

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

14.4 截断型奇异值分解



本节你将掌握的核心技能：

- ▶ 截断型 SVD：保留主成分，舍弃噪声和冗余信息。
- ▶ 将数据矩阵分解为多个秩一矩阵的叠加。
- ▶ 载荷 = 特征向量，得分 = 投影坐标。
- ▶ 低秩近似还原原始数据：前 p 个主成分高效还原数据且压缩存储。
- ▶ \mathbf{X} 在原空间不正交，但 \mathbf{V} 投影结果 \mathbf{Z} 完成正交化。

本章前文介绍了四种 SVD 分解，本节展开讲解截断型 SVD。截断型 SVD 应用特别广泛，比如主成分分析、数据降维等等。

矩阵乘法第二视角

形状为 $n \times D$ 原始数据矩阵 \mathbf{X} ，其经济型 SVD 分解为。

$$\mathbf{X}_{n \times D} = \mathbf{U}_{n \times D} \mathbf{S}_{D \times D} \mathbf{V}_{D \times D}^T \quad (1)$$

其中， \mathbf{U} 和 \mathbf{X} 的形状相同， \mathbf{U} 为半正交矩阵（列向量为单位向量且两两正交）； \mathbf{V} 为正交矩阵（列向量为单位向量且两两正交）。

如图 1 所示，根据矩阵乘法第二视角，原始数据矩阵 \mathbf{X} 的经济型 SVD 分解可以展开得到

$$\mathbf{X}_{n \times D} = \underbrace{\begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_D \end{bmatrix}}_{\mathbf{U}_{n \times D}} \underbrace{\begin{bmatrix} s_1 & & & \\ & s_2 & & \\ & & \ddots & \\ & & & s_D \end{bmatrix}}_{\mathbf{S}_{D \times D}} \underbrace{\begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_D^T \end{bmatrix}}_{\mathbf{V}_{D \times D}} = s_1 \mathbf{u}_1 \mathbf{v}_1^T + s_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + s_D \mathbf{u}_D \mathbf{v}_D^T = \sum_{j=1}^D s_j \mathbf{u}_j \mathbf{v}_j^T \quad (2)$$

这样，我们把数据矩阵 \mathbf{X} 写成 D 个形状相同的矩阵相加，具体如图 2 所示。

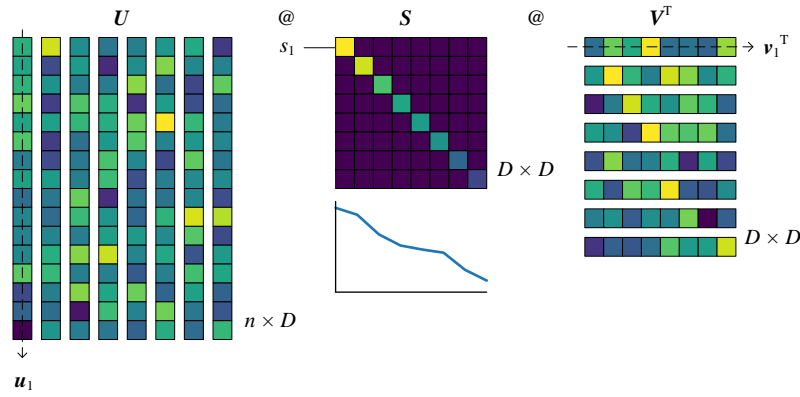
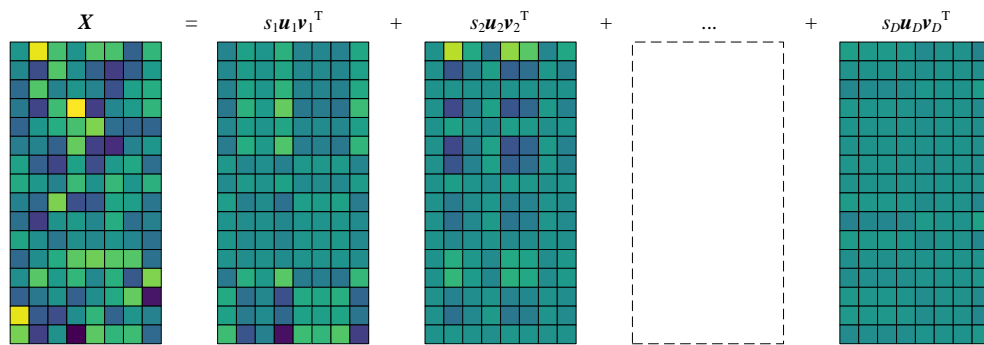


