

我是8位的

I am 8 bits, what about you?

随笔 - 205, 文章 - 0, 评论 - 103, 阅读 - 101万

导航

博客园
首页
新随笔
联系
订阅
管理

2022年3月						
日	一	二	三	四	五	六
27	28	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

公告

你的支持是我的动力
欢迎关注微信公众号“我是8位的”



昵称：我是8位的
园龄：4年7个月
粉丝：288
关注：5
+加关注

盖楼抽奖

#她的梦想在发光#

HWD科技女性故事有奖征集

分享最打动的科技女性故事

活动时间: 2022年3月8日-3月18日

马上参与



搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

积分与排名

积分 - 457097
排名 - 1198

线性代数笔记10——矩阵的LU分解

在线性代数中，LU分解(LU Decomposition)是矩阵分解的一种，可以将一个矩阵分解为一个单位下三角矩阵和一个上三角矩阵的乘积（有时是它们和一个置换矩阵的乘积）。LU分解主要应用在数值分析中，用来解线性方程、求反矩阵或计算行列式。

什么是LU分解

如果有一个矩阵A，将A表示成下三角矩阵L和上三角矩阵U的乘积，称为A的LU分解。

$$A = \begin{bmatrix} \blacksquare & 0 & 0 & 0 & 0 \\ \blacksquare & \blacksquare & 0 & 0 & 0 \\ \blacksquare & \blacksquare & \blacksquare & 0 & 0 \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & 0 \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \end{bmatrix} = \begin{bmatrix} \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ 0 & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ 0 & 0 & \blacksquare & \blacksquare & \blacksquare \\ 0 & 0 & 0 & \blacksquare & \blacksquare \\ 0 & 0 & 0 & 0 & \blacksquare \end{bmatrix}$$

更进一步，我们希望下三角矩阵的对角元素都为1：

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \blacksquare & 1 & 0 & 0 & 0 \\ \blacksquare & \blacksquare & 1 & 0 & 0 \\ \blacksquare & \blacksquare & \blacksquare & 1 & 0 \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & 1 \end{bmatrix} = \begin{bmatrix} \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ 0 & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ 0 & 0 & \blacksquare & \blacksquare & \blacksquare \\ 0 & 0 & 0 & \blacksquare & \blacksquare \\ 0 & 0 & 0 & 0 & \blacksquare \end{bmatrix}$$

一旦完成了LU分解，解线性方程组就会容易得多。

LU分解的步骤

上一章讲到，对于满秩矩阵A来说，通过左乘一个消元矩阵，可以得到一个上三角矩阵U。

$$\begin{aligned} A &= \begin{bmatrix} 2 & 1 \\ 8 & 7 \end{bmatrix} \\ \begin{bmatrix} 2 & 1 \\ 8 & 7 \end{bmatrix} &\xrightarrow{\begin{smallmatrix} (1,0) \\ (-4,1) \end{smallmatrix}} \begin{bmatrix} 2 & 1 \\ 0 & 3 \end{bmatrix} \Rightarrow E_{21} = \begin{bmatrix} 1 & 0 \\ -4 & 1 \end{bmatrix} \\ &\Rightarrow \underbrace{\begin{bmatrix} 1 & 0 \\ -4 & 1 \end{bmatrix}}_{E_{21}} \underbrace{\begin{bmatrix} 2 & 1 \\ 8 & 7 \end{bmatrix}}_A = \underbrace{\begin{bmatrix} 2 & 1 \\ 0 & 3 \end{bmatrix}}_U \\ &\Rightarrow A = E_{21}^{-1}U = LU \end{aligned}$$

可以看到，L实际上就是消元矩阵的逆。容易知道二阶矩阵的逆：

$$E_{21}^{-1} = \begin{bmatrix} 1 & 0 \\ 4 & 1 \end{bmatrix} = L$$

现在假设A是一个3×3矩阵，在不考虑行交换的情况下，通过消元得到上三角矩阵的过程是：

随笔分类 (211)

- ★★资源下载★★(1)
- Java并发编程(1)
- 程序员的数学(24)
- 单变量微积分(31)
- 多变量微积分(24)
- 概率(24)
- 机器学习(27)
- 软件设计(1)
- 数据分析(6)
- 数据结构与算法(27)
- 随笔(5)
- 线性代数(34)
- 项目管理(2)
- 转载(4)

随笔档案 (205)

- 2021年2月(1)
- 2020年3月(2)
- 2020年2月(6)
- 2020年1月(4)
- 2019年12月(7)
- 2019年11月(15)
- 2019年9月(3)
- 2019年8月(6)
- 2019年7月(1)
- 2019年6月(8)
- 2019年5月(3)
- 2019年4月(5)
- 2019年3月(7)
- 2019年2月(3)
- 2019年1月(7)
- 更多

