STAT 542 HW 3

Xiruo Li (xiruoli2)

Question 1

(a)

```
print(svm_linear)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.03163387

print(svm_radial)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##    0.5   10
##
## - best performance: 0.03315444

print(svm_polynomial)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##    0.5 0.01
##
## - best performance: 0.02314252
```

I choose three kernals: linear, radial, polynomial. I tune the cost in 10^(-2:2) for all kernels and gamma in (0.5,1,2) for radial/polynomial kernel. Then, I got the best model of each kernel by cross validation. Compared with these three model, kernel=polynomial with gamma=0.5, cost=0.01 has the lowest error,0.02544425.

(b)

```
##              Predicted Class
## Actual Class   4   9
##             4 187  13
##             9   4 173
```

(c)

(1):Linear kernal has the lowest number of parameters and computational cost. It is suitable when there is a large number of features.However, it perform badly for the non-linear seperable case. For Polynomial kernel, although it has the most number of parameters and computational cost, it is suitable for nonlinear seperable case. For Radial kernel, it has fewer parameters than polynomial kernel and it is useful for nonlinear seperable case.

(2):Cross validation has higher computational cost since, it train and test k times for k-fold cross validation. Also, it randomly divide data into train and test set, therefore, the result may have a high variance.

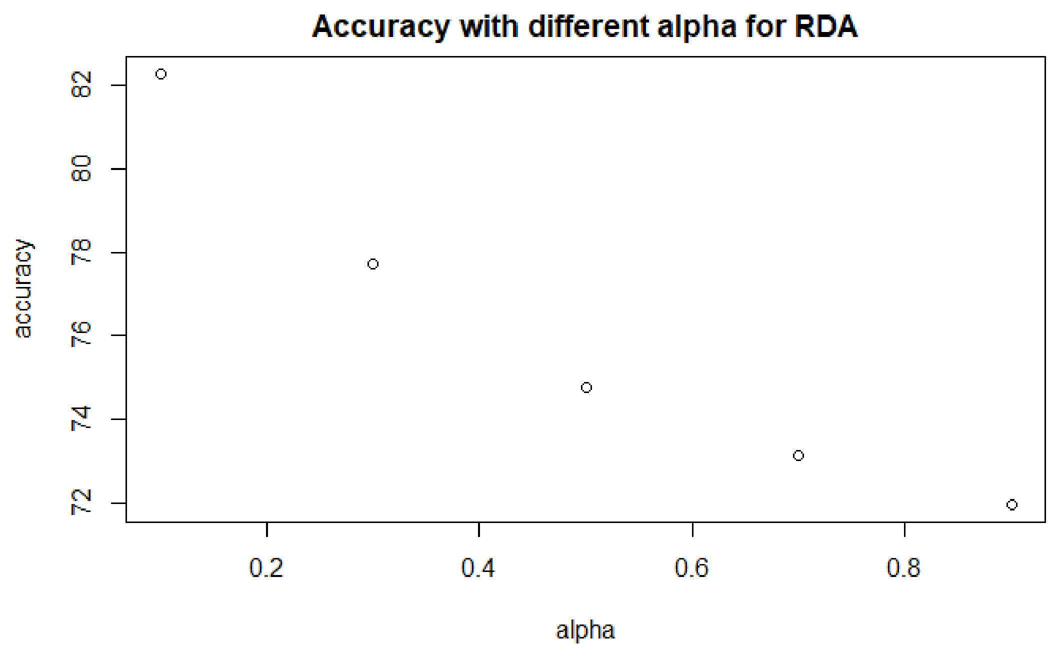
Question2

(a)

```
## [1] "Accuracy for LDA is: 82.56 %"
```

(b)

```
[1] "Accuracy for QDA is: 82.28 % ,when alpha=  0.1"

[1] "Accuracy for QDA is: 77.73 % ,when alpha=  0.3"

[1] "Accuracy for QDA is: 74.75 % ,when alpha=  0.5"

[1] "Accuracy for QDA is: 73.12 % ,when alpha=  0.7"

[1] "Accuracy for QDA is: 71.96 % ,when alpha=  0.9"
```



