AMATH 584 Assignment 4

Tyler Chen

Exercise 9.2

In Experiment 2, the singular values of A match the diagonal elements of a QR factor R approximately. Consider now a very different example. Suppose $Q = I$ and $A = R$, the $m \times m$ matrix (a Toeplitz matrix) with 1 on the main diagonal, 2 on the first superdiagonal, and 0 everywhere else.

(a) What are the eigenvalues, determinant, and rank of A?

(b) What is A^{-1}?

(c) Give a nontrivial upper bound on σ_m, the m-th singular value of A.

Solution

(a) Recall that the eigenvalues of a triangular matrix are the diagonal entries and the determinant of any matrix is the product of the eigenvalues. Then clearly all the eigenvalues are 1, and the determinant is 1. Since A has nonzero determinant A is full rank (rank m).

(b) We have,

$$a_{ij} = \begin{cases} 0 & i > j \\ 1 & i = j \\ 2 & i = j - 1 \\ 0 & i < j - 1 \end{cases}$$

Define,

$$b_{ij} = \begin{cases} 0 & i > j \\ (-2)^{i-j} & i \leq j \end{cases}$$

That is, B has 1 on the main diagonal, -2 on the fist superdiagonal, 4 on the second super diagonal, 8 on the 3rd superdiagonal, and so on.

I claim that $A^{-1} = B$.

From above we see $a_{ii} = 1$, $a_{i,i+1} = 2$, and $a_{ij} = 0$ for $j \notin \{i, i+1\}$. Then we can write AB as,

$$(ab)_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj} = a_{ii} b_{ij} + a_{i,i+1} b_{i+1,j}$$

Suppose $i < j$. Then,

$$(ab)_{ij} = 1 \cdot (-2)^{i-j} + 2 \cdot (-2)^{i+1-j} = (-2)^{i-j} - (-2)^{i-j} = 0$$

Now suppose $i = j$. Then,

$$(ab)_{ij} = a_{ii} b_{ii} + a_{i,i+1} b_{i+1,i} = 1 \cdot 1 + 2 \cdot 0 = 1$$

Finally, suppose $i > j$. Then,

$$(ab)_{ij} = 1 \cdot 0 + 2 \cdot 0 = 0$$

That is, $AB = I$ so $A^{-1} = B$.

(c) Write the SVD of A as $A = U\Sigma V^*$. Recall $Av_m = \sigma_m u_m$ for the smallest singular value σ_m of A. Then $A^{-1}u_m = v_m/\sigma_m$.

Since U and V are orthonormal, $\|u_m\| = \|v_m\| = 1$. Therefore,

$$\frac{1}{\sigma_m} = \left|\frac{1}{\sigma_m}\right| \|u_m\| = \left\|\frac{1}{\sigma_m} u_m\right\| = \left\|A^0 u_m\right\| \geq \left\|A^{-1}\right\| \|u_m\| = \left\|A^{-1}\right\|$$

Then, by definition of norm,

$$\begin{aligned}
\left\|A^{-1}\right\| &= \sup_{\|z\|=1} \left\|A^{-1}z\right\| \\
&\geq \left\|A^{-1}[0, ..., 0, 1]^T\right\| \\
&= \left\|[(-2)^{m-1}, ..., 4, -2, 1]^T\right\| \\
&= \sqrt{((-2)^{m-1})^2 + ... + (4)^2 + (-2)^2 + 1^2} \\
&\geq 2^{m-1}
\end{aligned}$$

Therefore,

$$\sigma_m \leq 2^{1-m}$$

Exercise 10.1

Determine the (a) eigenvalues, (b) determinant, and (c) singular values of a Householder reflector. For the eigenvalues, give a geometric argument as well as an algebraic proof.

Solution

Let $F = I - 2(vv^*)/(v^*v) \in \mathbb{C}^{n\times n}$ be a Householder reflector across a hyperplane H.