图 1. 矩阵乘法第二视角展开经济型 SVD 分解

图 2. 原始数据相当于由 D 个形状相同矩阵求和的结果

奇异值体现重要性

如图 3、图 4 所示，由于 u_j 和 v_j 都是单位向量，即向量 L^2 范数都为 1。

这些单位向量之间只存在方向分别，不存在大小的区别。因此，奇异值 s_j 的大小体现出方向的重要性。

对角方阵 S 的主对角线元素的作用，可以看作是缩放 U 的列向量；也可以看作是缩放 V^T 的行向量（即 V 的列向量）。

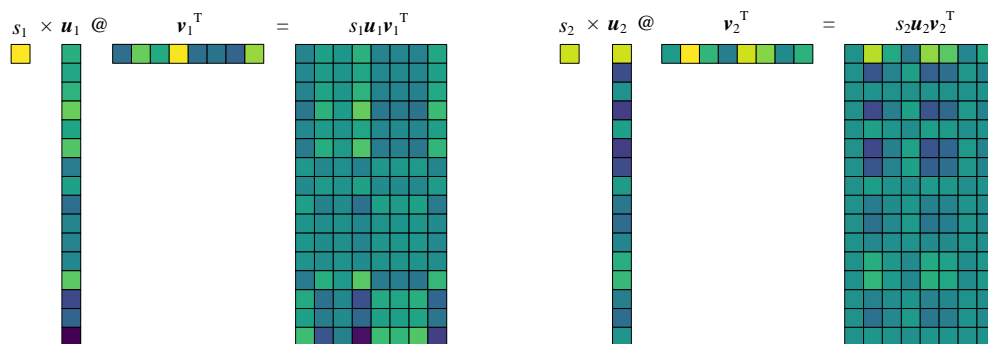


图 3. 前两个主成分还原部分原始数据

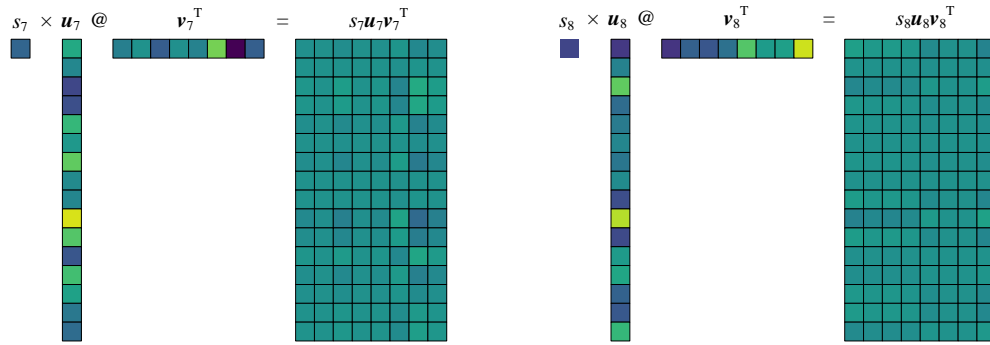


图 4. 最后两个主成分还原部分原始数据

从主成分分析角度来看，如果奇异值 $s_1, s_2 \dots s_D$ 由小到大排列， \mathbf{v}_1 就是第一主成分载荷 (loading)， \mathbf{u}_1 就是第一主成分因子得分 (score)，依此类推。