(c)

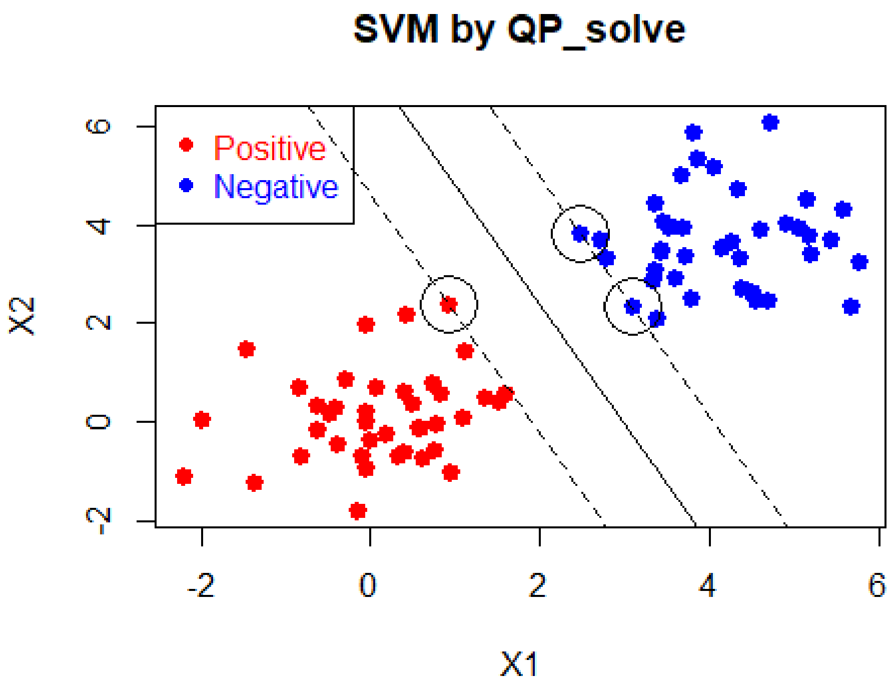
For LDA, it has an assumption that the covariance are same for all of class. This method is simple since it only use one pooled covariance matrix. But in the real cases, different class may have different covariance matrix. In such a case, the accuracy for LDA predication is low. For QDA, it abandon the assumption in LDA and use different covariance matrix for each class. Thus, it has better performance than LDA when covariance matrix is different for each class. But, the problem is that it has more parameters since it has k covariance matrix when there are k classes. Besides, since only the data in the kth class can be used to estimated kth covariance, when there is not enough data, it may has increased variance in the estimation of the optimal boundaries. For RDA, it's a compromise between LDA and QDA. It can be viewed as a more general method. When alpha is close to 0, it is close to LDA. Otherwise, it close to RDA fitting. When alpha increase, the error of training data will keep decreasing, since the model become flexible, like QDA. But the error for the test data first decrease, then increase, which shows a trade off between model complexity and the fitting of training data. We can get the best performance by tuning the alpha in RDA.

