计算方法Lab4

PB20000126葛哲凯

一、实验环境

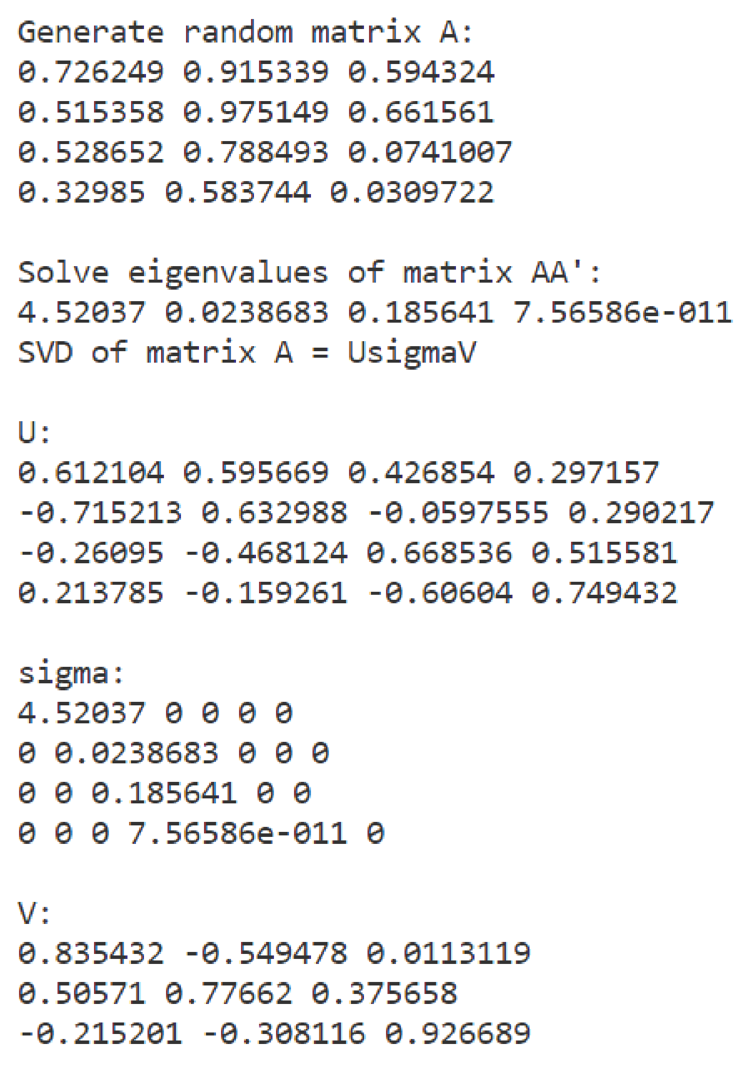
vscode，c++

二、随机矩阵SVD

自行生成一个4 × 3的随机矩阵A，应用Jacobi方法求解矩阵AAT 的特征值，计算矩阵A的SVD分解。

要求A的每个元素均为[0, 1]区间内的随机数

结果如下：

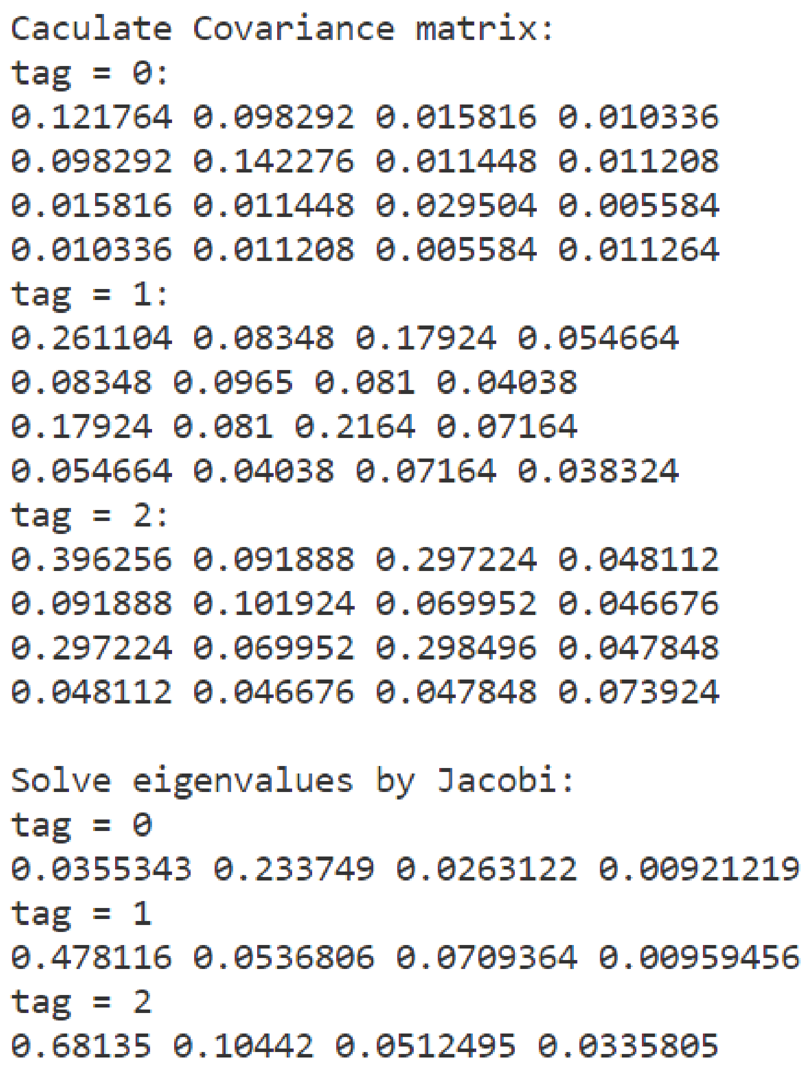


```
Generate random matrix A:
0.726249 0.915339 0.594324
0.515358 0.975149 0.661561
0.528652 0.788493 0.0741007
0.32985 0.583744 0.0309722

Solve eigenvalues of matrix AA':
4.52037 0.0238683 0.185641 7.56586e-011
SVD of matrix A = UsigmaV

U:
0.612104 0.595669 0.426854 0.297157
-0.715213 0.632988 -0.0597555 0.290217
-0.26095 -0.468124 0.668536 0.515581
0.213785 -0.159261 -0.60604 0.749432

sigma:
4.52037 0 0 0 0
0 0.0238683 0 0 0
0 0 0.185641 0 0
0 0 0 7.56586e-011 0

V:
0.835432 -0.549478 0.0113119
0.50571 0.77662 0.375658
-0.215201 -0.308116 0.926689
```

