# HW#3

*Zizhong Tian*

*10/30/2020*

```
##   missForest iteration 1 in progress...done!
##   missForest iteration 2 in progress...done!
##   missForest iteration 3 in progress...done!
```

## Question 1 (CASL 5.8 Exercise 2)

Generate a matrix $X$ and probabilities $p$ such that the linear Hessian $X^TX$ is well-conditioned but the logistic variation is not:

By definition, the logistic variation is $X^TDX$, where $D$ is a diagonal matrix with $i^{th}$ entry on the diagonal equal to $p_i(1-p_i)$. To make $X^TDX$ ill-conditioned, we need the diagonal entris of $D$ to be small. This is equivalent that $p_i$'s are mostly close to 0 or 1. Generate appropriate $X$ matrix and $p$ vector decribed above to give an example, then compute the condition numbers to confirm the desired situation:
Display the generation of $X$ and $p$, and the condition numbers for $X^TX$ and $X^TDX$, respectively.

```
set.seed(1023013)
n <- 100
beta <- rep(100, 10)
X <- matrix(rnorm(10*n), nrow=n, ncol=10)
XtX <- t(X) %*% X

p <- as.vector(1/(1+exp(-X %*% beta)))
D <- diag(p*(1-p))
XtDX <- t(X) %*% D %*% X

#Conditon number
kappa(XtX)
```

```
## [1] 3.558143
```

```
kappa(XtDX)
```

```
## [1] 5.864767e+12
```

Under this example, the condition number of the linear Hessian $X^TX$ is about 3.5581 (fairly small), while the condition number of the logistic variation $X^TDX$ is about $5.8648 \times 10^{12}$, which is so large. So, we can conclude that given the $X$ and $p$ properly like above, we can have $X^TX$ is well-conditioned but the logistic variation is ill-conditioned.

## Question 2

The function `glm_firstorder` will help to implement a first-order solution for the GLM maximum likelihood problem using only gradient information without using the Hessian matrix. I include both a constant step size version as well as an adaptive one using "Momentum".

For the constant step method, we just use the common gradient descent algorithm to update the "grad" in the function based on updated beta and the link function. We use a randomly generated Poisson data to check the consistency with the coefficients estimated using the standard "glm" function. Below, we can observe that the beta parameters are similar. We also record the iteration number to get the desired tolerance to compare with the adaptive step method below.

```
set.seed(1713948)
n <- 1000
truth <- c(4, 0.5, 0.3, 1)
X <- cbind(rep(1,n), matrix(rnorm(3*n), ncol=3))
lambda <- exp(X %*% truth)
Y <- rpois(n, lambda=lambda)
covs <- X[,-1]
q2.data <- data.frame(cbind(Y, covs))
form <- Y ~ .

fit.2a <- glm_firstorder(form, q2.data, mu_fun=function(eta) exp(eta))$beta
fit.ref <- glm(form, q2.data, family = poisson(link = "log"))$coefficients
cbind(fit.2a, fit.ref)
```

```
##                        fit.ref
## (Intercept) 3.9783947 4.0005274
## V2          0.5059980 0.5027752
## V3          0.2996782 0.2953425
## V4          1.0095698 0.9989495
```

```
count.a <- glm_firstorder(form, q2.data, mu_fun=function(eta) exp(eta))$iter
count.a
```

```
## [1] 157
```

For the adaptive updating using Momentum, we use the algorithm:

$$v_t = mv_{t-1} + \gamma \bigtriangledown_\beta J(\beta)$$

$$\beta = \beta + v_t$$

We set $m = 0.8$. Below, we also compare with the standard "glm", we observe that the estimates are even better. Also, as we record the iteration number to get convergence, we find that the Momentum method gives a faster rate to converge ($59<157$).

```
fit.2b <- glm_firstorder(form, q2.data, step="momentum", mu_fun=function(eta) exp(eta))$beta
cbind(fit.2b, fit.ref)
```

```
##                        fit.ref
## (Intercept) 4.0023559 4.0005274
## V2          0.5049590 0.5027752
## V3          0.2927950 0.2953425
## V4          0.9954619 0.9989495
```

```
count.b <- glm_firstorder(form, q2.data, step="momentum", mu_fun=function(eta) exp(eta))$iter
count.b
```

```
## [1] 59
```

## Question 3

The function `multiclass_logistic` will help to implement a classification model generalizing logistic regression to accommodate N-classes. Input a desire dataset and a formula, the function could return N sets of coefficient estimates based on N logistic regressions using second-order gradient descent (the method with inverse Hessian matrix). Using the softmax formula, the function could also return a set of probabilities for each individuals, representing the propensity to enter each levels. With these probabilities, the users could get the sense of multi-group classification.
To assess the performance of the classification model, we use `penguinsi` in the same package, construct the setting below, predict the "species" membership based on the "largest probability" of each individual. Implement the function, we could find the accuracy rate of classification (with 3 classes) is about 95.64%, which is a satisfactory result.
A set of coefficient estimates to help accomplish the classification, as well as the accuracy evaluation mentioned above, is outputted below.

```
q3.data <- penguinsi
form <- species ~ bill_length_mm + bill_depth_mm
q3_fit <- multiclass_logistic(form, q3.data, maxit = 50)
q3_fit$beta
```

```
##           [,1]       [,2]       [,3]
## [1,]  24.13553 -2.2102788  3.9988091
## [2,]  49.08624  0.5580412 -4.5376418
## [3,] -31.61429  0.3544237  0.7994035
```

```
max.ind <- apply(q3_fit$probs, 1, which.max)
predicted.Y <- unique(q3_fit$Y)[max.ind]
accuracy <- mean(predicted.Y==q3_fit$Y)
accuracy
```

```
## [1] 0.9563953
```