# Homework2

Yifang Zhang

## Q1. Derive the coordinate descent algorithm.

First, for $\beta_0$ we have:

$$\begin{cases} \dfrac{\partial}{\partial \beta_0} RSS(\beta_0) = -2\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j) \\ \dfrac{\partial}{\partial \beta_0}\lambda\sum_{i=1}^{p}|\beta_j| = 0 \end{cases}$$

Then set the derivative to zero and we will get:

$$-2\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j) + 0 \overset{set}{=} 0 \implies \hat{\beta_0} = \frac{\sum_{i=1}^{n} y_i - \sum_{i=1}^{n}\sum_{j=1}^{p} x_{ij}\beta_j}{n}$$

Now for $j = 1, 2, \cdots, p$, we have:

$$\begin{aligned} \frac{\partial}{\partial \beta_j} RSS(\beta_j) &= -2\sum_{i=1}^{n} x_{ij}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j) \\ &= -2\sum_{i=1}^{n} x_{ij}(y_i - \beta_0 - \sum_{k\neq j}^{p} x_{ik}\beta_k - x_{ij}\beta_j) \\ &= -2\sum_{i=1}^{n} x_{ij}(y_i - \beta_0 - \sum_{k\neq j}^{p} x_{ik}\beta_k) + 2\beta_j\sum_{i=1}^{n} x_{ij}^2 \end{aligned}$$

Then we set two terms $\rho_j$ and $z_j$ for $j = 1, 2, \cdots, p$ as following

$$\begin{cases} \rho_j = \sum_{i=1}^{n} x_{ij}(y_i - \beta_0 - \sum_{k\neq j}^{p} x_{ik}\beta_k) \\ z_j = \sum_{i=1}^{n} x_{ij}^2 \end{cases}$$

and rewrite $\frac{\partial}{\partial \beta_j} RSS(\beta_j)$ as:

$$\frac{\partial}{\partial \beta_j} RSS(\beta_j) = -2\rho_j + 2\beta_j z_j,\ j = 1, 2, \cdots, p$$

Since $|x|$ is not derivable at $x = 0$, we consider the subgradient method for $\frac{\partial}{\partial \beta_j}\lambda\sum_{i=1}^{p}|\beta_j| = \partial_{\beta_j}\lambda|\beta_j|$ as:

$$\lambda\partial_{\beta_j}|\beta_j| = \begin{cases} -\lambda, & \text{if } \beta_j < 0 \\ [-\lambda, \lambda], & \text{if } \beta_j = 0 \\ \lambda, & \text{if } \beta_j > 0 \end{cases}$$

So now we set the derivative of the lasso coordinate descent algorithm and set it to zero as following:

$$\partial_{\beta_j}[lasso] = \begin{cases} -2\rho_j + 2\beta_j z_j - \lambda, & \text{if } \beta_j < 0 \\ [-2\rho_j - \lambda, -2\rho_j + \lambda], & \text{if } \beta_j = 0 \\ -2\rho_j + 2\beta_j z_j + \lambda, & \text{if } \beta_j > 0 \end{cases} \overset{set\ to\ 0}{=} 0$$

**case1**: $\beta_j < 0$

$$-2\rho_j + 2\hat{\beta}_j z_j - \lambda = 0 \implies \hat{\beta}_j = \frac{2\rho_j + \lambda}{2z_j} = \frac{\rho_j + \frac{\lambda}{2}}{z_j} < 0 \implies \rho_j < -\frac{\lambda}{2}$$

**case2:** $\beta_j = 0$

$$\text{for } \hat{\beta}_j = 0 \implies 0 \in [-2\rho_j - \lambda, -2\rho_j + \lambda] \implies -\frac{\lambda}{2} \le \rho_j \le \frac{\lambda}{2}$$

**case3:** $\beta_j > 0$

$$-2\rho_j + 2\hat{\beta}_j z_j + \lambda = 0 \implies \hat{\beta}_j = \frac{\rho_j - \frac{\lambda}{2}}{z_j} > 0 \implies \rho_j > \frac{\lambda}{2}$$

