Lab 6

Attila Lazar

18.11.2020

1)

data

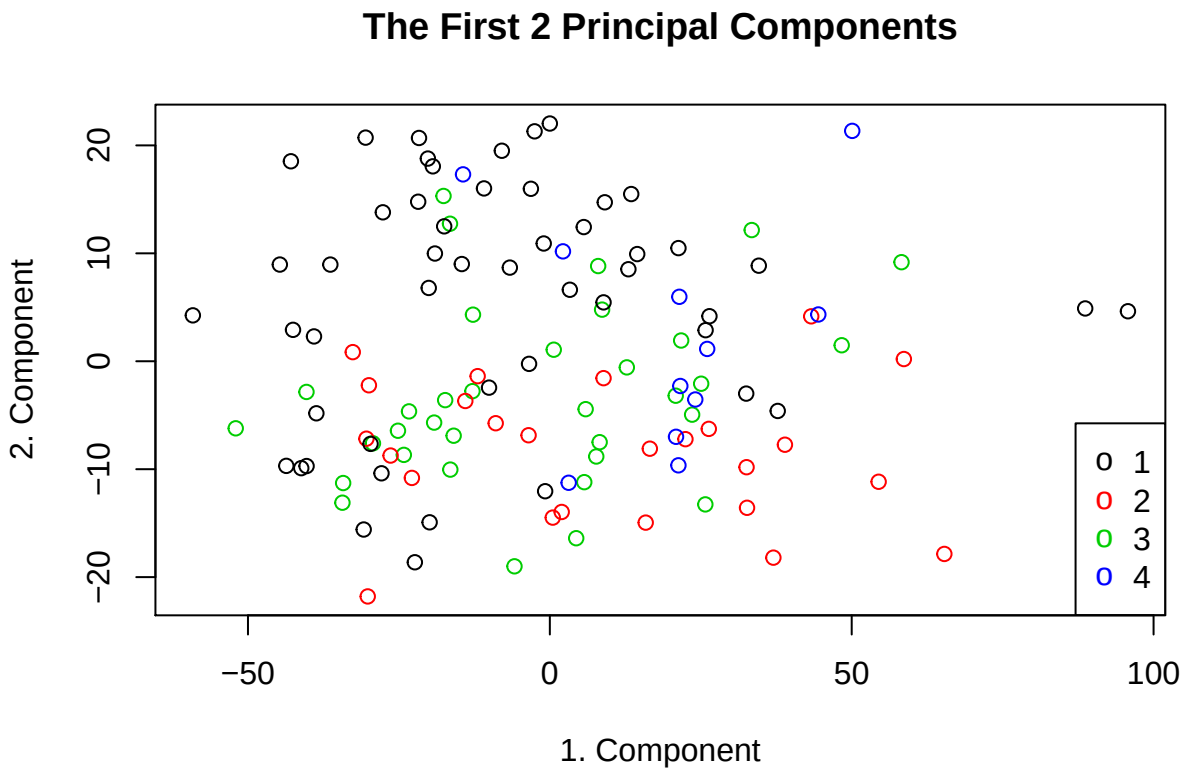
We load the dataset

```
#install.packages("rrcov")
data(olitos, package="rrcov")
```

a)

We visualize the first two principal components

```
X <- olitos[,-26]
pc <- princomp(X)
plot(pc$scores[,1], pc$scores[,2], col=olitos$grp, xlab="1. Component", ylab="2. Component", main="The F
legend("bottomright", legend=levels(olitos$grp), col=seq(1,4), pch = "o")
```



Based on these two components linear sepatation does not seem doable.

b)

We separate the data in test and training sets. Then we encode the response variable in 4 columns, with the value 1 denoting membership in the given class, 0 non-membership.

```
set.seed(1234)
n <- nrow(olitos)
train <- sample(1:n, round(n*2/3))
test <- (1:n) [-train]
response <- cbind(ifelse(olitos$grp==1, 1, 0), ifelse(olitos$grp==2, 1, 0), ifelse(olitos$grp==3, 1, 0)
data <- olitos[,-26]
head(response)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    1    0    0    0
## [3,]    0    1    0    0
## [4,]    0    1    0    0
## [5,]    0    1    0    0
## [6,]    0    1    0    0
```

