ZIP Practical Work : PLS2

Multivariate Modeling

Martin Guy

December 31, 2016

Introduction

We have normalized handwritten digits, automatically scanned from envelopes by the U.S. Postal Service in 16 x 16 grayscale images (from -1 to 1). The goal of this exercise is to recognize the right number that is written. The purpose is to continue the exercise we did for session 1 using Multivariate Regression and a **Principal Components Regression** and in session 2 using **Inter-Battery Analysis**. Now we will try **Partial Least Square 2** (PLS2) as a component based methodology to predict the digits.

1. Read the "zip_train.dat" and "zip_test.dat" files provided.

```
train.full <- read.table("zip_train.dat")
test <- read.table("zip_test.dat")
```

Select a 5% random sample (without replacement) of the train data

```
proportion <- 0.05
n.full <- dim(train.full)[1]  #original size of the training set
n <- floor(n.full * proportion) #new size of the training set

train <- train.full[sample(n.full,n),]
```

2. Define the response matrix (Y) and the predictor matrix (X).

```
X.train <- as.matrix(train[,-1])
X.test <- as.matrix(test[,-1])

Y.train <- class.ind(train[,1])
Y.test <- class.ind(test[,1])
```

Center the predictor matrix.

We center but not scale the data

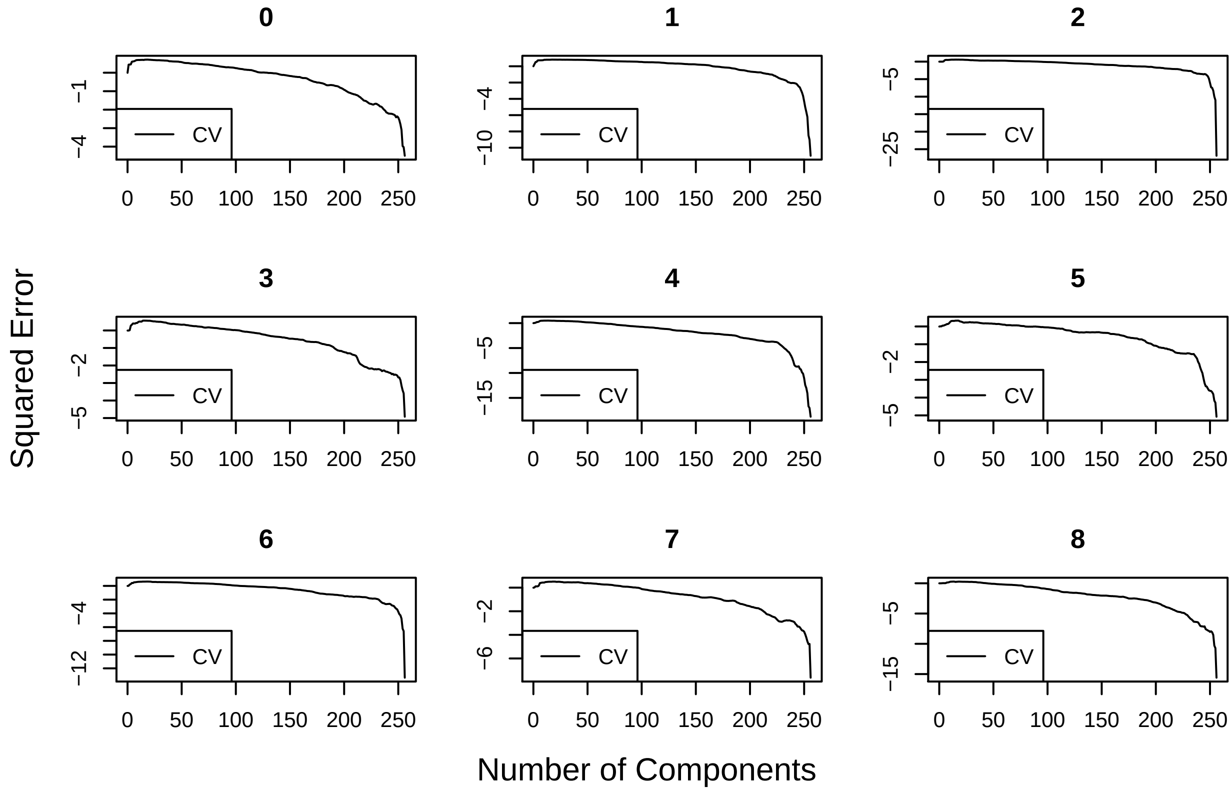
```
X.train.means <- colMeans(X.train)
X.train <-scale(X.train, scale=FALSE)
```

