regularization

Regularization

我们在选择用的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 loglikelihood($\hat{\theta}$) +2d (d is the number of parameters)

BIC: -2 loglikelihood($\hat{\theta}$) +2log(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-->∞ ， 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

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的比例说，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

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

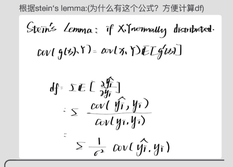
谁好谁坏要看数据，如果数据的signal noise很小，说明noise非常重要，lasso是最能降低variance的方法，relaxed lasso 可能只比lasso好一点点，但如果signal很强，这个时候更需要做的是减少bias，variance就不是那么重要了

plot

size of subsize--test error

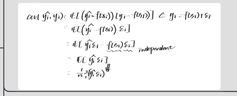
conclusion

根据stein's lemma:(为什么有这个公式？ 方便计算df)



size of subset--degree of freedom

如何计算df?



为什么$\hat{y}_1$和ϵ_1 不独立？ 因为残差是有可能拟合进$\hat{y}_1$

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

Lasso Regression

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dropout：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

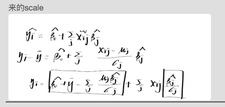
Q:为什么我们需要regularization呢？ 因为我们更重要的不是训练针对唯一数据集的function，而是不同的data，所以我们需要添加一个期望，同时，如果我们正常训练，我们的bias确实可能是0，但是方差可能会很大，所以我们需要添加一些额外的条件，比如，如果我们添加的条件是$||\beta||^2 \leq 1$，这样我们的方差就限制在1内了，这样也许可以实现一个bias-variance tradeoff

Ridge Regression

有三种形式：有截距项，无截距项和标准化

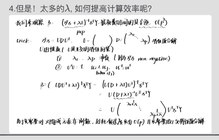
1. 如果做Ridge Regression，非常推荐做标准化，因为，Ridge中我们存在对β的惩罚项，这个惩罚项的假设就是每个β 都是同一个scale，否则结果不准确，因此非常推荐Ridge Regression 做标准化（之前的最小二乘法是不会受量纲影响的）

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



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？

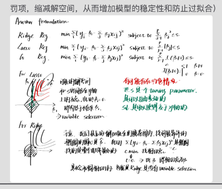


另外，对于lambda的取值范围，当lambda大于最大的特征值时，对角线元素会非常小，因此我们可以设置范围的最大值为最大的特征值

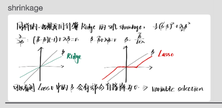
5. 在进行cross validation时，比如我们用的k=3， 如果我们用$\sum(y_i - \hat{y}_i)^2 + \lambda\beta^2$，注意这里只用了2/3 的数据，并没有用全部的数据，所以如果我们最后cv之后选择λ 时，必须对lambda进行rescale，在这里就是编程3/2， 如果本身我们的loss function就是取平均，那就不需要rescale。

注意

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）



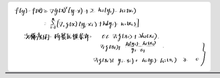
shrinkage



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-differentable 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？

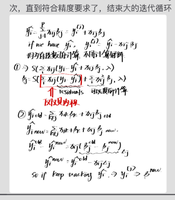
如果$f(x) = g(x) + \sum h_i(x_i)$,在这里，$g(x)$ 是一个convex+differentiable 的函数，$h_i(x_i)$ 是一个可加函数（separable or additive function),这时我们可以根据坐标下降法获得全局最优解，why?



思路：$H_0 = x_1^1, x_2^1$, 第一次固定x_2^1, 只变化$x_1^1 -> x_1^2$, 得到$H_0 = x_1^2, x_2^1$, 再只动$x_2^1 -> x_2^2$,直到converge，怎么动？

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 desent的方法来迭代求解的

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



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

双重循环：outer loop：先遍篇检查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

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？

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

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？

Relaxed Lasso