R with statistical learning

杨钰萍（mailto:yangyupingentian@outlook.com）

2018年3月20日

前言

数据结构分为向量、矩阵、数组、数据框、因子和列表.请参考R语言实战和Introduction to R for Data Science(https://courses.edx.org/courses/course-v1:Microsoft+DAT204x+2T2017/course/)

如果你使用的不是RStudio，而是VS2017或者RGUI，关于RMarkdown的使用请参考《R语言实战》Ch22

参考：R语言实战、计量经济学、应用线性回归模型、实用多元统计分析

以下部分语句涉及到简单的函数编写及调用，涉及到的数据及包参见下表

R包	MASS	ISLR	基础包
数据集	Boston	Auto、smarket	state.x77

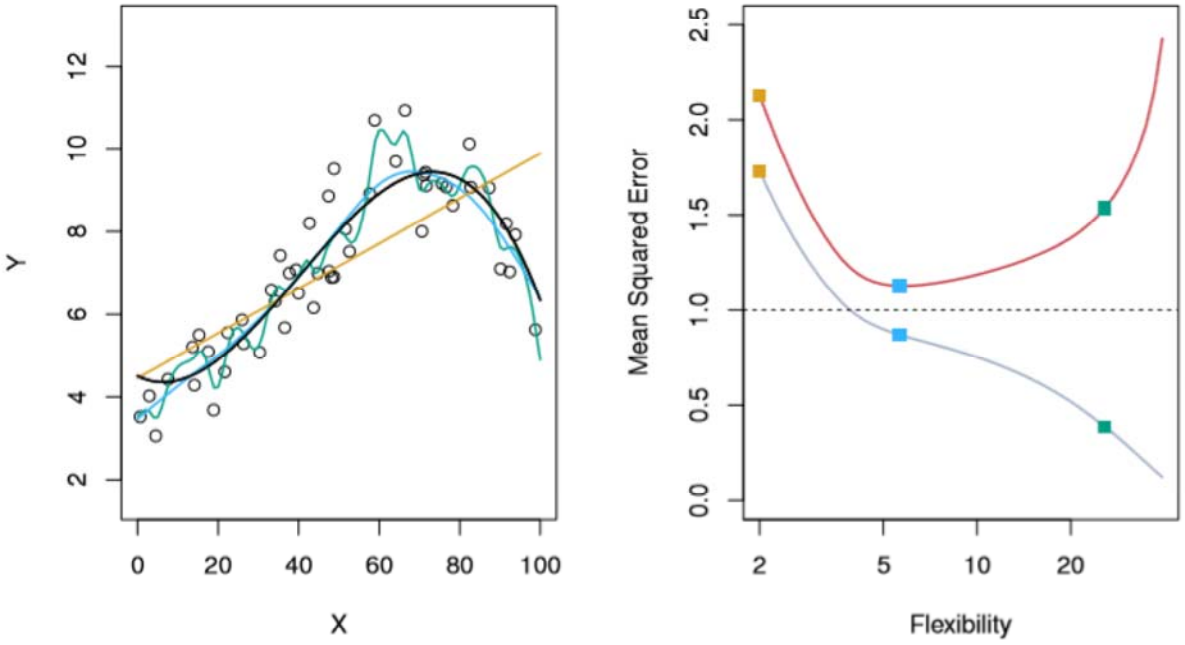
第一讲 统计基础

1 拟合效果检验

训练均方误差用来判断拟合的好不好，测试均方误差用来判断预测效果好不好。

当模型的光滑度增加时，训练均方误差会减小，但测试均方误差不一定会降低

训练均方误差小，测试均方误差大的现象成为**过拟合**，出现过拟合就意味着需要降低模型的光滑度，以此减小测试均方误差



左图黄蓝绿代表三种拟合模型，右图灰色曲线代表训练均方误差，红色曲线代表测试均方误差

从三个模型的角度，可以看到，**黄色**模型的训练均方误差和测试均方误差都很大，说明可能存在**欠拟合**情况；**绿色**模型的训练均方误差很小，测试均方误差较大，两者之间的差距很大，说明模型可能存在**过拟合**情况；**蓝色**模型的的训练均方误差和测试均方误差都相对较小，且两者之间的差距也较小，因此在这三个的模型中，**蓝色**模型相对较优。

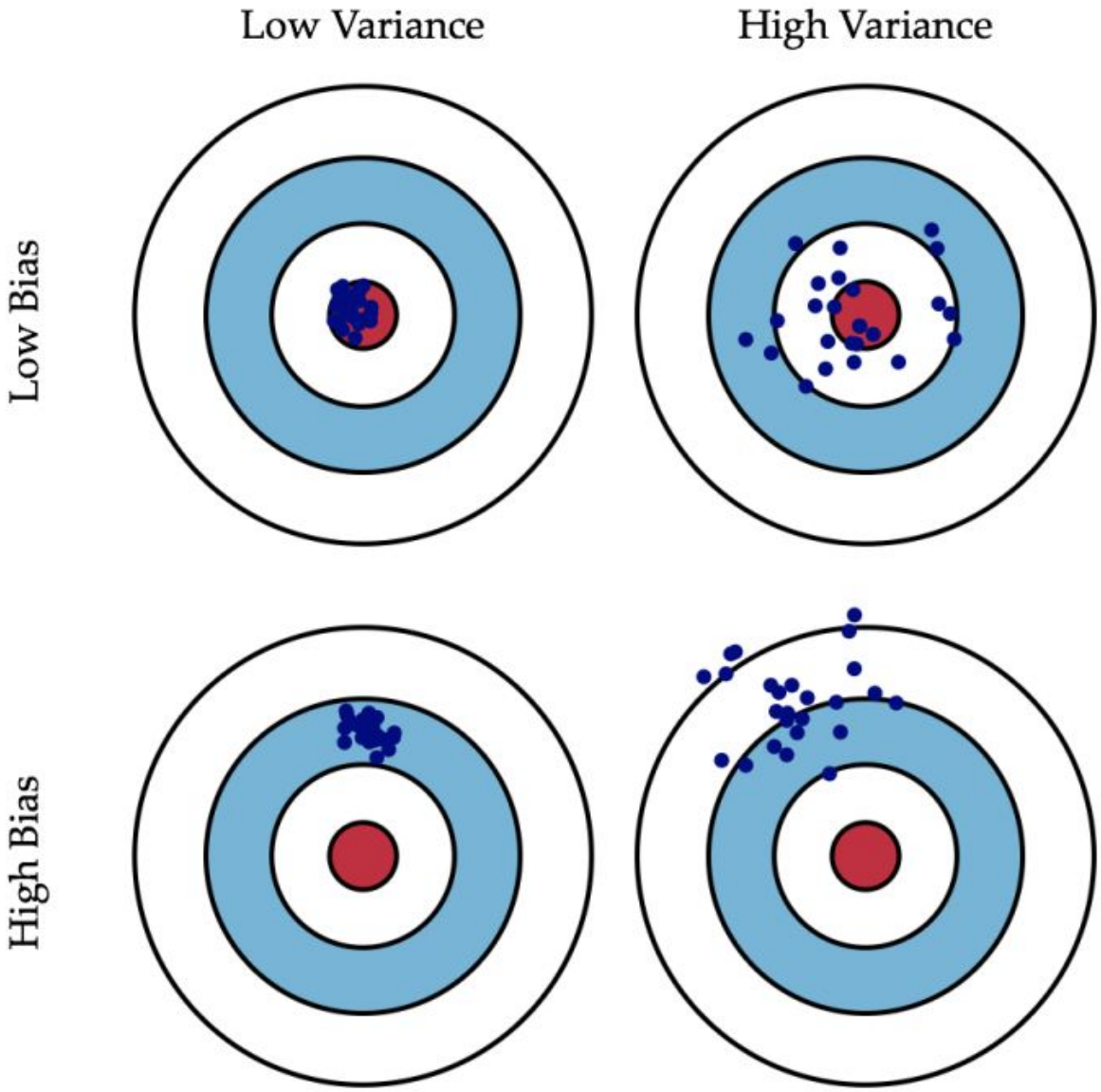
从右图图线的角度，**灰色**曲线代表了训练均方误差，可以看到随着模型光滑度的递增，训练均方误差越来越小。说明模型的训练均方误差可以通过增加模型的光滑度来改进。**红色**曲线代表了测试均方误差，图线呈**U**形，说明随着模型的光滑度增加，测试均方误差呈先减后增的态势。这就意味着，单一地通过增大模型光滑度，虽然可以减小训练均方误差，使得模型在训练集的表现变优，但是在超过**某一限度**时，会导致测试的均方误差增加，导致模型在测试集的表现不佳，出现过拟合情况。而判断这个限度在哪里，会在后面讨论（交叉验证等方法，还未讲到）

2 方差-偏差权衡

方差是指用不同的数据集去估计$f(.)$时，估计函数的改变量。对于光滑度低的模型，一般具有低方差性，也即当数据集中的某个点的位置进行偏移时，拟合函数不会有太大的改变。方差简单地说就是模型的**不稳定性**

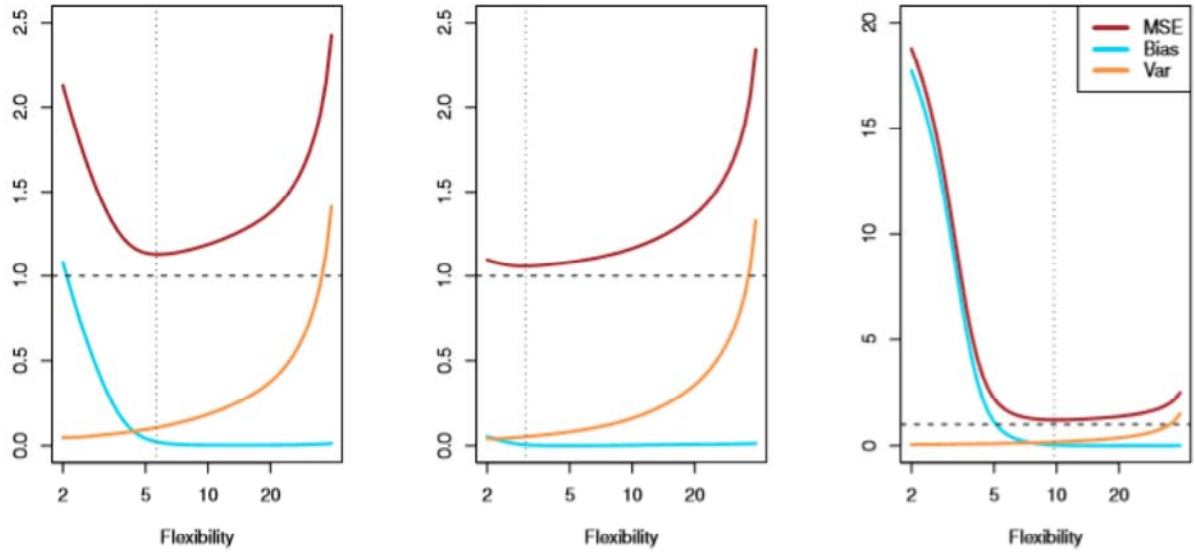
偏差是指为了选择一个简单的模型来逼近真是函数而被带入的误差。一般来说，光滑度越高的模型，偏差越小。偏差简单地说就是模型的**不准确性**

下图中，以打靶为例，左上表示低偏差-低方差（好），右上图表示低偏差-高方差（过拟合），左下图表示高偏差-低方差（欠拟合），右下图表示高偏差-高方差（差）



我们要测试均方误差**最小**，意味着方差与偏差同时很小，但在一个实际系统中，方差和偏差往往是不能兼得的，如果要降低模型的偏差，就会一定程度上提高模型的方差，反之亦然。出现这种情况的根本原因是，我们总是希望用有限的训练样本去估计无限的真实数据。当我们更相信这些数据的真实性时而忽视对模型的先验知识，就会尽量保证模型在训练样本的准确度，这样就可以减少偏差。但是，这样学习到的模型，很可能会失去一定的泛化能力，从而造成过拟合，降低模型在真实数据上的表现，增加模型的不确定性。相反，如果我们更加相信我们对于模型的先验知识，在学习模型的过程中对模型有更多的限制，就可以降低模型的方差，提高模型的稳定性，但也会使模型的偏差增大。

方差与偏差的权衡表现在数学模型上，实际上是模型光滑度的问题。我们可以用下图的方式来判断最优的光滑度。如下图所示是三个数据集的方差-偏差图，蓝色曲线表示不同光滑度下偏差的平方，橙色曲线表示方差，水平虚线表示不可约误差，红色曲线是测试均方误差（也就是前三个量的总和），垂直虚线所对应的光滑度是每一个数据集的最优测试均方误差所对应的模型参数。



但是在现实问题中，f 一般是未知的，这就意味着我们无法精确地计算测试均方误差、偏差和方差。但是，方差-偏差 权衡需要时刻谨记。

---

3 监督学习和无监督学习

有监督学习：回归、分类（区别在于输出的结果是否连续）

无监督学习：聚类、主成分

有监督和无监督的区别在于是否存在与预测变量相对应的响应变量y_i

---

4 先验概率和后验概率

先验概率是指根据以往经验和分析得到的概率

后验概率是在得到结果的信息后，重新修正的概率，是在考虑了事实后的条件概率，也就是事情已经发生，要求这件事情发生的原因是由某个因素引起的可能性的大小。后验概率的计算要以先验概率为基础。

假如给一些图片,这些图片中有的图上有动物的角,这些图片占了1/10(即**先验概率**),且已知在有角的条件下是犀牛的概率是0.8(类条件概率1,注意这个概率互补的概率是有角条件下不是犀牛的概率),已知在无角条件下是犀牛概率的是0.05(类条件概率2),现在拿起一张图,发现是一张犀牛的图,那么这张图上带角的概率有多大(求后验概率)

$$P(\text{图片上有动物的角}|\text{是犀牛}) = \frac{P(\text{图片上有动物的角且是犀牛})}{P(\text{是犀牛})}$$
$$= \frac{P(\text{是犀牛}|\text{图片上有动物的角}) * P(\text{图片上有动物的角})}{P(\text{是犀牛}|\text{图片上有动物的角}) * P(\text{图片上有动物的角}) + P(\text{是犀牛}|\text{图片上没有动物的角}) * P(\text{图片上没有动物的角})}$$

由图中公式可知**后验概率**为
$P(\text{图片上由动物的角}|\text{是犀牛}) = 0.8 * 0.1/(0.8 * 0.1 + 0.05 * 0.9) = 0.64$

---

5 参数方法和非参数方法

统计方法分为参数方法和非参数方法，

参数方法：基于模型的方法，把估计f的问题简化到估计一组参数

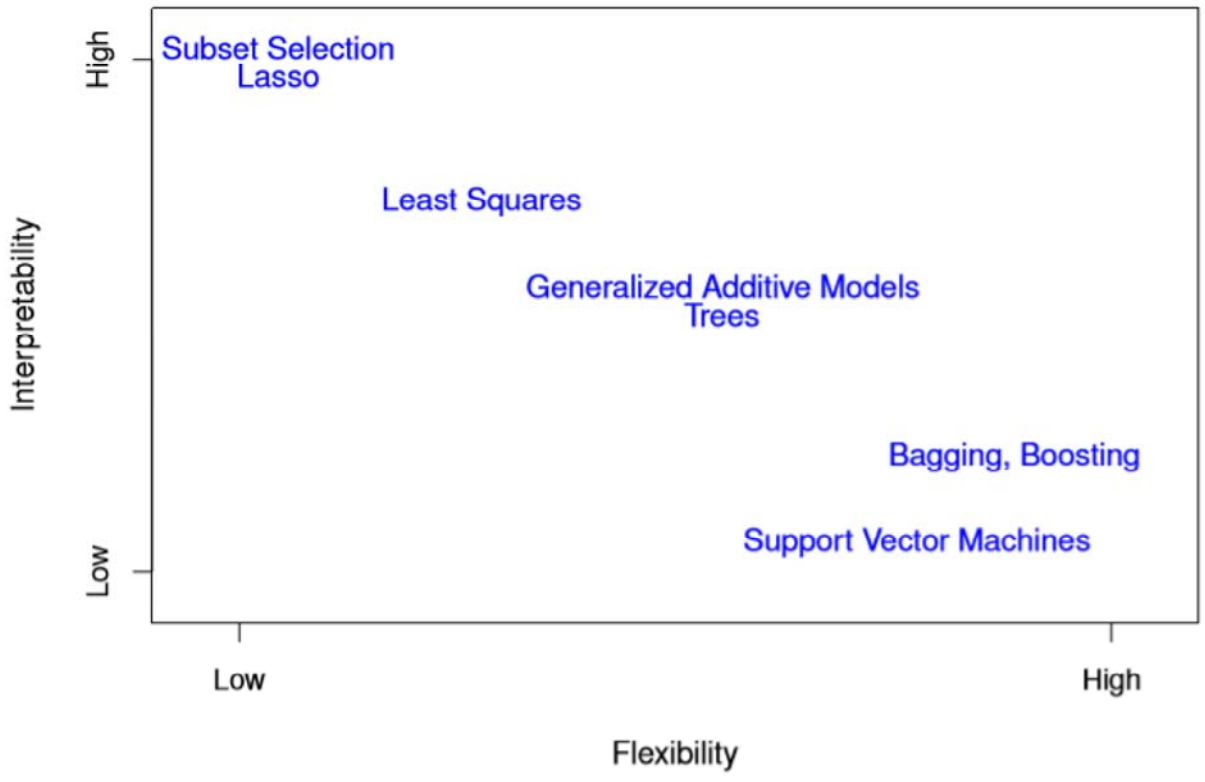
缺陷：选定的模型并非与真正的f在形式上是一致的，所以要拟合光滑度更高的模型，但是会带来过拟合问题

非参数方法：不需要对模型f的形式事先作出明确的假设，追求接近数据点的估计，估计函数在去粗和光滑处理后尽可能与更多数据点接近

优点：不限定函数形式，于是可能在更大的范围选择更适宜f形状的估计

缺点：需要大量观测点

下图是几种统计模型在光滑度和解释性之间的权衡



第二讲 线性回归

1 前言

下面不详述违反各项经典假设的原因及可能造成的后果，请参见计量经济学或者其它相关书籍

关于偏残差系数的数理意义请参考：赵松山《关于偏残差系数的讨论》，在这里不展开讨论

---

2 线性回归

2.1 简单线性回归

```
#Boston数据集在MASS包里面
library(MASS)
#fix(Boston)  #看一下Boston是一张怎样的数据表
#names(Boston)#看一下Boston数据表中的各个变量名称

#用lm.fit2.1 <-lm(medv ~ lstat, data = Boston)可以得到跟以下两句同样的结果
attach(Boston)
lm.fit2.1 <- lm(medv ~ lstat) #得到线性回归函数

lm.fit2.1 #系数
```

```
##
## Call:
## lm(formula = medv ~ lstat)
##
## Coefficients:
## (Intercept)        lstat
##       34.55        -0.95
```

```
summary(lm.fit2.1) #查看置信区间，p值，标准误，R^2，F统计量
```

```
##
## Call:
## lm(formula = medv ~ lstat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41   <2e-16 ***
## lstat       -0.95005    0.03873  -24.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

结果解读：星号个数表示显著性的强弱，“*”越多表示显著性越强

