HW7

Jingshi

3/25/2017

6.8 #2

a)

The correct answer is (iii) because lasso is more restirvtive and less flexible than least squares regression. Therefore, it could reduce overfitting issues and has less variance. When its increase in bias is less than its decrease in variance, lasso can improve prediction accuracy, compared with least squares regression.

b)

The correct answer is (iii) because ridge regression is also more restrictive and less flexible than least squares regression. Especially when λ is large. For similar reasons as the previous question, ridge regression could reduce overfitting issues and has less variance. When its increase in bias is less than its decrease in variance, ridge regression can improve prediction accuracy, compared with least squares regression.

6.8 #4

a)

The correct answer is (iii) steadily increase becasue as λ increases from 0, the restriction of this regression's coefficients will steadily increase. Thus, it gradually becomes less and less flexible. So, training RSS increases steadily.

b)

The correct answer is (ii) decrease initially, and then eventually start increasing in a U shape becasue as λ increases from 0, the restriction of this regression's coefficients will steadily increase. Thus, it gradually becomes less and less flexible. Test RSS will decreas at first. Then, after a boundary point, test RSS will increase.

c)

The correct anser is (iv) steadily decrease because as λ increases from 0, the restriction of this regression's coefficients will steadily increase. Thus, it gradually becomes less and less flexible. The bias will increase. By bias viriance trade off, the variance will decrease steadily.

6.8 #8

e)

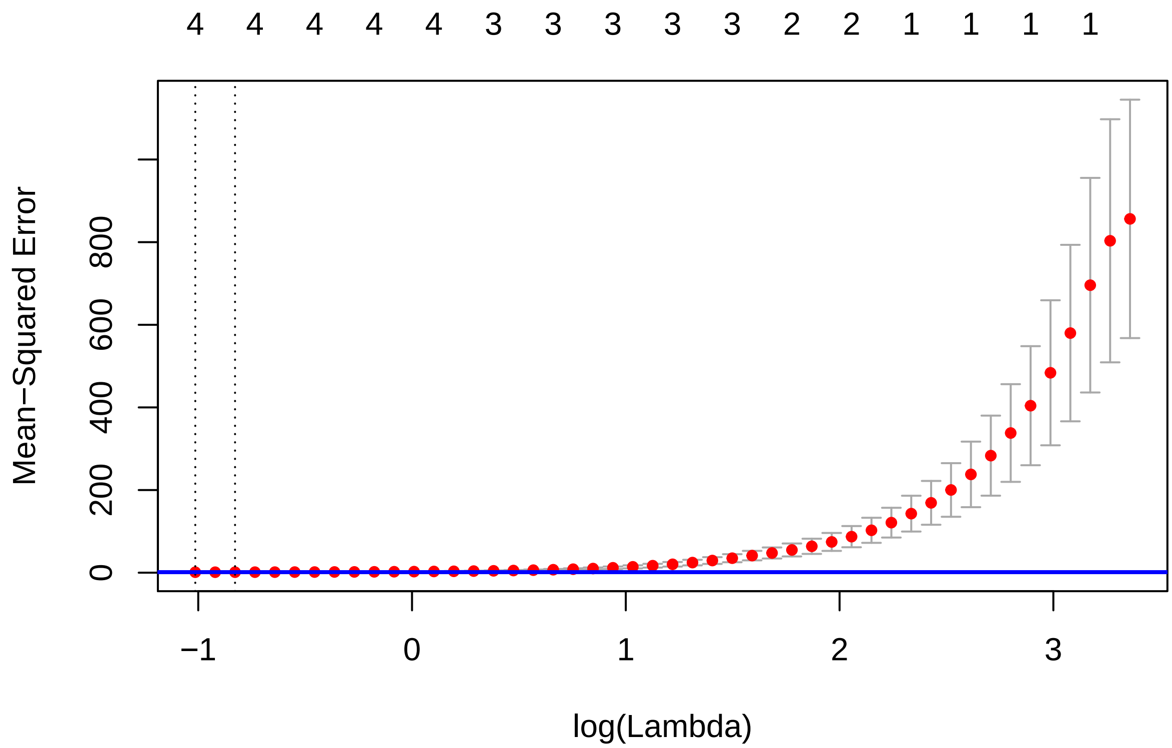
```r
#install.packages("glmnet")
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-5
```

```r
set.seed(6)
x <- rnorm(100)
e <- rnorm(100)
y <- 5 +  7* x +3 * x^2 + 6 * x^3 + e
my_matrix <- model.matrix(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7) + I(x^8) + I(x^9)
                          + I(x^10))
# Use cross-validation to select the optimal value of lambda.
cv_lasso <- cv.glmnet(my_matrix, y, alpha = 1)
# Create plots of the cross-validation error as a function of lambda.
plot(cv_lasso)
abline(h = min(cv_lasso$cvup), col = 4, lwd = 2)
```



