

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 Sparse linear models

13.1 Introduction

We introduced the topic of feature selection in Section 3.5.4, where we discussed methods for finding input variables which had high mutual information with the output. The trouble with this approach is that it is based on a myopic strategy that only looks at one variable at a time. This can fail if there are interaction effects. For example, if $y = \text{xor}(x_1, x_2)$, then neither x_1 nor x_2 on its own can predict the response, but together they perfectly predict the response. For a real-world example of this, consider genetic association studies: sometimes two genes on their own may be harmless, but when present together they cause a recessive disease (Balding 2006).

In this chapter, we focus on selecting sets of variables at a time using a model-based approach. If the model is a generalized linear model, of the form $p(y|x) = p(y|f(w^T x))$ for some link function f , then we can perform feature selection by encouraging the weight vector w to be sparse, i.e., to have lots of zeros. This approach turns out to offer significant computational advantages, as we will see below.

Here are some applications where feature selection/sparsity is useful:

- In many problems, we have many more dimensions D than training cases N . The corresponding design matrix is short and fat, rather than tall and skinny. This is called the small N , large D problem. This is becoming increasingly prevalent as we develop more high throughput measurement devices. For example, with gene microarrays, it is common to measure the expression levels of $D \sim 10,000$ genes, but to only get $N \sim 100$ such examples. (It is perhaps a sign of the times that even our data seems to be getting fatter...) We may want to find the smallest set of features that can accurately predict the response (e.g., growth rate of the cell) in order to prevent overfitting, to reduce the cost of building a diagnostic device, or to help with scientific insight into the problem.
- In Chapter 14, we will use basis functions centered on

the training examples, so $\varphi(x) = [\kappa(x, x_1), \dots, \kappa(x, x_N)]$, where κ is a kernel function. The resulting design matrix has size $N \times N$. Feature selection in this context is equivalent to selecting a subset of the training examples, which can help reduce overfitting and computational cost. This is known as a sparse kernel machine.

- In signal processing, it is common to represent signals (images, speech, etc.) in terms of wavelet basis functions. To save time and space, it is useful to find a sparse representation of the signals, in terms of a small number of such basis functions. This allows us to estimate signals from a small number of measurements, as well as to compress the signal. See Section 13.8.3 for more information.

Note that the topic of feature selection and sparsity is currently one of the most active areas of machine learning/ statistics. In this chapter, we only have space to give an overview of the main results.

已思考 42s

下面先给出逐字完整翻译（不漏译），随后是通俗解释，最后要点小结。

译文（完整、不遗漏）

13 稀疏线性模型

13.1 引言

我们在第 3.5.4 节介绍了特征选择这一主题，在那里我们讨论了寻找与输出具有高互信息的输入变量的方法。这个方法的问题在于，它基于一种短视（myopic）的策略，即每次只看一个变量。如果存在交互效应，这种方法就可能失败。举例来说，如果 $y = \text{xor}(x_1, x_2)$ ，那么无论是 x_1 还是 x_2 单独都无法预测响应，但它们放在一起却能完美预测响应。一个现实世界的例子是遗传关联研究：有时两个基因单独存在可能无害，但当同时存在时会导致隐性疾病（Balding 2006）。

在本章中，我们专注于使用一种基于模型的方法，一次选择一组变量。如果模型是广义线性模型，形式为 $p(y | x) = p(y | f(w^\top x))$ ，其中 f 是一个链接函数，那么我们可以通过鼓励权重向量 w 是稀疏的（即包含大量的零）来执行特征选择。事实证明，这种方法在计算上具有显著优势，我们将在下面看到。

下面是一些特征选择/稀疏性有用的应用场景：

- 在许多问题中，我们的维度 D 远多于训练样本数 N 。相应的设计矩阵是“矮而胖”的，而不是“高而瘦”的。这被称为“小 N 、大 D ”问题。随着我们开发更多高通量测量设备，这种情况变得越来越普遍。例如，在基因芯片中，常见的是测量 $D \sim 10,000$ 个基因的表达水

平，但只得到 $N \sim 100$ 个这样的样本。（也许这正是时代的写照：连我们的数据都在“变胖”……）我们可能希望找到能够准确预测响应（例如细胞的生长速率）的最小特征集合，以防止过拟合、降低构建诊断装置的成本，或帮助我们在该问题上获得科学洞见。

- 在第 14 章中，我们将使用以训练样本为中心的基函数，因此 $\phi(x) = [\kappa(x, x_1), \dots, \kappa(x, x_N)]$ ，其中 κ 是一个核函数。由此得到的设计矩阵大小为 $N \times N$ 。在这种情境下，特征选择等价于选择训练样本的一个子集，这有助于减少过拟合和计算成本。这被称为稀疏核机器。
- 在信号处理里，常见的做法是用小波基函数来表示信号（图像、语音等）。为了节省时间和空间，找到信号的稀疏表示（只用少量这类基函数）是有用的。这使我们能够用少量观测来估计信号，同时还能对信号进行压缩。参见第 13.8.3 节了解更多信息。

请注意，特征选择和稀疏性目前是机器学习/统计学中最活跃的话题之一。在本章中，我们只能概述主要结果。

通俗解释（逐段讲清楚）

1) “只看单个变量”的局限与交互效应

- 早先的方法是：逐个变量检查“它和目标的关系强不强”（比如用“互信息”衡量）。
- 但现实里，**变量之间会配合**：单看每一个都不显眼，可一组合起来就很关键。
- 经典例子是 **XOR (异或)**：
 - 规则：两个输入不同则输出 1，相同则输出 0。
 - 小表：
 - $(0,0) \rightarrow 0; (1,0) \rightarrow 1; (0,1) \rightarrow 1; (1,1) \rightarrow 0.$
 - 单看 x_1 或 x_2 各自，都与 y 没有稳定关系；但一起就能完全决定 y 。
- 现实类比：**两个基因**各自都没事，但一起出现就触发隐性病。这说明“只看单个变量”的策略会漏掉“配合效应”。

2) 基于模型的一次性“成组选特征”

- 思路：不再逐个变量打分，而是**直接建一个可预测的模型**，让模型自己挑（保留）对预测重要的变量集合。
- 采用的模型：**广义线性模型 (GLM)**，它先做线性加权 $w^\top x$ ，再套一个“链接函数” f （例如逻辑回归里的 S 形函数），最后得到预测分布 $p(y | f(w^\top x))$ 。
- **怎么让模型自己“挑”变量？**
 - 让权重向量 w 变得**稀疏**：大多数权重被推到 0，只有少数非零。
 - 常见做法（延伸解释）：在训练目标里加上**L1 正则化 (Lasso)** 这类“稀疏惩罚项”，优化后很多系数自动为 0，于是相当于把那些特征“剔除”了。

- **好处 (计算层面) :**

- 训练: 稀疏优化有高效算法 (如坐标下降), 在“特征超多”的场景很吃香。
- 预测: 只用非零权重对应的少数特征, **计算更快、内存更省。**

3) 三类典型应用场景

- (a) 小 N 、大 D (样本少、特征巨多)

- 设计矩阵 (行=样本, 列=特征) 会“矮胖” (行数少、列数多)。
- 如基因芯片: 1 万个基因特征, 但只有 100 个样本。
- 这时容易**过拟合**: 模型可能“记住”训练集的噪声。
- 稀疏化可以:
 - 只保留**最少且最有用**的特征→降低过拟合;
 - 降低设备/检测成本 (只测关键指标即可);
 - 让科学家更容易**解读哪些因素真重要**。

- (b) 核方法/基函数以训练样本为中心

- 把输入 x 变成 $\phi(x) = [\kappa(x, x_1), \dots, \kappa(x, x_N)]$, 相当于用每个训练样本当“参考点”去度量与新样本的相似度。
- 于是设计矩阵变成 $N \times N$; 若样本多, 计算和内存都很重。
- **稀疏化**在这里表现为: 只保留一小部分“代表性训练样本” (有点像只留少量“原型点”或“支持点”), 从而**减少过拟合与计算成本**。这类方法统称为**稀疏核机器**。

- (c) 信号处理与小波稀疏表示

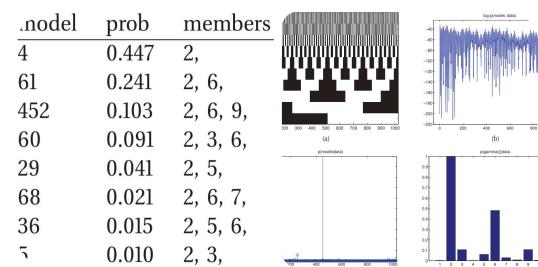
- 图像、语音等信号, 常能用小波基函数**紧凑表示**: 多数系数为 0, 只有少数显著。
- 益处:
 - **重建/估计更省样本**: 如果信号在某个基下本来就很稀疏, 只需较少的观测也能还原 (这与“压缩感知”思想一致)。
 - **压缩存储更高效**: 只保存那些非零或显著系数。
- 书中第 13.8.3 节会展开更多细节。

4) 为什么这是热门方向

- 现实数据越来越“高维” (指标多、分辨率高), 而可获得的标注样本却常常有限。
- 稀疏建模在**泛化、可解释性、效率**上三赢, 因此成为机器学习/统计学里非常活跃的研究主题。

要点小结

- 单变量筛选会漏交互：**像 XOR 或“两个基因一起致病”这类现象，要求我们成组看变量。
- 用模型做特征选择：**在 GLM 里让权重 稀疏化（大量 0），等价于自动“选特征”。常见技术是 L1 正则 (Lasso) 等。
- 计算更高效：**稀疏模型训练快、预测也只用少量特征，省时省内存。
- 三大应用：**
 - 小样本、超高维** (基因等) → 防过拟合、降成本、增可解释性；
 - 核方法** ($\phi(x) = [\kappa(x, x_i)]$) → 等价于选少量“代表性样本”，构成**稀疏核机器**；
 - 信号处理/小波** → 稀疏表示带来**少样本重建与压缩**。
- 整体趋势：**稀疏与特征选择是现代机器学习/统计的**核心热点**，兼顾预测准确、效率与可解释。



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 13.1 (a) All possible bit vectors of length 10 enumerated in Gray code order. (b) Score function for all possible models. (c) Posterior over all 1024 models. Vertical scale has been truncated at 0.1 for clarity. (d) Marginal inclusion probabilities. Figure generated by linregAllsubsetsGraycodeDemo.

13.2 Bayesian variable selection

A natural way to pose the variable selection problem is as follows. Let $\gamma_j = 1$ if feature j is “relevant”, and let $\gamma_j = 0$ otherwise. Our goal is to compute the posterior over models

$$p(\gamma|D) = e^{-f(\gamma)} \propto e^{-f(\gamma)} \quad (13.1)$$

where $f(\gamma)$ is the cost function:

$$f(\gamma) = -[\log p(D|\gamma) + \log p(\gamma)] \quad (13.2)$$

For example, suppose we generate $N = 20$ samples from a

$D = 10$ dimensional linear regression model, $y_i \sim N(w^T x_i, \sigma^2)$, in which $K = 5$ elements of w are non-zero. In particular, we use $w = (0.00, -1.67, 0.13, 0.00, 0.00, 1.19, 0.00, -0.04, 0.33, 0.00)$ and $\sigma^2 = 1$. We enumerate all $2^{10} = 1024$ models and compute $p(y|D)$ for each one (we give the equations for this below). We order the models in Gray code order, which ensures consecutive vectors differ by exactly 1 bit (the reasons for this are computational, and are discussed in Section 13.2.3).

The resulting set of bit patterns is shown in Figure 13.1(a). The cost of each model, $f(y)$, is shown in Figure 13.1(b). We see that this objective function is extremely “bumpy”. The results are easier to interpret if we compute the posterior distribution over models, $p(y|D)$. This is shown in Figure 13.1(c). The top 8 models are listed below:

The “true” model is {2, 3, 6, 8, 9}. However, the coefficients associated with features 3 and 8 are very small (relative to σ^2), so these variables are harder to detect. Given enough data, the method will converge on the true model (assuming the data is generated from a linear model), but for finite data sets, there will usually be considerable posterior uncertainty. Interpreting the posterior over a large number of models is quite difficult, so we will seek various summary statistics. A natural one is the posterior mode, or MAP estimate

$$\hat{y} = \operatorname{argmax} p(y|D) = \operatorname{argmin} f(y) \quad (13.3)$$

However, the mode is often not representative of the full posterior mass (see Section 5.2.1.3). A better summary is the median model (Barbieri and Berger 2004; Carvahlo and Lawrence 2007), computed using

$$\hat{y} = \{j : p(y_j = 1|D) > 0.5\} \quad (13.4)$$

This requires computing the posterior marginal inclusion probabilities, $p(y_j = 1|D)$. These are shown in Figure 13.1(d). We see that the model is confident that variables 2 and 6 are included; if we lower the decision threshold to 0.1, we would add 3 and 9 as well. However, if we wanted to “capture” variable 8, we would incur two false positives (5 and 7). This tradeoff between false positives and false negatives is discussed in more detail in Section 5.7.2.1.

The above example illustrates the “gold standard” for variable selection: the problem was sufficiently small (only

10 variables) that we were able to compute the full posterior exactly. Of course, variable selection is most useful in the cases where the number of dimensions is large. Since there are 2^D possible models (bit vectors), it will be impossible to compute the full posterior in general, and even finding summaries, such as the MAP estimate or marginal inclusion probabilities, will be intractable. We will therefore spend most of this chapter focussing on algorithmic speedups. But before we do that, we will explain how we computed $p(\gamma | D)$ in the above example.

已思考 38s

下面先给出**完整直译**（逐句、不漏译），随后是**通俗解释**，最后**要点小结**。

（说明：你给的文字里，式(13.1)的分母在排版中遗失了。我在译文中给出标准/应有的写法，并在括号里也标注了原文不完整的样字，方便对照。）

译文（完整、不遗漏）

图 13.1 (a) 以格雷码 (Gray code) 的顺序枚举的、长度为 10 的所有比特向量。 (b) 所有可能模型的评分函数。 (c) 1024 个模型的后验分布。为清晰起见，纵轴刻度在 0.1 处被截断。 (d) 边际纳入概率 (marginal inclusion probabilities)。该图由 `linregAllsubsetsGraycodeDemo` 生成。

13.2 贝叶斯变量选择

提出变量选择问题的一种自然方式如下。令 $\gamma_j = 1$ 表示第 j 个特征是“相关的”，否则令 $\gamma_j = 0$ 。我们的目标是计算关于模型的后验分布

$$p(\gamma | D) = \frac{e^{-f(\gamma)}}{\sum_{\gamma'} e^{-f(\gamma')}} \quad (13.1)$$

（原文排版为 $p(\gamma | D) = e^{-f(\gamma)} / \sum_{\gamma'} e^{-f(\gamma')}$ ，在你给的文字中分母丢失成 “ $p(\gamma | D) = e^{-f(\gamma)} \gamma e^{-f(\gamma')}$ ” 之类的残缺样式。）

其中 $f(\gamma)$ 为代价函数：

$$f(\gamma) = -[\log p(D | \gamma) + \log p(\gamma)] \quad (13.2)$$

例如，假设我们从一个 $D = 10$ 维的线性回归模型中生成 $N = 20$ 个样本， $y_i \sim N(w^\top x_i, \sigma^2)$ ，其中 w 的 $K = 5$ 个元素非零。具体地，我们使用 $w = (0.00, -1.67, 0.13, 0.00, 0.00, 1.19, 0.00, -0.04, 0.33, 0.00)$ 且 $\sigma^2 = 1$ 。我们枚举所有 $2^{10} = 1024$ 个模型，并为每一个计算 $p(\gamma | D)$ （相应公式见下文）。我们按

格雷码顺序对模型进行排序，这能保证相邻向量仅在 1 个比特上不同（这样做的原因与计算有关，将在第 13.2.3 节讨论）。

所得的比特模式集合见图 13.1(a)。每个模型的代价 $f(\gamma)$ 见图 13.1(b)。我们看到该目标函数极其“坑坑洼洼”（起伏很大）。如果改为计算模型的后验分布 $p(\gamma | D)$ ，结果更易于解释，如图 13.1(c) 所示。排名前 8 的模型如下所列：

model	prob	members
4	0.447	2
61	0.241	2, 6
452	0.103	2, 6, 9
60	0.091	2, 3, 6
29	0.041	2, 5
68	0.021	2, 6, 7
36	0.015	2, 5, 6
5	0.010	2, 3

“真实”的模型是 $\{2, 3, 6, 8, 9\}$ 。然而，与 σ^2 相比，与特征 3 和 8 相关的系数非常小，因此这些变量更难被检测到。只要数据足够多，该方法将会收敛到真实模型（假设数据由线性模型生成）；但对于有限数据集，通常会存在相当可观的后验不确定性。对大量模型的后验进行解释是相当困难的，因此我们将寻求各种摘要统计量。一个自然的做法是后验众数（MAP）估计

$$\hat{\gamma} = \arg \max_{\gamma} p(\gamma | D) = \arg \min_{\gamma} f(\gamma) \quad (13.3)$$

然而，众数往往不能代表整个后验质量（见第 5.2.1.3 节）。更好的摘要是**中位模型**（Barbieri and Berger 2004; Carvahlo and Lawrence 2007），其计算方式为

$$\hat{\gamma} = \{j : p(\gamma_j = 1 | D) > 0.5\} \quad (13.4)$$

这需要计算**后验边际纳入概率** $p(\gamma_j = 1 | D)$ 。它们显示在图 13.1(d) 中。我们看到，模型对变量 2 与 6 被纳入非常有信心；如果将判定阈值降低到 0.1，我们还会把 3 和 9 加进来。然而，如果我们还想“捕获”变量 8，就会引入两个**假阳性**（5 和 7）。关于**假阳性与假阴性**之间的权衡，在第 5.7.2.1 节有更详细的讨论。

上述例子展示了变量选择的“黄金标准”：问题足够小（只有 10 个变量），因此我们能够精

确计算完整的后验。当然，变量选择最有用的场景通常是维度很大。由于可能的模型数为 2^D (比特向量)，一般而言不可能计算完整后验；即便是找到摘要量（如 MAP 估计或边际纳入概率）也将是不可 tractable 的。因此，本章的大部分内容将聚焦于算法层面的加速。在此之前，我们将解释上例中我们是如何计算 $p(\gamma | D)$ 的。

通俗解释（逐段讲清）

图 13.1 在说什么？

- (a) 把 10 个特征是否选入用 10 位 0/1 表示 (1=选, 0=不选)，共有 $2^{10} = 1024$ 种。用格雷码排序，能保证相邻两种方案只差 1 位，便于做系统遍历或爬山搜索。
- (b) 每个模型都有一个“分数/代价” $f(\gamma)$ (越小越好)。曲线“坑坑洼洼”，说明优化很不平滑，直接找最优会很难。
- (c) 把代价转成后验概率 $p(\gamma | D)$ (通过式 13.1 的 softmax 形式)：能更直观看出“哪些模型更合理”。图里只把纵轴显示到 0.1，以免小柱子挤在一起看不清。
- (d) 边际纳入概率：对每个变量 j ，把所有包含它的模型的后验概率加起来，得到“变量 j 被纳入的总体可能性”。这是一个非常有用的、可解释的指标。

贝叶斯变量选择的基本思路

1. 用 $\gamma = (\gamma_1, \dots, \gamma_D)$ 表示“哪些特征被选入”。
2. 定义代价 $f(\gamma) = -[\log p(D | \gamma) + \log p(\gamma)]$ 。
 - 第一项是似然 (选了这些特征后，数据有多吻合)；
 - 第二项是先验 (我们事先偏好更简单或更稀疏的模型)。
3. 把代价通过式(13.1) 变成后验概率：代价越小、后验越大。
4. 有了 $p(\gamma | D)$ 后，我们可以：
 - 选MAP 模型 (后验最大的那个)；
 - 或者用中位模型：保留那些“被纳入概率超过 0.5”的特征。中位模型通常比单个 MAP 更“代表整体后验质量”。

为什么“真实模型”{2, 3, 6, 8, 9}没全被抓出来？

- 特征 3 和 8 的系数很小 (而噪声方差 $\sigma^2 = 1$ 不算小)，少量数据下就不容易把它们和噪声区分开。
- 数据量足够大时，方法会收敛到真相；但有限数据时存在后验不确定性：很多模型都有一定概率。
- 看 边际纳入概率：变量 2、6 的柱子很高，表示它们几乎肯定要选；3、9 中等；8 很矮。如果把阈值从 0.5 降到 0.1，你会多选到 3、9，但若还想抓到 8，就要付出两个假阳性 (把 5、7 错选进来) ——这就是查全率 vs. 准确率的经典权衡。

为什么要格雷码遍历？

- 相邻模型只改 1 位，有助于**重用计算**（例如快速更新似然/充分统计量），比随机跳来跳去更省时。这一点在第 13.2.3 节会展开。

为什么这个例子被称为“黄金标准”？

- 只有 10 个特征时，能把 1024 个模型都算一遍，得到“真后验”。
 - 但现实常是上万特征，模型数是 2^D 级的天文数字，**完整枚举不可能**，连求 MAP 或边际概率都可能算不动。
 - 所以后面章节会讲**加速技巧**（近似推断、贪心/采样/松弛/稀疏先验等）。
-

要点小结

- 用比特向量 γ 表示“选/不选”特征；目标是求后验 $p(\gamma \mid D)$ 。
- 代价 $f(\gamma) = -[\log p(D \mid \gamma) + \log p(\gamma)]$ ，代价小 \Leftrightarrow 后验大（式 13.1 为 softmax 形）。
- MAP 模型**：后验最大的一个；**中位模型**： $\{j : p(\gamma_j = 1 \mid D) > 0.5\}$ ，更能代表整体不确定性。
- 边际纳入概率**衡量“单个特征应否入模”的可信度；阈值越低，召回越高但假阳性更多。
- 小问题可枚举所有 2^D 个模型并精确计算；大问题做不到，需要**算法加速与近似**。
- 图 13.1 展示：代价表面很“起伏”、后验高度集中在少数模型上，以及变量 2、6 几乎必选，其它变量存在不确定与权衡。

MAP 模型、中位模型和边际纳入概率之间的关系

已思考 49s

下面把三者放在同一条主线上讲清楚：它们都是**同一个后验分布** $p(\gamma \mid D)$ 的不同“摘要方式”。

定义先摆好

- 边际纳入概率 (MIP)**

$$\pi_j = p(\gamma_j = 1 \mid D) = \sum_{\gamma: \gamma_j=1} p(\gamma \mid D)$$

它是“第 j 个变量被选入”的后验概率，对所有包含该变量的模型的后验质量求和得到。

- **MAP 模型 (后验众数)**

$$\hat{\gamma}_{\text{MAP}} = \arg \max_{\gamma} p(\gamma \mid D)$$

选概率最大的那一个具体模型（一组 0/1 决策）。

- **中位模型 (Median Probability Model, MPM)**

$$\hat{\gamma}_{\text{MPM}} = \{j : \pi_j > 0.5\}$$

对每个变量单独看：若“被选入”的概率超过 50%，就选它；否则不选。

关系与差异 (通俗图解)

1. 同源

- 三者都来自 $p(\gamma \mid D)$ 。
- MIP 是“按变量”把这个联合分布压扁；
- MPM 是对这些 MIP 做 0.5 阈值化；
- MAP 是从联合分布里挑一个峰值最多的“整套选择”。

2. 何时相同？

- 当后验高度集中（几乎所有概率都在某一个模型上）时：

$$p(\gamma^* \mid D) \approx 1 \Rightarrow \hat{\gamma}_{\text{MAP}} = \gamma^*, \quad \pi_j \approx \gamma_j^* \Rightarrow \hat{\gamma}_{\text{MPM}} = \gamma^*$$

这三者就一致或几乎一致。

3. 何时不同？

- 当后验分散在许多相近模型上时：
 - MAP 只能选其中一个峰值模型，常会少计或多计一些变量（取决于那个峰值模型恰好带了哪些变量）。
 - MPM 看“变量被包含的总体概率”，能把大家普遍同意的变量挑出来，而不被某个单独的模型牵着走。
- 例子：后验质量集中在 $\{x_1\}(0.45)$ 和 $\{x_2\}(0.45)$ 上，空模型 0.10。
 - MAP 会选 $\{x_1\}$ （或打平选 $\{x_2\}$ ）。

- $\pi_1 = \pi_2 = 0.45 < 0.5$, MPM 反而选**空模型**; 这反映了“两个变量是互为替代”的不确定性。
- 再例: 10 个高概率模型都包含 x_1 , 但其它变量各不相同、分摊了概率。
 - MAP 可能选到“ $x_1 +$ 一些偶然的附件”。
 - MPM 因 π_1 很高、其它很低, 只会选 x_1 , **更稳健**。

4. 与损失函数的关系

- 如果我们的损失是“每个变量选错记 1 分” (Hamming/逐坐标 0-1 损失), 则对每个 j 的最优决策是:
 - 选 1 的期望损失 = $1 - \pi_j$; 选 0 的期望损失 = π_j 。
 - 因而最佳阈值是 0.5 → MPM 是该损失下的贝叶斯最优。
- MAP 则是对“整套模型完全对/错”的 0-1 损失最优——这更苛刻, 因而在后验分散时不稳定。

5. 与预测性能

- 在线性回归等设置下 (如列近似正交、常见先验), 有结果表明 MPM 的预测往往不逊于甚至优于 MAP, 且更接近**贝叶斯模型平均 (BMA) **的预测; 直觉: MPM 保留“公认重要”的变量。
- 若目标是**解释/报告一个最可能的具体模型**, MAP 更符合口味; 若目标是**稳健预测或变量重要性判断**, MIP/MPM 更合适。

6. 计算层面

- MIP/MPM 只需要估计 π_j , 可用 MCMC 或近似推断;
 - MAP 需要在离散的 2^D 个模型里找全局最大, 常更难 (NP 难), 需要专门搜索/近似。
-

小结 (怎么用)

- MIP: 度量单个变量“应被纳入”的可信度 (0-1 概率)。
- MPM: 把 MIP 用 0.5 阈值化, 给出**按变量多数票的模型**; 对逐变量错分损失最优, 预测上常更稳健。
- MAP: 选后验概率最大的**单一模型**; 后验分散时可能不代表总体不确定性。
- **经验法则:**
 - 做**预测/稳健选择** → 看 MIP, 给 MPM (或设更低阈值换取召回)。
 - 做**解释/汇报一个模型** → 给 MAP, 但同时报告 MIP 以体现不确定性。

请用通俗易懂的话向我解释

已思考 29s

好，把这三样东西当成“三种看同一堆证据的法子”，不用公式也能懂。

想象有很多“模型方案”，每个方案就是一份**购物篮**：篮子里放着“被选进来的变量”。根据数据，每个篮子都有**支持票数**（概率大小）。

三个概念，像这么理解

- **MAP 模型（众数模型）**

就是**票数最多的那个篮子**。

直白：挑“最受欢迎的整套组合”。

- **边际纳入概率 (MIP)**

对**每个变量单独看**：把**所有**包含它的篮子的票数加起来，得到“这个变量被选上的总支持率”。

直白：统计“变量 A 被放进篮子里的概率有多大”。

- **中位模型 (Median/MPM)**

先算出每个变量的 MIP，再做个多数表决：

> 50% 就选，≤ 50% 就不选。

直白：按**每个变量的多数票**来装篮子。

他们的关系与差别

- 都是基于同一堆票（同一个后验分布），只是**视角不同**：

- MAP：看“整篮子的票”。

- MIP：看“某件商品的票”。

- 中位模型：把“某件商品的票”做多数表决，拼出一篮子。

- **什么时候三者差不多？**

当几乎所有票都投给同一个篮子时：MAP=中位模型，MIP 也都接近 0 或 1。

- **什么时候会不一样？**

当票**分散在好几个相近篮子**时：

- MAP 只能选一个赢家，可能把“偶然夹带”的变量一起选上。

- 中位模型更稳：只保留**大家普遍同意**的变量。

- MIP 能告诉你每个变量的“可信度”到底有多高。

两个一看就明白的小例子

1. 互相替代

45% 的票投给 $\{x_1\}$, 45% 投给 $\{x_2\}$, 10% 投给空篮。

- MAP: 选 $\{x_1\}$ (或 $\{x_2\}$, 并列时随便挑一个)。
- MIP: $x_1=0.45$, $x_2=0.45$ 。
- 中位模型: 两个都没过 0.5 → 选空篮。

含义: 我们不确定到底该选 x_1 还是 x_2 ; 中位模型选择“别乱加”。

2. 一个核心 + 若干附件

大多数高票篮子都包含 x_1 , 但其它变量各不相同。

- MAP: 可能选到 $\{x_1 + \text{某些附件}\}$ 。
- 中位模型: 只选 x_1 (因为它几乎人人同意)。
- MIP: 显示 x_1 概率很高, 其他都一般。

含义: 中位模型更“稳”和“简”。

实战怎么用

- 要一个可落地的“完整方案”(比如要上线就得定死一套): 给 MAP, 但同时把每个变量的 MIP 列出来说明不确定性。
- 要稳健预测或想知道“谁真的重要”: 看 MIP 排序, 再用中位模型(阈值默认 0.5; 想要更不漏可以把阈值降到 0.3 之类, 但会多一些假阳性)。
- 记住: 中位模型是在“每个变量选错同样疼”的情况下是最优多数决策; MAP 是在“整篮子一票定输赢”的标准下最优。

一句话总结

- MAP: 最可能的整套组合。
- MIP: 每个变量被选上的可信度。
- 中位模型: 把 MIP 做多数表决, 得到更稳、更代表共识的一套变量。

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。

13.2.1 The spike and slab model

The posterior is given by

$$p(\gamma|D) \propto p(\gamma)p(D|\gamma) \quad (13.5)$$

We first consider the prior, then the likelihood. It is common to use the following prior on the bit vector:

$$p(\gamma) = \prod_{j=1}^D \text{Ber}(\gamma_j | \pi_0) = \pi_0^{||\gamma||_0} (1 - \pi_0)^{D - ||\gamma||_0} \quad (13.6)$$

where π_0 is the probability a feature is relevant, and $||\gamma||_0 = \sum_{j=1}^D \gamma_j$ is the 0 pseudo-norm, that is, the number of non-zero elements of the vector. For comparison with later models, it is useful to write the log prior as follows:

$$\log p(\gamma | \pi_0) = ||\gamma||_0 \log \pi_0 + (D - ||\gamma||_0) \log(1 - \pi_0) \quad (13.7)$$

$$= ||\gamma||_0 (\log \pi_0 - \log(1 - \pi_0)) + \text{const} \quad (13.8)$$

$$= -\lambda ||\gamma||_0 + \text{const} \quad (13.9)$$

where $\lambda = \log 1 - \pi_0 / \pi_0$ controls the sparsity of the model.

