STAT542 Statistical Learning Homework 3

Huamin Zhang

Nov 1, 2017

Name: Huamin Zhang (huaminz2@illinois.edu)

Question 1

Install the quadprog package and utilize the function solve.QP to solve SVM. The solve.QP function is trying to perform the minimization problem:

$$\begin{aligned} minimize \quad & \frac{1}{2}\beta^T D\beta - d^T\beta \\ subjectto \quad & A^T\beta \geq b_0 \end{aligned}$$

a) [10 points]

Answer:

The primal of the linear separable SVM optimization problem is

$$\begin{aligned} minimize \quad & \frac{1}{2}||w||^2 \\ subject\ to \quad & y_i(x_i^T w + b) \geq 1 \quad i = 1, 2, \ldots, n \end{aligned}$$

where w is the normal vector of the dicision hyperplane and b is The intercept of the dicision hyperplane. Assume there is n observations in X and each observation has p features. Let D be a $(p+1) \times (p+1)$ positive definite matrix, and d be a (p+1)-column vector, A be an $(p+1) \times n$ matrix, and b_0 is an n-column vector.

Then the primal of the linear separable SVM optimization problem can be transformed into a form that can be solved by solve.QP if we set $d = [0, \ldots, 0]^T$, $b_0 = [1, \ldots, 1]^T$, $\beta = [w, b]^T$ and

$$D = \begin{bmatrix} I_{p\times p} & 0 \\ 0 & 0 \end{bmatrix} \quad A = \begin{bmatrix} y_1 x_{11} & \ldots & y_1 x_{p1} & y_1 \cdot 1 \\ \vdots & \ddots & \vdots & \vdots \\ y_n x_{n1} & \ldots & y_n x_{np} & y_n \cdot 1 \end{bmatrix}^T$$

.

```
library(quadprog)
# Create the linear separable data
set.seed(1); n <- 40; p <- 2
xpos <- matrix(rnorm(n*p,mean=0,sd=1),n,p)
xneg <- matrix(rnorm(n*p,mean=4,sd=1),n,p)
```

```
x <- rbind(xpos,xneg)
y <- matrix(c(rep(1,n),rep(-1,n)))
y_factor <- matrix(as.factor(c(rep(1,n),rep(-1,n))))
# X: Input matrix, each row is an observation vector
# Y: Response variable
# eps: The coefficient of a ridge matrix added to D to make sure it is positive definite
# Output:A (p+1)-vector. The fisrt p element is w and the p+1 element is b
HardSVM_primal <- function(X,Y,eps){
  n = dim(X)[1]
  p = dim(X)[2]
  # Conduct the parameter for solve.QP as we assumed
  D = diag(p + 1)
  D[p+1,p+1] = 0
  # To make sure the D is positive definite, we add a ridge to D
  D = D + eps*diag(p + 1)
  d = matrix(0,p+1)
  A = cbind(matrix(rep(Y, p), ncol = p)*X, Y)
  b0 = matrix(1, nrow = n)
  # Call the QP solver:
  result = solve.QP(Dmat = D, dvec = d, Amat = t(A), bvec = b0)
  return(result$solution)
}
# Calculte the SVM solution (decision line) for the linear separable data
HardSVM_primal_solution = HardSVM_primal(x,y,10^-7)
w_HardSVM_primal = matrix(HardSVM_primal_solution[1:p],nrow = 1,
                          dimnames = list(NULL,c('X1','X2')))
b_HardSVM_primal = HardSVM_primal_solution[p+1]
```

Here we compare our solution (the decision line) to the results produced by e1071 package and find they are almost the same, which means the code is correct.

```
# Calculte the SVM solution for the linear separable data with e1071 package
library('e1071')
# The cost is 1000 because this is a separable problem
svm.fit <- svm(y_factor ~ ., data = data.frame(x, y_factor), type='C-classification',
               kernel='linear',scale=FALSE, cost = 1000)
w = t(svm.fit$coefs) %*% svm.fit$SV
b = -svm.fit$rho
#b <- -(max(x[y == -1, ] %*% t(w)) + min(x[y == 1, ] %*% t(w)))/2
e1071_HardSVM_solution = c(w,b)
HardSVM_solution_compare = rbind(HardSVM_primal_solution,e1071_HardSVM_solution)
colnames(HardSVM_solution_compare) = c('X1','X2','b0')
HardSVM_solution_compare
```

```
##                                 X1         X2       b0
## HardSVM_primal_solution -0.9334294 -0.3849838 2.782384
## e1071_HardSVM_solution  -0.9338740 -0.3844957 2.781953
```

