Matrix Inversion Using Cholesky Decomposition

Marian Petruk, Luka Triska
Ukrainian Catholic University, Lviv
petruk@ucu.edu.ua, triska@ucu.edu.ua

December 15, 2017

Contents

1 Introduction

Matrix inversion using Cholesky decomposition technique is used for inversion of the positive-definite and symmetric matrices. Once one has such type of a matrix, the numerical approach based on Cholesky technique is more efficient than e.g. more general Gauss-Jordan elimination.

Classical matrix inversion algorithms that are based on Cholesky decomposition require $\mathcal{O}(n^3)$ operations. Krishnamoorthy & Menon (2011) suggest a modified method of matrix inversion based on the Cholesky decomposition. It has fewer operations comparing to the classical method. We will describe these methods below and implement in the code the idea of Krishnamoorthy & Menon (2011).

Cholesky decomposition technique may be applied to Hermitian matrices (i.e. complex square matrices that is equal to its own conjugate transpose). We will apply the method to matrices with real elements.

2 Algorithm description

2.1 Classical method with Cholesky decomposition

If $A \in \mathcal{R}^{N \times N}$ is a square symmetric positive-definite matrix then the Cholesky decomposition may transform it to

$$A = R^{\mathrm{T}} R \tag{1}$$

where R is an upper triangular matrix and R^{T} its transpose; $R^{\mathrm{T}} = L$ where L is a lower triangular matrix. The elements of $L = l_{ij}$, $0 \leq j \leq i \leq N-1$ are given by the expressions

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=0}^{i-1} l_{ik}^2} \tag{2}$$

$$l_{ij} = \frac{1}{r_{jj}} \left(a_{ij} - \sum_{k=0}^{i-1} l_{ik} l_{jk} \right) \tag{3}$$

It is important to follow a certain order when calculating the elements l_{ij} because each next element uses values of the previous elements. The order is the following: from the first element on the left to the right and then down to the next row starting again from the leftmost element.

The next step after the decomposition is to find the inverse matrix $X = A^{-1}$. The classical version of the second step is as follows.

If $X \in \mathcal{R}^{N \times N}$ and $X = A^{-1}$ then

$$AX = I \tag{4}$$

where I is the identity matrix of order N. Then we can rewrite this as

$$R^{\mathrm{T}} R X = I. \tag{5}$$

By letting $B = RX$, we derive

$$R^{\mathrm{T}} B = I \tag{6}$$

and

$$RX = B. \tag{7}$$

By solving equations (6) and (7) we may derive the inverse matrix A^{-1}.

2.2 Modified method with Cholesky decomposition

The modification proposed by Krishnamoorthy & Menon (2011) allows us to skip the process of solving the equation (6). So, the second step in finding the inverse matrix differs from described above and is simpler.

Let $B = \left(R^{\mathrm{T}}\right)^{-1}$. The diagonal elements of R^{T} and R are the same. The diagonal elements of R^{-1} are inversions of the diagonal elements of R. Therefore, we construct the matrix S with elements

$$s_{ij} = \begin{cases} 1/r_{ii} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Table 1: Number of operations for the matrix inverse

Method	Number of operations
Classical	$\frac{5}{6}n^3$
Modified	$\frac{1}{2}n^3$

Krishnamoorthy & Menon (2011) state that matrix S is the correct solution to the diagonal elements of matrix B; in other words, matrices B and S have mutual elements along the diagonal. Therefore, there is no need to compute B but just enough to use backward substitution in order to solve for x_{ij} from the equation

$$Rx_i = s_i \tag{9}$$

where x_i and s_i are the i-th columns in the matrices X and S respectively.

The formulae for the backward substitution method are

$$x_{ji} = \frac{s_{ji} - \sum\limits_{k=j+1}^{N-1} r_{jk}x_{ki}}{r_{jj}} \tag{10}$$

where $N-1 \geq i \geq j \geq 0$. In the numerical realization of the method, it is important to proceed in a certain order, because calculation of an element requires knowledge of the previous elements. Namely, we fix the ith columns in X and S starting from the rightmost column: the column index i varies from $N-1$ to 0. Then, solving the system (9), we start from the element of x_i with the highest index j (which is equal to i, because the first element of s_i which is not zero is s_{ii}) and solve for the elements x_{ji} of the vector x_i toward the top, i.e. the index j changes from from i to 0. In this way we calculate for the upper triangular elements of the matrix X. The lower triangular elements are just

$$x_{ij} = x_{ji}. \tag{11}$$

3 Time complexity

Classical Cholesky decomposition requires $\frac{1}{6}n^3$ operations. Then there are two equations to be solved using backward substitution. They require $\frac{1}{3}n^2$ operations each. So, the total number of operations with the classical method is $\frac{5}{6}n^3$ (Table 1).

