

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

1.4 向量标量乘法

本节介绍了向量的标量乘法及其几何意义，并通过代码和图示展示了如何在计算机中实现向量变换和可视化。从几何角度来看，标量乘法是对向量的伸缩变换。在 RGB 颜色空间中，向量标量乘法可以改变颜色的深浅，标量越小，颜色越暗，趋近于黑色。我们还引入了向量的线性组合。线性组合是多个向量通过标量加权求和生成新向量的过程。在此基础上，本节还比较了线性无关、线性相关两个概念。

标量乘法

向量标量乘法 (scalar multiplication of a vector) 是指向量的每个分量分别乘以同一标量，得到新的向量。

比如，给定非零 n 维列向量 \mathbf{a} 、标量 k ，两者乘法定义如下

$$k\mathbf{a} = k \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} ka_1 \\ ka_2 \\ \vdots \\ ka_n \end{bmatrix} \quad (1)$$

代码 1 计算向量标量乘法。下面聊聊其中关键语句。

- a** 定义了一个一维数组，代表向量。在 NumPy 中，`numpy.array()` 函数用于创建数组，可以是一维、二维或更高维的矩阵。
- b** 利用 `numpy.linalg.norm()` 计算向量 `a_vec` 的长度 (大小、模、L2 范数、欧几里得范数)。在 NumPy 中，`np.linalg.norm()` 是一个用于计算向量或矩阵的大小的函数。在默认情况下，如果输入为向量，函数计算的是欧几里得范数，也就是向量的长度。
- c** 计算 `a_vec` 与 `k` 的乘积。NumPy 允许直接用标量和数组相乘，表示对向量的每个分量都乘以 `k`。
- d** 再次调用 `numpy.linalg.norm()` 计算 `k_a_vec` 的长度。

代码 1. 向量标量乘法 |  LA_01_04_01.ipynb

```
## 初始化
import numpy as np

## 定义向量
a a_vec = np.array([3, 4])
b np.linalg.norm(a_vec)

## 标量乘法
k = -2
c k_a_vec = k * a_vec
d np.linalg.norm(k_a_vec)
```

几何视角

从几何角度来看，向量的标量乘法是对向量的伸缩变换；正数标量使向量同向伸缩，负数标量使向量反向伸缩。

显而易见，新向量 ka 与原非零向量 a **共线**；上一节提过，**共线** (colinear) 的意思是，两个向量位于同一直线上，方向相同、相反。

当 $|k| > 1$ 时，向量长度增大。如图 1 (a) 所示， $k > 1$ 时，新向量 ka 的方向与原向量 a 相同，长度增大；如图 1 (b) 所示， $k < -1$ 时， ka 与 a 反向，长度增大。

当 $0 < |k| < 1$ 时，向量缩短；如图 1 (c) 所示，当 $0 < k < 1$ 时，新向量 ka 的方向与原向量 a 相同，长度缩短；如图 1 (d) 所示， ka 与 a 反向，长度缩短。

