稀疏矩阵特征值项目报告

伪代码

Power Method

> 输入:
>
> - mat: 一个 n × n 的矩阵
>
> 输出:
>
> - powerMold1: 估计的最大特征值的模
> - powerEigVector: 估计的最大特征值对应的特征向量
>
> 过程:
>
> 1. 初始化 itTimes = 1（迭代次数）
> 2. 获取矩阵的维度 n
> 3. 初始化 powerInitVector 为一个元素都为 1 的 n 维列向量（初始向量归一化）
> 4. 初始化 powerEigVector 为一个元素都为 0 的 n 维列向量（初始特征向量为零向量）
> 5. 初始化 powerTol = 10 ^ -12（误差限）
> 6. 初始化 maxIt = 10000（最大迭代次数）
> 7. 初始化 powerErr = 10（初始误差）
> 8. 初始化 powerMold1 = 1（特征值模的初始值）
> 9. 初始化 powerMold2 = 1
> 10. 当 powerErr powerTol 且 itTimes <= maxIt 时执行以下步骤：
>   10.1. 将 powerEigVector = mat * powerInitVector（矩阵乘以当前向量，得到新向量）
>   10.2. 将 powerMold2 = max(abs(powerEigVector))（计算向量模的最大值）
>   10.3. 将 powerInitVector = powerEigVector / powerMold2（归一化新向量）
>   10.4. 将 powerErr = abs(powerMold1 - powerMold2)（计算误差）
>   10.5. 将 powerMold1 = powerMold2（更新特征值的模）
>   10.6. 将 itTimes = itTimes + 1（迭代次数加 1）
> 11. 输出 powerMold1 和 powerEigVector

QR算法

主体

输入：

- A: 一个 n × n 的矩阵
- k: 需要估计的前 k 个特征值

输出：

- a: 估计的前 k 个特征值

过程：

1. 初始化 maxIt = 10000（最大迭代次数）
2. 初始化 tol = 10 ^ -12（误差限）
3. 将 A0 = A（初始矩阵）
4. 将 a0 = diag(A)（初始特征值的估计）
5. 使用 Givens 函数对 A 进行 QR 分解，得到 Q 和 R
6. 将 A = R * Q（更新矩阵 A）
7. 将 a = diag(A)（更新特征值的估计）
8. 初始化 n = 1（迭代次数）
9. 当 max(abs(a - a0)) tol 且 n <= maxIt 时执行以下步骤：
   9.1. 将 a0 = a（更新特征值的估计）
   9.2. 使用 Givens 函数对 A 进行 QR 分解，得到 Q 和 R
   9.3. 将 A = R * Q（更新矩阵 A）
   9.4. 将 a = diag(A)（更新特征值的估计）
   9.5. 将 n = n + 1（迭代次数加 1）
10. 将 a = sort(abs(a), 'descend')（对特征值的估计按降序排序）
11. 将 a = a(1:k)（选取前 k 个特征值作为最终结果）
12. 输出 a

Givens函数

> 输入：
>
> - A: 一个 n × n 的矩阵
>
> 输出：
>
> - Q: Givens 变换得到的正交矩阵
> - R: Givens 变换得到的上三角矩阵
>
> 过程：
>
> 1. 获取矩阵的维度 n
> 2. 初始化 Q 为一个 n × n 的单位矩阵
> 3. 对于 j 从 1 到 n-1 进行循环：
>   3.1. 对于 i 从 n 到 j+1 进行循环：
>   3.1.1. 如果 A(i, j) 不等于 0，则执行 Givens 变换
>   3.1.1.1. 初始化 G 为一个 n × n 的单位矩阵
>   3.1.1.2. 获取第 j 行第 j 列元素，并将其赋值给 a
>   3.1.1.3. 获取第 i 行第 j 列元素，并将其赋值给 b
>   3.1.1.4. 计算旋转角度的模长 c = sqrt(a^2 + b^2)
>   3.1.1.5. 计算旋转角度的余弦值 cos_theta = a / c
>   3.1.1.6. 计算旋转角度的正弦值 sin_theta = b / c
>   3.1.1.7. 更新 G 的第 i 行第 i 列元素为 cos_theta
>   3.1.1.8. 更新 G 的第 j 行第 j 列元素为 cos_theta
>   3.1.1.9. 更新 G 的第 i 行第 j 列元素为 -sin_theta
>   3.1.1.10. 更新 G 的第 j 行第 i 列元素为 sin_theta
>   3.1.1.11. 使用 Givens 变换更新矩阵 A，即 A = G * A
>   3.1.1.12. 使用 Givens 变换更新矩阵 Q，即 Q = Q * G'
> 4. 最终的 A 矩阵即为 R 矩阵
> 5. 输出 Q 和 R