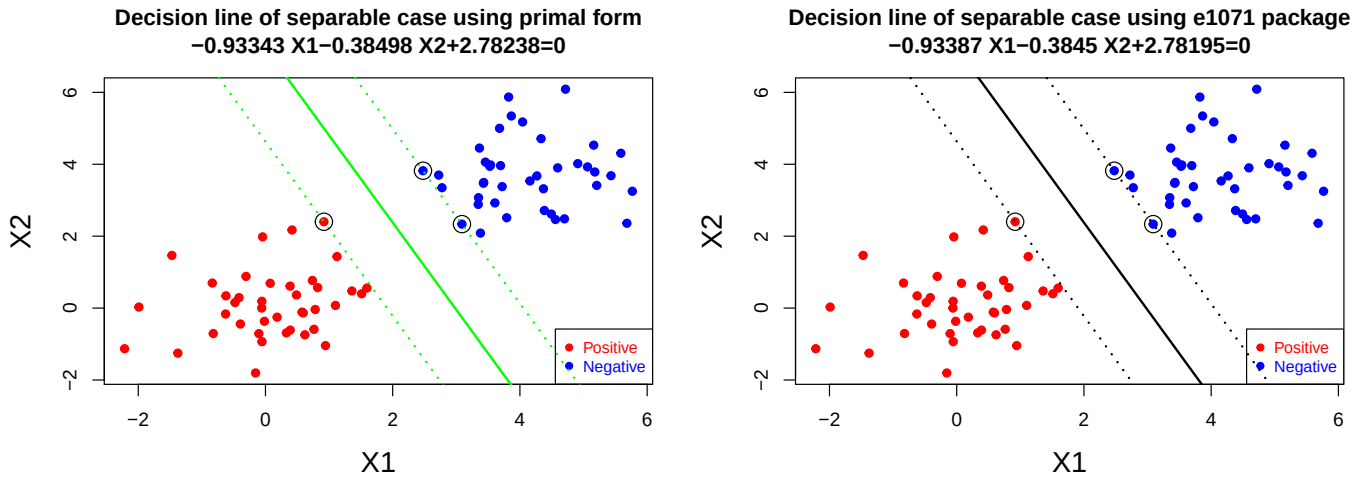
Also, we can compare the support vectors from our result to the results produced by e1071 package.

We find they both get three identical support vectors.(According to page limitation, we didn't show these three identical support vectors here. Please obtain them in the following plots)

```
# w: The normal vector of the dicision hyperplane
# b: The intercept of the dicision hyperplane
# X: Input matrix, each row is an observation vector
# Output: A matrix of support vectors
find_support_vector <- function(w,b,X){
  # The support vectors are the points whose distance from the hyperplane <= 1
  return(x[abs(x %*% t(w) + b) - 1 < 1e-2,])
}
# Our result: find_support_vector(w_HardSVM_primal, b_HardSVM_primal,x)
```

Finally, we use two plots to show our result(the left plot with green decision line) and the result from e1017 package(the right plot with black decision line). The support vectors are the points that are circled. These two polt are almost the same.

```
# Plot the data with dicision line and support vectors that are circled
plot_svm <- function(x,y,w,b,support_vector,color, title =''){
  plot(x,col=ifelse(y>0,"red", "blue"), pch = 19, cex = 0.8, lwd = 2,
     xlab = "X1", ylab = "X2", cex.lab = 1.5)
  legend("bottomright", c("Positive","Negative"),col=c("red", "blue"),
       pch=c(19, 19),text.col=c("red", "blue"), cex = 0.8)
  abline(a= -b/w[1,2], b=-w[1,1]/w[1,2], col= color, lty=1, lwd = 2)
  abline(a= (-b-1)/w[1,2], b=-w[1,1]/w[1,2], col= color, lty=3, lwd = 2)
  abline(a= (-b+1)/w[1,2], b=-w[1,1]/w[1,2], col= color, lty=3, lwd = 2)
  points(support_vector, col="black", cex=2)
  title(main = paste('Decision line ',title,'\n',round(w[1,1],5),' X1',
                     round(w[1,2],5),' X2+',round(b,5),'=0',sep=''))
}
# Compare the result of our code and e1071 package
par(mfrow=c(1,2))
plot_svm(x,y,w_HardSVM_primal,b_HardSVM_primal,find_support_vector
    (w_HardSVM_primal,b_HardSVM_primal,x),'green','of separable case using primal form')
plot_svm(x,y,w,b,x[svm.fit$index, ],'black','of separable case using e1071 package')
```



b) [10 points]

Answer:

The dual form of the linear separable SVM optimization problem is

$$\begin{aligned} &minimize \quad \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{n}\alpha_i \\ &subject\ to \quad \sum_{i=1}^{n}\alpha_i y_i = 0 \quad \alpha_i \geq 0 \quad i = 1, 2, \ldots, n \end{aligned}$$

where α_i is the Lagrange miltiplier which is a $n \times 1$ matrix. Assume there is n observations in X and each observation has p features. Let D be a $n \times n$ positive definite matrix, and d be a n-column vector, A be an $n \times (n+1)$ matrix, and b_0 is an (n+1)-column vector.

Then the dual form of the linear separable SVM optimization problem can be transformed into a form that can be solved by solve.QP if we set $d = [1, \ldots, 1]^T$, $b_0 = [0, \ldots, 0]^T$, $\beta = \alpha$, D is a $n \times n$ matrix where the (i,j)-th element is $y_i y_j x_i^T x_j$ and $A = [Y, I_{n\times n}]^T$ (the first meq constraints are treated as equality constraints).

