# 线性代数随笔【2】

## 线性方程组的解 *by Makiror Ouyang*

# 引言

严格来说这是之前的文章 *线性代数随笔【1】* 的续集，为此我反复品味了几遍，我决定从内容深度来拓展而不是广度，原因很简单——P1 是从几何理解然后把一堆概念扔出来，而没有实际解决问题的考虑，所以本篇我们将从数值的角度来入手。

这次我们将从实际点的角度来对之前的概念进行深化（后来感觉更应该是细化和通俗化）和补充，尽管你可能阅读时不清晰，我也建议你先把P1看完再来，似懂非懂的概念记住就好。

本篇补充/新涉及的内容包括：

- P1缺失的矩阵知识
- 常见的解线性方程组方式
    - 直接的消元和求解方法
    - 一些简单的，可能略微涉及数值分析的迭代求解法

在P1我们更注重依靠几何直觉的理解线代的基本概念，而本文的展开方向将是『线性方程组的解』，这更着重数值计算，可能会涉及到其他领域例如数值分析，但是这不要紧（）。

# 矩阵

我觉得第一个应该捡漏的就是有关矩阵的部分，因为我忽略了一些最基础的东西。倒也不是没提，但是真的不详细，我可能陷入了某人笔下的『知识的诅咒』吧XD

会有不少内容的本质和上篇重复，但是我觉得读者可以当成是看两本书一样（对于同一个学科，只看一本书导致的是片面的理解），毕竟相差一个月的我精神状态也能差很远（x）很多讨论问题的角度和方式也会发生变化。

## 矩阵乘法

两个矩阵 $A, B$ 相乘，则将矩阵 $A$ 的行与 $B$ 的列进行内积运算，例如 $A$ 的维度为 $m \times n$ ,$B$ 的维度为 $n \times p$，则它们乘积矩阵 $C$ 的维度为 $m \times p$。

之前如果没看到后面可能真的不知道这样做是『凭什么』，实际上很简单，我们之前提过线性方程组形如：

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases}$$

则可以以矩阵和向量的形式表示为：

$$Ax = b$$

其中，$A$ 是一个 $m \times n$ 的矩阵， $x$ 是一个 $n \times 1$ 的列向量， $b$ 是一个 $m \times 1$ 的列向量：

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

这样抽象了点，我们只需要同理地写一个例子，考虑一个线性方程组并将其写为矩阵的形式：

$$\begin{cases} a_{11}x_1 + a_{12}x_2 = b_1 \\ a_{21}x_1 + a_{22}x_2 = b_2 \end{cases}$$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

先忽略那个右侧常数项的向量，就看前面的两个矩阵相乘再联想，为什么要这样做矩阵乘法就显而易见了罢。但是这意味着矩阵乘法在 $A$ 的行数与 $B$ 的列数相等时才成立，对于相乘后的每个元素，写为：

$$(AB)_{i,j} = \sum_{r=1}^{n} a_{i,r} b_{r,j}$$

矩阵乘法满足一些性质：

- 乘法结合律： $(AB)C = A(BC)$
- 乘法左分配律： $(A + B)C = AC + BC$
- 乘法右分配律 $C(A + B) = CA + CB$

应该注意一下，和一般实数乘法什么的不同，矩阵乘法不满足乘法交换律。

## 单位矩阵

之前提及过，有一种特殊的矩阵叫单位矩阵，就是一种主对角线上的元素为1，其它元素都为零的对角矩阵。任何矩阵与其相乘都等于它本身、任何正整数次幂也等于它本身，就像实数乘法中的1一样。

我们可以用 $I_n$ 表示 $n$ 阶的单位矩阵，或者在无需特别强调阶数的情况下可以直接简写为 $I$，当然什么字母表示并不重要（以我见 $I$ 更常用，但我也见过使用 $E$ 的，反正看约定吧），而一个矩阵的逆矩阵满足 $A^{-1}A = I_n$，就像我们将一个数乘它的倒数就是1一样，一个矩阵与它的逆相乘就是单位矩阵。

我们可以用矩阵表示线性变换，单位矩阵也一样，但是它不改变任何向量的方向或者大小，它是一个恒等变化。而你可以将单位矩阵的每一列看作空间中的标准基向量，分别指向空间中坐标轴的方向。