```
confint(lm.fit2.1) #得到系数估计值的置信区间
```

```
##                 2.5 %     97.5 %
## (Intercept) 33.448457 35.6592247
## lstat       -1.026148 -0.8739505
```

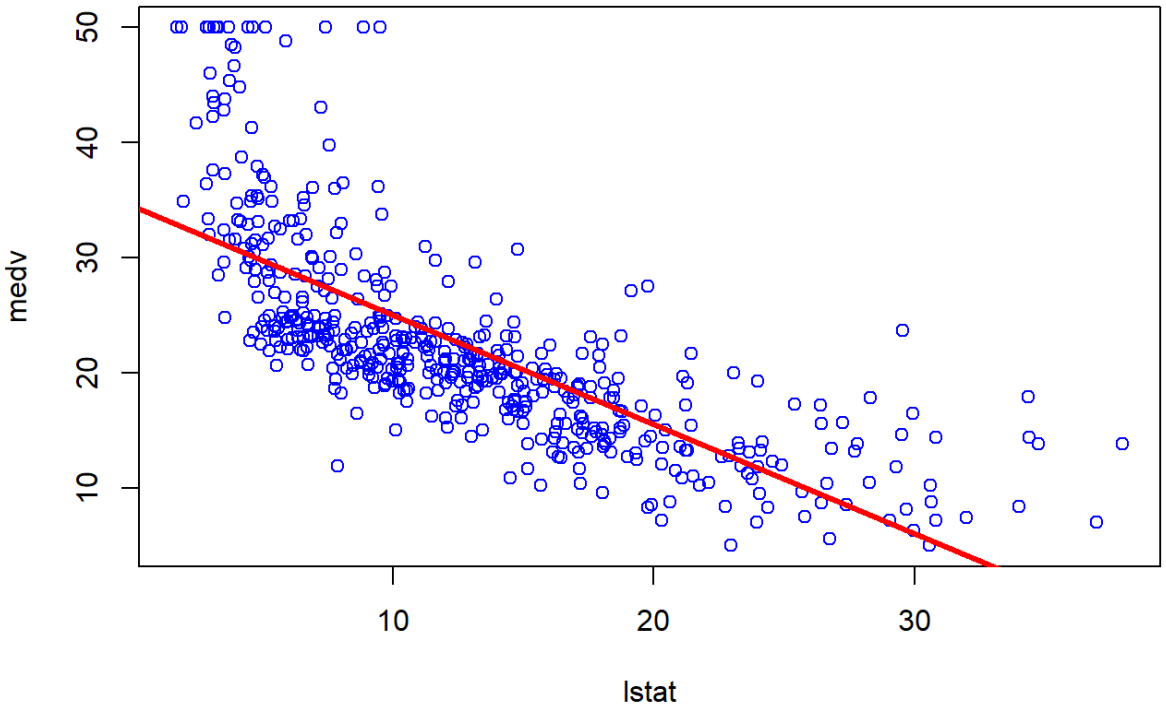
```
predict(lm.fit2.1, data.frame(lstat = (c(2, 6, 9))), interval = "confidence") #用lstat预测medv得到的置信区间
```

```
##        fit      lwr      upr
## 1 32.65374 31.67807 33.62942
## 2 28.85354 28.11121 29.59588
## 3 26.00340 25.39347 26.61332
```

```
predict(lm.fit2.1, data.frame(lstat = (c(2, 6, 9))), interval = "prediction") #用lstat预测medv得到的预测区间
```

```
##        fit      lwr      upr
## 1 32.65374 20.40284 44.90465
## 2 28.85354 16.61901 41.08808
## 3 26.00340 13.77618 38.23061
```

```
#画出函数拟合图
plot(lstat, medv, col="blue")
abline(lm.fit2.1, lwd = 3, col = "red")
```



注意： lm（应变量~自变量），plot（自变量，应变量），abline()不能单独使用，上面必有一行画图的命令

2.2 多元线性回归

```
lm.fit2.2.1 <- lm(medv ~ lstat + age, data = Boston) #medv关于lstat和age的回归模型
summary(lm.fit2.2.1)
```

```
##
## Call:
## lm(formula = medv ~ lstat + age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.981  -3.978  -1.283   1.968  23.158
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.22276    0.73085  45.458  < 2e-16 ***
## lstat       -1.03207    0.04819 -21.416  < 2e-16 ***
## age          0.03454    0.01223   2.826  0.00491 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.173 on 503 degrees of freedom
## Multiple R-squared:  0.5513, Adjusted R-squared:  0.5495
## F-statistic:   309 on 2 and 503 DF,  p-value: < 2.2e-16
```

```
lm.fit2.2.2 <- lm(medv ~ ., data = Boston) #medv关于所有变量的回归模型
summary(lm.fit2.2.2)
```

```
##
## Call:
## lm(formula = medv ~ ., data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.595  -2.730  -0.518   1.777  26.199
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
## crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
## zn           4.642e-02  1.373e-02   3.382 0.000778 ***
## indus        2.056e-02  6.150e-02   0.334 0.738288
## chas         2.687e+00  8.616e-01   3.118 0.001925 **
## nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
## rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
## age          6.922e-04  1.321e-02   0.052 0.958229
## dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
## rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
## tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
## ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
## black        9.312e-03  2.686e-03   3.467 0.000573 ***
## lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.745 on 492 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
## F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

```
lm.fit2.2.3 <- lm(medv ~ . - age, data = Boston) #medv关于除了age以外的所有变量的回归模型
summary(lm.fit2.2.3)
```

```
##
## Call:
## lm(formula = medv ~ . - age, data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.6054  -2.7313  -0.5188   1.7601  26.2243
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  36.436927   5.080119   7.172 2.72e-12 ***
## crim         -0.108006   0.032832  -3.290 0.001075 **
## zn            0.046334   0.013613   3.404 0.000719 ***
## indus         0.020562   0.061433   0.335 0.737989
## chas          2.689026   0.859598   3.128 0.001863 **
## nox         -17.713540   3.679308  -4.814 1.97e-06 ***
## rm            3.814394   0.408480   9.338  < 2e-16 ***
## dis          -1.478612   0.190611  -7.757 5.03e-14 ***
## rad           0.305786   0.066089   4.627 4.75e-06 ***
## tax          -0.012329   0.003755  -3.283 0.001099 **
## ptratio      -0.952211   0.130294  -7.308 1.10e-12 ***
## black         0.009321   0.002678   3.481 0.000544 ***
## lstat        -0.523852   0.047625 -10.999  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.74 on 493 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7343
## F-statistic: 117.3 on 12 and 493 DF,  p-value: < 2.2e-16
```

```
#用summary(lm(medv ~ lstat + age + lstat : age, data = Boston))可以得到跟以下语句一样的结果
summary(lm(medv ~ lstat * age, data = Boston)) #包含了lstat，age和lstat与age的交互项
```

```
##
## Call:
## lm(formula = medv ~ lstat * age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.806  -4.045  -1.333   2.085  27.552
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 36.0885359  1.4698355  24.553  < 2e-16 ***
## lstat       -1.3921168  0.1674555  -8.313 8.78e-16 ***
## age         -0.0007209  0.0198792  -0.036   0.9711
## lstat:age    0.0041560  0.0018518   2.244   0.0252 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.149 on 502 degrees of freedom
## Multiple R-squared:  0.5557, Adjusted R-squared:  0.5531
## F-statistic: 209.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

```
#如果要把二次项引入线性回归 ，要用I()，不能用lstat^2， 因为 "^" 表示交互项达到某一个次数，或者用下面的函数
lm.fit2.2.4 <- lm(medv ~ poly(lstat, 5)) #把lstat的一阶二阶至五阶全部写进去了
summary(lm.fit2.2.4)
```

```
##
## Call:
## lm(formula = medv ~ poly(lstat, 5))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.5433  -3.1039  -0.7052   2.0844  27.1153
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       22.5328     0.2318  97.197  < 2e-16 ***
## poly(lstat, 5)1 -152.4595     5.2148 -29.236  < 2e-16 ***
## poly(lstat, 5)2   64.2272     5.2148  12.316  < 2e-16 ***
## poly(lstat, 5)3  -27.0511     5.2148  -5.187 3.10e-07 ***
## poly(lstat, 5)4   25.4517     5.2148   4.881 1.42e-06 ***
## poly(lstat, 5)5  -19.2524     5.2148  -3.692 0.000247 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.215 on 500 degrees of freedom
## Multiple R-squared:  0.6817, Adjusted R-squared:  0.6785
## F-statistic: 214.2 on 5 and 500 DF,  p-value: < 2.2e-16
```

注意：如果自变量里面有定性变量，R会自动创建虚拟变量

2.3 选择“最佳模型”

选择最佳模型跟自变量的选取可以放在一起，选择最佳模型是对已有模型去进行比较，然后选自变量相当于是还在建模，但都有关于要不要把某一个自变量引进来的问题

但是一个模型好不好，主要看拟合的怎么样或者预测能力如何，根据模型不同的用途有不同的标准，以及可能会出现某两个检测方法得出不同结论的情况，这个需要从R^2等多种信息量准则去综合判断

方法1：anova检验（这个函数可以比较两个嵌套模型的拟合优度）（（必须要是**嵌套模型**））

```
#以下比较模型2.2.3与2.2.2
anova(lm.fit2.2.3, lm.fit2.2.2)
```

```
## Analysis of Variance Table
##
## Model 1: medv ~ (crim + zn + indus + chas + nox + rm + age + dis + rad +
##     tax + ptratio + black + lstat) - age
## Model 2: medv ~ crim + zn + indus + chas + nox + rm + age + dis + rad +
##     tax + ptratio + black + lstat
##   Res.Df   RSS Df Sum of Sq      F Pr(>F)
## 1    493 11079
## 2    492 11079  1  0.061834 0.0027 0.9582
```

结果解读：这里第一个模型嵌套在第二个模型中，结果显示，p值为0.96，意味着检验不显著，即不拒绝原假设，也就是不需要把age引入线性模型中。

方法2：用赤池信息量准则（AIC）((寻找可以最好地解释数据但包含最少自由参数的模型))(((没有嵌套模型的要求)))

```
AIC(lm.fit2.2.3, lm.fit2.2.2)
```

```
##             df      AIC
## lm.fit2.2.3 14 3025.611
## lm.fit2.2.2 15 3027.609
```

结果解读：这里表明lm.fit2.2.3更好，跟上面用ANOVA的结果一样

通过变量选择从大量的候选变量里面得到最终的自变量（较为流行以下两种）（（CH6的内容，下次再说））

方法3：逐步回归法（未展开）

方法4：全子集回归（未展开）

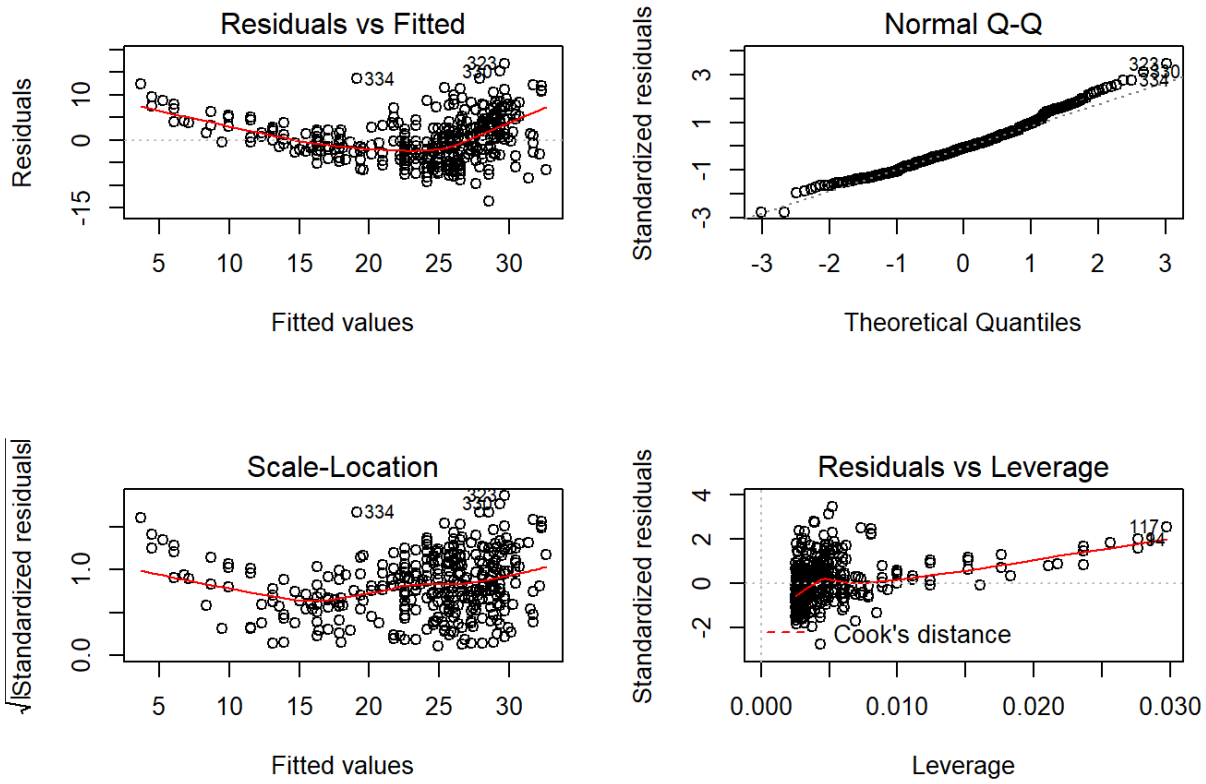
---

3 判断是否满足假设

3.1 总体判断

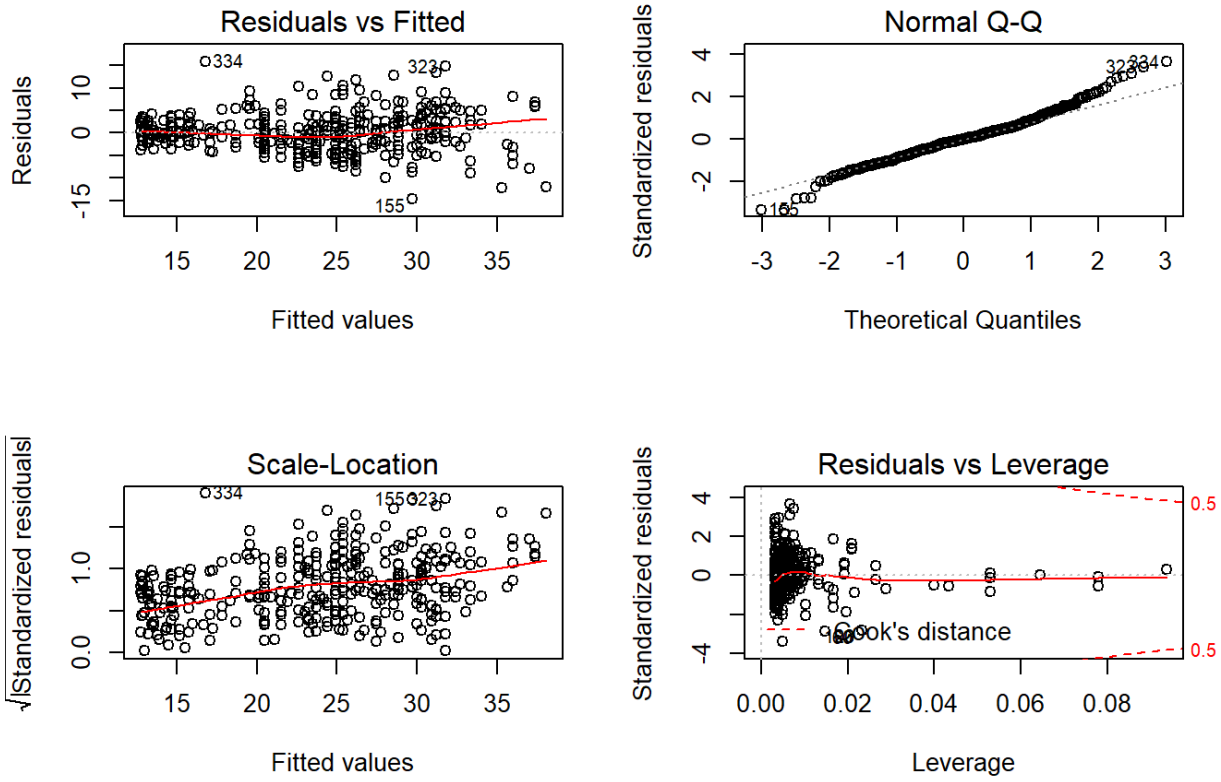
标准诊断图形

```
#Auto数据集在ISLR包里面
library(ISLR)
lm.fit3.1.1 <- lm(mpg ~ horsepower, data = Auto)
par(mfrow = c(2, 2))
plot(lm.fit3.1.1)
```



结果解读：左上图是**“残差与拟合图”**，用来判断线性性。该图中显示出了一个曲线关系，说明因变量与自变量不满足线性性，可能要在回归模型中加入一个二次项；右上图是**“正态Q-Q图”**，可以用来判断是否满足正态性假设。图中的点均落在呈45°角的直线上，说明满足正态性假设；左下图是**“位置尺度图”**，用来判断同方差性。该图中水平线周围的点呈随机分布，说明满足同方差性；右下图是**“残差与杠杆图”**，可以用来鉴别异常值点（可读性不佳，不展开，后面补充更好的鉴别图像）。

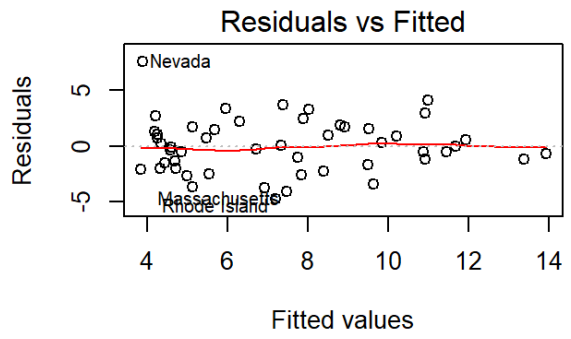
```
#加入二次项，与原来的回归诊断图进行比较
lm.fit3.1.2<- lm(mpg ~ horsepower + I(horsepower ^ 2), data = Auto)
par(mfrow = c(2, 2))
plot(lm.fit3.1.2)
```

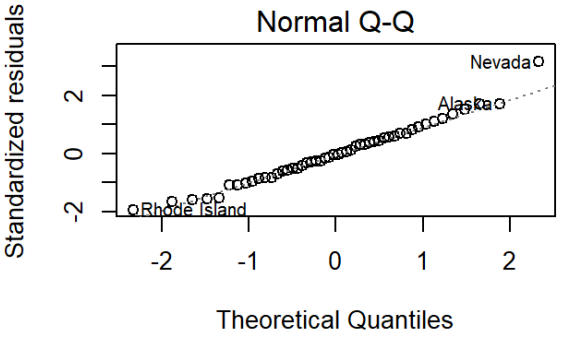


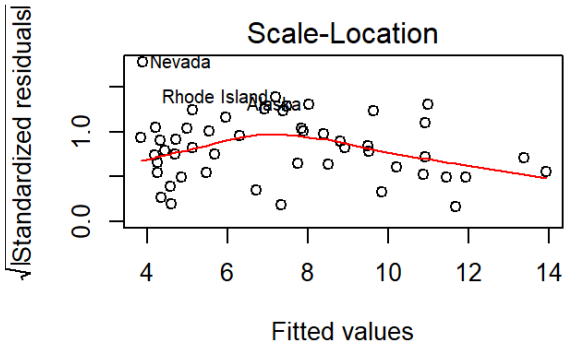
结果解读：加入二次项后，**“残差与拟合图”**趋于一条直线，说明此时满足了线性性

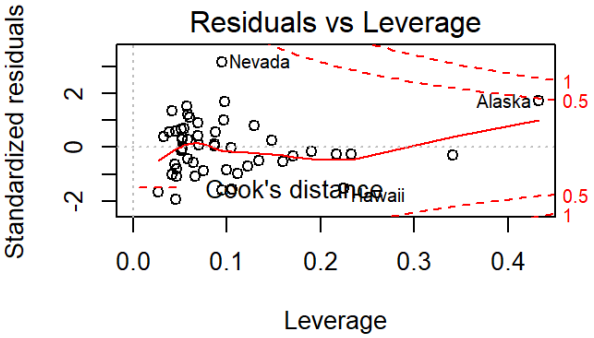
以下使用基础包中的state.x77数据集(这个数据集是一个矩阵)

