# Principal Component and Partial Least Squares Regression

*Krishna Kalyan*

## Introduction

The aim to this assignment was to automatically differentiate between acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL). Our training data has `7129` observations and `78` attributes and our test set has `7129` observations and `70` attributes. Our targets need to be manually extracted from `table_ALL_AML_predic.doc`.

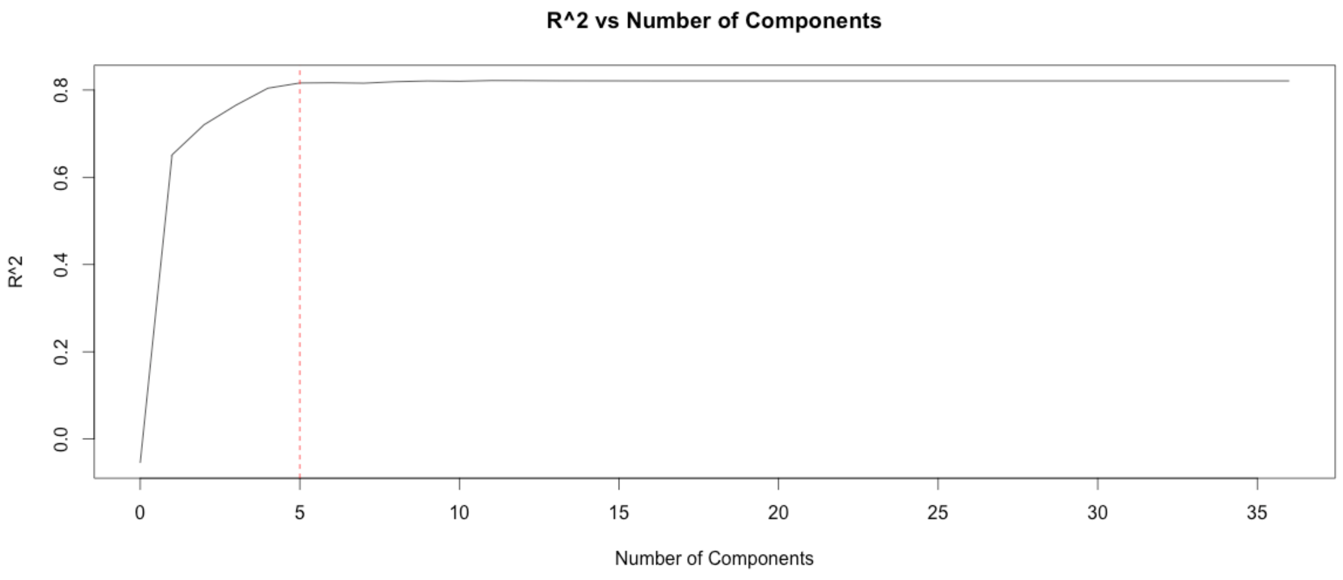
## Pre-Processing

Our data has several pre-processing steps to prepare out final dataset (Our dataset includes `tranining and test` set).

- Extract all numberical attributes from our dataset.
- Transpose numerical attributes.
- Ordering our dataset based on individuals in ascending order.
- Centering our trainig data and test data (with respect to mean of our training data). (Note: We do not scale our dataset as we want to retain variability in our attributes)
- Appending targets extracted from `table_ALL_AML_predic.doc` to our sorted dataset.

