第 4 章

线性模型中的高级特征选择技术

4

“我发现数学变得过于抽象，已经不合我意了，而计算机科学变得过于关心细枝末节——试图在一次计算中存储几毫秒和几千字节。在使用统计学解决实际问题的过程中，我发现了数学和计算机科学的结合之美。”

以上引自斯坦福大学罗伯特·提布施瓦尼教授，参见http://statweb.stanford.edu/~tibs/research_page.html。

迄今为止，我们研究了使用线性模型预测定量和定性结果的方法，着重讨论了特征选择技术，也就是如何排除无用的或不需要的预测变量。我们看到，线性模型在机器学习问题中是非常有效的。但在过去的二十多年中，人们已经开发和提炼了更新的技术，它们提供的预测能力和解释性已经远远超过了我们在前面章节中讨论过的线性模型。当今很多数据集具有数量庞大的特征，即使与观测值的数量相比也毫不逊色，这正如人们所称——高维性。如果你曾经参与过基因组的研究，那么这个问题就不证自明了。此外，随着我们要处理的数据规模不断增大，最优子集和逐步特征选择这样的技术会造成难以承受的时间成本——即使使用高速计算机。我说的可不是以分钟计的时间，在很多情况下，要得到一个最优子集的解需要花费数小时。

这种情况下有更好的解决方式。本章会讨论正则化的概念，正则化会对系数进行限制，甚至将其缩减到0。现在有很多种方法和方法组合可以实现正则化，我们会集中讨论**岭回归**和**最小化绝对收缩和选择算子**，最后讨论**弹性网络**，它将前面两种算法的优点合二为一。

4.1 正则化简介 regulation

你应该还记得我们的线性模型形式为$Y = B_0 + B_1x_1 + \cdots + B_nx_n + e$，还有最佳拟合试图最小化RSS。RSS是实际值减去估计值的差的平方和，可以表示为$e_1^2+e_2^2+\cdots+e_n^2$。

通过正则化，我们会在RSS的最小化过程中加入一个新项，称之为**收缩惩罚**。这个惩罚项包

含了一个希腊字母λ以及对β系数和权重的规范化结果。不同的技术对权重的规范化方法都不尽相同，我们随后都要进行相应的讨论。简言之，我们的模型中需要最小化，也就是RSS + λ（规范化后的系数）。我们会对λ进行选择，在模型构建过程中，λ被称为调优参数。请记住，如果$\lambda = 0$，模型就等价于OLS，因为规范化项目都被抵消了。

那么，正则化对我们来说意味着什么？正则化为什么会起作用？首先1正则化方法在计算上非常有效。如果使用最优子集法，我们需要在一个大数据集上测试2^p个模型，这肯定是不可行的。如果使用正则化方法，对于每个λ值，我们只需拟合一个模型，因此效率会有极大提升。另一个理由是我们在前言讨论过的偏差–方差权衡问题。在线性模型中，响应变量和预测变量之间的关系接近于线性，最小二乘估计接近于无偏，但可能有很高的方差。这意味着，训练集中的微小变动会导致最小二乘系数估计结果的巨大变动（James，2013）2正则化通过恰当地选择λ和规范化，可以使偏差–方差权衡达到最优，从而提高模型拟合的效果。最后3系数的正则化还可以用来解决多重共线性的问题。

4.1.1 岭回归 ridge regression

我们先研究什么是岭回归，以及它可以做什么和不能做什么。在岭回归中，规范化项是所有系数的平方和，称为**L2-norm（L2范数）**。在我们的模型中就是试图最小化RSS + $\lambda(\text{sum}\beta_{\text{权}}^2)$。当$\lambda$增加时，系数会缩小，趋向于0但永远不会为0。岭回归的优点是可以提高预测准确度，但因为它不能使任何一个特征的系数为0，所以在模型解释性上会有些问题，你需要多费一些唇舌。为了解决这个问题，我们使用LASSO。缺点: 不能进行特征选择

4.1.2 LASSO

区别于岭回归中的L2-norm，LASSO使用L1-norm，即所有特征权重的绝对值之和，也就是要最小化RSS + $\lambda(\text{sum}|\beta_{\text{权}}|)$。这个收缩惩罚项确实可以使特征权重收缩到0，相对于岭回归，这是一个明显的优势，因为可以极大地提高模型的解释性。

L1-norm为什么能够使权重（或者系数）变为0？其中的数学解释已经超过了本书范围（Tibsharini，1996）。

如果LASSO这么好，那还要岭回归做什么？话别说得这么急！存在高共线性或高度两两相关的情况下，LASSO可能会将某个预测特征强制删除，这会损失模型的预测能力。举例来说，如果特征A和B都应该存在于模型之中，那么LASSO可能会将其中一个的系数缩减到0。下面的引言非常好地总结了这个问题：

> “如果较少数目的预测变量有实际系数，其余预测变量的系数要么非常小，要么为0，那么在这样的情况下，LASSO性能更好。当响应变量是很多预测变量的函数，而且预测

变量的系数大小都差不多时，岭回归表现得更好。”

——James，2013

还是存在两全其美的机会的，这就引出了我们的下一个主题：弹性网络。

4.1.3　弹性网络 elastic net regression

弹性网络的强大之处在于，它既能做到岭回归不能做的特征提取，也能实现LASSO不能做的特征分组。重申一下，LASSO倾向于在一组相关的特征中选择一个，忽略其他。弹性网络包含了一个混合参数α，它和λ同时起作用。α是一个0和1之间的数，λ和前面一样，用来调节惩罚项的大小。请注意，当α等于0时，弹性网络等价于岭回归；当α等于1时，弹性网络等价于LASSO。实质上，我们通过对β系数的二次项引入一个第二调优参数，将L1惩罚项和L2惩罚项混合在一起。通过最小化（$\mathrm{RSS} + \lambda[(1-\alpha)(\mathrm{sum}|\beta_j|^2)/2 + \alpha(\mathrm{sum}|\beta_j|)]/N$）完成目标。

下面我们就来试试这些技术。使用leaps、glmnet和caret包在下面的商业案例中选择合适的特征并生成合适的模型。

4.2　商业案例

本章依然致力于癌症数据的研究——此处案例是前列腺癌。虽然这个数据集比较小，只有97个观测和9个变量，但通过与传统技术的比较，完全可以使你掌握正则化技术的发展。

4.2.1　业务理解

斯坦福大学医疗中心提供了97个病人的术前**前列腺特异性抗原**（PSA）数据，这些病人将要接受彻底的前列腺切除术（切除全部前列腺），以治疗前列腺癌。**美国癌症学会**估计，2014年有30 000名美国男性死于前列腺癌（http://www.cancer.org）。PSA是前列腺分泌的一种蛋白质，发现于前列腺癌患者的血液。我们的目标是，通过临床检测提供的数据建立一个预测模型。对于患者在手术后能够或应该恢复到什么程度，与其他指标相比，PSA可能是一个更有效的预后指标。手术之后，医生会在各个时间区间检查患者的PSA水平，并通过各种公式确定患者是否康复。术前预测模型和术后数据（这里没有提供）互相配合，就可能提高前列腺癌护理的水平，造福于每年数以千计的患者。

