Matrix Decompositions

Matrix Algebra for Data Analysis

Gaston **S**anchez

gastonsanchez.com

Readme

Introduction

Decompositions

Matrix decompositions, also known as matrix factorizations, allow us to express a matrix $\mathbf{X}$ as the product of a number of simpler matrices —usually 2 or 3— :

$$\mathbf{X} = \mathbf{AB}$$

or

$$\mathbf{X} = \mathbf{ABC}$$

Introduction (con't)

Importance

Matrix decompositions make it easier to study the properties of matrices. Likewise, many computation tasks become easier with decompositions.

Typical Applications

- solving systems of linear equations
- inverting a matrix
- analyzing numerical stability of a system
- understanding the structure of data
- finding basis for column space (or row space) of a matrix
- *etc*

Matrix Decompositions

We will cover the following decompositions

- Cholesky decomposition
- LU decomposition
- Spectral or Eigenvalue Decomposition (EVD)
- QR decomposition
- Singular Value Decomposition (SVD)

Some Assumptions

Real Matrices

For this lecture all matrices will be assumed to be real matrices, i.e. matrices containing elements in the set of Real numbers.

Rectangular $n > p$

Also for the sake of simplicity, we'll assume rectangular matrices as "tall" matrices, that is, more rows than columns.

Matrix Considerations

Focus

We will focus on **general rectangular matrices** and on **symmetric matrices** because of their relevance for multivariate statistics.

Specifically

- For symmetric matrices we'll consider **Cholesky**, **LU** and **EVD** decompositions.
- For general rectangular matrices we'll consider **QR** and **SVD** decompositions.

Matrix Considerations (con't)

Symmetric Matrices

For symmetric matrices $\mathbf{X}$ we typically look at decompositions of the form $\mathbf{X} = \mathbf{A}\mathbf{A}'$ or $\mathbf{X} = \mathbf{B}'\mathbf{B}$

General Rectangular Matrices

For rectangular matrices $\mathbf{X}$ we look at decompositions of the form $\mathbf{X} = \mathbf{A}\mathbf{B}$ or $\mathbf{X} = \mathbf{A}\mathbf{B}\mathbf{C}$

Important Concepts

Full Rank

If $\mathbf{X}$ is an $n \times p$ matrix, then $\mathbf{X}$ is of **full rank** if $\mathbf{Xb} = \mathbf{0}$ only if $\mathbf{b} = \mathbf{0}$.

Linear Independence

We also say the columns of $\mathbf{X}$ are *linearly independent*. Full row rank is defined in a similar way.

Important Concepts (con't)

Full Rank Decomposition

If $\mathbf{X}$ can be written as the product $\mathbf{AB}$, with $\mathbf{A}$ an $n \times k$ matrix of full rank, and $\mathbf{B}$ a $k \times p$ matrix of full rank, then $\mathbf{X}$ is said to have rank k.

The decomposition $\mathbf{X} = \mathbf{AB}$ is a *full rank* decomposition

Cholesky Decomposition

Cholesky Decomposition

Cholesky Decomposition

A real symmetric matrix (*p.d. or p.s.d.*) $\mathbf{X}$ can be decomposed as:

$$\mathbf{X} = \mathbf{L}\mathbf{L}'$$

where $\mathbf{L}$ is a lower triangular matrix with real diagonal entries.

Equivalently

The Cholesky decomposition of $\mathbf{X}$ can also be expressed as $\mathbf{X} = \mathbf{R}'\mathbf{R}$ with $\mathbf{R}$ an upper triangular matrix.

Cholesky Decomposition (con't)

For Positive Definite Matrices

If $\mathbf{X}$ is a *positive definite* matrix then $\mathbf{L}$ is a lower triangular matrix with real and positive diagonal entries.

For Positive Semi-Definite Matrices

If $\mathbf{X}$ is a *positive semi-definite* matrix then $\mathbf{L}$ is a lower triangular matrix with real and non-negative diagonal entries.

Cholesky Decomposition

$$\mathbf{X} = \mathbf{L}\mathbf{L}'$$

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{p1} & x_{p2} & \cdots & x_{pp} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{p1} & l_{p2} & \cdots & l_{pp} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & \cdots & l_{p1} \\ 0 & l_{22} & \cdots & l_{p2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & l_{pp} \end{bmatrix}$$

$$\mathbf{X} = \mathbf{R}'\mathbf{R}$$

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{p1} & x_{p2} & \cdots & x_{pp} \end{bmatrix} = \begin{bmatrix} r_{11} & 0 & \cdots & 0 \\ r_{12} & r_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ r_{1p} & r_{2p} & \cdots & r_{pp} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1p} \\ 0 & r_{22} & \cdots & r_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{pp} \end{bmatrix}$$

Cholesky Decomposition in R

`chol()` function

In R we have the function `chol()` that performs the Cholesky decomposition.

