

作者	生姜 DrGinger
脚本	生姜 DrGinger
视频	崔崔 CuiCui
开源学习资源	<a href="https://github.com/Visualize-ML">https://github.com/Visualize-ML</a>
平台	<a href="https://www.youtube.com/@DrGinger_Jiang">https://www.youtube.com/@DrGinger_Jiang</a> <a href="https://space.bilibili.com/3546865719052873">https://space.bilibili.com/3546865719052873</a> <a href="https://space.bilibili.com/513194466">https://space.bilibili.com/513194466</a>

## 2.2 转置



### 本节你将掌握的核心技能：

- ▶ 矩阵转置：将行、列互换的操作。
- ▶ 向量转置：行向量转置成列向量，列向量转置成行向量。
- ▶ 不同形状矩阵转置：细高、扁平、方阵。
- ▶ NumPy 转置操作：二维数组用.T 获取数组转置。
- ▶ 避开转置误区：转置不是对角镜像、不是左右/上下镜像，而是严格行列对换。
- ▶ 掌握各种转置运算的性质。

矩阵转置是个特别简单，但是又特别重要的线性代数运算。读者很容易犯错，因此我们特别拿出一节讲解矩阵转置。简单来说，矩阵转置是将矩阵的行列互换的一种基本运算。本节首先介绍了转置的定义，并通过列向量与行向量的转置说明其基本概念。接着，我们探讨了不同形状的矩阵如何在转置后发生变化，例如行向量变为列向量，细高矩阵变为扁平矩阵，方阵的转置保持形状，对称矩阵的转置不变等。最后，我们介绍了转置运算的重要性质，包括双重转置恢复原矩阵、转置对加法可交换、数乘对转置无影响等，为后续深入学习矩阵运算奠定基础。

### 什么是转置？

本书前文提过，列向量**转置** (transpose) 后获得行向量；反之亦然。

当把转置作用在矩阵上，产生的作用是矩阵的行列互换得到的新矩阵。转置是一种“矩阵 → 矩阵”运算。

**矩阵  $A$  的转置** (the transpose of a matrix  $A$ ) 记作  $A^T$  或  $A'$ 。NumPy 采用.T 代表数组转置运算，据此本书仅采用  $A^T$  记法。

如图 1 所示，一个  $m \times p$  矩阵  $A$  转置得到  $p \times m$  矩阵  $A^T$ 。

矩阵在转置前后，主对角线上的元素 (黑色框线) 位置保持不变。这是因为主对角线上的元素位于行索引、列索引相等的位置，即元素  $a_{i,i}$  在转置后仍然位于  $a_{i,i}$ 。

也就是说， $a_{1,1}$ 、 $a_{2,2}$ 、 $a_{3,3}$ 、等等元素转置之后位置不变。

把矩阵的每个元素看成是正方形，整个过程相当于矩阵  $A$  绕**主对角线** (main diagonal) 镜像。

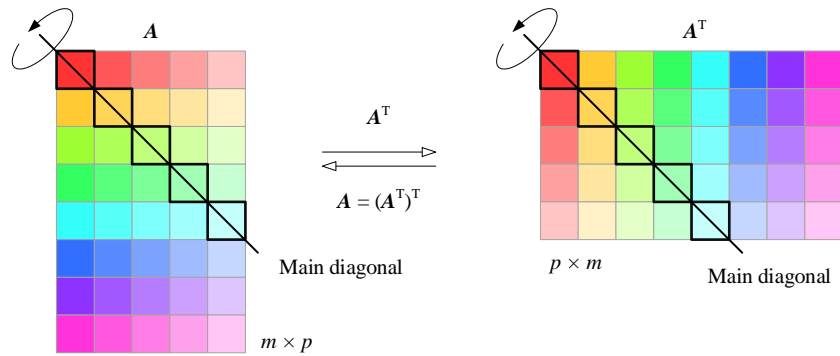


图 1. 矩阵转置

其实，矩阵转置一点也不复杂。把矩阵看成一个表格的话，矩阵转置，就是把表格的行、列进行对调。

## 列 → 行

如下图所示，将矩阵  $A$  写成一组列向量：

$$A = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_p \end{bmatrix} \quad (1)$$