```
states<-as.data.frame(state.x77[,c("Murder","Population","Illiteracy","Income","Frost")])#这边是为了把以矩阵形态存在的数据集变成数据框
lm.fit3.1.3 <- lm(Murder ~ ., data = states)
par(mfrow = c(2, 2))
plot(lm.fit3.1.3)
```









基于gvlma包

```
library(gvlma)
gvmodel <- gvlma(lm.fit3.1.3)
summary(gvmodel)
```

```
##
## Call:
## lm(formula = Murder ~ ., data = states)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7960 -1.6495 -0.0811  1.4815  7.6210
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.235e+00  3.866e+00   0.319   0.7510
## Population  2.237e-04  9.052e-05   2.471   0.0173 *
## Illiteracy  4.143e+00  8.744e-01   4.738 2.19e-05 ***
## Income      6.442e-05  6.837e-04   0.094   0.9253
## Frost       5.813e-04  1.005e-02   0.058   0.9541
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.535 on 45 degrees of freedom
## Multiple R-squared:  0.567,  Adjusted R-squared:  0.5285
## F-statistic: 14.73 on 4 and 45 DF,  p-value: 9.133e-08
##
##
## ASSESSMENT OF THE LINEAR MODEL ASSUMPTIONS
## USING THE GLOBAL TEST ON 4 DEGREES-OF-FREEDOM:
## Level of Significance =  0.05
##
## Call:
##  gvlma(x = lm.fit3.1.3)
##
##                     Value p-value                Decision
## Global Stat        2.7728  0.5965 Assumptions acceptable.
## Skewness           1.5374  0.2150 Assumptions acceptable.
## Kurtosis           0.6376  0.4246 Assumptions acceptable.
## Link Function      0.1154  0.7341 Assumptions acceptable.
## Heteroscedasticity 0.4824  0.4873 Assumptions acceptable.
```

结果解读：输出的结果除了summary(lm.fit3.1.3)的内容外，最后五行值得注意。它不仅给出了综合检验结果（Global Stat），还给出了偏度，峰度，异方差性的评价。当然输出结果表示，数据满足OLS回归模型的各项所有统计假设（Assumptions acceptable）

3.2 自相关性

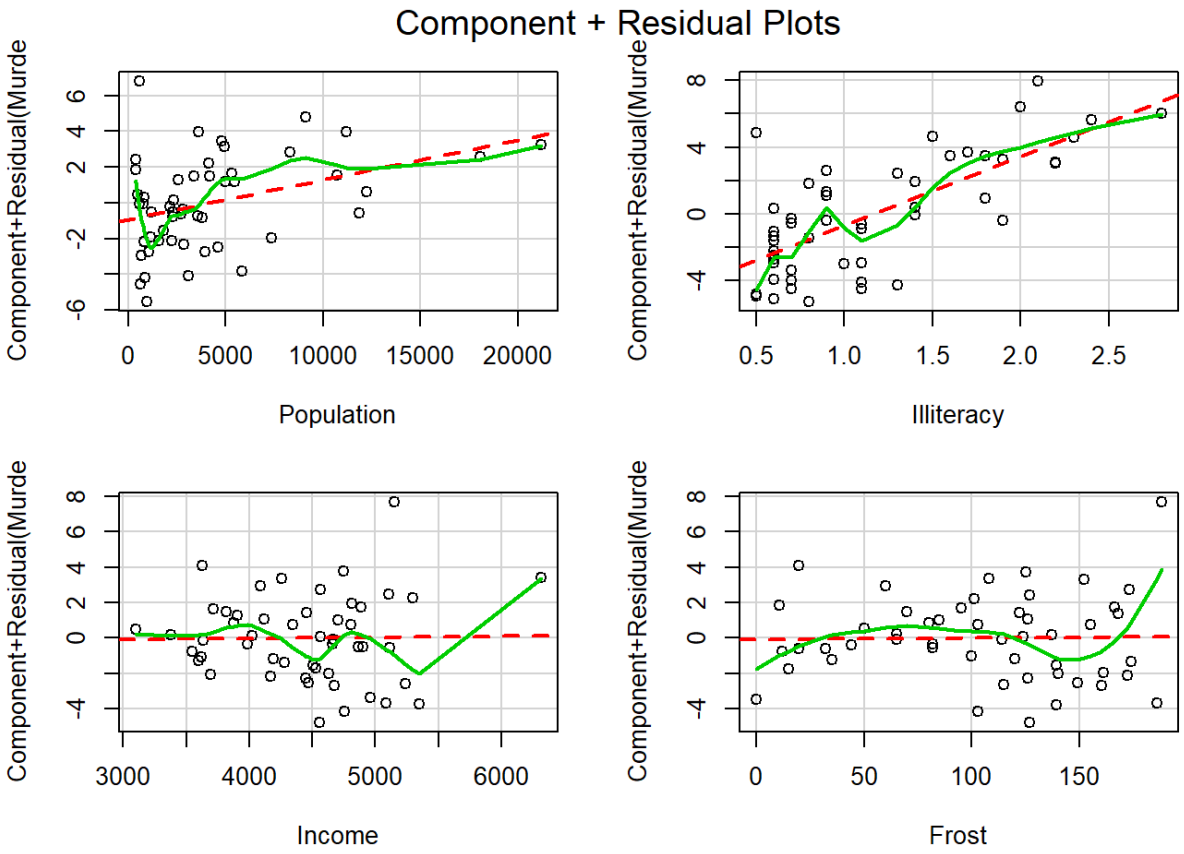
```
library(car)
durbinWatsonTest(lm.fit3.1.3)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1      -0.2006929      2.317691   0.218
##  Alternative hypothesis: rho != 0
```

结果解读：P>0.05,不能拒绝原假设，即不存在自相关性

3.3 线性性

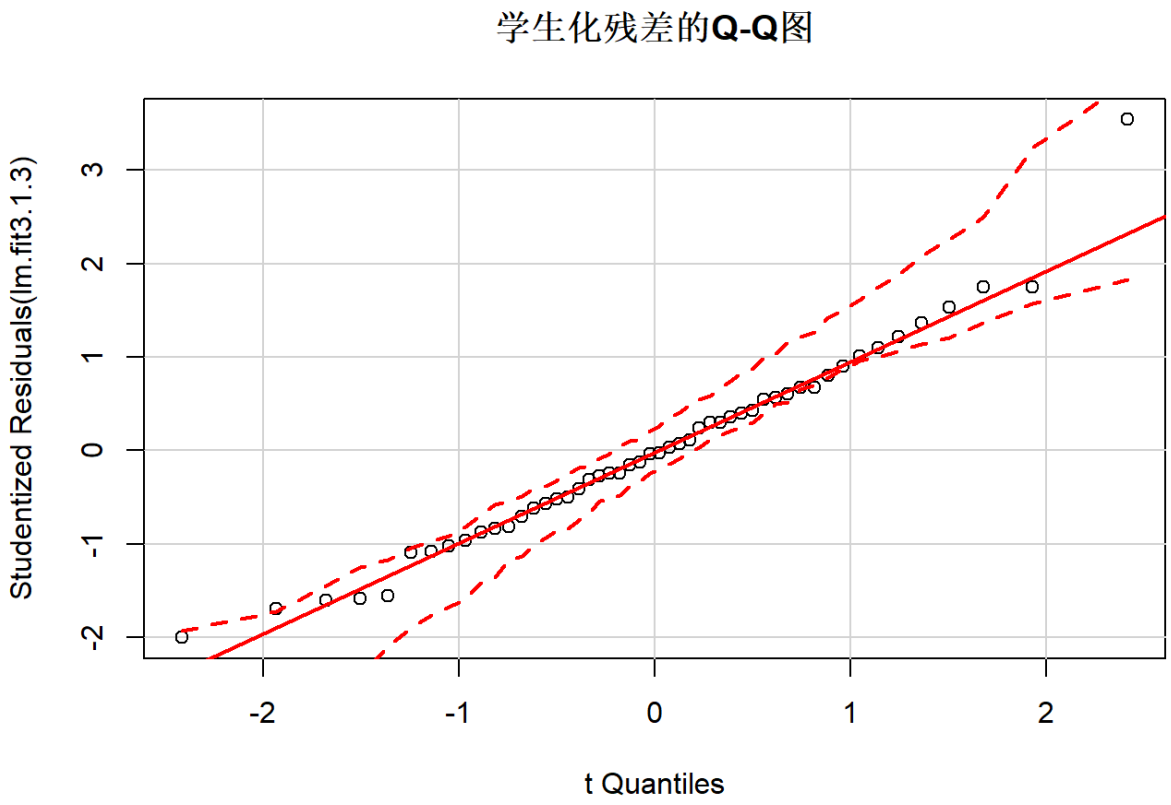
```
crPlots(lm.fit3.1.3) #具体可以再探究，跟标准诊断图形的左上图是吻合的，都表示满足线性性
```



结果解读：输出的是偏残差图，这边四幅图都显示拟合曲线和平滑拟合曲线都呈线性，表明满足线性性假设

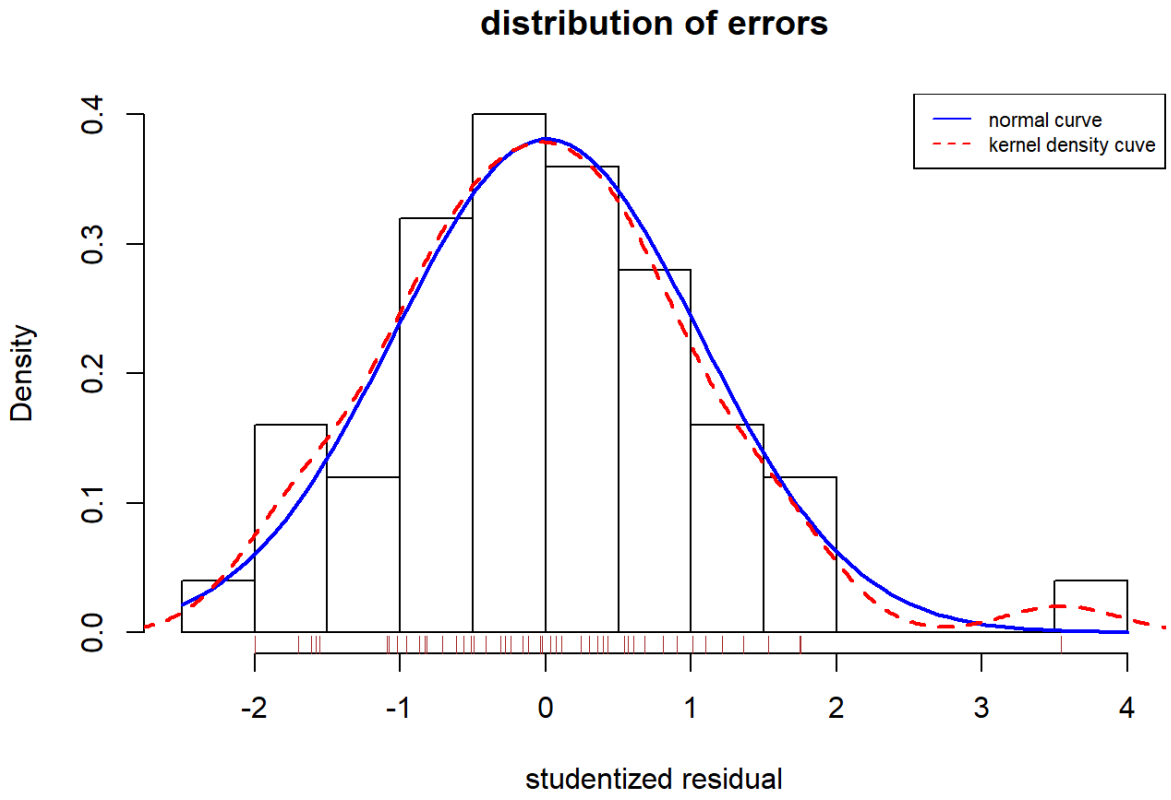
3.4 正态性

```
#car包提供大量函数，大大增强拟合和评价回归模型的能力
qqPlot(lm.fit3.1.3, label=row.names(states),id.method="identify",simulate = TRUE, mai
n = "学生化残差的Q-Q图")
```



结果解读：图中的点均散落在呈45°角的直线附近，且都落在置信区间内，说明较好地满足了正态性假设。但是发现存在一个异常值点（左上角），它是一个很大的正残差值（真实值-预测值），说明有数据被低估了。

```
#自己编写一个函数用来画学生化残差图
rsdplot <- function(fit, nbreaks = 10)
{
    z <- rstudent(fit)
    hist(z, breaks = nbreaks, freq = FALSE, xlab = "studentized residual", main = "di
stribution of errors")
    rug(jitter(z), col = "brown")
    curve(dnorm(x, mean = mean(z), sd = sd(z)), add = TRUE, col = "blue", lwd = 2)
    lines(density(z)$x, density(z)$y, col = "red", lwd = 2, lty = 2)
    legend("topright", legend = c("normal curve", "kernel density cuve"), lty = c(1,
2), col = c("blue", "red"), cex = 0.7)
}
rsdplot(lm.fit3.1.3)
```

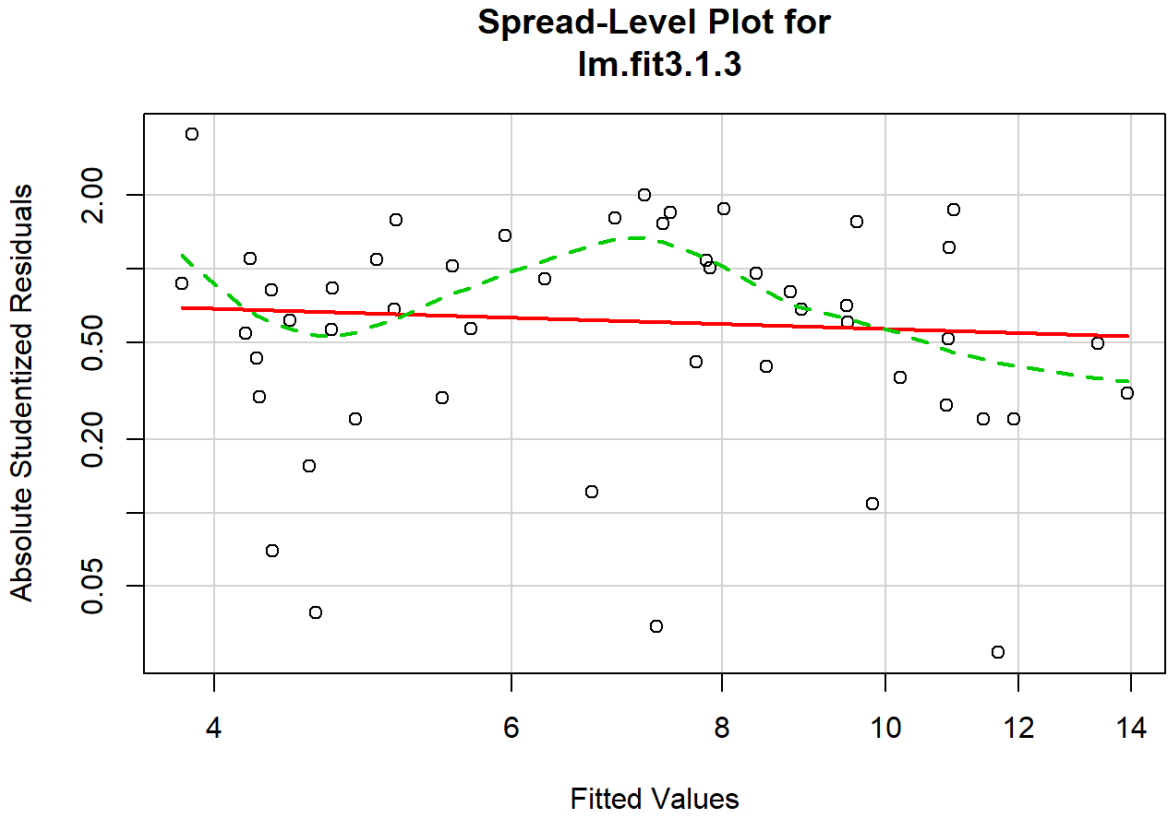


结果解读：除了一个很明显的离群点，其他误差值都很好地服从了正态分布，而且密度曲线跟正态分布曲线很接近。此外，这个柱状图和密度测量分布图能够很方便地看出分布的斜度（比Q-Q图容易）

3.5 同方差性

方法1 图形法

```
spreadLevelPlot(lm.fit3.1.3)
```




```
##
## Suggested power transformation:  1.209626
```

结果解读：“残差拟合值图”显示，散点在水平的残差拟合曲线周围呈水平随机分布，说明满足同方差性假设。

方法2 假设检验法

```
ncvTest(lm.fit3.1.3)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 1.746514    Df = 1     p = 0.1863156
```

结果解读：p=0.19,说明不显著，即满足原假设，也就是说，不存在异方差性

3.6 多重共线性

非正规方法：

方法1：增或减一个变量，回归系数大变

方法2：对重要自变量进行回归系数的单项检验，结果不显著

方法3：回归系数的代数符号与经验值相反

方法4：相关阵中两两自变量的相关系数较大

方法5：重要回归系数的置信区间较大

正规方法：计算方差膨胀因子（局限性：不能区别几个同时多重共线性）

```
sqrt(vif(lm.fit3.1.3)) > 2 #先计算方差膨胀因子，再与2比较大小
```

```
## Population Illiteracy     Income      Frost
##      FALSE      FALSE      FALSE      FALSE
```

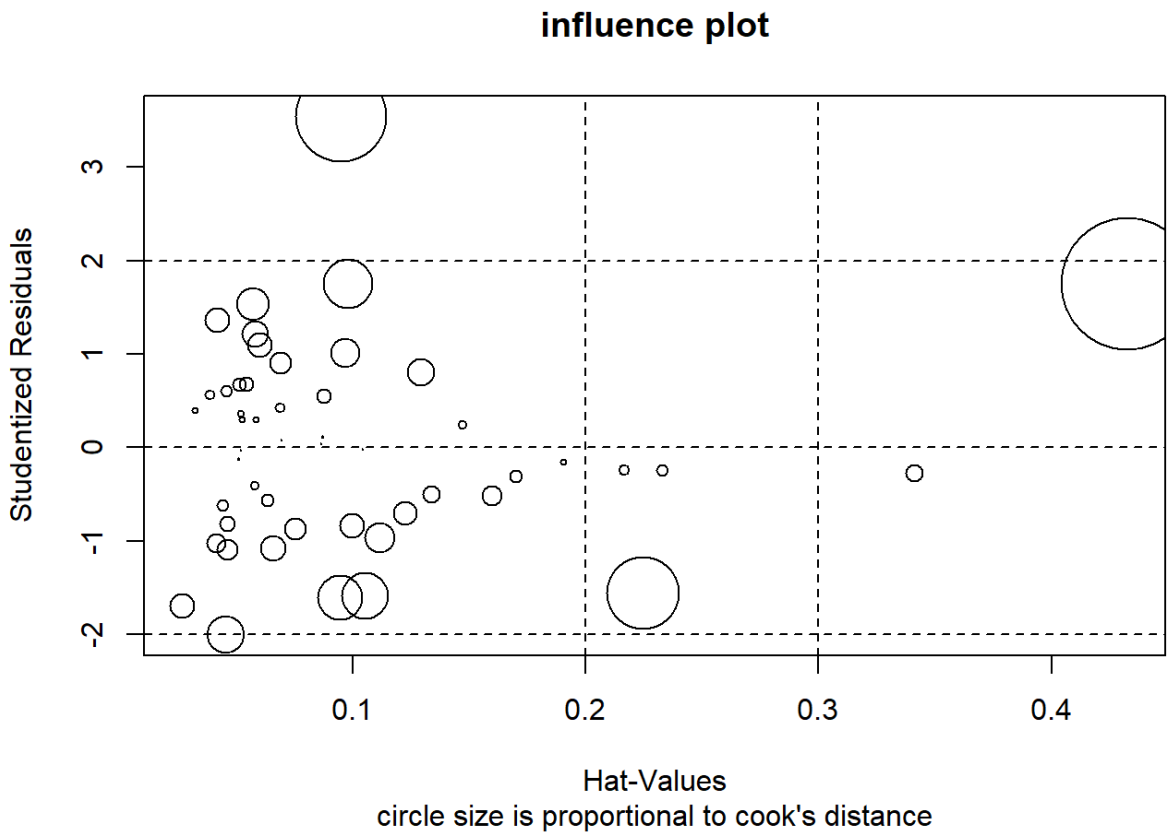
结果解读：TRUE表明存在多重共线性，FALSE表明不存在多重共线性，所以可以看到这边不存在多重共线性

---

4 判断是否存在异常值点

4.1 三种点总的判断

