16 Dive into Singular Value Decomposition

深入奇异值分解

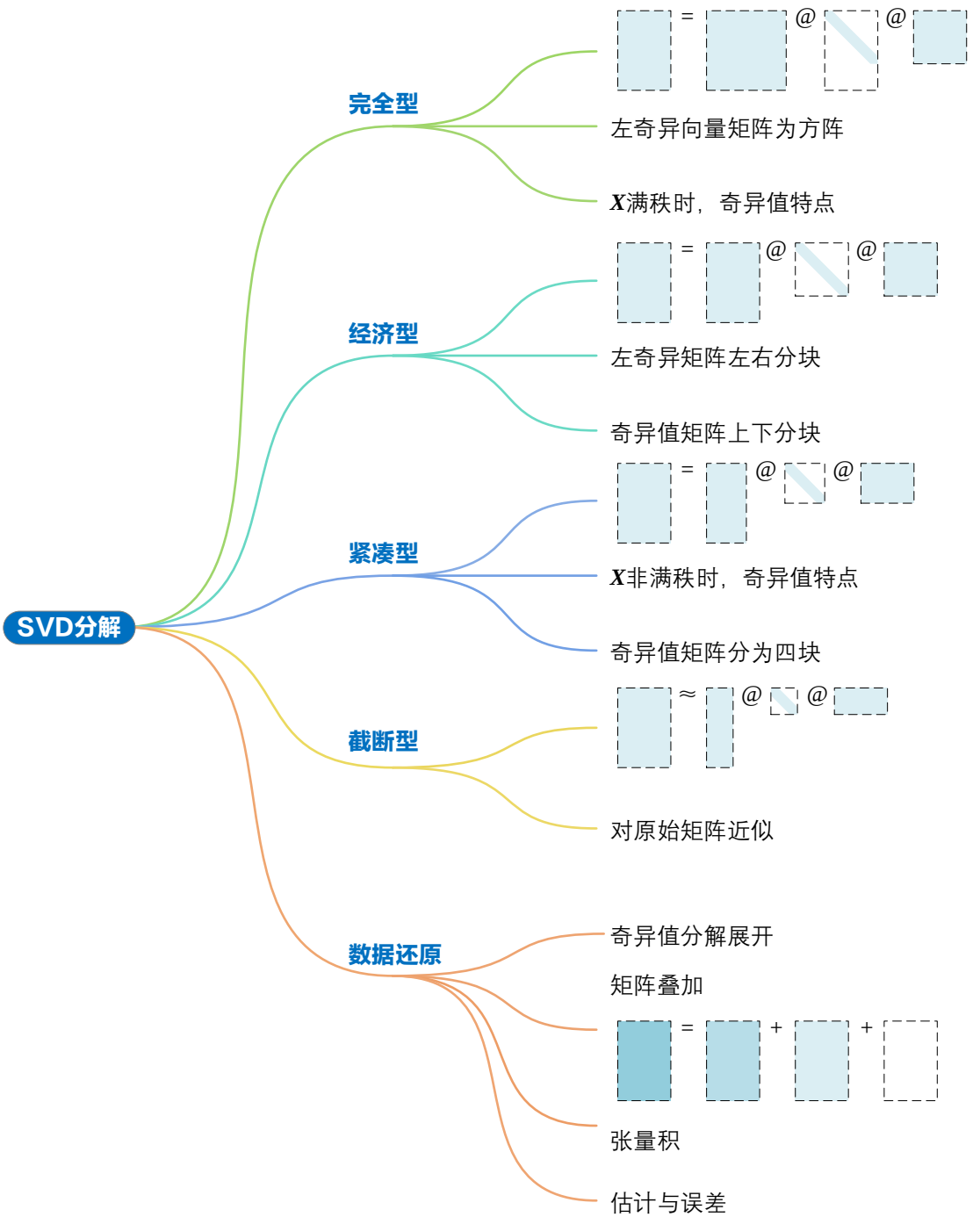
四种形式与数据还原

人不过是一根芦苇，世界最脆弱的生灵；但是，人是会思考的芦苇。

Man is but a reed, the most feeble thing in nature, but he is a thinking reed.

—— 布莱兹·帕斯卡 (Blaise Pascal) | 法国哲学家、科学家 | 1623 ~ 1662

- `matplotlib.pyplot.quiver()` 绘制箭头图
- `numpy.linspace()` 在指定的间隔内，返回固定步长的数据
- `numpy.linalg.svd()` 进行 SVD 分解
- `numpy.diag()` 以一维数组的形式返回方阵的对角线元素，或将一维数组转换成对角阵



16.1 完全型：U为方阵

上一章介绍过奇异值分解有四种形式：

- **完全型** (full)；
- **经济型** (economy-size, thin)；
- **紧凑型** (compact)；
- **截断型** (truncated)。

本章将深入介绍这四种奇异值分解。

首先回顾完全型 SVD 分解。图 1 所示为完全型 SVD 分解示意图。一般情况，丛书常见的数据矩阵 $\boldsymbol{X}$ 形状 $n > D$，即细高型。

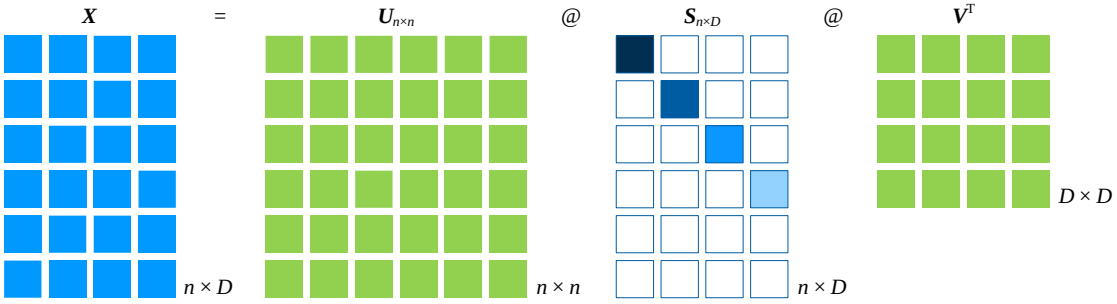


图 1. 完全型 SVD 分解

如图 1 所示，对于完全型 SVD 分解，左奇异向量矩阵 $\boldsymbol{U}$ 为方阵，形状为 $n \times n$。$\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n]$ 是张成 $\mathbb{R}^n$ 空间的规范正交基。

$\boldsymbol{S}_{n\times D}$ 的形状和 $\boldsymbol{X}$ 相同，为 $n \times D$。虽然 $\boldsymbol{S}_{n\times D}$ 也是对角阵，但是它不是方阵。

如果 $\boldsymbol{X}$ 满秩，$\text{rank}(\boldsymbol{X}) = D$，$\boldsymbol{S}$ 的主对角线元素 (奇异值 s_i) 一般排列顺序为：

$$s_1 \geq s_2 \geq \cdots s_D > 0 \tag{1}$$

右奇异向量矩阵 $\boldsymbol{V}$ 形状为 $D \times D$。$\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n]$ 是张成 $\mathbb{R}^D$ 空间的规范正交基。