The plot gives the estimated error for each λ (obtained from cross validation). Additionally, the "error bars" (one standard deviation above and below the estimated error) are also plotted. Moreover, the horizontal blue line is drawn where the cross validation error plus 1 standard deviation is minimal.

According to the plot, as λ increases, the mean squared error increases monotonically.

```r
my_best_lambda <- cv_lasso$lambda.min
# Take a look at my best lambda.
my_best_lambda
```

```
## [1] 0.362808
```

```r
fit_lasso <- glmnet(my_matrix, y, alpha = 1)
predict(fit_lasso, s = my_best_lambda, type = "coefficients")[1:11, ]
```

```
## (Intercept) (Intercept)           x      I(x^2)      I(x^3)      I(x^4)
##  5.19215856  0.00000000  7.03191624  2.67525007  5.94286468  0.03064618
##      I(x^5)      I(x^6)      I(x^7)      I(x^8)      I(x^9)
##  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
```

The estimated model:

$$\hat{y} = 5.19 + 7.03 \times x + 2.68 \times x^2 + 5.94 \times x^3 + 0.03 \times x^4$$

The resulting model has a term X^4 with very small estimated coefficient. However, the actual model does not have X^4 term. Besides this, the other coefficient estimates for the terms x, x^2, and x^3 are very close to the true model.

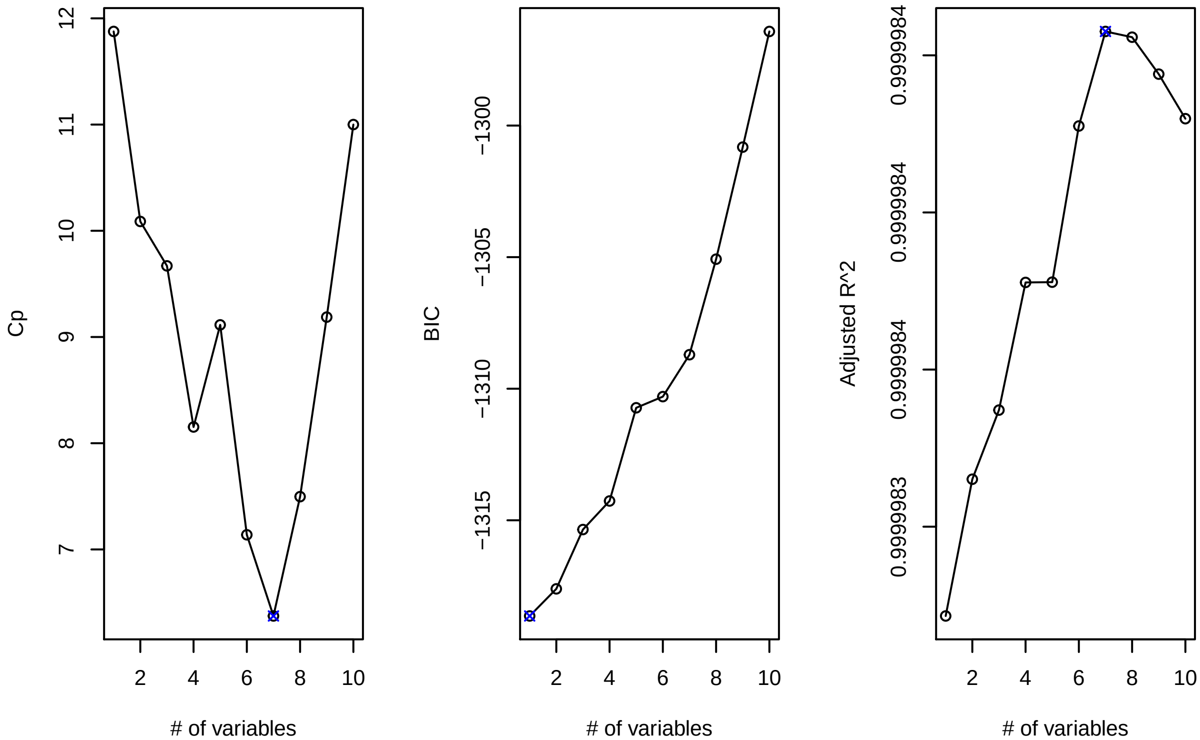
f)

