Linear Regression

1. Rewrite the WLS objective functions in terms of vectors and matrices as follows:

 (1)

Since W is a diagonal matrix of weights (so symmetric), then



To get the minimum of the above equation, take derivative of β with the following formulas 





1. Numerically speaking, I do not think inversion method is the fastest and most stable way to solve the linear system. Computing and applying the inverse matrix or pseudoinverse is a tremendously bad idea, since it is much more expensive and often numerically less stable than applying other algorithms, especially for high dimensional matrices.

Here are several ways to solve a system of linear equations Ax = b which provide more stability and are computationally efficient compared to inversion method.

1. Matrix decomposition: Gaussian or Gauss-Jordan elimination (considered as LU decomposition), Cholesky decomposition, QR decomposition, or SVD, and
2. Iterative method: conjugate gradient method.

Which method is optimal depends on the size and properties of the system matrix.

1. LU requires A to be square and performs well when A is sparse; it can be used for most linear systems.
2. Cholesky performs well for ***Hermitian positive-definite matrix*** A (A=LL\*, where L is a lower triangular matrix with real and positive diagonal entries, and L\* denotes the conjugate transpose of L)
3. QR decomposition requires A has linearly independent columns.
4. Conjugate gradient requires A to be symmetric and positive definite; it performs well when A is sparse and too large to be inverted directly for Cholesky.
5. Sinceis positive-definite, my method will be the Cholesky decomposition. To solve, here is the pseudo-code:

Function inputs:

X: N x P matrix

Y: N x 1 vector of responses

W: N x N diagonal matrix of weights

Function outputs:

: P x 1 vector of coefficient estimates

Pseudo-code:

1. Set *A = XTWX*;
2. Set *b = XTWy*;
3. Set *U* = Cholesky decomposition of *A*. Only the upper triangular part of *A* is used in R so that *A = UTU* when A is symmetric.
4. Solve ;
5. Solve ;
6. Returnas.

From the following R output, the inverse method is fastest for very small N and P values, but as N and P increase, LU and Cholesky methods perform much more quickly than inverse. LU is the most efficient method of the three and it is a partial Gaussian elimination method.

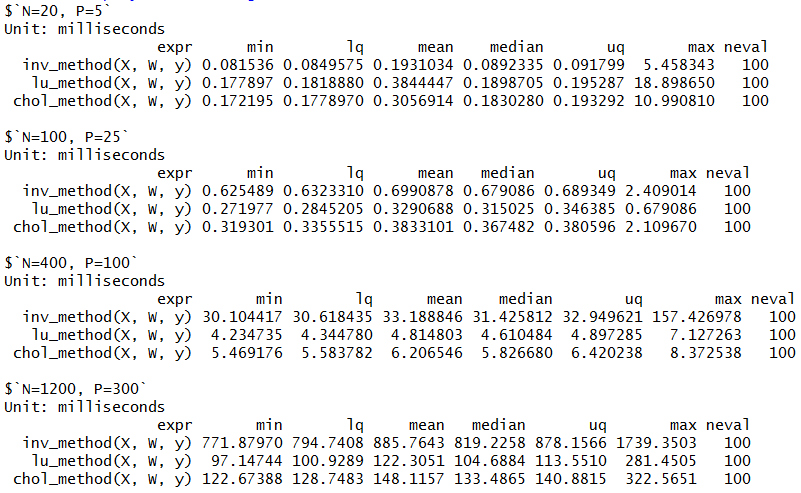


Figure 1: Performance benchmarking results

1. Since both LU and Cholesky perform well for sparse matrices, we benchmarked both of these methods and sparse Cholesky decomposition against inverse method for a sparse matrix with various sparsity level (1%, 5%, 25%). Here, *theta* represents the density of X, i.e., the proportion of entries which are non-zero. In the benchmark below, we can see more noticeable efficiency increase with higher sparsity and LU again performed the most efficiently.

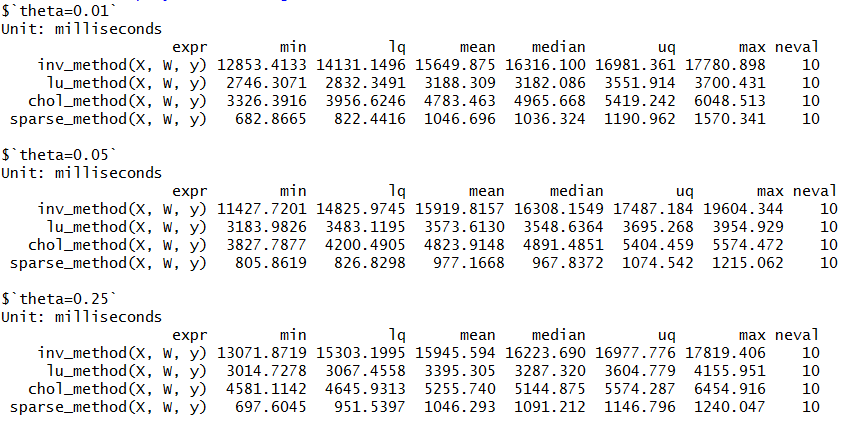


Figure 2: Benchmarking for various values of N, P, and density level

Generalized Linear Regression

1. The negative log likelihood is



With 

Taking derivative of w.r.t:



Since 









1. This is a one trial case ( sample size is 1), then the log likelihood function is



Here are a few notes about the R code:

1. Step size is fixed at stepsize = 0.01;
2. To handle probabilities close to 0 or 1, and a constant 10-5 to each log term in the log likelihood function;
3. Convergence is determined by using, where.

