The vignette for homework 3

```
library(bis557)
```

1. CASL 5.8 Exercise number 2. Include the write up in your homework-2 vignette.

We mentioned that the Hessian matrix in Equation 5.19 ($H(l) = X^t \cdot D \cdot X$)can be more ill- conditioned than the matrix X^tX itself. Generate a matrix X and probabilities p such that the linear Hessian (X^tX) is well-conditioned but the logistic variation is not.

Answer:

A matrix is ill-conditioned when the condition number is very high.And the condition number of a matrix is the ratio of the largest singular value to the smallest non-zero singular value.

Linear Hessian and logistic Hessian can both be written as $H(l) = X^t \cdot D \cdot X$. Here, for linear Hessian , $D = I$ and for logistic Hessian $D_{i,i} = p_i(1 - p_i)$. If we want to make logistic Hessian ill-conditioned, then we need to make the diagnose of D be close to zero.Since So $D_{i,i} = p_i(1 - p_i)$, we need to make p_i close to zero or one. I give following example.

Example:

```
set.seed(100)
n=100
p=2
X=matrix(rnorm(n*p),n,p)
X=cbind(1,X)
beta=matrix(c(100,100,100),1,3)
prob=exp(beta%*%t(X))/(1+exp(beta%*%t(X)))
quantile(prob)
#>           0%          25%          50%          75%         100%
#> 6.511136e-64 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00

#Linear Hessian
H_linear=t(X)%*%X

#Logistic Hessian
H_logistic=t(X)%*%diag(as.vector(prob*(1-prob)))%*%X

#singular values
svd(H_linear)$d
#> [1] 105.88201 100.03313  59.99898
#condition number of Linear Hessian
svd(H_linear)$d[1]/svd(H_linear)$d[3]
#> [1] 1.76473

#singular values
```

```
svd(H_logistic)$d
#> [1] 4.173237e-01 1.609634e-03 1.649944e-08
#condition number of logistic Hessian
svd(H_logistic)$d[1]/svd(H_logistic)$d[3]
#> [1] 25293209
```

Since the condition number of logistic Hessian is much higher than the linear Hessian, then we give an example which the linear Hessian (X^tX) is well-conditioned but the logistic variation is not.

2. Describe and implement a first-order solution for the GLM maximum likelihood problem using only gradient information, avoiding the Hessian matrix. Include both a constant step size along with an adaptive one. You may use a standard adaptive update Momentum, Nesterov, AdaGrad, Adam, or your own. Make sure to explain your approach and compare it's performance with a constant step size.

Answer:

The function we create for this question is **glm_first_order( )**.The information about this function is saved on the glm-first-order.r file and its test code is saved on the test-glm-first-order.r.

We know that the first order solution for the GLM maximum likelihood problem using only gradient information is:

$$\beta^{(new)} = \beta^{(old)} - \alpha\frac{dl(\beta)}{d\beta}|_{\beta=\beta^{(old)}}$$

Here α is step size. $\frac{dl(\beta)}{d\beta}$ is the gradient for the likelihood function $l(\beta)$.

$$\frac{dl(\beta)}{d\beta} = X^t(y - E(y))$$

Here $E(y) = g^{-1}(X^t\beta)$, g( ) is the link function.

Following is the example of how to use this function and comparison between constant step size and additive step size.

```
#Example , here we use the logistic regression.

set.seed(100)
maxit=10000

#constant step size
steps1=rep(0.001,maxit)

# additive step size
steps2=seq(5e-3,5e-7,length=maxit)

#create data
```

```
n=100
p=2
X=matrix(rnorm(n*p),n,p)
X=cbind(1,X)
beta=matrix(c(2,4,6),1,3)
prob=exp(beta%*%t(X))/(1+exp(beta%*%t(X)))
y=rbinom(n,size = 1, prob = prob)

#Run our glm model using constant step size.
fit1=glm_first_order(X,y, binomial(link="logit")$linkinv,steps1,maxit=maxit)

#Run our glm model using additive step size.
fit2=glm_first_order(X,y, binomial(link="logit")$linkinv,steps2,maxit=maxit)

fit1
#> $beta
#>          [,1]
#> [1,] 2.183821
#> [2,] 3.349309
#> [3,] 5.983555
fit2
#> $beta
#>          [,1]
#> [1,] 2.213754
#> [2,] 3.384980
#> [3,] 6.062959
beta
#>      [,1] [,2] [,3]
#> [1,]    2    4    6

sum((fit1$beta-t(beta))^2)
#> [1] 0.4574589
sum((fit2$beta-t(beta))^2)
#> [1] 0.4279046
```

From our results, we could see that the beta we get by using constant step size and additive step size are close to each other.By observing their difference with the true beta, the results of additive step size is a little better. But the performance may depend on different step sizes.

3. Describe and implement a classification model generalizing logistic regression to accommodate more than two classes.

Answer:

The function I create in this question is called **multi_logistic_regression( )**, the information of this function is saved on the multi-logistic-regression.r and its test code is saved on test-multi-logistic-regression.r

Suppose the responses have K classes. for class k, we could use logistic regression to calculate the probabilities whether the subjects belong to the kth class or the other classes. And for each subject, we select the class with the highest probability.In this question, we use the second order solution with gradient information and the Hessian matrix.

```r
#Example of this function
data("iris")
X = iris[,-5]
Y=iris$Species
fit=multi_logistic_regression(X,Y)
acc=sum(fit$y_pred==Y)/length(Y)
acc
#> [1] 0.98
```

Here, the accuracy is 98%. Thus our function has a good result.

Reference：Some of the codes are from the example professor presented at class.