也就是说，载荷 = 特征向量，得分 = 投影坐标。

⚠ 注意，有些文献把载荷定义为“特征向量 \times 奇异值”。

近似还原

如图 5 所示，用前 p 个主成分还原原始数据

$$\mathbf{X}_{n \times D} \approx \hat{\mathbf{X}}_{n \times D} = \mathbf{U}_{n \times p} \mathbf{S}_{p \times p} (\mathbf{V}_{D \times p})^T = \sum_{j=1}^p s_j \mathbf{u}_j \mathbf{v}_j^T \quad (3)$$

假设前 p 个奇异值 s_j 均大于 0；这样，矩阵 $\hat{\mathbf{X}}_{n \times D}$ 的秩为 p 。

也就是说，在 s_j 均大于 0 的前提下，上式中每叠加一层 $s_j \mathbf{u}_j \mathbf{v}_j^T$ ， $\hat{\mathbf{X}}_{n \times D}$ 的秩就增大 1。

⚠ 注意，使用不同 Python 函数完成奇异值分解，我们会发现结果单位向量常常存在符号不同；这是因为 $s_j \mathbf{u}_j \mathbf{v}_j^T$ 等价 $s_j (-\mathbf{u}_j) (-\mathbf{v}_j)^T$ ，也就是说 \mathbf{u}_j 、 \mathbf{v}_j 可以同时变号，不影响结果。

举个例子， $\hat{\mathbf{X}}_{n \times D} = s_1 \mathbf{u}_1 \mathbf{v}_1^T$ 的秩为 1； $\hat{\mathbf{X}}_{n \times D} = s_1 \mathbf{u}_1 \mathbf{v}_1^T + s_2 \mathbf{u}_2 \mathbf{v}_2^T$ 的秩为 2。

图 5 中， \mathbf{X} 和 $\hat{\mathbf{X}}$ 热图误差矩阵为：

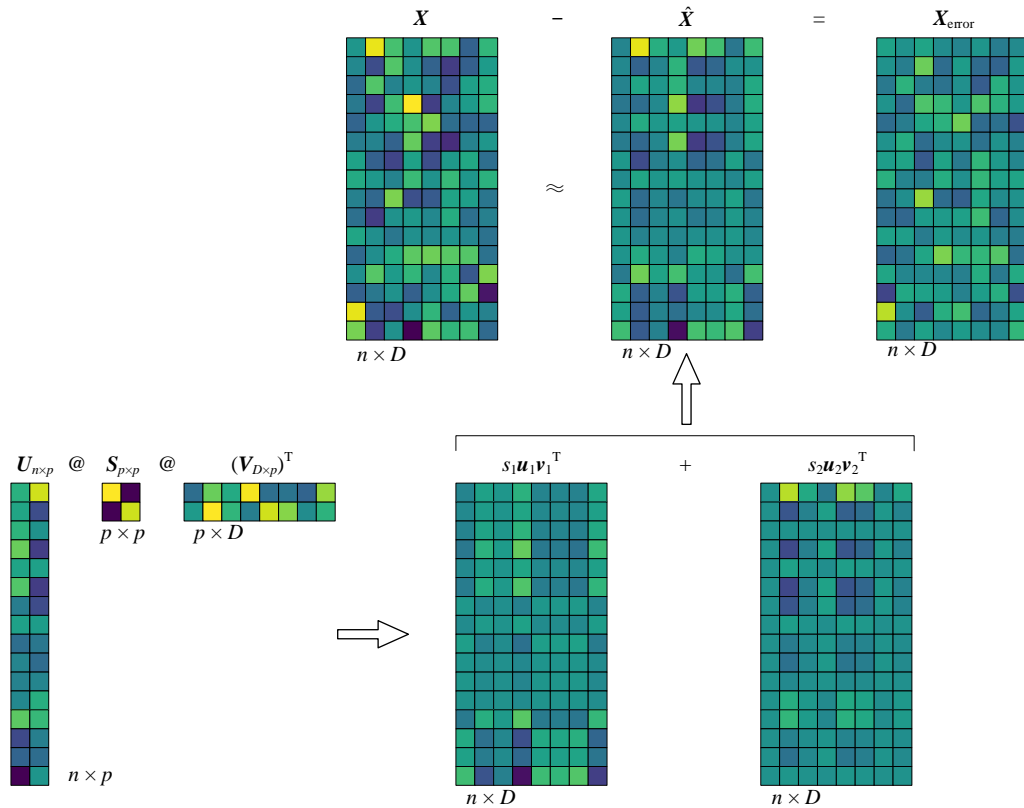
$$\mathbf{X}_{\text{error}} = \mathbf{X}_{n \times D} - \hat{\mathbf{X}}_{n \times D} \quad (4)$$