矩阵  $A$  转置  $A^T$  可以展开写成：

$$A^T = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_p^T \end{bmatrix} \quad (2)$$

如图 2 所示，这意味着，当我们将矩阵  $A$  表示为一组列向量时，对矩阵进行转置的操作就可以理解为将每个原来的列向量转换成相应的行向量。

比如，矩阵  $A$  的第 1 列在转置后就成为了新矩阵  $A^T$  的第 1 行，第 2 列则变成了第 2 行，依此类推。

通过这种方式，我们可以直观地看到转置操作是如何将原本竖直排列的元素转变为横向排列。元素的相对顺序 (色调的排序) 没有变化。

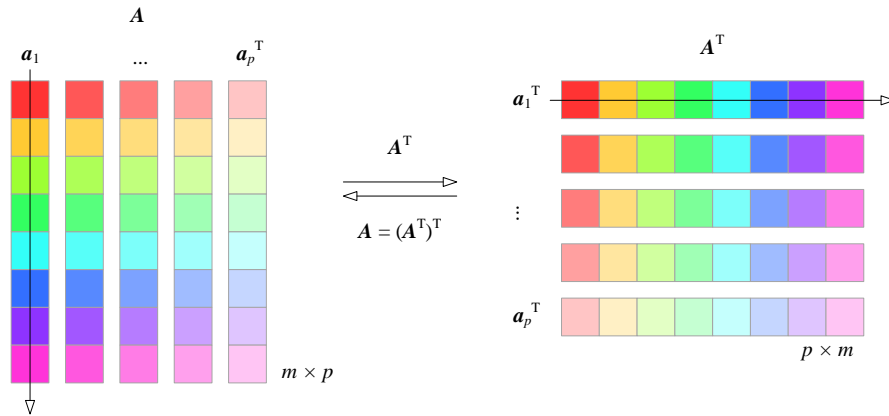


图 2. 矩阵转置将列变成行

例如,

$$A^T = \begin{bmatrix} 0 \\ 3 \\ 5 \end{bmatrix} \begin{bmatrix} 5 \\ 4 \\ 0 \end{bmatrix}^T = \begin{bmatrix} 0 & 3 & 5 \\ 5 & 4 & 0 \end{bmatrix} \quad (3)$$

原矩阵  $A$  中, 每列的第 1 个元素, 在转置后会变成  $A^T$  每行的第 1 个元素。

原矩阵每列的第 2 个元素, 会变成转置矩阵每行的第 2 个元素。

下面让我们利用 NumPy 完成矩阵 (二维数组) 的转置运算, 具体代码如代码 1。让我们聊聊其中关键语句。

线性大家已经熟悉 <sup>a</sup> 这两句。上节提过, `numpy.random.seed()` 设置随机数 `shengcheng` 的种子, 保证结果可复刻, 即每次运行代码时得到的随机数数组都一致。接着, `numpy.random.randint(0, 10, (5, 12))` 生成一个大小为  $5 \times 12$  的二维数组。`randint(0, 10)` 表示生成 0 到 9 之间的整数。

<sup>b</sup> 计算二维数组 (矩阵  $A$ ) 的转置, “`.T`” 是 NumPy 中获取矩阵转置的方法。

<sup>c</sup> 用 `seaborn.heatmap()` 创建一副热图, 也叫热力图。

`A[[0], :]` 表示选择矩阵  $A$  的第一行, 结果为二维数组, 如图 3 所示。图 3 还给出其他几行的转置。

`cmap = 'RdYlBu_r'` 指定了热图的色谱;

`square = True` 使热图的每个单元格保持为正方形;

`cbar = False` 禁用右侧的颜色条;

