HW3 : PCA and PLS Regression

Avantika Diwadkar

10/6/2022

#Libraries

```r
library(glmnet)
library(dplyr)
library(reshape2)
library(tibble)
library(pander)
```

Read in the gene expression and genotyping datasets

```r
#Read in the data
norm_expr <- read.table("~/Documents/Penn_State/Fall2022/HW1/Data/GEUVADIS_normalized_expressio

genot_df <- read.table("~/Documents/Penn_State/Fall2022/HW1/Data/GEUVADIS_chr20_processed.traw
```

Clean the dataset

```r
#Filter data for 1 gene
gene1_exp <- as.data.frame(norm_expr[1,])
gene1_vars <- genot_df %>%
  dplyr::filter(POS >= gene1_exp$start - 500000, POS <= gene1_exp$start + 500000) %>%
  dplyr::select(-CHR, -X.C.M, -POS, -COUNTED, -ALT)

#Wrangle data
gene1_exp <- gene1_exp %>% dplyr::select(-chromosome, -start, -end)
gene1_exp_df <- as.data.frame(t(gene1_exp))
gene1_exp_df <- gene1_exp_df[-1, ]
gene1_exp_df <- tibble::rownames_to_column(as.data.frame(gene1_exp_df), "Individuals")
names(gene1_exp_df) <- c("Individuals", "GeneExp")

gene1_var_df <- as.data.frame(t(gene1_vars))
names(gene1_var_df) <- as.vector(gene1_vars$SNP)
```

```r
gene1_var_df <- gene1_var_df[-1, ]
gene1_var_df <- tibble::rownames_to_column(gene1_var_df, "Individuals")
gene1_var_df$Individuals <- gsub(".*_","", gene1_var_df$Individuals)

#Final dataframe merge
gene1_data <- merge(gene1_exp_df, gene1_var_df, by="Individuals")
gene1_data[1:4, 1:8]
```

```
##   Individuals  GeneExp 20_49051640_A_G_b37 20_49051816_T_TAA_b37
## 1     HG00096 28.29987                   0                     0
## 2     HG00097 23.46323                   0                     0
## 3     HG00099 17.72964                   1                     1
## 4     HG00100 25.84449                   0                     0
##   20_49052954_T_C_b37 20_49053450_C_T_b37 20_49053871_C_A_b37
## 1                   1                   1                   1
## 2                   2                   2                   2
## 3                   2                   2                   2
## 4                   1                   1                   1
##   20_49054108_A_G_b37
## 1                   2
## 2                   2
## 3                   2
## 4                   2
```

Split data into training and testing

```r
#training data
X_train <- data.matrix(sapply(gene1_data[1:268,3:2152], as.numeric))
Y_train <- data.matrix(as.numeric(gene1_data$GeneExp[1:268]))

#testing data
X_test <- data.matrix(sapply(gene1_data[269:358,3:2152], as.numeric))
Y_test <- data.matrix(as.numeric(gene1_data$GeneExp[269:358]))

#data
#the response
X <- data.matrix(sapply(gene1_data[,3:2152], as.numeric))
#the outcome
Y <- data.matrix(as.numeric(gene1_data$GeneExp))
```

Run LASSO with glmnet

```r
set.seed(1233)

#Run lasso regression (alpha=1)
fit1 <- glmnet(x=X_train, y=Y_train,alpha = 1, lambda = 0.01, standardize = TRUE)
glm_pred <- predict(fit1, newx=X_test, s=0.01)

#calculate RMSE
sqrt(mean((glm_pred - Y_test)^2))
```

```
## [1] 195.4083
```

