几种稀疏线性系统求解问题的算法探究

1 稀疏矩阵 Cholesky 分解

Cholesky 分解是把一个对称正定的矩阵表示成一个下三角矩阵 $\mathbf{L}$ 和其转置 $\mathbf{L}^T$ 的乘积的分解，每一个对称正定矩阵都有唯一的 Cholesky 分解且 $\mathbf{L}$ 所有对角元素均为正实数。

1.1 LDL 分解

经典 Cholesky 分解的一个变形是 $\mathbf{LDL}$ 分解，即 $\mathbf{A}=\mathbf{LDL}^T$，其中 $\mathbf{L}$ 是一个单位下三角矩阵，$\mathbf{D}$ 是一个对角矩阵。该分解与经典 Cholesky 分解存在一定关系：

$$\mathbf{A}=\mathbf{LDL}^T=\mathbf{L}\mathbf{D}^{\frac{1}{2}}\left(\mathbf{D}^{\frac{1}{2}}\right)^T\mathbf{L}^T=\mathbf{L}\mathbf{D}^{\frac{1}{2}}\left(\mathbf{L}\mathbf{D}^{\frac{1}{2}}\right)^T$$

$\mathbf{LDL}$ 分解如果得以有效运行，构造及使用时所需求的空间及计算的复杂性与经典 Cholesky 分解是相同的，但是可以避免提取平方根。某些不存在 Cholesky 分解的不定矩阵，也可以运行 $\mathbf{LDL}$ 分解，此时矩阵 $\mathbf{D}$ 中会出现负数元素。

1.2 计算方法

1.2.1 Cholesky 算法

用于计算矩阵矩阵 $\mathbf{L}$ 的 Cholesky 算法，是以高斯消元法为基础而调整得来的。

1. 递归算法由 $i=1$ 开始，令 $\mathbf{A}^{(1)}=\mathbf{A}$;
2. 在步骤 i，矩阵 $\mathbf{A}^{(i)}$ 的形式如下：

$$\mathbf{A}^{(i)}=\begin{pmatrix}\mathbf{I}_{i-1} & 0 & 0\\ 0 & a_{i,i} & \mathbf{b}_i^*\\ 0 & \mathbf{b}_i & \mathbf{B}^{(i)}\end{pmatrix}$$

其中，$\mathbf{I}_{i-1}$ 代表 $i-1$ 维的单位矩阵；

3. 此时，定义矩阵 $\mathbf{L}_i$ 为

$$\mathbf{L}_i = \begin{pmatrix} \mathbf{I}_{i-1} & 0 & 0 \\ 0 & \sqrt{a_{i,i}} & 0 \\ 0 & \frac{1}{\sqrt{a_{i,i}}}\mathbf{b}_i & \mathbf{I}_{n-i} \end{pmatrix}$$

随即 $\mathbf{A}^i$ 可以被改写成

$$\mathbf{A}^{(i)} = \mathbf{L}_i \mathbf{A}^{(i+1)} \mathbf{L}_i^*$$

其中

$$\mathbf{A}^{(i+1)} = \begin{pmatrix} \mathbf{I}_{i-1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \mathbf{B}^{(i)} - \frac{1}{a_{i,i}}\mathbf{b}_i\mathbf{b}_i^* \end{pmatrix}$$

4. 重复上述步骤，得到 $\mathbf{A}^{n+1} = \mathbf{I}$。因此，所求的下三角矩阵 $\mathbf{L}$ 为

$$\mathbf{L} = \mathbf{L}_1 \mathbf{L}_2 \dots \mathbf{L}_n$$

1.2.2 LDL 分解

Cholesky 分解的另一种形式LDL分解的计算方式如下：

$$\begin{aligned} \mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^{\mathrm{T}} &= \begin{pmatrix} 1 & 0 & 0 \\ L_{21} & 1 & 0 \\ L_{31} & L_{32} & 1 \end{pmatrix} \begin{pmatrix} D_1 & 0 & 0 \\ 0 & D_2 & 0 \\ 0 & 0 & D_3 \end{pmatrix} \begin{pmatrix} 1 & L_{21} & L_{31} \\ 0 & 1 & L_{32} \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} D_1 & & (\text{symmetric}) \\ L_{21}D_1 & L_{21}^2 D_1 + D_2 & \\ L_{31}D_1 & L_{31}L_{21}D_1 + L_{32}D_2 & L_{31}^2 D_1 + L_{32}^2 D_2 + D_3 \end{pmatrix} \end{aligned}$$

