6703hw9

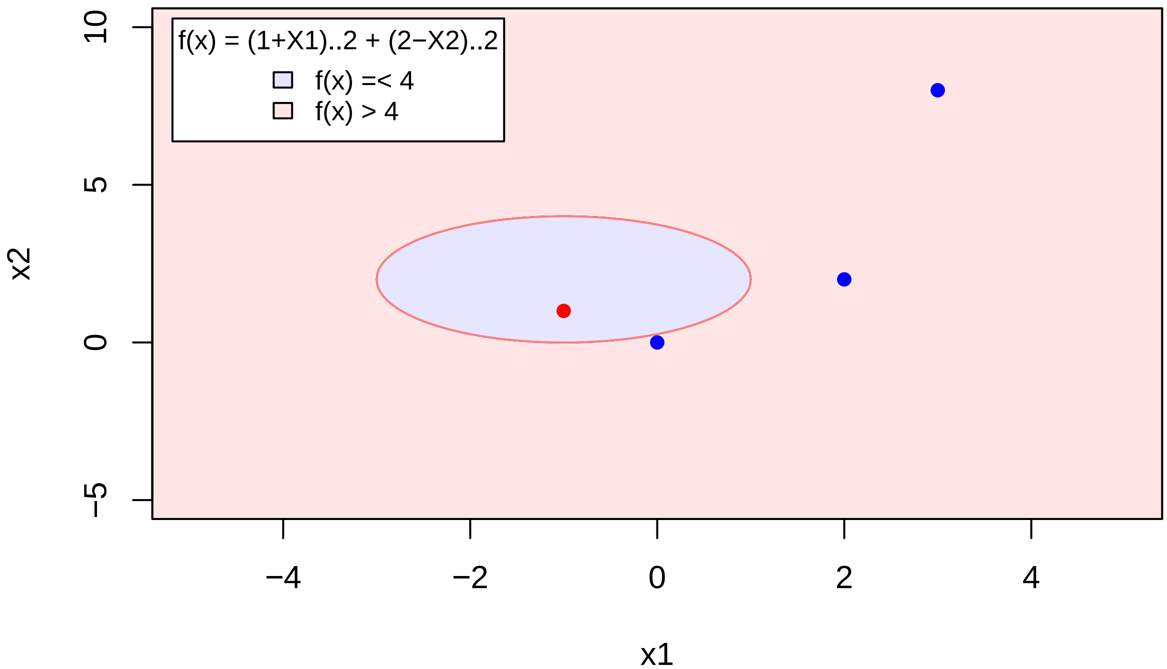
Yu Chen & Christopher Marais

2022-11-29

Chap 9 Exercise 2

(a)-(c)

```
x1 <- seq(-3,1,0.01)
x2 <- 2 - sqrt(4-(1+x1)^2)
x2.2 <- 2 + sqrt(4-(1+x1)^2)
plot(x1,x2,type="l",ylim=c(-5,10),xlim=c(-5,5))
lines(x1,x2.2)
#abline(h=-2)
#abline(h=4)
#abline(v=-3)
#abline(v=1)
# define colors
blue = rgb(0.9,0.9,1,1)
darkred = rgb(1,0.5,0.5,1)
red = rgb(1,0,0,0.1)
# fill color
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = red)
polygon(c(x1, rev(x1)),c(x2, rev(x2.2)),col = blue,border = darkred)
# add points
points(c(0,-1,2,3),c(0,1,2,8),pch=16,col=c("blue","red","blue","blue"))
# legend
suppressWarnings(legend("topleft",inset=.02,title = "f(x) = (1+X1)^2 + (2-X2)^2",
       c("f(x) =< 4","f(x) > 4"),
       fill=c(blue, red),
       cex=0.8,
       bg="white"))
```



The points which are out of the circle will satisfy the equation

$$(1 + X_1)^2 + (2 - X_2)^2 > 4$$

The points which are in the circle (or on the circle) will satisfy the equation

$$(1 + X_1)^2 + (2 - X_2)^2 \leq 4$$

Point (-1,1) is red. Other points (0,0) (2,2) (3,8) are blue.
d). Expand the equation, then

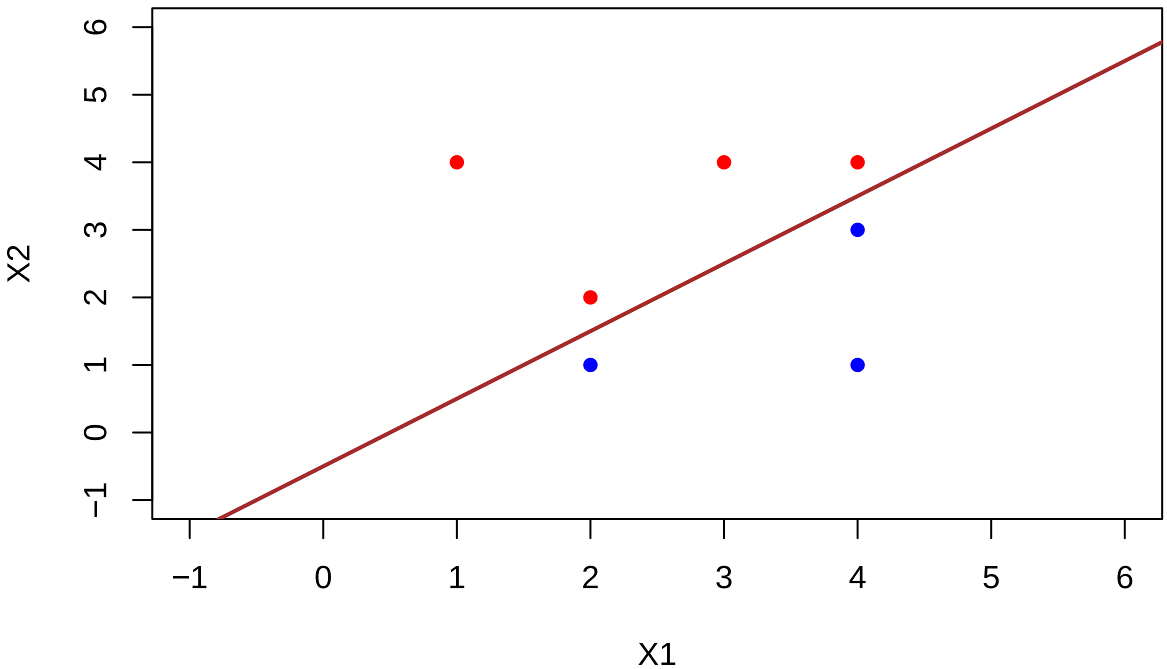
$$(1 + X_1)^2 + (2 - X_2)^2 > 4$$
$$X_1^2 + 2X_1 + X_2^2 - 4X_2 + 1 > 0$$

So we can see the decision boundary is linear in terms of X_1, X_1^2, X_2 and X_2^2.

Chap 9 Exercise 3

a).

```
plot(-1:6,-1:6,type="n",xlab='X1', ylab='X2')
points(c(3,2,4,1),c(4,2,4,4), col="red",pch=16)
points(c(2,4,4),c(1,3,1), col="blue",pch=16)
abline(-0.5,1,col='brown',lwd=2)
```



b). hyperplane:

$$-0.5 + X_1 - X_2 = 0$$

c).

