The honor code

(a) Please state the names of people who you worked with for this homework. You can also provide your comments about the homework here.

Cinidy Liu

(b) Please type/write the following sentences yourself and sign at the end. We want to make it *extra* clear that nobody cheats even unintentionally.

I hereby state that all of my solutions were entirely in my words and were written by me. I have not looked at another student's solutions and I have fairly credited all external sources in this write up.

I hereby state that all of my solutions were entirely in my words and were written by me. I have not looked at another student's solutions and I have fairly credited all external sources in this write up.

HW6

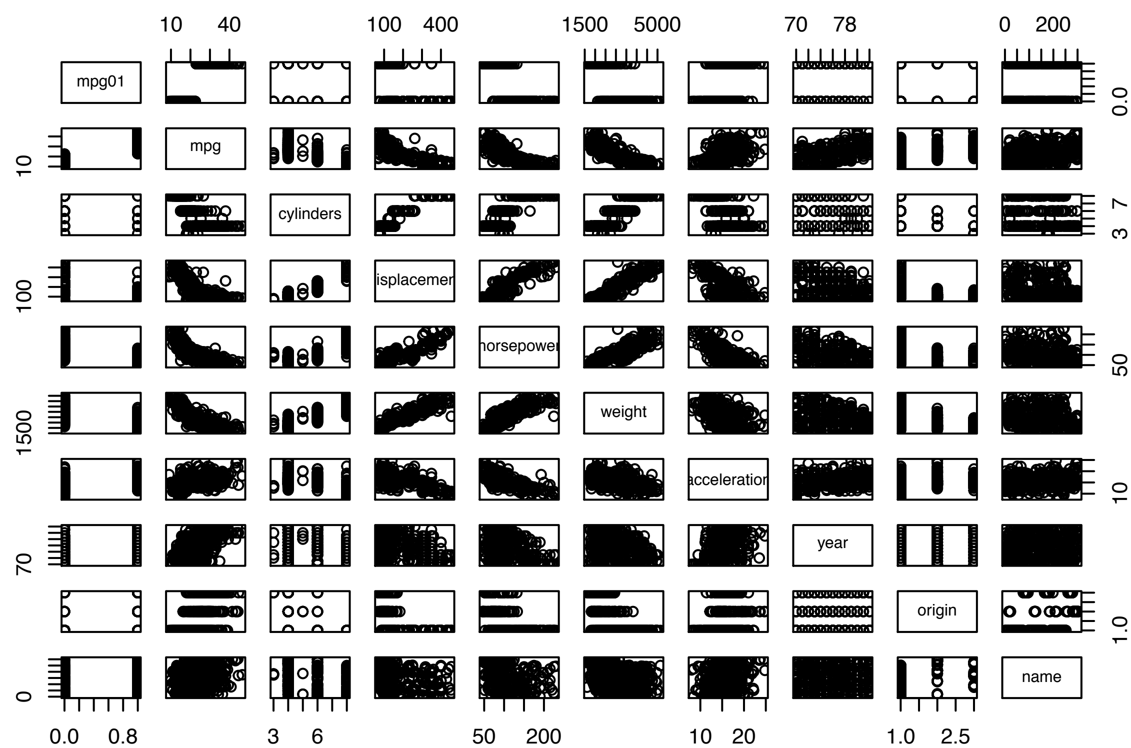
caojilin

4/15/2019

4.11

```r
#a
mpg = Auto$mpg
mpg01 = rep(0, length(mpg))
for (i in 1:length(mpg)) {
  mpg01[i] = if(mpg[i] > median(mpg)) 1 else 0
}
dat = data.frame(mpg01 = mpg01, Auto)
```

```r
#b
pairs(dat)
```



except for `mpg`, we see that `cylinders`, `horsepower` , `weight` ,`year` and `acceleration` are most likely to be useful in predicting `mpg01`

```r
#c
# split ratio 8:2
set.seed(16)
train_ind <- sample(seq_len(nrow(dat)), size = nrow(dat)*0.8)
train <- dat[train_ind, ]
test <- dat[-train_ind, ]
```

```r
#d LDA
fit = lda(mpg01 ~ cylinders + year + horsepower + weight + acceleration, data=train)
preds = predict(fit, test)
mean(preds$class != test$mpg01)
```

```
## [1] 0.06329114
```

```r
#e QDA
fit = qda(mpg01 ~ cylinders +year + horsepower + weight + acceleration, data=train)
preds = predict(fit, test)
mean(preds$class != test$mpg01)
```