In conclusion, from the derivative of lasso coordinate descent, for $j = 1, 2, \cdots, p$ we have:

$$\hat{\beta}_j = \begin{cases} \frac{\rho_j + \frac{\lambda}{2}}{z_j}, & \text{if } \rho_j < -\frac{\lambda}{2} \\ 0, & \text{if } \rho_j \in [-\frac{\lambda}{2}, \frac{\lambda}{2}] \\ \frac{\rho_j - \frac{\lambda}{2}}{z_j}, & \text{if } \rho_j > \frac{\lambda}{2} \end{cases}$$

Hence, in general we have:

$$\hat{\beta}_0 = \frac{\sum_{i=1}^n y_i - \sum_{i=1}^n \sum_{j=1}^p x_{ij}\beta_j}{n} \ ;$$

$$\hat{\beta}_j = \begin{cases} \frac{\rho + \frac{\lambda}{2}}{z_j}, & \text{if } \rho_j < -\frac{\lambda}{2} \\ 0, & \text{if } \rho_j \in [-\frac{\lambda}{2}, \frac{\lambda}{2}] \\ \frac{\rho - \frac{\lambda}{2}}{z_j}, & \text{if } \rho_j > \frac{\lambda}{2} \end{cases} \ ; \text{for } j = 1, 2, \cdots, p$$

where $\rho_j = \sum_{i=1}^n x_{ij}(y_i - \beta_0 - \sum_{k \ne j}^p x_{ik}\beta_k)$ and $z_j = \sum_{i=1}^n x_{ij}^2$.

# Q2

## (1). Analyze the prostate cancer data set from ESL.

```
##       lcavol  lweight age      lbph svi       lcp gleason pgg45       lpsa
## 1 -0.5798185 2.769459  50 -1.386294   0 -1.386294       6     0 -0.4307829
## 2 -0.9942523 3.319626  58 -1.386294   0 -1.386294       6     0 -0.1625189
## 3 -0.5108256 2.691243  74 -1.386294   0 -1.386294       7    20 -0.1625189
## 4 -1.2039728 3.282789  58 -1.386294   0 -1.386294       6     0 -0.1625189
## 5  0.7514161 3.432373  62 -1.386294   0 -1.386294       6     0  0.3715636
## 6 -1.0498221 3.228826  50 -1.386294   0 -1.386294       6     0  0.7654678
```

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

```
## Loaded glmnet 4.1-1
```

```
##
## Call:
## lm(formula = lpsa ~ lcavol + lweight + age + lbph + svi + lcp +
##     gleason + pgg45, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.64870 -0.34147 -0.05424  0.44941  1.48675
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.429170   1.553588   0.276  0.78334
## lcavol       0.576543   0.107438   5.366 1.47e-06 ***
## lweight      0.614020   0.223216   2.751  0.00792 **
## age         -0.019001   0.013612  -1.396  0.16806
## lbph         0.144848   0.070457   2.056  0.04431 *
## svi          0.737209   0.298555   2.469  0.01651 *
## lcp         -0.206324   0.110516  -1.867  0.06697 .
## gleason     -0.029503   0.201136  -0.147  0.88389
## pgg45        0.009465   0.005447   1.738  0.08755 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7123 on 58 degrees of freedom
## Multiple R-squared:  0.6944, Adjusted R-squared:  0.6522
## F-statistic: 16.47 on 8 and 58 DF,  p-value: 2.042e-12
```

```
##                    2.5 %      97.5 %
## (Intercept) -2.680674329 3.539014595
## lcavol       0.361482785 0.791603586
## lweight      0.167204780 1.060835229
## age         -0.046248270 0.008246226
## lbph         0.003813690 0.285882474
## svi          0.139585747 1.334831542
## lcp         -0.427546584 0.014898130
## gleason     -0.432120510 0.373114741
## pgg45       -0.001437213 0.020367537
```

