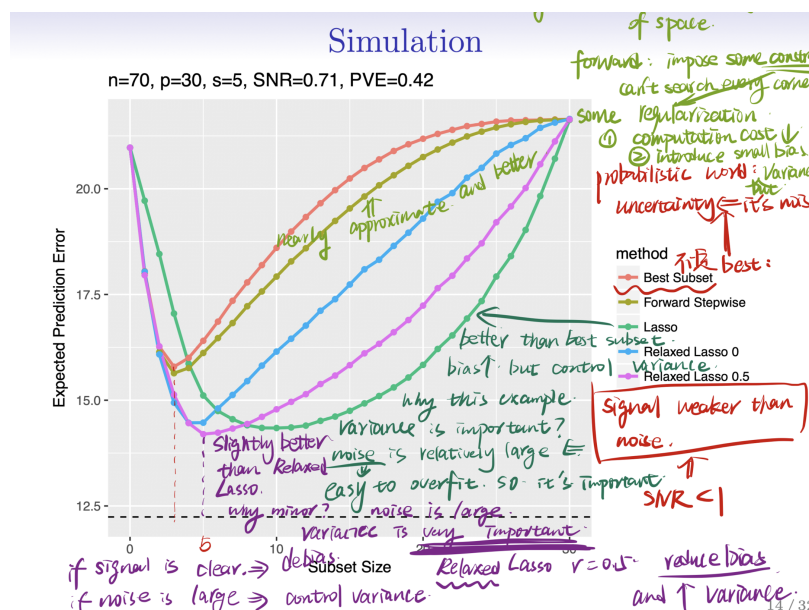


Regularization

- 我们在选择有用的variables的时候，需要遍历所有的subset，从而选择一个最好的模型，那怎么判断哪个模型更好呢？
 - AIC: $-2 \log \text{likelihood}(\theta) + 2d$ (d is the number of parameters)
 - BIC: $-2 \log \text{likelihood}(\theta) + 2 \log(n) * d$ (n is number of data)
 - BIC put more penalty to the number of parameters or degree of freedom
 - which one to use?
 - if choose a model for better prediction-->AIC 更好
 - if $n \rightarrow \infty$, BIC 更好, AIC判断的模型的df更大
 - In practice, cross validation is preferred.
 - forward
 - backward: take more time and not necessary
 - 因为我们的CV，假设我们把整个数据分成3份，123
 - 把1当作test data，23当作train data，得到最好的model, M_{21}
 - 把2当作test data，13当作train data，得到最好的model, M_{22}
 - 把3当作test data，12当作train data，得到最好的model, M_{23}
 - 这三个模型虽然df一样，但选择的subset可能不一样，那我们怎么得到最后的最好的model呢？到底选哪个呢？
 - 注意，这是我们cross validation的目标，cross validation并不在意到底是哪两个variable，而在意degree of freedom
 - 我们推广到general 的情形，每一个test data都可以获得一条test error的曲线，这样我们可以获得k个曲线，将这k个曲线做平均，选择一个平均test error最小的df，然后用整个data（不分123）来选择最好的subset variable
 - conclusion
 - forward一般比best subset快，且两者效果非常逼近
 - Lasso分为relaxed lasso和普通lasso，relaxed lasso效果最好
 - wide data ($n < p$) + low signal noise ratio，也就是noise影响很大，用lasso效果最好
 - Related Lasso：注意，lasso由于将很多beta shrinkage，也就是提高了这个model的bias啊，而想办法降低variance，从而达到trade off，那有没有可能我们再进一步降低bias呢，什么方法降低bias？OLS
 - size of subsize--test error
 - 1 best subset error 最高，由于我们search了所有可能的解空间，非常有可能导致过拟合，我们是probability world，是有一些uncertainty的，所以best subset不一定是最好的

