Homework 4

Casey Gibson

4/6/2018

Problem 1

a)

No (or at least it depends on what you mean by "variable selection" exactly, it is not able to select specific variables, but linear combinations of variables)

b)

No (or at least it depends on what you mean by "variable selection" exactly, it is not able to select specific variables, but linear combinations of variables)

c)

PLS takes into account the correlation between the predictors and the response, whereas PCR ignores it. In this way PCR is unsupervised, and PLS is supervised.

Problem 2

a)

$$p(\hat{X}) = \frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2}}{1+e^{\beta_0+\beta_1 x_1+\beta_2 x_2}}$$

$$p(\hat{X}) = \frac{e^{-6+.05*40+3.5}}{1+e^{-6+.05*40+3.5}} = .3775$$

b)

We need to solve for the inverse to get

$$\frac{e^{-6+.05*x+3.5}}{1+e^{-6+.05*x+3.5}} = .5$$

$$x = 50\ hours$$

Problem 3

We can use Bayes theorem to get

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{1}{2\sigma^2}(x-\mu_k)^2)}{\sum_{i=1}^{k} \pi_i \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{1}{2\sigma^2}(x-\mu_i)^2)}$$

If we then plugin the values from the problem we see

$$\pi_{yes} = .8, \pi_{no} = .2, \mu_{no} = 0, \mu_{yes} = 10, \hat{\sigma^2} = 36, x = 4$$

by wolfram alpha this is

$$p_{yes}(4) = .752$$

Problem 4

Consider

$$log(\frac{p_1(x_1, x_2)}{1 - p_1(x_1, x_2)}) = c_0 + c_1 x_1 + c_2 x_2$$

We can re-write the left hand side as

$$log(p_1(x_1, x_2)) - log(p_2(x_1, x_2))$$

because we only have two-classes $p_1(x_1, x_2) + p_2(x_1, x_2) = 1 \to p_2(x_1, x_2) = 1 - p_1(x_1, x_2)$

By Bayes theorem and independence we can re-write

$$p_1(x_2, x_2) = p_1(x_1)p_1(x_2) \propto p(X_1 = x_1|Y_1)p(X_2 = x_2|Y_1)p(Y_1)$$

$$= \pi_1 \frac{1}{\sqrt{2\pi\sigma_1}} exp(-\frac{1}{2\sigma_1^2}(x_1 - \mu_{11})^2) \frac{1}{\sqrt{2\pi\sigma_2}} exp(-\frac{1}{2\sigma_2^2}(x_2 - \mu_{12})^2)$$

$$= \pi_1 \frac{1}{2\pi\sigma_1\sigma_2} exp(-\frac{1}{2\sigma_1^2}(x_1 - \mu_{11})^2 + -\frac{1}{2\sigma_2^2}(x_2 - \mu_{12})^2)$$

taking logs we can see that

$$= -\frac{1}{2\sigma_1^2}(x_1 - \mu_{11})^2 + -\frac{1}{2\sigma_2^2}(x_2 - \mu_{12})^2 + log(\pi_1) - log(2\pi\sigma_1\sigma_2)$$

$$= -\frac{1}{2\sigma_1^2}(x_1^2 - 2\mu_{11}x_1 + \mu_{11}^2) + -\frac{1}{2\sigma_2^2}(x_2^2 - 2x_2\mu_{12} + \mu_{12}^2) + log(\pi_1) - log(2\pi\sigma_1\sigma_2)$$

By symmetry, we arrive at the same probability for p_2 except with $\mu_{22}, \mu_{21}, \sigma_2, \pi_2$ for parameters.

$$= -\frac{1}{2\sigma_1^2}(x_1^2 - 2\mu_{21}x_1 + \mu_{21}^2) + -\frac{1}{2\sigma_2^2}(x_2^2 - 2x_2\mu_{22} + \mu_{22}^2) + log(\pi_2) - log(2\pi\sigma_1\sigma_2)$$

Subtracting these two we get

$$-\frac{1}{2\sigma_1^2}x_1^2 + \frac{1}{\sigma_1^2}\mu_{11}x_1 - \frac{\mu_{11}^2}{2\sigma_1^2} - \frac{1}{2\sigma_2^2}x_2^2 + \frac{1}{\sigma_2^2}\mu_{12}x_2 - \frac{\mu_{12}^2}{2\sigma_2^2} + log(\pi_1)$$

