第 3 章

# 逻辑斯蒂回归与判别分析



“我们这个世界的真正逻辑寓于概率的计算之中。”

——詹姆斯·克拉克·麦克斯韦，苏格兰物理学家

OLS: Ordinary least squares

我们在第2章学习了如何使用**普通最小二乘法**预测定量结果，换句话说，就是线性回归。是时候换个主题了，下面将学习如何使用算法预测定性结果。这样的结果变量可能是二值变量（如男/女、是否购买、良性/恶性肿瘤），也可能是多值分类（如教育水平、眼睛颜色）。不管这种结果变量是二值的还是多值的，分析的任务都是对一个观测属于结果变量的某个类别的概率做出预测。换句话说，我们开发算法的目的是对观测进行分类。

开始研究分类问题之前，我们先讨论为什么OLS线性回归不是解决分类问题的合适技术，以及本章将要介绍的算法是如何解决这类问题的。然后讨论一个具体问题，预测肿瘤活体检查的结果是良性还是恶性。我们使用的数据集非常著名，这个数据集通常被称为“威斯康星乳腺癌数据集”。为解决这个问题，首先建立一个逻辑斯蒂回归模型并进行解释，然后研究模型检测方法以选择特征，期望得到最合适的模型。接下来，讨论线性判别分析方法和二次判别分析方法，并将这两种方法与逻辑斯蒂回归进行比较。在此之后，使用乳腺癌数据建立预测模型。最后介绍多元样条回归方法，以及选择解决实际问题的最佳综合算法，从而圆满解决整个问题。这些方法将为后续章节中的更高级机器学习方法打下基础。

## 3.1 分类方法与线性回归

既然我们在前一章学习了最小二乘回归方法，为什么不能把它用于定性结果呢？事实证明，你可以用，但后果自负。不妨假设你有一个结果需要预测，结果有三类：轻微、中等、严重。你和你的同事认为轻微与中等之间的差别，以及中等与严重之间的差别是相等的，是一个线性关系。你可以建立虚拟变量，用0表示轻微，1表示中等，2表示严重。如果你有理由相信这个，那么线性回归就是一个可接受的解决方案。但是，定性估计（如前面的例子）有可能产生高测量误差，从而使OLS发生偏离。多数商业问题没有一个在科学上能够被接受的方法，可以将一个定性的响应变量转化为定量的响应变量。如果我们有一个双结果响应变量，比如考试通过和考试失败，应

该怎么做？可以使用虚拟变量，将“考试失败”编码为0，“考试通过”编码为1，然后使用线性回归建立一个模型，模型预测值为观测考试通过还是考试失败的概率。但是概率范围为[0, 1]，模型中$Y$的估计值非常有可能超出这个范围。如果这样，那就有点不好解释了。

## 3.2　逻辑斯蒂回归

如前所述，分类问题最好使用位于0和1之间的概率进行建模。对于所有观测，我们有很多不同的函数可以实现这个功能，本章重点讨论逻辑斯蒂函数。逻辑斯蒂回归中使用的逻辑斯蒂函数如下所示：

logistic回归公式: $Y\text{的概率} = e^{B_0+B_1} / 1+e^{B_0+B_1 x}$

如果你曾经在赛马比赛或世界杯中下过一些小小的赌注，那就可以通过赔率更好地理解这个概念。逻辑斯蒂函数可以通过公式Probability($Y$)/(1 – Probability($Y$))转化为赔率。举例来说，如果巴西队赢得世界杯的概率是20%，那么赔率就是0.2/(1 – 0.2)，等于0.25，用赔率的话讲就是“1赔4”。

要从赔率回到概率，就要用赔率除以1加赔率的和。在世界杯的例子中，就是0.25/(1 + 0.25)，等于20%。此外，我们再考虑一下优势比。假设德国队赢得世界杯的赔率是0.18，我们可以通过优势比来比较巴西队和德国队的赔率。在本例中，优势比为巴西队的赔率除以德国队的赔率。可以得到优势比为0.25/0.18，等于1.39。这样，我们可以说巴西队赢得世界杯的可能性是德国队的1.39倍。

有一种方法可以看出逻辑斯蒂回归和线性回归之间的联系，逻辑斯蒂回归就是以对数发生比为响应变量进行线性拟合，即$\log(P(Y)/1 - P(Y)) = B_0 + B_1 x$。这里的系数是通过极大似然估计得到的，而不是通过OLS。极大似然的直观意义就是，我们要找到一对$B_0$和$B_1$的估计值，使它们产生的对观测的预测概率尽可能接近$Y$的实际观测结果，这就是所谓的似然性。与其他软件一样，R语言也可以实现极大似然估计，通过找到最优的$B$值组合来使似然性最大化。

请记住，逻辑斯蒂回归是一项强大的技术，可以预测分类问题，通常也是对分类问题进行建模的起点。因此，对于本章中将要面临的商业问题，我们使用逻辑斯蒂回归作为解决问题的首选方法。

### 3.2.1　业务理解

威斯康星大学的威廉·沃尔伯格博士1990年发布了威斯康星乳腺癌数据集。他收集数据的目标是想辨别肿瘤活体检查结果是良性的还是恶性的。他的团队使用**细针穿刺**（FNA）技术收集样本。如果医生通过检查发现肿瘤或者通过透视发现异常组织，下一步就是进行活体检查。FNA是取得活体组织的安全方法，很少出现并发症。病理学家对活体组织进行检查，试图确定诊断结果

（恶性或良性）。正如你所想像的，诊断结果并非无足轻重。良性的乳腺肿瘤并不危险，因为不存在异常组织扩散到身体其他部分的风险。如果良性肿瘤过大，就需要通过外科手术切除。从另一方面来说，恶性肿瘤必须进行医疗干预，治疗的程度与很多因素有关，一般都要进行外科手术，然后进行放射性治疗或化学治疗。所以，误诊造成的后果是很严重的。误诊为恶性（false positive，假阳性）会导致昂贵但不必要的治疗费用，还会使患者背负巨大的心理和生理负担。另一方面，误诊为良性（false negative，假阴性）会使患者得不到应有的治疗，造成癌细胞扩散，引起过早死亡。乳腺癌患者的早期医疗干预可以大大提高存活率。



我们的任务是开发尽可能准确的机器学习诊断算法，帮助医疗团队确定肿瘤性质。

### 3.2.2 数据理解和数据准备

数据集包含699个患者的组织样本，保存在有11个变量的数据框中，如下所示。

- ID：样本编码
- V1：细胞浓度
- V2：细胞大小均匀度
- V3：细胞形状均匀度
- V4：边缘黏着度
- V5：单上皮细胞大小
- V6：裸细胞核（16个观测值缺失）
- V7：平和染色质
- V8：正常核仁
- V9：有丝分裂状态
- class：肿瘤诊断结果，良性或恶性；这就是我们要预测的结果变量。

