Homework3

Yuning Li

2024-02-21

```r
library('splines')        ## for 'bs'
library('dplyr')          ## for 'select', 'filter', and others
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library('magrittr')       ## for '%<>%' operator
library('glmnet')         ## for 'glmnet'
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

```r
###  Linear regression examples ###

## load prostate data
prostate <-
  read.table(url(
    'https://web.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.data'))


## split prostate into testing and training subsets
prostate_train <- prostate %>%
  filter(train == TRUE) %>%
  select(-train)

prostate_test <- prostate %>%
  filter(train == FALSE) %>%
  select(-train)


## predict lpsa consider all other predictors
fit <- lm(lpsa ~ ., data=prostate_train)

## functions to compute testing/training error
L2_loss <- function(y, yhat)
  (y-yhat)^2
```

```r
error <- function(dat, fit, loss=L2_loss)
  mean(loss(dat$lcavol, predict(fit, newdata=dat)))
```

```r
## train_error
error(prostate_train, fit)
```

```
## [1] 1.647994
```

```r
## testing error
error(prostate_test, fit)
```

```
## [1] 1.474148
```

```r
## use glmnet to fit ridge regression
form  <- lpsa ~  lweight + age + lbph + lcp + pgg45 + lcavol + svi + gleason
x_inp <- model.matrix(form, data=prostate_train)
y_out <- prostate_train$lpsa
# Use glmnet to fit ridge regression
fit_ridge <- glmnet(x = x_inp, y = y_out, alpha = 0, lambda = seq(0.5, 0, -0.05))

# Display the coefficients for different lambda values
print(fit_ridge$beta)
```

```
## 9 x 11 sparse Matrix of class "dgCMatrix"
##   [[ suppressing 11 column names 's0', 's1', 's2' ... ]]
##
## (Intercept)  .            .            .             .            .
## lweight      0.515697286  0.525400575  0.5354170019  0.545775814  0.556311619
## age         -0.004889276 -0.005637483 -0.0064660393 -0.007390172 -0.008436051
## lbph         0.111264821  0.113946024  0.1167762533  0.119749176  0.122883708
## lcp          0.015168605  0.007943168 -0.0003765233 -0.010208916 -0.022217776
## pgg45        0.004390331  0.004515174  0.0046637167  0.004840594  0.005062742
## lcavol       0.329947512  0.341656357  0.3545916725  0.369018247  0.385454679
## svi          0.541710372  0.553480497  0.5659053246  0.579302886  0.594067995
## gleason      0.063402833  0.061373728  0.0588793776  0.055915056  0.052224299
##
## (Intercept)  .            .            .            .            .
## lweight      0.567083034  0.577904675  0.588577079  0.598728953  0.607640794
## age         -0.009617693 -0.010968049 -0.012524059 -0.014335139 -0.016468427
## lbph         0.126178477  0.129642591  0.133272193  0.137050755  0.140935937
## lcp         -0.036796539 -0.054965954 -0.078076538 -0.108262379 -0.149116876
## pgg45        0.005340520  0.005702607  0.006189804  0.006872589  0.007882003
## lcavol       0.404236189  0.426145259  0.452197566  0.483962606  0.524027412
## svi          0.610303587  0.628512630  0.649279407  0.673464940  0.702375090
## gleason      0.047591554  0.041501934  0.033149650  0.021047902  0.002236316
##
## (Intercept)  .
## lweight      0.613980850
## age         -0.019008894
## lbph         0.144825646
## lcp         -0.206830251
## pgg45        0.009471377
## lcavol       0.576765946
## svi          0.737805454
## gleason     -0.029463345
```

```r
# Tune the value of lambda based on cross-validation
cv_fit_ridge <- cv.glmnet(x = x_inp, y = y_out, alpha = 0)

# Display the optimal lambda
best_lambda <- cv_fit_ridge$lambda.min
print(best_lambda)
```

```
## [1] 0.08788804
```

```r
## functions to compute testing/training error with glmnet
error <- function(dat, fit_ridge, lam, form, loss=L2_loss) {
  x_inp <- model.matrix(form, data=dat)
  y_out <- dat$lpsa
  y_hat <- predict(fit_ridge, newx=x_inp, s=lam)  ## see predict.elnet
  mean(loss(y_out, y_hat))
}
```

```r
## train_error at lambda=0
error(prostate_train, fit_ridge, lam=0, form=form)
```

```
## [1] 0.4391999
```

```r
## testing error at lambda=0
error(prostate_test, fit_ridge, lam=0, form=form)
```