3. Perform a PLSR2 using "CV" or "LOO" for validation.

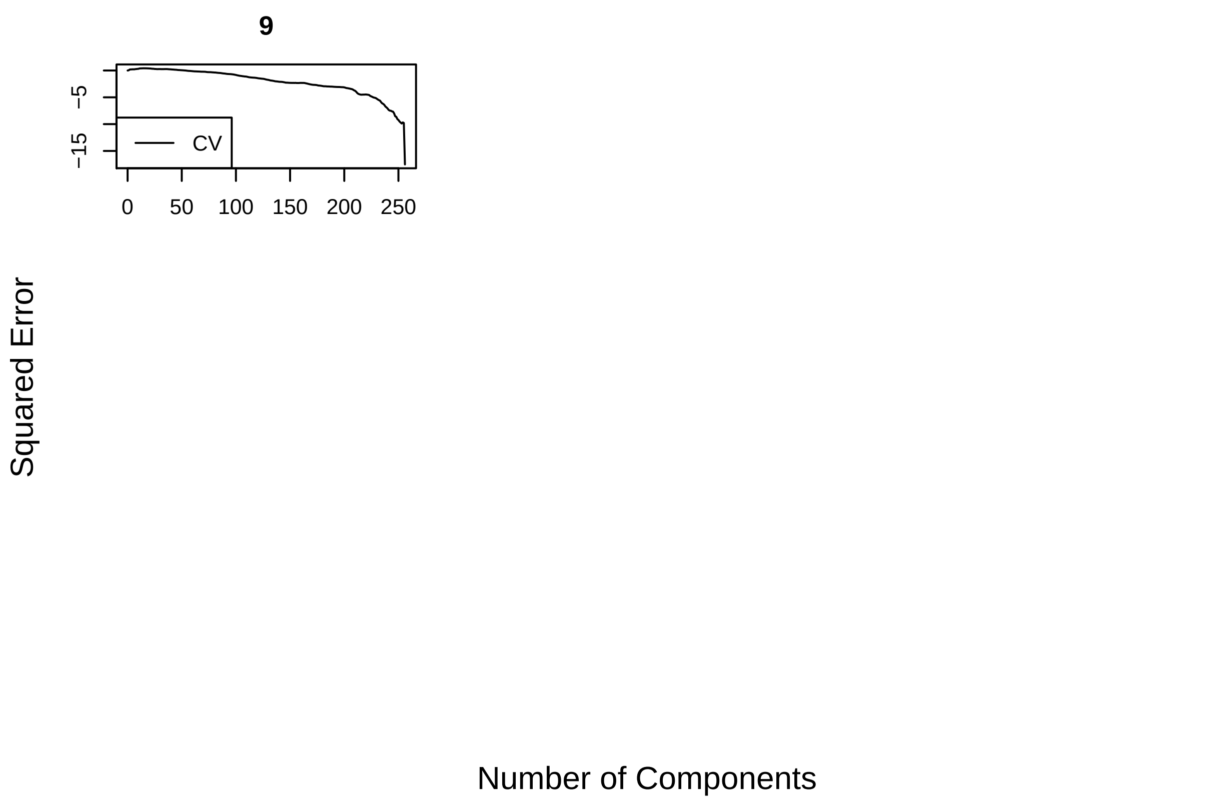
I decided to use cross-validation as it computes way faster than Leave-One-Out.

```
pls1 <- plsr(Y.train ~ X.train, validation="CV")


# Plot the components importance
plot(R2(pls1), legendpos = "bottomleft", main = 'R2 vs Number of Components',xlab = 'Number of Componen
```