再看两个特殊值。

如图 1 (e) 所示，当 $k = -1$ 时，新向量 ka 为原向量 a 的**反向量** (opposite vector)。

上一节提过，**反向量**是指将一个向量的每个分量取相反数得到的新向量。

几何角度来看，**反向量**是指方向与原向量相反但长度相等的向量，它与原向量关于原点对称。

如图 1 (f) 所示，当 $k = 0$ 时，新向量 ka 为零向量 0 。

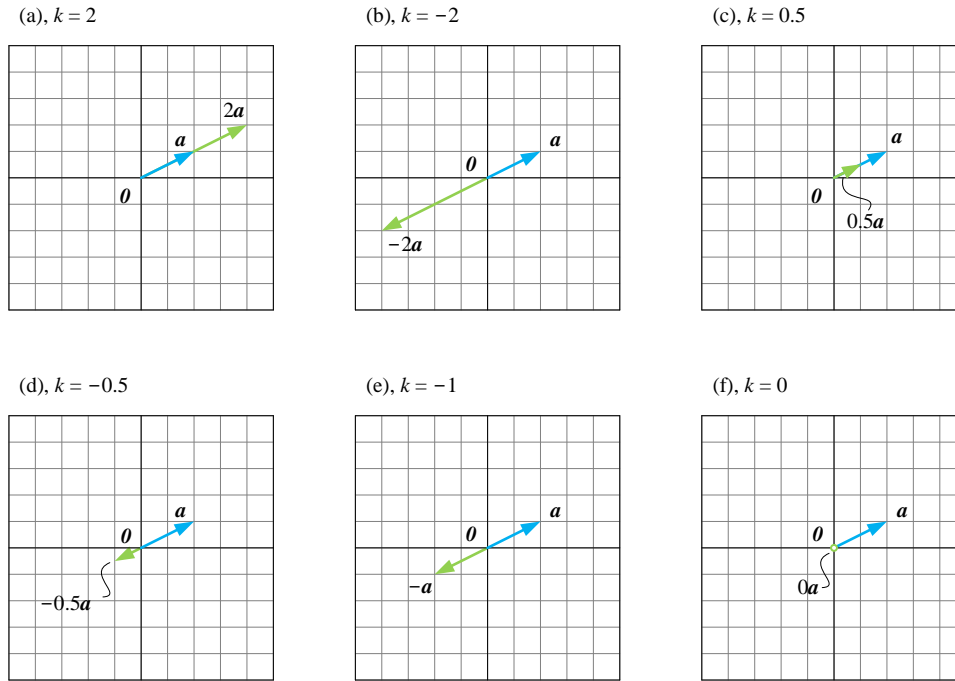


图 1. 向量的缩放

代码 2 利用 `matplotlib.pyplot.arrow()` 函数可视化向量标量乘法。下面聊聊其中关键语句。

a 和 **b** 都使用 Matplotlib 的 `matplotlib.pyplot.arrow()` 函数在二维坐标系中绘制一个带箭头的向量。`matplotlib.pyplot.arrow()` 函数需要我们提供起点坐标、终点坐标，以及箭头的样式参数。

在 **a** 中，`0, 0` 代表箭头的起点，而 `a_vec[0]`, `a_vec[1]` 代表箭头的 x 方向和 y 方向的分量，即向量 a 的坐标。

参数 `head_width=0.2` 设置箭头的宽度，使其在图中更清晰，`head_length=0.3` 设置箭头的长度，使其更容易识别。

参数 `fc='#00B0F0'` 表示箭头的填充颜色 (face color)，在这里是蓝色，`ec='#00B0F0'` 表示箭头的边缘颜色 (edge color)，与填充颜色一致，确保箭头颜色统一。

参数 `label="a"` 试图为箭头添加图例。

在 Matplotlib 中，可以使用多种方式指定颜色，包括十六进制颜色代码 `hex`、RGB 颜色、HTML 颜色名称、单字符缩写、灰度色阶、Colormap 颜色映射等等。十六进制颜色代码中，`'#FF0000'` 代表纯红色，`'#00FF00'` 为纯绿色，`'#0000FF'` 代表纯蓝色，`'#808080'` 则是一个灰色。

