

# King of Dumbs

笨蛋之王

级别: ♦ 进阶

## 电脑怎样把照片变成数字？

一张照片就是几千万个数，人工智能对它做的几乎全是乘法和加法。分三个层次讲向量、点积和矩阵，配两个互动小工具。

中文 发布于 2026年10月17日



扫码打开可互动的课程

<https://kingofdumbs.com/zh/lessons/vectors-and-matrices>

## 你将学到

- 照片就是一列数：向量和维数
- 点积：乘一乘再加起来，以及两支箭头有多同向
- 矩阵会挪动平面：它的列、面积和剩下的维数
- 矩阵乘法：把变换串起来，为什么顺序要紧，以及神经网络在做什么

### 本课会出现的符号

$E_a$  下标：右下角的标签 (♠) ·  $\Sigma$  求和：全部加起来 (♠) ·  $x_i$  下标  $i$ ：第  $i$  个元素 (♠) ·  $\approx$  约等于 (♠) ·  $\leq$  小于等于 / 大于等于 (♠) ·  $|x|$  绝对值：去掉正负号 (♠) ·  $\cdot$  点：乘（或点积） (♠♠) ·  $\sigma$  sigma：含义因课而异 (♠) ·  $\theta$  theta：通常是角度 (♠) ·  $\lambda$  lambda：一个缩放用的数 (♠) ·  $\|x\|$  范数：箭头的长度 (♠) ·  $x^T$  转置：把列放倒 (♠) ·  $\in$  属于 (♠) ·  $\mathbb{R}$   $\mathbb{Z}$  数集 (♠)

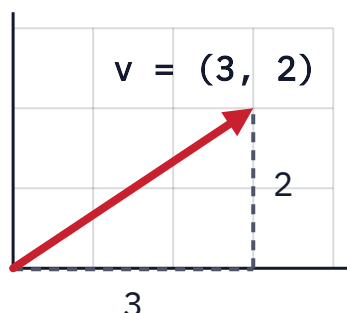
你给猫拍了张照片，在手机相册里搜”猫”，很多时候就能把它找出来。可电脑并看不见猫：它唯一会做的，就是飞快地加数和乘数。

这一课的傻问题是：**电脑怎样把照片变成数字？** 顺便再问一句：变成数字以后，它拿这些数做什么？答案就是三个听起来像考试题的概念：向量、点积和矩阵；再加上一道算术：乘一乘，再加起来。

选一张牌：♣ 从零开始，♦ 不怕简单的公式，♠ 想看符号。随时可以换。



## 照片就是一列数



长度  $\sqrt{(3^2 + 2^2)} \approx 3.6$

2 个数：平面

3 个数：空间

3600 万个数：画不出来，  
但算法一样

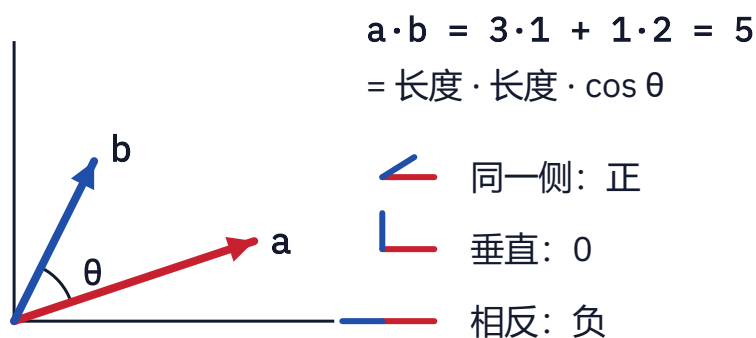
**向量**是一列有顺序的数，其中每个数叫它的**分量**。只有两个分量时，可以画成从中心出发的一支箭头： $v = (3, 2)$  就是”向右 3，向上 2”。它的**长度**用勾股定理算： $\sqrt{(3^2 + 2^2)} = \sqrt{13} \approx 3.6$  (OpenStax) 。

一张只有两个像素的灰度照片，就是平面上这样一支箭头；三个像素，就是空间里的一支箭头。再多就画不出来了，但算法一模一样：长度还是各分量平方和的平方根。**维数**就是向量里有多少个数。

彩色照片里，每个像素有三个 0 到 255 之间的数，每个**通道**（红、绿、蓝）一个，因为每个数占 8 位（Pillow，一个 Python 图像库）。所以打开的照片是一张 **高 × 宽 × 3** 的数表。JPEG 文件之所以小，是因为压缩过，但打开时会解压成这张表。要交给神经网络，就**把它全部排成一行**： $4000 \times 3000 \times 3 = 3600$  万个数。