```
influencePlot(lm.fit3.1.3,id.method="identify",main="influence plot",sub="circle size is proportional to cook's distance")
```



结果解读：纵坐标看离群点(>2或<-2),横坐标看高杠杆值点（>0.2或0.3），圈圈大小看对模型参数估计的影响大小，太大的可能是强影响点。从图中看到三种点均有存在

4.2 离群点

方法1：通过Q-Q图，落在置信区间外的是离群点

方法2：标准化残差值>2或<-2的可能是离群点

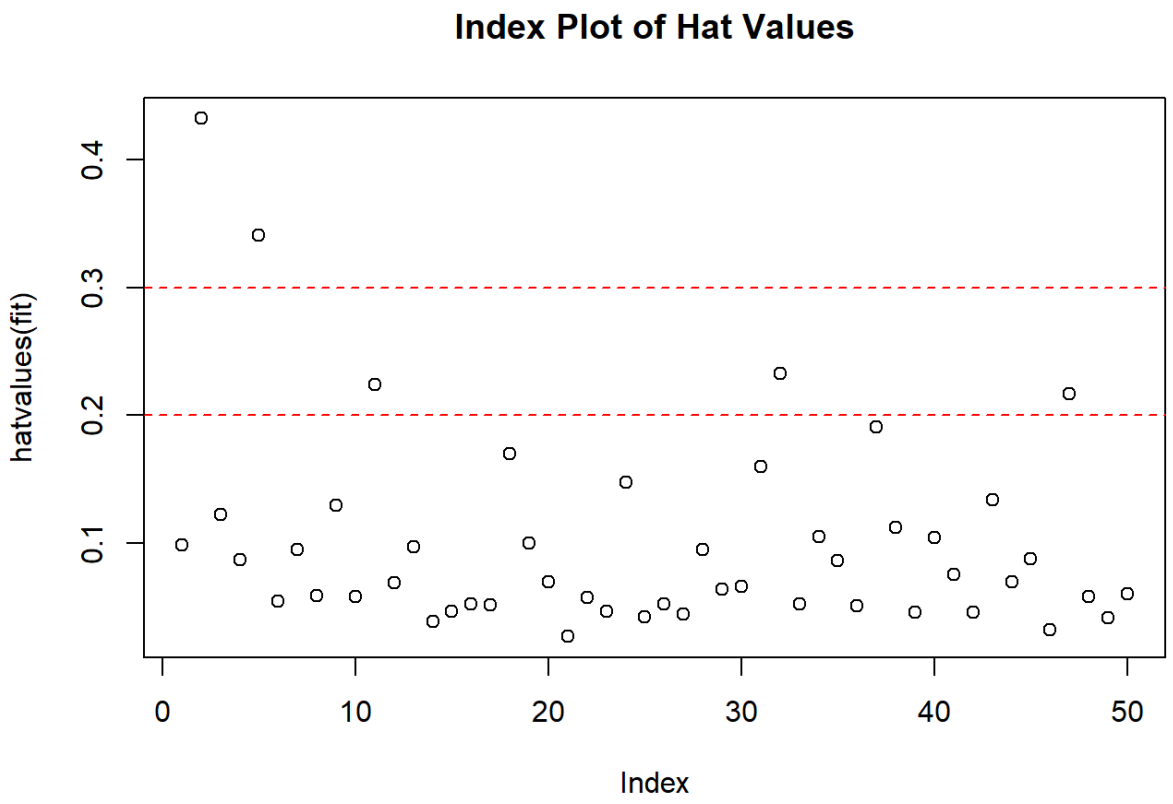
方法3：

```
#下面这个函数，是通过！单个！最大（-or+）残差值的显著性来判断是否具有离群点，如果不显著，就说明没有离群点，如果显著，必须删除这个离群点，然后再次使用这个函数用来判断是不是还有其他离群点的存在
outlierTest(lm.fit3.1.3)
```

```
##        rstudent unadjusted p-value Bonferonni p
## Nevada 3.542929         0.00095088     0.047544
```

4.3 高杠杆值点

```
hatplot <- function(fit)
{
    p <- length(coefficients(fit)) #p是参数个数
    n <- length(fitted(fit))#n是样本量
    plot(hatvalues(fit), main = "Index Plot of Hat Values")
    abline(h = c(2, 3) * p / n, col = "red", lty = 2)
    identify(1:n, hatvalues(fit), names(hatvalues(fit)))#定位函数
}
hatplot(lm.fit3.1.3)
```




```
## integer(0)
```

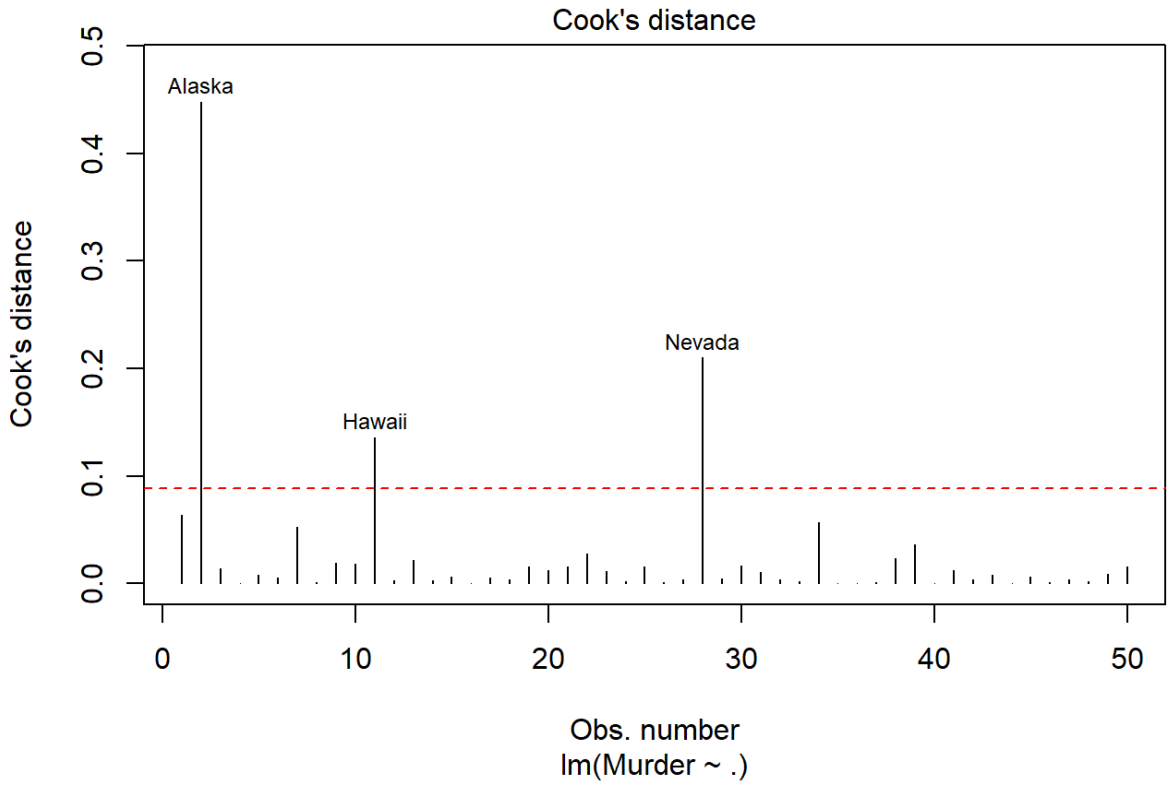
结果解读：只需要看图，水平线标注的是帽子均值2倍和3倍的位置，一般认为超过帽子均值2倍和3倍

的位置的点是高杠杆值点。

4.4 强影响值点

方法1：计算Cook距离并绘图（Cook's D图）（该图对于找强影响点很有用，但是不能反映这些点是怎么影响模型的）((这个图可以直接显示出来强影响点的名字))

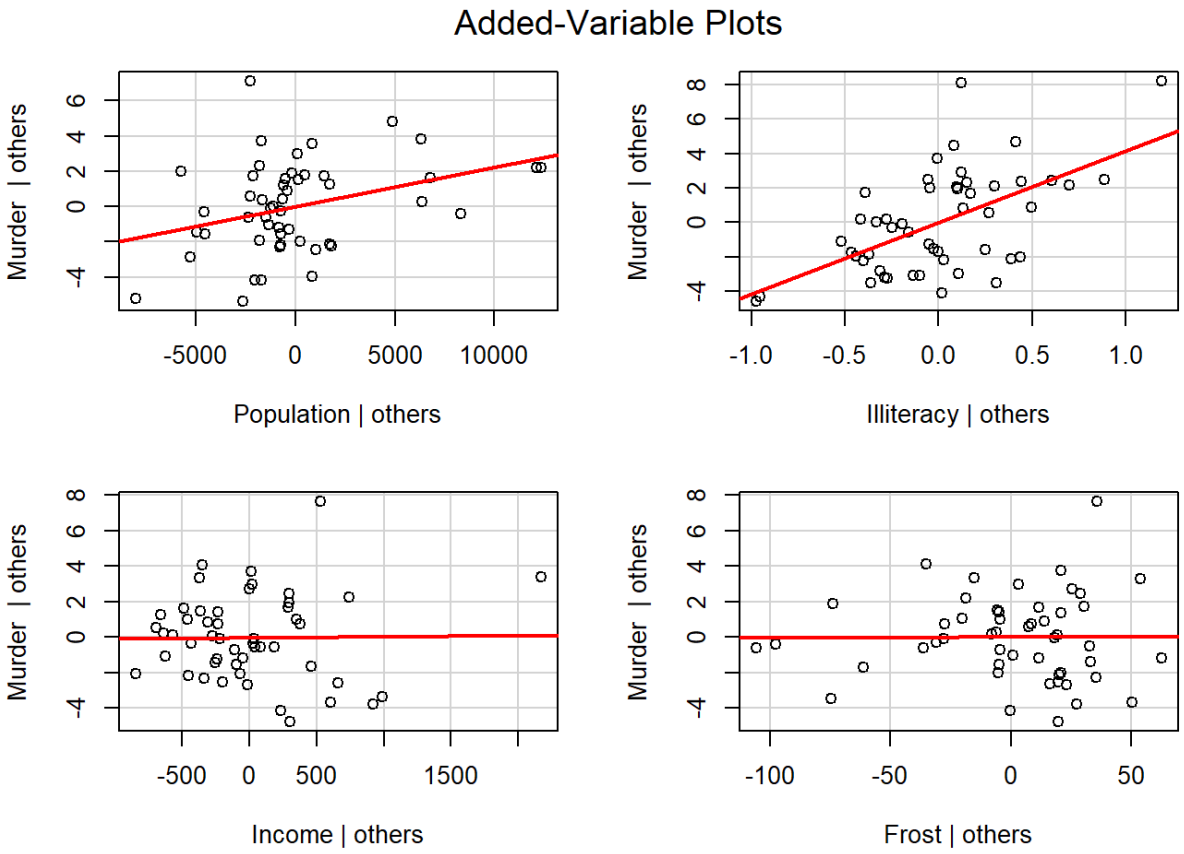
```
cutoff <- 4 / (nrow(states) - (length(lm.fit3.1.3$coefficients) - 1) - 1)
plot(lm.fit3.1.3, which = 4, cook.levels = cutoff)#？which=4表示前面的第四张图，但是第4张图并不是这样的(而且第1，2，3张图是一样的)#这里好像有点疑问
abline(h = cutoff, lty = 2, col = "red")
```



结果解读：从图中可以看到有三个强影响点（已被标出名字）

方法2：绘制变量添加图

```
avPlots(lm.fit3.1.3,id.method="identify")
```



结果解读：变量添加图是，预测变量X_k 与其余k-1个预测变量拟合的残差值（X轴）与响应变量Y与其余k-1个预测变量拟合的残差值（Y轴）画出的散点关系图，还给出了平滑拟合曲线。图中的直线表示相应预测变量的实际回归系数。如左下和右下两个图，直线为水平在O处，说明这两个预测变量是不显著的，最终的回归模型中，这两个变量前的系数应该是0。

---

5 问题的解决

5.1 违反线性性假设

```
boxTidwell(Murder ~ Population + Illiteracy, data = states)
```

```
##             Score Statistic   p-value MLE of lambda
## Population      -0.3228003 0.7468465     0.8693882
## Illiteracy       0.6193814 0.5356651     1.3581188
##
## iterations =  19
```

结果解读：结果表明使用变换，可以改善线性关系，但是由于计分检验的P值都>0.05，所以不能拒绝原假设，意味着不需要进行该变量变换

5.2 违反正态性假设

```
summary(powerTransform(states$Murder))
```

```
## bcPower Transformation to Normality
##
##                Est.Power Std.Err. Wald Lower Bound Wald Upper Bound
## states$Murder     0.6055   0.2639           0.0884           1.1227
##
## Likelihood ratio tests about transformation parameters
##                             LRT df       pval
## LR test, lambda = (0) 5.665991  1 0.01729694
## LR test, lambda = (1) 2.122763  1 0.14512456
```

结果解读：结果表明，用murder^0.6来正态化变量murder，但是发现不能拒绝lambda=1的假设，意味着，不需要进行该变量变换

5.3 违反同方差性

前面在<更好的方法（分散）>里面，检验同方差性的时候已经提到了

5.4 存在多重共线性

方法1：删除存在多重共线性的变量（把<更好的方法（分散）>里面，检测多重共线性，出现TRUE的变量删除）（（局限性：得不到关于被剔除变量的直接信息，且模型中剩余变量回归系数大小受模型外的相关变量的影响））

方法2：在多项式回归模型中，把任意给定的自变量表示成与其均值离差的形式，以大大降低一阶，二阶和更高阶的项之间的多重共线性

方法3：岭回归（专门处理多重共线性）((下次再讲))

方法4：主成分回归（参考《应用回归分析》何晓群，刘文卿）、因子分析这些

5.5 存在异常值点

如果能肯定地说明一个异常观察值是很大的测量误差的结果，剔除是合适的

如果异常观察值是准确的，他可能意味着模型的失败（遗漏重要自变量或选择了不正确的函数形式）

如果异常观察值是准确的，但找不到对于他的解释，那么可以抑制他的影响（而不是剔除）（（比如使用最小绝对离差法：通过极小化绝对离差之和来估计回归系数。这种方法具有对 异常数据和不合适模型并不敏感的性质，除了这个方法，还有其他稳健的方法可以参考《统计稳健性：关于当代应用问题的看法》罗伯特.V.霍格））

---

6 补充（待补充）

6.1 库克距离（待补充）

6.2 帽子矩阵（待补充）

6.3 异常值点的区分（待补充）

第三讲 经典分类

1 前言

1.1 Bayes、Fisher和距离判别法

从贝叶斯公式出发，我们可以得到LDA线性判别分析（分布满足：均值不同，方差相同，高斯分布）、QDA二次判别分析（分布满足：均值不同，方差不同，高斯分布）

Fisher判别法不足：不考虑各总体出现的概率大小，判别方法与错判后造成的损失无关，而这些恰好是贝叶斯的优点

正态等协差阵条件下，Bayes判别法与距离判别法等价

二分类且等协差条件下，Fisher判别法等价于距离判别法

正态等协差阵条件下，Bayes，Fisher线性判别法和距离判别法三者等价

Bayes要求知道总体分布类型（并不要求正态性），理论上可以说明Bayes判别在总体是非正态的情况下也适用，但是丧失正态性后，Bayes判别法具有的平均错判率最小的性质就不一定存在了

Fisher判别法和距离判别法对总体分布没有要求，只要求各类总体的二阶矩存在

当k个总体的均值向量共线性程度高时，Fisher判别法可以用较少的判别函数进行判别，这个时候，比Bayes简单

1.2 KNN跟线性回归的对比

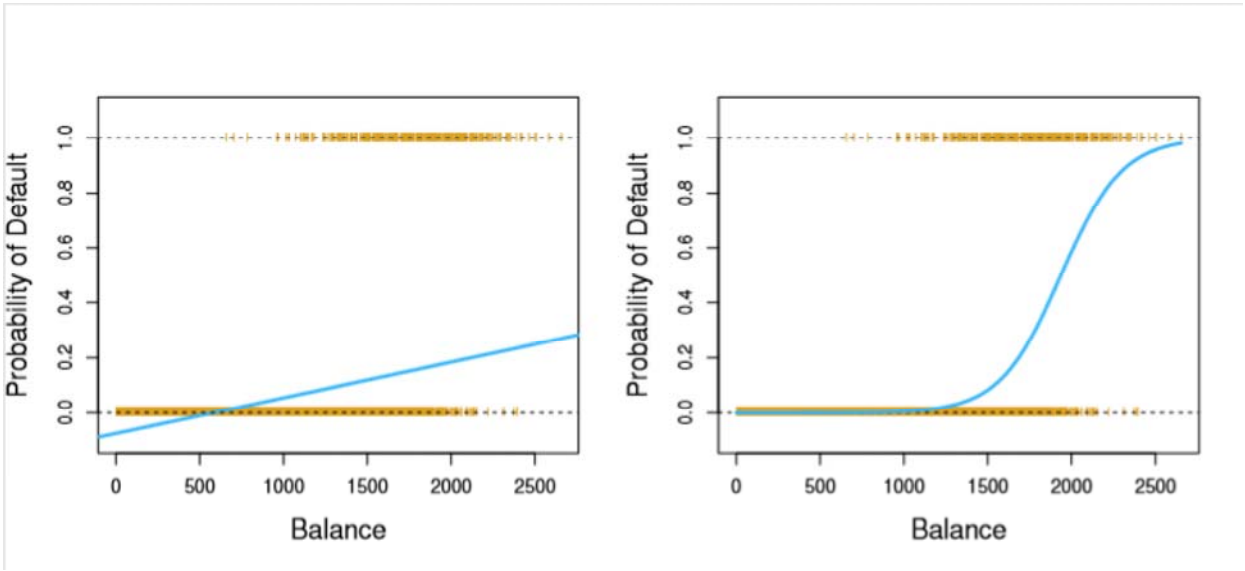
K最近邻(KNN)跟线性回归有一个对比（是非参方法与参数方法的对比）

1. 选定的参数形式接近 f 的真实形式，参数方法更优
2. 因为需要估计的系数较少，所以较容易拟合（优）
3. 系数有简单的解释，可以容易地进行统计显著性检验（优）
4. 假设太多，如果目标是预测的准确性，而所指定的函数与实际相差太远，参数方法的表现不佳（缺）
5. 如果 x 和Y的真实关系是近仅线性的，那么线性回归就会比较好， 非线性的情况， KNN 可能会大大优于线性回归
6. KNN 的预测效果随着维数增大而恶化（样本量少的话，参数方法会优于非参数方法）

1.3 分类时不使用线性回归

两水平以上的定性响应变量，哑变量的方法不能任意推广。这是因为不同种类的编码方式会产生完全不同的线性模型，导致测试观测产生不同的观测结果

两水平的定性相应变量，用线性回归模型，原则上一定有一些预测变量值是没有意义的（预测的概率>1或<0的，如以下右图），而考虑使用logistic回归模型（如下左图）就可以避免这一问题



1.4 多分类不使用Logistic回归

使用一个预测变量做Logistic回归时，如果其他预测变量与之有关系，那么模型可能会存在风险（只用一个预测变量得到的结果，可能与多个预测变量得到的结果完全不同——**混淆现象**）

当分类数超过2时，一般使用线性判别分析方法

这是因为当类别区分度高时，Logistic回归模型的参数估计不够稳定；n 较小时，在每一类响应分类中预测变量近似服从正态分布时，线性判别分析模型比Logistic回归模型更稳定

1.5 本讲数据集介绍

这一讲所使用的数据集**(Smarket数据集)**:2001年年初至2015年年末1250天S&P500股票指数的投资回报率，Lag1到Lag5表示过去五个交易日中的每个交易日的投资回报率，Volume表示前一日的股票成交量（十亿），Today表示当日的投资回报率，Direction表示这些数据在市场的走势方向（up：涨，down：跌）

---

2 Logistic回归

$$P_r(Y=1|X)=p(X)\ （1）$$

$$p(X)=\frac{e^{\beta_0+\beta_1X}}{1+e^{\beta_0+\beta_1X}}\ （2）$$

$$\frac{p(X)}{1-p(X)}=e^{\beta_0+\beta_1X}$$

$$\log(\frac{p(X)}{1-p(X)})=\beta_0+\beta_1X$$

β_0 和β_1 可以由极大似然法估计得到，把所估计出来的参数和$X=x$ 代入（2） 就可以知道$p(X)$ 的值，也就是（1） 中$P_r(Y=1|X)$ 的值，即Y 分到“1”类的概率，如果这个概率>0.5,则就将所要预测的响应变量分到“1”类，若概率<0.5,则将该响应变量分到“0”类