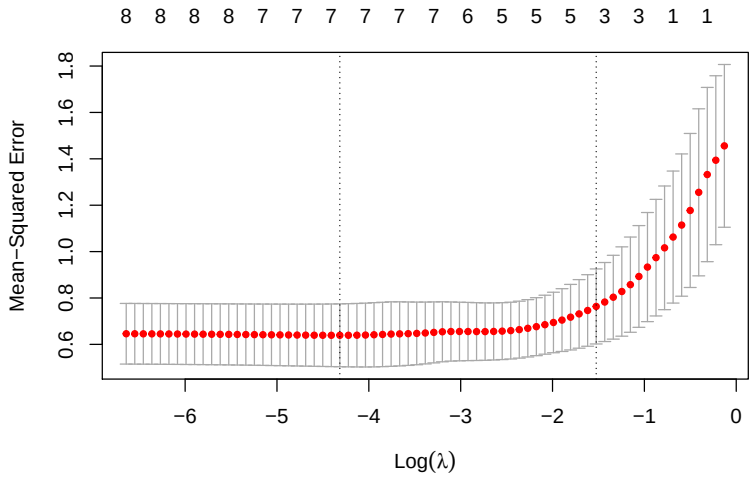
```
##                  [,1]
## Train Error 0.4391998
## Test Error  0.5212740
```

According to the p-value, only lcavol, lweight, lbph, svi variables are significant in our fitted model when the significant level is $\alpha = 0.05$. The training error is about 0.4392 and the test error is about 0.5213.

**i. Why standardization is approriate:**

We can find that our data have large differences between their ranges and are measured in different measurement units. lcavol, lweight, lbph, lcp and lpsa variables are logarithmized variables, while the unit of age variable is still years. Svi is a binary variable and gleason is a categorical variable. And the unit of pgg45 variable is percent. These differences in the ranges and measurement units of initial features can cause trouble to our machine learning models, so we have to standardize our data to uniform the range of our data.

**ii. Select the parameter with 5-fold CV using minimun/one standard deviation rule for CV, refit model and report TestErr:**




```
## Best Lambda under minimum rule
##                       0.0133582
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  0.163730802
## lcavol       0.543994702
## lweight      0.595959290
## age         -0.015001820
## lbph         0.134800309
## svi          0.668900088
## lcp         -0.144931892
## gleason      .
## pgg45        0.007383417
```

```
##                   [,1]
## Train Error 0.4393627
## Test Error  0.5165135
```

```
## Best Lambda under one-std rule
##                       0.2177054
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
## (Intercept) 0.4228791
## lcavol      0.4494666
## lweight     0.3837892
```

```
## age          .
## lbph         .
## svi          0.2118731
## lcp          .
## gleason      .
## pgg45        .
```

```
##                   [,1]
## Train Error 0.5210112
## Test Error  0.4005308
```

**a. Minimum CV rule:**

When we select the parameter with 5-fold CV using the minimum CV rule, the best $\lambda$ = 0.0134. Results indicated that except for the variable gleason, all variables are selected with its corresponding estimated coefficients shown above.

After refitting the model with seven selected variables, the test error is about 0.5165.

**b. One-standard deviation rule:**

When we select the parameter with 5-fold CV using the one-standard deviation CV rule, the best $\lambda$ = 0.2177. Results indicated that only the following three variables are selected: lcavol, lweight and svi. The corresponding estimated coefficients are shown above.

After refitting the model with three selected variables, the test error is about 0.4, which is smaller than previous two test errors.

