Cholesky Factorization

Kourosh Z.

4/14/2020

Introduction

Cholesky factorization: $A = RR^*$ form.

Assume that A is *a symmetric positive definite matrix (PD)* (this will be discussed in detail later). Recall that in Gaussian eliminations, we apply triangular transformations L_k from the left to introduce zeros in column k:

$$L_k = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -\ell_{k+1,k} & 1 & & \\ & & \vdots & & \ddots & \\ & & -\ell_{m,k} & & & 1 \end{bmatrix},$$

where

$$\ell_{jk} = \frac{x_{jk}}{x_{kk}},$$

for $k < j \leq m$. For symmetric matrices, the process of Gaussian elimination can be modified to reduce the computational cost. The main idea is that the elimination steps can be performed on the upper triangular part only. The lower triangular part of the matrix is set to zero a priori (note that this will eventually happen anyway) and due to symmetry the factor

$$\ell_{jk} = \frac{x_{jk}}{x_{kk}} = \frac{x_{kj}}{x_{kk}}$$

can still be constructed.

Special Case:

Consider a special case of an $m \times m$ Hermitian matrix A with $a_{11} = 1$:

$$A = \begin{bmatrix} 1 & w^* \\ w & K \end{bmatrix},$$

where

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_{m-1} \end{bmatrix}, \quad w^* = \begin{bmatrix} w_1 * & \cdots & w_{m-1}^* \end{bmatrix},$$

is a vector of size $m-1$ and K is a $(m-1)\times(m-1)$ PD matrix. For this particular matrix A, the first transformation L_1 is

$$L_1 = \begin{bmatrix} 1 & & \\ & \ddots & \\ -w & & 1 \end{bmatrix},$$

and

$$L_1 A = \begin{bmatrix} 1 & \\ -w & 1 \end{bmatrix} \begin{bmatrix} 1 & w^* \\ w & K \end{bmatrix} = \begin{bmatrix} 1 & w^* \\ 0 & K - ww^* \end{bmatrix}$$

In particular, since A is symmetric

$$\begin{aligned} L_1 A L_1^* &= \begin{bmatrix} 1 & 0 \\ -w & I \end{bmatrix} \begin{bmatrix} 1 & w^* \\ w & K \end{bmatrix} \begin{bmatrix} 1 & -w^* \\ 0 & I \end{bmatrix} \\ &= \begin{bmatrix} 1 & w^* \\ 0 & K - ww^* \end{bmatrix} \begin{bmatrix} 1 & -w^* \\ 0 & I \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & K - ww^* \end{bmatrix} \end{aligned}$$

Equivalently

$$A = \begin{bmatrix} 1 & 0 \\ w & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & K - ww^* \end{bmatrix} \begin{bmatrix} 1 & w^* \\ 0 & I \end{bmatrix}$$

General Case $a_{11} > 1$

Let $\alpha = \sqrt{a_{11}}$. It's straightforward to see

$$A = \begin{bmatrix} a_{11} & w^* \\ w & K \end{bmatrix} = \begin{bmatrix} \alpha & 0 \\ w/\alpha & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & K - ww^*/a_{11} \end{bmatrix} \begin{bmatrix} \alpha & w^*/\alpha \\ 0 & I \end{bmatrix}.$$

The matrix $K - ww^*/a_{11}$ is a lower principal sub-matrix of A and hence is PD. Therefor, using the same process, we can show that

$$A = \underbrace{R_1^* R_2^* \cdots R_{m-1}^*}_{R^*} \underbrace{R_{m-1} \cdots R_2 R_1}_{R} = R^* R,$$

where R_j's are upper-triangular and $r_{jj} > 0$. This decomposition is referred to as Cholesky factorization:

$$A = R^* R \implies R^{*-1} A = R \implies R_{m-1}^{*-1} \cdots R_2^{*-1} R_1^{*-1} A = R$$

The sequence $R_{m-1}^{*-1} \cdots R_2^{*-1} R_1^{*-1} A = R$ is analogous to $L_{m-1} \cdots L_2 L_1 A = U$ in Gaussian eliminations with some slight modifications. Let's consider the first step $R_1^{*-1} A$. First, note that

$$R_1^* = \begin{bmatrix} \alpha & 0 \\ w/\alpha & I \end{bmatrix} \implies R_1^{*-1} = \begin{bmatrix} 1/\alpha & 0 \\ -w/a_{11} & I \end{bmatrix}$$

