第五章　监督学习方法

线性回归和广义线性模型在预测不同类型的被解释变量这一问题上通常可以获得不错的效果，然而仍有其局限性。比如，这些模型要么都存在比较严格的函数形式假设，或者存在维数的诅咒。在“大数据”越来越流行的今天，高维数据、非结构化数据越来越普遍，传统的预测和拟合方法在这些问题上往往难以胜任，而一些**机器学习**（**machine learning**）方法在处理这些问题上具有较强的优势。特别的，**监督学习**（**supervised learning**）正是机器学习中用于预测和拟合的一系列方法。本章主要介绍一些常见的监督学习方法，包括收缩估计量（岭回归、Lasso 回归）以及基于决策树的方法，并简要介绍神经网络。

5.1　收缩估计量

在第一章的讨论中，我们论证了普通最小二乘是对于条件期望的最优线性逼近，而条件期望作为最优预测，因而 OLS 方法可以用于预测。然而如果我们关注预测，那么在均方误差的标准下，OLS 可能并不是最优的。为了解决这一问题，我们这里引入**收缩估计量**（**shringkage estimator**）。收缩估计量的引入一方面帮助建立更好的预测模型，而另一方面，也为变量选择提供了强有力的工具。为了理解收缩估计量的原理，我们可以从最初的 James-Stein 估计量开始。

5.1.1　James-Stein 估计量

考虑如下问题。假设 $\theta \in \mathbb{R}^K$ 为未知参数，观察到的样本 $x \in \mathbb{R}^K$，且 $x \sim N(\theta, I)$，即我们有 K 个参数，同时观察到了 K 个样本，每个样本服从 $x_k \sim N(\theta_k, 1)$，且样本之间相互独立。

为了预测 θ，我们可以使用最小二乘法，即最小化

$$\hat{\theta}^{LS} = \min_{\vartheta} \sum_{k=1}^{K} (x_k - \theta_k)^2$$

从而得到：$\hat{\theta}_k^{LS} = x_k$，由于 $\mathbb{E}\left(\hat{\theta}_k^{LS}\right) = \theta_k$，从而我们得到了无偏估计（当然，不是一致估计）。

然而，无偏的不一定是最好的。如果我们计算其均方误差，得到

$$MSE^{LS} = \mathbb{E}\left[\sum_{k=1}^{K} (x_k - \theta_k)^2\right] = \mathbb{E}\left[(x-\theta)'(x-\theta)\right] = K$$

然而 Stein（1956）指出，当 $K \geq 3$ 时，我们可以找到使得 MSE 更小的估计量（从而最小二乘估计是 inadmissible 的）。进一步，James 和 Stein（1961）提出了如下估计量：

$$\hat{\theta}^{JS} = \left(1 - \frac{K-2}{\|x\|_2^2}\right) x \tag{5.1}$$

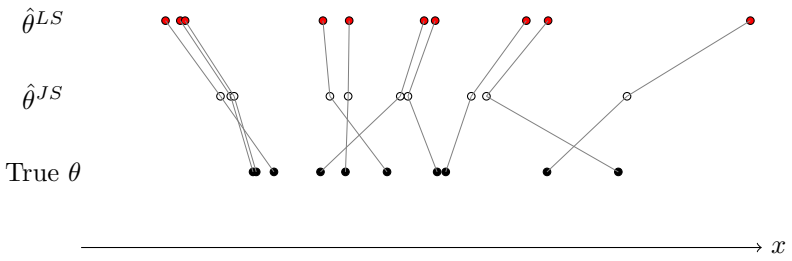


图 5.1: James-Stein 估计量

其中 $\|x\|_2 = \sqrt{x'x}$ 为 L^2 范数。可以证明以上估计量的均方误差为

$$MSE^{JS} = K - \mathbb{E}\left[\frac{(K-2)^2}{K-2+2M}\right] < K = MSE^{LS}$$

其中 $M \sim P\left(\frac{\|\theta\|_2^2}{2}\right)$。

如果我们观察式 (5.1)，如果 $K-2 < \|x\|_2^2$，那么 $\hat{\theta}^{JS}$ 的实质是将 x 向 0 进行了收缩 (shrink)，如图 (5.1) 所示。实际上，不仅仅可以向 0 进行收缩，取任意的向量 ϑ，估计量：

$$\hat{\theta}^{JS} = \left(1 - \frac{K-2}{\|x-\vartheta\|_2^2}\right)(x-\vartheta) + \vartheta = x + \frac{K-2}{\|x-\vartheta\|_2^2}(\vartheta - x)$$

实现了向向量 ϑ 的收缩。由于 ϑ 的任意性，当 x 与 ϑ 距离比较远时，$\|x-\vartheta\|_2^2$ 比较大，因而仅仅是轻微的向 ϑ 移动。实际上不管 ϑ 取任何值，以上定义的 $\hat{\theta}^{JS}$ 的均方误差总比 $\hat{\theta}^{LS}$ 小。因而 James-Stein 估计量通过收缩，放弃了无偏性，同时降低了方差，从而降低了总体的均方误差。

以上 James-Stein 估计量可以扩展到回归的情形。在线性回归中，如果我们令 OLS 估计量向 0 收缩，由于对 Y 的预测为 $X\hat{\beta}$，于是我们就有了 James-Stein 估计量：

$$\hat{\beta}^{JS} = \hat{\beta}^{OLS}\left[1 - \frac{(K-2)\hat{\sigma}^2}{\hat{\beta}X'X\hat{\beta}}\right]$$

同样我们也可以构造像任意 $\tilde{\beta}$ 收缩的 James-Stein 估计量。与之前类似，使用该估计量可以降低对 y 进行估计的均方误差。

5.1.2 岭回归

虽然我们可以将 OLS 估计量改进为收缩估计量，然而这仍然无法解决 OLS 的一些现实问题。比如，在高维（high-dimensional）问题中，OLS 通常表现较差，甚至是不可行的（比如 $K > N$，即解释变量个数大于样本量的情况）。为此，我们可以考虑直接对参数向量 β 进行收缩。

其中，**岭回归（ridge regression）**使用 L^2 范数实现对 β 的收缩。岭回归在最小二乘法的目标函数基础上，额外附加了 β 的 L^2 范数的限制，即限制 $\|\beta\|_2^2 \leq t$：

$$\hat{\beta}^r(t) = \arg\min_{\beta} \sum_{i=1}^{N} (y_i - x_i'\beta)^2 \tag{5.2}$$
$$s.t. \|\beta\|_2^2 \leq t$$

从而实现了 β 向 0 的收缩——在绝大多数情况下，向 0 收缩是非常自然且方便的选择，以上优化

问题如图5.2(a) 所示。解以上问题即最大化

$$\mathscr{L}(\beta,\lambda)=\sum_{i=1}^{N}\left(y_i-x_i'\beta\right)^2+\lambda\left(\|\beta\|_2^2-t\right)$$

以上优化问题的解取决于 t 的取值，一旦 t 确定了，根据库恩塔克定理，$\lambda(t)$ 也就确定了。或者，反过来，我们可以给定 λ，而不固定 t 的取值，那么以上最大化问题变为了

$$\hat{\beta}^r(\lambda)=\arg\min_{\beta}\sum_{i=1}^{N}\left(y_i-x_i'\beta\right)^2+\lambda\|\beta\|_2^2=\arg\min_{\beta}\left(Y-X\beta\right)'\left(Y-X\beta\right)+\lambda\beta'\beta$$

以上优化问题对 β 的大小进行了限制，因而以上问题通常也被成为**惩罚最小二乘**（**penalized least squares**）。一般的，我们将限制模型“大小”或者表现能力的方法统称为**正则化**（**regularization**），而惩罚最小二乘中的惩罚项正是通过收缩削弱模型的表现能力，或者让模型变得更“小”，因而我们也将惩罚项 $\lambda\|\beta\|_2^2$ 成为“正则化项”。

注意 $\lambda(t)$ 是一个 t 的减函数：当 $t\to\infty$ 时，$\lambda\to 0$，此时施加在 OLS 目标函数上的约束不再有约束力，从而 $\hat{\beta}^r$ 会逐渐收敛到 OLS 估计量 $\hat{\beta}$；而当 $\lambda\to\infty$，$t\to 0$，此时约束集逐渐向 0 向量靠拢，从而 $\hat{\beta}^r$ 会逐渐趋向于 0。

解以上问题，得到一阶条件：

$$\left(X'X+\lambda I\right)\beta=X'Y$$

从而得到

$$\hat{\beta}^r(\lambda)=\left(X'X+\lambda I\right)^{-1}X'Y$$

实际上，岭回归还可以看做是在特定先验下的贝叶斯线性回归（习题5.4）。注意到以上岭回归的计算式中，即使 X 不是列满秩的，矩阵 $(X'X+\lambda I)$ 也是可逆的，也可以获得估计值。特别是在 $K>N$ 的情况下，OLS 无法获得估计，而岭回归仍然可以获得估计值。

考虑 X 中没有常数项，且分量之间相互正交的情况。方便起见，我们记：

$$X=\left[\begin{array}{cccc}x^1 & x^2 & \cdots & x^K\end{array}\right]$$

其中 x^k 为 X 矩阵的第 k 列，不失一般性我们假设 $\sum_{i=1}^{N}x_i^k=0$，即我们考虑已经中心化的 K 个解释变量。如果 X 的分量之间正交，即 $\left(x^k\right)'x^l=0$，此时可以证明（习题5.5）岭回归的系数满足：

$$\hat{\beta}_k^r=\frac{\left(x^k\right)'x^k}{\left(x^k\right)'x^k+\lambda}\hat{\beta}_k$$

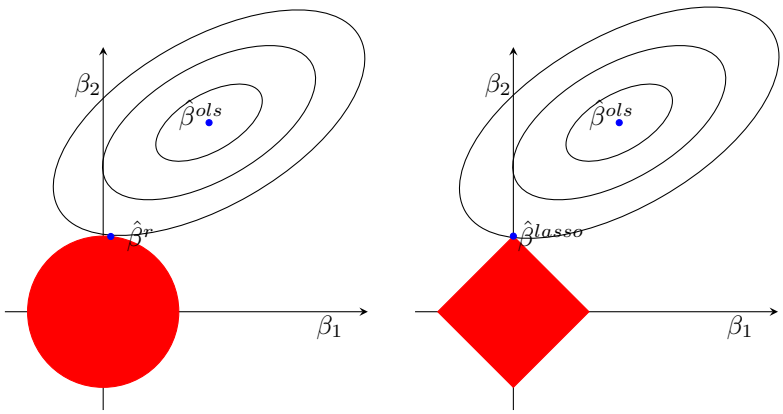
其中 $\left(x^k\right)'x^k=\sum_{i=1}^{N}\left(x_i^k\right)^2$ 为第 k 个变量的平方和。在这种情况下，岭回归即在 OLS 的基础上乘以一个收缩系数。而注意到 $\left(x^k\right)'x^k$ 的大小取决于 x^k 的方差以及样本量，从而对于方差不同的变量，其收缩幅度是不一样的，然而 x^k 的方差大并不代表对 y 的预测能力强，从而这样的收缩对于方差小的变量是不公平的。为此，我们有必要在进行岭回归（包括接下来的 Lasso 回归等方法）时，对每个解释变量进行标准化，即使用：

$$\mathring{x}_i^k=\frac{x_i^k-\overline{x^k}}{s_{x^k}}$$

其中 $\overline{x^k}$ 为 x^k 的均值，s_{x^k} 为 x^k 的标准差。

在 Stata 官方命令中，lasso 命令会自动帮助我们进行标准化，从而我们无需手动进行这一操作。而在一些软件和包[1]中，标准化这一步骤默认通过“运行中（on the fly）”标准化的方式进行，即

[1]如 Stata 中非官方的 Lasso 命令：lasso2、cvlasso 和 rlasso 等。



(a) 岭回归 (b) Lasso 回归

图 5.2: 岭回归与 Lasso 的目标函数及约束

解最小化问题：

$$\arg\min_{\beta} (Y - X\beta)' (Y - X\beta) + \lambda \|\Psi\beta\|_2^2$$

其中 $\Psi = \text{diag}[\psi_k]$ 为一个 $K \times K$ 的对角矩阵，其对角线元素为“惩罚载荷（penalty loadings）”，默认使用 $\psi_k = \sqrt{(x^k)' x^k / N}$，即 x^k 的标准差估计作为惩罚载荷，如此方差较大的变量会受到更多的惩罚，而方差较小的变量会受到较小的惩罚。可以证明以上做法与事前进行标准化的做法是等价的（习题5.6）。事实上以上目标函数即

$$\arg\min_{\beta} (Y - X\beta)' (Y - X\beta) + \lambda \sum_{k=1}^{K} \psi_k \beta_k^2$$