请大家逐行注释代码 2 中其他语句。

代码 2. 向量标量乘法的可视化 |  LA_01_04_02.ipynb

```

## 初始化
import numpy as np
import matplotlib.pyplot as plt

## 定义向量
a_vec = np.array([2, 1])

## 标量乘法
k = -2
k_a_vec = k * a_vec

## 可视化
plt.figure(figsize=(6,6))

# 绘制向量 a
plt.arrow(0, 0, a_vec[0], a_vec[1],
          head_width=0.2, head_length=0.3,
          fc='#00B0F0', ec='#00B0F0', label="a")

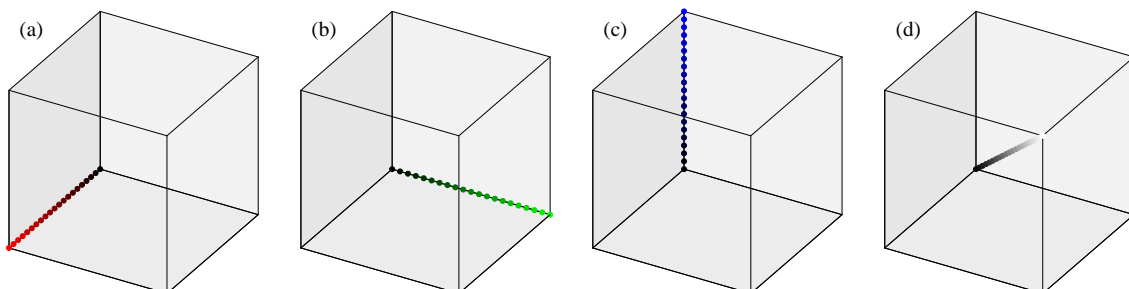
# 绘制向量 k * a
plt.arrow(0, 0, k_a_vec[0], k_a_vec[1],
          head_width=0.2, head_length=0.3,
          fc='#92D050', ec='#92D050', label="k*a")

# 装饰
plt.gca().set_aspect('equal', adjustable='box')
plt.xlim(-5, 5) # 设置 x 轴范围
plt.ylim(-5, 5) # 设置 y 轴范围
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.xticks(np.arange(-5,6))
plt.yticks(np.arange(-5,6))
plt.grid(color='gray', alpha=0.8, linestyle='--', linewidth=0.25)
plt.legend()

```

RGB 颜色变深

还是回到 RGB 颜色空间中，如图 2 所示，红色向量 e_1 、绿色向量 e_2 、蓝色向量 e_3 分别乘以标量 k ($0 < k < 1$)，随着 k 靠近 0，向量（散点）不断缩短，对应颜色不断变深。但是，向量方向不变，也就是说色调不变。



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

图 2. 红色、绿色、蓝色、白色颜色变深

我们知道，在 RGB 颜色空间中，3 维全 1 列向量， $[1, 1, 1]^T$ 对应白色。当我们用标量 k ($0 < k < 1$) 对全 1 列向量进行标量乘法，


$$k(\mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_3) = k \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} k \\ k \\ k \end{bmatrix} \quad (2)$$

得到的新颜色代表灰色；当 k 接近 1 时，灰度越浅，颜色接近白色；当 k 接近 0 时，颜色逐渐趋近黑色。 $k=0$ 我们得到黑色， $k=1$ ，我们得到白色。

因此，通过逐步调整标量 k 的值，我们可以在灰度范围内实现从黑到白的平滑渐变。

图 3 两个子图分别采用不同方式定义一组颜色，并基于这些颜色绘制了对应的散点序列，每组散点的颜色逐渐加深。颜色的变化通过标量乘法实现，即对每个基准颜色乘以一个连续变化的 k ($0 < k < 1$)，使颜色从原色逐渐向深色过渡。

按一定规则，或者随机给定 RGB 空间中不同的颜色向量，然后用一个介于 0 和 1 之间的渐变标量对其进行缩放，我们可以创造出丰富的渐变效果，让颜色在同一色调下展现不同的明暗层次。