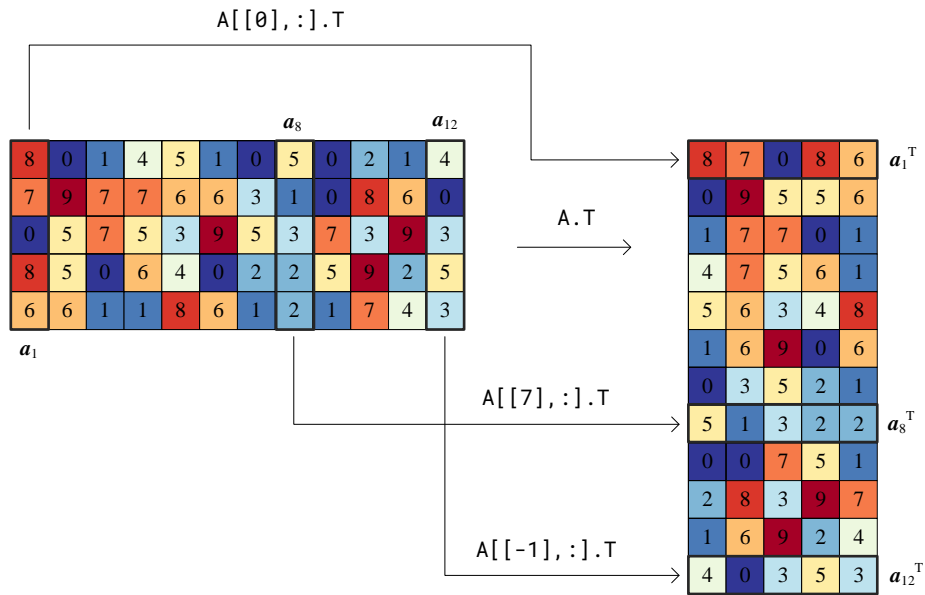
`annot = True` 在每个单元格中显示数字;

`vmin = 0` 和 `vmax = 9` 设置热图显示的数值范围;

`linecolor = 'w'` 设置单元格之间的边界颜色为白色;

`linewidths = 0.25` 设置单元格之间边界线的宽度。

<sup>d</sup> 和 <sup>c</sup> 类似。只不过 <sup>c</sup> 显示的是矩阵  $A$  第一行的转置 `A[[0], :].T`, 因此结果是第一行数据变成了列向量显示。

图 3. NumPy 二维数组的转置，列  $\rightarrow$  行代码 1. NumPy 二维数组的转置，列  $\rightarrow$  行 | LA\_02\_02\_01.ipynb

```

## 初始化
import numpy as np
import seaborn as sns

a np.random.seed(88)
A = np.random.randint(0, 10, (5, 12))

b ## 计算矩阵 A 的转置
A_T = A.T

c ## 行向量的转置
sns.heatmap(A[[0],:], cmap = 'RdYlBu_r', square = True,
            cbar = False, annot = True, vmin = 0, vmax = 9,
            linecolor = 'w', linewidths = 0.25)

d sns.heatmap(A[[0],:].T, cmap = 'RdYlBu_r', square = True,
            cbar = False, annot = True, vmin = 0, vmax = 9,
            linecolor = 'w', linewidths = 0.25)

```

## 行 $\rightarrow$ 列

反之，将矩阵  $A$  写成一组行向量：

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}^{(1)} \\ \mathbf{a}^{(2)} \\ \vdots \\ \mathbf{a}^{(m)} \end{bmatrix} \quad (4)$$

$\mathbf{A}^T$  则可以写成：

$$\mathbf{A}^T = \begin{bmatrix} \mathbf{a}^{(1)T} & \mathbf{a}^{(2)T} & \dots & \mathbf{a}^{(m)T} \end{bmatrix} \quad (5)$$

