STATS 216v – Introduction to Statistical Learning
Stanford University, Summer 2017

Problem Set 4

Due: August 16, 2017

Remember the university honor code. All work and answers must be your own.

---

1. ***14 points**: (a) 5 points (b) 5 points (c) 2 points (d) 2 points*

   (a) See the figure.

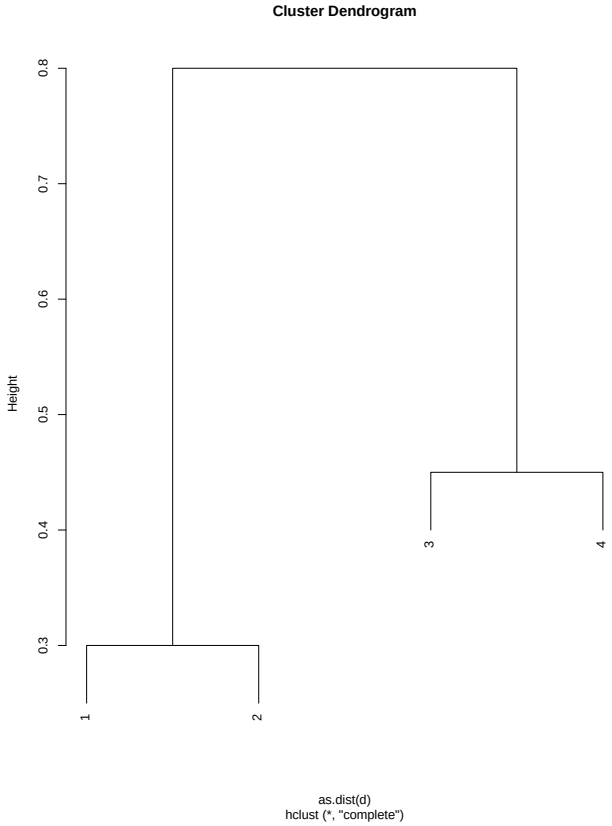


FIGURE 1: Complete linkage dendrogram

   (b) See the figure.

   (c) The clusters are (1,2) and (3,4).

   (d) The clusters are (1,2,3) and (4).

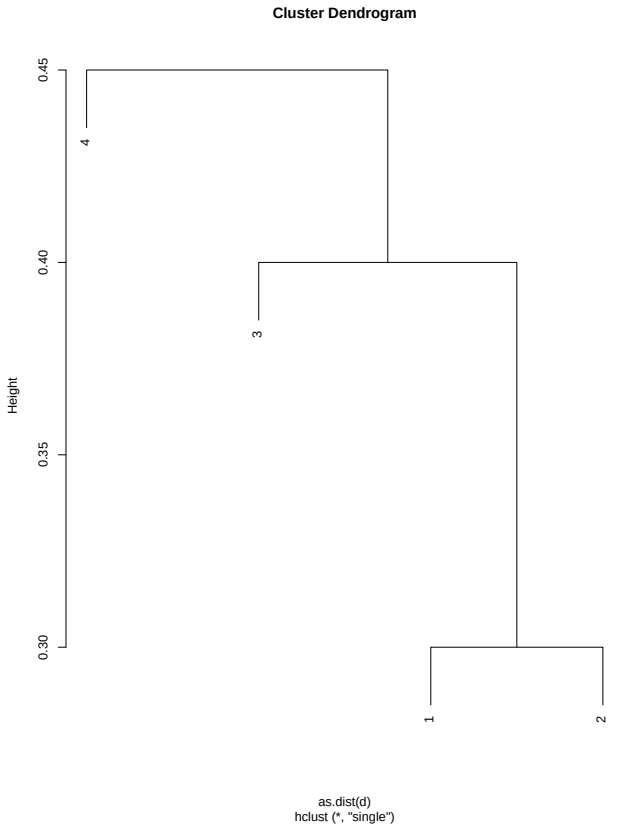
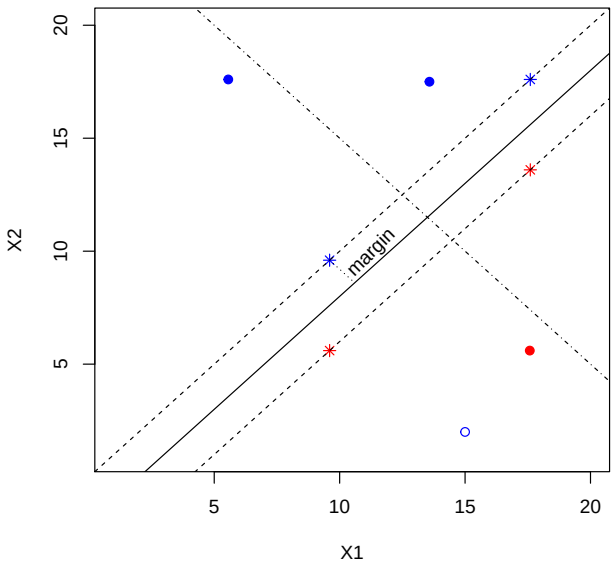


