否则

$$\text{prox}_R(b) = b - \lambda \frac{b}{\|b\|_2} = \frac{\|b\|_2 - \lambda}{\|b\|_2} b \quad (13.109)$$

把两种情况合并成**向量软阈值** (Wright et al., 2009) :

$$\text{prox}_R(b) = b - \frac{\max(\|b\|_2 - \lambda, 0)}{\max(\|b\|_2 - \lambda, 0) + \lambda} b \quad (13.110)$$

- 当 $p = \infty$ 时 (组- L_∞) , $C = \{z : \|z\|_1 \leq 1\}$ 是 ℓ_1 球。可用 Duchi 等 (2008) 的线性时间 $O(d_g)$ 投影算法来实现。

(二) EM 改造法

把 vanilla lasso 的 EM 程序稍作修改即可。若令 $\tau_j^2 = \tau_{g(j)}^2$ ($g(j)$ 为第 j 维所属的组) , σ^2 与 w 的条件分布与之前相同, 仅有两处改变:

- 权重精度 (E 步)** : 同一组共用一个精度, 满条件为

$$\frac{1}{\tau_g^2} \mid \gamma, w, \sigma^2, y, X \sim \text{InverseGaussian}\left(\frac{\gamma^2 \sigma^2}{\|w_g\|_2^2}, \gamma^2\right) \quad (13.111)$$

因此在 E 步可用

$$\mathbb{E}\left[\frac{1}{\tau_g^2}\right] = \frac{\gamma \sigma}{\|w_g\|_2}. \quad (13.112)$$

- 调参 γ 的后验** (只基于 G 个 τ_g^2) :

$$p(\gamma^2 \mid \tau) = \text{Ga}(a_\gamma + \frac{G}{2}, b_\gamma + \frac{1}{2} \sum_g \tau_g^2). \quad (13.113)$$

通俗解释

- 组 lasso 的本质**: 不再逐元素 $|w_j|$ 罚, 而是**按组罚“组向量的长度”** $\|w_g\|_2$ 。这样就会出现“**整组一起为 0 或一起非 0**”的效果 (组稀疏)。
- 为什么 GSM 有助于理解?**
 - 把“对 $\|w_g\|_2$ 的惩罚”改写成: 给每组一个**方差** τ_g^2 , 在该方差下 w_g 服从**零均值高斯**; 再给 τ_g^2 一个 Gamma 先验。
 - 如果组里某个分量很小, 会把 τ_g^2 估得小 \Rightarrow 整组都被**更强地收缩**;
 - 若组里已出现大系数, 会把 τ_g^2 估得大 \Rightarrow **放松**, 同组其他分量也更容易变大。这就是**组效应**为何自然产生。
- 怎么求解?**
 - 近端梯度**: 一步梯度 + 一步“组软阈值”。当组范数是 L_2 时, 近端就是**把组向量按长度整体缩放或置零**; 当是 L_∞ 时, 需要**投影到 ℓ_1 球** (有线性时间算法)。

2. **EM 改造**: 把每组的方差当隐变量推断; E 步给出**组级惩罚强度** $E[1/\tau_g^2] = \gamma\sigma/\|w_g\|_2$, M 步做一次加权的**最小二乘/MAP**。 γ 也可在贝叶斯框架下更新。

要点小结

- 组 lasso 的先验: $p(w) \propto \exp(-(\gamma/\sigma)\sum_g \|w_g\|_2)$, 可写成 **GSM**:
 $w_g \mid \tau_g^2 \sim N(0, \sigma^2 \tau_g^2 I)$, $\tau_g^2 \mid \gamma \sim Ga((d_g + 1)/2, \gamma^2/2)$ 。
- **组效应**: 同组间通过共享 τ_g^2 彼此牵引, 促成**整组稀疏**。
- **算法**:
 - **近端梯度**: 组- L_2 的近端 = **向量软阈值/整体缩放**; 组- L_∞ 需投影到 ℓ_1 球 (线性时间)。
 - **EM 版**: E 步更新组精度 $E[1/\tau_g^2] = \gamma\sigma/\|w_g\|_2$, 并更新 γ 的后验; M 步解加权最小二乘。
- 适用: 当“一个特征对应多个参数” (多类、多任务、one-hot 等) 且希望**整组选/整组丢**时, 组 lasso 比普通 L1 更贴切。
- 选择 L_2 还是 L_∞ :
 - L_2 : 允许组内大小不同;
 - L_∞ : 倾向组内幅度相近。

这一小节主要在讲什么? 我不太懂

已思考 10s

这小节干了两件事: (1) 用“高斯尺度混合 (GSM) /层次贝叶斯”的视角解释 Group Lasso 为什么会“整组选/整组丢”; (2) 给出两类常用求解算法 (近端梯度 & EM 变体)。配图 13.15 就是把“每一组共享一个方差”的层次模型画出来。

1) Group Lasso 的 GSM/贝叶斯解释 (图 13.15)

- Group Lasso 的惩罚是 $\sum_g \|w_g\|_2$ 。把它看成先验:

$$p(w \mid \gamma, \sigma^2) \propto \exp\left(-\frac{\gamma}{\sigma} \sum_g \|w_g\|_2\right)$$

- 这等价于**每组一个“方差”隐变量** τ_g^2 的层次模型 (高斯尺度混合, GSM) :

$$w_g \mid \sigma^2, \tau_g^2 \sim N(0, \sigma^2 \tau_g^2 I), \quad \tau_g^2 \mid \gamma \sim Ga\left(\frac{d_g+1}{2}, \frac{\gamma^2}{2}\right)$$

其中 d_g 是组大小。

- **直观：**同组参数共用 τ_g^2 。若组里某个权重很小，推断得到 τ_g^2 也小 \Rightarrow 整组收缩更强（都被压小，易变 0）；反之，有大权重时 τ_g^2 大 \Rightarrow 整组放松，同组其它权重也更容易变大。于是自然产生**组稀疏**（整组选/整组丢）。
-

2) 两类求解算法

A. 近端梯度 (Proximal Gradient)

- 目标：损失 $+ \sum_g \lambda \|w_g\|_p$ （通常 $p = 2$ 或 $p = \infty$ ）。
- 每次迭代对每组做一个小子问题（近端算子）：

$$\text{prox}(b_g) = \arg \min_z \|z - b_g\|_2^2 + \lambda \|z\|_p$$

其中 $b_g = \theta_k^{(g)} - t_k \nabla_{\theta_k} L(\theta_k)$ 。

- **组- L_2** （最常用）的近端是“**向量软阈值/整体缩放**”：
 - 若 $\|b_g\|_2 \leq \lambda \Rightarrow$ 直接 $w_g = 0$ （整组置零）；
 - 否则 $w_g = (1 - \lambda/\|b_g\|_2) b_g$ （整组按同一比例缩小）。
- **组- L_∞** ：需要把 b_g 投影到 ℓ_1 球（有线性时间算法），也能得到组稀疏，且有把组内幅度拉得更相近的倾向。

B. EM 变体 (层次贝叶斯做 MAP)

- 把 τ_g^2 当隐变量交替更新：
 - **E 步：**用当前 w_g 计算组级“惩罚强度”

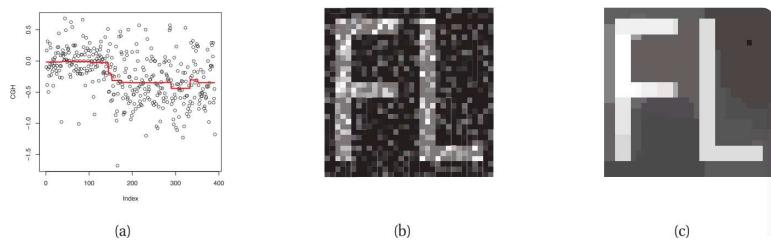
$$E[1/\tau_g^2] = \gamma \sigma / \|w_g\|_2$$
 （组越小 \Rightarrow 收缩越强）。
 - **M 步：**在这些强度下解一次**加权最小二乘/岭式**更新得到新 w 。
 同时可更新 γ 的后验（仅依赖各组 τ_g^2 ）。
 - 好处：思路统一、易扩展到别的模型；注意要防止数值上“误置零后翻不了身”的极端情况。
-

什么时候用哪个？

- 想要**大规模、稳定、易实现**：用**近端梯度（组向量软阈值）**；
- 已在做贝叶斯/层次建模，或要顺带估计不确定性：可用**EM 变体**。

要点小结

- 本节核心：**把 Group Lasso 看成“**每组共享方差**”的层次高斯 (GSM) 模型 \Rightarrow 解释了为何会产生**组稀疏**。
- 算法：**
 - 近端梯度：组- L_2 的近端 = **整组缩放或置零**；组- L_∞ 需投影到 ℓ_1 球。
 - EM 版：E 步给出组级精度，M 步解加权最小二乘，还可贝叶斯更新 γ 。
- 直观记忆：**一个组像拴在同一根“弹簧”上（同一个 τ_g^2 ）；弹簧紧时全体被拉向 0，松时全体都能长大——这就是 Group Lasso。



翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

Figure 13.16 (a) Example of the fused lasso. The vertical axis represents array CGH (chromosomal genome hybridization) intensity, and the horizontal axis represents location along a genome. Source: Figure 1 of (Hoefling 2010). (b) Noisy image. (c) Fused lasso estimate using 2d lattice prior. Source: Figure 2 of (Hoefling 2010). Used with kind permission of Holger Hoefling

13.5.2 Fused lasso

In some problem settings (e.g., functional data analysis), we want neighboring coefficients to be similar to each other, in addition to being sparse. An example is given in Figure 13.16(a), where we want to fit a signal that is mostly “off”, but in addition has the property that neighboring locations are typically similar in value. We can model this by using a prior of the form

$$p(w|\sigma^2) \propto \exp(-\lambda_1 \sum_{j=1}^D |w_j| - \lambda_2 \sum_{j=1}^{D-1} |w_j + w_{j+1}|) \quad (13.114)$$

This is known as the fused lasso penalty. In the context of functional data analysis, we often use $X = I$, so there is one

coefficient for each location in the signal (see Section 4.4.2.3). In this case, the overall objective has the form

$$J(w, \lambda_1, \lambda_2) = \sum_{i=1}^N (y_i - w_i)^2 + \lambda_1 \sum_{i=1}^N |w_i| + \lambda_2 \sum_{i=1}^{N-1} |w_{i+1} - w_i| \quad (13.115)$$

This is a sparse version of Equation 4.148.

It is possible to generalize this idea beyond chains, and to consider other graph structures, using a penalty of the form

$$J(w, \lambda_1, \lambda_2) = \sum_{s \in V} (y_s - w_s)^2 + \lambda_1 \sum_{s \in V} |w_s| + \lambda_2 \sum_{(s,t) \in E} |w_t - w_s| \quad (13.116)$$

This is called graph-guided fused lasso (see e.g., (Chen et al. 2010)). The graph might come from some prior knowledge, e.g., from a database of known biological pathways. Another example is shown in Figure 13.16(b-c), where the graph structure is a 2d lattice.

13.5.2.1 GSM interpretation of fused lasso

One can show (Kyung et al. 2010) that the fused lasso model is equivalent to the following hierarchical model

$$w | \sigma^2, \tau, \omega \sim N(0, \sigma^2 \Sigma(\tau, \omega)) \quad (13.117)$$

$$\tau_{2j} | \gamma_1 \sim \text{Expon}(\gamma_1) \quad j = 1: D \quad (13.118)$$

$$\omega_{2j} | \gamma_2 \sim \text{Expon}(\gamma_2) \quad j = 1: D - 1 \quad (13.119)$$

where $\Sigma = \Omega^{-1}$, and Ω is a tridiagonal precision matrix with

$$\text{main diagonal} = \{ 1 \tau_{2j} + 1 \omega_{2j-1} + 1 \omega_{2j} \} \quad (13.120)$$

$$\text{off diagonal} = \{ -1 \omega_{2j} \} \quad (13.121)$$

where we have defined $\omega_{-2} = \omega_{-D} = 0$. This is very similar to the model in Section 4.4.2.3, where we used a chain-structured Gaussian Markov random field as the prior, with fixed variance. Here we just let the variance be random. In the case of graph-guided lasso, the structure of the graph is reflected in the zero pattern of the Gaussian precision matrix (see Section 19.4.4).

13.5.2.2 Algorithms for fused lasso

It is possible to generalize the EM algorithm to fit the fused lasso model, by exploiting the Markov structure of the Gaussian prior for efficiency. Direct solvers (which don't use the latent variable trick) can also be derived (see e.g., (Hoefling 2010)). However, this model is undeniably more expensive to fit than the other variants we have considered.

下面先把原文完整翻译（含公式不漏译），再用通俗解释说明，最后给要点小结。

图注翻译

图 13.16

- (a) 融合 Lasso (fused lasso) 的一个例子。纵轴表示阵列 CGH (染色体基因组杂交) 的强度，横轴表示沿基因组的位置。来源：Hoefling (2010) 图 1。
- (b) 含噪声的图像。 (c) 采用二维网格先验的 fused lasso 估计结果。来源：Hoefling (2010) 图 2。经 Holger Hoefling 授权使用。

译文

13.5.2 融合 Lasso (Fused lasso)

在某些问题设置中（例如函数型数据分析），我们除了希望系数是稀疏的，还希望**相邻系数彼此相近**。图 13.16(a) 给出了一个例子：我们想拟合一个大多时候“关闭”（接近 0）、并且相邻位置通常数值相近的信号。我们可以用下面形式的先验建模：

$$p(w \mid \sigma^2) \propto \exp\left(-\frac{\lambda_1}{\sigma} \sum_{j=1}^D |w_j| - \frac{\lambda_2}{\sigma} \sum_{j=1}^{D-1} |w_{j+1} - w_j|\right) \quad (13.114)$$

这被称为**融合 lasso 惩罚**。在函数型数据分析的语境下，我们常取 $X = I$ ，于是信号中每个位置都有一个系数（见 4.4.2.3 节）。此时总体目标为

$$J(w, \lambda_1, \lambda_2) = \sum_{i=1}^N (y_i - w_i)^2 + \lambda_1 \sum_{i=1}^N |w_i| + \lambda_2 \sum_{i=1}^{N-1} |w_{i+1} - w_i| \quad (13.115)$$

这是式 4.148 的**稀疏版**。

这个想法可以从链结构推广到其他图结构，用下面形式的惩罚：

$$J(w, \lambda_1, \lambda_2) = \sum_{s \in V} (y_s - w_s)^2 + \lambda_1 \sum_{s \in V} |w_s| + \lambda_2 \sum_{(s,t) \in E} |w_s - w_t| \quad (13.116)$$

这称为**图引导的融合 lasso** (graph-guided fused lasso, 见 Chen et al., 2010)。图可以来自一些先验知识，例如生物通路数据库。另一个例子见图 13.16(b–c)，其中图结构是二维点阵。

13.5.2.1 融合 lasso 的 GSM 解释

可以证明 (Kyung et al., 2010)，融合 lasso 模型等价于如下层次模型：

$$w \mid \sigma^2, \tau, \omega \sim N(0, \sigma^2 \Sigma(\tau, \omega)) \quad (13.117)$$

$$\tau_j^2 \mid \gamma_1 \sim \text{Expon}\left(\frac{\gamma_1^2}{2}\right), \quad j = 1:D \quad (13.118)$$

$$\omega_j^2 \mid \gamma_2 \sim \text{Expon}\left(\frac{\gamma_2^2}{2}\right), \quad j = 1:D-1 \quad (13.119)$$

其中 $\Sigma = \Omega^{-1}$, Ω 为三对角精度矩阵, 其

$$\text{主对角} = \left\{ \frac{1}{\tau_j^2} + \frac{1}{\omega_{j-1}^2} + \frac{1}{\omega_j^2} \right\} \quad (13.120)$$

$$\text{副对角} = \left\{ -\frac{1}{\omega_j^2} \right\} \quad (13.121)$$

并约定 $\omega_0^{-2} = \omega_D^{-2} = 0$ 。这与 4.4.2.3 节中作为先验所用的**链式高斯马尔可夫随机场**非常相似, 只是这里把方差设为随机变量。在图引导的 lasso 情况下, 图结构会体现在高斯精度矩阵的**零模式**上 (见 19.4.4 节)。

13.5.2.2 融合 lasso 的算法

可以把 EM 算法推广到融合 lasso, 利用高斯先验的马尔可夫结构实现高效计算。也可以推导**不使用隐变量技巧**的直接求解器 (例如 Hoefling, 2010)。不过, 与前述其他变体相比, 这个模型训练代价**无疑更高**。

通俗解释

- 要解决的问题:** 很多信号 (如染色体拷贝数、温度曲线、图像像素) 既**稀疏** (大部分接近 0), 又**分段平滑/常数** (邻近位置应该相差不大, 只有在“拐点/边缘”处突然变化)。
- 做法:** 在普通 L1 (让系数稀疏) 的基础上, 再加一个**相邻差分的 L1 惩罚** (也叫**总变差 TV 惩罚**)。
 - 第一项 $\sum |w_j|$: 把很多系数推向 0;
 - 第二项 $\sum |w_{j+1} - w_j|$: 鼓励邻居相等 \Rightarrow 产生“**阶梯状/分段常数**”的解 (图 13.16(a) 红色折线)。
 - 在图像上, 这个惩罚会**去噪同时保边**: 图 13.16(b) 的噪声字母 “FL” 被 2D 融合 lasso 处理后 (c) 变得干净, 边缘清晰。
- 为什么能这么看 (GSM 解释):** 把两种 L1 惩罚都等价成“**高斯+随机方差**”的层次模型:
 - τ_j 控制单点的收缩强度 (对应 $|w_j|$ 惩罚);

- ω_j 控制相邻点之间的耦合强度（对应 $|w_{j+1} - w_j|$ 惩罚）。这样 Ω 变成三对角（或一般图上的稀疏）精度矩阵 \Rightarrow 邻域相连，远处独立。方差是随机的 \Rightarrow 自适应平滑：平坦区域强烈平滑，边缘处允许突变。
 - 算法层面：
 - 可用 EM（把 τ, ω 当隐变量交替更新）；
 - 也可用直接法/路径算法等。不过整体而言，比普通/组 lasso 更耗时（因为涉及差分耦合，求解线性系统更难）。
-

要点小结

- 融合 lasso = 稀疏 (L1) + 邻接差分的 L1 (TV) \Rightarrow 解通常分段常数、保边去噪。
- 链→图的推广：把一维邻接改成任意图 (V, E) 的邻接，适用于图像 (2D 点阵)、生物网络等。
- GSM/层次贝叶斯： $|w_j|$ 和 $|w_{j+1} - w_j|$ 的 L1 惩罚 \leftrightarrow 高斯先验 + 指数分布的随机方差 \Rightarrow 形成稀疏（或图稀疏）的精度矩阵。
- 应用：基因拷贝数分段检测、时间序列分段、图像去噪/分割等。
- 代价：训练比普通/组 lasso 更重，常用 EM 或专门的直接求解器来提速。

这一小节主要在讲什么？我不太懂

这小节讲的是**“融合 Lasso (fused lasso) ”：在做稀疏估计时，同时鼓励相邻位置的系数彼此接近**，从而得到分段常数/台阶状的结果。它适合那种“多数为0或很小、且在空间/时间上连续成片”的信号，比如基因组拷贝数曲线、图像去噪中的大块区域等。

核心思想（一句话）

在普通 Lasso 的 $\sum |w_j|$ 稀疏惩罚外，再加一个“相邻差分”的 $\sum |w_{j+1} - w_j|$ 惩罚（也叫总变差 TV）。

- 第一项让很多系数变 0（稀疏）；
- 第二项让相邻位置尽量相等（平滑成片，只在少数断点处跳变）。

目标（以一维、 $X = I$ 为例）：

$$\min_w \sum_i (y_i - w_i)^2 + \lambda_1 \sum_i |w_i| + \lambda_2 \sum_i |w_{i+1} - w_i|$$

这样得到的 w 往往是“**大块常值 + 少量跳变**”，既去噪又保边（图像里边缘被保留，噪声被抑制）。

可推广到图结构

不仅限于线性链（1D），还能在任意图 (V, E) 上做：

$\sum_{(s,t) \in E} |w_s - w_t|$ 让相邻节点（比如 2D 网格的像素）值接近，用于**图像或生物网络等**。

另一种视角（层次贝叶斯/GSM）

这套惩罚等价于：给 w 一个**高斯先验**，但其精度矩阵是**三对角/图稀疏**，并带**随机方差**（指数先验）。

直观上：平坦区域对应强平滑，边缘处允许跳变 \Rightarrow **自适应地保留突变**。

算法

可用**EM**（把那些方差当隐变量交替更新），也有**直接求解器**。但总体比普通/组 Lasso **更耗时**，因为变量之间通过差分项耦合了。

你需要记住的要点

- 目的：同时实现**稀疏**（值多为0）+**分段平滑/常数**（邻接相近）。
- 形式： $L1(|w|) + TV(|\Delta w|)$ 双惩罚。
- 场景：基因组拷贝数、时间序列分段、图像去噪/分割（2D 网格）。
- 代价：效果好、可保边，但优化**更贵**；常用 EM 或专门的 TV/路径求解方法。

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

13.5.3 Elastic net (ridge and lasso combined)

Although lasso has proved to be effective as a variable selection technique, it has several problems (Zou and Hastie 2005), such as the following:

- If there is a group of variables that are highly correlated (e.g., genes that are in the same pathway), then the lasso tends to select only one of them, chosen rather arbitrarily. (This is evident from the LARS algorithm: once one member of the group has been chosen, the remaining members of the group will not be very correlated with the new residual and hence will not be chosen.) It is usually better to select all the relevant variables in a group. If we

know the grouping structure, we can use group lasso, but often we don't know the grouping structure.