如图 4 所示，将矩阵写成一组行向量后，转置会将每个行向量转换为对应的列向量。但是元素的相对位置 (同一色调颜色的深浅渐变) 没有变化。

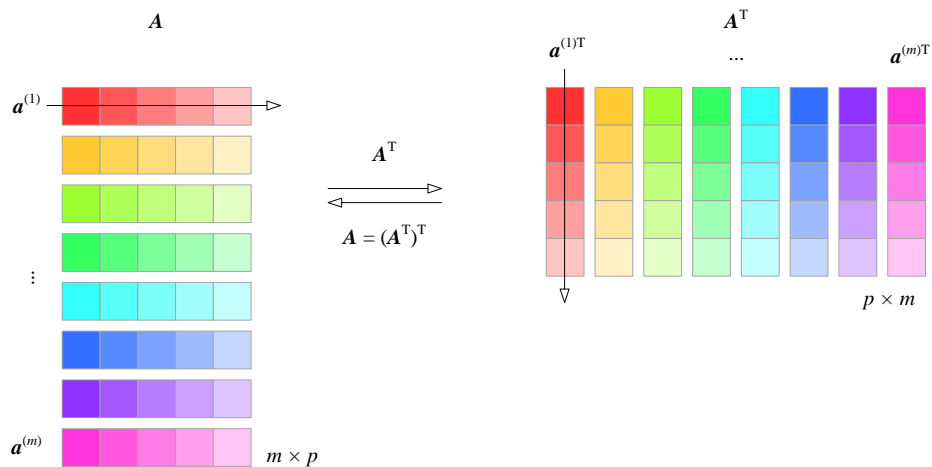


图 4. 矩阵转置将行变成列

回到刚才的例子，

$$\mathbf{A}^T = \begin{bmatrix} \begin{bmatrix} 0 & 5 \end{bmatrix} \\ \begin{bmatrix} 3 & 4 \end{bmatrix} \\ \begin{bmatrix} 5 & 0 \end{bmatrix} \end{bmatrix}^T = \begin{bmatrix} \begin{bmatrix} 0 \\ 3 \\ 5 \end{bmatrix} & \begin{bmatrix} 5 \\ 4 \\ 0 \end{bmatrix} \end{bmatrix} \quad (6)$$

原矩阵  $\mathbf{A}$  中，每行的第 1 个元素，在转置后会变成  $\mathbf{A}^T$  中每列的第 1 个元素。

每行的第 2 个元素，在转置后会变成  $\mathbf{A}^T$  中每列的第 2 个元素。

以此类推，原矩阵中第  $i$  行的所有元素，都会变成转置矩阵中第  $i$  列的对应元素。转置的本质就是将矩阵的行与列互换，将行向量的排列转换为列向量的排列方式。

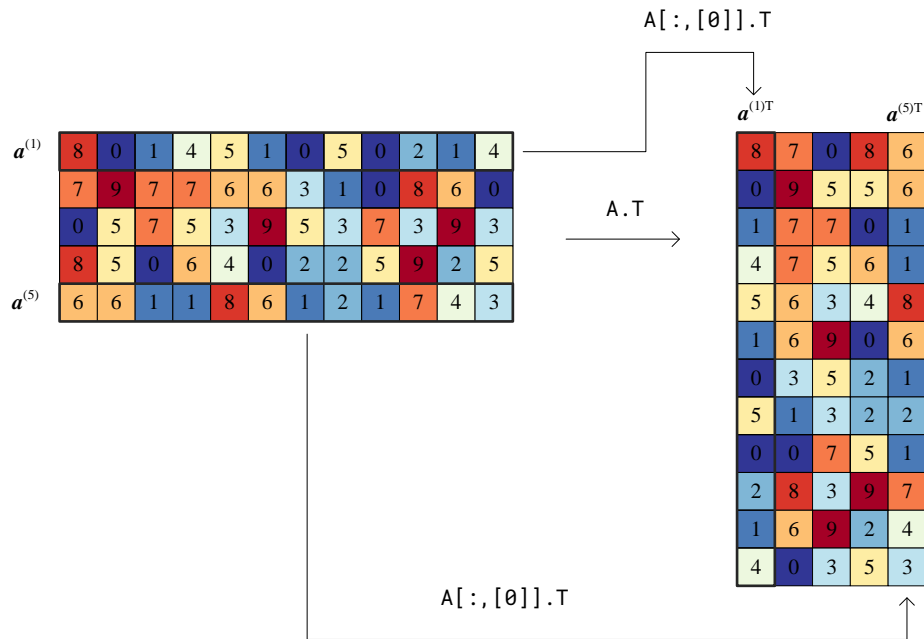


图 5. NumPy 二维数组的转置，行→列



LA\_02\_02\_01.ipynb 还可可视化矩阵  $A$  的行向量转置为列向量，具体如图 5 所示。

### 不同形状的矩阵转置

下面，让我们总结一下不同形状矩阵转置后的形状变换。

如图 6 所示，行向量的转置为列向量，比如

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad (7)$$

列向量的转置为行向量，比如

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}^T = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \quad (8)$$