根据 (2) 我们知道，误差也是若干层叠加，即

$$\mathbf{X}_{\text{error}} = \sum_{j=1}^D s_j \mathbf{u}_j \mathbf{v}_j^T - \sum_{j=1}^p s_j \mathbf{u}_j \mathbf{v}_j^T = s_{p+1} \mathbf{u}_{p+1} \mathbf{v}_{p+1}^T + \dots + s_D \mathbf{u}_D \mathbf{v}_D^T = \sum_{j=p+1}^D s_j \mathbf{u}_j \mathbf{v}_j^T \quad (5)$$



请大家回顾本书第 12 章第 3 节主成分分析中的误差计算。

图 5. 用前 p 个主成分还原原始数据

正交投影

下面，我们再从投影视角理解截断型 SVD 分解。

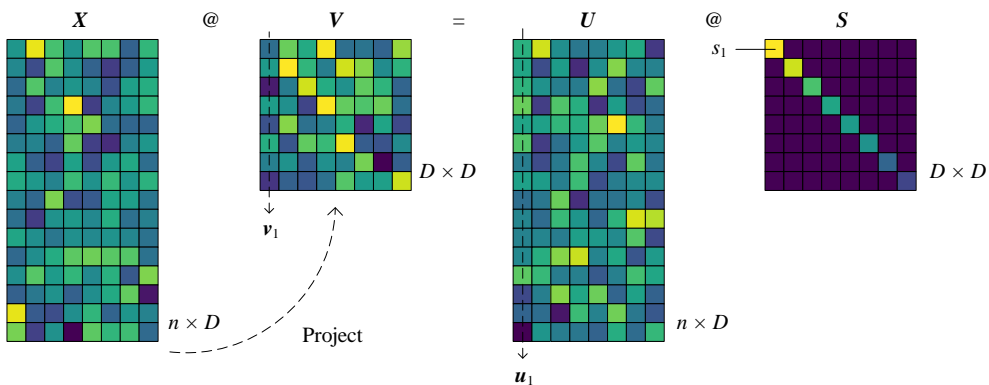


请大家回顾本书第 11 章第 4 节有关格拉姆矩阵谱分解相关内容。

将 (1) 写成

$$X_{n \times D} V_{D \times D} = U_{n \times D} S_{D \times D} \quad (6)$$

如图 6 所示，上式相当于将 X 投影到 V 空间中。

图 6. 原始数据向 V 投影

也用矩阵乘法第二视角，将 (6) 写成

$$\mathbf{X}_{n \times D} \underbrace{[\mathbf{v}_1 \ \mathbf{v}_2 \ \cdots \ \mathbf{v}_D]}_{\mathbf{V}_{D \times D}} = \underbrace{[\mathbf{u}_1 \ \mathbf{u}_2 \ \cdots \ \mathbf{u}_D]}_{\mathbf{U}_{n \times D}} \begin{bmatrix} s_1 & & & \\ & s_2 & & \\ & & \ddots & \\ & & & s_D \end{bmatrix} \quad (7)$$

进一步展开得到

$$[\mathbf{X}\mathbf{v}_1 \ \mathbf{X}\mathbf{v}_2 \ \cdots \ \mathbf{X}\mathbf{v}_D] = [s_1\mathbf{u}_1 \ s_2\mathbf{u}_2 \ \cdots \ s_D\mathbf{u}_D] \quad (8)$$