图 2 所示为矩阵 $\boldsymbol{X}_{6\times 4}$ 进行完全 SVD 分解的结果热图。

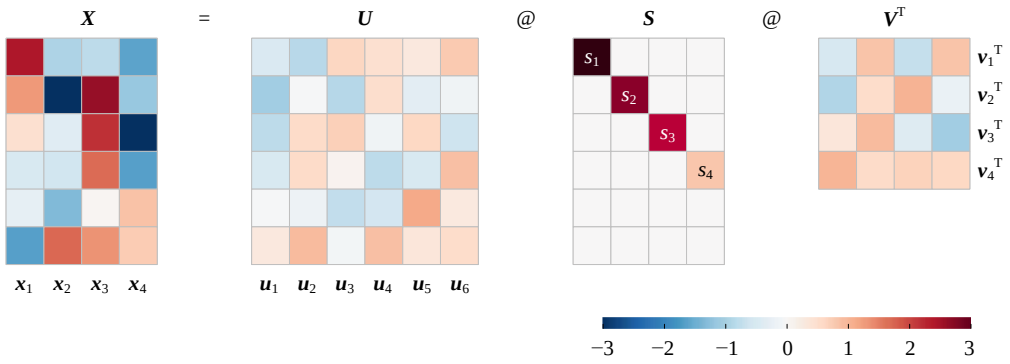


图 2. 数据 $\boldsymbol{X}$ 完全型 SVD 分解矩阵热图

Bk4_Ch16_01.py 中 Bk4_Ch16_01_A 部分绘制图 2。

```
# Bk4_Ch16_01_A

import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
PRECISION = 3

def svd(X,full_matrices):

    U, s, Vt = np.linalg.svd(X,full_matrices = full_matrices)

    # Put the vector singular values into a padded matrix

    if full_matrices:
        S = np.zeros(X.shape)
        np.fill_diagonal(S, s)
    else:
        S = np.diag(s)

    # Rounding for display
    return np.round(U, PRECISION), np.round(S, PRECISION), np.round(Vt.T, PRECISION)

# Repeatability
np.random.seed(1)

# Generate random matrix
X = np.random.randn(6, 4)

# manipulate X and reduce rank to 3
# X[:,3] = X[:,0] + X[:,1]
all_max = 2; all_min = -2

#%% full

U, S, V = svd(X, full_matrices = True)

fig, axs = plt.subplots(1, 4, figsize=(12, 3))

plt.sca(axs[0])
ax = sns.heatmap(X,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$X$')

plt.sca(axs[1])
```

```
ax = sns.heatmap(U,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$U$')

plt.sca(axs[2])
ax = sns.heatmap(S,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$S$')

plt.sca(axs[3])
ax = sns.heatmap(V.T,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$V^T$')
```

16.2 经济型：S去掉零矩阵，变方阵

在完全型 SVD 分解基础上，长方对角阵 $\boldsymbol{S}_{n\times D}$ 上下分块成一个对角方阵和一个零矩阵 $\boldsymbol{O}$：

$$\boldsymbol{S}_{n\times D}=\begin{bmatrix} s_1 & & & \\ & s_2 & & \\ & & \ddots & \\ & & & s_D \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}=\begin{bmatrix} \boldsymbol{S}_{D\times D} \\ \boldsymbol{O}_{(n-D)\times D} \end{bmatrix} \tag{2}$$

将 $\boldsymbol{U}_{n\times n}$ 写成左右分块矩阵 $[\boldsymbol{U}_{n\times D}, \boldsymbol{U}_{n\times(n-D)}]$，其中 $\boldsymbol{U}_{n\times D}$ 和 $\boldsymbol{X}$ 形状相同。

这样完全型 SVD 分解可以简化成经济型 SVD 分解：

$$\begin{aligned} \boldsymbol{X}_{n\times D} &= \begin{bmatrix} \boldsymbol{U}_{n\times D} & \boldsymbol{U}_{n\times(n-D)} \end{bmatrix} \begin{bmatrix} \boldsymbol{S}_{D\times D} \\ \boldsymbol{O}_{(n-D)\times D} \end{bmatrix} \boldsymbol{V}^{\mathrm{T}} \\ &= \left(\boldsymbol{U}_{n\times D}\boldsymbol{S}_{D\times D} + \boldsymbol{U}_{n\times(n-D)}\boldsymbol{O}_{(n-D)\times D} \right) \boldsymbol{V}^{\mathrm{T}} \\ &= \boldsymbol{U}_{n\times D}\boldsymbol{S}_{D\times D}\boldsymbol{V}^{\mathrm{T}} \end{aligned} \tag{3}$$

图 3 所示为完全型 SVD 简化成经济型 SVD 的示意图。比较完全型和经济型 SVD，分解结果中唯一不变的就是矩阵 $\boldsymbol{V}$，它一直保持方阵形态。

前文提到过，$\boldsymbol{U}_{n\times n}$ 为规范正交基，但是经济型 SVD 分解仅仅保留 $\boldsymbol{U}_{n\times D}$，而 $\boldsymbol{U}_{n\times(n-D)}$ 则完全被删掉。从向量空间角度来讲，$\boldsymbol{U}_{n\times D}$ 和 $\boldsymbol{U}_{n\times(n-D)}$ 有怎样的差异？