FIGURE 2: Single linkage dendrogram

2. ***13 points****: (a) 1 point (b) 4 points (c) 1 point (d) 2 points (e) 1 point (f) 2 points (g) 1 point (h) 1 point*

(a) The seven observations are plotted below, along with everything that is asked to be plotted later on in the problem. The seven filled-in points are the observations.



Grading notes for part (b): Two points for drawing the hyperplane, two points for giving an equation for the hyperplane. Any equation of the form $2c - cX_1 + cX_2 = 0$ *for some* c *is valid.*

(b) The optimal separating hyperplane is sketched as the solid black line in the figure in part (a).
The equation for this hyperplane is $2 - X_1 + X_2 = 0$.

Grading notes for part (c): Again, any inequality of the form $2c - cX_1 + cX_2 > 0$ *is valid.*

(c) Classify to Blue if $2 - X_1 + X_2 > 0$, and classify to Red otherwise.

Grading notes for part (d): One point for drawing the margin on the figure, one point for giving the correct width.

(d) The margin is indicated on the figure in part (a) by dotted lines running parallel to the optimal separating hyperplane. The width of the margin is approximately $\sqrt{2} = 1.4142$.

(e) The support vectors are indicated by stars in the figure in part (a).

(f) The seventh point is the one in the bottom right-hand corner of the figure. Because it is not a support vector, a slight movement of this observation would not affect the maximal margin hyperplane. As long as this observation is not close to the margin, it will have no effect.

(g) The hyperplane represented by the dash-dot line (with "negative slope") is not the optimal separating hyperplane. The equation for this hyperplane is $-25 + X_1 + X_2 = 0$.

(h) Consider the additional observation represented by the empty blue circle on the plot.
With the addition of this data point, the two classes are no longer separable by a hyperplane.

3. ***30 points**: (a) 0 pts. (b) 3 pt. (c) 3 pts. (d) 3 pts. (e) 3 pts. (f) 5 pts. (g) 5 pts. (h) 5 pts (i) 3pts.*

Grading notes: This problem should actually be very easy to grade because we will grade for completion. Student answers will vary significantly based on test set selection and other issues. Award full credit for any answer that is not obviously wrong. Award half credit (rounded down to the nearest integer number of points) for any part that is answered horribly wrongly.

(a) *(No answer expected here; nothing to grade)*

(b) The number of support vectors is 245 (or about 46% of training data), with about half coming from each class.

(c) The training error rate is 17.0%. The test error rate is 16.4%.

(d) The optimal cost is .32. This varies quite a bit due to randomness in the `tune` function.

(e) For optimal cost, the training error rate is 17.0%. The test error rate is 17.1%.

(f) (b) The number of support vectors is 392 (or about 80% of training data), with about half coming from each class.

(c) The training error rate is 23.6%. The test error rate is 27.3%.

(d) The optimal cost is 2.56.

(e) For optimal cost, the training error rate is 17.0%. The test error rate is 22.2%.

(g) (b) The number of support vectors is about 172 (or about 32% of training data), with about half coming from each class.

(c) The training error rate is 16.1%. The test error rate is 17.4%.

(d) The optimal cost is 5.12. The optimal degree is 2.

(e) For optimal cost and degree, the training error rate is 16.6%. The test error rate is 18.7%.

(h) (b) The number of support vectors is about 152 (or about 28.4% of training data), with about half coming from each class.

(c) The training error rate is 18.1%. The test error rate is 19.6%.