## 秩和零度

矩阵的秩是指该矩阵的行（或列）向量中的一个最大线性独立组的维数。从定义看，若有向量组 $\{v_1, v_2, \cdots, v_n\}$，存在不全为零的实数 $k_1, k_2, \cdots, k_n$，使得

$$k_1v_1 + k_2v_2 + \cdots + k_nv_n = 0$$

则我们称这个向量组是线性相关的，否则我们称其为线性独立。也就是在非所有标量都为零的情况下我们不能通过它们的线性组合来表示零向量，从几何来考虑可以理解为一组没有落在同一直线/平面/超平面上的向量，它们不存在共线/共面的关系。而秩实质就是矩阵中的列向量和行向量中线性独立的向量的数量，我们以列为例解释，若我们将矩阵 $A$ 中所有列向量生成的向量空间称为『列空间』，那它这个矩阵的秩 $rank(A)$ 就是列空间中线性独立的向量组。

一个矩阵的零空间（也就是核空间）包含了矩阵A的所有解使得它们与零向量相等的向量，而零化度描述的就是一个矩阵零空间的维度，换句话说就是这个集合中向量的数量。举个例子，考虑一个矩阵 $A$ 和方程 $Ax = 0$:

$$\begin{bmatrix} 1 & 2 \\ 3 & 6 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

其中矩阵的第二行是第一行的倍数，因此它们线性相关，然后我们看到方程组 $Ax = 0$，因为第二行是第一行的背书所以只存在一个自由变量，$x_2$ 可以取任何实数值，$x_1$ 取决于 $x_2$，因此这个矩阵的零空间包含了一个维度为1的线性无关变量，所以它的零化度是1。

这在之前讲过，上篇文章我们从值域的维度引入了秩这个概念，入手的方向是线性变换，也提及了秩-零化度定理，从那个角度我们说线性变换的秩加上零化度等于输入向量空间的维。但现在我们从矩阵论的角度补充就是，若我们用 $col(A)$ 表示矩阵 $A$ 的列数，则秩-零化度定理在这里表示为：

$$rank(A) + nullity(A) = col(A)$$

另外，若矩阵的秩等于其列数，则这个矩阵是满秩的。

## 子式

子式是将一个矩阵中某些行列交点组成的方阵的行列式形式，还有分几种，这个概念很常用，反正就先捡漏一些。

### 余子式和代数余子式

这里在P1讲过（讲行列式时提过）余子式的定义是，在行列式中将 $a_{ij}$ 的 $i$ 行 $j$ 列划去后的部分，叫 $a_{ij}$ 的余子式。阿就例如上面那个三阶行列式， $f$ 的余子式为：

$$M_{23}=\begin{vmatrix} a & b & \square \\ \square & \square & \square \\ g & h & \square \end{vmatrix}=\begin{vmatrix} a & b \\ g & h \end{vmatrix}$$

代数余子式就是，在这个余子式基础上乘以 $(-1)^{i+j}$，表示为：

$$C_{ij}=(-1)^{i+j}\cdot M_{ij}$$

### 主子式和Sylvester判别准则

主子式是从矩阵的左上角开始逐渐增大，定义为在 $n\times n$ 的矩阵 $A$ 中，由前 $k(1\leq k\leq n)$ 行/列组成的 $k\times k$ 子矩阵的行列式，叫 $k$ 阶主子式，表示为 $det(A_k)$。

我们可以通过主子式判断矩阵的可逆性，一个矩阵如果是可逆的，它的所有主子式都不为零。这是Sylvester判别准则的一部分，若矩阵的所有主子式都大于零，则这是正定矩阵；都非负则是半正定矩阵；都小于零则是负定矩阵；都非正则式半负定矩阵；正负都有则被称为不定矩阵。矩阵的正定、负定、半正定半负定、不定对应的是特征值的正负。

- 一个正定矩阵是可逆的，且它的逆也是正定矩阵。
- 正定矩阵的所有特征值都是正数。对于任意非零向量 $x$，都有 $x^\top Ax>0$。半正定矩阵、负定矩阵、半负定矩阵同理。