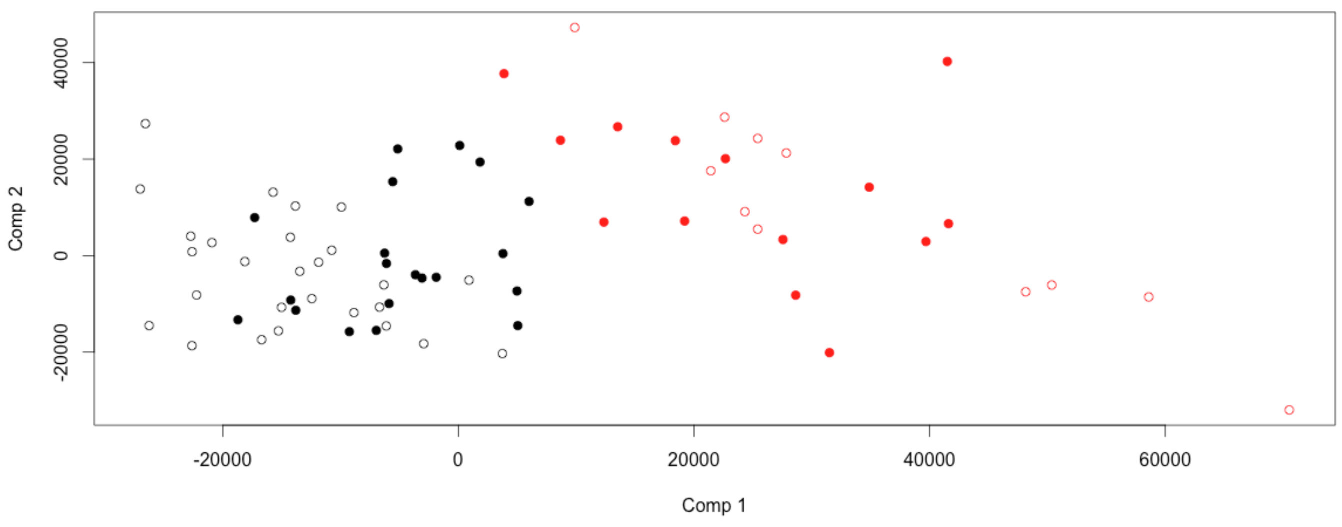
## Component Selection

Using the PLS packge we use `plsr` method train our model using `LOO` (Leave One Out) cross validation. We choose the components based on `R^2` observed. As observed in the plot below, we see that with `5` components fit our model well.



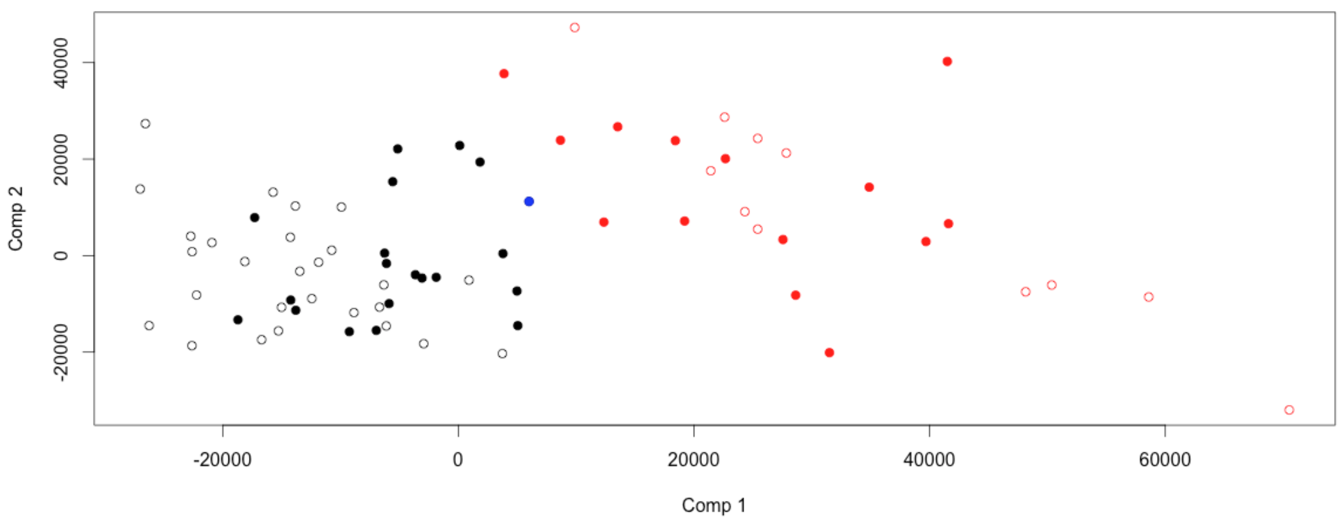
## Projection

We project the first two components of our training set and test set (as supplementary individuals) in the joint plot below. We see observe that first two components seperate our classes well (I am not a domain expert) (Dots coloured are `test dataset`). However its important to know that we are projecting only the first two components, If we could poject more components and observe, I am sure we could have a better seperation of classes and data.



