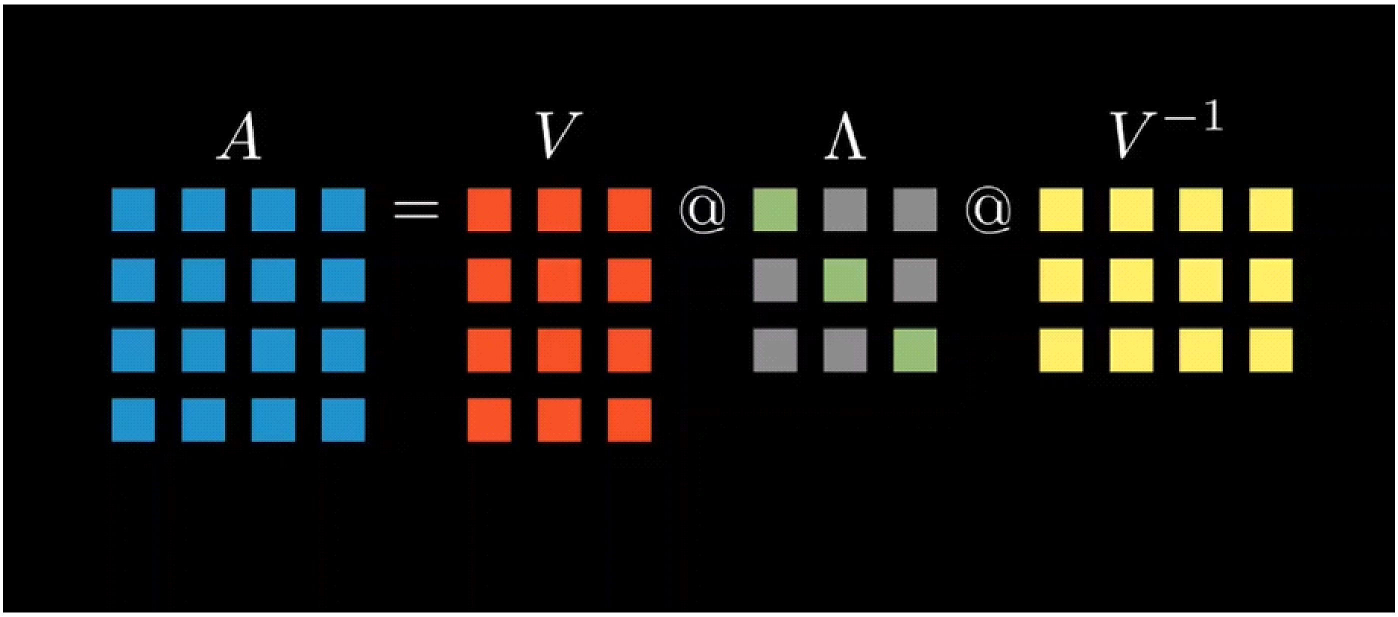
特征值分解（旋转-缩放-旋转）

2025年3月19日 10:41

特征值分解EVD

Eigenvalue Decomposition



公式$A=V\Lambda V^{-1}$

还是之前的例子：进行特征分解

设 $A=\begin{pmatrix}-2 & 1 & 1\\ 0 & 2 & 0\\ -4 & 1 & 3\end{pmatrix}$，求$A$的特征值与特征向量。

由之前的步骤：

① 构建由特征向量组成的矩阵V，验证P可逆

$V=\begin{bmatrix}\boxed{\begin{matrix}1\\0\\4\end{matrix}} & \begin{matrix}0\\1\\-1\end{matrix} & \begin{matrix}1\\0\\1\end{matrix}\end{bmatrix}$ ⇒ 行列式 $\det(V)=-3\neq 0$，可逆

② 构建由特征值构成的对角矩阵Λ

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

特征向量与对应的特征值位置保持一致

特征值从大到小排列

征值位置保持一致

③计算 V^{-1}

(1)计算余子式矩阵

由 $V=\begin{bmatrix}1&0&1\\0&1&0\\4&-1&1\end{bmatrix}$，可计算余子式:

$a_{11}=(-1)^{1+1}\begin{vmatrix}1&0\\-1&1\end{vmatrix}=1 \qquad a_{12}=0 \qquad a_{13}=\begin{vmatrix}-1&-3\\-1&0\end{vmatrix}=-4$

$a_{21}=-\begin{vmatrix}0&1\\-1&1\end{vmatrix}=-1 \qquad a_{22}=\begin{vmatrix}1&1\\4&1\end{vmatrix}=-3 \qquad a_{23}=\begin{vmatrix}1&0\\4&-1\end{vmatrix}=1$

$a_{31}=\begin{vmatrix}0&1\\1&0\end{vmatrix}=-1 \qquad a_{32}=-\begin{vmatrix}1&1\\0&0\end{vmatrix}=0 \qquad a_{33}=\begin{vmatrix}1&0\\0&1\end{vmatrix}=1$

$\therefore C=\begin{bmatrix}1&0&-4\\-1&-3&1\\-1&0&1\end{bmatrix}$

(2)计算伴随矩阵 C^*

$C^*=C^T=\begin{bmatrix}1&-1&-1\\0&-3&0\\-4&1&1\end{bmatrix}$

(3)计算 V^{-1}

$V^{-1}=\frac{1}{|V|}C^*=\frac{1}{-3}\begin{bmatrix}1&-1&-1\\0&-3&0\\-4&1&1\end{bmatrix}=\begin{bmatrix}-\frac{1}{3}&\frac{1}{3}&\frac{1}{3}\\0&1&0\\\frac{4}{3}&-\frac{1}{3}&-\frac{1}{3}\end{bmatrix}$

④最终结果:

旋转

缩放

$$\underset{A}{\begin{bmatrix}-2&1&1\\0&2&0\\-4&1&3\end{bmatrix}}=\underset{V}{\begin{bmatrix}1&0&1\\0&1&0\\4&-1&1\end{bmatrix}}\underset{\Lambda}{\begin{bmatrix}2&&\\&2&\\&&-1\end{bmatrix}}\underset{V^{-1}}{\begin{bmatrix}-\frac{1}{3}&\frac{1}{3}&\frac{1}{3}\\0&1&0\\\frac{4}{3}&-\frac{1}{3}&-\frac{1}{3}\end{bmatrix}}$$

