多元统计第四次作业

蒋文馨

2020 年 6 月 13 日

目录

1 Perpendicular Distance {1}

(a) Show that the perpendicular distance of the point (h, k) to the line $f(x, y) = ax+by+c = 0$ is $|ah + bk + c|/\sqrt{a^2 + b^2}$.

证明： 记 d 为点 (h, k) 到直线 f 的距离. 显然 a, b 不能同时为 0, 且当 $a = 0, b \neq 0$ 和 $a \neq 0, b = 0$ 或者 (h, k) 在直线 f 上时, 都有 $d = |ah + bk + c|/\sqrt{a^2 + b^2}$.

当 $a \neq 0, b \neq 0$ 且 (h, k) 不在直线 f 上时, 记 (p, q) 为 f 过点 (h, k) 的垂线的垂足, 故 $d = \|(h, k) - (p, q)\|$. 记 f 的斜率为 $k = -\frac{a}{b}$, 则有

$$k' = \frac{k-q}{h-p} = -\frac{1}{k} = \frac{b}{a} \tag{1}$$

$$ap + bq + c = 0 \tag{2}$$

$$d^2 = (h-p)^2 + (k-q)^2 \tag{3}$$

将 $h - p$ 和 $k - q$ 视为一个整体可以方便计算, 因此将2重写为

$$a(h-p) + b(k-q) = ah + bk + c \tag{4}$$

由1可得

$$k - q = \frac{b}{a}(h-p) \tag{5}$$

把5代入4和3, 得

$$h - p = \frac{ah + bk + c}{a + b^2/a} \tag{6}$$

$$d^2 = (1 + \frac{b^2}{a^2})(h-p) \tag{7}$$

把6代入7

$$d = |ah + bk + c|/\sqrt{a^2 + b^2}$$

综上, 当 a, b 不同时为零时, 点 (h, k) 到直线 f 的距离 d 为

$$d = |ah + bk + c|/\sqrt{a^2 + b^2}$$

□

这里也可以使用拉格朗日乘子法, 但下一题用了, 所以还是选择初等方法.

(b) Let $\mu(\mathbf{x}) = \beta_0 + \mathbf{x}^\tau \boldsymbol{\beta} = 0$ denote a hyperplane, where $\beta_0 \in \Re$ and $\boldsymbol{\beta} \in \Re^r$, and let $\mathbf{x}_k \in \Re^r$ be a point in the space. By minimizing $\|\mathbf{x} - \mathbf{x}_k\|^2$ subject to $\mu(\mathbf{x}) = 0$, show that the perpendicular distance from the point to the hyperplane is $|\mu(\mathbf{x}_k)| / \|\boldsymbol{\beta}\|$.

证明：将题目重写为标准形式:

$$\begin{aligned} \min_{\mathbf{x} \in \Re^r} f(\mathbf{x}) &= \|\mathbf{x} - \mathbf{x}_k\|^2 \\ s.t. \qquad \mu(\mathbf{x}) &= 0 \end{aligned} \tag{8}$$

构造拉格朗日函数

$$L(\mathbf{x}, \lambda) = (\mathbf{x} - \mathbf{x_k})'(\mathbf{x} - \mathbf{x_k}) + \lambda(\beta_0 + \mathbf{x}'\boldsymbol{\beta}) \tag{9}$$

令

$$\frac{\partial L}{\partial \mathbf{x}} = 2(\mathbf{x} - \mathbf{x}_k) + \lambda\boldsymbol{\beta} =: 0 \tag{10}$$

$$\frac{\partial L}{\partial \lambda} = \beta_0 + \mathbf{x}'\boldsymbol{\beta} =: 0 \tag{11}$$

由10可得

$$\mathbf{x} = \mathbf{x}_k - \frac{\lambda\boldsymbol{\beta}}{2} \tag{12}$$

把12代入11, 得

$$\lambda = \frac{2(\beta_0 + \mathbf{x}_k'\boldsymbol{\beta})}{\boldsymbol{\beta}'\boldsymbol{\beta}} \tag{13}$$

把12和13代回8, 得

$$\min_{\mathbf{x} \in \Re^r} f(\mathbf{x}) = \frac{\lambda^2 \boldsymbol{\beta}'\boldsymbol{\beta}}{4} = \frac{(\beta_0 + \mathbf{x}_k'\boldsymbol{\beta})^2}{\boldsymbol{\beta}'\boldsymbol{\beta}} \tag{14}$$

