LU decomposition (PA=LU, PAQ=LU) and QR factorization

Name : Anubothula Hema Priya
Entry Number : 2020MCB1228
Course : MA205 - Computational Lab
Professor : Dr. Manoranjan Mishra Sir

LU Decomposition

Introduction

LU decomposition of a matrix is a factorization of a given square matrix into two triangular matrices(one lower triangular and another upper triangular) such that product of these matrices is original matrix.

Mathematically it can be written as ,

$$A = LU;$$

where L = lower triangular matrix

U= Upper triangular matrix

Does LU Decomposition exists for every matrix?

Let

$$\Delta_1 = a_{11}, \Delta_2 = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}, \ldots, \Delta_n = \begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{vmatrix}$$

so called leading principal minors of the matrix.

The LU decompositions exists iff $\Delta_i \neq 0, i \in \{1, \ldots, n\}$

Theorem

If A is a Non- Singular Matrix,

∃ a permutation matrix P such that PA has triangular factorisation.

$$i.e., PA = LU$$

–Note: A be any square matrix and P be a permutation matrix if same order,

Then PA is the matrix that is obtained by interchanging rows of A.

Motivation

LU decomposition is a more efficient approach than Gauss elimination, especially when solving several equations with the same left-hand side. That is, for solving Ax = b with various b values for the same A. Note that in Gauss elimination both left hand side and right hand side are modified at the same time, that means if we want to solve the same equations for different values of b,the elimination step has to be done all over again. So, it would be more efficient if we can modify only A and save the results and use them repeatedly with different values of b. This provides the motivation for LU Decomposition.

LU Decomposition with partial and complete pivoting

The A = LU factorization discussed above assumes that there are no row interchanges. It is possible that a non singular matrix A cannot be factored directly as A = LU. For that we have to perform Indirect Factorization.

Indirect Factorization : Let A be a given N*N matrix. Assume that Gaussian elimination can be performed successfully to solve the linear system AX = B, but that row interchanges are required. Then there exists a permutation matrix P such that the product PA can be factored as a product of lower triangular matrix (L) and upper triangular matrix (U).

$$PA = LU$$

LU Decomposition with partial and complete pivoting

If both row and column interchanges are required. Then there exists two permutation matrices P and Q such that the product PAQ can be factored as a product of lower triangular matrix (L) and upper triangular matrix (U).

$$PAQ = LU$$

Detailed Algorithm

PA = LU

- Take a square matrix A.
- Now, Define L = identity matrix of same order as A.

  P = identity matrix of same order as A.

  U = A.

- do for i = 1, 2, ..., col-1

do for j = i+1, i+2, ..., row

if (|U(j, i)| ≥ |U(i, i)|)

Rj ↔ Ri in matrix U

Pj ↔ Pi

Set vec = 1:i-1

Now ,exchange row vectors L(i,vec) <-> L(l,vec)

end if

if (U(i,i) /= 0

$\lambda = U(j, i)/U(i, i)$

$Rj \longrightarrow Rj - \lambda\, Ri$

$L(j, i) = \lambda$

else Stop.

end if

end do

end do

L,U and corresponding permutation matrix P are obtained s.t,

$PA = LU$

PAQ = LU

- Take a square matrix A.

- Now, Define L = identity matrix of same order as A.
  P = identity matrix of same order as A.
  Q = identity matrix of same order as A.
  U = A.
- do for i = 1, 2, ..., col-1
  Search for absolute max element in U(i:col,i:row) subarray (l,m) be corresp. row and column position of that element

  if l ~= i or m ~= i
  if m ~= i
  Column(i) <-> Column(m) - in U
  Column(i) <-> Column(m) - in Q
  End if

```
if l ~= i
        R(i) <-> R(l) in  U
        R(i) <-> R(l) in  P

        Set vec = 1:i-1
        Now ,exchange row vectors   L(i,vec) <-> L(l,vec)
      End if

  End if

  do  for  j = i+1 : row
     Set   L(j,i)  as  U(j,i)/U(i,i);
      To update U, row(j) <-> row(j) - L(j,i)*row(i) in matrix U
  End do

End do
```

L,U and corresp.permutation matrices P,Q are obtained s.t,

PAQ = LU

Time Complexity of LU Decomposition

Eliminating the first column will require n additions and n multiplications for $n-1$ rows. Therefore, the number of operations for the first column is $2n(n-1)$. For the second column, we have $n-1$ additions and $n-1$ multiplication, and we do this for $(n-2)$ rows giving us $2(n-1)(n-2)$.