4.2.2　数据理解和数据准备

收集自97位男性的数据集保存在一个具有10个变量的数据框中，如下所示。

- lcavol：肿瘤体积的对数值
- lweight：前列腺重量的对数值
- age：患者年龄（以年计）
- lbph：**良性前列腺增生**（BPH）量的对数值，非癌症性质的前列腺增生。
- svi：贮精囊侵入，一个指标变量，表示癌细胞是否已经透过前列腺壁侵入贮精囊（1=是，0=否）。
- lcp：包膜穿透度的对数值，表示癌细胞扩散到前列腺包膜之外的程度。
- gleason：患者的Gleason评分；由病理学家进行活体检查后给出（2~10），表示癌细胞的变异程度——评分越高，程度越危险。
- pgg45：Gleason评分为4或5所占的百分比（高等级癌症）。
- lpsa：PSA值的对数值，响应变量。
- train：一个逻辑向量（TRUE或FALSE，用来区分训练数据和测试数据）。

这个数据集包含在ElemStatLearn这个R包内。加载所需的程序包和数据框之后，查看变量以及变量之间可能存在的联系，如下所示：

```
    > library(ElemStatLearn) #contains the data
> library(car) #package to calculate Variance Inflation Factor
> library(corrplot) #correlation plots
> library(leaps) #best subsets regression
> library(glmnet) #allows ridge regression, LASSO and elastic net
> library(caret) #parameter tuning
```

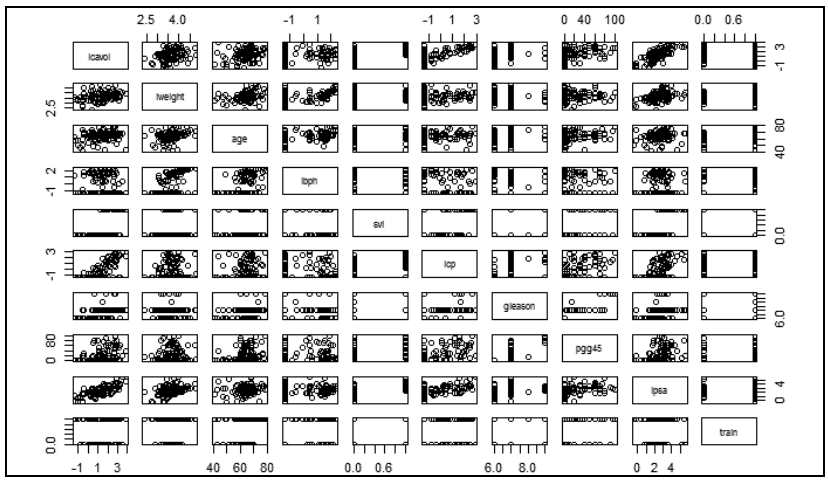
加载程序包之后，调出prostate数据集，查看数据结构，如下所示：

```
    > data(prostate)
> str(prostate)
'data.frame':97 obs. of 10 variables:
$ lcavol  : num -0.58 -0.994 -0.511 -1.204 0.751 ...
$ lweight : num 2.77 3.32 2.69 3.28 3.43 ...
$ age     : int 50 58 74 58 62 50 64 58 47 63 ...
$ lbph    : num -1.39 -1.39 -1.39 -1.39 -1.39 ...
$ svi     : int 0 0 0 0 0 0 0 0 0 0 ...
$ lcp     : num -1.39 -1.39 -1.39 -1.39 -1.39 ...
$ gleason : int 6 6 7 6 6 6 6 6 6 6 ...
$ pgg45   : int 0 0 20 0 0 0 0 0 0 0 ...
$ lpsa    : num -0.431 -0.163 -0.163 -0.163 0.372 ...
$ train   : logi TRUE TRUE TRUE TRUE TRUE TRUE ...
```

检查数据结构会发现几个问题，需要特别注意。检查数据集的特征就会发现，svi、lcp、gleason和pgg45的前10个观测值都具有相同的数值，只有一个例外——gleason的第三个观测值。为了保证这些特征作为输入特征是确实可行的，我们可以做出统计图或表格来理解数据。使用下面的plot()命令，将整个数据框作为输入值，即可建立散点图矩阵，如下所示：

```
> plot(prostate)
```

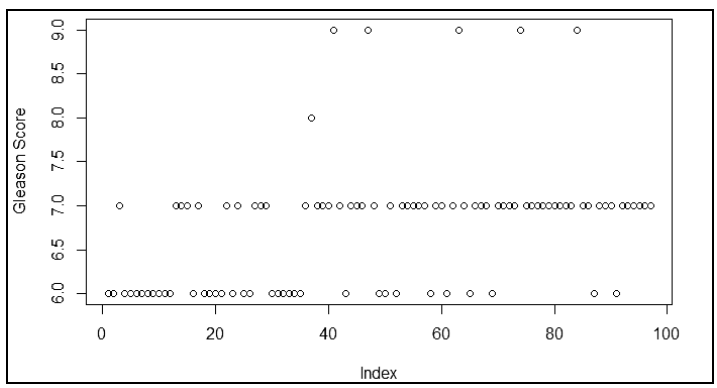
上述命令输出如下。



这么多变量放在一张图上确实有点让人搞不清状况，所以我们循序渐进。可以看出，结果变量lpsa和预测变量lcavol之间确实存在明显的线性关系。还可以看出，这些特征变量的离散度是比较合适的，而且在训练集和测试集之间的分布也比较平衡，可能的例外只有Gleason评分这个特征。请注意，这个数据集中，Gleason评分只有4个值。如果看一下位于train和gleason相交点的那个图，就会发现，有一个Gleason评分值既不属于测试集，也不输入训练集。这可能会使我们的分析过程产生问题，需要进行数据转换。所以，我们专门为这个特征建立一个统计图，如下所示：

```
> plot(prostate$gleason)
```

上述命令输出如下。



现在就看出问题了。每个点代表一个观测，X轴是数据框中观测的编号。只有1个观测的Gleason评分是8.0，只有5个观测的Gleason评分是9.0。你可以建立一个特征值表格来查看确切的数量。如下所示：

```
    > table(prostate$gleason)
 6  7  8  9
35 56  1  5
```

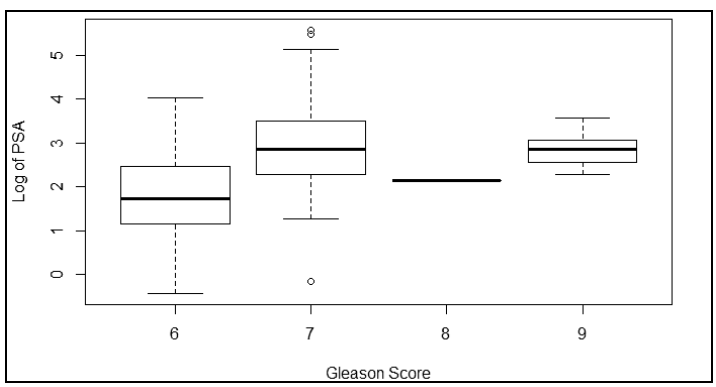
怎么办呢？我们有以下3种选择：

- 完全删除这个特征；
- 仅删除值为8.0和9.0的那些评分；
- 对特征重新编码，建立一个指标变量。

我认为，如果建立一个横轴为**Gleason Score**，纵轴为**Log of PSA**的箱线图，会对我们的选择有所帮助。在前面的章节中，我们使用ggplot2包建立箱线图，但R的基础包也可以建立箱线图，如下所示：

```
> boxplot(prostate$lpsa ~ prostate$gleason, xlab = "Gleason Score",
  ylab = "Log of PSA")
```

上述命令输出如下。



看了上面的图，我认为最好的选择是，将这个特征转换为一个指标变量，0表示评分为6，1表示评分为7或更高。删除特征可能会损失模型的预测能力。缺失值也可能会在我们将要使用的glmnet包中引起问题。