在一些软件包中，ψ_k 可以人为设定每个系数被惩罚的强度[2]。

与 OLS 相比，岭回归通过将 $\hat{\beta}$ 向 0 收缩，减小了 $\hat{\beta}$ 的变异性或者方差，而代价是岭回归所得到的系数并不是无偏的。需要注意的是，尽管岭回归减少了 $\hat{\beta}$ 的变异性，也不能保证 $\hat{Y}^r(\lambda) = X\hat{\beta}^r(\lambda)$ 比 OLS 的预测值更加精确。实际上，预测的均方误差取决于 $\hat{\beta}$ 的方差以及偏差两个方面，一方面岭回归降低了 $\hat{\beta}$ 的方差，而另一方面岭回归增加了偏差（bias），实际上两个效应现实中可能会相互抵消。

5.1.3 Lasso 回归

在岭回归的目标函数中，为了实现收缩，使用了 L^2 范数，即 $\|\beta\|_2^2$，而 Tibshirani（1996）提出可以使用 L^1 范数

$$\|\beta\|_1 = \sum_{k=1}^{K} |\beta_k|$$

对回归系数进行正则化，即 **Lasso（Least absolute shrinkage and selection operator）**回归，此时带约束的最优化问题就变为[3]

$$\hat{\beta}^{lasso}(t) = \arg\min_{\beta} \frac{1}{N} \sum_{i=1}^{N} (y_i - x_i'\beta)^2 \tag{5.3}$$

$$s.t. \|\beta\|_1 \leq t$$

[2]Stata 官方的 lasso 命令中，由于预先进行标准化，从而设定 $\psi_k = 1$，实际使用时也可以人为修改这一数值。

[3]为了与文献保持一致，这里目标函数除以 N 处理，这并不改变最优化结果。

或者等价的：

$$\hat{\beta}^{lasso}(\lambda) = \arg\min_{\beta} \frac{1}{N}\sum_{i=1}^{N}\left(y_i - x_i'\beta\right)^2 + \lambda\|\beta\|_1$$

可以证明，以上优化问题为一个凸优化（convex optimization）问题，从而以上的优化问题有非常有效的全局算法，比如 **LARS**（least angle regression，Efron 等人，2004）算法，因而尽管 Lasso 回归不像岭回归一样有闭式解，但是在计算上仍然是非常有效率的。

Lasso 回归的一个优点是其同时实现了收缩以及**变量选择**（variable selection）。为了说明这一点，如图 (5.2) 所示，图中黑色线为普通最小二乘目标函数的等高线，$\hat{\beta}^{ols}$ 为普通最小二乘估计量。随着 (β_1, β_2) 远离 $\hat{\beta}^{ols}$，目标函数的值也越大。由于岭回归使用了 L^2 范数对 β 的大小进行限制，图 (5.2.a) 中红色区域即岭回归中所给出的约束条件。可以看到当最小二乘目标函数的等高线与约束集相切时，最优化问题 (5.2) 达到了最优。注意，尽管相对于 $\hat{\beta}^{ols}$，$\hat{\beta}^{r}$ 实现了向 0 的收缩，然而岭回归并没有使得任何一个系数严格等于 0。

而图 (5.2.b) 则展示了 Lasso 回归的约束及最优化问题。由于 Lasso 回归使用了 L^1 范数对 β 的大小进行限制，其约束集并非光滑的，而是存在“棱角”，从图中可以看到，最优化问题 (5.3) 的解为角点解，即存在一些系数其值刚好为 0。由于这一性质，Lasso 回归不仅仅像岭回归一样对估计进行了收缩，同时还进行了变量选择，避免了过拟合现象。

由于 Lasso 回归可以对变量进行选择，一个很自然的问题是，这一过程是否可以将真实数据生成过程中非零的参数选择出来，以及估计量的统计性质，包括收敛速度和大样本分布的情况是否足够的好。对于收缩估计量 $\hat{\beta}^{s}$，如果其可以将真实的非零参数一致的选择出来，那么我们称其具有**选择一致性**（**selection consistency**），或者**稀疏一致性**（**sparsistency**）。如果我们记 β_0 为真实的 β，令 $S_0 = \{\beta_{0,k} \neq 0, k = 1, ..., K\}$ 为活跃集（active set），从而 S_0^c 即 β_0 中系数为 0 的集合，记 $K_0 = |S_0|$ 为 β_0 中不为 0 的分量的个数，即**稀疏指数**（**sparsity index**），同时记 $\hat{S} = \left\{\hat{\beta}_k^s \neq 0, k = 1, ..., K\right\}$，那么选择一致性定义即

$$P\left(\hat{S} = S_0\right) = 1$$

而如果在选择一致性的基础上，估计量额外满足 $\sqrt{N}\left(\hat{\beta}_{S_0}^s - \beta_{0,S_0}\right) \overset{a}{\sim} N(0, \Sigma)$，而其中 Σ 为估计量

$$\hat{\beta}^{oracle} = \arg\min_{\beta} \sum_{i=1}^{N}\left(y_i - x_i'\beta\right)^2$$
$$s.t.\ \beta_{S_0^c} = 0$$

的协方差矩阵，那么我们称估计量 $\hat{\beta}^{s}$ 具有 **oracle 性质**（**oracle property**, Fan 和 Li，2001）。Oracle 性质意味着估计量 $\hat{\beta}$ 的统计性质要至少要与假设知道哪些系数为 0 的前提下所得到的估计量（$\hat{\beta}^{oracle}$）性质一样好。

从预测的角度，Lasso 回归通常具有比较好的性质。可以证明，随着 K 和 N 同时趋向于无穷，在 λ 取合适的值的情况下（$\lambda = O\left(\sigma\sqrt{\ln K/N}\right)$），以下关于平均预测误差的不等式：

$$\frac{1}{N}\left\|X\left(\hat{\beta}^{lasso} - \beta_0\right)\right\|_2^2 \leq C \cdot \sigma^2 \frac{\log K}{N} K_0$$

以很大的概率成立（见 Bühlmann 和 van der Geer，2011），其中 $\left\|X\left(\hat{\beta}^{lasso} - \beta_0\right)\right\|_2^2$ 即 Lasso 回归的残差平方和，C 为常数。以上不等式被称为 oracle 不等式，意味着为了达到预测的一致性，真实的模型必须是**稀疏的**（**sparsity**），至少要求 $K_0 = o(N/\log K)$。

然而，Lasso 估计量的选择一致性需要更强的条件，其中比较重要的是不可代表性（irrepre-

sentability），即存在 $\gamma > 0$，有

$$\max_{j \in S_0^c} \left\| \left(X_{S_0}' X_{S_0} \right)^{-1} X_{S_0}' x_j \right\| \leq 1 - \gamma$$

以上条件实际上要求 S_0 中的 x 对于非 S_0 中的 x 的相关性足够的弱，或者解释能力足够的小。最理想的情况下，两类 x 应该正交，此时 $\gamma = 0$。可以证明在一定的条件和适当的 λ 的选取下，Lasso 回归可以以一个比较高的概率达到选择一致性（Hastie，Tibshirani 和 Wainwright，2015）。

然而具体到 oracle 性质，遗憾的是岭回归和 Lasso 回归通常**不具有** oracle 性质：岭回归通常不具有选择一致性，而 Lasso 回归虽然可能达到选择一致性，但是通常并不是一致估计量[4]。Oracle 性质之所以吸引人主要在于其可以保证选择一致性的前提下，同时对非零参数达到了一致且渐近正态的估计，从而可以进一步对其进行统计推断，而简单的岭回归和 Lasso 回归却无法达到这样的效果。

基于此，一种方法是将 Lasso 回归作为变量选择的工具，即挑选出 Lasso 回归中系数不为 0 的变量，进而对 $\hat{S}$ 集合中所对应的参数用 OLS 进行估计，这种做法也被称为**选择后估计**（**post-selection estimator**），或者**后 Lasso 估计**（**post Lasso estimator**）。Leeb 和 Pötscher（2005，2008）指出，通常 Lasso 回归中的一致性或者收敛性都是点态收敛，而非一致收敛，这导致不同的 β_0 会需要不同的样本量来达到需要的统计性质，而实践中往往 β_0 是未知的而样本量是固定的，从而根据 Lasso 进行变量选择后的 OLS 估计的统计性质仍然是复杂的问题。Berk 等人（2013）讨论了选择后估计的推断问题，提出可以通过拓宽置信区间的方法对参数进行合适的区间估计，而 Lee 等人（2016）则讨论了给定所选模型下的推断问题。

而另一方面，虽然有以上的缺点，选择后估计也的确有其优势。Belloni 和 Chernozhukov（2013）比较了模型选择后估计与 Lasso 估计，发现在高维稀疏模型中，即使 Lasso 方法可能无法达到选择一致性，不过选择后估计的在收敛速度和偏差方面表现得至少与 Lasso 估计一样好，在一些情况下其收敛速度和偏差可以严格由于单纯的 Lasso 估计，甚至在极端情况下，如果 Lasso 完美的选择出了正确的模型，选择后估计就成为 oracle 估计量。

5.1.4 超参数选择

在以上的岭回归和 Lasso 回归中，拉格朗日乘子 λ，或者等价的 t 控制了收缩的程度，一个更大的 λ 或者更小的 t 意味着更“小”的模型或者更多的向 0 收缩，从而偏差大但是方差小；反过来，更小的 λ 则意味着更“大”的模型。因而在这里同样有偏差-方差的权衡问题：应该存在一个“最优”的 λ 使得预测误差能够达到最小。

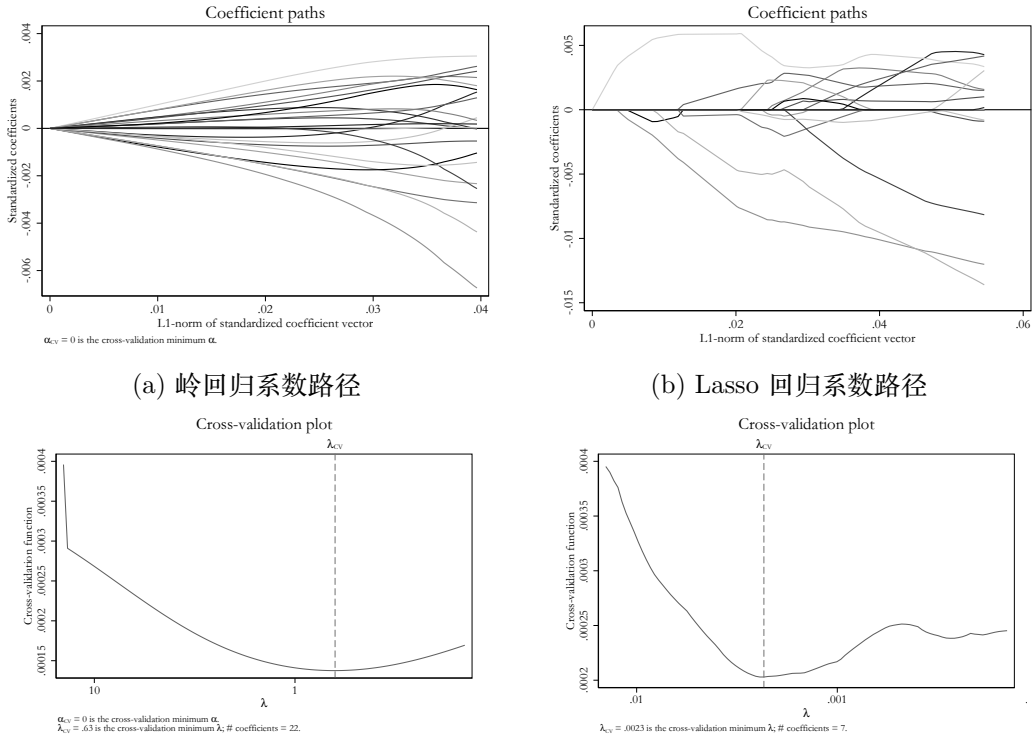
注意这里由于 $\hat{y} = x' \hat{\beta}^{lasso/r}$，从而参数 λ 并不参与模型的预测，λ 仅仅是为了“训练”[5]模型所需要的参数：$\hat{\beta}^{lasso/r}$ 是一个 λ 的函数，给定一个 λ 就会有一个对应的模型和参数，我们把这种训练模型之前需要设定的参数称为“**超参数**”（**hyperparameter**）。在实际应用时通常会调整超参数以找到预测效果最好的参数，因而超参数也被称为“**调整参数**”（**tuning parameter**）。

最常用的方法是数据驱动的交叉验证法。交叉验证法的步骤与2.2.2节介绍的步骤相同，出于计算速度的要求，一般进行 5-10 折的交叉验证即可。实际进行时会将样本随机分为 S 折，然后将 λ 从一个很大的数字到 0 进行格点（grid）化，针对每一个 λ，使用交叉验证的方法留出 1 折数据作为验证集，其他的 $S-1$ 折作为训练集，再在留出的 1 折数据中进行预测，最终挑出能够使得均方误差最小的 λ。