We can write the likelihood as follows:

$$p(D|\gamma) = p(y|X, \gamma) = p(y|X, w, \gamma)p(w|\gamma, \sigma^2)p(\sigma^2)dw d\sigma^2 \quad (13.10)$$

For notational simplicity, we have assumed the response is centered, (i.e., $y = 0$), so we can ignore any offset term μ .

We now discuss the prior $p(w|\gamma, \sigma^2)$. If $\gamma_j = 0$, feature j is irrelevant, so we expect $w_j = 0$. If $\gamma_j = 1$, we expect w_j to be non-zero. If we standardize the inputs, a reasonable prior is $N(0, \sigma^2 w)$, where $\sigma^2 w$ controls how big we expect the coefficients associated with the relevant variables to be (which is scaled by the overall noise level σ^2). We can summarize this prior as follows:

$$p(w_j | \sigma^2, \gamma_j) = \delta_0(w_j) \text{ if } \gamma_j = 0 \quad N(w_j | 0, \sigma^2 \sigma_w) \text{ if } \gamma_j = 1 \quad (13.11)$$

The first term is a "spike" at the origin. As $\sigma^2 w \rightarrow \infty$, the distribution $p(w_j | \gamma_j = 1)$ approaches a uniform distribution, which can be thought of as a "slab" of constant height.

Hence this is called the spike and slab model (Mitchell and Beauchamp 1988). We can drop the coefficients w_j for which $w_j = 0$ from the model, since they are clamped to zero under the prior. Hence Equation 13.10 becomes the following (assuming a Gaussian likelihood):

$$p(D|\gamma) = N(y|X\gamma w, \sigma^2 I N)N(w|\gamma D_y, \sigma^2 \sigma_w^2 I D_y) p(\sigma^2) dw d\sigma^2 \quad (13.12)$$

where $D_y = ||\gamma||_0$ is the number of non-zero elements in γ .

In what follows, we will generalize this slightly by defining a prior of the form $p(w|\gamma, \sigma^2) = N(w|\gamma D_y, \sigma^2 \Sigma_y)$ for any positive definite matrix Σ_y . Given these priors, we can now compute the marginal likelihood. If the noise variance is known, we can write down the marginal likelihood (using Equation 13.151) as follows:

$$p(D|\gamma, \sigma^2) = N(y|X\gamma w, \sigma^2 I)N(w|\gamma D_y, \sigma^2 \Sigma_y) dw = N(y|0, C_\gamma) \quad (13.13)$$

$$C_\gamma = \sigma^2 X \gamma \Sigma_y X^\top + \sigma^2 I N \quad (13.14)$$

If the noise is unknown, we can put a prior on it and integrate it out. It is common to use $p(\sigma^2) = IG(\sigma^2 | a, b\sigma)$.

Some guidelines on setting a, b can be found in (Kohn et al. 2001). If we use $a = b = 0$, we recover the Jeffrey's prior, $p(\sigma^2) \propto \sigma^{-2}$. When we integrate out the noise, we get the following more complicated expression for the marginal likelihood (Brown et al. 1998):

$$p(D|\gamma) = p(y|\gamma, w\gamma, \sigma^2)p(w\gamma|\gamma, \sigma^2)p(\sigma^2)dw\gamma d\sigma^2 \quad (13.15)$$

$$\propto |X^T \gamma X\gamma + \Sigma - 1|^{-1/2} |\Sigma\gamma|^{-1/2} (2b\sigma + S(\gamma))$$

$$-(2a\sigma + N - 1)/2 \quad (13.16)$$

where $S(\gamma)$ is the RSS:

$$S(\gamma) = y^T y - y^T X\gamma (X^T \gamma X\gamma + \Sigma - 1)^{-1} X^T \gamma y \quad (13.17)$$

When the marginal likelihood cannot be computed in closed form (e.g., if we are using logistic regression or a nonlinear model), we can approximate it using BIC, which has the form

$$\log p(D|\gamma) \approx \log p(y|X, \hat{w}\gamma, \hat{\sigma}^2) - \|y\|^2 \log N \quad (13.18)$$

where $\hat{w}\gamma$ is the ML or MAP estimate based on $X\gamma$, and $\|y\|^2$ is the "degrees of freedom" of the model (Zou et al. 2007). Adding the log prior, the overall objective becomes

$$\log p(y|D) \approx \log p(y|X, \hat{w}\gamma, \hat{\sigma}^2) - \|y\|^2 \log N - \lambda \|y\|^2 + \text{const} \quad (13.19)$$

We see that there are two complexity penalties: one arising from the BIC approximation to the marginal likelihood, and the other arising from the prior on $p(\gamma)$.

Obviously these can be combined into one overall complexity parameter, which we will denote by λ .

已思考 29s

下面先给出逐字完整翻译（不漏译），随后是通俗解释，最后要点小结。

（说明：原文里有少量排版/编号小错误，如“Equation 13.151”多了一个数字。我在译文仍按原文呈现，并在解释中指出。）

译文（完整、不遗漏）

13.2.1 脉冲-板块 (spike and slab) 模型

后验为

$$p(\gamma | D) \propto p(\gamma)p(D | \gamma) \quad (13.5)$$

我们先讨论先验，然后讨论似然。对比特向量使用如下的先验是常见的：

$$p(\gamma) = \prod_{j=1}^D \text{Ber}(\gamma_j \mid \pi_0) = \pi_0^{\|\gamma\|_0} (1 - \pi_0)^{D - \|\gamma\|_0} \quad (13.6)$$

其中 π_0 是某个特征是相关的概率, $\|\gamma\|_0 = \sum_{j=1}^D \gamma_j$ 是 0 伪范数, 也就是该向量中非零元素的个数。为了与后面的模型比较, 把对数先验写成下面的形式是有用的:

$$\log p(\gamma \mid \pi_0) = \|\gamma\|_0 \log \pi_0 + (D - \|\gamma\|_0) \log(1 - \pi_0) \quad (13.7)$$

$$= \|\gamma\|_0 (\log \pi_0 - \log(1 - \pi_0)) + \text{const} \quad (13.8)$$

$$= -\lambda \|\gamma\|_0 + \text{const} \quad (13.9)$$

其中 $\lambda = \log \frac{1 - \pi_0}{\pi_0}$ 控制模型的稀疏性。我们可以把似然写成

$$p(D \mid \gamma) = p(y \mid X, \gamma) = \int p(y \mid X, w, \gamma) p(w \mid \gamma, \sigma^2) p(\sigma^2) dw d\sigma^2 \quad (13.10)$$

为简化记号, 我们假设响应已中心化 (即 $\bar{y} = 0$) , 因此可以忽略任何偏置项 μ 。现在讨论先验 $p(w \mid \gamma, \sigma^2)$ 。如果 $\gamma_j = 0$, 特征 j 不相关, 所以我们期望 $w_j = 0$ 。如果 $\gamma_j = 1$, 我们期望 w_j 非零。如果把输入做了标准化, 一个合理的先验是 $N(0, \sigma_w^2 \sigma_w^2)$, 其中 σ_w^2 控制我们对相关变量系数大小的预期 (并且按总体噪声水平 σ^2 缩放)。我们可以把该先验概括为:

$$p(w_j \mid \sigma^2, \gamma_j) = \begin{cases} \delta_0(w_j) & \text{若 } \gamma_j = 0 \\ N(w_j \mid 0, \sigma_w^2 \sigma_w^2) & \text{若 } \gamma_j = 1 \end{cases} \quad (13.11)$$

第一项是在原点的一个“脉冲” (spike)。当 $\sigma_w^2 \rightarrow \infty$ 时, 分布 $p(w_j \mid \gamma_j = 1)$ 趋近于一个均匀分布, 可以被看作是一个“板块” (slab) 具有常数高度。因此这被称为**脉冲-板块模型** (Mitchell and Beauchamp 1988)。对于满足 $w_j = 0$ 的系数, 我们可以把它们从模型中去掉, 因为在先验下它们被夹定为 0。因此式(13.10) 变为下面形式 (假设高斯似然) :

$$p(D \mid \gamma) = \iint N(y \mid X_\gamma w_\gamma, \sigma^2 I_N) N(w_\gamma \mid 0_{D_\gamma}, \sigma_w^2 \sigma_w^2 I_{D_\gamma}) p(\sigma^2) dw_\gamma d\sigma^2 \quad (13.12)$$

其中 $D_\gamma = \|\gamma\|_0$ 是 γ 中非零元素的个数。接下来我们把它稍作推广: 把先验定义为 $p(w \mid \gamma, \sigma^2) = N(w_\gamma \mid 0_{D_\gamma}, \sigma^2 \Sigma_\gamma)$, 其中 Σ_γ 为任意**正定**矩阵¹。在给定这些先验的情况下, 我们现在可以计算边际似然。若噪声方差已知, 我们可以 (使用 Equation 13.151) 把边际似然写为:

$$p(D \mid \gamma, \sigma^2) = \int N(y \mid X_\gamma w_\gamma, \sigma^2 I) N(w_\gamma \mid 0, \sigma^2 \Sigma_\gamma) dw_\gamma = N(y \mid 0, C_\gamma) \quad (13.13)$$

$$C_\gamma = \sigma^2 X_\gamma \Sigma_\gamma X_\gamma^\top + \sigma^2 I_N \quad (13.14)$$

如果噪声未知，我们可以给它加一个先验并把它积分掉。常见做法是使用 $p(\sigma^2) = \text{IG}(\sigma^2 | a_\sigma, b_\sigma)$ 。关于如何设置 a, b 的一些指导可以在 (Kohn et al. 2001) 中找到。如果使用 $a = b = 0$ ，我们得到 Jeffrey's 先验 $p(\sigma^2) \propto \sigma^{-2}$ 。当我们把噪声积分掉时，会得到下面更复杂的边际似然表达式 (Brown et al. 1998)：

$$p(D | \gamma) = \iint p(y | \gamma, w_\gamma, \sigma^2) p(w_\gamma | \gamma, \sigma^2) p(\sigma^2) dw_\gamma d\sigma^2 \quad (13.15)$$

$$\propto |X_\gamma^\top X_\gamma + \Sigma_\gamma^{-1}|^{-1/2} |\Sigma_\gamma|^{-1/2} (2b_\sigma + S(\gamma))^{-(2a_\sigma+N-1)/2} \quad (13.16)$$

其中 $S(\gamma)$ 是残差平方和 (RSS)：

$$S(\gamma) = y^\top y - y^\top X_\gamma (X_\gamma^\top X_\gamma + \Sigma_\gamma^{-1})^{-1} X_\gamma^\top y \quad (13.17)$$

当边际似然不能闭式求（例如我们使用逻辑回归或非线性模型时），可以用 BIC 近似，它的形式为

$$\log p(D | \gamma) \approx \log p(y | X, \hat{\gamma}, \hat{\sigma}^2) - \frac{\|\gamma\|_0}{2} \log N \quad (13.18)$$

其中 $\hat{\gamma}$ 是基于 X_γ 的 ML 或 MAP 估计， $\|\gamma\|_0$ 是模型的“自由度” (Zou et al. 2007)。把对数先验加上去，整体目标变为

$$\log p(y | D) \approx \log p(y | X, \hat{\gamma}, \hat{\sigma}^2) - \frac{\|\gamma\|_0}{2} \log N - \lambda \|\gamma\|_0 + \text{const} \quad (13.19)$$

可以看到，这里有两个复杂度惩罚：一个来自对边际似然的 BIC 近似，另一个来自 $p(\gamma)$ 的先验。显然，这两者可以合并成一个总的复杂度参数，我们用 λ 表示之。

通俗解释（逐步把每一块讲明白）

1) 变量是否入模 = 一个个“开关”

- $\gamma_j \in \{0, 1\}$ 就像第 j 个特征的开关。
- 先验(13.6)：每个开关独立地以概率 π_0 打开。打开的个数就是 $\|\gamma\|_0$ 。
- 写成对数(13.7–13.9)后可以看出：等价于给“选入变量的个数”加了一个罚分 $\lambda \|\gamma\|_0$ 。
 - π_0 越小 $\rightarrow \lambda = \log \frac{1-\pi_0}{\pi_0}$ 越大 \rightarrow 越偏好稀疏（少选）。

2) 脉冲-板块先验 = “要么精准为0，要么大致服从高斯”

- 若 $\gamma_j = 0$ ：我们**强力相信** $w_j = 0$ (δ_0 就是一根“针状”的概率集中在 0)。
- 若 $\gamma_j = 1$ ：我们只规定一个“大致范围”，让 w_j 服从以 0 为均值、方差为 σ_w^2 的高斯。

- $\sigma_w^2 \rightarrow \infty$ 时，高斯几乎不收缩，像一块“厚板”（slab）：表示“只要非零就行，具体多大我们不太在意”。
- 直观：这是一种**显式做变量选择**的先验——某些系数被钉死为 0，另外一些允许自由取值。

3) 把为零的系数直接删掉，剩下的统一记为 w_γ

- 这样，似然里就只涉及被保留的列 X_γ 和对应的系数 w_γ （式 13.12）。

4) 积分消掉 w_γ 、 $\sigma^2 \Rightarrow$ 得到“边际似然”

- 目的：比较不同子模型 γ 时，不想被某个具体的 w 干扰，就把 w **积分掉**。
- 若 σ^2 已知：积分有闭式解（13.13–13.14），得到 $\gamma \sim N(0, C_\gamma)$ 。
 - $C_\gamma = \sigma^2(X_\gamma \Sigma_\gamma X_\gamma^\top + I)$ 体现了“模型复杂度越大，协方差越大”，天然带来**奥卡姆剃刀**。
- 若 σ^2 未知：再给 σ^2 一个逆伽马先验 $IG(a, b)$ ，继续积分，得到（13.16）。
 - 式中出现了 $|X_\gamma^\top X_\gamma + \Sigma_\gamma^{-1}|$ （“体积/复杂度”项）和 $S(\gamma)$ （某种“残差平方和”，见 13.17）。
 - Σ_γ^{-1} 起到类似岭回归的“稳定/防共线”作用。

5) 算不出边际似然时 \Rightarrow 用 BIC 近似

- 对许多非高斯/非线性模型，积分没闭式。BIC 给出一个简洁替代（13.18）：
 - “拟合好坏”由把 w 设为极大似然/后验 (\hat{w}_γ) 后的对数似然衡量；
 - “复杂度”罚项是 $\frac{1}{2} \|\gamma\|_0 \log N$ （变量越多、数据越少，罚越重）。
- 再加上先验带来的 $\lambda \|\gamma\|_0$ 罚（13.19），总目标就是：
对数似然 – 两个复杂度罚项。这两个罚项可合并成一个“总的 λ ”来控制稀疏度。

6) 这些公式想表达的核心

- 我们不只是“把训练误差压到最小”，还要为每多选一个变量**付费**；
- 这种付费来自两处：
 1. **统计学上的奥卡姆剃刀**（边际化或 BIC 近似自然产生）；
 2. **先验偏好稀疏**（独立伯努利先验转成 ℓ_0 罚）。
- 结果：你得到的是一种**贝叶斯化的子集选择准则**，能在“拟合好”与“够简洁”之间做平衡。

备注：文中“使用 Equation 13.151”应为编号疏漏，通常指代前一式（13.15）或就近的边际化公式。

要点小结

- **先验 (γ 的独立伯努利)**：等价于在目标中加 $\lambda \|\gamma\|_0$ 罚； π_0 越小 $\rightarrow \lambda$ 越大 \rightarrow 越稀疏。
- **脉冲-板块先验 (w 的分量)：**
 - $\gamma_j = 0 \Rightarrow w_j$ 用 δ_0 钉死为 0 (spike)；
 - $\gamma_j = 1 \Rightarrow w_j \sim N(0, \sigma^2 \Sigma_\gamma)$ (slab)。
- **边际似然**：把 w, σ^2 积分掉比较不同子模型；已知噪声有闭式 (13.13–13.14)，未知噪声用 IG 先验可得 (13.16)，其中 $S(\gamma)$ 是剩余平方和 (13.17)。
- **算不出时用 BIC**: $\log p(D | \gamma) \approx \text{"对数似然 at } \hat{w} \text{"} - \frac{1}{2} \|\gamma\|_0 \log N$ 。
- **最终目标 (13.19)**：

$$\text{拟合好坏} = \underbrace{\frac{1}{2} \|\gamma\|_0 \log N}_{\text{BIC 复杂度}} - \underbrace{\lambda \|\gamma\|_0}_{\text{稀疏先验}}$$

两个惩罚都鼓励**少而精**的变量集合，可合并为一个总的 λ 控制强度。

- **直觉**：这套方法“要么果断为 0，要么自由估计”，通过边际化/罚项自动执行奥卡姆剃刀，在有限数据下能更稳健地做变量选择与预测。

这一小节主要在讲什么？我不太懂

已思考 21s

这小节在干一件事：

用贝叶斯的方法做“选变量”，核心工具叫“脉冲-板块 (spike & slab) 先验”。

用大白话说核心思路

- 把“第 j 个变量要不要用”当作一个**开关** γ_j ：开=1，用；关=0，不用。
- 给这排开关一个**先验**：每个开关独立以概率 π_0 打开（越小越偏好少开几个）。这等价于给“被选中的变量个数”加一个惩罚 (ℓ_0 罚)，促稀疏。
- 对系数 w_j 用**脉冲-板块先验**：
 - 如果开关关着 ($\gamma_j = 0$)，就把 w_j 钉死为 0 (“脉冲” spike)。
 - 如果开关开着 ($\gamma_j = 1$)，就让 w_j 大致服从一个零均值高斯，方差可调 (“板块” slab，表示“非零但多大不太限制”)。
- 然后把系数 w (以及可能的噪声方差 σ^2) **积分掉**，得到每个开关组合 γ 的“边际似然” $p(D | \gamma)$ ；再和先验 $p(\gamma)$ 相乘，就得到后验 $p(\gamma | D)$ ，从而知道**哪些变量组合更靠谱**。

- 有时这个积分算不出闭式，就用 BIC 给个近似：

评分 = 拟合好坏 - (常数 × 入选变量数)。这个“常数”里包含了样本量 $\log N$ 和一个可调的稀疏强度 λ 。

这段文字依次讲了哪些点

- 后验公式**: $p(\gamma | D) \propto p(\gamma)p(D | \gamma)$ 。
- γ 的先验**: 独立伯努利 (每个变量以 π_0 概率被选)，等价于 $-\lambda\|\gamma\|_0$ 的惩罚， $\lambda = \log \frac{1-\pi_0}{\pi_0}$ 。
- w 的先验 (spike & slab) :**
 - 关: $\delta_0(w_j)$ (必为 0)；
 - 开: $N(0, \sigma^2 \Sigma_\gamma)$ (允许非零，幅度由方差控制)。
- 边际化计算:**
 - 噪声已知 \Rightarrow 有闭式协方差 C_γ ；
 - 噪声未知 \Rightarrow 给 σ^2 一个逆伽马先验，积分后得到包含行列式和残差平方和 $S(\gamma)$ 的表达式。
- 算不动时的近似**: 用 BIC，把“拟合”与“复杂度（变量个数）”平衡起来；再加上先验惩罚，得到总目标 (式 13.19)。

为什么要这么做

- 自动稀疏**: 能把明显没用的变量直接压成 0。
- 奥卡姆剃刀**: 边际化/惩罚让“更简单的模型”天然占优，防止过拟合。
- 可解释**: 你既能得到一套变量组合，也能看到每个变量被选上的可信度（上一小节里的“边际纳入概率”）。

关键词一图懂

- spike (脉冲)** : 把系数精确固定为 0。
- slab (板块)** : 给非零系数一个宽松的高斯分布。
- π_0 : 先验里“变量会被选”的概率 (小 \Rightarrow 更稀疏)。
- λ : 稀疏强度，越大越少选。
- 边际似然** $p(D | \gamma)$: 比较不同变量集合的“证据”。
- BIC**: 算不出边际似然时的快捷打分，= 拟合 - (常数 × 变量数)。

一句话总结

这小节教你：用“脉冲-板块先验”把“选变量”做成贝叶斯问题——要么把某些系数钉死为 0 (spike)，要么让它们自由但受高斯约束 (slab)；通过边际化或 BIC 近似，把“拟合

好坏”和“越简越好”同时纳入一个评分里，用一个 λ 控制稀疏强度，从而稳健地挑出该选的变量。

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。

13.2.2 From the Bernoulli-Gaussian model to l0 regularization

Another model that is sometimes used (e.g., (Kuo and Mallick 1998; Zhou et al. 2009; Soussen et al. 2010)) is the following:

$$y_i|x_i, w, \gamma, \sigma^2 \sim N(\gamma_j w_j x_i, \sigma^2) \quad (13.20)$$

$$\gamma_j \sim Ber(\pi_0) \quad (13.21)$$

$$w_j \sim N(0, \sigma^2 w) \quad (13.22)$$

In the signal processing literature (e.g., (Soussen et al. 2010)), this is called the BernoulliGaussian model, although we could also call it the binary mask model, since we can think of the γ_j variables as “masking out” the weights w_j .

Unlike the spike and slab model, we do not integrate out the “irrelevant” coefficients; they always exist. In addition, the binary mask model has the form $\gamma_j \rightarrow y \leftarrow w_j$, whereas the spike and slab model has the form $\gamma_j \rightarrow w_j \rightarrow y$. In the binary mask model, only the product $\gamma_j w_j$ can be identified from the likelihood.

One interesting aspect of this model is that it can be used to derive an objective function that is widely used in the (non-Bayesian) subset selection literature. First, note that the joint prior has the form

$$p(\gamma, w) \propto N(w|0, \sigma^2 w) \pi ||\gamma||_0 (1 - \pi_0) D - ||\gamma||_0 \quad (13.23)$$

Hence the scaled unnormalized negative log posterior has the form

$$f(\gamma, w) = -2\sigma^2 \log p(\gamma, w, y|X) = ||y - X(\gamma \cdot w)||_2^2 + \sigma^2 \sigma^2 w ||w||_2^2 + \lambda ||\gamma||_0 + \text{const} \quad (13.24)$$

where

$$\lambda = 2\sigma^2 \log(1 - \pi_0 \pi_0) \quad (13.25)$$

Let us split w into two subvectors, $w-\gamma$ and $w\gamma$, indexed by the zero and non-zero entries of γ respectively. Since $X(\gamma \cdot w) = X\gamma w\gamma$, we can just set $w-\gamma = 0$. Now consider the case where $\sigma^2 w \rightarrow \infty$, so we do not regularize the

non-zero weights (so there is no complexity penalty coming from the marginal likelihood or its BIC approximation). In this case, the objective becomes
 $f(\gamma, w) = \|y - X\gamma w\|_2^2 + \lambda \|\gamma\|_0$ (13.26)

This is similar to the BIC objective above. Instead of keeping track of the bit vector γ , we can define the set of relevant variables to be the support, or set of non-zero entries, of w . Then we can rewrite the above equation as follows:

$$f(w) = \|y - Xw\|_2^2 + \lambda \|w\|_0 \quad (13.27)$$

This is called L0 regularization. We have converted the discrete optimization problem (over $\gamma \in \{0, 1\}^D$) into a continuous one (over $w \in \mathbb{R}^D$); however, the 0 pseudo-norm makes the objective very non smooth, so this is still hard to optimize. We will discuss different solutions to this in the rest of this chapter.

