1 核心算法

1.1 Jacobi 方法

1.1.1 特征值的计算

首先找到矩阵绝对值最大的非对角元素 a_{pq}，再利用下面的公式求解迭代后矩阵：

$$
\begin{cases}
s = \dfrac{a_{qq} - a_{pp}}{2a_{pq}} \\
t = \begin{cases} t^2 + 2st - 1 = 0 \text{ 的绝对值较小的跟}, & s \neq 0 \\ 1 & s = 0 \end{cases} \\
c = \cos\theta = \dfrac{1}{\sqrt{1+t^2}} \\
d = \sin\theta = \dfrac{t}{\sqrt{1+t^2}} \\
b_{ip} = b_{pi} = ca_{pi} - da_{qi}, & i \neq p, q \\
b_{ip} = b_{pi} = da_{pi} + ca_{qi}, & i \neq p, q \\
b_{pp} = a_{pp} - ta_{pq}, \\
b_{qq} = a_{qq} + ta_{pq}, \\
b_{pq} = b_{qp} = 0, \\
b_{ij} = a_{ij}, & i \neq p, q; \quad j \neq p, q
\end{cases}
$$

如此迭代同时计算非对角元素平方和 $\sum\limits_{i \neq j} a_{ij}^2$，直至非对角元素平方和小于指定精度 ε，此时迭代矩阵的对角元即为所求特征值。伪代码见 Algorithm 1。

1.1.2 特征向量的计算

这里为了优化算法的数值计算结果，我尝试了两种方法计算特征向量。

旋转矩阵求解 利用单位阵在迭代过程中右乘旋转矩阵 G，表示为：

$$
g_{ij} = \begin{cases}
1, & i = j; \quad i, j \neq p, q \\
\cos\theta, & i, j = p; \quad i, j = q \\
\sin\theta, & i = p; \quad j = q \\
-\sin\theta, & i = q; \quad j = p
\end{cases}
$$

即可得到对应的特征向量矩阵。算法描述见 Algorithm 2。

高斯消元法求解 根据特征值的定义：

$$\boldsymbol{A}\boldsymbol{X} = \lambda \boldsymbol{X}$$

只需求解方程组 $(\boldsymbol{A} - \lambda \boldsymbol{I})\boldsymbol{X} = 0$ 即可得到相应的特征向量。算法详见 Algorithm 3。

1.2 SVD 分解

在 Jacobi 方法的技术上，我们若要将矩阵 $\boldsymbol{A}$ 分解为 $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}$ 只需：

1. 对实矩阵 $\boldsymbol{A}$, 应用 Jacobi 方法计算 $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$ 的特征值 $\lambda_1, \lambda_2, \lambda_3, \cdots, \lambda_n$, 满足 $\lambda_i > \lambda_j, i < j$；
2. 计算 $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$ 的特征向量矩阵 $\boldsymbol{U}$；
3. 计算 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 的特征向量矩阵 $\boldsymbol{V}$；
4. 计算 $\Sigma = diag(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \sqrt{\lambda_3}, \cdots, \sqrt{\lambda_n})$。

算法详见 Algorithm 4。

1.3 PCA 数据降维

在之前工作的基础上，对鸢尾花数据集进行数据降维：

1. 从文件读取数据集；
2. 选取前四个维度的数据组成矩阵 $\boldsymbol{X}^{\mathrm{T}}$；
3. 对矩阵 $\boldsymbol{X}$ 做去中心化处理并转置为 4×150 的 $\boldsymbol{X}$；
4. 计算协方差矩阵 $\frac{1}{m}\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$；
5. 利用 Jacobi 方法求解 $\frac{1}{m}\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$ 的特征值和特征向量；
6. 选取特征值最大的两个特征向量作为基向量，并将原始数据 $\boldsymbol{X}$ 投影至基向量组成的平面。

算法详见 Algorithm 5。

2 输出结果

2.0.1 任务 (a)

随机矩阵:

```
0.545  0.271  0.613
0.861  0.383  0.248
0.784  0.240  0.750
0.170  0.217  0.679
```

奇异值分解结果:

U 矩阵:

```
0.486  0.154  0.109  -0.853
0.500  -0.718  0.436  0.211
0.625  0.075  -0.726  0.277
0.351  0.675  0.520  0.388
```

```
非对角元元素平方和: 4.786127
非对角元元素平方和: 1.935603
非对角元元素平方和: 0.072186
非对角元元素平方和: 0.038661
非对角元元素平方和: 0.017327
非对角元元素平方和: 0.008252
非对角元元素平方和: 0.000253
非对角元元素平方和: 0.000069
非对角元元素平方和: 0.000019
非对角元元素平方和: 0.000002
非对角元元素平方和: 0.000000
非对角元元素平方和: 1.678484
非对角元元素平方和: 0.003256
非对角元元素平方和: 0.000314
非对角元元素平方和: 0.000005
非对角元元素平方和: 0.000000
```