从几何角度， \mathbf{X} 朝 \mathbf{v}_j 标量投影结果为 $s_j\mathbf{u}_j$ 。

$$\mathbf{X}\mathbf{v}_j = s_j\mathbf{u}_j \quad (9)$$

图 7 所示为原始数据朝 \mathbf{v}_1 投影结果为 $s_1\mathbf{u}_1$ 。由于 $\|\mathbf{u}_j\|=1$ ， $\|\mathbf{X}\mathbf{v}_j\|=s_j$ 。

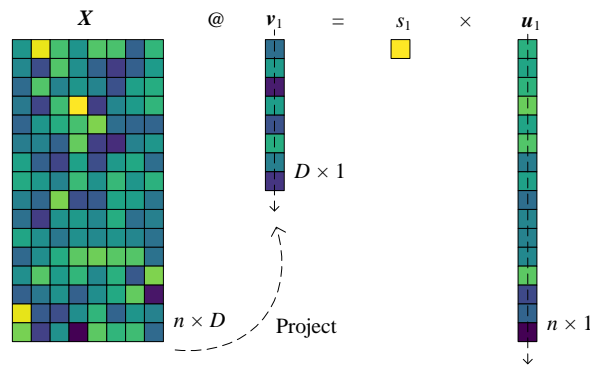


图 7. 原始数据向 \mathbf{v}_1 标量投影

如图 8 所示，数据矩阵 \mathbf{X} 向规范正交基 $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_D]$ 构成的 D 维空间投影写成

$$\mathbf{Z} = \mathbf{X}\mathbf{V} \quad (10)$$

\mathbf{Z} 代表 \mathbf{X} 在新的规范正交基 $[\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_D]$ 下的坐标。

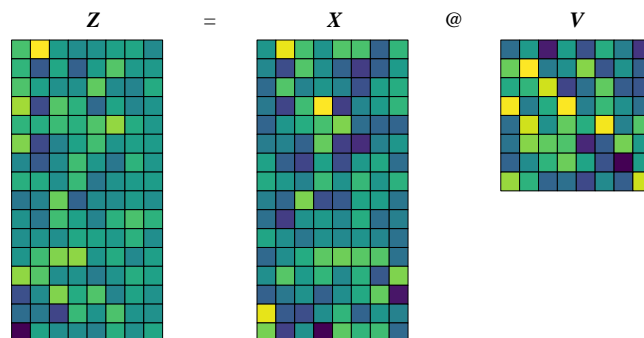


图 8. \mathbf{X} 向规范正交基 \mathbf{V} 投影

由于 $\mathbf{X} = \mathbf{USV}^T$ ，代入 (10) 得到：

$$\mathbf{Z} = \mathbf{USV}^T \mathbf{V} = \mathbf{US} = [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \cdots \quad \mathbf{u}_D] \begin{bmatrix} s_1 & & & \\ & s_2 & & \\ & & \ddots & \\ & & & s_D \end{bmatrix} = [s_1 \mathbf{u}_1 \quad s_2 \mathbf{u}_2 \quad \cdots \quad s_D \mathbf{u}_D] \quad (11)$$

即，

$$\underbrace{[\mathbf{z}_1 \quad \mathbf{z}_2 \quad \cdots \quad \mathbf{z}_D]}_{\mathbf{Z}} = \underbrace{[s_1 \mathbf{u}_1 \quad s_2 \mathbf{u}_2 \quad \cdots \quad s_D \mathbf{u}_D]}_{\mathbf{US}} \quad (12)$$

如图 9 所示，上式给了我们计算 \mathbf{Z} 的第二条路径。换句话说， \mathbf{u}_j 实际上就是“单位化”的投影坐标， s_j 是 \mathbf{z}_j 向量的模，即 $\|\mathbf{X}\mathbf{v}_j\| = \|\mathbf{z}_j\| = \|s_j \mathbf{u}_j\| = s_j \|\mathbf{u}_j\| = s_j$ 。