这是本书最后要回答的问题。

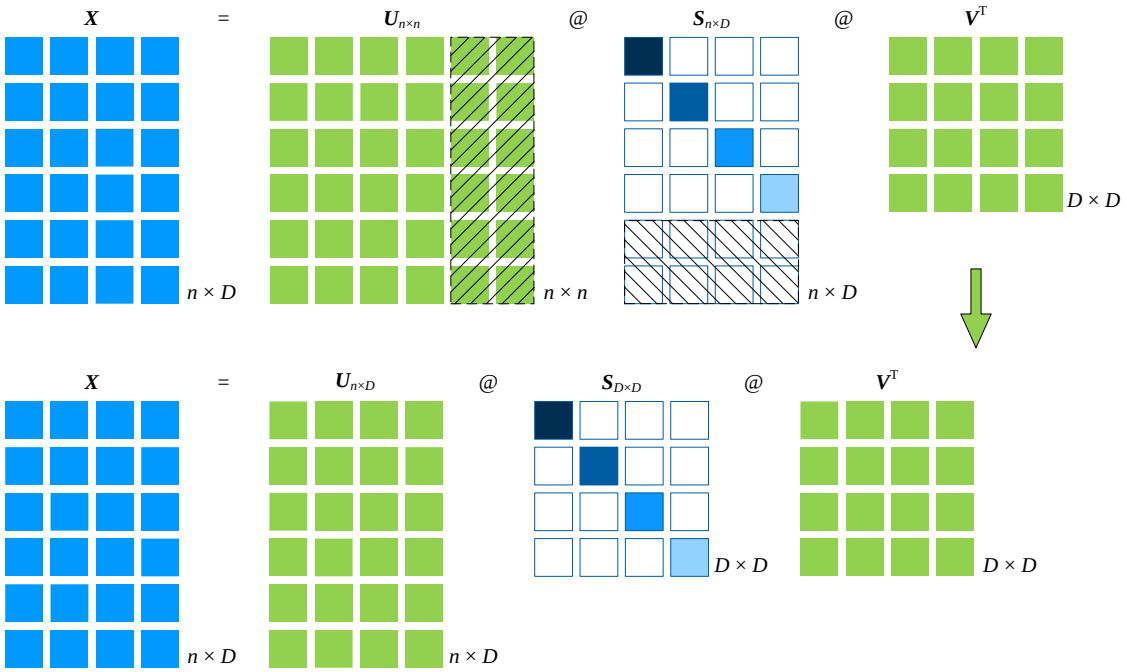


图 3. 从完全型到经济型

图 4 和图 5 比较完全型和经济型 SVD 分解结果热图。图 4 中阴影部分为删除的矩阵子块。

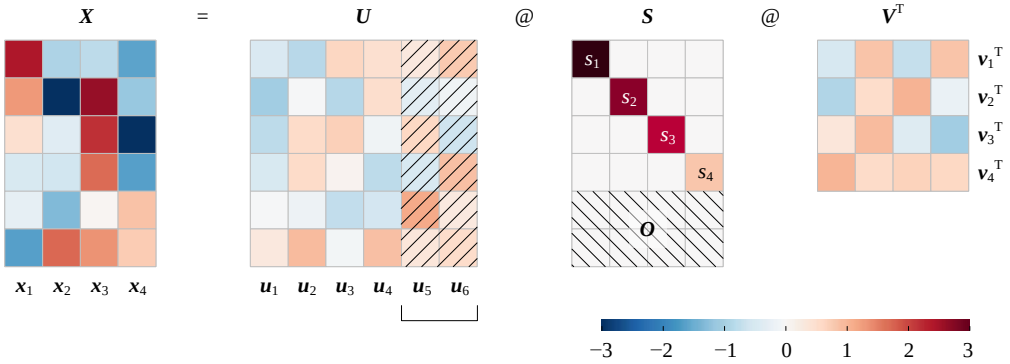


图 4. 数据 $\boldsymbol{X}$ 完全型 SVD 分解分块热图

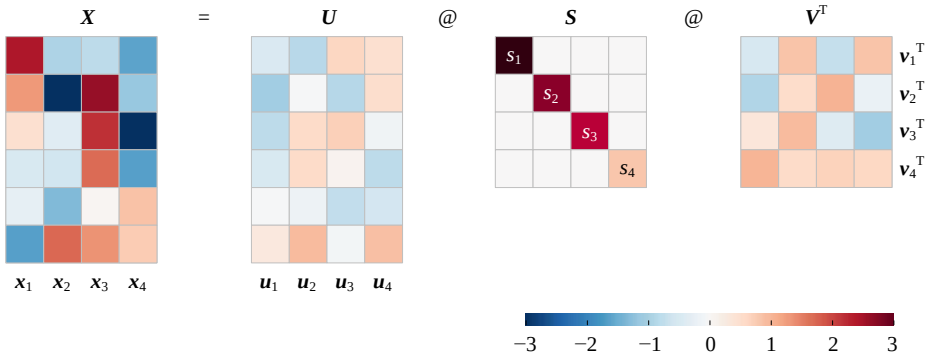
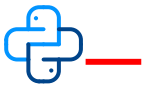


图 5. 数据 $\boldsymbol{X}$ 经济型 SVD 分解热图



Bk4_Ch16_01.py 中 Bk4_Ch16_01_B 部分绘制图 5。

```
# Bk4_Ch16_01_B

#%% Economy-size, thin

U, S, V = svd(X, full_matrices = False)

fig, axs = plt.subplots(1, 4, figsize=(12, 3))

plt.sca(axs[0])
ax = sns.heatmap(X,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$X$')


plt.sca(axs[1])
ax = sns.heatmap(U,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$U$')

plt.sca(axs[2])
ax = sns.heatmap(S,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$S$')

plt.sca(axs[3])
ax = sns.heatmap(V.T,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$V^T$')
```

16.3 紧凑型：非满秩

特别地，如果 $\text{rank}(\boldsymbol{X}) = r < D$，奇异值 s_i 满足：

$$s_1 \geq s_2 \geq \cdots s_r > 0, \quad s_{r+1} = s_{r+2} = \cdots s_D = 0 \tag{4}$$

奇异值矩阵 $\boldsymbol{S}$ 可以分成四个子块：

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{S}_{r\times r} & \boldsymbol{O}_{r\times(D-r)} \\ \boldsymbol{O}_{(D-r)\times r} & \boldsymbol{O}_{(D-r)\times(D-r)} \end{bmatrix} \tag{5}$$

上式中，矩阵 $\boldsymbol{S}_{r\times r}$ 对角线元素为非 0 奇异值。

将 (5) 代入完全型 SVD 分解 (3)，整理得到：

$$
\begin{aligned}
\boldsymbol{X}_{n\times D} &= \begin{bmatrix} \boldsymbol{U}_{n\times r} & \boldsymbol{U}_{n\times(D-r)} \end{bmatrix} \begin{bmatrix} \boldsymbol{S}_{r\times r} & \boldsymbol{O}_{r\times(D-r)} \\ \boldsymbol{O}_{(D-r)\times r} & \boldsymbol{O}_{(D-r)\times(D-r)} \end{bmatrix} \begin{bmatrix} (\boldsymbol{V}_{D\times r})^{\mathrm{T}} \\ (\boldsymbol{V}_{D\times(D-r)})^{\mathrm{T}} \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{U}_{n\times r}\boldsymbol{S}_{r\times r} & \boldsymbol{O}_{n\times(D-r)} \end{bmatrix} \begin{bmatrix} (\boldsymbol{V}_{D\times r})^{\mathrm{T}} \\ (\boldsymbol{V}_{D\times(D-r)})^{\mathrm{T}} \end{bmatrix} \\
&= \boldsymbol{U}_{n\times r}\boldsymbol{S}_{r\times r}(\boldsymbol{V}_{D\times r})^{\mathrm{T}}
\end{aligned} \tag{6}
$$

图 6 所示为从经济型 SVD 分解得到紧缩型 SVD 分解的示意图。再次强调，只有 $\boldsymbol{X}$ 为非满秩情况下，才存在紧缩型 SVD 分解。紧缩型 SVD 分解中，$\boldsymbol{U}$ 和 $\boldsymbol{V}$ 都不是方阵。

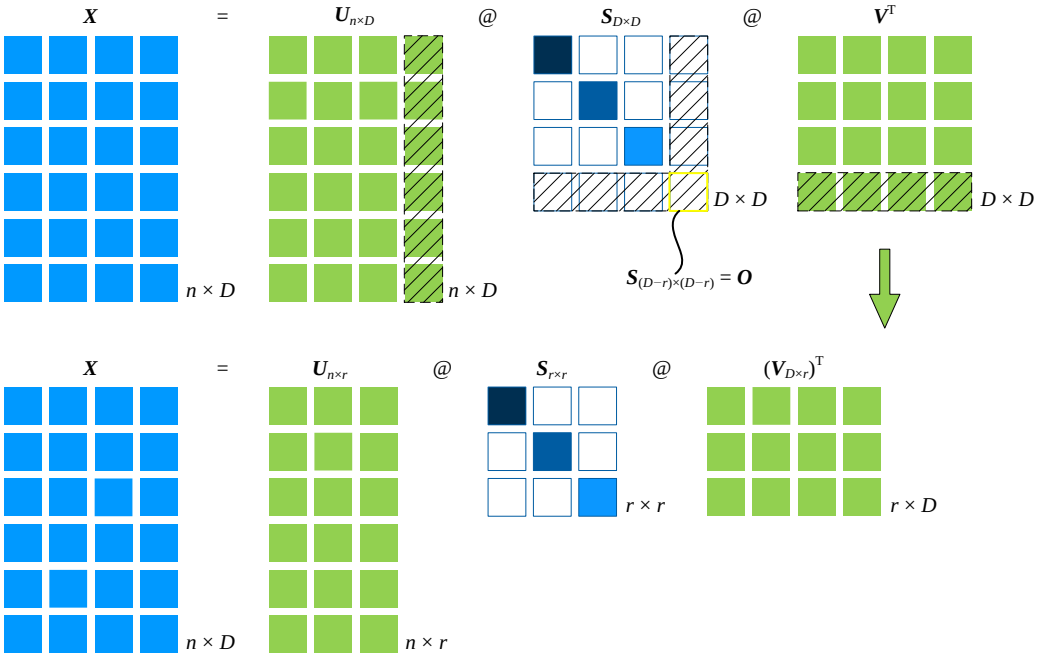


图 6. 从经济型到紧缩型

为了展示紧凑型 SVD 分解，我们用 $\boldsymbol{X}$ 第一、二列数据之和替代 $\boldsymbol{X}$ 矩阵第四列，即 $\boldsymbol{x}_4 = \boldsymbol{x}_1 + \boldsymbol{x}_2$。这样 $\boldsymbol{X}$ 矩阵列向量线性相关，rank($\boldsymbol{X}$) = 3，而 $s_4 = 0$。

