Exercises 1: Preliminaries

Devon Humphreys

1 Linear Weighted Regression

Given a simple linear model

$$y = X\beta + \epsilon \tag{1}$$

we wish to minimize the sum of squared residuals using the weighted least squares approach (WLS).

(A) We are given the solution in scalar sums and products for $\hat{\beta}$ as

$$\hat{\beta} = \arg\min_{\beta \in \mathcal{R}^P} \sum_{i=1}^{N} \frac{w_i}{2}(y_i - x_i^T \beta)^2 \,. \tag{2}$$

We wish to rewrite this optimization problem in terms of matrix algebra. This will give us the normal equations of linear least squares regression.

Recognize that:

$$\begin{aligned} \frac{1}{2}\sum_{i=1}^{N} w_i(y_i - x_i^T \beta)^2 &= \frac{1}{2}\sum_{i=1}^{N}(y_i - x_i^T \beta) w_i (y_i - x_i^T \beta) && (3)\\ &= \frac{1}{2}\sum_{i=1}^{N}(y_i w_i y_i - 2y_i w_i x_i^T \beta + x_i^T \beta w_i x_i^T \beta) && (4)\\ &= \frac{1}{2}(y^T W y) - 2(y^T W X \beta) + ((X\beta)^T W (X\beta)). && (5) \end{aligned}$$

Taking the partial derivative with respect to β and setting this to 0, we can find the optimal solution of this system of linear equations.

$$\begin{aligned} \nabla_\beta &= \frac{1}{2}\{\nabla_\beta y^T W y - 2\nabla_\beta y^T W X \beta + \nabla_\beta (X\beta)^T W (X\beta)\} && (6)\\ 0 &= \frac{1}{2}\{0 - 2y^T W X + 2X^T W X \beta\} && (7)\\ 0 &= -y^T W X + X^T W X \beta && (8)\\ -X^T W X \beta &= -X^T W y && (9)\\ \hat{\beta} &= (X^T W X)^{-1} X^T W y && (10) \end{aligned}$$

(B) The inversion method in the normal equations is not the fastest or most numerically stable way to solve a general system of linear equations as in the case of linear regression. Another class of methods rely on orthogonal decomposition. Such methods include (1) Cholesky factorization; (2) QR decomposition; and (3) singular value decomposition (SVD).

1. Cholesky Decomposition: the fastest of the three methods, but numerically unstable (that is, it suffers from underflow/overflow problems in floating point representation).
2. QR Decomposition: kind of a middle ground; a bit slower, but still fast and more numerically stable.
3. SVD: slowest, but the most numerically stable; especially useful for rank deficient matrices.

Pseudocode for QR decomposition:

$$X^T W y = X^T W X \beta \tag{11}$$

Recognize that

$$W^{\frac{1}{2}} X = QR\,, \tag{12}$$

where Q is orthonormal and R is a right triangular matrix.
Then

$$X^T W^{\frac{1}{2}} W^{12} y = X^T W^{\frac{1}{2}} W^{\frac{1}{2}} X \beta \tag{13}$$
$$(QR)^T W^{\frac{1}{2}} y = (QR)^T QR\beta \tag{14}$$
$$R^T Q^T W^{\frac{1}{2}} y = R^T Q^T QR\beta \tag{15}$$
$$Q^T W^{\frac{1}{2}} y = R\beta \tag{16}$$

Pseudocode for SVD:
We will factor the design matrix X into orthogonal components and a diagonal matrix containing the "singular values":

$$X = U\Sigma V^T \tag{17}$$

U and V are orthogonal matrices, and Σ is a diagonal matrix whose off-diagonal elements are 0.
Then we recognize that

$$\hat{\beta} = (X^T W X)^{-1} X^T W y \tag{18}$$
$$= ((U\Sigma V^T)^T W U\Sigma V^T)^{-1} (U\Sigma V^T) W y \tag{19}$$

...

(C) R code

```
library(microbenchmark)
library(Matrix)

## inversion method

# the matrix X contains our explanatory variables in y = Xb + e

inversion_solver <- function (n_obs, n_vars) {
  # create a random matrix for demonstration
  X <- rnorm(n_obs * n_vars, mean = 0, sd = 1)
  X <- matrix(X, nrow = n_obs, ncol = n_vars)

  # generate the vector y to be regressed on X
  y <- rnorm(n_obs, mean = 0, sd = 1)

  # solve the least squares problem using the inversion method
  b <- solve(t(X) %*% X) %*% t(X) %*% y

  return (b)
}

## QR method

qr_solver <- function (n_obs, n_vars) {
  # generate a random feature matrix X
  X <- rnorm(n_obs * n_vars, mean = 0, sd = 1)
  X <- matrix(X, nrow = n_obs, ncol = n_vars)

  # generate an observation vector y
  y <- rnorm(n_obs, mean = 0, sd = 1)
  y <- matrix(y, nrow = n_obs, ncol = n_vars)

  # solve the least squares problem using QR decomposition of X
  b <- qr.solve(X, y)
  return (b)
}

## Dealing with sparse matrices in R

inversion_sparse <- function (n_obs, n_vars, sparsity) {
  # define a random feature matrix X with sparsity of 95%
  X <- rnorm(n_obs * n_vars, mean = 0, sd = 1)
  X <- matrix(X, nrow = n_obs, ncol = n_vars)
  mask <- matrix(rbinom(n_obs * n_vars, 1, sparsity), nrow = n_obs)
  X <- X * mask
  X <- Matrix(X, sparse = T) # converts to sparse format
```

```
  # generate random observations in a vector y
  y <- rnorm(n_obs, mean = 1, sd = 1)

  # solve the least squares problem taking advantage of the sparse matrix format
  inv_mat <- solve(t(X) %*% X, sparse = T)
  b <- inv_mat %*% t(X) %*% y
  return (b)
}

## benchmarking the dense matrix solvers on increasing number of variables
var100 <- microbenchmark(inversion_solver(200, 100), qr_solver(200, 100), times = 10); var100
var1000 <- microbenchmark(inversion_solver(2000, 1000), qr_solver(2000, 1000), times = 10); var1000
var2000 <- microbenchmark(inversion_solver(5000, 2000), qr_solver(5000, 200), times = 10); var2000
var5000 <- microbenchmark(inversion_solver(2000, 5000), qr_solver(2000, 5000), times = 10); var5000

## benchmarking the sparse matrix solvers on increasing levels of sparsity and dimension
#sp05 <- microbenchmark(inversion_sparse(150, 50), inversion_solver(150, 50), times = 10); sp05
```