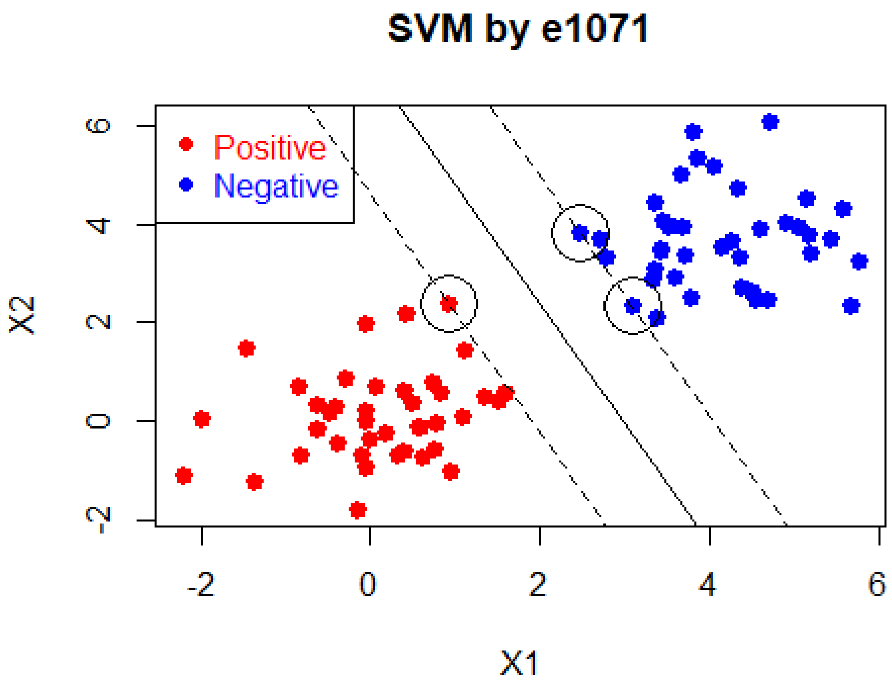
Question 3

(a)

```
#D,d,A,b0
eps=10^(-5)
Dmat=(y%*%t(y))*(x%*%t(x))+eps*diag(nrow(x))
dvec=matrix(1,2*n)
```

```
Amat=cbind(matrix(y,2*n,1),diag(2*n))
bvec=matrix(0, 2*n+1, 1)
```