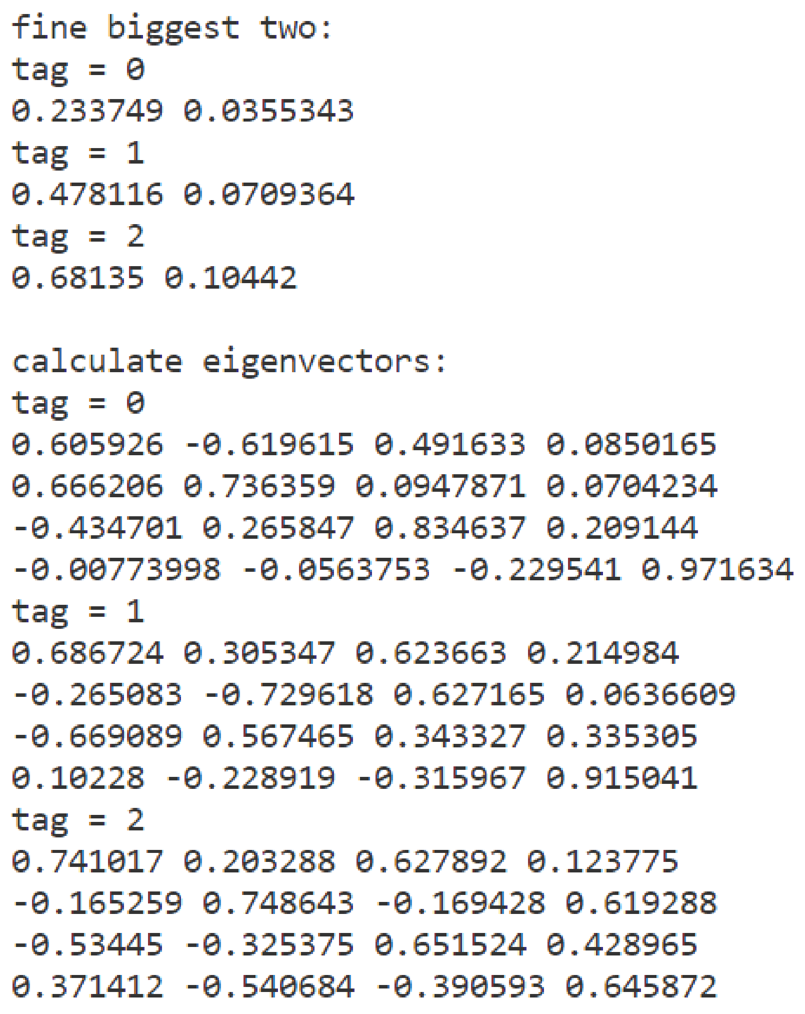
数据带入matlab验算，正确：

三、iris（鸢尾花）数据集

输出太多，一页屏幕放不下了。

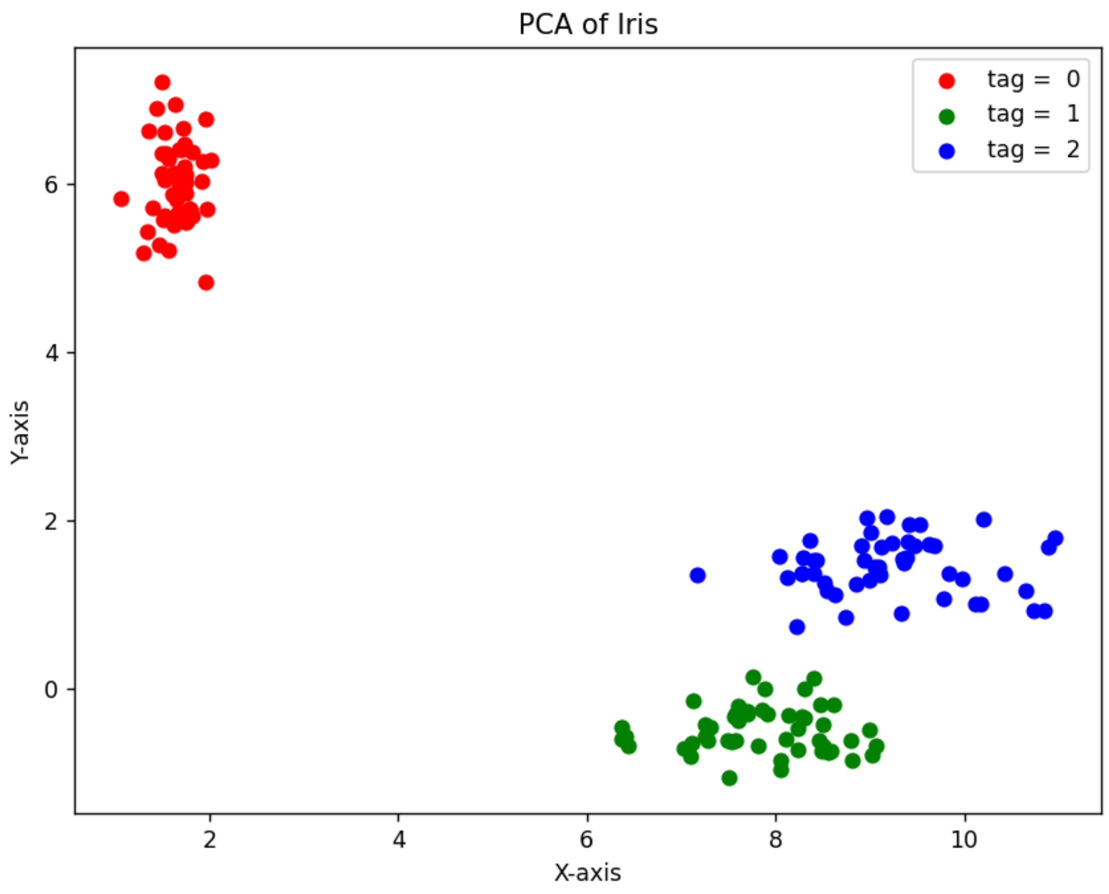


```
Caculate Covariance matrix:
tag = 0:
0.121764 0.098292 0.015816 0.010336
0.098292 0.142276 0.011448 0.011208
0.015816 0.011448 0.029504 0.005584
0.010336 0.011208 0.005584 0.011264
tag = 1:
0.261104 0.08348 0.17924 0.054664
0.08348 0.0965 0.081 0.04038
0.17924 0.081 0.2164 0.07164
0.054664 0.04038 0.07164 0.038324
tag = 2:
0.396256 0.091888 0.297224 0.048112
0.091888 0.101924 0.069952 0.046676
0.297224 0.069952 0.298496 0.047848
0.048112 0.046676 0.047848 0.073924

Solve eigenvalues by Jacobi:
tag = 0
0.0355343 0.233749 0.0263122 0.00921219
tag = 1
0.478116 0.0536806 0.0709364 0.00959456
tag = 2
0.68135 0.10442 0.0512495 0.0335805
```



```
fine biggest two:
tag = 0
0.233749 0.0355343
tag = 1
0.478116 0.0709364
tag = 2
0.68135 0.10442

calculate eigenvectors:
tag = 0
0.605926 -0.619615 0.491633 0.0850165
0.666206 0.736359 0.0947871 0.0704234
-0.434701 0.265847 0.834637 0.209144
-0.00773998 -0.0563753 -0.229541 0.971634
tag = 1
0.686724 0.305347 0.623663 0.214984
-0.265083 -0.729618 0.627165 0.0636609
-0.669089 0.567465 0.343327 0.335305
0.10228 -0.228919 -0.315967 0.915041
tag = 2
0.741017 0.203288 0.627892 0.123775
-0.165259 0.748643 -0.169428 0.619288
-0.53445 -0.325375 0.651524 0.428965
0.371412 -0.540684 -0.390593 0.645872
```

