PLS1 Practical Work : Leukemia

Multivariate Modeling

Martin Guy

December 21, 2016

FIB UPC UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Introduction

Although cancer classification has improved over the past 30 years, there has been no general approach for identifying new cancer classes (class discovery) or for assigning tumors to known classes (class prediction). The problem focuses in finding a classifier using Partial Least Squares (PLS1) regression approach on gene expression monitoring by DNA microarrays, to automatically differentiate between acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL). PLS1 will give the **principal components** our the dataset.

We will work on two datasets : the training set `data_set_ALL_AML_train.csv` with 38 samples and the test set `data_set_ALL_AML_independent.csv` with 34 samples. These datasets contain measurements corresponding to ALL and AML samples from Bone Marrow and Peripheral Blood.

1. Read the data files, enter the response manually into R

```
train <- read.csv("data_set_ALL_AML_train.csv", header = TRUE, sep = ";")
test <- read.csv("data_set_ALL_AML_independent.csv", header = TRUE, sep = ";")
```

The responses are located in the `table_ALL_AML_predic.doc` file, in the column "actual". There are two factors: `ALL` and `AML`. We will map them respectively to `0` and `1`. In this file, there are 27 `ALL` in the training data and 20 in the testing data. Rest is `AML`.

```
train.response <- append(rep(0,27), rep(1,11))
test.response <- append(rep(0,20), rep(1,14))


#Only numeric information is pertinent to solve the problem.
train <- select(train, contains("X"))
test <- select(test, contains("X"))
```

2. Form the data matrices X and Xt

```
X <- data.frame(t(train))
Xt <- data.frame(t(test))


rownames(X)
```

```
##  [1] "X1"  "X2"  "X3"  "X4"  "X5"  "X6"  "X7"  "X8"  "X9"  "X10" "X11"
## [12] "X12" "X13" "X14" "X15" "X16" "X17" "X18" "X19" "X20" "X21" "X22"
## [23] "X23" "X24" "X25" "X26" "X27" "X34" "X35" "X36" "X37" "X38" "X28"
## [34] "X29" "X30" "X31" "X32" "X33"
```

```
rownames(Xt)
```

```
##  [1] "X39" "X40" "X42" "X47" "X48" "X49" "X41" "X43" "X44" "X45" "X46"
## [12] "X70" "X71" "X72" "X68" "X69" "X67" "X55" "X56" "X59" "X52" "X53"
## [23] "X51" "X50" "X54" "X57" "X58" "X60" "X61" "X65" "X66" "X63" "X64"
## [34] "X62"
```

We can observe that the labels are not exactly in order, but no need to reorder. Indeed, it corresponds well to the `table_ALL_AML_predic.doc` file.

We do not forget to center (but not scale) the data.

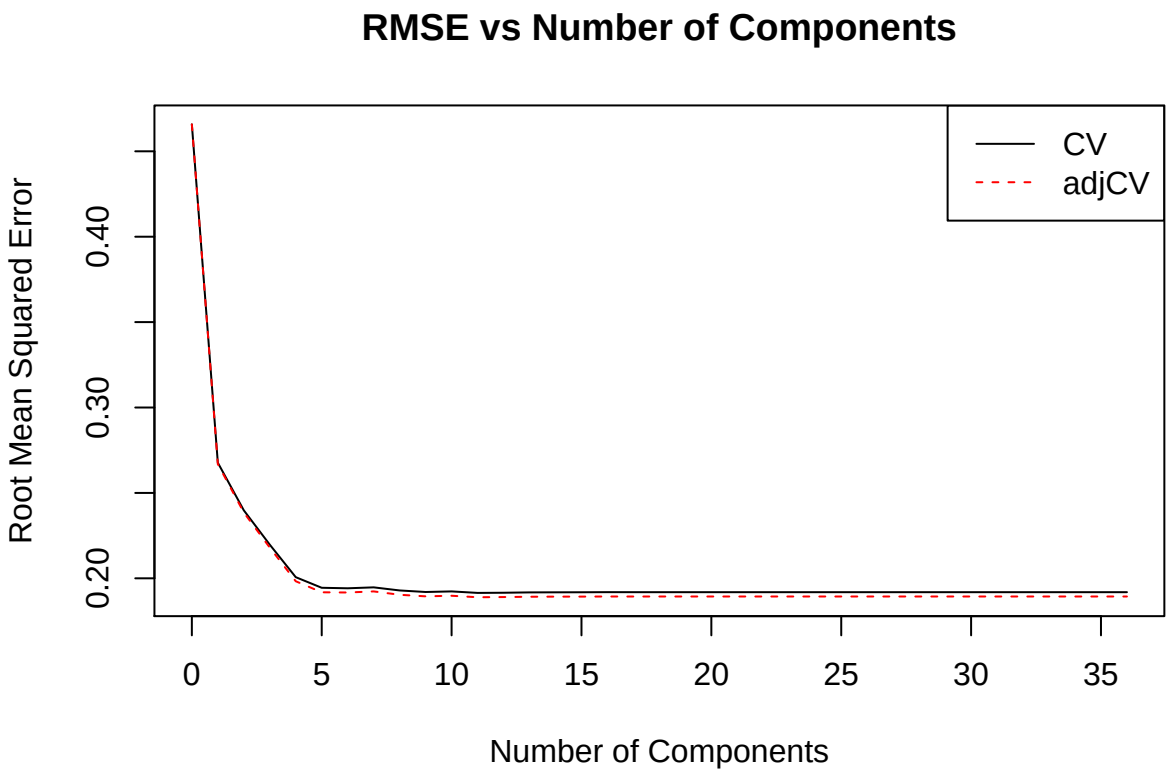
```
# We center but not scale the data
means = colMeans(X)
X <- data.frame(scale(X,scale = F))
Xt <- data.frame(scale(Xt,center=means,scale=F))

# Trainging and Testing dataset
training <- as.data.frame(cbind(train.response,X))
testing <- as.data.frame(cbind(test.response,Xt))
```