From the following table, we see that two sets of estimates are close.

Gradient descent method R: glm

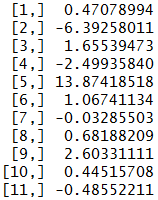
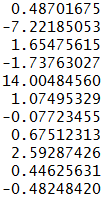
 

Figure 3: Comparison of results from Newton's method and gl

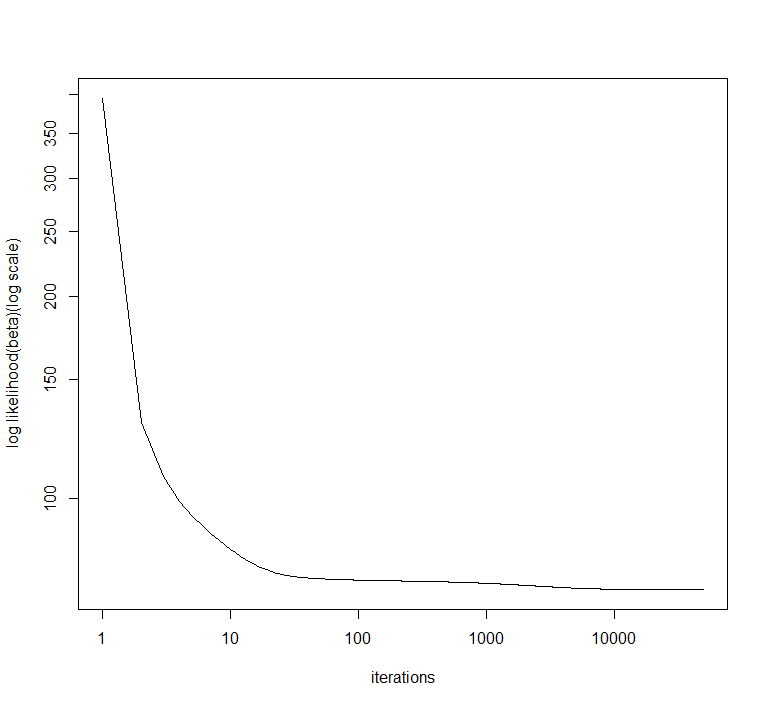


Figure 4: Log likelihood trace plot for gradient descent

1. First, we need to calculate the Hessian matrix of log likelihood function, which is a key part of Taylor series expansion.

From (A), we haveand 



Written in matrix form, 

where 

Let, then 

Recall Taylor series second-order expansion in general form:



Where, gradient evaluated at point a.

 Hessian evaluated at point a.

Then, we have 





So 



Where 



Where 

c\* is a constant that doesn’t involve β.

1. Here we use Newton's method to estimate. This iterative process requires far fewer iterations to achieve convergence since we are taking the curvature of the objective function  into account. We actually only use 10 iterations and achieve estimates which are exactly in line with estimates from ***glm***.

Newton's Method: 

Newton’s method R: glm

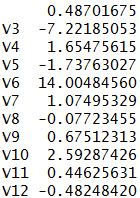
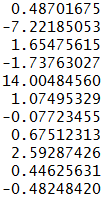
 

Figure 5: Comparison of results from Newton's method and glm

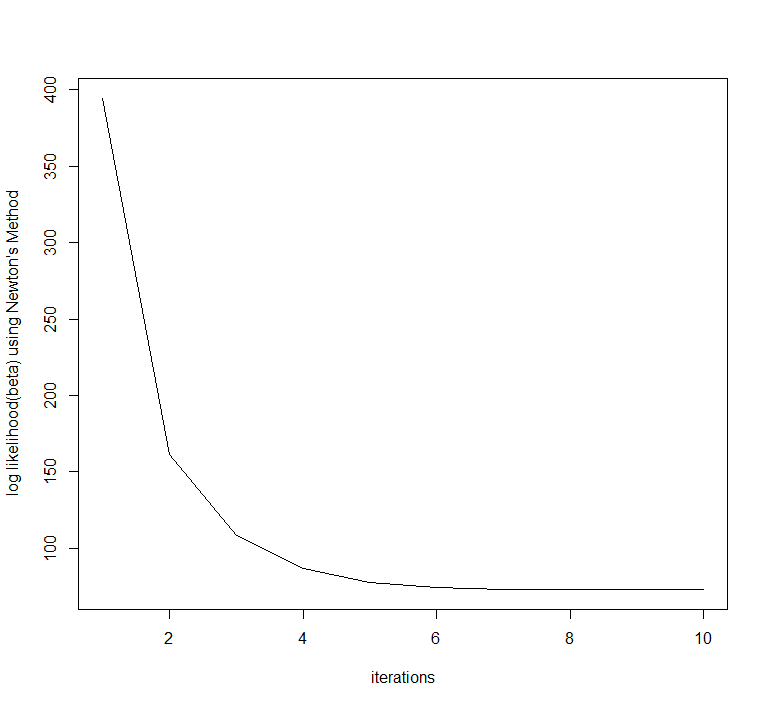


Figure 6: Log likelihood trace plot for Newton's method