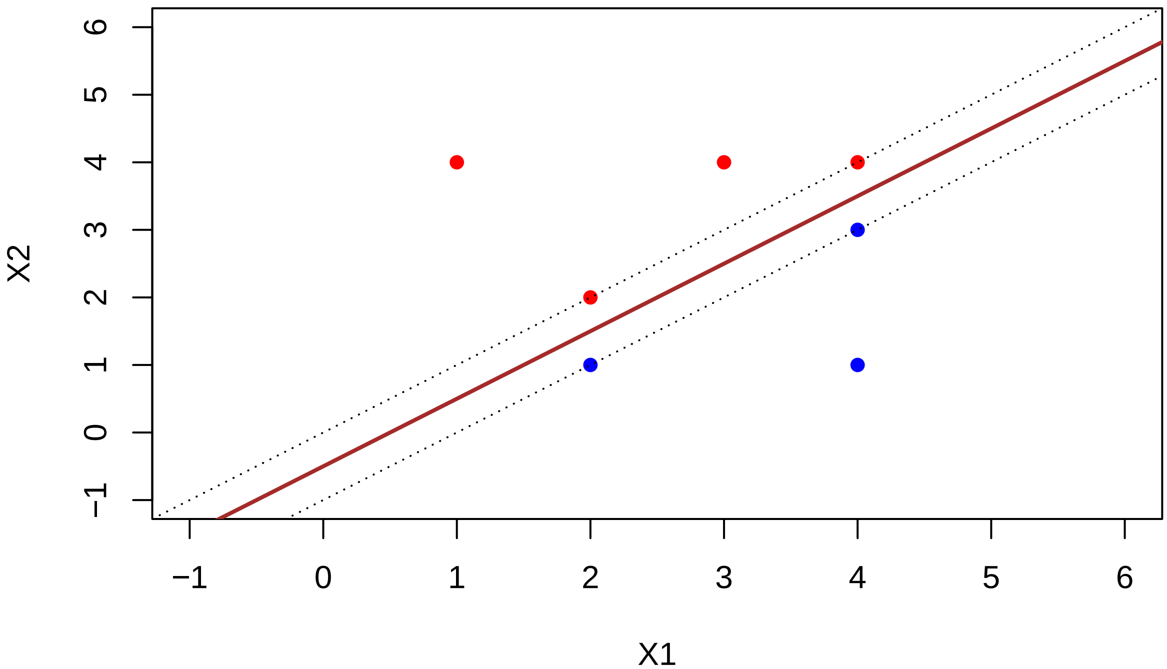
Classify to red if

$$0.5 - X_1 + X_2 > 0$$

and classify to Blue otherwise.

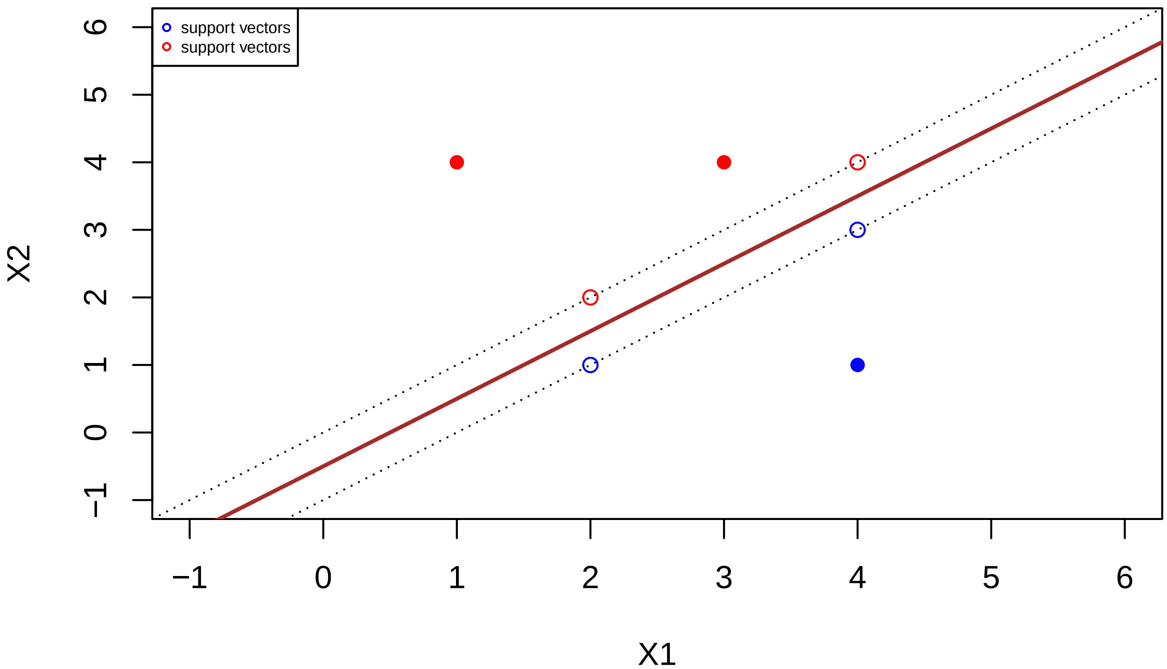
d).

```
plot(-1:6,-1:6,type="n",xlab='X1', ylab='X2')
points(c(3,2,4,1),c(4,2,4,4), col="red",pch=16)
points(c(2,4,4),c(1,3,1), col="blue",pch=16)
abline(-0.5,1,col='brown',lwd=2)
abline(-1, 1, col='black',lty='dotted')
abline(0, 1, col='black',lty='dotted')
```



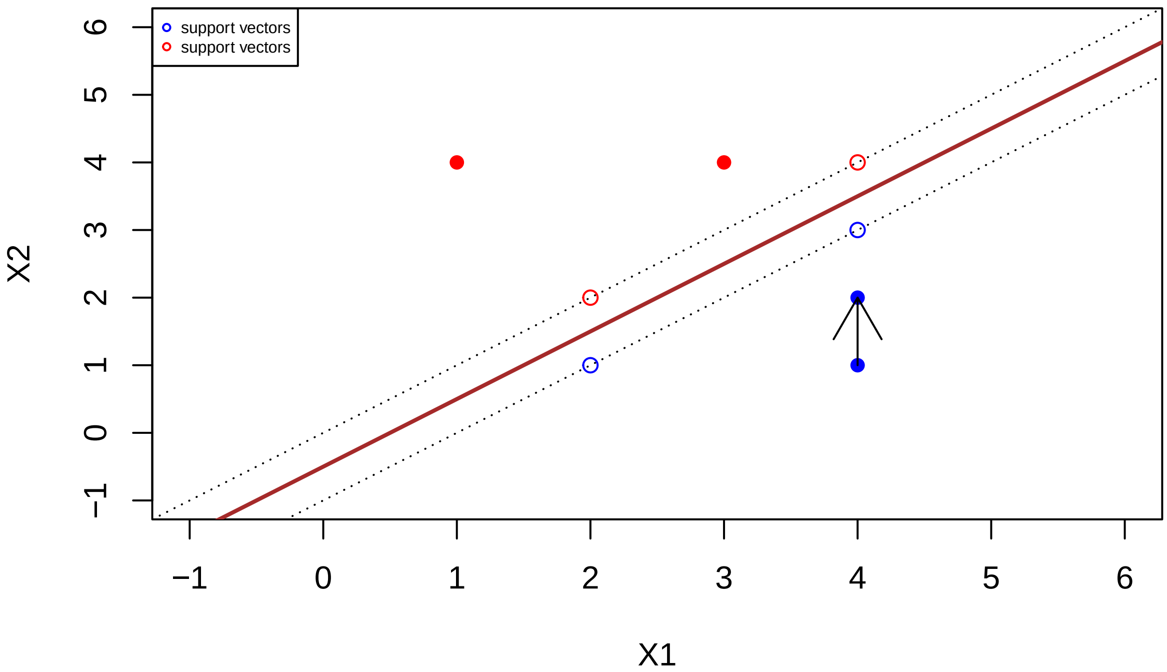
(e)

```
plot(-1:6,-1:6,type="n",xlab='X1', ylab='X2')
points(c(3,1),c(4,4), col="red",pch=16)
points(c(4),c(1), col="blue",pch=16)
points(c(2,2,4,4),c(1,2,3,4),col=c("blue","red","blue","red",pch=1))
abline(-0.5,1,col='brown',lwd=2)
abline(-1, 1, col='black',lty='dotted')
abline(0, 1, col='black',lty='dotted')
legend("topleft",pch=1,
       legend = c("support vectors","support vectors"),
       col=c("blue","red"),
       cex=0.5)
```



(f)