图 7 和图 8 比较经济型和紧凑型 SVD 分解。

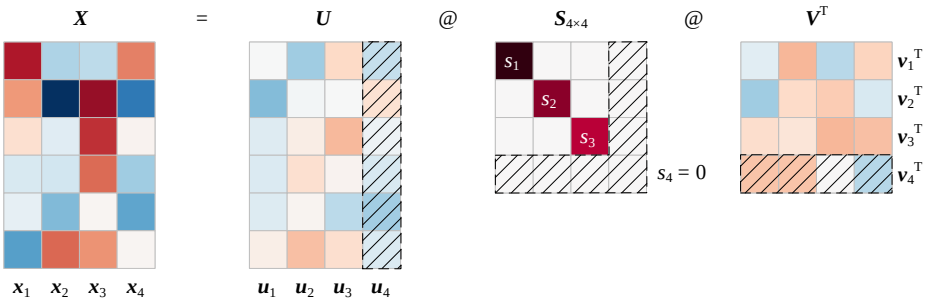


图 7. 数据 $\boldsymbol{X}$ 经济型 SVD 分解热图

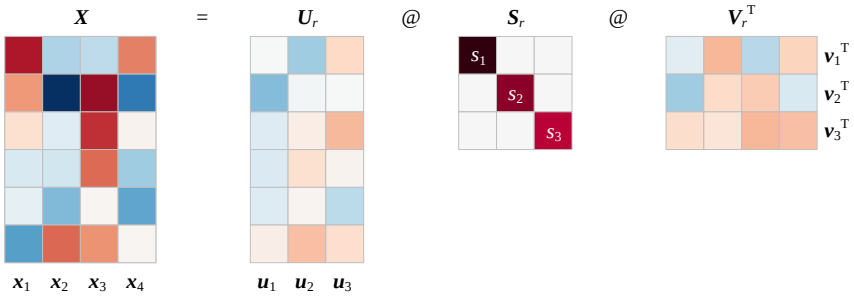


图 8. 数据 $\boldsymbol{X}$ 紧凑型 SVD 分解热图

Bk4_Ch16_01.py 中 Bk4_Ch16_01_C 部分绘制图 7。

```
# Bk4_Ch16_01_C

#%% Compact

import copy

X_rank_3 = copy.deepcopy(X);
# manipulate X and reduce rank to 3
X_rank_3[:,3] = X[:,0] + X[:,1]

U_rank_3, S_rank_3, V_rank_3 = svd(X_rank_3, full_matrices = False)

fig, axs = plt.subplots(1, 4, figsize=(12, 3))

plt.sca(axs[0])
ax = sns.heatmap(X_rank_3,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$X$')


plt.sca(axs[1])
ax = sns.heatmap(U_rank_3,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$U$')

plt.sca(axs[2])
ax = sns.heatmap(S_rank_3,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$S$')

plt.sca(axs[3])
ax = sns.heatmap(V_rank_3.T,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$V^T$')
```

16.4 截断型：近似

假设 $\text{rank}(\boldsymbol{X}) = r \leqslant D$，取紧缩型奇异值分解中前 p 个奇异值 ($p < r$) 对应的 $\boldsymbol{U}$、$\boldsymbol{S}$、$\boldsymbol{V}$ 矩阵成分，用它们还原原始数据就是截断型奇异值分解：

$$\boldsymbol{X}_{n\times D} \approx \hat{\boldsymbol{X}}_{n\times D} = \boldsymbol{U}_{n\times p}\boldsymbol{S}_{p\times p}\left(\boldsymbol{V}_{D\times p}\right)^{\mathrm{T}} \tag{7}$$

请读者自行补足上式中矩阵分块和对应的乘法运算。

图 9 所示为从紧缩型 SVD 分解到截断型 SVD 分解的示意图。

注意，(7) 不是等号，也就是截断型奇异值分解不能完全还原原始数据。换句话，截断型奇异值分解是对原矩阵 $\boldsymbol{X}$ 的一种近似。

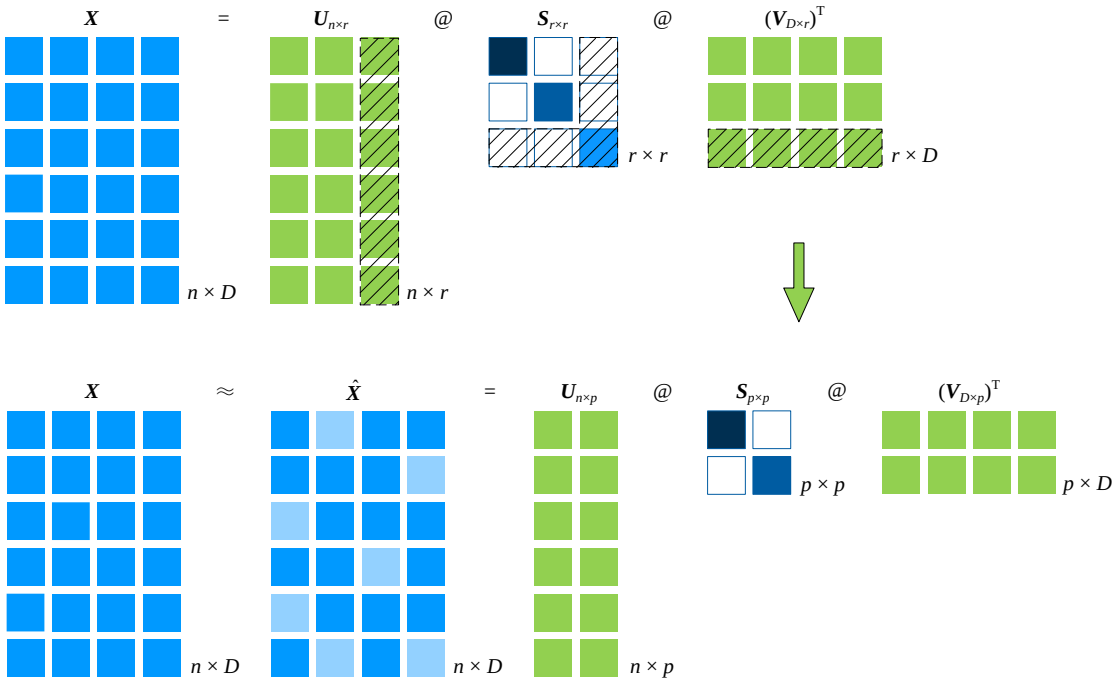


图 9. 从紧缩型到截断型

图 10 所示为 SVD 截断型分解热图计算过程，可以发现 $\boldsymbol{X}_{n\times D}$ 和 $\hat{\boldsymbol{X}}_{n\times D}$ 两幅热图存在一定误差。本章最后讲解矩阵数据还原和误差。