best subset selection

```r
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 3.3.2
```

```r
# I set beta 7 to be 7.
y <- 5 + 7 * x^7 + e
my_df <- data.frame(y = y, x = x)
fullfit <- regsubsets(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7) + I(x^8) + I(x^9)
                          + I(x^10), data= my_df,nvmax = 10)

reg.summary <- summary(fullfit)
par(mfrow = c(1, 3))
plot(reg.summary$cp, xlab = "# of variables", ylab = "Cp", type = "o")
points(which.min(reg.summary$cp), reg.summary$cp[which.min(reg.summary$cp)],
       col = "blue", cex = 1, pch = 4)
plot(reg.summary$bic, xlab = "# of variables", ylab = "BIC", type = "o")
points(which.min(reg.summary$bic), reg.summary$bic[which.min(reg.summary$bic)],
       col = "blue", cex = 1, pch = 4)
plot(reg.summary$adjr2, xlab = "# of variables", ylab = "Adjusted R^2", type = "o")
points(which.max(reg.summary$adjr2), reg.summary$adjr2[which.max(reg.summary$adjr2)],
       col = "blue", cex = 1, pch = 4)
```



As the output shows, Cp achived its lowest at 7 variables, BIC achieved its lowest at 1 variable, Adjusted R^2 achieved its lowest at 7 variables.

```
coef(fullfit, 1)
```

```
## (Intercept)      I(x^7)
##    4.900514    7.000573
```

```
coef(fullfit, 7)
```

```
## (Intercept)      I(x^3)      I(x^4)      I(x^5)      I(x^7)      I(x^8)
##  4.92087583  2.43622678  0.32696657 -2.85755666  8.11210119 -0.14228466
##      I(x^9)     I(x^10)
## -0.14080160  0.03190193
```

As the output shows, BIC chosen the correct model with a term X^7 and its coefficient estimates are very close to the true model.

lasso

```
mymatrix <- model.matrix(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6) + I(x^7) + I(x^8) + I(x^9)
                     + I(x^10), data = my_df)
lasso_cv <- cv.glmnet(mymatrix, y, alpha = 1)
lambda <- lasso_cv$lambda.min
# Take a look at lambda
lambda
```

```
## [1] 22.23937
```

```
lassofit <- glmnet(mymatrix, y, alpha = 1)
predict(lassofit, s = lambda, type = "coefficients")[1:11, ]
```

```
## (Intercept) (Intercept)           x      I(x^2)      I(x^3)      I(x^4)
##    9.027868    0.000000    0.000000    0.000000    0.000000    0.000000
##      I(x^5)      I(x^6)      I(x^7)      I(x^8)      I(x^9)
##    0.000000    0.000000    6.776606    0.000000    0.000000
```

This time, as the output shows, lasso picks the correct model with only 1 variable. The coefficient estimates for x^7 is very close to the true coefficient. However, the estimated intercept differs from the actual intercept a lot.