阅读排行榜

- 1. 使用Apriori进行关联分析 (一) (29768)
- 2. 线性代数笔记12——列空间和零空间 (28772)
- 3. FP-growth算法发现频繁项集 (一)——构建FP树(24430)
- 4. 寻找“最好” (2) ——欧拉-拉格朗日方程(23099)
- 5. 多变量微积分笔记3——二元函数的极值(22772)

评论排行榜

- 1. 隐马尔可夫模型 (一) (8)
- 2. 线性代数笔记12——列空间和零空间 (7)
- 3. 线性代数笔记3——向量2 (点积) (7)
- 4. FP-growth算法发现频繁项集 (一)——构建FP树(5)
- 5. 寻找“最好” (2) ——欧拉-拉格朗日方程(4)

推荐排行榜

- 1. 寻找“最好” (2) ——欧拉-拉格朗日方程(7)
- 2. FP-growth算法发现频繁项集 (一)——构建FP树(7)
- 3. 线性代数笔记3——向量2 (点积) (6)
- 4. FP-growth算法发现频繁项集 (二)——发现频繁项集(5)
- 5. 隐马尔可夫模型 (一) (5)

最新评论

- 1. Re:线性代数笔记3——向量2 (点积)
如果点积小于0，即夹角小于90°，这个写错了吧。应该是夹角大于90°
--猫猫猫猫猫大人

$$(E_{33}E_{31}E_{21})A = U$$

$$A = (E_{33}E_{31}E_{21})^{-1}U = E_{21}^{-1}E_{31}^{-1}E_{32}^{-1}U = LU$$

LU 分解的前提

并非所有矩阵都能进行LU分解，能够LU分解的矩阵需要满足以下三个条件：

- 1. 矩阵是方阵（LU分解主要是针对方阵）；
- 2. 矩阵是可逆的，也就是该矩阵是满秩矩阵，每一行都是独立向量；
- 3. 消元过程中没有0主元出现，也就是消元过程中不能出现行交换的初等变换。

LU分解的意义

LU分解的意义在于求解大型方程组。一个方程组可以简化为Ax = b的形式，其中A是n阶方阵，x是未知数组成的向量，b是n×1矩阵，例如：

$$\begin{cases} x + y + 2z = 1 \\ x - 2y - z = 0 \\ y - 2z = -2 \end{cases} \Rightarrow \underbrace{\begin{bmatrix} 1 & 1 & 2 \\ 1 & -2 & -1 \\ 0 & 1 & -2 \end{bmatrix}}_A \underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_x = \underbrace{\begin{bmatrix} 1 \\ 0 \\ -2 \end{bmatrix}}_b$$

以往求解的方式有两种，一是高斯消元法，二是对A求逆，使得x = A⁻¹b。第二种方式远比消元法复杂，先看一下消元法的计算量。假设A是n阶满秩方阵，如果不写成增广矩阵，即不考虑 b，那么第一次消元达到的效果是：

$$\begin{bmatrix} \blacksquare & \blacksquare & \cdots & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \cdots & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \cdots & \blacksquare & \blacksquare \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \blacksquare & \blacksquare & \cdots & \blacksquare & \blacksquare \end{bmatrix} \rightarrow \begin{bmatrix} \blacksquare & \blacksquare & \cdots & \blacksquare & \blacksquare \\ 0 & \Delta & \cdots & \Delta & \Delta \\ 0 & \Delta & \cdots & \Delta & \Delta \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \Delta & \cdots & \Delta & \Delta \end{bmatrix}$$

其中方块是A原来的元素，0是达到的效果，三角是经过消元运算后改变的元素。以第二行为例，为了使第一个元素为0，需要让第一行乘以某个数（第一行n个元素，共进行了n次乘法运算），再将第一行和第二行相加或相减（第二行n个数与第一行的n个数相加，共进行了n次加法运算）。如果把一组乘法和加法看成一次运算，那么第二行的消元共进行了n次运算；共有n-1行需要类似运算，所以第一次消元共进行了n(n - 1) ≈ n² 次运算。依次类推，第二次消元共进行了(n - 1)(n - 2) ≈ (n - 1)² 次运算.....消元到最后，变成了上三角矩阵U，总运算次数是：