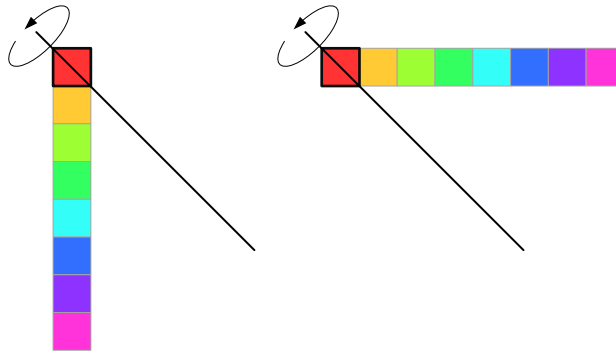


图 6. 行、列向量的相互转置

如图 1 所示，细高矩阵 (行数多于列数) 转置得到扁平矩阵 (行数少于列数)：

$$\begin{bmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \\ 4 & 8 \end{bmatrix}^T = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix} \quad (9)$$

反之，扁平矩阵转置得到细高矩阵：

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}^T = \begin{bmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \\ 4 & 8 \end{bmatrix} \quad (10)$$

如图 7 所示，方阵的转置仍然是方阵：

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}^T = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad (11)$$

但是元素位置绕主对角线镜像。

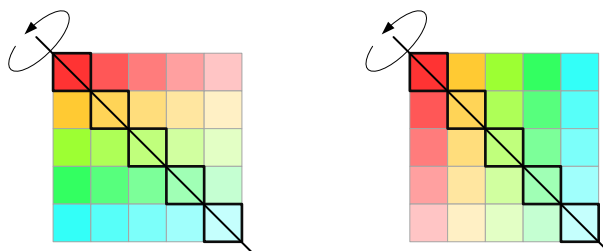


图 7. 方阵的转置

如图 8 所示，**对称矩阵** (symmetric matrix) 的转置为自身，比如

$$\begin{bmatrix} 1 & 8 \\ 8 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 8 \\ 8 & 6 \end{bmatrix} \quad (12)$$

也就是说，如果矩阵  $A$  是对称矩阵，则满足  $A = A^T$ 。矩阵形状是下一节要探讨的话题。

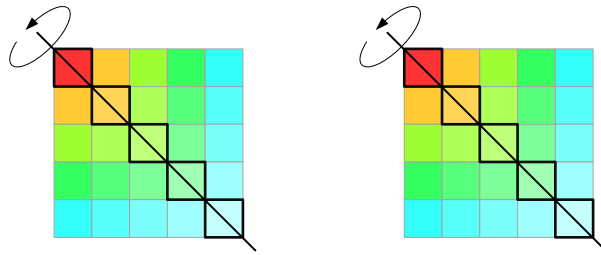
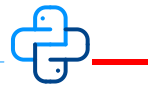