 请大家思考，如何让颜色变浅。

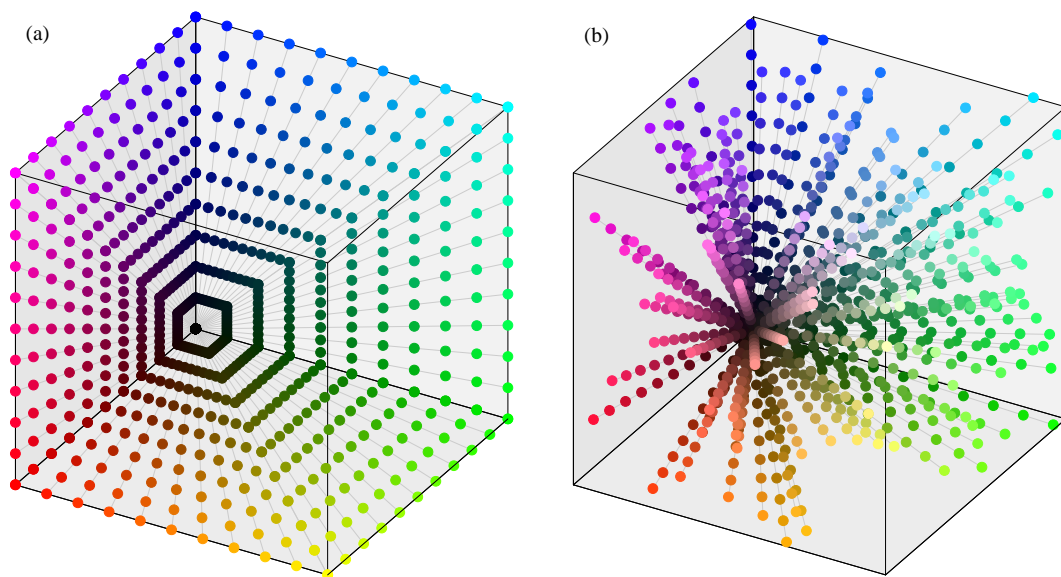


图 3. 颜色变深

线性组合

在向量运算中，标量乘法、向量加法的结合能够生成更广泛的空间。

给定两个非零、等维数向量 \mathbf{a} 、 \mathbf{b} ，我们可以通过标量 k 、 t 对它们进行加权求和，即

$$\mathbf{c} = k\mathbf{a} + t\mathbf{b}$$

(3)

其中 k 、 t 为实数。

上式实际是 \mathbf{a} 和 \mathbf{b} 的**线性组合** (linear combination)。**线性组合**指的是通过一组向量和对应的标量 (系数) 的加权和来构造新的向量。

几何上来看, 我们可以通过调整 k 、 t 的值, 生成一条穿过原点的直线 (当 \mathbf{a} 、 \mathbf{b} 共线), 或者填充整个平面 (当 \mathbf{a} 、 \mathbf{b} 不共线)。

下面, 让我们分析这两种情况。

线性无关

如果两个非零向量 \mathbf{a} 、 \mathbf{b} 不共线, 也就是说不存在标量 k 使得 $\mathbf{a} = k\mathbf{b}$ (或等价地, $\mathbf{b} = k\mathbf{a}$) 成立, 我们称非零向量 \mathbf{a} 、 \mathbf{b} **线性无关** (linearly independent)。

换句话说, \mathbf{a} 、 \mathbf{b} 不能通过标量缩放相互表示, 因此它们在几何上不会落在同一条直线上。

几何上, \mathbf{a} 、 \mathbf{b} 能够**张成** (span) 一个二维平面; 这个平面可以记做, $\text{span}(\mathbf{a}, \mathbf{b})$ 。本书前文提过**张成**这个概念。

简单来说, **张成**是指一组向量的所有线性组合构成的集合; 也就是说, 这些向量通过加法和数乘能够生成的所有可能的向量的集合。

让我们看两组例子。

如图 4 所示, 四个子图中 \mathbf{a} 、 \mathbf{b} 均不共线, 即线性无关。若 k 、 t 取所有实数, \mathbf{a} 、 \mathbf{b} 则可以**张成**整个平面。也就是说, 非零、二维向量 \mathbf{a} 、 \mathbf{b} **线性无关**, 平面内的向量都可以由 $k\mathbf{a} + t\mathbf{b}$ 通过线性组合的方式表示。

特别地, 若 k 、 t 取整数, 则向量 $k\mathbf{a} + t\mathbf{b}$ 终点位于规则网格的交点。它们具有一个共同特征: 彼此平行、间距相等, 并且经过原点。