After we get the solution of alpha. Then we can calculate the normal vector w and the intercept b of the dicision hyperplane by

$$\hat{w} = \sum_{i=1}^{n}\hat{\alpha}_i y_i x_i \quad \hat{b} = -\frac{max_{i:y_i=-1}x_i^T\hat{w} + min_{i:y_i=1}x_i^T\hat{w}}{2}$$

```
# X: Input matrix, each row is an observation vector
# Y: Response variable
# eps: The coefficient of a ridge matrix added to D to make sure it is positive definite
# Value(Output)
# W: The normal vector of the dicision hyperplane
# b0: The intercept of the dicision hyperplane
HardSVM_dual <- function(X,Y,eps){
  n = dim(X)[1]
  p = dim(X)[2]
  # Conduct the parameter for solve.QP as we assumed
  Q = sapply(1:n, function(i) Y[i]*t(X)[,i])
  D = t(Q)%*%Q + eps*diag(n)
  d = matrix(1, nrow=n)
  A = rbind(matrix(Y, nrow=1, ncol=n), diag(nrow=n))
  b0 = rbind( matrix(0, nrow=1, ncol=1) , matrix(0, nrow=n, ncol=1) )
  # Call the QP solver:
  sol = solve.QP(Dmat = D, dvec = d, Amat = t(A), bvec = b0, meq=1)
  # Get the solution of alpha
  alpha = matrix(sol$solution)
  X = as.matrix(X)
  # Calculate the normal vector w and intercept b0 of the dicision hyperplane
  W = rowSums(sapply(1:dim(X)[1], function(i) alpha[i]*Y[i]*X[i,]))
  b_y1 = sapply(which(Y == 1), function(i) X[i,] %*% W)
```

```
  b_y2 = sapply(which(Y == -1), function(i) X[i,] %*% W)
  b0 = -1/2 *(max(b_y2)+min(b_y1))
  return(c(W,b0))
}
# Calculte the SVM solution (decision line) for the linear separable data
HardSVM_dual_solution = HardSVM_dual(x,y,10^-7)
w_HardSVM_dual = matrix(HardSVM_dual_solution[1:p],nrow = 1,
                        dimnames = list(NULL,c('X1','X2')))
b_HardSVM_dual = HardSVM_dual_solution[p+1]
```

Here we compare our solution (the decision line) to the results produced by e1071 package and find they are almost the same, which means the code is correct.

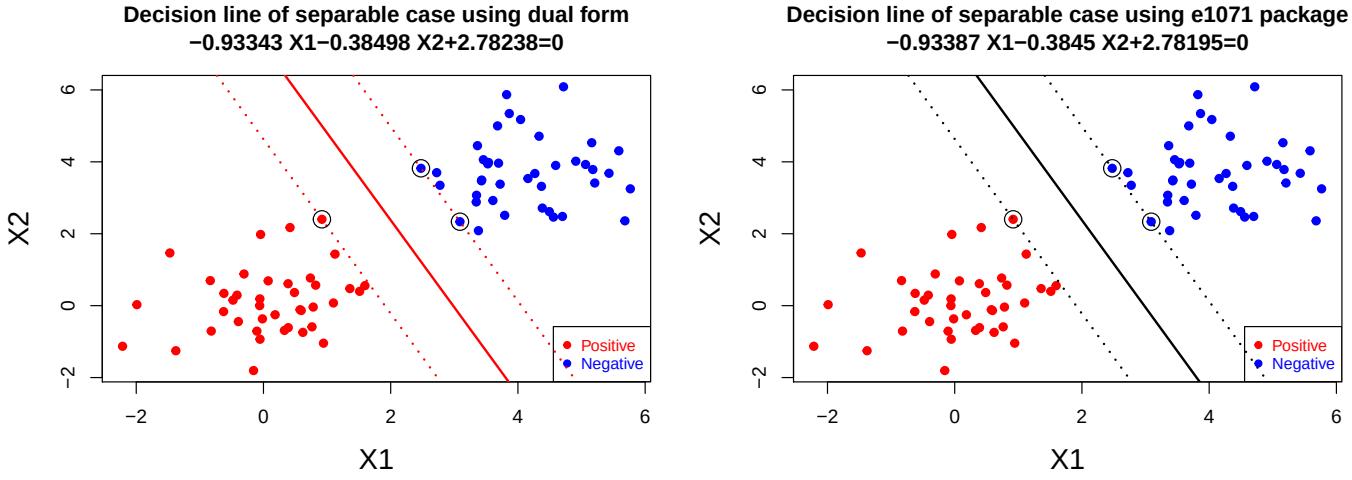
```
HardSVM_solution_compare = rbind(HardSVM_primal_solution,HardSVM_dual_solution,
                                 e1071_HardSVM_solution)
colnames(HardSVM_solution_compare) = c('X1','X2','b0')
HardSVM_solution_compare
```

```
##                                 X1         X2       b0
## HardSVM_primal_solution -0.9334294 -0.3849838 2.782384
## HardSVM_dual_solution   -0.9334294 -0.3849836 2.782384
## e1071_HardSVM_solution  -0.9338740 -0.3844957 2.781953
```

Also, we can compare the support vectors from our result to the results produced by e1071 package. **We find they both get three identical support vectors.**(According to page limitation, we didn't show these three identical support vectors here. Please obtain them in the following plots)

Finally, we use two plots to show our result(the left plot with red desicion line) and the result from e1017 package(the right plot with black decision line). The support vectors are the points that are circled. These two polt are almost the same.

```
# Plot the data with dicision line and support vectors that are circled
par(mfrow=c(1,2))
plot_svm(x,y,w_HardSVM_dual,b_HardSVM_dual,find_support_vector
    (w_HardSVM_dual,b_HardSVM_dual,x),'red','of separable case using dual form')
plot_svm(x,y,w,b,x[svm.fit$index, ],'black','of separable case using e1071 package')
```



c) [20 points]

Answer:

The dual form of the linear nonseparable SVM optimization problem is

$$\begin{aligned} &minimize \quad \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{n}\alpha_i \\ &subject\ to \quad \sum_{i=1}^{n}\alpha_i y_i = 0 \quad \alpha_i \geq 0 \quad -\alpha_i \geq -C \quad i = 1, 2, \ldots, n \end{aligned}$$

where α_i is the Lagrange miltiplier which is a $n \times 1$ matrix. Assume there is n observations in X and each observation has p features. Let D be a $n \times n$ positive definite matrix, and d be a n-column vector, A be an $n \times (2n+1)$ matrix, and b_0 is an (2n+1)-column vector.