Therefore, the total number of operations required for the full decomposition can be written as

Time Complexity of LU Decomposition

$$\sum_{i=1}^{n} 2(n-i)(n-i+1) = 2\sum_{j=0}^{n-1} j(j+1)$$

$$= 2\sum_{j=0}^{n-1} (j^2 + j)$$

$$= 2\left(\frac{1}{3}n^3 - \frac{1}{3}n\right)$$

Thus it is , **O(n^3)**

For partial, complete pivoting

PA = LU

For an n×n matrix A, we scan n rows of the first column for the largest value. At step k of the elimination, the pivot we choose is the largest of the n- (k+1) subdiagonal entries of column k, which costs O(nk) operations for each step of the elimination. So for an n-by-n matrix, there is a total of O(n2) comparisons.

PAQ = LU

When employing the complete pivoting strategy we scan for the largest value in the entire submatrix Ak:n,k:n, where k is the number of the elimination, instead of just the next subcolumn. This requires O((n−k)^2) comparisons for every step in the elimination to find the pivot. The total cost becomes O(n^3) comparisons for an n-by-n matrix.

Partial Vs Complete pivoting

Partial pivoting is the most common pivoting strategy because it is the cheapest, fastest algorithm to use while sacrifing very little accuracy. In practice, partial pivoting is just as accurate as complete pivoting.

Complete pivoting is theoretically the most stable strategy as it can be used even when a matrix is singular, but it is much slower than partial pivoting.

Advantages of LU Decomposition over Gauss Elimination

In many engineering applications,when you solve Ax = b, the matrix A remains unchanged, while the right hand side vector b keeps changing. The key idea behind solving using the LU factorization is to decouple the factorization phase from the actual solving phase. The factorization phase only needs the matrix A, while the actual solving phase makes use of the factored form of A and the right hand side to solve the linear system. Hence, once we have the factorization, we can make use of the factored form of A, to solve for different right hand sides at a relatively moderate computational cost.

Advantages of LU Decomposition over Gauss Elimination

The cost of factorizing the matrix A into LU is O(N3). Once you have this factorization, the cost of solving i.e. the cost of solving LUx=b is just O(N2), since the cost of solving a triangular system scales as O(N2). Hence, if you have 'r' right hand side vectors {b1,b2,…,br}, once you have the LU factorization of the matrix A, the total cost to solve Ax1=b1, Ax2=b2,…,Axr=br scales as O(N3+rN2).

On the other hand, if you do Gauss elimination separately for each right hand side vector bj, then the total cost scales as O(rN3), since each Gauss elimination independently costs O(N3).

Real life model examples

Determinant of a Matrix : Determinant of a matrix calculation becomes easier if we already have the LU decomposition of a matrix A = LU, we know that det(A) = det(L)det(U).

Solving a Linear System: Ax = b becomes trivial having LU factorization of a matrix, A = LU. We can solve Ly = b first, then we can solve Ux = y in a similar fashion. To find inverse of a matrix A, taking AX = I and by solving this system will easily give the result.

Real life model examples

The most common way of solving Linear Regression is via a least squares optimization that is solved using matrix factorization methods from linear regression, such as an LU decomposition or an SVD (Singular Value Decomposition, a dimensionality reduction method)

QR Factorization

Introduction

QR decomposition of a matrix is the factorization of a given square matrix into an orthogonal matrix Q and an upper triangular matrix R such that the product of these two matrices gives the original matrix.

There are several methods for actually computing the QR decomposition, such as by means of the Gram–Schmidt process, Householder transformations, or Givens rotations. Each has a number of advantages and disadvantages. We have computed it using the Gram–Schmidt process. A square matrix A can be decomposed into two square matrices Q and R such that A = QR where R is an upper triangular matrix and Q is an orthogonal matrix.