算法与代码实现

幂法

算法原理

有些实际问题往往不需要求出所有的特征值，只需要绝对值最大的主特征值。**幂法**是计算一个矩阵的主特征值及其特征向量的迭代法，特别适用于大型的稀疏矩阵。
设 λ_1 是n阶实矩阵A的主特征值，对应特征向量是 x_1 ,设A有n个线性无关的特征向量，则对于任意的非零向量 $v_0 \in R^n$ 可以用特征向量基表示：

$$v_0 = \alpha_1 x_1 + \alpha_2 x_2 + \ldots\ldots \alpha_n x_n$$

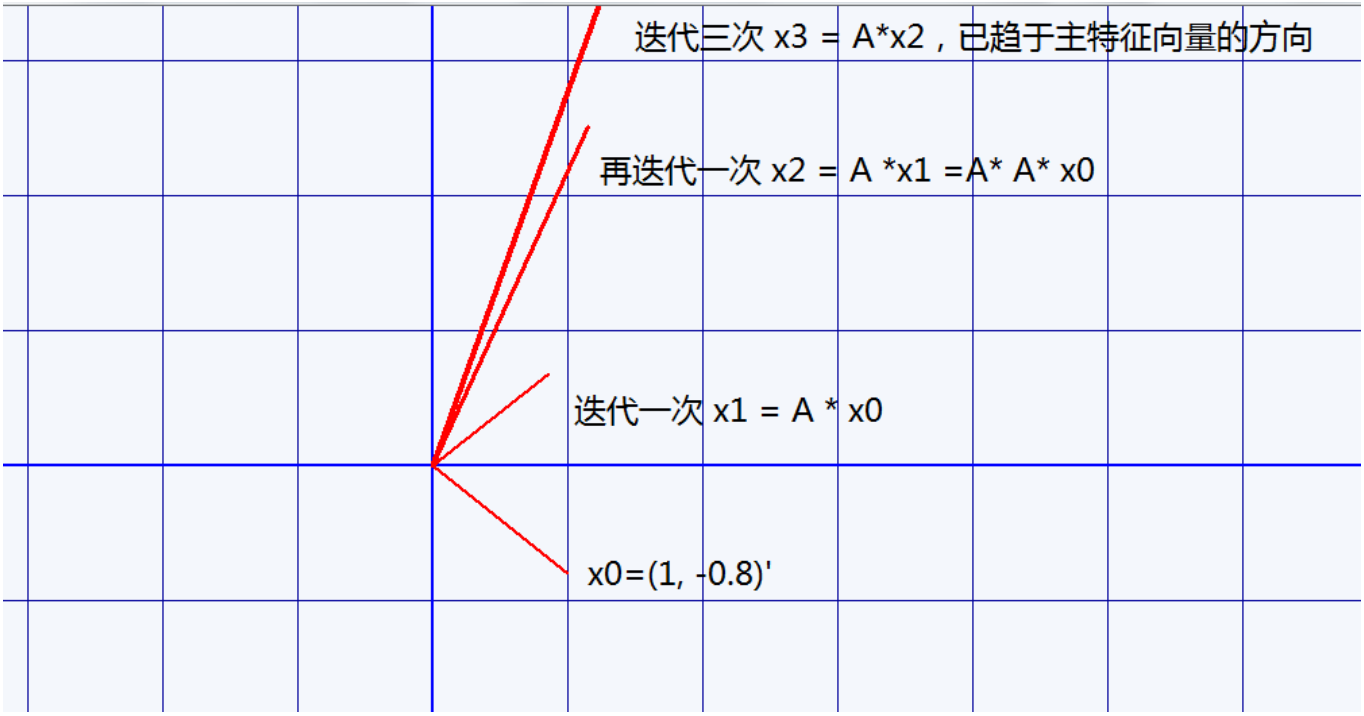
用A左乘得:

$$v_1 = Av_0 = \sum_{i=1}^{n} \alpha_i A x_i = \sum_{i=1}^{n} \alpha_i \lambda_i x_i$$