医疗团队对9个特征进行了评分和编码，评分的范围是1~10。

可以在R的MASS包中找到该数据框，名为biopsy。为进行数据准备，我们加载这个数据框，确认数据结构，将变量重命名为有意义的名称，还要删除有缺失项的观测，然后即可开始对数据进行可视化探索。下面就是我们开始工作的代码，首先加载库文件和数据集，然后使用str()函数检查数据内部结构。如下所示：

```
    > library(MASS)
> data(biopsy)

> str(biopsy)
'data.frame':   699 obs. of   11 variables:
$ ID    : chr  "1000025" "1002945" "1015425"
       "1016277" ...
$ V1    : int  5 5 3 6 4 8 1 2 2 4 ...
```

```
$ V2    : int  1 4 1 8 1 10 1 1 1 2 ...
$ V3    : int  1 4 1 8 1 10 1 2 1 1 ...
$ V4    : int  1 5 1 1 3 8 1 1 1 1 ...
$ V5    : int  2 7 2 3 2 7 2 2 2 2 ...
$ V6    : int  1 10 2 4 1 10 10 1 1 1 ...
$ V7    : int  3 3 3 3 3 9 3 3 1 2 ...
$ V8    : int  1 2 1 7 1 7 1 1 1 1 ...
$ V9    : int  1 1 1 1 1 1 1 1 5 1 ...
$ class: Factor w/ 2 levels "benign","malignant": 1 1 1 1 1 2 1 1
    1 1 ...
```

对数据结构的检查发现，特征是整数型变量，结果变量是一个因子，不需要将数据转换为其他结构。

我们可以删除ID列，如下所示：

```
> biopsy$ID = NULL
```

接着重命名变量，并确认操作结果：

```
> names(biopsy) <- c("thick", "u.size", "u.shape",
   "adhsn", "s.size", "nucl", "chrom", "n.nuc",
       "mit", "class")
> names(biopsy)
[1] "thick"   "u.size" "u.shape" "adhsn"
   "s.size"  "nucl"
   "chrom"   "n.nuc"
[9] "mit"     "class"
```

现在，我们要删除那些有缺失数据的观测。既然只有16个观测有缺失数据，所以删除它们不会有什么危险，因为它们只占所有观测的2%。我们不会在本章对如何处理缺失数据做全面讨论，你可以参考附录A，我在那里介绍了如何对数据进行处理。通过删除这些观测，我们建立了一个新的工作数据框。使用na.omit()函数后，一行代码就可以巧妙删除所有有缺失数据的观测。

```
> biopsy.v2 <- na.omit(biopsy)
```

根据你分析数据时使用的R包的不同，结果变量也可能要求是数值型，比如0或者1。为了满足这个要求，可以创建一个变量y，用0表示良性，用1表示恶性，然后使用ifelse()函数为y赋值，如下所示：

```
> y <- ifelse(biopsy$class == "malignant", 1, 0)
```

在分类问题中，我们可以通过很多方式来可视化地理解数据，我认为用什么方式都是个人的选择。在本例中，我喜欢做的一件事是检查各个特征的箱线图，并将这些箱线图按照分类结果并列排列。如果想知道哪个特征对于算法可能是重要的，那么这是一种特别好的方法，可以一目了然地看出数据的分布特点。根据我的经验，箱线图还提供了一种有效的故事表达方式，你可以直接展示给客户。有很多方法可以快速画出箱线图，lattice包和ggplot2包都可以非常好地完成这个任务。在本例中，我会使用ggplot2包以及一个扩展包reshape2。加载这些程序包之后，你需要用

dplyr包中的group_by()函数有相同功能

melt()函数建立一个数据框，这样做是为了在特征融合后就建立一个箱线图矩阵，使我们可以轻松地进行下面的可视化检查：

```
    > library(reshape2)
> library(ggplot2)
```

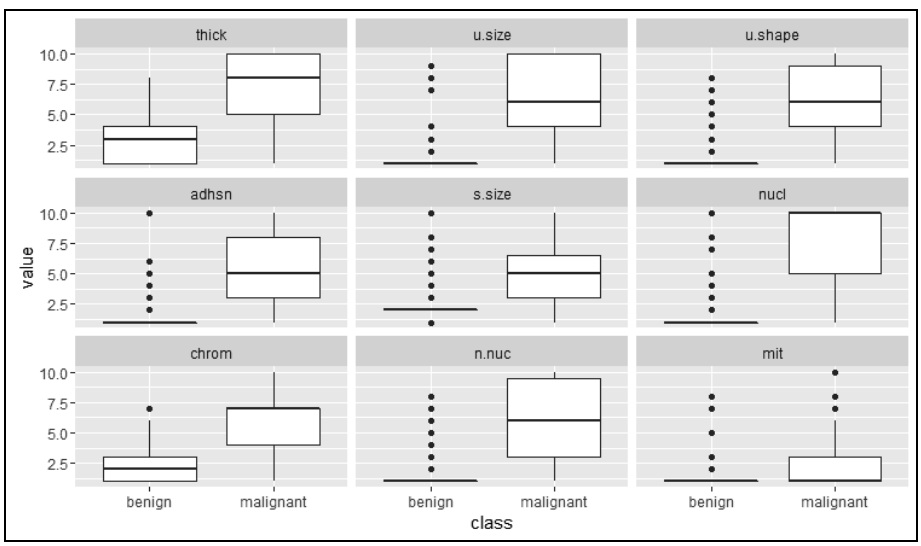
下面的代码将数据按值融合成一个总体特征，并按照class变量进行分组：

```
> biop.m <- melt(biopsy.v2, id.var = "class")
```

通过功能强大的ggplot2，我们可以建立一个3×3的箱线图矩阵，如下所示：

```
> ggplot(data = biop.m, aes(x = class, y = value))
+ geom_boxplot() + facet_wrap(~ variable, ncol = 3)
```

上述代码输出如下。




我们怎么解释箱线图呢？首先，在上面的屏幕截图中，白色矩形表示数据的上四分位数和下四分位数。换句话说，有一半的观测落在白色矩形范围内。白色矩形中间的黑色粗线段表示中位数。从矩形延伸出的直线与四分位距有关，如果没有离群点，则会终止于数据的最大值和最小值。黑点表示离群点。

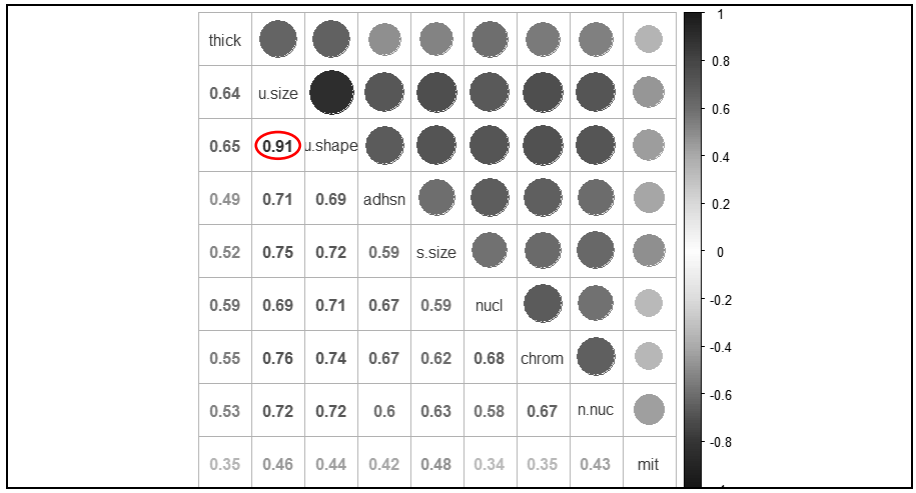
通过对上面箱线图的研究和判断，我们发现，很难确定哪个特征在分类算法中是重要的。但是，从中位数的间隔距离和相关分布来看，我觉得可以有把握地认为nuclei是一个重要特征。与之相反，不同class组的mitosis特征的中位数几乎没有区别，这说明它很可能是个无关特征。下面我们就来看一下！

