Lec 4: Sweep Operator

Ailin Zhang

Agenda

- Gauss Jordan wrap up
- The Sweep Operator

Gauss Jordan

$$\mathrm{GJ}[1:n][A|b] = [I|A^{-1}b] = A^{-1}[A|b],$$
$$\mathrm{GJ}[1:n][A|I] = [I|A^{-1}] = A^{-1}[A|I].$$

Gauss Jordan

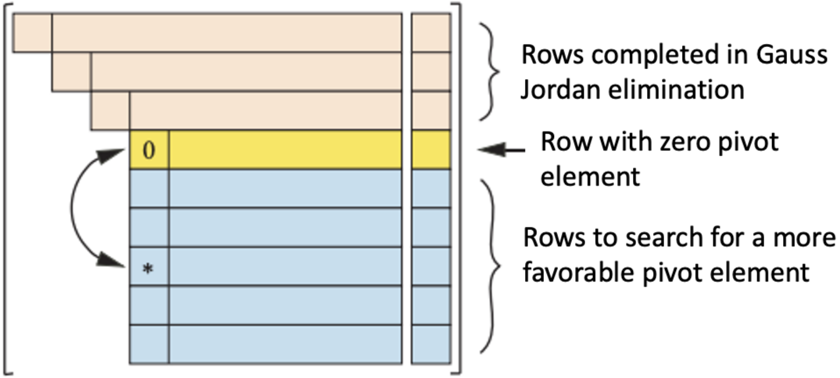
$$\mathrm{GJ}[1:n][A|b] = [I|A^{-1}b] = A^{-1}[A|b],$$
$$\mathrm{GJ}[1:n][A|I] = [I|A^{-1}] = A^{-1}[A|I].$$

Exercise:

Find A^{-1} where $A = \begin{bmatrix} 0 & 4 & -1 \\ 2 & -5 & 1 \\ 2 & -3 & 1 \end{bmatrix}$

Pivoting

In the previous example: Gauss-Jordan will not work because there is a zero in the pivot location, A_{11}.


0
*
Rows completed in Gauss Jordan elimination
Row with zero pivot element
Rows to search for a more favorable pivot element

Partial Pivoting: Exchange only rows

- Exchanging rows does not affect A^{-1}
- For increased numerical stability, make sure the largest possible pivot element is used. This requires searching in the partial column below the pivot element.
- Partial pivoting is usually sufficient.

To minimize the effect of roundoff, always choose the row that puts the largest pivot element on the diagonal, i.e., find i_p such that $|A_{i_p,i}| = max(|A_{k,i}|)$ for $k = i, \ldots, n$

Partial Pivoting:

Pivoting (that is row exchanges) can be expressed in terms of matrix multiplication

- Do pivoting during elimination, but track row exchanges in order to express pivoting with matrix P:

1. Let P be all zeros
2. Place a 1 in column j of row 1 to exchange row 1 and row j
3. If no row exchanged needed, place a 1 in column 1 of row 1
4. Repeat for all rows of P

P is a permutation matrix

Exercise

$$A = \begin{bmatrix} 2 & 4 & -2 & -2 \\ 1 & 2 & 4 & -3 \\ -3 & -3 & 8 & -2 \\ -1 & 1 & 6 & -3 \end{bmatrix}$$

Apply the elementary permutation matrix P_1 to permute the first and third rows of A.

Exercise

$$A = \begin{bmatrix} 2 & 4 & -2 & -2 \\ 1 & 2 & 4 & -3 \\ -3 & -3 & 8 & -2 \\ -1 & 1 & 6 & -3 \end{bmatrix}$$

Apply the elementary permutation matrix P_1 to permute the first and third rows of A.

$$P_1 * A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 4 & -2 & -2 \\ 1 & 2 & 4 & -3 \\ -3 & -3 & 8 & -2 \\ -1 & 1 & 6 & -3 \end{bmatrix} = \begin{bmatrix} -3 & -3 & 8 & -2 \\ 1 & 2 & 4 & -3 \\ 2 & 4 & -2 & -2 \\ -1 & 1 & 6 & -3 \end{bmatrix}$$