# 线性方程组的解

## 齐次线性方程组

如果 $b=0$，也就是 $Ax=0$，那这个线性方程组是齐次线性方程组，这个解集 $b$ 是我们在上篇提到的核空间（此处为了浅显点更应该叫零空间），补充说明这在此处指所有输入向量都映射到零向量的线性变换或矩阵的解空间，所以我们可以在讨论齐次线性方程组时使用。一个齐次线性方程组形如：

$$\begin{cases} a_{11}x_1+a_{12}x_2+\cdots+a_{1n}x_n=0 \\ a_{21}x_2+a_{22}x_2+\cdots+a_{2n}x_n=0 \\ \vdots \\ a_{m1}x_1+a_{m2}x_2+\cdots+a_{mn}x_n=0 \end{cases}$$

一个齐次线性方程组总有一个平凡解，也就是所有未知数都取零的解。如果存在一个非平凡解，也就是一个未知数不为零的解，则它一定存在无穷多个非平凡解，因为这意味着与这一个非平凡解成比例的解都满足方程组。

我们可以从矩阵的秩讨论齐次线性方程组的解，如果一个矩阵的秩等于未知变量的数量，则这个对应的方程组只有 $x = 0$ 的平凡解，因为方程组的系数矩阵式满秩的，这意味着它的列向量是线性独立的，无法通过线性组合得到非零解。或者如果矩阵的秩为零，它也是只有平凡解，因为所有行都是零向量，方程组不存在非零解。如果矩阵的秩小于未知变量的数量，则方程组有无穷多个非平凡解（也有平凡解）

要注意的是，因为齐次线性方程组的每个方程的右侧都等于零，如果系数矩阵的秩大于未知变量的数量，那么必然会存在线性无关的冗余方程，所以齐次线性方程组不会出现秩大于未知数数量的情况，不需要考虑。

## 增广矩阵与变换

我们可以考虑系数矩阵 $A$ 和结果向量 $b$ 水平地拼接在一起，形成一个 $m \times (n+1)$ 的矩阵，就是增广矩阵（Augmented matrix），形如（有时竖线可以忽略）：

$$\left[\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} & b_2 \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} & b_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} & b_m \end{array}\right]$$

为了更方便地求解线性方程组，我们可以先对增广矩阵进行初等变换，包括三种对矩阵的运算：

- 交换两行的位置
- 将某一行乘以一个非零的数
- 将某一行加上另一行的某倍数。

这其实就相当于，我们用消元法求解线性方程组时，可以将方程位置对调、可以将方程两边同乘非零常数、可以将方程两边同乘非零常数后加到另一个方程上，而保持解不变。这里是一些例子：

$$[A|b] = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \rightarrow \begin{bmatrix} d & e & f \\ a & b & c \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \rightarrow \begin{bmatrix} 2a & 2b & 2c \\ d & e & f \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \rightarrow \begin{bmatrix} a & b & c \\ d+2a & e+2b & f+2c \end{bmatrix}$$

## 阶梯形矩阵

在矩阵中，每一行的第一个非零元素被称为主元（或先导元素），若一个矩阵每一行的主元都在前一行的主元之后，

以及每一行的主元素下方的所有元素都是零，则这个矩阵是一个阶梯形矩阵（Row echelon form）。例如：

$$\begin{bmatrix} 3 & 2 & 2 & 6 \\ 0 & 1 & 4 & 3 \\ 0 & 0 & 2 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

在此基础上，若阶梯形矩阵的主元素都为1，且都在对角线上，主元的所在列除了它本身以外，其他元素都为零，那这个矩阵被称为行最简形矩阵，上面的矩阵不是一个行最简形矩阵，但是这个是：

$$\begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 4 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

任何矩阵都可以通过初等变换被转化为阶梯形矩阵，但是要注意的是，转化为阶梯形矩阵并不意味着这一定是行最简形矩阵。另外，计算所得的行阶梯矩阵的非零行的数量就是矩阵的秩。

## 高斯消元法

有了前置概念后，我们就能考虑使用高斯消元法来求解线性方程组。若按照本章第一节的定义，将一个线性方程组表示为

$$Ax = b$$

我们将其写为增广矩阵的形式 $[A|b]$，并使用初等变换将其转换为阶梯形矩阵，这里就是所谓『消元』。在此过程中我们通常会选择当前列中绝对值最大的元素作为主元（某种意义上来说这是自由的，但是你选择的主元素会很大程度影响计算的效率），并将其所在行交换到当前处理行，将所在行除以主元的值使其变为1，再进行行变换使下一行对应元素为零。重复此步骤直到处理完所有列或其满足无解条件。然后我们就把未知变量代回去进行求解，这个步骤也叫回代。