- 2 forward, put some shrinkage to 解空间, 可以认为是某种regularization, 两个好处, 一个是降低了计算成本, 一个是可以适当的增加bias然后大幅降低variance
- 3 relaxed lasso 0, relaxed lasso 0.5, lasso
 - 谁好谁坏要看数据, 如果数据的signal noise很小, 说明noise非常重要, lasso是最能降低variance的方法, relaxed lasso 可能只比lasso好一点点, 但如果signal很强, 这个时候更需要做的是减少bias, variance就不是那么重要了

- plot



- size of subset--degree of freedom
 - 根据stein's lemma:(为什么有这个公式? 方便计算df)

Stein's Lemma: if X, Y normally distributed.

$$\text{cov}(g(X), Y) = \text{cov}(X, Y) E[g'(X)]$$

$$\begin{aligned} df &= \sum E \left[\frac{\partial \hat{y}_i}{\partial y_i} \right] \\ &= \sum \frac{\text{cov}(\hat{y}_i, y_i)}{\text{cov}(y_i, y_i)} \\ &= \sum \frac{1}{\sigma^2} \text{cov}(\hat{y}_i, y_i) \end{aligned}$$

- 如何计算df?

$$\begin{aligned}
\text{cov}(\hat{y}_i, y_i) &= E[(\hat{y}_i - f(\mathbf{x}_i))(y_i - f(\mathbf{x}_i))] \in y_i = f(\mathbf{x}_i) + \epsilon_i \\
&= E[(\hat{y}_i - f(\mathbf{x}_i)) \epsilon_i] \\
&= E[\underbrace{\hat{y}_i \epsilon_i - f(\mathbf{x}_i) \epsilon_i}_{\text{independent}}] \\
&= E[\hat{y}_i \epsilon_i] \\
&= \frac{1}{n} \sum_i (\hat{y}_i \epsilon_i) \Downarrow
\end{aligned}$$

- 为什么 \hat{y}_i 和 ϵ_i 不独立？因为残差是有可能拟合进 \hat{y}_i
- 结论：df不仅仅和size of subset 有关，和search也有关，简单来说，如果size of subset=10，在一个本身就30个variable时，不进行搜索，df=10，但如果是搜索最好的10个variable，df>10
 - Lasso几乎是直线，因为他的regularization会令df减少（因为很多beta为0），但是search会令df增加，magic的事情是，两者抵消了
 - 越是best subset，search的越多，df越大

• SNR--test error

- 首先，SNR较低时，Noise很大，这个时候要降低variance，Lasso更好，SNR很高时，noise不重要，要降低bias，因此可能best subset也不错
- Relaxed Lasso is the best winner
- 其实，Ridge Regression 也很稳定，哪怕noise很大，甚至可能比Lasso更好，但由于这里讲的是variable selection，而Ridge是不能selection的，因此这里没有比较Ridge Regression

• regularization

- 有两种：一种是explicit form, 我们是真的能在表达式看到 λ 的，一种是implicit的，不是说没有，而是看上去没有，但实际上是蕴含着 λ 的。比如drop out：真正算出来和Ridge 是一模一样的
- Q: 为什么machine learning用了更多的parameter，但并不容易overfit?
- A: 这是因为比如LDA和QDA都是有closed form的，也就是他是直接跳到optimal的，但我们的machine learning是通过迭代一点一点达到iteration的，所以不容易overfit
- Q: 这种梯度下降的方法怎么确定就有regularization呢
- A: implicit regularization
- Q: 为什么我们需要regularization呢？因为我们更重要的不是训练针对唯一数据集的function，而是不同的data，所以我们需要添加一个期望，同时，如果我们正常训练，我们的bias确实可能是0，但是方差可能会很大，所以我们需要添加一些额外的条件，比如，如果我们添加的条件是 $\|\beta\|^2 \leq 1$ ，这样我们的方差就限制在1内了，这样也许可以实现一个bias-variance tradeoff
- Ridge Regression
 - 有三种形式：有截距项，无截距项和标准化
 - 注意