Sigma 矩阵:

```
1.768  0.000  0.000
0.000  0.566  0.000
0.000  0.000  0.144
0.000  0.000  0.000
```

V^T 矩阵:

```
0.704  0.311  0.638
-0.637  -0.121  0.761
0.314  -0.943  0.113
```

```
AA^T 的 jacobi 方法结果: 3.125  0.321  0.021  0.000
```

2.1 任务 (b)

输出

协方差矩阵:

```
0.283  0.130  -0.053  -0.170
0.130  0.060  -0.025  -0.078
-0.053  -0.025  0.010  0.032
-0.170  -0.078  0.032  0.102
```

可视化 利用 python 的 `pandas` 和 `matplotlib` 库将数据可视化：

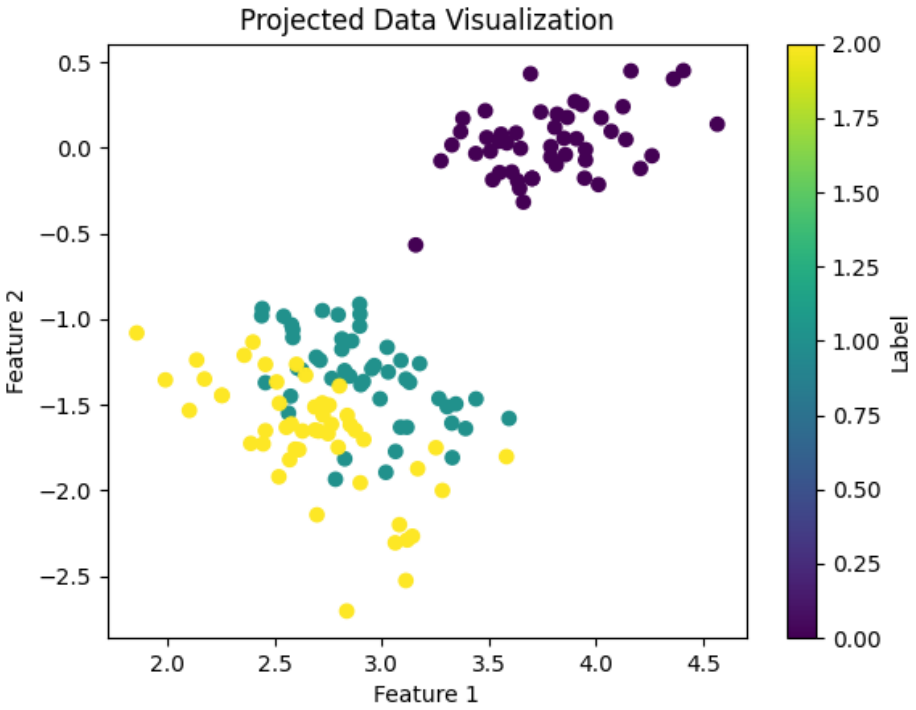

图 1: Iris Data

3 结果分析

3.1 Jacobi 迭代方法

从输出结果看，迭代过程中的对角元素平方和确实呈下降趋势；计算 $\det(\boldsymbol{A}\boldsymbol{A}^T - \lambda_i \boldsymbol{I})$ 得出结果：

$$\begin{pmatrix} -0.0408 \\ -1.68 \times 10^{-4} \\ 6.724 \times 10^{-8} \\ 0 \end{pmatrix} \approx 0$$

因此计算结果是对称矩阵特征值的近似值。

3.2 可视化结果

从可视化结果能够较明显地看出三个不同标签的数据集分别聚集在三个不同的区域，标签 2 和 3 代表的数据比较相似，离标签 1 的数据较远。成功将数据降维！

Algorithm 1: Eigenvalue Iteration

Input: Symmetric matrix A, tolerance ε

Output: Eigenvalues of A

Initialize $B = A$;
sum_squares $= \sum_{i \neq j} a_{ij}^2$;
while *sum_squares* $> \varepsilon$ **do**
    Find the largest off-diagonal element a_{pq} in absolute value;
    $s = \frac{a_{qq} - a_{pp}}{2a_{pq}}$;
    if $s \neq 0$ **then**
        $t = \frac{-s \pm \sqrt{s^2+1}}{s}$;
    else
        $t = 1$;
    end
    $\theta = \cos^{-1}(\frac{1}{\sqrt{1+t^2}})$;
    $c = \cos(\theta)$;
    $d = \sin(\theta)$;
    Update elements of B according to: **for** $i \neq p, q$ **do**
        $b_{ip} = b_{pi} = ca_{pi} - da_{qi}$;
        $b_{iq} = b_{qi} = da_{pi} + ca_{qi}$;
    end
    $b_{pp} = a_{pp} - ta_{pq}$;
    $b_{qq} = a_{qq} + ta_{pq}$;
    $b_{pq} = b_{qp} = 0$;
    $A = B$;
    Recalculate sum_squares $= \sum_{i \neq j} a_{ij}^2$;
end