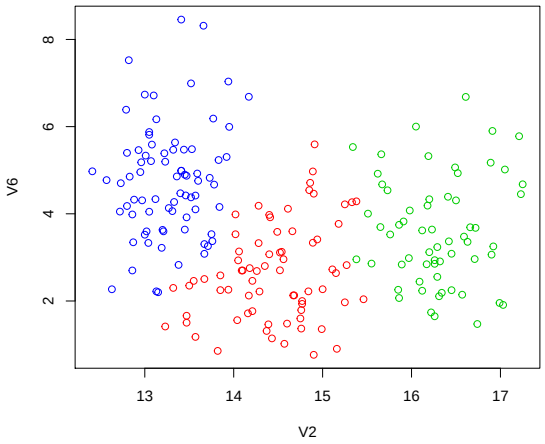
(d) The optimal cost is 0.16.

(e) For optimal cost, the training error rate is 32.1%. The test error rate is 33.4%. With only numeric predictors, this approach is conceptually the same as fitting svm with polynomial kernel with degree equals 2. The results are similar. The difference may be in the way svm treat the quadratic terms involving factor attributes.

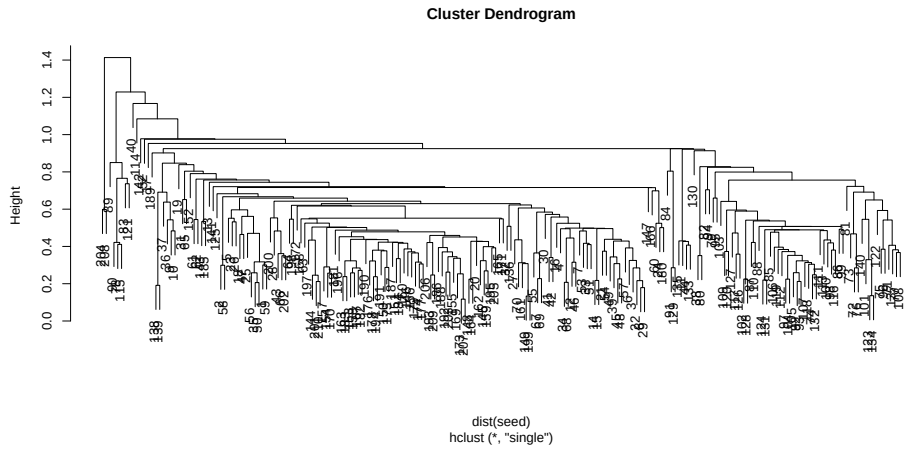
(i) Overall, the best results seem to come from a polynomial kernel with degree 2 and cost 1.

4. ***16 points**: (a) 4 points (b) 2 points (c) 4 points (d) 4 points (e) 2 points*

(a) *Grading notes for part (a): Award full credit for any figure that looks somewhat like the figure below. Student answers may vary due to randomness in kmeans starting position selection.*

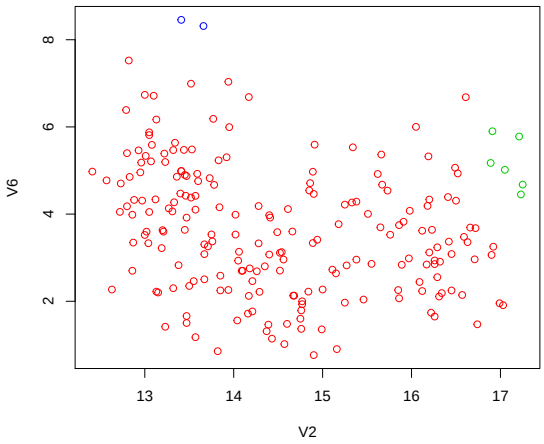


(b) The dendogram should look like this:



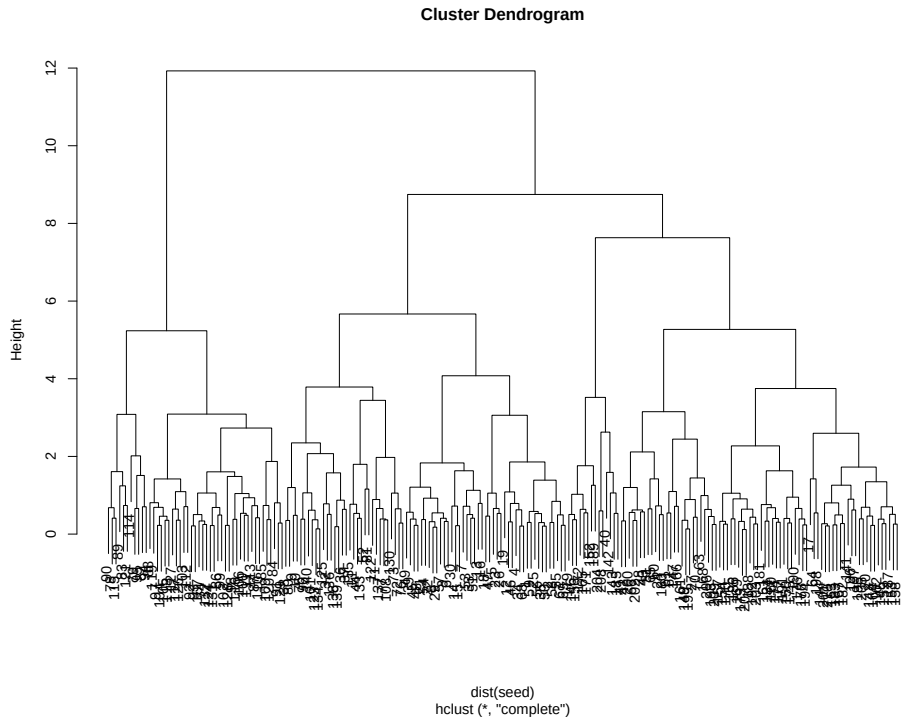
(c) *Grading notes for part (c): 3 points only plotting the figure, 1 points for comparison.*

The clustered figure should look like this:



It is very different from the results acquired by running kmeans. Here we have 1 dominant cluster and 2 very small clusters.

(d) *Grading notes for part (d): Award 2 points for only plotting the dendogram without noticing that the clusters are more balanced in size than the single linkage clusters.* The dendogram should look like this:



This dendrogram would result in 3 clusters of roughly equal size, whereas the single linkage clustering results in one large cluster and two small clusters, which we may not want.

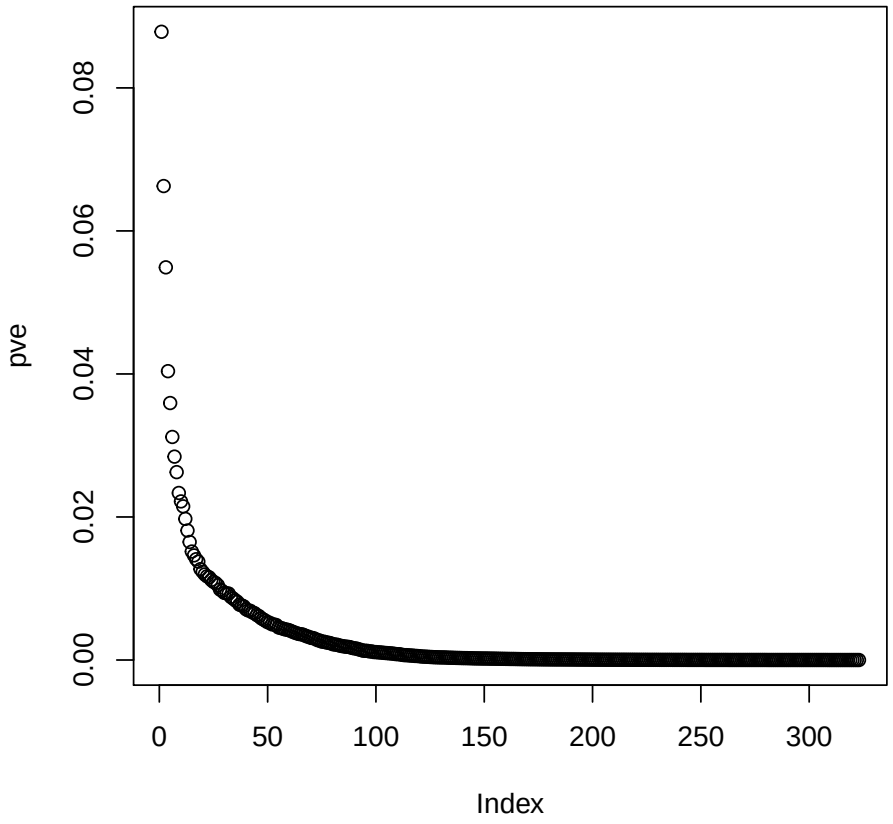
(e) *Grading notes for part (e): Many other multi-class classification techniques are also valid answers to this part of the question. Naming any one method earns full credit.*

