Inter Batteries Analysis

Krishna Kalyan

Introduction

The aim of this assignnment is to perform the Inter Batteries Analysis using linear algebra and matrix operation. We will also need to conduct expriments to decide the number of components. After a comparison study with other alogrithms we will conclude.

Inter Battries Analysis

The goal of IBA is to measure the relationship between both multivariate vectors by components derived from original variables. IBA is a compromise between CCA and PCA and its components are not orthogonal. Covaraiance matrix between the targets and the predictors are computed. This is followed by computing the eigen values (aib). `A` is calculated as given below. `B` is not required to be calculated. Inner product of the predictor matrix and `A` gives us `T` matrix, `U` is not required to be claculated.

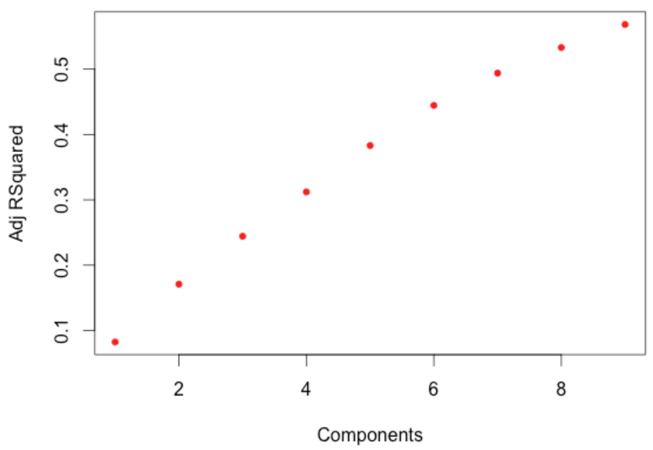
```
A = Vxy %*% aib$vectors %*% diag(aib$values^(-0.5))
```

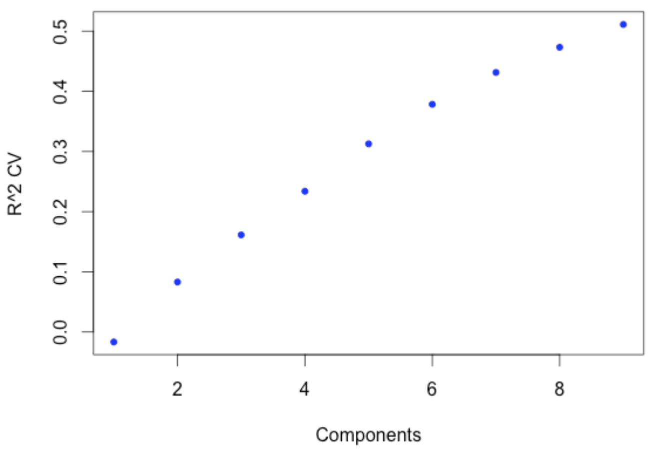
Data Analysis

We need to center the data (Only the predictors). I did this by stacking training and test set followed by centing. Since our data is centered we will need to remove the intercept. We should also take care of co-linearity as our covariance matrix has a lower rank. (This is because our dummy targets have redundant column).

Experiments

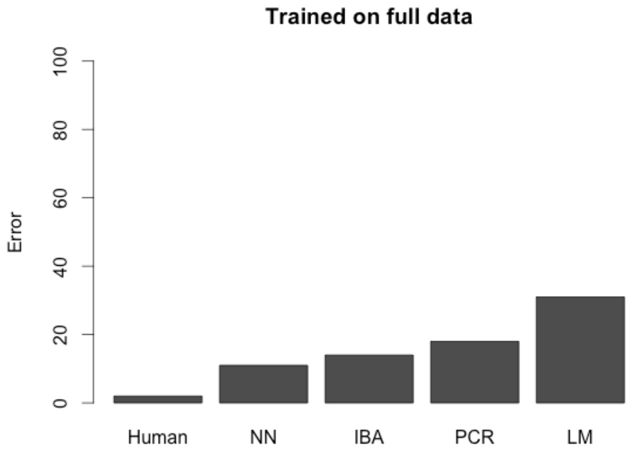
To decide how many componets to take, we conducted experiments by varying number of components with respect to Adjusted R^2 and the value of approximate cross validation calculated using the `PRESS` stastic. Observing the plot below we can see that the the best fit was obtained when we have `9` components.





Conclusion

Based on our experiments we get the plots below (Lower the better). These experiemtns were conducted by training on training set and evaluating error on the test set. We can observe that Inter Batteries Analysis (IBA) + Regression is slightly better than Principal Component Regression (PCR) model. Also the performance of the neural network on IBA components did not do as well as IBA with regression. The conclusions above hold true when the data set size is small and the number of dimensions are large.