- 1. 如果做Ridge Regression, 非常推荐做标准化, 因为, Ridge中我们存在对 β 的惩罚项, 这个惩罚项的假设就是每个 β 都是同一个scale, 否则结果不准确, 因此非常推荐Ridge Regression 做标准化 (之前的最小二乘法是不会受量纲影响的)
- 2. 另外, 由于我们进行回归之后得到的 β 是针对标准化数据得到的 β , 且 X_{test} 也是标准化之后的, 因此在得到结果之后还需要将 β 转化为原来的scale

$$\begin{aligned}\hat{y}_i &= \hat{\beta}_0 + \sum_j \tilde{x}_{ij} \hat{\beta}_j \\ y_i - \bar{y} &= \hat{\beta}_0 + \sum_j \frac{x_{ij} - \mu_j}{\sigma_j} \hat{\beta}_j \\ y_i &= \left[\hat{\beta}_0 + \bar{y} - \sum_j \frac{\mu_j \hat{\beta}_j}{\sigma_j} \right] + \sum_j x_{ij} \left[\frac{\hat{\beta}_j}{\sigma_j} \right]\end{aligned}$$

- 3. 如何决定 λ 呢? cross validation
- 4. 但是! 太多的 λ , 如何提高计算效率呢?

我们来观察 $\beta = (X^T X + \lambda I)^{-1} X^T Y$, 最耗时间的见手逆 $O(p^3)$

trick: $X = UDV^T$, $U = \begin{pmatrix} \text{ } \end{pmatrix}^{p \times p}$, $D = \begin{pmatrix} \lambda_1 & \dots & \lambda_p \end{pmatrix}$ 特征值分解
 U 的性质 (U 是正交的特征向量)

① $\lambda_1, \dots, \lambda_p$ 非负 (因为 $X^T X$ non-negative)

② $U^T U = I$ $u_i \perp u_j$ $u_i^T = u_i^T$ (when $i=j$)

$$\begin{aligned}\beta &= (UDV^T + \lambda I)^{-1} X^T Y = (U(D + \lambda I)U^T)^{-1} X^T Y \\ &= U(D + \lambda I)^{-1} U^T X^T Y \\ &= U \begin{pmatrix} \frac{1}{\lambda_1 + \lambda} & & \\ & \ddots & \\ & & \frac{1}{\lambda_p + \lambda} \end{pmatrix} U^T X^T Y\end{aligned}$$

由于 $X^T X$ 是对称元素在副数, 因此 λ 是实数且有 $O(p)$ 且 λ 是 $X^T X$ 的特征值分解

- 另外, 对于 λ 的取值范围, 当 λ 大于最大的特征值时, 对角线元素会非常小, 因此我们可以设置范围的最大值为最大的特征值
- 5. 在进行cross validation时, 比如我们用的 $k=3$, 如果我们用 $\sum (y_i - \hat{y}_i)^2 + \lambda \beta^2$, 注意这里只用了2/3的数据, 并没有用全部的数据, 所以如果我们最后cv之后选择 λ 时, 必须对 λ 进行rescale, 在这里就是编程3/2, 如果本身我们的loss function就是取平均, 那就不需要rescale。

• Lasso Regression

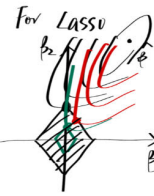
- Least Absolute Shrinkage: 为什么我们说是shrinkage? (注意shrinkage和regularization不同, 前者是放缩, 特征缩减, 也就是我们选择重要因子, 抛弃不太重要因子, 后者是正则化, 在损失函数中加入惩罚项, 缩减解空间, 从而增加模型的稳定性和防止过拟合)

Another formulation

Ridge Reg. $\min \sum (y_i - \beta_0 - \sum \beta_j x_{ij})^2$ subject to $\sum \beta_j^2 \leq s$

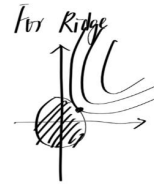
Lasso Reg. $\min \sum (y_i - \beta_0 - \sum \beta_j x_{ij})^2$ subject to $\sum |\beta_j| \leq s$

W Reg. $\min \sum (y_i - \beta_0 - \sum \beta_j x_{ij})^2$ subject to $\sum I(\beta_j \neq 0) \leq s$