(a) If a vector lies in the n dimensional hyperplane H it will not be affected by F. Therefore it is an eigenvector with eigenvalue 1. Similarly, if a vector is in the one dimensional vector space spanned by v it will become its negative. Therefore it is an eigenvector with eigenvalue -1. Since span(v) and H have dimensions summing to n, then these are all the eigenvectors/values.

Suppose λ is an eigenvalue of F with eigenvector x. Write,

$$\lambda x = Fx = \left(I - 2\frac{vv^*}{v^*v}\right)x = x - 2\frac{v^*x}{vv^*}v$$

So,

$$(1-\lambda)x = 2\frac{v^*x}{v^*v}v$$

Suppose $v^*x = 0$. Then $\lambda = 0$ and x is any vector orthogonal to v.

Now suppose $v^*x \neq 0$. Then x is a constant multiple of v. Rearrange to,

$$(1-\lambda)x = 2\frac{v^*v}{v^*v}x = 2x$$

So $\lambda = -1$, and as stated above x is a constant multiple of v.

These correspond exactly with the geometric understanding above.

(b) The determinant is equal to the product of the eigenvalues. As explained above, the eigenvalue 1 occurs only once (if x is a scalar multiple of v). Therefore $\det(F) = -1$.

(c) Let x be any vector and define $c = 2(v^*x)/(v^*v)$.

Therefore,

$$\|Fx\|^2 = \left\|\left(I - 2\frac{vv^*}{v^*v}\right)x\right\|^2 = \|x - cv\|^2 = (x-cv)^*(x-cv) = x^*x - cx^*v - \overline{c}v^*x + \overline{c}cv^*v$$

Recalling that $v^*x = \overline{x^*v}$,

$$-cx^*v - \overline{c}v^*x + \overline{c}cv^*v = -2\frac{(v^*x)(x^*v)}{v^*v} - 2\frac{(x^*v)(v^*x)}{v^*v} + 4\frac{(v^*x)(x^*v)}{v^*x} = 0$$

Therefore, $\|Fx\|^2 = \|x\|^2$. Since norms are positive, $\|Fx\| = \|x\|$. This means F does not stretch any vectors (i.e. the image of a unit sphere is a unit sphere). This proves all singular values are 1.

Exercise 10.4

Consider the 2×2 orthogonal matrices

$$F = \begin{bmatrix} -c & s \\ s & c \end{bmatrix}, J = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$

where $s = \sin\theta$ and $c = \cos(\theta)$ for some θ. The first matrix has $\det(F) = -1$ and is a reflector. The special case of a Householder reflector in dimension 2. The second has $\det(J) = 1$ and effects a rotation instead of a reflection. Such a matrix is called a Givens rotation.

(a) Describe exactly what geometric effects left-multiplications by F and J have on the plane $\mathbb{R}^2$.

(b) Describe an algorithm for QR factorization that is analogous to Algorithm 10.1 but based on Givens rotations instead of Householder reflections.

Solution

(a) Since $\det(F) = -\cos^2(\theta) - \sin^2(\theta) = -1$, it is a reflector. Clearly it is not the identity, so in $\mathbb{R}^2$ it must reflect across a 1 dimensional vector space. Suppose v is in this vector space. Then $Fv = v$ so $v \in \ker(I - F)$.

Observe $v = [\sin(\theta/2), \cos(\theta/2)]^*$ satisfies this condition. So F is a reflection across $\pi/2 - \theta/2$.

Since $\det(J) = \cos^2(\theta)\sin^2(\theta) = 1$, J is a rotator. Observe $J[1,0]^* = [\cos(\theta), -\sin(\theta)]^*$. That is, J rotates $[1,0]^*$ clockwise by θ about the origin. Since rotators rotate all vectors by the same amount, left multiplication by J effects a rotation clockwise by θ about the origin.

(b) In QR factorization, the key is taking $v \to \|v\| e_1$ during each step of the process. Rather than using a reflector, we could use a rotator to take $v \to \|b\| e_1$.