Partial Pivoting Example

Note that we can formally write,

$$(M_2 * P_2 * M_1 * P_1) * A = I$$

or

$$A = P_1^{-1} * M_1^{-1} * P_2^{-1} * M_2^{-1}$$

Gauss-Jordan performed on an arbitrary row

Matrix version of Gauss Jordan:

$$\mathrm{GJ}[1:m]\begin{bmatrix} A_{11} & A_{12} & \mid & I_1 & 0 \\ A_{21} & A_{22} & \mid & 0 & I_2 \end{bmatrix}$$

$$= \begin{bmatrix} I_1 & A_{11}^{-1}A_{12} & \mid & A_{11}^{-1} & 0 \\ 0 & A_{22} - A_{21}A_{11}^{-1}A_{12} & \mid & -A_{21}A_{11}^{-1} & I_2 \end{bmatrix}$$

$|A| = |A_{11}| \left| A_{22} - A_{21}A_{11}^{-1}A_{12} \right|$

Matrix Decomposition

- Theorem LUP decomposition

Any non-singular matrix A may be decomposed as:

$$LU = PA$$

where P is a permutation matrix, L is unit lower triangular, and U is upper triangular.

Application of LU Decomposition

$$\det(P) * \det(A) = \det(L) * \det(U)$$

$$(-1)^s * \det(A) = 1 * \prod_{i=1}^{n} U_{i,i}$$

where P is a permutation matrix and thus

$$\det(P) = \begin{cases} -1 & \text{if the number of elementary permutations is odd} \\ +1 & \text{if the number of elementary permutations is even} \end{cases}$$

L is unit lower triangular and thus,

$$\det(L) = 1$$

and U is upper triangular so

$$\det(U) = \text{ the product of the diagonal elements of } U$$

More Pivoting Strategies

- Full (or Complete) Pivoting: Exchange both rows and columns
    - Column exchange requires changing the order of the columns
    - For increased numerical stability, make sure the largest possible pivot element is used. This requires searching in the pivot row, and in all rows below the pivot row, starting the pivot column.
    - Full pivoting is less susceptible to roundoff, but the increase in stability comes at a cost of more complex programming and an increase in work associated with searching and data movement.

Matrix Decomposition

For linear system, $Ax = b$, one never actually computes $A^{-1}b$.

When $A = \mathbf{X}^\top \mathbf{X}$ is symmetric, some special computational methods are available:

- Sweep operator
- QR decomposition
- Cholesky decomposition

Sweep operator

The sweep operator has been introduced by Beaton (1964) as a tool for inverting symmetric matrices.

Sweep operator

The sweep operator arises from a close analysis of elimination applied to the equation $\mathbf{X}^\top X\beta = \mathbf{X}^\top \mathbf{Y}$. For a symmetric matrix$A$, $B = Sweep(A, k)$ is obtained as follows:

1. Divide k^{th} row and k^{th} column by a_{kk}:

$$b_{ik} = a_{ik}/a_{kk};\ b_{kj} = a_{kj}/a_{kk}$$

2. Subtract $a_{ik}a_{kj}/a_{kk}$ from the other entries:

$$b_{ij} = a_{ij} - a_{ik}a_{kj}/a_{kk}$$

3. Invert k^{th} diagonal element:

$$b_{kk} = -1/a_{kk}$$

Sweep operator

The sweep operator arises from a close analysis of elimination applied to the equation $\mathbf{X}^\top X\beta = \mathbf{X}^\top \mathbf{Y}$. For a symmetric matrix$A$, $B = Sweep(A, k)$ is obtained as follows:

1. Divide k^{th} row and k^{th} column by a_{kk}:
$$b_{ik} = a_{ik}/a_{kk};\ b_{kj} = a_{kj}/a_{kk}$$
2. Subtract $a_{ik}a_{kj}/a_{kk}$ from the other entries:
$$b_{ij} = a_{ij} - a_{ik}a_{kj}/a_{kk}$$
3. Invert k^{th} diagonal element:
$$b_{kk} = -1/a_{kk}$$

Question: What is the time complexity?

Sweep operator

The sweep operator arises from a close analysis of elimination applied to the equation $\mathbf{X}^\top X\beta = \mathbf{X}^\top \mathbf{Y}$. For a symmetric matrix$A$, $B = Sweep(A, k)$ is obtained as follows:

1. Divide k^{th} row and k^{th} column by a_{kk}:

$$b_{ik} = a_{ik}/a_{kk};\ b_{kj} = a_{kj}/a_{kk}$$

2. Subtract $a_{ik}a_{kj}/a_{kk}$ from the other entries:

$$b_{ij} = a_{ij} - a_{ik}a_{kj}/a_{kk}$$

3. Invert k^{th} diagonal element:

$$b_{kk} = -1/a_{kk}$$

Question: What is the time complexity? A single sweep operation visits every element of the matrix and requires roughly one divide per element. This leads to roughly $\mathcal{O}(n^2)$ divides.

