

Cholesky分解的单Pass随机算法

刘雪翠

上海理工大学理学院, 上海

收稿日期: 2023年12月7日; 录用日期: 2024年1月10日; 发布日期: 2024年1月18日

摘要

对于大规模数据矩阵, 数据的读取成本远高于算法本身的成本; 对存储在磁盘外部的流型数据, 往往只有一次读取数据的机会。而以往的Cholesky分解的随机算法都至少需要读取输入数据两次, 难以满足实际应用中的低成本需求。本文基于矩阵分解的单pass随机算法研究, 提出了Cholesky分解的单pass随机算法, 并给出了该算法的误差上界, 最后通过数值实验验证了该算法的可行性以及有效性。

关键词

Cholesky分解, 矩阵分解, 单Pass随机算法

Single-Pass Randomized Algorithm for Cholesky Decomposition

Xuecui Liu

College of Science, University of Shanghai for Science and Technology, Shanghai

Received: Dec. 7th, 2023; accepted: Jan. 10th, 2024; published: Jan. 18th, 2024

Abstract

For large-scale data matrices, the cost of data reading is much higher than the cost of algorithms. For streaming data stored outside the disk, there is only one chance to read the data. In the past, the randomized algorithm of Cholesky decomposition needed to read the original data at least twice, which was difficult to meet the low-cost requirements in practical applications. Based on the research of single-pass randomized algorithm with matrix decomposition, a single-pass randomized algorithm based on Cholesky decomposition is proposed, and the error upper bound of the algorithm is given, and finally the feasibility and effectiveness of the algorithm are verified by numerical experiments.

Keywords

Cholesky Decomposition, Matrix Factorization, Single-Pass Randomized Algorithm

Copyright © 2024 by author(s) and Hans Publishers Inc.

This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

<http://creativecommons.org/licenses/by/4.0/>



Open Access

1. 问题的提出

矩阵 *Cholesky* 分解的算法是数值线性代数的一个重要内容，其在最小二乘、线性回归、非线性优化等科学计算问题中具有广泛的应用。*Cholesky* 分解[1]源于矩阵的 *LU* 分解，即对对称正定矩阵 $A \in R^{m \times m}$ ，存在唯一的对角元素皆为正的下三角矩阵 $L \in R^{m \times m}$ ，使得 A 可以分解为 $A = LL^T$ 。为了增强数值稳定性，2002 年，Hammarling [2]提出了全主元的 *Cholesky* 分解。2004 年，Gu 和 Miranian [3]又提出了 *Cholesky* 分解的秩显算法。2016 年，Xiao 和 Gu [4]提出了 *Cholesky* 分解在分块形式下的低秩逼近算法，进一步给出了谱显分解算法，从而使得算法变得高效稳定。然而，这些确定性算法在矩阵的低秩逼近分解中时间成本过高，导致算法的效率较低。为了改进这些不足，文献[5]提出的主元随机 *Cholesky* 分解算法不仅有很好的低秩逼近效果，而且节省了大量时间成本。

但是这类随机算法的通病都是至少需要读取输入矩阵两次。而面对存储在内存外部的大规模数据和流型数据时，往往只有一次读取数据的机会。于是就有学者提出将单 *pass* 随机算法应用到矩阵分解的低秩逼近随机算法中。2011 年，Halko 和 Martinsson 等[6]首次提出了将单 *pass* 随机算法应用到矩阵的特征值分解，而后 Tropp 和 Yurtsever 等[7] [8]提出了一种新的 *SVD* 分解的单 *pass* 随机算法。接着，在 2020 年 Li 和 Yin [5]提出了 *LU* 分解的单 *pass* 随机算法。在此基础上，本文从随机矩阵投影的角度构造了一种新的随机算法——*Cholesky* 分解的单 *pass* 随机算法，并给出了相应的误差上界。同时通过数值实验对 *Cholesky* 分解的几种算法进行了比较。