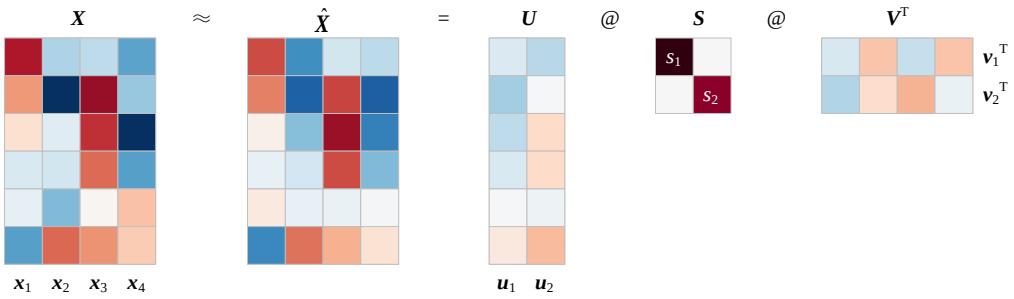


图 10. 采用截断型 SVD 分解还原数据运算热图

Bk4_Ch16_01.py 中 Bk4_Ch16_01_D 绘制图 10。

```
# Bk4_Ch16_01_D
#%% Truncated

num_p = 2;

U_truc = U[:,0:num_p]

S_truc = S[0:num_p,0:num_p]

V_truc = V[:,0:num_p]

X_hat = U_truc@S_truc@(V_truc.T)

# reproduce
fig, axs = plt.subplots(1, 4, figsize=(12, 3))

plt.sca(axs[0])
ax = sns.heatmap(X_hat,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$\hat{X}$')

plt.sca(axs[1])
ax = sns.heatmap(U_truc,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$U$')

plt.sca(axs[2])
ax = sns.heatmap(S_truc,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$S$')

plt.sca(axs[3])
ax = sns.heatmap(V_truc.T,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$V^T$')

# Error
fig, axs = plt.subplots(1, 3, figsize=(12, 3))

plt.sca(axs[0])
ax = sns.heatmap(X,cmap='RdBu_r',vmax = all_max,vmin = all_min,
```

```
                  cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$X$')


plt.sca(axs[1])
ax = sns.heatmap(X_hat,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                  cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$\hat{X}$')

plt.sca(axs[2])
ax = sns.heatmap(X - X_hat,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                  cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$E = X - \hat{X}$')
```

16.5 数据还原

上一章介绍过，经济型 SVD 分解可以展开写作：

$$\boldsymbol{X}_{n\times D} = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_D \end{bmatrix} \begin{bmatrix} s_1 & & & \\ & s_2 & & \\ & & \ddots & \\ & & & s_D \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1^{\mathrm{T}} \\ \boldsymbol{v}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{v}_D^{\mathrm{T}} \end{bmatrix}$$
$$= \underbrace{s_1\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}}}_{\hat{\boldsymbol{X}}_1} + \underbrace{s_2\boldsymbol{u}_2\boldsymbol{v}_2^{\mathrm{T}}}_{\hat{\boldsymbol{X}}_2} + \cdots + \underbrace{s_D\boldsymbol{u}_D\boldsymbol{v}_D^{\mathrm{T}}}_{\hat{\boldsymbol{X}}_D} \tag{8}$$

再次注意，上式中奇异值从大到小排列，即 $s_1 \geqslant s_2 \geqslant \ldots s_D$。图 11 所示为 SVD 分解展开示意图。图 12 所示上述运算热图，$D = 4$。

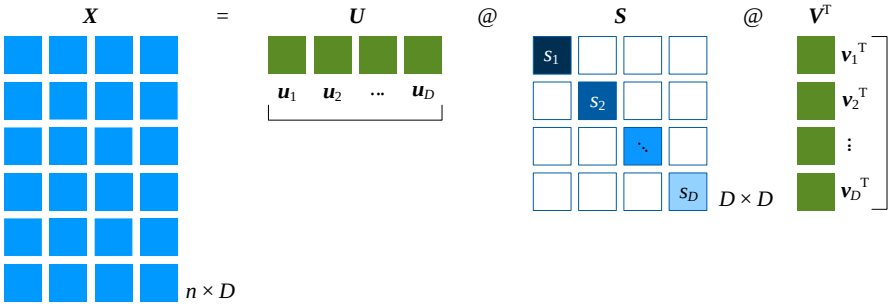


图 11. SVD 分解展开示意图

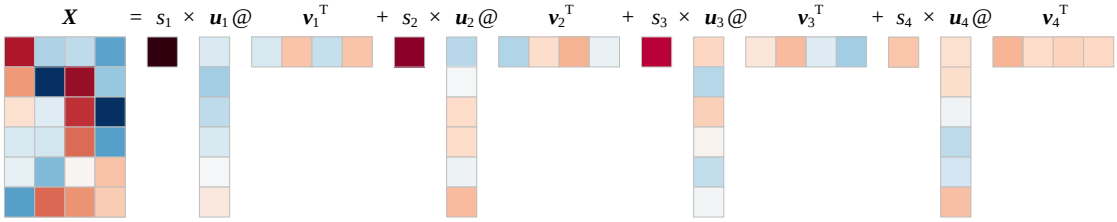


图 12. SVD 分解展开计算热图

组成部分

定义矩阵 $\boldsymbol{X}_j$ 为：

$$\boldsymbol{X}_j = s_j \boldsymbol{u}_j \boldsymbol{v}_j^{\mathrm{T}} \tag{9}$$

矩阵 $\boldsymbol{X}_j$ 形状和 $\boldsymbol{X}$ 相同。图 13 所示为矩阵 $\boldsymbol{X}_j$ (j = 1, 2, 3, 4) 四幅热图计算过程。

观察图 13 每幅矩阵 $\boldsymbol{X}_j$ 热图不难发现，矩阵 $\boldsymbol{X}_j$ 自身列向量之间存在倍数关系。也就是说，矩阵 $\boldsymbol{X}_j$ 的秩为 1，即 rank($\boldsymbol{X}_j$) = 1。

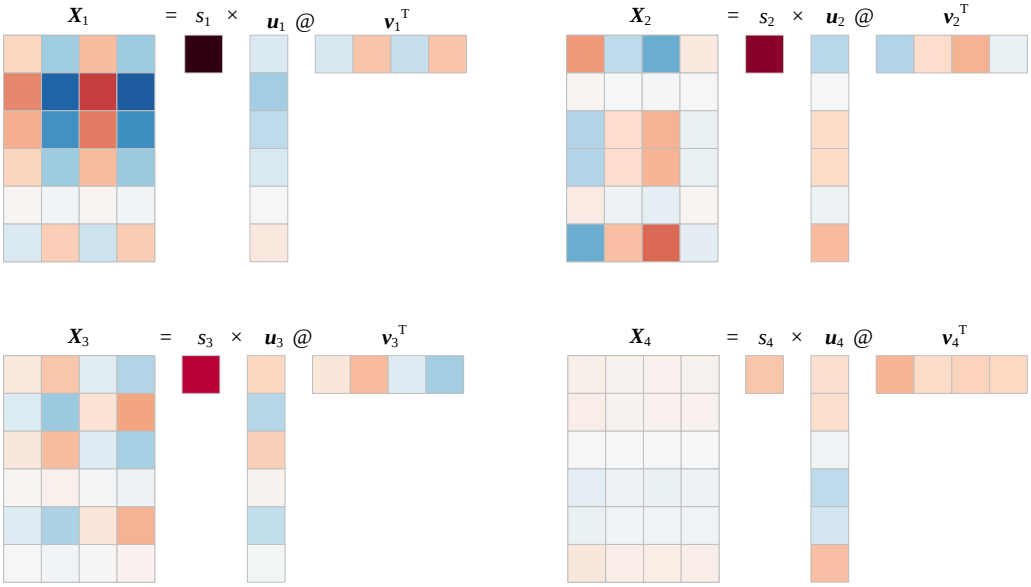


图 13. 还原数据的叠加成分

还原

(9) 代入 (8) 得到：

$$\boldsymbol{X}_{n\times D} = \boldsymbol{X}_1 + \boldsymbol{X}_2 + \cdots + \boldsymbol{X}_D \tag{10}$$

