HW 1 & 2 : Gram Schmidt orthogonalization and LASSO implementation with cyclic coordinate descent

Avantika Diwadkar

9/19/2022

Load libraries

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

Question 1: Implement Gram Schmidt orthogonalization algorithm

Run multiple regression using the mtcars demo data with miles per galon as outcome

```r
#I will be using mtcars as the example data to run multiple regression
#with miles per galon as outcome
mtdata <- mtcars %>% dplyr::select(mpg,disp,hp)
names(mtdata) <- c("y","x1", "x2")
mreg_model <- lm(y~x1+x2, data = mtdata)
#summary(mreg_model)
mreg_model$coefficients
```

```
## (Intercept)          x1          x2
## 30.73590425 -0.03034628 -0.02484008
```

Function to perform Gram Schmitt orthogonalization

```r
gram_schmidt_process <- function(y, x1, x2){
  #Initialize
  x0 <- as.vector(rep(1, length(y)))
  z0 <- x0
  #perform gram schmidt
  #Get gamma coeff and residuals
  gamma1 = as.vector(z0 %*% x2 / (z0 %*% z0))
  gamma_2 = as.vector(z0 %*% (x1) / (z0 %*% z0))
  z1 = as.vector(x2 - gamma1 * z0)
  gamma_12 = as.vector(z1 %*% (x1) / (z1 %*% z1))
  z2 = as.vector(x1 - gamma_2 * z0 - gamma_12 * z1)

  #Get beta coefficient
  beta_p = z2 %*% y/ (z2 %*% z2)
  return(beta_p)
}
```

Run Gram Schmidt orthogonalization function and compare coefficients with regression output

```
#Run
gm_coeff <- c(
  gram_schmidt_process(y=as.vector(mtdata$y), x1 = as.vector(mtdata$x1), x2 = as.vector(mtdata
  gram_schmidt_process(y=as.vector(mtdata$y), x1 = as.vector(mtdata$x2), x2 = as.vector(mtdata
names(gm_coeff) <- c("x1","x2")
gm_coeff
```

```
##          x1          x2
## -0.03034628 -0.02484008
```

Question 2: LASSO algorithm with cyclic coordinate descent

```
#Read in the data
norm_expr <- read.table("Data/GEUVADIS_normalized_expression_chr20", header=T)

genot_df <- read.table("Data/GEUVADIS_chr20_processed.traw", header=T)
```

Clean the data

```
#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$SNP)
gene1_var_df <- gene1_var_df[-1, ]
gene1_var_df <- tibble::rownames_to_column(gene1_var_df, "Individuals")
gene1_var_df$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
```

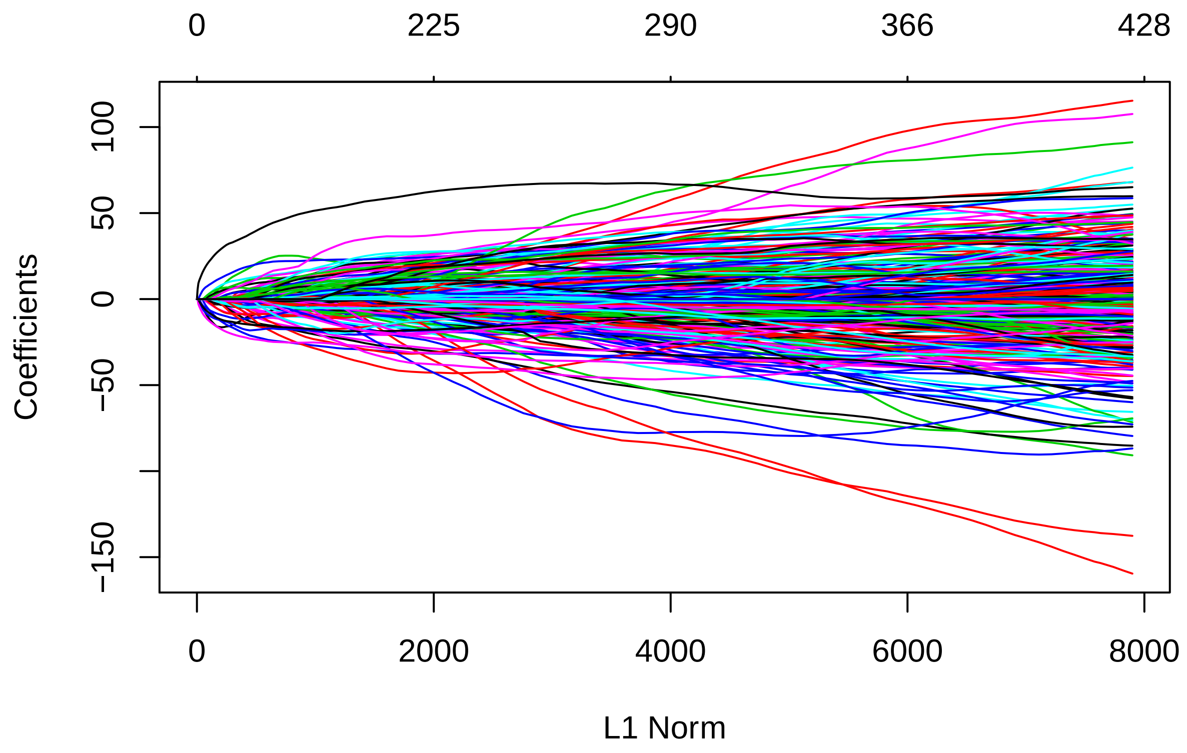
Run GLMNET for one gene

```
set.seed(1233)

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

#Run lasso regression (alpha=1)
```

```
fit <- glmnet(x=X, y=Y,alpha = 1)
plot(fit)
```