若进行k次左乘A得：

$$
\begin{aligned}
v_k &= A^k v_0 \\
&= \alpha_1 \lambda_1^k x_1 + \alpha_2 \lambda_2^k x_2 + \ldots\ldots + \alpha_n \lambda_n^k x_n \\
&= \lambda_1^k (\alpha_1 x_1 + \alpha_2 (\frac{\lambda_2}{\lambda_1})^k x_2 + \ldots\ldots + \alpha_n (\frac{\lambda_n}{\lambda_1})^k x_n)
\end{aligned}
$$

当$k \to \infty$ 时，由于$|\lambda_1| > |\lambda_2|$,所以$v_k = \alpha_1 \lambda_1^k x_1$, $\lambda_1 = \frac{(v_k)_i}{(v_{k-1})_i}$,其中主特征值是$v_k$ 和 v_{k-1} 的某个分量的比值。此时，v_k 可以近似的看成 λ_1 的特征向量，因为α和λ 都是常数，则$max\{v_k\} = \lambda_1$。



主要代码实现

在误差限和最大迭代次数内，依次更新向量，并取该向量的最大值，再依次更新特征值，满足条件后跳出循环。

```MATLAB
 while powerErr>powerTol && (itTimes <= maxIt)  %Calculating the greatest eigenvalue
and the corresponding eigenvector.
   powerEigVector=mat*powerInitVector;  %  矩阵乘以当前向量，得到新向量
   powerMold2=max(abs(powerEigVector));  %  计算向量模的最大值
   powerInitVector=powerEigVector/powerMold2;  %  归一化新向量
   powerErr=abs(powerMold1-powerMold2);  %  误差
   powerMold1=powerMold2;  %  更新特征值的模
   itTimes=itTimes+1;  %  迭代次数加1
 end
```

QR算法

算法理论

QR算法可以看做PowerMethod的矩阵化扩展，是一种用递归法求解矩阵特征值和特征向量的算法，在合适的递归轮数后，QR算法会得到一个上三角矩阵，进而求得特征值，Q_k可以进为矩阵的特征向量矩阵Q.QR算法具体如下：

$$
\begin{aligned}
&set\ A_0 = A \\
&for\ i = 1, 2 \cdots k(until convergence) \\
&\quad compute\ A_{k-1} = Q_k R_k \\
&\quad A_k = R_k Q_k \\
&end
\end{aligned}
$$

Givens旋转

Givens变换是一种将n维向量**x**在第(i，k)两个维度确定的坐标平面内进行旋转（从而将其中一个分量化0）的变换，因此它又叫平面旋转变换。

Algorithm 1 Givens trans

```
if b = 0 then
    c = 1
    S = 0
else
    if |b| > |a| then
        t = −a/b
        s = 1/√(1+t²)
        c = st
    else
        t = −b/a
        c = 1/√(1+t²)
        s = ct
    end if
end if
```

代码实现

```MATLAB
function [Q,R] = Givens(A)
n = size(A,1);  %  获取矩阵维度

Q = eye(n);  %  初始化Q为单位矩阵

for j = 1:n-1  %  遍历每一列
    for i = n:-1:j+1  %  从最后一行开始向上遍历
        if A(i,j)~=0  %  如果 A(i,j) 不等于 0，则执行 Givens 变换
                G = eye(n);  %  初始化 G 为单位矩阵
                a = A(j,j);  %  获取第 j 行第 j 列元素
                b = A(i,j);  %  获取第 i 行第 j 列元素
                c = sqrt(a^2 + b^2);  %  计算旋转角度的模长
                cos_theta = a/c;  %  计算旋转角度的余弦值
                sin_theta = b/c;  %  计算旋转角度的正弦值
                G(i,i) = cos_theta;  %  更新 G 的第 i 行第 i 列元素
                G(j,j) = cos_theta;  %  更新 G 的第 j 行第 j 列元素
                G(i,j) = -sin_theta;  %  更新 G 的第 i 行第 j 列元素
                G(j,i) = sin_theta;  %  更新 G 的第 j 行第 i 列元素
                A = G*A;  %  使用 Givens 变换更新矩阵 A
                Q = Q*G';  %  使用 Givens 变换更新矩阵 Q
        end
    end
end
R = A;  %  % 最终的 A 矩阵即为 R 矩阵
end
```

Arnoldi算法

算法理论

Arnoldi算法是一种用于计算大型稀疏或密集矩阵的Krylov子空间的迭代方法。它通常用于计算矩阵的特征值和特征向量，尤其在计算几个最大或最小的特征值时非常有效。Arnoldi算法的基本思想是通过构建一个正交基来逐步生成Krylov子空间。Krylov子空间是由初始向量和矩阵乘法生成的向量的线性组合构成的空间。使用Arnoldi算法，可以在迭代过程中逐步生成这些向量，并在每一步保持它们的正交性。