你可以非常简单地实现对指标变量的编码，通过一行代码即可。使用`ifelse()`命令，指定你想在数据框中转换的列，然后按照这个规则转换：如果观测值的特征值为x，则将其编码为y，否则将其编码为z。

```
> prostate$gleason <- ifelse(prostate$gleason == 6, 0, 1)
```

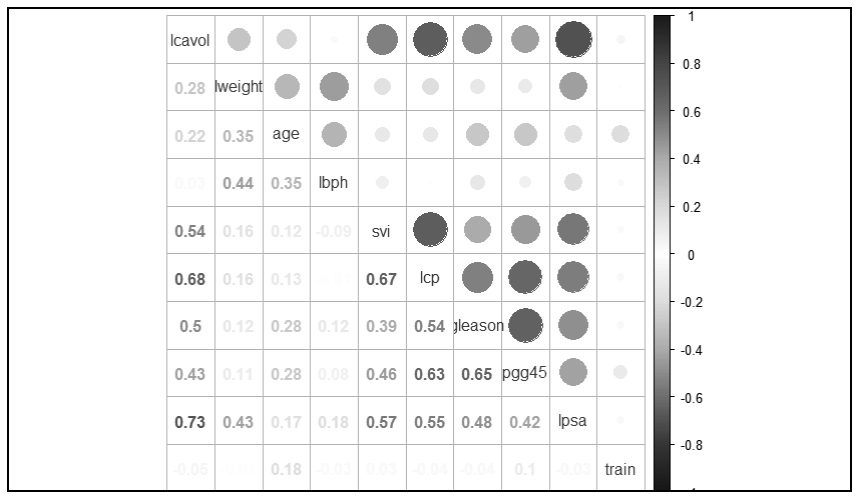
同样，我们建立一个表格，验证数据转换结果确如预想。如下所示：

```
    > table(prostate$gleason)
0   1
35 62
```

转换结果完美无缺！既然散点图矩阵比较难以理解，我们就再看看相关性统计图，它可以表示特征之间是否存在相关性或依赖。先用cor()函数建立一个相关性对象，然后利用corrplot库中的corrplot.mixed()函数做出相关性统计图，如下所示：

```
    > p.cor = cor(prostate)
> corrplot.mixed(p.cor)
```

上述命令输出如下。



这样又跳出两个问题。首先，PSA和肿瘤体积的对数（lcavol）高度相关。回想一下，在散点图矩阵中，它们表现出很强的线性相关关系。其次，多重共线性是个问题。例如，肿瘤体积还与包膜穿透相关，而包膜穿透还与贮精囊侵入相关。这真是一个很有趣的机器学习实战例子！

开始机器学习之前，必须先建立训练数据集和测试数据集。既然观测值中已经有一个特征指明这个观测值是否属于训练集，我们就可以使用subset()命令将train值为TRUE的观测值分到训练集中，将train值为FALSE的观测值分到测试集。将train删除也是必须的，因为我们不想把它作为预测特征。如下所示：

```
    > train <- subset(prostate, train == TRUE)[, 1:9]
> str(train)
'data.frame':67 obs.  of  9 variables:
 $ lcavol : num  -0.58 -0.994 -0.511 -1.204 0.751 ...
 $ lweight: num  2.77 3.32 2.69 3.28 3.43 ...
 $ age    : int  50 58 74 58 62 50 58 65 63 63 ...
 $ lbph   : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
 $ svi    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ lcp    : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
 $ gleason: num  0 0 1 0 0 0 0 0 0 1 ...
 $ pgg45  : int  0 0 20 0 0 0 0 0 0 30 ...
 $ lpsa   : num  -0.431 -0.163 -0.163 -0.163 0.372 ...
```

```
> test <- subset(prostate, train == FALSE)[, 1:9]
> str(test)
'data.frame':30 obs. of  9 variables:
 $ lcavol : num  0.737 -0.777 0.223 1.206 2.059 ...
 $ lweight: num  3.47 3.54 3.24 3.44 3.5 ...
 $ age    : int  64 47 63 57 60 69 68 67 65 54 ...
 $ lbph   : num  0.615 -1.386 -1.386 -1.386 1.475 ...
 $ svi    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ lcp    : num  -1.386 -1.386 -1.386 -1.386 -0.431 1.348 ...
 $ gleason: num  0 0 0 1 1 0 0 1 0 0 ...
 $ pgg45  : int  0 0 0 5 20 0 0 20 0 0 ...
 $ lpsa   : num  0.765 1.047 1.047 1.399 1.658 ...
```


4.3 模型构建与模型评价

数据已经准备好了，我们将开始构建模型。为了进行对比，先用最优子集回归建立一个模型，像我们在前两章中做的那样，然后使用正则化技术建立模型。

4.3.1 最优子集

下面的代码（或者说大部分代码）基本上是我们在第2章中使用过的代码的翻版。通过`regsubsets()`命令建立一个最小子集对象，然后指定训练数据集。选择出的特征随后用在测试集上，通过计算均方误差来评价模型。

我们建立模型的语法为`lpsa~.`，使用波形符号加句号说明，要使用数据框中除响应变量之外的所有变量进行预测。如下所示：

```
> subfit <- regsubsets(lpsa ~ ., data = train)
```

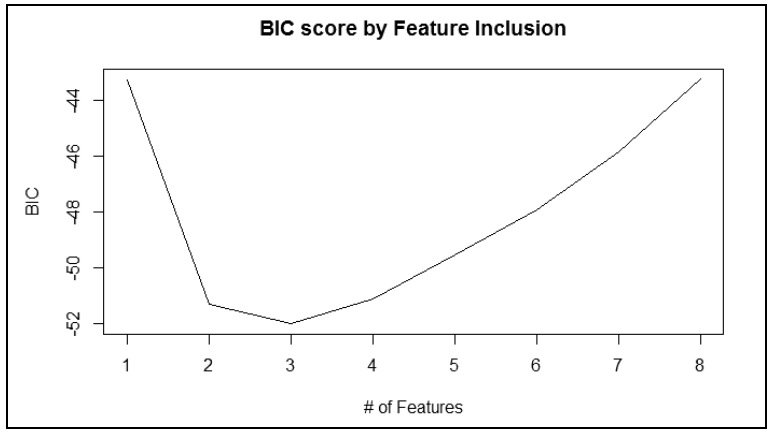
模型建立之后，你可以通过两行代码得到最优子集。第一行代码将摘要模型写入一个对象，然后从这个对象提取各个子集，使用`which.min()`命令确定最优子集。在本例中，我们使用第2章中讨论过的贝叶斯信息准则，如下所示：

```
    > b.sum <- summary(subfit)
> which.min(b.sum$bic)
  [1] 3
```

结果告诉我们，三特征模型具有最小的BIC值。可以通过一个统计图查看模型性能和子集组合之间的关系，如下所示：

```
> plot(b.sum$bic, type = "l", xlab = "# of Features", ylab = "BIC",
  main = "BIC score by Feature Inclusion")
```

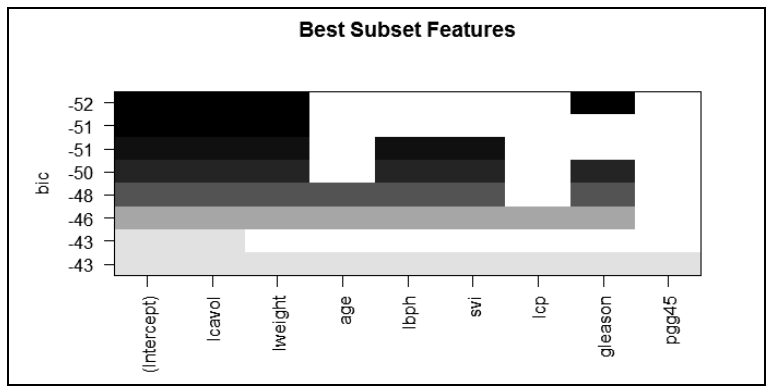
上述命令输出如下页图。



对实际模型做出统计图，可以让我们进行更详细的检查。如下所示：

```
> plot(subfit, scale = "bic", main = "Best Subset Features")
```

