Assignment 3

Linus Falk

October 30, 2022

1 Workout 2.3

Show that orthogonal matrices preserve the 2-norm and the Frobenius norm of matrices.

Let Q be a orthogonal matrix $Q \in \mathbb{R}^{mxn}$ and an arbitrary matrix $A \in \mathbb{R}^{nxm}$

$$\|QA\|_2^2 = (QAu)^T QA = A^T Q^T QA = A^T A = \|A\|_2^2 \tag{1}$$

We know that $\|A\|_F = \sum_{i=1}^m \sum_{j=i}^n a_{ij}^2 = \text{trace}(AA^T)$

$$\|QA\|_F^2 = \sum_{i=1}^m \sum_{j=i}^n q_{ij}^2 a_{ij}^2 = \text{trace}(QA(QA)^T)^2 = \text{trace}(QAA^T Q^T)^2 \tag{2}$$

The trace in invariant to cyclic permutations and so we got:

$$\text{trace}(QAA^T Q^T)^2 = \text{trace}(Q^T QA^T A)^2 = \text{trace}(A^T A)^2 = \|A\|_H^2 \tag{3}$$

2 Workout 2.8

Show that it requires $2n^2(m - \frac{n}{3})$ flops to compute R in the QR-factorization of $A \in \mathbb{R}^{m\times}, m \geq n$ using Householder transformations. This cost does not include the explicit construction of Q. Show that it is required $4(m^2n - mn^2 + \frac{n^3}{3}$

The Householder matrix multiplication in QR factorization:

$$\begin{gathered} HA = A - \frac{2}{u^T u} uu^T A = A - \beta u(u^T)A = A - \beta uw^T \\ \beta = \frac{2}{u^T u} \quad w = A^T u \end{gathered} \tag{4}$$

We get approximately mn flops for each of these calculations. We then proceed to calculate the sum when we reduce the size of H.

$$\begin{aligned} \sum_{k=1}^n 4(m-k)(n-k) = 4\sum_{i=1}^{n-1}(m-n+i)i = 4(m-n)\sum_{i=0}^{n-1} i + 4\sum_{i=0}^{n-1} i^2 = \\ 2(m-n)n(n-1) + 4\sum_{i=0}^{n-1} j^2 \approx 2(m-n)n(n-1) + 4\int_0^n x^2 \mathrm{dx} = \\ 2(m-n)n(n-1) + \frac{4}{3}n^3 = 2mn^2 - \frac{2}{3}n^3 \end{aligned} \tag{5}$$

3 Workout 3.3

Let $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma \geq 0$ be the singular values of $A \in \mathbb{R}^{m \times n}$ with $m \geq n$ and prove:

3.1 $\|A\|_2 = \sigma_1$

L_2 norm is defined as: max $\frac{\|Ax\|_2}{\|x\|_2}$

$$\begin{aligned} \|A\|_2^2 = max\frac{\|Ax\|_2}{\|x\|_2} = max\frac{x^TA^TAx}{x^Tx} = \lambda_1(A^TA) = \lambda_1((U\Sigma V^T)^TU\Sigma V^T) \\ \lambda_1(\Sigma^T\Sigma) = \sigma_1^2 \end{aligned} \tag{6}$$

The max is the largest vector i A

3.2 $\|A\|_F = \sqrt{\sigma_1^2 + \ldots + \sigma_n^2}$

$$\begin{aligned} \|A\|_H^2 = \text{trace}(A^TA) = \text{trace}((U\Sigma V^T)^TU\Sigma V^T) = \text{trace}(V\Sigma^TU^TU\Sigma V^T) \\ \text{trace}(V\Sigma^T\Sigma V^T) = \text{trace}(\Sigma^T\Sigma) = \sigma_1^2 + \sigma_2^2 + \ldots + \sigma_n^2 \end{aligned} \tag{7}$$

3.3 $\|A^{-1}\|_2 = \frac{1}{\sigma_n}$

Using that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$

$$\|A^{-1}\|_2^2 = max\frac{\|x\|_2}{\|Ax\|_2} = max\frac{x^Tx}{Ax^TA^TAx} = \frac{1}{\sigma_n^2} \tag{8}$$

3.4 cond$(A)_2 = \frac{\sigma_1}{\sigma_n}$

$$\text{Cond}_2(A) = \|A\|\|A^{-1}\| = [\text{from 1 and 3 we get}] = \frac{\sigma_1}{\sigma_n} \tag{9}$$

3.5 Rank(A) = number of nonzeros singular values

If we decompose A such that $A = U\Sigma V^T$ and U and V are orthogonal matrices. The multiplication of these invertible/orthogonal matrices with Σ doesn't change the rank of Σ. The rank of Σ is therefore the rank of A. The rank of Σ is equal the number of non-singular values.

