The vignette for homework 4

Since in this homework, we need to use python in Rmarkdown, so here we need to download package reticulate. And 'r-reticulate' is the conda environment I created in class whose python version is 3.9.

```
#install.packages("reticulate")
library(bis557)
library(reticulate)
```

```
library(devtools)
#> Loading required package: usethis
install_github("statsmaths/casl")
#> Skipping install of 'casl' from a github remote, the SHA1 (196caaff) has not changed since
        last install.
#>   Use `force = TRUE` to force installation
library(casl)
```

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

Since $\hat{\beta} = (X^tX + \lambda I)^{-1}X^TY$ and $X = U\Sigma V$,

$$\hat{\beta} = V(\Sigma\Sigma + \lambda I)^{-1}\Sigma U^TY$$

The function we create in this question is lm_ridge_py whose codes are in the file"lm_ridge_py.r" and test codes are in the file"test-lm-ridge-py.r".

The ridge regression with R implementation:

```
data("iris")
iris=na.omit(iris)
#colinear variables
iris$Sepal.Width2=iris$Sepal.Width*2
# the output of R
r_fit1=lm_ridge(Sepal.Length ~ ., iris,lambda=0.5)
```

The ridge regression with python implementation:

```
Y=matrix(iris$Sepal.Length, ncol = 1)
X=model.matrix(Sepal.Length ~ .,iris)
py_fit1=lm_ridge_py (Y,X,lambda =0.5)
```

Compare these two results:

```
r_fit1$coefficients
#>           [,1]
#> [1,]  1.4992495
```

```
#> [2,]  0.1437062
#> [3,]  0.7748818
#> [4,] -0.4515711
#> [5,] -0.2532605
#> [6,] -0.4311099
#> [7,]  0.2874123
py_fit1$coefficients
#>           [,1]
#> [1,]  1.4992495
#> [2,]  0.1437062
#> [3,]  0.7748818
#> [4,] -0.4515711
#> [5,] -0.2532605
#> [6,] -0.4311099
#> [7,]  0.2874123
```

We could observe that the results are the same.

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 fo fitting.

The function we create in this question is linear_model_py whose codes are in the file"linear-model-py.r" and its test codes are in "test-linear-model-py.r".

We first create a dataset with n individuals and p features. Then we run linear model with the data of m individuals each time. So we got n/m sets of beta, then calculate the average of these sets of beta.

```
n=1e6
p=4
X <- matrix(rnorm(n*p),n,p)
beta_true <- matrix(c(2,4,1.5,5),p,1)
Y <- X%*%beta_true
m=1e3
beta <- matrix(0,p,m)
for(i in 1: (n/m)){
  beta[,i] <- linear_model_py(Y[(1e3*(i-1)+1):(1000*i)],X[(1e3*(i-1)+1):
        (1000*i),])$coefficients
}
beta_py <- rowMeans(beta)
beta_py
#> [1] 2.0 4.0 1.5 5.0
beta_true
#>      [,1]
#> [1,]  2.0
#> [2,]  4.0
#> [3,]  1.5
#> [4,]  5.0
```

We could observe the beta we got is very precise.

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

For an orthogonal design matrix, lasso regression has a closed-form solution:

$$\hat{\beta}_i^{lasso} = sign(\hat{\beta}_i^{LS})(|\beta_i^{LS}| - \lambda)^+$$

reference:"https://stats.stackexchange.com/questions/17781/derivation-of-closed-form-lasso-solution"

For this case, we create function lm_lasso_py_or() whose codes are in "lm_lasso_py_or.r" file.

```
n=1000
p=4
X <- matrix(rnorm(n*p),n,p)
beta_true <- matrix(c(2,4,1.5,5),p,1)
Y <- X%*%beta_true
fit_casl= casl::casl_lenet(X,Y,lambda=0.01,maxit=1000L)
fit_lasso=lm_lasso_py_or(Y,X,lambda=0.01)
fit_casl
#>          [,1]
#> [1,] 1.991411
#> [2,] 3.990355
#> [3,] 1.490667
#> [4,] 4.989905
fit_lasso$coefficients
#>      [,1] [,2]
#> [1,] 1.99 1.99
#> [2,] 3.99 3.99
#> [3,] 1.49 1.49
#> [4,] 4.99 4.99
```

We could observe that the results are similar.

4. The final project proposal

Mechanisms of Action (MoA) Prediction

https://www.kaggle.com/isaienkov/mechanisms-of-action-moa-prediction-eda

Nowadays, understanding the underlying biological mechanism of a disease is the key factor to discover drug.Thus researchers seek to identify a protein target associated with a disease and then develop a molecule which could modulate that protein target.And we call the biological activity of that given molecule Mechanism-of-action or MoA.

Here in this project, with the dataset that includes gene expression and cell viability data in addition to the MoA annotations of more than 5,000 drugs,we need to predict multiple targets of the Mechanism of Action (MoA) response(s) of different samples.

Since it is a multi-labels classification problem and has a lot of features, it is challenging to select proper features and models. I'd like to consider PCA or lasso to conduct feature selection , and try to use and compare multiple classification models such as XGBoost and multilabel Neural Network.

I hope in this project, I will be able to find a good classification model or algorithm to improve the predication of MoAs of drugs based on their biological activity. The accuracy is based on the log loss:

$$score = -\frac{1}{M}\sum_{m=1}^{M}\frac{1}{N}\sum_{i=1}^{N}[y_{i,m}log(\hat{y_{i,m}}) + (1-y_{i,m})log(1-\hat{y_{i,m}})$$