(D) Consider the efficiency and stability of the above methods, but where X is a highly sparse rectangular matrix. Write an additional solver that can exploit the sparsity of A in a linear system $Ax = b$.

QR decomposition is the most efficient and appropriate way to handle this problem. We first store the sparse matrix X in a sparse matrix format using the Matrix library in R, as:

```
X = Matrix(X, sparse = T)
```

We find that it is represented in 122567 bytes compared to the 1600200 bytes for the normal storage wasting space on 0 entries. We then recall our QR algorithm for solving for $\hat{\beta}$ as

$$R^{-1}Q^T W^{\frac{1}{2}} y = \hat{\beta} \tag{20}$$

Letting $W = I$, we have that

$$R^{-1}Q^T y = \hat{\beta} \tag{21}$$

2 Generalized Linear Models

(A) We are given the general form of the negative log likelihood,

$$l(\beta) = -\ln \prod_{i=1}^{N} p(y_i|\beta) \tag{22}$$

Our task is to write the full likelihood for a binomial model using the logistic link function. The model for a single Bernoulli trial is

$$p^n(1-p)^{1-n} \tag{23}$$

First, let

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

Note that:

$$\begin{align}
1 - w_i &= 1 - \frac{1}{1+\exp(-x_i\beta)} \tag{25}\\
&= \frac{1+\exp(-x_i\beta)}{1+\exp(-x_i\beta)} - \frac{1}{1+\exp(-x_i\beta)} \tag{26}\\
&= \frac{\exp(-x_i\beta)}{1+\exp(-x_i\beta)} \tag{27}
\end{align}$$

The critical part of solving the gradient of this equation with respect to β is to find the gradient $\nabla_\beta w_i$. This is

$$\nabla_\beta w_i = \nabla_\beta \frac{1}{1+\exp(-x_i\beta)} \tag{28}$$

By the quotient rule of derivatives, we have that

$$\begin{align}
\nabla_\beta &= \frac{-\nabla_\beta(1+\exp(-x_i\beta)}{(1+\exp(-x_i\beta)^2} \tag{29}\\
&= \frac{x_i\exp(-x_i\beta}{(1+\exp(-x_i\beta)^2} \tag{30}\\
&= \frac{1}{1+\exp(-x_i\beta)}\frac{\exp(-x_i\beta)}{1+\exp(-x_i\beta)}x_i \tag{31}\\
&= w_i(1-w_i)x_i. \tag{32}
\end{align}$$

We can use this in our solution to the full gradient $\nabla_\beta l(\beta)$.

$$
\begin{aligned}
\nabla_\beta l(\beta) &= -\nabla_\beta \sum_{i=1}^{N} y_i ln(w_i) + (m_i - y_i) ln(1 - w_i) && (33)\\
&= -\sum_{i=1}^{N} y_i \nabla_\beta ln(w_i) + (m_i - y_i) \nabla_\beta ln(1 - w_i) && (34)\\
&= -\sum_{i=1}^{N} y_i \frac{1}{w_i} \nabla_\beta w_i + (m_i - y_i) \frac{1}{1 - w_i} \nabla_\beta (1 - w_i) && (35)\\
&= -\sum_{i=1}^{N} y_i \frac{1}{w_i} w_i (1 - w_i) x_i - (m_i - y_i) \frac{1}{1 - w_i} w_i (1 - w_i) x_i && (36)\\
&= -\sum_{i=1}^{N} y_i (1 - w_i) x_i - (m_i - y_i) w_i x_i && (37)\\
&= -\sum_{i=1}^{N} y_i x_i - y_i w_i x_i - m_i w_i x_i + y_i w_i x_i && (38)\\
&= -\sum_{i=1}^{N} y_i x_i - m_i w_i x_i && (39)\\
&= -\sum_{i=1}^{N} (y_i - m_i w_i) x_i && (40)\\
&= -(y - MW)^T X && (41)
\end{aligned}
$$