既然所有特征都是定量的，我们就可以像线性回归一样进行相关性分析。正如我们在线性回

归中讨论过的，逻辑斯蒂回归中的共线性也会使估计发生偏离。加载corrplot包，检查特征之间的相关性，就像在前一章中做的那样。这次使用另一种类型的相关性矩阵图，其中既有渐变椭圆，也有相关系数。如下所示：

```
    > library(corrplot)
> bc <- cor(biopsy.v2[, 1:9]) #create an object of
       the features
> corrplot.mixed(bc)
```

上述代码输出如下。




从相关系数可以看出，我们会遇到共线性问题，特别是细胞大小均匀度和细胞形状均匀度表现出非常明显的共线性。作为逻辑斯蒂回归建模过程的一部分，VIF分析是必不可少的，正如我们在线性回归中所做的那样。数据准备的最终任务是建立训练数据集和测试数据集。从原始数据中建立两个不同的数据集的目的在于，这样可以更准确地预测未使用数据或未知数据。

实际上，在机器学习中，我们不应该过于关心对已使用的观测的预测有多好，而应该把重点放在那些建立算法时没有使用过的观测的预测上面。所以，我们要使用训练数据建立并选择一个对测试数据能做出最好预测的最优算法。在本章中，我们会遵循这个评价标准来建立模型。

有多种方式可以将数据恰当地划分成训练集和测试集：50/50、60/40、70/30、80/20，诸如此类。你应该根据自己的经验和判断选择数据划分方式。在本例中，我选择按照70/30的比例划分数据。如下所示：

```
    > set.seed(123) #random number generator
> ind <- sample(2, nrow(biopsy.v2), replace = TRUE,
       prob = c(0.7, 0.3))
> train <- biopsy.v2[ind==1, ] #the training data
       set
```

```
> test <- biopsy.v2[ind==2, ] #the test data set
> str(test) #confirm it worked
'data.frame':   209 obs. of  10 variables:
 $ thick  : int  5 6 4 2 1 7 6 7 1 3 ...
 $ u.size : int  4 8 1 1 1 4 1 3 1 2 ...
 $ u.shape: int  4 8 1 2 1 6 1 2 1 1 ...
 $ adhsn  : int  5 1 3 1 1 4 1 10 1 1 ...
 $ s.size : int  7 3 2 2 1 6 2 5 2 1 ...
 $ nucl   : int  10 4 1 1 1 1 1 10 1 1 ...
 $ chrom  : int  3 3 3 3 3 4 3 5 3 2 ...
 $ n.nuc  : int  2 7 1 1 1 3 1 4 1 1 ...
 $ mit    : int  1 1 1 1 1 1 1 4 1 1 ...
 $ class  : Factor w/ 2 levels benign","malignant":
       1 1 1 1 1 2 1
       2 1 1 ...
```

为了确保两个数据集的结果变量是均衡的，我们做以下检查：

```
    > table(train$class)
   benign malignant
     302       172
> table(test$class)
   benign malignant
      142       67
```

两个数据集的结果变量的内部比例完全可以接受，既然如此，下面开始进行模型构建和模型评价。

### 3.2.3 模型构建与模型评价

在流程的这一部分，我们首先用所有输入变量建立一个逻辑斯蒂回归模型，然后用最优子集法对特征进行削减。在此之后，我们会试验**判别分析**和**多元自适应回归样条**（**MARS**）方法。

#### 1. 逻辑斯蒂回归模型

我们已经讨论了逻辑斯蒂回归背后的理论，现在可以开始拟合模型了。R中的`glm()`函数可以拟合广义线性模型，这是一系列模型，其中包括逻辑斯蒂回归模型。代码的语法和前一章中用过的`lm()`函数很相似，其中一个较大区别是我们必须在函数中使用`family = binomial`这个参数。该参数告诉R运行逻辑斯蒂回归，而不是其他版本的广义线性模型。首先在训练数据集上使用所有特征建立一个模型，看看这个模型在测试数据集上运行的效果。如下所示：

```
    > full.fit <- glm(class ~ ., family = binomial,
      data = train)
> summary(full.fit)
Call:
glm(formula = class ~ ., family = binomial, data =
      train)
Deviance Residuals:
```

```
    Min       1Q    Median      3Q       Max
-3.3397  -0.1387   -0.0716  0.0321    2.3559
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -9.4293     1.2273  -7.683 1.55e-14
      ***
thick         0.5252     0.1601   3.280 0.001039 **
u.size       -0.1045     0.2446  -0.427 0.669165
u.shape       0.2798     0.2526   1.108 0.268044
adhsn         0.3086     0.1738   1.776 0.075722 .
s.size        0.2866     0.2074   1.382 0.167021
nucl          0.4057     0.1213   3.344 0.000826
      ***
chrom         0.2737     0.2174   1.259 0.208006
n.nuc         0.2244     0.1373   1.635 0.102126
mit           0.4296     0.3393   1.266 0.205402
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05
      '.' 0.1 ' ' 1
(Dispersion parameter for binomial family taken to
       be 1)
    Null deviance: 620.989  on 473  degrees of
       freedom
Residual deviance:  78.373  on 464  degrees of
       freedom
AIC: 98.373
Number of Fisher Scoring iterations: 8
```

通过summary()函数，我们可以查看各个预测变量的系数及其*p*值。可以看到，只有两个特征的*p*值小于0.05（thickness和nuclei）。使用confint()函数可以对模型进行95%置信区间的检验。如下所示：

```
    > confint(full.fit)
                    2.5 %     97.5 %
(Intercept) -12.23786660 -7.3421509
thick         0.23250518  0.8712407
u.size       -0.56108960  0.4212527
u.shape      -0.24551513  0.7725505
adhsn        -0.02257952  0.6760586
s.size       -0.11769714  0.7024139
nucl          0.17687420  0.6582354
chrom        -0.13992177  0.7232904
n.nuc        -0.03813490  0.5110293
mit          -0.14099177  1.0142786
```

请注意，两个显著特征的置信区间不包括0。对于逻辑斯蒂回归模型的系数，你不能解释为“当*X*改变1个单位时*Y*会改变多少”。这时，优势比的作用就体现出来了。对数函数$\log(P(Y)/1-P(Y))=B_0+B_1x$的*B*系数可以通过exponent(beta)指数函数转化为优势比。

要在R中计算优势比，可以使用下面的exp(coef())函数形式：

odd ratio

```
    > exp(coef(full.fit))
 (Intercept)        thick       u.size      u.shape
     adhsn
8.033466e-05 1.690879e+00 9.007478e-01 1.322844e+00
     1.361533e+00
s.size         nucl        chrom        n.nuc  mit
1.331940e+00 1.500309e+00 1.314783e+00 1.251551e+00
    1.536709e+00
```