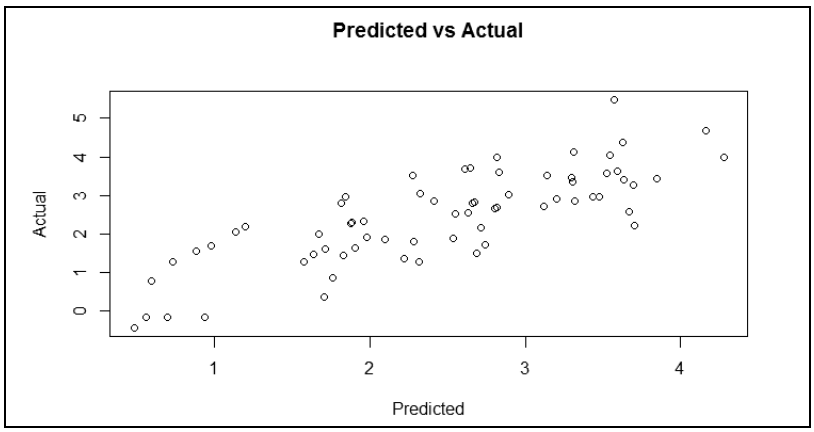
上述命令输出如下。



于是，上图告诉我们具有最小BIC值的模型中的3个特征是：lcavol、lweight和gleason。值得注意的是，lcavol包含在所有模型组合之中，这与我们之前的数据探索结果是一致的。现在，可以在测试集上试验模型了，但要先用模型的拟合值与实际值画一张图，看看最终解中的线性关系，并检查同方差性。需要用以上3个变量建立一个线性模型，因为是用OLS建立的模型，所以把它存储在名为ols的对象中。然后，使用OLS拟合出的模型就可以同训练集中的实际值进行对比了。如下所示：

```
    > ols <- lm(lpsa ~ lcavol + lweight + gleason, data = train)
> plot(ols$fitted.values, train$lpsa, xlab = "Predicted", ylab =
      "Actual", main = "Predicted vs Actual")
```

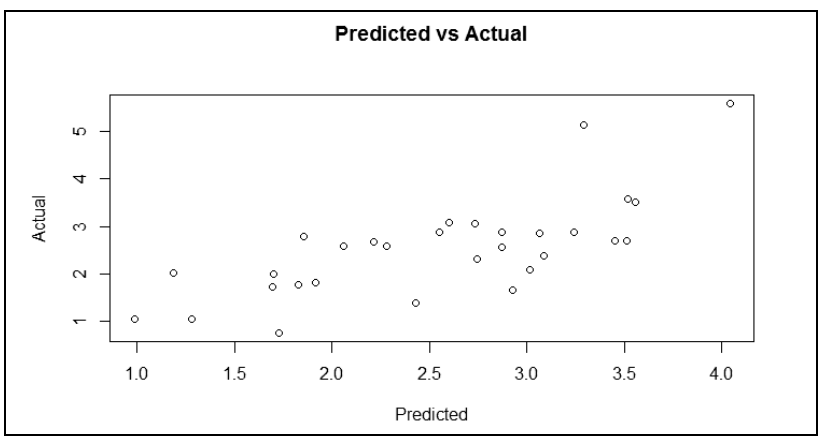
上述命令输出如下页图。



从图中可以看出，在训练集上线性拟合表现得很好，也不存在异方差性。然后看看模型在测试集上的表现，使用predict()函数并指定newdata = test，如下所示：

```
    > pred.subfit <- predict(ols, newdata = test)
> plot(pred.subfit, test$lpsa , xlab = "Predicted", ylab =
      "Actual", main = "Predicted vs Actual")
```

对象中的值可以用来生成统计图，表示预测值和实际值之间的关系。如下图所示。



这个图还不是太难看。总体来说，图中呈现一种线性关系，只不过当PSA值比较高时，有两个离群点。结束本小节之前，我们需要计算**均方误差**，以便在不同模型构建技术之间进行比较。这非常简单，只要算出残差并求出残差平方的均值即可，如下所示：

MSE: mean squared error

```
    > resid.subfit <- test$lpsa - pred.subfit
> mean(resid.subfit^2)
[1] 0.5084126
```

MSE值为0.508，以此为基准继续下面的内容。

4.3.2　岭回归

在岭回归中，我们的模型会包括全部8个特征，所以岭回归模型与最优子集模型的比较真是令人期待啊。我们要使用的程序包glmnet实际上已经加载过了。这个程序包要求输入特征存储在矩阵中，而不是在数据框中。岭回归的命令形式为glmnet(x=输入矩阵, y=响应变量, family=分布函数, alpha=0)。这里的alpha为0时，表示进行岭回归；alpha为1时，表示进行LASSO。

要准备好供glmnet使用的训练集数据也很容易，使用as.matrix()函数处理输入数据，并建立一个向量作为响应变量，如下所示：

```
> x <- as.matrix(train[, 1:8])
> y <- train[, 9]
```

现在可以使用岭回归了，我们把结果保存在一个对象中，可以为对象起一个恰当的名字，比如ridge。这里有一点非常重要，请一定注意：glmnet包会在计算λ值之前首先对输入进行标准化，然后计算非标准化系数。你需要指定响应变量的分布为gaussian，因为它是连续的；还要指定alpha = 0，表示进行岭回归。如下所示：

```
> ridge <- glmnet(x, y, family = "gaussian", alpha = 0)
```

这个对象包含了我们进行模型评价所需的所有信息。首先尝试print()命令，它会展示非0系数的数量，解释偏差百分比以及相应的λ值。程序包中算法默认的计算次数是100，但如果偏差百分比在两个λ值之间的提高不是很显著的话，算法会在100次计算之前停止。也就是说，算法收敛于最优解。为了节省篇幅，我只在下面列出了前5个和后10个λ结果：

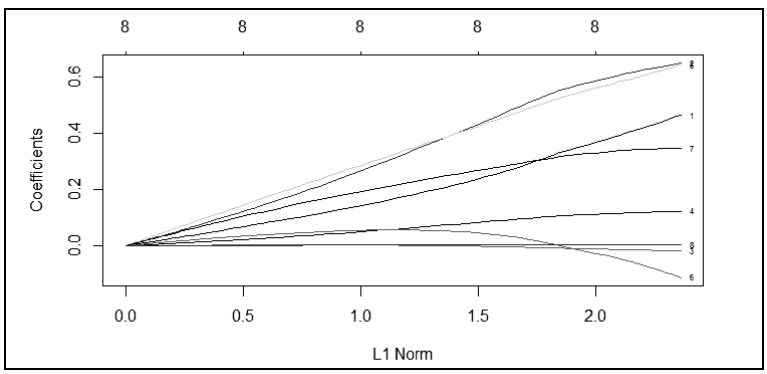
```
> print(ridge)
Call:  glmnet(x = x, y = y, family = "gaussian", alpha = 0)
      Df       %Dev   Lambda
 [1,]  8 3.801e-36 878.90000
 [2,]  8 5.591e-03 800.80000
 [3,]  8 6.132e-03 729.70000
 [4,]  8 6.725e-03 664.80000
 [5,]  8 7.374e-03 605.80000
 ..........................
[91,]  8 6.859e-01   0.20300
[92,]  8 6.877e-01   0.18500
[93,]  8 6.894e-01   0.16860
[94,]  8 6.909e-01   0.15360
[95,]  8 6.923e-01   0.13990
[96,]  8 6.935e-01   0.12750
[97,]  8 6.946e-01   0.11620
[98,]  8 6.955e-01   0.10590
[99,]  8 6.964e-01   0.09646
[100,] 8 6.971e-01   0.08789
```