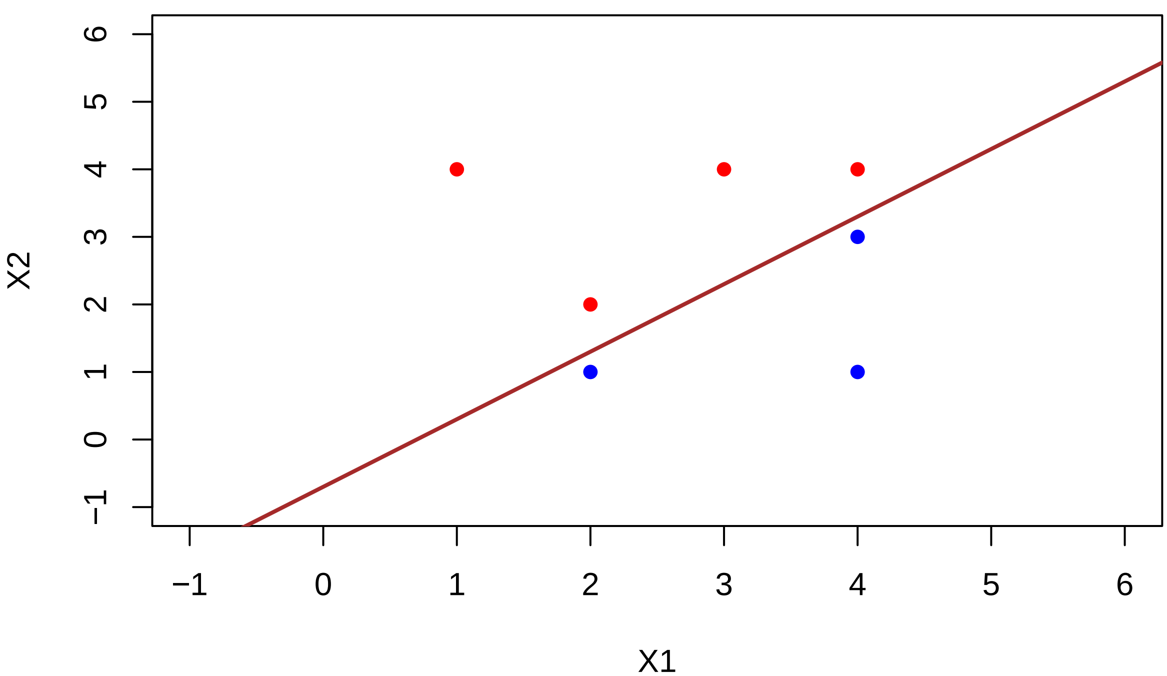
```
plot(-1:6,-1:6,type="n",xlab='X1', ylab='X2')
points(c(3,1),c(4,4), col="red",pch=16)
points(c(4),c(1), col="blue",pch=16)
points(c(2,2,4,4),c(1,2,3,4),col=c("blue","red","blue","red",pch=1))
abline(-0.5,1,col='brown',lwd=2)
abline(-1, 1, col='black',lty='dotted')
abline(0, 1, col='black',lty='dotted')
legend("topleft",pch=1,
       legend = c("support vectors","support vectors"),
       col=c("blue","red"),
       cex=0.5)
points(4,2,col="blue",pch=16)
arrows(4, 1, 4, 2)
```



We can see a slight movement of 7th observation would not affect the maximal margin hyperplane.

(g)

```
plot(-1:6,-1:6,type="n",xlab='X1', ylab='X2')
points(c(3,2,4,1),c(4,2,4,4), col="red",pch=16)
points(c(2,4,4),c(1,3,1), col="blue",pch=16)
abline(-0.7,1,col='brown',lwd=2)
```

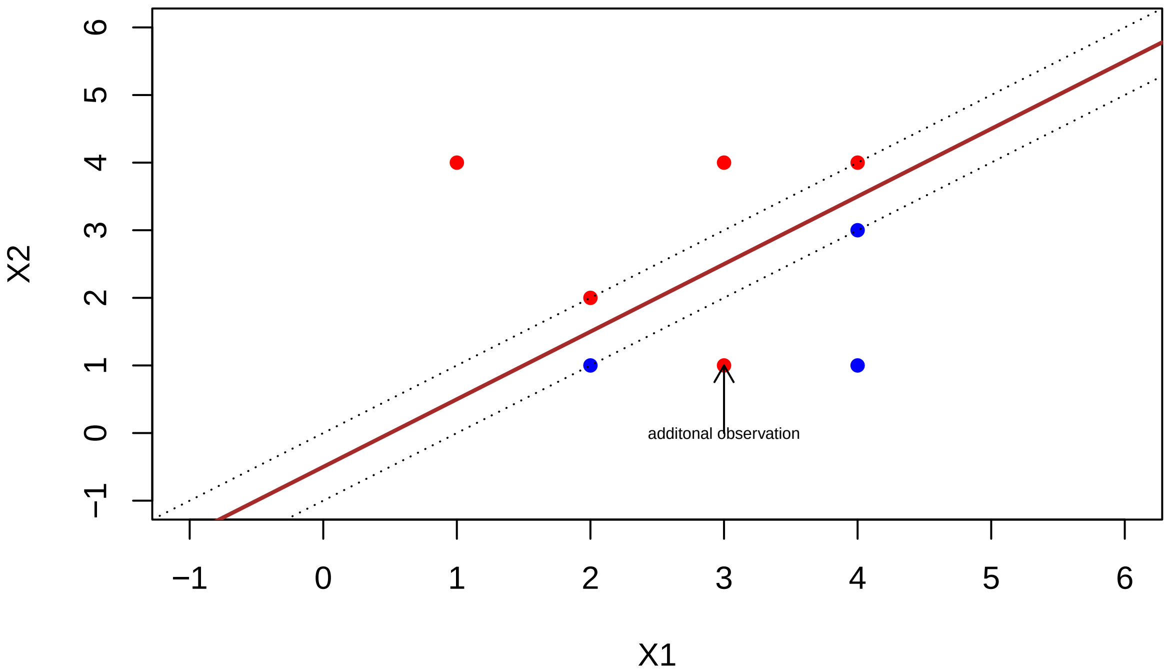


Equation for this hyperplane:

$$-0.7 + X_1 - X_2 = 0$$

(h)

```
plot(-1:6,-1:6,type="n",xlab='X1', ylab='X2')
points(c(3,2,4,1),c(4,2,4,4), col="red",pch=16)
points(c(2,4,4),c(1,3,1), col="blue",pch=16)
abline(-0.5,1,col='brown',lwd=2)
abline(-1, 1, col='black',lty='dotted')
abline(0, 1, col='black',lty='dotted')
points(3,1,col="red",pch=16)
arrows(3,0,3,1,length = 0.1)
text(3,0,"additonal observation",cex=0.5)
```



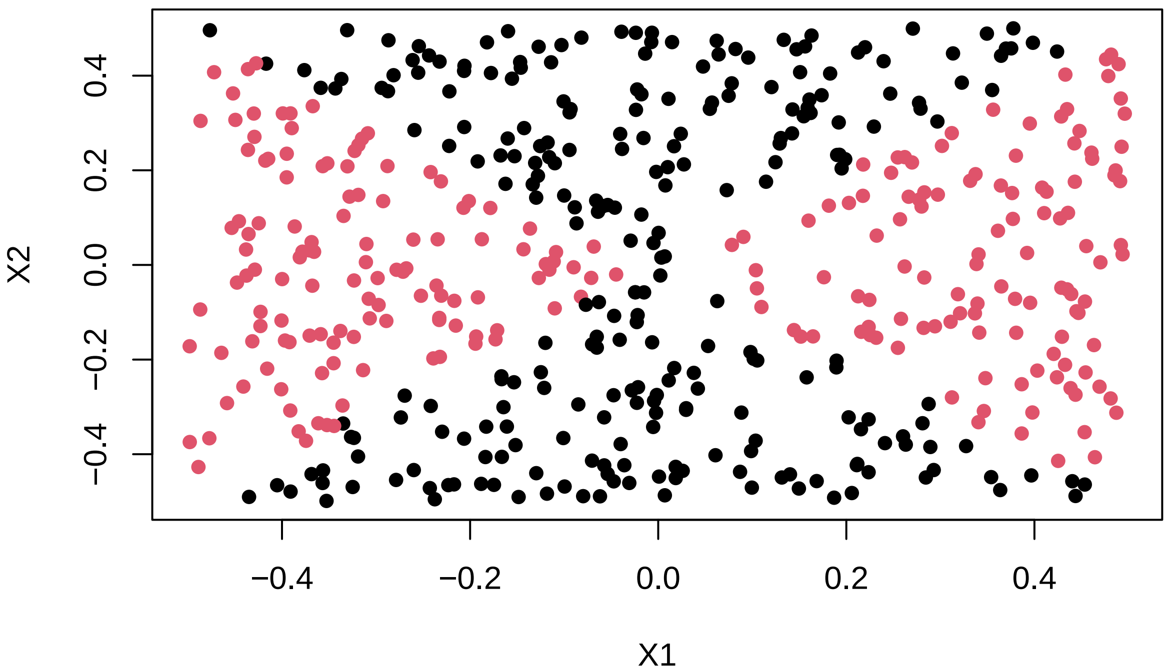
Chap 9 Exercise 5

In following plots, class y = 0 is col = "black" while y = 1 is col = "red". ### (a)

```
set.seed(0)
x1 <- runif (500) -0.5
x2 <- runif (500) -0.5
y <- 1*( x1^2-x2^2 > 0)
data <- data.frame(x1=x1,x2=x2,y=as.factor(y))
```