3. Perform the PLS1 regression of the training data

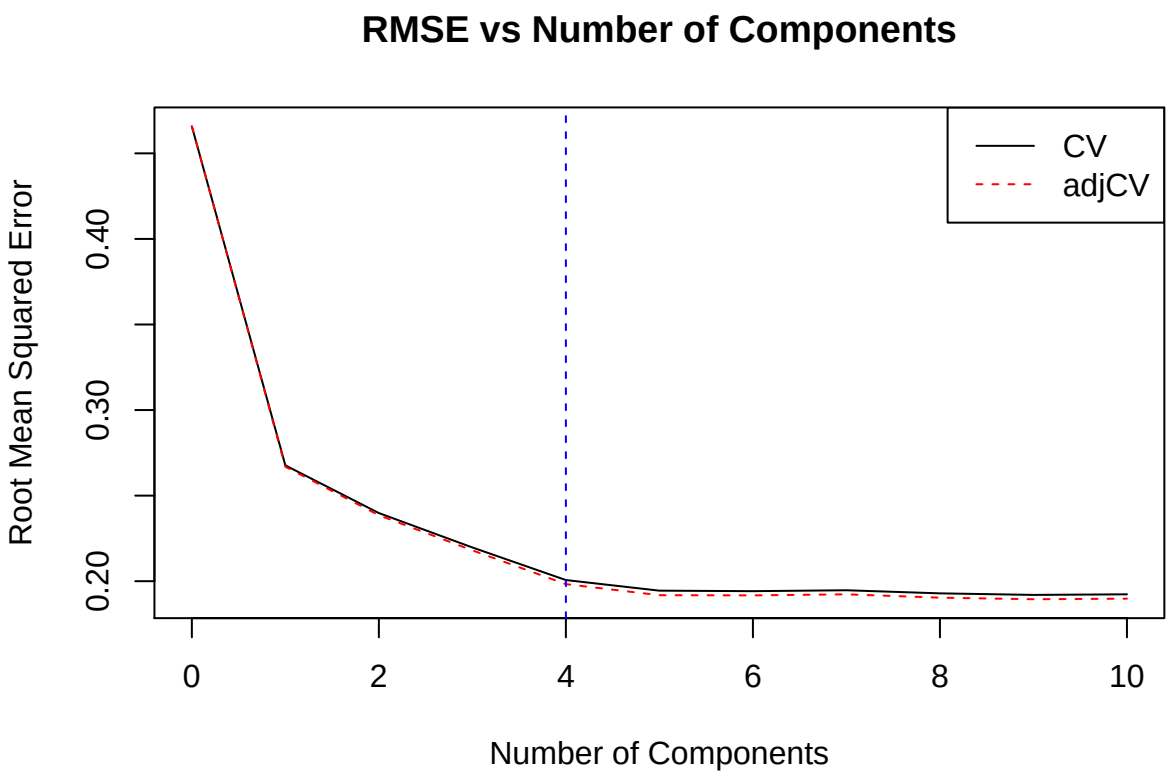
```
pls1 <- plsr(train.response ~ ., data = training, validation="LOO")

# Plot the components importance
plot(RMSEP(pls1), legendpos = "topright", main = 'RMSE vs Number of Components',xlab = 'Number of Compo
```



