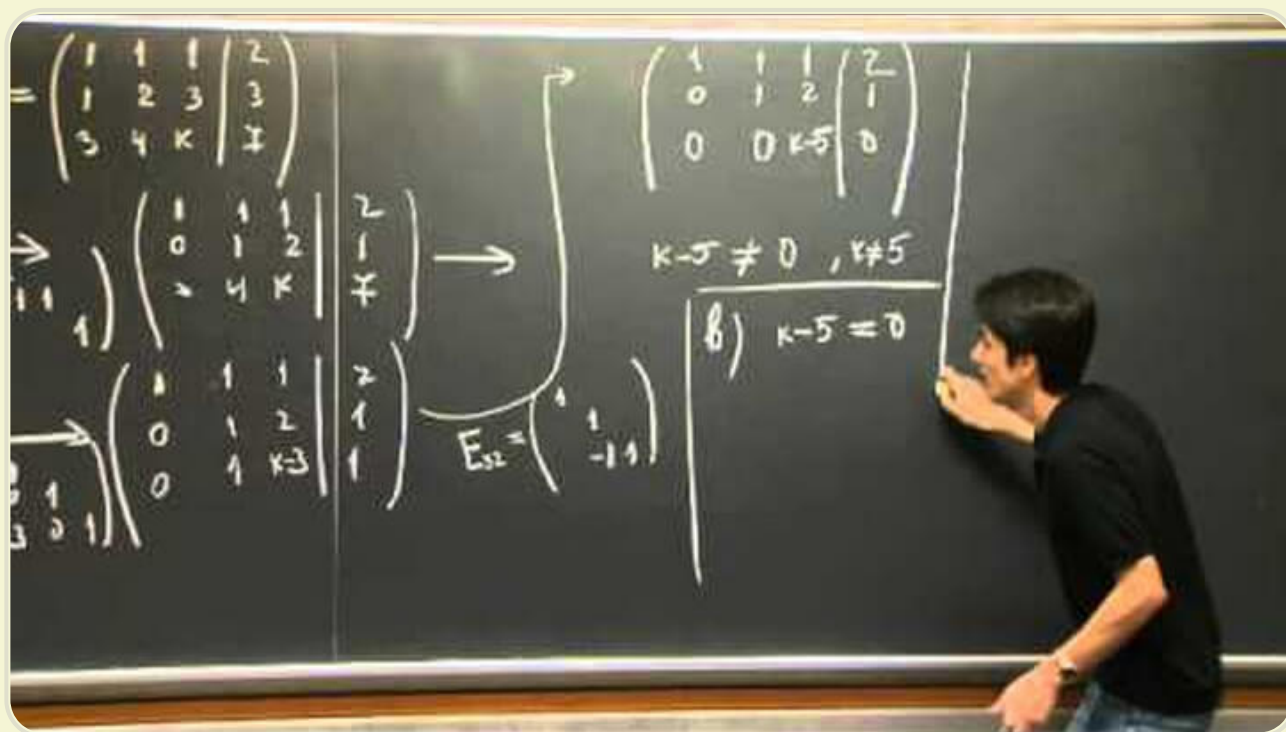


理解矩阵乘法

作者： 阮一峰

日期： 2015年9月 1日

大多数人在高中，或者大学低年级，都上过一门课《线性代数》。这门课其实是教矩阵。



刚学的时候，还蛮简单的，矩阵加法就是相同位置的数字加一下。

$$\begin{pmatrix} 2 & 1 \\ 4 & 3 \end{pmatrix} + \begin{pmatrix} 1 & 2 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 3 & 3 \\ 5 & 3 \end{pmatrix}$$

矩阵减法也类似。

矩阵乘以一个常数，就是所有位置都乘以这个数。

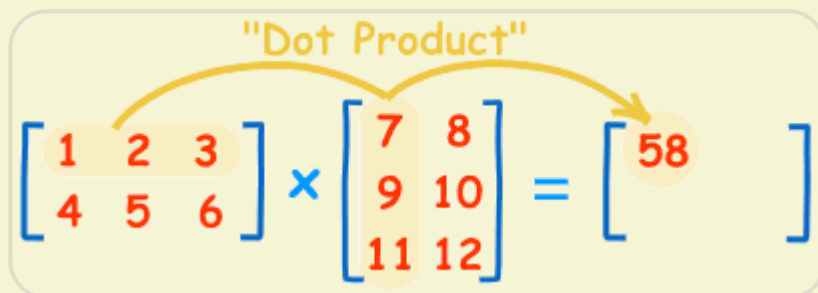
$$2 \times \begin{pmatrix} 2 & 1 \\ 4 & 3 \end{pmatrix} = \begin{pmatrix} 4 & 2 \\ 8 & 6 \end{pmatrix}$$

但是，等到矩阵乘以矩阵的时候，一切就不一样了。

$$\begin{pmatrix} 2 & 1 \\ 4 & 3 \end{pmatrix} \times \begin{pmatrix} 1 & 2 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 3 & 4 \\ 7 & 8 \end{pmatrix}$$

这个结果是怎么算出来的？

教科书告诉你，计算规则是，第一个矩阵第一行的每个数字（2和1），各自乘以第二个矩阵第一列对应位置的数字（1和1），然后将乘积相加（ $2 \times 1 + 1 \times 1$ ），得到结果矩阵左上角的那个值3。