## (2). Write code for coordinate descent algorithm using objective function 1 in Problem 1.

The R code of coordinate descent algorithm for lasso using objective function 1 is:

```
soft_thresh=function(rho,lambda){
  out=rep(0,length(rho))
  out[rho > lambda/2] = rho[rho > lambda/2] -lambda/2
  out[rho < -lambda/2] = rho[rho < -lambda/2] + lambda/2
  out
}

coordinate.descent=function(x,y,lambda,tol=1e-12,iter=100){
  beta=matrix(0,nrow=8,ncol=1)
  tol_curr=1
  p=ncol(x)
  z=rep(0,p)
  i=1

  while (tol < tol_curr && i<iter){
    beta_old=beta
    z=colSums(x^2)
    for (j in 1:p){
      rho_j=sum(x[,j]*(y-x[,-j]%*%beta_old[-j]))
      beta_old[j]=soft_thresh(rho_j,lambda)/z[j]
    }
    tol_curr=crossprod(beta-beta_old)
    beta=beta_old
    i=i+1
  }
```

```
  beta
}
```

Then we set a set of grid values for $\lambda$ from $10^2$ to $10^{-5}$ and use the following function to report the best $\lambda$ which will provide the minimum BIC value and then report the selected variables and estimated regression coefficients by the same function:

```
lasso.BIC=function(x,y,lambda){
  t=length(lambda)
  if (t==1){return(coordinate.descent(x,y,lambda))}
  else{
    BIC=rep(0,t)
    for (k in 1:t){
      model=glmnet(x,y,lambda=lambda[k],alpha=1)
      tLL=model$nulldev-deviance(model)
      n=model$nobs
      df=model$df
      BIC[k]=-tLL+log(n)*df
      }
    lambda.bic=lambda[which.min(BIC)]
    return(lambda.bic)
  }
}
```

```
## [1] 0.178865
```

```
##              [,1]
## [1,]  0.58849973
## [2,]  0.24184258
## [3,] -0.11562065
## [4,]  0.17395244
## [5,]  0.25380776
## [6,] -0.23039969
## [7,] -0.01160838
## [8,]  0.22085702
```

```
## [1] "Estimated Coefficients after scaling back:"
```

```
##               [,1]
## [1,]  0.572028433
## [2,]  0.612882712
## [3,] -0.018614253
## [4,]  0.143546024
## [5,]  0.729904932
## [6,] -0.198666742
## [7,] -0.019779186
## [8,]  0.009103676
```

The best $\lambda$ = 0.179 by BIC and the corresponding estimated coefficients after scaling back are shown above. We can find that by the coordinate descent algorithm with object function 1, all predict variables have nonzero estimated coefficients. In addition, this best $\lambda$ is quite close to the best $\lambda$ we got in Problem (1) under one-standard deviation rule, while the estimated coefficients are very close to what we got in Problem (1) by using minimum CV rule.

In fact, if we choose a set of larger $\lambda$, we can find something like following:

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

```
## [1] 10
```

```
##            [,1]
## [1,]  0.5694428
## [2,]  0.2380567
## [3,] -0.1020850
## [4,]  0.1671390
## [5,]  0.2408070
## [6,] -0.1905163
## [7,]  0.0000000
## [8,]  0.1912300
```

```
##            [,1]
## [1,] 0.47694921
## [2,] 0.19364568
## [3,] 0.00000000
## [4,] 0.09481132
## [5,] 0.14799136
## [6,] 0.00000000
## [7,] 0.00000000
## [8,] 0.05913080
```

The best $\lambda$ will just be the smallest value in our sets and the coordinate descent result shows that when the value of $\lambda$ becomes larger, more variables will be moved from our model. Since we still want to select the best $\lambda$ by BIC, I will still use $\lambda = 0.179$

Then we refit the model in lm() without intercept and in glmnet() where $\lambda = 0.179$ both as following:

```
refit.BIC1=lm(lpsa ~ lcavol+lweight+age+lbph+svi+lcp+gleason+pgg45+0,data=train)
mean((ytest-predict(refit.BIC1,newdata = xtest))^2)
```

```
## [1] 0.5179699
```

```
refit.glm1=glmnet(as.matrix(xtrain),as.matrix(ytrain),alpha = 1, lambda=bestlambda.BIC)
mean((ytest-predict(refit.glm1,newx = as.matrix(xtest)))^2)
```

```
## [1] 0.4648626
```

The test error is 0.518 and 0.465, respectively. And comparing to the test error we got in Problem (1), we can find that there is no big difference.