例 5.1. 我们使用 Hsiao、Ching 和 Wan（2012）的数据，使用其他国家和地区的 gdp 增长率对香港回归前的 gdp 增长率进行预测。注意到数据集中香港回归前的数据只有 18 条，然而排除了中国大陆和台湾省之后的国家和地区数目有 22 个，从而出现了 $N > K$ 的情况。我们可以考虑分别使

[4]在收缩估计量中我们通常会将“偏差”指代“渐近偏差”，即 $N \to \infty$ 时的偏差，从而这里存在渐近偏差意味着估计量并不是一致估计量。

[5]在机器学习中通常将模型参数的过程称为“训练”（train）模型。



(a) 岭回归系数路径 (b) Lasso 回归系数路径

(c) 岭回归交叉验证 (d) Lasso 回归交叉验证

图 5.3: 岭回归与 Lasso 回归的系数路径与交叉验证

用岭回归和 Lasso 回归来解决这一问题，并从这些可以用于相关 GDP 增长率的国家和地区中进行挑选。在 Stata 中，岭回归可以使用 elasticnet 命令进行，配合 alpha(0) 选项即得到了岭回归：

```
use "datasets/hcw.dta"
elasticnet linear HongKong Australia-Thailand if _n<=18, rseed(5) sel(cv) alpha
    (0)
```

其中 linear 代表要在线性回归中加入惩罚项，选项 sel(cv) 代表使用交叉验证选取 λ，而 alpha(0) 代表进行岭回归估计，rseed() 是为了保证每次运行结果都相同的一个随机数种子。估计完成后可以使用 lassoinfo 命令查看选取的最优 λ，也可以使用 coefpath 命令和 cvplot 命令可以分别画出系数路径图和交叉验证图。图5.3(a) 展示了岭回归的系数路径，该图左手边对应着较大的 λ 而右手边对应着较小的 λ；图5.3(c) 则展示了随着 λ 变化交叉验证的均方误差变化[6]。此外还可以及使用 etable 命令查看系数。此外，还可以在 sel(cv) 中加入 folds 选项以设定交叉验证的折数，比如 sel(cv, folds(9)) 即进行 9 折交叉验证。如果需要进行 Lasso 回归，可以设定 alpha(1)，或者直接使用 lasso 命令：

```
lasso linear HongKong Australia-Thailand if _n<=18, rseed(5) sel(cv)
```

除了以上的估计后命令外，还可以使用 lassocoef 命令查看 Lasso 回归保留了哪些变量，比如这里只有澳大利亚、意大利、日本、韩国、墨西哥、美国和马来西亚这几个国家被保留下来用于预测香港的 GDP 增长率。观察图5.3可以发现，Lasso 回归中多数国家的系数都被收缩到 0，而岭回归则并没有完成变量选择。当然，也可以使用 preidct 命令进行预测，在预测时可以使用经过收缩的 Lasso 回归系数，也可以使用选择后估计进行预测：

```
predict HongKong_lasso
label variable HongKong_lasso "Lasso"
predict HongKong_lasso_post, post
label variable HongKong_lasso_post "PostLasso"
```

[6]Stata 内部对 $\ln\lambda$ 进行一个等距的格点化，从而该图的横坐标是使用的对数坐标。

其中加入 post 选项的是进行后 Lasso 估计进行的预测。如果需要手动进行后 Lasso 估计，可以使用 Lasso 估计之后保留的 e(post_sel_vars)，该宏记录了所有被 Lasso 命令保留的变量，其中第一个变量为被解释变量：

```
lasso linear HongKong Australia-Thailand if _n<=18, rseed(5) sel(cv)
local post_vars=e(post_sel_vars)
reg `post_vars'
```

除了交叉验证外，一些信息准则，特别是 BIC 也会被用于 λ 的选择中，虽然这些信息准则应用在 Lasso 的理论基础仍然不是非常清晰，然而在一些模拟研究中，BIC 通常具有不错的表现。

此外，文献中对于 λ 的取值也有非常多的讨论（如 Bickel、Ritov 和 Tsybakov，2009；Belloni 和 Chernozhukov，2011；Belloni 等，2012；Belloni、Chernozhukov 和 Hansen，2014；Belloni、Chernozhukov 和 Wei，2016），其形式大概如：

$$\lambda = \frac{c}{\sqrt{N}}\hat{\sigma}\Phi^{-1}\left(1-\frac{\gamma}{2K}\right)$$

其中根据 Belloni 和 Chernozhukov（2011）建议，$c=1.1$，γ 为未能成功排除真值为 0 的参数的概率，Stata 中取 $\gamma = \frac{0.1}{\ln(\max\{K,N\})}$。根据如上形式就可以使用插入（plug-in）法估计 λ。

例 5.2. 在 Stata 中，如果使用 BIC 选取超参数，可以使用 sel(bic) 选项：

```
lasso linear HongKong Australia-Thailand if _n<=18, rseed(5) sel(bic)
```

而如果需要使用插入法，可以使用 sel(plugin)：

```
lasso linear HongKong Australia-Thailand if _n<=18, rseed(5) sel(plugin)
```

默认插入法假设异方差进行处理，也可以使用 sel(plugin, hom) 在同方差假设下进行估计。

5.1.5 其他收缩方法

作为一种收缩估计量，岭回归和 Lasso 回归分别使用了 L^2 范数和 L^1 范数，而类似的可以定义很多不同的收缩方法。

首先，我们可以将 Lasso 回归和岭回归结合起来，即同时使用 L^2 范数和 L^1 范数的线性组合作为惩罚项，即最小化

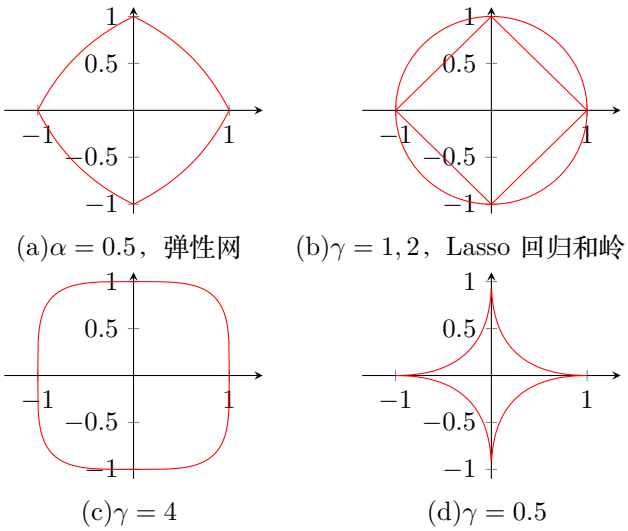
$$\hat{\beta}^{elastic}(\lambda,\alpha) = \arg\min_{\beta}\frac{1}{N}\sum_{i=1}^{N}(y_i - x_i'\beta)^2 + \lambda\left(\alpha\|\beta\|_1 + \frac{1-\alpha}{2}\|\beta\|_2^2\right)$$

其中 $\alpha\in[0,1]$ 为超参数。易知当 $\alpha=0$ 时以上即岭回归，而当 $\alpha=1$ 时即 Lasso 回归，以上回归被称为**弹性网**（**elastic net**，Zou 和 Hastie，2005）。图5.4(b) 展示了 $\alpha=0.5$ 时与以上最优化问题等价的约束优化问题的可行集，相比较于 Lasso 回归，其可行集虽然有不可导的“棱角”，但是更加光滑，因而相比较于 Lasso，其更容易保留变量，这也是弹性网提出的初衷：对于一些高度相关的变量，使用 Lasso 通常会丢掉其中的某些变量，而这些变量可能是重要的、有解释力的，而弹性网往往可以保留这些强相关的变量。

其次，如果我们将 L^1 和 L^2 范数改为更加一般的 L^γ 范数，那么我们就得到了 **Bridge** 估计量，即最小化

$$\hat{\beta}^{brige}(\lambda) = \arg\min_{\beta}\frac{1}{N}\sum_{i=1}^{N}(y_i - x_i'\beta)^2 + \lambda\sum_{k=1}^{K}|\beta_k|^\gamma$$

显然当 $\gamma=1$ 时，即 Lasso 回归，而当 $\gamma=2$ 时，就得到了岭回归。图5.4(b)-(d) 展示了 γ 取不同值时的可行集。值得注意的是，当 $\gamma<1$ 时，可行集并不是一个凸集，当然以上优化问题也不是一个凸优化问题，在计算上存在困难，因而文献中经常假设 $\gamma\geq 1$。



(a)$\alpha = 0.5$，弹性网 (b)$\gamma = 1, 2$，Lasso 回归和岭回归

(c)$\gamma = 4$ (d)$\gamma = 0.5$

图 5.4: 不同估计量的可行集

然而，$\gamma < 1$ 时的 Bridge 回归在偏差和选择一致性上都更有优势（Bühlmann 和 van der Geer，2011），相比较于 Lasso 具有更好的性质。然而优化问题限制了 $\gamma < 1$ 时的模型应用，此时我们可以使用 Zou（2006）提出的适应性 **Lasso**（**adaptive Lasso**）：

$$\hat{\beta}^{ada}(\lambda) = \arg\min_{\beta} \frac{1}{N}\sum_{i=1}^{N}(y_i - x_i'\beta)^2 + \lambda\sum_{k=1}^{K} w_k |\beta_k|$$

其中

$$w_k = \frac{1}{\left|\hat{\beta}_k^o\right|^{\theta}}$$

为权重，$\theta > 0$，$\hat{\beta}_j^o$ 为 β 的一个一致估计量作为初始估计量，比如 OLS 估计量。如果 $K > N$，也可以使用岭回归或者 Lasso 回归等估计量。如果初始估计量使用 Lasso 回归，注意到如果 $\left|\hat{\beta}_k^o\right| = 0$，那么 $w_k = \infty$，则 $\hat{\beta}_k^{ada} = 0$，从而在估计适应性 Lasso 时，只需要对第一步 Lasso 回归中非 0 的参数进行估计即可。以上步骤可以反复进行，即将 $\hat{\beta}^{ada}$ 作为初始估计量计算新的权重，再次计算适应性 Lasso，当然一般实践中只进行一次迭代效果已经足够好。

适应性 Lasso 具有很多的优良性质。首先从计算上来看，适应性 Lasso 仅仅在 Lasso 的基础上做了参数惩罚项上的权重调整，其计算代价相比 Lasso 是完全相同的。而从统计性质上，对于 $\theta > 0$，适应性 Lasso 总是给初始估计量中 $\left|\hat{\beta}_k^o\right|$ 比较小的参数以更大的惩罚，从而使得估计量更加容易收缩至 0，从而降低 S_0^c 中的参数被保留的可能性；而对于 $\left|\hat{\beta}_k^o\right|$ 比较大的参数以更小的惩罚，从而无偏性受到的影响更小。综合起来，适应性 Lasso 可以达到更好的统计性质，实际上 Zou（2006）证明适应性 Lasso 可以达到 oracle 性质。

例 5.3. 在 Stata 中，如果需要做适应性 Lasso，可以直接在 Lasso 命令中加入 sel(adaptive) 选项：

```
lasso linear HongKong Australia-Thailand if _n<=18, rseed(5) sel(adaptive)
lassocoef
```

可以发现经过适应性 Lasso 后，预测变量只剩下日本、韩国和墨西哥了。

以上 Lasso 的变种都集中在对惩罚项的设定上，而 Belloni，Chernozhukov 和 Wang（2011）

则建议将目标函数中的残差平方和改为其平方根，即

$$\hat{\beta}^{ada}(\lambda) = \arg\min_{\beta} \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - x_i'\beta)^2} + \frac{\lambda}{N}\sum_{k=1}^{K}|\beta_k|$$

即**平方根 Lasso**（**square-root Lasso**）。该方法的好处是在使用插入法计算 λ 的最优选择时，无须估计 σ。此外，该方法可以在误差项非正态分布时也达近乎 oracle 的收敛速度。

例 5.4. 在 Stata 中，如果需要做平方根 Lasso，可以使用 sqrtlasso 命令：

```
sqrtlasso HongKong Australia-Thailand if _n<=18, rseed(5)
lassocoef
```

由于平方根 Lasso 只适用于线性回归，从而无需加入 linear。

除此之外，常用的 Lasso 类方法还有**分组 Lasso**（**group Lasso**，Yuan 和 Lin，2006）以及 **SCAD**（**smoothly clipped absolute deviation**，Fan 和 Li，2001）等多种方法，在此不再赘述，感兴趣的读者可以参考 Chan 和 Mátyás（2022）以及 Bühlmann 和 van der Geer（2011）、Hastie，Tibshirani 和 Wainwright（2015）等相关材料。