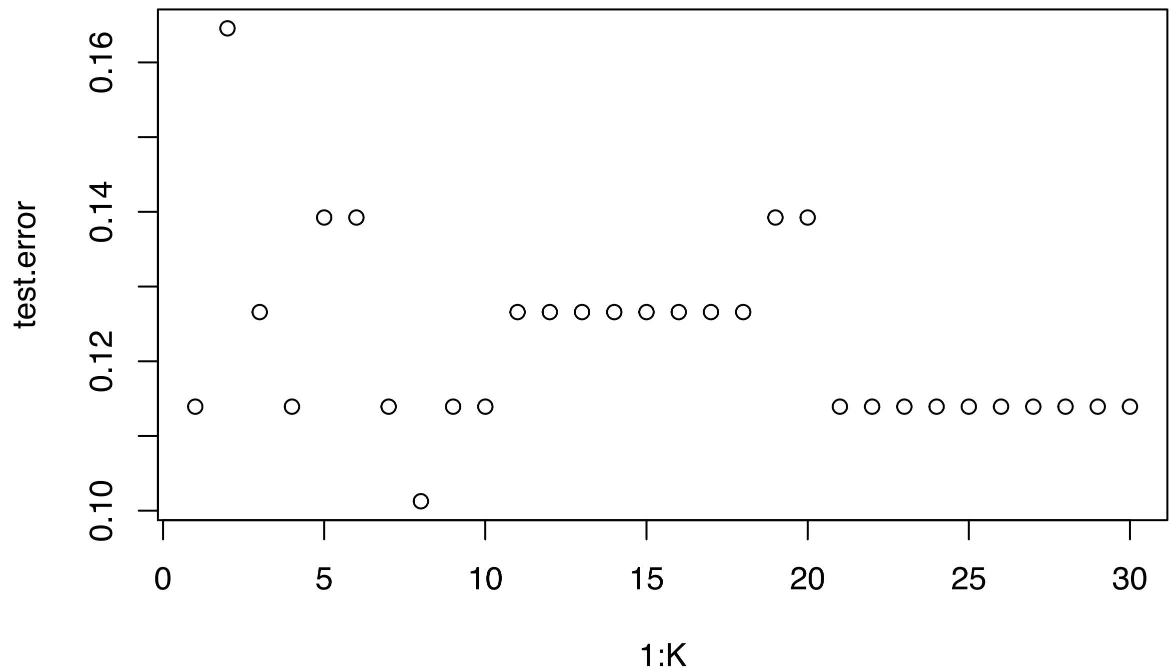
```
## [1] 0.1012658
```

```r
#f logistic regression
fit =  glm(mpg01 ~ cylinders + year + horsepower + weight + acceleration,
           family = "binomial", data = train)
preds = predict(fit, test, type = "response")
preds = preds > 0.5
mean(preds != test$mpg01)
```

```
## [1] 0.07594937
```

```r
#g KNN
set.seed(16)
train.knn <- dat[train_ind, -c(1,2,4,9,10)]
test.knn <- dat[-train_ind, -c(1,2,4,9,10)]

K = 30
test.error = rep(0, K)
for (i in 1:K) {
  preds = knn(train.knn, test.knn, cl=dat[train_ind, "mpg01"], k = i)
  test.error[i] = mean(preds != test$mpg01)
}
plot(test.error, x=1:K)
```




```r
which(test.error == min(test.error))
```

```
## [1] 8
```

```r
test.error[8]
```

```
## [1] 0.1012658
```

we see K=8 give us the smallest test error

1.