- In the $D > N$ case, lasso can select at most N variables before it saturates.

- If $N > D$, but the variables are correlated, it has been empirically observed that the prediction performance of ridge is better than that of lasso.

Zou and Hastie (Zou and Hastie 2005) proposed an approach called the elastic net, which is a hybrid between lasso and ridge regression, which solves all of these problems. It is apparently called the "elastic net" because it is "like a stretchable fishing net that retains 'all the big fish'" (Zou and Hastie 2005).

13.5.3.1 Vanilla version

The vanilla version of the model defines the following objective function:

$$J(w, \lambda_1, \lambda_2) = \|y - Xw\|^2 + \lambda_2 \|w\|^2 + \lambda_1 \|w\|_1 \quad (13.122)$$

Notice that this penalty function is strictly convex (assuming $\lambda_2 > 0$) so there is a unique global minimum, even if X is not full rank.

It can be shown (Zou and Hastie 2005) that any strictly convex penalty on w will exhibit a grouping effect, which means that the regression coefficients of highly correlated variables tend to be equal (up to a change of sign if they are negatively correlated). For example, if two features are equal, so $X_{:j} = X_{:k}$, one can show that their estimates are also equal, $\hat{w}_j = \hat{w}_k$. By contrast, with lasso, we may have that $\hat{w}_j = 0$ and $\hat{w}_k = 0$ or vice versa.

13.5.3.2 Algorithms for vanilla elastic net

It is simple to show (Exercise 13.5) that the elastic net problem can be reduced to a lasso problem on modified data. In particular, define

$$\tilde{X} = c \sqrt{X^T X} I_D, \tilde{y} = y / \sqrt{c} \quad (13.123)$$

where $c = (1 + \lambda_2)^{-1/2}$. Then we solve

$$\tilde{w} = \arg \min_w \|\tilde{y} - \tilde{X}w\|^2 + c\lambda_1 \|w\|_1 \quad (13.124)$$

and set $w = c^{-1} \tilde{w}$. We can use LARS to solve this subproblem; this is known as the LARS-EN algorithm. If we stop the algorithm after m variables have been included, the cost is $O(m^3 + Dm^2)$. Note that we can use $m = D$ if we wish, since \tilde{X} has rank D . This is in contrast to lasso, which cannot select more than N variables (before jumping to the OLS solution) if $N < D$. When using LARS-EN (or other 1 solvers), one typically uses cross-validation

to select λ_1 and λ_2 .

13.5.3.3 Improved version

Unfortunately it turns out that the “vanilla” elastic net does not produce functions that predict very accurately, unless it is very close to either pure ridge or pure lasso. Intuitively the reason is that it performs shrinkage twice: once due to the 2 penalty and again due to the 1 penalty. The solution is simple: undo the 2 shrinkage by scaling up the estimates from the vanilla version. In other words, if w^* is the solution of Equation 13.124, then a better estimate is

$$\hat{w} = 1 + \lambda_2 \tilde{w} \quad (13.125)$$

We will call this a corrected estimate. One can show that the corrected estimates are given by

$$\hat{w} = \operatorname{argmin}_w w^T X^T X + \lambda_2 I + \lambda_2 w - 2y^T X w + \lambda_1 \|w\|_1 \quad (13.126)$$

Now

$$X^T X + \lambda_2 I + \lambda_2 = (1 - \rho) \hat{\Sigma} + \rho I \quad (13.127)$$

where $\rho = \lambda_2 / (1 + \lambda_2)$. So the the elastic net is like lasso but where we use a version of $\hat{\Sigma}$ that is shrunk towards I . (See Section 4.2.6 for more discussion of regularized estimates of covariance matrices.)

13.5.3.4 GSM interpretation of elastic net

The implicit prior being used by the elastic net obviously has the form

$$p(w|\sigma^2) \propto \exp(-\gamma_1 \sigma \sum_j |w_j| - \gamma_2 \sum_j w_j^2) \quad (13.128)$$

which is just a product of Gaussian and Laplace distributions. This can be written as a hierarchical prior as follows (Kyung et al. 2010; Chen et al. 2011):

$$w_j|\sigma^2, \tau^2 \sim N(0, \sigma^2(\tau^2 - 2j + \gamma_2) - 1) \quad (13.129)$$

$$\tau^2 | \gamma_2 \sim \text{Expon}(\gamma_2) \quad (13.130)$$

Clearly if $\gamma_2 = 0$, this reduces to the regular lasso. It is possible to perform MAP estimation in this model using EM, or Bayesian inference using MCMC (Kyung et al. 2010) or variational Bayes (Chen et al. 2011).

已思考 20s

下面先把原文完整翻译（含公式不漏译），再用通俗解释说明，最后给要点小结。

译文

13.5.3 弹性网 (elastic net: 岭回归与 lasso 的结合)

虽然 lasso 作为变量选择技术很有效，但它有若干问题 (Zou 和 Hastie, 2005)，例如：

- 如果存在一组**高度相关的**变量（比如处于同一生物通路的基因），lasso 往往只挑出其中**一个**，而且挑哪个带有偶然性。（从 LARS 算法能看出来：一旦组里有一个被选入，剩下的那些与“新残差”的相关性就不高了，因此不会再被选中。）通常更好的做法是把一组里**所有相关的变量**都选上。如果我们**知道**分组结构，可以用 group lasso；但多数情况下我们并不知道。
- 当 $D > N$ 时，lasso 在“饱和”之前**最多只能选 N 个变量**。
- 当 $N > D$ 且变量彼此相关时，经验上常看到**岭回归的预测性能优于 lasso**。

Zou 和 Hastie (2005) 提出了“**弹性网**” (elastic net) 这一折衷方法，它介于 lasso 与岭回归之间，能解决上述问题。据说之所以叫“弹性网”，是因为它“像一张可拉伸的渔网，能把**所有大鱼都兜住**”。

13.5.3.1 基础 (vanilla) 版本

基础版模型的目标函数为

$$J(w, \lambda_1, \lambda_2) = \|y - Xw\|_2^2 + \lambda_2 \|w\|_2^2 + \lambda_1 \|w\|_1 \quad (13.122)$$

注意：只要 $\lambda_2 > 0$ ，这个惩罚是**严格凸**的，即使 X 非满秩也有**唯一全局最小值**。

可以证明 (Zou 和 Hastie, 2005)，只要对 w 的惩罚是**严格凸**的，就会出现“**成组效应**”：彼此高度相关的变量，其回归系数倾向于**相等**（若负相关则只是符号相反）。例如，如果两个特征完全相同 $X_{:j} = X_{:k}$ ，那么它们的估计也相等 $\hat{w}_j = \hat{w}_k$ 。相反，在 lasso 下可能出现 $\hat{w}_j = 0$ 而 $\hat{w}_k \neq 0$ (或反之)。

13.5.3.2 基础版的算法

可证明 (练习 13.5) 弹性网问题能化为在**改造过的数据**上做一次 lasso。具体地，定义

$$\tilde{X} = [\frac{c}{\sqrt{\lambda_2}} X \quad I_D], \quad \tilde{y} = [y \quad 0_{D \times 1}] \quad (13.123)$$

其中 $c = (1 + \lambda_2)^{-1/2}$ 。然后求解

$$\tilde{w} = \arg \min_{\tilde{w}} \|\tilde{y} - \tilde{X}\tilde{w}\|_2^2 + c\lambda_1 \|\tilde{w}\|_1 \quad (13.124)$$

最后取 $w = c \tilde{w}$ 。

可用 LARS 解这个子问题，称为 LARS-EN。若在纳入 m 个变量后停止，代价是 $O(m^3 + Dm^2)$ 。注意我们可以取 $m = D$ ，因为 \tilde{X} 的秩是 D 。这与 lasso 不同：若 $N < D$ ，lasso

在跳到 OLS 解之前不能选超过 N 个变量。使用 LARS-EN (或其它 ℓ_1 求解器) 时, 通常用 **交叉验证**来选择 λ_1, λ_2 。

13.5.3.3 改进版本

遗憾的是, “基础版”弹性网除非**非常接近**纯岭或纯 lasso, 否则预测精度往往不够好。直觉原因: 它把系数**收缩了两次**——一次来自 L_2 惩罚, 一次来自 L_1 惩罚。解决很简单: 把基础版得到的估计**按比例放大**, 抵消 L_2 收缩。也就是说, 若式 (13.124) 的解为 \tilde{w} , 更好的估计是

$$\hat{w} = (1 + \lambda_2) \tilde{w} \quad (13.125)$$

我们称其为**校正后的估计**。可证明这些校正估计等价于

$$\hat{w} = \arg \min_w w^\top \frac{X^\top X + \lambda_2 I}{1 + \lambda_2} w - 2y^\top X w + \lambda_1 \|w\|_1 \quad (13.126)$$

并且

$$\frac{X^\top X + \lambda_2 I}{1 + \lambda_2} = (1 - \rho) \hat{\Sigma} + \rho I, \quad \rho = \frac{\lambda_2}{1 + \lambda_2} \quad (13.127)$$

因此, 弹性网就像是 lasso, 但把协方差估计 $\hat{\Sigma}$ 向单位阵收缩后再用 (关于协方差的正则化, 见 4.2.6 节)。

13.5.3.4 弹性网的 GSM 解释

弹性网对应的隐式先验显然为

$$p(w | \sigma^2) \propto \exp\left(-\frac{\gamma_1}{\sigma} \sum_{j=1}^D |w_j| - \frac{\gamma_2}{2\sigma^2} \sum_{j=1}^D w_j^2\right) \quad (13.128)$$

即**高斯与拉普拉斯先验**的乘积。它可写成如下层次 (GSM) 先验 (Kyung 等, 2010; Chen 等, 2011) :

$$w_j | \sigma^2, \tau_j^2 \sim N(0, \sigma^2(\tau_j^{-2} + \gamma_2)^{-1}), \quad \tau_j^2 | \gamma_1 \sim \text{Exp}(\frac{\gamma_1^2}{2})$$

显然当 $\gamma_2 = 0$ 时就退化为普通 lasso。该模型可以用 EM 做 MAP, 也可以用 MCMC (Kyung 等, 2010) 或**变分 Bayes** (Chen 等, 2011) 做贝叶斯推断。

通俗解释

- **为什么需要弹性网？**

Lasso 会：① 对强相关的一组特征只挑一个；② 当 $D > N$ 时最多选 N 个；③ 在相关性强的情形，预测常不如岭回归稳。

Elastic Net = L1 + L2：既能选特征（L1），又能稳定/成组（L2），还能在 $D > N$ 时继续纳入超过 N 个变量（通过 LARS-EN）。

- **基础版目标：** $\|y - Xw\|^2 + \lambda_2 \|w\|_2^2 + \lambda_1 \|w\|_1$ 。

有了 L_2 ，目标严格凸，求解唯一；同时产生成组效应：相关特征的系数会靠拢甚至相等，不再随意只留一个。

- **怎么求？**

把问题改写成在扩展数据 (\tilde{X}, \tilde{y}) 上的 lasso，用 LARS-EN 快速沿正则化路径求解；再把解缩放回 w 。

- **改进版（校正）：** 基础版等于把系数“缩小两次”，会损失预测。把解乘以 $1 + \lambda_2$ 进行放缩可明显改进；等价于在协方差 $\hat{\Sigma}$ 被向单位阵收缩后的二次项上做 lasso。
- **贝叶斯/GSM 视角：** 弹性网 = 拉普拉斯（促稀疏） \times 高斯（促平滑/稳定）的乘积先验；可用 EM/MCMC/VB 做（更重）贝叶斯推断。

要点小结

- Elastic Net = L1 + L2：同时拥有稀疏选择与稳定分组优点。

- **解决痛点：**

1. 相关特征一组一个的问题 → 成组效应把它们一起保留；
2. $D > N$ 时 lasso 最多选 N 个 → LARS-EN 可继续前进；
3. 相关性强时预测不稳 → 加 L_2 更稳健，常优于纯 lasso。

- **算法：** 把 EN 化为 lasso（扩展数据）→ LARS-EN + 交叉验证选 λ_1, λ_2 ；改进版需校正放缩。

- **先验理解：** 等价于拉普拉斯 \times 高斯的层次先验（GSM）。

- **实践建议：** 如果特征高度相关或 D 与 N 规模不匹配，优先考虑弹性网； λ_1 控制稀疏强度， λ_2 控制组/稳定程度。

这一小节主要在讲什么？我不太懂

这小节讲的就是**弹性网** (Elastic Net) ——把 L1 (lasso) 和 L2 (ridge) 两种正则化合在一起，用来克服 lasso 的几个硬伤，并兼收并蓄两者的特点。

为什么要用弹性网？

lasso 很会“挑特征”，但有缺点：

- **强相关的一组特征时**，它常**只留一个**（随缘挑），丢掉同组里其他也有用的特征；
- 当 **特征数 D > 样本数 N** 时，lasso **最多选 N 个特征就“饱和”**；
- 变量相关性强时，ridge 的预测常比 lasso 更稳。

弹性网通过把两种惩罚**相加**来解决：

$$\min_w \|y - Xw\|_2^2 + \lambda_2 \|w\|_2^2 + \lambda_1 \|w\|_1$$

- L1 (λ_1) 让解稀疏，做特征选择；
- L2 (λ_2) 让解更稳、更“成组”，不容易只挑出组里一个。

直觉：像一张“可拉伸的渔网”，既能抓住“大鱼”（相关特征整组保留），又能滤掉小杂鱼（无关特征）。

它具体带来什么效果？

- **分组效应**：强相关特征的系数会**趋于相等**（而不是一个 0、一个非 0）。
- **可选超过 N 个特征**（在 $D > N$ 的情形下，配合 LARS-EN 算法可以继续往下加）。
- **预测更稳健**：在多重共线时，L2 部分抑制过拟合。

怎么算？

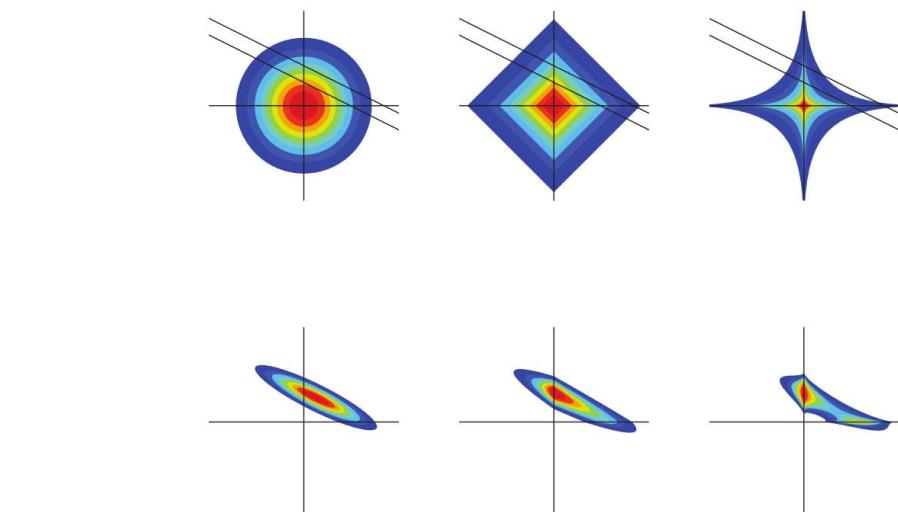
- **基础版可以化成一次 lasso 来解**（对数据做扩展/重缩放，再用 LARS 求路径），俗称 LARS-EN； λ_1, λ_2 用交叉验证调。
- **改进版**：基础版会被 L2+L1“缩两次”，预测略偏；把解再乘以 $(1 + \lambda_2)$ **校正**会更好（等价于在“向单位阵收缩”的协方差上做 lasso）。
- **贝叶斯视角**：等价于**拉普拉斯×高斯**的层次先验 (GSM)，也能用 EM/MCMC/变分做推断。

什么时候用？

- 特征**高度相关**、希望“**整组保留**”；
- D 和 N 比例悬殊（尤其 $D > N$ ）；
- lasso 过于激进、ridge 又太“稠密”时，想要**兼顾稀疏与稳定**。

小结（记住这几条）

- Elastic Net = L1 + L2：既选特征又稳健，解决 lasso 的“只挑一人、不能超 N、预测不稳”。
- 目标函数严格凸 ($\lambda_2 > 0$)，解唯一。
- LARS-EN高效求解；**改进版**需把基础解按 $(1 + \lambda_2)$ 放大以消除双重收缩。
- 调参： λ_1 控制稀疏程度， λ_2 控制稳定/分组程度；交叉验证选取即可。



翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

Figure 13.17 Top: plot of log prior for three different distributions with unit variance: Gaussian, Laplace and exponential power. Bottom: plot of log posterior after observing a single observation, corresponding to a single linear constraint. The precision of this observation is shown by the diagonal lines in the top figure. In the case of the Gaussian prior, the posterior is unimodal and symmetric. In the case of the Laplace prior, the posterior is unimodal and asymmetric (skewed). In the case of the exponential prior, the posterior is bimodal. Based on Figure 1 of (Seeger 2008). Figure generated by sparsePostPlot, written by Florian Steinke.