5.1.6 非线性模型与 Lasso

以上通过收缩进行正则化的方法可以很自然的推广到我们之前所遇到的非线性模型，包括非线性最小二乘、广义线性模型等，当然也包括 Tobit 等非线性模型。

比如，对于非线性最小二乘，其 Lasso 估计量为

$$\min_{\beta}\frac{1}{N}\sum_{i=1}^{N}\left[y_i - g(x_i,\beta)\right]^2 + \lambda\|\beta\|_1$$

当然正则化部分也可以使用岭回归、Bridge、适应性 Lasso 等多种不同的惩罚项形式。需要提示的是，由于 $g(\cdot,\cdot)$ 函数并非线性函数，因而如果某一个 $\hat{\beta}_j = 0$ 并不一定代表与之关联的变量被剔除掉。比如如果

$$g(x,\beta) = e^{\beta_1 x_1 + \beta_2 x_2}\cos(\beta_3 x_2)$$

显然如果 $\hat{\beta}_3 = 0$ 并不代表 x_2 被剔除了。当然，如果函数形式是一个“**单指数**（**single index**）”函数：

$$g(x,\beta) = g(x'\beta)$$

其中 $g(\cdot)$ 为一个非线性函数，那么 $\hat{\beta}_j = 0$ 意味着相对应变量被剔除。

另外一种常见的非线性模型的形式是极大似然估计。通常极大似然估计通过最大化对数似然函数

$$\max_{\beta} L(\beta|y,x)$$

得到，我们可以将目标函数乘以 -1 从而将其转化为一个求最小化的问题，再加入惩罚项，比如使用 L^1 正则化项：

$$\min_{\beta} -L(\beta|y,x) + \lambda\|\beta\|_1$$

比如，对于 Probit 和 Logit 回归，将式 (3.1) 带入就得到

$$\min_{\beta} -\frac{1}{N}\sum_{i=1}^{N}\left[y_i\ln(F(x_i'\beta)) + (1-y_i)\ln(1-F(x_i'\beta))\right] + \lambda\|\beta\|_1$$

解以上最优化问题即可得到 Probit/Logit 回归的 Lasso 回归结果。如果需要使用交叉验证法选取

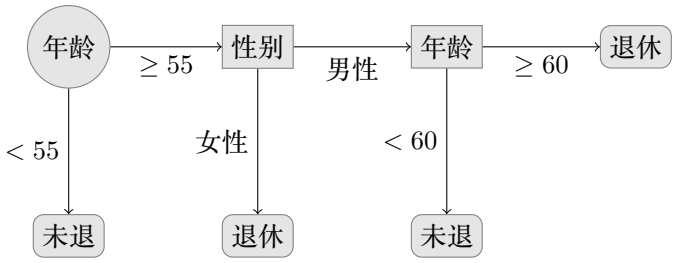


图 5.5: 预测退休的决策树

超参数（通常是 λ），可以通过最小化样本外的预测误差，或者最大化样本外的对数似然函数值进行选择。

需要注意的是，虽然在线性回归的目标函数中加入正则化项通常都是凸优化问题，并且可以使用 LARS 算法等优化方法，然而由于非线性模型中目标函数本身可能是非凸的，导致一般非线性模型的收缩估计量可能难以计算。不过，对于广义线性模型而言，通常存在比较好的优化算法，从而一般的 Probit、Logit、Poisson 回归如果加入正则化项都可以被高效的计算。Stata 以及 R、Python、Julia 等计量、统计软件中通常都可以实现以上这些回归的 Lasso 版本，在 Stata 中如果需要进行以上几种回归，只需要将 lasso 相关命令中的 linear 换成 probit、logit、poisson 即可。

5.2 决策树与随机森林

Lasso 回归在处理高维问题上有其优势，然而仍然需要对函数形式进行假设。我们之前已经学过一些非参数回归的方法，然而这些方法很容易面临维数的诅咒。在机器学习中，针对不同类型被解释变量，通常有不同的方法可以在不假设函数形式的前提下进行预测，比如基于树的方法（tree-based methods）、神经网络（neural networks）等，都可以用于回归和分类。这一节我们主要介绍基于树的方法。

5.2.1 决策树的基本概念

决策树（**decision tree**）使用一种称为**树**（**tree**）的数据结构解决分类和回归问题。数据结构中的“树”指的是一种层级结构，为了解释这个概念，我们考虑如何判断一个人是否已经退休。

图5.5展示了一个层级结构，该结构也是一个判断是否退休的规则。图中圆形的节点代表了所有的样本，我们称其为**根节点**（**root node**）。圆形中的“年龄”代表我们根据年龄是否大于等于 55 岁将样本区分为两个子样本，其中小于 55 岁的样本被判定为没有退休。由于该节点是从根节点划分出来的，我们将其称为根节点的**子节点**（**child node**），而根节点是该节点的**父节点**（**parent node**）。一般而言，子节点也可以有子节点，然而对于小于 55 岁的节点已经没有子节点，我们将其称为**叶子节点**（**leaf node**）。而对于大于等于 55 岁的样本，我们又可以根据性别判断是否退休：如果大于等于 55 的是女性样本，则预测为退休，否则需要继续根据年龄分组。此外，注意到图5.5展示的树中，每个节点如果有子节点，子节点一定是两个，我们将这种在每个非叶子节点上二分的树称为**二叉树**（**binary tree**）。

使用树这一结构进行建模这一思想有很长的历史，比如在统计学和经济学中，在基于树的方法广泛应用之前，Morgan 和 Sonquist（1963）就使用了类似的思路和结构。而决策树这一工具的广泛使用得益于一些算法的发明和推广，包括 Brieman 等人（1984）提出的 **CART**（classification and regression trees）、Quinlan（1986）提出的 **ID3**（iterative dichotomiser 3）以及 Quinlan（1993）提出的 ID3 算法的改进——**C4.5** 等。正如 CART 的名字所显示的那样，决策树可以解决两类问题：分类问题和回归问题，分别被称为**分类树**（**classification tree**）和**回归树**（**regression tree**）。

实际上不管是分类数还是随机树，其本质是对特征空间的划分（partition of feature space），即将所有 x 的取值范围划分为一个一个“矩形”，在每个矩形内部赋予同样的预测值。比如，图5.6(a)

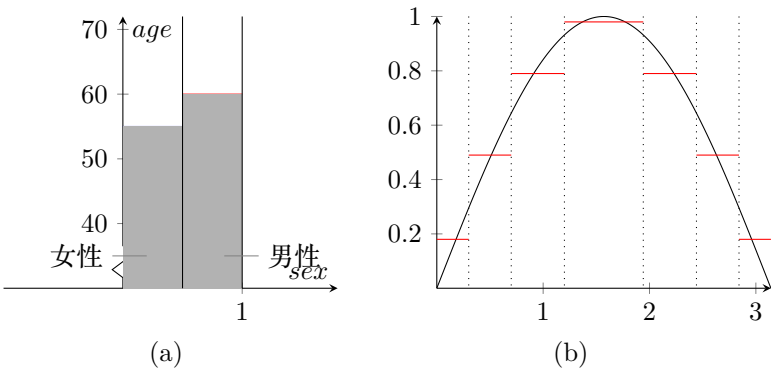


图 5.6: 特征空间划分

展示了退休的例子中的特征空间及其划分，灰色的两个矩形共同构成了预测为不退休的区域，而白色的部分构成了预测为退休的区域。类似的，图5.6(b) 展示了对一个一元函数（正弦函数）的预测，回归树所做的仅仅是将特征空间划分为很多不同的区域，在每个区域上用常数进行预测，从而呈现了阶梯型的函数。

我们可以首先将 $x_i = (x_{i1}, ..., x_{iK})'$ 的取值范围划分为 M 个区域：$R_1, ..., R_M$，对于回归树，可以定义一个阶梯函数：

$$f(x) = \sum_{m=1}^{M} c_m \cdot 1\{x \in R_m\}$$

c_m 即区域 R_m 的预测值，给定划分，我们可以使用 R_m 内的 y 的均值对 c_m 进行估计：

$$\hat{c}_m = \frac{\sum_{i=1}^{N} y_i \cdot 1\{x_i \in R_m\}}{\sum_{i=1}^{N} 1\{x_i \in R_m\}}$$

而对于分类树，如果可能的分类为 $l \in \{1, 2, ..., L\}$，可以定义 L 个阶梯函数：

$$f_l(x) = \sum_{m=1}^{M} p_{ml} \cdot 1\{x \in R_m\}$$

p_m 为区域 R_m 中 y 取值为 l 的概率，给定划分，我们可以使用 R_m 内的 y 取 l 的比例对 p_{ml} 进行估计：

$$\hat{p}_{ml} = \frac{\sum_{i=1}^{N} 1\{y_i = l\} \cdot 1\{x_i \in R_m\}}{\sum_{i=1}^{N} 1\{x_i \in R_m\}}$$

接下来，构建决策树的关键过程即如何获得该划分。

5.2.2 决策树的算法

构建决策树的算法通常是一种贪婪算法（greedy algorithm），即通过迭代的办法，针对每一个节点中的样本，寻找能够最大程度上降低预测误差的方法。因而首先一个重要的问题是，该如何定义划分时的预测误差下降程度？

对于回归树，我们可以使用预测误差的平方。现在考虑我们根据 x_k 对节点进行划分：

$$R_l = \{i | X_{ik} \leq s\}, R_r = \{i | X_{ik} > s\}$$

现在，我们可以根据这一划分分别计算两个子节点的 y 的均值：$\hat{c}_l, \hat{c}_r$，然而分别计算两个节点的

均方误差：

$$Q_l = \frac{\sum_{i=1}^{N} (y_i - \hat{c}_l)^2 \cdot 1\{i \in R_l\}}{\sum_{i=1}^{N} 1\{x_i \in R_l\}}$$

对于 RSS_r 进行类似定义。如此，我们可以在每个节点，找到能够最小化两个节点残差平方和的 k 和 s：

$$\min_{k,s} N_l Q_l + N_r Q_r$$

从而获得在该节点处的划分规则，其中 $N_l = \sum_{i=1}^{N} 1\{x_i \in R_l\}$，从而 $N_l Q_l + N_r Q_r$ 即两个节点的残差平方和。

现在，我们从根节点开始，我们可以递归地（recursively）实施以上步骤，将根节点划分为两个子节点，再将两个子节点各自划分为两个子节点，以此类推。以上步骤在每一个节点都划分为两个子节点，从而可以得到一颗二叉树。

而对于分类树，每个节点都有 y 属于不同类别的样本，我们可以定义一个节点的**纯度**（**purity**）或者相反的，**杂度**（**impurity**）。在计算每个节点 m 的概率估计 $\hat{p}_{ml}$ 之后，我们可以定义几种不同的杂度：

1. **错分率**（**classification error rate**）：如果我们将节点中 y 的取值个数最多的类别作为该节点的预测，那么被错误划分的比率即错分率，可以定义为

$$CER_m = 1 - \max_l \hat{p}_{ml}$$

2. **基尼系数**（**Gini index**）：

$$G_m = \sum_{l=1}^{L} \hat{p}_{ml}(1 - \hat{p}_{ml})$$

3. **交叉熵**（**cross-entropy**）：

$$CEnt_m = -\sum_{l=1}^{L} \hat{p}_{ml} \log \hat{p}_{ml}$$

这些指标越大，意味着节点内的类别更混乱，从而纯度更低或者杂度更高。CART 算法使用 Gini 系数做为杂度的指标，而 C4.5 则使用了交叉熵。

现在再次考虑以上的二分类问题。如果我们将节点 D 划分为两个子节点：D_1, D_2，为了评估划分之后的杂度，我们还需要进行加权，比如对于基尼系数，需要计算

$$G(D_1, D_2) = \frac{|D_1|}{|D|} G_{D_1} + \frac{|D_2|}{|D|} G_{D_2}$$

其中 $|D|$ 代表 D 节点的样本量。而对于交叉熵，可以定义信息增益：

$$Gain(D, D_1, D_2) = CEnt_D - \left(\frac{|D_1|}{|D|} CEnt_{D_1} + \frac{|D_2|}{|D|} CEnt_{D_2} \right)$$

以及增益率：

$$Gain\_ratio(D, D_1, D_2) = \frac{Gain(D, D_1, D_2)}{-\left(\frac{|D_1|}{|D|} \log \frac{|D_1|}{|D|} + \frac{|D_2|}{|D|} \log \frac{|D_2|}{|D|} \right)}$$

C4.5 算法中综合使用了信息增益和增益率（可参考周志华，2016）。

同样使用递归的办法，我们可以从根节点开始生成一颗二叉树。此外，这些算法还有一些其他的问题，比如缺失值的处理等，可以参考 Hastie，Tibshirani 和 Friedman（2009）以及周志华（2016），在此不再赘述。