(a) We need d units additions so $O(d)$ for $a+v$

We need d times multiplication, so $O(d)$ for $a^T v$

(b) A has $n \cdot d$ elements $A+B$ needs $O(n \cdot d)$ addition

also needs $O(n \cdot d)$ space to store

(c) $A \quad n\overset{d}{\begin{pmatrix} \text{—} \\ \vdots \\ \text{—} \end{pmatrix}} \quad d\underset{v}{\begin{pmatrix} | \end{pmatrix}}$ one multiplication needs d, there are n

so Av is $O(nd)$

$A^T B$ is $O(nd^2)$

(d) $A^T B v$ method 1: $(A^T B)$ takes $O(nd^2)$ $\quad (A^T B) \cdot v$ takes $O(d^2)$

$O(nd^2)$ dominant

method 2: $B \cdot v$ takes $O(nd)$. $\quad A^T (B \cdot v)$ takes $O(d \cdot n)$

So $O(nd)$ after drops constant

2.

the dominant terms

1. $X^TX \rightarrow X^TX+\lambda I \longrightarrow (X^TX+\lambda I)^{-1}$

$O(nl^2) + O(l^2) + O(l^3) = O(l^3+nl^2)$ (1)

2. $XX^T \longrightarrow XX^T+\lambda I \longrightarrow (XX^T+\lambda I)^{-1}$

$O(n^2 l) + O(n^2) + O(n^3) = O(n^3+n^2 l)$ (2)

3. if $n >> l$ (2)'s complexity becomes $O(n^3)$, (1) is preferable

if $n << l$ (1)'s complexity becomes $O(l^3)$, (2) is prefereable

4\. 1. if $l \gg n$, like the gene data set, then we can use the kernel trick to reduce the computation cost

2\. Sometimes it's easier to deal with the data in transformed feature space, such as predictors become linear in feature space, while not in the original space.

5. (a) $K(x,z) = (x_1x_2, \frac{x_1^2}{\sqrt{2}}, \frac{x_2^2}{\sqrt{2}})^T (z_1z_2, \frac{z_1^2}{\sqrt{2}}, \frac{z_2^2}{\sqrt{2}})$

$= (\frac{1}{\sqrt{2}} x_1 z_1 + \frac{1}{\sqrt{2}} x_2 z_2)^2$

$= (\frac{1}{\sqrt{2}} x^T z)^2$

(b) $K(x,z) = (1, x, \frac{x^2}{\sqrt{2!}}, \dots)^T (1, z, \frac{z^2}{\sqrt{2!}}, \dots)$

$= 1^2 + xz + \frac{(xz)^2}{2!} + \frac{(zx)^3}{3!} + \dots$

$= e^{xz}$

(c) $K(x,z) = (1 + x^T z)^2 + x^T z$

$= 1 + 3x_1z_1 + 3x_2z_2 + 2x_1z_1x_2z_2 + x_1^2 z_1^2 + x_2^2 z_2^2$

$= (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{3}x_1, \sqrt{3}x_2, 1)^T (z_1^2, z_2^2, \sqrt{2}z_1z_2, \sqrt{3}z_1, \sqrt{3}z_2, 1)$

$= \langle \phi(x), \phi(z) \rangle$

(d) $e^{-(x-z)^2}$ can not be written as $\phi(x)\phi(z)$

as $e^{-(x-z)^2} = 1 + (-(x-z)^2) + \dots$ where $-(x-z)^2$ cannot be splitted as a product of $f(x)$ and $g(z)$

6.

$$\hat{\Theta} = (\Phi^T\Phi + \lambda I_d)^{-1}\Phi^{-1}y$$

or

$$\hat{\Theta} = \Phi^T(\Phi\Phi^T + \lambda I_n)^{-1}y$$

7.
(1) suppose a new point is X_{new}, we'd like to predict $\hat{y}_{new}$

with kernelized $\hat{y}_{new} = \langle \phi(X_{new}), \hat{\Theta} \rangle = \phi(X_{new})^T \Phi^T(\Phi\Phi^T + \lambda I_n)^{-1}y$

$$= (K(X_1, X_{new}), \cdots, K(X_n, X_{new}))(K + \lambda I)^{-1}y$$

$$K = \begin{pmatrix} K(X_1,X_1) & K(X_1,X_2) & \cdots \\ K(X_2,X_1) & \ddots & \\ \vdots & & K(X_n,X_n) \end{pmatrix}$$