(b)

```r
plot(x1,x2,col=y+1,xlab="X1",ylab="X2",pch=16)
```

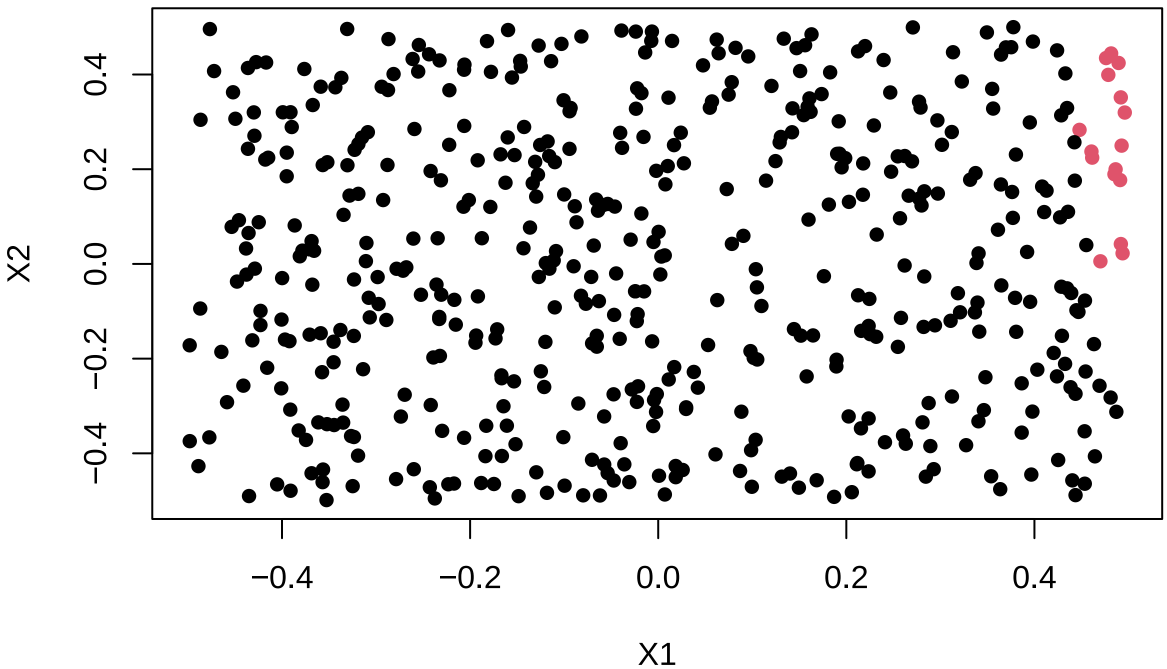


(c)

```r
model <- glm(y~x1+x2, family = "binomial", data=data)
```

(d)

```r
pred <- predict(model,type="response")
pred.class <- 1*(pred>0.5)
plot(x1,x2,col=pred.class+1,xlab="X1",ylab="X2",pch=16)
```



(e)

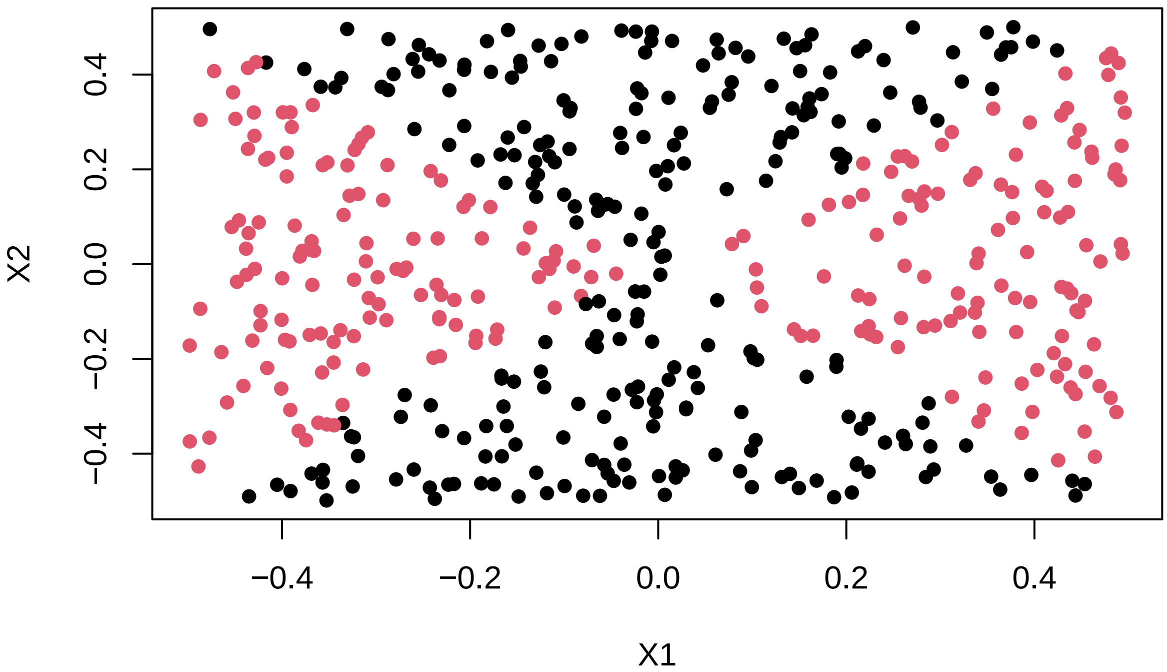
```
model2 <- glm(y~I(x1^2)+I(x2^2)+x1:x2,family = "binomial",data=data)
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

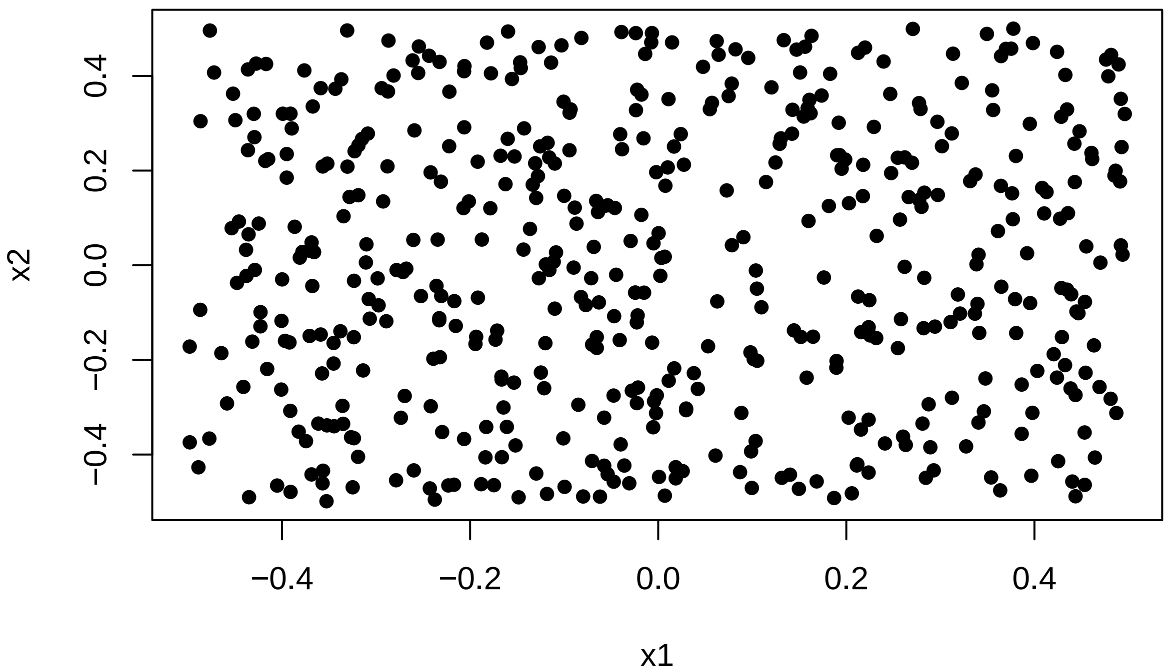
(f)