MNIST and Lasso Question

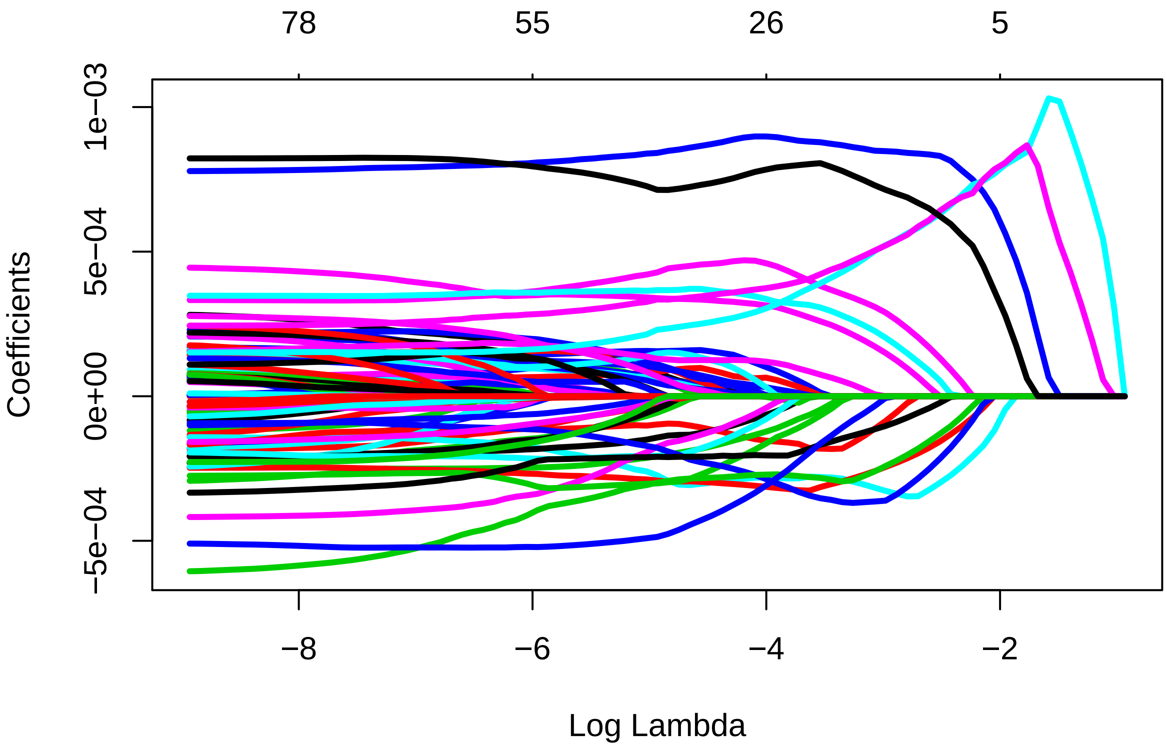
```r
library("glmnet")
load("mnist68.RData")
images_df <- mnist68

myv = rep(NA,784)
for (j in 1:784){myv[j] <- var(images_df[,j])}
myfeatures = (1:784)[myv > quantile(myv, .9)]
mydf = images_df[,c(myfeatures,785)]
mydf$labels = as.numeric(mydf$label==8)
```

a)