2. 预备知识

定义 1 [9] 设 $\mu \geq 1$ ， $a_1 > 0$ ， $a_2 > 0$ ，亚高斯随机矩阵 $\mathcal{A}(\mu, a_1, a_2, m, n)$ 是一个由所有 $m \times n$ 阶随机矩阵 $A = (\xi_{ij})$ 所构成的矩阵集， A 中的元素皆为服从 *i.i.d.* 的随机变量，且满足下述三个条件：

- 1) $E\left(|\xi_{ij}|^3\right) \leq \mu^3$;
- 2) $E\left(\xi_{ij}^2\right) \geq 1$;
- 3) $P\left(\|A\|_2 > a_1\sqrt{m}\right) \leq e^{-a_2m}$ 。

注 1 从文献[9]可知，如果 A 是 $m \times n$ ($m > n$) 阶亚高斯随机矩阵，则有 $A \in \mathcal{A}(\mu, a_1, a_2, m, n)$ ，且 $a_1 = 6\mu\sqrt{a_2 + 4}$ ，若矩阵 $A_{m \times n} \in \mathcal{N}(0, 1)$ 其中 $m > n$ ，有 $A \in \mathcal{A}(\mu, a_1, a_2, m, n)$ ，且 $\mu = \sqrt[3]{\frac{4}{2\pi}}$ 。

引理 1 [9] 若矩阵 $A \in \mathcal{A}(\mu, a_1, a_2, m, n)$ ，参数 m, n 满足 $m > \left(1 + \frac{1}{\ln n}\right)n$ ，则存在正常数 c_1, c_2 使得 $P\left(\sigma_n < c_1\sqrt{m}\right) \leq e^{-c_2m}$ 成立。

注 2 由文献[10]，可知引理 1 中参数 c_1, c_2 的确切值：

$$c_1 = \frac{b}{c^2 c_3} \left(\frac{b}{3e^2 c_3 a_1} \right)^{\frac{1}{\delta}}, \quad c_2 = \min \left(1, \frac{c''}{2\mu^6}, a^2 \right) - \frac{\ln 3}{m},$$

$$\text{其中, } c_3 = 4\sqrt{\frac{2}{\pi}} \left(\frac{2\mu^9}{a_1^3} + \sqrt{\pi} \right), \quad b = \min \left(\frac{1}{4}, \frac{c'}{5a_1\mu^3} \right), \quad c' = \left(\frac{27}{2^{13}} \right)^{\frac{1}{2}}, \quad c'' = \frac{27}{2^{11}}.$$

引理 2 [11] 设矩阵 $A \in \mathbb{R}^{m \times n}$ 的 SVD 分解为 $A = U\Sigma V^T$, 随机矩阵 $\Omega \in \mathbb{R}^{n \times k}$ 列满秩, $A\Omega$ 有 *thin-QR* 分解 $A\Omega = QR$, $V^T\Omega$ 的分块形式为 $V^T\Omega = (\hat{\Omega}_1^T, \hat{\Omega}_2^T)^T$, 其中 $\hat{\Omega}_1 \in \mathbb{R}^{(k-p) \times k}$, $\hat{\Omega}_2 \in \mathbb{R}^{(n-k+p) \times k}$. 设 $\hat{\Omega}_1$ 行满秩, r 是目标秩且满足 $r \leq k$, $0 \leq p \leq k-r$, 则有

$$\|(I - QQ^T)A\|_F \leq \sqrt{\frac{r\sigma_1^2 \sigma_{k-p+1}^2 \|\hat{\Omega}_2\|_2^2 \|\hat{\Omega}_1^?\|_2^2}{\sigma_{k-p+1}^2 \|\hat{\Omega}_2\|_2^2 \|\hat{\Omega}_1^?\|_2^2 + \sigma_1^2} + \sum_{i=r+1}^n \sigma_i^2}$$

算法 1 主元随机 Cholesky 分解算法(RCH) [5]

输入: 半正定矩阵 $A_{n \times n}$, 目标值 r , 过采样参数 $p \geq 2$;