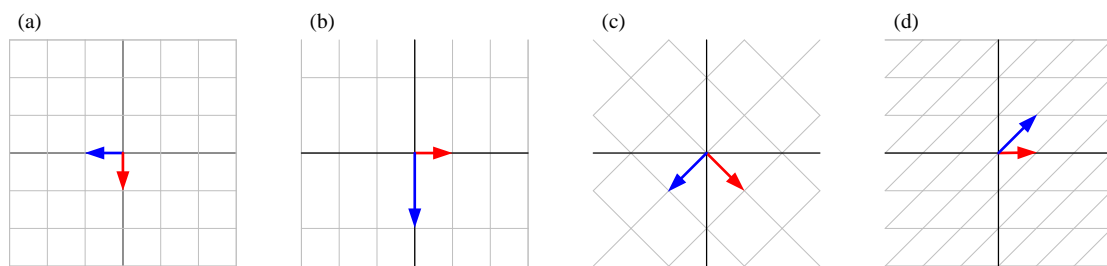
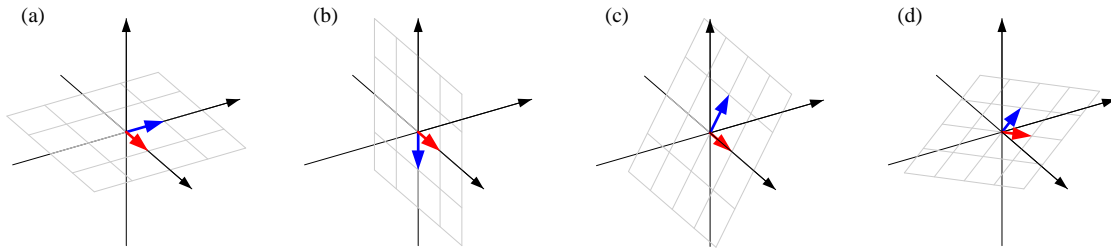


图 4. \mathbf{a} 、 \mathbf{b} 交织成平面网格, \mathbf{a} 、 \mathbf{b} 为 2 维向量

如图 5 所示, 两个不共线 (线性无关) 的 3 维向量 \mathbf{a} 、 \mathbf{b} 在三维空间中撑起的也是 (过原点) 二维平面。线性无关意味着 \mathbf{a} 、 \mathbf{b} 不能通过一个单独的数倍关系互相表示, 而是能“独立”地贡献新的方向。

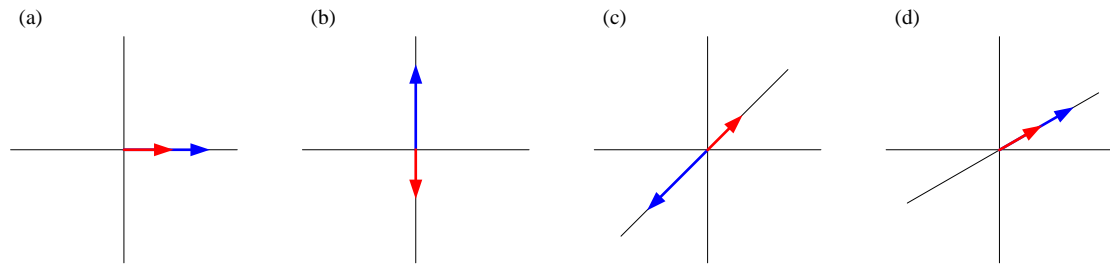
图 5. a 、 b 交织成平面网格， a 、 b 为 3 维向量

线性相关

然而，当 a 、 b 共线时，或称**线性相关** (linearly dependent)， a 、 b 的线性组合只能沿着 a 、 b 所在同一条经过原点的直线变化，无法扩展到更高维度的空间。

换句话说，所有由 a 、 b 线性组合得到的向量仍然落在两者共同所在的直线上，无法覆盖整个二维空间。

如图 6 所示，二维非零向量 a 、 b 共线，它们张成一条 (过原点的) 直线，无法撑起整个平面。

图 6. a 、 b 共线， a 、 b 为 2 维非零向量

如图 7 所示，在三维空间中，若两个非零向量 a 、 b 线性相关，它们的线性组合仍然只能沿着一条经过原点的直线变化。无论如何调整标量系数 k 、 t ，所有由 $ka + tb$ 生成的向量都局限在这条直线上，无法张成一个平面。