The modified Krishnamoorthy & Menon method requires solution of one equation only and thus $\frac{1}{3}n^3$ operations. That, combined with Cholesky decomposition ($\frac{1}{6}n^3$ operations) gives the total number of operations $\frac{1}{2}n^3$ (Table 1).

We have also directly measured the times of the execution. The description of our experiment and results are in the next section.

4 Experiment description

The data we used to test our program were generated with the matrices_generator module that we created. It generates a positive-definite matrix (i.e. matrix which

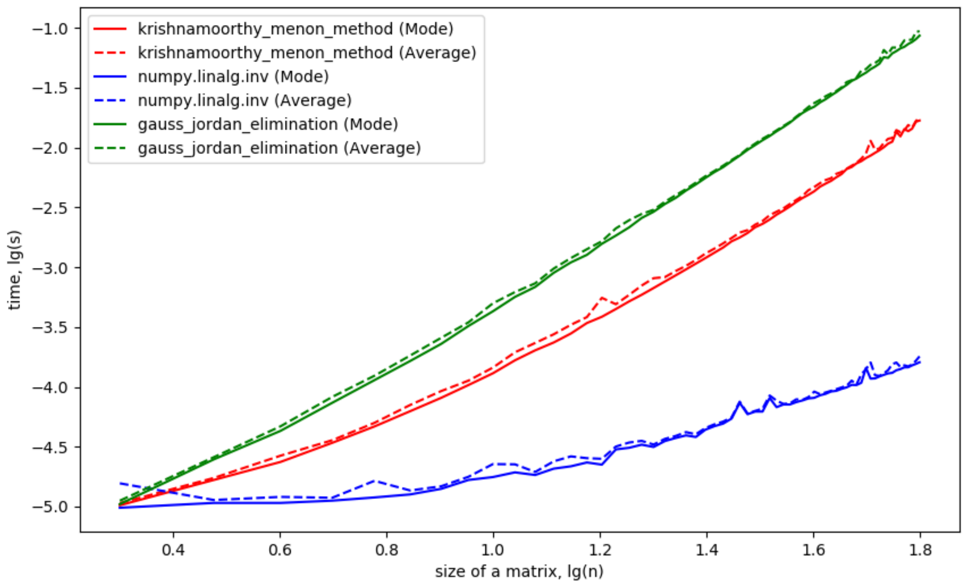

Figure 1: Times the algorithms spend to invert the matrix of the dimension $n \times n$. The mode (solid lines) and the average (dashed lines) times are shown. The green lines are for the Gauss-Jordan elimination, the red lines are for the modified Cholesky decomposition, the blue lines are for the method from the NumPy library.

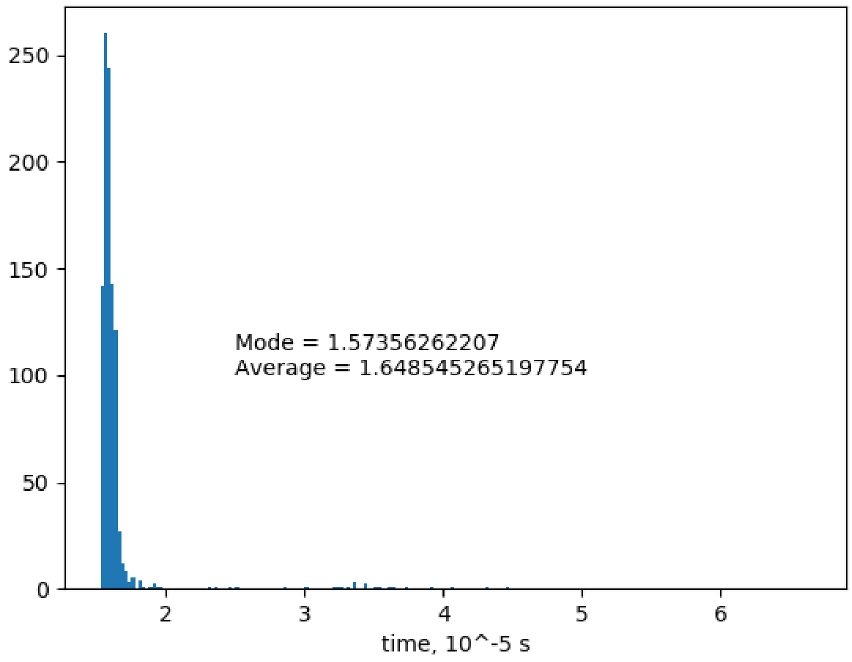

Figure 2: An example of the histogram for the times the code spent inverting the 3×3 matrix by the Krishnamoorthy & Menon method with Cholesky decomposition. The number of trials is 1000.