下面先给出**完整直译**（不漏译），然后是**通俗解释**，最后要点**小结**。

（说明：原文里有些排版小瑕疵，比如范数上的“ 2^2 ”应读作 $\|\cdot\|_2^2$ ，我在译文中按标准写法呈现。）

译文（完整）

13.2.2 从伯努利–高斯模型到 ℓ_0 正则化

另一个有时会被使用的模型（例如：(Kuo and Mallick 1998; Zhou et al. 2009; Soussen et al. 2010)）如下：

$$y_i | x_i, w, \gamma, \sigma^2 \sim N\left(\sum_j \gamma_j w_j x_{ij}, \sigma^2\right) \quad (13.20)$$

$$\gamma_j \sim Ber(\pi_0) \quad (13.21)$$

$$w_j \sim N(0, \sigma_w^2) \quad (13.22)$$

在信号处理文献中（如 (Soussen et al. 2010)），这被称为**伯努利–高斯模型**，尽管我们也可以称其为**二值掩码模型**，因为我们可以把 γ_j 变量看作是在“遮蔽 (masking out)”权重 w_j 。

与脉冲–板块模型不同，我们**不会把“不相关”的系数积分出去**；它们始终存在。此外，二值掩码模型的结构是 $\gamma_j \rightarrow y \leftarrow w_j$ ，而脉冲–板块模型的结构是 $\gamma_j \rightarrow w_j \rightarrow y$ 。在二值掩码模型中，**只有乘积** $\gamma_j w_j$ 能从似然中被识别出来。

该模型的一个有趣方面是：它可以用来推导一个在（非贝叶斯的）子集选择文献中被广泛使用的目标函数。首先，注意到联合先验的形式为

$$p(\gamma, w) \propto N(w | 0, \sigma_w^2 I) \pi_0^{\|\gamma\|_0} (1 - \pi_0)^{D - \|\gamma\|_0} \quad (13.23)$$

因此，按比例缩放的、未经归一化的负对数后验具有如下形式

$$f(\gamma, w) = -2\sigma^2 \log p(\gamma, w, y | X) = \|y - X(\gamma * w)\|_2^2 + \frac{\sigma^2}{\sigma_w^2} \|w\|_2^2 + \lambda \|\gamma\|_0 \quad (13.24)$$

其中

$$\lambda = 2\sigma^2 \log\left(\frac{1 - \pi_0}{\pi_0}\right) \quad (13.25)$$

把 w 按 γ 的 0/1 位置拆成两个子向量 $w_{-\gamma}$ 与 w_γ 。由于 $X(\gamma * w) = X_\gamma w_\gamma$ ，我们可以直接令 $w_{-\gamma} = 0$ 。现在考虑 $\sigma_w^2 \rightarrow \infty$ 的情形，此时我们不对非零权重做正则化（因此也就没有来自边际似然或其 BIC 近似的复杂度惩罚）。在这种情况下，目标函数变为

$$f(\gamma, w) = \|y - X_\gamma w_\gamma\|_2^2 + \lambda \|\gamma\|_0 \quad (13.26)$$

这与上面的 BIC 目标相似。不再显式跟踪比特向量 γ ，我们可以把“相关变量的集合”定义为 w 的**支撑集** (support)，即其非零位置的集合。于是我们可以把上式改写为

$$f(w) = \|y - Xw\|_2^2 + \lambda \|w\|_0 \quad (13.27)$$

这就叫做 ℓ_0 正则化。我们已经把离散优化问题（在 $\gamma \in \{0, 1\}^D$ 上）转换成了一个连续问题（在 $w \in \mathbb{R}^D$ 上）；然而，0 伪范数会让目标函数**非常不光滑**，因此这仍然很难优化。我们将在本章余下部分讨论不同的解决方案。

通俗解释（逐段讲透）

1) 这是个什么模型？为啥叫“伯努利–高斯 / 二值掩码”？

- 每个变量有两个成分：
 - γ_j 是 0/1 的“开关”（伯努利，概率 π_0 打开）。
 - w_j 是权重，服从高斯（均值 0、方差 σ_w^2 ）。
- 预测时， $\gamma_j w_j$ 乘在一起：
 - $\gamma_j = 0 \rightarrow$ 这个特征被“遮住”（mask 掉），贡献为 0；
 - $\gamma_j = 1 \rightarrow$ 放行，贡献就是 $w_j x_{ij}$ 。
- 所以它像给权重套了个**二值面罩**：有面罩（0）则完全不让过；摘面罩（1）就按高斯分布给个权重。

2) 它和“脉冲-板块 (spike & slab) ”有什么不同?

- spike & slab: γ_j 控制 w_j 的**分布形状** (不是直接乘) ; 结构是 $\gamma_j \rightarrow w_j \rightarrow y$ 。 $\gamma_j = 0$ 时 w_j 就被钉死为 0 (“脉冲”)。
- 这里 (伯努利-高斯 / 掩码) **不把无关系数积分掉**, w_j 总是存在; 结构是 $\gamma_j \rightarrow y \leftarrow w_j$ 。
- 关键差别: 在掩码模型里, 似然里只出现**乘积** $\gamma_j w_j$, 因此仅靠数据你分不清“是 $\gamma_j = 0$ 还是 $w_j = 0$ 导致的 0 贡献” (**不可辨识性**)。

3) 从贝叶斯到一个常见的 (非贝叶斯) 打分函数

- 写出联合先验 $p(\gamma, w)$:
 - w 是各向同性高斯 \rightarrow 带来一个 $\|w\|_2^2$ 的惩罚;
 - γ 是独立伯努利 \rightarrow 带来一个 $\|\gamma\|_0$ (选中变量个数) 的惩罚。
- 再把**负对数后验** (乘以 $2\sigma^2$ 做个尺度变换, 方便整理常数) 写出来 (式 13.24) :

$$\underbrace{\|y - X(\gamma.* w)\|_2^2}_{\text{拟合误差}} + \underbrace{\frac{\sigma^2}{\sigma_w^2} \|w\|_2^2}_{\text{权重 L2 罚}} + \underbrace{\lambda \|\gamma\|_0}_{\text{变量个数罚}} + \text{常数}$$

其中 $\lambda = 2\sigma^2 \log \frac{1-\pi_0}{\pi_0}$ 。

直觉: π_0 越小 (先验更稀疏) 、噪声 σ^2 越大 (怕过拟合), λ 就越大 \rightarrow 越惩罚“多选变量”。

4) 放宽对非零权重的约束, 得到“纯粹的子集选择”目标

- 令 $\sigma_w^2 \rightarrow \infty$: L2 罚 ($\sigma^2/\sigma_w^2\|w\|_2^2$) 消失 (不再缩小权重)。
- 把被屏蔽的位置的权重直接设 0 ($w_{-\gamma} = 0$), 只剩 $X_\gamma w_\gamma$ 。
- 此时目标退化为 (式 13.26) :

$$\|y - X_\gamma w_\gamma\|_2^2 + \lambda \|\gamma\|_0,$$

这就是典型的**子集选择**形式: “拟合误差 + 变量个数×罚分”。

- 再把“选哪些变量”不再通过 γ 明写, 而是用 w 的**支撑集** (非零位置) 表示, 就得到 (式 13.27) :

$$\|y - Xw\|_2^2 + \lambda \|w\|_0,$$

这就是著名的 ℓ_0 正则化。

5) 为何有用、也为何仍然困难?

- 有用: ℓ_0 正则直接鼓励**尽量少的非零系数**, 等价于直接做“挑最小变量子集”的目标, 很符合“少而精”的直觉, 也和前面 BIC 的“按变量子数惩罚”思想一致。
- 困难: $\|w\|_0$ 是**非凸、非光滑**的 (从 0 跳到 1 的离散计数), 优化上通常是**NP 难**。所以下续会介绍各种替代/近似 (如 L1/Lasso 替代、贪心法、局部搜索、概率近似)

等)。

要点小结

- 模型:** $\gamma_j \sim \text{Ber}(\pi_0)$, $w_j \sim N(0, \sigma_w^2)$, 预测用 $\sum_j \gamma_j w_j x_{ij}$; γ 像“掩码”, 决定某个权重是否起作用。
- 与 spike & slab 的区别:** 掩码模型里只出现乘积 $\gamma_j w_j$, 不可单独辨识二者; 且不会把“无关系数”积分掉。
- 负对数后验 (式 13.24)** = 拟合误差 + L2 权重罚 + 变量个数罚, 其中 $\lambda = 2\sigma^2 \log \frac{1-\pi_0}{\pi_0}$ 控制稀疏强度 (π_0 越小、 σ^2 越大 \rightarrow 越稀疏)。
- 当 $\sigma_w^2 \rightarrow \infty$:** 得到纯粹的**子集选择**目标
 $\|y - X_\gamma w_\gamma\|_2^2 + \lambda \|\gamma\|_0$,
进而写成对 w 的 ℓ_0 **正则化**: $\|y - Xw\|_2^2 + \lambda \|w\|_0$ 。
- 意义:** 把贝叶斯的掩码思路, 推导成了经典的 (非贝叶斯) ℓ_0 目标——“拟合好”与“变量越少越好”的直接权衡。
- 现实:** ℓ_0 非凸难优化, 后续通常用近似与算法技巧 (如 L1/Lasso、贪心 OMP、启发式搜索、贝叶斯近似等) 来求解。

这一小节主要在讲什么? 我不太懂

已思考 19s

这小节在讲一件核心事:

把一个“伯努利 + 高斯”的二值掩码模型, 推导成大家熟悉的 ℓ_0 正则化 (= 子集选择) 目标函数。

你可以这样理解——

用大白话串起来

1. 先搭个模型 (“二值面罩”)

- 每个特征有一个**开关** γ_j (伯努利, 开=1、关=0), 还有一个**权重** w_j (高斯, 均值0)。
- 预测时用的是 $\sum_j (\gamma_j w_j) x_{ij}$:
 - 开关关着 ($\gamma_j = 0$) \rightarrow 该特征被遮住, 贡献为 0;
 - 开关打开 ($\gamma_j = 1$) \rightarrow 用权重 w_j 正常发力。

- 所以它也叫**二值掩码模型**: γ 就像面罩, 决定某个权重是否起作用。

2. 和 spike&slab 不同点

- 这里**不把无关系数积分掉**, w_j 总是存在; 而 spike&slab 会把不相关的 w_j 钉死为 0 并“消掉”。
- 在掩码模型里, 似然里只看到乘积 $\gamma_j w_j$, 单靠数据分不清是“开关关了”还是“权重本身就是 0”(有点**不可辨识**的味道)。

3. 从这个贝叶斯模型推到一个常见的“打分函数”

- 写出**负对数后验**后, 得到一个很直观的目标:
拟合误差 + 权重的 L2 惩罚 + 被选特征个数的惩罚。
- 其中“被选特征个数”的惩罚系数 $\lambda = 2\sigma^2 \log \frac{1-\pi_0}{\pi_0}$:
 - π_0 越小 (先验更稀疏), λ 越大 \rightarrow 越不鼓励多选;
 - 噪声 σ^2 越大 (更怕过拟合), λ 也越大。

4. 把对非零权重的 L2 收缩拿掉, 就得到纯子集选择

- 让 $\sigma_w^2 \rightarrow \infty$ (不再对非零 w 做 L2 罚), 并把被遮住的位置的 w 设为 0, 目标就变成:

$$\underbrace{\|y - X_\gamma w_\gamma\|_2^2}_{\text{拟合误差}} + \underbrace{\lambda \|\gamma\|_0}_{\text{“选了几个特征”的罚}}.$$

- 再不显式写开关 γ , 而是直接用**“ w 的非零位置集合”代表“选了哪些特征”, 就得到 ℓ_0 正则**:

$$\|y - Xw\|_2^2 + \lambda \|w\|_0.$$

这就是**最经典的子集选择目标**: 拟合要好, 但每多选一个特征都要付费。

5. 意义与难点

- 意义: ℓ_0 正则化并非凭空来的, 它是**这个贝叶斯掩码模型的自然结果** (相当于做了一个 MAP 类的推导); 而且和前面 BIC 的“按变量个数罚”思路相呼应。
- 难点: $\|w\|_0$ 是**非凸、非光滑**的计数函数, 优化很难 (通常 NP 难), 所以后面章节才会用 L1 (Lasso)、贪心法等近似/算法来搞定。

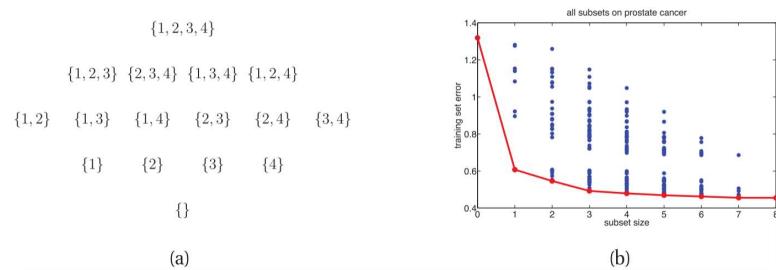
一个生活比喻

把建模当“采购零件”:

- 每个零件先有个**面罩** (γ): 戴上就不买 (不计入), 摘掉就可以买。
- 你要让**机器运转误差最小** (拟合误差小), 但**每多买一个零件都要花钱** ($\lambda \times$ 零件数)。
- 最终你要找“**误差 + 成本**”之和最小的采购单——这就是 ℓ_0 正则的精神。

要点小结

- 主题：**把“伯努利开关 + 高斯权重”的二值掩码模型，推导成 ℓ_0 正则化/子集选择目标。
- 过程：**负对数后验 = 拟合误差 + L2 权重罚 + (选中特征数) 罚；当不对非零权重做 L2 收缩时，化为 $\|y - Xw\|_2^2 + \lambda \|w\|_0$ 。
- λ 的直觉：**先验更稀疏 (π_0 小) 或噪声更大 (σ^2 大) $\rightarrow \lambda$ 大 \rightarrow 更少选特征。
- 意义：**把贝叶斯思路与经典子集选择连起来，解释了**为何 ℓ_0 正则合理**。
- 现实：** ℓ_0 难求解 \rightarrow 后续靠 L1 (Lasso)、贪心 OMP、启发式等方法近似解决。



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 13.2 (a) A lattice of subsets of {1, 2, 3, 4}. (b) Residual sum of squares versus subset size, on the prostate cancer data set. The lower envelope is the best RSS achievable for any set of a given size. Based on Figure 3.5 of (Hastie et al. 2001).

13.2.3 Algorithms

Since there are 2D models, we cannot explore the full posterior, or find the globally optimal model. Instead we will have to resort to heuristics of one form or another. All of the methods we will discuss involve searching through the space of models, and evaluating the cost $f(y)$ at each point. This requires fitting the model (i.e., computing $\text{argmax } p(D|w)$), or evaluating its marginal likelihood (i.e., computing $\int p(D|w)p(w)dw$) at each step. This is sometimes called the wrapper method, since we “wrap” our search for the best model (or set of good models) around a generic model-fitting procedure.

In order to make wrapper methods efficient, it is important that we can quickly evaluate the score function

for some new model, γ' , given the score of a previous model, γ . This can be done provided we can efficiently update the sufficient statistics needed to compute $f(\gamma)$. This is possible provided γ' only differs from γ in one bit (corresponding to adding or removing a single variable), and provided $f(\gamma)$ only depends on the data via $X\gamma$. In this case, we can use rank-one matrix updates/ downdates to efficiently compute $X^T \gamma' X\gamma$ from $X^T \gamma X\gamma$. These updates are usually applied to the QR decomposition of X . See e.g., (Miller 2002; Schniter et al. 2008) for details.

13.2.3.1 Greedy search

Suppose we want to find the MAP model. If we use the 0-regularized objective in Equation 13.27, we can exploit properties of least squares to derive various efficient greedy forwards search methods, some of which we summarize below. For further details, see (Miller 2002; Souassen et al. 2010).

- Single best replacement The simplest method is to use greedy hill climbing, where at each step, we define the neighborhood of the current model to be all models than can be reached by flipping a single bit of γ , i.e., for each variable, if it is currently out of the model, we consider adding it, and if it is currently in the model, we consider removing it. In (Souassen et al. 2010), they call this the single best replacement (SBR). Since we are expecting a sparse solution, we can start with the empty set, $\gamma = 0$. We are essentially moving through the lattice of subsets, shown in Figure 13.2(a). We continue adding or removing until no improvement is possible.
- Orthogonal least squares If we set $\lambda = 0$ in Equation 13.27, so there is no complexity penalty, there will be no reason to perform deletion steps. In this case, the SBR algorithm is equivalent to orthogonal least squares (Chen and Wigger 1995), which in turn is equivalent to greedy forwards selection. In this algorithm, we start with the empty set and add the best feature at each step. The error will go down monotonically with $\|\gamma\|_0$, as shown in Figure 13.2(b). We can pick the next best feature j^* to add to the current set γ_t by solving

$$j^* = \arg \min j \in \gamma_t \min w \|y - (X\gamma_t \cup j)w\|^2 \quad (13.28)$$

We then update the active set by setting $\gamma_{t+1} = \gamma_t \cup \{j^*\}$. To choose the next feature to add at step t , we need to solve $D-D_t$ least squares problems at step t , where $D_t =$

$|yt|$ is the cardinality of the current active set. Having chosen the best feature to add, we need to solve an additional least squares problem to compute $wt+1$.

- Orthogonal matching pursuits Orthogonal least squares is somewhat expensive. A simplification is to “freeze” the current weights at their current value, and then to pick the next feature to add by solving

$$j^* = \arg \min j \in yt \min \beta \|y - Xw - \beta x_{:,j} \|_2^2 \quad (13.29)$$

This inner optimization is easy to solve: we simply set $\beta = x_{T,:j} / \|x_{:,j}\|_2$, where $r_t = y - Xw$ is the current residual vector. If the columns are unit norm, we have

$$j^* = \operatorname{argmax}_{x_{T,:}} x_{T,:j} \quad (13.30)$$

so we are just looking for the column that is most correlated with the current residual. We then update the active set, and compute the new least squares estimate w_{t+1} using X_{yt+1} . This method is called orthogonal matching pursuits or OMP (Mallat et al. 1994). This only requires one least squares calculation per iteration and so is faster than orthogonal least squares, but is not quite as accurate (Blumensath and Davies 2007).

- Matching pursuits An even more aggressive approximation is to just greedily add the feature that is most correlated with the current residual. This is called matching pursuits (Mallat and Zhang 1993). This is also equivalent to a method known as least squares boosting (Section 16.4.6).
- Backwards selection Backwards selection starts with all variables in the model (the socalled saturated model), and then deletes the worst one at each step. This is equivalent to performing a greedy search from the top of the lattice downwards. This can give better results than a bottom-up search, since the decision about whether to keep a variable or not is made in the context of all the other variables that might depende on it. However, this method is typically infeasible for large problems, since the saturated model will be too expensive to fit.
- FoBa The forwards-backwards algorithm of (Zhang 2008) is similar to the single best replacement algorithm presented above, except it uses an OMP-like approximation when choosing the next move to make. A similar “dual-pass” algorithm was described in (Moghaddam et al. 2008).
- Bayesian Matching pursuit The algorithm of (Schniter et

al. 2008) is similar to OMP except it uses a Bayesian marginal likelihood scoring criterion (under a spike and slab model) instead of a least squares objective. In addition, it uses a form of beam search to explore multiple paths through the lattice at once.

13.2.3.2 Stochastic search

If we want to approximate the posterior, rather than just computing a mode (e.g. because we want to compute marginal inclusion probabilities), one option is to use MCMC. The standard approach is to use Metropolis Hastings, where the proposal distribution just flips single bits. This enables us to efficiently compute $p(\gamma | D)$ given $p(\gamma|D)$. The probability of a state (bit configuration) is estimated by counting how many times the random walk visits this state. See (O'Hara and Sillanpaa 2009) for a review of such methods, and (Bottolo and Richardson 2010) for a very recent method based on evolutionary MCMC.

However, in a discrete state space, MCMC is needlessly inefficient, since we can compute the (unnormalized) probability of a state directly using $p(\gamma, D) = \exp(-f(\gamma))$; thus there is no need to ever revisit a state. A much more efficient alternative is to use some kind of stochastic search algorithm, to generate a set S of high scoring models, and then to make the following approximation

$$p(\gamma|D) \approx e^{-f(\gamma)} \gamma \in S e^{-f(\gamma)} \quad (13.31)$$

See (Heaton and Scott 2009) for a review of recent methods of this kind.

13.2.3.3 EM and variational inference

It is tempting to apply EM to the spike and slab model, which has the form $\gamma_j \rightarrow w_j \rightarrow y$. We can compute $p(\gamma_j = 1 | w_j)$ in the E step, and optimize w in the M step.

However, this will not work, because when we compute $p(\gamma_j = 1 | w_j)$, we are comparing a delta-function, $\delta_0(w_j)$, with a Gaussian pdf, $N(w_j | 0, \sigma^2_w)$. We can replace the delta function with a narrow Gaussian, and then the E step amounts to classifying w_j under the two possible Gaussian models. However, this is likely to suffer from severe local minima.

An alternative is to apply EM to the Bernoulli-Gaussian model, which has the form $\gamma_j \rightarrow y \leftarrow w_j$. In this case, the posterior $p(\gamma|D, w)$ is intractable to compute because all the bits become correlated due to explaining away.

However, it is possible to derive a mean field approximation of the form ' $\int q(y_j)q(w_j)$ ' (Huang et al. 2007; Rattray et al. 2009).

已思考 23s

下面先给出逐字完整翻译（不漏译），再给通俗解释，最后给要点小结。

（说明：原文里个别公式/符号在排版上有些缺失，我在译文中写成标准形式并在解释里指出。）

译文（完整）

图 13.2 (a) $\{1, 2, 3, 4\}$ 的子集格 (lattice)。 (b) 前列腺癌数据集上，残差平方和 (RSS) 与子集大小的关系。红色下包络线表示对任何给定大小的变量子集所能达到的最佳 RSS。改绘自 (Hastie et al., 2001) 的图 3.5。

13.2.3 算法

由于模型共有 2^D 个，我们无法探索完整的后验，也无法找到全局最优模型。于是我们不得不借助各种启发式方法。下面要讨论的所有方法都涉及在模型空间中搜索，并在每个点上评估代价 $f(\gamma)$ 。这需要在每一步拟合模型（即计算 $\arg \max_w p(D | w)$ ），或评估其边际似然（即计算 $\int p(D | w)p(w) dw$ ）。这有时被称为包装法 (wrapper method)，因为我们把“寻找最佳模型（或一组好模型）”这件事包裹在一个通用的模型拟合程序外面。

为了让包装法高效，关键在于：在已知某个旧模型 γ 的得分的前提下，能快速评估某个新模型 γ' 的得分。只要能高效更新计算 $f(\gamma)$ 所需的充分统计量，就能做到这一点。特别地，只要 γ' 与 γ 只差 1 个比特（对应添加或删除一个变量），而且 $f(\gamma)$ 只通过 X_γ 与数据发生关系，那么就可以用秩一 (rank-one) 矩阵更新/降阶，把 $X_\gamma^\top X_\gamma$ 由 $X_{\gamma'}^\top X_{\gamma'}$ 快速更新得到。这些更新通常应用在 X 的 QR 分解上。详见 (Miller, 2002; Schniter et al., 2008)。

13.2.3.1 贪心搜索

假设我们想找到 MAP 模型。如果使用式 (13.27) 的 ℓ_0 正则化目标，就可以利用最小二乘的性质推导出若干高效的前向贪心方法，下面做一览；细节见 (Miller, 2002; Soussen et al., 2010)。

- **单次最佳替换 (Single Best Replacement, SBR)**

最简单的方法是贪心爬山：把当前模型的邻域定义为将 γ 的任意一个比特翻转后得到的所有模型——对每个变量，若当前不在模型里就考虑把它加进来；若在模型里就考虑删除。在 (Soussen et al., 2010) 中把它称为 SBR。因为我们期望解是稀疏的，可以

从空集 $\gamma = 0$ 开始。实质上，这就是在图 13.2(a) 的子集格上移动。不断加/删，直到无法改进为止。

- **正交最小二乘 (Orthogonal Least Squares, OLS)**

若在式 (13.27) 中令 $\lambda = 0$ (无复杂度惩罚)，就没有理由做删除步。在这种情形下，SBR 等价于 OLS (Chen & Wigger, 1995)，也等价于**前向逐步选择**：从空集开始，每步加入表现最佳的特征。误差会随 $\|\gamma\|_0$ 单调下降，如图 13.2(b)。第 t 步选下一个特征可解

$$j^{*} = \arg \min_{j \in \gamma_t} \min_w \|y - (X_{\gamma_t \cup j})w\|_2^2 \quad (13.28)$$

然后把活动集更新为 $\gamma_{t+1} = \gamma_t \cup \{j^*\}$ 。在第 t 步需要解 $D - D_t$ 个最小二乘问题来挑最优特征 (其中 $D_t = |\gamma_t|$)，挑好后再解一个最小二乘得到 $w^{(t+1)}$ 。

- **正交匹配追踪 (Orthogonal Matching Pursuit, OMP)**

OLS 代价偏高。一个简化是**冻结当前权重** w_t ，然后挑下一个特征解

$$j^{*} = \arg \min_{j \notin \gamma_t} \min_{\beta} \|y - Xw_t - \beta x_{:,j}\|_2^2 \quad (13.29)$$

内层最优是 $\beta = x_{:,j}^\top r_t / \|x_{:,j}\|_2^2$ ，其中 $r_t = y - Xw_t$ 是当前残差。若所有列已单位化，则

$$j^{*} = \arg \max_j x_{:,j}^\top r_t \quad (13.30)$$

即找与残差**最相关**的那一列。随后把该特征加入并在 $X_{\gamma_{t+1}}$ 上重算一次最小二乘得到 $w^{(t+1)}$ 。OMP 每轮只需一次最小二乘，因而比 OLS 快，但精度略逊 (Blumensath & Davies, 2007)。

- **匹配追踪 (Matching Pursuit, MP)**

更激进的近似：**只贪心加入与当前残差最相关的特征**（不在每步重算最小二乘）。称为 MP (Mallat & Zhang, 1993)，也等价于**最小二乘提升**（参见 16.4.6 节）。

- **后向选择 (Backwards Selection)**

从**饱和模型**（包含所有变量）出发，每步删除“最差”的一个。这相当于从子集格的顶部向下做贪心。因为是否保留某特征是在“其它特征都在场”的背景下决定的，有时会优于自底向上的搜索。但对大问题通常**不可行**，因饱和模型拟合过于昂贵。

- **FoBa (Forwards–Backwards)**

Zhang (2008) 的 FoBa 与上面的 SBR 类似，但在选择下一步动作时用 OMP 式的近似。类似的“双通道”算法见 (Moghaddam et al., 2008)。

- **贝叶斯匹配追踪 (Bayesian Matching Pursuit)**

(Schniter et al., 2008) 的算法与 OMP 相似，但用**贝叶斯边际似然**（在 spike-and-slab

先验下) 而非最小二乘作评分, 并采用**束搜索 (beam search) **同时探索多条路径。

13.2.3.2 随机化搜索

若我们想近似后验分布而非只找一个众数 (例如为了计算边际纳入概率), 可以用 MCMC。标准做法是 Metropolis-Hastings: 提议分布每次只翻转一个比特。这样可以高效地由 $p(\gamma \mid D)$ 计算 $p(\gamma' \mid D)$ 。某个状态 (比特配置) 的概率通过随机游走访问次数来估计。综述见 (O'Hara & Sillanpää, 2009); 基于进化 MCMC 的最新方法见 (Bottolo & Richardson, 2010)。

然而, 在离散状态空间里, MCMC 往往低效, 因为我们可以直接计算某状态的 (未归一化) 概率 $p(\gamma, D) = \exp(-f(\gamma))$, 因此没必要反复回访同一状态。更高效的替代是某种随机化搜索: 生成一组高分模型 S , 然后用下面的近似

$$p(\gamma \mid D) \approx \frac{e^{-f(\gamma)}}{\sum_{\gamma' \in S} e^{-f(\gamma')}} , \quad \gamma \in S \quad (13.31)$$

关于此类方法的综述见 (Heaton & Scott, 2009)。

13.2.3.3 EM 与变分推断

把 EM 用到 spike-and-slab (结构 $\gamma_j \rightarrow w_j \rightarrow y$) 上很诱人: E 步算 $p(\gamma_j = 1 \mid w_j)$, M 步优化 w 。但这行不通: E 步要比较的是狄拉克 $\delta_0(w_j)$ 与高斯密度 $N(w_j \mid 0, \sigma_w^2)$ 。可把 δ 换成一个很窄的高斯, 但 E 步就变成在两个高斯间硬分类, 极易陷入严重局部最优。

另一条路是把 EM 用到 伯努利-高斯 (掩码) 模型 (结构 $\gamma_j \rightarrow y \leftarrow w_j$)。此时后验 $p(\gamma \mid D, w)$ 因为互相解释 (explaining-away) 而使各比特相关, 变得不可 tractable。不过可以推导出一种均值场近似: $\prod_j q(\gamma_j) q(w_j)$ (Huang et al., 2007; Rattray et al., 2009)。

通俗解释 (逐段讲清)

图 13.2 想说明什么?