直观来看， b 只是 a 的若干倍数，因此它们指向同一方向或相反方向，本质上并没有提供额外的维度信息。这样的一组向量缺乏独立性，不能用于构造更高维度的空间结构。

本书后续会专门介绍这些向量空间概念，这里仅仅作作为铺垫。

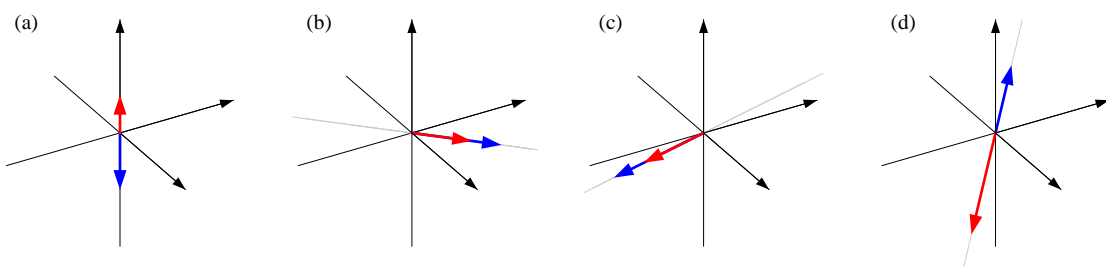


图 7. a 、 b 共线， a 、 b 为 3 维非零向量

请大家注意，向量的维数和它所处的空间维数是两个不同的概念。

一个二维非零向量，比如 $\mathbf{x} = [x_1, x_2]^T$ ，需要在二维平面中展示，但它本身张成的空间 $\text{span}(\mathbf{x})$ 是“一维的”，因为它的数乘结果都在同一条（过原点）直线上。无论如何放大或缩小这个向量，它仍然沿着原来的方向（或反方向），不会扩展到整个平面。

类似地，一个三维非零向量，比如 $\mathbf{x} = [x_1, x_2, x_3]^T$ ，虽然需要在三维空间中可视化，但它本身仍然是一维的。它的数乘结果都沿着固定的方向（或反方向），形成一条穿过原点的直线，而不是填满整个三维空间。

这就像房间里的一根细长的直线，无论两端如何延长，它依旧只是“一维的”，不会变成一个面或者一个体。

向量的维数表示它包含多少个分量，而它所处的空间维数决定了我们如何在几何上理解和展示这个向量。

颜色线性组合

在 RGB 颜色空间中，线性组合就是各个颜色的混合，下面让我们举例讲解。

纯红色 \mathbf{e}_1 、纯绿色 \mathbf{e}_2 的线性组合记作

$$k_1 \mathbf{e}_1 + k_2 \mathbf{e}_2 = k_1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + k_2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad (4)$$

如图 8 (a) 所示， k_1 、 k_2 由随机数发生器产生。随机数 k_1 、 k_2 满足 $[0, 1]$ 区间**连续均匀随机分布** (continuous uniform distribution)。简单来说，**连续均匀随机分布**是指在一个区间内，每个点被选中的概率（可能性）完全相同，没有偏向性。