Then the dual form of the linear separable SVM optimization problem can be transformed into a form that can be solved by solve.QP if we set $d = [1, \ldots, 1]^T$, $b_0 = [0, 0, \ldots, 0, -C, \ldots, -C]^T$, $\beta = \alpha$, D is a $n \times n$ matrix where the (i,j)-th element is $y_i y_j x_i^T x_j$ and $A = [Y, I_{n\times n}, -I_{n\times n}]^T$ (the first meq constraints are treated as equality constraints).

After we get the solution of alpha. Then we can calculate the normal vector w and the intercept b of the dicision hyperplane by

$$\hat{w} = \sum_{i=1}^{n}\hat{\alpha}_i y_i x_i \quad \hat{b} = mean_j:\ _{0\leq\alpha_j\leq C}(y_j - \sum_{i=1}^{n} y_i\hat{\alpha}_i x_i^T x_j)$$

Here is the code.

```
set.seed(2)
# Create the linear nonseparable data
n = 40 # number of data points for each class
p = 2 # dimension
# Generate the positive and negative examples
xpos = matrix(rnorm(n*p,mean=0,sd=1),n,p)
xneg = matrix(rnorm(n*p,mean=3,sd=1),n,p)
x = rbind(xpos,xneg)
y = matrix(c(rep(1,n),rep(-1,n)))
y_factor = matrix(as.factor(c(rep(1,n),rep(-1,n))))
# X: Input matrix, each row is an observation vector
# Y: Response variable
# C: C is a tuning parameter for "cost"
# eps: The coefficient of a ridge matrix added to D to make sure it is positive definite
# Value(Output)
# W: The normal vector of the dicision hyperplane
# b0: The intercept of the dicision hyperplane
SoftSVM_dual <- function(X,Y,C,eps){
  n = dim(X)[1]
  p = dim(X)[2]
  Q = as.matrix(sapply(1:n, function(i) Y[i]*t(X)[,i]))
```

```
  D = t(Q)%*%Q + eps*diag(n)
  d = matrix(1, nrow=n)
  A = rbind(matrix(Y, nrow=1, ncol=n), diag(nrow=n),-1 * diag(nrow=n))
  b0 = rbind(matrix(0, nrow=1, ncol=1) , matrix(0, nrow=n, ncol=1),
             matrix(-C, nrow=n, ncol=1) )
  # call the QP solver:
  sol = solve.QP(Dmat = D, dvec = d, Amat = t(A), bvec = b0, meq=1)
  # Get the solution of alpha
  alpha = matrix(sol$solution)
  X = as.matrix(X)
  # Calculate the normal vector w and intercept b0 of the dicision hyperplane
  eps_wb = 1e-5
  W = rowSums(sapply(1:dim(X)[1], function(i) alpha[i]*Y[i]*X[i,]))
  b0 = mean(sapply(which(alpha > eps_wb & alpha < C-eps_wb), function(i) Y[i]
                   - X[i,] %*% W))
  return(c(W,b0))
}
# Calculte the SVM solution (decision line) for the linear nonseparable data
# with the tuning parameter for "cost" C is 1
SoftSVM_dual_solution = SoftSVM_dual(x,y,1,1e-10)
w_SoftSVM_dual = matrix(SoftSVM_dual_solution[1:p],nrow = 1,
                        dimnames = list(NULL,c('X1','X2')))
b_SoftSVM_dual = SoftSVM_dual_solution[p+1]
```

Here we compare our solution (the decision line) to the results produced by e1071 package with C = 1, and find they are almost the same, which means the code is correct.