However, we can only rotate in one planes. To use a single Givens rotation would require us to rewrite v with respect to another basis such that v and $\|v\| e_1$ were in some plane spanned by 2 basis vectors. Rather than doing this, we can successively rotates on the planes spanned first by the k and $(k-1)$ basis to eliminate an entry in the k-th component, then in the $(k-1)$ and $(k-2)$ dimensions to eliminate the $(k-1)$-th component, and so on.

Let $a_k = v^{(0)} = [v_1, ..., v_k]^*$ be the $m-k$ through m entries of the $k-th$ column of A.

Define,

$$\theta_{i+1} = \text{atan2}(v^{(i)}_{k-i}, v^{(i)}_{k-i-1})$$

to be the angle from the positive x axis to the point $[v^{(i)}_{k-i}, v^{(i)}_{k-i-1}]^T$. Then observe,

$$J(\theta)\begin{bmatrix} v^{(i)}_{k-i-1} \\ v^{(i)}_{k-i} \end{bmatrix} i = \begin{bmatrix} \cos(\theta)v^{(i)}_{k-i-1} + \sin(\theta)v^{(i)}_{k-i} \\ -\sin(\theta)v^{(i)}_{k-i-1} + \cos(\theta)v^{(i)}_{k-i} \end{bmatrix} = \begin{bmatrix} \cos(\theta)v^{(i)}_{k-i-1} + \sin(\theta)v^{(i)}_{k-i} \\ 0 \end{bmatrix}$$

Define,

$$v^{(i+1)} = Q_{ki}v^{(i)} = \begin{bmatrix} I_{k-i-2} & & \\ & J(-\theta_i) & \\ & & I_i \end{bmatrix} v^{(i)}$$

Let $Q_k = Q_{kk}...Q_{k2}Q_{k1}$ so that $Q_k v = [x, 0, ..., 0]^*$ for some entry x.

Then define,

$$E_k = \begin{bmatrix} I_{m-k} & \\ & Q_k \end{bmatrix}$$

Then apply $E_{m-n+1}...E_{m-2}E_{m-1}A$ to get something upper triangular.

I wasn't sure about the correctness of the above algorithm so I I thought it wouldn't be that annoying to implement this in scipy to check myself. After spending too much time debugging indices (now I understand why matlab is 1 indexed) we have some working code:

```
# returns Q,R
def givens_qr(A):
    np.set_printoptions(precision=5)
    m,n=np.shape(A)
    E=np.identity(m)
    for i in range(m-1):
        v=A[i:,i]
        k=m-i
        Qk=np.identity(k)
        for j in range(1,k):
            theta=np.arctan2(v[k-j],v[k-j-1])
            Qj=np.identity(k)
            Qj[k-j-1:k-j+1,k-j-1:k-j+1]=np.array([[sp.cos(theta),sp.sin
                (theta)],[-sp.sin(theta),sp.cos(theta)]])
            v=Qj@v
            Qk=Qj@Qk

        Ek=np.identity(m)
        Ek[m-k:m,m-k:m]=Qk

        E=Ek@E
        A=Ek@A

    return(A,sp.transpose(E))
```

Exercise 11.3

Take $m = 50, n = 12$. Using MATLAB's `linspace` function, define t to be the m-vector corresponding to linearly spaced grid points form 0 to 1. Using MATLAB's `vander` and `fliplr`, define A to be the $m \times m$ matrix associated with least squares fitting on this grid by a polynomial of degree $n - 1$. Take b to be the function $\cos(4t)$ evaluated at the grid. Now, calculate and print (to sixteen-digit precision) the least square coefficient vector x by the following methods:

(a) Formation and solution of the normal equations, using MATLAB's `\`,

(d) QR factorization computed by MATLAB's `qr`

(e) `x = A\b` in MATLAB

(f) SVD, using MATLAB's `svd`

(g) The calculations above will produce four lists of twelve coefficients. In each list, shade with red pen the digits that appear to be wrong (affected by rounding error. Comment on what differences you observe. Do the normal equations exhibit instability? You do not have to explain your observations.