We observe there is not much improvement if we take more than 10 components. Then **we can zoom more**.

```
pls2 <- plsr(train.response ~ ., ncomp=10, data = training, validation="LOO")
plot(RMSEP(pls2), ncomp, legendpos = "topright", main = 'RMSE vs Number of Components',xlab = 'Number o
ncomp <- 4
abline(v = ncomp, lty=2, col="blue")
```



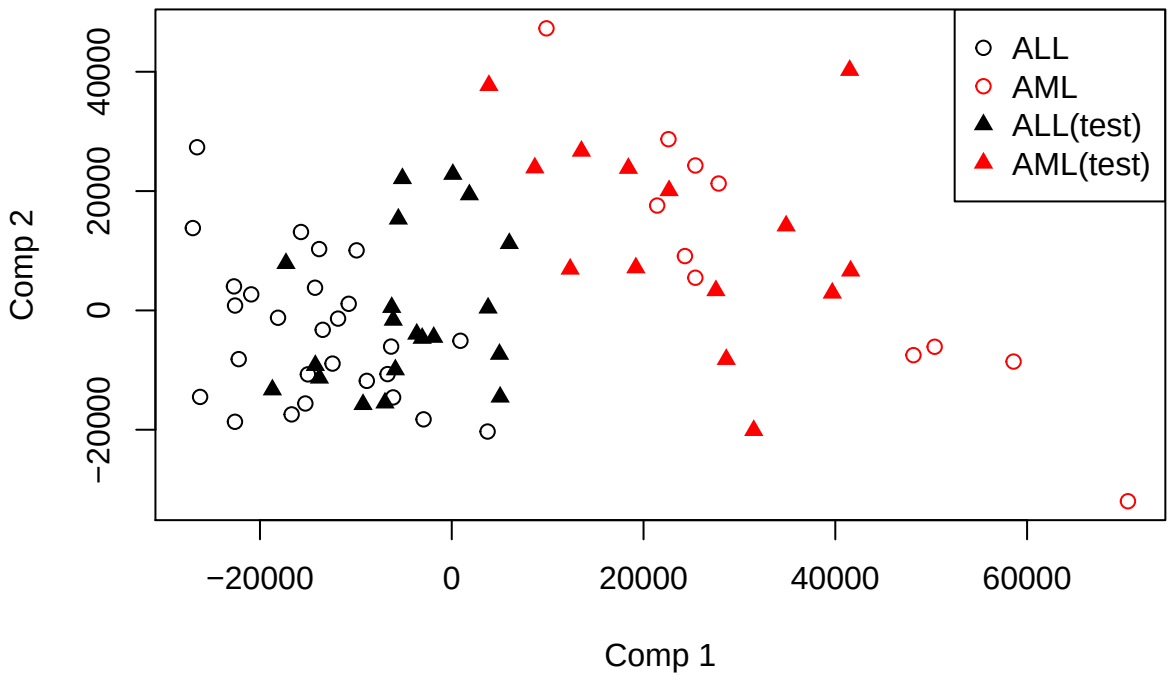
We see that after taking 4 components, the RMSE does not decrease that much if we keep taking more components. So **we select the first four components**.

4. Project the test data as supplementary individuals onto the selected PLS1 components

```
test.proj <- as.matrix(Xt) %*% pls2$projection[, 1:ncomp]
```

5. Perform a joint plot of the train and test individuals in the plane of the two first PLS1 components

```
plot(pls2$scores[, 1:2], col = as.factor(train.response))
points(test.proj[,1:2], pch=17, col=as.factor(test.response))
legend("topright",c("ALL","AML","ALL(test)","AML(test)"), pch=c(1,1,17,17),col=c("black","red","black",
```



On the training data, `ALL` and `AML` are well separated. On the testing data they are less separated.

```
R2(pls2)
```

```
## (Intercept)      1 comps      2 comps      3 comps      4 comps
##    -0.05478      0.65159      0.72039      0.76530      0.80421
##     5 comps      6 comps      7 comps      8 comps      9 comps
##     0.81612      0.81669      0.81567      0.81909      0.82079
##    10 comps
##     0.82013
```

And in fact, we see that we do not explain most of the variance that we can as we only use two components.