以第100行为例。可以看出非0系数，即模型中包含的特征的数量为8。请记住，在岭回归中，这个数是不变的。还可以看出解释偏差百分比为0.6971，以及这一行的调优系数λ的值为0.08789。

此处即可决定选择在测试集上使用哪个λ。这个λ值应该是0.08789，但是为了简单起见，在测试集上可以试一下0.10。此时，一些统计图是非常有用的。我们先看看程序包中默认的统计图，设定`label = TRUE`可以给曲线加上注释，如下所示：

```
> plot(ridge, label = TRUE)
```

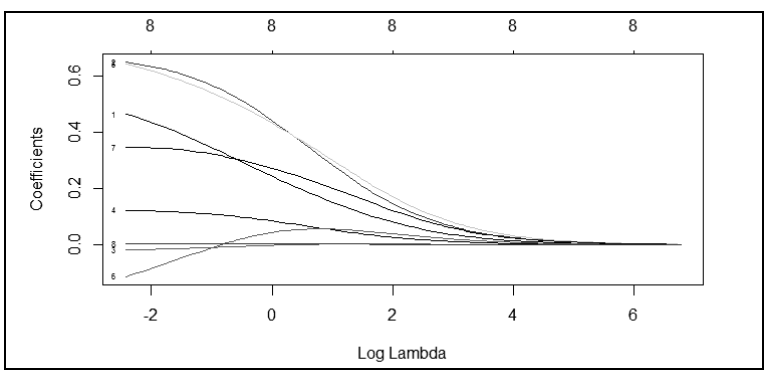
上述命令输出如下。



在默认图中，Y轴是**系数**值，X轴是**L1范数**，图中显示了系数值和L1范数之间的关系。图的上方有另一条X轴，其上的数值表示模型中的特征数。查看统计图的一种更好方式是，看系数值如何随着λ的变化而变化。只需在`plot()`命令中稍稍调整，加上参数`xvar="lambda"`即可。另一种选择是，看系数值如何随解释偏差百分比变化，将`lamda`换成`dev`即可。

```
> plot(ridge, xvar = "lambda", label = TRUE)
```

上述命令输出如下。



这张图非常有价值，因为它表明，当λ值减小时，压缩参数随之减小，而系数绝对值随之增大。要想看看当λ为一个特定值时的系数值，可以使用`coef()`命令。现在，看一下当λ为0.1时，系数值是多少。指定参数`s=0.1`，同时指定参数`exact=TRUE`，告诉glmnet在拟合模型时使用具

体的λ值，而不是从λ值的两侧选值插入。如下所示：

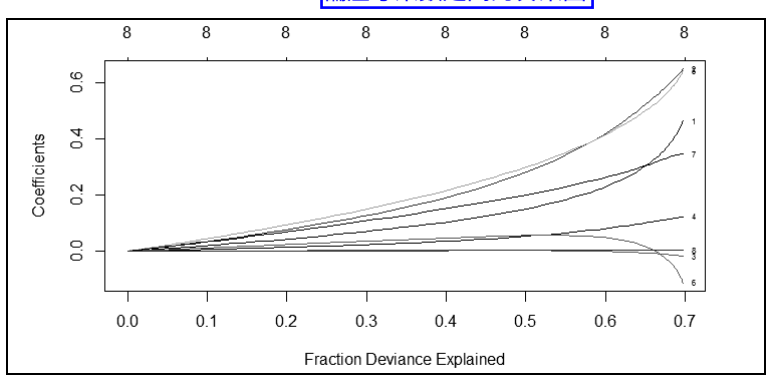
```
    > ridge.coef <- coef(ridge, s = 0.1, exact = TRUE)
> ridge.coef
9 x 1 sparse Matrix of class "dgCMatrix"
                      1
(Intercept)  0.13062197
lcavol       0.45721270
lweight      0.64579061
age         -0.01735672
lbph         0.12249920
svi          0.63664815
lcp         -0.10463486
gleason      0.34612690
pgg45        0.00428580
```

需要特别注意的是，age、lcp和pgg45的系数非常接近0，但还不是0。别忘了再看一下偏差与系数之间的关系图：

```
> plot(ridge, xvar = "dev", label = TRUE)
```

上述命令输出如下。

偏差与系数之间的关系图



同前两张图相比，我们可以从这张图中看出，当λ减小时，系数会增大，解释偏差百分比也会增大。如果将λ值设为0，就会忽略收缩惩罚，模型将等价于OLS。

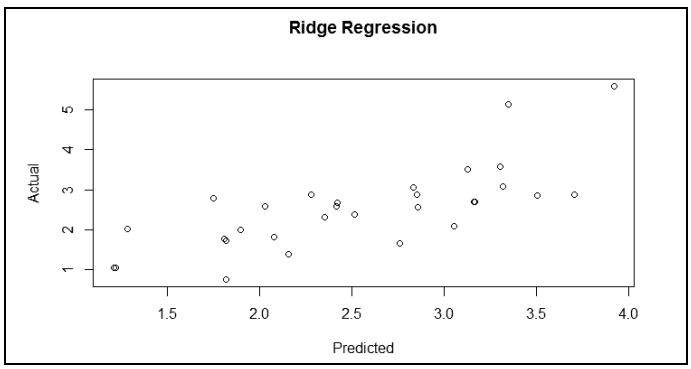
为了在测试集上证明这一点，需要转换特征，像我们在训练集上做的一样：

```
> newx <- as.matrix(test[, 1:8])
```

然后，使用predict()函数建立一个名为ridge.y的对象，指定参数type="response"以及λ值为0.10，画出表示预测值和实际值关系的统计图，如下所示：

```
    > ridge.y <- predict(ridge, newx = newx, type = "response", s =
      0.1)
> plot(ridge.y, test$lpsa, xlab = "Predicted", ylab = "Actual",main
      = "Ridge Regression")
```

上述命令输出如下。



表示岭回归中预测值和实际值关系的统计图看上去与最优子集的非常相似，同样地，在PSA测量结果比较大的一端有两个有趣的离群点。在实际情况下，我建议对离群点进行更深入的研究，搞清楚是它们真的与众不同，还是我们忽略了什么。这就是领域专家的用武之地。与MSE基准的比较可能告诉我们一些不同的事。先算出残差，然后算出残差平方的平均值：

```
    > ridge.resid <- ridge.y - test$lpsa
> mean(ridge.resid^2)
[1] 0.4789913
```

岭回归给出的MSE稍好一点E。现在是时候检验LASSO了，看看我们能否将误差再减少一些。

4.3.3 LASSO

下面运行LASSO就非常简单了，只要改变岭回归模型的一个参数即可。也就是说，在`glmnet()`语法中将`alpha=0`变为`alpha=1`。运行代码，看看模型的输出，检查前5个和后10个拟合结果：

```
    > lasso <- glmnet(x, y, family = "gaussian", alpha = 1)
> print(lasso)
Call: glmnet(x = x, y = y, family = "gaussian", alpha = 1)
Df %Dev Lambda
[1,] 0 0.00000 0.878900
[2,] 1 0.09126 0.800800
[3,] 1 0.16700 0.729700
[4,] 1 0.22990 0.664800
[5,] 1 0.28220 0.605800
.......................
[60,] 8 0.70170 0.003632
[61,] 8 0.70170 0.003309
[62,] 8 0.70170 0.003015
[63,] 8 0.70170 0.002747
[64,] 8 0.70180 0.002503
[65,] 8 0.70180 0.002281
```

```
[66,] 8 0.70180 0.002078
[67,] 8 0.70180 0.001893
[68,] 8 0.70180 0.001725
[69,] 8 0.70180 0.001572
```