Now if we try larger values like $\lambda = 1$ or $\lambda = 10$ and refit our model in lm(), the test error is:

```
## [1] 0.5211396
```

```
## [1] 0.576894
```

Even though less variables are considered in our model, the test error is becoming larger. And if we check the correlation between the response and eight predictors, we will find that only age and lbph have weaker correlation with lpsa comparing to other variables. So after checking different sets of $\lambda$ and different results, I confirmed that all variables should be selected.

```
##         lcavol lweight  age  lbph   svi   lcp gleason pgg45 lpsa
## lcavol    1.00    0.28 0.22  0.03  0.54  0.68    0.43  0.43 0.73
## lweight   0.28    1.00 0.35  0.44  0.16  0.16    0.06  0.11 0.43
## age       0.22    0.35 1.00  0.35  0.12  0.13    0.27  0.28 0.17
## lbph      0.03    0.44 0.35  1.00 -0.09 -0.01    0.08  0.08 0.18
## svi       0.54    0.16 0.12 -0.09  1.00  0.67    0.32  0.46 0.57
## lcp       0.68    0.16 0.13 -0.01  0.67  1.00    0.51  0.63 0.55
## gleason   0.43    0.06 0.27  0.08  0.32  0.51    1.00  0.75 0.37
```

```
## pgg45      0.43    0.11 0.28  0.08  0.46  0.63    0.75  1.00 0.42
## lpsa       0.73    0.43 0.17  0.18  0.57  0.55    0.37  0.42 1.00
```

## (3). Write code for coordinate descent algorithm using objective function 2 in Problem 1.

When the intercept is considered in our objective function, the algorithm to find the best $\lambda$ and implement coefficients is:

```
coordinate.intercept=function(x,y,lambda,tol=1e-12,iter=100){
  beta=matrix(0,nrow=9,ncol=1)
  tol_curr=1
  p=ncol(x)
  n=nrow(x)
  z=rep(0,p)
  i=1

  while (tol < tol_curr && i<iter){
    beta_old=beta
    z=colSums(x^2)
    for (j in 0:p){
      if (j==0){
        beta_old[j+1]=(sum(y)-sum(sapply(1:n,function(k) x[k,]%*%beta_old[-(j+1),])))/n
      }
      else{
        rho_j=sum(x[,j]*(y-x[,-j]%*%beta_old[-c(1,j+1)]))-beta_old[1]*sum(x[,j])
        beta_old[j+1]=soft_thresh(rho_j,lambda)/z[j]
      }
    }
    tol_curr=crossprod(beta-beta_old)
    beta=beta_old
    i=i+1
  }
  beta
}

lasso.BIC2=function(x,y,lambda){
  t=length(lambda)
  if (t==1){return(coordinate.intercept(x,y,lambda))}
  else{
    BIC=rep(0,t)
    for (k in 1:t){
      model=glmnet(x,y,lambda=lambda[k],alpha=1)
      tLL=model$nulldev-deviance(model)
      n=model$nobs
      df=model$df
      BIC[k]=-tLL+log(n)*df
    }
    lambda.bic=lambda[which.min(BIC)]
    return(lambda.bic)
  }
}
```

Still, we set the grid values for $\lambda$ from 10 to $10^{-6}$ and report the best $\lambda$ by BIC. The result is shown as following:

```
## [1] 0.2154435
```

```
##              [,1]
##  [1,]  2.45234509
##  [2,]  0.71081239
##  [3,]  0.29210185
##  [4,] -0.13965596
##  [5,]  0.21010706
##  [6,]  0.30656046
##  [7,] -0.27830889
##  [8,] -0.01403962
##  [9,]  0.26678277
```

```
## [1] "Estimated Coefficients after scaling back"
```

```
##               [,1]
##  [1,]  2.452345085
##  [2,]  0.572040798
##  [3,]  0.612885829
##  [4,] -0.018615313
##  [5,]  0.143549591
##  [6,]  0.729924942
##  [7,] -0.198687718
##  [8,] -0.019805817
##  [9,]  0.009104666
```

The best $\lambda = 0.215$ by BIC and the corresponding estimated coefficients after scaling back are shown above. We can find that by the coordinate descent algorithm with object function 2, all predict variables have nonzero estimated coefficients again. In addition there is no big difference between two best $\lambda$ and the estimated coefficients are almost the same, except that the estimated intercept is larger than what we got in Problem (1). Thus, in general there is no big difference between the best $\lambda$, the selected variables and the estimated coefficients obtained by two objective functions.