$$N = n^2 + (n - 1)^2 + (n - 2)^2 + \cdots + 2^2 + 1^2$$

$$set \Delta n = 1$$

$$N \approx \int_1^n n^2 dn \approx \frac{1}{3} n^3$$

经过约n³/3次运算后可以得到上三角矩阵U，由于是增广矩阵，所以可以逐步求解x。

2. Re:线性代数笔记10——矩阵的LU分解写的很好，不过LU分解的前提是错的，LU分解只需要第三个条件，如果允许行置换就是下面写到的PLU，可以分解所有矩阵

--wiki3D

3. Re:单变量微积分笔记20——三角替换1 (sin和cos) 很nice

--尹保棕

4. Re:线性代数笔记24——微分方程和exp(At)

有些图片挂了呢

--ccchendada

5. Re:寻找“最好” (2) ——欧拉-拉格朗日方程

提个issue，最速降线中

$v = \{2gh\}^{1/2}$ 与配图不一致，建议以起点为原点，向右伸出x轴，向下伸出y轴建立坐标系

--trustInU

LU分解的运算过程和高斯消元类似，首先经过 $n^3/3$ 次运算将A变成

LU，使 $Ax = b$ 变成 $(LU)x = L(Ux) = b$ ，再对L求逆，使得 $Ux = L^{-1}b$ ，最后求解。

看起来比高斯消元经历了更多的步骤，那为什么又说LU分解更快呢？

在实践中，b是输出，输出又经常变动，从 $Ax = b$ 频繁地变成 $Ax = b'$ ，此时高斯消元就需要全部重新计算（高斯消元用增广矩阵消元，变化过程是 $[A, b] \rightarrow [U, b']$ ），这对大型矩阵来说及其耗时。反观LU分解，因为它不依赖于b，所以计算一次后就可以存储U和 L^{-1} ，在输出变化后也只是需要简单的相乘。实际上，由于L已经是整理过的斜对角全是1的下三角矩阵，所以用高斯-诺当消元法对L求逆非常简单。

允许行交换

对于 $A = LU$ ，我们之前限制了行的互换，但如果不可避免的必须进行行互换，只需要把 $A = LU$ 变成 $PA = LU$ 就可以了，其中P是置换矩阵。实际上所有的 $A = LU$ 都可以写成 $PA = LU$ 的形式，当A没有行互换时，P就是单位矩阵。上一章叙述了置换矩阵的性质， $P^{-1} = P^T$ ，所以 $A = P^{-1}LU = P^T LU$

示例

$$A = \begin{bmatrix} 1 & 0 & 1 \\ a & a & a \\ b & b & a \end{bmatrix}$$

如果A存在LU分解，a,b满足什么条件？

使用消元法逐一消去主元：

$$\begin{aligned} E_{21}A &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & a & 0 \\ b & b & a \end{bmatrix}, & E_{21} &= \begin{bmatrix} 1 & 0 & 0 \\ -a & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ E_{31}E_{21}A &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & a & 0 \\ 0 & b & -b+a \end{bmatrix}, & E_{31} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -b & 0 & 1 \end{bmatrix} \\ E_{32}E_{31}E_{21}A &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & a & 0 \\ 0 & 0 & -b+a \end{bmatrix}, & E_{31} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -b/a & 1 \end{bmatrix} \end{aligned}$$

U

由于 E_{31} 中出现了 $-b/a$ ，所以 $a \neq 0$

$$E_{32}E_{31}E_{21}A = U \Rightarrow$$

$$L = E_{21}^{-1}E_{31}^{-1}E_{32}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ b & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & b/a & 1 \end{bmatrix}$$

b可以是任意常数。

作者：我是8位的

出处: <http://www.cnblogs.com/bigmonkey>

本文以学习、研究和分享为主，如需转载，请联系本人，标明作者和出处，非商业用途！

扫描二维码关注公众号“我是8位的”



随笔

分类: [线性代数](#)

标签: [LU分解](#), [LU分解的意义](#)

[好文要顶](#)[关注我](#)[收藏该文](#)

我是8位的
关注 - 5
粉丝 - 288
[+加关注](#)

4

推荐

0

反对

« 上一篇: [线性代数笔记9——消元矩阵与置换矩阵](#)
» 下一篇: [线性代数笔记11——向量空间](#)

posted on 2018-08-29 18:21 我是8位的 阅读(20363) 评论(1) 编辑 收藏 举报

[刷新评论](#) [刷新页面](#) [返回顶部](#)

登录后才能查看或发表评论，立即 [登录](#) 或者 [逛逛](#) 博客园首页

【推荐】华为 HWD 2022 故事征集，分享最打动你的科技女性故事
【推荐】华为开发者专区，与开发者一起构建万物互联的智能世界

WebForms升级到ASP.NET Core
fineui.com

- 编辑推荐:
- 革命性创新，动画杀手铜 @scroll-timeline
 - 戏说领域驱动设计（十二）—— 服务
 - ASP.NET Core 6框架揭秘实例演示[16]：内存缓存与分布式缓存的使用
 - .Net Core 中无处不在的 Async/Await 是如何提升性能的？
 - 分布式系统改造方案 —— 老旧系统改造篇



最新新闻:

- 乔布斯的创业搭档：他缺乏工程师才能，不得不锻炼营销能力来弥补
 - 美国大厂码农薪资曝光：年薪18万美元，够养家，不够买海景房
 - 两张照片就能转视频！Google提出FLIM帧插值模型
 - Android 再推 “杀手级” 功能，可回收 60% 存储空间
 - 溺在理财暴雷潮的投资人：本金63万，月兑25元不够卖菜
- » 更多新闻...

Powered by:

博客园

Copyright © 2022 我是8位的

Powered by .NET 6 on Kubernetes