Solution

We use the methods described in this chapter to solve a least squares solution in the listed ways. Instead of MATLAB we use the analogous NumPy functions.

```
def exercise_11_3():
    np.set_printoptions(precision=16)
    m,n=50,12

    # choose 50 linearly npaced grid points on the interval [0,1]
    ls=np.linspace(0,1,m)

    # evaluate 11th degree polynomial at grid points
    A=np.array([[ls[i]**j for j in range(n)] for i in range(m)])

    # evaluate function at gridpoints
    b=np.transpose(np.cos(4*ls))

    # solve normal equation
    At=np.transpose(A)
    parta=np.linalg.solve(At@A,At@b)

    # QR then solve
    Q,R=np.linalg.qr(A,mode='reduced')
    partd=np.linalg.solve(R,np.transpose(Q)@b)[0]

    # solve least squares directly
    parte=np.linalg.lstsq(A,b)[0]

    # SVD then solve
    U,s,Vh=np.linalg.svd(A,full_matrices=False)
    w=np.linalg.lstsq(np.diag(s),np.transpose(U)@b)[0]
    partf=np.transpose(Vh)@w
```

```
print(np.transpose([parta,partd,parte,partf]))
```

This gives output:

(a) normal equations	(d) QR factorization	(e) direct solve	(f) SVD factorization
9.9999999818173035e-01	1.0000000009965979e+00	1.0000000009965979e+00	1.0000000009965979e+00
2.1259641665186701e-07	-4.2274272260556245e-07	-4.2274265177333348e-07	-4.2274272837872218e-07
-8.0000021885339176e+00	-7.9999812356914406e+00	-7.9999812356934301e+00	-7.9999812356912292e+00
-4.1462123573614304e-05	-3.1876317444057278e-04	-3.1876314993206645e-04	-3.1876317678825039e-04
1.0667499807681406e+01	1.0669430795550085e+01	1.0669430795381054e+01	1.0669430795563699e+01
-5.8214001430143755e-03	-1.3820286633722034e-02	-1.3820285916466446e-02	-1.3820286680574223e-02
-5.6680433964156123e+00	-5.6470756308378469e+00	-5.6470756327989022e+00	-5.6470756307355092e+00
-3.9601899719993561e-02	-7.5316018359055548e-02	-7.5316014841892986e-02	-7.5316018505332760e-02
1.6541609690516932e+00	1.6936069567962235e+00	1.6936069526864230e+00	1.6936069569330794e+00
3.3293694535084475e-02	6.0321134788960951e-03	6.0321164899116431e-03	6.0321133980698605e-03
-3.8495713040492091e-01	-3.7424170534789569e-01	-3.7424170660312983e-01	-3.7424170532064682e-01
8.9869175581991409e-02	8.8040576403572934e-02	8.8040576630669998e-02	8.8040576399611048e-02

(g) We shade with red entries which appear to be wrong, and with blue entries which differ from their analogous entries with other methods by more than a few decimal points.

Note that the normal equations differ quite significantly from the other methods, displaying the numerical instability of the algorithm. None of the methods agreed entirely, but the other methods generally agreed to at least 10 decimal places.

Exercise

Consider the following least squares approach for ranking sports teams. Suppose we have four college football teams, called simply T1, T2, T3, and T4. These four teams play each other with the following outcomes:

- T1 beats T2 by 4 points: 21 to 17.
- T3 beats T1 by 9 points: 27 to 18.
- T1 beats T4 by 6 points: 16 to 10.
- T3 beats T4 by 3 points: 10 to 7.
- T2 beats T4 by 7 points: 17 to 10.

To determine ranking points $r_1, \ldots, r_4$ for each team, we do a least squares fit to the overdetermined linear system:

$$\begin{aligned} r_1 - r_2 &= 4, \\ r_3 - r_1 &= 9, \\ r_1 - r_4 &= 6, \\ r_3 - r_4 &= 3, \\ r_2 - r_4 &= 7 \end{aligned}$$

This system does not have a unique least squares solution, however.