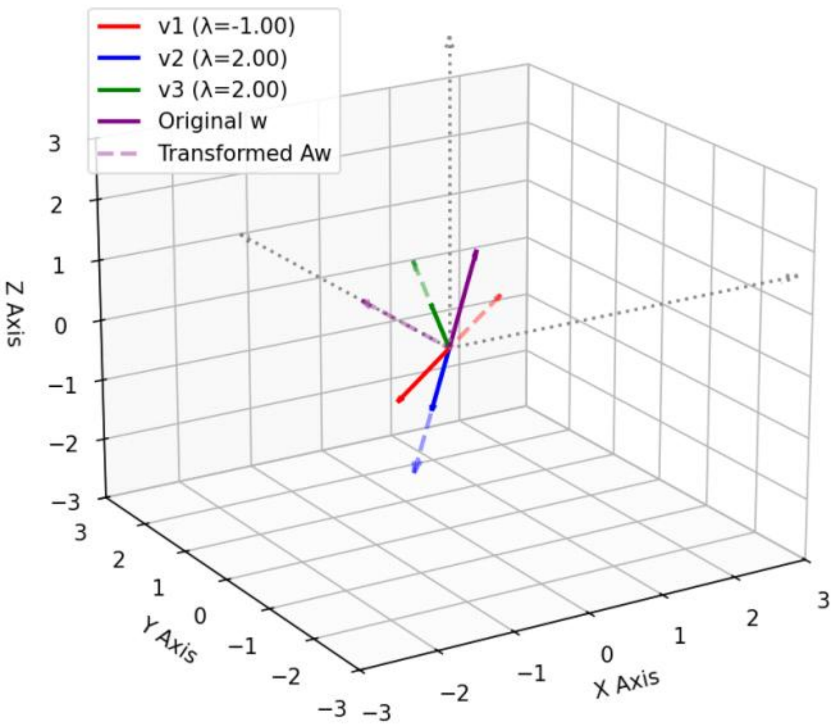
用代码绘制结果

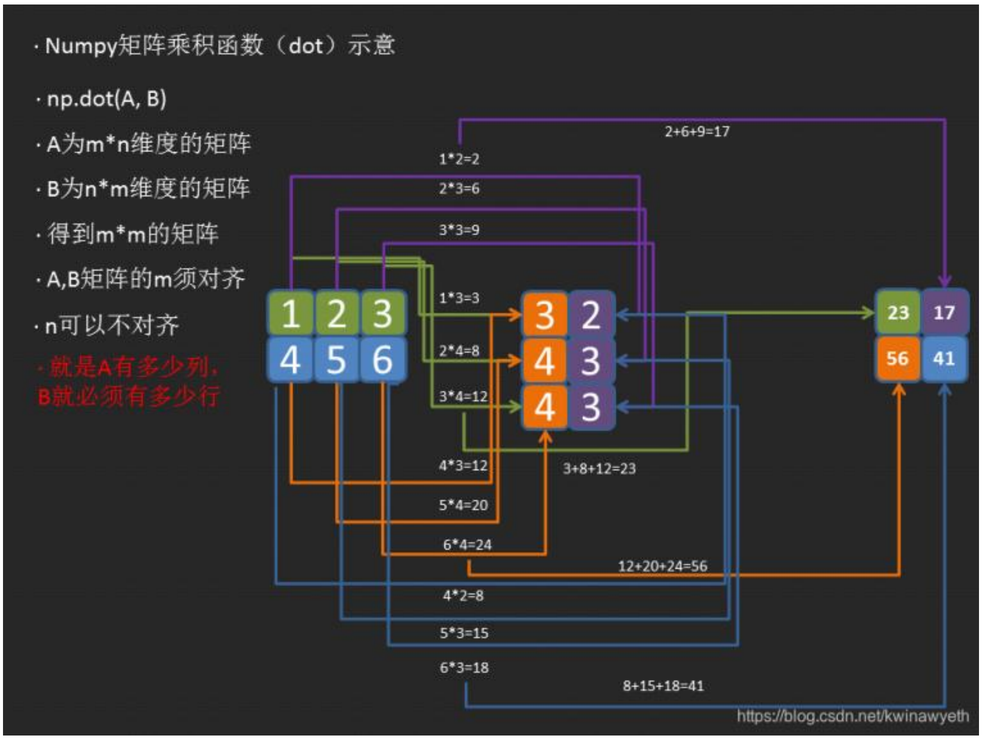
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
# 定义矩阵 A
```

```python
A = np.array([[-2, 1, 1],
              [0, 2, 0],
              [-4, 1, 3]])
# 计算特征值和特征向量
eigenvalues, eigenvectors = np.linalg.eig(A)
# 提取特征向量（实部）
v1 = eigenvectors[:, 0].real
v2 = eigenvectors[:, 1].real
v3 = eigenvectors[:, 2].real
# 计算变换后的向量
Av1 = A @ v1
Av2 = A @ v2
Av3 = A @ v3
# 定义示例向量 w 并计算变换后的结果
w = np.array([1, 1, 1])
Aw = A @ w
# 将w分解到特征向量基
coefficients = np.linalg.inv(eigenvectors) @ w
# 创建三维图形
fig = plt.figure(figsize=(12, 10))
ax = fig.add_subplot(111, projection='3d')
# 绘制坐标轴
ax.quiver(0, 0, 0, 5, 0, 0, color='gray', arrow_length_ratio=0.05, linestyle=':')
ax.quiver(0, 0, 0, 0, 5, 0, color='gray', arrow_length_ratio=0.05, linestyle=':')
ax.quiver(0, 0, 0, 0, 0, 5, color='gray', arrow_length_ratio=0.05, linestyle=':')
# 绘制特征向量及其变换
colors = ['red', 'blue', 'green']
for i, (vec, av, λ) in enumerate(zip([v1, v2, v3], [Av1, Av2, Av3], eigenvalues)):
    # 原始特征向量
    ax.quiver(0, 0, 0, *vec,
              color=colors[i],
              label=f'v{i+1} (λ={λ:.2f})',
              arrow_length_ratio=0.1,
              linewidth=2)