算法 2.1 基于 Gram-Schmidt 正交化的 Arnoldi 过程

给定非零向量 r, 计算 $v_1 = r/\|r\|_2$
for $j = 1, 2, \ldots, m-1$ **do**
    $w_j = Av_j$
    for $i = 1, 2, \ldots, j$ **do**
        $h_{ij} = (w_j, v_i)$
    end for
    $w_j = w_j - \sum_{i=1}^{j} h_{ij} v_i$
    $h_{j+1,j} = \|w_j\|_2$
    if $h_{j+1,j} = 0$ **then**
        break
    end if
    $v_{j+1} = w_j / h_{j+1,j}$
end for

```MATLAB
function  a= Arnoldi(A,k)
    tol = 1e-12;  %  误差限
    [m,n] = size(A);  %  矩阵大小
    p0=10;  %  Arnoldi迭代最大步数
    p = p0-1;  %  实际迭代步数
    Q = zeros(n,1+p);  %  存储 Arnoldi 过程中的正交基向量
    H = zeros(1+p);  %  存储 Hessenberg 矩阵
    Q(:,1) = rand(n,1);  %  初始化第一个正交基向量为随机向量
    Q(:,1) = Q(:,1)/norm(Q(:,1));   %  归一化第一个正交基向量
    H = Q(:,1)'*A*Q(:,1);  %  计算第一个 Hessenberg 矩阵元素
    q = A*Q(:,1) - Q(:,1)*H;  %  计算第一个残差向量

for m = 1:p
    beta = norm(q);  %  计算残差向量的模
    if beta <= tol
        Q = Q(:,1:1+m-1);  %  收敛时，截断正交基向量矩阵 Q
        H = H(1:1+m-1,1:1+m-1);   %  收敛时，截断 Hessenberg 矩阵 H
        return;
    end

    Q(:,1+m) = q/beta;  %  归一化残差向量得到新的正交基向量
    H(1+m,1+m-1) = beta;  %  更新 Hessenberg 矩阵元素
    w = A*Q(:,1+m);  %  计算新的向量
    H(1:1+m, 1+m) = Q(:,1:1+m)'*w;  %  更新 Hessenberg 矩阵元素
    q = w - Q(:,1:1+m)*H(1:1+m, 1+m);  %  计算新的残差向量


    % 用校正步骤保持正交性
    c = Q(:,1:1+m)'*q;
    q = q - Q(:,1:1+m)*c;
    H(1:1+m, 1+m) = H(1:1+m, 1+m) + c;
end

a=QRMethod(H,k);  %  调用 QR 方法计算 Hessenberg 矩阵的特征值
```

性能分析

时间复杂度

幂法、QR算法、Arnoldi算法的时间复杂度分别为O(k *(n^2 + n))*、*O(k* n^3)、O(k^3)，其中k为迭代次数

空间复杂度

幂法、QR算法、Arnoldi算法的空间复杂度分别为O(n)、O(n^2)、O(n^2+k^2)，其中k为迭代次数

结果分析

幂法

- 优点：简单而直观的方法，易于理解和实现。它对于具有主导特征值的矩阵收敛较快。
- 缺点：只能计算矩阵的最大特征值及其对应的特征向量，对于其他特征值和特征向量无法得到准确的结果。

QR算法

- 优点：能够计算矩阵的所有特征值及其对应的特征向量。具有良好的数值稳定性和收敛性。
- 缺点：计算复杂度较高，特别是在处理大型矩阵时，迭代次数较多，计算时间较长。

Arnoldi算法

- 优点：适用于计算稀疏矩阵的特征值和特征向量。可以计算矩阵的部分特征值和特征向量，对于大型稀疏矩阵具有较好的效率。
- 缺点：对于非稀疏矩阵的效果相对较差，对于具有多个主导特征值的矩阵收敛较慢。

在合适的误差限和迭代次数下，几种算法都可以算出与Matlab提供的eig和eigs函数算出的特征值相同的特征值；其中幂法速度快于QR算法，但幂法只能算最大特征值；而QR算法虽然可以算出所有特征值，但当矩阵维度很大时，运算时间便会很长，甚至无法算出结果；Arnoldi算法则优于两者，兼具两者的优点，在速度快的同时，又能算出所有特征值。