13.6 Non-convex regularizers

Although the Laplace prior results in a convex optimization problem, from a statistical point of view this prior is not ideal. There are two main problems with it. First, it does not put enough probability mass near 0, so it

does not sufficiently suppress noise. Second, it does not put enough probability mass on large values, so it causes shrinkage of relevant coefficients, corresponding to "signal". (This can be seen in Figure 13.5(a): we see that L1 estimates of large coefficients are significantly smaller than their ML estimates, a phenomenon known as bias.) Both problems can be solved by going to more flexible kinds of priors which have a larger spike at 0 and heavier tails. Even though we cannot find the global optimum anymore, these non-convex methods often outperform L1 regularization, both in terms of predictive accuracy and in detecting relevant variables (Fan and Li 2001; Schniter et al. 2008). We give some examples below.

13.6.1 Bridge regression

A natural generalization of L1 regularization, known as bridge regression (Frank and Friedman 1993), has the form

$$w^* = \text{NLL}(w) + \lambda \sum_j |w_j|^\beta \quad (13.131)$$

for $\beta \geq 0$. This corresponds to MAP estimation using a exponential power distribution given by

$$\text{ExpPower}(w|\mu, a, b) = b^{2a} \Gamma(1 + 1/b) \exp(-|x - \mu|^\beta a b) \quad (13.132)$$

If $\beta = 2$, we get the Gaussian distribution (with $a = \sigma \sqrt{2}$), corresponding to ridge regression; if we set $\beta = 1$, we get the Laplace distribution, corresponding to lasso; if we set $\beta = 0$, we get L0 regression, which is equivalent to best subset selection. Unfortunately, the objective is not convex for $\beta < 1$, and is not sparsity promoting for $\beta > 1$. So the L1 norm is the tightest convex approximation to the L0 norm.

The effect of changing β is illustrated in Figure 13.17, where we plot the prior for $\beta = 2$, $\beta = 1$ and $\beta = 0.4$; we assume $p(w) = p(w_1)p(w_2)$. We also plot the posterior after seeing a single observation, (x, y) , which imposes a single linear constraint of the form, $y = w^T x$, with a certain tolerance controlled by the observation noise (compare to Figure 7.11). We see that the mode of the Laplace is on the vertical axis, corresponding to $w_1 = 0$. By contrast, there are two modes when using $\beta = 0.4$, corresponding to two different sparse solutions. When using the Gaussian, the MAP estimate is not sparse (the mode does not lie on either of the coordinate axes).

已思考 8s

下面先把原文完整翻译（含公式），再用通俗解释说明，最后给要点小结。

图注翻译

图 13.17

上排：三个单位方差分布（高斯、拉普拉斯、指数幂分布）的对数先验等高线图。
下排：在观测到一个样本（等价于一个线性约束）之后的对数后验。上排中的两条斜线表示这次观测的精度/约束带宽。

- 高斯先验：后验是**单峰且对称**。
- 拉普拉斯先验：后验**单峰但不对称**（偏斜）。
- 指数幂先验：后验**双峰**。

基于 Seeger (2008) 图 1。由 sparsePostPlot (Florian Steinke 编写) 生成。

正文翻译

13.6 非凸正则化器

虽然拉普拉斯先验对应一个**凸**优化问题，但从统计角度看它并不理想，主要有两点问题：

- 1) 它在 0 点附近的概率质量不够大，**抑制噪声**不够强；
- 2) 它在大数值处的概率质量也不够大，导致把本应较大的、代表“信号”的系数也**往 0 缩小**（见图 13.5(a)： ℓ_1 对大系数的估计显著小于 ML 估计，这叫**偏差 bias**）。

这两点都可以通过使用**更灵活的先验**来解决：这些先验在 0 处有更尖的“**尖峰**”，在尾部更“**厚**”。尽管此时全局最优一般求不到，这些**非凸方法**在**预测精度和相关变量识别**上往往优于 ℓ_1 正则 (Fan & Li, 2001; Schniter et al., 2008)。下面给出一些例子。

13.6.1 桥回归 (Bridge regression)

对 ℓ_1 正则的一个自然推广是**桥回归** (Frank & Friedman, 1993)：

$$\hat{w} = \arg \min_w \text{NLL}(w) + \lambda \sum_j |w_j|^b \quad (13.131)$$

其中 $b \geq 0$ 。这对应于采用**指数幂分布** (Exponential Power) 的 MAP 估计，其密度为

$$\text{ExpPower}(w | \mu, a, b) = \frac{b}{2a\Gamma(1+1/b)} \exp\left(-\left|\frac{x-\mu}{a}\right|^b\right) \quad (13.132)$$

- 取 $b = 2$ 得到**高斯分布** (令 $a = \sigma\sqrt{2}$) , 对应**岭回归**;
- 取 $b = 1$ 得到**拉普拉斯分布**, 对应 lasso;
- 取 $b = 0$ 得到 ℓ_0 回归, 等价于**最佳子集选择**。

但当 $b < 1$ 时目标函数**非凸**; 当 $b > 1$ 又**不鼓励稀疏**。因此 ℓ_1 范数可以看成对 ℓ_0 的**最紧的凸近似**。

图 13.17 展示了改变 b 的效果。我们画出 $b = 2$ 、 $b = 1$ 、 $b = 0.4$ 的先验 (假设 $p(w) = p(w_1)p(w_2)$) , 再加入一个观测 (x, y) (即线性约束 $y = w^\top x$, 其噪声大小决定两条斜线之间的带宽)。

- **拉普拉斯** ($b = 1$) 时, 后验众数落在纵轴上 ($w_1 = 0$) , 体现稀疏。
- **指数幂** $b = 0.4$ 时, 后验出现**两个众数**, 对应两种不同的稀疏解。
- **高斯** ($b = 2$) 时, MAP 解不稀疏 (众数不在坐标轴上)。

通俗解释

- lasso 用拉普拉斯先验 → 优点: 凸、好解; 缺点:
 - 0 附近不够“尖” ⇒ 对纯噪声的小系数“清零”不够果断;
 - 尾部不够“厚” ⇒ 对真正很大的有用系数也会过度缩小 (产生偏差)。
- 解决思路: 用**更尖峰+更厚尾**的先验 (如指数幂 $b < 1$)。它们对应的正则是 $\sum |w_j|^b$
 - $b \downarrow (<1)$ ⇒ 更鼓励稀疏, 能更好地区分“真 0”和“大系数”; 但代价是**非凸**, 可能有**多个局部最优/多峰后验** (图 13.17 右列)。
 - $b = 2$ (高斯/岭) ⇒ 稠密、稳, 但不稀疏;
 - $b = 1$ (拉普拉斯/lasso) ⇒ 折中: 凸、易算, 但仍会有**系数被整体压小**的偏差。
- 图 13.17 的下排直观演示了加入一个观测后的变化:
 - 高斯先验 → **单峰对称**: 更像“平均化”, 不选零;
 - 拉普拉斯先验 → **单峰偏斜**: 会把某些维度推到 0;
 - 指数幂先验 ($b < 1$) → **双峰**: 提示可能存在多种稀疏解释 (非凸特征)。

要点小结

- **问题**: 拉普拉斯 (ℓ_1) 在 0 附近不够尖、尾部不够厚 ⇒ 对噪声清零不彻底、对大系数有**偏差**。
- **非凸正则**: $\sum |w_j|^b$ (桥回归) 统一了岭 ($b = 2$)、lasso ($b = 1$)、 ℓ_0 ($b = 0$)。

- **性质**: $b < 1$ 非凸但更稀疏、可减小偏差; $b > 1$ 不促稀疏。
- **图示结论**:
 - 高斯先验 → 单峰、不稀疏;
 - 拉普拉斯 → 单峰、可稀疏但有偏差;
 - 指数幂 ($b < 1$) → 多峰、非凸、可能更好抓到真正稀疏结构。
- **实践**: 若追求更强稀疏与更小偏差, 可考虑非凸正则 (SCAD、MCP、 $b < 1$ 等), 但需接受**优化更难与可能陷入局部最优的现实**。

这一小节主要在讲什么? 我不太懂

已思考 9s

这小节在说: **用比 Lasso (L_1) 更“尖、更厚尾”的先验/正则来做更好的稀疏估计——但代价是目标函数变成非凸, 优化更难**。具体以“桥回归 (Bridge regression)”为例说明。

核心思想

- Lasso 对应的拉普拉斯先验虽然让问题**凸**、好求解, 但有两个统计缺点:
 1. 在 0 附近概率不够集中 \Rightarrow 对纯噪声的小系数“清零”不够果断;
 2. 尾部不够厚 \Rightarrow 会把真正很大的“信号”系数也**过度缩小** (有偏, bias)。
- 解决思路: 用**更尖峰 (0 处更高)** + **更厚尾**的先验 \rightarrow 会产生**非凸正则**, 但往往在预测和**找对变量上**比 L_1 更好。

桥回归 (Bridge regression)

把惩罚从 $\sum |w_j|$ 推广为

$$\text{NLL}(w) + \lambda \sum_j |w_j|^b, \quad b \geq 0$$

这等价于使用**指数幂分布**先验。不同的 b 把常见方法统一起来:

- $b = 2$: 高斯/岭回归 (**稠密**、稳, 但不稀疏)
- $b = 1$: 拉普拉斯/Lasso (**凸**、能稀疏, 但会把大系数**缩小**)
- $b \rightarrow 0$: ℓ_0 /**最佳子集选择** (最“硬核”的稀疏, 但**最难解**)
- $0 < b < 1$: **非凸**, 比 L_1 更促稀疏、对大系数偏差更小, 但可能有多个局部最优

图 13.17 想表达什么

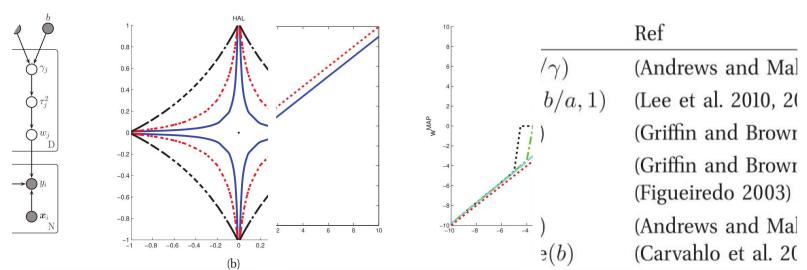
- 上排是三种先验（同一方差）的对数等高线：高斯（圆）、拉普拉斯（菱形）、指数幂 $b = 0.4$ （尖角更狠）。
- 下排是在加入一个线性观测后的对数后验：
 - 高斯先验：**后验单峰且对称 \rightarrow MAP 不落在坐标轴上，**不稀疏**。
 - 拉普拉斯：**后验单峰但偏斜，众数可能在轴上（把某一维推到 0） \rightarrow **稀疏，但有收缩**。
 - $b < 1$ ：后验常**双峰/多峰** \rightarrow 反映“可能有多种稀疏解”，也说明**非凸**、优化更难。

该怎么用（直觉指南）

- 想要**更强稀疏、对大系数少偏差**：可考虑 $0 < b < 1$ 的非凸正则（或其它非凸如 SCAD/MCP）。
- 代价：**可能陷入局部最优**，算法更复杂（常用重加权 L_1 /IRLS 等技巧）。
- 若追求**稳健+易解**：仍以 Lasso/Elastic Net 为首选。

要点小结

- L_1 易算但**0 处不够尖、尾不够厚** \Rightarrow 抑噪不彻底、对大系数有偏。
- 非凸正则**（桥回归 $|w|^b$ 等）用“更尖+更厚尾”的先验，常能**更准且更会选特征**。
- 统一视角**：岭 ($b = 2$) —Lasso ($b = 1$) — ℓ_0 ($b \rightarrow 0$)。
- 权衡**：效果 vs. 计算； $b < 1$ **效果好但难优化**， $b = 1$ **折中且凸**， $b = 2$ **稳但不稀疏**。



翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

Figure 13.18 (a) DGM for hierarchical adaptive lasso. (b) Contours of Hierarchical adaptive Laplace. Based on Figure 1 of (Lee et al. 2010).

Figure 13.19 Thresholding behavior of two penalty functions (negative log priors). (a) Laplace. (b) Hierarchical adaptive Laplace. Based on Figure 2 of (Lee et al. 2010).
 Table 13.2 Some scale mixtures of Gaussians.

Abbreviations: C+ = half-rectified Cauchy; Ga = Gamma (shape and rate parameterization); GT = generalized t; IG = inverse Gamma; NEG = Normal-ExponentialGamma; NG = Normal-Gamma; NJ = Normal-Jeffreys. The horseshoe distribution is the name we give to the distribution induced on w_j by the prior described in (Carvahlo et al. 2010); this has no simple analytic form. The definitions of the NEG and NG densities are a bit complicated, but can be found in the references. The other distributions are defined in the text.

13.6.2 Hierarchical adaptive lasso

Recall that one of the principal problems with lasso is that it results in biased estimates. This is because it needs to use a large value of λ to “squash” the irrelevant parameters, but this then over-penalizes the relevant parameters. It would be better if we could associate a different penalty parameter with each parameter. Of course, it is completely infeasible to tune D parameters by cross validation, but this poses no problem to the Bayesian: we simply make each τ_{2j} have its own private tuning parameter, γ_j , which are now treated as random variables coming from the conjugate prior $\gamma_j \sim \text{IG}(a, b)$.

The full model is as follows:

$$\gamma_j \sim \text{IG}(a, b) \quad (13.133)$$

$$\tau_{2j} | \gamma_j \sim \text{Ga}(1, \gamma_j / 2) \quad (13.134)$$

$$w_j | \tau_{2j} \sim N(0, \tau_{2j}) \quad (13.135)$$

See Figure 13.18(a). This has been called the hierarchical adaptive lasso (HAL) (Lee et al. 2010) (see also (Lee et al. 2011; Cevher 2009; Armagan et al. 2011)). We can integrate out τ_{2j} , which induces a $\text{Lap}(w_j | 0, 1/\gamma_j)$ distribution on w_j as before. The result is that $p(w_j)$ is now a scaled mixture of Laplacians. It turns out that we can fit this model (i.e., compute a local posterior mode) using EM, as we explain below. The resulting estimate, \hat{w}_{HAL} , often works much better than the estimate returned by lasso, \hat{w}_{L1} , in the sense that it is more likely to contain zeros in the right places (model selection consistency) and more likely to result in good predictions (prediction consistency) (Lee et al. 2010). We give an explanation for this behavior in Section 13.6.2.2.

13.6.2.1 EM for HAL

Since the inverse Gamma is conjugate to the Laplace, we

find that the E step for γ_j is given by

$$p(y_j|w_j) = IG(a + 1, b + |w_j|) \quad (13.136)$$

The E step for σ^2 is the same as for vanilla lasso. The prior for w has the following form:

$$p(w|\gamma) = j^{-1} 2\gamma_j \exp(-|w_j|/\gamma_j) \quad (13.137)$$

Hence the M step must optimize

$$\hat{w}^{(t+1)} = \operatorname{argmax}_w w \log N(y|Xw, \sigma^2) - j |w_j| E[1/\gamma_j] \quad (13.138)$$

The expectation is given by

$$E[1/\gamma_j] = a + 1/b + |w^{(t)}_j|/s(t)_j \quad (13.139)$$

Thus the M step becomes a weighted lasso problem:

$$\hat{w}^{(t+1)} = \operatorname{argmin}_w \|y - Xw\|^2/2 + j s(t)_j |w_j| \quad (13.140)$$

This is easily solved using standard methods (e.g., LARS).

Note that if the coefficient was estimated to be large in the previous iteration (so $w^{(t)}_j$ is large), then the scaling factor $s(t)_j$ will be small, so large coefficients are not penalized heavily. Conversely, small coefficients do get penalized heavily. This is the way that the algorithm adapts the penalization strength of each coefficient. The result is an estimate that is often much sparser than returned by lasso, but also less biased.

Note that if we set $a = b = 0$, and we only perform 1 iteration of EM, we get a method that is closely related to the adaptive lasso of (Zou 2006; Zou and Li 2008). This EM algorithm is also closely related to some iteratively reweighted 1 methods proposed in the signal processing community (Chartrand and Yin 2008; Candes et al. 2008).

13.6.2.2 Understanding the behavior of HAL

We can get a better understanding of HAL by integrating out γ_j to get the following marginal distribution,

$$p(w_j|a, b) = a^{-1} b^{-1} |w_j|^{a-1} (b + |w_j|)^{-a-1} \quad (13.141)$$

This is an instance of the generalized t distribution

(McDonald and Newey 1988) (in (Cevher 2009; Armagan et al. 2011), this is called the double Pareto distribution) defined as

$$GT(w|\mu, a, c, q) = c^{-1} a^{1/q} B(1/q, a) \frac{1}{1 + |w - \mu|^{q/c}} \quad (13.142)$$

where c is the scale parameter (which controls the degree of sparsity), and a is related to the degrees of freedom.

When $q = 2$ and $c = \sqrt{2}$ we recover the standard t distribution; when $a \rightarrow \infty$, we recover the exponential power distribution; and when $q = 1$ and $a = \infty$ we get the Laplace distribution. In the context of the current model, we see that $p(w_j|a, b) = GT(w_j|0, a, b/a, 1)$. The resulting

penalty term has the form

$$\pi\lambda(w_j) = -\log p(w_j) = (a + 1) \log(1 + |w_j| b) + \text{const} \quad (13.143)$$

where $\lambda = (a, b)$ are the tuning parameters. We plot this penalty in 2d (i.e., we plot $\pi\lambda(w_1) + \pi\lambda(w_2)$) in Figure 13.18(b) for various values of b . Compared to the diamond-shaped Laplace penalty, shown in Figure 13.3(a), we see that the HAL penalty looks more like a “star fish”: it puts much more density along the “spines”, thus enforcing sparsity more aggressively. Note that this penalty is clearly not convex. We can gain further understanding into the behavior of this penalty function by considering applying it to the problem of linear regression with an orthogonal design matrix. In this case, one can show that the objective becomes

$$\begin{aligned} J(w) &= 1/2 \|y - Xw\|^2 + D \sum_{j=1}^D \pi\lambda(|w_j|) \\ &= 1/2 \|y - \hat{y}\|^2 + 1/2 \sum_{j=1}^D (\hat{w}_{mle,j} - w_j)^2 + \sum_{j=1}^D \pi\lambda(|w_j|) \end{aligned} \quad (13.144)$$

where $\hat{w}^{mle} = XT^{-1}y$ is the MLE and $\hat{y} = X\hat{w}^{mle}$. Thus we can compute the MAP estimate one dimension at a time by solving the following 1d optimization problem:

$$\hat{w}_j = \operatorname{argmin}_{w_j} 1/2 (\hat{w}_{mle,j} - w_j)^2 + \pi\lambda(w_j) \quad (13.146)$$

In Figure 13.19(a) we plot the lasso estimate, \hat{w}^{L1} , vs the ML estimate, \hat{w}^{mle} . We see that the L1 estimator has the usual soft-thresholding behavior seen earlier in Figure 13.5(a). However, this behavior is undesirable since the large magnitude coefficients are also shrunk towards 0, whereas we would like them to be equal to their unshrunken ML estimates. In Figure 13.19(b) we plot the HAL estimate, \hat{w}^{HAL} , vs the ML estimate \hat{w}^{mle} . We see that this approximates the more desirable hard thresholding behavior seen earlier in Figure 13.5(b) much more closely.