当 j = 1 ~ D 时，将 $\boldsymbol{X}_j$ 一层层叠加最后还原原始数据矩阵 $\boldsymbol{X}$，如图 14 所示。

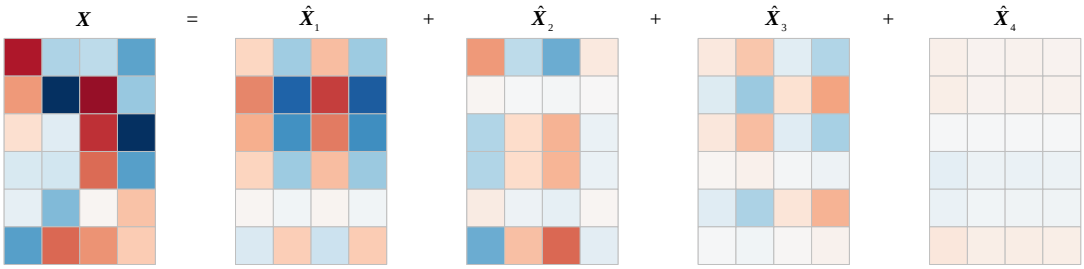


图 14. 还原原始数据

张量积

利用向量张量积，(8) 可以写成：

$$\begin{aligned}\boldsymbol{X} &= \underbrace{s_1\boldsymbol{u}_1 \otimes \boldsymbol{v}_1}_{\hat{\boldsymbol{X}}_1} + \underbrace{s_2\boldsymbol{u}_2 \otimes \boldsymbol{v}_2}_{\hat{\boldsymbol{X}}_2} + \cdots + \underbrace{s_D\boldsymbol{u}_D \otimes \boldsymbol{v}_D}_{\hat{\boldsymbol{X}}_D} \\ &= \hat{\boldsymbol{X}}_1 + \hat{\boldsymbol{X}}_2 + \cdots + \hat{\boldsymbol{X}}_D\end{aligned} \tag{11}$$

图 15 所示为张量积 $\boldsymbol{u}_j \otimes \boldsymbol{v}_j$ 计算热图，可以发现热图色差并不明显。这说明 $\boldsymbol{u}_j \otimes \boldsymbol{v}_j$ 本身并不能区分 $\boldsymbol{X}_j$。

然后再用奇异值 s_j 乘以对应张量积 $\boldsymbol{u}_j \otimes \boldsymbol{v}_j$ 得到 $\boldsymbol{X}_j$，具体如图 16 所示。可以发现 $\boldsymbol{X}_1$ 热图色差最明显。也就是说，奇异值 s_j 的大小决定了成分的重要性。

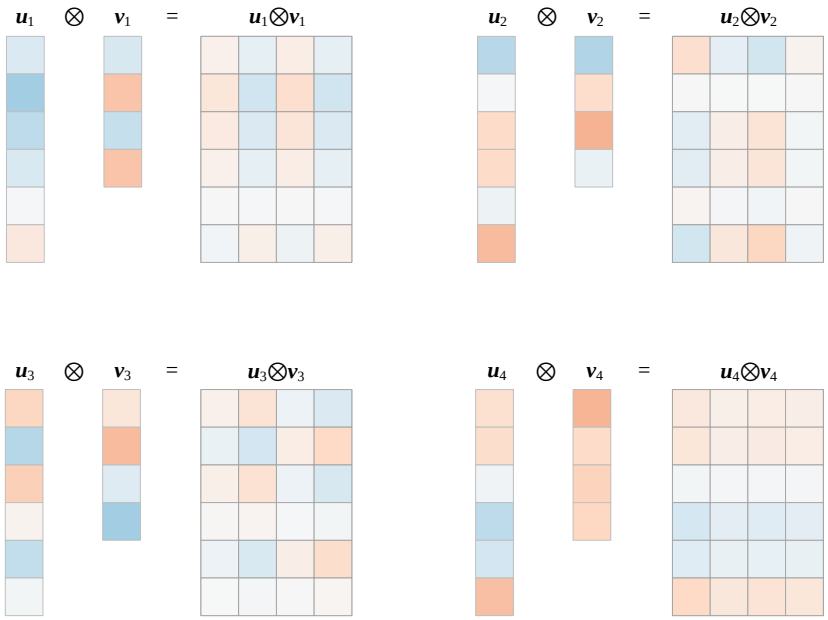


图 15. 向量张量热图

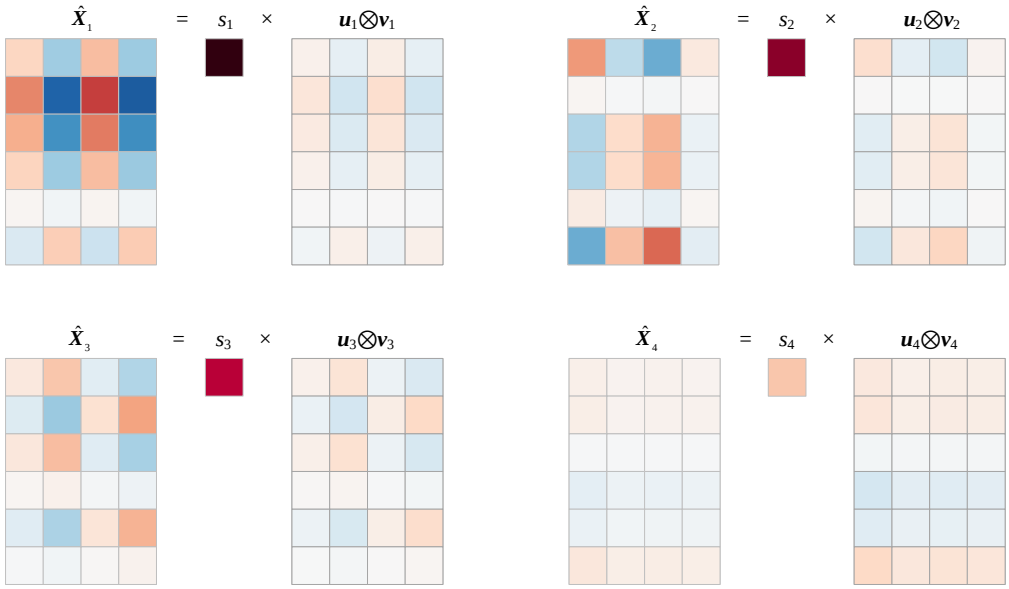


图 16. 奇异值标量乘张量积结果

正交投影

上一章指出 $\boldsymbol{v}_j$ 和 $\boldsymbol{u}_j$ 存在如下关系：

$$\boldsymbol{X}\boldsymbol{v}_j = s_j \boldsymbol{u}_j \tag{12}$$

将 (12) 代入 (11)，就得到：

$$\begin{aligned} \boldsymbol{X} &= \underbrace{\boldsymbol{X}\boldsymbol{v}_1 \otimes \boldsymbol{v}_1}_{\hat{\boldsymbol{X}}_1} + \underbrace{\boldsymbol{X}\boldsymbol{v}_2 \otimes \boldsymbol{v}_2}_{\hat{\boldsymbol{X}}_2} + \cdots + \underbrace{\boldsymbol{X}\boldsymbol{v}_D \otimes \boldsymbol{v}_D}_{\hat{\boldsymbol{X}}_D} \\ &= \boldsymbol{X}\left(\boldsymbol{v}_1 \otimes \boldsymbol{v}_1 + \boldsymbol{v}_2 \otimes \boldsymbol{v}_2 + \cdots + \boldsymbol{v}_D \otimes \boldsymbol{v}_D\right) \end{aligned} \tag{13}$$