5.2.3 过拟合与超参数选择

根据以上介绍的算法可以构建一棵树，然而一个很重要的问题是，如果以上步骤无限制的重复下去，最终每个叶子节点中将会都只有一个样本，从而在每个叶子节点中都达到了完美拟合，这显然是一种过拟合。为此，我们需要找到方法限制过拟合的问题。

在决策树中，我们通常将限制过拟合的过程称为“**剪枝**”（**pruning**）。剪枝操作可以通过预留验证集的方法操作，此时可以使用预剪枝和后剪枝两种方法：通过评估验证集上划分是否可以提高预测精度来决定是否对节点进行划分，即预剪枝；通过构建一颗完整的数，在验证集上判断是否山道两个叶子节点可以提高预测精度，即后剪枝。

另一种方法是通过一些规则限定树的大小。比如，我们可以规定一个杂度下降的一个最小幅度，如果划分一个节点杂度下降的幅度小于这个最小幅度，就停止划分该节点，并将该节点作为叶子节点。再比如，我们可以通过限制树的层数（深度）限制树的大小。

或者，我们可以限制叶子结点的数量。如果记 $|T|$ 为叶子结点的数量，我们可以要求 $|T|$ 小于等于某一个上限。此外，还可以将 $|T|$ 作为正则化项，通过最小化

$$\min \sum_{m=1}^{|T|} N_m Q_m + \alpha |T|$$

对于回归问题，Q_m 即均方误差，而对于分类问题，Q_m 则为基尼系数、交叉熵等的度量。注意其中 α 是一个超参数，可以通过交叉验证等方法确定 α。

5.2.4 集成学习与随机森林

集成学习（**ensemble learning**）是一类将多个不同的机器学习模型结合在一起完成预测任务的方法，这里多个不同的机器学习模型可以是不同的回归、不同的决策树等方法。比如，我们可以将一些深度比较浅的决策树结合在一起进行预测，往往可以达到比单一的树更好的预测效果。由于单颗树往往很不稳定，一些样本上的轻微变化都会导致最终模型的很大差异，使用集成学习方法往往可以解决这一问题。

集成学习大体可以分为两种，一种是不同的模型之间存在着相互依赖关系，必须按照顺序训练的方法，比如 boosting 方法（包括 AdaBoot 和 gradient boosting 等）；另一种是不同的模型之间相互不存在依赖关系，可以独立训练的方法，比如 bagging 和随机森林等方法。

Boosting 方法通常是一类**可加模型**（**additive model**）：

$$f(x) = \sum_{m=1}^{M} \beta_m b(x, \gamma_m)$$

如果令 $f_0(x) = 0$，boosting 方法迭代地方法逐步解出 β_m, γ_m 从而得到最终的模型。由于采用了迭代的方法，因而不同的 m 之间并不是独立的。常见的 boosting 方法如 AdaBoot 和 gradient boosting 的算法和性质可以参考 Hastie，Tibshirani 和 Friedman（2009），在此不再赘述。

另一种集成学习的方法是训练一些相互不依赖的模型。其中一种简单的方法是，将样本进行有放回抽样，即 bootstrap 样本，重复 B 次就可以得到 B 个不同的样本，对每一个 bootstrap 样本都进行模型训练，得到 B 个模型，然后将 B 个模型进行结合，比如简单的平均起来：

$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x)$$

就可以得到最终的预测。如果是分类问题，也可以通过投票的方式，即 B 颗树中预测出现最多的分类即为预测。这种方法也叫作 **bagging**（**bootstrap aggregation**）。该方法对于偏差并没有太大影响，但是可以大大降低方差，从而提高模型的预测能力。

注意到由于进行的是有放回抽样，从而一般情况下我们从 N 个样本中抽取容量为 N 的 bootstrap 样本，每次也不会相同，每个 bootstrap 样本可能包含着重复的样本。实际上如果 N 足够大，每个 bootstrap 样本大约只包含了 63.2% 的原始样本中的个体，而剩下的 36.8% 的样本往往可以用作验证集，即包外（out-of-bag）估计。

而**随机森林**（**random forest**）则在 bagging 的基础上，额外对特征也进行抽样。假设有 K 个预测变量，随机森林在每次 bootstrap 样本的训练时，仅仅使用随机抽取的 $\tilde{K}$（对于分类问题 $\approx \sqrt{K}$，回归问题 $\approx K/3$）个预测变量构建决策树，如此构建 B 颗树，从而构成随机森林。之所以对预测变量也进行抽样，其原理是降低了不同决策树之间的相关性，从而进一步降低了模型的方差。

例 5.5. Stata 中可以通过 pystacked[7]命令调用 Python 中的 scikit-learn 包[8]进行随机森林的估计。比如对于例3.1可以使用如下代码：

```
pystacked exit_labor `x', type(classify) m(rf) cmdopt1(n_estimators(100)
    max_leaf_nodes(20))
```

以上命令只能给定一些超参数（如树的数量、最大的叶子节点数、最小杂度下降以及树的深度等），对于超参数的选择，可以在 Stata 中通过调整超参数实现，当然也可以直接使用 Python 挑选好超参数后在 Stata 中直接使用。

练习题

问题 5.1. 证明：

$$\mathbb{E}\left[\left(\hat{\theta}-\theta_0\right)^2|X\right]=\mathbb{V}\left(\hat{\theta}|X\right)+\left[\text{Bias}\left(\hat{\theta}|X\right)\right]^2$$

即给定数据 X，均方误差可以分解为方差与偏差的平方之和。

问题 5.2. 对于预测问题，即如果可以观察到数据 X,Y，同时希望在 x 处预测，请证明：

$$\mathbb{E}\left[\left(y-\hat{f}\left(x\right)\right)^2|X,Y,x\right]=\mathbb{E}\left[\left(y-f\left(x\right)\right)^2|x\right]+\mathbb{V}\left[\hat{f}\left(x\right)|X,Y,x\right]+\left(\text{Bias}\left[\hat{f}\left(x\right)|X,Y,x\right]\right)^2$$

其中 $f\left(x\right)=\mathbb{E}\left(y|x\right)$，$\hat{f}\left(x\right)$ 为使用 X,Y 对 $f\left(x\right)$ 的预测模型。

问题 5.3. 请证明岭回归是一个有偏估计量，并写出岭回归的偏差（bias）。

问题 5.4. 如果设定模型 $y_i|x_i,\beta\sim N\left(x_i'\beta,\sigma^2\right)$，以及先验：$\beta\sim N\left(0,\tau^2I\right)$，请计算 β 的后验分布，并与岭回归结果进行比较。

问题 5.5. 证明如果 X 的每一列相互正交，那么岭回归的系数和 OLS 的回归系数满足：

$$\hat{\beta}_k^r=\frac{\left(x^k\right)'x^k}{\left(x^k\right)'x^k+\lambda}\hat{\beta}_k$$

问题 5.6. 证明使用运行中标准化的方法和事前标准化的方法是等价的。

问题 5.7. 对于线性回归，我们可以通过在 OLS 的目标函数上加入惩罚项，也可以通过其极大似然估计（最大化似然函数式 (1.9) 得到）得到，两者是等价的。请证明如果分别对其加入 L^1 或者 L^2 正则化项，其拉格朗日乘子：

$$\lambda_{LS}=2\hat{\sigma}^2\lambda_{ML}$$

其中 λ_{LS} 和 λ_{ML} 分别为使用 OLS 目标函数时和极大似然估计时的拉格朗日乘子。

问题 5.8. 为什么 bootstrap 样本只包含了约 63.2% 的原始样本中的个体?

[7]使用前需安装：net install pystacked, from(https://raw.githubusercontent.com/aahrens1/pystacked/main) replace
[8]具体使用手册可参考其官网：https://scikit-learn.org/stable/

第一章　计量经济学中的机器学习

在第??章中，我们已经介绍了一些机器学习的基本算法。机器学习在预测、分类等问题上已经展现了其强大的能力，然而计量经济学并非以预测为最终目的，而是以因果解释和结构参数识别为主要目的。由于机器学习方法往往需要通过正则化等手段降低方差，但是这会相应的带来偏差的问题，此外其系数往往很难进行推断，从而机器学习方法直接应用到计量经济学研究中仍然存在很多挑战。目前机器学习方法与计量经济学的结合已经有很多新进展，比如**双重/去偏机器学习**（**double/debiased machine learning**）和**因果森林**（**causal forest**）等。本章将主要介绍以上的新进展。

1.1　Lasso 与控制变量选择

考虑如下部分线性（partially linear）模型：

$$y_i = \alpha \times w_i + g(f_i) + \epsilon_i \tag{1.1}$$

我们假设 $\mathbb{E}(\epsilon_i | w_i, f_i) = 0$，从而外生性假设成立。此外，假设核心解释变量

$$w_i = m(f_i) + v_i \tag{1.2}$$

其中 $\mathbb{E}(v_i | f_i) = 0$，从而建模了协变量 f_i 对 w_i 的影响，或者 f_i 与 w_i 之间的相关性。由于 $g(\cdot)$ 和 $m(\cdot)$ 函数形式未知，我们可以使用一些控制变量 x_i 的线性组合来逼近两个函数，从而有

$$\begin{aligned} y_i &= \alpha \times w_i + x_i'\beta + r_{gi} + \epsilon_i \\ w_i &= x_i'\gamma + r_{mi} + v_i \end{aligned} \tag{1.3}$$

其中 x_i 可以是 f_i 的多项式逼近等，而 r_{gi}, r_{mi} 分别为用 $x_i'\beta$ 和 $x_i'\gamma$ 逼近 $g(\cdot)$ 和 $m(\cdot)$ 的误差。

由于使用了逼近的方法，$x_i = P(f_i)$ 可以是比较高维的。Belloni、Chernozhukov 和 Hansen（2014）研究了这一问题，发现如果简单的使用 Lasso 对 x_i 进行变量筛选，其结果 $\hat{\alpha}$ 通常是有偏、非正态的。由于对 x_i 进行变量选择时具有不确定性，$\hat{\alpha}$ 的分布有时会出现多峰的情形，这会造成统计推断上的困

难。为此，他们提出了一种“双重选择”（**double selection**）的方法。首先，将式 (1.3) 中的第二个式子带入到第一个式子，我们得到了：

$$\begin{aligned} y_i &= x_i'\zeta + r_i + \eta_i \\ w_i &= x_i'\gamma + r_{mi} + v_i \end{aligned} \tag{1.4}$$

其中 $\zeta = \alpha\gamma + \beta$，$r_i = \alpha r_{mi} + r_{gi}$，$\eta_i = \alpha v_i + \epsilon_i$。如此，为了选择 x_i，我们可以使用以上两个“简约式”进行选择。Belloni、Chernozhukov 和 Hansen（2014）提出可以分别对以上两个方程分别进行 Lasso 回归，然后取两个回归中系数非 0 变量的并集作为最终的控制变量。如果取

$$I = \left\{k : \hat{\zeta}_k^{lasso} \neq 0\right\} \cup \left\{k : \hat{\gamma}_k^{lasso} \neq 0\right\}$$

其中 $\hat{\zeta}_k^{lasso}$ 和 $\hat{\gamma}_k^{lasso}$ 分别为对式 (1.4) 两个回归进行 Lasso 回归的估计结果。最后，使用 I 中的变量进行如下回归：

$$y_i = \alpha \times w_i + \sum_k x_i'\xi + \varepsilon_i, k \in I$$

以上估计量被称为“后双重 **Lasso** 估计量”（**post-double-Lasso estimator**）。Belloni、Chernozhukov 和 Hansen（2014）证明在一定的稀疏性假设下，以上估计量为渐进正态的：

$$\left[\left(\mathbb{E}v_i^2\right)^{-1}\mathbb{E}\left(v_i^2\epsilon_i^2\right)\left(\mathbb{E}v_i^2\right)^{-1}\right]^{-1/2}\sqrt{N}\left(\hat{\alpha} - \alpha_0\right) \overset{a}{\sim} N\left(0,1\right)$$

例 1.1. 在 Stata 中，命令 dsregress[1] 实现了以上算法。比如在 NTV 的例子中，使用 1999 年的数据研究了观看到 NTV 的概率对于投票的影响。为了保证外生性其加入了一些人口、工资等滞后变量的多项式进行控制。我们可以使用双重选择的方法对这些多项式进行模型选择：

代码 1.1: NTV 的双重选择估计

```
// ntv_double_selec.do
clear
set more off
use datasets/NTV_Aggregate_Data_reshaped.dta
tsset tik_id year
// 社会经济变量以及人口、工资的多项式
forvalues i=1/5{
    gen lag_log_pop_`i'=L.logpop^`i'
    gen lag_log_wage_`i'=L.log_wage^`i'
```