Motivation behind the designing of QR factorization

The QR decomposition is frequently used to solve the linear least squares problem and is the foundation for the QR algorithm.

The least squares approximation of linear functions to data is known as linear least squares (LLS). It is a collection of formulations for addressing linear regression statistical problems, including variants for ordinary (unweighted), weighted, and generalized (correlated) residuals. Inverting the matrix of the normal equations and orthogonal decomposition methods are two numerical methods for linear least squares.

Designing efficient and stable strategies for identifying the eigenvalues of a matrix is one of the most important challenges in numerical analysis. Eigenvectors can be found using these eigenvalue techniques.Therefore, the QR algorithm, also known as QR iteration, is an eigenvalue algorithm that calculates a matrix's eigenvalues and eigenvectors.

General method of Gram-Schmidt QR factorization

A = QR, where Q and R are orthogonal and upper triangular matrices respectively. If A is invertible, then the factorization is unique if we require the diagonal elements of R to be positive. If A has n linearly independent columns, then the first n columns of Q form an orthonormal basis for the column space of A.

For any orthogonal matrix Q, the following condition holds :

QT Q = I implies QT = inv(Q) (QT = Transpose of matrix Q)

A = [a1……an] , Q = [q1,.....,qn] , R = [r1,.....,rn]

Here, a1 =[a11 ; a21 ; ……; an1]

a2 =[a12 ; a22 ; ……; an2] and so on….

an =[a1n ; a2n ; ……..; ann]

Here, r1 = [||a1|| ; 0 ; 0 ;.....; 0]

r2 = [<a2,q1> ; ||Per(a2)|| ;.....; 0] and so on….

rn = [<an,q1> ; <an,q2> ;.....; ||Per(an)||]

and || an || = Norm of nth column of matrix A = sqrt(a1n^2 + a2n^2 +..... +ann^2)

<an,q1> = Dot product of an and q1 (also called inner product)

Per(an) = Perpendicular projection of an on q1, q2, q3,..... qn-1.

1) We compute q1 = a1/(|| a1 ||)

Here, || a1 || represents the norm of 1st column of A.

r11 = || a1 ||

2) We calculate the inner product <a2,q1>, we get r21

r21 = <a2,q1>

3) Next, we calculate Per(a2), because we need q2 to be orthogonal to q1.

Per(a2) = a2 - (<a2,q1>)*q1

Algorithm

Let A be a square matrix, A= [aij]

Step 1: Read the matrix A = [aij], i,j = 1, 2, ….n

Step 2: Initialize Q and R to be identity matrices of dimension as same as A

Q = I(n,n), R = I(n,n)

Step 3: Initialize X to be a zero column matrix of same order as A

X = zeros(n,1)

Step 4: For i= 1,2…,n, j =1

Let b1 = || A(1:n,j) || ;

```
                Q(1:n,j) = A(1:n,j)/b1 ;
                   R(1,1) = b1;
Step 5:  For i = 2,3,.....,n
            Initialize g to be zero column matrix of n
                  g= zeros(n,1)
               For j = 1,......,i-1
                     g = g +(dot(A(1:n,i),Q(1:n,j)))*Q(1:n,j)
                     Here, dot represents dot product of two columns.
                     R(j,i)= dot(A(1:m,i),Q(1:m,j))
                end
   X = A(1:m,i) - g ;
   Q(1:m, i) = X/norm(X,2) ;
   R(i,i) = norm(X,2);
   end
```

Example

Consider A = [1 0 0; 1 1 0 ; 1 1 1]

Here, a1 = (1 ; 1; 1) , a2 = (0; 1; 1), a3 = (0; 0; 1)

We compute this by above mentioned process(Gram-schmidt)

Q = (q1; q2; q3), we obtain

q1 = [0.5774; 0.5774; 0.5774],

q2 = [-0.8165; 0.4082; 0.4082],

q3 = [0; -0.7071; 0.7071]

R = (r1; r2; r3), we obtain

Example

r1 = [1.7321 r2 = [1.1547 r3 = [0.5774
0 0.8165 0.4082
0] 0] 0.7071]

Here q1, q2 and q3 are orthonormal to each other.