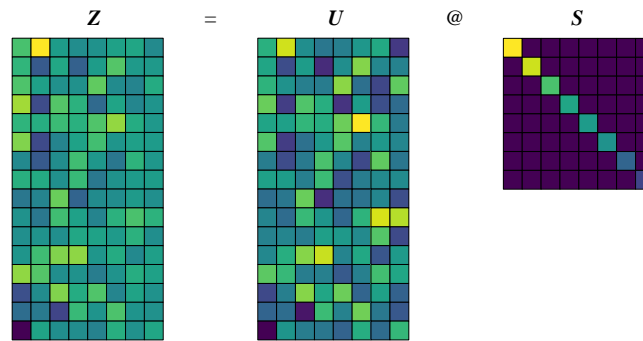


图 9. 第二条计算 \mathbf{Z} 的路径

正交化

对 \mathbf{Z} 求格拉姆矩阵：

$$\mathbf{Z}^T \mathbf{Z} = \begin{bmatrix} \mathbf{z}_1^T \\ \mathbf{z}_2^T \\ \vdots \\ \mathbf{z}_D^T \end{bmatrix} [\mathbf{z}_1 \quad \mathbf{z}_2 \quad \cdots \quad \mathbf{z}_D] = \begin{bmatrix} \mathbf{z}_1^T \mathbf{z}_1 & \mathbf{z}_1^T \mathbf{z}_2 & \cdots & \mathbf{z}_1^T \mathbf{z}_D \\ \mathbf{z}_2^T \mathbf{z}_1 & \mathbf{z}_2^T \mathbf{z}_2 & \cdots & \mathbf{z}_2^T \mathbf{z}_D \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{z}_D^T \mathbf{z}_1 & \mathbf{z}_D^T \mathbf{z}_2 & \cdots & \mathbf{z}_D^T \mathbf{z}_D \end{bmatrix} \quad (13)$$

请大家将上式写成向量内积形式。

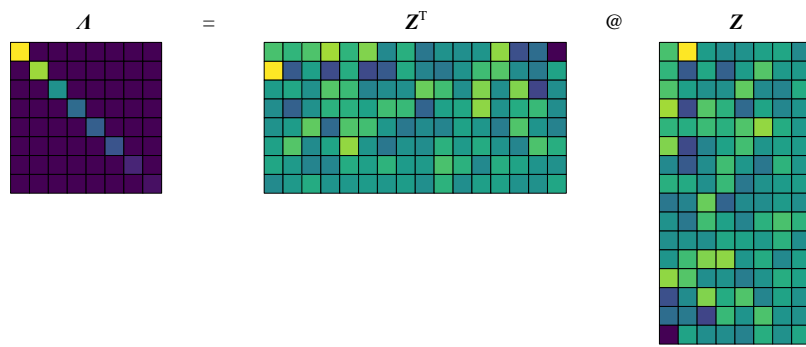
将 (11) 代入得到 (13)：

$$\mathbf{Z}^T \mathbf{Z} = (\mathbf{US})^T \mathbf{US} = \mathbf{S}^T \mathbf{U}^T \mathbf{U} \mathbf{S} = \begin{bmatrix} s_1^2 & & & \\ & s_2^2 & & \\ & & \ddots & \\ & & & s_D^2 \end{bmatrix} \quad (14)$$

如图 10 所示，发现 \mathbf{Z} 的格拉姆矩阵为对角阵，也就是说 \mathbf{Z} 的列向量两两正交，即：

$$\mathbf{z}_i^T \mathbf{z}_j = \mathbf{z}_j^T \mathbf{z}_i = \mathbf{z}_i \cdot \mathbf{z}_j = \mathbf{z}_j \cdot \mathbf{z}_i = \langle \mathbf{z}_i, \mathbf{z}_j \rangle = \langle \mathbf{z}_j, \mathbf{z}_i \rangle = 0, \quad i \neq j \quad (15)$$

回看图 8， $\mathbf{X} \rightarrow \mathbf{Z}$ 完成的的就是**正交化** (orthogonalization)。

图 10. Z 的格拉姆矩阵

分析鸢尾花照片

下面用截断奇异值分解分析鸢尾花照片。图 11 所示为作者拍的一张鸢尾花照片，经过黑白化处理后的每个像素都是 $[0, 1]$ 范围内的数字。所以整幅图片可以看成是一个数据矩阵。