图 8. 对称矩阵的转置

不同形状的矩阵转置后，形状可能会改变，但核心规则始终是行变列，列变行；而且，元素之间的相对位置没有变化。



LA\_02\_02\_02.ipynb 计算不同形状矩阵的转置，请大家自行学习。

## 几何视角

如图 9 所示， $A$  的行向量可以看成是平面中的三个箭头，而  $A$  的列向量可以看成是三维空间中的两个箭头。



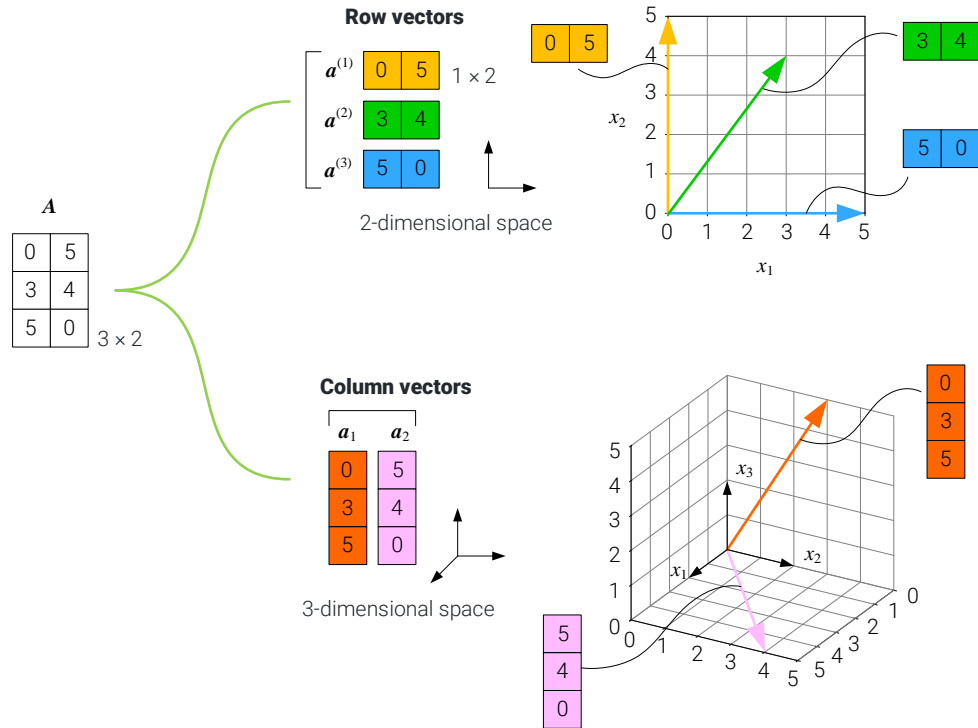
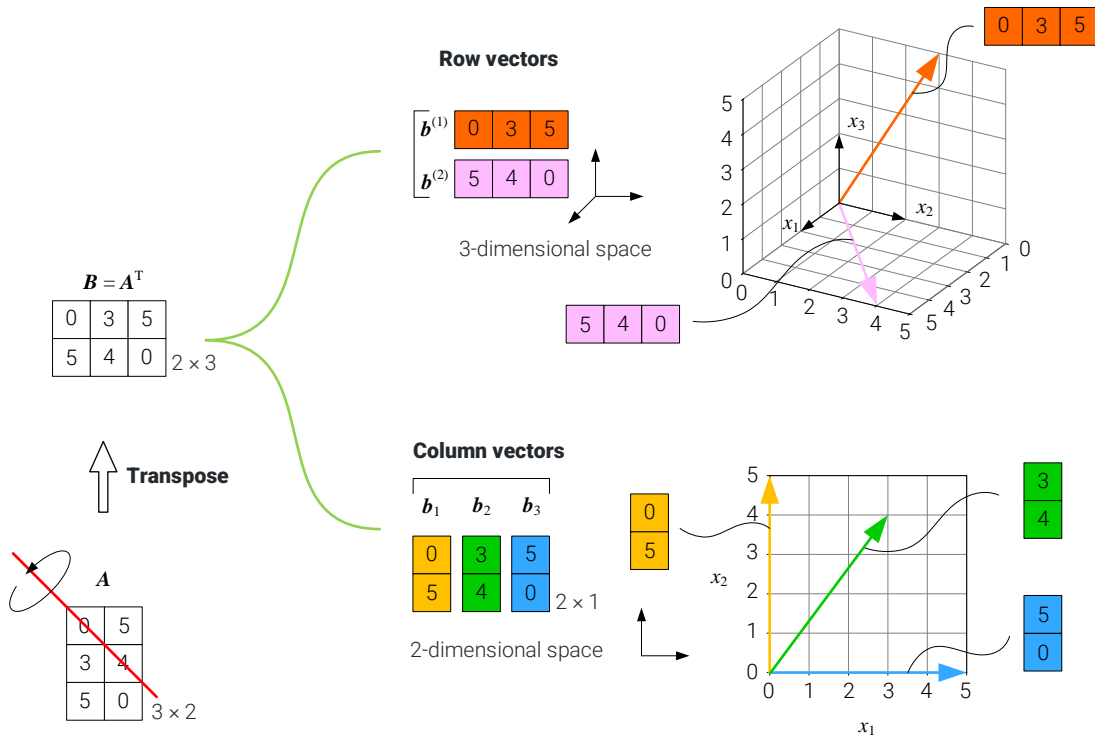
图 9. 矩阵  $A$  的行向量、列向量，图片来自《可视之美》