** Note

- NNIBA - Error value obtained was not tested on test dataset (We represent training error here). We can expect this error to incerase by 2 - 3 percent. (IBA + NN).
- Human - Human accuray is debatable as digit recorgnition can vary from person to person.

Code

```r
rm(list=ls())

library(plsdepot)
library(CatEncoders)
require(caTools)

setwd('/Users/krishna/MIRI/MVA/Assignment2')

train = read.csv('../zip_data/zip_train.dat',header = F, sep=" ")
test = read.csv('../zip_data/zip_test.dat',header = F, sep=" ")

head(train)

# As V258 has NAs
remove_col = c('V258')
train = train[,!names(train) %in% remove_col]
test = test[,!names(test) %in% remove_col]
ytrain = data.frame(train$V1)
ytest = data.frame(test$V1)

split = sample.split(train$V1, 0.05)
table(split)
# Stack data and center
remove_col = c('V1')
data = rbind(train,test)
data = scale(data, center = TRUE, scale = FALSE)

# Split Data
trainIB = data[1:nrow(train),]
testIB = data[-c(1:nrow(train)),]
split = sample.split(train$V1, 0.05)


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

# IBA Manual Train
source('Util.R')

trainIBs = subset(trainIB, split ==T)
trainy = subset(ytrain_ohe, split ==T)

TrainIB = interbatt(trainIBs,trainy)[[1]][,1:9]
AdjRsq =  matrix(9)
R2CV = matrix(9)

# When data is centered we need to remove the intercept
formula1 = cbind(C0,C1,C2,C3,C4,C5,C6,C7,C8,C9) ~  0 + .

n = nrow(TrainIB)
for(i in (1:9)){
  train_data = data.frame(TrainIB[,1:i], trainy)
  model1 = lm(formula1, data=train_data)
  adjr = sapply(summary(model1), function(x){x$adj.r.squared})
```

```
  AdjRsq[i] = mean(adjr)
  PRESS  = apply((as.data.frame(model1$residuals)/(1-ls.diag(model1)$hat))^2,2,sum)
  RMPRESS = sqrt(PRESS/n)
  R2cv_denom  = apply(trainy,2,var)*(n-1)
  R2cv = 1 - PRESS/R2cv_denom
  rcv =mean(R2cv)
  R2CV[i] = rcv
}

plot(1:9,AdjRsq,pch=19,col="red",cex=.7,xlab="Components",ylab="Adj RSquared")
plot(1:9,R2CV,pch=19,col="blue",cex=.7,xlab="Components",ylab="R^2 CV")

# We choose 9 components based on training R square error 0.54
train_data = data.frame(TrainIB[,1:9], trainy)
model1 = lm(formula1, data=train_data)

# IBA Test
A = interbatt(trainIBs,trainy)[[2]][,1:9]
TestIB = (as.matrix(testIB) %*% A)
yhat = predict(model1,data.frame(TestIB))
Yhat = data.frame(unname(apply(yhat, 1, which.max)) - 1)
eval_func(unlist(ytest),unlist(Yhat),cm_show = T)

# IBA on Full Data
TrainIB = interbatt(train,ytrain_ohe)[[1]][,1:9]
train_data = data.frame(TrainIB, ytrain_ohe)
model2 = lm(formula1, data=train_data)
adjr = sapply(summary(model1), function(x){x$adj.r.squared})

A = interbatt(train,ytrain_ohe)[[2]][,1:9]
TestIB = (as.matrix(testIB) %*% A)
yhat = predict(model2,data.frame(TestIB))
Yhat = data.frame(unname(apply(yhat, 1, which.max)) - 1)
eval_func(unlist(ytest),unlist(Yhat),cm_show = T)


# Prepare data for NN
X =  interbatt(trainIBs,trainy)[[1]][,1:9]
yindex = rownames(trainy)
y = ytrain[yindex,]
exportNN = data.frame(X,y)
head(exportNN)
write.table(exportNN,"opNN.csv", row.names=FALSE,sep=",",quote=F)
```

```r
interbatt = function(X,Y){
  Vxy = var(X,Y)
  rank = qr(Vxy)$rank
  print(paste("Rank of CoVar Matrix ",rank))
  aib = eigen(t(Vxy)%*%Vxy)
  A = Vxy %*% aib$vectors %*% diag(aib$values^(-0.5))
  TIB = as.matrix(X) %*% A
  rl = list(TIB, A)
  return(rl)
}

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)
}

plot_img = function(train_data,i,show_target=FALSE){
  CUSTOM_COLORS = colorRampPalette(colors = c("black", "white"))
  if(show_target==TRUE){
    print(train_data[i,1])
  }
  train_data = train_data[,-1]
  z = unname(unlist((train_data[i,])))
  k = matrix(z,nrow = 16,ncol = 16)
  rotate <- function(x) t(apply(x, 2, rev))
  image(rotate(t(k)), col = CUSTOM_COLORS(256))
}
```