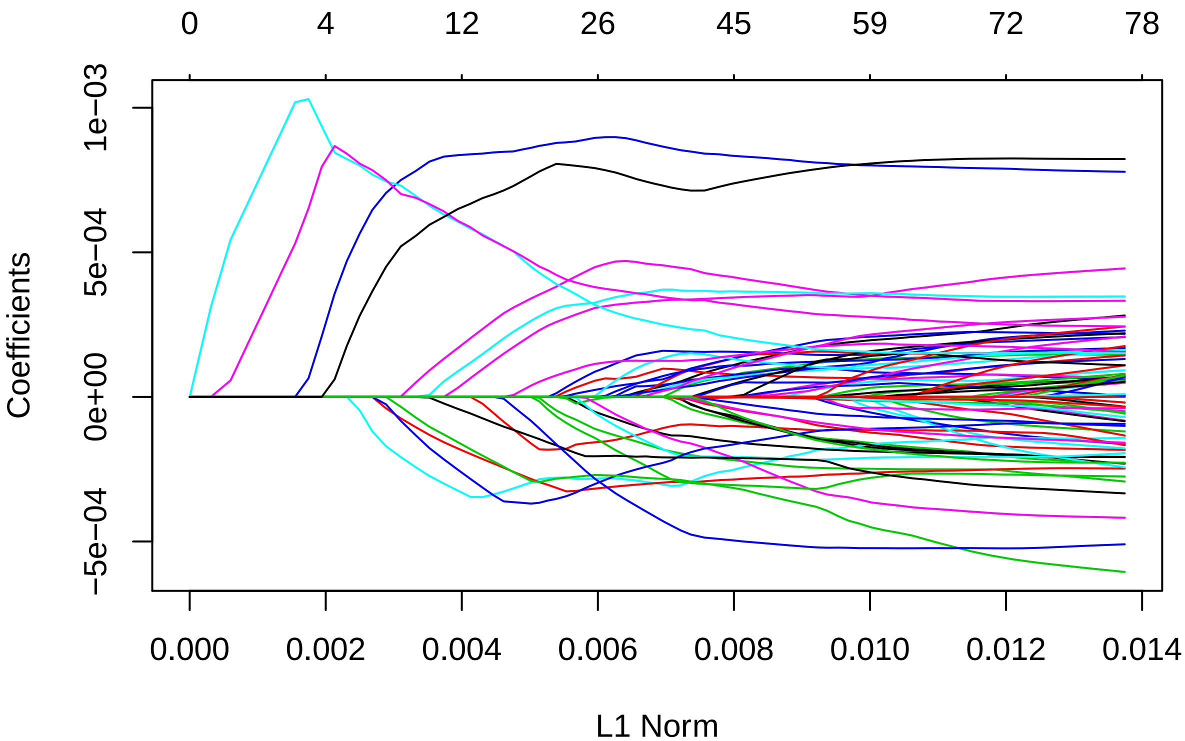
```r
# set up the proper model matrix from this data frame.
mymatrix <- model.matrix(labels ~. ,data = mydf)
# create a vector Y for the labels
y=mydf$labels
```

b)

```r
#cv_lasso <- cv.glmnet(mymatrix, y, alpha = 1)
lassofit <- glmnet(mymatrix, y, alpha = 1)
plot(lassofit, xvar = "lambda", lwd = 3)
```




```r
plot(lassofit)
```



The unusual thing I noticed is for some variables, some of the coefficients increase at first, then decrease, as λ increase or L1 Norm decrease. Rather than increasing/decreasing monotonically. This is because some features are dependent with each other, onece one disapear as λ increase, it will affect the other remaining features' trajectories. Moreover, λ is less than 1.

c)

```r
lasso_5 = glmnet(mymatrix, y, alpha = 1, lambda = 0.14)
coef(lasso_5)[,1]
```

```
##   (Intercept)   (Intercept)          V157          V158          V181 
##  2.800265e-01  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00 
##          V182          V185          V186          V187          V208 
##  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00 
##          V209          V212          V213          V214          V236 
##  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00 
##          V238          V239          V263          V264          V265 
##  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00 
##          V266          V291          V293          V319          V321 
##  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00 
##          V327          V346          V347          V349          V353 
##  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00 
##          V354          V355          V374          V375          V377 
##  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00 
##          V378          V379          V383          V402          V403 
##  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00 
##          V404          V411          V430          V431          V438 
##  0.000000e+00  0.000000e+00 -5.400519e-05  0.000000e+00  0.000000e+00 
##          V458          V459          V486          V491          V492 
##  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00 
##          V495          V496          V514          V518          V519 
##  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00 
##          V520          V521          V522          V523          V524 
##  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00 
##          V542          V545          V546          V547          V548 
##  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00 
##          V550          V551          V570          V573          V574 
##  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00 
##          V575          V578          V628          V629          V630 
##  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00 
##          V631          V632          V656          V657          V658 
##  0.000000e+00  0.000000e+00  5.562614e-04  8.296492e-04  7.804694e-04 
##          V659 
##  2.966307e-04
```

The last five features are V430, V656, V657, V658 and V659. I think some of them are not independent because their trajectories moves up and down when other features (in the five features) disapear, as λ increases.