13.6.3 Other hierarchical priors

Many other hierarchical sparsity-promoting priors have been proposed; see Table 13.2 for a brief summary. In some cases, we can analytically derive the form of the marginal prior for w_j . Generally speaking, this prior is not concave. A particularly interesting prior is the improper Normal-Jeffreys prior, which has been used in (Figueiredo 2003). This puts a non-informative Jeffreys prior on the variance, $Ga(\tau^2_j | 0, 0) \propto 1/\tau^2_j$; the resulting marginal has the form $p(w_j) = NJ(w_j) \propto 1/|w_j|$. This gives rise to a

thresholding rule that looks very similar to HAL in Figure 13.19(b), which in turn is very similar to hard thresholding. However, this prior has no free parameters, which is both a good thing (nothing to tune) and a bad thing (no ability to adapt the level of sparsity).

下面我先完整翻译文段与图注/表注，然后用通俗解释讲清楚思路，最后给要点小结。

图 13.18 与图 13.19、表 13.2 的图注/表注翻译

图 13.18

- (a) 分层自适应 Lasso (HAL, hierarchical adaptive lasso) 的有向图模型 (DGM)。
- (b) 分层自适应拉普拉斯 (HAL 先验) 的等值线。改编自 Lee 等 (2010) 图 1。

图 13.19

两种惩罚函数（也就是负对数先验）的“阈值化”行为。

- (a) 拉普拉斯 (Lasso)。
- (b) 分层自适应拉普拉斯 (HAL)。改编自 Lee 等 (2010) 图 2。

表 13.2

若干“高斯尺度混合” (scale mixtures of Gaussians)。缩写：

C^+ = 半截柯西； Ga = Gamma (形状/率参数化)； GT = 广义 t； IG = 逆 Gamma； NEG = Normal-Exponential-Gamma； NG = Normal-Gamma； NJ = Normal-Jeffreys。

“horseshoe (马蹄) 分布”指 Carvalho 等 (2010) 所述先验在 w_j 上诱导出的分布；它没有简洁的解析式。NEG 与 NG 的密度形式较复杂，见引用文献；其它分布本文已定义。

正文翻译：13.6.2 分层自适应 Lasso (HAL)

回想 Lasso 的主要问题之一是估计有偏：为了“压扁”无关的参数，需要把 λ 取得较大，但这又会过度惩罚真正相关的参数。更好的做法是每个参数配一个自己的惩罚强度。当然，用交叉验证去调 D 个参数完全不可行，但对贝叶斯方法来说这不成问题：令每个 τ_j^2 都拥有自己的调节参数 γ_j ，并把 γ_j 当作来自共轭先验的随机变量， $\gamma_j \sim IG(a, b)$ 。完整模型为

$$\gamma_j \sim IG(a, b) \quad (13.133)$$

$$\tau_j^2 \mid \gamma_j \sim Ga(1, \frac{\gamma_j^2}{2}) \quad (13.134)$$

$$w_j \mid \tau_j^2 \sim N(0, \tau_j^2) \quad (13.135)$$

见图 13.18(a)。这被称为**分层自适应 Lasso (HAL)** (Lee et al., 2010; 也见 Lee et al., 2011; Cevher, 2009; Armanag et al., 2011)。积分掉 τ_j^2 后, 在 w_j 上诱导出与前文相同的 Laplace 分布 $Lap(w_j \mid 0, 1/\gamma_j)$ 。因此 $p(w_j)$ 变成了**拉普拉斯的尺度混合**。该模型可以用 EM 进行拟合 (求一个局部后验众数), 下面说明。数值结果表明, 得到的估计 \hat{w}_{HAL} 往往比 Lasso 的 \hat{w}_{ℓ_1} 更好: 更可能在正确的位置得到 0 (模型选择一致性), 也更可能有好的预测 (预测一致性) (Lee et al., 2010)。我们在 13.6.2.2 节给出解释。

13.6.2.1 HAL 的 EM 算法

由于逆 Gamma 与拉普拉斯是共轭的, γ_j 的 E 步为

$$p(\gamma_j \mid w_j) = IG(a + 1, b + |w_j|) \quad (13.136)$$

σ^2 的 E 步与普通 Lasso 相同。 w 的先验为

$$p(w \mid \gamma) = \prod_j \frac{1}{2\gamma_j} \exp\left(-\frac{|w_j|}{\gamma_j}\right) \quad (13.137)$$

因此 M 步需要最优化

$$\hat{w}^{(t+1)} = \arg \max_w \log N(y \mid Xw, \sigma^2) - \sum_j |w_j| E\left[\frac{1}{\gamma_j}\right] \quad (13.138)$$

其中

$$E\left[\frac{1}{\gamma_j}\right] = \frac{a + 1}{b + |w_j^{(t)}|} = s_j^{(t)} \quad (13.139)$$

于是 M 步就是一个**带权的 Lasso**:

$$\hat{w}^{(t+1)} = \arg \min_w \|y - Xw\|_2^2 + \sum_j s_j^{(t)} |w_j| \quad (13.140)$$

用标准方法 (如 LARS) 即可求解。注意: 若上一步 $w_j^{(t)}$ 很大, 则 $s_j^{(t)}$ 很小, 因而**大系数惩罚得轻**; 而小系数惩罚得重。算法就这样**自适应地**为每个系数调整惩罚强度。结果通常比 Lasso **更稀疏但偏差更小**。

当 $a = b = 0$ 且只做 1 次 EM 迭代时, 此法与 Zou (2006) /Zou & Li (2008) 的**自适应 Lasso**很接近; 它也与信号处理领域的**迭代重加权** ℓ_1 方法 (Chartrand & Yin, 2008; Candes et al., 2008) 密切相关。

13.6.2.2 HAL 的行为机理

把 γ_j 积分掉可得 w_j 的边缘分布

$$p(w_j | a, b) = \frac{a}{2b} \left(1 + \frac{|w_j|}{b}\right)^{-(a+1)} \quad (13.141)$$

这是**广义 t 分布** (McDonald & Newey, 1988) 的一个特例,

$$\text{GT}(w | \mu, a, c, q) = \frac{q}{2c a^{1/q} B(1/q, a)} \left(1 + \frac{|w - \mu|^q}{a c^q}\right)^{-(a+1/q)} \quad (13.142)$$

其中 c 控制稀疏度, a 与自由度相关。取 $q = 2, c = \sqrt{2}$ 得标准 t; 当 $a \rightarrow \infty$ 得指数幂分布; 当 $q = 1, a = \infty$ 得拉普拉斯。对当前模型, 有

$$p(w_j | a, b) = \text{GT}(w_j | 0, a, b/a, 1).$$

对应的惩罚项 (负对数先验) 为

$$\pi_\lambda(w_j) = -\log p(w_j) = (a + 1) \log\left(1 + \frac{|w_j|}{b}\right) + \text{const}, \quad \lambda = (a, b) \quad (13.143)$$

在二维上 (画 $\pi_\lambda(w_1) + \pi_\lambda(w_2)$) 的等值线如图 13.18(b) 所示: 与菱形的拉普拉斯惩罚 (见图 13.3(a)) 相比, HAL 的等值线像“海星”, 在坐标轴方向有更大的密度——这意味着**更强的稀疏推动**。该惩罚显然**非凸**。

对**正交设计矩阵**的线性回归, 可把目标写成

$$J(w) = \frac{1}{2} \|y - Xw\|_2^2 + \sum_{j=1}^D \pi_\lambda(|w_j|) \quad (13.144)$$

$$= \frac{1}{2} \|y - \hat{y}\|_2^2 + \frac{1}{2} \sum_j (\hat{w}_j^{\text{mle}} - w_j)^2 + \sum_j \pi_\lambda(|w_j|) \quad (13.145)$$

其中 $\hat{w}^{\text{mle}} = X^\top y$, $\hat{y} = X\hat{w}^{\text{mle}}$ 。于是每一维的 MAP 可由 1 维问题得到:

$$\hat{w}_j = \arg \min_{w_j} \frac{1}{2} (\hat{w}_j^{\text{mle}} - w_j)^2 + \pi_\lambda(w_j) \quad (13.146)$$

图 13.19(a) 画出了 Lasso 的 \hat{w}_{ℓ_1} 与 \hat{w}_{mle} 的关系: 是**软阈值** (见图 13.5(a)), 连**大系数**也被向 0 缩—这并不理想。图 13.19(b) 画出了 HAL 的 \hat{w}_{HAL} 与 \hat{w}_{mle} : 它更接近**硬阈值** (见图 13.5(b)) 的理想行为—小的直接变 0, 大的基本不缩。

13.6.3 其它分层先验 (翻译)

已经提出了很多其它能促进稀疏的分层先验；表 13.2 给出了简要汇总。有些情况下可以解析出 w_j 的边缘先验，一般来说它们都**非凹/非凸**。一个特别有趣的先验是**不适当的 Normal-Jeffreys (NJ) 先验** (Figueiredo, 2003)：它在方差上放 Jeffreys 非信息先验 $\text{Ga}(\tau_j^2 | 0, 0) \propto 1/\tau_j^2$ ，从而在 w_j 上得到 $p(w_j) = \text{NJ}(w_j) \propto 1/|w_j|$ 。它诱导出的阈值规则与图 13.19(b) 的 HAL 很像，也很接近硬阈值。不过 NJ 没有可调参数：这既是优点（不用调参），也是缺点（无法按数据自适应稀疏度）。

通俗解释

- **为什么 HAL？**

Lasso 用同一个 λ 管所有系数：为了把“真 0”压到 0，不得不把 λ 设大，于是把“真大”的系数也压小了（**偏差**）。

HAL 让每个系数 w_j 拥有自己的“惩罚强度” γ_j （自动从数据里学），**大系数→惩罚小，小系数→惩罚大**，既能更狠地把噪声清掉，又能少缩真信号。

- **怎么做？**

把 Lasso 的拉普拉斯先验改成“**拉普拉斯的尺度混合**”，在 γ_j 上放逆 Gamma 先验。用 EM：

- E 步：按当前 w_j 更新 γ_j 的分布及其 $E[1/\gamma_j]$ ；
- M 步：解一个**带权 Lasso**（权重由上一步给出）。

这就像“**迭代重加权** ℓ_1 ”：越像 0 的分量权越大（惩罚重），越不像 0 的权越小（惩罚轻）。

- **它为什么更好？**

HAL 的边缘先验是**广义 t**，在 0 处更尖、尾更厚；对应的惩罚像“海星”有突出“脊柱”，更推稀疏。阈值曲线更像**硬阈值**：小的直接归零，大的基本保持不动——因此**更稀疏、偏差更小**，模型选择与预测往往更好。

- **其它选择**

表 13.2 罗列了很多同类的分层稀疏先验（NEG、NG、马蹄、NJ 等）。它们都可以看作“**高斯 + 随机方差**”的混合，目的都是：**0 附近更尖、尾更厚**。

要点小结

- HAL 通过为每个系数引入**私有惩罚强度** γ_j （带逆 Gamma 先验），把 Lasso 变成**自适应、分层**的模型。
- EM 实现：E 步得 $E[1/\gamma_j] = \frac{a+1}{b+|w_j|}$ ，M 步变成**带权 Lasso**。

- 边缘先验 $p(w_j)$ 为广义 t, 负对数惩罚 $(a + 1) \log(1 + |w_j|/b)$: 比拉普拉斯更促稀疏、对大系数更少收缩; 效果接近硬阈值。
- HAL 的估计常较 Lasso: 更稀疏、偏差更小、模型选择/预测更稳。
- 这一类方法普遍是非凸的 (优化更难), 但统计性能往往优于纯 ℓ_1 。
- 还有多种相关分层先验 (NEG/NG/马蹄/NJ 等), 都可视作高斯尺度混合。

翻译并解释以下内容, 翻译不能有遗漏, 解释请通俗易懂且详尽, 因为我的数学基础不太好, 最后总结要点。

13.7 Automatic relevance determination (ARD)/sparse Bayesian learning (SBL)

All the methods we have considered so far (except for the spike-and-slab methods in Section 13.2.1) have used a factorial prior of the form $p(w) = \prod_j p(w_j)$. We have seen how these priors can be represented in terms of Gaussian scale mixtures of the form $w_j \sim N(0, \tau_{2j}^2)$, where τ_{2j} has one of the priors listed in Table 13.2. Using these latent variances, we can represent the model in the form $\tau_{2j} \rightarrow w_j \rightarrow y \leftarrow X$. We can then use EM to perform MAP estimation, wherein the E step we infer $p(\tau_{2j} | w_j)$, and in the M step we estimate w from y , X and τ . This M step either involves a closed-form weighted L2 optimization (in the case of Gaussian scale mixtures), or a weighted L1 optimization (in the case of Laplacian scale mixtures). We also discussed how to perform Bayesian inference in such models, rather than just computing MAP estimates.

In this section, we discuss an alternative approach based on type II ML estimation (empirical Bayes), whereby we integrate out w and maximize the marginal likelihood wrt τ . This EB procedure can be implemented via EM, or via a reweighted L1 scheme, as we will explain below. Having estimated the variances, we plug them in to compute the posterior mean of the weights, $E[w|\tau, D]$; rather surprisingly (in view of the Gaussian prior), the result is an (approximately) sparse estimate, for reasons we explain below.

In the context of neural networks, this method is called automatic relevance determination or ARD

(MacKay 1995b; Neal 1996): see Section 16.5.7.5. In the context of the linear models we are considering in this chapter, this method is called sparse Bayesian learning or SBL (Tipping 2001). Combining ARD/SBL with basis function expansion in a linear model gives rise to a technique called the relevance vector machine (RVM), which we will discuss in Section 14.3.2.

13.7.1 ARD for linear regression

We will explain the procedure in the context of linear regression; ARD for GLMs requires the use of the Laplace (or some other) approximation. case can be It is conventional, when discussing ARD / SBL, to denote the weight precisions by $\alpha_j = 1/\tau^2 j$, and the measurement precision by $\beta = 1/\sigma^2$ (do not confuse this with the use of β in statistics to represent the regression coefficients!). In particular, we will assume the following model:

$$p(y|x, w, \beta) = N(y|w^T x, 1/\beta) \quad (13.147)$$

$$p(w) = N(w|0, A^{-1}) \quad (13.148)$$

where $A = \text{diag}(\alpha)$. The marginal likelihood can be computed analytically as follows:

$$p(y|X, \alpha, \beta) = N(y|Xw, \beta I_N)N(w|0, A)dw \quad (13.149)$$

$$= N(y|0, \beta I_N + XA^{-1}XT) \quad (13.150)$$

$$= (2\pi)^{-N/2} |C\alpha|^{-1/2} \exp(-\frac{1}{2} y^T C\alpha^{-1} y) \quad (13.151)$$

where

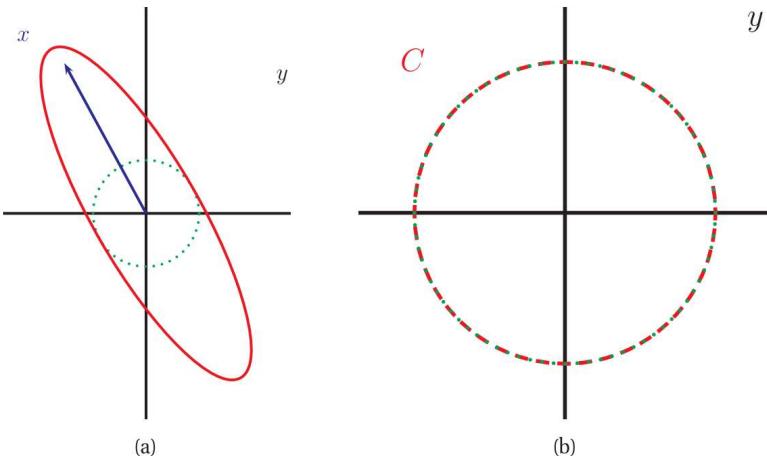
$$C\alpha = \beta^{-1} I_N + XA^{-1}XT \quad (13.152)$$

Compare this to the marginal likelihood in Equation 13.13 in the spike and slab model; modulo the $\beta = 1/\sigma^2$ factor missing from the second term, the equations are the same, except we have replaced the binary $y_j \in \{0, 1\}$ with continuous $\alpha_j \in \mathbb{R}_+$. In log form, the objective becomes

$$l(\alpha, \beta) - \frac{1}{2} \log p(y|X, \alpha, \beta) = \log |C\alpha| + y^T C\alpha^{-1} y \quad (13.153)$$

To regularize the problem, we may put a conjugate prior on each precision, $\alpha_j \sim Ga(a, b)$ and $\beta \sim Ga(c, d)$. The modified objective becomes $(\alpha, \beta) - \frac{1}{2} \log p(y|X, \alpha, \beta) + j \log Ga(\alpha_j|a, b) + \log Ga(\beta|c, d)$ (13.154)

$$= \log |C\alpha| + y^T C\alpha^{-1} y + j(a \log \alpha_j - b \alpha_j) + c \log \beta - d \beta \quad (13.155)$$



翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

Figure 13.20 Illustration of why ARD results in sparsity. The vector of inputs x does not point towards the vector of outputs y , so the feature should be removed. (a) For finite α , the probability density is spread in directions away from y . (b) When $\alpha = \infty$, the probability density at y is maximized. Based on Figure 8 of (Tipping 2001).

13.7 Automatic relevance determination (ARD)/sparse Bayesian learning (SBL)

All the methods we have considered so far (except for the spike-and-slab methods in Section 13.2.1) have used a factorial prior of the form $p(w) = \prod_j p(w_j)$. We have seen how these priors can be represented in terms of Gaussian scale mixtures of the form $w_j \sim N(0, \tau_{w_j}^2)$, where $\tau_{w_j}^2$ has one of the priors listed in Table 13.2. Using these latent variances, we can represent the model in the form $\tau_{w_j}^2 \rightarrow w_j \rightarrow y \leftarrow X$. We can then use EM to perform MAP estimation, wherein the Estepweinfer $p(\tau_{w_j}^2 | w_j)$, and in the M step we estimate w from y , X and τ . This M step either involves a closed-form weighted L2 optimization (in the case of Gaussian scale mixtures), or a weighted L1 optimization (in the case of Laplacian scale mixtures). We also discussed how to perform Bayesian inference in such models, rather than just computing MAP estimates.

In this section, we discuss an alternative approach based on type II ML estimation (empirical Bayes), whereby we integrate out w and maximize the marginal likelihood wrt τ . This EB procedure can be implemented via EM, or via a

reweighted l1 scheme, as we will explain below. Having estimated the variances, we plug them in to compute the posterior mean of the weights, $E[w|\tau, D]$; rather surprisingly (in view of the Gaussian prior), the result is an (approximately) sparse estimate, for reasons we explain below.

In the context of neural networks, this method is called automatic relevance determination or ARD (MacKay 1995b; Neal 1996): see Section 16.5.7.5. In the context of the linear models we are considering in this chapter, this method is called sparse Bayesian learning or SBL (Tipping 2001). Combining ARD/SBL with basis function expansion in a linear model gives rise to a technique called the relevance vector machine (RVM), which we will discuss in Section 14.3.2.