可能的解空间
和s空间在y轴
上的点,此时 $\beta_j=0$.
所有 β_j 都不在 β_j .
→ variable selection

有可能存在多个情况
也是是个tuning parameter.
是可以自由变动的
SL是可以使得关于y轴的

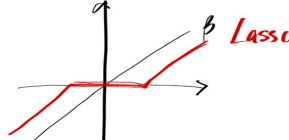
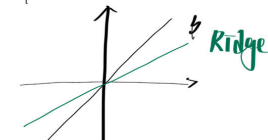


注意, 我们最后的解必须在可能范围内, 找到最靠外的
椭圆和圆心的点. 此时 $\sum (y_i - \beta_0 - \sum \beta_j x_{ij})^2$ 是椭圆
找的是整个曲线最小的
c min. 找相切点.
c=0. → 找 β_j 的相切点 β_j
是不会和圆相切. 也就是Ridge是没有variable selection

shrinkage

同样的, 如果我们计算 Ridge 的 soft shrinkage, $\frac{1}{2}(\beta - \beta)^2 + \lambda \beta^2$

$$\frac{\partial}{\partial \beta} = (\beta - \beta)(1) + \lambda \beta = 0 \quad \beta - \beta + \lambda \beta = 0 \quad \beta = \frac{\beta}{1 + \lambda}$$



可以看到 Lasso 中的 β 会有部分直接降为0. → variable selection

- Lasso由于是non-differentiable 的损失函数, 是没有close-form的解的, 我们现在的解是用 coordinate descent 的方法来迭代求解的

- 首先, 如果有一个convex+differentiable (differentiable保证任何局部最优解都是全局最优解) 的函数, 能否得到一个全局最优解? 可以, 直接对每个自变量求导使得每个分量的导数等于0即可
- 如果是non-differentiable 的函数呢? 不一定, 例如下图, 每一个闭合曲线上的所有点都是同一个值, 越靠近, value越小, 因为我们希望越往中间靠近, 但假设我们在尖锐点处, 由于我们是坐标下降, 不管是x轴下降还是y轴下降, 我们都是不能降低value的, 所以就在这里卡死了, 但是这里明显不是全局最优解, 那怎么解决呢?
- 如果 $f(x) = g(x) + \sum h_i(x_i)$, 在这里, $g(x)$ 是一个convex+differentiable 的函数, $h_i(x_i)$ 是一个可加函数 (separable or additive function), 这时我们可以根据坐标下降法获得全局最优解, why?

$$f(y) - f(x) \geq \nabla g(x)^T (y - x) + \sum h_i(y_i) - h_i(x_i)$$

$$= \sum_{i=1}^n [\nabla_i g(x) (y_i - x_i) + h_i(y_i) - h_i(x_i)]$$

次梯度法的最优性条件 $0 \in \nabla_i g(x_i) + \partial h_i(x_i)$

$$\left. \begin{aligned} \nabla_i g(x_i) + \frac{h_i(y_i) - h_i(x_i)}{y_i - x_i} \\ \nabla_i g(x_i) (y_i - x_i) + h_i(y_i) - h_i(x_i) \geq 0 \end{aligned} \right\}$$

- 思路: $H_0 = x_1^1, x_2^1$, 第一次固定 x_2^1 , 只变化 $x_1^1 \rightarrow x_1^2$, 得到 $H_0 = x_1^2, x_2^1$, 再只动 $x_2^1 \rightarrow x_2^2$, 直到 converge, 怎么动?
- 但是, 太多 lambda 了, 怎么计算呢? 和 Ridge 一样, Ridge 是给 lambda 规定了一个范围, 但是 Lasso 没有, 因为 Lasso 甚至不是 differentiable 的, 没办法特征值分解, 同样, 我们来看 Lasso 的时间空间复杂度在哪里, 是矩阵乘一个向量, 这个算法会让代码运行十分缓慢, 怎么解决?
 - 1. track \hat{y} , 从而减少矩阵运算: 注意我们之前计算的每一个都是只针对 β_j , 因此每次迭代, 都要对 p 个 β_j 进行循环寻找 β_{new} , 大循环是对迭代次数的循环, 在中间, 每一次都对每一个 β_j 进行覆盖, 只覆盖一次, 直到符合精度要求了, 结束大的迭代循环