当我们完成消元过程后，就能判断方程组的解了，首先如果消元结果没有类似于 $0 = 1$ 这样的矛盾，且没有出现自由变量，每一个未知数都有一个数与之对应，那这就是方程组的唯一解。

如果消元后的矩阵有一行系数都为0，但最右侧的常数列不为零，则方程组无解，例如（关注第三行）：

$$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 4 \\ 0 & 0 & 3 \end{bmatrix}$$

若消元出现了没有对应的主元变量的变量，也就是所谓的自由变量，那方程组有无穷多解，因为自由变量可以取任何值，例如（关注第三行）：

$$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 4 \\ 0 & 0 & 0 \end{bmatrix}$$

我们应该理解的是，高斯消元法会发生改变的是方程中变量的系数，而变量本身并没有参与消元，以简化省略方程组

对变量的处理。

此外，我们还能使用高斯消元法来求一个矩阵的逆矩阵。假设我们要求一个矩阵 $A$ 的逆矩阵，则先将其与一个单位矩阵 $I$ 合并，形成一个增广矩阵 $[A|I]$，并将其转换为阶梯形矩阵（如果无法转换则代表该矩阵不可逆），化简后我们进行行变换使得增广矩阵的左侧变为单位矩阵，此时它的右侧就是矩阵 $A$ 的逆矩阵。

## LU分解

我们也可以使用LU分解来解线性方程，之所以叫LU分解，是因为它的主要目标就是将我们要求解的矩阵分成两个三角矩阵，一个下三角矩阵 $L$（Lower Triangular）和上三角矩阵 $U$（Upper Triangular）。形如：

$$A = LU = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

分解后等式 $Ax = b$ 就可以改写为 $L(Ux) = b$，我们定义 $Ux = y$，则有等式：

$$\begin{gathered} Ly = b \\ Ux = y \end{gathered}$$

我们求解关于向量 $y$ 的方程 $Ly = b$，并将结果代入求解关于向量 $x$ 的方程 $Ux = y$ 即可。这个可能稍微抽象点，所以我们以一个非常简单的情况来理解好了，考虑线性方程组及其增广矩阵形式：

$$\begin{cases} 2x_1 + 3x_2 = 8 \\ 4x_1 + 7x_2 = 14 \end{cases}$$

$$\left[\begin{array}{cc|c} 2 & 3 & 8 \\ 4 & 7 & 14 \end{array}\right]$$

显而易见，我们可以对其进行变换，将第二行减去两倍的第一行得到矩阵 $U$，并将乘数 $2$ 记录于 $L$ 对应的地方，分解后的 $L$ 和 $U$：

$$L = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}, U = \begin{bmatrix} 2 & 3 \\ 0 & 1 \end{bmatrix}$$

分解其实很简单，下三角矩阵 $L$ 实质就是记录高斯消元过程的步骤，用于对上三角矩阵 $U$ 进行行变换的。现在我们继续按照上述的步骤来解这个方程组，先考虑方程 $Ly = b$：

$$\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 8 \\ 14 \end{bmatrix}$$

$$\begin{gathered} y_1 = 8 \\ 2(8) + y_2 = 14 \\ y_2 = -2 \end{gathered}$$

$$y = \begin{bmatrix} 8 \\ -2 \end{bmatrix}$$

我们再将 $y$ 代入到方程 $Ux = y$ 中继续求解：

$$\begin{bmatrix} 2 & 3 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 8 \\ -2 \end{bmatrix}$$

$$\begin{gathered} 2x_1 + 3x_2 = 8 \\ x_2 = -2 \\ 2x_1 + 3(-2) = 8 \\ x_1 = 7 \end{gathered}$$

$$x = \begin{bmatrix} 7 \\ -2 \end{bmatrix}$$

所以，该线性方程组的解为 $\begin{cases} x_1 = 7 \\ x_2 = -2 \end{cases}$

前面我们求解方程 $Ly = b$ 的方法叫前向替代，这是指适用于解下三角形矩阵与向量的乘积的方法，因为下三角形矩阵的结构适合从前向后逐步代入求解未知变量，在下三角矩阵中，每个方程只包含前面方程中已知的未知数，因此我们可以逐个解出未知数，而不需要回头重新计算已知的值。对应地，我们求解方程 $Ux = y$ 的方法就叫后向替代，适用于上三角矩阵与向量的乘积求解。