这部分内容改编自“鸢尾花书”《机器学习》。

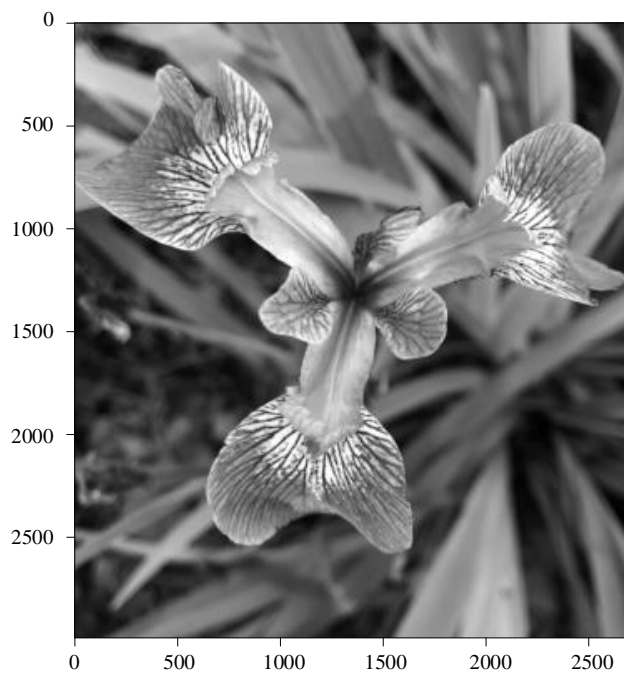


图 11. 鸢尾花图片，经过黑白处理，来自《机器学习》

图 12 所示为利用 SVD 分解得到的奇异值随主成分变化。前 10 个奇异值贡献最大，主导数据结构。

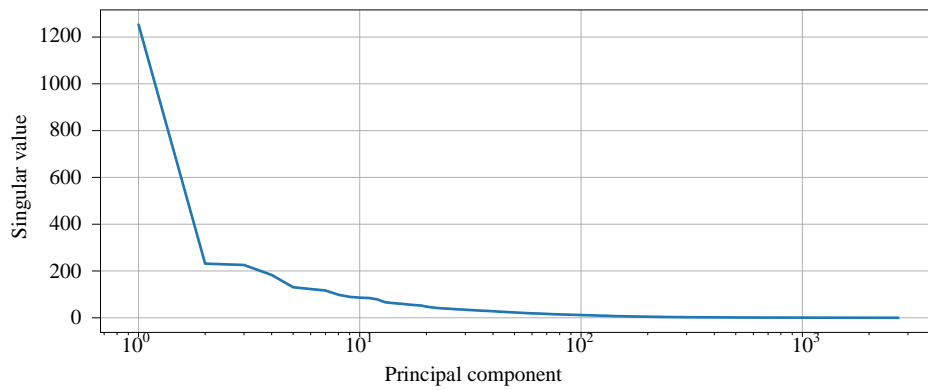


图 12. 奇异值随主成分变化，来自《机器学习》

图 13 所示为利用前 4 个主元还原鸢尾花照片。这幅图相当于由 4 个秩一矩阵叠加而成，具体如图 14 所示。

图 13 左图中我们仅仅能够看到“格子”。

图 15 所示为利用前 64 个主元还原鸢尾花图片，图形已经很清晰，我们已经能够看到鸢尾花的样子。相比原图片，图 15 的数据发生大幅压缩，仅仅保留了大概 2.5% (64/2714)。