R2 vs Number of Components



R2 vs Number of Components

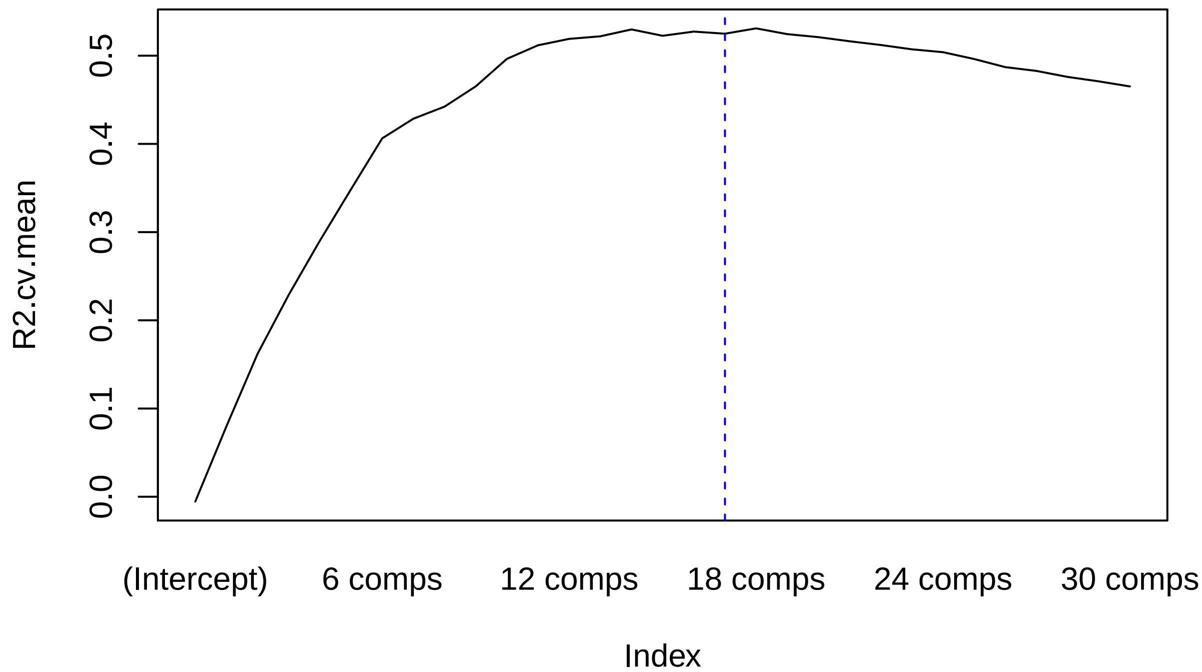


We observe this is getting worse as we take more and more components. Then **we can zoom more**. Let's zoom to 30 components.

```r
pls2 <- plsr(Y.train ~ X.train, ncomp=30, validation="CV")



R2.cv <- R2(pls2)$val[1,,]
#head(R2.cv)

R2.cv.mean <- apply(R2.cv,2,mean)
R2.cv.mean
```

```
##  (Intercept)      1 comps      2 comps      3 comps      4 comps
## -0.005517231  0.079841449  0.162021674  0.228885475  0.290281403
##      5 comps      6 comps      7 comps      8 comps      9 comps
##  0.348603522  0.406296772  0.428533499  0.442255144  0.465184780
##     10 comps     11 comps     12 comps     13 comps     14 comps
##  0.496371771  0.511800812  0.519064972  0.521962593  0.529764766
##     15 comps     16 comps     17 comps     18 comps     19 comps
##  0.522558745  0.527322607  0.524933255  0.530960706  0.524425126
##     20 comps     21 comps     22 comps     23 comps     24 comps
##  0.521027434  0.516352489  0.512122481  0.507220923  0.503941816
##     25 comps     26 comps     27 comps     28 comps     29 comps
##  0.496200763  0.487026640  0.482721126  0.475906143  0.470879013
##     30 comps
##  0.465152303
```

```r
plot(R2.cv.mean,type="l",xaxt="n")
axis(1,at=1:ncol(R2.cv),lab=colnames(R2.cv),tick=FALSE)

nd <- which.max(R2.cv.mean)-1
abline(v = nd, lty=2, col="blue")
```




```
print(paste("Number of selected components:", nd))
```

```
## [1] "Number of selected components: 18"
```