13.7.1 ARD for linear regression

We will explain the procedure in the context of linear regression; ARD for GLMs requires the use of the Laplace (or some other) approximation. It is conventional, when discussing ARD / SBL, to denote the weight precisions by $\alpha_j = 1/\tau^2 j$, and the measurement precision by $\beta = 1/\sigma^2$ (do not confuse this with the use of β in statistics to represent the regression coefficients!). In particular, we will assume the following model:

$$p(y|x, w, \beta) = N(y|w^T x, 1/\beta) \quad (13.147)$$

$$p(w) = N(w|0, A^{-1}) \quad (13.148)$$

where $A = \text{diag}(\alpha)$. The marginal likelihood can be computed analytically as follows:

$$p(y|X, \alpha, \beta) = N(y|Xw, \beta I)N(w|0, A)dw \quad (13.149)$$

$$= N(y|0, \beta I + XA^{-1}XT) \quad (13.150)$$

$$= (2\pi)^{-N/2} |\beta I + XA^{-1}XT|^{-1/2} \exp(-\frac{1}{2} y^T C \alpha - \frac{1}{2} \alpha^T y) \quad (13.151)$$

where

$$C\alpha = \beta^{-1} I + XA^{-1}XT \quad (13.152)$$

Compare this to the marginal likelihood in Equation 13.13 in the spike and slab model; modulo the $\beta = 1/\sigma^2$ factor missing from the second term, the equations are the same, except we have replaced the binary $y_j \in \{0, 1\}$ with continuous $\alpha_j \in \mathbb{R}_+$. In log form, the objective becomes

$$l(\alpha, \beta) = -\frac{1}{2} \log p(y|X, \alpha, \beta) = \log |C\alpha| + y^T C \alpha - \frac{1}{2} \alpha^T y \quad (13.153)$$

To regularize the problem, we may put a conjugate prior on each precision, $\alpha_j \sim Ga(a, b)$ and $\beta \sim Ga(c, d)$. The modified objective becomes

$$l(\alpha, \beta) = -\frac{1}{2} \log p(y|X, \alpha, \beta) + j \log Ga(\alpha_j|a, b) + \log Ga(\beta|c, d)$$

d) (13.154)

$$= \log |C\alpha| + yTC - 1/\alpha y + j(a \log \alpha_j - b\alpha_j) + c \log \beta - d\beta$$

(13.155)

This is useful when performing Bayesian inference for α and β (Bishop and Tipping 2000). However, when performing (type II) point estimation, we will use the improper prior $a = b = c = d = 0$, which results in maximal sparsity.

Below we describe how to optimize (α, β) wrt the precision terms α and β .⁷ This is a proxy for finding the most probable model setting of y in the spike and slab model, which in turn is closely related to l_0 regularization. In particular, it can be shown (Wipf et al. 2010) that the objective in Equation 13.153 has many fewer local optima than the l_0 objective, and hence is much easier to optimize. Once we have estimated α and β , we can compute the posterior over the parameters using

$$p(w|D, \hat{\alpha}, \hat{\beta}) = N(\mu, \Sigma) \quad (13.156)$$

$$\Sigma^{-1} = \hat{\beta} X^T X + A \quad (13.157)$$

$$\mu = \hat{\beta} \Sigma X^T y \quad (13.158)$$

The fact that we compute a posterior over w , while simultaneously encouraging sparsity, is why the method is called “sparse Bayesian learning”. Nevertheless, since there are many ways to be sparse and Bayesian, we will use the “ARD” term instead, even in the linear model context. (In addition, SBL is only “being Bayesian” about the values of the coefficients, rather than reflecting uncertainty about the set of relevant variables, which is typically of more interest.)

13.7.2 Whence sparsity?

If $\hat{\alpha}_j \approx 0$, we find $\hat{w}_j \approx \hat{w}_{mle,j}$, since the Gaussian prior shrinking w_j towards 0 has zero precision. However, if we find that $\hat{\alpha}_j \approx \infty$, then the prior is very confident that $w_j = 0$, and hence that feature j is “irrelevant”. Hence the posterior mean will have $\hat{w}_j \approx 0$. Thus irrelevant features automatically have their weights “turned off” or “pruned out”.

We now give an intuitive argument, based on (Tipping 2001), about why ML-II should encourage $\alpha_j \rightarrow \infty$ for irrelevant features. Consider a 1d linear regression with 2 training examples, so $X = x = (x_1, x_2)$, and $y = (y_1, y_2)$. We can plot x and y as vectors in the plane, as shown in Figure 13.20. Suppose the feature is irrelevant for

predicting the response, so x points in a nearly orthogonal direction to y . Let us see what happens to the marginal likelihood as we change α . The marginal likelihood is given by $p(y|x,\alpha,\beta) = N(y|0,C)$, where

$$C = 1/\beta I + 1/\alpha x x^T \quad (13.159)$$

If α is finite, the posterior will be elongated along the direction of x , as in Figure 13.20(a). However, if $\alpha = \infty$, we find $C = 1/\beta I$, so C is spherical, as in Figure 13.20(b). If $|C|$ is held constant, the latter assigns higher probability density to the observed response vector y , so this is the preferred solution. In other words, the marginal likelihood “punishes” solutions where α_j is small but $X_{:,j}$ is irrelevant, since these waste probability mass. It is more parsimonious (from the point of view of Bayesian Occam’s razor) to eliminate redundant dimensions.

13.7.3 Connection to MAP estimation

ARD seems quite different from the MAP estimation methods we have been considering earlier in this chapter. In particular, in ARD, we are not integrating out α and optimizing w , but viceversa. Because the parameters w_j become correlated in the posterior (due to explaining away), when we estimate α_j we are borrowing information from all the features, not just feature j . Consequently, the effective prior $p(w|\alpha)$ is non-factorial, and furthermore it depends on the data D (and σ^2). However, in (Wipf and Nagarajan 2007), it was shown that ARD can be viewed as the following MAP estimation problem:

$$w^{\text{ARD}} = \operatorname{argmin}_w \|w\|_2^2 + g_{\text{ARD}}(w) \quad (13.160)$$

$$g_{\text{ARD}}(w) = \min_{\alpha \geq 0} \sum_j \alpha_j w_j^2 + \log |\mathbf{C}\alpha| \quad (13.161)$$

The proof, which is based on convex analysis, is a little complicated and hence is omitted. Furthermore, (Wipf and Nagarajan 2007; Wipf et al. 2010) prove that MAP estimation with non-factorial priors is strictly better than MAP estimation with any possible factorial prior in the following sense: the non-factorial objective always has fewer local minima than factorial objectives, while still satisfying the property that the global optimum of the non-factorial objective corresponds to the global optimum of the ℓ_0 objective — a property that ℓ_1 regularization, which has no local minima, does not enjoy.

下面先给出你贴出内容的**完整翻译（不遗漏）**，随后用大白话**详细解释**，最后**总结要点**。

翻译

图 13.20 说明了为什么 ARD 会产生稀疏性。输入向量 x 并不指向输出向量 y 的方向，因此这个特征应该被移除。(a) 当 α 有限时，概率密度会沿着远离 y 的方向“拉长”。(b) 当 $\alpha = \infty$ 时，在 y 处的概率密度被最大化。改编自 (Tipping 2001) 的图 8。

13.7 自动相关性确定 (ARD) / 稀疏贝叶斯学习 (SBL)

到目前为止我们讨论的所有方法（第 13.2.1 节的 spike-and-slab 方法除外）都使用了可分解 (factorial) 的先验 $p(w) = \prod_j p(w_j)$ 。我们看到这些先验可以表示成高斯尺度混合 (GSM)： $w_j \sim N(0, \tau_j^2)$ ，其中 τ_j^2 的先验在表 13.2 中给出。借助这些潜在方差，我们可把模型写成 $\tau_j^2 \rightarrow w_j \rightarrow y \leftarrow X$ 。于是可以用 EM 做 MAP 估计：在 E 步推断 $p(\tau_j^2 | w_j)$ ，在 M 步用 y, X, τ 来估计 w 。这个 M 步要么是封闭形式的加权 ℓ_2 优化（高斯尺度混合），要么是加权 ℓ_1 优化（拉普拉斯尺度混合）。我们也讨论过如何在这些模型中做贝叶斯推断而不仅仅是求 MAP。

本节讨论一种替代思路：基于 II 型极大似然（经验贝叶斯，EB）。具体地，我们积分掉 w ，对 τ 最大化边缘似然。该 EB 过程可以通过 EM 实现，也可通过一种重加权 ℓ_1 的方案实现（下面会解释）。在估计出这些方差之后，把它们代回去计算权重的后验均值 $E[w | \hat{\tau}, D]$ ；颇为意外的是（考虑到先验是高斯），结果是**（近似）稀疏**的，原因见下文。

在神经网络语境中，这个方法叫自动相关性确定 (ARD)（MacKay 1995b; Neal 1996；见 16.5.7.5 节）。在本章的线性模型语境，这个方法叫稀疏贝叶斯学习 (SBL)（Tipping 2001）。把 ARD/SBL 与线性模型的基函数展开结合，得到相关向量机 (RVM)，见 14.3.2 节。

13.7.1 线性回归中的 ARD

我们以线性回归来说明 (GLM 的 ARD 需要 Laplace 或其它近似)。讨论 ARD/SBL 时的习惯记号是：权重精度 $\alpha_j = 1/\tau_j^2$ ，观测精度 $\beta = 1/\sigma^2$ （不要把这里的 β 和统计里表示回归系数的 β 混淆）。假设模型为

$$p(y | x, w, \beta) = N(y | w^\top x, 1/\beta) \quad (13.147)$$

$$p(w) = N(w | 0, A^{-1}) \quad (13.148)$$

其中 $A = \text{diag}(\alpha)$ 。边缘似然可以解析计算：

$$p(y | X, \alpha, \beta) = \int N(y | Xw, \beta I_N) N(w | 0, A) dw \quad (13.149)$$

$$= N(y | 0, \beta I_N + XA^{-1}X^\top) \quad (13.150)$$

$$= (2\pi)^{-N/2} |C_\alpha|^{-1/2} \exp(-\frac{1}{2} y^\top C_\alpha^{-1} y) \quad (13.151)$$

其中

$$C_\alpha = \beta^{-1} I_N + X A^{-1} X^\top \quad (13.152)$$

把它与 spike-and-slab 的式 (13.13) 对比：除去第二项里缺少的 $\beta = 1/\sigma^2$ 因子外，形式相同，只是把二元的 $\gamma_j \in \{0, 1\}$ 换成了连续的 $\alpha_j \in \mathbb{R}_+$ 。取对数后目标为

$$\ell(\alpha, \beta) = -\frac{1}{2} \log p(y \mid X, \alpha, \beta) = \log |C_\alpha| + y^\top C_\alpha^{-1} y \quad (13.153)$$

为正则化，我们可给每个精度放共轭先验 $\alpha_j \sim \text{Ga}(a, b)$, $\beta \sim \text{Ga}(c, d)$ 。修正后的目标为

$$\ell(\alpha, \beta) = -\frac{1}{2} \log p(y \mid X, \alpha, \beta) + \sum_j \log \text{Ga}(\alpha_j \mid a, b) + \log \text{Ga}(\beta \mid c, d) \quad (13.154)$$

$$= \log |C_\alpha| + y^\top C_\alpha^{-1} y + \sum_j (a \log \alpha_j - b \alpha_j) + c \log \beta - d \beta \quad (13.155)$$

这在做 α, β 的贝叶斯推断时很有用 (Bishop and Tipping 2000)。但在进行** (II型) 点估计**时，我们会用不适当先验 $a = b = c = d = 0$ ，这会给出最大的稀疏性。

下面描述如何对 (α, β) 优化 (关于精度 α, β)。这相当于在 spike-and-slab 里寻找最可能的 γ 设定的代理问题，进而与 ℓ_0 正则化密切相关。特别地，(Wipf et al. 2010) 证明：式 (13.153) 的目标函数比 ℓ_0 目标局部极小点更少，因此更易优化。一旦得到了 α, β ，就可计算参数的后验

$$p(w \mid D, \hat{\alpha}, \hat{\beta}) = N(\mu, \Sigma) \quad (13.156)$$

$$\Sigma^{-1} = \hat{\beta} X^\top X + A \quad (13.157)$$

$$\mu = \hat{\beta} \Sigma X^\top y \quad (13.158)$$

我们一边计算 w 的后验，一边促进稀疏，这就是“稀疏贝叶斯学习”这一名称的来源。不过，由于有很多方式能“既稀疏又贝叶斯”，即使在这里是线性模型，我们仍用“ARD”这个术语。（另外，SBL 只在系数数值上“做贝叶斯”，而没有表达对相关特征集合的不确定性，而这通常更受关注。）

13.7.2 稀疏性从何而来？

如果 $\hat{\alpha}_j \approx 0$ ，那么 $\hat{w}_j \approx \hat{w}_j^{\text{mle}}$ ，因为把 w_j 拉向 0 的高斯先验的精度为 0 (相当于无约束)。但如果 $\hat{\alpha}_j \approx \infty$ ，先验就非常确信 $w_j = 0$ ，于是该特征 j 是“无关”的，后验均值会有 $\hat{w}_j \approx 0$ 。因此不相关的特征会被自动“关掉/剪枝”。

下面给出一个直观论证 (Tipping 2001)：考虑只有 2 个训练样本的一维线性回归，令 $X = x = (x_1, x_2)$, $y = (y_1, y_2)$ 。如图 13.20，把 x 与 y 画成平面向量。假设该特征对预测 y 无用，即 x 与 y 几乎正交。看 α 改变时边缘似然 $p(y | x, \alpha, \beta) = N(y | 0, C)$ 的变化，其中

$$C = \frac{1}{\beta} I + \frac{1}{\alpha} xx^\top \quad (13.159)$$

当 α 有限时，后验沿着 x 的方向被拉长 (图 13.20a)。而当 $\alpha = \infty$ 时， $C = \frac{1}{\beta} I$ 是球形 (图 13.20b)。若保持 $|C|$ 不变，后一种在观测点 y 处的密度更大，因此是更优解。换言之，边缘似然会“惩罚”那种 α_j 很小但 $X_{:,j}$ 又不相关的情形，因为这会浪费概率质量。从奥卡姆剃刀角度，更简洁的是去掉冗余维度。

13.7.3 与 MAP 估计的联系

ARD 看起来与前面章节的 MAP 法很不同：ARD 是积分掉 α 的反面——我们积分 w 而优化 α 。因为在后验里，参数 w_j 会因“解释相互排斥”而相关，估计 α_j 时会借用全部特征的信息，不仅是第 j 个。因此有效先验 $p(w | \hat{\alpha})$ 不是可分解的，而且还依赖数据 D (以及 σ^2)。不过 (Wipf & Nagarajan 2007) 证明，ARD 可视作如下的 MAP 问题：

$$\hat{w}_{ARD} = \arg \min_w \frac{\beta}{2} \|y - Xw\|_2^2 + g_{ARD}(w) \quad (13.160)$$

$$g_{ARD}(w) = \min_{\alpha \geq 0} \sum_j \alpha_j w_j^2 + \log |C_\alpha| \quad (13.161)$$

证明基于凸分析，这里略去。此外 (Wipf & Nagarajan 2007; Wipf et al. 2010) 还证明：用非可分解先验做 MAP 比任何可分解先验都更好，意义是：非可分解目标的局部极小更少，同时其全局最优与 ℓ_0 目标的全局最优一致——而 ℓ_1 正则虽然没有局部极小，却不具备这种性质。

通俗解释

它在干什么？

ARD/SBL 不再像 Lasso 那样直接给每个 w_j 加 ℓ_1 惩罚，而是给每个权重一个方差（或精度） τ_j^2 / α_j 。

- α_j 大 \Rightarrow 先验坚信 w_j 应该非常接近 0 \Rightarrow 这个特征“无关”。
- α_j 小 \Rightarrow 先验几乎不约束 w_j \Rightarrow 让数据来决定它的大小。

怎么做？

1. 先把 w 积分掉，得到只与 α, β 有关的边缘似然；

2. 通过 EM 或其它办法最大化这个边缘似然 (这叫II型极大似然/经验贝叶斯) ;
3. 得到 $\hat{\alpha}, \hat{\beta}$ 后, 计算 w 的后验均值 μ 作为估计;
4. 许多 α_j 会被推到很大 (趋向 ∞) , 于是对应的 w_j 非常接近 0 \Rightarrow 自然稀疏。

为什么会自动稀疏?

把每个特征的输入向量 $X_{:,j}$ 与目标 y 看成平面向量:

- 若 $X_{:,j}$ 与 y 几乎没关系 (近似正交) , 让 α_j 有限会把概率“拉长”到一个**不必要的方向**, 降低在 y 处的密度;
- 把 α_j 设得很大等于**关掉**这条方向, 反而能在 y 处取得更高的密度 (更大边缘似然) 。

这就是“自动相关性确定”——模型自己把无用特征的权重关掉。

它与 Lasso 有啥不同?

- Lasso: 对所有系数**统一加** $\lambda|w_j|$ 的惩罚, 容易把大的真系数也**压小 (有偏)** ;
- ARD: 每个系数有自己的“强度旋钮” α_j , 对大的、重要的系数**少罚**, 对不重要的系数**重罚到 0**, 更像是**数据自适应**的可分/不可分先验组合;
- 理论上, ARD对应的优化等价于一个**非可分解先验的 MAP**, 其目标相较 ℓ_0/ℓ_1 有**更少的坏极小**, 更容易优化到好解。

用法 (线性回归) 一眼看懂版

- 设定初值: α_j 全部相同、 β 来自噪声初估;
- 反复:
 1. 计算 C_α 与边缘似然梯度 (或 EM 的期望量) ;
 2. 更新 α, β (让与 y 不相关的特征的 $\alpha_j \uparrow$) ;
- 收敛后: 输出 w 的后验均值 μ , 把 α_j 特别大的特征视为**剪枝**掉了。

要点总结

- ARD/SBL 的核心: 积分掉 w , 对每个权重的**精度** α_j 做II型极大似然; $\alpha_j \rightarrow \infty$ 时, 自动把 w_j 逼到 0 \Rightarrow 稀疏。
- 边缘似然: $p(y|X, \alpha, \beta) = N(0, \beta^{-1}I + XA^{-1}X^\top)$, 优化目标 $\ell = \log |C_\alpha| + y^\top C_\alpha^{-1}y$ 。
- 估计完成后: $w | D, \hat{\alpha}, \hat{\beta} \sim N(\mu, \Sigma)$, $\mu = \hat{\beta} \Sigma X^\top y$ 。
- 为何稀疏: 让无关方向 (与 y 近乎正交的 $X_{:,j}$) 的 α_j 变大能提升边缘似然 (奥卡姆剃刀) , 因此会被**自动关闭**。
- 与 MAP 的关系: 等价于最小化 $\frac{\beta}{2} \|y - Xw\|^2 + g_{ARD}(w)$, 其中 g_{ARD} 是对 α 最优化后的**非可分解惩罚**; 这类目标较 ℓ_0 更易优化, 较任何可分解先验的 MAP 更少局部极

小。

- 与 Lasso 对比：**ARD 是自适应、贝叶斯、非可分解的稀疏化；Lasso 是固定、凸、可分解的 ℓ_1 惩罚，可能产生偏差并对相关特征处理欠佳。
- 实践提示：**需要矩阵运算（可能较重），通常对数据标准化更稳定；可用 EM、重加权 ℓ_1 或直接梯度法来更新 α, β 。

如果你愿意，我可以把这套更新步骤写成伪代码/NumPy 代码，或者给一个小数据的演示来直观看到 α_j 变大的“剪枝”过程。

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

13.7.4 Algorithms for ARD

In this section, we review several different algorithms for implementing ARD.

