SDS 385: Exercises 1 - Preliminaries

August 23, 2016

Professor James Scott

Spencer Woody

Problem 1

(A)

$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{N} \frac{w_i}{2} \left(y_i - x_i^T \beta\right)^2 \tag{1}$$
$$= \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2}(Y - X\beta)^T W (Y - X\beta) \tag{2}$$

$$\frac{1}{2}(Y - X\beta)^T W (Y - X\beta) = \frac{1}{2}(Y^T - \beta^T X^T) W (Y - X\beta) \tag{3}$$
$$= \frac{1}{2}(Y^T W - \beta^T X^T W)(Y - X\beta) \tag{4}$$
$$= \frac{1}{2}(Y^T W Y - \beta^T X^T W Y - Y^T W X\beta + \beta^T X^T W X \beta) \tag{5}$$
$$= \frac{1}{2}(Y^T W Y - 2(X\beta)^T W Y + \beta^T X^T W X \beta) \tag{6}$$
$$= \frac{1}{2} Y^T W Y - (X\beta)^T W Y + \frac{1}{2}\beta^T X^T W X \beta, \tag{7}$$

because

$$\beta^T X^T W Y = (X\beta)^T W Y, \tag{8}$$

and

$$Y^T W X \beta = (Y^T W X \beta)^T \because Y^T W X \beta \in \mathbb{R}^1 \tag{9}$$
$$(Y^T W X \beta)^T = (W X \beta)^T Y = (X\beta)^T W^T Y = (X\beta)^T W Y. \tag{10}$$

We want to minimize the objective function from Eqn. (7), so we take the gradient with respect to β and set it equal to zero. For each of the three terms, their are respective gradients with respect to β are

(i)

$$\frac{\partial}{\partial \beta} \frac{1}{2} Y^T W Y = 0 \tag{11}$$

(ii)

$$\frac{\partial}{\partial \beta} - (X\beta)^T W Y = -X^T W Y \tag{12}$$

(iii)

$$\frac{\partial}{\partial \beta} \frac{1}{2} \beta^T X^T W X \beta = \frac{1}{2}\beta^T (X^T W X + (X^T W X)^T) \tag{13}$$
$$= X^T W X \beta. \tag{14}$$

Summing these terms and equaling them to zero yields

$$X^T W X \beta - X^T W Y = 0 \therefore \tag{15}$$
$$(X^T W X)\hat{\beta} = X^T W Y \tag{16}$$

Problem 1 continued on next page. . .

(B) The brute force method of solving Eqn. (16) is the *inversion method*, i.e.

$$\hat{\beta} = (X^T W X)^{-1} X^T W y. \tag{17}$$

However, this method is computationally expensive. Therefore I propose an alternative methods to solving this matrix equation using the Cholesky decomposition. **Cholesky Decomposition**
Let

$$C = X^T W X, \;\; D = X^T W y \tag{18}$$

so

$$C\hat{\beta} = D. \tag{19}$$

We decompose matrix C into a product of a lower-triangular matrix and an upper-triangular matrix, such that $U = L^T$ so

$$C = LU = LL^T \therefore \tag{20}$$

$$LL^T \hat{\beta} = D. \tag{21}$$

Furthermore we define matrix $A = L^T \hat{\beta}$. Thus we are left with two matrix equations to solve.

$$LA = D \tag{22}$$

$$L^T \hat{\beta} = A \tag{23}$$

This method will be much less computationally intensive than the inversion method because of the fact that the two left-matrices L and $U = L^T$ are triangular. We still must invert L and L^T but this is simpler than taking an inverse of a more complicated matrix $X^T W X$. This is similar to an LU decomposition, with the exception that we necessarily have two triangular matrices that are transposes of one another. Therefore, this method gains a computational advantage over LU decomposition from symmetric exploitation.

(C) Code for implementing this method is shown in the appendix to this paper.

(D)

Problem 2

(A) We have $y_i \sim \text{Binomial}(m_i, w_i)$, where

$$w_i = \frac{1}{1+\exp(-x_i^T\beta)}, \quad 1 - w_i = \frac{\exp(-x_i^T\beta)}{1+\exp(-x_i^T\beta)}, \tag{24}$$

so the negative log likelihood is

$$\ell(\beta) = -\log\left\{\prod_{i=1}^N p(y_i|\beta)\right\} \tag{25}$$

$$= -\log\left\{\prod_{i=1}^N \binom{m_i}{y_i}(w_i)^{y_i}(1-w_i)^{m_i-y_i}\right\} \tag{26}$$

$$= -\left\{\sum_{i=1}^N \left(\log\binom{m_i}{y_i} + y_i\log(w_i) + (m_i - y_i)\log(1-w_i)\right)\right\} \tag{27}$$