Run Principal Component Regression

Step 1: Normalize and scale the data Step 2: Generate covariance matrix Step 3: Perform Single Value Decomposition

```r
set.seed(1)
#standardize the data
X.scaled <- scale(X, center = T, scale = T)
Y.scaled <- scale(Y, center = T, scale = T)

#Single Value Decomposition of the scaled vector
Xsvd <- svd(X.scaled, nu = 0)
Xsvd$d <- Xsvd$d/sqrt(nrow(X.scaled) - 1)

#Add Principal component scores
Xsvd$scores <- X.scaled %*% Xsvd$v

#Add names to eigenvectors
dimnames(Xsvd$v) <- list(colnames(X.scaled), paste0("PC", seq_len(ncol(Xsvd$v))))

#View top 10 PCs with variance explained by each
#Xsvd$scores[,1:20] %>% head(1)
Xsvd$var <- Xsvd$d^2/sum(Xsvd$d^2)
pc_df <- data.frame("PCs" = paste0("PC", seq_len(ncol(Xsvd$v))), "Scores" = as.vector(Xsvd$sco
pc_df %>% head(10)
```

```
##     PCs     Scores Variance_explained
## 1   PC1  18.279164         0.06531712
## 2   PC2  -8.424923         0.05422389
## 3   PC3   1.994324         0.04803224
## 4   PC4 -12.589615         0.04693614
## 5   PC5   9.645772         0.04216777
## 6   PC6  -2.118952         0.04200894
## 7   PC7   9.729784         0.03413867
## 8   PC8   8.331388         0.03215538
## 9   PC9  -5.814618         0.03144519
## 10 PC10   7.438896         0.02807713
```

Run Regression with Principal Components

```r
set.seed(1)
pcs <- as.data.frame(Xsvd$scores)
names(pcs) <- paste0("PC", seq_len(ncol(pcs)))
Y_pcr <- cbind(Y.scaled, pcs[,1:10])
pcrmodel <- lm(Y.scaled ~ ., data = Y_pcr)
summary(pcrmodel)
```

```
##
## Call:
## lm(formula = Y.scaled ~ ., data = Y_pcr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.86320 -0.81684 -0.00276  0.80091  1.89206
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.529e-16  5.319e-02   0.000    1.000
## PC1         -2.260e-03  4.495e-03  -0.503    0.615
## PC2          4.234e-03  4.933e-03   0.858    0.391
## PC3         -1.692e-03  5.242e-03  -0.323    0.747
## PC4         -3.215e-03  5.303e-03  -0.606    0.545
## PC5         -1.257e-03  5.594e-03  -0.225    0.822
## PC6          6.671e-03  5.605e-03   1.190    0.235
## PC7         -4.930e-03  6.218e-03  -0.793    0.428
## PC8          3.925e-03  6.406e-03   0.613    0.541
## PC9         -2.891e-03  6.478e-03  -0.446    0.656
## PC10         7.824e-03  6.856e-03   1.141    0.255
##
## Residual standard error: 1.006 on 347 degrees of freedom
## Multiple R-squared:  0.01542,    Adjusted R-squared:  -0.01296
## F-statistic: 0.5434 on 10 and 347 DF,  p-value: 0.8589
```

Partial Least Square Regression

Step1 : Initialize Step 2 : Iteration to derive mth direction Zm and update Xm so it is orthogonal to Zm Step 3: Repeat for X1 .. Xp and use Z1.. Zp in lm model

```
p = ncol(as.data.frame(X_test))
dimnames(X_test) <- NULL
Z_list <- list()

for (j in seq(1, p)){
  Xj = as.matrix(X_test[,j])
  Xjm = matrix(0, ncol=11, nrow = nrow(X_test))
  Xjm[,1] <- Xj[,1]
  for (m in seq(1, 10)){ #10 pls components
    #initialize power
    if (m==1){f=m} else {f = m-1}
    #initialize Xjm
    Xjf = Xjm[,f]

    #calculate phi
    phi = t(Xjf) %*% Y_test
    #vector for direction m
    Zm = phi %*% Xjf

    #update Xjf for k=f+1
    k=f+1
    Xjk = Xjf - ((t(Zm) %*% Xjf)/(t(Zm) %*% Zm))*as.vector(Zm)
    Xjm[,k] <- Xjk[,1]
  }
  #Save Z component
  Z_list[[j]] <- Zm
}
```

As PLS function is giving error I will be using the pls package for regression

```
library(pls)
set.seed(777)
pls_df <- as.data.frame(cbind(sapply(gene1_data[,3:200], as.numeric),as.numeric(gene1_data$Gen
pls_df <- pls_df %>% dplyr::rename("Y"="V199")
pls.fit <- plsr(Y~.,data=pls_df,scale=T, ncomp=10)
summary(pls.fit)
```

```
## Data:    X dimension: 358 198
##  Y dimension: 358 1
## Fit method: kernelpls
## Number of components considered: 10
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X   21.878   41.191   46.186   59.028   68.080   78.154   83.126    86.99
## Y    0.829    1.646    5.888    7.026    7.873    8.619    9.718    10.96
##    9 comps  10 comps
## X    89.53     91.79
## Y    12.34     13.14
```