$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & \\ & \end{bmatrix}$$

也就是说，结果矩阵第m行与第n列交叉位置的那个值，等于第一个矩阵第m行与第二个矩阵第n列，对应位置的每个值的乘积之和。

怎么会有这么奇怪的规则？

我一直没理解这个规则的含义，导致《线性代数》这门课就没学懂。研究生时发现，线性代数是向量计算的基础，很多重要的数学模型都要用到向量计算，所以我做不了复杂模型。这一直让我有点伤心。

前些日子，受到[一篇文章](#)的启发，我终于想通了，矩阵乘法到底是什么东西。关键就是一句话，矩阵的本质就是线性方程式，两者是一一对应关系。如果从线

性方程式的角度，理解矩阵乘法就毫无难度。

下面是一组线性方程式。

$$\begin{cases} 2x + y = 3 \\ 4x + 3y = 7 \end{cases}$$

矩阵的最初目的，只是为线性方程组提供一个简写形式。

$$\begin{pmatrix} 2 & 1 \\ 4 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 3 \\ 7 \end{pmatrix}$$

老实说，从上面这种写法，已经能看出矩阵乘法的规则了：系数矩阵第一行的2和1，各自与 x 和 y 的乘积之和，等于3。不过，这不算严格的证明，只是线性方程式转为矩阵的书写规则。

下面才是严格的证明。有三组未知数 x、y 和 t，其中 x 和 y 的关系如下。

$$\begin{cases} a_{11}x_1 + a_{12}x_2 = y_1 \\ a_{21}x_1 + a_{22}x_2 = y_2 \end{cases}$$

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

x 和 t 的关系如下。

$$\begin{cases} b_{11}t_1 + b_{12}t_2 = x_1 \\ b_{21}t_1 + b_{22}t_2 = x_2 \end{cases}$$

$$\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

有了这两组方程式，就可以求 y 和 t 的关系。从矩阵来看，很显然，只要把第二个矩阵代入第一个矩阵即可。

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

从方程式来看，也可以把第二个方程组代入第一个方程组。

$$\begin{cases} a_{11}(b_{11}t_1 + b_{12}t_2) + a_{12}(b_{21}t_1 + b_{22}t_2) = y_1 \\ a_{21}(b_{11}t_1 + b_{12}t_2) + a_{22}(b_{21}t_1 + b_{22}t_2) = y_2 \end{cases}$$

上面的方程组可以整理成下面的形式。

$$\begin{cases} (a_{11}b_{11} + a_{12}b_{21})t_1 + (a_{11}b_{12} + a_{12}b_{22})t_2 = y_1 \\ (a_{21}b_{11} + a_{22}b_{21})t_1 + (a_{21}b_{12} + a_{22}b_{22})t_2 = y_2 \end{cases}$$

$$\begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

最后那个矩阵等式，与前面的矩阵等式一对照，就会得到下面的关系。

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

矩阵乘法的计算规则，从而得到证明。

=====

以下为广告部分。欢迎大家在我的网络日志[投放广告](#)。

【赞助商广告】

如果你想换工作，花五分钟，浏览一下，也许人生从此就会不同。



的目标是，只要你有2年以上的互联网工作经验（一线互联网公司更好），它就极有可能帮你拿到年薪 20W--80W 的 Offer。



它最近对514位用户的一份实证分析，有两点发现。

(1) 所有离职的工程师之中，BAT员工最受欢迎，新工作的薪酬通常可以翻番。

(2) 只有40%的程序员跳槽去了C轮、D轮和上市公司，其余60%都去了天使轮、A轮和B轮的公司，其中不乏资深的高级程序员。原因是这些创业公司刚刚起步、产品模式初步得到认可，上升空间大、技术人才需求量大、愿意给出较多的期权。



结合上面两点，年青程序员可以这样设计自己的职业生涯：

1、至少有一次大公司的成功工作经历。

所谓"成功"，是指参与过知名的产品开发或主力项目经验。这一般代表你已经具备：良好的代码规范、团队协作能力、与大牛一起工作的开阔眼界成熟的技术体系。

2、然后，加入一家高速发展的创业公司。

帮助你实现这条职业道路。

(完)

文档信息

- 版权声明：自由转载-非商用-非衍生-保持署名（创意共享3.0许可证）
- 发表日期：2015年9月 1日

相关文章

■ 2021.09.22: [俄罗斯总理的几何题](#)

9月1日是俄罗斯的知识节，因为这一天是各级学校的开学日，象征进入知识宝库的日子。

■ 2019.01.28: [Prolog 语言入门教程](#)

Prolog 是一种与众不同的语言，不用来开发软件，专门解决逻辑问题。比如，"苏格拉底是人，人都会死，所以苏格拉底会死"这一类的问题。

■ 2018.09.05: [哈希碰撞与生日攻击](#)

一、哈希碰撞是什么？所谓哈希（hash），就是将不同的输入映射成独一无二的、固定长度的值（又称"哈希值"）。它是最常见的软件运算之一。

■ 2017.12.13: [图像与滤波](#)

我对图像处理一直很感兴趣，曾经写过好几篇博客（1，2，3，4）。



Weibo | Twitter | GitHub

Email: yifeng.ruan@gmail.com