输出: 置换矩阵 $P_{n \times n}$, 下三角矩阵 $L_{n \times (k+p)}$, 使得 $\|PAP^T - LL^T\| \leq \varepsilon$, ε 是误差界。

- 1) 生成一个 $n \times (k+p)$ 阶的高斯采样矩阵 Ω ;
- 2) 计算 $Y = A\Omega$;
- 3) 对 Y 进行列主元 LU 分解, 得到 $PY = L_y U_y$;
- 4) 令 $B = L_y^T PAP^T (L_y^T)^T$, 其中 $L_y^T = (L_y^T L_y)^{-1} L_y^T$;
- 5) 对 B 作 Cholesky 分解, 得 $B = L_b L_b^T$;
- 6) 令 $L = L_y L_b$;
- 7) 返回 L, P .

3. Cholesky 分解的单 Pass 随机算法(SPRCH)和误差界

3.1. Cholesky 分解的单 Pass 随机算法

本节基于 LU 分解的单 pass 随机算法, 介绍了一种新的 Cholesky 分解的单 pass 随机算法, 与算法 1 相比, 它即保留了在低秩逼近算法中较好的效果, 又只需对 A 访问一次, 大大的节约了计算成本。类似于算法 1 的构造, 我们可以给出算法 2 的具体思想如下:

我们可以根据 $PAP^T \approx L_y L_y^T PAP^T (L_y^T)^T (L_y^T)^T$, 然后考虑将算法 1 中的 $B = L_y^T PAP^T (L_y^T)^T$ 的 A 替换掉, 我们将等式两边同时左乘 L_y , 右乘 L_y^T 得到:

$$L_y B L_y^T \approx L_y L_y^T PAP^T (L_y^T)^T (L_y^T)^T \approx PAP^T$$

因此我们得到

$$L_y B L_y^T \approx PAP^T$$

又因为 $Y_2 = \Omega_1^T Y_1$ 将等式两边同时左乘 $\Omega_1^T P^T$, 右乘 $P\Omega_1$, 我们得到:

$$\Omega_1^T P^T L_y B L_y^T P \Omega_1 \approx \Omega_1^T Y_1 = Y_2$$

然后我们得到 B 的一个近似值:

$$B \approx (\Omega_1^T P^T L_y)^\dagger Y_2 (L_y^T P \Omega_1)^\dagger$$

因此我们就得到了一个新的算法 *Cholesky* 分解的单 *pass* 随机算法(*SPRCH*)

取流型数据 $Y_1 \leftarrow A \Omega_1$, $Y_2 \leftarrow \Omega_1^T Y_1$, 则可通过以下更新公式计算:

$$Y_1(i,:) \leftarrow A(i,:) \Omega_1, Y_{2,i+1} \leftarrow Y_{2,i} + \Omega_1^T Y_1(i,:), i = 1, 2, 3, \dots, n.$$

算法 2 *Cholesky* 分解的单 *pass* 随机算法(*SPRCH*)

输入: 半正定矩阵 $A_{n \times n}$, 目标值 r , 过采样参数 $p \geq 2$;

输出: 置换矩阵 $P_{n \times n}$, 下三角矩阵 $L_{n \times (k+p)}$, 使得 $\|PAP^T - LL^T\| \leq \varepsilon$, ε 是误差界。

- 1) 生成一个 $n \times (k+p)$ 阶的高斯采样矩阵 Ω_1 ;
 - 2) 计算 $Y_1 = A \Omega_1$, $Y_2 = \Omega_1^T Y_1$;
 - 3) 对 Y_1 进行列主元 *LU* 分解, 得到 $PY_1 = L_y U_y$;
 - 4) 令 $B = (\Omega_1^T P^T L_y)^\dagger Y_2 (L_y^T P \Omega_1)^\dagger$;
 - 5) 对 B 作 *Cholesky* 分解, 得 $B = L_b L_b^T$;
 - 6) 令 $L = L_y L_b$;
 - 7) 返回 L, P 。
-