```
## [1] 0.5214441
```

```r
## train_error at lambda=0.03
error(prostate_train, fit_ridge, lam=0.05, form=form)
```

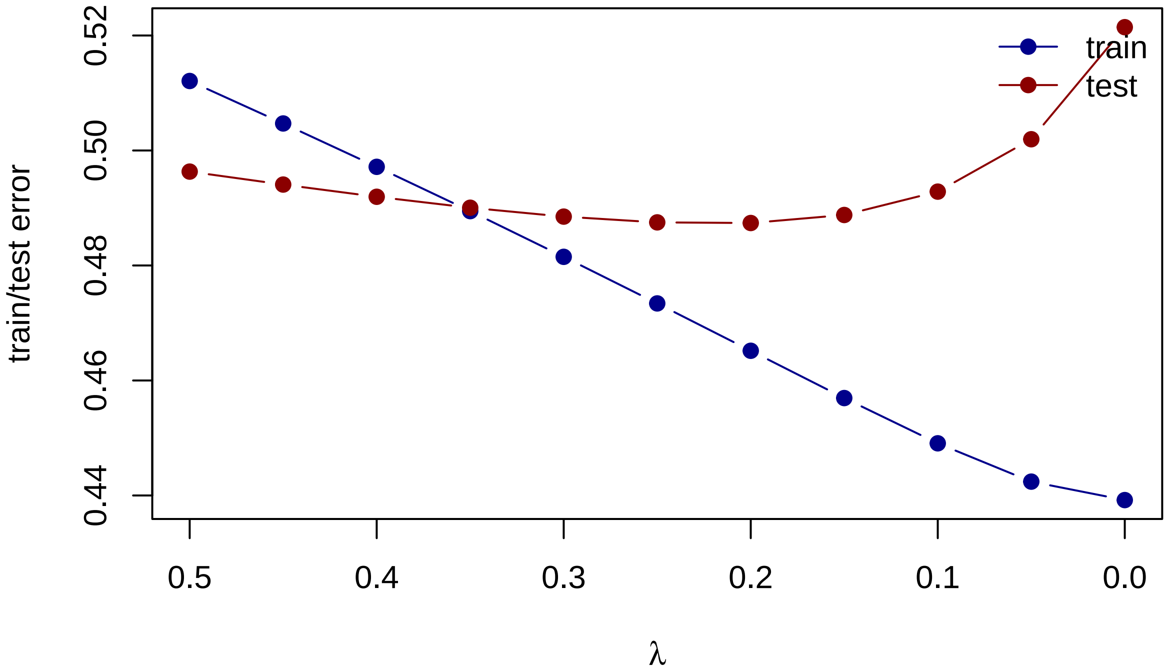
```
## [1] 0.4424079
```

```r
## testing error at lambda=0.03
error(prostate_test, fit_ridge, lam=0.05, form=form)
```

```
## [1] 0.5019552
```

```r
## compute training and testing errors as function of lambda
err_train_1 <- sapply(fit_ridge$lambda, function(lam)
  error(prostate_train, fit_ridge, lam, form))
err_test_1 <- sapply(fit_ridge$lambda, function(lam)
  error(prostate_test, fit_ridge, lam, form))
```

```r
## plot test/train error
plot(x=range(fit_ridge$lambda),
     y=range(c(err_train_1, err_test_1)),
     xlim=rev(range(fit_ridge$lambda)),
     type='n',
     xlab=expression(lambda),
     ylab='train/test error')
points(fit_ridge$lambda, err_train_1, pch=19, type='b', col='darkblue')
points(fit_ridge$lambda, err_test_1, pch=19, type='b', col='darkred')
legend('topright', c('train','test'), lty=1, pch=19,
       col=c('darkblue','darkred'), bty='n')
```




```r
colnames(fit_ridge$beta) <- paste('lam =', fit_ridge$lambda)
print(fit_ridge$beta %>% as.matrix)
```

```
##                lam = 0.5   lam = 0.45     lam = 0.4   lam = 0.35    lam = 0.3
## (Intercept)  0.000000000  0.000000000  0.0000000000  0.000000000  0.000000000
## lweight      0.515697286  0.525400575  0.5354170019  0.545775814  0.556311619
## age         -0.004889276 -0.005637483 -0.0064660393 -0.007390172 -0.008436051
## lbph         0.111264821  0.113946024  0.1167762533  0.119749176  0.122883708
## lcp          0.015168605  0.007943168 -0.0003765233 -0.010208916 -0.022217776
## pgg45        0.004390331  0.004515174  0.0046637167  0.004840594  0.005062742
## lcavol       0.329947512  0.341656357  0.3545916725  0.369018247  0.385454679
## svi          0.541710372  0.553480497  0.5659053246  0.579302886  0.594067995
## gleason      0.063402833  0.061373728  0.0588793776  0.055915056  0.052224299
##               lam = 0.25    lam = 0.2   lam = 0.15    lam = 0.1   lam = 0.05
## (Intercept)  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
## lweight      0.567083034  0.577904675  0.588577079  0.598728953  0.607640794
## age         -0.009617693 -0.010968049 -0.012524059 -0.014335139 -0.016468427
## lbph         0.126178477  0.129642591  0.133272193  0.137050755  0.140935937
## lcp         -0.036796539 -0.054965954 -0.078076538 -0.108262379 -0.149116876
## pgg45        0.005340520  0.005702607  0.006189804  0.006872589  0.007882003
## lcavol       0.404236189  0.426145259  0.452197566  0.483962606  0.524027412
## svi          0.610303587  0.628512630  0.649279407  0.673464940  0.702375090
## gleason      0.047591554  0.041501934  0.033149650  0.021047902  0.002236316
##                  lam = 0
## (Intercept)  0.000000000
## lweight      0.613980850
## age         -0.019008894
## lbph         0.144825646
## lcp         -0.206830251
## pgg45        0.009471377
## lcavol       0.576765946
## svi          0.737805454
## gleason     -0.029463345
```

```r
# Plot path diagram with effective degrees of freedom
plot(x = range(fit_ridge$lambda),
     y = range(as.matrix(fit_ridge$beta)),
     type = 'n',
     xlab = expression(df(lambda)),
     ylab = 'Coefficients')

# Plot coefficients
for (i in 1:nrow(fit_ridge$beta)) {
  points(x = fit_ridge$lambda, y = fit_ridge$beta[i,], pch = 19, col = '#00000055')
  lines(x = fit_ridge$lambda, y = fit_ridge$beta[i,], col = '#00000055')
}

# Add vertical line at df value chosen by cross-validation
abline(v = best_lambda, col = 'red', lty = 2)

# Add labels for coefficients
text(x = 0, y = fit_ridge$beta[, ncol(fit_ridge$beta)],
     labels = rownames(fit_ridge$beta),
     xpd = NA, pos = 4, srt = 45)

# Add horizontal line at y = 0
abline(h = 0, lty = 3, lwd = 2)
```