[1] 此外还有 dslogit、dspoisson 等分别针对 Logit 回归和 Poisson 回归的拓展。

```
}
gen lag_nurses=L.nurses
gen lag_doctors_pc=L.doctors_pc
local E_controls "lag_log_pop_* lag_log_wage_* lag_nurses
    lag_doctors_pc"
reghdfe Votes_SPS_ Watch_probit `E_controls' if year
    ==1999, a(region) cluster(region)
dsregress Votes_SPS_ Watch_probit if year==1999, controls
    ((i.region) `E_controls') cluster(region) selec(cv)
```

注意在以上代码中，选项 controls 为控制变量，其中括号中的为不使用 Lasso 选择的变量，而括号外的变量使用 Lasso 进行变量选择。

而对于线性面板数据模型

$$y_{it} = x_{it}'\beta + \alpha_i + \epsilon_{it}$$

Belloni 等人（2016）提出，为了消除个体固定效应，我们可以首先对数据进行组内去平均：

$$\ddot{x}_{it} = x_{it} - \bar{x}_i$$
$$\ddot{y}_{it} = y_{it} - \bar{y}_i$$

进而使用正则化：

$$\hat{\beta}^{cl-lasso} = \min_{\beta} \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \left(\ddot{y}_{it} - \ddot{x}_{it}'\beta\right)^2 + \frac{\lambda}{NT} \sum_{k=1}^{K} \phi_k \left|\beta_k\right|$$

其中 ϕ_k 为惩罚载荷（penalty loadings），其中

$$\phi_k^2 = \frac{1}{NT} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \ddot{x}_{it} \ddot{\epsilon}_{it} \right)^2$$

其中 $\ddot{\epsilon}_{it}$ 不可观测，可以通过迭代的方式计算[2]。以上方法被称为聚类 **Lasso**（**cluster-Lasso**）。经过变量选择后，可以继续使用选出的变量继续进行固定效应估计，即后聚类 **Lasso**（**post-cluster-Lasso**）。

1.2 冗余参数与正交化

在计量经济学中，我们通常会使用机器学习对一些冗余参数（**nuisance parameters**）进行建模，目的是为了对核心参数进行更好的估计。比如在式 (1.3)

[2]见 Belloni 等人（2016）的 Appendix A。

中，α 是我们关心的核心参数，而 β, γ 则是冗余参数。如果使用机器学习的方法对这些参数进行估计，通常正则化会导致这些参数有偏，或者在 Lasso 中被错误地删除，这样就会导致核心参数的估计也被影响。为此，一个比较常见的方法是通过 **Neyman 正交化**（**Neyman orthogonalization**，Neyman，1959，1979）的手段降低这种影响。

为了说明这点，我们考虑式 (1.1) 的部分线性模型。该模型中，对于 α 的识别基于如下矩条件：

$$\mathbb{E}\left(w_i \epsilon_i\right) = \mathbb{E}\left(\left[y_i - \alpha \times w_i - g\left(f_i\right)\right] w_i\right) = 0$$

其中 $g\left(f_i\right)$ 是需要使用机器学习的方法（如 Lasso 等）进行估计和预测的部分，而我们对其系数并不感兴趣，从而是这里的冗余参数。现在考虑如果我们没有很好的估计 $g\left(\cdot\right)$，而是有一个偏差，比如实际中带入的函数为

$$g\left(f_i\right) + t\left[\tilde{g}\left(f_i\right) - g\left(f_i\right)\right] \triangleq g\left(f_i\right) + th\left(f_i\right)$$

其中 $\tilde{g}\left(f_i\right) - g\left(f_i\right)$ 为偏离的方向，而 t 为偏离的程度，从而实际上我们使用的矩条件为

$$\mathbb{E}\left(\left[y_i - \alpha \times w_i - g\left(f_i\right) - th\left(f_i\right)\right] w_i\right)$$

然而如果我们考虑将以上矩条件对 t 求导数，会得到

$$\partial_t \mathbb{E}\left(\left[y_i - \alpha \times w_i - g\left(f_i\right) - th\left(f_i\right)\right] \epsilon_i\right)\big|_{t=0} = -\mathbb{E}\left[w_i \times h\left(f_i\right)\right]$$

由于 w_i 与 f_i 通常是相关的，从而以上偏导数通常不为 0。这也就意味着，如果我们不能很好地估计 $g\left(\cdot\right)$，而是使用了一个带有偏差的版本，那么矩条件本身也会有偏差，从而如果直接令矩条件等于 0 得到估计也会有较大偏差。以上计算的导数可以看做是矩条件对 $g\left(\cdot\right)$ 函数求导，也被称为 Gâteaux 导数，也可以记为 $\partial_g \mathbb{E}\left(\left[y_i - \alpha \times w_i - g\left(f_i\right)\right] w_i\right)$。

现在，我们考虑一个“正交化”的版本。仿照??节的做法，对于式 (1.1)，两边分别对 f_i 求条件期望并相减，得到

$$y_i - \mathbb{E}\left(y_i | f_i\right) = \alpha \times \left[w_i - \mathbb{E}\left(w_i | f_i\right)\right] + \epsilon_i$$

如果记 $\mathbb{E}\left(y_i | f_i\right) = \psi\left(f_i\right)$，而 $\mathbb{E}\left(w_i | f_i\right) = m\left(f_i\right)$，那么此时估计 α 的矩条件为

$$\begin{aligned}
\mathbb{E}\left(\epsilon_i\left[w_i - \psi\left(f_i\right)\right]\right) &= \mathbb{E}\left[\left(\left[y_i - \psi\left(f_i\right)\right] - \alpha \times \left[w_i - m\left(f_i\right)\right]\right)\left(w_i - m\left(f_i\right)\right)\right] \\
&= \mathbb{E}\left[\left(y_i - \psi\left(f_i\right)\right)\left(w_i - m\left(f_i\right)\right) - \alpha\left(w_i - m\left(f_i\right)\right)^2\right] \\
&= 0
\end{aligned}$$

现在我们考虑 $\psi\left(\cdot\right)$ 和 $m\left(\cdot\right)$ 两个函数作为冗余参数，分别使用机器学习的方法

进行估计，同样实际使用的估计版本可能带有偏差：

$$\psi(f_i) + t\left[\tilde{\psi}(f_i) - \psi(f_i)\right] \triangleq \psi(f_i) + th_\psi(f_i)$$
$$m(f_i) + t\left[\tilde{m}(f_i) - m(f_i)\right] \triangleq m(f_i) + th_m(f_i)$$

从而同样计算

$$\begin{aligned}
&\partial_t \mathbb{E}\left[(y_i - \psi(f_i) - th_\psi(f_i))(w_i - m(f_i) - th_m(f_i)) - \alpha(w_i - m(f_i) - th_m(f_i))^2\right] \\
=&\mathbb{E}\left[-h_\psi(f_i)(w_i - m(f_i)) - h_m(y_i - \psi(f_i)) + 2th_\psi(f_i)h_m(f_i)\right] \\
&+ \mathbb{E}\left[2\alpha h_m(f_i)(w_i - m(f_i) - th_m(f_i))\right]
\end{aligned}$$

根据定义，有

$$\begin{aligned}
\mathbb{E}\left[h_\psi(f_i)(w_i - m(f_i))\right] &= \mathbb{E}\left(\mathbb{E}\left[h_\psi(f_i)(w_i - m(f_i))\,|f_i\right]\right) \\
&= \mathbb{E}\left(h_\psi(f_i)\,\mathbb{E}\left[(w_i - m(f_i))\,|f_i\right]\right) \\
&= 0
\end{aligned}$$

类似的有 $\mathbb{E}\left[h_m(y_i - \psi(f_i))\right] = 0$，从而以上导数可以化简为 $2t\mathbb{E}\left[h_\psi(f_i)h_m(f_i)\right] - 2\alpha t\mathbb{E}\left[h_m(f_i)\right]^2$，从而当 $t = 0$ 时，有

$$\partial_t \mathbb{E}\left[(y_i - \psi(f_i) - th_\psi(f_i))(w_i - m(f_i) - th_m(f_i)) - \alpha(w_i - m(f_i) - th_m(f_i))^2\right]\Big|_{t=0} = 0$$

这也就意味着，即使估计的冗余参数 $\psi(\cdot)$ 和 $m(\cdot)$ 存在偏差，其对估计 α 的矩条件的影响也是更高阶的影响，从而影响较小。

在以上步骤中，我们分别将被解释变量 y 和核心解释变量 w 中有关 x 的部分剔除，再将残差部分单独拿出用于进一步估计，从而也被称为“**排除**”(**partialing-out**) 估计。以上对于 α 的估计步骤可以看做是 Chernozhukov、Hansen 和 Spindler(2015a,b) 的特例，其中 $\psi(\cdot)$ 和 $m(\cdot)$ 两个函数可以使用 Lasso、后 Lasso 等方法结合多项式逼近等方法获得，当然也可以使用其他机器学习方法，如决策树、随机森林等。如果使用 Lasso 等方法，可以结合 Belloni、Chernozhukov 和 Hansen（2014）的双重选择方法，如此便诞生了“排除回归”(partialing-out regression)，其计算步骤如下：

1. 使用 w_i 对 x_i（f_i 的多项式等逼近）进行 Lasso 回归，并进行后 Lasso 估计，得到残差 $\hat{e}_w$；
2. 使用 y_i 对 x_i 进行 Lasso 回归，并进行后 Lasso 估计，得到残差 $\hat{e}_y$；
3. 使用 $\hat{e}_y$ 对 $\hat{e}_w$ 做回归得到 α 的估计。

例 1.2. Stata 中的 poregress 命令实现了以上算法，比如我们可以简单的将例1.1中的命令改为 poregress 即可：

```
poregress Votes_SPS_ Watch_probit if year==1999, controls
    ((i.region) `E_controls') cluster(region) selec(cv)
```

此外，工具变量也是实证中常用的方法，而过多的工具变量通常会导致第一阶段的过拟合，从而使得 2SLS 回归的结果向 OLS 偏离。为了解决工具变量过多的问题，Belloni 等（2012）推荐在第一阶段使用 Lasso 回归选取工具变量，实证中也往往使用该方法（如 Gilchrist 和 Sands，2016 等）。而 Chernozhukov、Hansen 和 Spindler（2015a,b）更进一步讨论了工具变量估计时的正交化方法。

考虑如下工具变量设定：

$$y_i = \alpha \times w_i + x_i'\beta + \epsilon_i \tag{1.5}$$

以及第一阶段：

$$w_i = x_i'\gamma + z_i'\delta + u_i$$

其中 $\epsilon_i \perp (x_i, z_i)$，而 x 和 z 都可能是高维的。注意 z 作为工具变量只有在排除了 x 的影响之后才可以作为变异的来源，我们不妨记

$$z_i = \Pi x_i + \zeta_i$$

其中 ζ_i 为排除了 x 相关部分之后的部分。如果记 x 的维数为 K_x，z 的维数为 K_z，那么显然 Π 为 $K_z \times K_x$ 维的矩阵，而 ζ_i 为 $K_z \times 1$ 维的向量。

现在，将第一阶段对 x_i 进行投影可以得到

$$w_i = x_i'\gamma + x_i'\Pi'\delta + \rho_i^w \triangleq x_i'\vartheta + \rho_i^w$$

进而将结构方程对 x_i 进行投影可以得到

$$y_i = \alpha x_i'\vartheta + x_i'\beta + \rho_i^y \triangleq x_i'\theta + \rho_i^y$$

如此该模型中的冗余参数为 $\eta = (\theta', \vartheta', \gamma, \delta)$。

现在，考虑对结构方程式 (1.5) 两边对 x_i 求条件期望，并相减，得到：

$$\mathbb{E}(y_i|x_i) = \alpha \mathbb{E}(w_i|x_i) + x_i'\beta$$

使用式 (1.5) 减去上式，得到

$$\rho_i^y = \alpha \rho_i^w + \epsilon_i \tag{1.6}$$

然而注意到 ρ_i^w 与 ϵ_i 仍然存在相关性，所以需要为其找工具变量。由于 ρ_i^w 中已经排除了 x_i 的影响，所以一个很自然的想法是使用 ζ_i 作为 ρ_i^w 的工具变量，然而 ζ_i 仍然是高维的。注意到

$$\rho_i^w = u_i + \zeta_i'\delta$$

从而可以使用 $\zeta_i'\delta = x_i'\gamma + z_i'\delta - x_i'\vartheta$ 作为工具变量。令 $\upsilon_i = \zeta_i'\delta$，那么矩条件即

$$\mathbb{E}\left[\left(\rho_i^y - \alpha\rho_i^w\right)\upsilon_i\right] = 0$$

以上即使用 υ_i 作为 ρ_i^w 的工具变量对式 (1.6) 进行估计的结果。

可以证明以上矩条件是 Neyman 正交的，即

$$\frac{\partial}{\partial\eta}\mathbb{E}\left[\left(\rho_i^y - \alpha\rho_i^w\right)\upsilon_i\right] = 0$$

