Exercises 1: Preliminaries

Devon Humphreys

1 Linear Regression

Given a simple linear model

$$\mathbf{y} = \mathbf{X}\beta + \epsilon\,,$$

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

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

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:

$$\frac{1}{2}\sum_{i=1}^{N} w_i(y_i - x_i^T\beta)^2 = \frac{1}{2}\sum_{i=1}^{N}(y_i - x_i^T\beta)w_i(y_i - x_i^T\beta)$$

$$= \frac{1}{2}\sum_{i=1}^{N}(y_i w_i y_i - 2y_i w_i x_i^T\beta + x_i^T\beta w_i x_i^T\beta)$$

$$= \frac{1}{2}\{\mathbf{y^T W y} - 2\mathbf{y^T W X}\beta + (\mathbf{X}\beta)^\mathbf{T}\mathbf{W}(\mathbf{X}\beta)\}\,.$$

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.

$$\nabla_\beta = \frac{1}{2}\{\frac{\partial}{\partial\beta}\mathbf{y^T W y} - 2\frac{\partial}{\partial\beta}\mathbf{y^T W X}\beta + \frac{\partial}{\partial\beta}(\mathbf{X}\beta)^\mathbf{T}\mathbf{W}(\mathbf{X}\beta)\} = 0$$

$$\frac{1}{2}\{0 - 2\mathbf{y^T W X} + 2\mathbf{X^T W X}\beta\} = 0$$

$$-\mathbf{y^T W X} + \mathbf{X^T W X}\beta = 0$$

$$\mathbf{X^T W X}\beta = \mathbf{X^T W y}$$

Therefore

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

(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:

$$\mathbf{X^T W y} = \mathbf{X^T W X}\beta\,.$$

Recognize that

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

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

Then

$$\mathbf{X^T W^{\frac{1}{2}} W^{12} y} = \mathbf{X^T W^{\frac{1}{2}} W^{\frac{1}{2}} X}\beta\,.$$

$$(\mathbf{QR})^{\mathbf{T}}\mathbf{W^{\frac{1}{2}} y} = (\mathbf{QR})^{\mathbf{T}}\mathbf{QR}\beta\,.$$

$$\mathbf{R^T Q^T W^{\frac{1}{2}} y} = \mathbf{R^T Q^T Q R} \beta \, .$$

$$\mathbf{Q^T W^{\frac{1}{2}} y} = \mathbf{R} \beta \, .$$

Pseudocode for SVD:

We will factor the design matrix $\mathbf{X}$ into orthogonal components and a diagonal matrix containing the "singular values":

$$\mathbf{X} = \mathbf{U \Sigma V^T}$$

$\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, and $\mathbf{\Sigma}$ is a diagonal matrix whose off-diagonal elements are 0.

Then we recognize that

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

$$= ((\mathbf{U \Sigma V^T})^{\mathbf{T}} \mathbf{W U \Sigma V^T})^{-1} (\mathbf{U \Sigma V^T}) \mathbf{W y}$$

...

(C) R code for SVD

```
library(microbenchmark)
library(ggplot2)
library(Matrix)
library(SparseM)

#inversion method
norm_regression = function(y, X, W){
  beta_hat = solve(t(X) %*% W %*% X) %*% t(X) %*% W %*% y
  return(beta_hat)
}

#singular value decomposition
svd_regression = function(y, X, W){
  sigma = diag(svd(X)$d)
  U = svd(X)$u
  V = svd(X)$v
  beta = V %*% solve(sigma) %*% t(U) %*% y
```

```
  return(beta)
}


#example matrices
N = 1000
P = 200

X = matrix(rnorm(N*P, mean = 0, sd = 1), nrow = N, ncol = P, byrow = T)
W = diag(rep(1, N))
y = rnorm(N, mean = 0, sd = 1)
mask = matrix(rbinom(N*P, 1, 0.05), nrow = N)
X = mask*X
X[1:10, 1:10]

#benchmarking
tmn = microbenchmark(norm_regression(y, X, W))
tmsvd = microbenchmark(svd_regression(y, X, W))
boxplot(tmn)
boxplot(tmsvd)
```

(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 $\mathbf{Ax} = \mathbf{b}$.

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

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

Letting $\mathbf{W} = \mathbf{I}$, we have that

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

In R:

```
#sparse matrix operations - solving Ax = b in a sparse A
qr_regression_sparse = function(y, X){
  Q = qr.Q(qr(X))
  R = qrR(qr(X))
  return(solve(R) %*% t(Q) %*% y)
}
```

2 Generalized Linear Models

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

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

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

$$p^n (1-p)^{1-n} \, .$$

First, let

$$w_i = \frac{1}{1+e^{-x_i\beta}}$$

Note that:

$$1 - w_i = 1 - \frac{1}{1+e^{-x_i\beta}} = \frac{1+e^{-x_i\beta}}{1+e^{-x_i\beta}} - \frac{1}{1+e^{-x_i\beta}} = \frac{e^{-x_i\beta}}{1+e^{-x_i\beta}}$$

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

$$\nabla_\beta w_i = \nabla_\beta \frac{1}{1+e^{-x_i\beta}}$$

By the quotient rule of derivatives, we have that

$$\nabla_\beta = \frac{-\nabla_\beta (1+e^{-x_i\beta})}{(1+e^{-x_i\beta})^2}$$

$$= \frac{x_i e^{-x_i\beta}}{(1+e^{-x_i\beta})^2}$$

$$= \frac{1}{1+e^{-x_i\beta}} \frac{e^{-x_i\beta}}{1+e^{-x_i\beta}} x_i$$

$$= w_i(1-w_i)x_i\,.$$

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

$$\nabla_\beta l(\beta) = -\nabla_\beta \sum_{i=1}^{N} y_i ln(w_i) + (m_i - y_i) ln(1-w_i)$$

$$= -\sum_{i=1}^{N} y_i \nabla_\beta ln(w_i) + (m_i - y_i)\nabla_\beta ln(1-w_i)$$

$$= -\sum_{i=1}^{N} y_i \frac{1}{w_i} \nabla_\beta w_i + (m_i - y_i)\frac{1}{1-w_i}\nabla_\beta(1-w_i)$$

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

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

$$= -\sum_{i=1}^{N} y_i x_i - y_i w_i x_i - m_i w_i x_i + y_i w_i x_i$$

$$= -\sum_{i=1}^{N} y_i x_i - m_i w_i x_i$$

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

$$= -(\mathbf{y} - \mathbf{MW})^{\mathbf{T}}\mathbf{X}$$