```
pred2 <- predict(model2,type="response",newdata = data)
pred2.class <- 1*(pred2>0.5)
plot(x1,x2,col=pred2.class+1,xlab="X1",ylab="X2",pch=16)
```



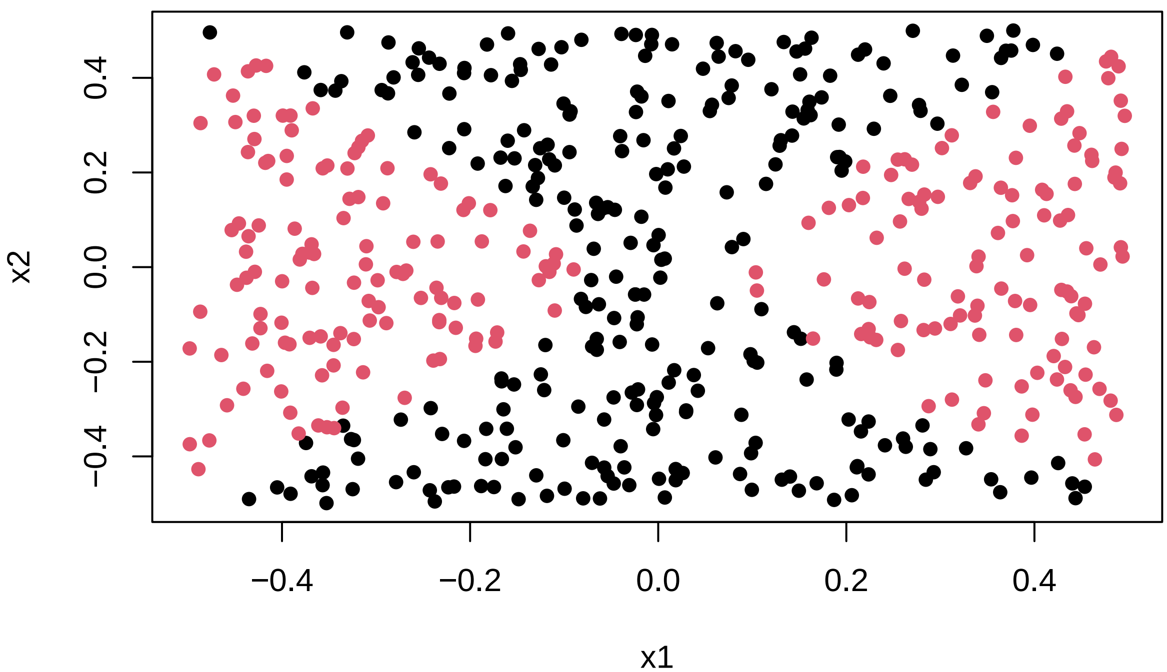
(g)

```
e5g.svmfit <- svm(y~.,data=data,kernel="linear",cost=0.1,scale=FALSE)
svm.pred <- predict(e5g.svmfit,data)
plot(x1,x2,col=svm.pred,pch=16)
```



(h)

```
e5h.svmfit <- svm(y~.,data=data,kernel="radial",gamma=1,cost=1)
svm.pred2 <- predict(e5h.svmfit,data)
plot(x1,x2,col=svm.pred2,pch=16)
```



(i)

Both logistic regression and SVM without non-linear terms have a bad performance. When introduce non-linear terms to logistic regression and SVM with a radial kernel, the models almost have exact predictions on train data. Overall, the two models have quite similar behavior in this case.

Chap 9 Exercise 8

(a)

```
set.seed(0)
n <- nrow(OJ)
train <- sample(n,800)
OJ.train <- OJ[train,]
OJ.test <- OJ[-train,]
```

(b)

```
svmfit.8b <- svm(Purchase~.,data=OJ.train,cost=0.01,kernel="linear",scale=FALSE)
summary(svmfit.8b)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, cost = 0.01, kernel = "linear",
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  603
##
##  ( 303 300 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

From the summary, a linear kernel was used with cost=0.01, and that there were 603 support vectors, 303 in CH class and 300 in MM class.

(c)

```
svm.pred.train <- predict(svmfit.8b,OJ.train)
svm.pred.test <- predict(svmfit.8b,OJ.test)
table(predict=svm.pred.train, train.truth=OJ.train$Purchase)
```

```
##        train.truth
## predict  CH  MM
##      CH 416 103
##      MM  75 206
```

```
table(predict=svm.pred.test, test.truth=OJ.test$Purchase)
```

```
##        test.truth
## predict  CH  MM
##      CH 134  46
##      MM  28  62
```

```
paste("train error:", (103+75)/800)
```

```
## [1] "train error: 0.2225"
```

```
paste("test error:", round((46+28)/270,4))
```

```
## [1] "test error: 0.2741"
```

Training error rate is 22.25% and test error rate is 27.41%.

(d)

```
set.seed(1)
tune.out <- tune(svm,Purchase~.,data=OJ.train,kernel="linear",
                 ranges = list(cost=c(0.01,0.1,0.5,1,5,10)))
bestmod <- tune.out$best.model
```

(e)

```
## train error
pred.train <- predict(bestmod,OJ.train)
table(predict=pred.train,train.truth = OJ.train$Purchase)
```

```
##        train.truth
## predict  CH  MM
##      CH 428  72
##      MM  63 237
```

```
paste("train error:", (74+62)/800)
```

```
## [1] "train error: 0.17"
```

```
## test error
pred.test <- predict(bestmod,OJ.test)
table(predict=pred.test,test.truth = OJ.test$Purchase)
```

```
##        test.truth
## predict  CH  MM
##      CH 144  28
##      MM  18  80
```

```
paste("test error:", round((27+18)/270,4))
```

```
## [1] "test error: 0.1667"
```

The train error rate is 17% and the test error rate is 16.67%.

(f)

```
#radial kernel
svm.radial <- svm(Purchase~.,data=OJ.train,kernel="radial")
summary(svm.radial)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  367
##
##  ( 187 180 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
pred.train <- predict(svm.radial,OJ.train)
pred.test <- predict(svm.radial,OJ.test)
table(predict=pred.train, train.truth=OJ.train$Purchase)
```

```
##        train.truth
## predict  CH  MM
##      CH 445  71
##      MM  46 238
```

```
table(predict=pred.test, test.truth=OJ.test$Purchase)
```

```
##        test.truth
## predict  CH  MM
##      CH 144  33
##      MM  18  75
```

```
paste("train error:", (71+46)/800)
```

```
## [1] "train error: 0.14625"
```

```
paste("test error:", round((18+33)/270,4))
```

```
## [1] "test error: 0.1889"
```

```
#best model
set.seed(1)
tune.out2 <- tune(svm,Purchase~.,data=OJ.train,kernel="radial",
                  ranges = list(cost=c(0.01,0.1,0.5,1,5,10)))
bestmod2 <- tune.out2$best.model

# train error rate
pred.train2 <- predict(bestmod2,OJ.train)
table(predict=pred.train2,train.truth = OJ.train$Purchase)
```

```
##        train.truth
## predict  CH  MM
##      CH 446  71
##      MM  45 238
```

```r
paste("train error:", (71+45)/800)
```

```
## [1] "train error: 0.145"
```

```r
## test error
pred.test2 <- predict(bestmod2,OJ.test)
table(predict=pred.test2,test.truth = OJ.test$Purchase)
```

```
##        test.truth
## predict  CH  MM
##      CH 142  31
##      MM  20  77
```

```r
paste("test error:", round((31+20)/270,4))
```

```
## [1] "test error: 0.1889"
```

From the summary, a radial kernel was used with cost=1, and that there were 367 support vectors, 187 in CH class and 180 in MM class. Based on the svm model with radial, the train error rate is 14.625% and the test error rate is 18.89%. Use tune method and we get the best model is when cost = 0.5. The train error rate is 14.5% and the test error rate is 18.89%.

(g)

```r
#polynomial kernel
svm.poly <- svm(Purchase~.,data=OJ.train,kernel="polynomial",degree=2)
summary(svm.poly)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "polynomial",
##     degree = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  438
##
##  ( 223 215 )
```

```
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
pred.train <- predict(svm.poly,OJ.train)
pred.test <- predict(svm.poly,OJ.test)
table(predict=pred.train, train.truth=OJ.train$Purchase)
```

```
##        train.truth
## predict  CH  MM
##      CH 458 106
##      MM  33 203
```

```
table(predict=pred.test, test.truth=OJ.test$Purchase)
```

```
##        test.truth
## predict  CH  MM
##      CH 146  42
##      MM  16  66
```

```
paste("train error:", round((106+33)/800,4))
```

```
## [1] "train error: 0.1737"
```

```
paste("test error:", round((42+16)/270,4))
```

```
## [1] "test error: 0.2148"
```

```
#best model
set.seed(1)
tune.out3 <- tune(svm,Purchase~.,data=OJ.train,kernel="polynomial",degree=2,
                 ranges = list(cost=c(0.01,0.1,0.5,1,5,10)))
bestmod3 <- tune.out3$best.model

# train error rate
pred.train3 <- predict(bestmod3,OJ.train)
table(predict=pred.train3,train.truth = OJ.train$Purchase)
```

```
##        train.truth
## predict  CH  MM
##      CH 457  86
##      MM  34 223
```

```
paste("train error:", (86+34)/800)
```

```
## [1] "train error: 0.15"
```

```r
## test error
pred.test3 <- predict(bestmod3,OJ.test)
table(predict=pred.test3,test.truth = OJ.test$Purchase)
```

```
##        test.truth
## predict  CH  MM
##      CH 145  39
##      MM  17  69
```

```r
paste("test error:", round((39+17)/270,4))
```

```
## [1] "test error: 0.2074"
```

From the summary result, a polynomial kernel was used with cost=1 and degree=2, and that there were 438 support vectors, 223 in CH class and 215 in MM class. Based on the svm model with polynomial kernel, the train error rate is 17.37% and the test error rate is 21.48%. Use tune method and we get the best model is when cost = 5. The train error rate is 15% and the test error rate is 20.74%.