3.2. 误差分析

$$\begin{aligned} \|PAP^T - LL^T\|_F &= \|PAP^T - L_y L_b L_b^T L_y^T\|_F = \|PAP^T - L_y B L_y^T\|_F \\ &= \|PAP^T - L_y (\Omega_1^T P^T L_y)^\dagger Y_2 (L_y^T P \Omega_1)^\dagger L_y^T\|_F \\ &\leq \|PAP^T - L_y L_y^\dagger P A P^T\|_F + \|L_y L_y^\dagger P A P^T - L_y (\Omega_1^T P^T L_y)^\dagger Y_2 (L_y^T P \Omega_1)^\dagger L_y^T\|_F \\ &\leq \left(1 + \|L_y (\Omega_1^T P^T L_y)^\dagger \Omega_1^T\|_2\right) \cdot \|(I - L_y L_y^\dagger) P A\|_F \end{aligned}$$

接下来估计 $\|L_y (\Omega_1^T P^T L_y)^\dagger \Omega_1^T\|_2$ 的上界, 取特殊的 p 值 0, 首先对 L_y 做 *thin-svd* 分解 $L_y = \hat{U} \hat{\Sigma} \hat{V}^T$, 其中 $\hat{U} \in R^{n \times r}$, $\hat{\Sigma} \in R^{r \times r}$, $\hat{V} \in R^{r \times r}$, 即可得到

$$\begin{aligned} \|L_y (\Omega_1^T P^T L_y)^\dagger \Omega_1^T\|_2 &= \|\hat{U} \hat{\Sigma} \hat{V}^T (\Omega_1^T P^T \hat{U} \hat{\Sigma} \hat{V}^T)^\dagger \Omega_1^T\|_2 \\ &= \|\hat{U} \hat{\Sigma} \hat{V}^T (\hat{\Sigma} \hat{V}^T)^{-1} (\Omega_1^T P^T \hat{U})^\dagger \Omega_1^T\|_2 \end{aligned}$$

由于 P 是置换矩阵, 且 \hat{U} 为列正交矩阵, 因此, $\Omega_1^T P^T \hat{U}$ 是一个 *Gaussian* 随机矩阵, 由引理 1, 有

$$\left\| \left(\Omega_1^T P^T \hat{U} \right)^\dagger \right\|_2 = \frac{1}{\sigma_r \left(\Omega_1^T P^T \hat{U} \right)} \leq \frac{1}{c_1 \sqrt{l}}$$

成立的概率不低于 $1 - e^{-c_2 l}$ 。

类似地，有 $\left\| \Omega_1^T \right\|_2 \leq a_1 \sqrt{n}$ 成立的概率不低于 $1 - e^{-a_2 n}$ ，则 $\left\| L_y \left(\Omega_1^T P^T L_y \right)^\dagger \Omega_1^T \right\|_2 \leq \frac{a_1 \sqrt{n}}{c_1 \sqrt{l}}$ 成立的概率不低于 $1 - e^{-a_2 n} - e^{-c_2 l}$ 。接下来估计 $\left\| \left(I - L_y L_y^\dagger \right) P A \right\|_F$ 的上界。

对算法 2，考虑对 PY 进行 QR 分解，即 $PY = QR$ ，则相应的可以得到 $L_y L_y^\dagger = QQ^T$ ，此外将引理 2 中的 Ω 用算法 2 中的 Ω_1 替换，同时取特殊的 p 值 0，则有以下结论成立：

$$\left\| \left(I - L_y L_y^\dagger \right) P A \right\|_F \leq \sqrt{\frac{r \sigma_1^2 \sigma_{r+1}^2 \left\| \tilde{\Omega}_2 \right\|_2^2 \left\| \tilde{\Omega}_1 \right\|_2^2}{\sigma_{r+1}^2 \left\| \tilde{\Omega}_2 \right\|_2^2 \left\| \tilde{\Omega}_1 \right\|_2^2 + \sigma_1^2} + \sum_{i=r+1}^n \sigma_i^2}$$