- 左图(a) 是“所有子集”的结构图: 最底下是空集, 上面一层是一元子集, 再往上是二元子集……顶层是全体 $\{1, 2, 3, 4\}$ 。很多算法就是在这张“子集格”里上上下下移动。
- 右图(b) 画的是“训练误差 (RSS) 随所选变量个数的变化”: 蓝点是每个具体子集的 RSS; 红线是“在该大小下最好的”那一个 (下包络)。通常变量越多, 训练误差越小, 但并不代表泛化越好——后面要靠正则/验证集来决定停在哪一层。

包装法 (wrapper) 是啥?

- 每试一个子集，都要“把这个子集交给一个通用的拟合器”算分（拟合/边际似然），因此叫“外面包着一个搜索器”。
- 为了快：每次只加/删一个变量，复用上一步的分解（如 QR），做~~秩一更新~~，就不用从头算。

贪心家族怎么区别？

- SBR：在“加一个/删一个”的邻域里挑最好的那一步，来回加减直到不能改进。
- OLS ($\lambda = 0$)：只加不删，每步把当前能把误差降得最多的特征加进来；每步要解很多小的最小二乘来比较候选。
- OMP：先用当前权重算残差，挑与残差最相关的特征加进来；每步只解一次最小二乘，快一些。
- MP：更简：只看相关性，连重算最小二乘都不做，最快但最粗。
- 后向选择：从全体开始每次删一个；决策背景最“充分”，但一开始就要拟合超大的模型，常用不起。
- FoBa：前向为主，必要时后向纠正一步，介于二者之间。
- Bayesian MP：像 OMP，但评分用贝叶斯边际似然，并用束搜索一次走多条路，兼顾速度与全局性。

想要的不只是一个解，而是后验分布？

- MCMC 可以近似整个位分布，但在离散空间里会老“踩回来”。
- 更实际：用随机化搜索先收集一批高分模型 S ，再在这批里按 $e^{-f(\gamma)}$ 归一化当作近似后验（式 13.31）。

EM/变分能不能用？

- 直接对 spike-and-slab 用 EM：E 步要把“一个无限尖的 δ ”和“一个高斯”比较，容易崩成坏的局部解。
 - 换掩码模型做 EM：位之间因为“互相解释”强相关，精确 E 步不可行；但可以做均值场（把联合后验拆成每个变量的独立近似）。
-

要点小结

- 搜索空间巨大： 2^D 个子集，无法全扫 \Rightarrow 采用包装式搜索 + 快速秩一更新。
- 贪心路线图：
 - SBR (加/删一步) \rightarrow OLS (只加、每步多次最小二乘) \rightarrow OMP (看残差相关性、每步一次最小二乘) \rightarrow MP (只看相关性)。
 - 反向：后向选择；折中：FoBa；贝叶斯打分：Bayesian MP。

- **训练误差 vs 子集大小**: 变量越多训练误差越小, 但泛化不一定更好; 看图 13.2(b) 的红线就是“在该大小下最优的训练误差”。
- **近似后验**: MCMC 可行但在离散空间低效; 用**随机化搜索**收集高分集合 S , 再在其中归一化近似 $p(\gamma \mid D)$ (式 13.31)。
- **EM/变分**:
 - spike-and-slab 上 EM 易陷局部最优;
 - 掩码模型精确 E 步不可行, 但可用**均值场** $\prod_j q(\gamma_j)q(w_j)$ 近似。

实战小贴士: 中等规模先用 OMP/FoBa 这类快贪心, 配合交叉验证选停点或选 λ ; 若想要不确定性评估, 再用**随机化搜索 + 归一化**或轻量 MCMC 做后验近似。

这一小节主要在讲什么? 我不太懂

已思考 23s

这小节其实在讲两件事: **怎么在巨大的“子集空间”里找变量子集**, 以及**这些搜索要怎么又快又靠谱**。配套的图 13.2 只是帮你把“子集空间长什么样、误差随子集大小怎么变”直观化。

先把问题说清

- 一共有 2^D 种“选哪些变量”的组合, **不可能全都试**。
- 我们用一个**打分函数** $f(\gamma)$ (越小越好) 评估每个组合 γ 的好坏。
每试一个组合, 都要把模型“跑一遍”求分——这种**把搜索包在拟合器外**的方法叫 **wrapper**。

怎么让 wrapper 跑得快

- **每次只改一个变量** (加一个或删一个), 这样两次模型非常接近。
- **用秩一更新** (在 QR 分解上做小修小补) 把上一步的计算复用掉, 就不用每次从头算。

图 13.2 在说啥

- (a) 是**子集格**: 最底层是空集, 往上依次是包含 1 个、2 个、... 的子集, 最上是全体。搜索算法就是在这张“格子”上爬来爬去。

- (b) 蓝点：给定“选 k 个变量”的所有组合的训练误差 (RSS)；红线：每个 k 能达到的最好训练误差。
规律：变量越多，训练误差越小（红线单调下降），但这不代表泛化更好，所以需要正则/停下来的规则。
-

具体有哪些搜索法（从贪心到随机）

1) 贪心家族（一次走一步，简单高效）

- SBR (单次最佳替换)：从空集开始，每次加或删一个变量，看哪一步能把分数降得最多，就走哪一步，直到不能再改进。
- OLS 前向选择 ($\lambda=0$ 时)：只加不删；每步把能最大幅度降低残差的变量加进来。误差会随“已选个数”单调下降。
- OMP (正交匹配追踪)：先算当前残差，挑与残差最相关的变量加进来；每轮只解一次最小二乘，比 OLS 快些，精度略差。
- MP (匹配追踪)：更粗糙；只按“相关性最大”加，不重算权重，最快但最不精。
- 后向选择：从全选开始每次删掉最差的一个；决策更“全面”，但一上来就要拟合超大模型，大规模用不起。
- FoBa (前后向结合)：以“前向加”为主，必要时来一步“后向删”纠错，是 SBR 的加速版。
- Bayesian MP：像 OMP，但评分用贝叶斯边际似然 (spike-and-slab 先验)，并用束搜索一次并行探索多条路径。

2) 随机化/后验近似（要的不止一个解）

- MCMC (MH 翻转一位)：通过随机游走近似整个后验（想算“边际纳入概率”时很有用）。但在离散空间里会老回头，效率不高。
- 随机化搜索集合 S ：更实用的替代是先用随机/启发式搜索收集一批高分模型 S ，再按 $e^{-f(\gamma)}$ 在集合里归一化，近似后验；不必反复访问同一状态。

3) EM / 变分（为什么不太好用，或怎么近似）

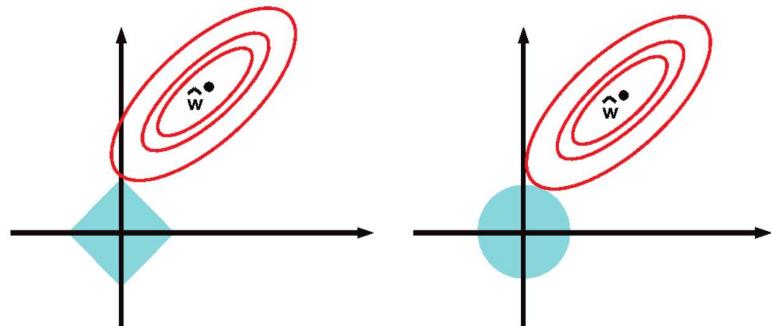
- 在 spike-and-slab 上用 EM：E 步要拿“ δ 脉冲”跟“高斯”比，容易陷入很坏的局部最优 → 不推荐。
 - 在伯努利-高斯掩码上用 EM：位之间因“互相解释”变得强相关，精确 E 步不可解；可用均值场变分把后验近似为 $\prod_j q(\gamma_j)q(w_j)$ 。
-

该如何选用

- 想快速给出一套可用子集：先上 OMP / FoBa / SBR；用验证集或信息准则决定在哪一层停。
- 想评估不确定性（谁更可能被选）：用随机化搜索 + 集合归一化，或轻量的 MCMC。
- 规模很大：务必用“每次只改一位 + 程序更新”，否则算不动。

一句话总结

这小节是在教你：**在 2^D 的巨大子集空间里，如何用一系列贪心、随机与近似推断的方法高效搜索好模型**；同时用图 13.2 说明“子集格”的结构和“变量越多训练误差越低，但不代表更好”的事实。



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 13.3 Illustration of 1 (left) vs 2 (right) regularization of a least squares problem. Based on Figure 3.12 of (Hastie et al. 2001).

13.3 1 regularization: basics

When we have many variables, it is computationally difficult to find the posterior mode of $p(\gamma|D)$. And although greedy algorithms often work well (see e.g., (Zhang 2008) for a theoretical analysis), they can of course get stuck in local optima.

Part of the problem is due to the fact that the γ_j variables are discrete, $\gamma_j \in \{0, 1\}$. In the optimization community, it is common to relax hard constraints of this form by replacing discrete variables with continuous variables. We can do this by replacing the spike-and-slab style prior, that assigns finite probability mass to the event that w_j

=0, to continuous priors that "encourage" $w_j = 0$ by putting a lot of probability density near the origin, such as a zero-mean Laplace distribution. This was first introduced in Section 7.4 in the context of robust linear regression. There we exploited the fact that the Laplace has heavy tails. Here we exploit the fact that it has a spike near $\mu = 0$. More precisely, consider a prior of the form

$$p(w|\lambda) = \prod_{j=1}^D \text{Lap}(w_j|0, 1/\lambda) \propto \prod_{j=1}^D e^{-\lambda|w_j|} \quad (13.32)$$

We will use a uniform prior on the offset term, $p(w_0) \propto 1$. Let us perform MAP estimation with this prior. The penalized negative log likelihood has the form

$$f(w) = -\log p(D|w) - \log p(w|\lambda) = \text{NLL}(w) + \lambda\|w\|_1 \quad (13.33)$$

where $\|w\|_1 = \sum_{j=1}^D |w_j|$ is the 1 norm of w . For suitably large λ , the estimate \hat{w} will be sparse, for reasons we explain below. Indeed, this can be thought of as a convex approximation to the non-convex ℓ_0 objective

$$\operatorname{argmin}_w \text{NLL}(w) + \lambda\|w\|_0 \quad (13.34)$$

In the case of linear regression, the 1 objective becomes

$$f(w) = N \sum_{i=1}^N (y_i - (w_0 + w^T x_i))^2 + \lambda\|w\|_1 \quad (13.35)$$

$$= \text{RSS}(w) + \lambda\|w\|_1 \quad (13.36)$$

where $\lambda = 2\lambda\sigma^2$. This method is known as basis pursuit denoising or BPDN (Chen et al. 1998). The reason for this term will become clear later. In general, the technique of putting a zero-mean Laplace prior on the parameters and performing MAP estimation is called 1 regularization. It can be combined with any convex or non-convex NLL term. Many different algorithms have been devised for solving such problems, some of which we review in Section 13.4.

13.3.1 Why does L1 regularization yield sparse solutions?

We now explain why 1 regularization results in sparse solutions, whereas 2 regularization does not. We focus on the case of linear regression, although similar arguments hold for logistic regression and other GLMs.

The objective is the following non-smooth objective function:

$$\operatorname{min}_w \text{RSS}(w) + \lambda\|w\|_1 \quad (13.37)$$

We can rewrite this as a constrained but smooth objective (a quadratic function with linear constraints):

$$\operatorname{min}_w \text{RSS}(w) \text{ s.t. } \|w\|_1 \leq B \quad (13.38)$$

where B is an upper bound on the 1-norm of the weights: a small (tight) bound B corresponds to a large penalty λ , and vice versa. Equation 13.38 is known as lasso, which

stands for "least absolute shrinkage and selection operator" (Tibshirani 1996). We will see why it has this name later. Similarly, we can write ridge regression

$$\min w \text{RSS}(w) + \lambda \|w\|_2^2 \quad (13.39)$$

or as a bound constrained form:

$$\min w \text{RSS}(w) \text{s.t. } \|w\|_2 \leq B \quad (13.40)$$

In Figure 13.3, we plot the contours of the RSS objective function, as well as the contours of the ℓ_2 and ℓ_1 constraint surfaces. From the theory of constrained optimization, we know that the optimal solution occurs at the point where the lowest level set of the objective function intersects the constraint surface (assuming the constraint is active). It should be geometrically clear that as we relax the constraint B , we "grow" the ℓ_1 "ball" until it meets the objective; the corners of the ball are more likely to intersect the ellipse than one of the sides, especially in high dimensions, because the corners "stick out" more. The corners correspond to sparse solutions, which lie on the coordinate axes. By contrast, when we grow the ℓ_2 ball, it can intersect the objective at any point; there are no "corners", so there is no preference for sparsity.

To see this another way, notice that, with ridge regression, the prior cost of a sparse solution, such as $w = (1, 0)$, is the same as the cost of a dense solution, such as $w = (1/\sqrt{2}, 1/\sqrt{2})$, as long as they have the same ℓ_2 norm:

$$\|(1, 0)\|_2 = \|(1/\sqrt{2}, 1/\sqrt{2})\|_2 = 1 \quad (13.41)$$

However, for lasso, setting $w = (1, 0)$ is cheaper than setting $w = (1/\sqrt{2}, 1/\sqrt{2})$, since

$$\|(1, 0)\|_1 = 1 < \|(1/\sqrt{2}, 1/\sqrt{2})\|_1 = \sqrt{2} \quad (13.42)$$

The most rigorous way to see that ℓ_1 regularization results in sparse solutions is to examine conditions that hold at the optimum. We do this in Section 13.3.2.

已思考 19s

下面先给出**完整直译**（不漏译；文中“1”“2”均指 ℓ_1 、 ℓ_2 ），然后是**通俗解释**，最后**要点小结**。

译文（完整、不遗漏）

图 13.3 最小二乘问题下 ℓ_1 （左）与 ℓ_2 （右）正则化的示意图。改绘自 (Hastie et al., 2001) 的图 3.12。

13.3 ℓ_1 正则化：基础

当我们有很多变量时，计算 $p(\gamma | D)$ 的后验众数会在计算上变得困难。并且尽管贪心算法通常表现不错（例如，参见 (Zhang 2008) 的理论分析），但它们当然可能陷入局部最优。

问题的一部分来自于变量 γ_j 是离散的， $\gamma_j \in \{0, 1\}$ 。在优化领域，通常会通过把离散变量替换为连续变量来放松这类硬约束。我们可以通过把“把 $w_j = 0$ 的事件赋予有限概率质量”的“脉冲-板块”类型的先验，替换为**连续先验**来做到这一点：这种先验通过在原点附近放置大量概率密度来“鼓励” $w_j = 0$ ，例如零均值的拉普拉斯分布。该分布在第 7.4 节的**稳健线性回归**背景中首次出现。那里我们利用的是拉普拉斯分布的**厚尾**；而这里我们利用的是它在 $\mu = 0$ 附近的**尖峰**。更准确地说，考虑如下形式的先验

$$p(w | \lambda) = \prod_{j=1}^D \text{Lap}(w_j | 0, 1/\lambda) \propto \prod_{j=1}^D e^{-\lambda|w_j|}. \quad (13.32)$$

我们对偏置项采用均匀先验 $p(w_0) \propto 1$ 。在该先验下做 MAP 估计。带惩罚的负对数似然为

$$f(w) = -\log p(D | w) - \log p(w | \lambda) = \text{NLL}(w) + \lambda \|w\|_1, \quad (13.33)$$

其中 $\|w\|_1 = \sum_{j=1}^D |w_j|$ 是 w 的 ℓ_1 范数。对足够大的 λ ，估计值 \hat{w} 将是稀疏的，原因我们将在下面解释。事实上，这可以被看作是对非凸 ℓ_0 目标的一个**凸近似**：

$$\arg \min_w \text{NLL}(w) + \lambda \|w\|_0. \quad (13.34)$$

在线性回归情形， ℓ_1 目标变为

$$f(w) = \sum_{i=1}^N -\frac{1}{2\sigma^2} (y_i - (w_0 + w^\top x_i))^2 + \lambda \|w\|_1 \quad (13.35)$$

$$= \text{RSS}(w) + \tilde{\lambda} \|w\|_1, \quad \text{其中 } \tilde{\lambda} = 2\lambda\sigma^2. \quad (13.36)$$

这种方法称为**基追踪去噪** (basis pursuit denoising, BPDN) (Chen et al., 1998)。该术语的原因稍后会变得清晰。一般而言，在参数上放置零均值拉普拉斯先验并进行 MAP 估计的技术称为“ ℓ_1 正则化”。它可以与任何凸或非凸的 NLL 项结合。为求解这类问题，人们设计了许多算法，部分将在 13.4 节回顾。

13.3.1 为什么 ℓ_1 正则化会产生稀疏解？

现在我们解释为什么 ℓ_1 正则化得到的是稀疏解，而 ℓ_2 正则化不会。我们聚焦于线性回归的情形，尽管类似的论证对逻辑回归和其他 GLM 也成立。

目标函数是如下**非光滑**目标：

$$\min_w \text{RSS}(w) + \lambda \|w\|_1. \quad (13.37)$$

我们可以把它改写为一个**有约束但光滑**的目标（一个带线性约束的二次函数）：

$$\min_w \text{RSS}(w) \quad \text{s.t.} \quad \|w\|_1 \leq B, \quad (13.38)$$

其中 B 是权重 ℓ_1 范数的上界：较小（更紧）的 B 对应较大的惩罚 λ ，反之亦然。² 式 (13.38) 被称为 **lasso**，意为“最小绝对收缩与选择算子”(Tibshirani, 1996)。我们稍后会看到它为何得此名。类似地，岭回归可以写成

$$\min_w \text{RSS}(w) + \lambda \|w\|_2^2 \quad (13.39)$$

或其**界约束形式**：

$$\min_w \text{RSS}(w) \quad \text{s.t.} \quad \|w\|_2^2 \leq B. \quad (13.40)$$

在图 13.3 中，我们画出了 RSS 目标函数的等高线，以及 ℓ_2 与 ℓ_1 约束面的等高线。由约束优化理论我们知道，最优解出现在目标函数的最低等高线与约束面相交之处（假设约束是活跃的）。从几何上看，随着我们放宽约束 B ，我们“扩张” ℓ_1 “球”直到它与目标相遇；在高维里，角比边更容易与椭圆等高线相交，因为角“更突出”。这些角对应**稀疏解**，它们落在坐标轴上。相反，当我们扩张 ℓ_2 球时，它可以在任何点与目标相交； ℓ_2 球**没有角**，因此**不偏好稀疏**。

换一种方式来看：在岭回归中，一个稀疏解（如 $w = (1, 0)$ ）的先验代价与一个稠密解（如 $w = (1/\sqrt{2}, 1/\sqrt{2})$ ）相同，只要它们的 ℓ_2 范数相同：

$$\|(1, 0)\|_2 = \|(1/\sqrt{2}, 1/\sqrt{2})\|_2 = 1. \quad (13.41)$$

然而，对 lasso 来说，令 $w = (1, 0)$ 比令 $w = (1/\sqrt{2}, 1/\sqrt{2})$ 便宜，因为

$$\|(1, 0)\|_1 = 1 < \|(1/\sqrt{2}, 1/\sqrt{2})\|_1 = \sqrt{2}. \quad (13.42)$$

最严谨地说明 ℓ_1 正则产生稀疏解的办法，是检查在最优点处成立的条件。我们将在 13.3.2 节这样做。

通俗解释（一步步讲清）

图 13.3 的直观

- 红色椭圆：最小二乘 (RSS) 的等高线，朝中心 (\hat{w}) 越小越好。
- 左图蓝色“菱形”是 ℓ_1 球 ($|w_1| + |w_2| \leq B$)；右图蓝色“圆”是 ℓ_2 球 ($w_1^2 + w_2^2 \leq B$)。

- 在有约束的优化里，最优点就是“**最小的红椭圆恰好碰到蓝边界**”的那个点。
- ℓ_1 的边界有“尖角”，最容易在**坐标轴上相交** → **某些坐标被压成 0** (稀疏)。
- ℓ_2 边界是圆滑的，没有角 → 很少把坐标压成严格 0，只会整体变小 (收缩但不选零)。

为什么要用 ℓ_1

- 直接做“选或不选”的 ℓ_0 目标是离散的、非凸的，很难求解。
- 用“零均值拉普拉斯先验 + MAP”得到的 ℓ_1 目标，是 ℓ_0 的**凸近似**：易算、效果好，还能产生稀疏。
- 在线性回归里， ℓ_1 正则也叫 **lasso** 或 **BPDN** (基追踪去噪)。

λ 与 B 的关系

- 惩罚大 (λ 大) \Leftrightarrow 允许的 ℓ_1 半径 B 小：模型被“勒得更紧”，更稀疏；
- 惩罚小 (λ 小) $\Leftrightarrow B$ 大：更接近无约束最小二乘，变量更多、系数更大。

ℓ_1 为啥更偏爱 0？再用一个数值对比

- 同样的 ℓ_2 长度， $(1,0)$ 与 $(1/\sqrt{2}, 1/\sqrt{2})$ 代价相等 (岭回归**不区分稀疏/稠密**)。
- 但 ℓ_1 下， $(1,0)$ 的“路程”更短 ($1 \text{ vs } \sqrt{2}$)，所以**更便宜**，自然更爱稀疏。

要点小结

- **动机：**直接在 $\gamma \in \{0, 1\}^D$ 上搜索太难；用**连续的拉普拉斯先验**把问题放松成 ℓ_1 正则。
- **ℓ_1 目标：** $\text{NLL}(w) + \lambda \|w\|_1$ ；在线性回归里是 $\text{RSS}(w) + \tilde{\lambda} \|w\|_1$ (BPDN/Lasso)。
- **几何直觉：** ℓ_1 球有“角”，容易与 RSS 等高线在坐标轴处相交 → 产生**稀疏 (很多系数 =0)**； ℓ_2 球圆滑 → 只收缩不致零。
- **与 ℓ_0 的关系：** ℓ_1 是 ℓ_0 的**凸近似**，计算上易解，又能逼近“选子集”的效果。
- **参数作用：** λ 越大 (或等价地 B 越小) → 越稀疏； λ 越小 → 更像普通最小二乘。
- **更严谨的解释：**最优化条件 (下一节) 可严格证明 ℓ_1 会给出稀疏解。

这一小节主要在讲什么？我不太懂

已思考 11s

这小节的主线就三件事：**为什么不用离散“选不选”(ℓ_0)，而用连续的 ℓ_1 正则； ℓ_1 的目标长什么样；它为何能让系数自动变成很多 0 (稀疏) 而 ℓ_2 做不到。**

用大白话串起来

- 原始难题：**直接在“变量选/不选”的比特向量 $\gamma \in \{0, 1\}^D$ 上找最优 (ℓ_0 子集选择) 太难，容易卡在局部解。
- 换个表述（连续放松）：**给权重 w 用**拉普拉斯先验**（在 0 附近很尖），做 MAP \rightarrow 得到目标

$$\text{损失} + \lambda \|w\|_1$$

这就是** ℓ_1 正则**（线性回归场景也叫 Lasso/BPDN）。它是 ℓ_0 的**凸近似**，更好求。

3. 图 13.3 怎么看：

- 红椭圆=最小二乘 (RSS) 的等高线；
- 左图蓝“菱形”= ℓ_1 约束 ($|w_1| + |w_2| \leq B$)，有**尖角**；
- 右图蓝“圆”= ℓ_2 约束 ($w_1^2 + w_2^2 \leq B$)，**圆滑**。

在有约束最小化里，“最优点”是红椭圆刚好**碰到**蓝边界的地方。 ℓ_1 的尖角更容易落在**坐标轴上** \rightarrow 某些坐标被压成 0（稀疏）； ℓ_2 没角，只会把所有系数一起缩小，很少变成严格 0。

4. 再用一个数字直觉：

- 岭回归(ℓ_2): $(1, 0)$ 和 $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ 的代价一样（同一 ℓ_2 长度） \rightarrow 不偏好稀疏。
- Lasso(ℓ_1): $|1| + |0| = 1 < \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}} = \sqrt{2}$ \rightarrow **更偏好** $(1, 0)$ ，因此更易把一维压成 0。

5. 参数怎么影响稀疏： λ 大 \Rightarrow 惩罚重 \Rightarrow 允许的 $\|w\|_1$ 更小（等价的界 B 更紧） \Rightarrow **更稀疏**； λ 小 \Rightarrow 更接近普通最小二乘。

一句话结论

- **目的：**把难求的 ℓ_0 子集选择，换成好求的** ℓ_1 正则**。
- **形式：** $\text{NLL}(w) + \lambda \|w\|_1$ （线性回归即 RSS + $\lambda \|w\|_1$ ）。
- **原因：** ℓ_1 约束有“角”，几何上天然偏好把部分权重压到 0 \rightarrow **自动选特征**； ℓ_2 没有这个性质。
- **别名：**Lasso / BPDN；可用于线性回归、逻辑回归等 GLM。

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。

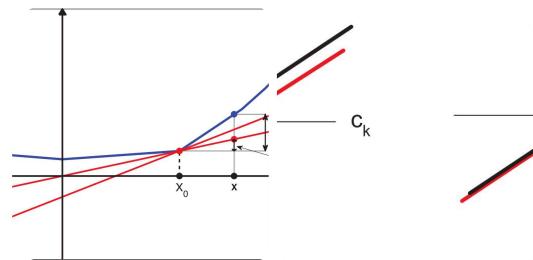
13.3.2 Optimality conditions for lasso

The lasso objective has the form

$$f(\theta) = \text{RSS}(\theta) + \lambda \|w\|_1 \quad (13.43)$$

Unfortunately, the $\|w\|_1$ term is not differentiable whenever $w_j = 0$. This is an example of a non-smooth optimization problem.