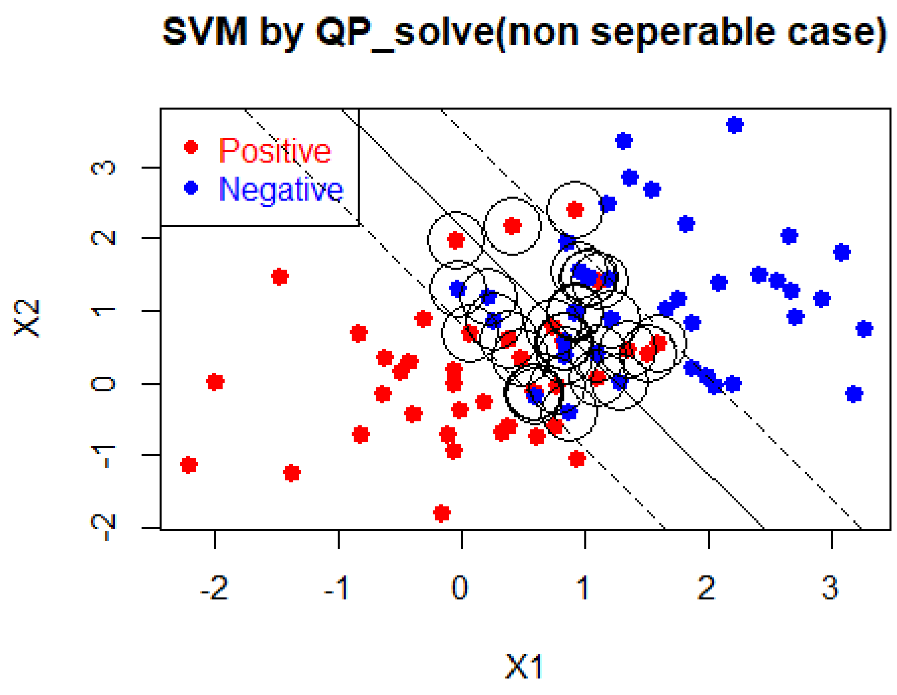


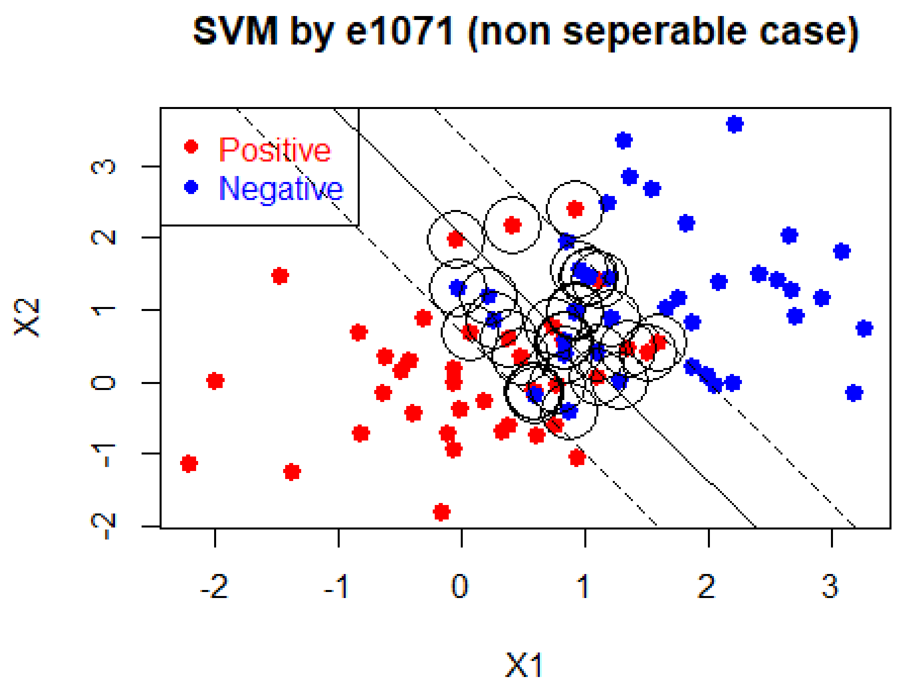
```
#comparison

##          QP_solve      e1071
## beta1  -0.933426 -0.9338740
## beta2  -0.384982 -0.3844957
## beata0  2.782373  2.7819531
```

Both plots by QP_solve and e1071 show the support vectors and decision line. By comparison, beta and beta0 are similar for two methods.

(b)

```
#c,D,d,A,b0
c=1
Dmat=(y %*% t(y) )* (x %*% t(x)) + eps*diag(nrow(x))
dvec=matrix(1, nrow=n*2)
Amat=cbind(matrix(y,2*n,1),diag(2*n),diag(-1,2*n))
bvec<-rbind(matrix(0, 2*n+1, 1), matrix(-c, 2*n, 1))
```