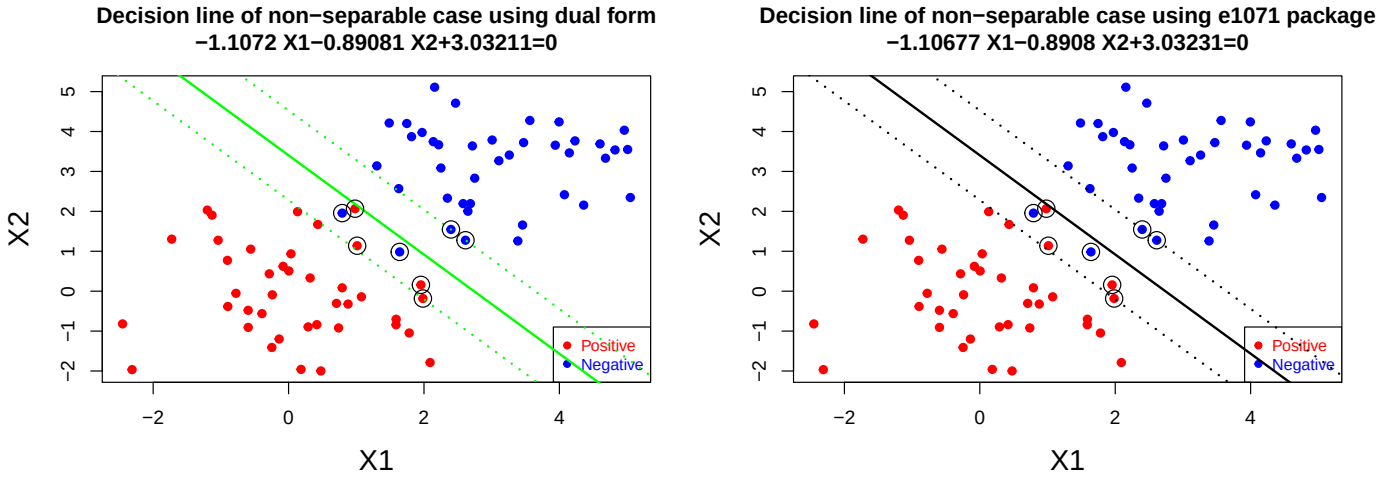
```
# Calculte the SVM solution (decision line) for the linear nonseparable
# data with e1071 package. The cost C is 1
svm.fit = svm(y_factor ~ ., data = data.frame(x, y_factor), type='C-classification',
              kernel='linear',scale=FALSE, cost = 1)
w = t(svm.fit$coefs) %*% svm.fit$SV
b = -svm.fit$rho
e1071_SoftSVM_solution = c(w,b)
SoftSVM_solution_compare = rbind(SoftSVM_dual_solution,e1071_SoftSVM_solution)
colnames(SoftSVM_solution_compare) = c('X1','X2','b0')
SoftSVM_solution_compare
```

```
##                                X1         X2       b0
## SoftSVM_dual_solution  -1.107197 -0.8908079 3.032114
## e1071_SoftSVM_solution -1.106770 -0.8907972 3.032305
```

Also, we can compare the support vectors from our result to the results produced by e1071 package. **We find they both get eight identical support vectors.(Please obtain them in the following plots)**

Finally, we use two plots to show our result(the left plot with green decision line) and the result from e1017 package(the right plot with black decision line). The support vectors are the points that are circled. These two polt are almost the same.

```r
# Plot the data with dicision line and support vectors that are circled
# find_support_vector(w_SoftSVM_dual,b_SoftSVM_dual,x)
par(mfrow=c(1,2))
plot_svm(x,y,w_SoftSVM_dual,b_SoftSVM_dual,find_support_vector
    (w_SoftSVM_dual,b_SoftSVM_dual,x),'green','of non-separable case using dual form')
plot_svm(x,y,w,b,x[svm.fit$index, ],'black','of non-separable case using e1071 package')
```



d) [20 points]

Answer:

To build the SVM model and evaluate how well/bad your model fits, we split the data into two part. 90% is training data, and 10% is test data.

```r
library(ElemStatLearn)
data(SAheart)
# Data clean
# Transform Present/Absent to 1/0 on attribute 'famhist'
SAheart[,'famhist'] = as.vector(SAheart[,'famhist'])
SAheart[SAheart[,'famhist'] == 'Present','famhist'] = 1
SAheart[SAheart[,'famhist'] == 'Absent','famhist'] = 0
SAheart[,'famhist'] = as.numeric(SAheart[,'famhist'])
# Transform 1/0 to 1/-1 on attribute 'chd'
SAheart[SAheart[,'chd'] == 0,'chd'] = -1
# Split the data. 90% is training data and the other are test data
set.seed(1)
train_index = sample(dim(SAheart)[1])[1:floor(dim(SAheart)[1] * 0.9)]
train_x = SAheart[train_index,-10]
train_y = SAheart[train_index,10]
test_x = SAheart[-train_index,-10]
test_y = SAheart[-train_index,10]
```

Here we built the function svm_acc to calculte the predict accurracy and the function

svm_cv to perform the cross validation.

```
# X: Input matrix, each row is an observation vector
# Y: Response variable
# w: The normal vector of the dicision hyperplane
# b: The intercept of the dicision hyperplane
# Output(acc): The predict accurracy of the data based on the hyperplane w*x+b = 0
svm_acc <- function(X,Y,w,b){
  X = as.matrix(X)
  predict = sign(X %*% t(w) + b)
  acc = sum(predict == Y) / length(Y)
  return(acc)
}

# X: Input matrix, each row is an observation vector
# Y: Response variable
# C: a sequence of tuning parameters for "cost"
# nfold: the fold of cross validation
# Output(result): A data frame of the sequence of tuning parameters
#                 the corresponding cross validation accurracy
svm_cv <- function(X,Y,C,nfold){
  # Reorder the data
  random_index = sample(dim(X)[1])
  X = X[random_index,]
  Y = Y[random_index]
  cv_acc_dual = rep(NA,length(C))
  cv_acc_e1071 = rep(NA,length(C))
  p = dim(X)[2]
  for(i in 1:length(C)){
    acc = rep(NA,nfold)
    acc_e1071 = rep(NA,nfold)
    size = floor(dim(X)[1]/nfold)
    for(j in 1:nfold){
      # Split the data into train and validation part
      index = ((j-1)*size+1):(j*size)
      train_x = X[-index,]
      train_y = Y[-index]
      train_y_factor = matrix(as.factor(train_y))
      validation_x = X[index,]
      validation_y = Y[index]
      # Fit the model with our method
      SoftSVM_dual_solution = SoftSVM_dual(train_x,train_y,C[i],10^-7)
      w_SoftSVM_dual = matrix(SoftSVM_dual_solution[1:p],nrow = 1)
      b_SoftSVM_dual = SoftSVM_dual_solution[p+1]
      # Calculate the accurracy on the validation data with our model
      acc[j] =  svm_acc(validation_x,validation_y,w_SoftSVM_dual,b_SoftSVM_dual)
      # Fit the model with e1071 package
      svm.fit <- svm(train_y_factor ~ ., data = data.frame(train_x, train_y_factor),
                     type='C-classification', kernel='linear',scale=FALSE,
```

```
                    cost = C[i])
      # Calculate the accurracy on the validation data with e1071 package
      acc_e1071[j] =  sum(as.numeric(as.vector(predict(svm.fit,validation_x))) ==
                          validation_y) / length(validation_y)
    }
    # Calculate the cross validation accurracy
    cv_acc_dual[i] = mean(acc)
    cv_acc_e1071[i] = mean(acc_e1071)
  }
  result = rbind(C,cv_acc_dual,cv_acc_e1071)
  return(result)
}
```

