刘程华 2018011687 计91

1、Let A be an $n\times n$ idempotent matrix. Prove on p.6 that A is diagonalizable.

下面列举三个可对角化矩阵A的等价条件：

1. 每一特征值 $\lambda_i\in\sigma(A)$ 的代数重数等于几何重数，即 $dimN\left((A-\lambda_i I)^n\right)=\dim N\left(A-\lambda_i I\right)$ 。
2. 每一特征值 $\lambda_i\in\sigma(A)$ 的指标(index)等于1，也就是说 A 的Jordan矩阵的每一个Jordan分 块的阶数为1。
3. 最小多项式为 $m_A(t)=(t-\lambda_1)\cdots(t-\lambda_k)$, 也就是说 $(A-\lambda_1 I)\cdots(A-\lambda_k I)=0$

P为一个幂等矩阵(或称投影矩阵)。特征值 λ 满足 $\lambda^2=\lambda$, 因此 P 的特征值为0或1。下面我们用三种方法证明幂等矩阵可对角化，习惯导致，题目中的A在叙述中为P。

证明1： 根据定义, $(I-P)^2=I-2P+P^2=I-P$, 即知 $I-P$ 是幂等矩阵。再者，$P^n=P^{n-1}=\cdots=P$, 类似地， $(I-P)^n=(I-P)^{n-1}=\cdots=I-P$。对于 $\lambda=0$, $\dim N\left(P^n\right)=\dim N(P)$; 对于 $\lambda=1$, $\dim N\left((P-I)^n\right)=\dim N(P-I)$

证明2： 考虑Jordan形式 $P=SJS^{-1}$ ，其中Jordan矩阵为

$$J=\begin{bmatrix}J_1 & & \\ & \ddots & \\ & & J_m\end{bmatrix}=J_1\oplus\cdots\oplus J_m$$

每一 J_i 是 $n_i\times n_i$ 阶Jordan分块，形式如下：

$$J_i=\begin{bmatrix}\lambda & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1\\ & & & \lambda\end{bmatrix}$$

写出 $P^2=SJ^2S^{-1}=S\left(J_1^2\oplus\cdots\oplus J_m^2\right)S^{-1}$, 故 $P^2=P$等价于 $J_i^2=J_i, 1\le i\le m$ 。观察Jordan分块的幂矩阵形式可推断每一 J_i 是 1×1 阶。

证明3： 分开三种情况讨论。若 $\sigma(P)=\{0,1\}, P^2=P$ 或写为 $P(P-I)=0$, 则最小多项式为 $m_P(t)=t(t-1)$。若 $\sigma(P)=\{0\}$, 对于任一 $\mathbf{x}\in\mathbb{C}^n$, 设 $P\mathbf{x}=\mathbf{y}$, 即有 $P\mathbf{x}=P^2\mathbf{x}=P\mathbf{y}$, 推得 $P\mathbf{y}=\mathbf{y}$。但1不是 P 的特征值，因此 $\mathbf{y}=\mathbf{0}$。换句话说， $N(P)=\mathbb{C}^n$意味 $P=0$, 故 $m_P(t)=t$。若 $\sigma(P)=\{1\}$, 则 $\sigma(I-P)=\{0\}$ 同样道理， $N(I-P)=\mathbb{C}^n$, 即得 $P=I$, 故 $m_P(t)=t-1$。

2、Prove the four equations on p.7.