$+$

$$\frac{1}{2\sigma_1^2}x_1^2 - \frac{1}{\sigma_1^2}\mu_{21}x_1 + \frac{\mu_{21}^2}{2\sigma_1^2} + \frac{1}{2\sigma_2^2}x_2^2 - \frac{1}{\sigma_2^2}\mu_{22}x_2 + \frac{\mu_{22}^2}{2\sigma_2^2} - log(\pi_2)$$

$$c_0 = log(\frac{\pi_1}{\pi_2}) + \frac{\mu_{22}^2}{2\sigma_2^2} - \frac{\mu_{11}^2}{2\sigma_1^2} + \frac{\mu_{21}^2}{2\sigma_1^2} - \frac{\mu_{12}^2}{2\sigma_2^2}$$

$$c_1 = \frac{\mu_{11}}{\sigma_1^2} - \frac{\mu_{21}}{\sigma_1^2}$$

$$c_2 = \frac{\mu_{21}}{\sigma_2^2} - \frac{\mu_{22}}{\sigma_2^2}$$

Problem 5

a)

```r
library(MASS)
n <- length(Boston$crim)
Boston[1,]
```

```
##      crim zn indus chas   nox    rm  age  dis rad tax ptratio black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.09   1 296    15.3 396.9  4.98
##   medv
## 1   24
```

```r
ind = sample(rep(1:5,length=n))
folds <- lapply(split(1:n,ind), function(i) Boston[i,])
```

b)

```r
library (pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

```r
set.seed(2)
train1 <- rbind(folds$`2`,folds$`3`,folds$`4`,folds$`5`)
train2 <- rbind(folds$`1`,folds$`3`,folds$`4`,folds$`5`)
train3 <- rbind(folds$`1`,folds$`2`,folds$`4`,folds$`5`)
train4 <- rbind(folds$`1`,folds$`2`,folds$`3`,folds$`5`)
train5 <- rbind(folds$`1`,folds$`2`,folds$`3`,folds$`4`)

pcr.fit=pcr(train1$crim~., data=train1, scale=TRUE, validation="CV")
aa1 <- pcr.fit[["validation"]][["PRESS"]]
pcr_aa <- which(aa1==min(aa1))
pcr.pred=predict(pcr.fit,folds$`1`, ncomp = pcr_aa)
mse1_pcr <- mean((pcr.pred - folds$`1`$crim)^2)

pcr.fit=pcr(train2$crim~., data=train2, scale=TRUE, validation="CV")
aa1 <- pcr.fit[["validation"]][["PRESS"]]
pcr_aa <- which(aa1==min(aa1))
pcr.pred=predict(pcr.fit,folds$`2`, ncomp = pcr_aa)
mse2_pcr <- mean((pcr.pred - folds$`2`$crim)^2)

pcr.fit=pcr(train3$crim~., data=train3, scale=TRUE, validation="CV")
aa1 <- pcr.fit[["validation"]][["PRESS"]]
pcr_aa <- which(aa1==min(aa1))
pcr.pred=predict(pcr.fit,folds$`3`, ncomp = pcr_aa)
mse3_pcr <- mean((pcr.pred - folds$`3`$crim)^2)

pcr.fit=pcr(train4$crim~., data=train4, scale=TRUE, validation="CV")
aa1 <- pcr.fit[["validation"]][["PRESS"]]
pcr_aa <- which(aa1==min(aa1))
pcr.pred=predict(pcr.fit,folds$`4`[,2:ncol(folds$`4`)], ncomp = pcr_aa)
```

```r
mse4_pcr <- mean((pcr.pred - folds$`4`$crim)^2)

pcr.fit=pcr(train5$crim~., data=train5, scale=TRUE, validation="CV")
aa1 <- pcr.fit[["validation"]][["PRESS"]]
pcr_aa <- which(aa1==min(aa1))
pcr.pred=predict(pcr.fit,folds$`5`, ncomp = pcr_aa)
mse5_pcr <- mean((pcr.pred - folds$`5`$crim)^2)

print ("mse")
```

```
## [1] "mse"
```

```r
print (mean(c(mse1_pcr,mse2_pcr,mse3_pcr,mse4_pcr,mse5_pcr)))
```

```
## [1] 43.56426
```

c)

```r
set.seed(2)
pls.fit=plsr(train1$crim~., data=train1, scale=TRUE, validation="CV")
aa1 <- pls.fit[["validation"]][["PRESS"]]
pls_aa <- which(aa1==min(aa1))
pls.pred=predict(pls.fit,folds$`1`, ncomp = pls_aa)
mse1_pls <- mean((pls.pred - folds$`1`$crim)^2)

pls.fit=plsr(train2$crim~., data=train2, scale=TRUE, validation="CV")
aa1 <- pls.fit[["validation"]][["PRESS"]]
pls_aa <- which(aa1==min(aa1))
pls.pred=predict(pls.fit,folds$`2`, ncomp = pls_aa)
mse2_pls <- mean((pls.pred - folds$`2`$crim)^2)

pls.fit=plsr(train3$crim~., data=train3, scale=TRUE, validation="CV")
aa1 <- pls.fit[["validation"]][["PRESS"]]
pls_aa <- which(aa1==min(aa1))
pls.pred=predict(pls.fit,folds$`3`, ncomp = pls_aa)
mse3_pls <- mean((pls.pred - folds$`3`$crim)^2)

pls.fit=plsr(train4$crim~., data=train4, scale=TRUE, validation="CV")
aa1 <- pls.fit[["validation"]][["PRESS"]]
pls_aa <- which(aa1==min(aa1))
pls.pred=predict(pls.fit,folds$`4`, ncomp = pls_aa)
mse4_pls <- mean((pls.pred - folds$`4`$crim)^2)

