博客园 cnblogs.com 首页 新闻 博问 专区 闪存 班级 代码改变世界 注册 登录

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 XML
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

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28——复矩阵和快速傅立叶变换

原文 | https://mp.weixin.qq.com/s/YzPoPnRb-gEm_EiV9et0TA

实矩阵也可能碰到复特征值，因此无可避免地在矩阵运算中碰到复数。

矩阵当然也有可能包含复数，最重要的复矩阵是傅立叶矩阵，它用于傅立叶变换。一种特殊的傅立叶变换是快速傅立叶变换（fast Fourier transform），简称FFT，在计算机中很常用，特别是涉及到大数据时，FFT将把傅立叶变换中的n阶方正阵乘法的运算次数从n^2降低到$n\log_2 n$，这是一个巨大的进步。

本文相关前置知识

复数和复平面、复平面上的旋转

傅立叶矩阵中w的由来

标准正交矩阵及其性质

复矩阵的特征值

复向量

先给出一个复向量，即向量的分量中至少有一个是复数：

$$z=\begin{bmatrix}z_1\\z_2\\\vdots\\z_n\end{bmatrix},\qquad z\in C^n$$

虽然这个向量在表达上和普通的实向量没什么区别，但这个向量不再属于实空间R^n，而是属于复空间C^n，即n维复空间。

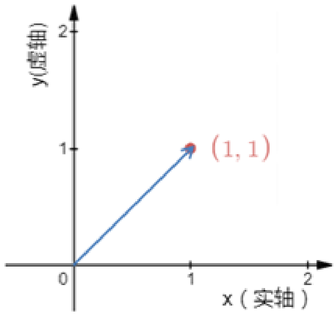
模长

关于复向量的第一个问题是模长怎么计算?

由于向量中有复数分量，再用过$\mathbf{z}^T\mathbf{z}$的方式是无法计算出模长的，比如下面的(1, i)：

$$\begin{bmatrix}1\\i\end{bmatrix}\begin{bmatrix}1 & i\end{bmatrix}=1^2+i^2=0$$

但很明显，(1, i)在复平面上的模长不是0：



我们知道一个复数的模长的平方等于这个复数与它的共轭复数的乘积，因此可以通过下面的方式计算复向量的模长：

$$|\boldsymbol{z}|^2 = |z_1|^2 + |z_2|^2 + \cdots + |z_n|^2 = \bar{z}_1 z_1 + \bar{z}_2 z_2 + \cdots + \bar{z}_n z_n = \bar{\boldsymbol{z}}^{\mathrm{T}} \boldsymbol{z}$$

$$|(1,\mathrm{i})|^2 = \begin{bmatrix} 1 & -\mathrm{i} \end{bmatrix} \begin{bmatrix} 1 \\ \mathrm{i} \end{bmatrix} = 2, \qquad |(1,\mathrm{i})| = \sqrt{2}$$

通常用$\mathbf{z}^{\mathrm{H}}$（H来自Hermite）表示共轭向量的转置：

$$\boldsymbol{z}^{\mathrm{H}} = \bar{\boldsymbol{z}}^{\mathrm{T}}, \qquad |\boldsymbol{z}|^2 = \boldsymbol{z}^{\mathrm{H}} \boldsymbol{z}$$

点积

与模长类似，如果有两个复向量**p**和**q**，它们的点积也不能简单地定义成$\mathbf{p}^{\mathrm{T}}\mathbf{q}$，而是$\mathbf{p}^{\mathrm{H}}\mathbf{q}$：

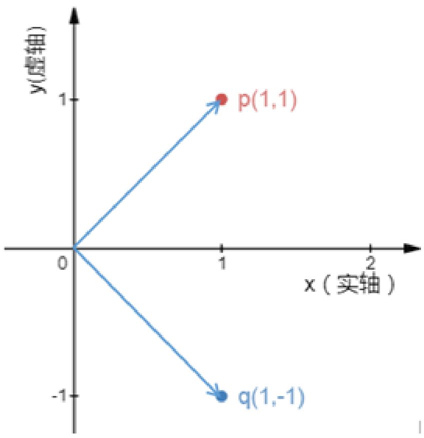
$$\boldsymbol{p}^{\mathrm{H}} \boldsymbol{q} = \bar{\boldsymbol{p}}^{\mathrm{T}} \boldsymbol{q} = \bar{p}_1 q_1 + \bar{p}_2 q_2 + \cdots + \bar{p}_n q_n$$