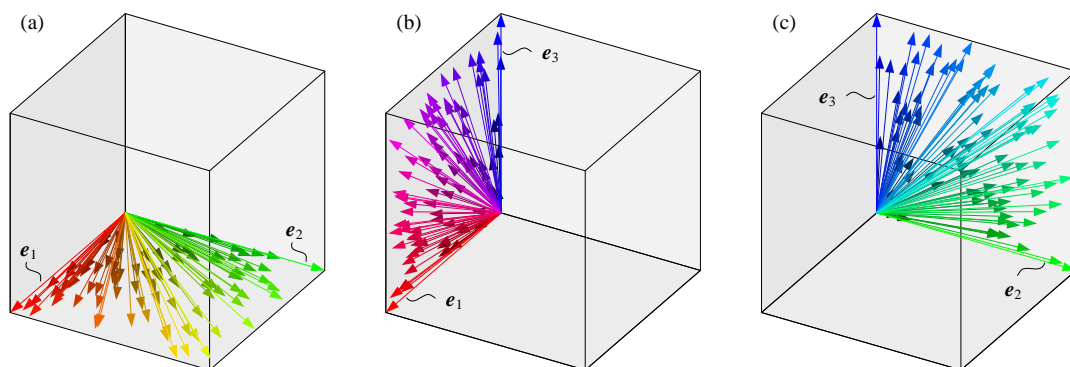


图 8. 两个颜色的线性组合

可以想象当 k_1 、 k_2 取得所有可能得实数， $k_1 \mathbf{e}_1 + k_2 \mathbf{e}_2$ 构成的向量将**张成**整个 $x_1 x_2$ 平面。

如图 8 (b) 所示为纯红色 \mathbf{e}_1 、纯蓝色 \mathbf{e}_3 的线性组合

$$k_1 \mathbf{e}_1 + k_2 \mathbf{e}_3 = k_1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + k_3 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (5)$$

随机数 k_1 、 k_3 也是满足 $[0, 1]$ 区间连续均匀随机分布。请大家自行分析图 8 (c)。

而纯红色 \mathbf{e}_1 、纯绿色 \mathbf{e}_2 、纯蓝色 \mathbf{e}_3 的加权混合得到的是如图 9 所示多彩的 RGB 色彩空间。

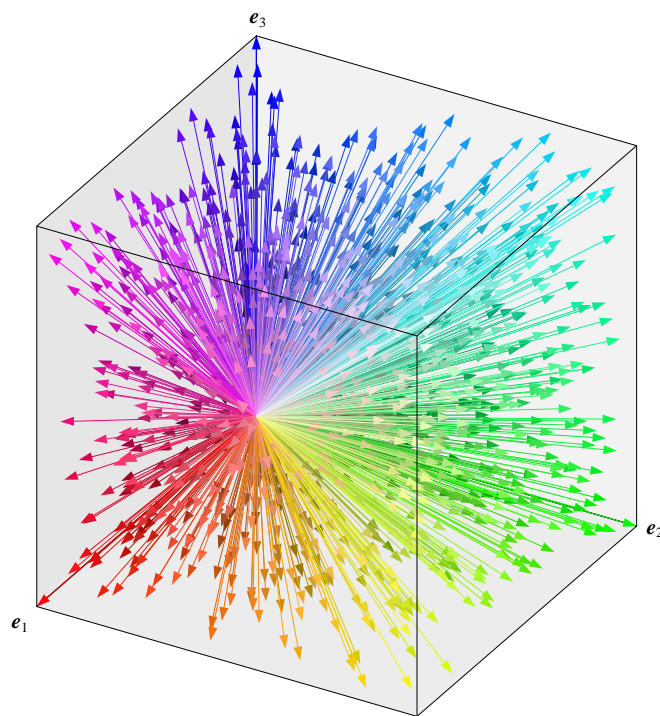


图 9. 三个颜色向量的线性组合



请大家用 DeepSeek/ChatGPT 等工具完成本节如下习题。

- Q1.** \mathbf{a} 、 \mathbf{b} 、 \mathbf{c} 均为 2 维非零向量，如果三者线性相关，几何上有什么特点？
- Q2.** \mathbf{a} 、 \mathbf{b} 、 \mathbf{c} 均为 3 维非零向量，如果三者线性相关，几何上又有什么特点？
- Q3.** 混合纯红色向量 $[1, 0, 0]^T$ 、深红色向量 $[1/2, 0, 0]^T$ ，得到的颜色有什么特点？
- Q4.** 混合白色向量 $[1, 1, 1]^T$ 、灰色向量 $[1/2, 1/2, 1/2]^T$ ，得到的颜色有什么特点？
- Q5.** 请尝试编写 Python 代码绘制图 3 (a) 这幅散点图。
- Q6.** 请尝试编写 Python 代码绘制图 9 这幅箭头图。
- Q7.** 如下哪些组向量的线性组合能够“撑起”整个平面？

► $\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

▶ $\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}$

▶ $\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

▶ $\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}$