优势比可以解释为特征中1个单位的变化导致的结果发生比的变化。如果系数大于1，就说明当特征的值增加时，结果的发生比会增加。反之，系数小于1就说明，当特征的值增加时，结果的发生比会减小。在本例中，除u.size之外的所有特征都会增加对数发生比。

我们进行数据探索时发现一个问题——潜在的多重共线性。同线性回归一样，在逻辑斯蒂回归中也可以算出VIF统计量，如下所示：

```
    > library(car)
> vif(full.fit)
   thick u.size u.shape adhsn s.size nucl
     chrom  n.nuc
1.2352 3.2488 2.8303 1.3021 1.6356 1.3729
      1.5234 1.3431
  mit
1.059707
```

没有一个VIF值大于5，根据VIF经验法则，共线性看来不成为一个问题。下面的任务就是进行特征选择，先别急，我们先编写一些代码，看看模型在训练数据集和测试数据集上表现如何。

首先要建立一个向量，表示预测概率，如下所示：

```
    > train.probs <- predict(full.fit, type =
       "response")
> train.probs[1:5] #inspect the first 5 predicted
       probabilities
[1] 0.02052820 0.01087838 0.99992668 0.08987453
       0.01379266
```

下一步需要评价模型在训练集上执行的效果，然后再评价它在测试集上的拟合程度。快速实现评价的方法是生成一个混淆矩阵。在后面的章节中，我们使用的混淆矩阵是由caret包实现的，InformationValue包也可以实现混淆矩阵。这时，我们需要用0和1来表示结果。函数区别良性结果和恶性结果使用的默认值是0.50，也就是说，当概率大于等于0.50时，就认为这个结果是恶性的：

```
> trainY <- y[ind==1]
> testY <- y[ind==2]
> confusionMatrix(trainY, train.probs)
    0   1
0 294   7
1   8 165
```

预测 预测 实际 实际

矩阵的行表示预测值，列表示实际值。对角线上的元素是预测正确的分类。右上角的值7表示

误为恶性的预测数，左下角的值8表示误为良性的预测数。可以查看错误预测的比例。如下所示：

```
> misClassError(trainY, train.probs)
[1] 0.0316
```

看上去我们干得相当不错，训练集上只有3.16%的预测错误率。如前所述，我们必须正确预测未知数据，换句话说，必须正确预测测试集。

在测试集上建立混淆矩阵的方法和在训练集上一样：

```
> test.probs <- predict(full.fit, newdata = test,
   type = "response")
> misClassError(testY, test.probs)
[1] 0.0239
> confusionMatrix(testY, test.probs)
    0   1
0 139   2
1   3  65
```

(2+3)/(139+2+3+65)=0.0239

看上去，我们使用全部特征建立的模型效果非常好，差不多98%的预测正确率真是让人激动。但是，我们还是要看看是否有改进的空间。想象一下，假如你或你的亲人是被误诊的患者，会怎么样？就像前面提到的，后果很严重。出于这种考虑，还可能有更好的方式来建立分类算法吗？下面我们就来看一下！

### 2. 使用交叉验证的逻辑斯蒂回归

k-fold cross-validation

交叉验证的目的是提高测试集上的预测正确率，以及尽可能避免过拟合。**K折交叉验证**的做法是将数据集分成$K$个相等的等份，每个等份称为一个**K子集**（K-set）。算法每次留出一个子集，使用其余$K-1$个子集拟合模型，然后用模型在留出的那个子集上做预测。将上面$K$次验证的结果进行平均，可以使误差最小化，并且获得合适的特征选择。你也可以使用**留一交叉验证**方法，这里的$K$等于$N$。模拟表明，LOOCV可以获得近乎无偏的估计，但是会有很高的方差。所以，大多数机器学习专家都建议将$K$的值定为5或10。

bestglm包可以自动进行交叉验证，这个包依赖于我们在线性回归中使用过的leaps包。交叉验证的语法和数据格式存在注意事项，所以我们按部就班地进行：

```
    > library(bestglm)
Loading required package: leaps
```

加载程序包之后，需要将结果编码成0或1。如果结果仍为因子，程序将不起作用。使用这个程序包的另外一个要求是结果变量（或$y$）必须是最后一列，而且要删除所有没有用的列。通过以下代码新建一个数据框即可快速实现上述要求：

```
> X <- train[, 1:9]
> Xy <- data.frame(cbind(X, trainY))
```

以下代码使用数据进行交叉验证：

```
> bestglm(Xy = Xy, IC="CV",
  CVArgs=list(Method="HTF", K=10,
  REP=1), family=binomial)
```

在上面的代码中，Xy = Xy指的是我们已经格式化的数据框，IC = "CV"告诉程序使用的信息准则为交叉验证，CVArgs是我们要使用的交叉验证参数。HTF方法就是K折交叉验证，后面的数字K = 10指定了均分的份数，参数REP = 1告诉程序随机使用等份并且只迭代一次。像glm()函数一样，我们需要指定参数family = binomial。顺便说一句，如果指定参数family = gaussian，你就可以使用bestglm()函数进行线性回归。完成上面的交叉验证分析之后，会得到下面的结果：Best Model有三个特征，它们是thick、u.size和nucl。Morgan-Tatar搜索的结果表明，程序对所有可能的子集进行了简单而彻底的搜索。如下所示：

```
    Morgan-Tatar search since family is non-gaussian.
CV(K = 10, REP = 1)
BICq equivalent for q in (7.16797006619085e-05,
    0.273173435514231)
Best Model:
Estimate Std. Error   z value     Pr(>|z|)
(Intercept) -7.8147191 0.90996494 -8.587934
     8.854687e-18
thick        0.6188466 0.14713075  4.206100
     2.598159e-05
u.size       0.6582015 0.15295415  4.303260
     1.683031e-05
nucl         0.5725902 0.09922549  5.770596
     7.899178e-09
```

我们可以把这些特征放到glm()函数中，看看模型在测试集上表现如何。predict()函数不能用于bestglm生成的模型，所以下面的步骤是必需的：

```
> reduce.fit <- glm(class ~ thick + u.size + nucl,
   family = binomial, data = train)
```

同前面一样，下面的代码可以在测试集上比较预测值和实际值：

```
> test.cv.probs <- predict(reduce.fit, newdata =
   test, type = "response")
> misClassError(testY, test.cv.probs)
[1] 0.0383
> confusionMatrix(testY, test.cv.probs)
    0    1
0 139    5
1   3   62
```

精简了特征的模型和全特征模型相比，精确度略有下降，但这并不是“世界末日”。我们可以用bestglm包再试一次，这次使用信息准则为BIC的最优子集方法：

```
    > bestglm(Xy = Xy, IC = "BIC", family = binomial)
Morgan-Tatar search since family is non-gaussian.
BIC
```

```
BICq equivalent for q in (0.273173435514231,
      0.577036596263757)
Best Model:
 Estimate Std. Error    z value     Pr(>|z|)
(Intercept) -8.6169613  1.03155250 -8.353391
      6.633065e-17
thick        0.7113613  0.14751510  4.822295
      1.419160e-06
adhsn        0.4537948  0.15034294  3.018398
      2.541153e-03
nucl         0.5579922  0.09848156  5.665956
      1.462068e-08
n.nuc        0.4290854  0.11845720  3.622282
      2.920152e-04
```