To handle non-smooth functions, we need to extend the notion of a derivative. We define a subderivative or subgradient of a (convex) function $f : I \rightarrow \mathbb{R}$ at a point θ_0 to be a scalar g such that $f(\theta) - f(\theta_0) \geq g(\theta - \theta_0) \quad \forall \theta \in I$ (13.44) where I is some interval containing θ_0 . See Figure 13.4 for an illustration. We define the set of subderivatives as the interval $[a, b]$ where a and b are the one-sided limits $a = \lim_{\theta \rightarrow \theta_0^-} f(\theta) - f(\theta_0) / (\theta - \theta_0)$, $b = \lim_{\theta \rightarrow \theta_0^+} f(\theta) - f(\theta_0) / (\theta - \theta_0)$ (13.46). The set $[a, b]$ of all subderivatives is called the subdifferential of the function f at θ_0 and is denoted $\partial f(\theta_0)$. For example, in the case of the absolute value function $f(\theta) = |\theta|$, the subderivative is given by $\partial f(\theta) = \begin{cases} -1 & \text{if } \theta < 0 \\ [-1, 1] & \text{if } \theta = 0 \\ 1 & \text{if } \theta > 0 \end{cases}$ (13.47). If the function is everywhere differentiable, then $\partial f(\theta) = \{df(\theta) / d\theta\}$. By analogy to the standard calculus result, one can show that the point $\hat{\theta}$ is a local minimum of f iff $0 \in \partial f(\hat{\theta})$.



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 13.4 Illustration of some sub-derivatives of a

function at point x_0 . Based on a figure at <http://en.wikipedia.org/wiki/Subderivative>.

Figure 13.5 Left: soft thresholding. The flat region is the interval $[-\lambda, +\lambda]$. Right: hard thresholding.

13.3.2 Optimality conditions for lasso

The lasso objective has the form

$$f(\theta) = \text{RSS}(\theta) + \lambda \|w\|_1 \quad (13.43)$$

Unfortunately, the $\|w\|_1$ term is not differentiable whenever $w_j = 0$. This is an example of a non-smooth optimization problem.

To handle non-smooth functions, we need to extend the notion of a derivative. We define a subderivative or subgradient of a (convex) function $f : I \rightarrow \mathbb{R}$ at a point θ_0 to be a scalar g such that

$$f(\theta) - f(\theta_0) \geq g(\theta - \theta_0) \quad \forall \theta \in I \quad (13.44)$$

where I is some interval containing θ_0 . See Figure 13.4 for an illustration. We define the set of subderivatives as the interval $[a, b]$ where a and b are the one-sided limits

$$a = \lim_{\theta \rightarrow \theta^-} f(\theta) - f(\theta_0) / (\theta - \theta_0), \quad b = \lim_{\theta \rightarrow \theta^+} f(\theta) - f(\theta_0) / (\theta - \theta_0) \quad (13.46)$$

The set $[a, b]$ of all subderivatives is called the subdifferential of the function f at θ_0 and is denoted $\partial f(\theta)|_{\theta_0}$. For example, in the case of the absolute value function $f(\theta) = |\theta|$, the subderivative is given by

$$\partial f(\theta) = \begin{cases} \{-1\} & \text{if } \theta < 0 \\ [-1, 1] & \text{if } \theta = 0 \\ \{+1\} & \text{if } \theta > 0 \end{cases} \quad (13.47)$$

If the function is everywhere differentiable, then $\partial f(\theta) = \{df(\theta) d\theta\}$. By analogy to the standard calculus result, one can show that the point $\hat{\theta}$ is a local minimum of f iff $0 \in \partial f(\theta)|_{\hat{\theta}}$.

Let us apply these concepts to the lasso problem. Let us initially ignore the non-smooth penalty term. One can show (Exercise 13.1) that

$$\partial_w \text{RSS}(w) = a_j w_j - c_j \quad (13.48)$$

$$a_j = 2 \sum_{i=1}^n x_i^2 j_i \quad (13.49)$$

$$c_j = 2 \sum_{i=1}^n x_i (y_i - w^T - j_i) \quad (13.50)$$

where $w-j$ is w without component j , and similarly for x_i , $-j$. We see that c_j is (proportional to) the correlation between the j 'th feature $x_{:,j}$ and the residual due to the other features, $r-j = y - X_{:,j} - w-j$. Hence the magnitude of c_j is an indication of how relevant feature j is for predicting y (relative to the other features and the current parameters).

Adding in the penalty term, we find that the subderivative is given by

$$\partial w_j f(w) = (ajw_j - c_j) + \lambda \partial w_j \|w\|_1 \quad (13.51)$$

$$= \begin{cases} ajw_j - c_j - \lambda & \text{if } w_j < 0 \\ [-c_j - \lambda, -c_j + \lambda] & \text{if } w_j = 0 \\ ajw_j - c_j + \lambda & \text{if } w_j > 0 \end{cases} \quad (13.52)$$

We can write this in a more compact fashion as follows:

$$XT(Xw - y)_j \in \begin{cases} \{-\lambda\} & \text{if } w_j < 0 \\ [-\lambda, \lambda] & \text{if } w_j = 0 \\ \{\lambda\} & \text{if } w_j > 0 \end{cases} \quad (13.53)$$

Depending on the value of c_j , the solution to $\partial w_j f(w) = 0$ can occur at 3 different values of w_j , as follows:

1. If $c_j < -\lambda$, so the feature is strongly negatively correlated with the residual, then the subgradient is zero at $\hat{w}_j = c_j + \lambda$ $aj < 0$.
2. If $c_j \in [-\lambda, \lambda]$, so the feature is only weakly correlated with the residual, then the subgradient is zero at $\hat{w}_j = 0$.
3. If $c_j > \lambda$, so the feature is strongly positively correlated with the residual, then the subgradient is zero at $\hat{w}_j = c_j - \lambda$ $aj > 0$.

In summary, we have

$$\hat{w}_j(c_j) = \begin{cases} (c_j + \lambda)/aj & \text{if } c_j < -\lambda \\ 0 & \text{if } c_j \in [-\lambda, \lambda] \\ (c_j - \lambda)/aj & \text{if } c_j > \lambda \end{cases} \quad (13.54)$$

We can write this as follows:

$$\hat{w}_j = \text{soft}(c_j aj; \lambda aj) \quad (13.55)$$

where

$$\text{soft}(a; \delta) = \text{sign}(a)(|a| - \delta)_+ \quad (13.56)$$

and $x_+ = \max(x, 0)$ is the positive part of x . This is called soft thresholding. This is illustrated in Figure 13.5(a), where we plot \hat{w}_j vs c_j . The dotted line is the line $w_j = c_j/aj$ corresponding to the least squares fit. The solid line, which represents the regularized estimate $\hat{w}_j(c_j)$, shifts the dotted line down (or up) by λ , except when $-\lambda \leq c_j \leq \lambda$, in which case it sets $w_j = 0$.

By contrast, in Figure 13.5(b), we illustrate hard thresholding. This sets values of w_j to 0 if $-\lambda \leq c_j \leq \lambda$, but it does not shrink the values of w_j outside of this interval. The slope of the soft thresholding line does not coincide with the diagonal, which means that even large coefficients are shrunk towards zero; consequently lasso is a biased estimator. This is undesirable, since if the likelihood indicates (via c_j) that the coefficient w_j should be large, we do not want to shrink it. We will discuss this issue in more detail in Section 13.6.2. Now we finally can understand why Tibshirani invented the term “lasso” in

(Tibshirani 1996): it stands for “least absolute selection and shrinkage operator”, since it selects a subset of the variables, and shrinks all the coefficients by penalizing the absolute values. If $\lambda = 0$, we get the OLS solution (of minimal 1 norm). If $\lambda \geq \lambda_{\max}$, we get $\hat{w} = 0$, where $\lambda_{\max} = \|XTy\|_{\infty} = \max_j |y^T x_j|$ (13.57)

This value is computed using the fact that 0 is optimal if $(XTy)_j \in [-\lambda, \lambda]$ for all j . In general, the maximum penalty for an 1 regularized objective is

$$\lambda_{\max} = \max_j |\nabla_j NLL(0)| \quad (13.58)$$

已思考 14s

下面先把两张图的图注和小节内容完整翻译（不漏译），再用“大白话”解释，最后给要点小结。

图注翻译

图 13.4 在点 x_0 处，一个函数的一些**次导数 (sub-derivative) **的示意。改绘自维基百科“Subderivative”。

图 13.5 左：软阈值 (平坦区间为 $[-\lambda, +\lambda]$)。**右：**硬阈值。

译文 (完整、不遗漏)

13.3.2 Lasso 的最优化条件

Lasso 的目标为

$$f(\theta) = RSS(\theta) + \lambda \|w\|_1 \quad (13.43)$$

不幸的是，只要某个 $w_j = 0$ ，项 $\|w\|_1$ 就不可导。这是**非光滑优化问题**的一个例子。

为处理非光滑函数，我们需要把“导数”的概念推广。设 $f : I \rightarrow \mathbb{R}$ 为 (凸) 函数，在点 θ_0 处，称标量 g 是 f 的一个**次导 (次梯度)**，如果

$$f(\theta) - f(\theta_0) \geq g(\theta - \theta_0), \quad \forall \theta \in I \quad (13.44)$$

其中 I 是包含 θ_0 的某个区间。示意见图 13.4。

把所有次导组成的集合定义为一个区间 $[a, b]$ ，其中 a, b 为单侧极限

$$a = \lim_{\theta \rightarrow \theta_0^-} \frac{f(\theta) - f(\theta_0)}{\theta - \theta_0}, \quad b = \lim_{\theta \rightarrow \theta_0^+} \frac{f(\theta) - f(\theta_0)}{\theta - \theta_0} \quad (13.46)$$

区间 $[a, b]$ 称为函数 f 在 θ_0 处的次微分 (subdifferential) , 记为 $\partial f(\theta)|_{\theta_0}$ 。例如, 绝对值函数 $f(\theta) = |\theta|$ 的次导为

$$\partial f(\theta) = \begin{cases} \{-1\}, & \theta < 0 \\ [-1, 1], & \theta = 0 \\ \{+1\}, & \theta > 0 \end{cases} \quad (13.47)$$

若函数处处可导, 则 $\partial f(\theta) = \{\frac{df(\theta)}{d\theta}\}$ 。类似标准微积分结论可证明: 点 $\hat{\theta}$ 是 f 的一个局部极小值, 当且仅当 $0 \in \partial f(\theta)|_{\hat{\theta}}$ 。

把这些概念用于 lasso。先忽略非平滑的惩罚项, 可证明 (练习 13.1)

$$\frac{\partial}{\partial w_j} \text{RSS}(w) = a_j w_j - c_j \quad (13.48)$$

$$a_j = 2 \sum_{i=1}^n x_{ij}^2 \quad (13.49)$$

$$c_j = 2 \sum_{i=1}^n x_{ij} (y_i - w_{-j}^\top x_{i,-j}) \quad (13.50)$$

其中 w_{-j} 表示去掉第 j 分量的 w , $x_{i,-j}$ 类似。可见 c_j 与第 j 个特征列 $x_{:,j}$ 和由其他特征造成的残差 $r_{-j} = y - X_{:,-j} w_{-j}$ 的相关性成正比。因此 $|c_j|$ 的大小表示在“考虑了其他特征及当前参数后”, 特征 j 对预测 y 的相关强度。

把惩罚项加进去, 次导为

$$\partial_{w_j} f(w) = (a_j w_j - c_j) + \lambda \partial_{w_j} \|w\|_1 \quad (13.51)$$

$$= \begin{cases} \{a_j w_j - c_j - \lambda\}, & w_j < 0 \\ [-c_j - \lambda, -c_j + \lambda], & w_j = 0 \\ \{a_j w_j - c_j + \lambda\}, & w_j > 0 \end{cases} \quad (13.52)$$

更紧凑地写为

$$(X^\top (Xw - y))_j \in \begin{cases} \{-\lambda\}, & w_j < 0 \\ [-\lambda, \lambda], & w_j = 0 \\ \{\lambda\}, & w_j > 0 \end{cases} \quad (13.53)$$

根据 c_j 的取值, $\partial_{w_j} f(w) = 0$ 的解有三种情况:

1. 若 $c_j < -\lambda$ (该特征与残差强负相关), 则在 $w_j = \frac{c_j + \lambda}{a_j} < 0$ 处为零。
2. 若 $c_j \in [-\lambda, \lambda]$ (该特征与残差弱相关), 则在 $w_j = 0$ 处为零。

3. 若 $c_j > \lambda$ (该特征与残差强正相关) , 则在 $\hat{w}_j = \frac{c_j - \lambda}{a_j} > 0$ 处为零。

总结为

$$\hat{w}_j(c_j) = \begin{cases} (c_j + \lambda)/a_j, & c_j < -\lambda \\ 0, & c_j \in [-\lambda, \lambda] \\ (c_j - \lambda)/a_j, & c_j > \lambda \end{cases} \quad (13.54)$$

也可写成

$$\hat{w}_j = \text{soft}\left(\frac{c_j}{a_j}; \frac{\lambda}{a_j}\right) \quad (13.55)$$

其中

$$\text{soft}(a; \delta) = \text{sign}(a)(|a| - \delta)_+, \quad (x)_+ = \max(x, 0). \quad (13.56)$$

这称为**软阈值**。图 13.5(a) 展示了以 c_j 为横轴、 \hat{w}_j 为纵轴时的曲线。虚线 $w_j = c_j/a_j$ 是最小二乘解；实线是正则化估计 $\hat{w}_j(c_j)$ ：把虚线整体下移（或上移） λ ，但当 $c_j \in [-\lambda, \lambda]$ 时，直接令 $w_j = 0$ 。

对比之下，图 13.5(b) 是**硬阈值**：在 $[-\lambda, \lambda]$ 内把 w_j 置 0，但在该区间外**不收缩**。软阈值曲线的斜率**不等于对角线**，意味着即使很大的系数也会被拉向 0；因此 lasso 是一个**有偏估计器**。这并不理想：若似然（通过 c_j ）表明 w_j 应当很大，我们并不希望再把它缩小。第 13.6.2 节将更详细讨论这一点。

现在也就能理解 Tibshirani 在 (Tibshirani, 1996) 中为何创造术语“lasso”了：它是“**最小绝对值选择与收缩算子**” (least absolute selection and shrinkage operator) 的缩写——既通过惩罚绝对值**选择**变量子集，又**收缩**所有系数。若 $\lambda = 0$ ，得到 OLS 解（在 OLS 解中取 ℓ_1 范数最小的那个）。若 $\lambda \geq \lambda_{\max}$ ，则 $\hat{w} = 0$ ，其中

$$\lambda_{\max} = \|X^T y\|_\infty = \max_j |y^T x_{:,j}| \quad (13.57)$$

这个结论来自事实：若对所有 j 有 $(X^T y)_j \in [-\lambda, \lambda]$ ，则 0 为最优解。一般而言，**任意** ℓ_1 正则目标的最大惩罚为

$$\lambda_{\max} = \max_j |\nabla_j \text{NLL}(0)|. \quad (13.58)$$

通俗解释（逐段讲清）

1) 为什么要“次导/次梯度”？

ℓ_1 里有 $|w_j|$ 。在 $w_j = 0$ 处没普通导数，但还是想写“驻点=0”的条件，于是用**次导**：凡是

能让“红直线是蓝函数在点 x_0 的支撑线”的斜率都算（图 13.4）。对 $|\theta|$ 而言：左斜率是 -1 ，右斜率是 $+1$ ，在 0 处“允许的斜率”是整个区间 $[-1, 1]$ 。

2) Lasso 的一阶最优条件长什么样？

- 先把 RSS 的导数写开： $\partial \text{RSS} / \partial w_j = a_j w_j - c_j$ 。
 - $a_j = 2 \sum x_{ij}^2$ (和第 j 列强度相关)。
 - $c_j = 2 \sum x_{ij} \times (\text{其他特征所剩下的残差}) \rightarrow$ 就是“**第 j 列与残差的相关性**”。
- 再加上 ℓ_1 惩罚的次导 (在 0 处是 $[-\lambda, \lambda]$)，得到式 (13.52)/(13.53) 的三段式条件。

3) 三种情形 = 软阈值

- 强负相关** $c_j < -\lambda$: 给个负的解 $(c_j + \lambda)/a_j$ (往 0 方向“加回 λ ”)。
- 弱相关** $|c_j| \leq \lambda$: 直接置零。
- 强正相关** $c_j > \lambda$: 给个正的解 $(c_j - \lambda)/a_j$ 。

把三段式合起来，就是

$$\hat{w}_j = \text{sign}\left(\frac{c_j}{a_j}\right) \left(\left| \frac{c_j}{a_j} \right| - \frac{\lambda}{a_j} \right)_+$$

即**软阈值** (图 13.5 左)：先把最小二乘的系数整体向 0 推 λ/a_j ，若推过了就变成 0。

4) 软 vs 硬阈值；偏差问题

- 硬阈值**: $|c_j| \leq \lambda$ 直接 0；其余保持原值 (图 13.5 右)。
- 软阈值**: 不仅会把小系数变 0，还会把**大系数也缩小一点** (因此有偏)。这点以后会谈怎么缓解 (如弹性网、后期去偏等)。

5) λ 的两个基准

- $\lambda = 0$: 退化成 OLS (在 OLS 等价解里选 ℓ_1 最小的那个)。
- $\lambda \geq \lambda_{\max}$: **全零解**，其中
 $\lambda_{\max} = \|X^T y\|_\infty$ (或更一般地 $\max_j |\nabla_j \text{NLL}(0)|$)。

要点小结

- 非光滑 \Rightarrow 用次导/次微分**: 在 $w_j = 0$ 处用 $[-1, 1]$ 作为 $|w_j|$ 的“允许斜率”。
- RSS 的导数**: $\partial \text{RSS} / \partial w_j = a_j w_j - c_j$ ，其中 c_j 量化“**第 j 列与残差相关性**”。
- Lasso 的一阶条件**: 合并 ℓ_1 次导得到三段式；解即**软阈值**:

$$\hat{w}_j = \text{soft}\left(\frac{c_j}{a_j}; \frac{\lambda}{a_j}\right)$$

——把小系数推到 0，把大系数也向 0 收缩。

- **软阈值 vs 硬阈值**: 硬阈值不收缩区间外的系数；软阈值会收缩全部 \rightarrow Lasso 有偏。
- λ 的极值: 0 时为 OLS; $\geq \lambda_{\max}$ 时全零；一般情形的 $\lambda_{\max} = \max_j |\nabla_j \text{NLL}(0)|$ 。
- **直觉**: 看与残差的相关性 c_j 。若相关性弱（落在 $[-\lambda, \lambda]$ ），就把该特征踢出；若强，就选入且轻度收缩。

这一小节主要在讲什么？我不太懂

这小节其实就是在回答两件事：Lasso 的“驻点条件”该怎么写，以及它为什么会被系数推成 0（软阈值）。因为 ℓ_1 在 0 处不可导，只能用“次梯度/次导数”来写最优条件。

核心思路（用最少的数学）

1. 为什么要次梯度？

$|w_j|$ 在 $w_j = 0$ 处没有普通导数，但允许的“斜率范围”是 $[-1, 1]$ 。非光滑最优化里：

最优点 \hat{w} 满足 0 落在目标的次梯度里： $0 \in \partial f(\hat{w})$ 。

2. 把 Lasso 的一阶条件写出来（线性回归）：

- 先对 RSS 求导： $\frac{\partial \text{RSS}}{\partial w_j} = a_j w_j - c_j$ 。
 - $a_j = 2 \sum_i x_{ij}^2$ （第 j 列强度）；
 - $c_j = 2 \sum_i x_{ij} \cdot$ （“其它变量拟合后剩下的残差”）
—— c_j 就是“第 j 个特征与残差的相关性”。
- 加上 ℓ_1 的次梯度（在 0 处是 $[-\lambda, \lambda]$ ），得到三段式条件：
 - 若 $c_j < -\lambda$: $\hat{w}_j = (c_j + \lambda)/a_j < 0$
 - 若 $|c_j| \leq \lambda$: $\hat{w}_j = 0$ （被剔除）
 - 若 $c_j > \lambda$: $\hat{w}_j = (c_j - \lambda)/a_j > 0$

3. 这三段式 = 软阈值（soft-thresholding）

$$\hat{w}_j = \text{soft}\left(\frac{c_j}{a_j}; \frac{\lambda}{a_j}\right) = \text{sign}(z)(|z| - \delta)_+,$$

把“最小二乘系数”整体向 0 推一段 (δ)，推过界就变 0。

- 图 13.5 左：软阈值（中间平段 $[-\lambda, \lambda]$ ）。
- 右图是硬阈值（只清零，不收缩），与 Lasso 不同。
- 因为软阈值会把大系数也稍微缩小，Lasso 有偏（bias）。

4. λ 如何控制稀疏

- $\lambda = 0$: 退化为普通最小二乘 (OLS)。
- λ 越大, $|c_j|$ 需要越大才“过阈值”被选中 \rightarrow **更稀疏**。
- 足够大时全零: $\lambda \geq \lambda_{\max} = \|X^T y\|_\infty \Rightarrow \hat{w} = 0$ 。
- 更一般的 ℓ_1 问题: $\lambda_{\max} = \max_j |\nabla_j \text{NLL}(0)|$ 。

图 13.4 在说什么

用一条“支撑线”的斜率来定义**次导**: 在不可导点 (比如绝对值的 0 点), 允许的斜率是一段区间, 而不是一个数。

一句话理解

看相关性 c_j : 弱 ($|c_j| \leq \lambda$) 就清零, 强就选入并向 0 轻度收缩 (软阈值)。这就是 Lasso “选择与收缩”的由来。

要点小结

- Lasso 目标非光滑 \Rightarrow **用次梯度最优条件**。
- c_j : 特征与当前残差的相关性, 是“该特征该不该留”的信号。
- **软阈值解**: 小于阈值清零, 超过阈值也会被**收缩** (\rightarrow 有偏)。
- λ 越大越稀疏; λ_{\max} 给出“全零”的门槛。
- 这套条件既解释了 Lasso **为什么稀疏**, 也给出了**坐标更新/求解器的闭式步** (后面算法会用到)。

你应该先向我解释好Lasso

已思考 28s

好的! 先把 Lasso (least absolute shrinkage and selection operator, 最小绝对收缩与选择算子) 用最直白的方式讲清楚, 再给你能直接上手的要点与做法。

Lasso 是什么?

在回归里, 我们既想**拟合好**, 又想**用更少的变量**避免过拟合。

Lasso 用下面这个目标同时做到两件事:

$$\min_{w, w_0} \underbrace{\text{RSS}(w)}_{\text{拟合误差}} + \lambda \underbrace{\|w\|_1}_{\text{绝对值之和}} \quad (\text{线性回归})$$

- w_0 是截距（通常不惩罚）。
- $\lambda \geq 0$ 是“收紧”强度：越大越稀疏。
- 也可以把 RSS 换成对数似然 → 逻辑回归等 GLM 一样能用 Lasso。

一句话：用“绝对值之和”的罚项 (ℓ_1) 去收缩所有系数，同时把很多小系数直接推到 0，就完成了**特征选择**。

为什么 Lasso 会把系数变成 0?

两层直觉：

1. 几何直觉

有约束写法： $\min \text{RSS} \text{ s.t. } \|w\|_1 \leq B$ 。

ℓ_1 约束是个带“尖角”的菱形（高维里是有很多“角”的多面体）。最优点是“红色等高线”第一次碰到约束边界的地方。**角最突出**，最容易被碰到；角对应某些坐标为 0 \Rightarrow 稀疏。

2. 代数直觉（软阈值）

对每个特征 j ，看它与“当前残差”的相关性 c_j 。

- 若 $|c_j| \leq \lambda$ ：直接把 w_j 设为 0；
- 若 $|c_j| > \lambda$ ：把最小二乘系数向 0 推回一截（收缩 λ ），仍可能非零。

这条分段公式就叫**软阈值**（soft-thresholding）。Lasso 的解本质上就是对每个坐标做软阈值。

对比：岭回归 (ℓ_2) 是“圆形”约束，**没有角**，只会让系数变小，很少变成严格 0，所以不做选择，只做收缩。

怎么解 Lasso (实用算法)

坐标下降（最常用，快、稳）

标准化数据（每列零均值、单位方差），截距不惩罚。重复下面步骤直到收敛：

1. 计算对第 j 列的**部分残差**： $r = y - X_{-j}w_{-j}$
2. 相关性： $z_j = x_j^\top r$ ，尺度： $a_j = \|x_j\|_2^2$
3. 软阈值更新：

$$w_j \leftarrow \frac{\text{sign}(z_j) \max(|z_j| - \lambda, 0)}{a_j}$$

整个过程对每个 j 来回扫一圈，很快就收敛。多数库（glmnet、sklearn）都用这个思想。

正则路径：从 $\lambda_{\max} = \|X^T y\|_\infty$ （此时全零）开始，逐步减小 λ ，沿着路径热启动，效率很高。

λ 怎么选？

- **K 折交叉验证**：在一串 λ 上评估验证误差，选最小的那一个；
 - **一倍标准误差规则 (1-SE)**：选更简单但验证误差在最优值 1 个标准误差之内的 λ （更稳健）。
 - 也可用信息准则 (AIC/BIC)，但 CV 更通用。
-

与其它方法怎么比？

- **岭回归 (ℓ_2)**：不稀疏，善于处理多重共线但不做选择；
 - **弹性网 ($\ell_1 + \ell_2$)**：在变量强相关时，Lasso 可能只选其中一个且不稳定；加入 ℓ_2 能成组入选、稳定性更好；
 - **子集选择 (ℓ_0)**：最直接但**非凸难解**；Lasso 是它的**凸近似**，可扩展到高维。
-

实战注意

- **一定要标准化特征**（截距除外），否则惩罚不公平。
 - **有偏**：大系数也会被收缩（软阈值），若关心无偏，可在选好子集后**再用无惩罚回归回拟一次**（“去偏”）。
 - **强相关特征**：优先考虑**弹性网**；或先做特征分组/主成分。
 - $p \gg n$ ：Lasso 依然可用（最多选到 n 个非零）。
 - **解释时报告**：选中变量、系数、 λ 、交叉验证误差、稳定性（可做重采样看选择频率）。
-

一个 10 秒小例子（只看更新规则）

假设某一步对特征 j 得到 $z_j = 0.35$ 、 $a_j = 1$ 、 $\lambda = 0.4$ 。

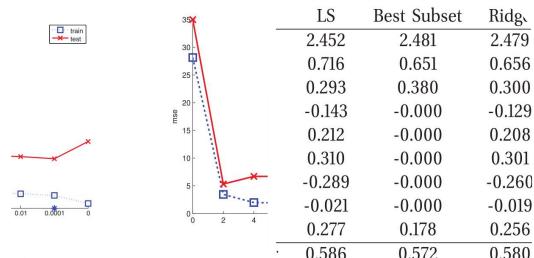
因为 $|z_j| = 0.35 \leq 0.4 \Rightarrow$ 软阈值变 0，于是这一列被剔除。

若把 λ 降到 0.2，则 $w_j = \text{sign}(0.35) \times (0.35 - 0.2)/1 = 0.15$ (被选入但收缩)。

要点小结

- 定义:** $\min \text{损失} + \lambda \|w\|_1$, λ 控制稀疏强度。
- 机理:** ℓ_1 的“角”+ 软阈值 \Rightarrow 大量系数被压成 0 (选择)；其余被拉向 0 (收缩)。
- 求解:** 坐标下降 = “相关性 $z_j \rightarrow$ 软阈值 \rightarrow 更新 w_j ”；用正则路径与交叉验证挑 λ 。
- 优缺点:** 可解释、可扩展、能做特征选择；对强相关敏感、估计有偏 (可用弹性网或去偏)。

如果你愿意，我可以用你手头的数据跑一遍 **坐标下降 + 交叉验证**，把“正则路径”和被选变量列表画出来。



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 13.6 (a) MSE vs λ for lasso for a degree 14 polynomial. Note that λ decreases as we move to the right. Figure generated by linregPolyLassoDemo. (b) MSE versus polynomial degree. Note that the model order increases as we move to the right. See Figure 1.18 for a plot of some of these polynomial regression models. Figure generated by linregPolyVsDegree.