图 10 所示为矩阵转置  $A^T$  的行列向量。而  $A^T$  的行向量是三维空间的两个箭头， $A^T$  的列向量是平面中的三个箭头。

图 10. 矩阵  $A$  的转置之后矩阵的行向量、列向量，图片来自《可视之美》

## 注意事项

矩阵转置并不是简单的镜像操作，而是一个明确的行、列对调过程，请初学者务必注意以下几点。

“主对角线”不是“对角线”。矩阵转置不是对角线镜像。如图 11 (a) 所示，对于非方阵，沿对角线镜像没有意义。

只有方阵 (行数等于列数) 会让转置看起来像对角线镜像，但这仅是特殊情况。不同于对角线，矩阵的主对角线是指矩阵中行、列序号相同元素构成的连线。前文提过，矩阵  $A$  的主对角线就是元素  $a_{i,i}$  构成的连线。

矩阵转置也不是左右镜像，如图 11 (b)。转置前后，列的顺序会转化为行的顺序，即转置严格地将第  $i$  行的元素调整到第  $i$  列。例如，如果原矩阵的第 1 列在左侧，转置后它成为第 1 行，而不是移动到最右侧。

矩阵转置也不是上下镜像，如图 11 (c)。矩阵上方的行不会因为转置而移动到下方。

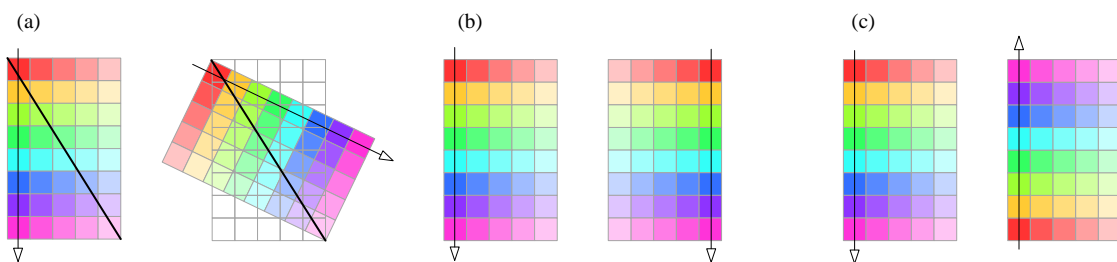


图 11. 矩阵转置的误区

## 重要性质

矩阵转置如下几个重要性质值得大家重视。

**双重转置恢复原矩阵**，即

$$\left(A^T\right)^T = A \quad (13)$$

矩阵转置的定义是将矩阵的行变成列，列变成行。如果对已经转置过的矩阵再次转置，就会恢复原来的行列结构。比如，给定原矩阵为

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{bmatrix} \quad (14)$$

对矩阵  $A$  转置得到

$$A^T = \begin{bmatrix} a_{1,1} & a_{2,1} & a_{3,1} \\ a_{1,2} & a_{2,2} & a_{3,2} \end{bmatrix} \quad (15)$$

$A^T$  再转置得到

$$(\mathbf{A}^T)^T = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{bmatrix} \quad (16)$$

**转置运算对矩阵加法可交换**，即

$$(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T \quad (17)$$

矩阵加法是按元素相加的，因此转置操作不会影响加法结果的每个元素的相对位置。

比如，给定如下两个形状相同的矩阵

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \end{bmatrix} \quad (18)$$

先加后转置，即计算  $(\mathbf{A} + \mathbf{B})^T$

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} a_{1,1} + b_{1,1} & a_{1,2} + b_{1,2} \\ a_{2,1} + b_{2,1} & a_{2,2} + b_{2,2} \\ a_{3,1} + b_{3,1} & a_{3,2} + b_{3,2} \end{bmatrix} \quad (19)$$