If P is a projector, $I-P$ is also a projector, called the complementary projector to P

$I-P$ project onto $R(I-P)$，P是幂等矩阵，我们证明以下四个等式

- $R(I-P)=N(P)$

  若 $\mathrm{x}\in N(P)$，则 $P\mathrm{x}=0$，故 $\mathrm{x}=\mathrm{x}-P\mathrm{x}=(I-P)\mathrm{x}$，亦即 $\mathrm{x}\in R(I-P)$；若 $\mathbf{x}\in R(I-P)$, 就有 $\mathbf{x}=(I-P)\mathbf{y}$, 故 $P\mathbf{x}=P(I-P)\mathbf{y}=0\mathbf{y}=\mathbf{0}$, 即 $\mathbf{x}\in N(P)$ 合并以上结果证明 $N(P)=R(I-P)$
- $R(P)=N(I-P)$ 我们有

  $$(I-P)^2=I-2P+P^2=I-P$$

  $I-P$ 为幂等距阵，故 $N(I-P)=R(I-(I-P))=R(P)$
- $R(P)\cap N(P)=\{0\}$ 若 $\mathrm{x}\in R(P)\cap N(P)$, 就有 $\mathrm{y}\in\mathbb{R}^n$ 使得 $\mathrm{x}=P\mathrm{y}$, 由第二条知存在z使得 $\mathrm{x}=(I-P)\mathrm{z}$因此$\mathrm{z}=P\mathrm{y}+P\mathrm{z}$, 等号两边左乘 P, 立得

  $$P\mathbf{z}=P^2\mathbf{y}+P^2\mathbf{z}=P\mathbf{y}+P\mathbf{z}$$

  故 $\mathrm{x}=P\mathrm{y}=0$
- $N(I-P)\cap N(P)=\{0\}$ 由第二条和第三条立得。

3、Turn pseudo code of CGS and MGS on p. 22 into a R function and offer your demo examples. (Meaning that write your own function that can perform CGS or MGS, as you wish. Then give some numeric examples to show how to use it.) Finally, search online the existing R **function(s) and check your results.**

code：

```
#CGS
classical_gram_schmidt<-function (A, tol = .Machine$double.eps^0.5)
{
  stopifnot(is.numeric(A), is.matrix(A))
  m <- nrow(A)
  n <- ncol(A)
  if (m < n)
    stop(" rows of must be greater or equal colums! ")
  Q <- matrix(0, m, n)
  R <- matrix(0, n, n)
  for (k in 1:n) {
    Q[, k] <- A[, k]
    if (k > 1) {
      for (i in 1:(k - 1)) {
        R[i, k] <- t(Q[, i]) %*% Q[, k]
        Q[, k] <- Q[, k] - R[i, k] * Q[, i]
      }
    }
    R[k, k] <- (t(Q[, k]) %*% Q[, k ])^0.5
    if (abs(R[k, k]) <= tol)
      stop("Matrix does not have full rank.")
    Q[, k] <- Q[, k]/R[k, k]
  }
  return(list(Q = Q, R = R))
}
#MGS
modified_gram_schmidt<-function (A, tol = .Machine$double.eps^0.5)
{
  stopifnot(is.numeric(A), is.matrix(A))
  m <- nrow(A)
  n <- ncol(A)
  if (m < n)
    stop(" rows of must be greater or equal colums! ")
  Q <- matrix(0, m, n)
  R <- matrix(0, n, n)
  for (k in 1:n) {
    Q[, k] <- A[, k]
  }
  for (i in 1:n) {
        R[i, i] <- (t(Q[, i]) %*% Q[, i])^0.5
        if (abs(R[i, i]) <= tol)
          stop("Matrix does not have full rank.")
        Q[, i] <- Q[, i]/R[i, i]
        if(i<n)
        for(j in (i+1):n){
          R[i,j]<- (t(Q[, i]) %*% Q[, j])
          Q[, j]<-Q[, j]-R[i,j]*Q[,i]
        }
  }
  return(list(Q = Q, R = R))
}
```

上述代码考虑了不满秩和行数小于列数的情况，鲁棒性较好。避免赘述，下面只列举了一个数值案例。

```
A <- matrix(c(4, 3, -2, 1), ncol = 2)
> classical_gram_schmidt(A)
$Q
     [,1] [,2]
[1,]  0.8 -0.6
[2,]  0.6  0.8


$R
     [,1] [,2]
[1,]    5   -1
[2,]    0    2
###################
> modified_gram_schmidt(A)
$Q
```

```
     [,1] [,2]
[1,]  0.8 -0.6
[2,]  0.6  0.8

$R
     [,1] [,2]
[1,]    5   -1
[2,]    0    2
###################
> library(pracma)
> gramSchmidt(A)
$Q
     [,1] [,2]
[1,]  0.8 -0.6
[2,]  0.6  0.8

$R
     [,1] [,2]
[1,]    5   -1
[2,]    0    2
```