Then, **we select 18 components** here as the mean of the R2 value is the greatest. Let's see now how much of the variance is explained with this number of components:

```
var.exp <- rep(0, nd)
curr <- 0

for(i in 1:nd)
{
  curr <- curr + pls2$Xvar[i]
  var.exp[i] = curr/pls2$Xtotvar
}
var.exp <- var.exp*100

print(paste("Variance explained with ", nd," components: ", round(var.exp[nd]*100)/100, "%", sep=""))
```

```
## [1] "Variance explained with 18 components: 69.73%"
```

4. Predict the responses in the test data, be aware of the appropriate centering.

```
X.test <- scale(X.test, center = X.train.means, scale=FALSE)
test.proj <- as.matrix(X.test) %*% pls2$projection[, 1:nd]

train.pls.data <- data.frame(pls2$scores[,1:nd])
model <- lm(Y.train~., data=train.pls.data)
pred.test.prob <- predict(model, data.frame(test.proj) , type="response")
```

Compute the average R2 in the test data.

Lol, dunno that shit.

5. Assign every test individual to the maximum response and compute the error rate.

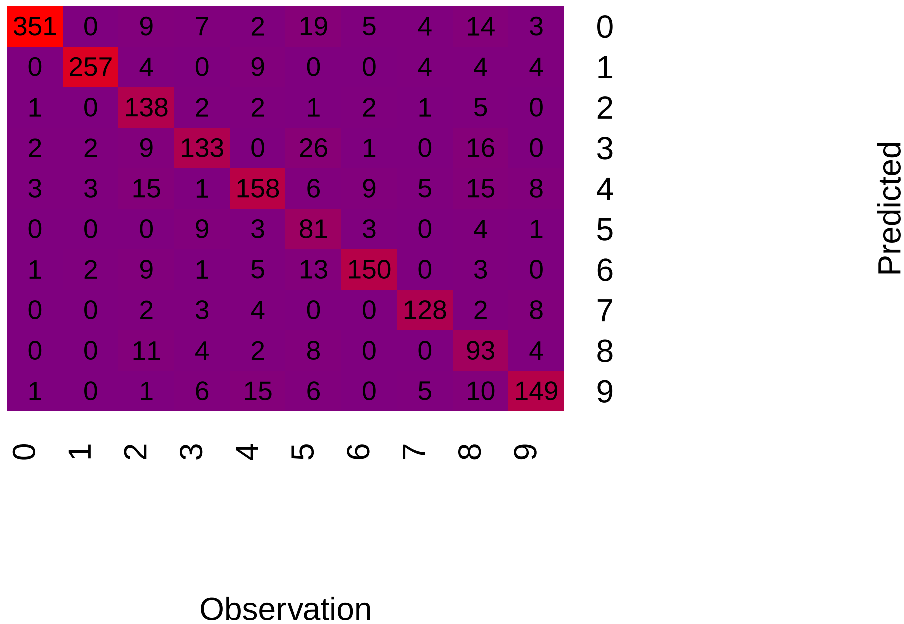
```
pred.test.numbers <- c()

n.test <- nrow(pred.test.prob)
for(i in 1:n.test)
{
  pred.test.numbers[i] <- unname(which.max(pred.test.prob[i,])-1)
}

(pred.acc <- mean(pred.test.numbers == test[,1])) #accuracy
```

```
## [1] 0.8161435
```

```
(pred.err <- 1-pred.acc) #error rate
```

```
## [1] 0.1838565
```

We obtained this confusion matrix:



Comparison with previous works

Using PCR, we had around **79%** of accuracy. With IBA, we had around **80%** of accuracy. Now, we have nearly **82%**, which is better.

Conclusion

We can observe that our model here is slightly better than previous work. Moreover, PLS2 algorithm seems really powerful as it reduces here a dataset of 256 variables to 18 components with the most variability.