(a) Show that if $(r_1, \ldots, r_4)^T$ solves the least squares problem above then so does the vector $(r_1 + c, \ldots, r_4 + c)^T$ for any constant c.
To make the solution unique, we can fix the total number of ranking points, say, at 20. To do this, we add the following equation to those listed above:

$$r_1 + r_2 + r_3 + r_4 = 20.$$

(b) Explain why the least squares solution to the six equations listed will satisfy this last equation exactly.

(c) Use Matlab to determine the values r_1, r_2, r_3, r_4 that most closely satisfy these equations, and based on your results, rank the four teams. [This method of ranking sports teams is a simplification of one introduced by Ken Massey in 1997. It has evolved into a part of the famous BCS (Bowl Championship Series) model for ranking college football teams and is one factor in determining which teams play in bowl games.]

Solution

(a) We write this system as $Ar = b$ for,

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & -1 \end{bmatrix} \qquad r = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix} \qquad b = \begin{bmatrix} 4 \\ 9 \\ 6 \\ 3 \\ 7 \end{bmatrix}$$

Let c be a constant and define $s = (c, c, c, c)$. Observe $s \in \ker(A)$. Let $r = (r_1, ..., r_4)^T$ so that $(r_1 + c, ..., r_4 + c)^T = r + s$.

Then clearly,

$$\|b - Ar\| = \|b - Ar - As\| = \|b - A(r + s)\|$$

This proves that if r is a least squares solution to the system $Ar = b$, then $r + s$ is also a solution. □

(b) To make the solution unique we append the equation $r_1 + r_2 + r_3 + r_4 = 20$ so that,

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad r = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix} \qquad b = \begin{bmatrix} 4 \\ 9 \\ 6 \\ 3 \\ 7 \\ 20 \end{bmatrix}$$

Denote the old system with 5 equations by A_0. From above, if $(r_1, ..., r_4)$ is a solution then so is $(r_1 + c, ..., r_4 + c)$. By appropriate choice of c we can find a least squares solution such that $r_1 + c + ... + r_4 + c = r_1 + ... + r_4 + 4c$ has any value. Moreover, this choice of c is uniquely determined by the value of the sum.

Observe $\|b - Ar\|^2 = \|b - A_0 r\|^2 + (20 - (r_1 + r_2 + r_3 + r_4))^2$. The first term is minimized for a least squares solution to $A_0 r = b$. The second term, $(20 - (r_1 + ... + r_4))^2$, is non-negative. We have just argued that there is a least squares solution to $A_0 r = b$ such that $r_1 + ... + r_4 = 20$. Then the second term is zero and therefore minimal.

Any vector which is not a least squares solution will not minimize the first term, and any other least squares solution to $A_0 r = b$ will not minimize the second term. Finally, since the square function is strictly increasing, $\|b - Ar\|$ is minimized when $\|b - Ar\|^2$ is minimized. Therefore, the least squares solution to $Ar = b$ is the *unique* least squares solution to $A_0 r = b$ which satisfies $r_1 + ... + r_4 = 20$.

(c) We solve using NumPy,

```
def exercise_4_4():
    A=np.array(
            [[1,-1,0,0],
             [-1,0,1,0],
             [1,0,0,-1],
             [0,0,1,-1],
             [0,1,0,-1],
             [1,1,1,1]]
            )
    b=np.transpose(np.array([[4,9,6,3,7,20]]))
    At=np.transpose(A)
    x=np.linalg.solve(At@A,At@b)
    print(x)
```

This gives solution for r,

```
[[ 5.25 ]
 [ 4.625]
```

```
[ 9.125]
[ 1.   ]]
```

That is,

$$r_1 = 5.25 \qquad r_2 = 4.625 \qquad r_3 = 9.125 \qquad r_4 = 1$$

So the least squares fit with this method ranks the teams from best to worst as, $T3, T1, T2, T4$.

This corresponds at the very least roughly with our intuition, since team 3 never lost, teams 1 and 2 lost, but team 1 won more than team 2 (including a head to head win) and team 4 lost all their games).