jupyter PlottingMatrixMultiplication-zh Last Checkpoint: 20 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run Markdown

可视化矩阵乘法

在*线性转换和矩阵*视频中，你学习了向量可以如何分解为基向量 $\hat{i}$ 和 $\hat{j}$。你还学习了如何转换向量：将该向量的 x 和 y 值乘以*转换的*基向量 $\hat{i_T}$ 和 $\hat{j_T}$，并对结果求和（请参阅*方程 1*）。

$$transformed\ \vec{v} = x\,\hat{i_T} + y\,\hat{j_T}$$

方程 1

你了解到，这种通过使用*转换的*基向量转换向量的方法与矩阵乘法一样（请参阅*方程 2*）。

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = x\begin{bmatrix} a \\ c \end{bmatrix} + y\begin{bmatrix} b \\ d \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

方程 2

在此 Lab 中，你将：

- 绘制一个分解为基向量 $\hat{i}$ 和 $\hat{j}$ 的向量
- 绘制一个利用*方程 1* 的向量转换过程
- 展示可以通过矩阵乘法实现同一向量转换（*方程 2*）

绘制分解为基向量 $\vec{i}$ 和 $\vec{j}$ 的向量 $\vec{v}$

在该 Lab 的第一部分，我们将如下所示地定义向量 $\vec{v}$：

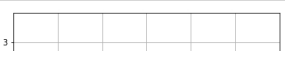
$$\vec{v} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

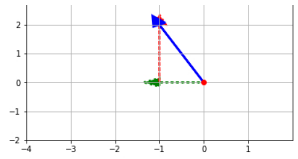
下面概述了以下 Python 代码包含的绘制向量 $\vec{v}$、$\vec{i}$ 和 $\vec{j}$ 的步骤。

1. 使用 *import* 方法使 NumPy 和 Matlibplot python 软件包变得可用
2. 定义向量 $\vec{v}$
3. 定义基向量 $\vec{i}$
4. 定义基向量 $\vec{j}$
5. 将 ***v_ihat*** 定义为 x 乘以基向量 $\vec{i}$
6. 将 ***v_jhat*** 定义为 y 乘以基向量 $\vec{j}$
7. 使用 Matlibplot 绘制分解为 ***v_ihat*** 和 ***v_jhat*** 的向量 $\vec{v}$
   A. 创建变量 ***ax*** 来表示图形的坐标轴
   B. 使用 ***ax*** 和 *plot* 方法绘制原点（红点，位于 0,0）
   C. 使用 ***ax*** 和 *arrow* 方法绘制向量 ***v_ihat***（绿色虚线箭头，原点为 0,0）
   D. 使用 ***ax*** 和 *arrow* 方法绘制向量 ***v_jhat***（红色虚线箭头，原点为 v_ihat 的顶端）
   E. 使用 ***ax*** 和 *arrow* 方法绘制向量 _$\vec{v}$_（蓝色箭头，原点为 0,0）
   F. 设定 x 轴的格式
      a. 使用 *xlim* 方法设置上下限
      b. 使用 ***ax*** 和 *set_xticks* 方法设置主要刻度
   G. 设定 y 轴的格式
      a. 使用 *ylim* 方法设置上下限
      b. 使用 ***ax*** 和 *set_yticks* 方法设置主要刻度
   H. 使用 *grid* 方程创建网格线
   I. 使用 *show* 方程显示图形

In [1]:

```
# 导入 NumPy 和 Matplotlib
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

# 定义向量V
v = np.array([-1,2])

# 定义一个单位向量i作为基向量1
i_hat = np.array([1,0])

# 定义一个单位向量j作为基向量2
j_hat = np.array([0,1])

# Define v_ihat - as v[0](即x) multiplied by basis vector ihat
v_ihat = v[0] * i_hat

# Define v_jhat_t - as v[1](即y) multiplied by basis vector jhat
v_jhat = v[1] * j_hat

# Plot that graphically shows vector v (color='b') - whose position can be   绘制 这部分和向量Lab类似
# decomposed into v_ihat and v_jhat

# Creates axes of plot referenced 'ax'
ax = plt.axes()

# Plots red dot at origin (0,0)
ax.plot(0,0,'or')


# Plots vector v_ihat as dotted green arrow starting at origin 0,0
ax.arrow(0, 0, *v_ihat, color='g', linestyle='dotted', linewidth=2.5, head_width=0.30,
         head_length=0.35)

# Plots vector v_jhat as dotted red arrow starting at origin defined by v_ihat
ax.arrow(v_ihat[0], v_ihat[1], *v_jhat, color='r', linestyle='dotted', linewidth=2.5,
         head_width=0.30, head_length=0.35)

# Plots vector v as blue arrow starting at origin 0,0
ax.arrow(0, 0, *v, color='b', linewidth=2.5, head_width=0.30, head_length=0.35)


# Sets limit for plot for x-axis
plt.xlim(-4, 2)

# Set major ticks for x-axis
major_xticks = np.arange(-4, 2)
ax.set_xticks(major_xticks)


# Sets limit for plot for y-axis
plt.ylim(-2, 4)

# Set major ticks for y-axis
major_yticks = np.arange(-2, 4)
ax.set_yticks(major_yticks)

# Creates gridlines for only major tick marks
plt.grid(b=True, which='major')

# Displays final plot
plt.show()
```