试过所有可能的子集之后，以上4个特征提供了最小的BIC评分。我们看看这个模型在测试集上的预测效果，如下所示：

```
    > bic.fit <- glm(class ~ thick + adhsn + nucl +
       n.nuc, family = binomial, data = train)
> test.bic.probs <- predict(bic.fit, newdata =
       test, type = "response")
    > misClassError(testY, test.bic.probs)
    [1] 0.0239
    > confusionMatrix(testY, test.bic.probs)
        0    1
    0 138    1
    1   4   66
```

我们得到5个错误的预测，和全特征模型一样。那么问题来了：哪一个模型更好？在任何正常情况下，如果具有相同的泛化效果，经验法则会选择最简单的或解释性最好的模型。我们当然可以开启一个全新的分析，使用新的随机数以及新的训练集和测试集划分比例。但是，不妨假定我们已经做了逻辑斯蒂回归能做的所有操作，最后还是回到了全特征模型和BIC最小模型，仍然需要讨论模型选择的方法。下面讨论判别分析的方法，在最后的模型推荐中，我们有可能选择使用这种方法建立的模型。

## 3.3　判别分析概述 Discriminant Analysis

**判别分析**又称**费舍尔判别分析**，也是一项常用的分类技术。当分类很确定时，判别分析可以有效替代逻辑斯蒂回归。当你遇到分类结果很确定的分类问题时，逻辑斯蒂回归的估计结果可能是不稳定的，即置信区间很宽，不同样本之间的估计值会有很大变化（James，2013）。判别分析不会受到这个问题的困扰，实际上，它会比逻辑回归做得更好，泛化能力更强。反之，如果特征和结果变量之间具有错综复杂的关系，判别分析在分类任务上的表现就会非常差。在乳腺癌这个例子中，逻辑斯蒂回归在训练集和测试集上的表现都非常好，分类的结果并不确定。出于同逻辑回归进行比较的目的，我们研究一下判别分析，包括**线性判别分析**和**二次判别分析**。

LDA: Linear Discriminant Analysis　QDA: Quadratic Discriminant Analysis

判别分析使用**贝叶斯定理**确定每个观测属于某个类别的概率。如果你有两个类别，比如良性和恶性，判别分析会计算观测分别属于两个类别的概率，然后选择高概率的类别作为正确的类别。

贝叶斯定理定义了在$X$已经发生的条件下$Y$发生的概率——等于$Y$和$X$同时发生的概率除以$X$发生的概率，公式如下：

$$Y / X\text{的概率} = \frac{\mathrm{P}(X+Y)}{\mathrm{P}(X)}$$

公式中的分子表示一个具有某些特征的观测属于某个分类水平的可能性，分母表示一个具有这些特征的观测属于所有分类水平的可能性。同样地，分类原则认为如果$X$和$Y$的联合分布已知，那么给定$X$后，决定观测属于哪个类别的最佳决策是选择那个有更大概率（后验概率）的类别。

获得后验概率的过程如下所示。

(1) 收集已知类别的数据。
(2) 计算先验概率——代表属于某个类别的样本的比例。
(3) 按类别计算每个特征的均值。
(4) 计算每个特征的方差–协方差矩阵。在线性判别分析中，这会是一个所有类别的混合矩阵，给出线性分类器；在二次判别分析中，会对每个分类建立一个方差–协方差矩阵。
(5) 估计每个分类的正态分布（高斯密度）。
(6) 计算`discriminant`函数，作为一个新对象的分类原则。
(7) 根据`discriminant`函数，将观测分配到某个分类。

我们还可以给出确定后验概率的扩展表示形式，如下所示。

- $\pi_k$ = 分类$k$中的样本数/总体样本数，是一个随机选择的观测属于第$k$个分类的先验概率。
- $f_k(X) = \mathrm{P}(X = x \mid Y = k)$ 是一个观测属于第$k$个分类的概率密度函数。假设概率服从正态分布（高斯分布）。如果有多个特征，假设概率服从多元高斯分布。
- 通过$p_k(X)$ = $Y$在给定$X$时的概率，我们按以下方式调整贝叶斯定理。$-p_x(X) = \pi_k f_k(X) / \sum_{l=1}^{k} \pi_l f_l(X)$。这是给定一个观测的特征值时，这个观测属于第$k$个分类的后验概率。
- 假设$k = 2$并且先验概率相同，$\pi_1 = \pi_2$，则当$2x(\mu_1 - \mu_2) > \mu_1^2 - \mu_2^2$时，观测值被分配到第一个分类中，否则就被分配到第二个分类。这个公式称为决策边界。判别分析会生成$k - 1$个决策边界，也就是说，如果有三个类别（$k = 3$），那么就会有两个决策边界。

尽管线性判别分析简单而又优雅，它仍然具有局限性。线性判别分析假设每种类别中的观测服从多元正态分布，并且不同类别之间的具有同样的协方差。二次判别分析仍然假设观测服从正态分布，但假设每种类别都有自己的协方差。

为什么这种假设很重要？当你放宽相同协方差假设，就意味着允许二次项进入判别分数的计

算，这在线性判别分析中是不可能的。其中的数学意义理解起来有些挑战性，已经超出了本书的讨论范围。重要的是要记住，二次判别分析技术比逻辑斯蒂回归更具灵活性，同时还要时刻牢记偏差—方差权衡的问题。使用更有灵活性的技术可以得到偏差更小的结果，但很可能具有更高的方差。和很多灵活的技术一样，需要一个高鲁棒性的训练数据集来降低高分类方差。

## 判别分析应用

线性判别分析（LDA）可以用MASS包实现，我们为了使用biopsy数据集已经加载了这个包。LDA的语法和`lm()`以及`glm()`函数非常相似。

现在可以开始LDA模型的拟合了，如下所示：

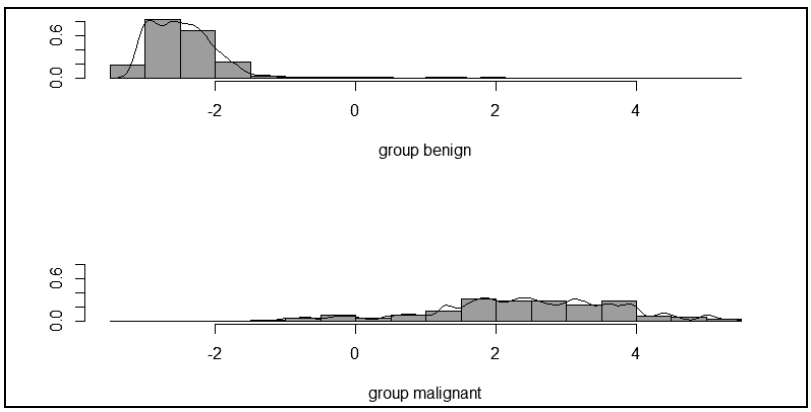
```
    > lda.fit <- lda(class ~ ., data = train)
> lda.fit
Call:
lda(class ~ ., data = train)
Prior probabilities of groups: 分组先验概率
   benign malignant
0.6371308 0.3628692
Group means:
    thick  u.size u.shape   adhsn  s.size    nucl
      chrom
benign    2.9205 1.30463 1.41390 1.32450 2.11589
      1.39735 2.08278
malignant 7.1918 6.69767 6.68604 5.66860 5.50000
      7.67441 5.95930
           n.nuc     mit
benign    1.22516 1.09271
malignant 5.90697 2.63953
Coefficients of linear discriminants:
               LD1
Thick    0.19557291
u.size   0.10555201
u.shape  0.06327200
adhsn    0.04752757
s.size   0.10678521
nucl     0.26196145
chrom    0.08102965
n.nuc    0.11691054
mit     -0.01665454
```