如此我们可以使用如下步骤估计 α：

1. 使用 w 对 x 和 z 进行 Lasso 回归得到 $\hat{\gamma}$ 和 $\hat{\delta}$ 以及 $\hat{w}$；
2. 使用 y 对 x 进行 Lasso 回归得到 $\hat{\theta}$ 以及残差 $\hat{\rho}^y$；
3. 使用 $\hat{w}$ 对 x 进行 Lasso 回归得到 $\hat{\vartheta}$；
4. 计算 $\hat{\rho}^w = w - x_i'\hat{\vartheta}$ 以及 $\hat{\upsilon} = x_i'\hat{\gamma} + z_i'\hat{\delta} - x_i'\hat{\vartheta}$；
5. 使用 $\hat{\upsilon}$ 作为 $\hat{\rho}^w$ 的工具变量估计式 (1.6)。

例 1.3. Stata 中可以使用 poivregress 完成以上步骤。比如以下代码中，我们使用出生年份、出生季度与省份的交互作为教育的工具变量，估计了教育的回报问题：

代码 1.2: 使用 Lasso 挑选工具变量

```
// lasso_iv.do
clear all
set maxvar 100000
use datasets/chfs_ind.dta,clear
gen exper=a3007
gen exper2=exper^2
gen edu=a2012
gen ln_income=log(1+labor_inc)
gen quarter=floor(a2006/4)
gen birth_year=a2005
poivregress ln_income (edu=i.quarter#i.province#i.
    birth_year), controls(exper*) sel(cv) cluster(province
    )
```

当然，以上工具仍然很弱，严谨的分析仍然需要对第一阶段的弱工具情况进行分析。

1.3 双重机器学习

以上正交化的方法虽然可以使得估计量更加可靠，但是仍然会存在一些偏差。为了说明这一点，我们仍然考虑式 (1.1) 的部分线性模型。使用正交化的方法，估计量可以写为

$$\hat{\alpha}=\left(\frac{1}{N}\sum_{i=1}^{N}\hat{v}_i^2\right)^{-1}\frac{1}{N}\sum_{i=1}^{N}\hat{v}_i\left(y_i-\hat{g}\left(f_i\right)\right)$$

其中 $\hat{v}_i=w_i-\hat{m}\left(f_i\right)$，$\hat{m}\left(\cdot\right),\hat{g}\left(\cdot\right)$ 可以使用机器学习的方法得到，这里我们需要其收敛速度比 $N^{-1/4}$ 更快，通常在 Lasso 等回归中该条件可以通过稀疏性保证。

以上的估计量中，$\frac{1}{N}\sum_{i=1}^{N}\hat{v}_i^2\xrightarrow{p}\mathbb{E}\left(v_i^2\right)$，而收敛速度主要取决于第二项。如果我们将 $\hat{v}_i=w_i-\hat{m}\left(f_i\right)$ 以及

$$y_i-\hat{g}\left(f_i\right)=\alpha\left(m\left(f_i\right)+v_i\right)+g\left(f_i\right)-\hat{g}\left(f_i\right)+\epsilon_i$$

带入到第二项，我们会发现可以将

$$\frac{1}{\sqrt{N}}\sum_{i=1}^{N}\hat{v}_i\left(y_i-\hat{g}\left(f_i\right)\right)$$

分解为如下几种成分：

1. 两个误差项的乘积，即

$$\frac{1}{\sqrt{N}}\sum_{i=1}^{N}v_i\epsilon_i\overset{a}{\sim}N\left(0,\sigma_{v\epsilon}^2\right)$$

   这一项是常见的一项；

2. 估计误差的乘积，或者正则化的偏差乘积项，即

$$\frac{1}{\sqrt{N}}\sum_{i=1}^{N}\left(\hat{g}\left(f_i\right)-g\left(f_i\right)\right)\left(\hat{m}\left(f_i\right)-m\left(f_i\right)\right)$$

   这一项是两个估计误差的乘积，即使两个成分的收敛速度都比较慢，其乘积也会更快，如果收敛速度比 $N^{-1/4}$ 更快，也会随着 $N\to\infty$ 变为无穷小项；

3. 误差项与估计误差的乘积项，即诸如

$$\frac{1}{\sqrt{N}}\sum_{i=1}^{N}v_i\left(\hat{g}\left(f_i\right)-g\left(f_i\right)\right),\frac{1}{\sqrt{N}}\sum_{i=1}^{N}\epsilon_i\left(\hat{m}\left(f_i\right)-m\left(f_i\right)\right)$$

这些项收敛速度慢，是主要的问题所在。

为了说明第三项，如果假设 $\hat{g}(f_i) = g(f_i) + \frac{\epsilon_i}{N^{1/2-\delta}}$，从而 $\hat{g}(f_i) - g(f_i) = O_p\left(N^{-1/2+\delta}\right)$，这是一个近乎于参数模型的相对比较快的收敛速度。然而如果带入到上面，我们会发现

$$\frac{1}{\sqrt{N}}\sum_{i=1}^{N} v_i\left(\hat{g}(f_i) - g(f_i)\right) = \sqrt{N}O_p\left(N^{-1/2+\delta}\right) = O_p\left(N^{\delta}\right) \to \infty$$

从而这一项会带来一定的偏差。注意这里一个重要的阻碍在于，由于估计 $\hat{g}(\cdot)$ 所使用的样本中包含了可能与 v_i 相关的信息，或者样本 i 即参与了估计 $\hat{g}(\cdot)$，又参与了对 α 的估计，从而 V_i 与 $\hat{g}(f_i) - g(f_i)$ 可能是相关的。

Chernozhukov 等（2018）提出可以使用“**交叉拟合**”(**cross-fitting**）的办法消除这一项。所谓交叉拟合，即估计冗余参数 $\hat{g}, \hat{m}$ 所用的样本与估计 α 所用的样本为不同样本，如此，我们就消除了 V_i 与 $\hat{g}(f_i) - g(f_i)$ 之间的相关性。比如，如果估计冗余参数 $\hat{g}, \hat{m}$ 所用的样本与估计 α 所用的样本是独立的，且注意到 $\mathbb{E}(v_i|f_i) = 0$，那么有

$$\begin{aligned}\mathbb{V}\left(v_i\left(\hat{g}(f_i) - g(f_i)\right)\right) &= \mathbb{E}\left[v_i^2\left(\hat{g}(f_i) - g(f_i)\right)^2\right] - \left(\mathbb{E}\left[v_i\left(\hat{g}(f_i) - g(f_i)\right)\right]\right)^2 \\ &= \mathbb{E}\left[\left(\hat{g}(f_i) - g(f_i)\right)^2 \mathbb{E}\left(v_i^2|f_i\right)\right]\end{aligned}$$

由于 $\hat{g}(f_i) - g(f_i) = o_p(1)$，从而

$$\mathbb{V}\left(\frac{1}{\sqrt{N}}\sum_{i=1}^{N} v_i\left(\hat{g}(f_i) - g(f_i)\right)\right) = \frac{\sum_{i=1}^{N}\mathbb{E}\left[\left(\hat{g}(f_i) - g(f_i)\right)^2 \mathbb{E}\left(v_i^2|f_i\right)\right]}{N} \xrightarrow{p} 0$$

从而以上第三项也会是一个 $o_p(1)$。

为了实现交叉拟合，一种简单的方法是将样本一分为二：I 和 I^c。其中，样本 I 用于估计冗余参数 $\hat{g}, \hat{m}$；然后将 I^c 中的样本带入到 $\hat{g}, \hat{m}$ 中，得到样本外残差，再使用这些样本外残差估计 α。不过，这种做法会使得我们失去大约一半的样本。解决这一问题的方法很简单，就是将 I 和 I^c 的角色互换：再使用 I^c 估计冗余参数，使用 I 估计 α 即可。

或者，我们可以将样本随机分为 S 份：$I_1, I_2, ..., I_S$，然后使用如下步骤估计 α：

1. 将 $I_s^c = I/I_s$，即第 s 份样本的补集用于估计冗余参数（如 g, m）；
2. 使用第 1 步估计的模型，在 I_s 上计算残差；
3. 重复以上步骤 S 次，计算所有样本的“样本外残差”；
4. 使用样本外残差带入正交化的矩条件估计 α。

可以证明，在一定的条件（收敛速度，比如稀疏性等）下，以上的估计是 $\sqrt{N}$ 收敛且渐进正态的。

双重机器学习可以看做是以上 Neyman 正交化方法的拓展，即使用交叉拟合的方式进一步减少偏误。因而，对于以上 Neyman 正交化的方法，我们都可以使用双重机器学习的方法进行估计。比如，以上介绍的正交化的回归方法和工具变量方法都可以扩展到双重机器学习方法。

例 1.4. 例1.2的例子中，我们可以使用交叉拟合版本的 poregress 命令；xporegress 进行交叉拟合：

```
xporegress Votes_SPS_ Watch_probit if year==1999,
    controls((i.region) `E_controls') cluster(region)
    selec(cv) xfold(15)
```

其中 xfold 选项即 S 的值。

例 1.5. 针对上例中的数据，我们也可以使用其他机器学习方法，如随机森林、梯度提升（gradient boosting）等机器学习方法。为此我们可以直接在 Stata 中调用 Python，或者也可以使用 ddml 命令[3]配合 pystacked 命令等方式进行。比如，针对以上的部分线性模型，我们可以使用如下代码：

代码 1.3: DDML 命令进行双重机器学习

```
// ntv_ddml.do
clear
set more off
use datasets/NTV_Aggregate_Data_reshaped.dta
tsset tik_id year
// 控制变量
gen lag_nurses=L.nurses
gen lag_doctors_pc=L.doctors_pc
gen lag_pop=L.logpop
gen lag_wage=L.log_wage
local controls "lag_doctors_pc lag_nurses lag_pop
    lag_wage i.region"
// DDML
ddml init partial, kfolds(10) // 初始化模型，10折的交叉拟
    合
ddml E[Y|X]: pystacked Votes_SPS_ `controls', type(
    regress) method(rf) cmdopt1(n_estimators(200))
ddml E[D|X]: pystacked Watch_probit `controls', type(
    regress) method(rf) cmdopt1(n_estimators(200))
```

[3]使用方法可以参考 https://statalasso.github.io/docs/ddml/models/

```
ddml crossfit
ddml estimate, cluster(region) // 等价于 reg
    Y1_pystacked_1 D1_pystacked_1, cluster(region)
```

其中 ddml init 用于声明模型，partial 代表部分线性模型；kfolds 为样本划分的折数；E[Y|X] 用于设定预测 y 的模型，这里使用了随机森林；E[D|X] 用于设定预测 w 的模型[4]。ddml crossfit 命令就会进行交叉拟合，并将预测的样本外残差放入到 Y1_pystacked_1 D1_pystacked_1 两个变量中；最后可以使用 ddml estimate 命令进行估计。

例 1.6. Dube 等（2020）使用双重机器学习估计了在线劳动市场（Mturk 网站）的劳动供给弹性问题，其基本回归方程为

$$\begin{aligned}\ln(duration) &= -\eta \ln(reward) + g(x) + \epsilon \\ \ln(reward) &= m(x) + v\end{aligned}$$

其中 $duration$ 为任务空缺的持续期，而 $reward$ 为任务的奖励。除了雇主、发布时间、任务时间要求等固定效应外，他们还将任务的标题、描述等使用 Doc2Vec 嵌入方法结合词袋模型等扩展了文本维度，从而 x 的维度很高。为此，他们使用随机森林对 $\ln(duration)$ 和 $\ln(reward)$ 进行了交叉拟合，再将交叉拟合的样本外残差进行回归，从而进行了双重机器学习的估计。

1.4 双重机器学习与因果推断

因果推断是计量经济学的核心问题之一，传统的文献中，我们通常关注平均处理效应（ATE）、处理组平均处理效应（ATT）等的识别和估计问题。然而，在处理效应的分析中，处理效应的异质性不仅仅在理论上可能会影响处理效应的评估，同样在现实中也是我们非常感兴趣的话题。传统的方法对于识别**异质性处理效应（heterogeneous treatment effects）**往往需要较强的假设，或者统计性质较差，而机器学习方法的引入为解决这一问题提供了有力工具。

实际上，我们以上讨论的部分线性模型可以看做是对同质性处理效应的建模，即假设数据生成过程为：

$$y_i = \tau w_i + g(x_i) + u_i$$

其中 $w = 0/1$ 为处理变量。根据该设定，其反事实为

$$\begin{aligned}y_i(0) &= g(x_i) + u_i \\ y_i(1) &= \tau + g(x_i) + u_i\end{aligned}$$

