homework-4

```r
library(BIS557)
library(reticulate)
use_condaenv("r-reticulate")
library(casl)
```

Part.1

ridge_py() implements a numerically-stable ridge regression that takes into account colinear (or nearly colinear) regression variables, using Python code. The coefficients match the ones from the R implementation.

```r
library(BIS557)
#python implementation
fit_py <- ridge_py(Sepal.Length ~ .- Species, iris, contrasts = list(Species = "contr.sum"), lambda = 0
fit_py$coefficients
#>            [,1]
#> [1,]  1.4991827
#> [2,]  0.7459804
#> [3,]  0.7282092
#> [4,] -0.5654540
#r implementation
fit_r <- ridge(Sepal.Length ~ .- Species, iris,
               contrasts = list(Species = "contr.sum"), lambda = 0.5)
fit_r$coefficients
#>            [,1]
#> [1,]  1.4991827
#> [2,]  0.7459804
#> [3,]  0.7282092
#> [4,] -0.5654540
```

Part.2

A function named lm_ofc() was created and it fits a linear model by reading in contiguous rows from a data frame. QR decomposition is used in this implementation. The results are close to the ones of the lm() function.

```r
set.seed(1234)
library(BIS557)
n <- 1e6
d <- 5
df <- as.data.frame(matrix(runif(n * d, min = -1, max = 1), nrow = n, ncol = d))
names(df) <- c("y", "V1", "V2", "V3", "V4")
N <-200
beta <- matrix(0, nrow = N, ncol = d)
#select contiguous rows from a data frame
for (i in 1:N){
start <- (i-1) * n/N + 1
end <- i * n/N
df_new <- df[start:end, ]
fit_linear_model <- lm_ofc(y ~ ., df_new)
beta[i, ] <- fit_linear_model$coefficients
}
#fitted model
coefficients <- apply(beta, 2, mean)
coefficients
#> [1] -0.0003846903 -0.0011663374 -0.0012348670  0.0011435936 -0.0010746171
#lm results
fit_lm <- lm(y  ~ ., df)
fit_lm$coefficients
#>  (Intercept)           V1           V2           V3           V4
#> -0.000377412 -0.001183632 -0.001265702  0.001165479 -0.001065048
```

Part. 3

install_github("statsmaths/casl")

lasso() function fits a LASSO regression model in Python based on the soft-threshold function. The coefficients are the same as the "casl" results.

```
library(BIS557)
data(iris)
form <- Sepal.Length ~ .; df <- iris
df_no_na <- model.frame(form, df)
X0 <- qr.Q(qr(model.matrix(form, df_no_na, contrasts.arg = list(Species = "contr.sum"))))
yname <- as.character(form)[2]
y0 <- matrix(df_no_na[,yname],ncol = 1)
#lasso results using casl package
fit_casl <- casl_lenet(X0, y0, lambda = 0.01, maxit = 1e5)
fit_casl
#>            [,1]
#> [1,] -70.065925
#> [2,]   0.000000
#> [3,]   7.688435
#> [4,]   0.000000
#> [5,]   0.000000
#> [6,]   0.000000
#python implementation
fit_py <- lasso(Sepal.Length  ~ ., iris, lambda = 0.01, contrasts = list(Species = "contr.sum"))
fit_py$coefficients
#>            [,1]
#> [1,] -70.065925
#> [2,]   0.000000
#> [3,]   7.688435
#> [4,]   0.000000
#> [5,]   0.000000
#> [6,]   0.000000
```

Part.4

For the final project I plan to implement a convolutional neural network that will be able to detect and classify dog breeds based on a training dataset. Backward propagation will be used and out-of-sample prediction accuracy will be calculated to estimate the effectiveness of this network.

Reference:

[1] https://numpy.org/doc/stable/reference/generated/numpy.linalg.svd.html

[2] https://numpy.org/doc/stable/reference/generated/numpy.sign.html

[3] Textbook P182 - P193