R code

```r
mySweep <- function(A, m)
{
n <- dim(A)[1]
for (k in 1:m)
{
for (i in 1:n)
 for (j in 1:n)
    if (i!=k  & j!=k)
       A[i,j] <- A[i,j] - A[i,k]*A[k,j]/A[k,k]
for (i in 1:n)
  if (i!=k)
    A[i,k] <- A[i,k]/A[k,k]
for (j in 1:n)
  if (j!=k)
    A[k,j] <- A[k,j]/A[k,k]
A[k,k] <- - 1/A[k,k]
}
return(A)
}
A = matrix(c(1,2,3,7,11,13,17,21,23), 3,3)
solve(A)
```

```
##       [,1] [,2]  [,3]
## [1,]  1.00 -3.0  2.00
## [2,] -0.85  1.4 -0.65
## [3,]  0.35 -0.4  0.15
```

```r
mySweep(A,3)
```

Proposition 1

Proposition

Suppose $V_{p\times m} = U_{p\times m}A_{m\times m}$*, and* $B = Sweep(A, k)$*. Then* $\hat{V} = \hat{U}B$*, where:*

- $\hat{U} = U$ *except that its* k^{th} *column is* v_k*;*
- $\hat{V} = V$ *except that its* k^{th} *coumn is* $-u_k$*.*

Proof:

Proposition 2

Proposition

If A is a symmetric invertible matrix and we sweep on each diagonal element of A the result is $B = Sweep(A, 1:n) = -A^{-1}$

Proof:

Proposition 3

Proposition

If $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ *and we sweep on each of the diagonal entries of* A_{11}*, we get* $B = \begin{bmatrix} -A_{11}^{-1} & A_{11}^{-1}A_{12} \\ A_{21}A_{11}^{-1} & A_{22} - A_{21}A_{11}^{-1}A_{12} \end{bmatrix}$

Proof: Start with an simple example $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$

Proposition 3

Proposition

If $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ *and we sweep on each of the diagonal entries of* A_{11}*, we get* $B = \begin{bmatrix} -A_{11}^{-1} & A_{11}^{-1}A_{12} \\ A_{21}A_{11}^{-1} & A_{22} - A_{21}A_{11}^{-1}A_{12} \end{bmatrix}$

Proof: Take advantage of proposition 2.

Proposition 4

We construct a matrix $\mathbf{Z} = [\mathbf{X}\mathbf{Y}]$, and let

$$A = \mathbf{Z}^\top \mathbf{Z} = \begin{bmatrix} \mathbf{X}^\top\mathbf{X} & \mathbf{X}^\top\mathbf{Y} \\ \mathbf{Y}^\top\mathbf{X} & \mathbf{Y}^\top\mathbf{Y} \end{bmatrix}$$

be the cross-product matrix. Then

$$\mathrm{SWP}[1:p]A = \begin{bmatrix} -(\mathbf{X}^\top\mathbf{X})^{-1} & (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y} \\ \mathbf{Y}^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1} & \mathbf{Y}^\top\mathbf{Y} - \mathbf{Y}^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y} \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{\mathrm{Var}(\hat\beta)}{\sigma^2} & \hat\beta \\ \hat\beta^\top & \mathrm{RSS} \end{bmatrix}$$

where $\mathrm{RSS} = \|\mathbf{Y} - \mathbf{X}\hat\beta\|_{\ell_2}^2$ is the residual sum of squares.

Other Application of sweep operator

Perform sweep operator on multivariate normal distribution

$$\begin{bmatrix} \Omega & x - \mu \\ x^t - \mu^t & 0 \end{bmatrix}$$

Why Sweep Operator?

1. Efficient for computing the central statistics used in multiple regression
   - Start from a square correlation or covariance matrix.
   - Compute multiple correlation, residual variance, regression slopes, and standard errors of slopes, plus some other
2. An efficient way to compute a whole series of regressions, as in stepwise regression.
3. (Drawback) If it is applied many times to the same matrix, rounding error can accumulate.