4、Follow our deduce fashion on p.24 **to investigate all the possible errors then make comparisons of CGS versus MGS. You are welcome to comment on the differences.**

$a_1 = (1, \varepsilon, 0, 0)^T, a_2 = (1, 0, \varepsilon, 0)^T, a_3 = (1, 0, 0, \varepsilon)^T$

CGS: $q_1 = (1, \varepsilon, 0, 0)^T$ $q_2 = (0, -1, 1, 0)^T/\sqrt{2}$ $q_3 = (0, -1, 0, 1)^T/\sqrt{2}$

$$\text{MGS: } \begin{aligned} q_1 &= (1, \varepsilon, 0, 0)^T \\ q_2 &= (0, -1, 1, 0)^T/\sqrt{2} \\ q_3 &= (0, -1, -1, 2)^T/\sqrt{6} \end{aligned}$$

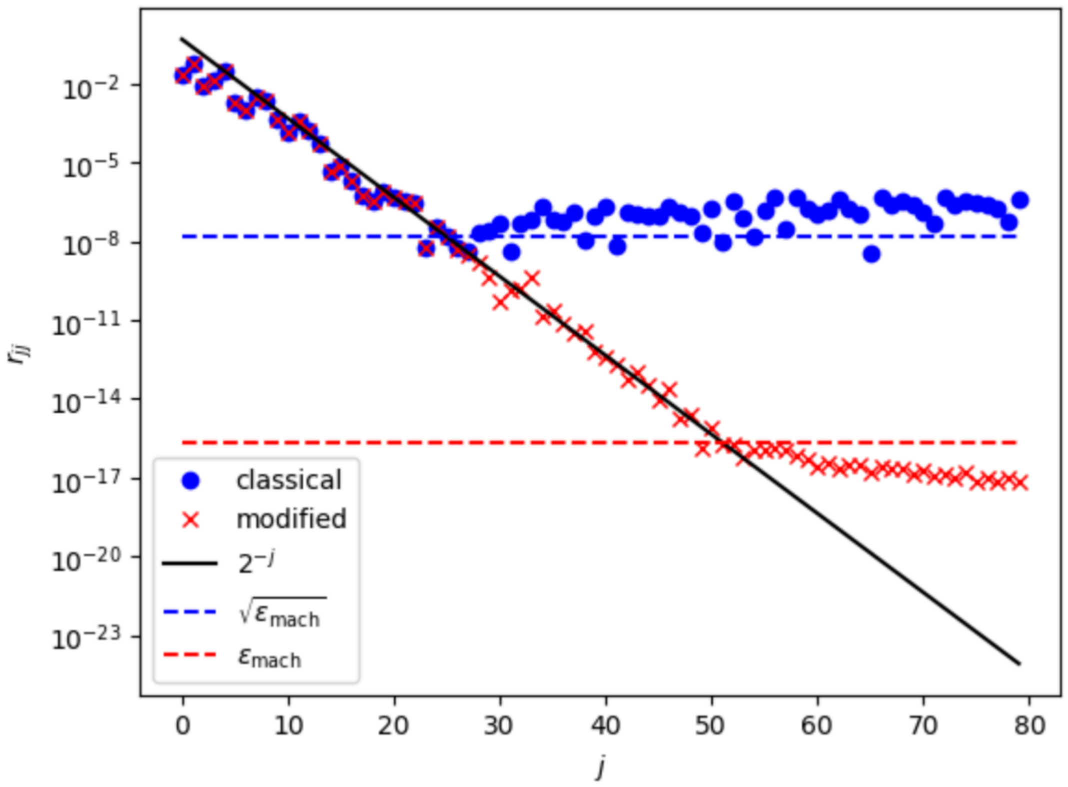
检查正交性： CGS $q_1^T q_3 = -\epsilon/\sqrt{2}$ MGS $q_1^T q_3 = -\epsilon/\sqrt{6}$