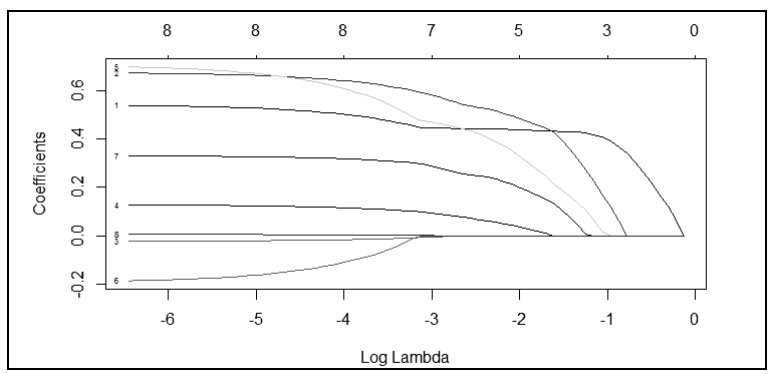
请注意，模型构建过程在69步之后停止了，因为解释偏差不再随着λ值的增加而减小。还要注意，Df列现在也随着λ变化。初看上去，当λ值为0.001572时，所有8个特征都应该包括在模型中。然而，出于测试的目的，我们先用更少特征的模型进行测试，比如7特征模型。从下面的结果行中可以看到，λ值大约为0.045时，模型从7个特征变为8个特征。因此，使用测试集评价模型时要使用这个λ值。如下所示：

```
[31,] 7 0.67240 0.053930
[32,] 7 0.67460 0.049140
[33,] 7 0.67650 0.044770
[34,] 8 0.67970 0.040790
[35,] 8 0.68340 0.037170
```

和岭回归一样，可以在图中画出结果。如下所示：

```
> plot(lasso, xvar = "lambda", label = TRUE)
```

上述命令输出如下。



这张图很有趣，真正展示了LASSO是如何工作的。请注意标号为8、3和6的曲线的表现，这几条曲线分别对应特征pgg45、age和lcp。看上去lcp一直接近于0，直到作为最后一个特征被加入模型。可以通过与岭回归中同样的操作看看7特征模型的系数值，将λ值放入coef()函数，如下所示：

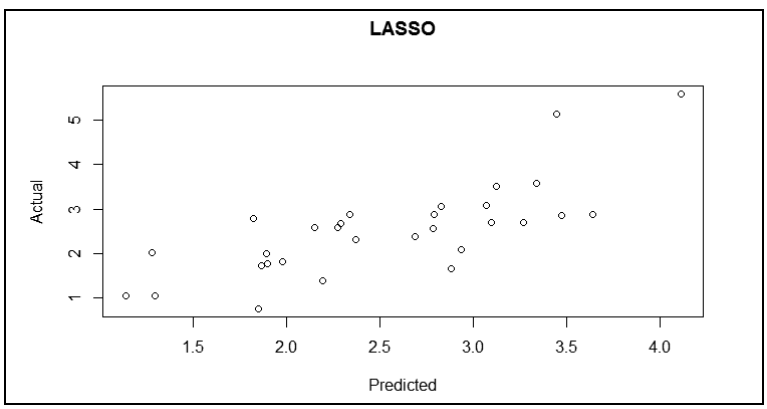
```
    > lasso.coef <- coef(lasso, s = 0.045, exact = TRUE)
> lasso.coef
9 x 1 sparse Matrix of class "dgCMatrix"
                            1
(Intercept) -0.1305852115
lcavol       0.4479676523
lweight      0.5910362316
age         -0.0073156274
```

```
lbph          0.0974129976
svi           0.4746795823
lcp           .
gleason       0.2968395802
pgg45         0.0009790322
```

LASSO算法在λ值为0.045时，将lcp的系数归零。下面是LASSO模型在测试集上的表现：

```
    > lasso.y <- predict(lasso, newx = newx, type = "response", s =
      0.045)
> plot(lasso.y, test$lpsa, xlab = "Predicted", ylab = "Actual",
      main = "LASSO")
```

上述命令输出如下。



像以前一样，算出MSE的值：

```
    > lasso.resid <- lasso.y - test$lpsa
> mean(lasso.resid^2)
[1] 0.4437209
```

看起来我们的统计图和前面一样，只是MSE值有了一点点改进，重大改进的最后希望只能寄托在弹性网络上面了。要进行弹性网络建模，还可以继续使用glmnet包，要做的调整是不但要解出λ值，还要解出弹性网络参数α。回忆一下，α = 0表示岭回归惩罚，α = 1表示LASSO惩罚，弹性网络参数为0≤α≤1。同时解出两个不同的参数会非常麻烦，令人心生怯意，但是，我们可以求助于R中的老朋友——caret包。

4.3.4 弹性网络

caret包旨在解决分类问题和训练回归模型，它配有一个很棒的网站，帮助人们掌握其所有功能：http://topepo.github.io/caret/index.html。这个软件包有很多功能可以使用，其中一些会在后面的章节中用到。现在的目的集中于找到λ和弹性网络混合参数α的最优组合。可以通过下面3个简单的步骤完成。

Elastic Net produces a regression model that is penalized with both the L1-norm and L2-norm. The consequence of this is to effectively shrink coefficients (like in ridge regression) and to set some coefficients to zero (as in LASSO).

(1) 使用R基础包中的expand.grid()函数，建立一个向量存储我们要研究的α和λ的所有组合。

(2) 使用caret包中的trainControl()函数确定重取样方法，像第2章一样，使用LOOCV。

(3) P在caret包的train()函数中使用glmnet()训练模型来选择α和λ。

一旦选定参数，我们会像在岭回归和LASSO中做的那样，在测试数据上使用它们。

我们的组合网格应该足够大，以便能获得最优模型；但又不能太大，导致计算上不可行。对于本章这种规模的数据集，不用担心会出现这样的问题，但一定要时刻牢记。

可以按照下面的规则试验这两个超参数。

- α从0到1，每次增加0.2；请记住，α被绑定在0和1之间。
- λ从0到0.20，每次增加0.02；0.2的λ值是岭回归λ值（$\lambda=0.1$）和LASSOλ值（$\lambda=0.045$）之间的一个中间值。

可以使用expand.grid()函数建立这个向量并生成一系列数值，caret包会自动使用这些数值。caret包通过下列代码生成α值和λ值：

```
> grid <- expand.grid(.alpha = seq(0, 1, by = .2), .lambda =
  seq(0.00, 0.2,  by = 0.02))
```

使用table()函数，可以看到α和λ的全部66种组合：

```
    > table(grid)
      .lambda
.alpha 0 0.02 0.04 0.06 0.08 0.1 0.12 0.14 0.16 0.18 0.2
   0   1    1    1    1    1   1    1    1    1    1   1
   0.2 1    1    1    1    1   1    1    1    1    1   1
   0.4 1    1    1    1    1   1    1    1    1    1   1
   0.6 1    1    1    1    1   1    1    1    1    1   1
   0.8 1    1    1    1    1   1    1    1    1    1   1
   1   1    1    1    1    1   1    1    1    1    1   1
```