从结果可以看出，在分组先验概率中，良性概率大约为64%，恶性概率大约为36%。下面再看看分组均值，这是按类别分组的每个特征的均值。线性判别系数是标准线性组合，用来确定观测的判别评分的特征。评分越高，越可能被分入恶性组。

对LDA模型使用`plot()`函数，可以画出判别评分的直方图和密度图，如下所示：

```
> plot(lda.fit, type = "both")
```

上述命令输出如下。




可以看出，组间有些重合，这表明有些观测被错误分类。

LDA模型可以用predict()函数得到3种元素（class、posterior和x）的列表。class元素是对良性或恶性的预测，posterior是值为x的评分可能属于某个类别的概率，x是线性判别评分。通过下面的函数，我们仅提取恶性观测的概率：

```
    > train.lda.probs <- predict(lda.fit)$posterior[,
      2]
> misClassError(trainY, train.lda.probs)
    [1] 0.0401
    > confusionMatrix(trainY, train.lda.probs)
        0    1
    0 296   13
    1   6  159
```

很不幸，我们的LDA模型在训练集上表现得比逻辑斯蒂回归模型差多了。但最重要的是LDA模型在测试集上的表现，如下所示：

```
> test.lda.probs <- predict(lda.fit, newdata =
   test)$posterior[, 2]
> misClassError(testY, test.lda.probs)
[1] 0.0383
> confusionMatrix(testY, test.lda.probs)
    0    1
0 140    6
1   2   61
```

基于LDA模型在训练集上的糟糕表现，它在测试集上的表现比我预期的要好。从正确分类率的角度看，LDA模型表现得依然不如逻辑斯蒂回归模型（LDA模型：96%，逻辑斯蒂回归模型：98%）。

下面用二次判别分析（QDA）模型来拟合数据。在R中，QDA也是MASS包的一部分，函数为qda()。建模过程直截了当，我们将模型存储在一个名为qda.fit的对象中，如下页所示：

```
    > qda.fit = qda(class ~ ., data = train)
> qda.fit
Call:
qda(class ~ ., data = train)
Prior probabilities of groups:
   benign malignant
0.6371308 0.3628692
Group means:
    Thick u.size u.shape adhsn s.size   nucl  chrom
     n.nuc
benign    2.9205 1.3046 1.4139 1.3245 2.1158
      1.3973 2.0827 1.2251
malignant 7.1918 6.6976 6.6860 5.6686 5.5000
      7.6744 5.9593 5.9069
               mit
benign    1.092715
malignant 2.639535
```

结果中有分组均值，和LDA一样；但是没有系数，因为这是二次函数，我们前面讨论过了。

使用与LDA相同的代码，可以得到QDA模型在训练集和测试集上的预测结果：

```
> train.qda.probs <- predict(qda.fit)$posterior[,
  2]
> misClassError(trainY, train.qda.probs)
[1] 0.0422
> confusionMatrix(trainY, train.qda.probs)
    0    1
0 287    5
1  15  167
> test.qda.probs <- predict(qda.fit, newdata =
  test)$posterior[, 2]
> misClassError(testY, test.qda.probs)
[1] 0.0526
> confusionMatrix(testY, test.qda.probs)
    0    1
0 132    1
1  10   66
```

根据混淆矩阵可以立即断定，QDA模型在训练数据集上表现得最差。QDA在测试集上的分类效果也很差，有11个预测错误，而误为恶性的比例尤其高。

## 3.4　多元自适应回归样条方法 Multivariate Adaptive Regression Splines, MARS

如果有一种建模技术具有以下特点，你是不是会喜出望外？

- ❑ 对于回归和分类问题，都可以灵活建立线性模型和非线性模型。
- ❑ 支持变量的交互项。
- ❑ 简单易懂，易于解释。

- 几乎不需要数据预处理。
- 可以处理所有类型的数据：数值型、因子等。
- 可以很好地预测未知数据，也就是说，在偏差–方差的权衡方面做得非常好。

如果这些特点都让你心动，那么就根本不用我再向你推荐MARS模型了。几个月前我注意到了这种方法，而且发现这种方法的效果出奇地好。实际上，在我最近的工作中，这种方法在测试数据上的效果超过了随机森林和提升树。它立刻就成为了我的基准模型，其他模型都是用来和它做比较的。这种方法的另外一个优点是，它否定了我正在做的很多特征工程方面的工作。这些工作中，很多都要使用**证据权重法**（**WOE**）和**信息价值法**（**IV**）来捕获非线性并对变量进行重新编码。我本来是想花些篇幅介绍WOE和IV方法的，但经过一些测试，我发现这些技术能做的事情，MARS也做得非常好（比如捕获非线性），所以干脆就不再介绍WOE和IV方法了。



要理解MARS非常容易。首先，我们从一个前面已经讨论过的线性模型或广义线性模型开始。然后，为了捕获非线性关系，添加一个铰链（hinge）函数。这些铰链作用于输入特征，相当于系数的改变。举例来说，假设有这样一个模型：*Y*=12.5（截距）+1.5（变量1）+3.3（变量2），其中变量1和变量2的范围都在1和10之间。下面，我们看看变量2的铰链函数如何发挥作用：

$$Y = 11（新截距）+1.5（变量1）+ 4.26734(\max(0, 变量2 - 5.5))$$

于是，我们可以这样理解铰链函数：在0和变量2减去5.50的结果中，它找出其中的最大值。所以，只要变量2的值大于5.5，铰链函数的值就会乘以系数，否则它的值就是0。这种方法可以给每个变量配置多个铰链，也可以用于交互项。

MARS方法的另一个有趣之处是，它可以自动进行变量选择。这是通过交叉验证实现的，默认的方法是通过前向过程建模——这非常类似于前向逐步回归——然后通过后向过程精简模型。因为模型完成前向过程之后，有可能出现对数据的过拟合，所以要通过后向过程根据**广义交叉验证**对输入特征进行精简，并去除铰链。

$$GCV=RSS/N*(1 - 参数有效数量/N)^2$$
$$参数有效数量 = 输入特征数量 + 惩罚系数*(输入特征数量-1)/2$$

在earth包中，对于加法模型来说，惩罚系数=2；对于乘法模型，其中有交互项，惩罚系数=3。

在R中，你可以调整的参数非常少。在下面的例子中，我会演示一种简单而又有效实现MARS的方法。如果你的需求很多，那么可以学习Stephen Milborrow制作的非常棒的在线教程Notes on the earth package，通过以下链接可以学习MARS的更多灵活用法：

http://www.milbo.org/doc/earth-notes.pdf

介绍完MARS以后，我们开始学习其用法。你可以使用MDA包，但因为我是通过earth包学习的，所以我还是使用这个包。代码和前面的例子很相似，在那里我们使用了`glm()`函数。但需要