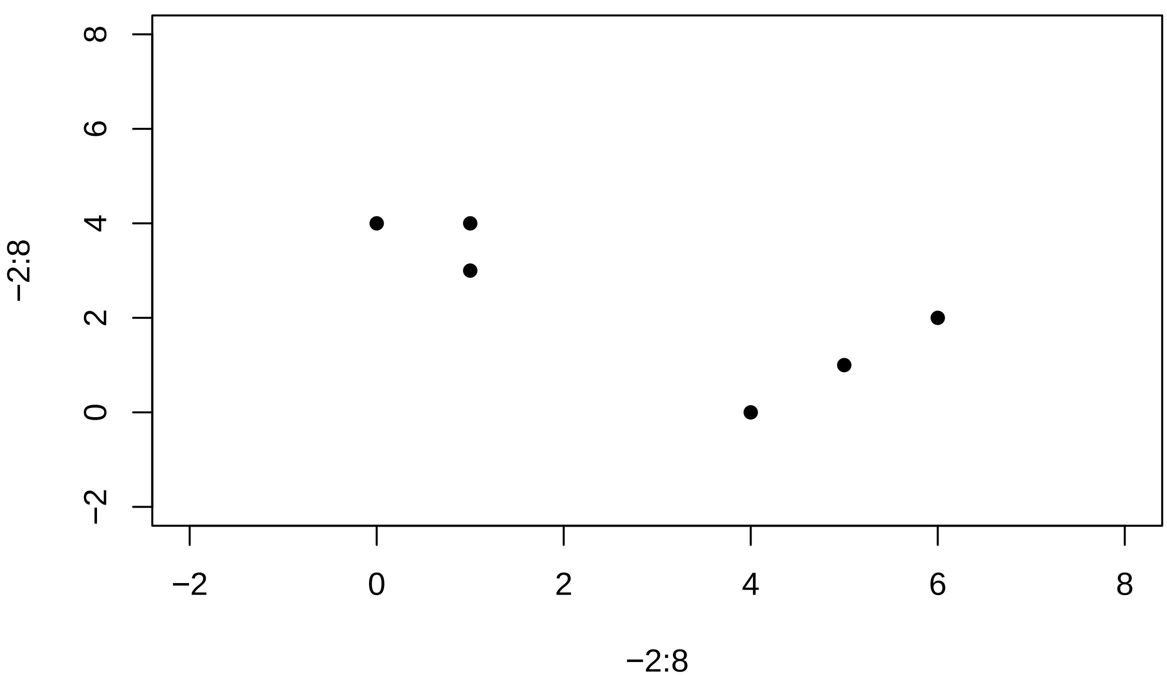
(h)

Overall, when we use svm model with cost = 0.5 and radial kernel, we will get the best result (both train error rate and test error rate) on this data.

Chap10 Exercise 3

(a)

```r
point <- data.frame(X1 = c(1, 1, 0, 5, 6, 4), X2 = c(4, 3, 4, 1, 2, 0))
plot(-2:8,-2:8,type="n")
points(point,pch=16)
```



(b)

```r
set.seed(1)
labels <- sample(c(1,2),6,replace = TRUE);labels
```

```
## [1] 1 2 1 1 2 1
```

```r
point <- cbind(labels,point)
point
```

```
##   labels X1 X2
## 1      1  1  4
## 2      2  1  3
## 3      1  0  4
## 4      1  5  1
## 5      2  6  2
## 6      1  4  0
```

(c)

```r
c1 <- c(mean(point[labels==1,2]),mean(point[labels==1,3]));c1
```

```
## [1] 2.50 2.25
```

```r
c2 <- c(mean(point[labels==2,2]),mean(point[labels==2,3]));c2
```

```
## [1] 3.5 2.5
```

Centroid for label = 1 is (2.50 2.25), and Centroid for label = 2 is (3.5 2.5).

(d)

```r
distance <- function(x0,y0,df){
  return(sqrt((x0-df[2])^2+(y0-df[3])^2))
}
point$c1 <- apply(point,1,distance,x0=2.5,y0=2.25)
point$c2 <- apply(point,1,distance,x0=3.5,y0=2.5)
point$assign <- 2 - (point$c1 < point$c2);point$assign
```

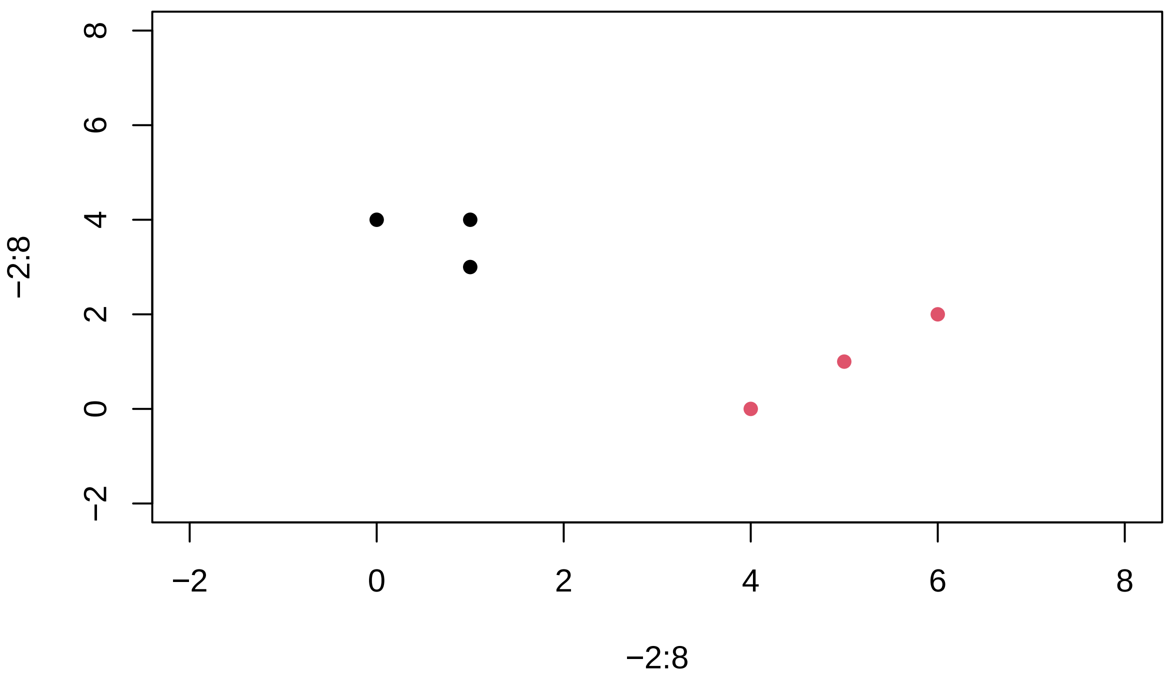
```
## [1] 1 1 1 2 2 2
```

(e)

```
labels <- point$assign
final <- 0
while (!all(final == labels)) {
  final <- labels
  c1 <- c(mean(point[final==1,2]),mean(point[final==1,3]))
  c2 <- c(mean(point[final==2,2]),mean(point[final==2,3]))
  c1.dist <- apply(point,1,distance,x0=c1[1],y0=c1[2])
  c2.dist <- apply(point,1,distance,x0=c2[1],y0=c2[2])
  labels <- 2 - (c1.dist < c2.dist)
}
```

(f)