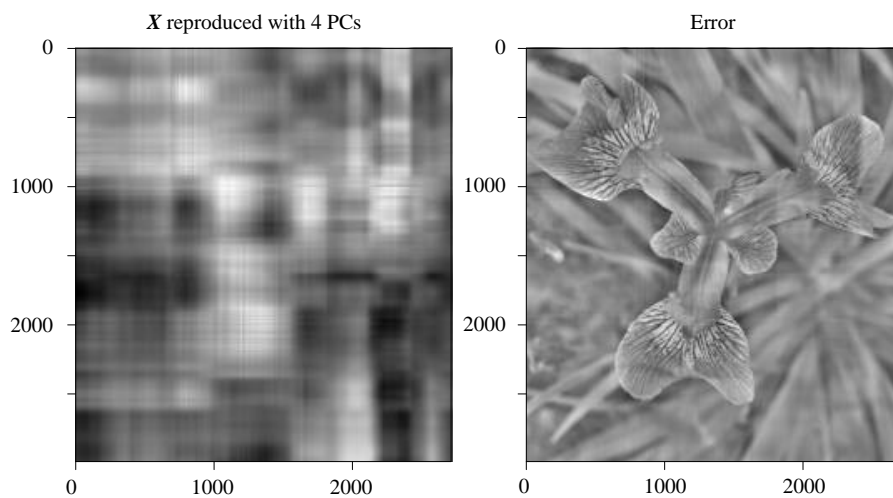


图 13. 利用第 1、2、3、4 主元还原鸢尾花照片

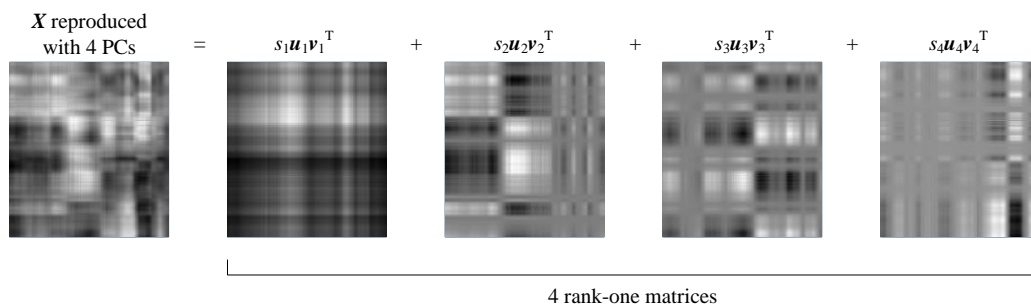


图 14. 前 4 个秩一矩阵叠加

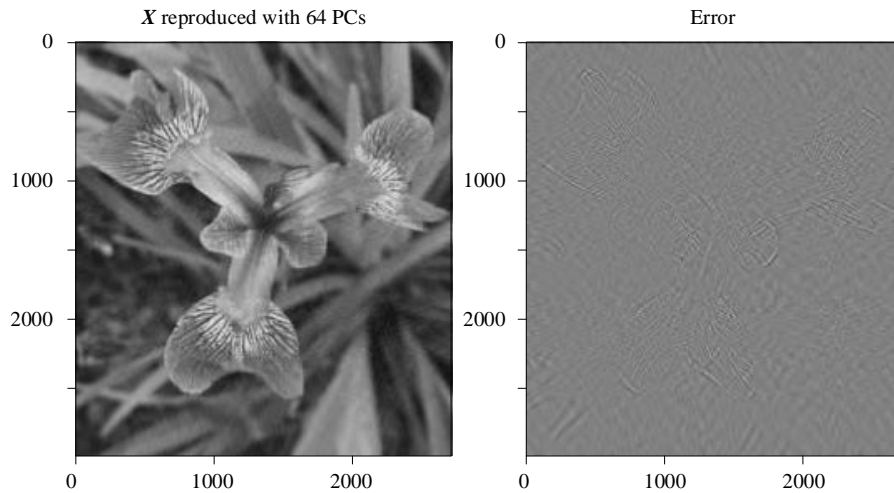


图 15. 利用前 64 个主元还原鸢尾花照片



请大家用 DeepSeek/ChatGPT 等工具完成本节如下习题。

Q1. 请对如下矩阵完成经济型奇异值分解，然后保留第一主成分近似还原原始矩阵，并计算误差。

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}$$