# Lasso CV

Tucker Morgan - tlm2152

3/20/2022

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.2

## Warning: package 'ggplot2' was built under R version 4.1.2

## Warning: package 'tibble' was built under R version 4.1.2

## Warning: package 'tidyr' was built under R version 4.1.2

## Warning: package 'dplyr' was built under R version 4.1.2
```

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.1.3
```

```
source("./shared_code/setup.R")
```

```
## Warning: package 'beepr' was built under R version 4.1.2

## Warning: package 'gtsummary' was built under R version 4.1.3
```

```
source("./shared_code/partition.R")
source("./shared_code/data_prep.R")
source("./shared_code/lasso.R")
source("./shared_code/auc_calc_lasso.R")
```

```
## Warning: package 'pROC' was built under R version 4.1.3
```

```
part_bc <- partition(p = 0.8, data = bc)

bc_trn <-
  part_bc %>%
  filter(part_id == "train") %>%
  select(-part_id)

bc_tst <-
  part_bc %>%
  filter(part_id == "test") %>%
  select(-part_id)
```

```
source("./shared_code/cv_folding.R")

bc_trn_folds <-
  cv_sets(training = bc_trn) %>%
  select(-fold_p)
```

```r
set.seed(100)

X <- bc_trn_folds[, -c(1,32)]
X <- as.matrix(X)
Y <- bc_trn_folds$diagnosis
lambda_vec <- seq(0, 0.4, length = 5) # lambda vector for testing

# creating a simple example function for testing
ex_func <- function(x, y, lambda_vec){
  glmnet(x = x, y = y,
         standardize = TRUE,
         alpha = 1,
         lambda = lambda_vec,
         family = "binomial"(link = "logit"))
}

ex_fit <- ex_func(x = X, y = Y, lambda_vec = 0) #%>% coef() # just for example, not stored

cv_function <- function(k = 5, training, func, lambda_vec){

  auc_list = list()
  mean_auc_list = list()
# first, a for loop to iterate over a lambda vector
  for (j in 1:length(lambda_vec)){
    # and now we have a for loop to iterate over each fold, k = 5 here
    for (i in 1:k){
      # this will identify the training set as not i
      trn_set =
        training %>%
        filter(fold_id != i) %>%
        select(-fold_id)
      # and this assigns i to be the test set
      tst_set =
        training %>%
        filter(fold_id == i) %>%
        select(-fold_id)
      # making matrices
      X_trn <- as.matrix(trn_set[,-1])
      X_tst <- as.matrix(tst_set[,-1])
      Y_trn <- trn_set$diagnosis
      # fitting our function based on training set
      trn_fit = func(x = X_trn, y = Y_trn, lambda_vec = lambda_vec[j])
      # calculating AUC
      trn_pred <- predict(trn_fit,
                          newx = X_tst,
                          type = "response")
      trn_roc <- pROC::roc(tst_set$diagnosis, trn_pred)

      auc_list[[i]] = trn_roc$auc
    }
    # calculating mean cv auc for each lambda
    auc_df = data.frame("auc" = do.call(rbind, auc_list))
    mean_auc = mean(auc_df$auc)
    mean_auc_list[[j]] = data.frame("mean_auc" = mean_auc, "lambda" = lambda_vec[j])
  }
  # creating dataframe to show lambda values and corresponding mean AUC
  res = as.data.frame(do.call(rbind, mean_auc_list))

  return(res)
}

cv_function(training = bc_trn_folds, func = ex_func, lambda_vec = lambda_vec)
```

The cross-validation function seems to work as intended, let's try updating to work with Jimmy's lasso function.

```
cv_jt <- function(k = 5, training, func, lambda_vec){

  auc_list = list()
  # mean_auc_list = list()
  # we have a for loop to iterate over each fold, k = 5 here

  pb <- progress_bar$new(
      format = " folding [:bar] :percent eta: :eta",
      total = k, clear = FALSE, width = 60)

  for (i in 1:k) {


    # this will identify the training set as not i
    trn_set =
      training %>%
      filter(fold_id != i) %>%
      select(-fold_id)

    # and this assigns i to be the test set
    tst_set =
      training %>%
      filter(fold_id == i) %>%
      select(-fold_id) %>%
      rename(y = diagnosis)

    # making matrices
    X_trn <- trn_set[,-1]
   # X_tst <- tst_set[,-1]
    Y_trn <- trn_set$diagnosis




    # fitting our function based on training set
    trn_fit = func(x = X_trn, y = Y_trn, lambda_vec = lambda_vec)

    # calculating AUC, updates required, add AUC to end of i-th matrix and store
    # rbind matrices together, group on lambda and take mean AUC from there

    trn_roc <- auc_calc_lasso(trn_fit, tst_set)
    trn_roc <- trn_roc %>% filter(!is.na(lambda))

    auc_list = bind_rows(auc_list, trn_roc)
   # auc_list[[i]] = trn_roc

    pb$tick()
  }


   # calculating mean cv auc for each lambda
   # auc_df = data.frame("auc" = do.call(rbind, auc_list))
   #  mean_auc = mean(auc_df$auc)
   # mean_auc_list[[j]] = data.frame("mean_auc" = mean_auc, "lambda" = lambda_vec[j])

  res <-
    auc_list %>%
    group_by(lambda) %>%
    summarise(mean_auc = mean(auc_vals))

  # creating dataframe to show lambda values and corresponding mean AUC
  # res = as.data.frame(do.call(rbind, mean_auc_list))

  return(res)
}

lambda_vec <- rep(seq(10, 2, -1), 4) / c(rep(1, 9), rep(10, 9), rep(100, 9), rep(1000, 9))

cv_res <- cv_jt(k = 5, training = bc_trn_folds, func = lasso, lambda_vec = lambda_vec)
cv_res %>% knitr::kable()
```

| lambda | mean_auc |
|---:|---:|
| 2e-03 | 0.8987679 |
| 3e-03 | 0.8963980 |
| 4e-03 | 0.8953984 |
| 5e-03 | 0.8947197 |
| 6e-03 | 0.8950532 |
| 7e-03 | 0.8951564 |
| 8e-03 | 0.8941699 |
| 9e-03 | 0.8938875 |
| 1e-02 | 0.8918259 |
| 2e-02 | 0.7754069 |
| 3e-02 | 0.6789104 |
| 4e-02 | 0.6789054 |
| 5e-02 | 0.6794759 |
| 6e-02 | 0.6801020 |
| 7e-02 | 0.6225511 |
| 8e-02 | 0.6275511 |
| 9e-02 | 0.6275511 |
| 1e-01 | 0.6275511 |
| 2e-01 | 0.6275511 |
| 3e-01 | 0.6275511 |
| 4e-01 | 0.6275511 |
| 5e-01 | 0.6275511 |

| lambda | mean_auc |
|---:|---:|
| 6e-01 | 0.6275511 |
| 7e-01 | 0.6275511 |
| 8e-01 | 0.6275511 |
| 9e-01 | 0.6226312 |
| 1e+00 | 0.5981390 |
| 2e+00 | 0.5000000 |
| 3e+00 | 0.5000000 |
| 4e+00 | 0.5000000 |
| 5e+00 | 0.5000000 |
| 6e+00 | 0.5000000 |
| 7e+00 | 0.5000000 |
| 8e+00 | 0.5000000 |
| 9e+00 | 0.5000000 |
| 1e+01 | 0.5000000 |