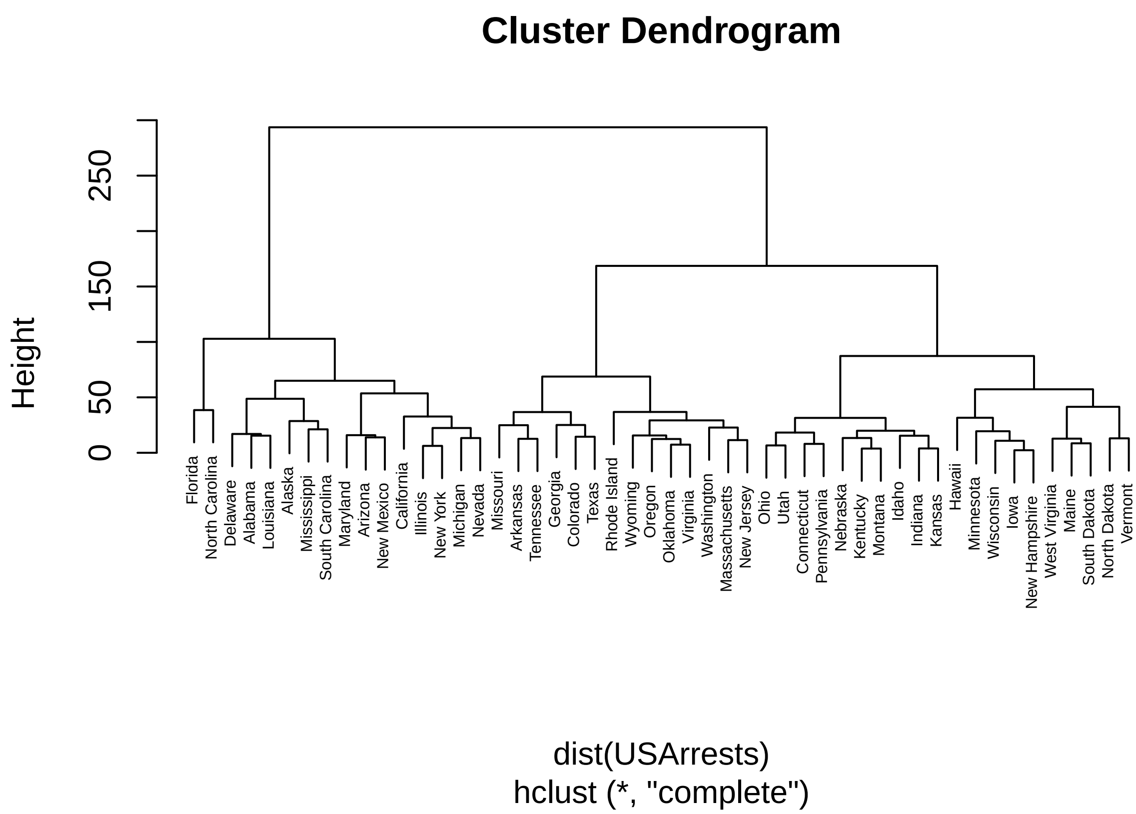
```
plot(-2:8,-2:8,type="n")
points(point[,c("X1","X2")],col=final,pch=16)
```



Chap10 Exercise 9

(a)

```
hc.complete <- hclust(dist(USArrests),method="complete")
plot(hc.complete,cex=0.5)
```



(b)

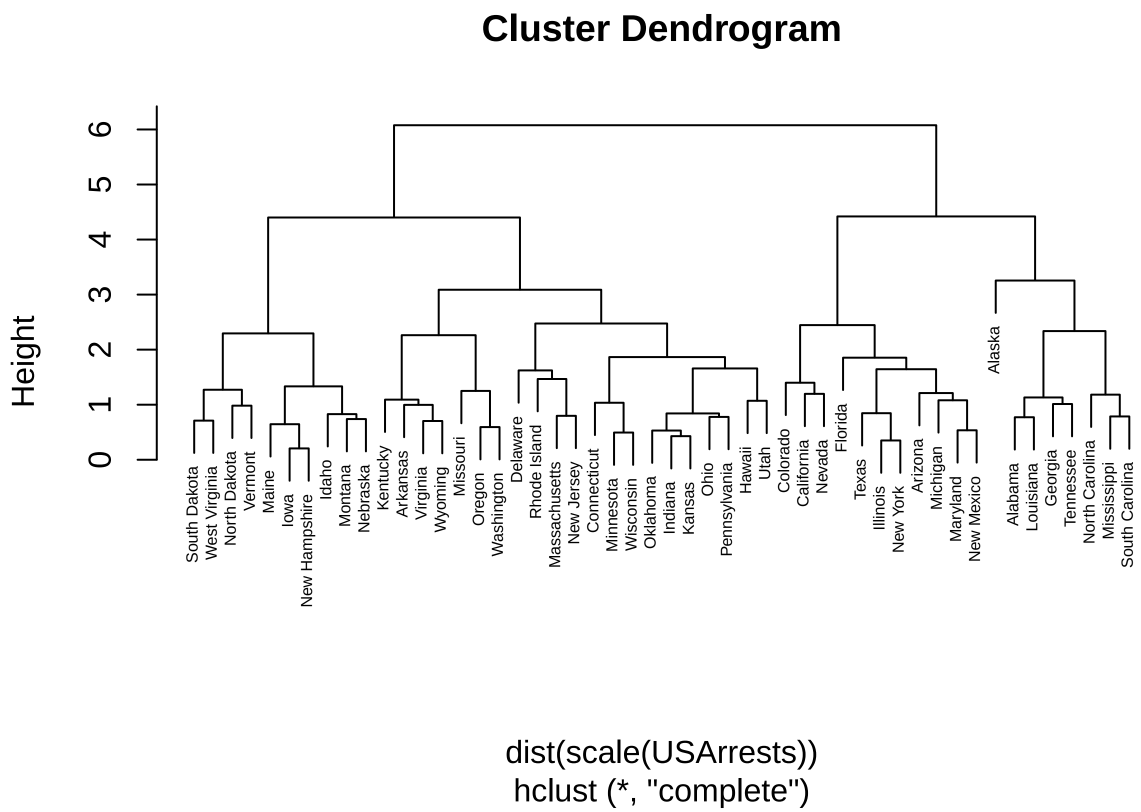
The states are assigned to 3 classes. Shows below:

```
cutree(hc.complete,3)
```

```
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              1              1              2              1
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              2              3              1              1              2
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              3              3              1              3              3
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              3              3              1              3              1
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##              2              1              3              1              2
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##              3              3              1              3              2
##     New Mexico       New York North Carolina   North Dakota           Ohio
##              1              1              1              3              3
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##              2              2              3              2              1
##   South Dakota      Tennessee          Texas           Utah        Vermont
##              3              2              2              3              3
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##              2              2              3              3              2
```

(c)

```
hc.scale <- hclust(dist(scale(USArrests)), method="complete")
plot(hc.scale,cex=0.5)
```



(d)

After scale, the dendrogram height reduce a lot. In this case, the variables should be scaled efore computing the inter-obersvation dissimilarities because variables have different unit.