Customized lasso script with cyclic co-ordinate descent

Lasso regression is a form of shrinkage method that utilizes the L1 penalty regularization to automatically perform model selection in linear regression.

The steps involve - standardizing the predictor variables to be centered around 0 with variance = 1 and outcome variables to be centered around 0, generating the soft thresholding function for single predictor, and performing lasso with cyclic co-ordinate descent

```
#Standardize the data before running lasso
#This step is performed by default in glmnet
X_std <- scale(X, center=T, scale=T)
Y_std <- scale(Y, center=T, scale=F)

#Function coordinate descent step for single predictor
run_coord_desc <- function(i, X, Y, weights, lambda){
  #Compute prediction
  pred <- X[,i]*weights[i]
```

```r
  #Compute Rho
  rho <- (X[,i]%*%(Y-pred))/nrow(X)

  #Soft thresholding
  if (rho < (-lambda)){
    new_weight = rho + lambda
  } else if (rho > lambda){
    new_weight = rho - lambda
  } else {
    new_weight = 0
  }

  #Return
  new_weight
}

#Cyclic coordinate descent for multiple predictors
run_lasso_cyclic_desc <- function(X, Y, lambda, tolerance){
  #Get number of features
  no_of_feat <- dim(X)[2]

  #Initialize
  weights <- rep(0, no_of_feat)
  wgt_change <- rep(0, no_of_feat)
  converged <- F

  if (!converged){
    for (i in seq(1, no_of_feat)){
      new_weight <- run_coord_desc(i,X,Y, weights,lambda)

      # Calculate change in weight for feature
      wgt_change[i] = abs(new_weight - weights[i])

      # assign new weight
      weights[i] = new_weight
    }

    #Get max change in weight
    max_wgt_change <- max(wgt_change)
    if (max_wgt_change < tolerance){
      converged <- T
    }

  }

  #Return
  weights
```

```
}
```

Compare output from lasso CCD with glmnet for set of lambdas

```
set.seed(480)
lambda_list = c(0.05,0.1, 0.5, 1, 5, 10, 15, 16)

for (lambda in lambda_list){

  #Run glmnet
  fit_df <- as.data.frame(as.matrix(coef(fit, s = lambda)))[-1,]
  coeff1 <- mean(fit_df)

  #Run lasso CCD function
  lasso_fit <- run_lasso_cyclic_desc(X_std, Y_std, lambda=lambda, tolerance=1e-5)
  lasso_fit_df <- as.data.frame(lasso_fit) %>% dplyr::mutate(Coeff = colnames(X))
  coeff2 <- mean(lasso_fit_df$lasso_fit)


  #Get root mean square error to compare the two models
 message(paste0("The root mean square error between the two models for lambda=",
                lambda ," is ",sqrt(mean((coeff2 - coeff1)^2)), "\n"))
}
```

```
## The root mean square error between the two models for lambda=0.05 is 0.340736215544611

## The root mean square error between the two models for lambda=0.1 is 0.335540993208798

## The root mean square error between the two models for lambda=0.5 is 0.315183393495511

## The root mean square error between the two models for lambda=1 is 0.286301156335237

## The root mean square error between the two models for lambda=5 is 0.0665165627510851

## The root mean square error between the two models for lambda=10 is 0.00203847490216321

## The root mean square error between the two models for lambda=15 is 0.000600590810293883

## The root mean square error between the two models for lambda=16 is 0
```