可以确认这就是我们想要的结果——α值在0和1之间，λ值在0和0.2之间。

对于重取样方法，我们要在代码中将method参数指定为LOOCV。还有其他的重取样方法可以选择，比如自助法和K折交叉验证法。trainControl()函数中有更多选择，我们会在后续章节进行研究。

在trainControl()函数中，你还可以通过selectionFunctioin()函数指定模型选择方法。对于定量型响应变量，使用算法的默认选择**均方根误差**即可完美实现：

```
> control <- trainControl(method = "LOOCV")
```

现在可以使用train()函数确定最优的弹性网络参数了。这个函数和lm()很相似，只需在函数语法中加上method="glmnet", trControl=control, tuneGrid=grid。将结果存储在一个名为enet.train的对象中：

```
> enet.train <- train(lpsa ~ ., data = train, method = "glmnet",
  trControl = control, tuneGrid = grid)
```

调用这个对象，可以看到能够得出最小RMSE值的参数组合，如下所示：

```
    > enet.train
glmnet
67 samples
 8 predictor
No pre-processing
Resampling:
Summary of sample sizes: 66, 66, 66, 66, 66, 66, ...
Resampling results across tuning parameters:
  alpha  lambda  RMSE  Rsquared
  0.0    0.00    0.750 0.609
  0.0    0.02    0.750 0.609
  0.0    0.04    0.750 0.609
  0.0    0.06    0.750 0.609
  0.0    0.08    0.750 0.609
  0.0    0.10    0.751 0.608
   ......................
  1.0    0.14    0.800 0.564
  1.0    0.16    0.809 0.558
  1.0    0.18    0.819 0.552
  1.0    0.20    0.826 0.549
```

我们选择最优模型的原则是RMSE值最小，模型最后选定的最优参数组合是$\alpha=0$，$\lambda=0.08$。

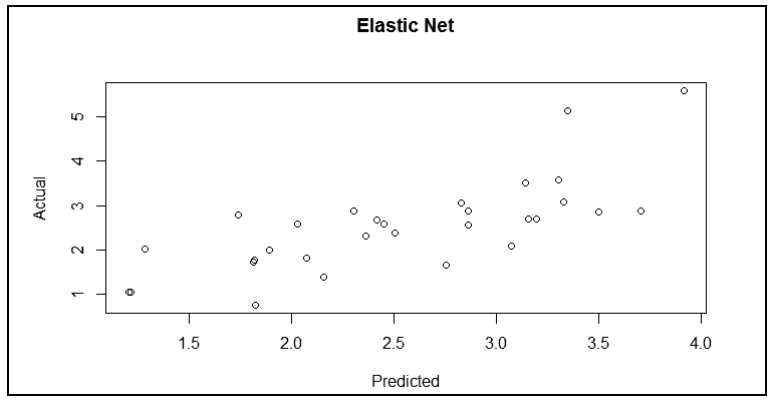
实验设计得到的最优调优参数是$\alpha=0$和$\lambda=0.08$，相当于glmnet中$s=0.08$的岭回归。回忆一下，我们在4.3.2节中得出的λ是0.10。R方为61%，真是乏善可陈。

在测试集上验证模型的过程和前面一样：

```
    > enet <- glmnet(x, y, family = "gaussian", alpha = 0, lambda =
      .08)
> enet.coef <- coef(enet, s = .08, exact = TRUE)
> enet.coef
9 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept)  0.137811097
lcavol       0.470960525
lweight      0.652088157
age         -0.018257308
lbph         0.123608113
svi          0.648209192
lcp         -0.118214386
gleason      0.345480799
```

```
pgg45         0.004478267
> enet.y <- predict(enet, newx=newx, type="response", s=.08)
> plot(enet.y, test$lpsa, xlab="Predicted", ylab="Actual",
      main="Elastic Net")
```

上述命令输出如下。



和以前一样，计算MSE：

```
    > enet.resid <- enet.y - test$lpsa
> mean(enet.resid^2)
[1] 0.4795019
```

这个模型的误差与岭回归很接近。在测试集上，LASSO模型在误差方面表现最好。模型可能过拟合了！我们的三特征最优子集模型是最容易解释的，但考虑误差的话，却更应该接收另外一种技术得出的模型。可以在glmnet包中使用10折交叉验证来确定哪个模型更好。

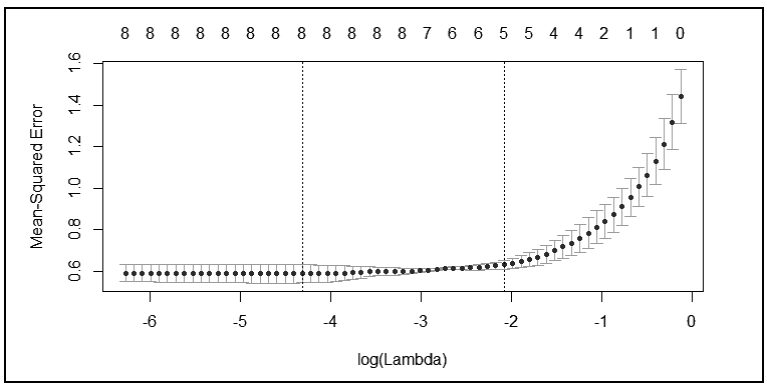
4.3.5　使用 glmnet 进行交叉验证

我们通过caret包使用过LOOCV，现在试试K折交叉验证。glmnet包在使用`cv.glmnet()`估计λ值时，默认使用10折交叉验证。在K折交叉验证中，数据被划分成k个相同的子集（折），每次使用$k-1$个子集拟合模型，然后使用剩下的那个子集做测试集，最后将k次拟合的结果综合起来（一般取平均数），确定最后的参数。

在这个方法中，每个子集只有一次用作测试集。在glmnet包中使用K折交叉验证非常容易，结果包括每次拟合的λ值和响应的MSE。默认设置为$\alpha=1$，所以如果你想试试岭回归或弹性网络，必须指定α值。因为我们想看看尽可能少的输入特征的情况，所以还是使用默认设置，但由于训练集中数据量的原因，只分3折：

```
    > set.seed(317)
> lasso.cv = cv.glmnet(x, y, nfolds = 3)
> plot(lasso.cv)
```

上述代码的输出如下。



CV统计图和glmnet中其他统计图有很大区别，它表示λ的**对数值**和**均方误差**之间的关系，还带有模型中特征的数量。图中两条垂直的虚线表示取得MSE最小值的logλ（左侧虚线）和距离最小值一个标准误差的logλ。如果有过拟合问题，那么距离最小值一个标准误的位置是非常好的解决问题的起点。你还可以得到这两个λ的具体值，如下所示：

```
> lasso.cv$lambda.min #minimum
[1] 0.0133582
> lasso.cv$lambda.1se #one standard error away
[1] 0.124579
```

使用lambda.1se可以完成下面的过程，查看系数并在测试集上进行模型验证：

```
> coef(lasso.cv, s = "lambda.1se")
9 x 1 sparse Matrix of class "dgCMatrix"
 1
(Intercept) -0.13543760
lcavol 0.43892533
lweight 0.49550944
age .
lbph 0.04343678
svi 0.34985691
lcp .
gleason 0.21225934
pgg45 .

> lasso.y.cv = predict(lasso.cv, newx=newx, type = "response",
    s = "lambda.1se")

> lasso.cv.resid = lasso.y.cv - test$lpsa

> mean(lasso.cv.resid^2)
[1] 0.4465453
```