```
library(ISLR)
library(MASS)
#划分训练样本和测试样本
dat <- Smarket[, c("Year", "Lag1", "Lag2", "Direction")]
train <- (dat$Year < 2005)#输出布尔值
dattrain <- dat[train,] #以2001-2004年的数据为训练样本
dattest <- dat[!train,] #2005年的数据为测试样本

#Logistic回归建模
glm.fit <- glm(Direction ~ Lag1+Lag2, family = binomial, data = dattrain)#训练集：2001-2004的数据

#Logistic回归预测
glm.probs <- predict(glm.fit, dattest, type = "response")#测试集：2005年的数据，得到2005年的走势预测
glm.pred <- rep("Down", 252) #创建一个有1250个“DOWN”元素的向量
glm.pred[glm.probs > 0.5] = "UP" #把向量中上涨概率大于0.5的变成“UP”

tablogs<-table(glm.pred, dattest$Direction)
tablogs
```

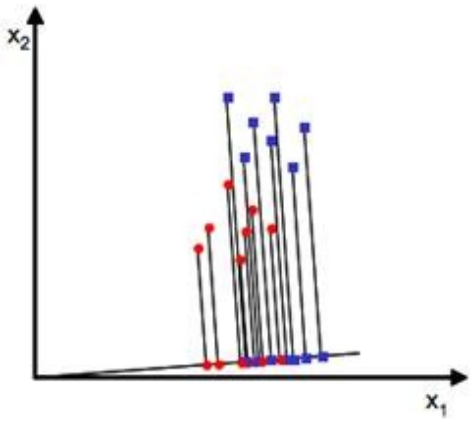
```
##
## glm.pred Down  Up
##     Down   35  35
##     UP     76 106
```

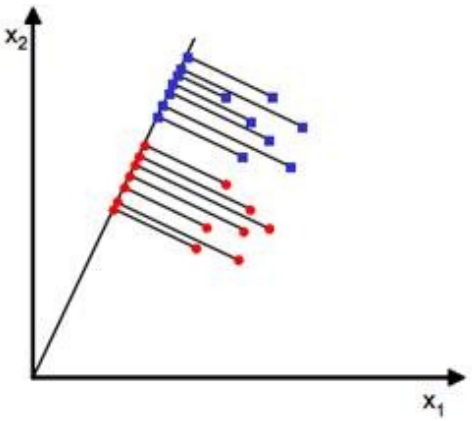
```
sum(diag(prop.table(tablogs))) #计算正确分类率
```

```
## [1] 0.5595238
```

3 线性判别分析

线性判别分析





$$W = A'X$$

3.1 Bayes方法

```
#数据划分
dattrain1 <- dattrain[dattrain$Direction == "Up",]
dattrain2 <- dattrain[dattrain$Direction == "Down",]
dattrain1 <- as.matrix(dattrain1[, c("Lag1", "Lag2")])
dattrain2 <- as.matrix(dattrain2[, c("Lag1", "Lag2")])
```

准则：

1、后验概率最大

2、错判损失最小，以下认为错判损失相同

把$X = x$ 分到能使$\delta_k(x) = -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \log\pi_k$ 最大的那一个“k ”类(后验概率最大)

```
#计算两类样本均值(列均值)
dattrain1_mean <- apply(dattrain1, 2, mean)
dattrain2_mean <- apply(dattrain2, 2, mean)
pred <- rep("Down", nrow(as.matrix(dattest)))
post <- rep(0, nrow(as.matrix(dattest)))
p1 <- log(nrow(dattrain1) / nrow(dattrain))
p2 <- log(nrow(dattrain2) / nrow(dattrain))
for (i in 1:nrow(as.matrix(dattest))) {
    delta1 <- -1 / 2 * (as.matrix(dattest[, c("Lag1", "Lag2")])[i,] - dattrain1_mea
n) %*% solve(cov(dattrain1)) %*% (as.matrix(dattest[, c("Lag1", "Lag2")])[i,] - dattr
ain1_mean) + p1
    delta2 <- -1 / 2 * (as.matrix(dattest[, c("Lag1", "Lag2")])[i,] - dattrain2_mea
n) %*% solve(cov(dattrain2)) %*% (as.matrix(dattest[, c("Lag1", "Lag2")])[i,] - dattr
ain2_mean) + p2

    if (delta1 > delta2) {
        pred[i] = "Up"
        post[i] = exp(delta1) / (exp(delta1) + exp(delta2))
    } else {
        pred[i] = "Down"
        post[i] = exp(delta2) / (exp(delta1) + exp(delta2))
    }
}
Bayesresults <- cbind(dattest, as.data.frame(pred), post)
Bayesresults[1:10,]
```

```
##      Year   Lag1   Lag2 Direction pred      post
## 999  2005 -0.134  0.008      Down   Up 0.5098307
## 1000 2005 -0.812 -0.134      Down   Up 0.5212584
## 1001 2005 -1.167 -0.812      Down   Up 0.5334764
## 1002 2005 -0.363 -1.167        Up   Up 0.5232352
## 1003 2005  0.351 -0.363      Down   Up 0.5068117
## 1004 2005 -0.143  0.351        Up   Up 0.5057980
## 1005 2005  0.342 -0.143      Down   Up 0.5048589
## 1006 2005 -0.610  0.342        Up   Up 0.5124110
## 1007 2005  0.398 -0.610      Down   Up 0.5081952
## 1008 2005 -0.863  0.398        Up   Up 0.5152596
```

```
sum(Bayesresults$Direction == Bayesresults$pred) / nrow(dattest) #计算正确分类率
```

```
## [1] 0.5634921
```

3.2 LDA分类器

分类器会将$X = x$ 分到使$\delta_k(x) = x \cdot \mu_k/\sigma^2 - (\mu_k^2)/(2\sigma^2) + log\pi_k$ 最大的那一个“k ”类

```
#前提假设： 每一类中的观测都来自于一个均值不同，方差相同的正态分布假设上
#用LDA建模
lda.fit <- lda(Direction ~ Lag1 + Lag2, data = dattrain)
lda.fit
```

```
## Call:
## lda(Direction ~ Lag1 + Lag2, data = dattrain)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
##
## Coefficients of linear discriminants:
##             LD1
## Lag1 -0.6420190
## Lag2 -0.5135293
```

```
#用LDA预测
lda.pred <- predict(lda.fit, dattest)
tablda <- table(dattest$Direction, lda.pred$class)
tablda
```

```
##
##        Down  Up
##   Down   35  76
##   Up     35 106
```

```
sum(diag(prop.table(tablda))) #计算正确分类率
```

```
## [1] 0.5595238
```

```
cbind(dattest, lda.pred$class)[1:10,]#输出原数据的类别和训练后的类别
```

```
##      Year   Lag1   Lag2 Direction lda.pred$class
## 999  2005 -0.134  0.008      Down             Up
## 1000 2005 -0.812 -0.134      Down             Up
## 1001 2005 -1.167 -0.812      Down             Up
## 1002 2005 -0.363 -1.167        Up             Up
## 1003 2005  0.351 -0.363      Down             Up
## 1004 2005 -0.143  0.351        Up             Up
## 1005 2005  0.342 -0.143      Down             Up
## 1006 2005 -0.610  0.342        Up             Up
## 1007 2005  0.398 -0.610      Down             Up
## 1008 2005 -0.863  0.398        Up             Up
```

```
round(lda.pred$posterior, 3)[1:10,] #输出前十个后验概率，保留三位小数
```

```
##       Down    Up
## 999  0.490 0.510
## 1000 0.479 0.521
## 1001 0.467 0.533
## 1002 0.474 0.526
## 1003 0.493 0.507
## 1004 0.494 0.506
## 1005 0.495 0.505
## 1006 0.487 0.513
## 1007 0.491 0.509
## 1008 0.484 0.516
```

3.3 QDA分类器

分类器会将$X = x$ 分到使$\delta_k(x) = x^T\Sigma^(-1)\mu_k - 1/2\mu_k^T\Sigma^(-1)\mu_k + log\pi_k$ 最大的那一个“k ”类

```
#前提假设：  每一类的观测都服从一个均值不同协方差矩阵相同的多元高斯分布
#用QDA建模
qda.fit <- qda(Direction ~ Lag1 + Lag2, data = dattrain)
qda.fit
```

```
## Call:
## qda(Direction ~ Lag1 + Lag2, data = dattrain)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
```

```
#用QDA预测
qda.pred <- predict(qda.fit, dattest)
tabqda <- table(dattest$Direction, qda.pred$class)
tabqda
```

```
##
##        Down  Up
##   Down   30  81
##   Up     20 121
```

```
sum(diag(prop.table(tabqda))) #计算正确分类率
```

```
## [1] 0.5992063
```

```
cbind(dattest, qda.pred$class)[1:10,] #输出原数据的类别和训练后的类别
```

```
##      Year   Lag1   Lag2 Direction qda.pred$class
## 999  2005 -0.134  0.008      Down             Up
## 1000 2005 -0.812 -0.134      Down             Up
## 1001 2005 -1.167 -0.812      Down             Up
## 1002 2005 -0.363 -1.167        Up             Up
## 1003 2005  0.351 -0.363      Down             Up
## 1004 2005 -0.143  0.351        Up             Up
## 1005 2005  0.342 -0.143      Down             Up
## 1006 2005 -0.610  0.342        Up             Up
## 1007 2005  0.398 -0.610      Down             Up
## 1008 2005 -0.863  0.398        Up             Up
```

```
round(qda.pred$post, 3)[1:10,]#输出前十个后验概率，保留三位小数
```

```
##       Down    Up
## 999  0.487 0.513
## 1000 0.476 0.524
## 1001 0.464 0.536
## 1002 0.474 0.526
## 1003 0.490 0.510
## 1004 0.491 0.509
## 1005 0.492 0.508
## 1006 0.485 0.515
## 1007 0.489 0.511
## 1008 0.482 0.518
```

```
#比较LDA和QDA
sum(diag(prop.table(tabqda))) > sum(diag(prop.table(tablda)))
```

```
## [1] TRUE
```

结果解读：QDA模型的正确分类率大于LDA模型，说明这组数据QDA的分类效果比LDA的分类效果好

3.4 距离判别法公式分类

计算各类重心即各组的均值，某次观测分到i类，如果它离第i类的重心距离最近（这里的距离可以用欧氏距离，但是马氏距离用的多）

距离计算：$D(X,G_i)=(X-\mu_i^{'})(\Sigma_i^{-1})(X-\mu_i),i=1,2$ 其中，$\mu_1,\mu 2,\Sigma_1,\Sigma_2$ 分别是总体 G_1,G_2 的均值向量和协方差矩阵

二分类（两总体）当$\Sigma_1=\Sigma_2=\Sigma$ 时（多分类也是类似的做法）

距离比较：$W(X)=D(X,G_2)-D(X,G_1)=2[X-\frac{1}{2}(\mu_1+\mu_2)]^{'}\Sigma^{-1}(\mu_1-\mu_2)$

判别标准：

(1) $W(X)>0,X\in G_1$

(2) $W(X)<0,X\in G_2$

(3) $W(X)=0$, 待判

```
#计算两类样本均值(列均值)
dattrain1_mean <- apply(dattrain1, 2, mean)
dattrain2_mean <- apply(dattrain2, 2, mean)
#计算协方差矩阵
sigma <- (cov(dattrain1) * nrow(dattrain1) + cov(dattrain2) * nrow(dattrain2)) / (nro
w(dattrain1) + nrow(dattrain2) - 2)
solve(sigma) #求逆矩阵
```

```
##            Lag1       Lag2
## Lag1 0.66041852 0.01461352
## Lag2 0.01461352 0.66012566
```

```
a <- solve(sigma) %*% (dattrain1_mean - dattrain2_mean) #%*%表示矩阵的乘法,Sigma的逆*（mu_1-mu_2）

b = 1 / 2 * (dattrain1_mean + dattrain2_mean)#两个分类的的均值向量的均值
#判断测试集里的属于哪一类
pred <- rep("Down", nrow(as.matrix(dattest)))##全都是"Down"
for (i in 1:nrow(as.matrix(dattest))) {
    W = t(a) %*% t(as.matrix((dattest[, c("Lag1", "Lag2")] - b)))[, i]

    if (W > 0) {
        pred[i] = "Up"
    } else {
        pred[i] = "Down"
    }
}
Distanceresults <- cbind(dattest, as.data.frame(pred))
Distanceresults[1:10,]
```

```
##      Year   Lag1   Lag2 Direction pred
## 999  2005 -0.134  0.008      Down   Up
## 1000 2005 -0.812 -0.134      Down   Up
## 1001 2005 -1.167 -0.812      Down   Up
## 1002 2005 -0.363 -1.167        Up   Up
## 1003 2005  0.351 -0.363      Down Down
## 1004 2005 -0.143  0.351        Up Down
## 1005 2005  0.342 -0.143      Down Down
## 1006 2005 -0.610  0.342        Up   Up
## 1007 2005  0.398 -0.610      Down   Up
## 1008 2005 -0.863  0.398        Up   Up
```

```
sum(Distanceresults$Direction == Distanceresults$pred) / nrow(dattest) #计算正确分类率
```

```
## [1] 0.547619
```

3.5 Fisher判别法公式分类

(假设是p个因子，二分类)

类之间的变异$(\overline{Y}_1 - \overline{Y}_2)$ 尽可能大，类内部的变异$S_P^2 = \frac{(n_1-1)S_1^2 + (n_2-1)S_2^2}{n_1+n_2-2}$ 尽可能小,也就是 $\lambda = \frac{(\overline{Y}_1 - \overline{Y}_2)^2}{S_P^2}$ 越大越好,其中$\overline{Y}_1 = A^{'}\overline{X}_1, \overline{Y}_2 = A^{'}\overline{X}_2$ 。$\longrightarrow$ 得到适当的X的线性组合A，得到判别函数$Y = A^{'}X = a_1x_1 + a_2x_2 + \cdots + a_px_p$

参考相关文献：$A = \begin{pmatrix} a_1 \\ a_2 \\ \cdots \\ a_p \end{pmatrix} = \begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1p} \\ S_{21} & S_{22} & \cdots & S_{2p} \\ \cdots & & & \\ S_{p1} & S_{p2} & \cdots & S_{pp} \end{bmatrix}^{-1} \begin{pmatrix} \overline{X}_{11} - \overline{X}_{21} \\ \overline{X}_{12} - \overline{X}_{22} \\ \cdots \\ \overline{X}_{1p} - \overline{X}_{2p} \end{pmatrix}$

判别的界值：　　$= \dfrac{n_1\overline{Y}_1 + n_2\overline{Y}_2}{n_1 + n_2}$

判别标准：　y_0

(1) $\overline{Y}_1 < \overline{Y}_2$ 时，$Y < y_0$ ，则$X \in G_1$,否则$X \in G_2$

(2) $\overline{Y}_1 > \overline{Y}_2$ 时，$Y < y_0$ ，则$X \in G_2$,否则$X \in G_1$

(3) $\overline{Y}_1 = \overline{Y}_2$ 时，待判

```
c <- 1 / (nrow(dattrain1) + nrow(dattrain2) - 2) * a #%*%表示矩阵的乘法
y0 <- (dattrain1_mean %*% c * nrow(dattrain1) + dattrain2_mean %*% c * nrow(dattrain
2)) / nrow(dattrain)
pred <- rep("Down", nrow(as.matrix(dattest)))
for (i in 1:nrow(as.matrix(dattest))) {
    y = t(c) %*% t(as.matrix(dattest[, c("Lag1", "Lag2")]))[, i]
    if (y > y0) {
        pred[i] = "Up"
    } else {
        pred[i] = "Down"
    }
}
Fisherresults <- cbind(dattest, as.data.frame(pred))
Fisherresults[1:10,]
```

```
##      Year   Lag1   Lag2 Direction pred
## 999  2005 -0.134  0.008      Down   Up
## 1000 2005 -0.812 -0.134      Down   Up
## 1001 2005 -1.167 -0.812      Down   Up
## 1002 2005 -0.363 -1.167        Up   Up
## 1003 2005  0.351 -0.363      Down Down
## 1004 2005 -0.143  0.351        Up Down
## 1005 2005  0.342 -0.143      Down Down
## 1006 2005 -0.610  0.342        Up   Up
## 1007 2005  0.398 -0.610      Down   Up
## 1008 2005 -0.863  0.398        Up   Up
```

```
sum(Fisherresults$Direction == Fisherresults$pred) / nrow(dattest) #计算正确分类率
```

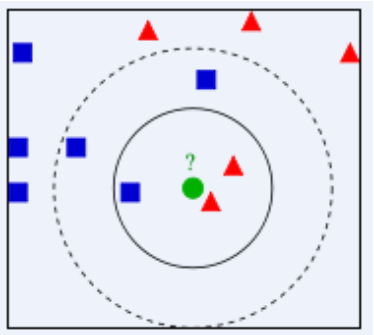
```
## [1] 0.547619
```

结果解读：在这个数据下，用Fisher和距离判别法公式实际得到的结果是一样的

---

4 KNN分类器

KNN决策过程



$K=3$ 时，离绿色的点最近的3个点为2个红色点，1个蓝色点，因此把绿点分到红色类

$K=5$ 时，离绿色的点最近的5个点为2个红色点，3个蓝色点，因此把绿点分到蓝色类

```
library(class)
#knn()函数要求参数是矩阵形式
train.X <-as.matrix(dattrain[,c(2,3)])#训练集自变量的矩阵
test.X <-as.matrix(dattest[,c("Lag1","Lag2")])#测试集自变量的矩阵
train.Direction <-as.matrix(dattrain[,"Direction"])#测试集因变量的矩阵

set.seed(1)#设置随机种子是为了结果具有可重复性
#K=1
knn.predK1 <-knn(train.X, test.X, train.Direction, k = 1)
tabK1<-table(knn.predK1, dattest[,"Direction"])
tabK1
```

```
##
## knn.predK1 Down Up
##       Down   43 58
##       Up     68 83
```

```
sum(diag(prop.table(tabK1))) #计算正确分类率
```

```
## [1] 0.5
```

```
#K=3
knn.predK3 <- knn(train.X, test.X, train.Direction, k = 3)
tabK3<-table(knn.predK3, dattest[,"Direction"])
tabK3
```