复平面上有两个向量p(1, i)和q(1, -i)，二者的点积是：

$$\boldsymbol{p} = \begin{bmatrix} 1 & \mathrm{i} \end{bmatrix}, \qquad \bar{\boldsymbol{p}} = \begin{bmatrix} 1 \\ -\mathrm{i} \end{bmatrix}$$

$$\boldsymbol{p}^{\mathrm{H}} \boldsymbol{q} = \bar{\boldsymbol{p}}^{\mathrm{T}} \boldsymbol{q} = \begin{bmatrix} 1 & -\mathrm{i} \end{bmatrix} \begin{bmatrix} 1 \\ -\mathrm{i} \end{bmatrix} = 1 + (-\mathrm{i})^2 = 0$$

二者的点积是0，因此可以判断两个复向量互相垂直：



复矩阵

我们曾讲过，对于一个矩阵**A**来说，如果$\mathbf{A}^{\mathrm{T}}=\mathbf{A}$，那么**A**是对称矩阵，实际上这个结论仅对实矩阵有效，对复矩阵可不管用。

厄米特矩阵

如果一个复矩阵是对称矩阵，那么：

$$\bar{\boldsymbol{A}}^{\mathrm{T}} = \boldsymbol{A}$$

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轴建立座标系
--trustInU

通常写作另一种方式：

$$A^{\mathrm{H}} = A$$

这种对称复矩阵称为厄米特矩阵（或埃米特矩阵，Hermitian matrix），比如下面这个：

$$\bar{A}^{\mathrm{T}} = A = \begin{bmatrix} 2 & 3+\mathrm{i} \\ 3-\mathrm{i} & 5 \end{bmatrix}$$

酉矩阵

上一章讲到，一个复对称矩阵的特征值仍然是实数，且可以找到互相垂直的特征向量，其对角元素都是实数。

假设有一个由n个标准正交向量组成的复矩阵$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n]$，这里的正交意味：

$$\boldsymbol{q}_i^H \boldsymbol{q}_j = \bar{\boldsymbol{q}}_i^T \boldsymbol{q}_j = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$$

$$\boldsymbol{Q}^H \boldsymbol{Q} = \bar{\boldsymbol{Q}}^T \boldsymbol{Q} = \boldsymbol{I}$$

这个复空间的正交矩阵Q称为酉矩阵（unitary matrix）。

傅立叶矩阵和快速傅立叶变换

傅立叶变换是一种分析信号的方法，它可分析信号的成分，也可用这些成分合成信号。许多波形可作为信号的成分，比如正弦波、方波、锯齿波等，傅立叶变换用正弦波作为信号的成分。

在电子工程或计算机中，n×n矩阵的行和列都是从0开始的，到n-1结束，由于傅立叶变换经常用在计算机上，所以我们在讨论傅立叶矩阵的时候遵从这种下标规则。

傅立叶矩阵

型如F_n的复矩阵是傅立叶矩阵：

$$F_n = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{n-1} \\ 1 & w^2 & w^4 & \cdots & w^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w^{n-1} & w^{2(n-1)} & \cdots & w^{(n-1)^2} \end{bmatrix}$$