png

使用*转换的*向量 $\vec{i_T}$ 和 $\vec{j_T}$ *转换*向量

在此部分，我们将绘制使用*转换的*向量 $\vec{i_T}$ 和 $\vec{j_T}$ *转换*向量 $\vec{v}$ 的结果。向量 $\vec{v}$、$\vec{i_T}$ 和 $\vec{j_T}$ 的定义如下所示。

$$\vec{v} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

$$\vec{i_T} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

$$\vec{j_T} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

TODO：使用向量 $\vec{i_T}$ 和 $\vec{j_T}$ 计算并绘制*转换的*向量 $\vec{v_T}$

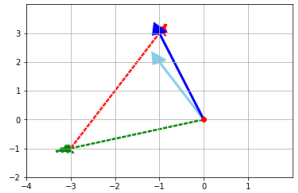
在此部分，你将创建*转换的*向量 $\vec{i_T}$ 和 $\vec{j_T}$，并使用它们根据上述*方程 1 转换*向量 $\vec{v}$

1. 通过将 x 和 y 替换为 3 和 1 定义向量 $\vec{i_T}$（请参阅 ***TODO 1.***:）。
2. 通过将 x 和 y 替换为 1 和 2 定义向量 $\vec{j_T}$（请参阅 ***TODO 2.***:）。
3. 通过将向量 $\vec{i_T}$ 和 $\vec{j_T}$ 相加定义向量 $\vec{v_T}$（请参阅 ***TODO 3.***:）。
4. 绘制向量 $\vec{v_T}$：复制向量 $\vec{v}$ 的 *ax.arrow(...)* 语句，并在 *ax.arrow(...)* 语句中设置 *color = 'b'*以将向量 $\vec{v_T}$ 绘制为蓝色向量（请参阅 ***TODO 4.***:）。

注意:

- 要*运行*代码：
  - 点击"保存"图标（位于顶部菜单栏的"*文件*"下的磁盘图标）以保存你的代码。
  - 选择"*内核*"和"*重新启动并运行所有*"以运行代码。

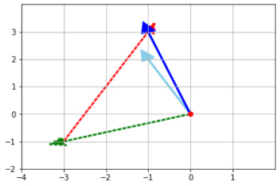
In [2]:
```
# Define vector v
v = np.array([-1, 2])

# DONE 1.: Define vector i_hat as transformed vector i_hat(ihat_t)
# where x=3 and y=1 instead of x=1 and y=0 替换x y值
ihat_t = np.array([3, 1])

# DONE 2.: Define vector j_hat as transformed vector j_hat(jhat_t)
# where x=1 and y=2 instead of x=0 and y=1 替换x y值
jhat_t = np.array([1, 2])

# Define v_ihat_t - as v[0](x) multiplied by transformed vector ihat
v_ihat_t = v[0] * ihat_t

# Define v_jhat_t - as v[1](y) multiplied by transformed vector jhat
v_jhat_t = v[1] * jhat_t

# DONE 3.: Define transformed vector v (v_t) as
# vector v_ihat_t added to vector v_jhat_t
v_t = v_ihat_t + v_jhat_t


# Plot that graphically shows vector v (color='skyblue') can be transformed
# into transformed vector v (v_trfm - color='b') by adding v[0]*transformed
# vector ihat to v[0]*transformed vector jhat


# Creates axes of plot referenced 'ax'
ax = plt.axes()

# Plots red dot at origin (0,0)
ax.plot(0,0,'or')


# Plots vector v_ihat_t as dotted green arrow starting at origin 0,0
ax.arrow(0, 0, *v_ihat_t, color='g', linestyle='dotted', linewidth=2.5, head_width=0.30,
         head_length=0.35)

# Plots vector v_jhat_t as dotted red arrow starting at origin defined by v_ihat
ax.arrow(v_ihat_t[0], v_ihat_t[1], *v_jhat_t, color='r', linestyle='dotted', linewidth=2.5,
         head_width=0.30, head_length=0.35)

# Plots vector v as blue arrow starting at origin 0,0
ax.arrow(0, 0, *v, color='skyblue', linewidth=2.5, head_width=0.30, head_length=0.35)

# TODO 4.: Plot transformed vector v (v_t) a blue colored vector(color='b') using
# vector v's ax.arrow() statement above as template for the plot
ax.arrow(0, 0, *v_t, color='b', linewidth=2.5, head_width=0.30, head_length=0.35)


# Sets limit for plot for x-axis
plt.xlim(-4, 2)

# Set major ticks for x-axis
major_xticks = np.arange(-4, 2)
ax.set_xticks(major_xticks)


# Sets limit for plot for y-axis
plt.ylim(-2, 4)

# Set major ticks for y-axis
major_yticks = np.arange(-2, 4)
ax.set_yticks(major_yticks)

# Creates gridlines for only major tick marks
plt.grid(b=True, which='major')

# Displays final plot
plt.show()
```