$$\begin{aligned}
 \hat{y}_i &= \sum_{j=1}^p x_{ij} \beta_j = y_i^{(1)} + x_{ij} \beta_j \\
 \text{if we have } \hat{y}_i, \quad y_i^{(1)} &= \hat{y}_i - x_{ij} \beta_j \\
 &\text{因为直接数值计算, 不用计算矩阵} \\
 ① \Rightarrow S(\sum_{j=1}^p x_{ij} (y_i - \hat{y}_i + x_{ij} \beta_j), \lambda) \\
 \beta_j &= S(\underbrace{\sum_{j=1}^p x_{ij} (y_i - \hat{y}_i)}_{\substack{\uparrow \text{Residuals} \\ \text{只计算内积}}}, \lambda) \\
 ② \hat{y}_i^{old} &= \sum_{k=1}^p x_{ik} \beta_k + x_{ij} \beta_j^{old} \\
 \hat{y}_i^{new} &= \sum_{k=1}^p x_{ik} \beta_k + x_{ij} \beta_j^{new} \\
 \hat{y}_i^{old} - \hat{y}_i^{new} &= x_{ij} (\beta_j^{old} - \beta_j^{new}) \Delta_j \\
 \hat{y}_i^{new} &= \hat{y}_i^{old} - x_{ij} \Delta_j \\
 \text{So if keep tracking } \hat{y}_i &\Rightarrow y_i^{(1)} \Rightarrow \beta^{new}
 \end{aligned}$$

- cons: Lasso 其实应该是稀疏的, 但现在我们每次都进行更新, 浪费时间, 这个算法还可以 improve
- 2. active set: 因为我们知道他稀疏, 比如 1000 个变量, 只有 10 个变量不是 0, 那我们怎么样可以只更新这 10 个 beta 呢?
 - 双重循环: outer loop: 先通篇检查 beta 哪些是 0, 把不是 0 的存为 active set, 然后对 active set 中的 beta 进行更新, 直到 converge, 在再回去检查, 看 active set 是否变了, 若变了, 重新更新, 否则直接输出
 - mm, m, nin 的初始值均为 0, 如果 beta 要更新, 则把他认为是 active set 中的一个, mm 存 size of active set, m 存 active set 的 index
 - 这个算法的复杂度也是和初始值有关系的, 如果 true=2, 那我从 0 开始搜索和从 100 开始搜索, 复杂度肯定不一样, 那怎么继续减少复杂度呢? --> warm start

- 3. warm start: 我们已知当lambda趋于无穷时, beta都等于0, 那么我们令 lambda从大到小递减, 每次只减少一点, 这样可以允许每次只增加一个不等于0的beta, 也就减少了active set的数量, 因此复杂度大大减少
- Relaxed Lasso

Best subset selection

$$\min_{\beta} \frac{1}{2} \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j)^2$$

subject to $-Mz_j \leq \beta_j \leq Mz_j, z_j = 0, 1$

$\sum_{j=1}^p z_j \leq k$ 将非零的p个数据列在p个之内

⇒ put some constraints to best subset.

Relaxed Lasso:

$$\hat{\beta}_{\lambda}^{\text{Relax}}(\gamma) = \gamma \hat{\beta}_{\lambda} + (1-\gamma) \hat{\beta}_{\lambda}^{\text{OLS}}$$

$\gamma=0$ 时 $\hat{\beta}_{\lambda}^{\text{Relax}} = \hat{\beta}^{\text{OLS}}$ 根据OLS. $\hat{\beta}^{\text{OLS}} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} \leftarrow \text{OLS}$

$\gamma=1$ 时 $\hat{\beta}_{\lambda}^{\text{Relax}} = \gamma \hat{\beta}_{\lambda}$ 是Lasso