$$(F_n)_{jk} = w^{j\times k}, \qquad j,k = 0,1,2,\cdots,n-1$$

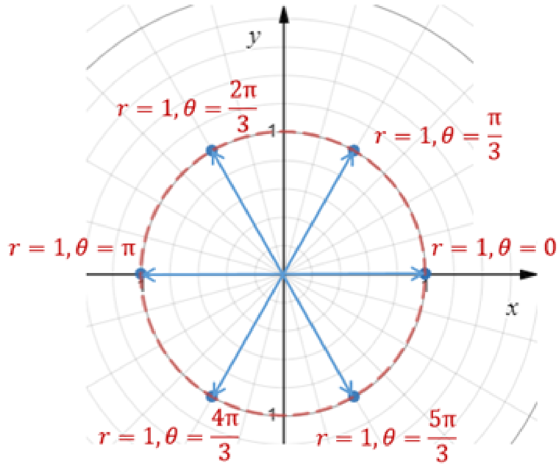
矩阵中的每个元素都不为0，是个全矩阵。w是个特殊的值：

$$w = \cos\left(\frac{2\pi}{n}\right) + \mathrm{i}\sin\left(\frac{2\pi}{n}\right) = \mathrm{e}^{\frac{\mathrm{i}2\pi}{n}}, \qquad w^n = 1$$

相关链接：傅立叶矩阵中w的由来

复数和复平面、复平面上的旋转

在计算w的乘方的时候，我们需要考虑用极坐标表示复平面。在极坐标下，w表示模长为1的向量从（1，0）开始，绕原点逆时针旋转了2π/n，如此一来，我们就可以知道n等于任意值时w的位置，并且也同样知道w的乘方的位置。对于w来说，w^n的模长仍然等于1，只是旋转的角度有所不同。比如n=6时，$w=e^{i2\pi/6}= e^{i\pi/3}$，$w^2=(e^{i\pi/3})^2= e^{i2\pi/3}$。



同理，n=4时，w=$e^{i2\pi/4}$，正好落在虚轴上，w= i，w^4=1。我们写出4阶傅立叶矩阵：

$$F_4=\begin{bmatrix}1&1&1&1\\1&w&w^2&w^3\\1&w^2&w^4&w^6\\1&w^3&w^6&w^9\end{bmatrix}=\begin{bmatrix}1&1&1&1\\1&i&i^2&i^3\\1&i^2&i^4&i^6\\1&i^3&i^6&i^9\end{bmatrix}=\begin{bmatrix}1&1&1&1\\1&i&-1&-i\\1&-1&1&-1\\1&-i&-1&i\end{bmatrix}$$

傅立叶矩阵可以得到一个四点（离散的）傅立叶变换，它的逆矩阵可以得到傅立叶逆变换。此外，傅立叶矩阵的列向量是正交的，所以很容易求得逆矩阵。实际上傅立叶矩阵可以分解成一系列稀疏矩阵，这些矩阵有大量的0元素，所以相应的乘法和求逆都很简单。

F_4的列向量正交，这意味着任意两个列向量的点积为0，但如果你还是用过去的点积计算方法就会发现它并不是0（当然有时候会凑巧等于0），比如第2列和第4列：

$$\begin{bmatrix}1\\i\\-1\\-i\end{bmatrix}^T\begin{bmatrix}1\\-i\\-1\\i\end{bmatrix}=1+1+1+1=4$$

前面介绍过，复向量的点积不是这么算的，正确算法应该是取共轭的转置：

$$\begin{bmatrix}1\\i\\-1\\-i\end{bmatrix}^H\begin{bmatrix}1\\-i\\-1\\i\end{bmatrix}=\begin{bmatrix}1\\-i\\-1\\i\end{bmatrix}^T\begin{bmatrix}1\\-i\\-1\\i\end{bmatrix}=1-1+1-1=0$$

F_4的列向量的模长是2，为了使矩阵更完美，可以把它除以2，于是矩阵的各列就变成了标准正交向量：

$$F_4=\frac{1}{2}\begin{bmatrix}1&1&1&1\\1&i&-1&-i\\1&-1&1&-1\\1&-i&-1&i\end{bmatrix}$$