## Conclusions

Using the `predict` method in the PLS function we observe a test set accuracy of `0.97` and obsercing the confusion matrix we observe that only one point was mis-classified. With `glm` model we observe almost the same test set accuracy of `0.97` and one point mis-classified. Observation number `43` was misclassifed in the test-set by both the models (This can be observed by the blue dot below).



We can see why that point was misclassified based on the plot above. It has values that represented both the classes in the test set. If we observed carefully we see that it is right in the middle (almost) of our training set class seperation.

# Code

```r
rm(list=ls())
library(doMC)
library(dplyr)
library(pls)
library(plsdepot)
registerDoMC(8)

setwd('/Users/krishna/MIRI/MVA/Octane_PLS/')
source('targets_prep.R')

# Preperare Training Data
train = read.csv('exer_leukemia/data_set_ALL_AML_train.csv', sep=';')
train_num = select(train,starts_with("X"))
train_t = data.frame(t(train_num))
rownames(train_t) = as.integer(substring(rownames(train_t),2))
train_prep = train_t[order(as.integer(rownames(train_t))),]
train_prep_center = data.frame(scale(train_prep, center = T, scale = F) )
train_prep_center$target = target_train

# Prepare Test data
test = read.csv('exer_leukemia/data_set_ALL_AML_independent.csv', sep=';')
test_num = select(test,starts_with("X"))
test_t = data.frame(t(test_num))
rownames(test_t) = as.integer(substring(rownames(test_t),2))
test_prep = test_t[order(as.integer(rownames(test_t))),]
test_prep_center = data.frame(scale(test_prep,
                                    center = colMeans(train_prep),
                                    scale = F))

test_prep_center$target = target_test

# Model
model1 = plsr(target ~ ., data=train_prep_center, validation = "LOO")

plot(R2(model1), main = 'R^2 vs Number of Components',
     xlab = 'Number of Components', ylab = 'R^2')
abline(v=5, cex=0.9,col='red',lty = 2)

# Test data Projection
test_projection = as.matrix(test_prep_center[,-c(7130)]) %*% model1$projection[,1:5]
plot(model1$scores[,1:2], col = as.factor(target_train))
points(test_projection[,1:2], pch=19, col = as.factor(target_test))

source('Util.R')

# Evaluation
yhat_pred = predict(model1,test_prep_center)
yhat_comp = data.frame(yhat_pred)
yhat = yhat_comp[,5] >= .5
y = test_prep_center$target
eval_pls = eval_func(y,yhat, cm_show = T)

# Logist Regression Model
train_glm = data.frame(cbind(model1$scores[,1:5],target_train))
names(train_glm) = c(paste0('C.',rep(1:5)),'target')

train_glm$target = as.factor(train_glm$target)
```

```r
test_glm = as.data.frame(test_projection)
names(test_glm) = paste0('C.',rep(1:5))
model2 = glm(target ~. , data=train_glm, family="binomial")
summary(model2)
yhat = predict(model2, test_glm, type='response') > .5
eval_glm = eval_func(y,yhat, cm_show = T)


test_projection = as.matrix(test_prep_center[,-c(7130)]) %*% model1$projection[,1:5]
plot(model1$scores[,1:2], col = as.factor(target_train))
points(test_projection[,1:2], pch=19, col = as.factor(target_test))
points(test_projection[5,1],test_projection[5,2],pch=19, col = 'blue')
```

```r
# Train
all = rep(0,27)
aml = rep(1,11)
target_train = c(all,aml)

# Test
all1 = rep(0,((49-39) + 1))
aml1 = rep(1,((54-50) + 1))
all2 = rep(0,((56-55) + 1))
aml2 = rep(1,((58-57) + 1))
all3 = rep(0,((59-59) + 1))
aml3 = rep(1,((66-60) + 1))
all4 = rep(0,((72-67) + 1))
target_test = c(all1,aml1,all2,aml2,all3,aml3,all4)
```

```r
eval_func = function(y, yhat, cm_show = FALSE){
  metrics = c()
  cm = table(y,yhat)
  if(cm_show == TRUE){
    print(cm)
  }
  total = sum(cm)
  no_diag = cm[row(cm) != (col(cm))]
  acc = sum(diag(cm))/total
  error = sum(no_diag)/total
  metrics = c(acc,error)
  return(metrics)
}
```