may be decomposed to the lower and upper matrices) in three steps: i) generate a real matrix Y, ii) calculate Y^{T}, iii) produce the positive-definite matrix by $A = Y^{\mathrm{T}}Y$.

With this step, we ended up with 499 matrices, with n ranging from 2 to 500 (the json file containing that data took up a staggering 810 megabytes (!). However, we have used matrices up to $n = 63$ due to the approaching deadline and lack of computing power).

We tested Cholesky decomposition part of our code on an example of matrix from Wikipedia (https://en.wikipedia.org/wiki/Cholesky_decomposition)

$$A = LL^T$$
$$A = \begin{bmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 6 & 1 & 0 \\ -8 & 5 & 3 \end{bmatrix} \times \begin{bmatrix} 2 & 6 & -8 \\ 0 & 1 & 5 \\ 0 & 0 & 3 \end{bmatrix} \tag{12}$$

We tested the part for the matrix inverse by comparing our results with the results from verified tools (i.e. Wolfram Alpha, NumPy library).

The inverse matrix A^{-1} is:

$$A^{-1} = \begin{bmatrix} 49.3611 & -13.5555 & 2.1111 \\ -13.5555 & 3.7777 & -0.5555 \\ 2.1111 & -0.5555 & 0.1111 \end{bmatrix} \tag{13}$$

The modified Cholesky method for the matrix inverse is the main algorithm of our project. In order to measure its time complexity and compare it to the time complexity of the other two methods, we inverse each matrix 1000 times, for each matrix dimension and for each method. Then we calculated the average time and the mode (the most frequent time) for these trials of the matrix inverse.

The typical time, that three methods spent to calculate matrix inverse of an $n \times n$ dimension , are shown on Fig. 1.

First, we see from this plot that the average is typically larger than the mode and also more fluctuating. The reason of such behavior is visible on the Fig. 2 where the time for each of 1000 runs of the same code for the same matrix are shown. We realized that there are times considerably larger than the average and the mode. This is the effect of some background processes which slow down executions of our code. The long duration runs could be quite big, even till 60 units, making the average value more fluctuating than the mode.

Secondly, the slope of the red and green lines on the log-log plot (we measured it by hand from Fig. 1) demonstrates that the time complexity of the 'modified' Cholesky method is proportional to n^3. The Gauss-Jordan elimination method for the matrix inverse (the 'classical' one) is also proportional to n^3 (the same slope in the log-log plot) but with the larger coefficient (the green line is above the red line). This is in agreement with the Table 1.

Thirdly, we obtained that the power w in the dependence of t on n^w is smaller than 3 for the matrices of the small dimensions ($n < 5$), for the classical and modified methods.

5 Conclusions

Performing this project, we have learned how to generate a positive-definite matrix, how to solve systems of linear equations using backward substitution, how to split a matrix into lower and upper triangular matrices.

We improved our knowledge of matrices, matrix inversion and transposition. We understand what a numerical experiment is, saw what it is like to work with random data (running our code, each time we derived different Fig. 2), understand what histogram is and touch the statistical data analysis.

Working on our algorithm, we learned how to work with NumPy, SciPy, and matplotlib.

The most difficult part of the research was to grasp the main concept, idea from a research paper. It was hard because we have no previous experience in reading and understanding such papers.

We have also stumbled upon the problem of lacking the data for testing, so we had to learn how to generate all the data and then to work with them.

It is remained unclear to us why the time complexity for matrices with dimensions $n < 5$ is not $\mathcal{O}(n^3)$ but $\mathcal{O}(n^w)$ with $w < 3$.

6 GitHub repository

https://github.com/marianpetruk/ad_fontes_2017

Bibliography

Krishnamoorthy & Menon (2011) Matrix Inversion Using Cholesky Decomposition
https://arxiv.org/abs/1111.4144