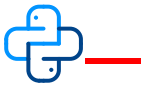
这就是本书前文反复提到的正交投影。以 $\boldsymbol{v}_1$ 为例，数据 $\boldsymbol{X}$ 在 $\boldsymbol{v}_1$ 上的投影在 $\mathbb{R}^D$ 中的像就是 $\boldsymbol{X}\boldsymbol{v}_1 \otimes \boldsymbol{v}_1$。$\text{span}(\boldsymbol{v}_1)$ 是 $\mathbb{R}^D$ 的子空间，维度为 1。

这就意味着 $\boldsymbol{X}\boldsymbol{v}_1 \otimes \boldsymbol{v}_1$ 的秩为 1，即 $\text{rank}\left(\boldsymbol{X}\boldsymbol{v}_1 \otimes \boldsymbol{v}_1\right) = 1$。

而之所以选择 $\boldsymbol{v}_1$ 做第一投影方向，就是因为在所有的一维方向中，$\boldsymbol{v}_1$ 方向对应的奇异值 s_1 最大化。大家可能又会好奇，几何视角下，奇异值 s_1 到底是什么？

卖个关子，这个问题在本书优化部分回答。

`Bk4_Ch16_01.py` 中 `Bk4_Ch16_01_E` 计算张量积，以及绘制还原原始数据过程热图。



```
# Bk4_Ch16_01_E

#%% Tensor products
```

```
u1_outer_v1 = np.outer(U[:,0][:, None], V[:,0][:, None]);

u2_outer_v2 = np.outer(U[:,1][:, None], V[:,1][:, None]);

u3_outer_v3 = np.outer(U[:,2][:, None], V[:,2][:, None]);

u4_outer_v4 = np.outer(U[:,3][:, None], V[:,3][:, None]);

# visualize tensor products

fig, axs = plt.subplots(1, 4, figsize=(12, 3))

plt.sca(axs[0])
ax = sns.heatmap(u1_outer_v1,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$u1v1^T$')

plt.sca(axs[1])
ax = sns.heatmap(u2_outer_v2,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$u2v2^T$')

plt.sca(axs[2])
ax = sns.heatmap(u3_outer_v3,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$u3v3^T$')

plt.sca(axs[3])
ax = sns.heatmap(u4_outer_v4,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$u4v4^T$')


X_1 = S[0,0]*u1_outer_v1
# X_1 = S[0,0]*U[:,0][:, None]@V[:,0][None, :];
X_2 = S[1,1]*u2_outer_v2
# X_2 = S[1,1]*U[:,1][:, None]@V[:,1][None, :];
X_3 = S[2,2]*u3_outer_v3
# X_3 = S[2,2]*U[:,2][:, None]@V[:,2][None, :];
X_4 = S[3,3]*u4_outer_v4
# X_4 = S[3,3]*U[:,3][:, None]@V[:,3][None, :];

# visualize components
fig, axs = plt.subplots(1, 4, figsize=(12, 3))

plt.sca(axs[0])
ax = sns.heatmap(X_1,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$\hat{X}_1$')

plt.sca(axs[1])
ax = sns.heatmap(X_2,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$\hat{X}_2$')

plt.sca(axs[2])
ax = sns.heatmap(X_3,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$\hat{X}_3$')

plt.sca(axs[3])
ax = sns.heatmap(X_4,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
```

```
ax.set_aspect("equal")
plt.title('$\hat{X}_4$')
```

16.6 估计与误差

把数据矩阵 $\boldsymbol{X}$ 对应的热图看做一幅图像，本节介绍如何采用较少数据尽可能还原原始图像，并准确知道误差是多少。

两层叠加

奇异值按大小排列，选取 s_1 和 s_2 还原原始数据，s_1 最大，s_2 其次。

根据上一节讨论，从图像还原角度，s_1 对应 $\boldsymbol{X}_1$，$\boldsymbol{X}_1$ 还原了 $\boldsymbol{X}$ 图像大部分特征；s_2 对应 $\boldsymbol{X}_2$，$\boldsymbol{X}_2$ 在 $\boldsymbol{X}_1$ 基础上进一步还原 $\boldsymbol{X}$。

$\boldsymbol{X}_1$ 和 $\boldsymbol{X}_2$ 叠加得到 $\hat{\boldsymbol{X}}$。如图 17 所示，$\boldsymbol{X}$ 和 $\hat{\boldsymbol{X}}$ 热图的相似度已经很高：

$$\boldsymbol{X}_{n\times D} \approx \hat{\boldsymbol{X}}_{n\times D} = \boldsymbol{X}_1 + \boldsymbol{X}_2 \tag{14}$$

$\boldsymbol{X}$ 和 $\hat{\boldsymbol{X}}$ 热图误差矩阵 $\boldsymbol{E}$ 为：

$$\boldsymbol{E} = \boldsymbol{X}_{n\times D} - \hat{\boldsymbol{X}}_{n\times D} \tag{15}$$

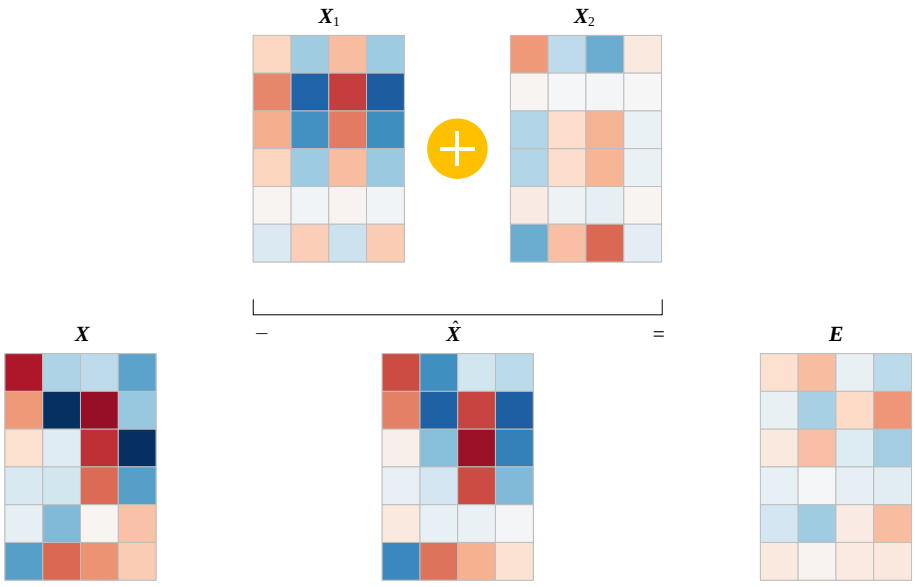


图 17. 利用前两个奇异值对应的矩阵还原数据

将 (14) 展开写成：

$$\boldsymbol{X} \approx \hat{\boldsymbol{X}} = s_1\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}} + s_2\boldsymbol{u}_2\boldsymbol{v}_2^{\mathrm{T}} = \begin{bmatrix}\boldsymbol{u}_1 & \boldsymbol{u}_2\end{bmatrix}\begin{bmatrix}s_1 & \\ & \boldsymbol{u}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{v}_1^{\mathrm{T}} \\ \boldsymbol{v}_2^{\mathrm{T}}\end{bmatrix} \tag{16}$$