对计算转置  $(\mathbf{A} + \mathbf{B})^T$

$$(\mathbf{A} + \mathbf{B})^T = \begin{bmatrix} a_{1,1} + b_{1,1} & a_{2,1} + b_{2,1} & a_{3,1} + b_{3,1} \\ a_{1,2} + b_{1,2} & a_{2,2} + b_{2,2} & a_{3,2} + b_{3,2} \end{bmatrix} \quad (20)$$

对  $\mathbf{A}$ 、 $\mathbf{B}$  分别先转置

$$\mathbf{A}^T = \begin{bmatrix} a_{1,1} & a_{2,1} & a_{3,1} \\ a_{1,2} & a_{2,2} & a_{3,2} \end{bmatrix}, \quad \mathbf{B}^T = \begin{bmatrix} b_{1,1} & b_{2,1} & b_{3,1} \\ b_{1,2} & b_{2,2} & b_{3,2} \end{bmatrix} \quad (21)$$

然后计算  $\mathbf{A}^T + \mathbf{B}^T$  两者之和

$$\mathbf{A}^T + \mathbf{B}^T = \begin{bmatrix} a_{1,1} + b_{1,1} & a_{2,1} + b_{2,1} & a_{3,1} + b_{3,1} \\ a_{1,2} + b_{1,2} & a_{2,2} + b_{2,2} & a_{3,2} + b_{3,2} \end{bmatrix} \quad (22)$$

通过计算发现，无论是先加后转置，还是先转置后加，最终结果都相同。

**数乘对转置无影响**，即

$$(k\mathbf{A})^T = k\mathbf{A}^T \quad (23)$$

其中， $k$  为标量；数乘是对矩阵的所有元素进行同样的缩放，而转置仅改变元素的位置，不影响数值，因此数乘运算可以与转置交换顺序。

举个例子，先数乘得到

$$k\mathbf{A} = \begin{bmatrix} ka_{1,1} & ka_{1,2} \\ ka_{2,1} & ka_{2,2} \\ ka_{3,1} & ka_{3,2} \end{bmatrix} \quad (24)$$

对  $k\mathbf{A}$  转置

$$(k\mathbf{A})^T = \begin{bmatrix} ka_{1,1} & ka_{2,1} & ka_{3,1} \\ ka_{1,2} & ka_{2,2} & ka_{3,2} \end{bmatrix} \quad (25)$$

换个计算路径，先对  $\mathbf{A}$  转置

$$\mathbf{A}^T = \begin{bmatrix} a_{1,1} & a_{2,1} & a_{3,1} \\ a_{1,2} & a_{2,2} & a_{3,2} \end{bmatrix} \quad (26)$$

再对其进行数乘

$$k\mathbf{A}^T = \begin{bmatrix} ka_{1,1} & ka_{2,1} & ka_{3,1} \\ ka_{1,2} & ka_{2,2} & ka_{3,2} \end{bmatrix} \quad (27)$$

通过计算发现，无论是先数乘后转置还是先转置后数乘，最终的矩阵都相同。

⚠ 注意，转置、矩阵乘法、逆矩阵结合后，还有几个性质值得我们注意，现在还不需要我们费心。



请大家用 DeepSeek/ChatGPT 等工具完成本节如下习题。

**Q1.** LA\_02\_01\_01.ipynb 反复调用 `seaborn.heatmap()` 绘制类似热图，请将重复代码写成一个 Python 函数，方便后续可视化调用。

**Q2.** 请用随机数生成器创建一个  $5 \times 8$  矩阵，然后提取这个矩阵的每一行；最后，再用 `numpy.block()` 把这些行向量合成为原始矩阵。

<https://numpy.org/doc/2.2/reference/generated/numpy.block.html>

**Q3.** 请编写 Python 代码验证“双重转置恢复原矩阵”；判断矩阵是否相同，请用 `numpy.array_equal()`。

**Q4.** 请编写 Python 代码验证“转置运算对矩阵加法可交换”。

**Q5.** 请编写 Python 代码验证“数乘对转置无影响”。

**Q6.** 请编写 Python 尝试复刻图 9、图 10。