使用matplotlib作出原数据集投影到二维平面上的图像:

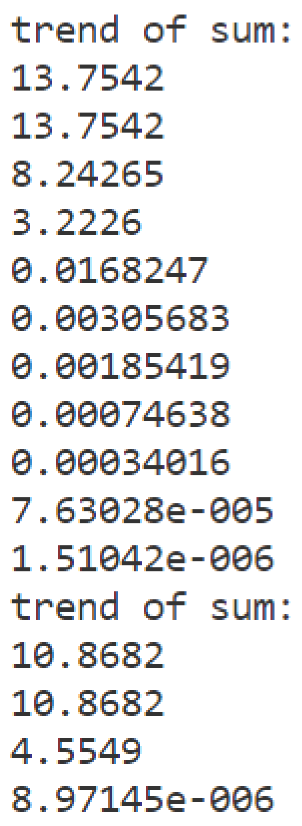


可以看到，对Iris数据集进行PCA分析之后，起到了聚类的效果

四、分析Jacobi算法：

使用上面随机生成的矩阵，分别查看AA^T和A^TA，在Jacobi算法中，非对角元素的平方和的变化：

可以看到，sum缩小的速度是非常快的，在10^1的数量级次数下，就减少到10^{-6}数量级



```
trend of sum:
13.7542
13.7542
8.24265
3.2226
0.0168247
0.00305683
0.00185419
0.00074638
0.00034016
7.63028e-005
1.51042e-006
trend of sum:
10.8682
10.8682
4.5549
8.97145e-006
```

五、核心0.代码Jacobi算法

算法过程参考课本

```cpp
//Jacobi算法求矩阵特征值
vector<double> Jacobi(vector<vector<double>> matrix){
    int n = matrix.size();  //矩阵阶数
    double epsilon = 1;
    int p,q;
    vector<double> eigenvalue0(n);
    vector<double> eigenvalue1(n);
    vector<vector<double>> matrix0 = matrix;
    vector<vector<double>> matrix1 = matrix;

    while (epsilon > e){
        double max_element = 0;
        for (int i = 0;i < n;i++){
            for (int j = i + 1;j<n;j++){
                if (abs(matrix1[i][j]) > max_element){
                    max_element = abs(matrix1[i][j]);
                    p = i; q = j;
                }
            }
        }
        double s = (matrix1[q][q]-matrix1[p][p])/(2*matrix1[p][q]);
        double t;
        if (s == 0){
            t = 1;
        }else{
            double t1 = -s - sqrt(s*s+1);
            double t2 = -s + sqrt(s*s+1);
            t = abs(t1) > abs(t2) ? t2 : t1;
        }

        double c = 1/sqrt(1+t*t);
        double d = t/sqrt(1+t*t);


        for (int i = 0;i<n;i++){
            for (int j = 0;j<n;j++){
                matrix0[i][j] = matrix1[i][j];
            }
        }

        for (int i = 0;i < n;i++){
            if (i == p || i == q) continue;
            matrix1[i][p] = c*matrix0[p][i] - d*matrix0[q][i];
            matrix1[p][i] = matrix1[i][p];
            matrix1[i][q] = c*matrix0[q][i] + d*matrix0[p][i];
            matrix1[q][i] = matrix1[i][q];
        }
        matrix1[p][p] = matrix0[p][p] - t*matrix0[p][q];
        matrix1[q][q] = matrix0[q][q] + t*matrix0[p][q];
        matrix1[p][q] = 0;
        matrix1[q][p] = 0;

        for (int i = 0;i < n;i++){
            eigenvalue0[i] = eigenvalue1[i];
            eigenvalue1[i] = matrix1[i][i];
        }

        epsilon = norm(eigenvalue0,eigenvalue1);

    }

    return eigenvalue1;
}
```