Table 13.1 Results of different methods on the prostate cancer data, which has 8 features and 67 training cases. Methods are: LS = least squares, Subset = best subset regression, Ridge, Lasso. Rows represent the coefficients; we see that subset regression and lasso give sparse solutions. Bottom row is the mean squared error on the test set (30 cases). Based on Table 3.3. of (Hastie et al.

2009). Figure generated by prostateComparison.

13.3.3 Comparison of least squares, lasso, ridge and subset selection

We can gain further insight into 1 regularization by comparing it to least squares, and 2 and 0 regularized least squares. For simplicity, assume all the features of X are orthonormal, so $XTX = I$. In this case, the RSS is given by

$$RSS(w) = \|y - Xw\|^2 = y^T y + w^T X^T X w - 2w^T X^T y \quad (13.59)$$

$$= \text{const} + k w^2 - 2k i w_k x_i^T y_i \quad (13.60)$$

so we see this factorizes into a sum of terms, one per dimension. Hence we can write down the MAP and ML estimates analytically, as follows:

- MLE The OLS solution is given by

$$\hat{w}_{OLS,k} = x_k^T y \quad (13.61)$$

where x_k is the k 'th column of X . This follows trivially from Equation 13.60. We see that $\hat{w}_{OLS,k}$ is just the orthogonal projection of feature k onto the response vector (see Section 7.3.2).

- Ridge One can show that the ridge estimate is given by

$$\hat{w}_{Ridge,k} = \hat{w}_{OLS,k} + \lambda \quad (13.62)$$

- Lasso From Equation 13.55, and using the fact that $a_k = 2$ and $\hat{w}_{OLS,k} = c_k/2$, we have

$$\hat{w}_{Lasso,k} = \text{sign}(\hat{w}_{OLS,k}) |\hat{w}_{OLS,k}| - \lambda/2 \quad (13.63)$$

This corresponds to soft thresholding, shown in Figure 13.5(a).

- Subset selection If we pick the best K features using subset selection, the parameter estimate is as follows

$$\hat{w}_{SS,k} = \hat{w}_{OLS,k} \text{ if } \text{rank}(|\hat{w}_{OLS,k}|) \leq K \text{ otherwise} \quad (13.64)$$

where rank refers to the location in the sorted list of weight magnitudes. This corresponds to hard thresholding, shown in Figure 13.5(b).

Figure 13.6(a) plots the MSE vs λ for lasso for a degree 14 polynomial, and Figure 13.6(b) plots the MSE vs polynomial order. We see that lasso gives similar results to the subset selection method.

As another example, consider a data set concerning prostate cancer. We have $D = 8$ features and $N = 67$ training cases; the goal is to predict the log prostate-specific antigen levels (see (Hastie et al. 2009, p4) for more biological details). Table 13.1 shows that lasso gives

better prediction accuracy (at least on this particular data set) than least squares, ridge, and best subset regression. (In each case, the strength of the regularizer was chosen by cross validation.) Lasso also gives rise to a sparse solution. Of course, for other problems, ridge may give better predictive accuracy. In practice, a combination of lasso and ridge, known as the elastic net, often performs best, since it provides a good combination of sparsity and regularization (see Section 13.5.3).

已思考 23s

下面先把图注与正文完整翻译（不漏译），再给通俗解释，最后要点小结。

译文（完整）

图 13.6

(a) 14 次多项式的 lasso：均方误差 (MSE) 随 λ 变化的曲线。注意从左往右， λ 在减小。图由 `linregPolyLassoDemo` 生成。

(b) MLE (普通最小二乘) 随多项式阶数的 MSE 曲线。注意从左往右，模型阶数在增大。关于这些多项式回归模型的一部分可视化，见图 1.18。图由 `linregPolyVsDegree` 生成。

表 13.1

前列腺癌数据上的不同方法结果。该数据集有 8 个特征、67 个训练样本。方法包括：LS=最小二乘，Subset=最佳子集回归，Ridge=岭回归，Lasso。表中各行是系数；可以看到子集回归和 lasso 给出了稀疏解。最后一行是测试集（30 个样本）上的 MSE。改编自 (Hastie et al., 2009) 的表 3.3。图由 `prostateComparison` 生成。

13.3.3 最小二乘、lasso、岭回归与子集选择的比较

通过把 ℓ_1 正则与最小二乘、 ℓ_2 与 ℓ_0 正则化的最小二乘相比，我们可以获得进一步的直觉。为简单起见，假设 X 的所有特征列都正交且单位范数，即 $X^\top X = I$ 。在这种情况下，RSS 为

$$\text{RSS}(w) = \|y - Xw\|^2 = y^\top y + w^\top X^\top Xw - 2w^\top X^\top y \quad (13.59)$$

$$= \text{const} + \sum_k w_k^2 - 2 \sum_k w_k (x_{:k}^\top y) \quad (13.60)$$

因此它按维度分解成项之和。于是可以把 MAP 和 ML 的估计都写成解析式：

- MLE (普通最小二乘)

$$\hat{w}_k^{\text{OLS}} = x_{:k}^\top y \quad (13.61)$$

其中 $x_{:k}$ 是 X 的第 k 列。由式 (13.60) 直接得到。可见 \hat{w}_k^{OLS} 只是把响应向量在第 k 个特征方向上的正交投影 (见 7.3.2 节)。

- 岭回归

可以证明岭回归的估计为

$$\hat{w}_k^{\text{ridge}} = \frac{\hat{w}_k^{\text{OLS}}}{1 + \lambda}. \quad (13.62)$$

- Lasso

由式 (13.55) 并利用 $a_k = 2$ 与 $\hat{w}_k^{\text{OLS}} = c_k/2$ 可得

$$\hat{w}_k^{\text{lasso}} = \text{sign}(\hat{w}_k^{\text{OLS}}) \left(|\hat{w}_k^{\text{OLS}}| - \frac{\lambda}{2} \right)_+. \quad (13.63)$$

这对应于软阈值 (见图 13.5(a))。

- 子集选择

若用子集选择挑出最佳的 K 个特征，则参数估计为

$$\hat{w}_k^{\text{SS}} = \begin{cases} \hat{w}_k^{\text{OLS}}, & \text{若 } \text{rank}(|\hat{w}_k^{\text{OLS}}|) \leq K \\ 0, & \text{否则} \end{cases} \quad (13.64)$$

其中 rank 指在按绝对值大小排序列表中的位置。这对应于硬阈值 (见图 13.5(b))。

图 13.6(a) 画出了 14 次多项式的 lasso：MSE 随 λ 的变化；图 13.6(b) 画出了 MSE 随多项式阶数的变化。可以看到 lasso 给出的结果与子集选择方法相近。

再看一个例子：关于前列腺癌的数据集。我们有 $D = 8$ 个特征、 $N = 67$ 个训练样本；目标是预测对数化的 PSA (前列腺特异抗原) 水平 (更多生物学细节见 (Hastie et al., 2009) 第 4 页)。表 13.1 显示，在这个特定数据集上，lasso 比最小二乘、岭回归和最佳子集回归有更好的预测精度 (每个方法的正则强度都通过交叉验证选择)。Lasso 也得到一个稀疏解。当然，在其他问题上，岭回归可能给出更好的预测。在实践中，弹性网 (lasso + 岭) 往往效果最好，因为它在稀疏性与正则化之间给出了良好的折中 (见 13.5.3 节)。

通俗解释

1) 四个方法一图懂 (在“列正交”这个理想化前提下)

- 最小二乘 (OLS)：每个系数就是“该特征与目标的相关性”($x_{:k}^\top y$)。不做任何惩罚，容易过拟合。

- **岭回归 (ℓ_2)**：把 OLS 系数等比例缩小到 $\hat{w}/(1 + \lambda)$ 。不会把系数变为 0，**只收缩不选择**。
- **Lasso (ℓ_1)**：先看 OLS 系数的绝对值，如果不超过阈值 $\lambda/2$ 就**直接变 0**；超过阈值就**减去 $\lambda/2$** （软阈值）。因此既“选择”又“收缩”。
- **最佳子集 (ℓ_0)** **：只保留绝对值最大的 K 个 OLS 系数，其他全置 0（硬阈值）。能选得很“干脆”，但搜索开销大、易不稳定。

2) 两张图讲动态

- **图 13.6(a) (固定 14 次多项式, 调 λ)**：
 - 从左到右 λ 变小：
 - λ 很大 \rightarrow 模型太平滑/太简单 \rightarrow 训练误差小幅、测试误差**偏大**（欠拟合）；
 - 逐步减小 \rightarrow 测试 MSE 先**下降并达到最佳**；
 - 再继续减小到接近 0 \rightarrow 约束消失、模型过于灵活 \rightarrow 测试 MSE **回升**（过拟合）。
 - 这说明 λ 就像“连续版的模型阶数”控制器。
- **图 13.6(b) (最小二乘, 调多项式阶数)**：
 - 阶数从小到大：测试 MSE 先降后升；高阶时急剧上升（严重过拟合）。
 - 这和 (a) 的趋势呼应：lasso 调 λ 的效果，和直接调模型阶数很像，但 lasso 不需要显式枚举子集。

3) 表 13.1 (前列腺癌) 告诉我们什么？

- **稀疏性**：子集与 lasso 把一些特征的系数设为 0，而岭回归只是缩小。
- **预测**：在这个数据集上，Lasso 的测试 MSE 最低 (0.564)，优于子集 (0.572)、岭 (0.580)、最小二乘 (0.586)。
- **结论不绝对**：换数据集可能岭回归更好；**弹性网**常更稳，因为它“既选又稳” ($\ell_1 + \ell_2$)。

要点小结

- 在列正交的简化情形下：
 - OLS： $\hat{w}_k = x_{:,k}^\top y$ （投影）。
 - Ridge： $\hat{w}_k = \hat{w}_k^{\text{OLS}} / (1 + \lambda)$ （均匀缩小）。
 - Lasso： $\hat{w}_k = \text{soft}(\hat{w}_k^{\text{OLS}}; \lambda/2)$ （阈值+收缩）。
 - Subset：取 OLS 绝对值最大的 K 个（硬阈值）。
- **性能曲线**：无论调 λ 还是调阶数，测试误差通常**先降后升**（偏差-方差权衡）。

- **实证：**在前列腺癌例子里，lasso 最好且稀疏；但通用做法是用交叉验证选 λ ，并把 **弹性网** 作为默认强力基线。
- **直觉复盘：** $\ell_2 \rightarrow$ 收缩不选； $\ell_1 \rightarrow$ 既收缩又选； $\ell_0 \rightarrow$ 只选不收缩（但难解）。

↳ **直觉复盘**： $\ell_2 \rightarrow$ 收缩不选； $\ell_1 \rightarrow$ 既收缩又选； $\ell_0 \rightarrow$ 只选不收缩（但难解）

这是什么意思

已思考 18s

这句“直觉复盘： $\ell_2 \rightarrow$ 收缩不选； $\ell_1 \rightarrow$ 既收缩又选； $\ell_0 \rightarrow$ 只选不收缩（但难解）”的意思是：

- **ℓ_2 (岭回归) —“收缩不选”**

给系数加平方罚 ($\sum w_j^2$)。作用是把**所有系数都往 0 拉小**（收缩），但很少把谁变成**严格 0**，所以**不会真正删掉变量**。优点是稳定、抗多重共线；缺点是模型不稀疏、解释性差。

- **ℓ_1 (Lasso) —“既收缩又选”**

给系数加绝对值罚 ($\sum |w_j|$)。它会把**很多小系数直接压成 0**（选/不选），同时把**剩下的系数也往 0 拉一点**（收缩）。所以**既能做特征选择又能防过拟合**；但系数会有偏差，而且遇到强相关特征时可能只留一个、选择不稳定。

- **ℓ_0 (最佳子集) —“只选不收缩（但难解）”**

直接惩罚**非零系数的个数** ($\|w\|_0$)。它会**挑出一个子集**，把其他系数设为 0，但**保留下来的系数不再缩小**（通常用 OLS 估它们）。效果最接近“纯粹的选变量”，但这是**组合优化**（要在 2^D 个子集中找最优），**计算非常难、容易不稳定**。

一个 10 秒的小例子

设最小二乘（不加罚）得到 $w_{OLS} = [1.2, 0.3, 0.05]$ 。

- **$\ell_2 (\lambda=1)$** → 大家一起缩小：[0.6, 0.15, 0.025]（**没有 0**）。
- **$\ell_1 (\lambda \approx 0.2)$** → 软阈值：[1.0, 0.1, 0]（**把很小的那维清零，其余也被拉小**）。
- **ℓ_0 (选 2 个变量)** → 只留绝对值最大的两个：[1.2, 0.3, 0]（**清零但不缩小**）。

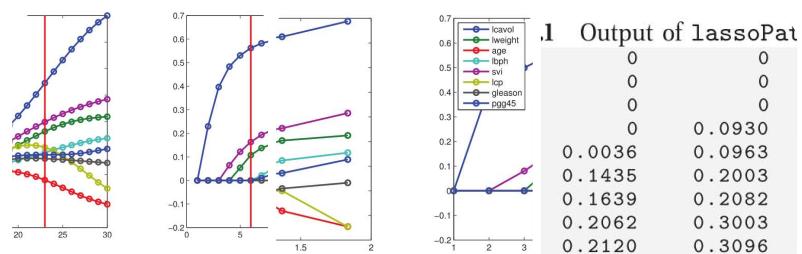
什么时候用谁？

- 想要**稳定预测**、特征很多且相关强 → ℓ_2 或 **弹性网** ($\ell_1+\ell_2$)。

- 想要自动选变量、模型可解释且稀疏 $\rightarrow \ell_1$ (Lasso) 。
- 维度不大、愿意花算力找最小子集 $\rightarrow \ell_0/\text{最佳子集}$ (或用近似贪心法) 。

一句话：

- ℓ_2 : 把所有系数“缩小”但不删人；
- ℓ_1 : 既“删人”（很多设为 0），又“缩小”幸存者；
- ℓ_0 : 只“删人”不“缩小”，但找最优删谁很难。



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 13.7 (a) Profiles of ridge coefficients for the prostate cancer example vs bound on 2 norm of w, so small t (large λ) is on the left. The vertical line is the value chosen by 5-fold CV using the 1SE rule. Based on Figure 3.8 of (Hastie et al. 2009). Figure generated by ridgePathProstate. (b) Profiles of lasso coefficients for the prostate cancer example vs bound on 1 norm of w, so small t (large λ) is on the left. Based on Figure 3.10 of (Hastie et al. 2009). Figure generated by lassoPathProstate.

Figure 13.8 Illustration of piecewise linearity of regularization path for lasso on the prostate cancer example. (a) We plot $\hat{w}_j(B)$ vs B for the critical values of B. (b) We plot vs steps of the LARS algorithm. Figure generated by lassoPathProstate.

13.3.4 Regularization path

As we increase λ , the solution vector $w^*(\lambda)$ will tend to get sparser, although not necessarily monotonically. We can plot the values $\hat{w}_j(\lambda)$ vs λ for each feature j; this is known as the regularization path.

This is illustrated for ridge regression in Figure 13.7(a),

where we plot $\hat{w}_j(\lambda)$ as the regularizer λ decreases. We see that when $\lambda = \infty$, all the coefficients are zero. But for any finite value of λ , all coefficients are non-zero; furthermore, they increase in magnitude as λ is decreased. In Figure 13.7(b), we plot the analogous result for lasso. As we move to the right, the upper bound on the 1 penalty, B , increases. When $B = 0$, all the coefficients are zero. As we increase B , the coefficients gradually “turn on”. But for any value between 0 and $B_{\max} = \|\hat{w}_{OLS}\|_1$, the solution is sparse.

Remarkably, it can be shown that the solution path is a piecewise linear function of B (Efron et al. 2004). That is, there are a set of critical values of B where the active set of non-zero coefficients changes. For values of B between these critical values, each non-zero coefficient increases or decreases in a linear fashion. This is illustrated in Figure 13.8(a). Furthermore, one can solve for these critical values analytically. This is the basis of the LARS algorithm (Efron et al. 2004), which stands for “least angle regression and shrinkage” (see Section 13.4.2 for details).

Remarkably, LARS can compute the entire regularization path for roughly the same computational cost as a single least squares fit (namely $O(\min(ND^2, DN^2))$).

In Figure 13.8(b), we plot the coefficients computed at each critical value of B . Now the piecewise linearity is more evident. Below we display the actual coefficient values at each step along the regularization path (the last line is the least squares solution):

By changing B from 0 to B_{\max} , we can go from a solution in which all the weights are zero to a solution in which all weights are non-zero. Unfortunately, not all subset sizes are achievable using lasso. One can show that, if $D > N$, the optimal solution can have at most N variables in it, before reaching the complete set corresponding to the OLS solution of minimal 1 norm. In Section 13.5.3, we will see that by using an L^2 regularizer as well as an L^1 regularizer (a method known as the elastic net), we can achieve sparse solutions which contain more variables than training cases. This lets us explore model sizes between N and D .

下面先把图注与正文完整翻译（不漏译），再用通俗解释说明，最后给要点小结。

图注翻译

图 13.7

- (a) 前列腺癌示例的岭回归系数随 $\|w\|_2$ 的上界 (记作 t) 的轨迹图; 因此小 t (也就是大 λ) 在左侧。红色竖线是使用 5 折交叉验证并采用 1SE 规则选出的取值。改绘自 (Hastie et al., 2009) 图 3.8。图由 `ridgePathProstate` 生成。
- (b) 前列腺癌示例的 lasso 系数随 $\|w\|_1$ 的上界 (记作 t) 的轨迹图; 因此小 t (也就是大 λ) 在左侧。改绘自 (Hastie et al., 2009) 图 3.10。图由 `lassoPathProstate` 生成。

图 13.8 说明了前列腺癌示例中 lasso 的正则化路径呈分段线性的性质。(a) 我们在若干临界的 B 取值处, 绘制 $\hat{w}_j(B)$ 与 B 的关系。(b) 我们按 LARS 算法的步数绘制系数的变化。图由 `lassoPathProstate` 生成。

清单 13.1 `lassoPathProstate` 的输出: 沿着 lasso 正则化路径, 每一步 (每个临界 B) 对应的一组系数; 最后一行是最小二乘解。

正文翻译

13.3.4 正则化路径

当我们增大 λ 时, 解向量 $\hat{w}(\lambda)$ 往往会更稀疏 (虽然不一定严格单调)。我们可以对每个特征 j 作图, 画出 $\hat{w}_j(\lambda)$ 随 λ 的变化; 这被称为**正则化路径** (regularization path)。

图 13.7(a) 展示了岭回归的情况, 我们在减小正则强度 λ 时绘制 $\hat{w}_j(\lambda)$ 。可见当 $\lambda = \infty$ 时, 所有系数为 0; 但对任何有限的 λ , 所有系数都是非零的; 并且当 λ 继续减小时, 它们的绝对值会增大。

在图 13.7(b) 中, 我们画出了 lasso 的对应结果。向右移动时, ℓ_1 罚的上界 B 在增大。当 $B = 0$ 时, 所有系数为 0。随着 B 的增大, 系数逐渐被“激活”。但是在 0 到 $B_{\max} = \|\hat{w}_{OLS}\|_1$ 之间的任何取值下, 解都是稀疏的。

一个很有意思的事实是: 可以证明解路径是 B 的**分段线性函数** (Efron 等, 2004)。也就是说, 存在一些 B 的临界值, 使得**非零系数的活跃集合**在这些点发生变化; 在任意两个临界值之间, 每个非零系数都按**线性**方式增加或减少。这在图 13.8(a) 中有所说明。并且, 这些临界值可以**解析地**求出。这就是 LARS 算法 (“最小角回归与收缩”) 的基础 (细节见 13.4.2)。令人惊讶的是, LARS 计算**整条**正则化路径所需的时间, 大致与一次最小二乘拟合相当 (复杂度约为 $O(\min(ND^2, DN^2))$)。

在图 13.8(b) 中，我们绘制了每个**临界值** B 处得到的系数；此时“分段线性”的特征更明显。下方给出了沿正则化路径每一步的实际系数值（最后一行是最小二乘解）：

(对应清单 13.1 的数值)

通过把 B 从 0 改到 B_{\max} ，我们可以从“所有权重为 0 的解”走到“所有权重均非零的解”。不幸的是，用 lasso 并不能实现**所有的子集大小**。可以证明，如果 $D > N$ ，在达到对应于“ ℓ_1 范数最小的 OLS 解”的完整集合之前，最优解中最多只有 N 个变量是非零的。在 13.5.3 节我们将看到，通过同时使用 ℓ_2 与 ℓ_1 正则（即**弹性网**），我们可以得到**包含的变量数多于训练样本数**的稀疏解，从而让我们能够探索介于 N 与 D 之间的模型规模。

通俗解释（逐点讲清）

1) 正则化路径是啥？

把惩罚强度慢慢从“非常强”（ λ 很大、或 B 很小）调到“很弱”（ λ 很小、或 B 很大），记录**每个系数的取值轨迹**。这条“轨迹”叫正则化路径。它告诉你哪个变量在什么时候被选入、系数怎么变化。

2) 岭 vs Lasso 的路径

- **岭** (ℓ_2)：除了 $\lambda = \infty$ 的极端点，其他任何有限 λ **都不为 0**——所以路径上所有线从**不贴在 0 上**，只是整体从 0 平滑长出来。
- **Lasso** (ℓ_1)：从左（大 λ 、小 B ）开始，大家全是 0；随着 B 增大，**一条条曲线从 0 处“抬头”**——谁先抬头，说明谁更相关；而且**在很大范围内解保持稀疏**（很多系数仍是 0）。

3) 为什么说 Lasso 的路径是“分段线性”？

在某些关键的 B 值（临界点）上，**活跃集合**发生变化：

- 新变量进入（从 0 变非零）或已有变量退出（回到 0）；
- 在两个相邻临界点之间，所有当前非零的系数**线性**地随 B 变化。

这就是图 13.8(a)(b) 看到的“折线感”。LARS 就是沿这些临界点一步步走，把整条路径一次性算出来。

4) 交叉验证怎么选解？

图 13.7(a)(b) 的**红竖线**：用 5 折 CV 选到的“好点”。“**1SE 规则**”是：选一个**更简单**（更稀疏）的模型，只要它的验证误差**在最优误差的 1 个标准误差范围内**——稳健不易过拟合。

5) Lasso 的一个“上限”

当**特征数 D 大于样本数 N** 时，Lasso 的最优解**最多只有 N 个非零系数**。所以如果你想探

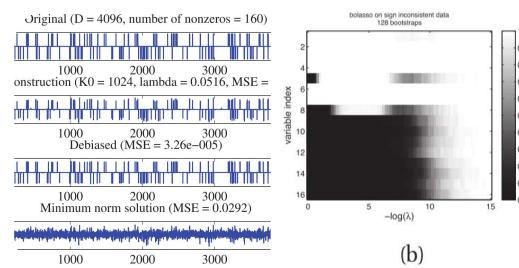
索“比 N 还多的变量”那一段模型规模，Lasso 做不到；这时用弹性网 $(\ell_1 + \ell_2)$ 可以突破这个限制，同时保持一定稀疏与稳定性。

6) 清单 13.1 怎么看？

每一列是一组特征的系数；每一行对应路径上的一个临界点（LARS 的一步）。从上到下，越来越多的系数开始离开 0，直到最后一行变成 OLS 的非稀疏解。

要点小结

- 正则化路径：** $\hat{w}_j(\lambda)$ 或 $\hat{w}_j(B)$ 随惩罚强度的轨迹；展示变量的“进场顺序”和系数变化。
- 岭：** 除了 $\lambda = \infty$ 外，系数不为 0；路径平滑增长 \Rightarrow 只收缩不选择。
- Lasso：** 从全 0 开始，系数分批激活；在 $0 < B < B_{\max}$ 的大区间里解是稀疏的。
- 分段线性：** Lasso 的路径在 B 上是分段线性；LARS 可在“接近一次 OLS 的代价”下把整条路径算出。
- 选择模型：** 用交叉验证（常配 1SE 规则）在路径上定位一个稀疏而泛化好的解。
- 规模限制：** 若 $D > N$ ，Lasso 的非零最多 N 个；弹性网能得到多于 N 个非零、在稀疏与稳定间更折中。



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 13.9 Example of recovering a sparse signal using lasso. See text for details. Based on Figure 1 of (Figueiredo et al. 2007).

Figure 13.10 (a) Probability of selection of each variable (white = large probabilities, black = small probabilities) vs. regularization parameter for Lasso. As we move from left to right, we decrease the amount of regularization, and therefore select more variables. (b) Same as (a) but for

bolasso. (c) Probability of correct sign estimation vs. regularization parameter. Bolasso (red, dashed) and Lasso (black, plain): The number of bootstrap replications is in {2, 4, 8, 16, 32, 64, 128, 256}. Based on Figures 1-3 of (Bach 2008). Figure generated by bolassoDemo.

13.3.5 Model selection

It is tempting to use L_1 regularization to estimate the set of relevant variables. In some cases, we can recover the true sparsity pattern of w^* , the parameter vector that generated the data. A method that can recover the true model in the $N \rightarrow \infty$ limit is called model selection consistent. The details on which methods enjoy this property, and when, are beyond the scope of this book; see e.g., (Buhlmann and van de Geer 2011) for details.

Instead of going into a theoretical discussion, we will just show a small example. We first generate a sparse signal w^* of size $D = 4096$, consisting of 160 randomly placed ± 1 spikes. Next we generate a random design matrix X of size $N \times D$, where $N = 1024$. Finally we generate a noisy observation $y = Xw^* + \epsilon$, where $\epsilon \sim N(0, 0.012)$. We then estimate w from y and X . The original w^* is shown in the first row of Figure 13.9. The second row is the L_1 estimate $w^{\hat{L}1}$ using $\lambda = 0.1\lambda_{\max}$. We see that this has "spikes" in the right places, but they are too small. The third row is the least squares estimate of the coefficients which are estimated to be non-zero based on $\text{supp}(\hat{w}^{L1})$. This is called debiasing, and is necessary because lasso shrinks the relevant coefficients as well as the irrelevant ones. The last row is the least squares estimate for all the coefficients jointly, ignoring sparsity. We see that the (debiased) sparse estimate is an excellent estimate of the original signal. By contrast, least squares without the sparsity assumption performs very poorly.