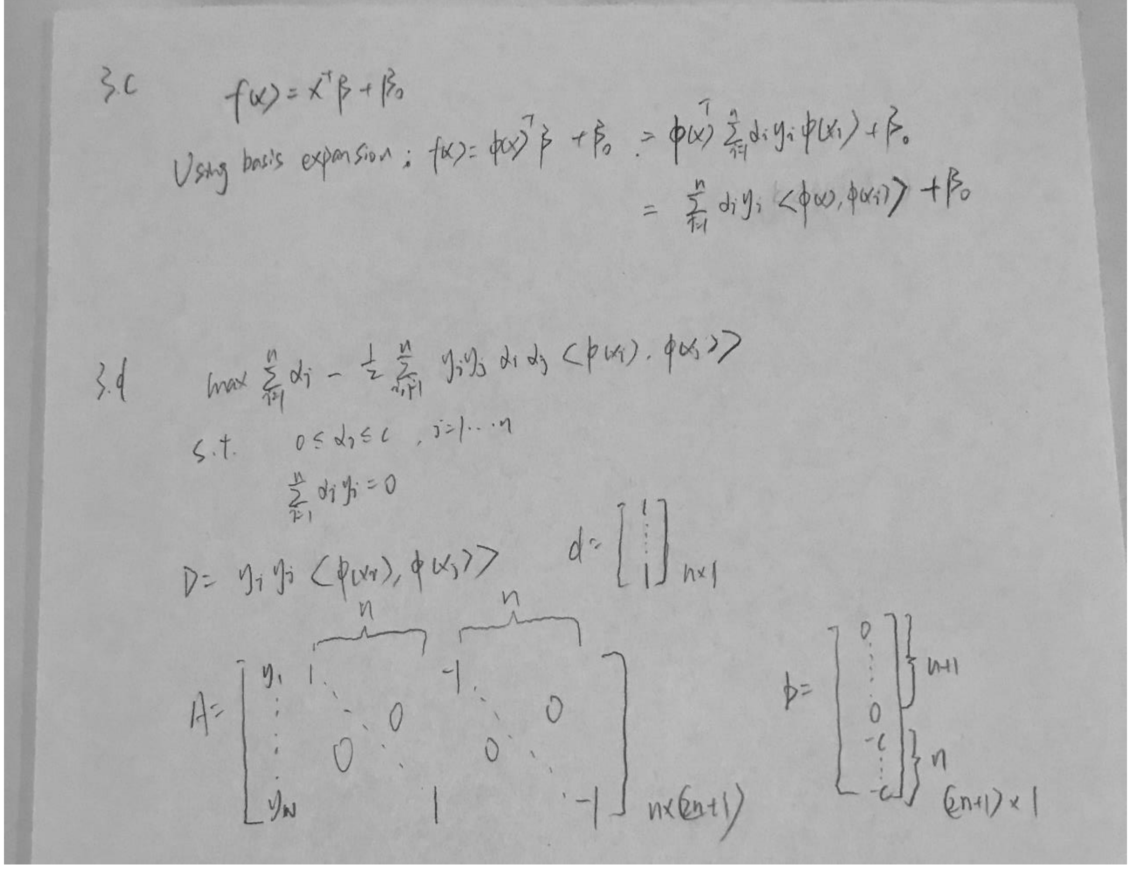




```
#comparison

##           QP_solve      e1071
## beta1  -1.2556610 -1.2565658
## beta2  -0.7375098 -0.7374771
## beata0  1.5922924  1.5245487
```

Both plots by QP_solve and e1071 show the support vectors and decision line. By comparison, beta and beta0 are similar for two methods.

(c)(d)



3.c $f(x) = x^T\beta + \beta_0$

Using basis expansion: $f(x) = \phi(x)^T\beta + \beta_0 = \phi(x)^T \sum_{i=1}^{n} \alpha_i y_i \phi(x_i) + \beta_0$

$$= \sum_{i=1}^{n} \alpha_i y_i \langle \phi(x), \phi(x_i) \rangle + \beta_0$$

3.d $\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \phi(x_i), \phi(x_j) \rangle$

s.t. $0 \le \alpha_i \le c$, $i = 1 \dots n$

$\sum_{i=1}^{n} \alpha_i y_i = 0$

$D = y_i y_j \langle \phi(x_i), \phi(x_j) \rangle$ $\quad d = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{n\times 1}$

$$A = \begin{bmatrix} y_1 & \overbrace{\begin{matrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{matrix}}^{n} & \overbrace{\begin{matrix} -1 & & 0 \\ & \ddots & \\ 0 & & -1 \end{matrix}}^{n} \\ \vdots & & \\ y_N & & \end{bmatrix}_{n\times(2n+1)} \qquad b = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -c \\ \vdots \\ -c \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix} 0 \\ \vdots \\ 0 \end{matrix}}\right\} n+1 \\ \left.\vphantom{\begin{matrix} -c \\ \vdots \\ -c \end{matrix}}\right\} n \end{matrix}_{(2n+1)\times 1}$$