[4]为了符号统一我们这里使用了 w，文献中常用 d，仅仅是记法问题。

那么处理效应 $y_i(1) - y_i(0) = \tau$，从而平均处理效应与处理组平均处理效应相等，都为 τ。在无混淆分配的假设，即 $w_i \amalg (y_i(0), y_i(1)) | x_i$ 的条件下，可以自然得到 $u_i \amalg w_i | x_i$，从而我们可以使用双重机器学习的方法对以上问题进行估计，注意到如果设定

$$w_i = m(x_i) + v_i$$

那么 $m(x_i)$ 即为 $P(w_i = 1|x_i)$，或者倾向得分函数。此外，针对异质性处理效应 CATE 的估计，Foster 和 Syrgkanis（2019）继续在以上基础上假设

$$y_i = \tau(x_i) w_i + g(x_i) + u_i$$

其中 $\tau(x_i)$ 为处理效应的异质性估计。Python 中的 EconML 包实现了这一算法。

然而以上模型对于处理效应同质性的假设仍然太强。为此我们不妨暂时放弃函数形式假设，先从识别层面探讨可能的异质性处理效应的识别问题。为此我们现引入如下假设：

假设 1.1. 无混淆分配假设：$w_i \amalg (y_i(0), y_i(1)) | x_i$

假设 1.2. 共同支撑假设：对于任意的 $x_i = x$，倾向得分 $p(x) \triangleq P(w_i = w | x_i = x) > 0, w = 0/1$

假设 1.3. *SUTVA* 假设：$Y_i = Y_i(W_i)$

如此，如果记 $g(w, x) = \mathbb{E}(y_i | w_i = w, x_i = x)$，那么：

$$\begin{aligned} g(w, x) &= \mathbb{E}(y_i | w_i = w, x_i = x) \\ &= \mathbb{E}(y_i(w) | w_i = w, x_i = x) \\ &= \mathbb{E}(y_i(w) | x_i = x) \end{aligned}$$

以上等式意味着为了获得 $g(w, x)$ 的估计，我们可以使用根据 w 分组，使用 y 对 x 做回归分别得到 $g(0, x)$ 以及 $g(1, x)$，将两者相减就可以得到

$$g(1, x) - g(0, x) = \mathbb{E}(y_i(1) - y_i(0) | x_i = x) = CATE(x)$$

其中 $CATE(x)$ 为给定特征 x 的平均处理效应，或者**条件平均处理效应（conditional average treatment effects）**。Chernozhukov 等（2017）即考虑了这种数据生成过程：

$$\begin{aligned} y &= g(w, x) + u \\ w &= m(x) + v \end{aligned}$$

其中无混淆分配假设即 $\mathbb{E}(u|w, x) = 0$，该设定实际上允许处理变量 w 与 x 之间任意的交互，或者说处理效应可以是 x 的任意函数，从而以上模型也被称为

“交互模型”(interactive model)。如果需要进一步得到 ATE 和 ATT，可以计算

$$ATE = \mathbb{E}\left[g\left(1,x\right) - g\left(0,x\right)\right]$$
$$ATT = \mathbb{E}\left[g\left(1,x\right) - g\left(0,x\right)|w=1\right]$$

进一步，我们也可以将以上的识别方法写成逆概率加权的形式：

$$\begin{aligned}\mathbb{E}\left(\frac{w_i y_i}{p\left(x_i\right)}|x_i = x\right) &= \frac{\mathbb{E}\left(w_i y_i|x_i = x\right)}{p\left(x\right)} \\ &= \frac{\mathbb{E}\left(w_i y_i\left(1\right)|x_i = x\right)}{p\left(x\right)} \\ &= \frac{\mathbb{E}\left(w_i|x_i = x\right)\mathbb{E}\left(y_i\left(1\right)|x_i = x\right)}{p\left(x\right)} \\ &= \mathbb{E}\left(y_i\left(1\right)|x_i = x\right)\end{aligned}$$

类似的，

$$\mathbb{E}\left[\frac{\left(1-w_i\right)y_i}{1-p\left(x_i\right)}|x_i = x\right] = \mathbb{E}\left(y_i\left(0\right)|x_i = x\right)$$

从而条件平均处理效应可以使用

$$CATE\left(x\right) = \mathbb{E}\left[\frac{w_i y_i}{p\left(x_i\right)} - \frac{\left(1-w_i\right)y_i}{1-p\left(x_i\right)}|x_i = x\right]$$

进行识别。

而更进一步，我们可以证明

$$\begin{aligned}\mathbb{E}\left(y_i\left(1\right)|x_i = x\right) &= \mathbb{E}\left[g\left(1,x_i\right) + \frac{w_i\left(y_i - g\left(1,x_i\right)\right)}{p\left(x_i\right)}|x_i = x\right] \\ \mathbb{E}\left(y_i\left(0\right)|x_i = x\right) &= \mathbb{E}\left[g\left(0,x_i\right) + \frac{\left(1-w_i\right)\left(y_i - g\left(0,x_i\right)\right)}{1-p\left(x_i\right)}|x_i = x\right]\end{aligned} \tag{1.7}$$

以上方程虽然看起来是非常多余的，因为根据以上的讨论，单独的 $g\left(1,x\right)$ 或者 $\frac{w_i y_i}{p(x_i)}$ 就可以识别条件平均处理效应，而这么做看起来多此一举。然而实际上，这样做有其好处：在实证中，$g\left(\cdot,\cdot\right)$ 函数和 $p\left(x_i\right)$ 都需要通过模型进行估计，从而都可能会存在设定错误或者偏误。而这一方法的好处在于其是“**双向稳健**”(**doubly robust**) 的，即只要 $g\left(\cdot,\cdot\right)$ 函数和 $p\left(x_i\right)$ 只要有一个估计准确，那么以上等式即成立。

基于此，Chernozhukov（2017）提出可以通过定义

$$\psi\left(\tau,\eta;y_i,w_i,x_i\right) = g\left(1,x_i\right) - g\left(0,x_i\right) + \frac{w_i\left(y - g\left(1,x_i\right)\right)}{p\left(x_i\right)} - \frac{\left(1-w_i\right)\left(y - g\left(0,x_i\right)\right)}{1-p\left(x_i\right)} - \tau$$

并使用矩条件 $\mathbb{E}\psi\left(\tau,\eta;y_i,w_i,x_i\right) = 0$ 求解出平均处理效应 τ，其中 $\eta = \left(g\left(1,x\right),g\left(0,x\right),p\left(x\right)\right)$

为冗余参数。可以证明以上的矩条件是满足 Neyman 正交化条件的，即

$$\partial_\eta \mathbb{E}\psi\left(\tau,\eta;y,w,x\right)|_{\eta=\eta_0}=0 \tag{1.8}$$

从而我们可以使用双重机器学习的方法对 τ 进行估计。

例 1.7. 如上方法可以使用 ddml 命令直接进行估计，比如对于 NSW 项目的政策评估：

代码 1.4: DDML 命令估计平均处理效应

```
// ddml_nsw.do
clear
set more off
use datasets/Lalonde_nsw/nsw.dta
// append using datasets/Lalonde_nsw/cps_controls.dta
local controls "age-nodegree"
ddml init interactive, kfolds(10)
ddml E[Y|X,D]: pystacked re78 'controls', type(regress)
    method(rf) cmdopt1(n_estimators(200))
ddml E[D|X]: pystacked treat 'controls', type(class)
    method(rf) cmdopt1(n_estimators(200))
ddml crossfit
ddml estimate, robust
```

或者也可以对处理组平均处理效应进行评估。根据 Farrell（2015），有

$$\begin{aligned}\mathbb{E}\left(y_i\left(1\right)|w_i=1\right)&=\mathbb{E}\left[\frac{w_i y_i}{p_1}\right]\\ \mathbb{E}\left(y_i\left(0\right)|w_i=1\right)&=\mathbb{E}\left[\frac{w_i g\left(0,x_i\right)}{p_1}+\frac{p\left(x_i\right)}{p_1}\frac{\left(1-w_i\right)\left(y_i-g\left(0,x_i\right)\right)}{1-p\left(x_i\right)}\right]\end{aligned}$$

其中 p_1 为处理组样本量占总样本量的比例，从而处理组平均处理效应：

$$\mathbb{E}\left(y_i\left(1\right)-y_i\left(0\right)|w_i=1\right)=\mathbb{E}\left[\frac{w_i\left(y_i-g\left(0,x_i\right)\right)}{p_1}-\frac{p\left(x_i\right)}{p_1}\frac{\left(1-w_i\right)\left(y_i-g\left(0,x_i\right)\right)}{1-p\left(x_i\right)}\right]$$

Chernozhukov（2017）提出可以根据以上条件，定义

$$\psi\left(\tau,\eta;y_i,w_i,x_i\right)=\frac{w_i\left(y_i-g\left(0,x_i\right)\right)}{p_1}-\frac{p\left(x_i\right)}{p_1}\frac{\left(1-w_i\right)\left(y_i-g\left(0,x_i\right)\right)}{1-p\left(x_i\right)}-\tau\frac{w_i}{p_1}$$

其中 $\eta=\left(g\left(0,x\right),p\left(x\right),p_1\right)$ 为冗余参数。同样，以上矩条件满足 Neyman 正交化条件，从而可以使用双重机器学习的方法对其进行估计。Knaus（2022）对以上方法提供了一个综述。

1.5 因果树方法

我们知道，回归树通过不断划分特征空间的方法可以实现对 y 的预测或者对 w 的分类，这是一种“非参数”的方法：我们无需假设 y 或者 w 的数据生成过程，甚至也无需设置特征对 y 的影响的异质性，分类和回归树本身就可以发现这些“交互”作用。那么是否可以将分类和回归树的方法直接用于处理效应的估计呢，从而解决异质性处理效应的估计问题呢?

比如，我们可以构造一颗预测 y 的回归树，假设叶子结点的集合为 $\Pi = \{\ell_1, ..., \ell_L\}$，我们可以仿照以上逆概率加权的方法，使用

$$\hat{\tau}_\ell = \frac{1}{\#\ell} \sum_{i \in \ell} \left[\frac{w_i y_i}{p(x_i)} - \frac{(1 - w_i) y_i}{1 - p(x_i)} \right]$$

对 CATE 进行估计。如此，我们无需对异质性做任何假设，而是使用数据驱动的方法对 CATE，或者异质性的处理效应进行了建模。

然而以上方法存在很严重的问题。为了构建回归树，我们使用了逐步增加节点的纯度的方法，而纯度的增加就是节点内 y 的相似程度增加，而以上的估计本质上是在节点内对处理组和对照组的 y 进行加权比较，按照这样的算法，以上的比较结果将会是被低估的，从而引入了偏差。

Athey 和 Imbens（2016）提出了适应因果推断的“**因果树**”（**causal tree**），使用了一种“诚实方法”（honest approach）构建树，从而达到对异质性处理效应的识别。该方法的一个重要改进是将样本分为了两部分：训练集 I^{tr} 和估计集 I^{est}，当然，为了选择超参数还可以多分出一个验证集。重要的在于，在构建树的过程中仅仅使用训练集，而估计政策效应则使用估计集数据，因而这种树的构造方法也被称为“**双样本树**”（**double-sample tree**）。此外，为了适应政策效应的评估，Athey 和 Imbens（2016）还改进了构造树时的准则，使用期望的 MSE 作为准则估计，从而可以更好的估计政策效应。

随后，Wager 和 Athey（2019）进一步将因果树推广为**因果森林**（**causal forest**）。他们提出了两种不同的方法：

1. 双样本树，即使用 Athey 和 Imbens（2016）的方法，每次从样本中进行有放回抽样，在每个 bootstrap 样本中都进行双样本树的建模，最后将每颗树得到的 CATE 平均得到最终的估计；

2. 倾向得分树：不使用双样本的方法，而是直接以 w_i 作为训练目标构建树。

Wager 和 Athey（2019）证明以上因果森林的估计结果是渐进正态的，且其渐进方差可以方便的估计出。

随后，Athey 等（2019）又对以上方法做了更多拓展，提出了“**广义随机森林**”（**generalized random forest**）以解决更多的参数估计问题，这里不再赘述。广义随机森林在 R 语言中可以使用 grf 包进行估计，在 Python 中也可以使用 EconML 包进行分析。

习题

练习 1.1. 证明如果式 (1.7) 中的 $p(x_i)$ 如果正确，那么任意的 $g(1,x)$ 都使得等式成立。

练习 1.2. 证明如果式 (1.7) 中的 $g(1,x)$ 如果正确，那么任意的 $p(x_i) > 0$ 都使得等式成立。

练习 1.3. 证明式 (1.8)