png

计算和绘制*转换的*向量 $\vec{v_T}$ 的解决方案

上述代码的输出应该与下面的输出相符。如果你需要任何帮助或想要检查你的答案，请点击此处查看解决方案 notebook。



计算和绘制*转换的*向量 $\vec{v_T}$ 的解决方案视频

你可以在**线性映射 Lab 解决方案**部分找到解决方案视频。你可能需要重新打开一个浏览器窗口，以便轻松地在向量 Lab Jupyter Notebook 和此 Lab 的解决方案视频之间轻松切换。

矩阵乘法

在此部分，我们将演示可以通过矩阵实现上述部分的相同向量转换结果（*方程 2*）。向量 $\vec{v}$ 和 $\vec{ij}$ 的定义如下所示。

$$\vec{v} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

$$\vec{ij} = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$$

TODO：矩阵乘法

在此部分，定义***转换的***向量 $\vec{v_T}$：使用函数 ***matmul*** 将 2x2 矩阵 $\vec{ij}$ 与向量 $\vec{v}$ 相乘。

1. 将下面的 **None** 替换为***转换的***向量 $\vec{v_T}$ 的定义代码：使用 ***matmul*** 函数将矩阵 $\vec{ij}$ 与向量 $\vec{v}$ 相乘（请参阅 ***TODO 1.***）

注意：

- 在导入 Numpy 软件包时，使用了别名 ***np***，因此在下面调用 ***matmul*** 函数时，确保使用别名 ***np***。
- 要*运行*代码：
  - 点击"保存"图标（位于顶部菜单栏的"*文件*"下的磁盘图标）以保存你的代码。
  - 选择"*内核*"和"*重新启动并运行所有*"以运行代码。

In [3]:
```
# 定义向量v
v = np.array([-1,2])

# 定义一个2*2的矩阵ij
ij = np.array([[3, 1],[1, 2]])

# TODO 1.: Demonstrate getting v_trfm by matrix multiplication
# by using matmul function to multiply 2x2 matrix ij by vector v
# to compute the transformed vector v (v_t)
v_t = np.matmul(ij, v)

# Prints vectors v, ij, and v_t
print("\nMatrix ij:", ij, "\nVector v:", v, "\nTransformed Vector v_t:", v_t, sep="\n")
```

```
Matrix ij:
[[3 1]
 [1 2]]

Vector v:
[-1  2]

Transformed Vector v_t:
[-1  3]
```

矩阵乘法的解决方案

上面关于*转换的*向量 $\vec{v_T}$ 的输出应该与下面的解决方案相符。注意，在 NumPy 中，向量以水平方式表示，因此向量 $\vec{v}$ 将定义为上述 *[-1 2]* 的形式。如果你需要任何帮助或想要检查你的答案，请点击此处查看解决方案 notebook。

你通过此矩阵乘法完成了*方程 2*中的计算过程（使用*转换的*向量 $\hat{\vec{i}_T}$ 和 $\hat{\vec{j}_T}$，如下所示）。

$$\begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \end{bmatrix} = -1 \begin{bmatrix} 3 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -1*3+2*1 \\ -1*1+2*2 \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$$

转换的 $\vec{v_T}$ 的值将如下所示，NumPy 会将其写成 *[-1 3]*：

$$tranformed\ \vec{v_T} = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$$

矩阵乘法的解决方案视频

你可以在**线性映射 Lab 解决方案**部分找到解决方案视频。你可能需要重新打开一个浏览器窗口，以便轻松地在向量 Lab Jupyter Notebook 和此 Lab 的解决方案视频之间轻松切换。