Then we refit the model in lm() without intercept and in glmnet() with $\lambda = 0.215$ again as following:

```
refit.BIC2=lm(lpsa ~ lcavol+lweight+age+lbph+svi+lcp+gleason+pgg45,data=train)
mean((ytest-predict(refit.BIC2,newdata = xtest))^2)
```

```
## [1] 0.521274
```

```
refit.glm2=glmnet(as.matrix(xtrain),as.matrix(ytrain),alpha = 1, lambda=bestlam.BIC2)
mean((ytest-predict(refit.glm1,newx = as.matrix(xtest)))^2)
```

```
## [1] 0.4648626
```

The test error is 0.52 and 0.465, respectively. Comparing to what we got in Problem (2), basically there is no difference.

## (4). Can we still rely on the p-value and confidence intervals obtained?

**Answer:** No, because inferential statistics like p-values or confidence intervals are only valid when the model is chosen independent of the data we used. While when we selected the model size we actually have utilized the information form the data. That means when we used the same data twice to both select the model size and do the inference, we will get too optimistic results, including too narrow confidence intervals and too samll p-values.

# Q3. Classify three iris species by the following four classifiers.

## (1). LDA

```
## [1] 0.9333333
```

```
##                 pred.LDA.classes
## observed.classes setosa versicolor virginica
##       setosa         12          0         0
##       versicolor      0         11         0
##       virginica       0          3        19
```

The prediction accurancy on the test data by using LDA is about 93.33%. Confusion matrix is shown above.

## (2). QDA

```
## [1] 0.9333333
```

```
##                 pred.QDA.classes
## observed.classes setosa versicolor virginica
##       setosa         12          0         0
##       versicolor      0          9         2
##       virginica       0          1        21
```

The prediction accurancy on the test data by using QDA is 93.33%, which is the same as the accurancy under LDA. Confusion matrix is shown above.

## (3). Logistic regression

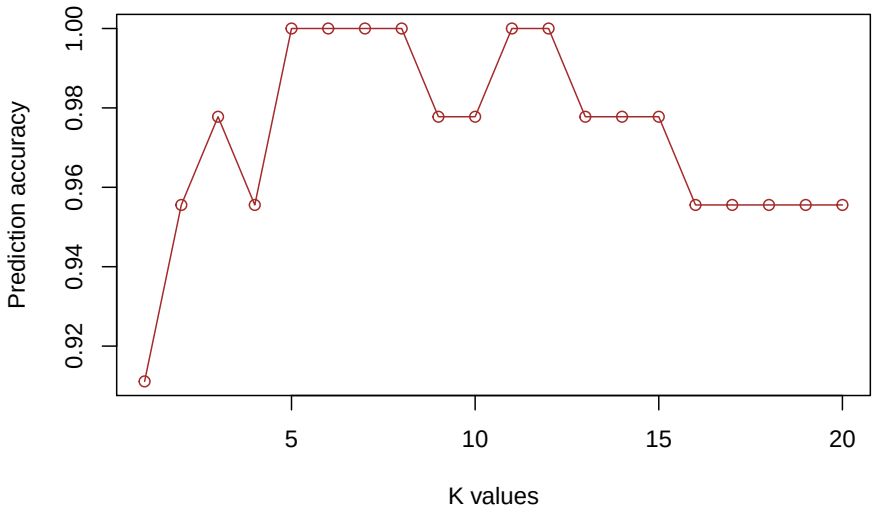
```
## # weights:  18 (10 variable)
## initial  value 115.354290
## iter  10 value 7.509920
## iter  20 value 3.767000
## iter  30 value 2.971296
## iter  40 value 2.565087
## iter  50 value 1.349492
## iter  60 value 1.211743
## iter  70 value 1.169731
## iter  80 value 1.031143
## iter  90 value 0.941652
## iter 100 value 0.841988
## final  value 0.841988
## stopped after 100 iterations
```

```
## [1] 0.9555556
```

```
##                 pred.mullog
## observed.classes setosa versicolor virginica
##       setosa         12          0         0
##       versicolor      0         11         0
##       virginica       0          2        20
```