另外，LU分解主要有两种，在其中一种也就是上面的Doolittle解法中 $L$ 的下三角是不包含主对角线在内的，$L$ 的主对角线为1，而另外一种叫Crout解法的 $U$ 主对角线才是1。但是之后如果有提及或者应用其变体，在默认情况下仍然是 $L$ 的主对角线为1

还有，LU分解的存在是有条件的，被分解的矩阵 $A$ 必须是一个方阵，且所有主子式不为零。

## PLU 分解

PLU分解是LU分解的一个变体，适用于所有的方阵（非方阵可能不适应）在LU分解的基础上引入了一个置换矩阵 $P$（Permutation matrix），也就是只由0和1组成的，且每行和每列都有且只有一个1，其余都是0的方阵，即：

$$A = PLU$$

在基本的LU分解中，我们不会考虑原始矩阵某些主元为零的情况，这样可能会导致分解会失败。而在PLU分解中我们通过这个置换矩阵 $P$ 记录行交换操作，这样我们就能通过行交换确保主元不为零。若我们要用一个三阶置换矩阵表示行1与行2位置互换，可以表示为：

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

同理地我们将置换矩阵应用到分解的过程，考虑一个矩阵 $A$，并将 $P$ 初始化为一个三阶的单位矩阵。

$$A=\begin{bmatrix}2&1&1\\4&3&3\\8&7&9\end{bmatrix},P=\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$$

我们希望选取一个主元，以便在高斯消元过程中能够迅速将该主元下方的元素变为零，通常，选择绝对值最大的元素作为主元是一个好策略，因为这可以最大程度地减小舍入误差，所以在这个例子中我们选择第一列中绝对值最大的元素 8 作为主元，并将其移到第一行，为了表示第一行和第三行交换，矩阵 $P$ 变为：

$$P=\begin{bmatrix}0&0&1\\0&1&0\\1&0&0\end{bmatrix}$$

这样我们就可以继续进行LU分解来解方程辣。毕竟PLU分解存在的意义很大程度是数值稳定性，所以它更多在实际应用会被重视（这对于处理矩阵中存在较小主元或近似奇异的情况非常重要）

# 线代的数值分析

Uh非要说的话其实这里的内容也可以归于上一章，但是从这里开始其实内容也会涉及到简单的数值分析，而为了 ~~水篇幅~~ 更好的组织文章结构在这里开一个二级标题，实际内容和上一章有很多关系。

## 矩阵的收敛性

之前提过，我们可以使用范数来度量向量的大小，而在一个向量序列 $\{v_1,v_2,\cdots,v_n\}$ （它们的维度相同）中，对于一个任意（一般比较小的）正数 $\epsilon$ ，存在一个正整数 $N$ 使得当 $n>N$ 时序列中每个向量都满足 $\|v_n-v\|<\epsilon$，则我们称这个向量序列在范数度量下收敛到向量 $v$，或 $v$ 是这个向量序列的极限。

如果你看过我以往的文章 *【微积分入门】从极限到黎曼积分的探索* 的话，对这个概念不会陌生，我们期望向量序列中的项与收敛到的向量 $v$ 的差值都小于 $\epsilon$ ，也就是 $\epsilon$ 在此作为一个收敛的精度。而正整数 $N$ 表示从这项开始序列中的每一项都满足 $\|v_n-v\|<\epsilon$，即对收敛范围的控制。当 $n$ 趋近于无穷大时，序列中的向量 $v_n$ 收敛到向量 $v$，表示为：

$$\lim_{n\to\infty}v_n=v$$

同理地，一个矩阵序列也可以收敛于一个矩阵，前提也是它们的维度相同，在矩阵序列 $\{A_1,A_2,\cdots,A_n\}$ ，对于正数 $\epsilon$ 存在一个正整数 $N$ 使得当 $n>N$ 时序列中每个矩阵都满足 $\|A_n-A\|<\epsilon$，则我们称这个矩阵序列在范数度量（注意选择合适的矩阵范数）下收敛到矩阵 $A$，或 $A$ 是这个矩阵序列的极限。

$$\lim_{n\to\infty}A_n=A$$

## Jacobi迭代法

之前我们解的那些很简单的矩阵，所用的方法，什么消元阿分解啊都叫做直接解法都是针对那种中小型且比较密集的矩阵的，而对于那种大且稀疏的矩阵可能会变得不好处理，而我们可以考虑使用迭代解法来逼近方程组的解，先从最简单的迭代解法之一Jacobi迭代法开始好了。