这个模型的误差为0.45，只有5个特征，排除了age、lcp和pgg45。

4.4　模型选择

通过对数据集的分析和研究，我们得出5个不同模型。下面是这些模型在测试集上的误差。

- 最优子集模型：0.51
- 岭回归模型：0.48
- LASSO模型：0.44
- 弹性网络模型：0.48
- LASSO交叉验证模型：0.45

仅看误差的话，7特征LASSO模型表现最好。但是，这个最优模型能解决我们试图回答的问题吗？我们通过交叉验证得到λ值约为0.125的模型，它更简约，也可能更加合适。我更倾向于选择它，因为其解释性更好。

说到这里，显然需要来自肿瘤专家、泌尿科专家和病理学家的专业知识，来帮助我们搞清什么是最有意义的。确实如此，但同时也需要更多数据。在本例的样本规模之下，仅改变随机数种子或重新划分训练集和测试集都可能使结果发生大的改变（你可以试试）。到头来，这些结果非但不能提供答案，还可能引起更多问题。但是，这很糟糕吗？当然不是，除非在项目开始之初你就犯下了严重的错误，对你能做的事情夸下海口。我们在第1章就提出了合理的警告，要审慎稳妥地推进你的任务。

4.5　正则化与分类问题

上面使用的正则化技术同样适用于分类问题，二值分类和多值分类皆可。因此，结束本章之前，我们再介绍一下可以用于逻辑斯蒂回归问题的示例代码。更具体地说，是可以用于上一章乳腺癌数据集的代码。在具有定量型响应变量的回归问题中，正则化是一种处理高维数据集的重要技术。

逻辑斯蒂回归示例

回忆一下，在我们分析过的乳腺癌数据中，肿瘤是恶性的概率可以用逻辑斯蒂函数表示如下：

$$\mathrm{P}(\text{malignant}) = 1/1 + e^{-(B_0 + B_1X_1 + B_nX_n)}$$

因为这个函数中有线性的部分，所以可以使用L1和L2正则化。和前一章一样，先加载并准备好乳腺癌数据：

```
> library(MASS)
> biopsy$ID = NULL
> names(biopsy) = c("thick", "u.size", "u.shape", "adhsn",
```

```
"s.size", "nucl", "chrom", "n.nuc", "mit", "class")
> biopsy.v2 <- na.omit(biopsy)
> set.seed(123)
> ind <- sample(2, nrow(biopsy.v2), replace = TRUE, prob = c(0.7,
  0.3))
> train <- biopsy.v2[ind==1, ]
> test <- biopsy.v2[ind==2, ]
```

转换数据，生成输入矩阵和标签：

```
> x <- as.matrix(train[, 1:9])
> y <- train[, 10]
```

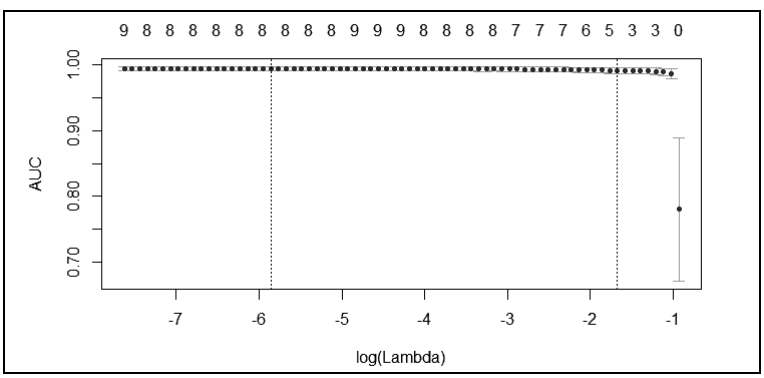
在函数cv.glmnet中，将family的值设定为binomial，将measure的值设定为曲线下面积（auc），并使用5折交叉验证：

```
> set.seed(3)
> fitCV <- cv.glmnet(x, y, family = "binomial",
      type.measure = "auc",
      nfolds = 5)
```

绘制fitCV，可以看出AUC和λ的关系：

```
> plot(fitCV)
```

绘图命令输出如下。



非常有意思！仅加入一个特征就可以使AUC有立竿见影的提高。下面看看在一个标准误差之处的模型系数：

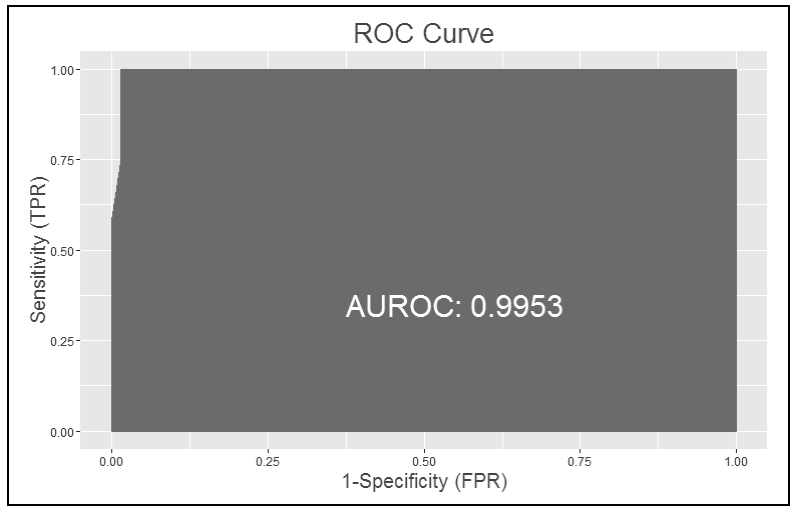
```
> fitCV$lambda.1se
[1] 0.1876892
> coef(fitCV, s = "lambda.1se")
10 x 1 sparse Matrix of class "dgCMatrix"
 1
(Intercept) -1.84478214
thick 0.01892397
u.size 0.10102690
```

```
u.shape 0.08264828
adhsn .
s.size .
nucl 0.13891750
chrom .
n.nuc .
mit .
```

可以看出，选择出的4个特征是thickness、u.size、u.shape和nucl。和前一章一样，通过误差和auc，我们看看这个模型在测试集上的表现：

```
> library(InformationValue)
> predCV <- predict(fitCV, newx = as.matrix(test[, 1:9]),
    s = "lambda.1se",
    type = "response")
actuals <- ifelse(test$class == "malignant", 1, 0)
misClassError(actuals, predCV)
[1] 0.0622
> plotROC(actuals, predCV)
```

上述代码输出如下。



结果显示，这个模型的效果与前面的逻辑斯蒂回归模型基本一样。看上去，lambda.1se还不是最优的选择。我们看看使用lambda.min选择的模型是否可以再次改善样本预测结果：

```
> predCV.min <- predict(fitCV, newx = as.matrix(test[, 1:9]),
    s = "lambda.min",
    type = "response")
> misClassError(actuals, predCV.min)
[1] 0.0239
```

就是它了！这个错误率和第3章中的一样低。

4.6 小结

本章的目标是，通过一个小数据集介绍如何对线性模型应用高级特征选择技术。数据集的结果变量是定量的，但我们使用的glmnet包也支持定性的结果变量（二值分类和多值分类）。我们介绍了正则化及其包含的3种技术，并应用这些技术构建模型，然后进行了比较。正则化是一项强大的技术，与其他建模技术相比，既可以提高计算效率，还可以提取更有意义的特征。此外，我们还开始使用caret包在训练模型时使多个参数达到最优化。直到现在，我们还是仅讨论了线性模型。接下来的两章中，我们会开始使用非线性模型解决分类问题和回归问题。