## 点积：乘一乘，再加起来



$a = (a_1, a_2)$  和  $b = (b_1, b_2)$  的**点积**是  $a \cdot b = a_1 b_1 + a_2 b_2$ ：把分量逐个相乘，再加起来（OpenStax）。分量更多也一样。 $a = (3, 1)$ ,  $b = (1, 2)$ ： $3 \cdot 1 + 1 \cdot 2 = 5$ 。两张各有 3600 万个数的照片，就是 **3600 万次乘法**。

用箭头来看还有另一种读法： $a \cdot b = \mathbf{a}$  的长度  $\times$   $\mathbf{b}$  的长度  $\times$  **夹角的余弦**（OpenStax）。**同向时余弦为 1，成直角时为 0，相反时为 -1**。所以两支**垂直**的箭头，点积为 0。

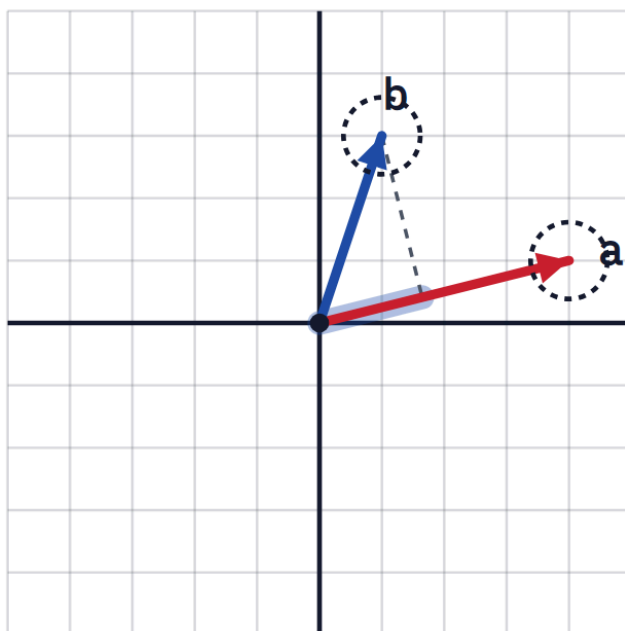
把点积除以两个长度，就只剩下余弦，它衡量两个方向有多像：这就是**余弦相似度**。拿照片来说有个妙处。把照片里的每个数都乘以 1.2，照片看起来更亮：**箭头变长了，但没有转动**，它和原图的余弦相似度仍然是 1（自行计算，前提是没有哪个数超过 255）。

### 笨蛋之王说：

乘一乘，加一加，就这么多。靠这个我能比价格、比箭头、比照片，还剩时间睡个午觉。

## 动手试试：两支箭头

拖动箭头的尖端，或者用滑块、键盘方向键来移动。找一个让点积等于 0 的位置：这时两支箭头互成直角。蓝色宽带是  $b$  在  $a$  上的影子：两支箭头越同向，影子越长。



a, 横向

4 a, 纵向  1

b, 横向

1 b, 纵向  3

点积  $a \cdot b$   $4 \cdot 1 + 1 \cdot 3 = 7$

长度  $\|a\| = 4.12 \cdot \|b\| = 3.16$

夹角  $57.5^\circ$

余弦（方向有多像） 0.54

$a = (4, 1)$ ,  $b = (1, 3)$ ,  $a \cdot b = 7$ 。指向同一侧：点积为正。



互动小工具。在网站上试一试：<https://kingofdumbs.com/zh/lessons/vectors-and-matrices>

## 矩阵会挪动平面

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

第 1 行:  $2 \cdot 1 + 1 \cdot 2 = 4$

第 2 行:  $0 \cdot 1 + 1 \cdot 2 = 2$



**矩阵**是一张长方形的数表；有  $m$  行  $n$  列的矩阵叫  $m \times n$  矩阵（OpenStax）。矩阵乘向量，就是用矩阵的**每一行**和这个向量做点积。 $A = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}$ ,  $x = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ ：第一行得  $2 \cdot 1 + 1 \cdot 2 = 4$ ，第二行得  $0 \cdot 1 + 1 \cdot 2 = 2$ ，所以  $Ax = \begin{pmatrix} 4 \\ 2 \end{pmatrix}$ 。