所以

$$d = \sqrt{f_{\min}} = \frac{|\mu(\mathbf{x}_k)|}{\|\boldsymbol{\beta}\|}$$

□

2 Concave Function {11}

Show that the functional $F_D(\boldsymbol{\alpha})$ in (11.40) is concave; i.e., show that, for $\theta \in (0,1)$ and $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \Re^n$

$$F_D(\theta\boldsymbol{\alpha} + (1-\theta)\boldsymbol{\beta}) \geq \theta F_D(\boldsymbol{\alpha}) + (1-\theta)F_D(\boldsymbol{\beta}).$$

证明：将 F_D 重写为矩阵形式:

$$F_D(\boldsymbol{\alpha}) = \mathbf{1}'\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}'\mathbf{H}\boldsymbol{\alpha} \tag{15}$$

其中 $\mathbf{H} = [y_i y_j \mathbf{x}_i' \mathbf{x}_j]_{i\times j} \simeq \mathbf{I}$, $y_i = \pm 1, i = 1, 2, \cdots, n$, $\mathbf{H}$ 是对称半正定矩阵 (若假设 $\mathbf{x}_i$ 线性无关 $i = 1, 2, \cdots, n$, 则为对称正定矩阵). 因此

$$\begin{aligned}
& F_D(\theta\boldsymbol{\alpha} + (1-\theta)\boldsymbol{\beta}) - \theta F_D(\boldsymbol{\alpha}) - (1-\theta)F_D(\boldsymbol{\beta}) \\
=& -\frac{1}{2}[(\theta\boldsymbol{\alpha} + (1-\theta)\boldsymbol{\beta})'\mathbf{H}(\theta\boldsymbol{\alpha} + (1-\theta)\boldsymbol{\beta}) - \theta\boldsymbol{\alpha}'\mathbf{H}\boldsymbol{\alpha} - (1-\theta)\boldsymbol{\beta}'\mathbf{H}\boldsymbol{\beta}] \\
=& \frac{(1-\theta)\theta}{2}(\boldsymbol{\alpha}'\mathbf{H}\boldsymbol{\alpha} + \boldsymbol{\beta}'\mathbf{H}\boldsymbol{\beta} - \boldsymbol{\alpha}'\mathbf{H}\boldsymbol{\beta} - \boldsymbol{\beta}'\mathbf{H}\boldsymbol{\alpha}) \\
=& \frac{(1-\theta)\theta}{2}(\boldsymbol{\alpha} - \boldsymbol{\beta})'\mathbf{H}(\boldsymbol{\alpha} - \boldsymbol{\beta})
\end{aligned} \tag{16}$$

因为 $\theta \in (0,1)$, $\mathbf{H}$ 对称半正定, 所以 $F_D(\theta\boldsymbol{\alpha} + (1-\theta)\boldsymbol{\beta}) - \theta F_D(\boldsymbol{\alpha}) - (1-\theta)F_D(\boldsymbol{\beta}) \geq 0$. 即,

$$F_D(\theta\boldsymbol{\alpha} + (1-\theta)\boldsymbol{\beta}) \geq \theta F_D(\boldsymbol{\alpha}) + (1-\theta)F_D(\boldsymbol{\beta}).$$

□

另一种方法:

由 $\mathbf{H}$ 对称知

$$\nabla F_D(\boldsymbol{\alpha}) = \mathbf{1} - \mathbf{H}\boldsymbol{\alpha}$$

$$\nabla^2 F_D(\boldsymbol{\alpha}) = -\mathbf{H}$$

由 $\mathbf{H}$ 半正定, $-\mathbf{H}$ 半负定, 知 F_D 为凹函数.

□

3 Nonlinear-SVM to WDBC Data Set {12}

Apply nonlinear-SVM to a binary classification wdbc data set. Make up a two-way table of values of (C, γ) and for each cell in that table compute the CV/10 misclassification rate. Find the pair (C, γ) with the smallest CV/10 misclassification rate. Compare this rate with results obtained using LDA and that using a classification tree.