对于标准正交的实矩阵来说，矩阵的逆等于矩阵的转置，傅立叶矩阵可化简为标准正交的复矩阵，具有同样的性质，F_4的逆矩阵就是它共轭的转置：

$$F_4^HF_4=\boldsymbol{I},\qquad F_4^{-1}=F_4^H$$

由于F_4的逆矩阵就是F_4共轭的转置，所以F_4的逆矩阵和F_4具有同样的性质。

快速傅立叶变换

什么是快速傅立叶变换呢？举个例子，F_6与F_3之间存在着某种奇妙的联系，F_8与F_4，F_{64}与F_{32}也一样，我们可以把这种联系描述出来。

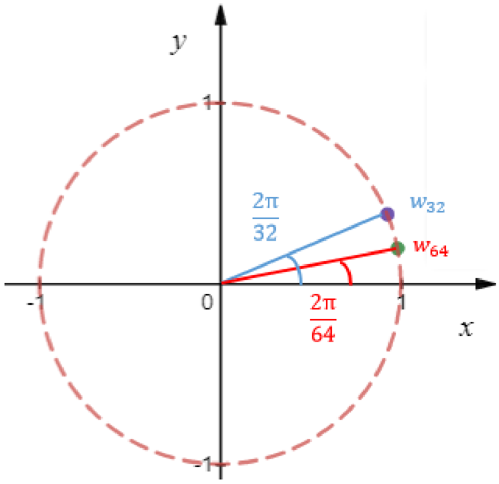
以F_{64}与F_{32}为例，F_{64}是一个64阶方阵，$w^{64} = 1$，w = 1；同理对于F_{32}来说，$w^{32} = 1$，w = 1。

$$F_{64} \to w_{64} = e^{\frac{i2\pi}{64}}, \qquad (w_{64})^{64} = 1$$

$$F_{32} \to w_{32} = e^{\frac{i2\pi}{32}}, \qquad (w_{32})^{32} = 1$$

$$(w_{64})^2 = w_{32}$$

w_{64}和w_{32}的模长相等，w_{64}的幅角是w_{32}的2倍：



既然如此，F_{64}和F_{32}也应该存在某种联系。实际上F_{64}与由两个F_{32}和两个零矩阵构成的方阵有关：

$$F_{2n} = \begin{bmatrix} I & D \\ I & -D \end{bmatrix} \begin{bmatrix} F_n & 0 \\ 0 & F_n \end{bmatrix} P$$

这种分解称为快速傅立叶变换。其中P是一个2n×2n的置换矩阵，D是由w的幂构成的对角矩阵：

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ & & & \vdots & & & \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ & & & \vdots & & & \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}, \qquad D = \begin{bmatrix} 1 & & & & \\ & w & & & \\ & & w^2 & & \\ & & & \ddots & \\ & & & & w^{n-1} \end{bmatrix}$$

P的效果是使得所乘行向量**x**中序号为奇数的分量x_1, x_3, x_5......提到前面，偶数序号的分量x_2, x_3, x_6......放到后面。例如：

$$\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_8 \end{bmatrix}, \qquad P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{x}P = \begin{bmatrix} x_1 & x_3 & x_5 & x_7 & x_2 & x_6 & x_6 & x_8 \end{bmatrix}$$

可以看到，快速傅立叶变换实际上使用的是分治算法。计算64阶傅立叶变换的计算量是64^2，而经过一次变换后，计算量变成了2×32^2（2个32阶的傅立叶矩阵）再加上一些修正项，而修正项主要来自于和对角矩阵D的乘法，共32次。继续对F_{32}进行分解……知道矩阵尺度为1。对于n阶矩阵，可将n^2次计算降至$(n/2)\ \log_2 n$。

作者：我是8位的

出处：https://mp.weixin.qq.com/s/YzPoPnRb-gEm_EiV9et0TA

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！

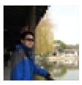
扫描二维码关注作者公众号“我是8位的”

随笔

分类: 线性代数

标签: 复矩阵, 快速傅里叶变换

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

0 推荐 0 反对

« 上一篇： 闲话复数（2）——欧拉公式
» 下一篇： 线性代数笔记29——正定矩阵和最小值

posted on 2019-11-26 17:09 我是8位的 阅读(1797) 评论(0) 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

广告

JAVA Office文档在线编辑APIs

简单易用的Word，Excel，PowerPoint在线编辑接口

cloud.e-iceblue.cn

打开

编辑推荐：

· 革命性创新，动画杀手锏 @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...

历史上的今天：

2018-11-26 寻找“最好” （4）——不等约束和KKT条件

Powered by:
博客园
Copyright © 2022 我是8位的
Powered by .NET 6 on Kubernetes