`chol()` output

The output of `chol()` is the matrix $\mathbf{R}$ (upper triangular) such that $\mathbf{X} = \mathbf{R}'\mathbf{R}$. If the argument `pivot = TRUE`, then `"pivot"` and `"rank"` are also returned.

Cholesky Decomposition in R

Apply `chol()`

```
# X matrix (positive semi-definite)
set.seed(11)
Y = matrix(rnorm(20), 5, 4)
(X = t(Y) %*% Y)
```

```
##        [,1]    [,2]    [,3]    [,4]
## [1,]  5.896 -1.4814  1.8063 -1.3254
## [2,] -1.481  4.0254  0.5179  0.9336
## [3,]  1.806  0.5179  4.5617 -0.3928
## [4,] -1.325  0.9336 -0.3928  1.6186
```

```
# Cholesky decomposition
R_chol = chol(X)
R_chol
```

```
##       [,1]    [,2]   [,3]     [,4]
## [1,] 2.428 -0.6101 0.7439 -0.54587
## [2,] 0.000  1.9113 0.5084  0.31423
## [3,] 0.000  0.0000 1.9364 -0.07565
## [4,] 0.000  0.0000 0.0000  1.10281
```

Note that R returns $\mathbf{R}$, that is an *upper triangular matrix*

```
# X = R'R
t(R_chol) %*% R_chol
```

```
##        [,1]    [,2]    [,3]    [,4]
## [1,]  5.896 -1.4814  1.8063 -1.3254
## [2,] -1.481  4.0254  0.5179  0.9336
## [3,]  1.806  0.5179  4.5617 -0.3928
## [4,] -1.325  0.9336 -0.3928  1.6186
```

LU Decomposition

LU Decomposition

LU

An $n \times n$ square matrix $\mathbf{X}$ can be decomposed, with proper row and/or column orderings or permutations, as

$$\mathbf{X} = \mathbf{LU}$$

where

- $\mathbf{L}$ is an $n \times n$ *Lower triangular* matrix
- $\mathbf{U}$ is an $n \times n$ *Upper triangular* matrix

Without a proper ordering or permutations in $\mathbf{X}$*, the factorization may fail to materialize*

LU Decomposition (con't)

LU

$$\mathbf{X} = \mathbf{LU}$$

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix}$$

Eigen-Value Decomposition

Eigen-Value Decomposition

EVD

An $n \times n$ symmetric matrix $\mathbf{X}$ can be decomposed as:

$$\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}'$$

where

- $\mathbf{U}$ is a $n \times p$ column **orthonormal** matrix containing the eigen-vectors of $\mathbf{X}$
- $\mathbf{\Lambda}$ is a $p \times p$ **diagonal** matrix containing the eigen-values of $\mathbf{X}$

Eigen-Value Decomposition

EVD

$$\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}'$$

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} = \begin{bmatrix} u_{11} & \cdots & u_{1p} \\ u_{21} & \cdots & u_{2p} \\ \vdots & \ddots & \vdots \\ u_{n1} & \cdots & u_{np} \end{bmatrix} \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_p \end{bmatrix} \begin{bmatrix} u_{11} & \cdots & u_{n1} \\ u_{12} & \cdots & u_{n2} \\ \vdots & \ddots & \vdots \\ u_{1p} & \cdots & u_{np} \end{bmatrix}$$

Likewise

$$\mathbf{X} = \sum_{k=1}^{p} \lambda_k \mathbf{u_k} \mathbf{u_k'}$$

EVD in R

`eigen()` function

R provides the function `eigen()` to perform a eigen-value decomposition of a given matrix

`eigen()` output

A list with the following components

- `values` a vector containing the eigen-values
- `vectors` a matrix whose columns contain the eigen-vectors

EVD example in R

```
# X matrix
set.seed(22)
Y = matrix(rnorm(20), 5, 4)
X = t(Y) %*% Y

# eigen-value decomposition
EVD = eigen(X)

# elements returned by svd()
names(EVD)


## [1] "values"  "vectors"



# vector of singular values
(lambda = EVD$values)


## [1] 15.615  4.090  2.175  0.187
```

```
# matrix of eigen-vectors
(U = EVD$vectors)


##         [,1]    [,2]     [,3]    [,4]
## [1,]  0.5708  0.7407 -0.33863  0.1043
## [2,] -0.2742  0.5295  0.76797  0.2338
## [3,]  0.2772 -0.3206  0.04462  0.9046
## [4,]  0.7226 -0.2612  0.54181 -0.3408



# U osrthonormal (U'U = I)
t(U) %*% U


##             [,1]       [,2]       [,3]       [,4]
## [1,]  1.000e+00 -3.331e-16 -1.110e-16 -2.776e-17
## [2,] -3.331e-16  1.000e+00  3.608e-16  1.665e-16
## [3,] -1.110e-16  3.608e-16  1.000e+00  5.274e-16
## [4,] -2.776e-17  1.665e-16  5.274e-16  1.000e+00
```