Of course, to perform model selection, we have to pick λ . It is common to use cross validation. However, it is important to note that cross validation is picking a value of λ that results in good predictive accuracy. This is not usually the same value as the one that is likely to recover the "true" model. To see why, recall that L_1 regularization performs selection and shrinkage, that is, the chosen coefficients are brought closer to 0. In order to prevent relevant coefficients from being shrunk in this way, cross

validation will tend to pick a value of λ that is not too large. Of course, this will result in a less sparse model which contains irrelevant variables (false positives). Indeed, it was proved in (Meinshausen and Bühlmann 2006) that the prediction-optimal value of λ does not result in model selection consistency. In Section 13.6.2, we will discuss some adaptive mechanisms for automatically tuning λ on a per-dimension basis that does result in model selection consistency. A downside of using 1 regularization to select variables is that it can give quite different results if the data is perturbed slightly. The Bayesian approach, which estimates posterior marginal inclusion probabilities, $p(\gamma_j = 1 | D)$, is much more robust. A frequentist solution to this is to use bootstrap resampling (see Section 6.2.1), and to rerun the estimator on different versions of the data. By computing how often each variable is selected across different trials, we can approximate the posterior inclusion probabilities. This method is known as stability selection (Meinshausen and Bühlmann 2010).

We can threshold the stability selection (bootstrap) inclusion probabilities at some level, say 90%, and thus derive a sparse estimator. This is known as bootstrap lasso or bolasso (Bach 2008). It will include a variable if it occurs in at least 90% of sets returned by lasso (for a fixed λ). This process of intersecting the sets is a way of eliminating the false positives that vanilla lasso produces. The theoretical results in (Bach 2008) prove that bolasso is model selection consistent under a wider range of conditions than vanilla lasso. As an illustration, we reproduced the experiments in (Bach 2008). In particular, we created 256 datasets of size $N = 1000$ with $D = 16$ variables, of which 8 are relevant. See (Bach 2008) for more detail on the experimental setup. For dataset n , variable j , and sparsity level k , define $S(j, k, n) = \mathbb{I}(\hat{w}_j(\lambda_k, D_n) \neq 0)$. Now define $P(j, k)$ be the average of $S(j, k, n)$ over the 256 datasets. In Figure 13.10(a-b), we plot P vs $-\log(\lambda)$ for lasso and bolasso. We see that for bolasso, there is a large range of λ where the true variables are selected, but this is not the case for lasso. This is emphasized in Figure 13.10(c), where we plot the empirical probability that the correct set of variables is recovered, for lasso and for bolasso with an increasing number of bootstrap samples.

Of course, using more samples takes longer. In practice, 32 bootstraps seems to be a good compromise between speed and accuracy. With bolasso, there is the usual issue of picking λ . Obviously we could use cross validation, but plots such as Figure 13.10(b) suggest another heuristic: shuffle the rows to create a large black block, and then pick λ to be in the middle of this region. Of course, operationalizing this intuition may be tricky, and will require various ad-hoc thresholds (it is reminiscent of the “find the knee in the curve” heuristic discussed in Section 11.5.2 when discussing how to pick K for mixture models). A Bayesian approach provides a more principled method for selecting λ .

13.3.6 Bayesian inference for linear models with Laplace priors

We have been focusing on MAP estimation in sparse linear models. It is also possible to perform Bayesian inference (see e.g., (Park and Casella 2008; Seeger 2008)). However, the posterior mean and median, as well as samples from the posterior, are not sparse; only the mode is sparse. This is another example of the phenomenon discussed in Section 5.2.1, where we said that the MAP estimate is often untypical of the bulk of the posterior. Another argument in favor of using the posterior mean comes from Equation 5.108, which showed that plugging in the posterior mean, rather than the posterior mode, is the optimal thing to do if we want to minimize squared prediction error. (Schniter et al. 2008) shows experimentally, and (Elad and Yavnch 2009) shows theoretically, that using the posterior mean with a spike-and-slab prior results in better prediction accuracy than using the posterior mode with a Laplace prior, albeit at slightly higher computational cost. variable index variable index P(correct support)

下面先把两幅图和两小节**完整翻译**（不漏译），再用**通俗解释说明**，最后给**要点小结**。

图注翻译

图 13.9 用 lasso 恢复稀疏信号的一个例子。细节见正文。改绘自 (Figueiredo et al., 2007) 的图 1。

(图中从上到下：原始信号（维度 $D = 4096$, 非零个数 = 160）； ℓ_1 重建 ($K_0 = 1024$, $\lambda = 0.0516$, $MSE=0.0027$)；去偏后的结果 ($MSE=3.26e-005$)；最小范数（最小二乘）解 ($MSE=0.0292$)。)

图 13.10

- (a) Lasso：每个变量被选中的概率（白=大概率，黑=小概率）随正则化参数的变化。越往右正则越弱，因此会选更多变量。
 - (b) 与 (a) 相同，但方法为 bolasso。
 - (c) 正确符号（正确支持集）被估计出来的概率随正则化参数的变化。红色虚线为 bolasso，黑色实线为 lasso；自举（bootstrap）重复次数 $\in \{2, 4, 8, 16, 32, 64, 128, 256\}$ 。改绘自 (Bach, 2008) 的图 1-3。图由 `bolassoDemo` 生成。
-

13.3.5 模型选择（翻译）

使用 ℓ_1 正则化来估计“相关变量的集合”很有吸引力。在某些情形下，我们可以恢复出生成数据的参数向量 w^{*} 的真实稀疏模式。若某个方法在 $N \rightarrow \infty$ 的极限下能恢复出**真实模型**，则称其为**模型选择一致**（model selection consistent）。哪些方法在何种条件下满足这一性质，细节超出本书范围；可参见 (Bühlmann and van de Geer, 2011)。我们不做理论讨论，只展示一个小例子。

我们先生成一个维度 $D = 4096$ 的稀疏信号 w^{*} ，它由 160 个随机位置的 ± 1 脉冲组成。再生成大小为 $N \times D$ 的随机设计矩阵 X ，其中 $N = 1024$ 。最后生成带噪的观测 $y = Xw^{*} + \varepsilon$ ，其中 $\varepsilon_i \sim N(0, 0.01^2)$ 。然后用 y 与 X 来估计 w 。原始 w^{*} 在图 13.9 第一行。第二行是 ℓ_1 估计 \hat{w}_{L1} ($\lambda = 0.1 \lambda_{\max}$)。可以看到“脉冲”的位置对了，但幅值偏小。第三行是在 $\text{supp}(\hat{w}_{L1})$ (\hat{w}_{L1} 判定为非零的支撑集) 上做的**最小二乘**估计，这称为**去偏**（debiasing），因为 lasso 会把相关和不相关的系数都向 0 收缩。最后一行是在**所有系数**上做的**最小二乘**（忽略稀疏性）。可见，**（去偏后的）稀疏估计**对原始信号的近似非常好；相比之下，不考虑稀疏的**最小二乘**表现很差。

当然，要做模型选择必须选 λ 。通常用交叉验证。然而要注意：交叉验证选的是**预测精度好的** λ ，这通常**不是能恢复“真实模型”的那个值**。原因在于， ℓ_1 正则既做**选择**也做**收缩**：被选中的系数会被往 0 拉。为了不把有用的系数也拉小，交叉验证会倾向于选一个**不太大**的 λ 。这会导致模型**不够稀疏**，包含无关变量（**假阳性**）。事实上，(Meinshausen and Bühlmann, 2006) 证明：**预测最优**的 λ 并不会带来模型选择一致性。在 13.6.2 节，我们将讨论按维度自适应地调 λ 的方法，它们可以达到模型选择一致性。

用 ℓ_1 做变量选择的一个缺点是：数据稍有扰动，结果可能差异很大。贝叶斯方法（估计后验边际纳入概率 $p(\gamma_j = 1 | D)$ ）要稳健得多。一个频率学派的解决方案是**自举 (bootstrap) 重采样**（见 6.2.1 节），在不同版本的数据上重复运行估计器。通过统计每个变量被选中的频率，我们可以近似后验纳入概率。这称为**稳定性选择 (stability selection)** (Meinshausen and Bühlmann, 2010)。

我们可以把稳定性选择（自举）的纳入概率在某个阈值（比如 90%）处**阈值化**，从而得到一个稀疏估计器。这称为**自举 lasso** 或 **bolasso** (Bach, 2008)：如果一个变量在（固定 λ 下）由 lasso 返回的集合中出现的频次至少为 90%，就把它纳入。把这些集合取交集是一种剔除 vanilla lasso 产生的假阳性的方法。(Bach, 2008) 的理论结果证明：**bolasso 在更宽的条件下是模型选择一致的**。作为演示，我们复现实验：构造 256 个数据集，每个 $N = 1000$ 、 $D = 16$ ，其中 8 个是真正相关的。更详细的设置见 (Bach, 2008)。对数据集 n 、变量 j 、稀疏程度 k ，定义

$$S(j, k, n) = \mathbf{1}(\hat{w}_j(\lambda_k, \mathbf{D}_n) \neq 0).$$

令 $P(j, k)$ 为 $S(j, k, n)$ 在 256 个数据集上的平均。图 13.10(a-b) 画出了 lasso 与 bolasso 的 P 随 $-\log \lambda$ 的变化。可以看到：**bolasso 在一个很大的 λ 区间内都能选到真正的变量**，而 lasso 做不到。图 13.10(c) 进一步强调了这一点：我们画出了“恢复正确变量集合”的经验概率；随着 bootstrap 次数增加，bolasso（红虚线）明显优于 lasso（黑实线）。当然，更多的采样会更耗时；实践中 32 次**自举**在速度与准确率之间往往是个不错的折中。

使用 bolasso 仍需选 λ 。可以用交叉验证，但像图 13.10(b) 这样的图也启发了另一个启发式：在图中“把行顺一下”，形成一大块黑色区域，然后把 λ 选在这个区域的中间。当然，要把这种直觉工程化并不容易，需要一些经验阈值（这让人联想到 11.5.2 节中“在曲线拐点处取值”的启发式，用于混合模型里选 K ）。**贝叶斯方法**提供了一种更为原则性的 λ 选择方式。

13.3.6 带拉普拉斯先验的线性模型的贝叶斯推断（翻译）

在稀疏线性模型中，我们一直关注 MAP 估计。也可以进行**贝叶斯推断**（例如 Park & Casella, 2008; Seeger, 2008）。但是，**后验均值与后验中位数**，以及从后验中抽样得到的样本，都不是稀疏的；只有**后验众数 (MAP)** 是稀疏的。这再次体现了 5.2.1 节讨论的现象：**MAP 往往并不代表后验的大部分质量。而支持用后验均值的另一个论据来自式 (5.108)：如果我们的目标是最小化平方预测误差**，把后验均值代入要比把后验众数代入更优。（Schniter et al., 2008）在实验上、(Elad and Yavneh, 2009) 在理论上都表明：在**spike-and-slab 先验**下使用**后验均值**，比在**拉普拉斯先验**下使用**后验众数**，能得到更好的预测精度（代价是稍高的计算量）。

通俗解释

1) 用 Lasso 做“真模型”的找回：要小心

- Lasso 的 λ 用交叉验证选，**优化的是预测误差**，而不是“把真变量全找回”。预测最优的 λ 往往**太小**，会把一些无关变量也放进来（假阳性），因此**不保证**“模型选择一致”。
- 一种简单补救是去偏（debias）：先用 Lasso 找到“可能非零”的位置，再在这些位置上用**无惩罚的最小二乘**重估系数（幅值更接近真值）。图 13.9 就展示了：
 - 仅 ℓ_1 重建：位置对、幅值小；
 - 去偏后：幅值也对，MSE 显著下降；
 - 完全不用稀疏假设的最小二乘：被噪声拖垮。

2) 选择稳定性：用自举投票 (bolasso)

- 单次 Lasso 对数据扰动很敏感；bootstrap 多次抽样、重复 Lasso，把**出现频率**当作“被选概率”，再阈值化（比如 $\geq 90\%$ 就留下）。
- 这等价于“多数表决 + 交集消假阳性”。理论与实验证明：bolasso **更容易恢复真模型**。
- 代价：多跑几次 \Rightarrow 耗时增加；经验上 32 次是个好平衡。

3) 贝叶斯视角：别只看 MAP

- $\ell_1 + \text{MAP}$ 给稀疏、好解释；但若追求**预测**，用**后验均值**往往更优，尤其配 spike-and-slab 先验（虽然计算更贵）。
- 这也解释了为何“Lasso 的系数偏小要去偏”：MAP 会收缩；而后验均值在合适先验下能综合不确定性、预测更好。

要点小结

- **Lasso 重建稀疏信号**：能找对位置，但幅值被收缩；**在选出的子集上二次拟合（去偏）**可显著提升精度。
- **交叉验证选 λ** ：倾向于**预测最优**而非**选择最优**；不一定恢复真支持集（不具“选择一致性”）。
- **稳定性选择 / bolasso**：对多次 bootstrap 的选择结果做投票（或交集）；**更稳健**、更接近真模型，代价是多次训练。
- **正则路径解读**图 13.10：bolasso 的“白区”更宽，表示在较大范围的 λ 上能稳定选到真变量；恢复正确集合的概率也更高。

- **贝叶斯推断**: 后验众数 (MAP) 稀疏但不代表后验主体; 就**预测损失 (MSE) **而言, **后验均值 + spike-and-slab** 往往优于 **拉普拉斯 + MAP**。
- **实务建议**:
 1. 选子集: 考虑 **稳定性选择/bolasso** 或 **弹性网**;
 2. 估系数: 先 Lasso 选、再去偏;
 3. 关注预测: 可优先考虑**后验均值或做模型平均**来利用不确定性。

这一小节主要在讲什么? 我不太懂

已思考 9s

这小节在讲两件事: **用 L1 (Lasso) 做“模型选择”时的能力与局限, 以及更稳、更准的改进做法** (含自举版 lasso 与贝叶斯思路)。

通俗解释

1. **Lasso 能找稀疏但会“缩水”**
 - 例子里真信号只有少量非零 (稀疏)。
 - Lasso 基本能把**非零的位置**找对, 但因为“软阈值收缩”, **幅值被压小**。
 - 解决: **去偏 (debias)** ——先用 Lasso 选出可能非零的坐标, 再只在这些坐标上**做无惩罚最小二乘重估**, 幅度就回来了, 预测误差显著下降。
 - 若完全不用稀疏假设直接最小二乘, 会被噪声淹没, 效果很差。
2. **交叉验证挑的 λ 更偏向“预测好”, 不一定“选对人”**
 - CV 的目标是**最小化预测误差**, 因此会避免把重要系数收缩太狠, 通常得到**更小的 $\lambda \rightarrow$ 模型不够稀疏**, 容易带入无关变量 (假阳性)。
 - 理论上, “预测最优的 λ ”一般**不具备**“把真支持集恢复出来”的一致性。
3. **Lasso 的选择会不稳定: 数据小扰动, 结果可能变**
 - 频率学派的补救: **稳定性选择 / bolasso**
 - 多次 bootstrap 重采样, 每次跑 Lasso;
 - 统计每个变量被选中的**频率**, 设阈值 (如 $\geq 90\%$ 就保留)。
 - 直观是“多数表决、取交集”, 能**消除许多假阳性**, 在更宽的 λ 区间里**更稳地选到真变量** (实验图里红色虚线优于黑线)。

- 代价：要多跑若干次（经验上 32 次常够用）。 λ 仍需选择，可用 CV，或借助“频率图”找稳定区间中段的 λ （启发式）。

4. 贝叶斯角度：预测时更偏向“后验均值”，不一定追求稀疏

- 拉普拉斯先验 + MAP = Lasso，本质给出**稀疏的众数**。
 - 但**后验均值/中位数或采样**通常**不稀疏**，却常有**更好的预测**（尤其用 spike-and-slab 先验时）；
 - 这再次提醒：**稀疏可解释** ≠ **预测最优**。如果目标是最小化平方预测误差，理论上用**后验均值**更合适（但计算更贵）。
-

关键要点

- Lasso = **选位置 + 收缩幅度** → 选得对但系数偏小；**去偏**可显著改进。
- CV 选 λ 为预测服务，不保证“选对真变量”；Lasso 的选择对数据扰动**不稳定**。
- Bolasso (bootstrap + Lasso) : 用出现频率筛选变量，**更稳、更接近真模型**，代价是多次训练。
- **贝叶斯替代**：若目标是预测，用**后验均值**（特别是 spike-and-slab 先验）通常优于 Lasso 的 MAP；若目标是**稀疏可解释**，可用 Lasso/bolasso/弹性网并做去偏。

一句话：Lasso 是选特征的好起点，但想“既选得稳，又估得准”，常需**去偏 + 稳定性选择（或弹性网）**；若更在意预测性能，**贝叶斯平均**往往更强。

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。

13.4 |1 regularization: algorithms

In this section, we give a brief review of some algorithms that can be used to solve 1 regularized estimation problems. We focus on the lasso case, where we have a quadratic loss. However, most of the algorithms can be extended to more general settings, such as logistic regression (see (Yuan et al. 2010) for a comprehensive review of 1 regularized logistic regression). Note that this area of machine learning is advancing very rapidly, so the methods below may not be state of the art by the time you read this chapter. (See (Schmidt et al. 2009; Yuan et al.

2010; Yang et al. 2010) for some recent surveys.)

13.4.1 Coordinate descent

Sometimes it is hard to optimize all the variables simultaneously, but it is easy to optimize them one by one. In particular, we can solve for the j 'th coefficient with all the others held fixed:

$$w^* j = \operatorname{argmin}_z f(w + z e_j) - f(w) \quad (13.65)$$

where e_j is the j 'th unit vector. We can either cycle through the coordinates in a deterministic fashion, or we can sample them at random, or we can choose to update the coordinate for which the gradient is steepest.

The coordinate descent method is particularly appealing if each one-dimensional optimization problem can be solved analytically. For example, the shooting algorithm (Fu 1998; Wu and Lange 2008) for lasso uses Equation 13.54 to compute the optimal value of w_j given all the other coefficients.

See (Yuan et al. 2010) for some extensions of this method to the logistic regression case. The resulting algorithm was the fastest method in their experimental comparison, which concerned document classification with large sparse feature vectors (representing bags of words). Other types of data (e.g., dense features and/or regression problems) might call for different algorithms.

13.4.2 LARS and other homotopy methods

The problem with coordinate descent is that it only updates one variable at a time, so can be slow to converge. Active set methods update many variables at a time. Unfortunately, they are more complicated, because of the need to identify which variables are constrained to be zero, and which are free to be updated.

Active set methods typically only add or remove a few variables at a time, so they can take a long if they are started far from the solution. But they are ideally suited for generating a set of solutions for different values of λ , starting with the empty set, i.e., for generating regularization path. These algorithms exploit the fact that one can quickly compute $w^*(\lambda_k)$ from $w^*(\lambda_{k-1})$ if $\lambda_k \approx \lambda_{k-1}$; this is known as warm starting. In fact, even if we only want the solution for a single value of λ , call it λ_* , it can sometimes be computationally more efficient to compute a set of solutions, from λ_{\max} down to λ_* , using warm-starting; this is called a continuation method or

homotopy method. This is often much faster than directly “cold-starting” at λ^* ; this is particularly true if λ^* is small. Perhaps the most well-known example of a homotopy method in machine learning is the LARS algorithm, which stands for “least angle regression and shrinkage” (Efron et al. 2004) (a similar algorithm was independently invented in (Osborne et al. 2000b,a)). This can compute $w^*(\lambda)$ for all possible values of λ in an efficient manner.

LARS works as follows. It starts with a large value of λ , such that only the variable that is most correlated with the response vector y is chosen. Then λ is decreased until a second variable is found which has the same correlation (in terms of magnitude) with the current residual as the first variable, where the residual at step k is defined as $r_k = y - X_{\cdot}F_k w_k$, where F_k is the current active set (c.f., Equation 13.50). Remarkably, one can solve for this new value of λ analytically, by using a geometric argument (hence the term “least angle”). This allows the algorithm to quickly “jump” to the next point on the regularization path where the active set changes. This repeats until all the variables are added.

It is necessary to allow variables to be removed from the active set if we want the sequence of solutions to correspond to the regularization path of lasso. If we disallow variable removal, we get a slightly different algorithm called LAR, which tends to be faster. In particular, LAR costs the same as a single ordinary least squares fit, namely $O(ND \min(N,D))$, which is $O(ND^2)$ if $N > D$, and $O(N^2D)$ if $D > N$. LAR is very similar to greedy forward selection, and a method known as least squares boosting (see Section 16.4.6).

There have been many attempts to extend the LARS algorithm to compute the full regularization path for l_1 regularized GLMs, such as logistic regression. In general, one cannot analytically solve for the critical values of λ . Instead, the standard approach is to start at λ_{\max} , and then slowly decrease λ , tracking the solution as we go; this is called a continuation method or homotopy method. These methods exploit the fact that we can quickly compute $w^*(\lambda_k)$ from $w^*(\lambda_{k-1})$ if $\lambda_k \approx \lambda_{k-1}$; this is known as warm starting. Even if we don’t want the full path, this method is often much faster than directly “cold-starting” at the desired value of λ (this is particularly true if

λ is small).

The method described in (Friedman et al. 2010) combines coordinate descent with this warmstarting strategy, and computes the full regularization path for any ℓ_1 regularized GLM. This has been implemented in the `glmnet` package, which is bundled with PMTK.

已思考 9s

下面先给出**完整直译**（不漏译），然后用**通俗解释**讲清楚，最后**要点小结**。

译文（完整）

13.4 ℓ_1 正则化：算法

本节对若干可用于求解 ℓ_1 正则估计问题的算法作一段简要回顾。我们聚焦在 `lasso`（损失是二次型）情形。不过，大多数算法都能推广到更一般的设置，例如**逻辑回归**（关于 ℓ_1 正则化逻辑回归的全面综述，见 Yaun et al. 2010）。需要注意的是，这一领域发展非常迅速，所以当你读到本章时，下面的方法也许已经不是最新的了（可参见 Schmidt et al. 2009；Yaun et al. 2010；Yang et al. 2010 的近期综述）。

13.4.1 坐标下降 (Coordinate descent)

有时要同时优化所有变量很难，但逐个优化就容易。具体地，我们可以在其他系数都固定时，解第 j 个系数：

$$w_j^* = \arg \min_z f(w + z e_j) - f(w) \quad (13.65)$$

其中 e_j 是第 j 个单位向量。我们可以按顺序循环各坐标，也可以随机抽样坐标，或者选择梯度最陡的坐标来更新。

如果每个一维子问题都能**解析地**求解，坐标下降法尤其吸引人。比如，`lasso` 的 **shooting 算法** (Fu 1998; Wu and Lange 2008) 就利用式 (13.54) 在给定其它系数时直接算出最优的 w_j 。

关于把该方法扩展到**逻辑回归**的做法，可见 Yaun et al. (2010)。在他们的实验对比（面向“词袋”稀疏大特征的文档分类）中，这一算法是**最快的**。对于其他类型数据（例如**稠密**特征和/或**回归**问题），可能需要不同的算法。

13.4.2 LARS 与其他同伦 (homotopy) 方法

坐标下降的一个问题是每次只更新一个变量，因此收敛可能慢。主动集（active set）方法一次更新多个变量。遗憾的是它们更复杂，因为要识别哪些变量被约束为0，哪些是可以更新的。

主动集方法通常一次只增加或删除少量变量，如果一开始距离解很远，可能需要很久。但它们非常适合按不同 λ 值生成一系列解，从空集开始，也就是用于生成正则化路径。这些算法利用了这样一个事实：若 $\lambda_k \approx \lambda_{k-1}$ ，我们能很快地由 $\hat{w}(\lambda_{k-1})$ 计算出 $\hat{w}(\lambda_k)$ ；这叫做热启动（warm starting）。事实上，即便我们只想要某个单独的 λ^* 的解，有时从 λ_{\max} 一路算到 λ^* 更省时；这叫延续法/同伦法（continuation/homotopy）。这通常比在 λ^* 处直接**冷启动（cold-start）**更快，特别是当 λ^* 很小时。

机器学习中最知名的同伦法之一是 LARS 算法（least angle regression and shrinkage，Efron et al. 2004；另见 Osborne et al. 2000b,a 的独立工作）。它能高效地为所有可能的 λ 计算 $\hat{w}(\lambda)$ 。

LARS 的工作方式如下：从很大的 λ 开始，此时只选择与响应向量 y 相关性最大的那个变量。然后逐渐减小 λ ，直到发现第二个变量使其与当前残差的（绝对）相关性与第一个变量相同；其中第 k 步的残差定义为

$$r_k = y - X_{:,F_k} w_{F_k},$$

F_k 是当前的活跃集（参见式 13.50）。一个重要事实是：利用几何论证（“最小角”由此得名），可以解析地解出这个新的 λ 值。这使算法能快速“跳到”正则化路径上活跃集发生变化的下一个点；重复此过程，直到所有变量都被加入。

如果我们希望这串解恰好对应 lasso 的正则化路径，就必须允许把变量从活跃集删除。若不允许删除，就得到稍有不同、通常更快的 LAR 算法。LAR 的计算代价与一次普通最小二乘相同： $O(ND \min(N, D))$ ，即 $N > D$ 时为 $O(ND^2)$ ，而 $D > N$ 时为 $O(N^2D)$ 。LAR 与贪心前向选择以及最小二乘 boosting（见 16.4.6 节）非常相近。

已有许多尝试将 LARS 扩展到 ℓ_1 正则化的广义线性模型（GLM）（如逻辑回归）的整条路径上。一般来说，临界的 λ 值无法解析求解。标准做法是从 λ_{\max} 开始缓慢减小 λ ，一路跟踪解——这就是延续/同伦法。它们同样依赖热启动：当 $\lambda_k \approx \lambda_{k-1}$ 时， $\hat{w}(\lambda_k)$ 可由 $\hat{w}(\lambda_{k-1})$ 快速得到。即使不需要整条路径，这种方法也常比在目标 λ 处直接冷启动更快（尤其当 λ 很小时）。

Friedman et al. (2010) 描述的方法把坐标下降与上述热启动结合，能为任意 ℓ_1 正则化的 GLM 计算完整正则化路径。该方法实现在 glmnet 包中，并已和 PMTK 一起提供。