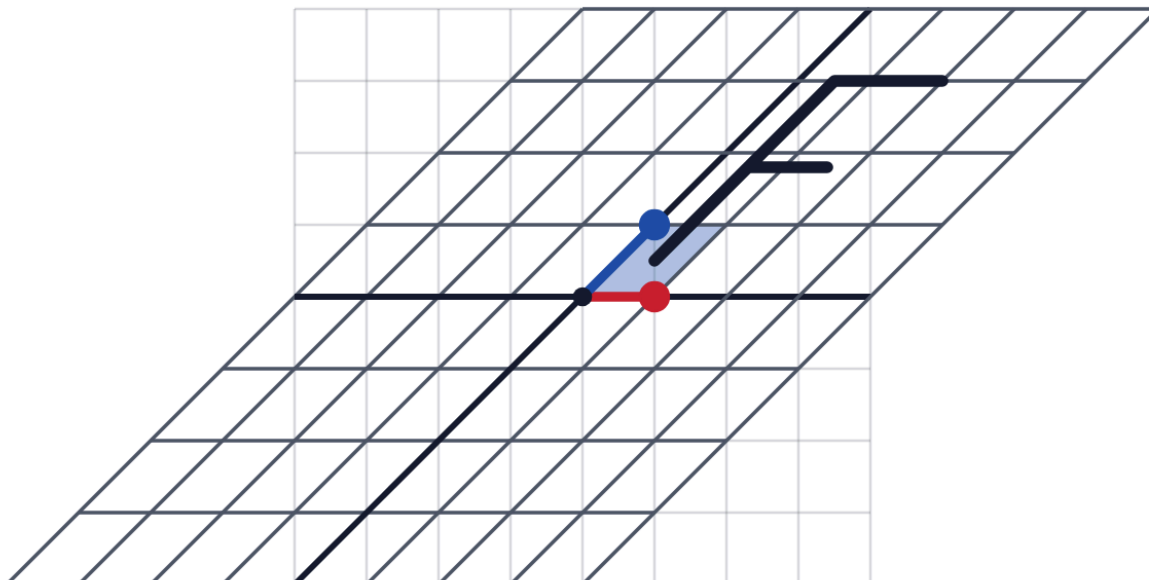
读矩阵有个窍门：**第一列是 (1, 0) 去的地方，第二列是 (0, 1) 去的地方**（Austin）。在这个例子里， $(1, 0)$  变成  $(2, 0)$ ， $(0, 1)$  变成  $(1, 1)$ ：矩阵把图形横向拉长，又**推斜**了。知道了这两点，就知道一切点去哪里，因为**任何一个点都是这两者的组合**。

边长为 1 的小方块变成一个平行四边形。矩阵为  $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$  时，它的面积是  $ad - bc$ ：在例子里是  $2 \cdot 1 - 1 \cdot 0 = 2$ ，变成两倍（Austin）。如果等于 0，矩阵就把整个平面**压扁**成一条直线：少了一个维度，而且回不去了。 $\begin{pmatrix} 1 & -1 \\ 2 & -2 \end{pmatrix}$  就是这样。如果是负数，图形会**像照镜子一样翻过来**。

Pillow 就是这样**旋转**或**推斜**照片的：对每个新像素，它用一个矩阵算出**它来自原图的哪个点**（再加上一个平移）。

## 动手试试：让网格变形

调一调矩阵的四个数，或者试试下面的按钮。红色箭头是第一列，即  $(1, 0)$  去的地方；蓝色箭头是第二列。找一个能把网格压成一条直线的矩阵：面积会变成 0。再看看镜像时字母 F 会怎样。



试一个

什么也不做

拉伸

转 90°

推斜

镜像

压扁

a (左上)



1

b (右上)



1

c (左下)



0

d (右下)



1

(1, 0) 变成 (1, 0)

(0, 1) 变成 (1, 1)

面积乘以 (ad - bc) **1**

剩下的维数 2: 平面还是平面

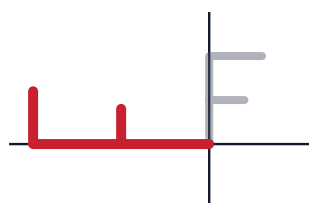
矩阵  $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ : 面积乘以 1, 剩下 2 维。



