线性代数回顾

本节来重新回顾一下线性代数的相关内容。

矩阵

矩阵是以矩形排列的数组，例如：

$$A = \begin{bmatrix} 1 & 3 & 4 & 2 \\ 3 & 5 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 3 \\ 3 & 5 \\ 4 & 2 \end{bmatrix}$$

矩阵也是二维数组的一种表现形式。一般用：行数×列数 表示矩阵的维度。例如上文A是一个2×4的矩阵，而B是一个3×2的矩阵。

有时会使用$\mathbb{R}^{i \times j}$表示维度为$i \times j$的全体矩阵的集合（注意同实数集区分）

引用矩阵中的元素，一般使用$A_{i,j}$表示矩阵A的第i行第j列的元素，比如上文中$A_{1,4} = 2$，$A_{2,3} = 0$。

向量

向量是一类维度固定为$n \times 1$（n为任意正数）的矩阵。例如：

$$y = \begin{bmatrix} 3 \\ 4 \\ 5 \\ 9 \end{bmatrix}$$

向量中元素使用y_i引用$y_{i,1}$的元素，例如$y_3 = 5$。

对于编程语言来说，对向量中元素的引用分为1-索引与0-索引两种机制，即向量y第一个元素是y_1还是y_0的区别。例如`Python`是0-索引而`Mathematica`是1-索引。

（一个书写习惯：一般使用A，B这样的大写字母表示矩阵，a，b表示数字，x，y表示向量，当然不一定要时时刻刻遵守，不过这种习惯为大部分人所接受）

矩阵运算

矩阵的加法

矩阵加法是两个**同维度**的矩阵间做的运算。例如下面两个矩阵：

$$A=\begin{bmatrix}0&2\\1&0\\0&3\end{bmatrix}\qquad B=\begin{bmatrix}1&3\\3&5\\4&2\end{bmatrix}$$

定义$A+B$为A，B对应位置元素相加的矩阵，即：

$$A+B=\begin{bmatrix}0+1&2+3\\1+3&0+5\\0+4&3+2\end{bmatrix}=\begin{bmatrix}1&5\\4&5\\4&5\end{bmatrix}$$

对于维度不相同的矩阵，**不给出**相加的定义。

标量乘法

例如下面这个矩阵：

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

定义矩阵标量乘法$a\times A$为矩阵A全部元素乘以a的矩阵，取$a=2$，于是：

$$a\times A=\begin{bmatrix}0\times 2&2\times 2\\1\times 2&0\times 2\\0\times 2&3\times 2\end{bmatrix}=\begin{bmatrix}0&4\\2&0\\0&6\end{bmatrix}$$

标量除法A/a可以定义为$A\times\frac{1}{a}$。

矩阵乘法

两个矩阵$A^{n\times m}$，$B^{m\times s}$（注意A的列数m同B的行数相同）的乘积被定义为一个矩阵$C^{n\times s}$，其中元素如下：

$$C_{i,j}=\sum_{a,b=1,1}^{m,m}A_{i,a}B_{b,j}=A_{i,1}B_{1,j}+A_{i,2}B_{2,j}+\ldots A_{i,m}B_{m,j}$$

这么写可能显得有点抽象，一个简单的记忆方法是C中第i行第j列的元素是A的第i行行向量与B的第j列列向量做向量点乘的结果。（这也解释了A的列数为何要同B的行数相同）

简单使用一个例子，下面是两个矩阵

$$A=\begin{bmatrix}1&3\\3&5\\4&2\end{bmatrix}\qquad B=\begin{bmatrix}1&3&4&2\\3&5&0&1\end{bmatrix}$$

于是计算$A\times B$为C，并举例$C_{2,3}$计算过程：

$$C=\begin{bmatrix}10&18&4&5\\18&34&12&11\\10&22&16&10\end{bmatrix}$$

$$C_{2,3}=A_{2,1}B_{1,3}+A_{2,2}B_{2,3}=3\times 4+5\times 0=12$$

对于实际问题，我们往往也不需要人手动计算过于繁杂的矩阵乘法，`Python`中的`numpy.dot`，`Mathematica`的`Dot`函数都可以帮我们快速完成运算。

矩阵乘法性质

> - $A \times B \neq B \times A$（这条是指大部分时间矩阵乘法是不可交换的）
> - $A \times (B \times C) = (A \times B) \times C$（矩阵乘法总是可结合的）
> - n**阶单位矩阵**定义为下面这样一个矩阵：
>
> $$I^{n \times n} = \overbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}}^{n\text{个元素}} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} (\text{另一种写法})$$

矩阵的逆

若A是一个$m \times m$的方阵，那么定义A的逆A^{-1}由如下性质得出：

$$A^{-1}A = AA^{-1} = I^{m \times m}$$

对于**逆矩阵**的计算公式可以参考这篇博客逆矩阵计算或者是线性代数的课本，不过现在使用软件计算显然是更为简单快捷的方法，在这里给出`Python`与`Mathematica`中计算矩阵`A`的逆的方法：

- `Python`中使用函数`numpy.linalg.inv(A)`。
- `Mathematica`中使用函数`Inverse[A]`。

奇异矩阵的内容请自行翻阅线性代数课本中相关章节，在本课程中不对是否存在逆矩阵做过多讨论。

矩阵转置

一个矩阵A的转置矩阵A^T中的元素应该有下述特点：

$$A^T_{i,j} = A_{j,i}$$

通俗来说就是将元素以对角线对称过去的新矩阵，例如：

$$A = \begin{bmatrix} 1 & 6 \\ 3 & 5 \\ 4 & 2 \end{bmatrix} \iff A^T = \begin{bmatrix} 1 & 3 & 4 \\ 6 & 5 & 2 \end{bmatrix}$$

同样，在这里给出`Python`与`Mathematica`中计算矩阵`A`的转置的方法：

- `Python`中使用函数`A.transpose()`或者`A.T`（注意A是`numpy.array`对象）均可。
- `Mathematica`中使用函数`Transpose[A]`或者`A` `esc`+tr+`esc`（打出一个T的上标）均可。