pls.fit=plsr(train5$crim~., data=train5, scale=TRUE, validation="CV")
aa1 <- pls.fit[["validation"]][["PRESS"]]
pls_aa <- which(aa1==min(aa1))
pls.pred=predict(pls.fit,folds$`5`, ncomp = pls_aa)
mse5_pls <- mean((pls.pred - folds$`5`$crim)^2)

print ("mse pls")
```

```
## [1] "mse pls"
```

```r
print (mean(c(mse1_pls,mse2_pls,mse3_pls,mse4_pls,mse5_pls)))
```

```
## [1] 43.55214
```

d)

Although PCR performed slightly better, the MSE's are very comparable, so I would choose PLSR because it chose fewer components than PCR did, therefore there are fewer parameters and the model is more parsimonious.

e)

Yes, even though the PLS model doesn't use all the components, it still uses all the features, as long as we define "feature" as "predictor" which Intro to Statistical Learning seems to. This is because a component is a linear combination of all the features.

Problem 6

a)

```r
library(ISLR)
fit.glm =glm(Direction ~ Lag1 + Lag2 + Lag3 +Lag4 + Lag5 + Volume,data=Weekly, family= binomial)

summary(fit.glm)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

Lag2 is significant at the $\alpha = .05$ level.

b)

```
probs = predict(fit.glm , type="response")
pred.glm = rep("Down", length(probs))
pred.glm[probs > 0.5 ] = "Up"
table(pred.glm, Weekly$Direction)
```

```
##
## pred.glm Down  Up
##     Down   54  48
##     Up    430 557
```

We can see that the model is correct

```
print ((54+557)/(54+557 + 430 +48))
```

```
## [1] 0.5610652
```

56% of the time

When the market goes up, the model predicts up

```
print (557/(48+557))
```

```
## [1] 0.9206612
```

However, when the market goes down the model is only correct

```
print (54/(54+430))
```

```
## [1] 0.1115702
```

c)

```
train = (Weekly$Year < 2009)
Weekly.20092010 = Weekly[!train,]
Direction.20092010 =  Weekly$Direction[!train]
fit.glm2 =  glm(Direction ~ Lag2, data= Weekly, family=binomial , subset=train)
summary(fit.glm2)
```

```
##
## Call:
## glm(formula = Direction ~ Lag2, family = binomial, data = Weekly,
##     subset = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.536  -1.264   1.021   1.091   1.368
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.20326    0.06428   3.162  0.00157 **
## Lag2         0.05810    0.02870   2.024  0.04298 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1354.7  on 984  degrees of freedom
```

```
## Residual deviance: 1350.5  on 983  degrees of freedom
## AIC: 1354.5
##
## Number of Fisher Scoring iterations: 4
```

```r
probs2 = predict(fit.glm2, Weekly.20092010, type="response")
pred.glm2 = rep("Down",length(probs2))
pred.glm2[probs2>.5] = "Up"
table(pred.glm2,Direction.20092010)
```

```
##          Direction.20092010
## pred.glm2 Down Up
##      Down    9  5
##      Up     34 56
```

```r
print ("correct predition")
```

```
## [1] "correct predition"
```

```r
print ((9+56)/(9+5+34+56))
```

```
## [1] 0.625
```

d)

```r
library(MASS)
fit.lda  = lda(Direction~ Lag2, data= Weekly, subset =train)
fit.lda
```

```
## Call:
## lda(Direction ~ Lag2, data = Weekly, subset = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
##
## Coefficients of linear discriminants:
##            LD1
## Lag2 0.4414162
```

```r
pred.lda =  predict(fit.lda, Weekly.20092010)
table(pred.lda$class, Direction.20092010)
```

```
##       Direction.20092010
##        Down Up
##   Down    9  5
##   Up     34 56
```

LDA gives the same result as above.

e)

```r
library(MASS)
fit.qda  = qda(Direction~ Lag2, data= Weekly, subset =train)
fit.qda
```

```
## Call:
## qda(Direction ~ Lag2, data = Weekly, subset = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##              Lag2
## Down -0.03568254
## Up    0.26036581
```

```r
pred.qda =  predict(fit.qda, Weekly.20092010)
table(pred.qda$class, Direction.20092010)
```

```
##       Direction.20092010
##        Down Up
##   Down    0  0
##   Up     43 61
```

Correct prediction is

```r
print (61/(43+61))
```

```
## [1] 0.5865385
```

f)

```r
library(class)
train.X = as.matrix(Weekly$Lag2[train])
test.X = as.matrix(Weekly$Lag2[!train])
train.Direction = Weekly$Direction[train]
set.seed(1)
pred.knn = knn(train.X, test.X, train.Direction, k=1)
table(pred.knn, Direction.20092010)
```

```
##         Direction.20092010
## pred.knn Down Up
##     Down   21 30
##     Up     22 31
```

The correct prediction percentage is

```r
print ((21+31)/(21+31+30+22))
```

```
## [1] 0.5
```

g)

Logistic regression and LDA have the best error rate.