Now, if we compute the product of Q and R, we get the original matrix.

Computational Cost

If we choose a square matrix of order n, then the computational cost will be of order O(2n^3).

For i = 2, j = 1(for j there are 4n iterations)

For i =3, j =1, 2 and so on……..

For i =n, j = 1,2,.....,n-1

So, Computational cost = O((1+2+.....+n-1)*4n)

= O(n(n-1)*2n)

= O(2n^3)

Comparison of QR factorization Vs Householder reflections

Description of Householder reflections:

A Householder reflection is a transformation that takes a vector and reflects it about a plane. We can use this to calculate QR factorisation of an m x n matrix (m >= n). X is an arbitrary m dimensional vector of A such that ||X|| = |α|, here α is scalar.

$$\alpha = -e^{i \arg x_k} \left\| \mathbf{x} \right\|$$

Where, e1 = [1, 0…..,0]T (Transpose of matrix)

$$\begin{aligned}\mathbf{u} &= \mathbf{x} - \alpha \mathbf{e}_1, \\ \mathbf{v} &= \frac{\mathbf{u}}{\|\mathbf{u}\|}, \\ Q &= I - 2\mathbf{v}\mathbf{v}^{\mathsf{T}}.\end{aligned}$$

Here, Q is the m x m householder matrix, both symmetric and orthogonal.

$$Q\mathbf{x} = \begin{bmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

This can be used to convert an m-by-n matrix A to upper triangular form gradually. To begin, multiply A by the Householder matrix Q1 obtained by selecting the first matrix column for x. As a result, the left column of the matrix Q1A contains zeros (except for the first row).

$$Q_1 A = \begin{bmatrix} \alpha_1 & \star & \cdots & \star \\ 0 & & & \\ \vdots & & A' & \\ 0 & & & \end{bmatrix}$$

Thus, we get a iterative matrix,

$$Q_k = \begin{bmatrix} I_{k-1} & 0 \\ 0 & Q'_k \end{bmatrix}.$$

After t iterations of this process, t = min(m-1,n)

$$R = Q_t \cdots Q_2 Q_1 A$$

Here, R is an upper triangular matrix. Now Q is

$$Q = Q_1^{\mathsf{T}} Q_2^{\mathsf{T}} \cdots Q_t^{\mathsf{T}} = Q_1 Q_2 \cdots Q_t,$$

Therefore , A = QR is a QR decomposition of A.

Now, the computational cost for the Householder algorithm takes O(2/3n^3).

Advantages/Disadvantages of Gram-Schmidt over Householder Method

Advantages :

It is very easy to implement, which makes it a useful algorithm.

Disadvantages :

Therefore, by comparing the time complexity, the householder method is numerically more stable than the Gram-Schmidt method. Householder method provides good orthogonality as compared to Gram-Schmidt method.

Real life model application

Real time face recognition:

A real-time face recognition is done based on Gram-Schmidt orthogonalization for linear discriminant analysis (GSLDA). The GSLDA technique avoids big matrices computations, such as computing the inverse or diagonalization of matrices, which can be time-consuming and inaccurate. GSLDA, on the other hand, overcomes the degenerate eigenvalue problem of linear discriminant analysis (LDA) to produce superior recognition performance than LDA. The proposed method's superior performance has been proven by experimental findings on genuine face databases.

References for QR Factorization

1) https://en.wikipedia.org/wiki/Gram%E2%80%93Schmidt_process
2) https://youtu.be/FAnNBw7d0vg
3) https://youtu.be/qmRC8mTPGI8
4) https://people.math.harvard.edu/~knill/teaching/math21b2017/14-gramschmidt.pdf#:~:text=The%20Gram-Schmidt%20process%20is%20tied%20to%20the%20factorization,matrices%20in%20every%20time%20step%20of%20the%20evolution.

Thanks you sir !