通俗解释

- **坐标下降**: 一次只调一个系数，其它先不动；对 lasso，每次更新就用“**软阈值闭式解**”。数据是高维稀疏文本时，这招往往**最快最稳**。但因为“一次一步”，可能要多轮才能收敛。
 - **主动集 / 同伦 (延续)**：
 - 同一时间**只关注可能非零的那几个系数**（活跃集），一起更新；
 - 从 λ_{\max} 开始，逐步减小 λ ，每一步都用上一步的解当**热启动** → 很快；
 - LARS/LAR 是代表：在路径的“转折点”间**直线前进**，能几乎以“一次 OLS 的代价”得到**整条路径**；想看“变量何时进入/退出”，它最方便。
 - **推广到 GLM**: 逻辑回归等没有漂亮的解析转折点，用“ λ 递减 + 热启动”的延续法最实用；glmnet 就是“**坐标下降 + 热启动**”，工业界/学界都在用。
-

要点小结

- 目标：高效求解 ℓ_1 正则（尤其 lasso），**并走整条正则化路径**。
- **坐标下降**：逐坐标做**软阈值更新**；实现简单、扩展到 GLM 容易；在稀疏高维文本任务上**常最快**。
- **主动集/同伦**：一次处理多变量；从大 λ 向小 λ **热启动**，生成整条路径比冷启动快得多。
- **LARS/LAR**：经典同伦法；LARS 允许进出活跃集（对准 lasso 路径），LAR不删变量更快；与前向选择/LS boosting 类似。
- **GLM 场景**：临界 λ 不可解析，采用“递减 λ +热启动”；glmnet=坐标下降+路径跟踪，是通用高效实现。

这一小节主要在讲什么？我不太懂

已思考 13s

这小节的核心就是：**讲清楚用什么算法来高效求解 ℓ_1 正则（尤其是 Lasso）**，以及**为什么很多时候要沿着“正则化路径”一步步算**。

用大白话说

- **目标：**给出几种实用算法，帮你在“损失 + $\lambda\|w\|_1$ ”的模型里，既快又稳地找到系数 w （以及一串不同 λ 下的解）。
- **两大类思路：**
 1. **坐标下降** (Coordinate Descent)
 - 一次只更新一个系数，其余先不动；对 Lasso，每次更新都有**软阈值的闭式解**，实现很简单。
 - 可顺序循环、随机挑选，或挑“梯度最陡”的坐标更新。
 - 在高维稀疏数据（如词袋文本）上常常**最快**；也能拓展到逻辑回归（如 `glmnet` 的做法）。

2. 主动集 / 同伦 (延缓) 方法 (Active-set & Homotopy)

- 同时更新**一小撮可能非零**的变量（活跃集）。
- 从**大** λ （解全为 0）开始，逐步**减小** λ ，每一步用上一步的解做**热启动**（warm start），沿着**正则化路径**前进。
- LARS/LAR 是典型代表：在线性回归里几乎能**解析**地找到“路径的拐点”，因此**一站一站跳**过去，算整条路径的代价接近**一次 OLS**。LARS 允许变量进出活跃集（更贴合 Lasso 路径），LAR 不删变量更快。
- 到了 GLM（如逻辑回归），拐点不能解析，就用“ λ 递减 + 热启动”的续接法；`glmnet` = 坐标下降 + 热启动，能把**整条路径**算出来。

什么时候用哪种？

- **只要一个 λ 的解：** λ 较小、问题难时，先从 λ_{\max} 往下“热启动”通常更快；否则坐标下降直接求也行。
- **想看变量进入的顺序 / 需要一串模型：**用 LARS/LAR 或 `glmnet` 路径更合适。
- **数据超高维且稀疏（如文本）：**优先**坐标下降**（快且内存友好）。
- **特征强相关、想稳定：**可考虑路径法并配合弹性网 $(\ell_1 + \ell_2)$ 。

关键词速记

- **正则化路径：** λ 从大到小，系数如何“从 0 被激活”的整条轨迹。
- **热启动：**用 $\hat{w}(\lambda_{k-1})$ 当 $\hat{w}(\lambda_k)$ 的初值 $(\lambda_k \approx \lambda_{k-1})$ 。
- **LARS/LAR：**用几何法快速跳到下一个“拐点”；代价≈一次 OLS。
- **软阈值：**坐标下降更新 Lasso 系数的闭式公式，会“变 0 + 收缩”。

一句话总结

这小节告诉你：解 Lasso 最常用的是“坐标下降（软阈值）”和“同伦/路径法（LARS、glmnet）”；前者简单高效、易扩展，后者善于沿整条正则化路径快速得到一系列解，并通过热启动显著加速。

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。

13.4.3 Proximal and gradient projection methods

In this section, we consider some methods that are suitable for very large scale problems, where homotopy methods made be too slow. These methods will also be easy to extend to other kinds of regularizers, beyond ℓ_1 , as we will see later. Our presentation in this section is based on (Vandenberghe 2011; Yang et al. 2010).

Consider a convex objective of the form

$$f(\theta) = L(\theta) + R(\theta) \quad (13.66)$$

where $L(\theta)$ (representing the loss) is convex and differentiable, and $R(\theta)$ (representing the regularizer) is convex but not necessarily differentiable. For example, $L(\theta) = \text{RSS}(\theta)$ and $R(\theta) = \lambda \|\theta\|_1$ corresponds to the BPDN problem. As another example, the lasso problem can be formulated as follows: $L(\theta) = \text{RSS}(\theta)$ and $R(\theta) = IC(\theta)$, where $C = \{\theta : \|\theta\|_1 \leq B\}$, and $IC(\theta)$ is the indicator function of a convex set C , defined as

$$IC(\theta) = 0 \quad \theta \in C \quad +\infty \quad \text{otherwise} \quad (13.67)$$

In some cases, it is easy to optimize functions of the form in Equation 13.66. For example, suppose $L(\theta) = \text{RSS}(\theta)$, and the design matrix is simply $X = I$. Then the obective becomes $f(\theta) = R(\theta) + \frac{1}{2} \|\theta - y\|_2^2$. The minimizer of this is given by $\text{proxR}(y)$, which is the proximal operator for the convex function R , defined by

$$\text{proxR}(y) = \operatorname{argmin}_z R(z) + \frac{1}{2} \|z - y\|_2^2 \quad (13.68)$$

Intuitively, we are returning a point that minimizes R but which is also close (proximal) to y . In general, we will use this operator inside an iterative optimizer, in which case we want to stay close to the previous iterate. In this case, we use

$$\text{proxR}(\theta) = \underset{z}{\operatorname{argmin}} z^T R(z) + \frac{1}{2} \|z - \theta\|^2 \quad (13.69)$$

The key issues are: how do we efficiently compute the proximal operator for different regularizers R , and how do we extend this technique to more general loss functions L ? We discuss these issues below.

13.4.3.1 Proximal operators

If $R(\theta) = \lambda \|\theta\|_1$, the proximal operator is given by componentwise soft-thresholding:

$$\text{proxR}(\theta) = \text{soft}(\theta, \lambda) \quad (13.70)$$

as we showed in Section 13.3.2. If $R(\theta) = \lambda \|\theta\|_0$, the proximal operator is given by componentwise hard-thresholding:

$$\text{proxR}(\theta) = \text{hard}(\theta, \sqrt{2\lambda}) \quad (13.71)$$

where $\text{hard}(u, a) = u \mathbb{I}(|u| > a)$.

If $R(\theta) = \text{IC}(\theta)$, the proximal operator is given by the projection onto the set C :

$$\text{proxR}(\theta) = \underset{z \in C}{\operatorname{argmin}} \|z - \theta\|^2 \quad (13.72)$$

For some convex sets, it is easy to compute the projection operator. For example, to project onto the rectangular set defined by the box constraints $C = \{\theta : j \leq \theta_j \leq u_j\}$ we can use

$$\text{projC}(\theta)_j = \begin{cases} l & \text{if } \theta_j \leq l \\ \theta_j & \text{if } l \leq \theta_j \leq u_j \\ u_j & \text{if } \theta_j \geq u_j \end{cases} \quad (13.73)$$

To project onto the Euclidean ball $C = \{\theta : \|\theta\|^2 \leq 1\}$ we can use

$$\text{projC}(\theta) = \theta / \|\theta\| \quad \text{if } \|\theta\| > 1 \\ \theta \quad \text{if } \|\theta\| \leq 1 \quad (13.74)$$

To project onto the 1-norm ball $C = \{\theta : \|\theta\|_1 \leq 1\}$ we can use

$$\text{projC}(\theta) = \text{soft}(\theta, \lambda) \quad (13.75)$$

where $\lambda = 0$ if $\|\theta\|_1 \leq 1$, and otherwise λ is the solution to the equation

$$\sum_{j=1}^D \max(|\theta_j| - \lambda, 0) = 1 \quad (13.76)$$

We can implement the whole procedure in $O(D)$ time, as explained in (Duchi et al. 2008). We will see an application of these different projection methods in Section 13.5.1.2.

已思考 14s

下面先把原文完整翻译（不漏译），再用通俗解释讲清楚，最后给要点小结。

译文

13.4.3 近端 (proximal) 与梯度投影方法

本节讨论一些适合**超大规模问题**的方法，在这类问题上，同伦/路径方法可能太慢。这些方法也很容易推广到**I1 之外的其他正则器**（稍后可见）。本节表述主要基于 (Vandenberghe, 2011; Yang et al., 2010)。

考虑如下凸目标：

$$f(\theta) = L(\theta) + R(\theta) \quad (13.66)$$

其中 $L(\theta)$ (损失) 是**凸且可微的**， $R(\theta)$ (正则项) 是**凸但不一定可微的**。比如， $L(\theta) = \text{RSS}(\theta)$ 、 $R(\theta) = \lambda \|\theta\|_1$ 就对应 **BPDN** 问题。再如，**lasso** 也可写成： $L(\theta) = \text{RSS}(\theta)$ 、 $R(\theta) = I_C(\theta)$ ，其中

$C = \{\theta : \|\theta\|_1 \leq B\}$ ，而 $I_C(\theta)$ 是凸集 C 的**指示函数**：

$$I_C(\theta) = \begin{cases} 0, & \theta \in C \\ +\infty, & \text{否则} \end{cases} \quad (13.67)$$

在某些情形下，式 (13.66) 这样的函数容易优化。比如令 $L(\theta) = \text{RSS}(\theta)$ ，且设计矩阵 $X = I$ 。此时目标变为

$$f(\theta) = R(\theta) + \frac{1}{2} \|\theta - y\|_2^2.$$

它的极小点由**近端算子 (proximal operator)**给出：

$$\text{prox}_R(y) = \arg \min_z R(z) + \frac{1}{2} \|z - y\|_2^2 \quad (13.68)$$

直观地说，我们返回一个既让 R 小、又离 y 近 (proximal, 邻近) 的点。更一般地，在迭代优化中我们希望靠近**上一次迭代点**，于是用

$$\text{prox}_R(\theta_k) = \arg \min_z R(z) + \frac{1}{2} \|z - \theta_k\|_2^2. \quad (13.69)$$

关键问题是：**怎样高效计算不同正则器 R 的近端算子？** 以及**怎样把该技巧推广到更一般的损失 L ？** 下面讨论这两点。

13.4.3.1 近端算子

- 若 $R(\theta) = \lambda \|\theta\|_1$ ，近端算子是**逐元素软阈值**：

$$\text{prox}_R(\theta) = \text{soft}(\theta, \lambda) \quad (13.70)$$

(见 13.3.2 节)。

- 若 $R(\theta) = \lambda \|\theta\|_0$ ，近端算子是**逐元素硬阈值**：

$$\text{prox}_R(\theta) = \text{hard}(\theta, \sqrt{2\lambda}), \quad (13.71)$$

其中 $\text{hard}(u, a) = u \mathbf{1}(|u| > a)$ 。

- 若 $R(\theta) = I_C(\theta)$, 近端算子就是把点投影到集合 C 上:

$$\text{prox}_R(\theta) = \arg \min_{z \in C} \|z - \theta\|_2^2 = \text{proj}_C(\theta). \quad (13.72)$$

对某些凸集, 投影易于计算。例如:

1. **盒约束** $C = \{\theta : \ell_j \leq \theta_j \leq u_j\}$:

$$\text{proj}_C(\theta)_j = \begin{cases} \ell_j, & \theta_j \leq \ell_j \\ \theta_j, & \ell_j \leq \theta_j \leq u_j \\ u_j, & \theta_j \geq u_j \end{cases} \quad (13.73)$$

2. **欧氏球** $C = \{\theta : \|\theta\|_2 \leq 1\}$:

$$\text{proj}_C(\theta) = \begin{cases} \frac{\theta}{\|\theta\|_2}, & \|\theta\|_2 > 1 \\ \theta, & \|\theta\|_2 \leq 1 \end{cases} \quad (13.74)$$

3. ℓ_1 球 $C = \{\theta : \|\theta\|_1 \leq 1\}$:

$$\text{proj}_C(\theta) = \text{soft}(\theta, \lambda), \quad (13.75)$$

其中若 $\|\theta\|_1 \leq 1$ 则 $\lambda = 0$; 否则 λ 取满足

$$\sum_{j=1}^D \max(|\theta_j| - \lambda, 0) = 1 \quad (13.76)$$

的值。整个过程可在 $O(D)$ 时间内实现 (见 Duchi et al., 2008)。这些不同投影方法的一个应用将在 13.5.1.2 节出现。

通俗解释

这段在干嘛?

把你的目标写成“光滑损失 L + 可能不光滑的正则 R ”。直接最小化可能困难, 于是采用“靠近上一步 + 考虑 R 的惩罚”的折中:

找 z 使 $R(z)$ 小且 z 离上一轮 θ_k 近。

这个折中的求解器就叫**近端算子** prox_R 。它像一个**带规则的‘去噪/收缩’操作**:

- 对 ℓ_1 , 它就是**软阈值** (小的元素被推到 0, 剩下的被缩小) ;
- 对 ℓ_0 , 是**硬阈值** (小于阈值的直接置 0, 其余不变) ;

- 对**指示函数**, 就是**把点投影到可行集合** (盒子、球等)。

在大问题里, 我们会**迭代**: 先对光滑的 L 走一步 (通常是梯度步), 再用 prox_R 做一次“规则化的靠近”。(这就是常说的**近端梯度/梯度投影思想**; 后续可以扩展到很多正则器。)

为什么实用?

- 只要能高效算 prox_R , 整个大问题就能被分解成**简单的逐元素门槛/投影操作**, **非常快、可并行**;
- 不仅支持 ℓ_1 , 也能支持** ℓ_0 、稀疏组、盒/球/多面体约束**等很多 R;
- 对超大规模 (特征多、样本多) 的场景, 比“整批求解正则化路径”的同伦法更合适。

几个投影例子怎么理解?

- 盒约束**: 对每个坐标, 把越界的值截到下/上界;
- ℓ_2 **球**: 如果向量太长, 就把它**缩放**到单位长度;
- ℓ_1 **球**: 做一次**软阈值**, 阈值 λ 通过“让新向量的 ℓ_1 正好等于 1”来确定, 可在线性时间内完成。

要点小结

- 把目标写成 $f = L + R$: L 光滑、 R 可能不光滑。
- 近端算子** prox_R : 求 $R(z) + \frac{1}{2}\|z - \theta\|^2$ 的极小点; 在算法里反复使用, 起到**收缩/投影**的作用。
- 典型近端:
 - $R = \lambda\|\theta\|_1 \rightarrow$ **软阈值**;
 - $R = \lambda\|\theta\|_0 \rightarrow$ **硬阈值** (阈值 $\sqrt{2\lambda}$) ;
 - $R = I_C \rightarrow$ **投影到集合** C (盒、 ℓ_2 球、 ℓ_1 球等)。
- 这些操作都很快 (有的 $O(D)$), 适合**超大规模**问题, 也为后续的**近端梯度/梯度投影**算法铺路 (稍后扩展到更多正则器与模型)。

这一小节主要在讲什么? 我不太懂, 和前面的坐标下降、主动集方法是否做的同一件事

已思考 23s

这小节在讲：用“**近端/梯度投影** (proximal / projected-gradient) ”**这一路算法来解 ℓ_1 正则化问题**，它把目标写成

$$f(\theta) = L(\theta) + R(\theta)$$

(L 是光滑可微的损失，比如 RSS； R 是凸但可能不光滑的正则，比如 $\lambda\|\theta\|_1$ 或 “ $\|\theta\|_1 \leq B$ ”这类约束），然后每一步做两件事：

(1) 沿着 L 走一步梯度；(2) 用 R 的“**近端算子**”把结果做一次“收缩/投影”。

其中“**近端算子**” prox_R 就是“在尽量减小 R 的同时又离当前点不远”的解；对常见 R 有现成的、简单又快的计算——

- $R = \lambda\|\theta\|_1 \rightarrow$ **软阈值**（很多分量被推到 0）；
- $R = \lambda\|\theta\|_0 \rightarrow$ **硬阈值**；
- $R = I_C$ (集合指示函数) \rightarrow **投影到集合 C** (盒约束、 ℓ_2 球、 ℓ_1 球等)。

它们很适合**超大规模**问题，也容易推广到 ℓ_1 之外的各种正则与约束。

和“坐标下降、主动集/同伦法”是同一件事吗？

目标一样 (都是为了解 ℓ_1 正则/稀疏问题)，**但做法与适用场景不一样**：

- **坐标下降 (13.4.1)**
 - 一次只更新一个坐标，其它先不动；对 Lasso 每次更新都有**软阈值的闭式解**。
 - 优点：实现简单、内存友好，对**高维稀疏** (如文本) 非常快。
 - 可能需要多轮“扫一遍又一遍”才能收敛；强相关时会慢些。
 - 可视为“**近端思想在单坐标上的特例**” (逐坐标做 prox)。
- **主动集/同伦 (LARS/LAR, 13.4.2)**
 - 维护一小撮“**可能非零**”的变量一起更新；
 - 从大 λ 到小 λ 沿**正则化路径**往前走，靠**热启动**快速前进；LARS 还能几乎解析地跳到“拐点”。
 - **最适合想要“一整条路径”** (看变量何时进入/退出) 的时候；但在超大规模或非二次损失时不一定最省时。
- **近端/梯度投影 (本节 13.4.3)**
 - 一次更新整个向量：先做一次**梯度步** (只看 L)，再做一次**近端步** (把正则 R 落实为软阈值/投影)。
 - **可并行、可扩展**：换个 R 只需换 prox；加盒/球/ ℓ_1 球等约束也只是换个投影。

- 更适合超大规模、只关心单个 λ 的情况；要路径也能“从 λ_{\max} 往下热启动”，但不像 LARS 有那种解析“拐点”。

一句话对比：

- 坐标下降 = “逐坐标近端”；
 - 主动集/同伦 = “沿路径跳拐点（热启动）”；
 - 近端/梯度投影 = “整向量梯度 + 整向量近端（收缩/投影）”，**大规模场景更通用**。
-

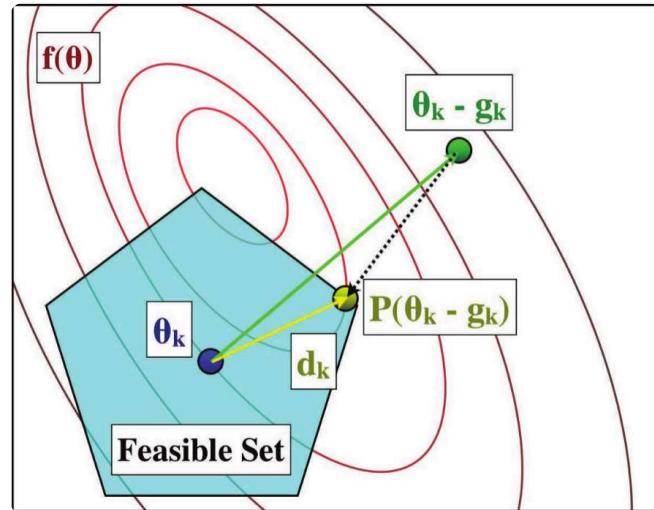
直观小示例（不写公式）

每一轮：

- 用当前 θ_k 计算损失 L 的梯度，往“更小的方向”走半步得到临时点 u 。
 - 对 u 做一次 prox：
 - 若是 ℓ_1 惩罚，就对 u 逐元素软阈值（小的推到 0）；
 - 若是“ ℓ_1 球内”的约束，就把 u 投影回到这只“钻石形”球里。
 得到 θ_{k+1} 。重复即可。
-

要点小结

- 本节讲 **近端/梯度投影框架**：把目标拆成 L （光滑）+ R （可能不光滑），每步“**梯度步 + 近端步**”。
- 常见 prox： $\ell_1 \rightarrow$ 软阈值； $\ell_0 \rightarrow$ 硬阈值；指示函数→投影（盒、 ℓ_2/ℓ_1 球等）。
- 与前两节关系**：目标相同、思路不同；坐标下降是“逐坐标 prox”，主动集/同伦是“沿路径热启动”，而近端法是“整向量 prox”，**更适合超大规模、可扩展到多种正则与约束**。
- 实战建议：
 - 只要能高效算 prox、数据很大 → **近端/投影法优先**；
 - 要整条路径或看变量“进场顺序” → **LARS/路径法**；
 - 超高维稀疏文本 → **坐标下降常最强**。



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 13.11 Illustration of projected gradient descent. The step along the negative gradient, to $\theta_k - g_k$, takes us outside the feasible set. If we project that point onto the closest point in the set we get $\theta_{k+1} = \text{proj}_{\Theta}(\theta_k - g_k)$. We can then derive the implicit update direction using $d_k = \theta_{k+1} - \theta_k$. Used with kind permission of Mark Schmidt.

13.4.3.2 Proximal gradient method We now discuss how to use the proximal operator inside of a gradient descent routine. The basic idea is to minimize a simple quadratic approximation to the loss function, centered on the θ_k :

$$\theta_{k+1} = \underset{z}{\operatorname{argmin}} z^T R(z) + L(\theta_k) + g^T k (z - \theta_k) + \frac{1}{2} t_k \|z - \theta_k\|^2 \quad (13.77)$$

where $g_k = \nabla L(\theta_k)$ is the gradient of the loss, t_k is a constant discussed below, and the last term arises from a simple approximation to the Hessian of the loss of the form $\nabla^2 L(\theta_k) \approx t_k I$. Dropping terms that are independent of z , and multiplying by t_k , we can rewrite the above expression in terms of a proximal operator as follows:

$$\theta_{k+1} = \underset{z}{\operatorname{argmin}} z^T t_k R(z) + \frac{1}{2} t_k \|z - u_k\|^2 = \text{prox}_{t_k R}(u_k) \quad (13.78)$$

$$u_k = \theta_k - t_k g_k \quad (13.79)$$

$$g_k = \nabla L(\theta_k) \quad (13.80)$$

If $R(\theta)=0$, this is equivalent to gradient descent. If $R(\theta)=\|C(\theta)\|_1$, the method is equivalent to projected gradient descent, sketched in Figure 13.11. If $R(\theta)=\lambda\|\theta\|_1$, the

method is known as iterative soft thresholding. There are several ways to pick τ_k , or equivalently, $\alpha_k = 1/\tau_k$. Given that $\alpha_k I$ is an approximation to the Hessian $\nabla^2 L$, we require that

$$\alpha_k(\theta_k - \theta_{k-1}) \approx g_k - g_{k-1} \quad (13.81)$$

in the least squares sense. Hence

$$\alpha_k = \operatorname{argmin}_\alpha \|\alpha(\theta_k - \theta_{k-1}) - (g_k - g_{k-1})\|_2^2 = (\theta_k - \theta_{k-1})^T(g_k - g_{k-1}) / (\theta_k - \theta_{k-1})^T(\theta_k - \theta_{k-1}) \quad (13.82)$$

This is known as the Barzilai-Borwein (BB) or spectral stepsize (Barzilai and Borwein 1988; Fletcher 2005; Raydan 1997). This stepsize can be used with any gradient method, whether proximal or not. It does not lead to monotonic decrease of the objective, but it is much faster than standard line search techniques. (To ensure convergence, we require that the objective decrease "on average", where the average is computed over a sliding window of size $M+1$.) When we combine the BB stepsize with the iterative soft thresholding technique (for $R(\theta) = \lambda\|\theta\|_1$), plus a continuation method that gradually reduces λ , we get a fast method for the BPDN problem known as the SpaRSA algorithm, which stands for "sparse reconstruction by separable approximation" (Wright et al. 2009). However, we will call it the iterative shrinkage and thresholding algorithm. See Algorithm 12 for some pseudocode, and SpaRSA for some Matlab code.

13.4.3.3 Nesterov's method

A faster version of proximal gradient descent can be obtained by expanding the quadratic approximation around a point other than the most recent parameter value. In particular, consider performing updates of the form

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

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

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

This is known as Nesterov's method (Nesterov 2004; Tseng 2008). As before, there are a variety of ways of setting τ_k ; typically one uses line search.

When this method is combined with the iterative soft thresholding technique (for $R(\theta) = \lambda\|\theta\|_1$), plus a continuation method that gradually reduces λ , we get a fast method for the BPDN problem known as the fast iterative shrinkage thresholding algorithm or FISTA (Beck and Teboulle 2009).

下面先把原文逐句完整翻译（不漏译），再用通俗解释说明，最后要点小结。

图注翻译

图 13.11 投影梯度下降 (projected gradient descent) 的示意图。沿着负梯度方向迈一步，到达 $\theta_k - g_k$ 这点，会把我们带到**可行集合**之外。如果把该点**投影回**集合中的最近点，我们得到

$\theta_{k+1} = \text{proj}_{\Theta}(\theta_k - g_k)$ 。随后可用 $d_k = \theta_{k+1} - \theta_k$ 得到隐式的更新方向。图经 Mark Schmidt 许可使用。

13.4.3.2 近端梯度法 (Proximal gradient method)

我们现在讨论如何在梯度下降的框架里使用**近端算子**。基本思想是：围绕 θ_k 对损失函数做一个**简单二次近似并最小化**它：

$$\theta_{k+1} = \arg \min_z R(z) + L(\theta_k) + g_k^\top (z - \theta_k) + \frac{1}{2t_k} \|z - \theta_k\|_2^2 \quad (13.77)$$

其中 $g_k = \nabla L(\theta_k)$ 是损失的梯度， t_k 是一个常数（下文讨论），最后一项来自对 Hessian 的近似

$\nabla^2 L(\theta_k) \approx \frac{1}{t_k} I$ 。把与 z 无关的项去掉并整体乘以 t_k ，上述式子可改写成一个**近端形式**：

$$\theta_{k+1} = \arg \min_z t_k R(z) + \frac{1}{2} \|z - u_k\|_2^2 = \text{prox}_{t_k R}(u_k) \quad (13.78)$$

$$u_k = \theta_k - t_k g_k \quad (13.79)$$

$$g_k = \nabla L(\theta_k) \quad (13.80)$$

- 若 $R(\theta) = 0$ ，这就退化为普通的**梯度下降**。
- 若 $R(\theta) = I_C(\theta)$ （集合 C 的指示函数），该方法等价于**投影梯度下降**（见图 13.11）。
- 若 $R(\theta) = \lambda \|\theta\|_1$ ，该方法称为**迭代软阈值** (iterative soft thresholding)。

有多种选择 t_k （或等价的 $\alpha_k = 1/t_k$ ）的方法。既然 $\alpha_k I$ 是 Hessian $\nabla^2 L$ 的近似，我们要求在最小二乘意义下满足

$$\alpha_k (\theta_k - \theta_{k-1}) \approx g_k - g_{k-1} \quad (13.81)$$

因此