互动小工具。在网站上试一试: <https://kingofdumbs.com/zh/lessons/vectors-and-matrices>

## 矩阵乘法：顺序很重要

先转 90°，再拉伸



$$S \cdot R = (0 \ -2 \ / \ 1 \ 0)$$

先拉伸，再转 90°



$$R \cdot S = (0 \ -1 \ / \ 2 \ 0)$$

R: 转 90° · S: 横向拉伸 2 倍

两个矩阵相乘时，结果里的每个数，都是**第一个矩阵的某一行和第二个矩阵的某一列**做点积（OpenStax）。只有当第一个矩阵的列数等于第二个矩阵的行数时才能相乘： **$m \times k$  矩阵乘  $k \times n$  矩阵，得到  $m \times n$  矩阵。**

对一个点先做 B 再做 A，等于一次做矩阵 **AB**。注意：写法和做的顺序是反的（Austin）。而且 **AB 一般不等于 BA**（OpenStax）。设 R 为转 90°，S 为横向拉伸到两倍：先转再拉是  $S \cdot R = (0 \ -2 \ / \ 1 \ 0)$ ，先拉再转是  $R \cdot S = (0 \ -1 \ / \ 2 \ 0)$ 。

神经网络的一层会把输入乘上一个**权重矩阵**（PyTorch）。一张  $100 \times 100$  的灰度照片有 10000 个数；一个有 1000 个输出的层用的是  $1000 \times 10000$  的矩阵，每张照片要做 **1000 万次乘法**（自行计算）。层与层之间还有一个**不是乘法的步骤**；《逻辑回归还是神经网络？》那一课讲了它是什么，以及网络靠它多得到了什么。

显卡会**一次放进很多张照片**，每张一列，于是一切就变成一次巨大的矩阵乘法（NVIDIA）。

**笨蛋之王说：**

我看不见猫，只看见 3600 万个数。难的是知道该拿哪些矩阵去乘它们。



## 最后挑战

1. 一张  $100 \times 100$  像素的彩色照片。电脑打开它时要存多少个数？

- a. 100
- b. 10000, 每个像素一个
- c. 30000: 每个像素三个, 红、绿、蓝

2. 两支箭头互成直角。它们的点积是多少？

- a. 1
- b. 0
- c. 要看它们有多长

3. 把照片转  $90^\circ$ , 再横向拉伸。这和先拉伸、再旋转一样吗？

- a. 一样: 乘法怎么换顺序都一样
- b. 不一样: 矩阵相乘, 顺序很重要
- c. 只有照片是正方形时才一样



## 三句话总结

1. 在电脑眼里, 照片是一列数 (每个像素三个, 取值 0 到 255): 一个向量, 它的维数就是数的个数。
2. 点积把两列数逐项相乘再相加: 既能算出总价, 就像水果摊的账; 也能说明两支箭头有多同向 (垂直时为 0)。
3. 矩阵是一张能挪动平面上每个点的数表, 矩阵乘法把这些变换串起来, 而且顺序要紧: 这正是神经网络对每张照片做上百万次的运算。

## 参考资料

- 向量、长度和点积：OpenStax, *Calculus Volume 3*, [2.1](#) 和 [2.3](#) (水果摊的例子, 定理 2.4 和 2.5)。矩阵及其乘法：OpenStax, *College Algebra 2e*, [7.5](#)。
- 变换、行列式和维数：D. Austin, *Understanding Linear Algebra*, [2.2](#)、[2.5](#)、[2.6](#)、[3.4](#) 和 [3.5](#)。
- 像素和图像变换：Pillow 13.0, [Concepts](#) 和 [ImageTransform](#)。从照片到张量：torchvision 0.29, [ToTensor](#)。
- 神经网络：PyTorch 2.14, [nn.Linear](#) 和 [cosine\\_similarity](#)。NVIDIA, [Matrix Multiplication Background](#) 和 [Linear/Fully-Connected Layers](#)。

**\*\*这一课是怎么做的：\*\***在 Claude 的帮助下，根据上面的资料调研和撰写，资料查阅于 2026 年 10 月 17 日。没有单独链接的数字，是根据定义算出来的。例子（4000 × 3000 的照片、两家店、有 1000 个输出的层）都是编的。插图都是原创的。

## 答案

1. **c)** 30000: 每个像素三个, 红、绿、蓝. 对。  $100 \times 100 \times 3$ : 一个 30000 维的向量。
2. **b)** 0. 答对了。互相垂直, 点积就是 0, 不管多长。
3. **b)** 不一样: 矩阵相乘, 顺序很重要. 没错。就像袜子和鞋。

---

King of Dumbs 的课程, kingofdumbs.com。这是离线版; 可以动手玩的原版在:

<https://kingofdumbs.com/zh/lessons/vectors-and-matrices>

更新于 2026年10月17日.