MGS中$q_1^T q_3$更接近0，因此MGS效果更好。

对于CGS来说，在构建第n个正交基时，n+1及以后的向量是不可见的。在计算的过程中不可避免地引入误差，而后面的向量只有在其被计算的时候才使用，这时误差可能已经通过传递积累到一定程度了，所以说其稳定性较差。MGS的计算量越往后是逐步减少的，因此前面计算积累的误差的影响就不会剧烈地扩散开，所以MGS可以使求得的矩阵**Q**的正交性更好，同时矩阵**R**的误差也被控制在计算机精度的水平。下面我们通过实验验证一下。

借助python，我们构造一个正方形矩阵（80×80）奇异值介于2^{-1}和 2^{-80}，使用不同方法执行分解，并检查结果的准确性和正交性。

绘制结果如下图，由图可知CGS的误差水平大约为计算机精度的平方根，MGS的误差水平大约为计算机精度。CGS误差的累积速度太快而无法保持稳定性，MGS是更好的。



Code:

```
using LinearAlgebra
U = qr(randn(80,80)).Q # U = random 80×80 orthogonal matrix
V = qr(randn(80,80)).Q # V = random 80×80 orthogonal matrix
Σ = Diagonal(2.0 .^ (-1
                     :-1:-80)) # Σ = diagonal matrix with entries 2^-1, 2^-2,
..., 2^-80
A = U * Σ * V'
# Classical Gram-Schmidt (Trefethen algorithm 7.1), implemented in the simplest
way
# (We could make it faster by unrolling loops to avoid temporaries arrays etc.)
function clgs(A)
    m,n = size(A)
    T = float(eltype(A))
    Q = similar(A, T)
    R = zeros(T,n,n)
    @views for j = 1:n
        aⱼ = A[:,j]
        vⱼ = copy(aⱼ) # use copy so that modifying vⱼ doesn't change aⱼ
        for i = 1:j-1
            qᵢ = Q[:,i]
            R[i,j] = qᵢ'aⱼ
            vⱼ .-= R[i,j] .* qᵢ
        end
        R[j,j] = norm(vⱼ)
        Q[:,j] .= vⱼ ./ R[j,j]
    end
    return Q, R
```

```julia
end

# Modified Gram-Schmidt (Trefethen algorithm 8.1)
function mgs(A)
    m,n = size(A)
    T = float(eltype(A))
    Q = similar(A, T)
    R = zeros(T,n,n)
    @views for j = 1:n
        aⱼ = A[:,j]
        vⱼ = copy(aⱼ)
        for i = 1:j-1
            qᵢ = Q[:,i]
            R[i,j] = qᵢ'vⱼ # ⟵ NOTICE: mgs has vⱼ, clgs has aⱼ
            vⱼ .-= R[i,j] .* qᵢ
        end
        R[j,j] = norm(vⱼ)
        Q[:,j] .= vⱼ ./ R[j,j]
    end
    return Q, R
end

QC, RC = clgs(A);
QM, RM = mgs(A);

using PyPlot

n = size(A,2)
semilogy(diag(RC), "bo")
semilogy(diag(RM), "rx")
semilogy(2.0 .^ -(1:n), "k-")
semilogy(fill(sqrt(eps()), n), "b--")
semilogy(fill(eps(), n), "r--")
legend(["classical","modified",L"2^{-j}", L"\sqrt{\epsilon_\mathrm{mach}}",
L"\epsilon_\mathrm{mach}"], loc="lower left")
ylabel(L"r_{jj}")
xlabel(L"j")
```