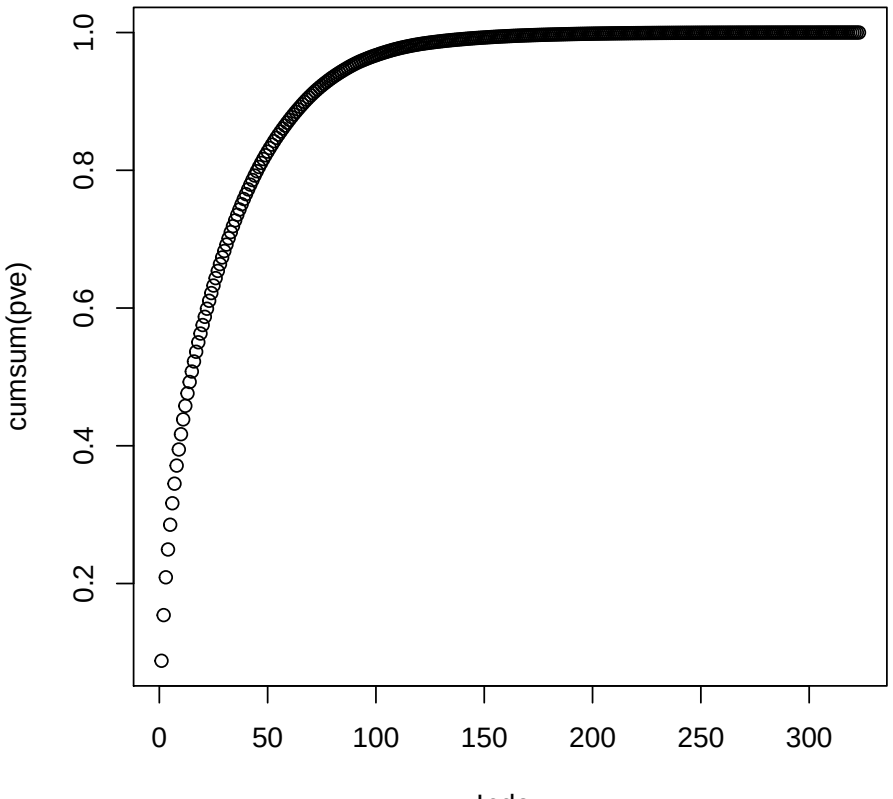
We can use one v.s. all svm, discriminant analysis or logistic regression for multiple classes.

5. ***20 points**: (a) 0 pts. (b) 5 pts. (c) 5 pts. (d) 5pts. (e) 5 pts.*

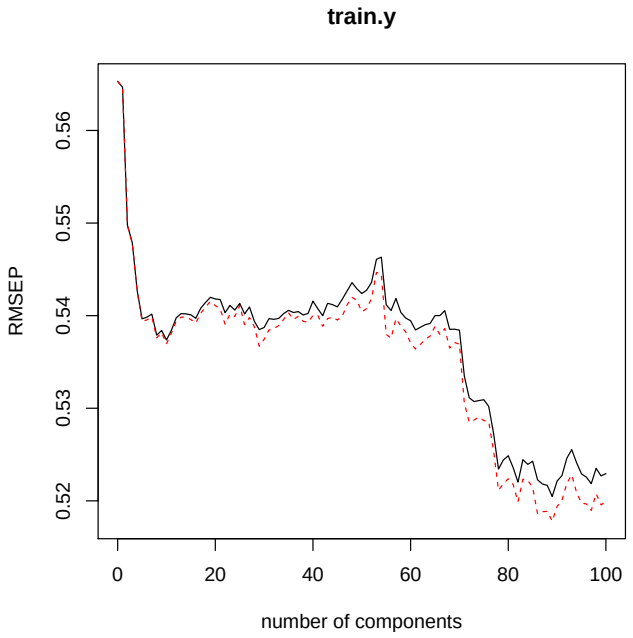
(a) There is nothing to grade.

(b) *Grading notes for part (b): 2 points for each plot.*


pve
0.00
0.02
0.04
0.06
0.08
0
50
100
150
200
250
300
Index


cumsum(pve)
0.2
0.4
0.6
0.8
1.0
0
50
100
150
200
250
300

(c) The plot is below.