综上所述，可以得到算法 2 的误差界有如下结论。

定理 1 在算法 2 的假设条件下，Cholesky 分解的随机算法误差界满足：

$$\left\| P A P^T - L L^T \right\|_F \leq \left(1 + \frac{a_1 \sqrt{n}}{c_1 \sqrt{l}} \right) \sqrt{\frac{r \sigma_1^2 \sigma_{r+1}^2 \left\| \hat{\Omega}_2 \right\|_2^2 \left\| \hat{\Omega}_1 \right\|_2^2}{\sigma_{r+1}^2 \left\| \hat{\Omega}_2 \right\|_2^2 \left\| \hat{\Omega}_1 \right\|_2^2 + \sigma_1^2} + \sum_{i=r+1}^n \sigma_i^2}$$

成立的概率不低于 $1 - e^{-a_2 n} - e^{-c_2 l}$ 。

4. 数值结果

4.1. 不同过采样参数选取对于算法的精度及运行时间的影响

取固定阶数为 1000×1000 的奇异值快速下降的数据矩阵 A ，对于不同的参数用两种随机算法做十次独立的实验，并记录误差和运行时间分别取平均值为 e_{RCH} 和 e_{SPRCH} ， t_{RCH} 和 t_{SPRCH} 。

从图 1 中可以看出两种算法的误差精度都很高。但随着 p 值增加， $SPRCH$ 的误差逐渐减小而 RCH 的误差逐渐增大。图 2 中可以发现 $SPRCH$ 在时间成本上更有优势。需要指出的是，大规模数据计算中读取成本比算法运行成本高很多，新算法 $SPRCH$ 只需要读取原始矩阵 A 一次，而 RCH 至少要读取两次，新算法总体成本比 RCH 算法成本低。

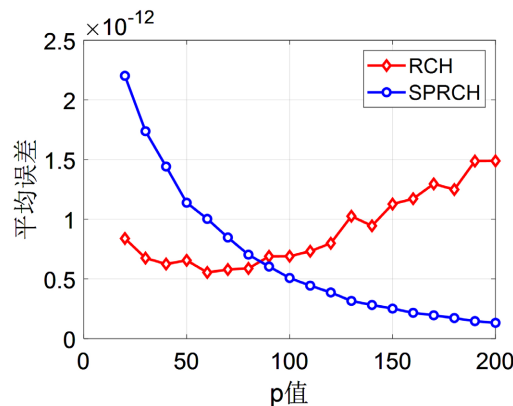


Figure 1. Error line chart for two algorithms for p from 20 to 200
图 1. 参数 p 从 20 到 200 时两种不同算法的误差折线图

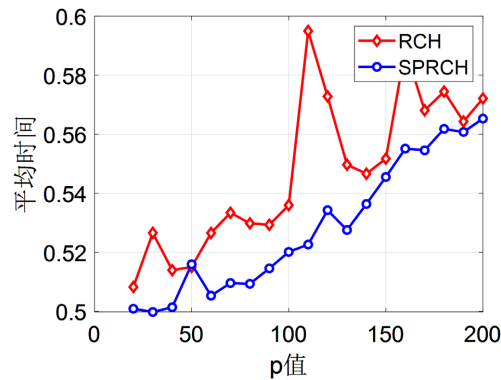


Figure 2. Line chart of running time for two algorithms for p from 20 to 200

图 2. 参数 p 从 20 到 200 时两种不同算法的运行时间折线图

4.2. 不同规模矩阵对于算法的精度及运行时间的影响

根据上节选取合适的过采样参数, 将这两种不同的算法和 *matlab* 中自带的 *Cholesky* 分解作比较, 进行独立的十次实验, 然后记录运行时间及误差分别取平均值为 t_{CH} 、 t_{RCH} 、 t_{SPRCH} 和 e_{CH} 、 e_{RCH} 、 e_{SPRCH} 。