考虑 $Ax = b$ 我们将方程分解为 $A = L + D + U$，对角矩阵 $D$ ，其对角线上的元素与原矩阵 $A$ 的对角线上的元素相同，而下三角矩阵 $L$ 和上三角矩阵 $U$ 的主对角线的元素都为 0：

$$L = \begin{bmatrix} 0 & & & \\ a_{21} & 0 & & \\ \vdots & \ddots & \ddots & \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{bmatrix}, D = \begin{bmatrix} a_{11} & & & \\ & a_{22} & \ddots & \\ & & & a_{nn} \end{bmatrix}, U = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ & 0 & \ddots & \vdots \\ & & \ddots & a_{n-1,n} \\ & & & 0 \end{bmatrix}$$

但是这并不是LDU分解（一种LU分解的变体），这里不要搞混了。经过代入和移项后后，我们得到

$$(D + L + U)x = b$$
$$Dx = -(L + U)x + b$$

我们将方程的两边同时乘以 $D$ 的逆矩阵 $D^{-1}$，得到

$$D^{-1}Dx = D^{-1}(-(L + U)x + b)$$

由于一个矩阵乘以它的逆矩阵得到的是单位矩阵 $I$，所以我们可以直接消去左边的 $D^{-1}D$ 得到了等式：

$$x = -D^{-1}(L + U)x + D^{-1}b$$

啊然后我们就能使用Jacobi迭代公式来更新未知数 $x$ 的估计值，此处我们约定使用上标来表示迭代的次数，并且此处的 $i, j$ 另有用所以用 $k$ 做为索引，对于每个 $x$ 我们可以这样表示，或写作求和形式，这里提供两种形式方便理解：

$$x^{(k+1)} = -D^{-1}(L + U)x^{(k)} + D^{-1}b$$

$$x_i^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sum_{j \neq i} a_{ij}x_j^{(k)}), i = 1 \cdots n$$

我们现在要进一步讨论收敛性的问题，首先，若一个矩阵的每一行，主对角线元素的绝对值都大于其他元素绝对值之和，即满足

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|, i = 1 \cdots n$$

则我们称这个矩阵是严格对角占优的，通常具有这个性质的矩阵的收敛性会更好。或者一个正定的线性方程组一般也会有更好的收敛性。

我们更多得考虑收敛的条件，我们可以考虑这些方面来判断收敛并终止迭代：

- 定义一个容差 $\epsilon$，当每次迭代中的估计解 $x^{(k)}$ 与上一次迭代的 $x^{(k-1)}$ 的差小于这个容差则判断迭代已满足收敛所需的精度，即满足：

$$||x^{(k)} - x^{(k-1)}|| < \epsilon$$

- 定义每次迭代计算的残差向量（可以理解为误差）为 $r^{(k)} = b - Ax^{(k)}$ 的范数，并且与一个容差 $\epsilon$ 进行比较，如果解的误差小于这个容差则判断迭代已收敛并满足所需的精度，即：

$$||r^{(k)}|| < \epsilon$$

### 谱半径

矩阵的谱半径是矩阵的特征值（eigenvalues）的绝对值的最大值，对于一个矩阵 $A$ 的谱半径，我们可以表示为 $\rho(A)$，若它的维度为 $n \times n$ 其特征值为 $\lambda_1, \lambda_2, \cdots, \lambda_n$ ，则定义为：

$$\rho(A) = \max_i |\lambda_i|$$

一个矩阵的谱半径小于1，通常情况下这意味着它的收敛性良好（一个矩阵式严格对角占优时，它的谱半径必然小于1）。通俗的说，我们可以使用谱半径来判断这个矩阵是否有望收敛，但不能直接通过这个值来判断它是否已收敛，我们可以借此来在迭代时判断是否应该结束一个没有必要的迭代。