Hence

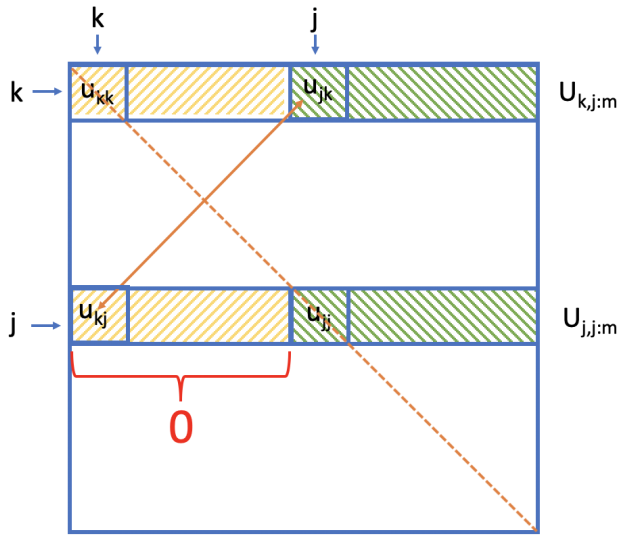
$$R_1^{*-1} A = \begin{bmatrix} 1/\alpha & 0 \\ -w/a_{11} & I \end{bmatrix} \begin{bmatrix} a_{11} & w^* \\ w & K \end{bmatrix} = \begin{bmatrix} a_{11}/\alpha & w^*/\alpha \\ 0 & K - ww^*/a_{11} \end{bmatrix}$$

This operation can be though of a two step process, where in the first step, we perform a Gaussian elimination, followed by multiplying the first row with $1/\alpha$

$$\begin{bmatrix} a_{11} & w^* \\ w & K \end{bmatrix} \overset{\text{GE}}{\rightarrow} \begin{bmatrix} a_{11} & w^* \\ 0 & K - ww^*/a_{11} \end{bmatrix} \overset{\text{Scale}}{\rightarrow} \begin{bmatrix} a_{11}/\alpha & w^*/\alpha \\ 0 & K - ww^*/a_{11} \end{bmatrix}$$

Similar process can be used to describe the application of R_k^{*-1} to the working matrix.

An important distinction between this process and regular Gaussian elimination is that due to the symmetry, the calculation can be performed on the upper triangular part of the matrix only. The following figure illustrates the idea:



- Gaussian Elimination Step:
- $U_{j,k:m} = U_{j,k:m} - (u_{kj}/u_{kk}) * U_{k,k:m}$
- No subtraction is needed in the shaded yellow region as it will eventually be zero. So we fill up the lower triangular portion with zeros a priori.
- The factor u_{kj} is present as u_{jk} due to symmetry.
- Reduced cost elimination is performed on green shaded region only
- $U_{j,j:m} = U_{j,j:m} - (u_{jk}/u_{kk}) * U_{k,j:m}$

Algorithm

Algorithm 23.1. Cholesky Factorization

$R = A$
for $k = 1$ **to** m
    for $j = k + 1$ **to** m
        $R_{j,j:m} = R_{j,j:m} - R_{k,j:m}\overline{R}_{kj}/R_{kk}$
    $R_{k,k:m} = R_{k,k:m}/\sqrt{R_{kk}}$

Remark: Compare this with Gaussian Elimination:

Algorithm 20.1. Gaussian Elimination without Pivoting

$U = A,\ L = I$
for $k = 1$ **to** $m - 1$
    for $j = k + 1$ **to** m
        $\ell_{jk} = u_{jk}/u_{kk}$
        $u_{j,k:m} = u_{j,k:m} - \ell_{jk} u_{k,k:m}$

The essential difference is:

$$\text{Gaussian Eliminations:} \quad U_{j,k:m} \leftarrow U_{j,k:m} - U_{k,k:m} * \frac{u_{jk}}{u_{kk}}$$

$$\text{Cholesky:} \quad R_{j,j:m} \leftarrow R_{j,j:m} - R_{k,j:m} * \frac{\bar{r}_{kj}}{r_{kk}}$$

```
cholesky <- function(A){
  if(!isSymmetric(A)){
    stop('A must be symmetric')
  }
  m <- nrow(A)
  R <- matrix(0, nrow = nrow(A), ncol = ncol(A))
  for(i in 1:m){
    R[i,i:m] <- A[i,i:m]
  }
  for(k in 1:(m-1)){
    for(j in (k+1):m){
      R[j,j:m] <- R[j,j:m] - R[k,j:m] * R[k,j] / R[k,k]
    }
    R[k,k:m] <- R[k,k:m] / sqrt(R[k,k])
  }
  R[m,m] <- R[m,m] / sqrt(R[m,m])

  return(R)
}

A <- matrix(runif(16, 2,5), nrow = 4)
A <- A %*% t(A)

R <- cholesky(A)
A - t(R) %*% R
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0    0    0    0
## [2,]    0    0    0    0
## [3,]    0    0    0    0
## [4,]    0    0    0    0
```