Algorithm 2: Eigenvector Calculation

Input: Rotation angle θ, pivot indices p, q

Output: Eigenvector matrix V

```
Initialize V as the identity matrix;
while sum_squares > ε do
    .....
    for i = 1 to n do
        for j = 1 to n do
            if i = j and i ≠ p, q then
                g_ij = 1;
            else
                if i = p and j = q then
                    g_ij = sin(θ);
                else
                    if i = q and j = p then
                        g_ij = −sin(θ);
                    else
                        if i = p and j = p or i = q and j = q then
                            g_ij = cos(θ);
                        else
                            g_ij = 0;
                        end
                    end
                end
            end
        end
    end
    Update V = V × G;
    Recalculate sum_squares = Σ_{i≠j} a_ij^2;
end
```

Algorithm 3: Gaussian Elimination for Solving $\boldsymbol{AX} = \lambda \boldsymbol{X}$

Input: Matrix $\boldsymbol{A}$, eigenvalue λ

Output: Eigenvector $\boldsymbol{X}$ corresponding to λ

Construct matrix $\boldsymbol{B} = \boldsymbol{A} - \lambda \boldsymbol{I}$;

Augment $\boldsymbol{B}$ with zero vector on the right to form the augmented matrix $\Big[\boldsymbol{B}|\boldsymbol{0}\Big]$;

Use Gaussian elimination to row reduce the augmented matrix to row-echelon form;

Extract the solutions $\boldsymbol{X}$ from the row-echelon form of the augmented matrix;

Normalize the eigenvector $\boldsymbol{X}$ to obtain a unit vector (if desired);

Algorithm 4: SVD Decomposition using Jacobi Method

Input: Matrix $\boldsymbol{A}$ of size $m \times n$

Output: Matrices $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, $\boldsymbol{V}$ for SVD decomposition of $\boldsymbol{A}$

`// Step 1: Compute` $\boldsymbol{AA}^{\mathrm{T}}$ `eigenvalues and eigenvectors`

Compute $\boldsymbol{C} = \boldsymbol{AA}^{\mathrm{T}}$;

Use Jacobi method to compute eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ of $\boldsymbol{C}$;

Sort eigenvalues λ_i in descending order;

Compute eigenvector matrix $\boldsymbol{U}$ corresponding to the eigenvalues of $\boldsymbol{C}$;

`// Step 2: Compute` $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ `eigenvalues and eigenvectors`

Compute $\boldsymbol{D} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$;

Use Jacobi method to compute eigenvalues $\mu_1, \mu_2, \ldots, \mu_n$ of $\boldsymbol{D}$;

Sort eigenvalues μ_i in descending order;

Compute eigenvector matrix $\boldsymbol{V}$ corresponding to the eigenvalues of $\boldsymbol{D}$;

`// Step 3: Compute` $\boldsymbol{\Sigma}$

Initialize $\boldsymbol{\Sigma}$ as a diagonal matrix of size $m \times n$;

Set $\Sigma_{ii} = \sqrt{\lambda_i}$ for $i = 1, 2, \ldots, n$ (assuming $m \geq n$);

Algorithm 5: PCA Dimensionality Reduction using Jacobi Method

Input: File containing Iris dataset

Output: Projected data onto the principal components

`// Step 1: Read data from file`

Read Iris dataset from file into matrix $\boldsymbol{X}$;

`// Step 2: Select the first four dimensions`

Extract the first four dimensions of $\boldsymbol{X}$ to form matrix $\boldsymbol{X}^{\mathrm{T}}$;

`// Step 3: Center and transpose matrix` $\boldsymbol{X}$

Center matrix $\boldsymbol{X}$ and transpose to obtain a 4×150 matrix $\boldsymbol{X}$;

`// Step 4: Compute the covariance matrix`

Compute the covariance matrix $\boldsymbol{C} = \frac{1}{m}\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}$, where m is the number of samples;

`// Step 5: Use Jacobi method to find eigenvalues and eigenvectors`

Use Jacobi method to compute eigenvalues and eigenvectors of $\boldsymbol{C}$;

`// Step 6: Select the top two eigenvectors`

Select the eigenvectors corresponding to the largest eigenvalues as the principal components;

`// Step 7: Project the original data onto the principal component subspace`

Project the original data $\boldsymbol{X}$ onto the subspace spanned by the selected principal components;