```
##
## knn.predK3 Down Up
##       Down   48 54
##       Up     63 87
```

```
sum(diag(prop.table(tabK3))) #计算正确分类率
```

```
## [1] 0.5357143
```

```
#K=9
knn.predK9 = knn(train.X, test.X, train.Direction, k = 9)
tabK9<-table(knn.predK9, dattest[,"Direction"])
tabK9
```

```
##
## knn.predK9 Down Up
##       Down   45 61
##       Up     66 80
```

```
sum(diag(prop.table(tabK9))) #计算正确分类率
```

```
## [1] 0.4960317
```

5 ROC曲线(不会画)

		预测			
		0	1	total	假阳性率
真实	0	a	b	a+b	b/(a+b)
	1	c	d	c+d	d/(c+d)
	total	a+c	b+d	a+b+c+d	真阳性率（势、灵敏度）
	预测阴性率	a/(a+c)	d/(b+d)	预测阳性率（精确度）	

6 分类方法比较

```
#results表示相对应分类方法的正确区分率
methods<-c("Logistic回归","Bayes方法","LDA分类器","QDA分类器","距离判别法公式分类","Fish
er判别法公式分类","KNN(K=1)","KNN(K=3)","KNN(K=9)")
results<-c(sum(diag(prop.table(tablogs))),sum(Bayesresults$Direction == Bayesresults
$pred) / nrow(dattest),sum(diag(prop.table(tablda))),sum(diag(prop.table(tabqda))),su
m(Distanceresults$Direction == Distanceresults$pred) / nrow(dattest),sum(Fisherresult
s$Direction == Fisherresults$pred) / nrow(dattest),sum(diag(prop.table(tabK1))),sum(d
iag(prop.table(tabK3))),sum(diag(prop.table(tabK9))))
comparement<-data.frame(methods,results)
comparement
```

```
##                 methods   results
## 1         Logistic回归 0.5595238
## 2            Bayes方法 0.5634921
## 3            LDA分类器 0.5595238
## 4            QDA分类器 0.5992063
## 5   距离判别法公式分类 0.5476190
## 6 Fisher判别法公式分类 0.5476190
## 7              KNN(K=1) 0.5000000
## 8              KNN(K=3) 0.5357143
## 9              KNN(K=9) 0.4960317
```

结果解读：上述结果表示在这组数据的情况下，QDA分类器的分类效果是相对最好的，KNN的K取值从1->3,分类效果有所提升，但从3->9，分类结果反而变差，说明随着K的继续增加，结果不会再有更进一步的改进

6.1 LDA与QDA的比较

观测训练数据量相对较**小**，则LDA优于QDA，反之，QDA优于LDA.因为当数据量**大**时，K类协方差矩阵相同的假设就很容易站不住脚

LDA是关于X的一次函数，QDA是关于X的二次函数。QDA的光滑性更高，即拥有高方差，低偏差。而LDA的光滑性不如QDA，也即拥有更低的方差，且具有改善预测效果的潜力

在具体数据中，QDA与LDA哪个分类器较好，可以在同一个ROC曲线中绘制QDA,LDA，Bayes三个分类器的曲线进行判断

6.2 LDA与Logistic、KNN的比较

LDA与Logistic两者都是关于x的线性函数，意味着都产生线性决策边界，而KNN是彻底的非参数方法，对决策边界的形状没有任何假设（也即没有明确的分界线）

当高斯分布的假设近似成立时，LDA比Logistic好，当该假设不成立时，Logistic模型更好

当决策边界高度非线性时，KNN优于LDA与Logistic

KNN的**缺点**在于：（1）必须小心选择光滑水平（2）由于不需要确定、给出参数，因此没有办法确定哪些变量是重要的

6.3 总结

光滑度（从低到高）：

1、LDA与Logistic 2、QDA 3、KNN

QDA的光滑度不如KNN但优于LDA与Logistic，对决策边界的形状做了一些假设，且不是线性的。因此QDA的在实际中的应用更广泛一些

如果对于同一观测，LDA与Logistic很好，且Logistic优于LDA,而QDA很差，考虑该观测可能是非正态分布（可能是t分布）

第四讲 支持向量机

1 前言

通常把最大间隔分类器、支持向量分类器和支持向量机都简单叫做“**支持向量机**”，但他们是具有严格的区分的（下面会区分）

支持向量机被认为是适应性最广的分类器之一

---

2 最大间隔分类器(MMC)

2.1 超平面

p 维空间中超平面是$p-1$ 维的平面仿射子空间，比如说，2维空间中的超平面是
$\beta_0+\beta_1X_1+\beta_2X_2=0$,是一条直线（1维）； 3维空间中的超平面是
$\beta_0+\beta_1X_1+\beta_2X_2+\beta_3X_3=0$,是一个平面（2维）； p 维空间中的超平面就是
$\beta_0+\beta_1X_1+\beta_2X_2+\ldots+\beta_pX_p=0$,这是$p-1$ 维的。

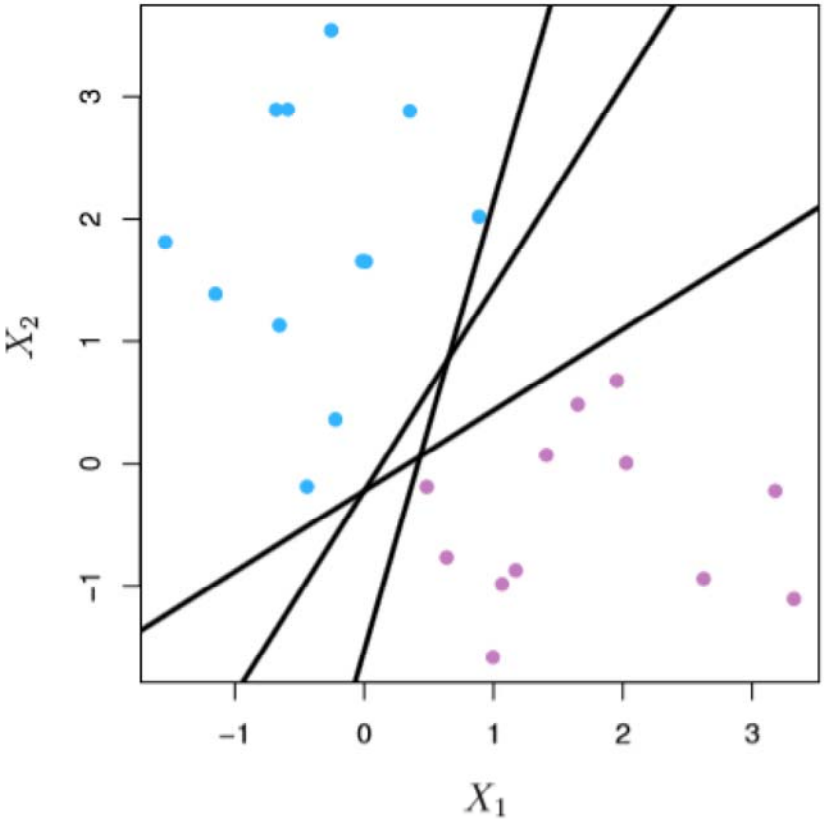
超平面可以把一个空间分割成两个部分

2.2 最大间隔超平面

由于分割超平面不唯一，需要找到一个分割超平面，使所有到这个超平面的训练观测的垂直最小距离最大，这样的超平面就叫做最大间隔超平面。比如说下图，有好几条线（分割平面）可以用来分割红、蓝点（左图画出了三种分割平面，当然还有更多），右图标出了所有到**这一个**超平面的训练观测的垂直最小距离（有三个点），以同样的方式标出另外两个超平面与观测的垂直最小距离，可以发现右图所示的超平面有**最大**的**垂直最小距离**。

相当于是这三个点决定了这个最大间隔超平面，如果这三个点的位置有所改变，最大间隔超平面也会改变，所以这三个点起到了某种支持作用，被称作**“支持向量”**。这也意味着，最大间隔超平面只由支持向量决定，与其他向量无关。

最大间隔超平面对单个观测的变化极其敏感，有可能会出现**过拟合**情况



2.3 构建最大间隔分类器

n 个观测$x_1,x_2,\ldots,x_n$ 和其对应的类别标签$y_1,y_2,\ldots,y_n\in\{-1,1\}$，假设超平面是
$\beta_0+\beta_1x_{i1}+\ldots+\beta_px_{ip}=0$

那么最大间隔分类器就是一个优化问题$\max_{\beta_0,\beta_1,...,\beta_p} M$，

满足$\sum_{j=1}^{p} \beta_j^2 = 1(1)$

$$yi(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M, i = 1, \ldots, n(2)$$

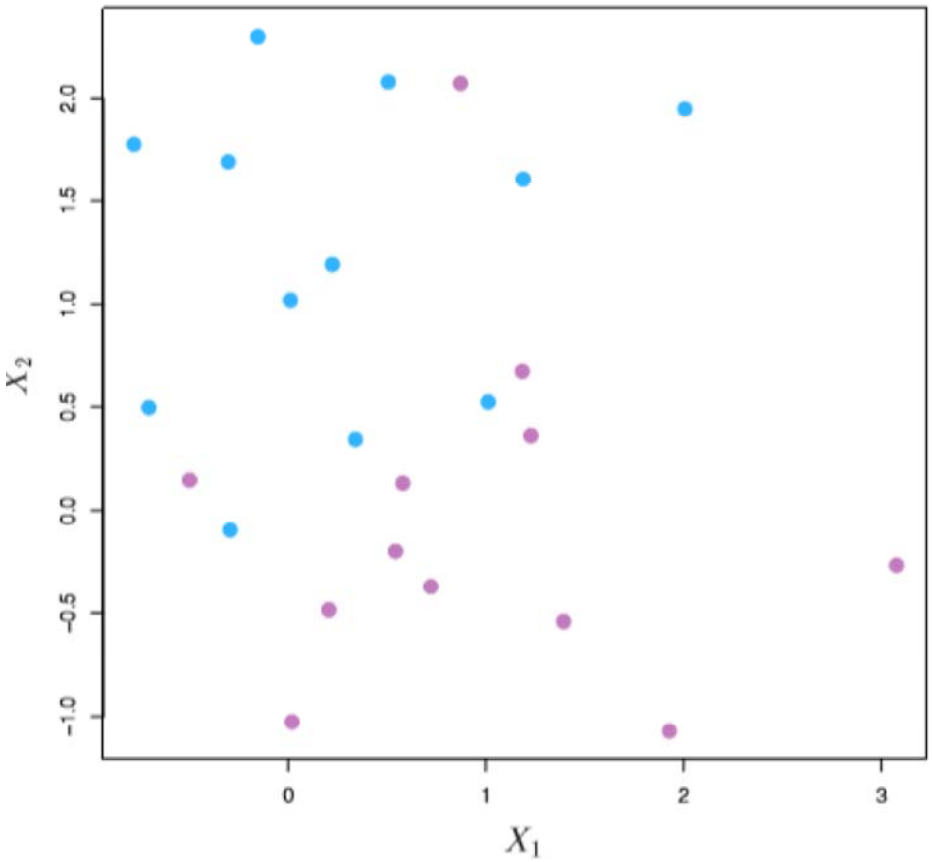
(1)式的约束使得第i 个观测到超平面的垂直距离为$yi(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip})$，使得(2)式有意义。(2)式保证了每个观测都在超平面正确的一侧,且与超平面的距离至少为M 。所以问题就转化为了去找$\beta_0, \beta_1, \ldots, \beta_p$ 以此最大化M ，这边通常使用**引入拉格朗日对偶变量+序列最小最优化算法(SMO算法)**来求解（不讲）

通过最大化间隔，使得该分类器对数据分类是具有了最大把握。使用最大间隔而非最小间隔，是因为最大间隔能获得最大稳定性与区分的确信度（离最大间隔越远的观测的分类越具有可信度），从而获得良好的推广能力（超平面的M 越大，这个分类器的性能越好，推广能力也就越好）

---

3 支持向量分类器(SVC)

上面讲的都是线性可分的情况，但是大多数情况下，是线性不可分的，比如说下图的情况，就无法用一条直线来分割



那么在线性不可分情况下，就要考虑将最大间隔分类器进行推广——>**支持向量分类器**

3.1 概念

支持向量分类器也叫做软件隔分类器，允许小部分观测误分以保证大部分观测可以被更好地分类，提高分类器的稳定性

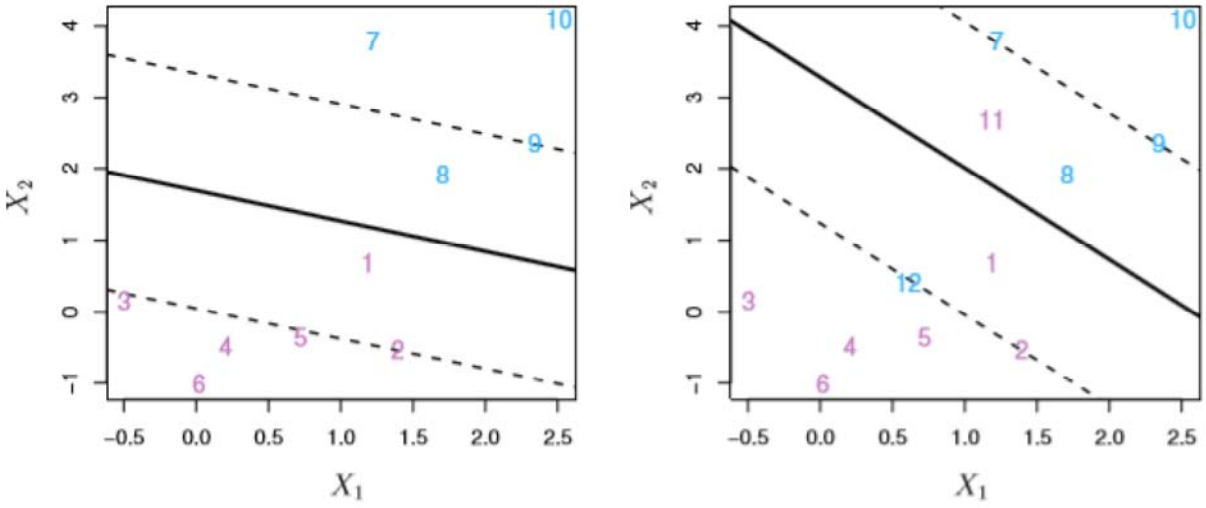
观测误分有两种情况：

（1）落在间隔错误的一侧（离超平面太近）

（2）落在超平面错误的一侧（入曹营）

左图的**1、8**两个点，穿过了间隔，落入了间隔错误的一侧，但还是在超平面正确的一侧

右图的**1、8**两个点，穿过了间隔，落入了间隔错误的一侧，但还是在超平面正确的一侧，**11、12**两个点直接穿过了超平面落在了超平面错误的一侧



3.2 构建支持向量分类器

n 个观测$x_1, x_2, \ldots, x_n$ 和其对应的类别标签$y_1, y_2, \ldots, y_n \in \{-1, 1\}$，假设超平面是 $\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} = 0$

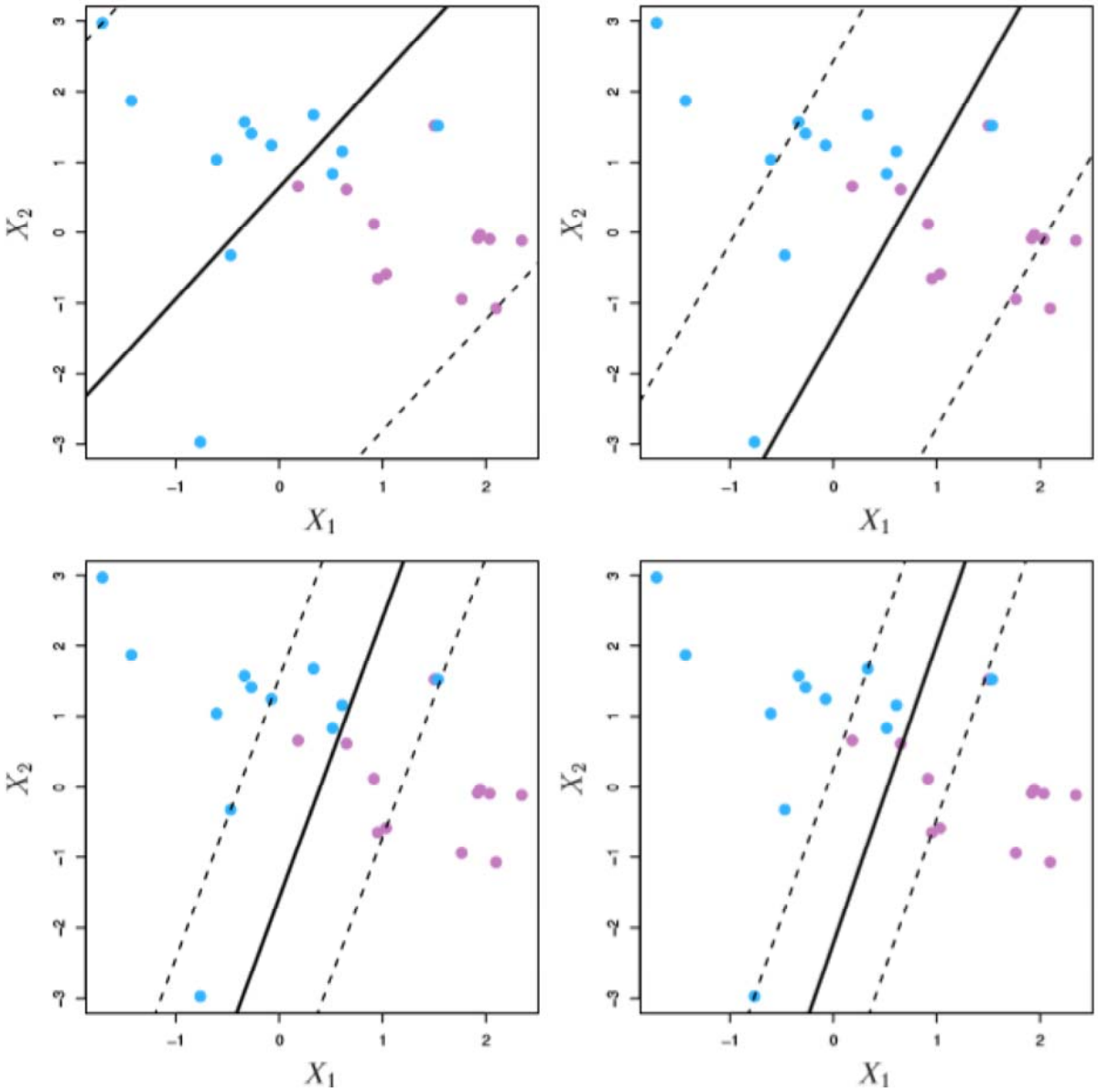
那么支持向量分类器就是一个优化问题$\max \min_{\beta_0, \beta_1, \ldots, \beta_p, \epsilon_1, \epsilon_2, \ldots, \epsilon_n} M$，

满足$\sum_{j=1}^{p} \beta_j^2 = 1 (1)$

$$yi(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i), i = 1, \ldots, n (2)$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C (3)$$

$\epsilon_1, \epsilon_2, \ldots, \epsilon_n$ 是松弛变量，使得分类器允许训练观测中有小部分观测可以落在间隔的错误的一侧($\epsilon_i > 0$)或是超平面错误的一侧($\epsilon_i > 1$)，C 是非负调节参数，代表了容忍度，C 越大，则间隔越大，落在间隔错误的一侧的观测就越多。

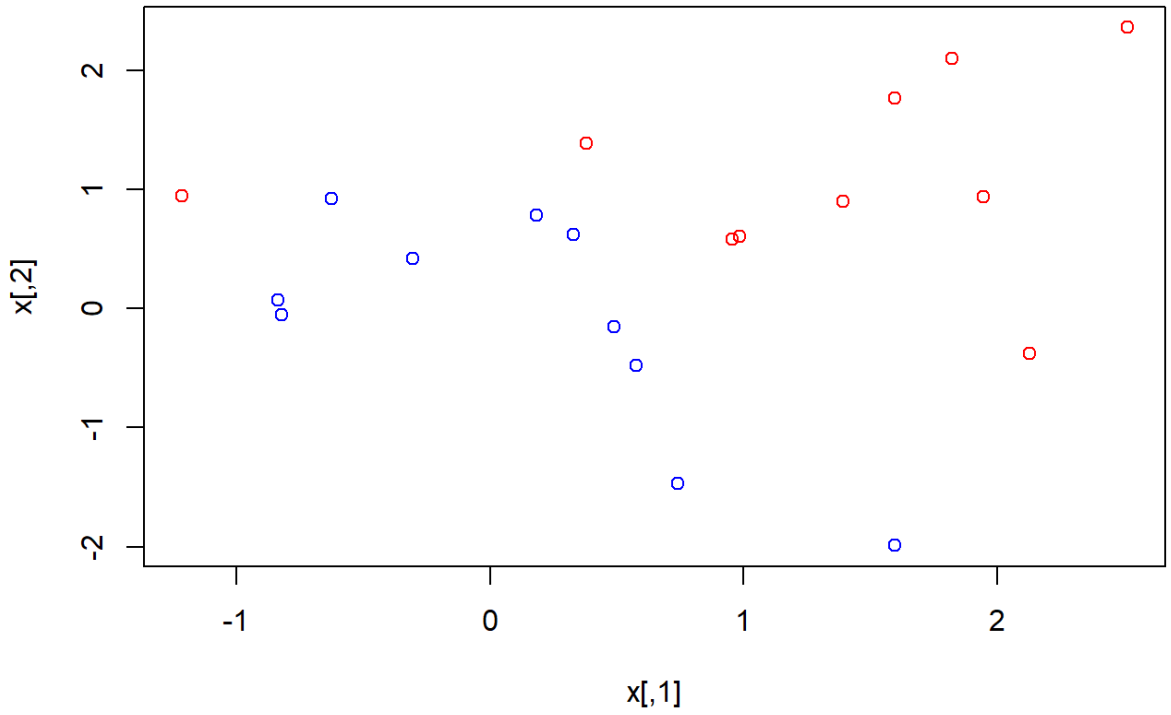


C 越来越小，间隔越来越小，落在间隔上的观测也越来越少

在支持向量分类器中的支持向量是那些落在间隔上和落在间隔错误一侧的观测，以上图（右）为例，**1、2、7、8、9、11、12**是该分类器的支持向量。当C 比较大的时候，穿过间隔的观测就多，这意味着支持向量的数量会增加，从而确定该超平面涉及到的观测点就相应地增加了，使得该分类器具有较低的方差和较高的偏差

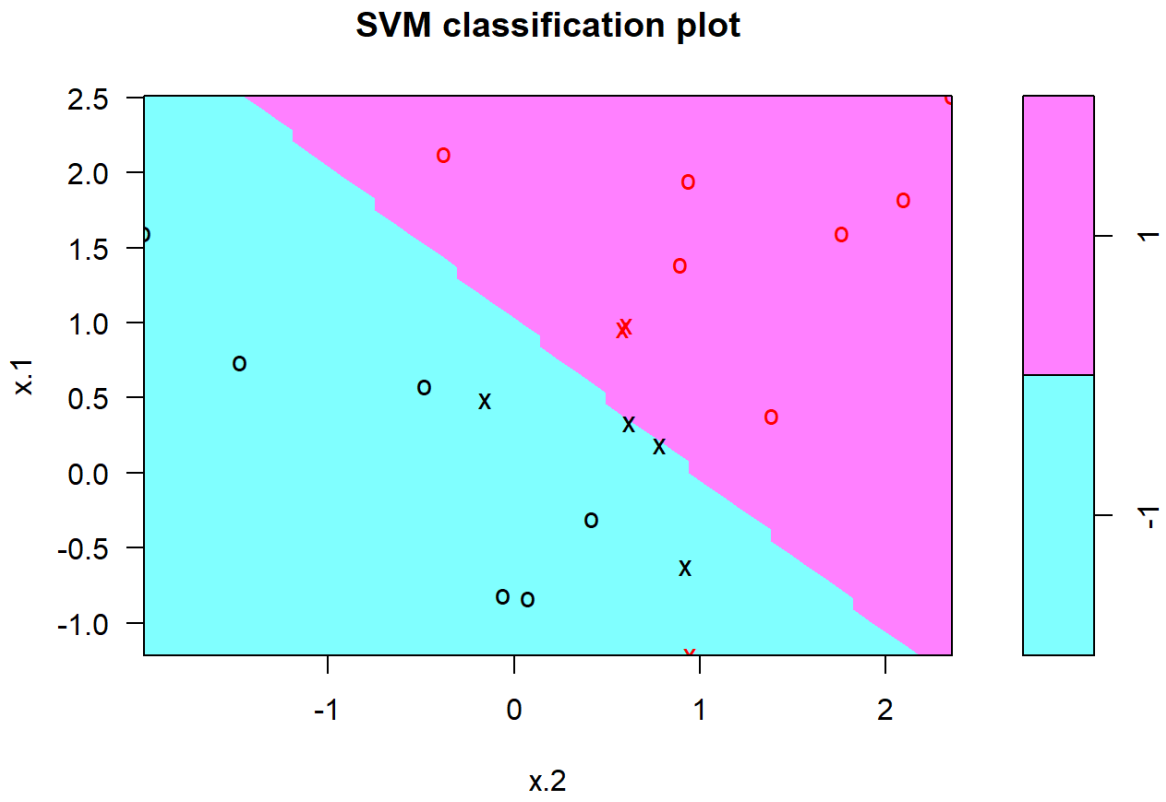
3.3 实践