$n \times n$

Computing K takes $n \times n$ times the cost of $K(X,Z)$

$K(X,Z) = (1+X^TZ)^P$ where X^TZ takes $O(l)$

assume $P = 2^m$ ($P = 2^{m+1}$ doesn't affect the cost much)

Then $(1+X^TZ)^P = (((1+X^TZ)^2)^2\cdots)^2 = (1+X^TZ)^{\overbrace{2\times2\times2\cdots\times2}^{m}}$

Squaring a number can be seen as $O(1)$

We need $m = \log P$ times squaring

So the overall cost for computing $K(X,Z)$ is $O(l + \log P)$

inverting a $n \times n$ matrix is $O(n^3)$

The two dominant costs are $O(n^3 + n^2(l + \log p))$

(2)

$$\hat{y}_{new} = \langle \phi(X_{new}), \hat{\theta} \rangle = \phi(X_{new})^T (\Phi^T \Phi + \lambda I_d)^{-1} \Phi^T y$$ Non-kernelized

$\Phi^T \Phi$ is $d \times d$

$$\underset{\Phi^T}{d\begin{pmatrix} \overset{n}{\rule{2em}{0.4pt}} \\ \\ \end{pmatrix}} \; \underset{\Phi}{n\begin{pmatrix} \overset{d}{|} \\ \\ \end{pmatrix}}$$

the total computation for $\Phi^T \Phi$ is $O(d^2 n)$

while inverting $d \times d$ matrix takes $O(d^3)$

These two costs dominate, so the overall is $O(d^3 + d^2 n)$

Compare: if $n >> d$, we prefer the non-kernelized, which is faster than the kernelized

if $n << d$, we prefer the kernelized $(\Phi^T \Phi + \lambda I_d)^{-1} \Phi^T y$, which has the cost lower than non-kernelized

4.2

(a) $\hat{y} = \underset{K}{\text{argmax}}\ P(Y=k)\, P(X \mid Y=k)$ $\pi(k) = P(Y=k)$, $P(X \mid Y=k)$ is multivariate normal

$$\pi(2)\frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(X-u_2)^T\Sigma^{-1}(X-u_2)\right\} > \pi(1)\frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(X-u_1)^T\Sigma^{-1}(X-u_1)\right\}$$

$\downarrow$ take log on both sides

$$-\frac{1}{2}(X-u_2)^T\Sigma^{-1}(X-u_2) + \ln(\pi(2)) + \text{constants} > -\frac{1}{2}(X-u_1)^T\Sigma^{-1}(X-u_1) + \ln(\pi(1)) + \text{constants}$$

$$X^T\Sigma^{-1}u_2 - \frac{1}{2}u_2^T\Sigma^{-1}u_2 + \ln(\pi(2)) > X^T\Sigma^{-1}u_1 - \frac{1}{2}u_1^T\Sigma^{-1}u_1 + \ln(\pi(1))$$

$$X^T\Sigma^{-1}(u_2-u_1) > \frac{1}{2}u_1^T\Sigma^{-1}u_2 - \frac{1}{2}u_1^T\Sigma^{-1}u_1 + \underbrace{\ln\left(\frac{N_1}{N}\right) - \ln\left(\frac{N_2}{N}\right)}_{\rightarrow \ln\left(\frac{N_2}{N_1}\right)}$$

if left likelihood > right hand side likelihood, we classify to class 2

if left < right, we classify to class 1

(b) $(X^TX)\beta = X^TY$

$$\begin{pmatrix} 1^T & 1^T \\ X^{(2)T} & X^{(1)T} \end{pmatrix}\begin{pmatrix} 1 & X^{(2)} \\ 1 & X^{(1)} \end{pmatrix}\begin{pmatrix} \beta_0 \\ \beta \end{pmatrix} = \begin{pmatrix} 1^T & 1^T \\ X^{(2)T} & X^{(1)T} \end{pmatrix}\begin{pmatrix} N/N_2 1 \\ -N/N_1 1 \end{pmatrix}$$