    # 变换后的向量
    ax.quiver(0, 0, 0, *av,
              color=colors[i],
              alpha=0.4,
              linestyle='--',
              arrow_length_ratio=0.1,
              linewidth=2)
# 绘制示例向量及其变换
ax.quiver(0, 0, 0, *w,
          color='purple',
          label='Original w',
          arrow_length_ratio=0.1,
          linewidth=2)
ax.quiver(0, 0, 0, *Aw,
          color='purple',
          alpha=0.4,
          linestyle='--',
```

```python
              label='Transformed Aw',
              arrow_length_ratio=0.1,
              linewidth=2)
# 设置坐标轴
ax.set_xlim([-3, 3])
ax.set_ylim([-3, 3])
ax.set_zlim([-3, 3])
ax.set_xlabel('X Axis')
ax.set_ylabel('Y Axis')
ax.set_zlabel('Z Axis')
ax.set_title('3D Visualization of Matrix Transformation\n'
             'Eigenvectors Show Directional Scaling', pad=20)
# 调整视角
ax.view_init(elev=25, azim=-45)
# 添加图例
ax.legend(loc='upper left', bbox_to_anchor=(0.05, 0.95))
plt.show()
# 打印分解信息
print("="*50)
print("特征值:", eigenvalues)
print("\n特征向量矩阵（每列为一个特征向量）:")
print(eigenvectors)
print("\n示例向量 w 的分解系数:", coefficients.round(2))
print("\n验证变换结果:")
print("Aw（直接计算）:", Aw)
print("Aw（通过特征分解）:",
      sum(coefficients[i] * eigenvalues[i] * eigenvectors[:, i] for i in range(3)).round(2))
```






```python
import numpy as np
# 设置打印选项（保留3位小数，禁用科学计数法）
np.set_printoptions(precision=3, suppress=True)
# 定义原始矩阵
A = np.array([[-2, 1, 1],
              [0, 2, 0],
              [-4, 1, 3]])
# 特征值分解
eigenvalues, eigenvectors = np.linalg.eig(A)
D = np.diag(eigenvalues)
V = eigenvectors
V_inv = np.linalg.inv(V)
# 重构原始矩阵
A_reconstructed = V @ D @ V_inv
# 输出结果
print("特征值：\n", eigenvalues)
print("\n特征向量矩阵V：\n", V)
print("\n对角矩阵D：\n", D)
print("\n特征向量逆矩阵V_inv：\n", V_inv)
print("\n重构矩阵：\n", A_reconstructed)
print("\n原始矩阵与重构矩阵差异：\n", A - A_reconstructed)
```

```
特征值:
 [-1.  2.  2.]

特征向量矩阵V:
 [[-0.707 -0.243  0.302]
 [ 0.     0.     0.905]
 [-0.707 -0.97   0.302]]

对角矩阵D:
 [[-1.  0.  0.]
 [ 0.  2.  0.]
 [ 0.  0.  2.]]

特征向量逆矩阵V_inv:
 [[-1.886  0.471  0.471]
 [ 1.374 -0.    -1.374]
 [ 0.     1.106  0.   ]]

重构矩阵:
 [[-2.  1.  1.]
 [ 0.  2.  0.]
 [-4.  1.  3.]]

原始矩阵与重构矩阵差异:
 [[ 0.  0. -0.]
 [ 0.  0.  0.]
 [ 0.  0. -0.]]
```