$$= -\left\{\sum_{i=1}^N \left(\log\binom{m_i}{y_i} + y_i\log\left(\frac{1}{1+\exp(-x_i^T\beta)}\right) + (m_i - y_i)\log\left(\frac{\exp(-x_i^T\beta)}{1+\exp(-x_i^T\beta)}\right)\right)\right\} \tag{28}$$

$$= -\left\{\sum_{i=1}^N \left(\log\binom{m_i}{y_i} - y_i\log(1+\exp(-x_i^T\beta)) - (m_i - y_i)x_i^T\beta - m_i\log(1+\exp(-x_i^T\beta)) + y_i\log(1+\exp(-x_i^T\beta))\right)\right. \tag{29}$$

$$= -\left\{\sum_{i=1}^N \left(\log\binom{m_i}{y_i} - (m_i - y_i)x_i^T\beta - m_i\log(1+\exp(-x_i^T\beta))\right)\right\} \tag{30}$$

$$= \sum_{i=1}^N \left((m_i - y_i)x_i^T\beta + m_i\log(1+\exp(-x_i^T\beta)) - \log\binom{m_i}{y_i}\right) \tag{31}$$

$$\tag{32}$$

The gradient for this expression is,

$$\frac{\nabla\ell(\beta)}{d\beta} = \sum_{i=1}^N \left((m_i - y_i)x_i - m_i\frac{1}{1+\exp(-x_i^T\beta)}\exp(-x_i^T\beta)x_i\right) \tag{33}$$

$$= \sum_{i=1}^N \left((m_i - y_i)x_i - m_i w_i\exp(-x_i^T\beta)x_i\right) \tag{34}$$

$$= \sum_{i=1}^N (m_i - y_i - m_i w_i\exp(-x_i^T\beta))x_i \tag{35}$$

$$= \sum_{i=1}^N (m_i w_i - y_i)x_i \tag{36}$$

Text here.

(B)

(C)

(D)

(E)

```
########################################################
######### Created by Spencer Woody on 24 Aug 2016 #########
########################################################

library(Matrix)
library(microbenchmark)

### No. 1 pt C

# Set N, P, X, W, and y

N <- 2000
P <-  500

X <- matrix(rnorm(N * P), nrow = N)
y <- matrix(rnorm(N), nrow = N)
W <- diag(rep(1, N))

# Inversion method

Inv.method <- function(X.Inv, W.Inv, y.Inv) {
    XtWX <- (t(X.Inv)*diag(W.Inv)) %*% X.Inv
    XtWY <- (t(X.Inv)*diag(W.Inv)) %*% y.Inv
    bhat.Inv <- solve(XtWX) %*% XtWY
    return(bhat.Inv)
}

Cho.decomp <- function(X.Cho, W.Cho, y.Cho) {
    D.Cho <- (t(X.Cho)*diag(W.Cho)) %*% y.Cho
    C.Cho <- (t(X.Cho)*diag(W.Cho)) %*% X.Cho

    U.Cho <- chol(C.Cho)
    L.Cho <- t(U.Cho)

    u <- forwardsolve(L.Cho, D.Cho)
    bhat.Cho <- backsolve(U.Cho, u)

    return(bhat.Cho)
}

microbenchmark(
    Inv.method(X, W, y),
    Cho.decomp(X, W, y),
    times=5)


### No. 1 pt D

N <- 2000
P <-  500

X <- matrix(rnorm(N * P), nrow = N)
mask <- matrix(rbinom(N * P, 1, 0.05), nrow = N)
```

```r
X <- mask * X

Inv.methodSPARSE <- function(X.Inv, W.Inv, y.Inv) {
    X <- Matrix(X, sparse = TRUE)
    XtWX <- (t(X.Inv)*diag(W.Inv)) %*% X.Inv
    XtWY <- (t(X.Inv)*diag(W.Inv)) %*% y.Inv
    bhat.Inv = Matrix::solve(XtWX, XtWY, sparse = TRUE)
    return(bhat.Inv)
}

Inv.methodSPARSE2 <- function(X.Inv, W.Inv, y.Inv) {
    XtWX <- (t(X.Inv)*diag(W.Inv)) %*% X.Inv
    XtWY <- (t(X.Inv)*diag(W.Inv)) %*% y.Inv
    bhat.Inv = solve(XtWX, XtWY)
    return(bhat.Inv)
}

microbenchmark(
    Inv.methodSPARSE(X, W, y),
    Inv.methodSPARSE2(X, W, y),
    Cho.decomp(X, W, y),
    times=5)


microbenchmark(
    solve(XtWX, XtWY),
    Matrix::solve(XtWX, XtWY, sparse = TRUE),
    solve(XtWX) %*% XtWY,
    times = 5
)



# END
```