从几何角度来看，Jacobi 迭代法可以被理解为将解空间中的解沿着坐标轴方向进行迭代调整。每个 Jacobi 迭代步骤都相当于在一个坐标轴方向上进行一次调整，然后依次在每个坐标轴上进行调整，直到达到收敛条件。

## Gauss-Seidel 迭代

Gauss-Seidel（高斯-赛德尔）迭代也是一种常见的迭代法，名字比较长后面为了方便统一叫G-S迭代。
G-S的前半部分基本和Jacobi是同理的，更新公式也一样：

$$x_i^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)}), i = 1 \cdots n$$

但是，在Jacobi迭代中变量的更新使用的是上次迭代的旧值，而在G-S迭代中，每个变量的更新依赖于当前迭代中已经更新过的其他变量的新值。细说，在GS迭代中的 $x_j^{(k)}$ 是已经更新过的它的最新值，这是在 $j < i$ 的情况下，如果 $j \geq i$ 则使用的是未更新的旧值。而在Jacobi迭代中的 $x_j^{(k)}$ 我们使用的都是上一次迭代的旧值，Jacobi迭代的更新发生在每一轮迭代的开始时，新的值将在下一轮迭代中使用而非G-S那样及时地更新使用。
由于G-S变量的及时更新，在很多时候它的收敛速度会比Jacobi快，但是由于它的更新顺序依赖于先前变量的值，所以我们在实际应用中如果要进行并行计算会比较麻烦。反观Jacobi迭代如果要并行处理就很简单，每个进程可以独立计算每一行，非常合适。

## SOR方法和松弛因子

SOR（Successive Over Relaxation）方法在Jacobi迭代和G-S迭代的基础上引入了一个松弛因子，松弛因子可以通过加权改进迭代的效率。基本思想无非就是在每个迭代步骤中将当前迭代的解与迭代的解进行线性结合，并调整这个参数来控制这个线性组合的权重。

通常我们用 $\omega$ 表示松弛因子，其取值范围通常在0-2之间。在使用松弛因子的迭代方法中，更新公式改写为：

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + (\frac{\omega}{a_{i,i}})(b_i - \sum_{j \neq i}(a_{i,j}x_j^{(k)})), i = 1 \cdots n$$

当 $\omega = 1$ 时，它与之前的两种方法无异（差别只在于两种方法之间的区别，就是变量更新），而当 $\omega < 1$ 时我们采用的是欠松弛，相对的收敛会更稳定但是速度对应的也会慢，而 $\omega > 1$ 则叫做过松弛，迭代速度会更快但是可能会比较不稳定。你或许可以将这个参数理解为我们在收敛过程中是否更看重上次的迭代解。

在最后，我们再了解一种换了一个角度的迭代法。

## 线性共轭梯度

在我曾经的文章 *从初中数学开始的Linear Regression* 提及过梯度下降算法，用于求解最小化函数的问题，它的主要思想也是通过不断迭代来更新参数，使目标函数的值逐渐减小，直到达到一个满意的最小值或近似最小值。

共轭梯度（Conjugate gradient，后面为了方便我们称其为CG）也是一种迭代方法，但它要求矩阵 $A$ 是对称且正定的。

> 在写文章的前两天我已经在学校的特种兵作息间隙中啃完了关键的 *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*。
> （这玩意最好是 Without the Agonizing Pain，但确实很推荐看原文 ）
> http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf
>
> 另外CG的Wiki的参考资料都是法语我真的原地气急败坏

若有两个向量 $u, v$，$A$ 是一个对称正定矩阵，若它们满足 $u^\top Av = 0$ 则我们说这两个向量组矩阵 $A$ 的作用下是共轭的，并且这也意味着它们的向量内积为0，则在 $A$ 的作用下互不干扰。

与之前对残差的定义一样，我们定义在第 $k$ 论迭代时 $Ax$ 与 $b$ 的误差，残差向量 $r_k = b - Ax^{(k)}$，在迭代的最开始我们要初始化一个残差，一般初始化为 $r_0 = b - Ax_0$，并且初始化一个搜索的方向 $p_0$，一般我们可以取 $p_0 = r_0$，这保证了 $r_0$ 和 $p_0$ 共轭，在什么都不知道的情况下我们使用这样的方法选择收敛方向一般不会对后续的收敛造成负面影响。