13.7.4.1 EM algorithm

The easiest way to implement SBL/ARD is to use EM. The expected complete data log likelihood is given by

$$\begin{aligned} Q(\alpha, \beta) &= E \log N(y|Xw, \sigma^2 I) + \log N(w|0, A^{-1}) \quad (13.162) \\ &= 1/2 E [-N \log \beta - \beta \|y - Xw\|^2 + j \log \alpha_j - \text{tr}(Aw w^T)] \\ &\quad + \text{const} \quad (13.163) \end{aligned}$$

$$\begin{aligned} &= 1/2 N \log \beta - \beta/2 \|y - X\mu\|^2 + \text{tr}(XTX\Sigma) + 1/2 j \log \alpha_j - 1/2 \text{tr}[A(\mu\mu^T + \Sigma)] + \text{const} \quad (13.164) \end{aligned}$$

where μ and Σ are computed in the E step using Equation 13.158. Suppose we put a $G(a, b)$ prior on α_j and a $G(c, d)$ prior on β . The penalized objective becomes

$$Q(\alpha, \beta) = Q(\alpha, \beta) + j(a \log \alpha_j - b \alpha_j) + c \log \beta - d \beta \quad (13.165)$$

Setting $dQ/d\alpha_j = 0$ we get the following M step:

$$\alpha_j = 1 + 2a E w_j^2 + 2b = 1 + 2a m_j^2 + \sum_{jj} + 2b \quad (13.166)$$

If $\alpha_j = \alpha$, and $a = b = 0$, the update becomes

$$\alpha = D E [w^T w] = D \mu^T \mu + \text{tr}(\Sigma) \quad (13.167)$$

The update for β is given by

$$\beta - 1 \text{ new} = \|y - X\mu\|^2 + \beta - 1 j(1 - \alpha_j \sum_{jj}) + 2d N + 2c \quad (13.168)$$

13.7.4.2 Fixed-point algorithm

A faster and more direct approach is to directly optimize the objective in Equation 13.155. One can show (Exercise 13.3) that the equations $d\alpha_j/d\alpha_j = 0$ and $d\beta/d\beta = 0$ lead to the following fixed point updates:

$$\alpha_j \leftarrow \gamma_j + 2a_m^2 j + 2b \quad (13.169)$$

$$\beta - 1 \leftarrow \|y - X\mu\|^2 + 2d_N - j \gamma_j + 2c \quad (13.170)$$

$$\gamma_j = 1 - \alpha_j \Sigma_j \quad (13.171)$$

The quantity γ_j is a measure of how well-determined w_j is by the data (MacKay 1992). Hence $\gamma = j \gamma_j$ is the effective degrees of freedom of the model. See Section 7.5.3 for further discussion.

Since α and β both depend on μ and Σ (which can be computed using Equation 13.158 or the Laplace approximation), we need to re-estimate these equations until convergence. (Convergence properties of this algorithm have been studied in (Wipf and Nagarajan 2007).) At convergence, the results are formally identical to those obtained by EM, but since the objective is non-convex, the results can depend on the initial values.

13.7.4.3 Iteratively reweighted L1 algorithm

Another approach to solving the ARD problem is based on the view that it is a MAP estimation problem. Although the log prior $g(w)$ is rather complex in form, it can be shown to be a non-decreasing, concave function of $|w_j|$. This means that it can be solved by an iteratively reweighted L1 problem of the form

$$w^{t+1} = \operatorname{argmin}_w NLL(w) + j \lambda(t) j |w_j| \quad (13.172)$$

In (Wipf and Nagarajan 2010), the following procedure for setting the penalty terms is suggested (based on a convex bound to the penalty function). We initialize with $\lambda(0) j = 1$, and then at iteration $t + 1$, compute $\lambda(t+1) j$ by iterating the following equation a few times:

$$\lambda_j \leftarrow X_{:,j} \sigma^2 I + X_{:,j} \operatorname{diag}(1/\lambda_j) \operatorname{diag}(|w^{(t+1)}_j|) - 1 X_{:,j}^T - 1 X_{:,j}^2 \quad (13.173)$$

We see that the new penalty λ_j depends on all the old weights. This is quite different from the adaptive lasso method of Section 13.6.2. To understand this difference, consider the noiseless case where $\sigma^2 = 0$, and assume $D = N$. In this case, there are $D = N$ solutions which perfectly reconstruct the data, $Xw = y$, and which have sparsity $\|w\|_0 = N$; these are called basic feasible solutions or BFS. What we want are solutions that satisfy $Xw = y$ but which are much sparser than this. Suppose the method has found a BFS. We do not want to increase the penalty on a weight just because it is small (as in adaptive lasso), since that will just reinforce our current local optimum. Instead, we want to increase the penalty on a weight if it is small

and if we have $\|w(t+1)\| < N$. The covariance term $(X\text{diag}(1/\lambda_j)\text{diag}(|w(t+1)|)) - 1$ has this effect: if w is a BFS, this matrix will be full rank, so the penalty will not increase much, but if w is sparser than N , the matrix will not be full rank, so the penalties associated with zero-valued coefficients will increase, thus reinforcing this solution (Wipf and Nagarajan 2010).

13.7.5 ARD for logistic regression

Now consider binary logistic regression, $p(y|x,w) = \text{Ber}(y|\text{sigm}(w^T x))$, using the same Gaussian prior, $p(w) = N(w|0, A^{-1})$. We can no longer use EM to estimate α , since the Gaussian prior is not conjugate to the logistic likelihood, so the E step cannot be done exactly. One approach is to use a variational approximation to the E step, as discussed in Section 21.8.1.1. A simpler approach is to use a Laplace approximation (see Section 8.4.1) in the E step. We can then use this approximation inside the same EM procedure as before, except we no longer need to update β . Note, however, that this is not guaranteed to converge.

An alternative is to use the techniques from Section 13.7.4.3. In this case, we can use exact methods to compute the inner weighted L1 regularized logistic regression problem, and no approximations are required.

已思考 29s

下面先把你给出的段落完整翻译（含公式编号，不遗漏），然后用通俗语言详细解释，最后给出要点总结。

翻译

13.7.4 ARD 的算法

本节回顾若干实现 ARD 的不同算法。

13.7.4.1 EM 算法

实现 SBL/ARD 最简单的方法是使用 EM。期望的完整数据对数似然为

$$Q(\alpha, \beta) = E[\log N(y | Xw, \sigma^2 I) + \log N(w | 0, A^{-1})] \quad (13.162)$$

$$= \frac{1}{2} E[N \log \beta - \beta \|y - Xw\|^2 + \sum_j \log \alpha_j - \text{tr}(Aww^T)] + \text{const} \quad (13.163)$$

$$= \frac{1}{2}N \log \beta - \frac{\beta}{2}(\|y - X\mu\|^2 + \text{tr}(X^\top X \Sigma)) + \frac{1}{2} \sum_j \log \alpha_j - \frac{1}{2} \text{tr}[A(\mu\mu^\top + \Sigma)] \quad \text{(13.164)}$$

其中 μ, Σ 在 E 步用式 (13.158) 计算。设在 α_j 上放 $\text{Ga}(a, b)$ 先验、在 β 上放 $\text{Ga}(c, d)$ 先验。则带惩罚的目标为

$$Q'(\alpha, \beta) = Q(\alpha, \beta) + \sum_j (a \log \alpha_j - b\alpha_j) + c \log \beta - d\beta \quad (13.165)$$

令 $\frac{\partial Q'}{\partial \alpha_j} = 0$ 得到如下 M 步：

$$\alpha_j = \frac{1+2a}{E[w_j^2] + 2b} = \frac{1+2a}{m_j^2 + \Sigma_{jj} + 2b} \quad (13.166)$$

若 $\alpha_j = \alpha$ 且 $a = b = 0$, 更新式变为

$$\alpha = \frac{D}{E[w^\top w]} = \frac{D}{\mu^\top \mu + \text{tr}(\Sigma)} \quad (13.167)$$

β 的更新为

$$\beta_{\text{new}}^{-1} = \frac{\|y - X\mu\|^2 + \beta^{-1} \sum_j (1 - \alpha_j \Sigma_{jj}) + 2d}{N + 2c} \quad (13.168)$$

13.7.4.2 不动点算法

一种更快、更直接的方法是直接优化式 (13.155)。可证明 (练习 13.3) 令 $\frac{d}{d\alpha_j} = 0$ 、 $\frac{d}{d\beta} = 0$ 得到如下不动点更新：

$$\alpha_j \leftarrow \frac{\gamma_j + 2a}{m_j^2 + 2b} \quad (13.169)$$

$$\beta^{-1} \leftarrow \frac{\|y - X\mu\|^2 + 2d}{N - \sum_j \gamma_j + 2c} \quad (13.170)$$

$$\gamma_j = 1 - \alpha_j \Sigma_{jj} \quad (13.171)$$

量 γ_j 衡量了由数据对 w_j 的“确定程度” (MacKay 1992)。因此 $\gamma = \sum_j \gamma_j$ 是模型的**有效自由度**。见 7.5.3 节以获更多讨论。

由于 α, β 都依赖 μ, Σ (可用式 (13.158) 或 Laplace 近似计算)，需要重复估计直到收敛。

(该算法的收敛性质见 Wipf and Nagarajan 2007。) 在收敛时，结果在形式上与 EM 相同，但由于目标非凸，结果可能依赖初值。

13.7.4.3 迭代重加权 ℓ_1 算法

解决 ARD 的另一种方法基于把它视为一个 MAP 估计问题。尽管对数先验 $g(w)$ 的形式较复杂，但可证明它是 $|w_j|$ 的**单调不减、凹函数**。这意味着问题可以通过如下**迭代重加权**

ℓ_1 形式求解：

$$w^{t+1} = \arg \min_w \text{NLL}(w) + \sum_j \lambda_j^{(t)} |w_j| \quad (13.172)$$

在 (Wipf and Nagarajan 2010) 中，建议如下设定惩罚项（基于对惩罚函数的一个凸上界）。初始化 $\lambda_j^{(0)} = 1$ ，在第 $t+1$ 次迭代，用下式迭代若干次得到 $\lambda_j^{(t+1)}$ ：

$$\lambda_j \leftarrow (x_{:,j}^\top (\sigma^2 I + X \text{diag}(1/\lambda) \text{diag}(|w^{(t+1)}|)^{-1} X^\top)^{-1} x_{:,j})^{1/2} \quad (13.173)$$

可见新的惩罚 λ_j 依赖于旧的**所有**权重。这与 13.6.2 节的自适应 lasso 很不同。为理解这种差异，考虑**无噪声**情形 $\sigma^2 = 0$ ，并假设 $D > N$ 。此时存在 $\binom{D}{N}$ 个解精确满足 $Xw = y$ ，且稀疏度 $\|w\|_0 = N$ ；这些解称为**基本可行解** (BFS)。我们希望得到满足 $Xw = y$ 但**比这更稀疏**的解。假设方法当前找到一个 BFS。我们不希望仅仅因为某个权重很小就增加其惩罚（如自适应 lasso 中那样），那只会加固当前的局部最优。我们希望当某个权重很小且 $\|w^{(t+1)}\|_0 < N$ 时才增加惩罚。式中的协方差项

$$(X \text{diag}(1/\lambda) \text{diag}(|w^{(t+1)}|)^{-1} X^\top)^{-1}$$

正体现了这一点：若 w 是 BFS，该矩阵满秩，惩罚不会明显增大；若 w 比 N 还稀疏，该矩阵不满秩，于是与零系数相关的惩罚会增加，从而**强化**这种解 (Wipf and Nagarajan 2010)。

13.7.5 逻辑回归中的 ARD

现在考虑二元逻辑回归 $p(y | x, w) = \text{Ber}(y | \sigma(w^\top x))$ ，仍取同样的高斯先验 $p(w) = N(w | 0, A^{-1})$ 。此时不能再用 EM 来估计 α ，因为高斯先验与逻辑似然**不共轭**，E 步无法精确完成。一种方法是像 21.8.1.1 节那样在 E 步使用**变分近似**；更简单的方法是在 E 步使用**Laplace 近似**（见 8.4.1 节）。随后把该近似带入与先前相同的 EM 过程中（只是无需更新 β ）。但注意，这**并不保证收敛**。

另一种方法是使用 13.7.4.3 节的技术。在这种情况下，可以对内部的**加权 ℓ_1 逻辑回归子问题**使用**精确求解方法**，无需任何近似。

通俗解释

目的：给 ARD/SBL 提供几套可落地的求解器。

1) EM 实现

- **E 步：**在当前 α, β 下，计算 w 的后验 $N(\mu, \Sigma)$ （式 13.156–13.158）。
- **M 步：**用上一步的 μ, Σ 更新每个 α_j 与 β 。

- 直观地说：如果 $\mu_j^2 + \Sigma_{jj}$ (“这个系数的大小+不确定性”) 小，则 α_j 大，即更强地把 w_j 压向 0；反之亦然。
- β 由残差与不确定性共同决定。
- 重复直到收敛。实现简单，数值稳健，但可能比后面的办法慢。

2) 不动点更新

- 直接对边缘似然的对数目标求导=0，得到闭式的“自举”更新式 (13.169)–(13.171)。
- $\gamma_j = 1 - \alpha_j \Sigma_{jj}$ 可理解为“第 j 个系数被数据确定了多少”。 $\sum \gamma_j$ 就是模型的**有效自由度**。
- 反复用这些式子更新，通常比 EM 快。但因为问题**非凸**，结果会依赖初始化。

3) 迭代重加权 ℓ_1

- 把 ARD 看成一个带**凹惩罚**的 MAP 问题 \Rightarrow 可以外层迭代、内层解一次**带权 ℓ_1** （或其逻辑回归版本）。
- 每一轮用式 (13.173) 根据**整体协方差结构**更新每个权重的惩罚 λ_j 。
- 与“自适应 Lasso”不同：这里不是“谁小罚谁”，而是看当前解是否比基本可行解还**更稀疏**；若是，就**提高零系数的惩罚**来巩固更稀疏的解。对高维、追求极稀疏的场景常更给力。

4) 逻辑回归的 ARD

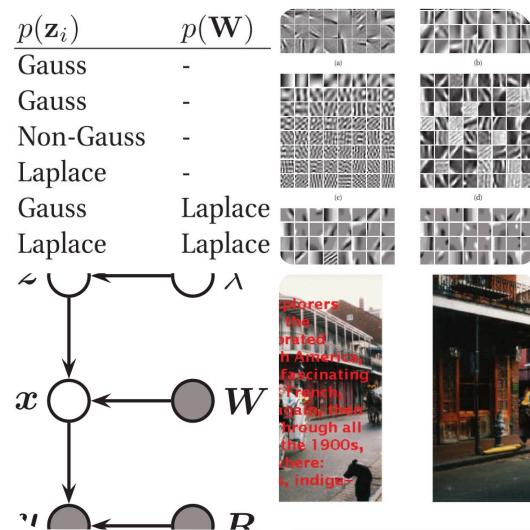
- 因为不共轭，标准 EM 做不了精确 E 步。两条路：
 - 变分/Laplace 近似出 μ, Σ 再照 EM 更新 α ；
 - 直接用**重加权 ℓ_1** 的思路，把内层换成**加权 ℓ_1 逻辑回归的精确求解**。第二种通常更稳。
-

要点总结

- 三类实现：**
 - EM**：E 步算 w 的高斯后验；M 步用 (13.166)(13.168) 更新 α, β 。稳健易写。
 - 不动点**：用 (13.169)–(13.171) 直接更新，通常更快； γ_j 反映有效自由度。
 - 重加权 ℓ_1** ：外层更新权重惩罚 λ_j ，内层解带权 ℓ_1 （或逻辑回归版本）。能逼出更稀疏的解。
- 非凸性**：目标非凸，解依赖初始化；但这些方法实践效果好。
- 逻辑回归**：用变分/Laplace 近似做 EM，或走重加权 ℓ_1 的“精准子问题”路线。

- **直觉:** α_j 大就把 w_j “关掉”；算法通过残差与不确定性自动调节 α_j ，实现**自适应稀疏**。

如果你想，我可以把三种算法分别写成伪代码/小示例，帮你对着数据跑一遍，直观看到 α_j 如何把不相关特征“剪枝”。



翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

Table 13.3 Summary of various latent factor models. A dash “-” in the $p(\mathbf{W})$ column means we are performing ML parameter estimation rather than MAP parameter estimation. Summary of abbreviations: PCA = principal components analysis; FA = factor analysis; ICA = independent components analysis; MF = matrix factorization.

Figure 13.21 Illustration of the filters learned by various methods when applied to natural image patches. (Each patch is first centered and normalized to unit norm.) (a) ICA. Figure generated by `icaBasisDemo`, kindly provided by Aapo Hyvärinen. (b) sparse coding. (c) PCA. (d) non-negative matrix factorization. (e) sparse PCA with low sparsity on weight matrix. (f) sparse PCA with high sparsity on weight matrix.

Figure 13.22 Schematic DGM for compressed sensing. We observe a low dimensional measurement y generated by passing x through a measurement matrix R , and possibly subject to observation noise with variance σ^2 . We assume that x has a sparse decomposition in terms of the

dictionary W and the latent variables z . The parameter λ controls the sparsity level.

Figure 13.23 An example of image inpainting using sparse coding. Left: original image. Right: reconstruction. Source: Figure 13 of (Mairal et al. 2008). Used with kind permission of Julien Mairal.

13.8 Sparse coding

So far, we have been concentrating on sparse priors for supervised learning. In this section, we discuss how to use them for unsupervised learning.

In Section 12.6, we discussed ICA, which is like PCA except it uses a non-Gaussian prior for the latent factors z_i . If we make the non-Gaussian prior be sparsity promoting, such as a Laplace distribution, we will be approximating each observed vector x_i as a sparse combination of basis vectors (columns of W); note that the sparsity pattern (controlled by z_i) changes from data case to data case. If we relax the constraint that W is orthogonal, we get a method called sparse coding. In this context, we call the factor loading matrix W a dictionary; each column is referred to as an atom. In view of the sparse representation, it is common for $L > D$, in which case we call the representation overcomplete.

In sparse coding, the dictionary can be fixed or learned. If it is fixed, it is common to use a wavelet or DCT basis, since many natural signals can be well approximated by a small number of such basis functions. However, it is also possible to learn the dictionary, by maximizing the likelihood

$$\log p(D|W) = \sum_{i=1}^N \log z_i N(x_i|Wz_i, \sigma^2 I)p(z_i)dz_i \quad (13.174)$$

We discuss ways to optimize this below, and then we present several interesting applications. Do not confuse sparse coding with sparse PCA (see e.g., (Witten et al. 2009; Journee et al. 2010)): this puts a sparsity promoting prior on the regression weights W , whereas in sparse coding, we put a sparsity promoting prior on the latent factors z_i . Of course, the two techniques can be combined; we call the result sparse matrix factorization, although this term is nonstandard. See Table 13.3 for a summary of our terminology.

13.8.1 Learning a sparse coding dictionary

Since Equation 13.174 is a hard objective to maximize, it is

common to make the following approximation:

$$\log p(D|W) \approx N \sum_{i=1}^N z_i \log N(x_i|Wz_i, \sigma^2 I) + \log p(z_i) \quad (13.175)$$

If $p(z_i)$ is Laplace, we can rewrite the NLL as

$$NLL(W, Z) = N \sum_{i=1}^N \|x_i - Wz_i\|_2^2 + \lambda \|z_i\|_1 \quad (13.176)$$

To prevent W from becoming arbitrarily large, it is common to constrain the 2 norm of its columns to be less than or equal to 1. Let us denote this constraint set by

$$C = \{W \in RD \times L \text{ s.t. } w^T j w_j \leq 1\} \quad (13.177)$$

Then we want to solve $\min_{W \in C, Z \in RN \times L} NLL(W, Z)$.

For a fixed z_i , the optimization over W is a simple least squares problem. And for a fixed dictionary W , the optimization problem over Z is identical to the lasso problem, for which many fast algorithms exist. This suggests an obvious iterative optimization scheme, in which we alternate between optimizing W and Z .