如果 $\mathbf{A}$ 是实数矩阵，下述递归计算式计算适用于矩阵 $\mathbf{D}$ 及 $\mathbf{L}$ 中的所有元素：

$$\begin{aligned} D_j &= A_{jj} - \textstyle\sum_{k=1}^{j-1} L_{jk}^2 D_k \\ L_{ij} &= \frac{1}{D_j}\left(A_{ij} - \textstyle\sum_{k=1}^{j-1} L_{ik}L_{jk}D_k\right), \quad \text{for } i > j \end{aligned}$$

1.3 线性方程组求解

Cholesky 分解主要被用于方程组 $\mathbf{A}\mathbf{x} = \mathbf{b}$ 的求解。如果 $\mathbf{A}$ 是对称正定的，我们可以先求出 $\mathbf{A} = \mathbf{L}\mathbf{L}^T$，随后用向后替换法对 $\mathbf{y}$ 求解 $\mathbf{L}\mathbf{y} = \mathbf{b}$，再以向前替换法对 $\mathbf{x}$ 求解 $\mathbf{L}^{\mathbf{T}}\mathbf{x} = \mathbf{y}$ 即得最终解。另一种可避免在计算 $\mathbf{L}\mathbf{L}^T$ 时需要解平方根的方法就是计算 $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^{\mathrm{T}}$，然后对 $\mathbf{y}$ 求解 $\mathbf{L}\mathbf{y} = \mathbf{b}$，最后求解 $\mathbf{D}\mathbf{L}^{\mathrm{T}}\mathbf{x} = \mathbf{y}$。对于可以被改写成对称矩阵的线性方程组，Cholesky 分解及其 LDL变形是一个较高效率及较高数值稳定性的求解方法。

2 稀疏矩阵LU分解

2.1 稀疏排序

首先要进行排序，其目的是为了减少矩阵LU分解过程中产生的注入元，求解矩阵的最优排序方法是个NP完全问题（不能够在合理的时间内进行求解），对具体矩阵而言，目前也没有方法或指标来判定哪种算法好。因此实测不同的算法，对比产生的注入元，是确定排序算法的可靠依据。目前主要的排序算法有最小度排序 (MMD，minimum degree ordering algorithm) 和嵌套排序 (nested dissection) 两种。

2.2 分解算法

分解算法主要有三种 (这些方法主流求解器的核心代码都在万行以上)：

1. General Technique (通用方法)

   主要采用消去树等结构进行LU分解。适用于非常稀疏的非结构化矩阵。

2. Fontal Scheme (波前法)

   在分解过程中，将连续多行合并为一个密集子块，对子块采用高等数学库进行分解。

   波前法求解基本步骤：

   1. 输入增广矩阵（l个常数列） $\tilde{A} = [\,A \quad b\,] = \left[\,a_{ij}\right]_{n(n+1)}$，精度要求 ε 2) 对 $k = 1, 2, \cdots n-1$ 执行
      1. 行选主元
         - $\left|a_{kj_k}\right| = \max\limits_{k \le j \le n} \left|a_{kj}\right|$；
         - 记录行对应的非零元, Front；
         - 记录主元 j_k, $AIW[k][0] = j_k$；
         - 记录行对应的非零元数目 $AIW[k][2] = \mathrm{Len}\left(\underset{a_{kj} \neq 0}{Front}\right)$；
         - 记录主元在 $\underset{a_{kj} \neq 0}{Front}$ 中的位置 $AIW[k][1] = \mathrm{Index}^{(a_{kj_k})}\left(\underset{a_{kj} \neq 0}{Front}\right)$
      2. 若 $\left|a_{kj_k}\right| \le \varepsilon$，则矩阵奇异，停止计算，否则转 3
      3. 对 $l = k+1, \cdots n$ 执行；若 $\left|a_{lj_k}\right| > \varepsilon$ 则对 $m = 1, 2 \cdots n+1$ 执行 $a_{lm} = a_{\mathrm{km}} - a_{\mathrm{lm}} \times \frac{a_{kj_k}}{a_{lj_k}}$
   2. 回代求解