$$\begin{pmatrix} N & N_1\hat{u}_1^T + N_2\hat{u}_2^T \\ N_1\hat{u}_1 + N_2\hat{u}_2 & X^{(2)T}X^{(2)} + X^{(1)T}X^{(1)} \end{pmatrix}\begin{pmatrix} \beta_0 \\ \beta \end{pmatrix} = \begin{pmatrix} 0 \\ N(\hat{u}_2-\hat{u}_1) \end{pmatrix}$$ $N\beta_0 + (N_1\hat{u}_1^T + N_2\hat{u}_2^T)\beta = 0$ ①

$$X^{(2)T}X^{(2)} + X^{(1)T}X^{(1)} = \bar{X}^{(2)T}\bar{X}^{(2)} + \bar{X}^{(1)T}\bar{X}^{(1)} + N_1\hat{u}_1\hat{u}_1^T + N_2\hat{u}_2\hat{u}_2^T = (N-2)\hat{\Sigma} + N_1\hat{u}_1\hat{u}_1^T + N_2\hat{u}_2\hat{u}_2$$

$$(N_1\hat{u}_1 + N_2\hat{u}_2)\beta_0 + \left[(N-2)\hat{\Sigma} + N_1\hat{u}_1\hat{u}_1^T + N_2\hat{u}_2\hat{u}_2^T\right]\beta = N(\hat{u}_2-\hat{u}_1)$$ plug in ①

$$\left[(N-2)\hat{\Sigma} + \frac{N_1N_2}{N}(\hat{u}_2-\hat{u}_1)(\hat{u}_2-\hat{u}_1)^T\right]\beta = N(\hat{u}_2-\hat{u}_1)$$

(c) from part (b) we have

$$(N-2)\hat{\Sigma}\beta + \frac{N_1 N_2}{N}(\hat{u}_2-\hat{u}_1)(\hat{u}_2-\hat{u}_1)^T\beta = N(\hat{u}_2-\hat{u}_1)$$

Since $(\hat{u}_2-\hat{u}_1)^T\beta$ is scalar

$\hat{\Sigma}\beta$ is a linear combination of $\hat{u}_2-\hat{u}_1$, and $\hat{\Sigma}_B\beta$ is a multiple of $\hat{u}_2-\hat{u}_1$

$\hat{\Sigma}\beta \propto \hat{u}_2-\hat{u}_1$ hence it's in the direction of $\hat{u}_2-\hat{u}_1$

$\beta \propto \hat{\Sigma}^{-1}(\hat{u}_2-\hat{u}_1)$

(d) Since in part (b) and (c), we assume labels are different,
so if two classes are distinct, $\beta \propto \hat{\Sigma}^{-1}(\hat{u}_2-\hat{u}_1)$ still holds.

(e) $\hat{\beta} = \lambda\hat{\Sigma}^{-1}(\hat{u}_2-\hat{u}_1)$ $\quad \hat{\beta}_0 = -\frac{\lambda}{N}(N_1\hat{u}_1^T + N_2\hat{u}_2^T)\hat{\Sigma}^{-1}(\hat{u}_2-\hat{u}_1)$

$$\hat{\beta}_0 + X^T\hat{\beta}_1 = X^T\lambda\hat{\Sigma}^{-1}(\hat{u}_2-\hat{u}_1) \geq \frac{\lambda}{N}(N_1\hat{u}_1^T + N_2\hat{u}_2^T)\hat{\Sigma}^{-1}(\hat{u}_2-\hat{u}_1) \quad (2)$$

We see when $N_1 = N_2$ (2) is equivalent to LDA in part (a). $\ln(\frac{N_2}{N_1}) = 0$
when $N_1 \neq N_2$, they are different.