(d) 89 components.

(e) The `pcr` test RMSE is 0.5501. This outperforms ordinary linear regression, but is slightly worse that produced by the Lasso, Boosting and Random Forest.

Appendix: Relevant R code

```
# Question 1
d <- matrix(c(0, .3, .4, .7,
              .3, 0, .5, .8,
              .4, .5, 0, .45,
              .7, .8, .45, 0),
            nrow = 4, ncol = 4)

#part a
complete_h <- hclust(as.dist(d), method = "complete")
plot(complete_h)

#part b
single_h <- hclust(as.dist(d), method = "single")
plot(single_h)



# Question 2
x1=c(13.58,9.6,17.6,5.56,9.6,17.6,17.58)
x2=c(17.5,9.6,17.6,17.6,5.6,13.6,5.6)
toydata<-data.frame("X1"=x1,"X2"=x2)
#(a)(e)
plot(toydata$X1,toydata$X2,col=c(rep("blue",4),rep("red",3)),
     pch=c(19,8,8,19,8,8,19),xlab="X1",ylab="X2",xlim=c(1,20),ylim=c(1,20))

#(b),(c)
abline(-2,1,lty=1)

#(d)
abline(0,1,lty=2)
abline(-4,1,lty=2)
lines(c(9.6,10.6),c(9.6,8.6),lty=3)
text(10, 9, "margin", pos=4, cex = 1, srt = 45)

#(g)
abline(25,-1,lty=4)

#(h)
points(15,2,col="blue",pch=1)



# Question 3
library(ISLR)
```

```
library(e1071)
#(a)
set.seed(2017)
train = sample(1:nrow(OJ), 535)
test = setdiff(1:nrow(OJ), train)
#(b)
svmLinear = svm(Purchase ~ ., data = OJ, kernel = 'linear',
                cost = 1, scale = FALSE, subset = train)
summary(svmLinear)
#(c)
mean(predict(svmLinear, newdata = OJ[train, ]) !=
       OJ[train, 'Purchase'])   # training error
mean(predict(svmLinear, newdata = OJ[test, ]) !=
       OJ[test, 'Purchase'])            # test error
#(d)
tuneLinear = tune(svm, Purchase ~ ., data = OJ[train, ],
                  kernel = 'linear', ranges = list(cost = 0.01*2^(1:10)))
summary(tuneLinear)
optimalCost = tuneLinear$best.parameters
#(e)
svmLinearOptimal = svm(Purchase ~ ., data = OJ,
       kernel = 'linear', cost = optimalCost, scale = FALSE, subset = train)
mean(predict(svmLinearOptimal, newdata = OJ[train, ]) !=
       OJ[train, 'Purchase'])   # training error
mean(predict(svmLinearOptimal, newdata = OJ[test, ]) !=
       OJ[test, 'Purchase'])    # test error
#(f)
svmRadial = svm(Purchase ~ ., data = OJ,
                kernel = 'radial', cost = 1, scale = FALSE, subset = train)
summary(svmRadial)
mean(predict(svmRadial, newdata = OJ[train, ]) !=
       OJ[train, 'Purchase'])   # training error
mean(predict(svmRadial, newdata = OJ[test, ]) !=
       OJ[test, 'Purchase'])            # test error
tuneRadial = tune(svm, Purchase ~ ., data = OJ[train, ],
                  kernel = 'radial', ranges = list(cost = 0.01*2^(1:10)))
summary(tuneRadial)
optimalCost = tuneRadial$best.parameters
svmRadialOptimal = svm(Purchase ~ ., data = OJ,
       kernel = 'radial', cost = optimalCost, scale = FALSE, subset = train)
mean(predict(svmRadialOptimal, newdata = OJ[train, ]) !=
       OJ[train, 'Purchase'])   # training error
mean(predict(svmRadialOptimal, newdata = OJ[test, ]) !=
       OJ[test, 'Purchase'])    # test error
#(g)
svmPolynomial = svm(Purchase ~ ., data = OJ,
```

```
  kernel = 'polynomial', cost = 1, degree = 2, scale = FALSE, subset = train)
summary(svmPolynomial)
