Partial Least Squares 2

Krishna Kalyan

Introduction

The purpose of this exercise is to use PLSR2 as a component based methodology to predict the digits. PLS2 refers to the situation when the response block Y has more than one variable. It is an iterative algorithm with allows missing data. In this exercise I will not compare my results with Multivariate Regression, Principal Components Regression and IBA as my results were partially correct.

Pre-Processing

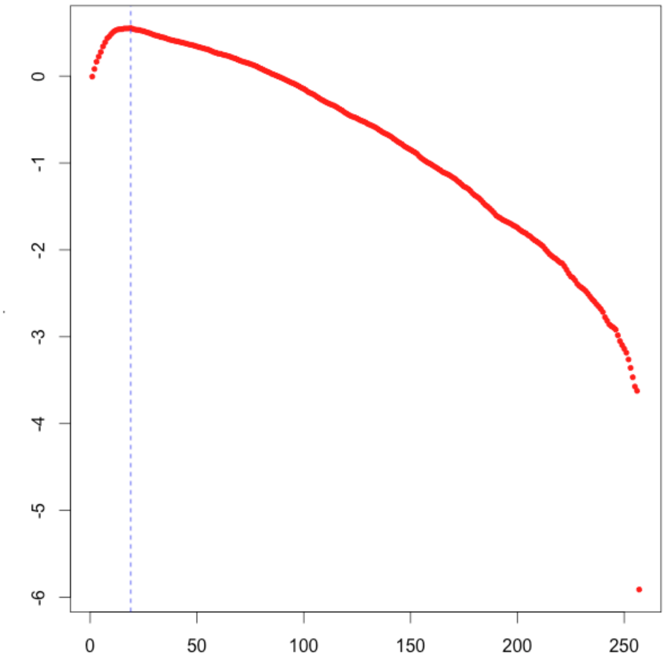
The pre-processing steps have been described below

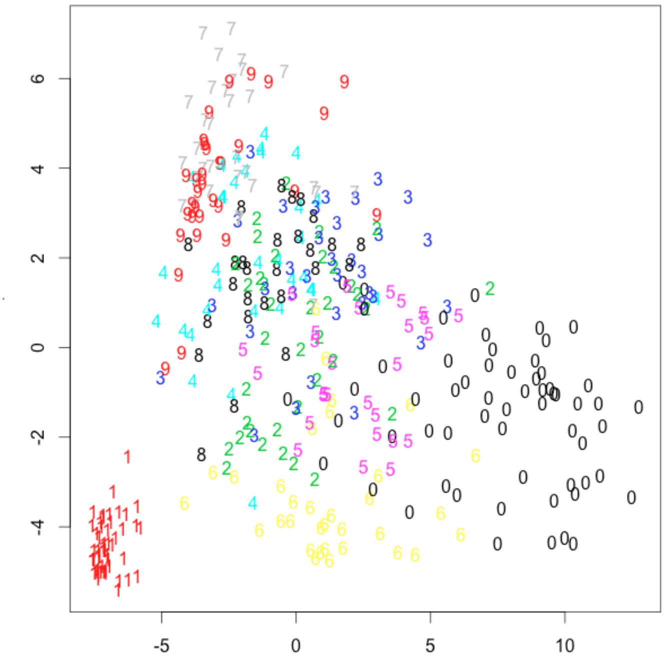
- Sample 5% training data.
- Center training data.
- Center test data with respect to the column mean of training data set.
- Dummmy code class labels.

Partial Least Squares

Using PLS2 we choose `19` components based on `R^2` value obtained by cross validation on the training set. This can be observed in the plot below. We plot first `2` compoments, We observed that data well seperates the digits `0` and `1` better than other digits.

We observed that that our training accuracy to be `0.96`. This was kind of suspicious. On evaluating our model on test dataset we observe our accuracy to be lower as expected `0.79`.





Conclusions

Compared to what we have seen, that `PLS2` like other algorithms do well when we have small datasets. It would be difficult to compare it to other methods we observe as we can expect high variance in small subsets of data. We also observed that `LDA` model learns better representaion on training data with `PLS2` components.

Thank you professor for the classes.

Code

```r
library(caTools)
library(data.table)
library(pls)
library(CatEncoders)
source('../Assignment PLS2/utils.R')
set.seed(1337)
par(mfrow=c(1,2))

# 1 Read and choose 5 percent
setwd('/Users/krishna/MIRI/MVA/zip_data')
train = fread("zip_train.dat", showProgress = TRUE)
test = fread("zip_test.dat", showProgress = TRUE)

split = sample.split(train$V1, 0.05)
ts = data.frame(subset(train, split ==T))

# 2 Define the response matrix (Y) and the predictor matrix (X) + Center Predictor
X = ts[,-1]
X = scale(X, center = T, scale = F)
y = matrix(ts$V1)

# Prepare y OHE
ohe_model = OneHotEncoder.fit(y)
y_ohe = as.matrix(transform(ohe_model,y))
y_ohe = as.data.frame(y_ohe)
names(y_ohe) = c("C0","C1","C2","C3","C4","C5","C6","C7","C8","C9")

# 3 Perform a PLSR2 using CV for validation + Components to retain
train_data = data.frame(X,y_ohe)
formula1 = cbind(C0,C1,C2,C3,C4,C5,C6,C7,C8,C9) ~ .
model1 = plsr(formula1, data=train_data, validation = "CV")
r2_cv = R2(model1)$val[1,,]
r2_mean = apply(r2_cv, MARGIN = 2, mean)
max_comp = which.max(r2_mean)
plot(1:length(r2_mean),r2_mean, cex=.6, pch =19, col='red', ylab = "RSquare", xlab="C
omponents")
abline(v=max_comp, col='blue', lty = 2)
max_comp

# 19 Components to retain
comps_fix = data.frame(model1$scores[,1:max_comp])
train_data_comp = data.frame(y_ohe,comps_fix)
model2 = lm(formula1, data = train_data_comp)
y_tr = predict(model2,comps_fix)
ytr_class = data.frame(unname(apply(y_tr, 1, which.max)) - 1)
eval1 = eval_func(unlist(y),unlist(ytr_class), cm=T)
# Training Accuracy - 0.96
# Error - 0.04

# 4 Predict the responses in the test data
X_test = data.frame(test)[,-1]
X_test = scale(X_test, scale=F, center = colMeans(X))
X_test_proj = data.frame(as.matrix(X_test) %*% model1$projection[,1:max_comp])
Y_test = matrix(test$V1)

# 5 Compute Error
y_ohe_test = as.matrix(transform(ohe_model,Y_test))
y_ohe_test = as.data.frame(y_ohe)
names(y_ohe_test) = c("C0","C1","C2","C3","C4","C5","C6","C7","C8","C9")
Yhat = predict(model2,X_test_proj)
Yhat_class = data.frame(unname(apply(Yhat, 1, which.max)) - 1)
eval2 = eval_func(unlist(Y_test),unlist(Yhat_class), cm=T)
# Test Accuracy - 0.79
# Error - 0.21

# Training Plot
plot(comps_fix[,1:2], pch=3, col='white', xlab='Comp1', ylab='Comp2')
text(comps_fix[,1:2], labels=y, col=as.numeric(as.factor(y)))

# Training Error with LDA
model3 = lda(model1$scores[,1:max_comp],y,CV=T)
eval3 = eval_func(y,model3$class, cm_show = T)
eval3
# 0.95616438 0.04383562

model4 = lda(X, y, CV=T)
eval4 = eval_func(y,model4$class, cm_show = T)
eval4
# 0.5260274 0.4739726
```

```
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)
}
```