The prediction accurancy on the test data by using multinormal logistic regression is about 95.56%, which is higher than perivous two classifiers. Confusion matrix is shown above.

## (4). K-nearest neighbors




```
## [1] 5
```

```
##                  pred.knn5
## observed.classes setosa versicolor virginica
##       setosa         12          0         0
##       versicolor      0         11         0
##       virginica       0          0        22
```

For K-nearest neighbors, we tried a set of K values and found that when $K$ = 5 we will get the highest prediction accuracy on the test data, which is 1. That means when $K$ = 5, our predicition of species on the test data is completely accurate. The confusion matrix under $K$ = 5 is given above.

**Summary:** After comparing perivous four classification methonds we can find that, first if we choose the appropriate K value, KNN method can provide a completely accurate prediction on the test iris data. Also, logistic regression method is a little better than LDA and QDA. Finally, we can find that all four classification methods provide completely accurate prediction for setosa on the test data.

## Q4. Use sparse logistic regression to classify the subject.

```
setwd("~/Desktop")
genedata=read.csv(file = "~/Desktop/HW2_Ch10Ex11.csv", header = F)
type=rep(0,40)
type[21:40]=1
set.seed(1007)
cvfit=cv.glmnet(t(genedata),type,alpha=1,family="binomial",nfold=5)
cvfit$lambda.min
```

```
## [1] 0.004292158
```

```
cvfit$lambda.1se
```

```
## [1] 0.005944151
```

```
sparse.logis=glmnet(t(genedata),type,family = "binomial", alpha = 1,
                    lambda = cvfit$lambda.min)
coefsparse=as.matrix(coef(sparse.logis))
num.effevarible=nrow(genedata)-sum(coefsparse == 0)
num.effevarible
```

```
## [1] 24
```

```
res.sparse=predict(sparse.logis,newx = t(genedata), type = "response")
pred.sparse=ifelse(res.sparse>0.5,1,0)
accuracy.sparse=mean(pred.sparse == type)
accuracy.sparse
```

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

**1). The selection of $\lambda$ :**

Here I selected the best $\lambda$ for lasso by using 5-fold cross-validation. I reported the best $\lambda$ under both the minimum CV rule and the one-standard deviation rule for CV, i.e. $\lambda_{min}$ = 0.0043 and $\lambda_{1se}$ = 0.0059. The difference is quite small.

Then I tried 10-fold CV as well and found that the best $\lambda$ under the minimum CV rule is still 0.0043 and the best $\lambda$ under one-standard deviation rule is 0.0062 which is very close to the privous one.

Finally, I chose $\lambda$ = 0.0043 to classify the subject due to its consistency.

**2). Summary:**

First, by using sparse logistic regression with L1 penalty and the best $\lambda$ = 0.0043, we found that only 24 genes are effective with a nonzero estimated coefficient. That means, among a large number of variables (1000 genes), only 24 are effective and selected in our model to predict the type of our samples.

Then we used the model we have gotten on the gene expression data again and obtained the prediction accuracy on that original data set. The result shows that prediction accuracy is 1, which means the sparse logistic regression with lasso penalty and the best $\lambda$ obtained by using 5-fold CV with the minimum rule can classify the subject completely accurate. Also it shows that those 24 genes contain enough information to classify the subject as healthy or diseased.