6. Obtain the logistic regression model to predict the response in the training data

```
train.pls.data <- data.frame(pls2$scores[,1:4])

model <- glm(as.factor(train.response) ~ .,family=binomial(link='logit'),data=train.pls.data)
```

```
pred.train <- unname(predict(model, train.pls.data))
pred.train <- as.numeric(pred.train > 0.5)

acc.train <- sum(pred.train==train.response)/length(pred.train)
paste("Accuracy on training data:", acc.train)
```

```
## [1] "Accuracy on training data: 1"
```

We have an **accuracy of 100%** on the training data. While this is a perfect score, this may be a sign of **overfitting**. It is more accurate then to see the accuracy on test data.

7. Predict the probability of AML leukemia in the test sample.

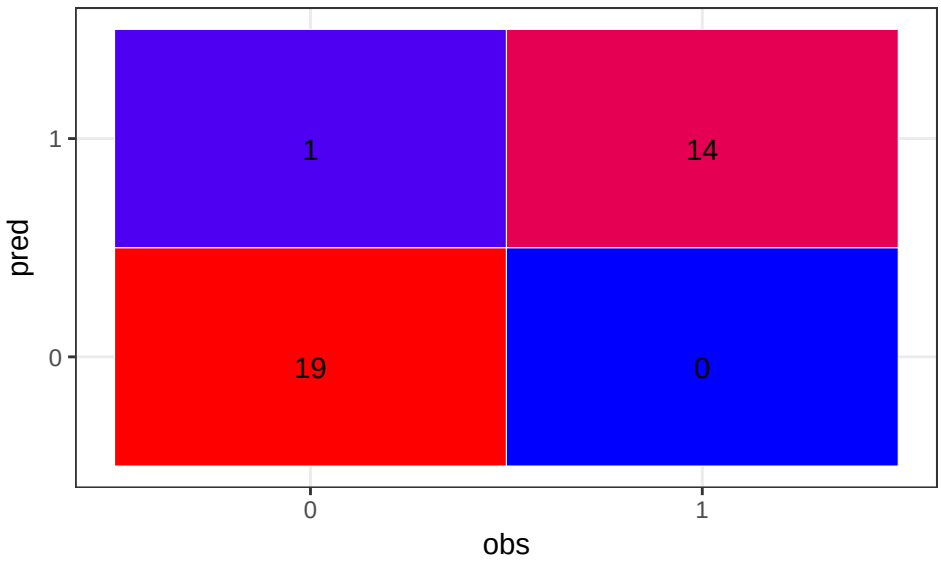
```
pred.test.prob <- predict(model, data.frame(test.proj) , type="response")
pred.test <- unname(pred.test.prob)
pred.test <- as.numeric(pred.test > 0.5)

acc.test <- sum(pred.test==test.response)/length(pred.test)
paste("Accuracy on testing data:", acc.test)
```

```
## [1] "Accuracy on testing data: 0.970588235294118"
```

We obtain here a really good accuracy on the test dataset, even though the sample size is really small.

We can take a look at the confusion matrix:



Now let's finish with predicting the probability of AML leukemia in the test sample. We will compare them with the **prediction strength** (denoted by `PS`) from the `table_ALL_AML_predic.doc` file. These prediction strength have been put in the `paperProbAML.txt` file in order to load them easily.

```
##     test.response pred.test.prob   PS
## X50           AML         1.0000 0.97
## X51           AML         1.0000 1.00
## X52           AML         0.9998 0.61
## X53           AML         1.0000 0.89
## X54           AML         0.9997 0.23
## X57           AML         1.0000 0.22
## X58           AML         1.0000 0.74
## X60           AML         1.0000 0.06
## X61           AML         1.0000 0.40
## X62           AML         0.9999 0.58
## X63           AML         1.0000 0.69
## X64           AML         1.0000 0.52
## X65           AML         1.0000 0.60
## X66           AML         1.0000 0.27
```

Conclusion

We can observe that our model here is really accurate thanks to Partial Least Square Algorithm, and also **really sure** about the predictions. Nevertheless, this is most probably due to overfitting. It could be indeed better to try with a much larger dataset, and for instance perform cross-validation to evaluate this model. Even though, PLS1 algorithm seems really powerful as it reduces here a dataset of 7130 variables to 4 components with the most variability.