3. Multi-fontal method (多波前法)

   将符号结构相同的多行(列) 合并为多个密集子块，以这些密集子块为单位进行LU分解。这些子块的生成，消去，装配，释放等都需要特定的数据结构及算法。

3 QR 分解

QR 分解就是将一个矩阵（可以不是方阵）分解成一个正交矩阵和一个上三角矩阵。即

$$A = QR$$

其中 Q 是正交矩阵，即 Q 中的列向量互相正交，即 $Q^T Q = I$； R 是上三角矩阵。当 A 为非奇异矩阵时，这个分解是唯一的。

实现 QR 分解的算法有多种，包括 Gram–Schmidt 正交化，Householder 变换，以及 Givens 旋转。

3.1 Givens 旋转

Givens 旋转在数值计算中的主要作用在于将矩阵中的元素置换为0。具体来说，就是将原矩阵做成一个正交的 Givens 矩阵，然后将原矩阵特定位置上的元素变成0。下面以一个 2×2 矩阵的例子进行说明。

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

在Givens 旋转一般考虑同一列中的两个元素，并通过左乘正交阵的方法把下方的元素变成0，针对上面的矩阵 AA，我们考虑第一列，并对 A 阵左乘一个正交阵，即

$$\begin{bmatrix} 0.3162 & 0.8944 \\ -0.8944 & 0.3162 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 3.1623 & 4.4272 \\ 0 & -0.6325 \end{bmatrix}$$

其中左乘的正交阵被称为 Givens 矩阵，具有形式：

$$G = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

对于多维的矩阵，Givens 矩阵一般写作下面的形式。

$$G = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

QR 分解与 Givens 旋转 下面我们来说明如何通过 Givens 旋转 来实现 QR 分解。其实原理很简单，就是通过将原矩阵 A 的主对角线下方的元素都通过 Givens 旋转置换成0，形成上三角矩阵 R，同时左乘的一系列 Givens 矩阵相乘得到一个正交阵 Q。

可以通过下图来理解，其中 ×× 表示没有发生变化的元素，mm 表示值改变的元素，每一个向右的箭头表示原矩阵左乘了一次 Givens 矩阵。

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \rightarrow \begin{bmatrix} \mathrm{m} & \mathrm{m} & \mathrm{m} \\ \times & \times & \times \\ 0 & \mathrm{m} & \mathrm{m} \end{bmatrix} \rightarrow \begin{bmatrix} \mathrm{m} & \mathrm{m} & \mathrm{m} \\ 0 & \mathrm{m} & \mathrm{m} \\ 0 & \times & \times \end{bmatrix} \rightarrow \begin{bmatrix} \mathrm{m} & \mathrm{m} & \mathrm{m} \\ 0 & \times & \times \\ 0 & 0 & \mathrm{m} \end{bmatrix}$$

即最终可以得到如下的形式：

$$Q^T A = R$$

从而得到

$$A = QR$$

相比于其他两种实现 QR 分解的方法，基于 Givens 选择的方法的优点在于其计算易于并行化，且在面对稀疏矩阵 (Sparse Matrix) 是可以减少计算量。

4 使用 Eigen 库对三种算法的性能进行测试

由于使用直接解法对于求解的矩阵没有太多要求（部分算法要求矩阵为方阵），我们随机在生成的矩阵中随机保留 1/500 个非零元，并令其它位置置零。我们分别对 $100 \times 100, 200 \times 200, 500 \times 500, 1000 \times 1000, 1500 \times 1500, 2000 \times 2000$ 稀疏度为 0.2% 的稀疏矩阵进行了测试，结果汇总如下：

求解时间（单位: ms）	100	200	500	1000	1500	2000
SparseLU	15	36	214	7948	57717	212855
SparseQR	36	133	9577	137915	621839	1869020
Cholesky (LDLt)	3	5	16	52	231	1251

从上表中可以看出，在稀疏度较低（低于 1%），且矩阵规模不大的情况下，Cholesky(LDLt) 算法远远优于另两种；相比之下，LU和QR解法适合求解更加一般矩阵，其结果相比起 Cholesky 算法更为稳定。理论上来说，Cholesky 分解法的性能是LU分解法的两倍左右，但 Eigen 库对于对称正定矩阵的LDLT分解减少了大量重复运算，极大地提高求解速度。

但由于三种算法的时间复杂度其实都是 $O(n^3)$，随稀疏矩阵复杂度的增加，在求解一些稍稠密的稀疏矩阵或规模较大的稀疏矩阵时，直接解法的性能远远不如迭代解法；其次，直接解法中的分解算法相比起迭代解法更为复杂，其并行加速算法的实现也相对更加困难。