```
#先建立线性可分的数据集
set.seed(1)
x = matrix(rnorm(20 * 2), ncol = 2)
y = c(rep(-1, 10), rep(1, 10))
x[y == 1,] = x[y == 1,] + 1
plot(x, col = (3 - y))
```




```
dat = data.frame(x = x, y = as.factor(y))
```

```
library(e1071)
#由于这边用了线性核函数，不需要去考虑gamma参数
svmfit3.3.1 = svm(y ~ ., data = dat, kernel = "linear", cost = 10, scale = FALSE)
plot(svmfit3.3.1, dat)#“X”是支持向量，“O”是非支持向量
```




```
svmfit3.3.1$index#用以确定是哪7个支持向量
```

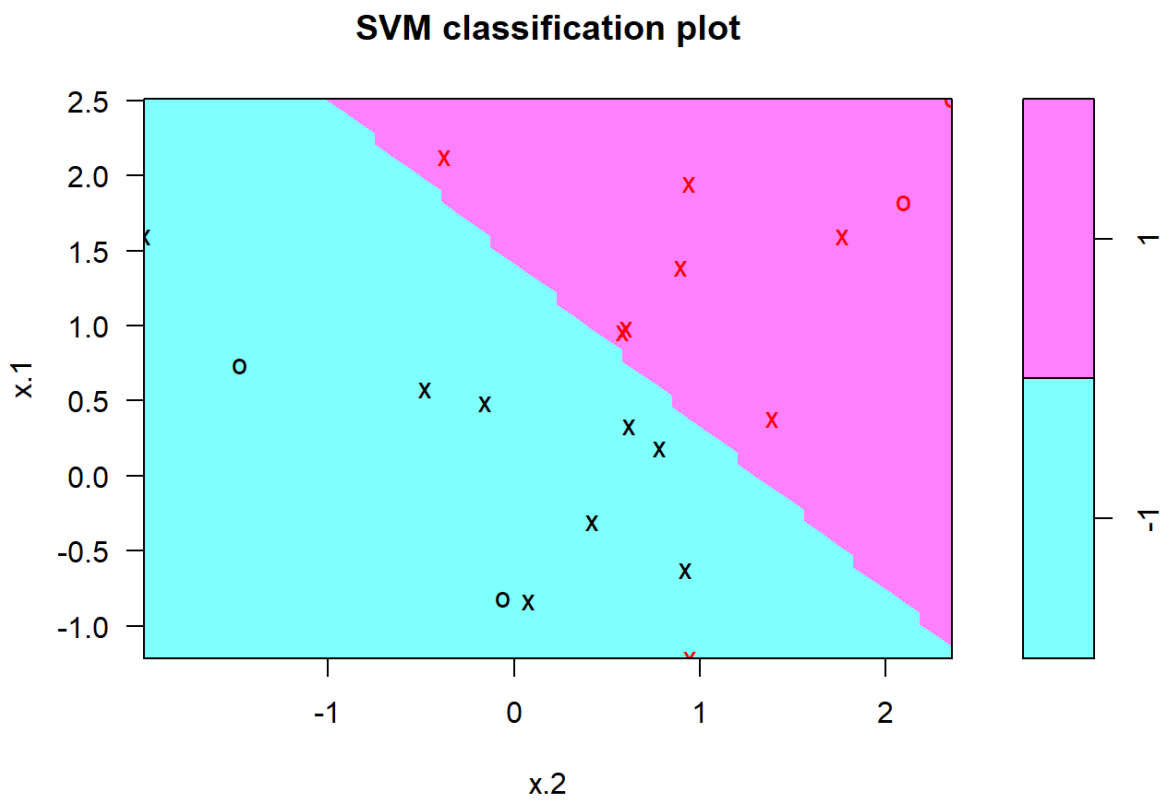
```
## [1]  1  2  5  7 14 16 17
```

```
summary(svmfit3.3.1)
```

```
##
## Call:
## svm(formula = y ~ ., data = dat, kernel = "linear", cost = 10,
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.5
##
## Number of Support Vectors:  7
##
##  ( 4 3 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

结果解读：cost为10的分类器，gamma的值是默认的特征变量的个数的倒数，模型将数据分为两类，有7个支持向量，一类4个，一类3个

```
#改变cost值
svmfit3.3.2 = svm(y ~ ., data = dat, kernel = "linear", cost = 0.1, scale = FALSE)
plot(svmfit3.3.2, dat)
```




```
svmfit3.3.2$index
```

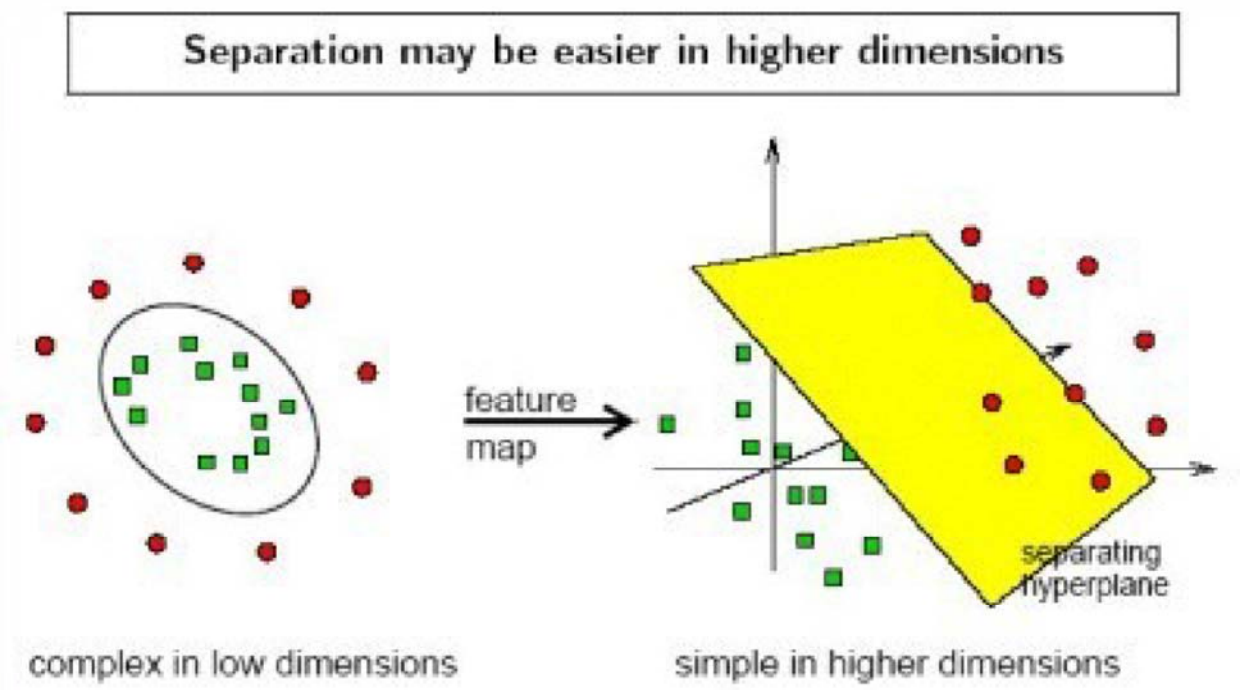
```
##  [1]  1  2  3  4  5  7  9 10 12 13 14 15 16 17 18 20
```

结果解读：该模型的cost相比前一个模型小，显然从图中就可以看到，支持向量比上一个模型多了很多

4 核函数

4.1 引入核函数

到支持向量分类器为止，还都是线性回归。但是很多时候，预测变量与响应变量之间的关系是非线性的，这个时候再使用线性回归就会使分类结果大打折扣。我们需要使用预测变量的函数来扩大特征空间，将变量映射到一个高维特征空间，以此使其线性可分（如下图）。在引入支持向量机的概念之前，我们先介绍一下核函数。



原始的扩大特征空间的方法诸如使用二次多项式，三次多项式，可能会导致出现数量庞大的特征，导致**计算量**激增，所以我们引入**核函数**

对于线性学习器，考虑以下函数

$f(x) = \sum_{i=1}^{p} \beta_i \phi_i(x) + \beta_0$

分类函数也就是决策规则可以用测试点和训练点的内积表示(引入了拉格朗日对偶变量α 后)

$f(x) = \sum_{i=1}^{p} \alpha_i y_i \langle \phi_i(x_i), \phi(x) \rangle + \beta_0$ （4.2中解释径向核函数还要提到这个公式）

原始的映射方法建立非线性学习器步骤：

（1）使用一个非线性映射将数据变换到一个特征空间$x->\phi(x)$

（2）在特征空间计算内积$\langle \phi_i(x_i), \phi(x) \rangle$

（3）在特征空间使用线性学习器分类

使用核函数方法可以在特征空间中直接计算内积，也即合并了上面的（1）（2）这两个步骤

下面给出核函数的数学定义：

核是一个函数κ，对所有$x, z \in \chi$，满足$\kappa(x, z) = \langle \phi(x), \phi(z) \rangle$

使用恰当的核函数来代替内积，可以隐式地将非线性训练数据映射到高维空间，而不增加可调参数的个数

补充：任何将计算表示为数据点的内积的方法，都可以用核方法进行非线性扩展

4.2 几个核函数

下面的（1）（2）（3）（4）是**e1071包的svm()函数**中的四种核函数

（1）线性核函数$\kappa(x_i, x_j) = \langle x_i, x_j \rangle$

（2）多项式核函数$\kappa(x_i, x_j) = (\gamma \langle x_i, x_j \rangle + R)^d$

适合正交归一化的数据（**向量正交且模为1**），属于全局核函数，允许相距很远的数据点对核函数的值有影响。d越大，映射的维度就越高，计算量也就越大，要注意d过大可能会出现过拟合。

（3）径向核函数

$$\kappa(\quad,\quad)=\exp\Big(-\gamma\sum_{m=1}^{p}(\qquad-\qquad)^2\Big)=\exp\big(-\gamma|\quad-\qquad|^2\big)$$

高斯核函数$\kappa(x_i,x_j)=\exp\left(-\frac{|x_i-x_j|^2}{2\sigma^2}\right)$ 实际就是径向核函数 $x_{im}\quad x_{jm}\quad x_i\quad x_j$

对数据中存在的噪声具有良好的抗干扰能力，属于局部和函数，当数据点离中心点变远时，取值会变小，随着gamma的减小（σ^2 的增大），函数的作用范围会变小（gamma决定函数的作用范围）

补充：径向核函数是如何起作用的：

如果测试观测x 离训练观测x_i 很远，那么径向核函数$\kappa(\phi(x_i),\phi(x))$ 就会很小，那么在分类函数中，就意味着这个训练观测x_i 对测试观测x 几乎没有影响，也就意味着，距离x 远的训练观测x_i 对预测x 的类别几乎没有帮助

注意在e1071包的svm()函数中，用的是径向核函数的表达式而非高斯核函数的表达式

（4）Sigmoid核函数$\tanh(\gamma\langle x_i,x_j\rangle+R)$（我这边没用过）

来源于神经网络，广泛应用于深度学习和机器学习中，此时，SVM实现的是一种多层感知器神经网络

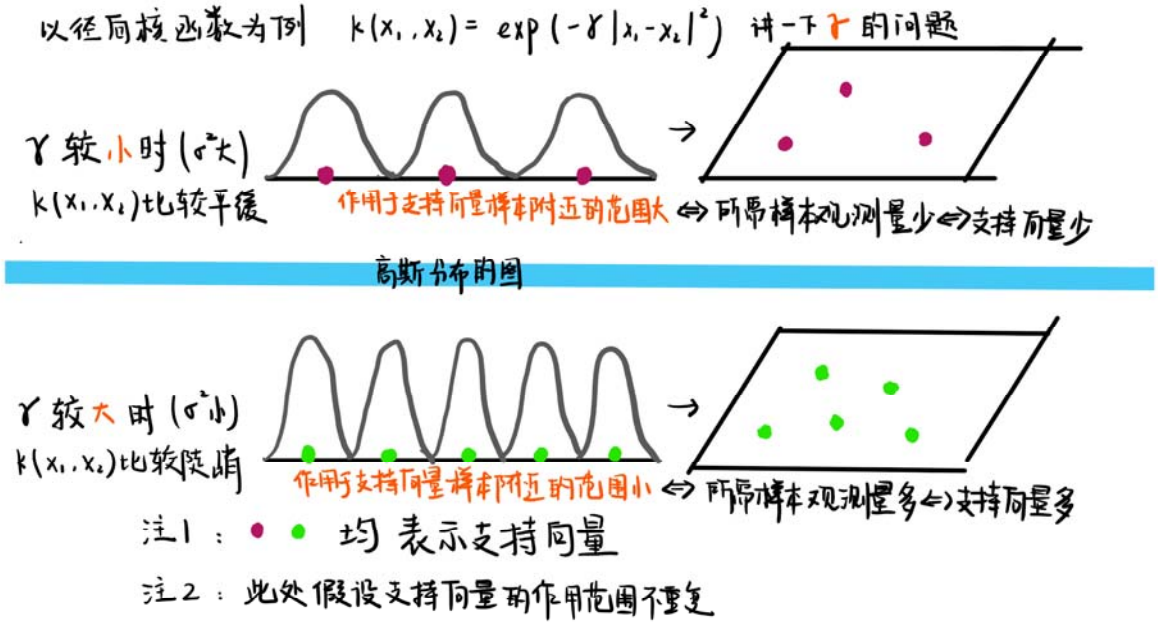
（5）字符串核函数

是定义在字符串集合上的核函数，度量一对字符串的相似度，在文本分类、信息检索方面有应用

svm()函数中有两个参数，gamma和cost 从上述式子来看，（1）是不需要考虑gamma的，即使用系统默认（默认是特征变量个数的倒数）即可，（2）（3）（4）均要考虑gamma参数的选取。cost参数与上面模型中的C的意义（容忍度）不同，意为观测穿过间隔的成本，cost越小，间隔越宽，穿过间隔就更容易，支持向量就越多。这两者对于模型的意义刚好相反。