解: 这里使用 e1071 包的 tune 函数选择最优的 cost 和 gamma 参数. 为了使比较更公平, **所有方法均计算 10 折 CV 下的错误率**. wdbc 数据第 3 到 32 列进行标准化.

```r
#数据读入及预处理
library(readr)
wdbc <- read_csv("wdbc.txt", col_names = FALSE)
#rename colomn of data
data.colname = "diag"
for (i in 1:30) {
  data.colname = c(data.colname, paste('X', i, sep = ""))
}
data = wdbc[, -1]#delete id of patients, diag=X2
names(data) = data.colname
data$diag = as.factor(data$diag)
data[, 2:31] = apply(data[, 2:31], 2, scale)
```

为了减小计算量, 先取大步长估计最优参数范围, 再减小步长搜索. 不过即使是这样, 每次运行也需要很长的时间.

```r
library(e1071)
library(MASS)
library(rpart)
library(randomForest)
set.seed(0)
#choose range of gamma and cost
gamma = seq(0, 1, .1)
cost = 10 ^ (-3:4)
tuned = tune(svm,
             diag ~ .,
             data = data,
             range = list(gamma = gamma, cost = cost))
tuned$best.parameters
```

```
##    gamma cost
## 35   0.1    1
```

```r
plot(tuned)
```

```r
#choose range of gamma and cost
gamma = seq(0, .2, .02)
cost = seq(.01, 2, .01)
```

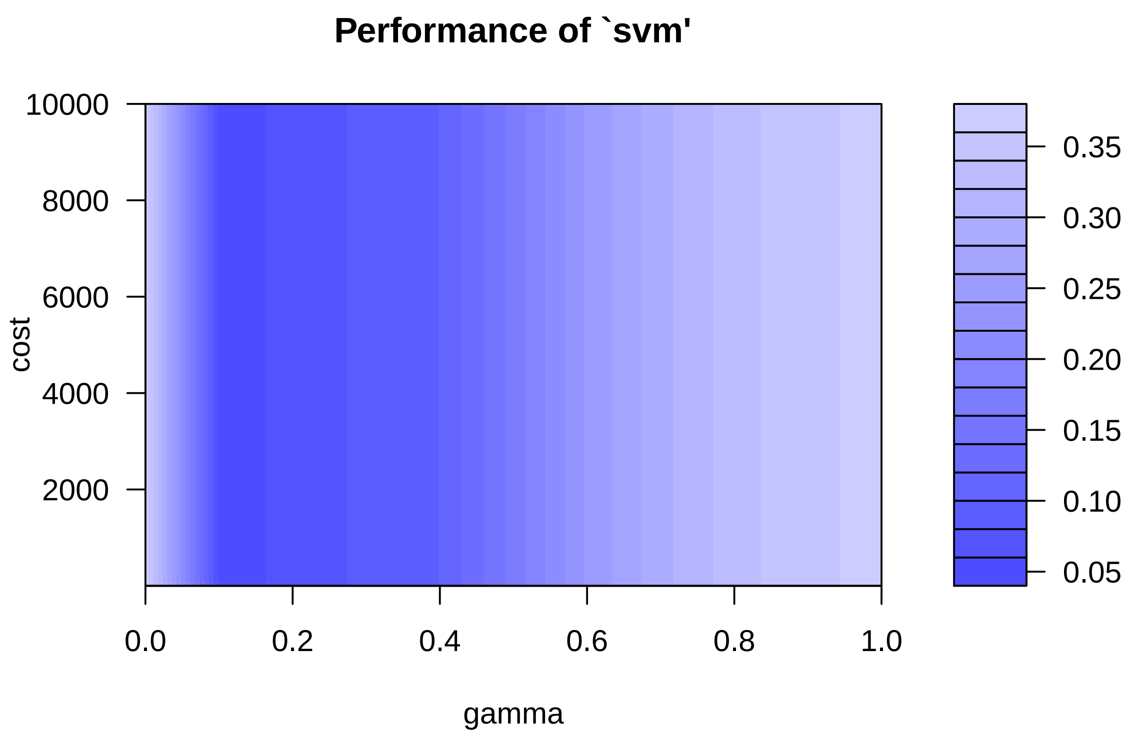

图 1: 不同 gamma 和 cost 下 SVM 的错误率: 颜色越深表示错误率越低. 从图中可以看出错误率的变化趋势, 缩小区间再进行进一步搜索, 减小计算量.

```
tuned2 = tune(svm,
              diag ~ .,
              data = data,
              range = list(gamma = gamma, cost = cost))
```

```
## [1] "最佳的参数取值约为 gamma: 0.04; cost: 1.92"
```

10 折 CV 下不同 (C, γ) 的错误率见表1. (为了使变化更明显, 这里用了第一次 tune 的数据)