Here we use a sequence of tuning parameters $(0.001, 0.01, 0.1, 0.5, 1, 2, 10)$ and we will do 5-fold cross validation on the training data to select the tuning parameter.

```
set.seed(0)
round(svm_cv(train_x,train_y,c(0.001,0.01,0.1,0.5,1,2,10),5),5)
```

```
##                 [,1]    [,2]    [,3]    [,4]    [,5]    [,6]     [,7]
## C            0.00100 0.01000 0.10000 0.50000 1.00000 2.00000 10.00000
## cv_acc_dual  0.67711 0.69157 0.72771 0.73976 0.73735 0.73735  0.73735
## cv_acc_e1071 0.67711 0.69639 0.73012 0.74217 0.73494 0.73494  0.73735
```

Thus, we choose the best tuning parameter with highest accurracy $C = 0.5$. Then we use svm model with this tuning parameter to fit all the train data again.

```
p = dim(train_x)[2]
# Fit the model with our method using the best tuning parameter C=0.5
SoftSVM_dual_solution = SoftSVM_dual(train_x,train_y,0.5,10^-7)
w_SoftSVM_dual = matrix(SoftSVM_dual_solution[1:p],nrow = 1)
b_SoftSVM_dual = SoftSVM_dual_solution[p+1]
# Fit the model with e1071 package using the best tuning parameter C=0.5
train_y_factor = matrix(as.factor(train_y))
svm.fit <- svm(train_y_factor ~ ., data = data.frame(train_x, train_y_factor),
               type='C-classification', kernel='linear',scale=FALSE, cost = 0.5)
# Calculate the accurracy on the train and test data with our model
train_error_SoftSVM_dual = svm_acc(train_x,train_y,w_SoftSVM_dual,b_SoftSVM_dual)
test_error_SoftSVM_dual = svm_acc(test_x,test_y,w_SoftSVM_dual,b_SoftSVM_dual)
# Calculate the accurracy on the train and test data with e1071 package
train_error_e1071= sum(as.numeric(as.vector(predict(svm.fit,train_x))) ==
                           train_y) / length(train_y)
test_error_e1071= sum(as.numeric(as.vector(predict(svm.fit,test_x))) ==
                           test_y) / length(test_y)
# Show the result
result = data.frame(c(train_error_SoftSVM_dual,train_error_e1071),
                    c(test_error_SoftSVM_dual,test_error_e1071))
colnames(result) = c('Train_error','Test_error')
rownames(result) = c('Our method','e1071 pachage')
result
```

```
##                Train_error Test_error
## Our method       0.7325301  0.7234043
## e1071 pachage    0.7325301  0.7234043
```

So our predict result are the same as e1071 package.

Question 2 [20 points]

Answer:

Assume $\xi_1 < \xi_2 < \cdots < \xi_K$. Since we have

$$f(X) = \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3 \quad (1)$$

Then we have

$$when \quad X \le \xi_1 \quad f^{'}(X) = \beta_1 + 2\beta_2 X + 3\beta_3 X^2 \quad (2)$$

$$f^{''}(X) = 2\beta_2 + 6\beta_3 X \quad (3)$$

$$when \quad X \ge \xi_K \quad f^{'}(X) = \beta_1 + 2\beta_2 X + 3\beta_3 X^2 + \sum_{k=1}^{K} 3\theta_k (X - \xi_k)^2 \quad (4)$$

$$f^{''}(X) = 2\beta_2 X + 6\beta_3 X^2 + \sum_{k=1}^{K} 6\theta_k (X - \xi_k) \quad (5)$$

Since we have the following constrains $f(X)$ is linear for$X \le \xi_1$ and $X \ge \xi_K$, we have

$$f^{''}(X) \equiv 0 \quad (6)$$

whatever X and $\xi_k, k = 1, \ldots, K$ is.