4.3 参数详解

gamma参数



γ 越大，也即 σ^2 越小，训练准确率越高。理论上，γ 趋向于无穷大时，可以拟合任何数据，但是对测试集的分类会比较差，具有低偏差，高方差性，这就会有过拟合的问题。而γ 较小时，具有高偏差，低方差性，可能会出现欠拟合问题。

cost参数

cost小，穿过间隔的成本小，导致支持向量多，容易出现过拟合，生成较为复杂的边界，具有低偏差，高方差性

cost大，穿过间隔的成本大，导致支持向量少，容易出现欠拟合，生成较为平滑的边界，具有高偏差，低方差性

4.4 核函数的选择

当特征数》样本数 使用线性核函数

特征数《样本数 使用径向核函数

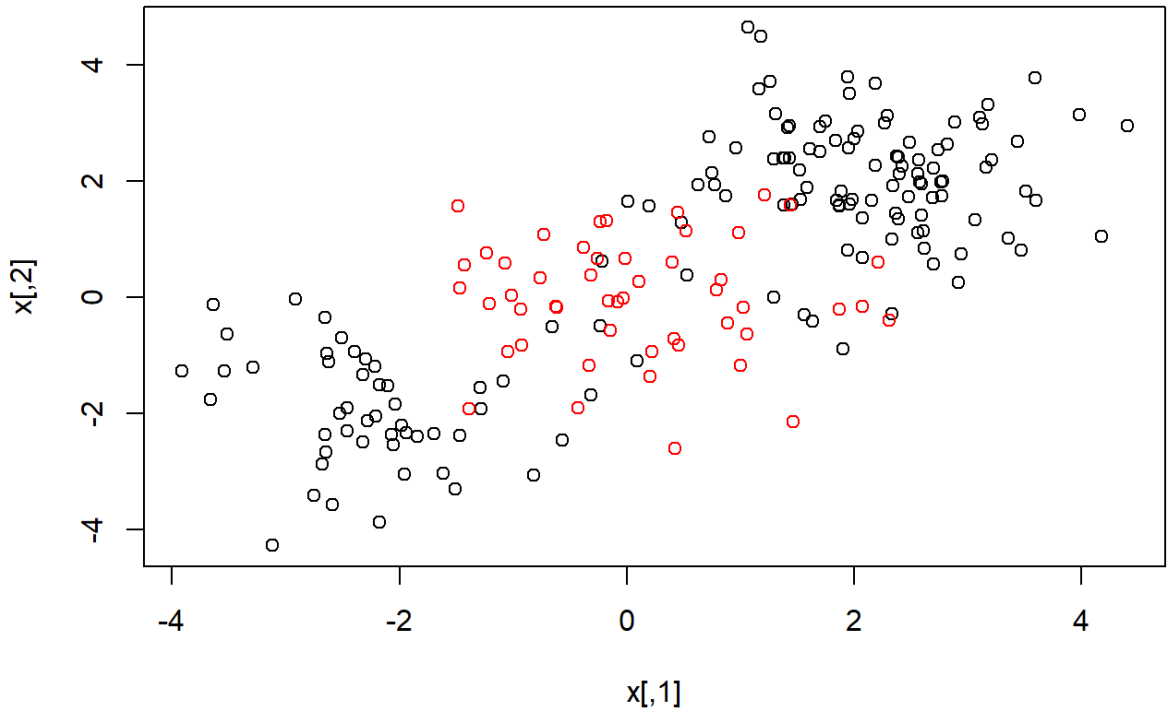
特征数和样本数一样都很大 使用线性核函数（因为线性核和径向核的效果差不多，但线性核在效率上更优）

---

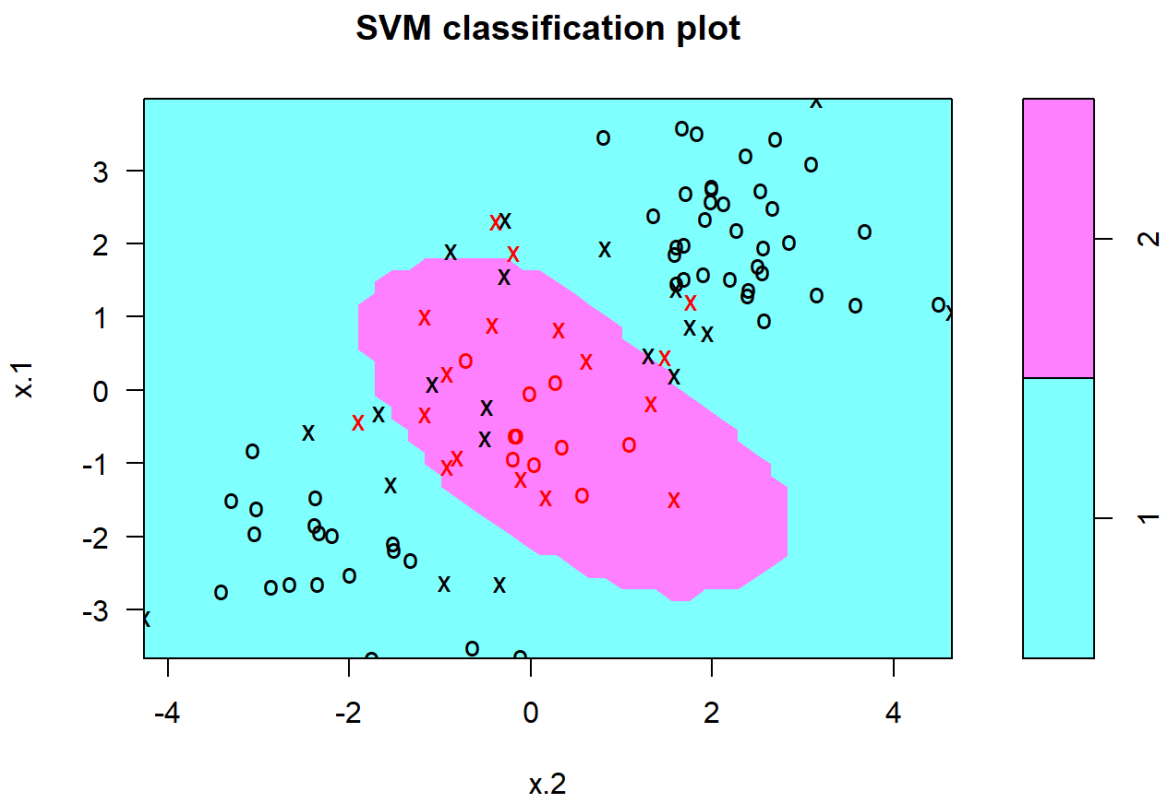
5 支持向量机(SVM)

支持向量机是支持向量分类器的扩展，在支持向量分类器中引入了**核函数。**

```
#建立一个线性不可分的数据集
set.seed(1)
x = matrix(rnorm(200 * 2), ncol = 2)
x[1:100,] = x[1:100,] + 2
x[101:150,] = x[101:150,] - 2
y = c(rep(1, 150), rep(2, 50))
dat = data.frame(x = x, y = as.factor(y))
plot(x, col = y)
```



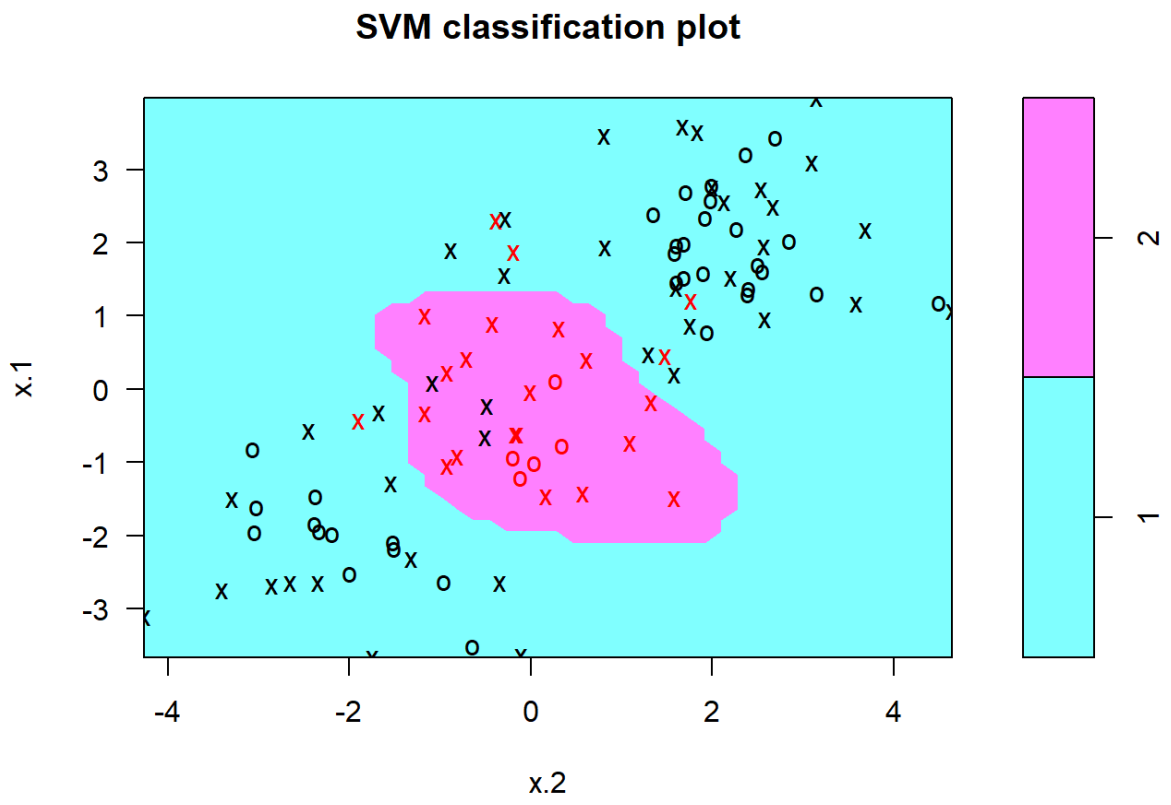

```
train = sample(200, 100)
svmfit5.1 = svm(y ~ ., data = dat[train,], kernel = "radial", gamma = 1, cost = 1)#这边用了径向核函数，因此除了参数cost，也要规定gamma参数
plot(svmfit5.1, dat[train,])
```




```
summary(svmfit5.1)
```

```
##
## Call:
## svm(formula = y ~ ., data = dat[train, ], kernel = "radial",
##     gamma = 1, cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  1
##
## Number of Support Vectors:  37
##
##  ( 17 20 )
##
##
## Number of Classes:  2
##
## Levels:
##  1 2
```

```
#改变gamma的值
svmfit5.2 = svm(y ~ ., data = dat[train,], kernel = "radial", gamma = 5, cost = 1)
plot(svmfit5.2, dat[train,])
```




```
summary(svmfit5.2)
```

```
##
## Call:
## svm(formula = y ~ ., data = dat[train, ], kernel = "radial",
##     gamma = 5, cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  5
##
## Number of Support Vectors:  61
##
##  ( 22 39 )
##
##
## Number of Classes:  2
##
## Levels:
##  1 2
```

```
set.seed(1)
#下面这个函数是用来通过交叉验证来选择径向核函数最优的gamma和cost#用tune.svm()也能得到一样的结果
tune.out = tune(svm, y ~ ., data = dat[train,], kernel = "radial", ranges = list(cost = c(0.1, 1, 10, 100, 1000), gamma = c(0.5, 1, 2, 3, 4)))
summary(tune.out)#这个语句给出了不同cost和gamma值对应的交叉验证误差，而且给出了最好的模型的cost=1，gamma=2
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1     2
##
## - best performance: 0.12
##
## - Detailed performance results:
##     cost gamma error dispersion
## 1  1e-01   0.5  0.27 0.11595018
## 2  1e+00   0.5  0.13 0.08232726
## 3  1e+01   0.5  0.15 0.07071068
## 4  1e+02   0.5  0.17 0.08232726
## 5  1e+03   0.5  0.21 0.09944289
## 6  1e-01   1.0  0.25 0.13540064
## 7  1e+00   1.0  0.13 0.08232726
## 8  1e+01   1.0  0.16 0.06992059
## 9  1e+02   1.0  0.20 0.09428090
## 10 1e+03   1.0  0.20 0.08164966
## 11 1e-01   2.0  0.25 0.12692955
## 12 1e+00   2.0  0.12 0.09189366
## 13 1e+01   2.0  0.17 0.09486833
## 14 1e+02   2.0  0.19 0.09944289
## 15 1e+03   2.0  0.20 0.09428090
## 16 1e-01   3.0  0.27 0.11595018
## 17 1e+00   3.0  0.13 0.09486833
## 18 1e+01   3.0  0.18 0.10327956
## 19 1e+02   3.0  0.21 0.08755950
## 20 1e+03   3.0  0.22 0.10327956
## 21 1e-01   4.0  0.27 0.11595018
## 22 1e+00   4.0  0.15 0.10801234
## 23 1e+01   4.0  0.18 0.11352924
## 24 1e+02   4.0  0.21 0.08755950
## 25 1e+03   4.0  0.24 0.10749677
```

```
tabsvm<-table(true = dat[-train, "y"], pred = predict(tune.out$best.model, newdata = dat[-train,]))#tune.out$best.model就是使交叉验证误差最小的cost和gamma所建立的分类器
tabsvm
```

```
##     pred
## true  1  2
##    1 74  3
##    2  7 16
```

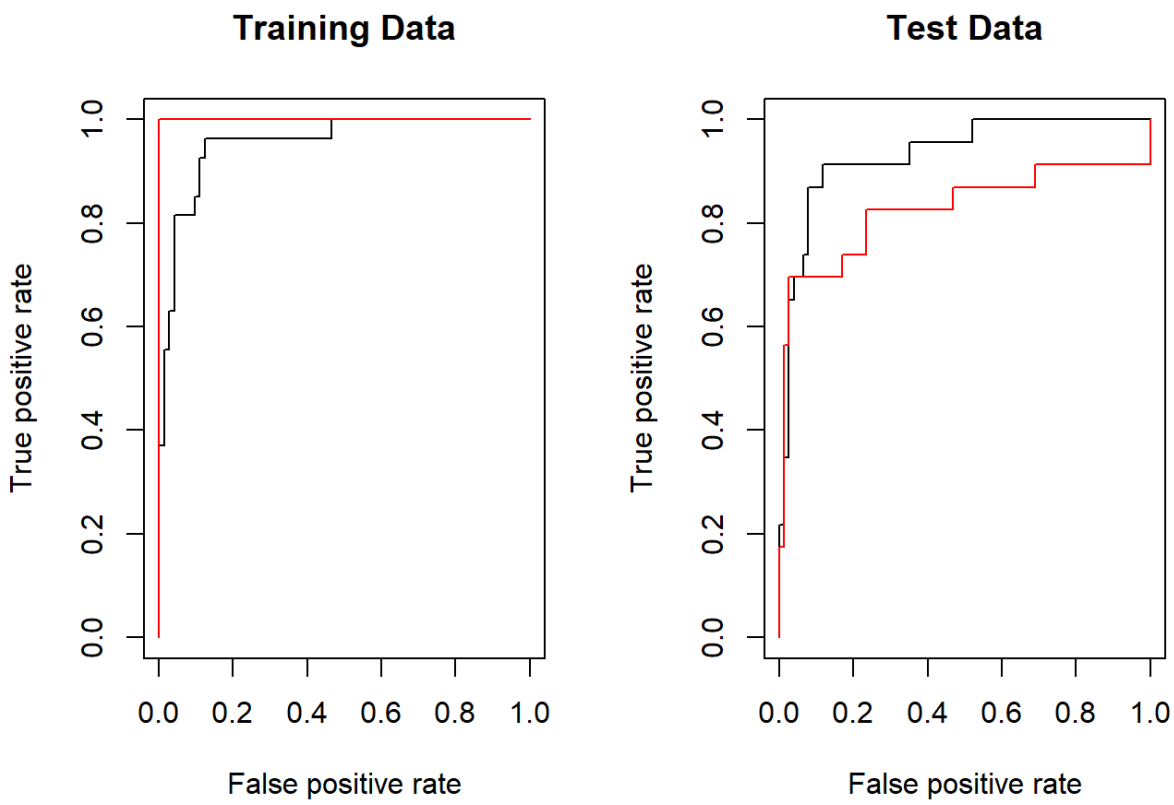
```
sum(diag(prop.table(tabsvm)))#计算正确分类率
```

```
## [1] 0.9
```

6 ROC曲线

```
#比较不同gamma取值的ROC曲线
library(ROCR)
rocplot = function(pred, truth, ...) {
  predob = prediction(pred, truth)
  perf = performance(predob, "tpr", "fpr")
  plot(perf, ...)
}
#训练集数据
svmfit.gamma2 = svm(y ~ ., data = dat[train,], kernel = "radial", gamma = 2, cost =
1, decision.values = T)
fittedgamma2 = attributes(predict(svmfit.gamma2, dat[train,], decision.values = TRU
E))$decision.values
par(mfrow = c(1, 2))
rocplot(fittedgamma2, dat[train, "y"], main = "Training Data")
svmfit.gamma50 = svm(y ~ ., data = dat[train,], kernel = "radial", gamma = 50, cost
= 1, decision.values = T)
fittedgamma50 = attributes(predict(svmfit.gamma50, dat[train,], decision.values = T))
$decision.values
rocplot(fittedgamma50, dat[train, "y"], add = T, col = "red")

#测试集数据
fittedgamma2 = attributes(predict(svmfit.gamma2, dat[-train,], decision.values = T))
$decision.values
rocplot(fittedgamma2, dat[-train, "y"], main = "Test Data")
fittedgamma50 = attributes(predict(svmfit.gamma50, dat[-train,], decision.values =
T))$decision.values
rocplot(fittedgamma50, dat[-train, "y"], add = T, col = "red")
```



结果解读：训练集数据的ROC图显示增加gamma可以更加光滑地拟合数据，但是在测试集数据中，gamma2比gamma50模型能提供更准确的预测结果

---

7 多分类的SVM

7.1 一类对一类

构建$\binom{K}{2}$个SVM模型，每个SVM模型可以分隔两个类

具体:使用$\binom{K}{2}$个SVM模型对一个测试观测进行分类，记录这个测试观测被分到每个类别的次数，分到次数最多的这个类，就是这个测试观测的最终预测类别

a,b,c三类

可以做出三个SVM模型：a|b,a|c,b|c

测试观测分到：a,a,b

那么这个测试观测最终被分到类“a”

7.2 一类对其余

第m个类与其余K-1个类构建SVM模型，这样的话一共有K个SVM模型，把x^*分到使$f(x^*) = \beta_{0m} + \beta_{1m}x_1^* + \beta_{2m}x_2^* + \ldots + \beta_{pm}x_p^*$最大的那个类“m”。这意味着把这个测试观测预测的类“m”，而不是其他的类别是具有高度的信心的

---

8 总结

8.1 SVM与Logistic回归比较

不同类别的 可以很好地被分离时，SVM表现更好；不同类别存在较多重叠时，Logistic回归表现更好

8.2 损失函数+惩罚项

支持向量分类器里面的优化问题也可以写成以下形式： 但是这边不再要求$\sum \beta^2 = 1$，而是要求间隔对应的值是1（这种模式出现在大多数参考书籍里面）

$\min_{\beta_0,\beta_1,...,\beta_p} \left\{ \sum_{i=1}^{n} \max[0, 1 - y_i f(x_i)] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$

这里的λ 相当于是前面的容忍度C ,$\lambda \sum_{j=1}^{p} \beta_j^2$ 是惩罚项，用以控制方差-偏差权衡，$\sum_{i=1}^{n} \max[0, 1 - y_i f(x_i)]$ 是损失函数（这种损失函数是铰链损失函数），是模型对数据拟合程度的某种量化。如果被正确分类，那么$y_i f(x_i) \geq 1$，这就意味着$\max[0, 1 - y_i f(x_i)] = 0$，损失为0，即在间隔之外的正确分类的观测不会影响分类器，而那些在间隔错误一侧的观测（也就是那些支持向量），使$y_i f(x_i) < 1$，则$\max[0, 1 - y_i f(x_i)] = 1 - y_i f(x_i) > 0$，由于最终的优化目标是min，这与其相悖，因此会影响到分类器的构建

这一part先讲到这里，后面还会深入