我们通过计算步长 $\alpha_k$，使得 $x_{k+1} = x_k + \alpha_k p_k$ 最小化 $||r_k||$，计算公式为：

$$\alpha_k = \frac{r_k^\top r_k}{p_k^\top Apk}$$

其中的分子 $r_k^\top r_k$ 是当前残差的二次范数的平方，可以理解为当前的残差大小。分母 $r_k^\top r_k$ 表示搜索方向 $p_k$ 应用在矩阵 $A$ 的长度的平方，也就是在搜索方向上步长导致的变化。这个比例 $\alpha$ 告诉我们的就是搜索方向要移动多远才能在下一步减小残差 $r_k$，如果 $\alpha$ 较大的情况下我们就能一次地移动更远以接近最优解。这样，我们就能很简单地更新

解 $x^(k)$ 使其往减小残差的方向前进：

$$x^{(k)} = x^{(k)} + \alpha_k p_k$$

同样地，在下一次迭代开始之前我们应该先更新收敛方向，分为两步：

$$\beta_k = \frac{r_{(k+1)}^\top r_{(k+1)}}{r_k^\top r_k}$$

$$p_{k+1} = r_{k+1} + \beta_k p_k$$

在第一个公式，$\beta$ 是计算搜索方向的系数，实际上是在衡量两个连续步骤的残差 $r_k$ 和 $r_{k+1}$ 之间的关系，若这个值为零则意味着它们之间已经没有共轭性了，我们这么做是为了确保新的搜索方向与之前的搜索方向共轭。第二行公式我们计算新的收敛方向，这个 $\beta$ 在这里的作用嘛...简单地说就是根据之前搜索方向导致的残差大小来决定是更『关注』还是『保留』之前的搜索方向，然后我们再结合新的残差方向和之前搜索方向提供的信息在搜索空间中前进。

同样，在最后我们可以简单粗暴地为残差向量定义一个容差 $\epsilon$ 来结束迭代，即 $\|r_k\| \leq \epsilon$，或者通过解的变化来判断迭代是否已经收敛，若这个值为 $\epsilon$，则是满足 $\|x_{k+1} - x_k\| \leq \epsilon$ 时结束迭代。

# 结尾

结束了，上次的 *线性代数随笔【1】* 的流量相对而言不错所以我还是尽快写了2（鞠躬感谢），大概查资料和写用了断断续续的十几个小时吧。组织这次Blog内服的经验为我正在写的机器学习库提供了很多灵感，这对我而言也是相当有意义的。

想说声抱歉的就是这篇博客有不小一部分是我在学校以分钟为单位的零散时间组织的，而这些内容只是我乱七八糟的知识体系中的一部分，所以内容的尺度一直变来变去，我已经在我精神状态允许的范围内将它们连接起来了（这可能也是P2仍然叫**“随笔”**的原因）。

近期我正在改造我的网站，之后可能会陆续上传之前文章（尤其是数学相关的）的修正、补充以及英翻 → http://www.makiror.xyz。另外我想尝试写一些其他类型的文章，也欢迎各位给我点什么建议。

## 一些参考链接

对不起这次轮到参考链接写不完了！我独自赴日回来后一直在~~闭关修炼~~，啃了很多书籍讲义论文以及公开课...语言涉及中日英德法都有，反正我就留下一些我认为对读者学习更有帮助的资料吧。

书籍：

线性代数及其应用 （*Linear Algebra and Its Applications*）：https://www.zhihu.com/topic/20129721/hot

*Introduction to Linear Algebra* by William Gilbert Strang: https://math.mit.edu/~gs/linearalgebra/

*Linear Algebra* by Jim Hefferon (6th Edition 2023 edition): https://math.mit.edu/~gs/linearalgebra/ila6/indexila6.html

论文：
*An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*:
http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf

（对深入某个内容感兴趣的小伙伴可以私信或者电邮我，我有看过不少其他针对文章某个内容深入分析的论文）

讲义：
*Numerical Linear Algebra* by Lloyd N. Trefethen and David Bau: https://www.stat.uchicago.edu/~lekheng/courses/309/books/Trefethen-Bau.pdf
線形代数学A（2017年，京都大学）：https://www.kurims.kyoto-u.ac.jp/~shimizu/LectLA2017.html

MAKIROR gzanan@gmail.com 22:24 16/09/2023