According to (3) and (6), we can get

$$\beta_2 = 0 \quad \beta_3 = 0 \quad (7)$$

According to (5), (6) and (7), we can get

$$\sum_{k=1}^{K} \theta_k (X - \xi_k) = (\sum_{k=1}^{K} \theta_k) X - \sum_{k=1}^{K} \theta_k \xi_k \equiv 0$$

$$\Longrightarrow \sum_{k=1}^{K} \theta_k = 0 \quad (8)$$

$$\sum_{k=1}^{K} \theta_k \xi_k = 0 \quad (9)$$

Finally, we need to prove the fomula (1) can be written as

$$f(X) = \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \alpha_k (d_k(X) - d_{K-1}(X)) \quad (10)$$

where

$$d_k(X) = \frac{(X-\xi_k)_+^3 - (X-\xi_K)_+^3}{\xi_K - \xi_k} \quad \text{and} \quad \alpha_k = \theta_k(\xi_K - \xi_k)$$

According to (7), fomula (10) can be written as

$$\begin{aligned} f(X) =& \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \sum_{k=1}^{K-2} \alpha_k (d_k(X) - d_{K-1}(X)) \\ =& \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K-2} \alpha_k d_k(X) - \sum_{k=1}^{K-2} \alpha_k d_{K-1}(X) \\ =& \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K-2} \theta_k [(X-\xi_k)_+^3 - (X-\xi_K)_+^3] - \sum_{k=1}^{K-2} \theta_k (\xi_K - \xi_k) \frac{(X-\xi_{K-1})_+^3 - (X-\xi_K)_+^3}{\xi_K - \xi_{K-1}} \\ =& \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K-2} \theta_k (X-\xi_k)^3 + [-\sum_{k=1}^{K-2} \theta_k \frac{\xi_K - \xi_k}{\xi_K - \xi_{K-1}}](X-\xi_{K-1})_+^3 \\ &+ [-\sum_{k=1}^{K-2} \theta_k + \sum_{k=1}^{K-2} \theta_k \frac{\xi_K - \xi_k}{\xi_K - \xi_{K-1}}](X-\xi_K)_+^3 \quad (11) \end{aligned}$$

Now let's foucus on these two item.

$$A = -\sum_{k=1}^{K-2} \theta_k \frac{\xi_K - \xi_k}{\xi_K - \xi_{K-1}} \quad (12)$$

$$B = -\sum_{k=1}^{K-2} \theta_k + \sum_{k=1}^{K-2} \theta_k \frac{\xi_K - \xi_k}{\xi_K - \xi_{K-1}} \quad (13)$$

According to (8) and (9), we have

$$\sum_{k=1}^{K-2} \theta_k = -\theta_{K-1} - \theta_K \quad (14)$$

$$\sum_{k=1}^{K-2} \theta_k \xi_k = -\theta_{K-1}\xi_{K-1} - \theta_K \xi_K \quad (15)$$

Thus according to (14) and (15), formula (12) can be written as

$$\begin{aligned} A =& -\sum_{k=1}^{K-2} \theta_k \frac{\xi_K - \xi_k}{\xi_K - \xi_{K-1}} \\ =& (-\sum_{k=1}^{K-2} \theta_k) \frac{\xi_K}{\xi_K - \xi_{K-1}} + (\sum_{k=1}^{K-2} \theta_k \xi_k) \frac{1}{\xi_K - \xi_{K-1}} \\ =& \frac{\xi_K(\theta_{K-1} + \theta_K)}{\xi_K - \xi_{K-1}} - \frac{\theta_{K-1}\xi_{K-1} + \theta_K \xi_K}{\xi_K - \xi_{K-1}} \\ =& \frac{\theta_{K-1}(\xi_K - \xi_{K-1})}{\xi_K - \xi_{K-1}} \\ =& \theta_{K-1} \quad (16) \end{aligned}$$

And according to (12), (13), (14) and (16),we have

$$\begin{aligned} B =& -\sum_{k=1}^{K-2} \theta_k - A \\ =& \theta_{K-1} + \theta_K - \theta_{K-1} \\ =& \theta_K \quad (17) \end{aligned}$$

According to (11), (12), (16) and (17), formula (11) can be written as

$$
\begin{aligned}
f(X) &= \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K-2} \theta_k (X-\xi_k)^3 + A(X-\xi_{K-1})_+^3 + B(X-\xi_K)_+^3 \\
&= \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K-2} \theta_k (X-\xi_k)^3 + \theta_{K-1}(X-\xi_{K-1})_+^3 + \theta_K (X-\xi_K)_+^3 \\
&= \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K} \theta_k (X-\xi_k)^3 \quad (18)
\end{aligned}
$$

Thus, we have proved formula (10) can be written as the form of formual (1)(or (18))

Question 3 [20 points]

Question: Can we know the Mental Health Issues of a shooter accorrding to his/her own feature(like Race, Gender) and the features of the mass shooting event(like Location, Date, Fatalities, Injured, etc)? In other words, is there a correlation between the Mental Health Issues of a shooter and the features of him/her and the mass shooting event?

We write a function `mass.shootings.data_clean` do the data clean. First, since we use the SVM model, we can't handle the text information. Thus, we will remove the "Title" and "Summary" information. Second, we combine some levels in "Mental Health Issues", "Race" and "Gender" features. Third, we split the "Date" feature into "Day", "Month" and "Year" features. Finally, we remove the "Location" feature since we think the level of "Location" is too much and it can be replaced by the "Latitude" and "Longitude" information. **Due to the page limitation, here we will not show the functions we write for Question 3. You can find them in `mass.shootings.data.R` file**

```
mass.shootings = read.csv("Mass Shootings Dataset Ver 2.csv", stringsAsFactors = FALSE, head
source("mass.shootings.data.R")
# data clean
data = mass.shootings.data_clean(mass.shootings)
colnames(data)
```

```
##  [1] "Fatalities"           "Injured"              "Total.victims"
##  [4] "Mental.Health.Issues" "Race"                 "Gender"
##  [7] "Latitude"             "Longitude"            "Year"
## [10] "Month"                "Day"
```

```
# The summary of cleanned data
# summary(data)
```