$A = LDL^T$ decomposition

A second (perhaps more common) form of Cholesky factorization is $A = LDL^T$, where L is lower triangular and D is diagonal. This decomposition is equivalent to $A = R^T R$:

$$A = LDL^T = LD^{1/2}D^{1/2}L^T = \underbrace{(LD^{1/2})}_{R^T}\underbrace{(LD^{1/2})^T}_{R} = R^T R$$

where $R = (LD^{1/2})^T = D^{1/2}L^T$. The derivation of this version can be obtained by equating both sides.

$$\begin{aligned}
\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{21} & a_{22} & a_{32} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ \ell_{21} & 1 & 0 \\ \ell_{31} & \ell_{32} & 1 \end{bmatrix} \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix} \begin{bmatrix} 1 & \ell_{21} & \ell_{31} \\ 0 & 1 & \ell_{32} \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} d_1 & 0 & 0 \\ d_1\ell_{21} & d_2 & 0 \\ d_1\ell_{31} & d_2\ell_{32} & d_3 \end{bmatrix} \begin{bmatrix} 1 & \ell_{21} & \ell_{31} \\ 0 & 1 & \ell_{32} \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} d_1 & d_1\ell_{21} & d_1\ell_{31} \\ d_1\ell_{21} & d_2 + d_1\ell_{21}^2 & d_1\ell_{21}\ell_{31} + d_2\ell_{32} \\ d_1\ell_{31} & d_1\ell_{21}\ell_{31} + d_2\ell_{32} & d_1\ell_{31}^2 + d_2\ell_{32}^2 + d_3 \end{bmatrix}
\end{aligned}$$

Equating both sides, we obtain:

$$\begin{aligned}
a_{11} &= d_1 && \implies d_1 = a_{11} \\
a_{21} &= d_1\ell_{21} && \implies \ell_{21} = a_{21}/d_1 \\
a_{31} &= d_1\ell_{31} && \implies \ell_{31} = a_{31}/d_1 \\
a_{22} &= d_1\ell_{21}^2 + d_2 && \implies d_2 = a_{22} - d_1\ell_{21}^2 \\
a_{32} &= d_1\ell_{21}\ell_{31} + d_2\ell_{32} && \implies \ell_{32} = (a_{32} - d_1\ell_{21}\ell_{31})/d_2 \\
a_{33} &= d_1\ell_{31}^2 + d_2\ell_{32}^2 + d_3 && \implies d_3 = a_{33} - d_1\ell_{31}^2 - d_2\ell_{32}^2
\end{aligned}$$

From these equations, it is easy to see that the general case is as follows:

$$\begin{aligned}
d_j &= a_{jj} - \sum_{k=1}^{j-1} d_k\ell_{jk}^2 \\
\ell_{ij} &= \left[a_{ij} - \sum_{k=1}^{j-1} d_k\ell_{ik}\ell_{jk}\right]/d_j
\end{aligned}$$

for $j = 1, \cdots, n$ **do**
  $c_{jj} = a_{jj} - \sum_{k=1}^{j-1} d_k\ell_{jk}^2$;
  $d_j = c_{jj}$;
  for $i = j+1, \cdots, n$ **do**
    $c_{ij} = a_{ij} - \sum_{k=1}^{j-1} d_k\ell_{ik}\ell_{jk}$;
    $\ell_{ij} = c_{ij}/d_j$;
  end
end

Algorithm 1: LDL^T

```
LDLt <- function(A){
  n <- nrow(A)
  L <- matrix(0, nrow = nrow(A), ncol = ncol(A))
  diag(L) <- 1
  d <- matrix(0, nrow = 1, ncol = ncol(A))
  for(j in 1:n){
    if(j == 1){
      cjj <- A[j,j]
    }else{
      cjj <- A[j,j] - sum(d[1:(j-1)] * L[j, 1:(j-1)] ^ 2)
    }
    d[j] <- cjj
    if(j == n){
      next
    }
    for(i in (j+1):n){
      if(j == 1){
        cij <- A[i,j]
      }else{
        cij <- A[i,j] - sum(d[1:(j-1)] * L[i, 1:(j-1)] * L[j,1:(j-1)])
      }
      L[i,j] <- cij / d[j]
    }
  }

  return(list(L = L, d = d))
}

A <- matrix(runif(16, 2,5), nrow = 4)
A <- A %*% t(A)

L <- LDLt(A)
D <- matrix(0, nrow = nrow(A), ncol = nrow(A))
diag(D) <- L$d
A - L$L %*% D %*% t(L$L)
```

```
##      [,1]         [,2]         [,3] [,4]
## [1,]    0 0.000000e+00 0.000000e+00    0
## [2,]    0 7.105427e-15 7.105427e-15    0
## [3,]    0 0.000000e+00 0.000000e+00    0
## [4,]    0 0.000000e+00 0.000000e+00    0
```

Modified Cholesky

If the matrix A is not PD, then the Cholesky decomposition does not exist. However, the Cholesky decomposition algorithm can be modified to approximate A with a "close enough" PD matrix. The idea is as follows:

We would like the elements of the diagonal matrix D to be sufficiently positive, while controlling the entries in L from getting too large.