Then we train a multivariate Least Squares Regression model on the training set

```
model1 <- lm(response~., data=data, subset=train)
#summary(model1)
```

Te resulting model contains 4 different models for each response variable. the variable *X9* seem to have the most overall significance.

c)

We predict values of our test-set and print the confusion matrix

```
pred.y <- predict(model1, data[test,])
pred <- apply(pred.y, 1, which.max)
confmat1 <- table(pred, olitos[test,'grp'])
confmat1
```

```
##
## pred  1  2  3  4
##    1 14  0  1  4
##    2  3  7  0  1
##    3  4  0  4  0
##    4  0  0  1  1
```

The missclassification rate is:

```
paste0((1-sum(diag(confmat1))/sum(confmat1)) * 100, "%")
```

```
## [1] "35%"
```

2)

a)

We load the dataset, extract train and test sets

```
bank <- read.csv2("data/bank.csv")
#bank <- rbind(bank, bank[which(bank$y == "yes"),])

set.seed(1234)
train <- sample(1:nrow(bank), 3000)
test <- (1:nrow(bank)) [-train]
```

We encode the response variable in 2 columns of "0" and "1" and then construct a multivariate Least Regression Model using the training set

```
x <- bank[,-17]
y1 <- ifelse(bank$y==levels(bank$y)[1], 1, 0)
y2 <- ifelse(bank$y==levels(bank$y)[2], 1, 0)
data <- cbind(x,y1,y2)
model2 <- lm(cbind(y1,y2)~., data=data, subset=train)
```

```
pred.num <- predict(model2, data[test,])
pred <- ifelse(apply(pred.num, 1, which.max) == 1, "no", "yes")
confmat2 <- table(bank[test,'y'], pred)
confmat2
```

```
##      pred
##         no  yes
##   no  1340   19
##   yes  122   40
```

with a missclassification rate of:

```
paste0((1-sum(diag(confmat2))/sum(confmat2)) * 100, "%")
```

```
## [1] "9.27021696252466%"
```

We are interested in keeping False Negatives Low thus in high sensitivity:

```
confmat2[2,2] / (confmat2[2,1] + confmat2[2,2])
```

```
## [1] 0.2469136
```

As a strategy to get less False Negatives, we try to sample a balanced population. We sample 1000 "yes" and 1000 "no" samples using bootrapping from the training dataset. We train our model and test with predictions from the test dataset

```
bsrp.y <- sample(which(data[train, "y1"] == 1), 1000, replace=TRUE)
bsrp.n <- sample(which(data[train, "y1"] == 0), 1000, replace=TRUE)
bsrp <- as.vector(rbind(bsrp.y,bsrp.n))

model2b <- lm(cbind(y1,y2)~., data=data, subset=bsrp)

pred.num <- predict(model2b, data[test,])
pred <- ifelse(apply(pred.num, 1, which.max) == 1, "no", "yes")
confmat2b <- table(bank[test,'y'], pred)
confmat2b
```

```
##      pred
##         no  yes
##   no  1328   31
##   yes   99   63
```

missclassification error

```
paste0((1-sum(diag(confmat2b))/sum(confmat2b)) * 100, "%")
```

```
## [1] "8.54700854700855%"
```

and sensitivity

```
confmat2b[2,2] / (confmat2b[2,1] + confmat2b[2,2])
```

```
## [1] 0.3888889
```

We still get a fair amount of false negatives, our missclassification rate and sensitivity improved a little bit.