According to our quesiton, we will choose the features we will use and remove the obvservation with missing value. Here we write a function `mass.shootings.missing_value` deal with the missing value. Then we choose "Race","Gender" as the shooters' feature and "Total.victims","Latitude","Longitude" as the events' feature. Here are three things to mentioned. First, we only use the "Total.victims" information since it is the sum of "Fatalities" and "Injured" in most of the time. Second, we don't use the Date information. The main reason is the event date is sparse in the 3D space of Year-Month-Day. We don't have enough data, thus it will be overfitting if we use date information.(Maybe you can try only use the month information.)

```
# Remove all unknown and na data
data2 = na.omit(mass.shootings.missing_value(data))
# Choose the feature according to our question
#feature = c("Mental.Health.Issues","Fatalities","Injured","Race","Gender","Latitude","Lon
feature = c("Mental.Health.Issues","Total.victims","Race","Gender","Latitude","Longitude")
#feature = c("Mental.Health.Issues","Total.victims","Race","Gender")
data2 = data2[,feature]
```

To build the SVM model and evaluate how well/bad your model fits, we split the data into two part. 80% is training data, and 20% is test data.

```
# Split the data. 90% is training data and the other are test data
set.seed(0)
train_index = sample(dim(data2)[1])[1:floor(dim(data2)[1] * 0.8)]
train_x = data2[train_index,-1]
train_y = data2[train_index,1]
test_x = data2[-train_index,-1]
test_y = data2[-train_index,1]
```

Here we will do a 3-fold cross validation over training data to select the best C value and the best kernel. We will try four kernels 'linear','polynomial','sigmoid','radial'. Here we write a funciton to do the cross-validation. You can find it in `mass.shootings.data.R` file.

```
# Function SVM_CV
# Usage: SVM_CV(X,Y,C,kernel,nfold)
# X: Input matrix, each row is an observation vector
# Y: Response variable
# C: a sequence of tuning parameters for "cost"
# kernel: a sequence of kernel
# nfold: the fold of cross validation
# Output(result): A matrix of the sequence of tuning parameters
#                 the corresponding cross validation accurracy
round(SVM_CV(train_x,train_y,c(0.1,1,5,10,20,50,100),
            c('linear','polynomial','sigmoid','radial'),3),5)
```

```
##                0.1       1       5      10      20      50     100
## linear     0.60417 0.62500 0.61806 0.62500 0.62500 0.62500 0.61806
## polynomial 0.53472 0.52083 0.49306 0.50000 0.48611 0.50694 0.52778
## sigmoid    0.52778 0.52778 0.52778 0.52778 0.52778 0.52778 0.52778
## radial     0.52778 0.59028 0.60417 0.60417 0.61111 0.61806 0.61806
```

Thus, when we use the best value of $C = 10$ and the "linear" kernel. We use the parameter to fit the model with all training data and evaluate on the test data.

```
# Fit the model
svm.fit <- svm(train_y ~ ., data = data.frame(train_x,train_y),
               type='C-classification', kernel='linear', cost = 10)
# Make prediction
predict_y_with_coor = predict(svm.fit,test_x,decision.values=TRUE)
# Calculate the accuracy
```

```
train_acc = sum(predict(svm.fit,train_x) == train_y) / dim(train_x)[1]
test_acc = sum(predict(svm.fit,test_x) == test_y) / dim(test_x)[1]
cbind(train_acc,test_acc)
```

```
##      train_acc  test_acc
## [1,] 0.6438356 0.7297297
```

Also, we write a function `confusion_matrix` to calculate the confusion matrix, specificity and sensitivity.

```
# Function confusion_matrix
# Usage: confusion_matrix(y,svm_pred_y)
# y: The truth y value
# svm_pred_y: Predict value of y
# Output(result): A list of confusion matrix,specificity,sensitivity
result = confusion_matrix(test_y,predict_y_with_coor)
result$result
```

```
##                               Condition positive Condition negative
## Predicted condition positive                  16                  8
## Predicted condition negative                   2                 11
```

```
c(result$specificity,result$sensitivity)
```

```
## [1] 0.5789474 0.8888889
```

We also fit a model that do not use "Latitude","Longitude" information. This analysis part will not be showed here accorrding to the page limitaion. The best C=50 and the best kernel is "polynomial". You can find it in the Rmd file.

Finally, we built a ROC curve and calculate the AUC value that is 0.7515.

```
library("pROC")
svm.ROC = roc(response = test_y,
              predictor = as.vector(attr(predict_y_with_coor, "decision.values")),
              levels = levels(test_y))
# Plot the roc curve. Here we do not show it due to the page limitation.
# plot(svm.ROC,col = "green")
svm.ROC$auc
```

```
## Area under the curve: 0.7515
```

Conclusion: **We can know the Mental Health Issues of a shooter accorrding to his/her own feature(like Race, Gender) and the features of the mass shooting event(like Location, Date, Fatalities, Injured, etc) with the accuracy 0.7297297.**

According to several metric like accuracy = 0.7297297, specificity = 0.5789474, sensitivity = 0.8888889 and AUC value = 0.7515, we can say that **there is a weak correlation between the Mental Health Issues of a shooter and the features of him/her and the mass shooting event.** We can predict well if the shooter really has a Mental Health Issues. However, if the shooter doesn't, we do pretty bad. If we have more information about the shooter, or about the event such as extract some information from "Title" and "Summary"(do text mining), the model may predict better.