EVD example in R (con't)

```
# X equals U L U'
U %*% diag(lambda) %*% t(U)
```

```
##        [,1]   [,2]   [,3]   [,4]
## [1,]  7.584 -1.401  1.485  5.244
## [2,] -1.401  3.614 -1.767 -2.769
## [3,]  1.485 -1.767  1.778  3.466
## [4,]  5.244 -2.769  3.466  9.092
```

```
# compare to X
X
```

```
##        [,1]   [,2]   [,3]   [,4]
## [1,]  7.584 -1.401  1.485  5.244
## [2,] -1.401  3.614 -1.767 -2.769
## [3,]  1.485 -1.767  1.778  3.466
## [4,]  5.244 -2.769  3.466  9.092
```

QR Decomposition

QR Decomposition

QR

An $n \times p$ matrix $\mathbf{X}$ can be decomposed as

$$\mathbf{X} = \mathbf{QR}$$

where

- $\mathbf{Q}$ is an $n \times p$ *orthonormal* matrix
- $\mathbf{R}$ is an *upper triangular* matrix

QR Decomposition (con't)

QR

$$\mathbf{X} = \mathbf{QR}$$

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1p} \\ q_{21} & q_{22} & \cdots & q_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ q_{n1} & q_{n2} & \cdots & q_{np} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1p} \\ 0 & r_{22} & \cdots & r_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{pp} \end{bmatrix}$$

$\mathbf{Q}$ basis

If $\mathbf{X}$ has k linearly independent columns, then the first k columns of $\mathbf{Q}$ form an orthonormal basis for the column space of $\mathbf{X}$

QR Decomposition in R

QR in R

In R we can use the function `qr()` to compute the QR decomposition of a matrix.

`qr()` output

A list with the following elements:

- `qr` a matrix with the same dimensions as the input `x`
- `qraux` a vector of length `ncol(x)` with info about $\mathbf{Q}$
- `rank` the rank of `x`
- `pivot` information of the pivoting strategy

QR Decomposition in R

```
# matrix
set.seed(33)
(X = matrix(rnorm(15), 5, 3))
```

```
##         [,1]    [,2]     [,3]
## [1,] -0.1359  0.4986  0.16735
## [2,] -0.0408 -0.7552 -0.02928
## [3,]  1.0105  0.7786  1.87585
## [4,] -0.1583  0.7546  0.24463
## [5,] -2.1566 -1.0995  0.70215
```

```
# QR decomposition
QR = qr(X)
QR
```

```
## $qr
##          [,1]    [,2]     [,3]
## [1,]  2.39112  1.2554  0.13427
## [2,]  0.01706  1.2758  0.63494
## [3,] -0.42262 -0.3731 -1.91816
## [4,]  0.06619 -0.6286 -0.01121
## [5,]  0.90194  0.3557  0.42923
##
## $rank
## [1] 3
##
## $qraux
## [1] 1.057 1.582 1.903
##
## $pivot
## [1] 1 2 3
##
## attr(,"class")
## [1] "qr"
```

QR Decomposition in R (con't)

Use `qr.Q()` to extract the $\mathbf{Q}$ matrix of a QR decomposition

```
# Q matrix
qr.Q(QR)


##           [,1]     [,2]     [,3]
## [1,] -0.05685  0.44678  0.05667
## [2,] -0.01706 -0.57518 -0.17632
## [3,]  0.42262  0.19442 -0.88400
## [4,] -0.06619  0.65658  0.08517
## [5,] -0.90194  0.02564 -0.42070
```

Use `qr.R()` to extract the $\mathbf{R}$ matrix of a QR decomposition

```
# R matrix
qr.R(QR)


##       [,1]  [,2]    [,3]
## [1,] 2.391 1.255  0.1343
## [2,] 0.000 1.276  0.6349
## [3,] 0.000 0.000 -1.9182
```