注意，一定要设定如何精简模型，并且使响应变量服从二元分布。我的设定是使用5折交叉验证来选择模型（pmethod="cv"，nfold = 5），重复3次（ncross = 3）；使用没有交互项的加法模型（degree = 1），每个输入特征只使用一个铰链函数（minspan = -1）。在我使用的数据中，交互项和多个铰链函数都会导致过拟合。当然，你的情况会有所不同。代码如下所示：

```
> library(earth)
> set.seed(1)
> earth.fit <- earth(class ~ ., data = train,
                     pmethod = "cv",
                     nfold = 5,
                     ncross = 3,
                     degree = 1,
                     minspan = -1,
                     glm=list(family=binomial)
                     )
```

下面查看模型摘要。乍一看，它有点让人摸不着头脑。当然，我们可以看到模型公式和逻辑斯蒂回归系数，还有铰链函数，再后面是一些注释和广义R方的数值，等等。MARS模型的生成过程是：先根据数据建立一个标准线性回归模型，它的响应变量在内部编码为0和1；对特征（变量）进行精简之后，生成最终模型，并将其转换为一个GLM模型。所以，我们可以不用考虑R方的值：

```
> summary(earth.fit)
Call: earth(formula=class~., data=train,
  pmethod="cv",
glm=list(family=binomial), degree=1, ncross=3,
  nfold=5, minspan=-1)
GLM coefficients
             malignant
(Intercept) -6.5746417
u.size 0.1502747
adhsn 0.3058496
s.size 0.3188098
nucl 0.4426061
n.nuc 0.2307595
h(thick-3) 0.7019053
h(3-chrom) -0.6927319
Earth selected 8 of 10 terms, and 7 of 9 predictors
  using pmethod="cv"
Termination condition: RSq changed by less than
  0.001 at 10 terms
Importance: nucl, u.size, thick, n.nuc, chrom,
  s.size, adhsn,
  u.shape-unused, ...
Number of terms at each degree of interaction: 1 7
  (additive model)
Earth GRSq 0.8354593 RSq 0.8450554 mean.oof.RSq
  0.8331308 (sd 0.0295)
GLM null.deviance 620.9885 (473 dof) deviance
  81.90976 (466 dof)
  iters 8
```

```
pmethod="backward" would have selected the same
  model:
8 terms 7 preds, GRSq 0.8354593 RSq 0.8450554
  mean.oof.RSq
  0.8331308
```

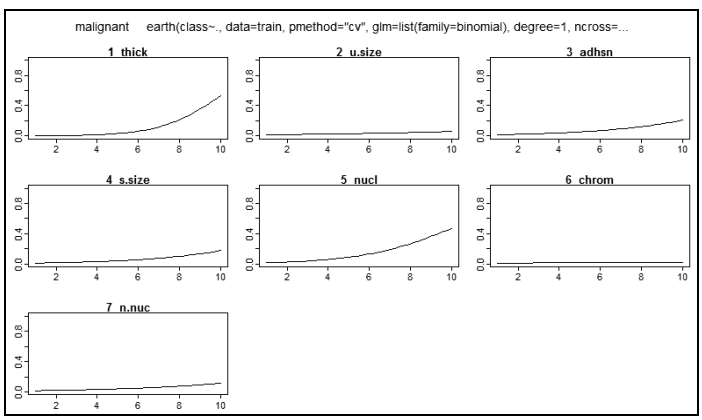
模型有8项，包括截距和7个预测变量。其中两个预测变量有铰链函数，这就是浓度和染色质变量。如果浓度大于3，就会用系数0.7019乘以铰链函数的值，否则这一项就是0。对于染色质，如果它的值小于3，那么就用系数乘以铰链函数值，否则这一项就是0。



还可以做出统计图。第一张图是用plotmo()函数生成的，它展示了保持其他预测变量不变，某个预测变量发生变化时，响应变量发生的改变。你可以清楚地看到铰链函数对浓度所起的作用：

```
> plotmo(earth.fit)
```

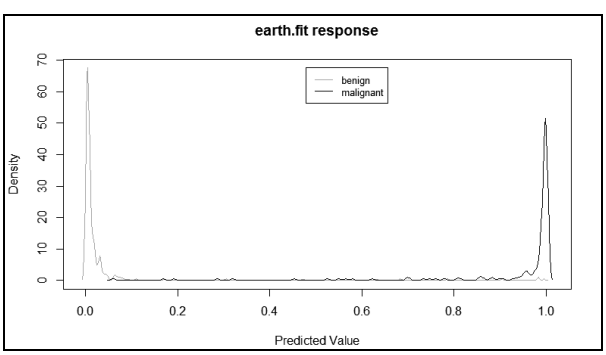
上述命令输出如下。




通过plotd()函数，可以生成按类别标签分类的预测概率密度图：

```
> plotd(earth.fit)
```

上述命令输出如下。



下面看看变量之间的相对重要性。先解释一下，nsubsets是精简过程完成之后包含这个变量的模型的个数。对于gcv和rss列，其中的值表示这个变量贡献的gcv和rss值的减少量（gcv和rss的范围都是0~100）：

```
> evimp(earth.fit)
        nsubsets   gcv   rss
nucl           7 100.0 100.0
u.size         6  44.2  44.8
thick          5  23.8  25.1
n.nuc          4  15.1  16.8
chrom          3   8.3  10.7
s.size         2   6.0   8.1
adhsn          1   2.3   4.6
```

我们再看一下模型在测试集上的表现：

```
> test.earth.probs <- predict(earth.fit, newdata =
  test, type = "response")
> misClassError(testY, test.earth.probs)
[1] 0.0287
> confusionMatrix(testY, test.earth.probs)
    0    1
0 138    2
1   4   65
```

这完全可以和逻辑斯蒂回归模型相比。下面比较前面讲过的各种模型，看看哪个模型是最好的选择。

## 3.5　模型选择

从上述的工作中，你能得到什么结论？我们从模型中计算出混淆矩阵和错误率，为的就是有一个依据，可是在选择分类模型时还是毫无头绪。对于分类模型的比较，**受试者工作特征**（**ROC**）图是一个很有效的工具。简言之，ROC基于分类器的性能对其进行可视化、组织和选择（Fawcett，2006）。在ROC图中，Y轴是**真阳性率**（**TPR**），X轴是**假阳性率**（**FPR**）。计算过程非常简单，如下所示。

TPR = 正确分类的阳性样本数/所有阳性样本数
FPR = 错误分类的阴性样本数/所有阴性样本数

如果将ROC画出来，可以生成一条曲线，然后可以算出**曲线下面积**了。AUC就是一项衡量分类器性能的有效指标，直观地说，AUC等于一个观测者在面对一对随机选出的案例（其中一个是阳性案例，一个是阴性案例）时，能正确识别出阳性案例的概率（Hanley JA，McNeil BJ，1982）。在我们的例子中，只要将观测者换成我们的算法即可依此进行评价。