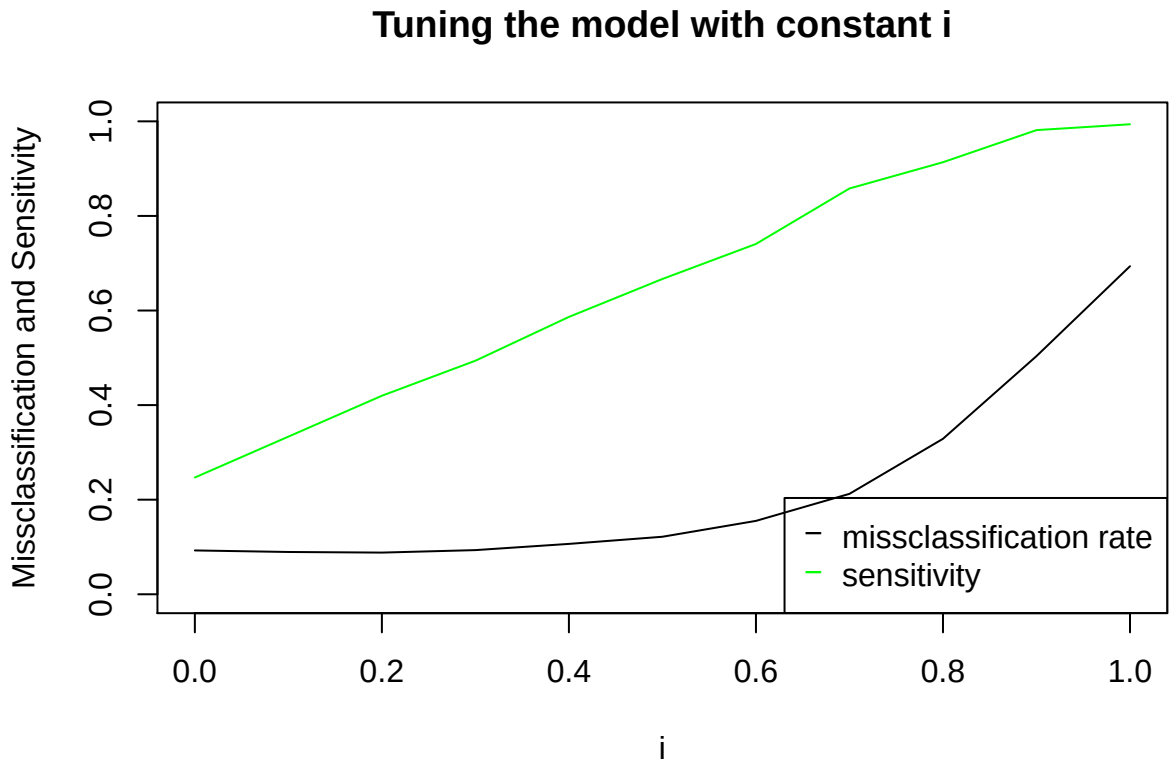
As next Strategy we simply add a constant number i to one of the predicted classes. since the class with the higher predicted response column is predicted as the samples response class, this method effectively pushes the separation boundary in the direction of one class.

We evaluate the effect on the missclassification-rate and sensitivity by adding values from 0 to 1 to the "yes" column of the response

```
specificity <- array()
accuracy <- array()
i <- array()

for (x in seq(0, 1, 0.1)) {
  pred.num <- predict(model2, data[test,])
  pred.num[,2] <- pred.num[,2] + x
  pred <- ifelse(apply(pred.num, 1, which.max) == 1, "no", "yes")
  confmatrix <- table(bank[test,'y'], pred)
  specificity <- c(specificity, confmatrix[2,2] / (confmatrix[2,1] + confmatrix[2,2]))
  accuracy <- c(accuracy, sum(diag(confmatrix)) / sum(confmatrix))
  i <- c(i, x)
}

plot(i, 1-accuracy, type='l', ylim=c(0,1), col="black", ylab="Missclassification and Sensitivity", main=
lines(i, specificity, type='l', col="green")
legend("bottomright", legend=c('missclassification rate','sensitivity'), col = c("black", "green"), pch
```



With a tuning constant of 0.5 we get slightly more missclassified samples but much less false negatives

```
pred.num <- predict(model2, data[test,])
pred.num[,2] <- pred.num[,2] + 0.5
pred <- ifelse(apply(pred.num, 1, which.max) == 1, "no", "yes")
confmat2c <- table(bank[test,'y'], pred)
confmat2c
```

```
##      pred
##         no  yes
##   no  1228  131
##   yes   54  108
```

missclassification error

```
paste0((1-sum(diag(confmat2c))/sum(confmat2c)) * 100, "%")
```

```
## [1] "12.1630506245891%"
```

and sensitivity

```
confmat2c[2,2] / (confmat2c[2,1] + confmat2c[2,2])
```

```
## [1] 0.6666667
```