Table 1. Comparison of three algorithms under different scale matrices

表 1. 不同规模矩阵下三种算法的比较

| | 2^9 | 2^{10} | 2^{11} | 2^{12} | 2^{13} |
|-------------|------------|------------|------------|------------|------------|
| t_{CH} | 0.026032 | 0.117295 | 0.811678 | 2.029463 | 8.593510 |
| t_{RCH} | 0.049066 | 0.600490 | 0.271498 | 0.721979 | 3.481178 |
| t_{SPRCH} | 0.208580 | 0.512950 | 0.236635 | 0.523648 | 2.561754 |
| e_{CH} | 4.4651 | 2.5433 | 6.0396 | 7.1385 | 6.6852 |
| e_{RCH} | 2.4871E-13 | 8.5132E-13 | 7.4698E-13 | 1.3596E-12 | 1.0525E-12 |
| e_{SPRCH} | 3.8134E-13 | 6.4950E-13 | 6.5797E-13 | 1.1416E-12 | 1.0130E-12 |

从表 1 可以看出, 随着矩阵规模不断增大, 本文所提算法——*Cholesky* 分解的单 *pass* 随机算法相对于传统的 *Cholesky* 分解以及主元随机 *Cholesky* 分解(*RCH*)算法在时间成本和误差上有着较大的优势。故新算法在应对大规模数据和流型数据矩阵的问题中有更好的应用前景。

参考文献

- [1] 李庆扬, 王能超, 易大义. 数值分析[M]. 第 5 版. 武汉: 华中科技大学出版社, 2018.
- [2] Higham, N.J. (2002) Accuracy and Stability of Numerical Algorithms. Society for Industrial and Applied Mathematics. <https://doi.org/10.1137/1.9780898718027>
- [3] Gu, M. and Miranian, L. (2004) Strong Rank Revealing Cholesky Factorization. *Electronic Transactions on Numerical Analysis*, **17**, 76-92.
- [4] Xiao, J.W. and Gu, M. (2016) Spectrum-Revealing Cholesky Factorization for Kernel Methods. *Proceedings of the 16th IEEE International Conference on Data Mining*, Barcelona, 12-15 December 2016, 1293-1298. <https://doi.org/10.1109/ICDM.2016.0175>

-
- [5] Li, H. and Yin, S. (2020) Single-Pass Randomized Algorithms for LU Decomposition. *Linear Algebra and Its Applications*, **595**, 101-122. <https://doi.org/10.1016/j.laa.2020.03.001>
 - [6] Halko, N., Martinsson, P.G. and Tropp, J.A. (2011) Finding Structure with Randomness: Probabilities Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Review*, **53**, 217-288. <https://doi.org/10.1137/090771806>
 - [7] Tropp, J.A., Yurtsever, A., Udell, M. *et al.* (2018) More Practical Skeching Algorithms for Low-Rank Matrix Approximation. Statistics Department, Caltech, Pasadena.
 - [8] Tropp, J.A., Yurtsever, A., Udell, M. *et al.* (2019) Streaming Low-Rank Matrix Approximation with an Application to Scientific Simulation. *SIAM Journal on Scientific Computing*, **41**, A2430-A2463. <https://doi.org/10.1137/18M1201068>
 - [9] Litvak, A.E., Pajor, A. and Rudelson, M. (2005) Smallest Singular Value of Random Matrices and Geometry of Random Polytopes. *Advances in Mathematics*, **195**, 491-523. <https://doi.org/10.1016/j.aim.2004.08.004>
 - [10] Shabat, G., Shmueli, Y., Aizenbud, Y., *et al.* (2018) Randomized LU Decomposition. *Applied and Computational Harmonic Analysis*, **44**, 246-272. <https://doi.org/10.1016/j.acha.2016.04.006>
 - [11] Gu, M. (2015) Subspace Iteration Randomization and Singular Value Problems. *SIAM Journal on Scientific Computing*, **37**, A1139-A1173. <https://doi.org/10.1137/130938700>