4 Workout 3.5

Let A be an $m \times n$ matrix off full rank r = min(m,n). If C is another $m \times n$ matrix such that $\|C - A\|_2 < \sigma_r$, then show C also has full rank.

Using SVD let D be the nearest singular matrix to A:

$$A = \begin{bmatrix} \sigma_1 & 0 & \dots & \dots & 0 \\ 0 & \sigma_2 & \dots & \dots & 0 \\ \vdots & \dots & \ddots & \vdots & \vdots \\ 0 & \dots & \dots & \sigma_{r-1} & 0 \\ 0 & \dots & \dots & \dots & \sigma_r \\ 0 & \dots & \dots & \dots & 0 \end{bmatrix}, D = \begin{bmatrix} \sigma_1 & 0 & \dots & \dots & 0 \\ 0 & \sigma_2 & \dots & \dots & 0 \\ \vdots & \dots & \ddots & \vdots & \vdots \\ 0 & \dots & \dots & \sigma_{r-1} & 0 \\ 0 & \dots & \dots & \dots & 0 \\ 0 & \dots & \dots & \dots & 0 \end{bmatrix} \tag{10}$$

The L2 norm $\|A - D\|_2 = \|\Sigma_A - \Sigma_D\|_2 = \sigma_r$ if $\|A - C\|_2 < \sigma_r$ then the rank of C most be of the same A.

5 WO 2.9 programming Python

```
#Implement a function for QR factorization

def grot(a,b):
    if abs(a) > abs(b):
        t = b/a; c = 1/np.sqrt(1+t**2); s = c*t
    else:
        t = a/b; s = 1/np.sqrt(1+t**2); c = s*t

   return np.array([c, s])


def G(R, c, s, i):
    R[i][i] = s
    R[i][i - 1] = c
    R[i - 1][i] = -c
    R[i - 1][i - 1] = s
    return R


def qrfac(A, mode = "R"):
    m, n = np.shape(A)
    if mode == "R":
        R = A
        for j in range(1, n + 1, 1):
            for i in range(m, j, -1):
                Q = np.eye(m)
                [a, b] = grot(R[i - 1][j - 1],
                R[i - 2][j - 1])
                R = np.transpose(G(Q, a, b, i - 1)) @ R
        return np.triu(R)
```

```
    elif mode == "RQ":
        Qsave = np.eye(m)
        R = A
        for j in range(1, n + 1, 1):
            for i in range(m, j, -1):
                Q = np.eye(m)
                [a, b] = grot(R[i - 1][j - 1],
                R[i - 2][j - 1])
                R = np.transpose(G(Q, a, b, i - 1)) @ R
                Qsave = Qsave @ Q
        return np.triu(R), Qsave[0:m][0:m]

    else:
        print('Input mode types "R" or "QR" ')

A = np.array([[0.8147, 0.0975, 0.1576],
[0.9058, 0.2785, 0.9706],
[0.1270, 0.5469, 0.9572],
[0.9134, 0.9575, 0.4854],
[0.6324, 0.9649, 0.8003]])

B = np.array([[2, 3, 5],
              [1, 2, -1],
              [2, 5, 3],
              [1, -1, 0]])

C = np.array([[6, 5, 0],
              [5, 1, 4],
              [0, 4, 3]])


r, q = qrfac(A,"RQ")
print(q@r-A)

r, q = qrfac(B,"RQ")
print(q@r-B)

r, q = qrfac(C,"RQ")
print(q@r-C)

#OUTPUT

[[ 0.00000000e+00  0.00000000e+00  2.77555756e-17]
 [-2.22044605e-16  1.11022302e-16 -3.33066907e-16]
 [-2.77555756e-17 -2.22044605e-16 -5.55111512e-16]
 [-4.44089210e-16 -4.44089210e-16  0.00000000e+00]
 [-3.33066907e-16 -4.44089210e-16 -3.33066907e-16]]

[[-2.22044605e-16 -4.44089210e-16  0.00000000e+00]
```

```
 [ 0.00000000e+00  0.00000000e+00  3.33066907e-16]
 [ 0.00000000e+00 -8.88178420e-16  0.00000000e+00]
 [ 0.00000000e+00  1.55431223e-15  2.50130237e-16]]

[[-1.77635684e-15 -8.88178420e-16  2.84775512e-17]
 [-8.88178420e-16  2.22044605e-16 -8.88178420e-16]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00]]
```