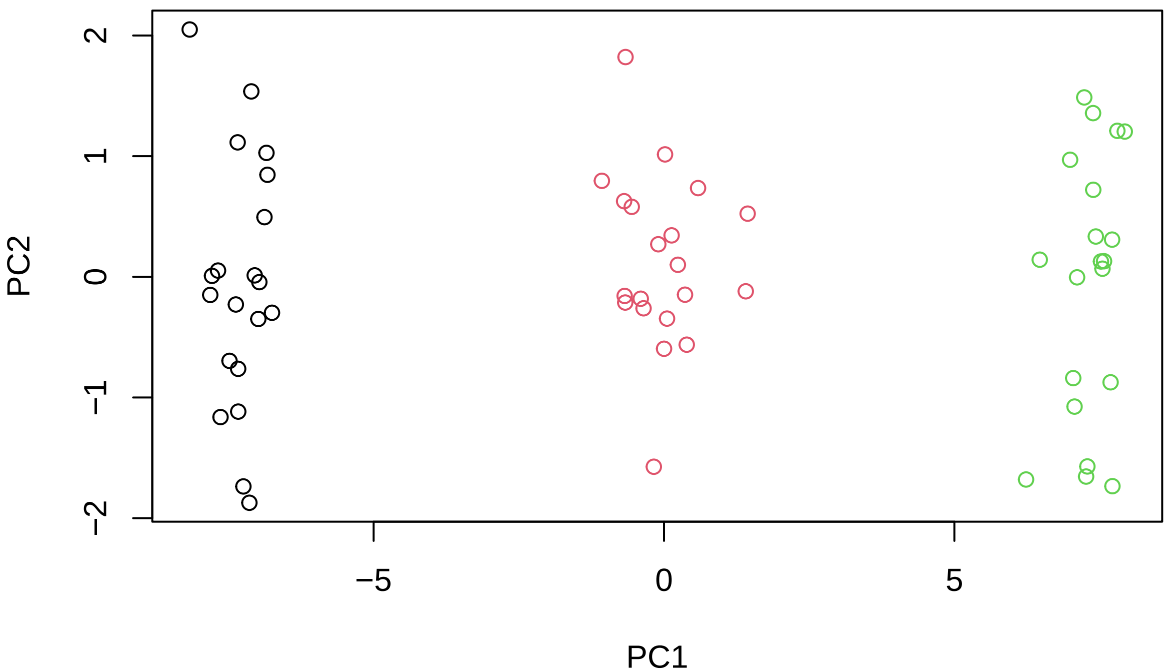
Chap10 Exercise 10

(a)

```
set.seed(1)
data <- matrix(c(rnorm(20*50,mean = 1,sd=1),
                 rnorm(20*50,mean = 3,sd=1),
                 rnorm(20*50,mean = 5,sd=1)),
               ncol = 50,
               byrow = T)
labels <- rep(1:3,each=20)
data <- as.data.frame(data)
```

(b)

```r
pr.out <- prcomp(data,scale=T)
plot(pr.out$x[,1:2],col=labels)
```



(c)

```r
km.out <- kmeans(data,3,nstart = 20)
km.out$cluster
```

```
##  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [39] 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```r
table(km.out$cluster,labels)
```

```
##    labels
##      1  2  3
##   1  0 20  0
##   2  0  0 20
##   3 20  0  0
```

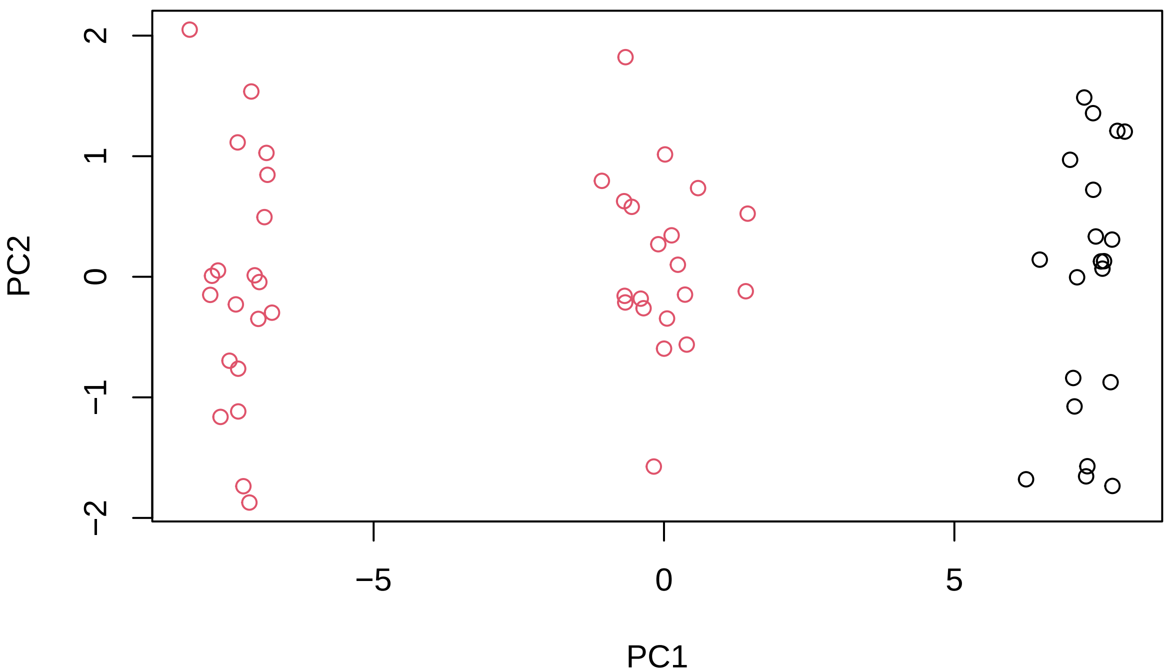
As we can see, the K-means assign first 20 observations to 1, next 20 observations to 2 and last 20 observations to 3. This type of classification is the same as the true labels.

(d)

```r
km.out2 <- kmeans(data,2,nstart=20)
km.out2$cluster
```

```
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [39] 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```r
plot(pr.out$x[,1:2],col=km.out2$cluster)
```



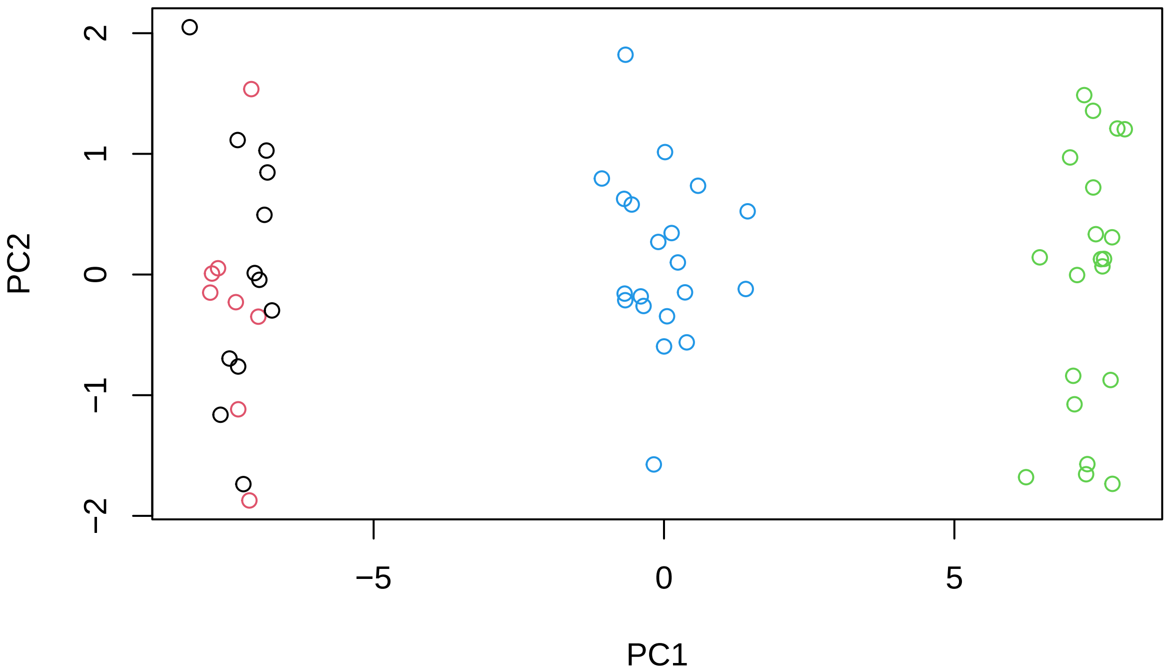
From the cluster results and plot, we can see that all points of one of 3 classes is assigned to another class. Overall there are two clusters.

(e)

```r
km.out4 <- kmeans(data,4,nstart=20)
km.out4$cluster
```

```
##  [1] 1 1 2 2 2 1 1 1 2 2 1 1 2 1 2 2 1 1 1 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [39] 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

```r
plot(pr.out$x[,1:2],col=km.out4$cluster)
```



Points are assigned to 4 clusters. From the plot, we can see that the points in one of the classifications are subdivided into 2 classes.

(f)

```
km.pc <- pr.out$x[,1:2]
km.out.pc <- kmeans(km.pc,3,nstart=20)
km.out.pc$cluster
```

```
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [39] 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

```
table(pc = km.out.pc$cluster,truth = labels)
```

```
##    truth
## pc   1  2  3
##   1  0 20  0
##   2 20  0  0
##   3  0  0 20
```

From the table, we can see that perform K-means clustering on 60x2 matrix of principle component score vectors will get the same result as on the raw data.

(g)

```
km.scale <- kmeans(scale(data),3,nstart=20)
km.scale$cluster
```

```
##  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [39] 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```r
table(scale=km.scale$cluster,truth=labels)
```

```
##      truth
## scale  1  2  3
##     1  0 20  0
##     2  0  0 20
##     3 20  0  0
```

We can see that the result (after scale the variable) in (g) is the same as (b). Since the data set is well separated in 3 classes, there is also well separate after scale.