Pick numbers δ and β such that

1. $d_j \geq \delta$; i.e., make d_j sufficiently large.

2. $\ell_{ij}\sqrt{d_j} \leq \beta$; i.e., prevent ℓ_{ij} from getting too large.

To acheive the above two condition we impose the following modification:

$$d_j = \max\left\{|c_{jj}|, \left(\frac{\Theta_j}{\beta}\right)^2, \delta\right\},$$

where $\Theta_j = \max_{j<i\leq m} |c_{ij}|$.

Note:

$$|m_{ij}| = |\ell_{ij}\sqrt{d_j}| = \frac{|c_{ij}|}{\sqrt{d_j}} \leq \frac{|c_{ij}|}{\sqrt{\left(\frac{\Theta_j}{\beta}\right)^2}} = \left(\frac{|c_{ij}|}{\Theta_j}\right)\beta \leq \beta$$

```
mod.Cholesky <- function(A, delta = 1e-8, beta = 100){
  n <- nrow(A)
  L <- matrix(0, nrow = nrow(A), ncol = ncol(A))
  C <- matrix(0, nrow = nrow(A), ncol = ncol(A))
  diag(L) <- 1
  d <- matrix(0, nrow = 1, ncol = ncol(A))
  for(j in 1:n){
    if(j == 1){
      C[j,j] <- A[j,j]
    }else{
      C[j,j] <- A[j,j] - sum(d[1:(j-1)] * L[j, 1:(j-1)] ^ 2)
    }
    if(j == n){
      thetaj <- 0
      d[j] <- max(abs(C[j,j]), (thetaj/beta)^2, delta)
      next
    }else{
      thetaj <- max(abs(C[(j+1):n,j]))
      d[j] <- max(abs(C[j,j]), (thetaj/beta)^2, delta)
    }
    for(i in (j+1):n){
      if(j == 1){
        C[i,j] <- A[i,j]
      }else{
        C[i,j] <- A[i,j] - sum(d[1:(j-1)] * L[i, 1:(j-1)] * L[j,1:(j-1)])
      }
      L[i,j] <- C[i,j] / d[j]
    }
  }

  D <- matrix(0, nrow = n, ncol = n)
  diag(D) <- d
  A.mod <- L %*% D %*% t(L)

  return(list(A.mod = A.mod, L = L, d = d))
}

A <- matrix(runif(16, 2,5), nrow = 4)
```

```r
A <- A %*% t(A)

L <- mod.Cholesky(A)

n <- nrow(A)
D <- matrix(0, nrow = n, ncol = n)
diag(D) <- L$d
A - L$L %*% D %*% t(L$L)
```

```
##              [,1] [,2] [,3]          [,4]
## [1,] 0.000000e+00    0    0  7.105427e-15
## [2,] 0.000000e+00    0    0  0.000000e+00
## [3,] 0.000000e+00    0    0 -7.105427e-15
## [4,] 7.105427e-15    0    0  0.000000e+00
```

```r
## Testing PSD
A <- matrix(runif(16, 2,5), nrow = 4)
A <- A %*% t(A)
EE <- eigen(A, symmetric = T)
A <- t(EE$vectors) %*% diag((EE$values - min(EE$values) - 0.03)) %*% EE$vectors

L <- mod.Cholesky(A)
A - L$A.mod
```

```
##               [,1]          [,2]          [,3]          [,4]
## [1,]  0.000000e+00 -1.776357e-15 -3.552714e-15  7.105427e-15
## [2,] -1.776357e-15  0.000000e+00  0.000000e+00  0.000000e+00
## [3,]  0.000000e+00  0.000000e+00  0.000000e+00 -7.105427e-15
## [4,]  0.000000e+00  0.000000e+00  0.000000e+00 -4.206952e+00
```

```r
eigen(L$A.mod, symmetric = T)
```

```
## eigen() decomposition
## $values
## [1] 175.29414157   3.59231674   1.21715023   0.01477309
##
## $vectors
##            [,1]       [,2]       [,3]        [,4]
## [1,]  0.3799623 -0.6233460 -0.2313348  0.64308060
## [2,] -0.2132281  0.6002037 -0.5907756  0.49525093
## [3,]  0.3429057 -0.1181540 -0.7297136 -0.57963215
## [4,] -0.8322128 -0.4870676 -0.2549246 -0.07211404
```