表 1: (γ,cost) 表

		cost							
		e-3	e-2	0.1	1	10	e+2	e+3	e+4
	0	0.373	0.373	0.373	0.373	0.373	0.373	0.373	0.373
	0.1	0.373	0.373	0.068	$\boxed{0.044}$	0.053	0.054	0.054	0.054
	0.2	0.373	0.373	0.285	0.065	0.063	0.063	0.063	0.063
	0.3	0.373	0.373	0.373	0.081	0.086	0.086	0.086	0.086
	0.4	0.373	0.373	0.373	0.100	0.100	0.100	0.100	0.100
γ	0.5	0.373	0.373	0.373	0.178	0.167	0.167	0.167	0.167
	0.6	0.373	0.373	0.373	0.262	0.243	0.243	0.243	0.243
	0.7	0.373	0.373	0.373	0.318	0.294	0.294	0.294	0.294
	0.8	0.373	0.373	0.373	0.357	0.331	0.331	0.331	0.331
	0.9	0.373	0.373	0.373	0.366	0.357	0.357	0.357	0.357
	1.0	0.373	0.373	0.373	0.369	0.364	0.364	0.364	0.364

使用 10-CV 计算 LDA, CART, Random Forest 及调参后 SVM 的错误率. (为了使比较更公平, 这里重新计算了 SVM 的错误率.) 需要注意的是: CART 进行了剪枝.

```
#randomly shuffle the data
shuffledata = data[sample(nrow(data)), ]
#create 10 equally size folds
folds = cut(seq(1, nrow(shuffledata)), breaks = 10, labels = FALSE)
#record performance of methods
cart.rcd = lda.rcd = qda.rcd = vector(length = 10)
forest.rcd = vector(length = 10)
svm.rcd = vector(length = 10)
#perform 10 fold cross validation
#parameter setting
```

```
c = 1.92; g = 0.04
for (i in 1:10) {
  testIndexes = which(folds == i, arr.ind = TRUE)
  testData = shuffledata[testIndexes,]
  trainData = shuffledata[-testIndexes,]
  #SVM
  svm.out = svm(diag ~ ., trainData, cost = c, gamma = g)
  svm.rcd[i] = sum(list(predict(svm.out,
                              newdata = testData[, -1])) !=
                   testData[, 1]) / dim(testData)[1]

  #LDA
  lda.out = lda(diag ~ ., trainData)
  lda.rcd[i] = sum(list(predict(lda.out, newdata = testData[, -1])$class) !=
                   testData[, 1]) / dim(testData)[1]

  #qDA
  qda.out = qda(diag ~ ., trainData)
  qda.rcd[i] = sum(list(predict(qda.out, newdata = testData[, -1])$class) !=
                   testData[, 1]) / dim(testData)[1]

  #CART
  dtree = rpart(diag ~ ., data = trainData)
  #prune
  which.min.xerror = which.min(dtree$cptable[, "xerror"])
  cutoff = dtree$cptable[which.min.xerror, "xerror"] +
    dtree$cptable[which.min.xerror, "xstd"]
  cart.out = prune(dtree,
             cp = dtree$cptable
             [min(which(dtree$cptable[, "xerror"] <cutoff)), "CP"]
             )
  cart.prdt = predict(cart.out, newdata = testData[, -1])
  cart.prdt.class = vector(length = dim(testData)[1])
  cart.prdt.class[cart.prdt[, 1] > 0.5] = "B"
  cart.prdt.class[cart.prdt[, 1] <= 0.5] = "M"
  cart.rcd[i] = sum(cart.prdt.class != testData[, 1]) / dim(testData)[1]

  #RandomForest
```

```
  forest.out = randomForest(diag ~ ., data = trainData)
  forest.rcd[i] = sum(list(predict(forest.out,
                                   newdata = testData[, -1])) !=
                          testData[, 1]) / dim(testData)[1]
}
```

```
## [1] "SVM's misclassification rate is 0.0157894736842105"

## [1] "LDA's misclassification rate is 0.043984962406015"

## [1] "QDA's misclassification rate is 0.0457393483709273"

## [1] "CART's misclassification rate is 0.0756892230576441"

## [1] "Random Forest's misclassification rate is 0.0405075187969925"
```

可以看出四种方法错误率都不高. 效果最好的是 SVM(虽然是否能低于 0.02 取决于种子, 但 SVM 总是最好的). RF 和 CART 相比有了明显的提升; 然而 QDA 和 LDA 差不多. 我还试着对 wdbc 取 log 以改进 LDA 的错误率, 但是改进过于轻微.