homework-4

```r
library(bis557)
```

Question 1

In Python, implement a numerically-stable ridge regression that takes into account colinear (or nearly colinear) regression variables. Show that it works by comparing it to the output of your R implementation.

The R implementation:

```r
data(iris)
iris$target = ifelse(iris$Species=="setosa", 0, ifelse(iris$Species=="versicolor", 1,
  2))
X = iris[,1:3]
y = iris$target
ridge(X, y, 0.01)
#> $coefficients
#>             [,1]
#> [1,] -0.23152809
#> [2,]  0.09850707
#> [3,]  0.54588106
#>
#> $lambda
#>      [,1] [,2] [,3]
#> [1,] 0.01 0.00 0.00
#> [2,] 0.00 0.01 0.00
#> [3,] 0.00 0.00 0.01
```

The python output (notebook) shows that the coefficients are -0.23178751, 0.09881172 and 0.54603559, which are exactly the same as R output.

Question 2

Create an "out-of-core" implementation of the linear model that reads in contiguous rows of a data frame from a file, updates the model. You may read the data from R and send it to your Python functions for fitting. (please see python notebook)

Question 3

Implement your own LASSO regression function in Python. Show that the results are the same as the function implemented in the casl package.

```r
casl_util_soft_thresh <-
function(a, b)
{
  a[abs(a) <= b] <- 0
  a[a > 0] <- a[a > 0] - b
  a[a < 0] <- a[a < 0] + b
  a
```

```
}

casl_lenet_update_beta <-
function(X, y, lambda, alpha, b, W)
{
  WX <- W * X
  WX2 <- W * X^2
  Xb <- X %*% b

  for (i in seq_along(b))
  {
    Xb <- Xb - X[, i] * b[i]
    b[i] <- casl_util_soft_thresh(sum(WX[,i, drop=FALSE] *
                                    (y - Xb)),
                                  lambda*alpha)
    b[i] <- b[i] / (sum(WX2[, i]) + lambda * (1 - alpha))
    Xb <- Xb + X[, i] * b[i]
  }
  b
}

# the casl lasso function
casl_lenet <-
function(X, y, lambda, alpha = 1, b=matrix(0, nrow=ncol(X), ncol=1),
         tol = 1e-5, maxit=50L, W=rep(1, length(y))/length(y))
{
  for (j in seq_along(lambda))
  {
    if (j > 1)
    {
      b[,j] <- b[, j-1, drop = FALSE]
    }

    # Update the slope coefficients until they converge.
    for (i in seq(1, maxit))
    {
      b_old <- b[, j]
      b[, j] <- casl_lenet_update_beta(X, y, lambda[j], alpha,
                                       b[, j], W)
      if (all(abs(b[, j] - b_old) < tol)) {
        break
      }
    }
    if (i == maxit)
    {
      warning("Function lenet did not converge.")
    }
  }
  b
}
```

```r
#check consistency
diabetes = read.table("https://web.stanford.edu/~hastie/Papers/LARS/diabetes.data",
  header = T)
X = as.matrix(diabetes[, 1:10])
y = diabetes$Y

casl_lenet(X, y, lambda=1)
#> Warning in casl_lenet(X, y, lambda = 1): Function lenet did not converge.
#>              [,1]
#>  [1,]   0.3279708
#>  [2,] -19.6516085
#>  [3,]   7.1969435
#>  [4,]   0.3035790
#>  [5,]  -0.2384569
#>  [6,]  -0.3613295
#>  [7,]  -0.9590866
#>  [8,]  14.5782324
#>  [9,]  10.1761675
#> [10,]  -0.2634052
```

The results are the same as the LASSO function in python.

Question 4

Topic: optimize the K parameter in k-nearest neighbors(KNN) algorithm via cross-validation.
Benchmark comparison: ROC (Receiver Operating Characteristic) Curve to evaluate the diagnostic ability of binary classifiers.
Application: predict the outcome of diabetes using the Pima Indians Diabetes Database.