(Mumford 1994) called this kind of approach an analysis-synthesis loop, where estimating the basis W is the analysis phase, and estimating the coefficients Z is the synthesis phase. In cases where this is too slow, more sophisticated algorithms can be used, see e.g., (Mairal et al. 2010).

A variety of other models result in an optimization problem that is similar to Equation 13.176. For example, non-negative matrix factorization or NMF (Paatero and Tapper 1994; Lee and Seung 2001) requires solving an objective of the form

$$\min_{W \in C, Z \in RL \times N} \sum_{i=1}^N \|x_i - Wz_i\|_2^2 \text{ s.t. } W \geq 0, z_i \geq 0 \quad (13.178)$$

(Note that this has no hyper-parameters to tune.) The intuition behind this constraint is that the learned dictionary may be more interpretable if it is a positive sum of positive “parts”, rather than a sparse sum of atoms that may be positive or negative. Of course, we can combine NMF with a sparsity promoting prior on the latent factors. This is called non-negative sparse coding (Hoyer 2004). Alternatively, we can drop the positivity constraint, but impose a sparsity constraint on both the factors z_i and the dictionary W . We call this sparse matrix factorization. To ensure strict convexity, we can use an elastic net type penalty on the weights (Mairal et al. 2010) resulting in

$$\min_{W, Z} \sum_{i=1}^N \|x_i - Wz_i\|_2^2 + \lambda \|z_i\|_1 + \gamma \|w_j\|_1 \leq 1 \quad (13.179)$$

There are several related objectives one can write down.

For example, we can replace the lasso NLL with group lasso or fused lasso (Witten et al. 2009).

We can also use other sparsity-promoting priors besides the Laplace. For example, (Zhou et al. 2009) propose a model in which the latent factors z_i are made sparse using the binary mask model of Section 13.2.2. Each bit of the mask can be generated from a Bernoulli distribution with parameter π , which can be drawn from a beta distribution. Alternatively, we can use a non-parametric prior, such as the beta process. This allows the model to use dictionaries of unbounded size, rather than having to specify L in advance. One can perform Bayesian inference in this model using e.g., Gibbs sampling or variational Bayes. One finds that the effective size of the dictionary goes down as the noise level goes up, due to the Bayesian Occam's razor. This can prevent overfitting. See (Zhou et al. 2009) for details.

13.8.2 Results of dictionary learning from image patches

One reason that sparse coding has generated so much interest recently is because it explains an interesting phenomenon in neuroscience. In particular, the dictionary that is learned by applying sparse coding to patches of natural images consists of basis vectors that look like the filters that are found in simple cells in the primary visual cortex of the mammalian brain (Olshausen and Field 1996). In particular, the filters look like bar and edge detectors, as shown in Figure 13.21(b). (In this example, the parameter λ was chosen so that the number of active basis functions (non-zero components of z_i) is about 10.) Interestingly, using ICA gives visually similar results, as shown in Figure 13.21(a). By contrast, applying PCA to the same data results in sinusoidal gratings, as shown in Figure 13.21(c); these do not look like cortical cell response patterns.¹⁰ It has therefore been conjectured that parts of the cortex may be performing sparse coding of the sensory input; the resulting latent representation is then further processed by higher levels of the brain.

Figure 13.21(d) shows the result of using NMF, and Figure 13.21(e-f) show the results of sparse PCA, as we increase the sparsity of the basis vectors.

13.8.3 Compressed sensing

Although it is interesting to look at the dictionaries

learned by sparse coding, it is not necessarily very useful.

However, there are some practical applications of sparse coding, which we discuss below.

Imagine that, instead of observing the data $x \in RD$, we observe a low-dimensional projection of it, $y = Rx +$ where $y \in RM$, R is a $M \times D$ matrix, $M < D$, and ϵ is a noise term (usually Gaussian). We assume R is a known sensing matrix, corresponding to different linear projections of x . For example, consider an MRI scanner: each beam direction corresponds to a vector, encoded as a row in R . Figure 13.22 illustrates the modeling assumptions.

Our goal is to infer $p(x|y, R)$. How can we hope to recover all of x if we do not measure all of x ? The answer is: we can use Bayesian inference with an appropriate prior, that exploits the fact that natural signals can be expressed as a weighted combination of a small number of suitably chosen basis functions. That is, we assume $x = Wz$, where z has a sparse prior, and W is suitable dictionary. This is called compressed sensing or compressive sensing (Candes et al. 2006; Baraniuk 2007; Candes and Wakin 2008; Bruckstein et al. 2009).

For CS to work, it is important to represent the signal in the right basis, otherwise it will not be sparse. In traditional CS applications, the dictionary is fixed to be a standard form, such as wavelets. However, one can get much better performance by learning a domain-specific dictionary using sparse coding (Zhou et al. 2009). As for the sensing matrix R , it is often chosen to be a random matrix, for reasons explained in (Candes and Wakin 2008). However, one can get better performance by adapting the projection matrix to the dictionary (Seeger and Nickish 2008; Chang et al. 2009).

13.8.4 Image inpainting and denoising

Suppose we have an image which is corrupted in some way, e.g., by having text or scratches sparsely superimposed on top of it, as in Figure 13.23. We might want to estimate the underlying “clean” image. This is called image inpainting. One can use similar techniques for image denoising.

We can model this as a special kind of compressed sensing problem. The basic idea is as follows. We partition the image into overlapping patches, y_i , and concatenate them to form y . We define R so that the i 'th row selects

out patch i . Now define V to be the visible (uncorrupted) components of y , and H to be the hidden components. To perform image inpainting, we just compute $p(yH|yV, \theta)$, where θ are the model parameters, which specify the dictionary W and the sparsity level λ of z . We can either learn a dictionary offline from a database of images, or we can learn a dictionary just for this image, based on the non-corrupted patches. Figure 13.23 shows this technique in action. The dictionary (of size 256 atoms) was learned from 7×106 undamaged 12×12 color patches in the 12 mega-pixel image. An alternative approach is to use a graphical model (e.g., the fields of experts model (S. and Black 2009)) which directly encodes correlations between neighboring image patches, rather than using a latent variable model. Unfortunately such models tend to be computationally more expensive.

好的！下面我把你给出的这段内容逐句完整翻译，然后用通俗方式解释梳理，最后给出要点总结。为便于阅读，我按原来的小节顺序来。

图注翻译

表 13.3 各种潜在因子模型的总结。 $p(W)$ 一栏中的短横线 “-” 表示我们做的是极大似然 (ML) 参数估计而不是 MAP 参数估计。缩写：PCA=主成分分析；FA=因子分析；ICA=独立成分分析；MF=矩阵分解。

图 13.21 把不同方法应用到自然图像小块（每块先做去均值并归一化到单位范数）时学到的滤波器示意图。(a) ICA (由 `icaBasisDemo` 生成, Aapo Hyvärinen 提供)。(b) 稀疏编码 (sparse coding)。(c) PCA。(d) 非负矩阵分解 (NMF)。(e) 权重矩阵稀疏度较低的稀疏 PCA。(f) 权重矩阵稀疏度较高的稀疏 PCA。

图 13.22 压缩感知 (compressed sensing) 的示意贝叶斯网。我们观测到的是把 x 通过测量矩阵 R 后得到的低维观测 y , 并且可能叠加方差为 σ^2 的观测噪声。我们假设 x 可以用字典 W 和潜变量 z 的稀疏分解表示。参数 λ 控制稀疏程度。

图 13.23 使用稀疏编码做图像修补 (image inpainting) 的例子。左：原图。右：重建结果。来源：Mairal 等 (2008) 图 13, 经作者 Julien Mairal 授权使用。

正文翻译

13.8 稀疏编码 (Sparse coding)

到目前为止，我们一直专注于用于**监督学习**的稀疏先验。本节讨论如何把它们用于**无监督学习**。

在第 12.6 节我们讨论了 ICA，它类似 PCA，但对潜在因子 z_i 使用**非高斯先验**。如果把这种非高斯先验设为**促稀疏的**（例如拉普拉斯分布），我们就会把每个观测向量 x_i 近似为**基向量 (W 的列) 的稀疏组合**；注意稀疏模式（由 z_i 控制）会随着样本而改变。如果放松“W 必须正交”的约束，就得到一种称为**稀疏编码**的方法。在这里，把载荷矩阵 W 称为**字典 (dictionary)**；每一列称为一个**原子 (atom)**。由于稀疏表示，常见的是 $L > D$ ，此时我们称表示是**过完备 (overcomplete)**的。

在稀疏编码中，字典既可以**固定**也可以**学习**。若固定，常见做法是用小波或 DCT 基，因为许多自然信号可由少量这样的基函数很好地近似。不过也可以通过**极大化似然**

$$\log p(D|W) = \sum_{i=1}^N \log \int N(x_i | W z_i, \sigma^2 I) p(z_i) dz_i \quad (13.174)$$

来**学习字典**。下面我们讨论如何优化它，然后给出几个有趣的应用。请不要把**稀疏编码**与**稀疏 PCA**混淆（参见 Witten 等，Journee 等）：后者在**回归权重** W 上施加促稀疏先验；而稀疏编码在**潜在因子** z_i 上施加促稀疏先验。当然两者可以结合，我们把结果称为**稀疏矩阵分解 (sparse matrix factorization)**，尽管这个叫法并不规范。见表 13.3 以了解本文术语的汇总。

13.8.1 学习稀疏编码字典

由于式 (13.174) 难以直接最大化，通常做如下近似：

$$\log p(D|W) \approx \sum_{i=1}^N \max_{z_i} [\log N(x_i | W z_i, \sigma^2 I) + \log p(z_i)] \quad (13.175)$$

如果 $p(z_i)$ 是拉普拉斯分布，我们可以把负对数似然写成

$$\text{NLL}(W, Z) = \sum_{i=1}^N \frac{1}{2} \|x_i - W z_i\|_2^2 + \lambda \|z_i\|_1 \quad (13.176)$$

为避免 W 变得任意大，通常约束它的列的 ℓ_2 范数不超过 1。用集合

$$C = \{W \in \mathbb{R}^{D \times L} \text{ s.t. } w_j^T w_j \leq 1\} \quad (13.177)$$

表示该约束。于是我们要解 $\min_{W \in C, Z \in \mathbb{R}^{N \times L}} \text{NLL}(W, Z)$ 。对**固定的** z_i ，关于 W 的优化是一个简单的**最小二乘问题**；对**固定的字典** W ，关于 Z 的优化问题与 lasso 完全相同，现有很多快速算法。这启发了一个显而易见的**交替优化**方案：在 W 与 Z 之间交替更新。

(Mumford 1994 把这种方法称为“分析-合成 (analysis-synthesis) 循环”：估计字典 W 是“分析”，估计系数 Z 是“合成”。) 当该方法过慢时，可用更复杂的算法（例如 Mairal 等 2010）。

还有一些模型会导致与式 (13.176) 类似的优化问题。例如，**非负矩阵分解 (NMF)** 需要求解

$$\min_{W \in \mathbb{C}, Z \in \mathbb{R}^{L \times N}} \frac{1}{2} \sum_{i=1}^N \|x_i - Wz_i\|_2^2 \text{ s.t. } W \geq 0, z_i \geq 0 \quad (13.178)$$

(注意这没有超参数需要调。) 该约束背后的直觉是：若字典是若干正“部件”的正和，可能更易解释，而不是若干正负原子的稀疏和。当然也可以把 NMF 与对潜因子促稀疏的先验结合，这叫**非负稀疏编码** (Hoyer 2004)。

或者，去掉非负性约束，但同时在**因子** z_i 和**字典** W 上都施加稀疏约束。我们称之为**稀疏矩阵分解**。为保证严格凸性，可以在权重上使用类似弹性网的惩罚 (Mairal 等 2010)，得到

$$\min_{W, Z} \frac{1}{2} \sum_{i=1}^N \|x_i - Wz_i\|_2^2 + \lambda \|z_i\|_1 \text{ s.t. } \|w_j\|_2^2 + \gamma \|w_j\|_1 \leq 1 \quad (13.179)$$

还可以写下多个相关目标函数。例如，把 lasso 的 ℓ_1 惩罚替换为**组 lasso**或**融合 lasso** (Witten 等，2009)。

我们也可使用拉普拉斯之外的其他促稀疏先验。例如，Zhou 等 (2009) 提出让潜因子 z_i 通过**二值遮罩模型** (第 13.2.2 节) 变稀疏。遮罩的每一位可由参数为 π 的伯努利分布生成，而 π 又可来自 beta 分布。或者使用**非参数先验**，如**beta 过程**。这使模型可以使用**无界大小**的字典，而不必事先指定 L 。在该模型中可用 Gibbs 采样或变分 Bayes 进行推断。会发现：由于**贝叶斯奥卡姆剃刀**，有效字典大小会随噪声水平升高而减小，从而避免过拟合。细节见 Zhou 等 (2009)。

13.8.2 基于图像小块的字典学习结果

稀疏编码之所以引起广泛关注，一个原因是它解释了神经科学中的一个现象：把稀疏编码应用于**自然图像的小块**，得到的字典基向量看起来就像哺乳动物**初级视觉皮层简单细胞**的滤波器 (Olshausen & Field, 1996)。这些滤波器像**条纹与边缘检测器**，如图 13.21(b)。

(这个例子里， λ 设得使每个 z_i 的非零元素大约为 10 个。) 有趣的是，用 ICA 得到的效果在视觉上也很相似 (图 13.21(a))。而对同一数据使用 PCA 得到的是**正弦光栅** (图 13.21(c))，看起来不像皮层细胞的响应模式。因此有人猜想大脑皮层的某些部分可能在对感觉输入做**稀疏编码**，然后更高层再进一步处理该潜在表示。图 13.21(d) 是 NMF 的结果，图 13.21(e-f) 是**稀疏 PCA** 的结果，且随着对基向量稀疏度要求增加而变化。

13.8.3 压缩感知 (Compressed sensing)

尽管观察学习到的字典很有趣，但未必实用。不过稀疏编码确实有一些实际应用，下面讨论。

设想我们不是直接观测 $x \in \mathbb{R}^D$ ，而是观测它的一个**低维投影**

$$y = Rx + \varepsilon$$

其中 $y \in \mathbb{R}^M$ 、 R 是 $M \times D$ 矩阵、 $M \ll D$ ， ε 是噪声（通常高斯）。我们假设 R 是已知的**传感矩阵**，对应不同的线性投影。例如在 MRI 扫描仪中，每个射线方向对应一个向量，作为 R 的一行。图 13.22 给出了建模假设。

我们的目标是求 $p(x|y, R)$ 。既然没量完所有维度，怎么可能恢复完整的 x 呢？答案是：**用合适的先验做贝叶斯推断**，利用自然信号可表示为**少量适当基函数的线性组合**这一事实。也就是假设 $x = Wz$ ，其中 z 服从稀疏先验， W 是合适的字典。这就是**压缩感知** (Candes 等 2006; Baraniuk 2007; Candes & Wakin 2008; Bruckstein 等 2009)。

要让 CS 有效，关键在于选择**合适的基**，否则表示不稀疏。传统 CS 的字典常固定为**标准基**（例如小波）。不过，用稀疏编码**学习领域特定的字典**会明显更好 (Zhou 等 2009)。至于传感矩阵 R ，通常选择**随机矩阵**（缘由见 Candes & Wakin 2008）。但也可以**根据字典自适应地设计投影矩阵**，获得更好效果 (Seeger & Nickisch 2008; Chang 等 2009)。

13.8.4 图像修补与去噪

假设图像被破坏了，比如被稀疏地叠加了文字或划痕（图 13.23）。我们希望估计底层的“干净”图像，这叫**图像修补** (inpainting)。图像去噪也可用相似技术。

可以把它建模为一种特殊的压缩感知问题。基本思路如下：把图像分成**重叠的小块** y_i ，再把它们连接起来形成 y 。构造 R ，使第 i 行选择出第 i 个小块。把 y 的**可见（未损坏）分量**记为 V ，**隐藏（损坏）分量**记为 H 。**做修补就是计算** $p(y_H|y_V, \theta)$ ，**其中 θ 是模型参数（给出字典 W 以及 z 的稀疏度 λ ）**。**字典可以离线从一个图像库学，也可以仅用该图像的未损坏 patch 在线学习**。图 13.23 展示了该技术：字典有 256 个原子，是在一张 1200 万像素图中 700 万个未损坏的 12×12 彩色 patch 上学习的。另一种方法是使用**图模型**（例如 fields of experts; S. & Black 2009），它直接编码邻近图像块之间的相关性，而不是用潜变量模型。但这类模型往往**计算开销更大**。

通俗解释

- **稀疏编码**：把每个数据 x_i 表示成字典 W 的列（“原子”）的**少量线性组合** Wz_i 。向量 z_i 里大多数分量是 0（这由 ℓ_1 惩罚或拉普拉斯先验控制），只有少数是非零。

- **为什么要稀疏?** 更容易解释、更抗噪、能从少量测量恢复信号 (压缩感知) , 还与大脑视觉皮层的编码方式相呼应。
- **过完备字典:** 列数 L 大于数据维度 D ——字典很“丰富”, 更容易用少数原子组合出复杂形状。
- **怎么学字典?** 把问题拆成两步**交替最优化**:
 1. 固定字典 W , 对每个样本解一个 lasso 得到 z_i ;
 2. 固定所有 z_i , 解一个**最小二乘**得到 W , 并把每列 w_j 约束到单位球内。
重复这两步直到收敛。这样做简单、有效。
- **与其他方法的关系:**
 - ICA 也用稀疏 (非高斯) 先验, 但通常要求**分量独立且** (常) 保持某种正交/方差约束; 学到的视觉滤波器与稀疏编码类似。
 - PCA 追求方差最大, 得到的是**正弦/余弦状基**, 不稀疏。
 - NMF 额外要求 W, Z **非负**, 更易解释 (“部件加和”), 也可与稀疏先验结合。
 - **稀疏矩阵分解:** 同时对 Z 和 W 加稀疏约束, 可用弹性网式限制来保证问题良态。
- **压缩感知:** 只测量到低维 $y = Rx + \varepsilon$, 但如果 $x = Wz$ 在某个字典下很稀疏, 就能重建 x 。
 - 字典 W 很关键: 选得对, z 稀疏; 选不好, 稀疏性消失就难重建。
 - 传感矩阵 R 常取随机, 但也可以根据字典**自适应设计**, 提升重建质量。
- **图像修补/去噪:** 把图像分块, 已知的像素当“可见”, 未知的当“隐藏”, 用稀疏先验和学到的字典**推断缺失像素**。既能补字、补划痕, 也能去噪。

要点总结

- 稀疏编码 = 用**少量字典原子**表示数据: $x \approx Wz$, $\|z\|_0$ 小。
- 学字典的常用做法: **交替优化** (解 lasso 得 z , 解最小二乘得 W) , 并对 W 的列做 $\|w_j\|_2 \leq 1$ 约束。
- 可与多种思想组合: NMF (非负) 、**稀疏矩阵分解** (同时稀疏 W 与 Z) 、组/融合 lasso 等。
- 学到的视觉字典呈**边缘/条纹滤波器**, 与视觉皮层简单细胞相似; PCA 得到的是正弦状基。
- **压缩感知:** 在低维测量 $y = Rx + \varepsilon$ 的情况下, 若 x 在某字典下稀疏, 就能重建 x 。字典可固定 (如小波) 也可**数据驱动学习**。