要在R中画出ROC图，你可以使用ROCR包。我认为这是一个功能强大的包，它可以让你只

用3行代码就可以画出ROC图。这个程序包还配有一个网站，上面有示例和演示，你可以通过以下网址找到该网站：

http://rocr.bioinf.mpi-sb.mpg.de

下面我要展示4个不同的ROC图：全特征模型、使用BIC选择特征的简化模型、MARS模型和一个糟糕模型。这个所谓的糟糕模型只有1个预测特征，会和其他两个模型形成有效对比。因此，下面加载ROCR包，使用thick这个特征建立这个性能糟糕的模型。为简单起见，我们将其命名为bad.fit。如下所示：

```
    > library(ROCR)
> bad.fit <- glm(class ~ thick, family = binomial,
      data = test)
> test.bad.probs = predict(bad.fit, type =
      "response") #save
      probabilities
```

现在即可使用测试数据集绘制ROC图，每个模型3行代码。首先建立一个对象，保存对实际分类的预测概率，然后使用这个对象建立一个带有TPR和FPR的对象。在此之后，使用`plot()`函数绘制ROC图。我们从全特征模型（我喜欢称之为全模型）开始，它就是我们在3.2节建立的那个初始模型：

```
> pred.full <- prediction(test.probs, test$class)
```

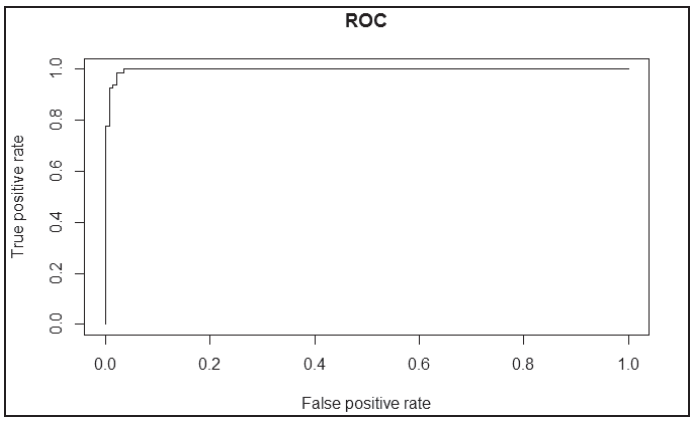
下面是带有TPR和FPR的performance对象：

```
> perf.full <- performance(pred.full, "tpr", "fpr")
```

下面使用`plot()`命令画图，图的标题设为`ROC`，参数`col = 1`指定曲线颜色为黑色：

```
> plot(perf.full, main = "ROC", col = 1)
```

上述命令输入如下。



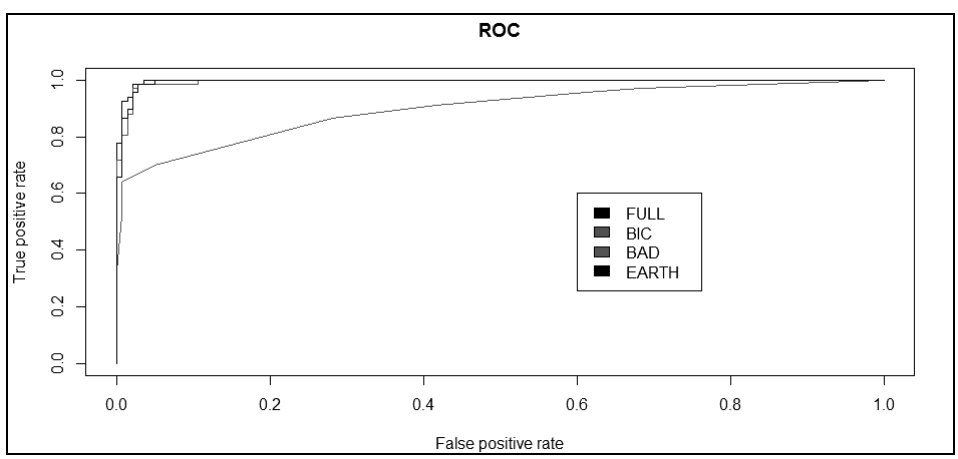
如前所述，曲线的Y轴表示TPR，X轴表示FPR。如果你有一个完美无缺的、没有假阳性错误的分类器，那么就会有一条从X轴0.0处垂直上升的线。如果模型的预测和随机选择没什么区别，那么就会有一条从左下角到右上角的对角线。回忆一下，全模型预测错了5个样本：3个假阳性和2个假阴性。为了便于比较，可以使用同样的代码加上其他模型。首先加上使用BIC建立的模型（参考3.2节），如下所示：

```
    > pred.bic <- prediction(test.bic.probs,
      test$class)
> perf.bic <- performance(pred.bic, "tpr", "fpr")
> plot(perf.bic, col = 2, add = TRUE)
```

plot()命令中的add = TURE将曲线加入现有图中。最后，加入那个表现糟糕的模型和MARS模型，并加入一个图例，如下所示：

```
    > pred.bad <- prediction(test.bad.probs,
      test$class)
> perf.bad <- performance(pred.bad, "tpr", "fpr")
> plot(perf.bad, col = 3, add = TRUE)
    > pred.earth <- prediction(test.earth.probs,
      test$class)
    > perf.earth <- performance(pred.earth, "tpr",
      "fpr")
    > plot(perf.earth, col = 4, add = TRUE)
    > legend(0.6, 0.6, c("FULL", "BIC", "BAD",
      "EARTH"), 1:4)
```

上述代码片段输出如下。




可以看到，全特征模型、BIC模型和MARS模型基本上重叠在一起。显而易见，**糟糕的**模型表现得和我们预想的一样差。

现在，我们能做的最后一件事就是计算AUC。这也可以通过ROCR包建立performance对象来

实现，只要将tpr和fpr换成auc即可。代码和输出如下所示：

```
> performance(pred.full, "auc")@y.values
[[1]]
[1] 0.9972672
> performance(pred.bic, "auc")@y.values
[[1]]
[1] 0.9944293
> performance(pred.bad, "auc")@y.values
[[1]]
[1] 0.8962056
> performance(pred.earth, "auc")@y.values
[[1]]
[1] 0.9952701
```

最高的AUC值是全模型的99.7%，BIC模型是99.4%，糟糕模型是89.6%，MARS模型是99.5%。所以，从各个方面来看，排除掉糟糕模型，其他几个模型在预测能力方面没有什么区别。我们该怎么做？一个简单的解决方案就是，将训练集和测试集重新随机化，把各种分析再做一遍，比如使用60/40的划分比例和一个不同的随机数种子。但是，如果我们还是得到相同的结果，又该怎么办？我认为，一个纯粹的统计学家会建议选择最简约的模型，但其他人可能更倾向于全特征模型。这就又归结到各种因素的权衡问题了，比如模型准确性与解释性、简约性与扩展性之间的权衡。在本章的例子中，我们完全可以选择更简单的模型，它具有和全模型一样的正确性。不用说你也能知道，仅通过GLM或判别分析不可能总有这样的预测效果。在接下来的章节中，我们会继续研究这个问题，使用一些更复杂的技术来提高模型的预测能力。机器学习的美妙之处就在于——“条条大路通罗马”。

## 3.6 小结

我们在本章研究了如何使用基于概率的线性模型预测定性响应变量，介绍了三种方法：逻辑斯蒂回归、判别分析和MARS。除此之外，还介绍了ROC图，这是一种可视化的有统计意义的模型选择技术。我们还简要讨论了需要注意的模型选择问题和权衡问题。在后续的章节中，我们还会使用乳腺癌数据集，看看更加复杂的技术在这个数据集上的应用效果。