mean(predict(svmPolynomial, newdata = OJ[train, ]) !=
       OJ[train, 'Purchase'])   # training error
mean(predict(svmPolynomial, newdata = OJ[test, ]) !=
       OJ[test, 'Purchase'])            # test error
tunePolynomial = tune(svm, Purchase ~ ., data = OJ[train, ],
  kernel = 'polynomial', ranges = list(cost = 0.01*2^(1:10), degree = 2))
summary(tunePolynomial)
optimalCost = tunePolynomial$best.parameters$cost
optimalDegree = tunePolynomial$best.parameters$degree
svmPolynomialOptimal = svm(Purchase ~ ., data = OJ,
  kernel = 'polynomial', cost = optimalCost,
  degree = optimalDegree, scale = FALSE, subset = train)
mean(predict(svmPolynomialOptimal, newdata = OJ[train, ]) !=
       OJ[train, 'Purchase'])   # training error
mean(predict(svmPolynomialOptimal, newdata = OJ[test, ]) !=
       OJ[test, 'Purchase'])    # test error
#(h)
#adding all the quadratic terms
OJ2=cbind(OJ,OJ[,c(2,4:7,10:13,15:17)]^2)
id=c(2,4:7,10:13,15:17)
for(i in 1:(length(id)-1)){
  for(j in (i+1):length(id)){
    OJ2=cbind(OJ2,OJ[,id[i]]*OJ[,id[j]])
  }
}
for(i in 1:length(id)){
  OJ2=cbind(OJ2,OJ[,id[i]]*OJ[,8])
}
for(i in 1:length(id)){
  OJ2=cbind(OJ2,OJ[,id[i]]*OJ[,9])
}
for(i in 1:length(id)){
  OJ2=cbind(OJ2,OJ[,id[i]]*OJ[,18])
}
for(i in 1:length(id)){
  OJ2=cbind(OJ2,OJ[,id[i]]*as.numeric(OJ[,14])-1)
}
colnames(OJ2)=c("Purchase",1:143)
svmLinearsq = svm(Purchase ~ ., data = OJ2,
    kernel = 'linear', cost = 0.05, scale = FALSE, subset = train)
summary(svmLinearsq)
mean(predict(svmLinearsq, newdata = OJ2[train, ]) !=
       OJ2[train, 'Purchase'])  # training error
mean(predict(svmLinearsq, newdata = OJ2[test, ]) !=
```

```
       OJ2[test, 'Purchase'])           # test error
tuneLinearsq = tune(svm, Purchase ~ ., data = OJ2[train, ],
       kernel = 'linear', ranges = list(cost = 0.01*2^(1:10)))
summary(tuneLinearsq)
optimalCost = tuneLinearsq$best.parameters
svmLinearsqOptimal = svm(Purchase ~ ., data = OJ2,
       kernel = 'linear', cost = optimalCost, scale = FALSE, subset = train)
mean(predict(svmLinearsqOptimal, newdata = OJ2[train, ]) !=
       OJ2[train, 'Purchase'])  # training error
mean(predict(svmLinearsqOptimal, newdata = OJ2[test, ]) !=
       OJ2[test, 'Purchase'])   # test error



#4
#(a)
seed<-read.csv("SeedData.csv")
set.seed(2017)
points <- seed[,c(2,6)]
plot(points)

km.out <- kmeans(seed, 3, nstart = 20)
plot(points, col = (km.out$cluster + 1))

#(b) & (c)
hc <- hclust(dist(seed), method = "single")
plot(hc)
plot(points, col = cutree(hc,3)+1)

#(d)
hc2 <- hclust(dist(seed), method = "complete")
plot(hc2)

#5
#(a)
load("als.Rdata")
library(pls)
train.X <- train.X
test.X <- test.X
pr.out <- prcomp(train.X, scale = TRUE)

# (b)
pr.var <- pr.out$sdev^2
pve <- pr.var / sum(pr.var)
plot(pve)
plot(cumsum(pve))
```

```
# (c)
pcr.fit <- pcr(train.y ~ ., data = train.X, scale = TRUE, validation = "CV",
               ncomp = 100)
validationplot(pcr.fit)

# (d)
which.min(RMSEP(pcr.fit)$val[2,1,1:101])

# (e)
pcr.pred <- predict(pcr.fit, test.X, ncomp = 86)
pcr.RMSE <- sqrt(mean((pcr.pred - test.y)^2))

pcr.RMSE
```