- **图像修补/去噪是稀疏编码的实用场景：对重叠小块做稀疏重建，恢复缺失或受损像素。**

如果你想，我可以用一个小的数值例子，演示如何交替优化学出一个 1D/2D 的字典并做重建。

这一小节主要在讲什么？我不太懂

已思考 8s

这小节（13.8 稀疏编码 Sparse coding）在讲——**如何在无监督学习里，用“少量基向量的线性组合”来表示数据，并把这些基向量（字典）也一起学出来**。核心思想和做法是：

一句话版

把每个样本 x_i 表示成字典 W 的列向量的**很少几个**的线性组合：

$$x_i \approx W z_i, \quad z_i \text{ 很稀疏 (大多数分量为0)}$$

同时从数据里**学习**出这个字典 W 。

直观类比

想像有一盒“乐高零件”（字典的列），每张小图片/信号 x_i 都用**极少的**几块零件拼出来（稀疏系数 z_i ）。我们一边推测每张图用了哪些零件（求 z_i ），一边改进零件形状（学 W ），让总体拼得更好。

数学目标（但很容易理解）

把“重建误差 + 稀疏惩罚”最小化：

$$\min_{W,Z} \sum_{i=1}^N \frac{1}{2} \|x_i - W z_i\|_2^2 + \lambda \|z_i\|_1$$

- 第一项：拼出来的和原样本要像（误差小）。
- 第二项：用的零件要**少** (ℓ_1 促稀疏)。
- 为防止字典无限放大，约束每列 $\|w_j\|_2 \leq 1$ 。

算法：交替优化

1. 固定 W ，对每个样本解一个 lasso 得 z_i ；

2. 固定所有 z_i , 做一次**最小二乘**更新 W ;
两步反复直到收敛。 (慢时可用更高效的近似与加速方法。)
-

和 PCA / ICA / 稀疏PCA / NMF 的区别

- PCA: 学到的是最大方差方向, 基向量常像正弦/余弦栅格; 不追求“每个样本只用少量基”。
 - ICA: 给潜变量 z 用非高斯 (可稀疏) 先验, 但通常带独立/正交等约束; 学到的滤波器外观和稀疏编码相近。
 - 稀疏 PCA: 把稀疏性加在权重矩阵 W 上; 而稀疏编码把稀疏性加在系数 z 上。
 - NMF: 要求 $W, Z \geq 0$, 更偏“部件分解”; 也能再加稀疏先验 (非负稀疏编码)。
-

为什么有趣/有用

- 神经科学联系: 在自然图像小块上学到的字典, 长得像视觉皮层简单细胞的边缘/条纹检测器。
 - 压缩感知: 只测少量投影 $y = Rx + \varepsilon$, 若 $x = Wz$ 在某字典下很稀疏, 就能从 y 重建 x ; 字典可事先固定 (如小波) 或数据驱动学出来, 效果更好。
 - 图像修补/去噪: 把图像分块, 对缺失/被文字遮挡的像素用稀疏表示来推断恢复, 实际效果很好 (文中示例)。
-

小结要点

- 稀疏编码 = 学字典 W + 稀疏系数 z , 让每个样本由少数“原子”组合而成。
- 标准目标: 重建误差 + ℓ_1 稀疏惩罚; 常用交替最优化 (lasso \leftrightarrow 最小二乘)。
- 可扩展到 NMF、组/融合 lasso、稀疏矩阵分解 等变体。
- 与 PCA/ICA 的差异在于: 是否追求样本级的**稀疏使用**、是否对 W 施加正交/非负/稀疏约束。
- 典型应用: 压缩感知、图像修补/去噪; 并且与视觉皮层滤波器相呼应。

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 53s

下面给你一个“整章速览包”，把你前面贴过的 13 章内容（稀疏线性模型/变量选择）收成三块：①知识点清单 ②子章节之间怎么串起来 ③最后的章总结。——尽量短平快但不失重点。

① 知识点总结 (分点 | 关键词 | 用途)

- **子集选择 (l0) 与贝叶斯解释**

- 关键词：Spike&Slab、Bernoulli–Gaussian（二元掩码）、MAP、BIC、 $\|\gamma\|_0$ 、离散搜索
- 用途：直接做变量是否入模的决策；连接到经典“最优子集/前向后向/OMP”等贪心法与随机搜索（MCMC/随机探测）。

- **l1 正则 (Lasso) 基础**

- 关键词：Laplace 先验、MAP、软阈值、子梯度、KKT、几何解释（菱形球角）、正则路径、LARS
- 用途：把难的 l0 变成凸问题，得到稀疏解；可画整条路径做模型挑选；与岭回归/最优子集的对照理解偏差-方差权衡。

- **为什么 l1 会稀疏**

- 关键词：l1 球角点更易与误差等高线相交；与 l2（圆球）对比；OLS vs Ridge vs Lasso 的闭式阈值化（软/硬阈值）
- 用途：理解“设为 0”与“收缩”的来源，知道 lasso 是有偏的（大系数被压小）。

- **模型选择与稳定性**

- 关键词：交叉验证（偏向预测最优≠真模型）、去偏(debiasing)、稳定选择/BoLasso（自助法频率）、 λ_{\max}
- 用途：挑 λ 时区分“预测好”与“找真子集”；用重复抽样抑制假阳性。

- **l1 算法族**

- 关键词：坐标下降(shooting)、LARS/同伦（整条路径）、近端法/prox、投影梯度、ISTA/SpaRSA（BB 步长）、FISTA、EM（GSM）
- 用途：从小到超大规模、从单点解到整条路径，都有现成高效解法；GSM+EM给出统一贝叶斯视角。

- **扩展正则化**

- *Group Lasso*: $\sum_g \lambda_g \|w_g\|_2$ 或 $\|w_g\|_\infty$, GSM 解释 (一组一方差) ;
用途: 成组选择 (多任务/类别哑变量/同一特征跨任务) 。
 - *Fused Lasso*: $\|w\|_1 + \sum |w_{j+1} - w_j|$, TV/图引导;
用途: 邻接/图结构平滑+稀疏 (信号分段常数、图像去噪/去块) 。
 - *Elastic Net*: $\lambda_1 \|w\|_1 + \lambda_2 \|w\|_2^2$ (修正版去二次收缩) ;
用途: 相关特征成组进入、 $D > N$ 时可选 $> N$ 个特征、兼顾稳定性与稀疏性。
 - **非凸稀疏化**
 - 关键词: Bridge $\|w\|_b^b$ ($0 < b < 1$)、层次自适应 Lasso (HAL, 广义 t/双 Pareto 先验) 、Normal-Jeffreys、Horseshoe
 - 用途: 更尖的 0 峰+更重尾 → 更少假阳性、更小收缩偏差 (近似硬阈值) , 代价是优化非凸/需EM或变分。
 - **ARD / SBL (经验贝叶斯)**
 - 关键词: Type-II ML、边缘似然 $p(y|X, \alpha, \beta)$ 、每维精度 α_j 、自动关掉不相关特征、EM/不动点/重加权 ℓ_1
 - 用途: 不对 w 做 MAP, 而是先估方差再取后验均值; 得到近似稀疏且局部极值更少, 线性/逻辑回归均可用。
 - **无监督的稀疏表示 (Sparse Coding)**
 - 关键词: 字典学习 $x \approx Wz$ 、 ℓ_1 约束在 z 、交替最优化、过完备字典、与 PCA/ICA/NMF 对比
 - 用途: 学习边缘/条纹等视觉原子; 压缩感知 ($y = Rx$ 反演) 、图像修补/去噪等实际工程。
-

② 子章节关联 (依赖 | 对比 | 递进)

- **依赖/铺垫**
 - 13.2 (l0/Spike&Slab) → 提出“真稀疏”的目标与离散搜索难度
 - 13.3 (l1 基础) → 用 Laplace 把 l0 放松成凸问题; 给出几何与子梯度阈值化
 - 13.4 (算法) → 为 13.3 的目标提供可扩展解法 (坐标/同伦/近端/EM)
- **对比/取舍**
 - Ridge vs Lasso vs Subset: 无稀疏/软稀疏/硬稀疏; 偏差-方差与稳定性
 - Lasso vs Elastic Net: 单纯稀疏 vs 稀疏+组效应、相关特征
 - Lasso vs HAL/非凸: 凸但有偏 vs 非凸更接近硬阈值
 - MAP (因子化先验) vs ARD (非因子化有效先验/Type-II) : 局部极值多 vs 少、但需边缘化

- 递进/扩展

- $\ell_1 \rightarrow$ Group/Fused/Elastic Net (融入结构与稳定性)
- 监督 (回归/分类) → 无监督 (字典学习/压缩感知/修补)

(可选) Mermaid 总览

```
flowchart TD A[10/Spike&Slab(13.2)] --> B[11 基础与最优性(13.3)] B --> C[算法: 坐标/同伦/近端/EM(13.4)] B --> D[扩展: Group/Fused/ElasticNet(13.5)] B --> E[非凸: Bridge/HAL/层次先验(13.6)] E --> D B --> F[ARD/SBL Type-II ML(13.7)] C --> F D --> G[稀疏编码/CS/修补(13.8)] E --> G F --> G
```

关系句

- "13.2 给出理想目标 (ℓ_0)，13.3 用 ℓ_1 做凸近似，13.4 提供效率可控的算法族。"
- "13.5 把结构先验 (组、邻接、混合惩罚) 引入到 ℓ_1 体系。"
- "13.6 通过**非凸层次先验**减小 ℓ_1 的收缩偏差。"
- "13.7 走经验贝叶斯路线，用边缘似然自动关掉不相关维 (ARD/SBL)。"
- "13.8 把稀疏思想用于无监督表示与感知任务 (字典学习、压缩感知、图像修补)。"

③ 本章总结 (把握全局与落地建议)

- 核心主题：**在高维/小样本设置里，通过**稀疏性**做变量选择与正则化，提升**泛化、可解释性与计算可行性**。
- 三条主要路线：**
 - $\ell_0/\text{子集选择}$** (最理想但离散困难)
 - $\ell_1/\text{凸近似}$** (Lasso 及其扩展: Group/Fused/ElasticNet; 算法成熟、工业首选)
 - 层次/经验贝叶斯** (Spike&Slab、HAL、ARD/SBL; 更强的选择一致性与偏差控制)
- 算法工具箱：**坐标下降适合大稀疏设计；LARS 得到整条路径；近端/投影梯度 (ISTA/FISTA/SpaRSA) 适合超大规模与广义损失；GSM+EM 提供统一贝叶斯实现。
- 实践要点：**
 - 明确目标：**预测优先** → 可用 CV 选 λ (可能更稠密)；**找真子集** → 稳定选择/BoLasso/层次先验/ARD 更合适。
 - 特征强相关或 $D > N \rightarrow$ Elastic Net / Group Lasso；有序/空间结构 → Fused/Graph-guided。
 - 关心偏差 (系数被过度缩小) → **去偏** 或考虑 **非凸/层次** 方法。
 - 资源受限/大规模 → **近端梯度** + warm start；需要整条路径 → LARS/glmnet。

- **外延应用：**把“稀疏先验 + 高效算法”迁移到**无监督表示**：字典学习、压缩感知与图像修补/去噪，连接到神经科学（简单细胞滤波器）与实际工程。

一句话收尾：本章从“选谁进模型”出发，给出了**从离散→凸→层次贝叶斯**的全景方案，并配齐了算法、扩展与应用，让你能在不同数据结构与目标（预测/解释）下，**稳准省地做稀疏建模**。

请用通俗易懂的话总结本章，使我能够理解

已思考 27s

下面把整章用“白话”说清楚，读完就能抓住重点、知道该怎么用。

这章在解决什么问题？

当特征很多、样本不多时，**不是每个特征都重要**。如果我们能只让**少数几个**起作用（其余系数变成 0），模型就更稳、更好解释，也更能泛化——这就叫**稀疏**。本章讲的就是：**怎样选特征并把无关系数压成 0**。

三条主路线（从理想到实用）

1. $\mathbb{I}0$ 子集选择（理想但难）

直接问“选不选”：每个特征一个开关（0/1）。这最贴近“真相”，但组合太多，计算上很难。贝叶斯里对应 **spike-and-slab** 先验（要么正好 0，要么是个分布）。

2. $\mathbb{I}1$ (Lasso, 最常用)

不再枚举开关，而是给所有系数收一个“罚金” $\sum|\mathbf{w}|$ 。

- **几何直觉：** $\mathbb{I}1$ 的“约束球”像菱形，角容易卡在坐标轴上 → 很多系数被推成 0（稀疏）。
- **计算：**是**凸优化**，有成熟算法；解是**软阈值**（小的直接变 0，大的也会被“缩小”，因此有偏）。
- **选多大罚金 λ ：**为了预测好一般用交叉验证；但它未必找回“真模型”，可配合**稳定选择/bolasso**，再对非零系数做**去偏**。

3. ARD / SBL (经验贝叶斯，自动开关)

给每个系数一个**专属方差**（或精度 α_j ）。先把 \mathbf{w} 积掉，直接用数据来估这些方差；不重要的特征会把 α_j 推到**很大**，等价于把对应系数**自动置 0**。

直觉：如果某特征方向和输出几乎无关，让它占“概率质量”反而拉低边缘似然，最优做法就是把它关掉。

优点：常比直接做 ℓ_0 的小坑更少，解更稳；线性/逻辑回归都能用。

算法怎么落地？

- **坐标下降**：一维一维地更新（快、实现简单）。
 - **LARS/同伦**：一口气算出整条正则路径（ λ 从大到小），代价≈一次最小二乘。
 - **近端/投影梯度**（ISTA、FISTA、SpaRSA）：大规模、通用损失函数的首选。
 - **EM (GSM 技巧)**：把 Laplace/组范数写成“高斯+随机方差”的层次模型，E 步估方差，M 步解二次/加权 ℓ_1 。
-

重要扩展（按数据结构加“常识”）

- **Group Lasso**：一组特征要么一起上、要么一起下（多任务、哑变量、同一词跨类别）。
 - **Fused Lasso / 图引导**：既稀疏又让**相邻位置相近**（时间序列分段常数、图像网格）。
 - **Elastic Net ($\ell_1 + \ell_2$)**：处理**强相关特征**，让它们**成组进入**；在 $D \gg N$ 时能选超过 N 个特征。
 - **非凸惩罚 (Bridge、HAL 等)**：在 0 处更尖、尾更重，**更接近硬阈值**，把无关项干净地砍掉且对大系数**更少压缩**，但优化更难。
-

正则路径与选择 λ

把每个系数随 λ 的变化画出来就是**正则路径**：

- 岭回归 (ℓ_2)：所有系数同时平滑增长，**不稀疏**。
 - Lasso：从全 0 开始，逐个“**点亮**”；路径分段线性，正好被 LARS 高效追踪。
- 实践上：预测优先 → 用 CV 选 λ ；要找真子集 → 配合稳定选择/层次先验/ARD。
-

超出监督学习：稀疏表示 (Sparse Coding)

把观测 x 表示成少量“字典原子”的线性组合 $x \approx Wz$ ，对 z 用 ℓ_1 。交替优化 W 和 z 。

- 学到的字典在图像上像**边缘/条纹滤波器**（与 V1 皮层简单细胞相似）。
- **压缩感知**：只量 $M \ll D$ 个线性投影 $y = Rx$ ，利用稀疏先验把 x 复原。
- **图像修补/去噪**：把坏掉的像素当“缺失”，用稀疏表示从好块推断坏块。

选型小抄（一句话指南）

- 特征很多、相关强 → Elastic Net (或 Group Lasso 若知道分组)。
- 需要“分段平滑”或有网格/图结构 → Fused/Graph Lasso。
- 追求解释性/真子集 → 稳定选择/BoLasso、非凸/HAL、或 ARD/SBL。
- 超大规模/任意损失 → 近端梯度 (FISTA)；要路径 → LARS/glmnet。
- 无监督表示/重建 → Sparse Coding/压缩感知。

一句话收尾

这章告诉我们：在高维里，“少即是多”。用 ℓ_1 /层次贝叶斯/ARD 等工具，把不必要的系数压成 0，仅保留**少量、可信的特征**，既让模型**更准**，也让它**更懂人**；再按数据结构加上组、邻接或 ℓ_2 ，就能既稳又稀疏地解决真实世界的问题。

这里的变量选择与正则化，其中变量是指什么，正则化是指什么

已思考 17s

好问题！这章里说的“**变量选择**”和“**正则化**”分别指：

变量 (variable) 指什么？

- 就是**自变量/特征/解释变量**——在数据矩阵 X 里的一列。
例子：预测房价时，面积、卧室数、学区评分、是否地铁旁等，每一项都是一个“变量”。
- 若做了特征工程（多项式、交互项、分箱、基函数、词袋、卷积字典原子等），**变换后的一列**也算一个变量。
- 在“组”场景（group lasso）里，**一组相关列**可被当成“一个变量组”。

变量选择 (variable selection) 是什么？

- 目标：**决定哪些变量要用，哪些不要。**
- 数学上就是让某些系数 $w_j = 0$ (不用)，留下少数非零 (要用)。
- 典型做法：

- 直接做子集选择 (ℓ_0 , 或贝叶斯的 spike-and-slab) ——最贴近“开关”，但计算很难；
- 用能产生稀疏解的正则化 (比如 L1/Lasso、Group Lasso、Fused Lasso、HAL/非凸)，在优化时**自动把不重要的系数推到 0**，等价完成变量选择。
- 作用：提高解释性、降方差、防过拟合。

正则化 (regularization) 是什么？

- 在损失函数后面加一个“**复杂度惩罚项**”，限制模型自由度、让估计更稳：

$$\min_w \text{Loss}(w) + \lambda \Omega(w)$$

其中 $\Omega(w)$ 是正则项， λ 控制强度。

- 常见正则项与效果：
 - L2 (Ridge)** : $\Omega(w) = \|w\|_2^2$ 。缩小系数 (shrinkage)，但一般不变 0，不做变量选择；对强相关特征更稳。
 - L1 (Lasso)** : $\Omega(w) = \|w\|_1$ 。既缩小又能把很多系数推到 0 → 产生稀疏 → 顺带完成变量选择。
 - Elastic Net**: L1+L2，既能选又能稳，适合强相关特征、多于样本的特征。
 - Group/Fused**: 在组或邻接结构上做惩罚，促使**整组**选入/剔除，或**相邻位置相近**。
 - 非凸惩罚 (Bridge、HAL 等)** : 在 0 点更尖、尾更重，**更接近硬阈值**，对大系数更少偏差，但优化更难。
- 作用：**控制复杂度、防过拟合、改善数值稳定性**；有的还能**诱导稀疏**从而实现变量选择。

两者的关系 (一句话)

- 变量选择是目标** (选出该用的特征)；
- 正则化是手段** (通过加惩罚来控制复杂度)。
- L1/Group/Fused/非凸**这类正则化 **既是正则化 又能实现变量选择**；而 L2 只正则、不选择。

小例子

- 房价模型：如果用 OLS 全上，容易过拟合。
 - 用 Ridge：所有系数都被“缩小”，但都在；
 - 用 Lasso：自动把“是否地铁旁”“学区评分”等保留，把“是否顶楼”“是否有壁炉”等可能直接压成 0，**实现变量选择**；

- 用 Elastic Net：当“面积”“客厅面积”“卧室数”强相关时，会成组进入且更稳。

要点

- 变量 = X 的列（或列的组/变换后列）。
- 变量选择 = 让多数系数变 0，挑出关键特征。
- 正则化 = 在损失后加惩罚控制复杂度；某些正则化（L1 系）还能自动完成变量选择。