CHAPTER IV
Matrix Decompositions

PHOK Ponna and NEANG Pheak

Institute of Technology of Cambodia
Department of Applied Mathematics and Statistics (AMS)

2022–2023

Contents

Contents

Many complex matrix operations cannot be solved efficiently or with stability using the limited precision of computers. Matrix decompositions are methods that reduce a matrix into constituent parts that make it easier to calculate more complex matrix operations. Matrix decomposition methods, also called matrix factorization methods, are a foundation of linear algebra in computers,even for basic operations such as solving systems of linear equations, calculating the inverse, and calculating the determinant of a matrix.

Definition 1

A **matrix decomposition** is a way of reducing a matrix into its constituent parts. It is an approach that can simplify more complex matrix operations that can be performed on the decomposed matrix rather than on the original matrix itself. A common analogy for matrix decomposition is the factoring of numbers, such as the factoring of 10 into 2×5. For this reason,matrix decomposition is also called **matrix factorization**. Like factoring real values, there are many ways to decompose a matrix, hence there are a range of different matrix decomposition techniques. Two simple and widely used matrix decomposition methods are the LU matrixdecomposition and the QR matrix decomposition. Next, we will take a closer look at each ofthese methods.

Contents

The LU decomposition is for square matrices and decomposes a matrix into L and U components.

$$A = L.U$$

Or, without the dot notation.

$$A = LU$$

Where A is the square matrix that we wish to decompose, L is the lower triangle matrixand U is the upper triangle matrix.
The LU decomposition is found using an iterative numerical process and can fail for those matrices that cannot be decomposed or decomposed easily. A variation of this decomposition that is numerically more stable to solve in practice is called the LUP decomposition, or the LU decomposition with partial pivoting.

$$A = L.U.P$$

The rows of the parent matrix are re-ordered to simplify the decomposition process and the additional P matrix specifies a way to permute the result or return the result to the original order. There are also other variations of the LU. The LU decomposition is often used to simplify the solving of systems of linear equations, such as finding the coefficients in a linear regression, as well as in calculating the determinant and inverse of a matrix.
The LU decomposition can be implemented in Python with the lu() function. More specifically, this function calculates an LPU decomposition. The example below first defines a 3×3 square matrix. The LU decomposition is calculated, then the original matrix is reconstructed from the components.

```
# LU decomposition
from numpy import array
from scipy.linalg import lu
# define a square matrix
A = array([
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]])
print(A)
# factorize
```

```
P, L, U = lu(A)
print(P)
print(L)
print(U)
# reconstruct
B = P.dot(L).dot(U)
print(B)
```

Running the example first prints the defined 3×3 matrix, then the P, L, and U components of the decomposition, then finally the original matrix is reconstructed.

Contents

The QR decomposition is for $m \times n$ matrices (not limited to square matrices) and decomposes a matrix into Q and R components.

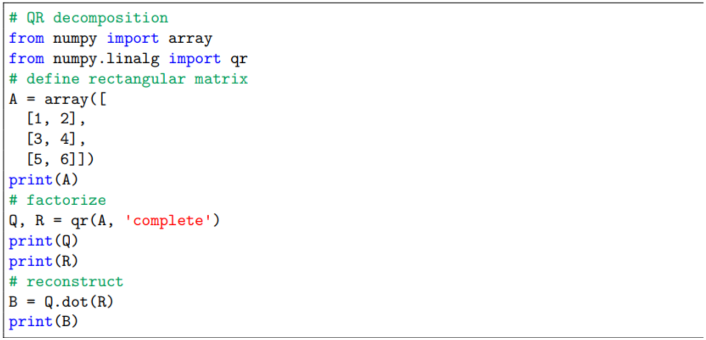
$$A = Q.R$$

Or, without the dot notation.

$$A = QR$$

Where A is the matrix that we wish to decompose, Q a matrix with the size $m \times m$, and R is an upper triangle matrix with the size $m \times n$. The QR decomposition is found using an iterative numerical method that can fail for those matrices that cannot be decomposed, or decomposed easily. Like the LU decomposition, the QR decomposition is often used to solve systems of linear equations, although is not limited to square matrices.

The QR decomposition can be implemented in NumPy using the qr() function. By default, the function returns the Q and R matrices with smaller or reduced dimensions that is more economical. We can change this to return the expected sizes of $m \times m$ for Q and $m \times n$ for R by specifying the mode argument as ‘complete’, although this is not required for most applications. The example below defines a 3×2 matrix, calculates the QR decomposition, then reconstructs the original matrix from the decomposed elements.



```
# QR decomposition
from numpy import array
from numpy.linalg import qr
# define rectangular matrix
A = array([
  [1, 2],
  [3, 4],
  [5, 6]])
print(A)
# factorize
Q, R = qr(A, 'complete')
print(Q)
print(R)
# reconstruct
B = Q.dot(R)
print(B)
```

Contents

The **Cholesky decomposition** is for square symmetric matrices where all values are greater than zero, so-called positive definite matrices. For our interests in machine learning, we will focus on the Cholesky decomposition for real-valued matrices and ignore the cases when working with complex numbers. The decomposition is defined as follows:

$$A = L.L^T$$

Or without the dot notation:

$$A = LL^T$$

Where A is the matrix being decomposed, L is the lower triangular matrix and L^T is the transpose of L. The decompose can also be written as the product of the upper triangular matrix, for example:

$$A = U^T.U$$

Where U is the upper triangular matrix.

The Cholesky decomposition is used for solving linear least squares for linear regression, as well as simulation and optimization methods.
When decomposing symmetric matrices, the Cholesky decomposition is nearly twice as efficient as the LU decomposition and should be preferred in these cases.
While symmetric, positive definite matrices are rather special, they occur quite frequently in some applications, so their special factorization, called Cholesky decomposition, is good to know about.
When you can use it, Cholesky decomposition is about a factor of two faster than alternative methods for solving linear equations.

The Cholesky decomposition can be implemented in NumPy by calling the cholesky()function. The function only returns L as we can easily access the L transpose as needed. The example below defines a 3×3 symmetric and positive definite matrix and calculates the Cholesky decomposition, then the original matrix is reconstructed.

```
# Cholesky decomposition
from numpy import array
from numpy.linalg import cholesky
# define symmetrical matrix
A = array([
  [2, 1, 1],
  [1, 2, 1],
  [1, 1, 2]])
print(A)
# factorize
L = cholesky(A)
print(L)
# reconstruct
B = L.dot(L.T)
print(B)
```

Running the example first prints the symmetric matrix, then the lower triangular matrix from the decomposition followed by the reconstructed matrix.