上式实际上就是主成分分析中，用前两个主元还原原始数据对应的计算，具体热图如图 18 所示。本系列丛书《概率统计》一册将从 z 分数、协方差矩阵和相关性系数等角度讲解主成分分析的不同技术途径，而《数据科学》一册将从数据分析角度再谈主成分分析。

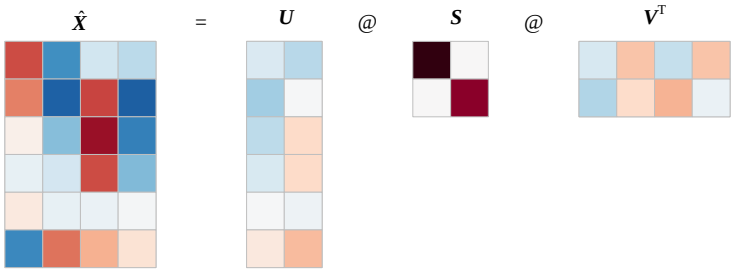


图 18. 用前两个主元还原原始数据

采用 s_1 和 s_2 还原原始数据时，误差 $\boldsymbol{E}$ 具体为：

$$\begin{aligned}\boldsymbol{X} - \hat{\boldsymbol{X}} &= s_3\boldsymbol{u}_3\boldsymbol{v}_3^{\mathrm{T}} + s_4\boldsymbol{u}_4\boldsymbol{v}_4^{\mathrm{T}} + \cdots + s_D\boldsymbol{u}_D\boldsymbol{v}_D^{\mathrm{T}} \\ &= \boldsymbol{X}\left(\boldsymbol{v}_3 \otimes \boldsymbol{v}_3 + \boldsymbol{v}_4 \otimes \boldsymbol{v}_4 + \cdots + s_D\boldsymbol{u}_D\boldsymbol{v}_D^{\mathrm{T}}\right) \\ &= \boldsymbol{X}\sum_{j=3}^{D}\boldsymbol{v}_j \otimes \boldsymbol{v}_j\end{aligned} \tag{17}$$

三层叠加

图 19 所示为利用前三个奇异值对应的矩阵还原数据，可以发现 $\boldsymbol{X}$ 和 $\hat{\boldsymbol{X}}$ 热图误差进一步缩小。

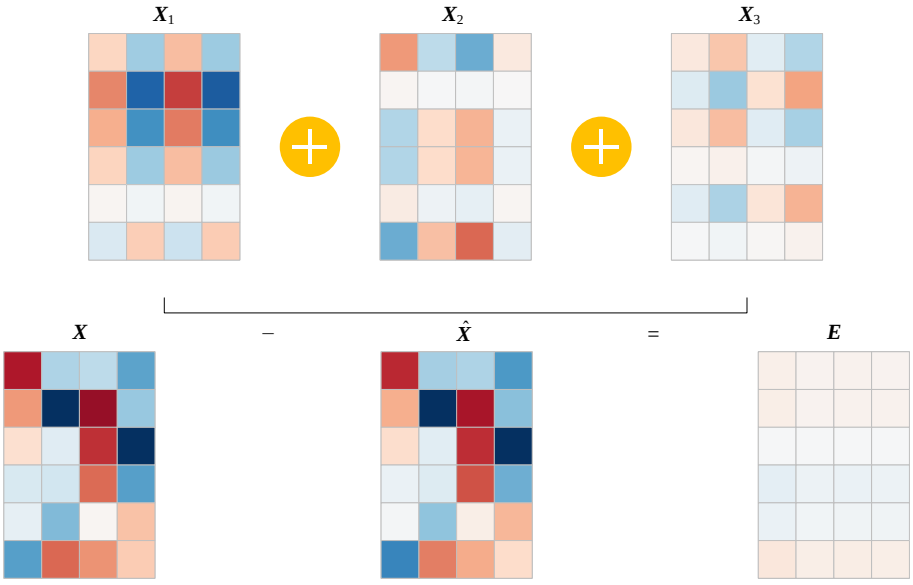
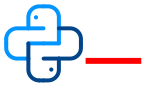


图 19. 利用前三个奇异值对应的矩阵还原数据

当 $D=4$ 时，采用 s_1、s_2、s_3 还原原始数据时，误差 $\boldsymbol{E}$ 只剩一个成分：

$$\boldsymbol{X}-\hat{\boldsymbol{X}}=s_4\boldsymbol{u}_4\boldsymbol{v}_4^{\mathrm{T}}=\boldsymbol{X}\boldsymbol{v}_4\otimes\boldsymbol{v}_4 \tag{18}$$

如果采用全部成分还原原始数据，请大家计算误差矩阵是否为 $\boldsymbol{O}$ 矩阵。



Bk4_Ch16_01.py 中 Bk4_Ch16_01_F 绘制本节数据还原和误差热图。

```
# Bk4_Ch16_01_F

#%% Reproduction and error

fig, axs = plt.subplots(1, 3, figsize=(12, 3))

plt.sca(axs[0])
ax = sns.heatmap(X,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$X$')

plt.sca(axs[1])
ax = sns.heatmap(X_1 + X_2,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$\hat{X}_1 + \hat{X}_2$')


plt.sca(axs[2])
ax = sns.heatmap(X - (X_1 + X_2),cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$X - (\hat{X}_1 + \hat{X}_2)$')


fig, axs = plt.subplots(1, 3, figsize=(12, 3))


plt.sca(axs[0])
ax = sns.heatmap(X,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$X$')

plt.sca(axs[1])
ax = sns.heatmap(X_1 + X_2 + X_3,cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$\hat{X}_1 + \hat{X}_2 + \hat{X}_3$')


plt.sca(axs[2])
ax = sns.heatmap(X - (X_1 + X_2 + X_3),cmap='RdBu_r',vmax = all_max,vmin = all_min,
                 cbar_kws={"orientation": "horizontal"})
ax.set_aspect("equal")
plt.title('$X - (\hat{X}_1 + \hat{X}_2 + \hat{X}_3)$')
```

如下四幅图最能概括本章的核心内容。奇异值四种不同形式，每种都有特殊意义，它们之间都有内在联系。

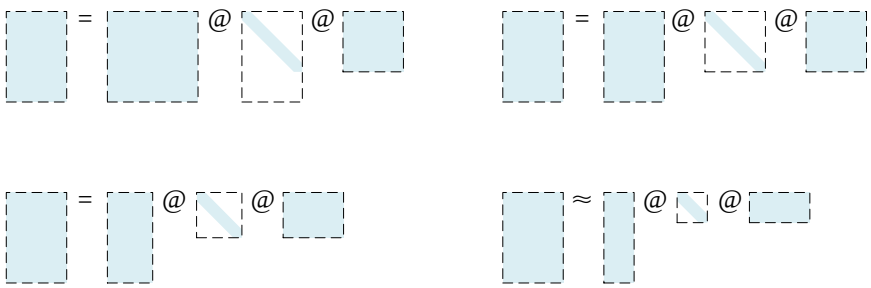


图 20. 总结本章重要内容的四副图

再次强调，矩阵分解的内核还是矩阵乘法。相信大家已经在本章奇异值分解中看到矩阵乘法的不同视角、分块矩阵乘法等内容。

此外，张量积和正交投影这两个工具在解释奇异值分解上有立竿见影的效果。

本章留了个悬念，奇异值分解中的奇异值的几何内涵到底是什么？我们将在本书优化部分回答这个问题。在那里，大家会用优化视角一睹奇异值的几何本质。

本章虽然是矩阵分解板块的最后一章，但是本书有关矩阵分解的故事远没有结束。本书后续会从几何角度、数据角度、空间角度、应用角度一次次回顾这些线性代数的有力武器。