Singular Value Decomposition

Singular Value Decomposition

SVD

An $n \times p$ matrix $\mathbf{X}$ can be decomposed as:

$$\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}'$$

where

- $\mathbf{U}$ is a $n \times p$ column **orthonormal** matrix containing the left singular vectors
- $\mathbf{\Lambda}$ is a $p \times p$ **diagonal** matrix containing the singular values of $\mathbf{X}$
- $\mathbf{V}$ is a $p \times p$ column **orthonormal** matrix containing the right singular vectors

Singular Value Decomposition

SVD

$$\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}'$$

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} = \begin{bmatrix} u_{11} & \cdots & u_{1p} \\ u_{21} & \cdots & u_{2p} \\ \vdots & \ddots & \vdots \\ u_{n1} & \cdots & u_{np} \end{bmatrix} \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_p \end{bmatrix} \begin{bmatrix} v_{11} & \cdots & v_{p1} \\ \vdots & \ddots & \vdots \\ v_{1p} & \cdots & v_{pp} \end{bmatrix}$$

Likewise

$$\mathbf{X} = \sum_{k=1}^{p} \lambda_k \mathbf{u_k} \mathbf{v_k'}$$

svd() in R

svd() function

R provides the function `svd()` to perform a singular value decomposition of a given matrix

svd() output

A list with the following components

- `d` a vector containing the singular values
- `u` a matrix whose columns contain the left singular vectors
- `v` a vector whose columns contain the right singular vectors

SVD example in R

```
# X matrix
set.seed(22)
X = matrix(rnorm(20), 5, 4)

# singular value decomposition
SVD = svd(X)

# elements returned by svd()
names(SVD)
```

```
## [1] "d" "u" "v"
```

```
# vector of singular values
(d = SVD$d)
```

```
## [1] 3.9516 2.0224 1.4748 0.4324
```

```
# matrix of left singular vectors
(U = SVD$u)
```

```
##         [,1]     [,2]    [,3]      [,4]
## [1,] -0.4251 -0.53913 -0.7233  0.009794
## [2,]  0.5269 -0.76863  0.2860  0.056100
## [3,]  0.5753  0.05000 -0.4421  0.131072
## [4,]  0.2215  0.05273 -0.1702 -0.951234
## [5,] -0.4021 -0.33655  0.4131 -0.273371
```

```
# matrix of right singular vectors
(V = SVD$v)
```

```
##         [,1]    [,2]     [,3]    [,4]
## [1,]  0.5708 -0.7407  0.33863  0.1043
## [2,] -0.2742 -0.5295 -0.76797  0.2338
## [3,]  0.2772  0.3206 -0.04462  0.9046
## [4,]  0.7226  0.2612 -0.54181 -0.3408
```

SVD example in R (con't)

```
# U orthonormal  (U'U = I)
t(U) %*% U
```

```
##           [,1]       [,2]       [,3]       [,4]
## [1,] 1.000e+00  1.388e-16  2.776e-17  0.000e+00
## [2,] 1.388e-16  1.000e+00 -2.776e-17 -8.327e-17
## [3,] 2.776e-17 -2.776e-17  1.000e+00  5.551e-17
## [4,] 0.000e+00 -8.327e-17  5.551e-17  1.000e+00
```

```
# V orthonormal  (V'V = I)
t(V) %*% V
```

```
##           [,1]       [,2]       [,3]       [,4]
## [1,]  1.000e+00 -1.110e-16 -5.551e-17  1.110e-16
## [2,] -1.110e-16  1.000e+00  8.327e-17  1.943e-16
## [3,] -5.551e-17  8.327e-17  1.000e+00 -8.327e-17
## [4,]  1.110e-16  1.943e-16 -8.327e-17  1.000e+00
```

```
# X equals U D V'
U %*% diag(d) %*% t(V)
```

```
##         [,1]     [,2]     [,3]    [,4]
## [1,] -0.5121  1.85809 -0.76391 -0.9222
## [2,]  2.4852 -0.06603  0.08196  0.8616
## [3,]  1.0078 -0.16276  0.74303  2.0029
## [4,]  0.2928 -0.19986 -0.08402  0.9366
## [5,] -0.2090  0.30056 -0.79289 -1.6157
```

```
# compare to X
X
```

```
##         [,1]     [,2]     [,3]    [,4]
## [1,] -0.5121  1.85809 -0.76391 -0.9222
## [2,]  2.4852 -0.06603  0.08196  0.8616
## [3,]  1.0078 -0.16276  0.74303  2.0029
## [4,]  0.2928 -0.19986 -0.08402  0.9366
## [5,] -0.2090  0.30056 -0.79289 -1.6157
```

Summary

Matrix Decompositions

Matrix	Decomposition
Symmetric	Cholesky
Symmetric	LU
Symmetric	Spectral (EVD)
Rectangular	QR
Rectangular	Singular Value Decomposition (SVD)