Elements of Mathematics
Exercise Sheet 3

Submission due date: **16.11.2021, 10:15h**

THEORY

1 Inverse Matrix

Please prove the following statements.

1. An invertible matrix $A \in \mathbb{F}^{n\times n}$ has exactly one inverse matrix.
2. The inverse A^{-1} of an invertible matrix $A \in \mathbb{F}^{n\times n}$ is also invertible, with inverse $\left(A^{-1}\right)^{-1} = A$.
3. The product of two invertible matrices, say A and B, is invertible with inverse
$$(AB)^{-1} = B^{-1}A^{-1}.$$
4. A diagonal matrix
$$D = \operatorname{diag}(d_1, \dots, d_n) = \begin{pmatrix} d_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & d_n \end{pmatrix} \in \mathbb{F}^{n\times n}$$
is invertible if and only if $d_i \neq 0$ for all $i = 1, \dots, n$. What is its inverse?
Hint: It may be useful to split up the equivalence $\Leftrightarrow$ into $\Rightarrow$ and $\Leftarrow$ and to prove each of them separately.

(8 Points)

Solution:

1. Suppose $BA = I$ and $AC = I$, then
$$B = BI = B(AC) = (BA)C = IC = C.$$

   In the next subtasks we verify that the suggested inverse, say $\widetilde{A}$, satisfies the determining requirement $A\widetilde{A} = A\widetilde{A} = I$.
2. Let $B := A^{-1}$ and $\widetilde{B} := A$, then by definition of the inverse for A we find
$$B\widetilde{B} = A^{-1}A = I$$
and
$$\widetilde{B}B = AA^{-1} = I.$$
Thus $B^{-1} = (A^{-1})^{-1} = \widetilde{B} = A$.
3. Let $C := AB$ and $\widetilde{C} := B^{-1}A^{-1}$, then by exploiting the rules for matrix computations we obtain
$$C\widetilde{C} = (AB)B^{-1}A^{-1} = A(BB^{-1})A^{-1} = A \cdot I \cdot A^{-1} = AA^{-1} = I$$
and similarly
$$\widetilde{C}C = B^{-1}A^{-1}(AB) = B^{-1}(A^{-1}A)B = B^{-1} \cdot I \cdot B = B^{-1}B = I.$$
Thus $C^{-1} = (AB)^{-1} = \widetilde{C} = B^{-1}A^{-1}$.

4. We again split the proof for the equivalence ("$\Leftrightarrow$", "if and only if") into two statements ("$\Rightarrow$", "$\Leftarrow$").
"$\Leftarrow$": First, let $d_i \neq 0$ for all i (thus we can divide by d_i) and set
$$\widetilde{D} := \mathrm{diag}(d_1^{-1}, \dots, d_n^{-1}) = \begin{pmatrix} d_1^{-1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & d_n^{-1} \end{pmatrix} \in \mathbb{F}^{n\times n}.$$
Then by definition of the matrix product we can quickly verify that
$$D\widetilde{D} = I \quad \text{and} \quad \widetilde{D}D = I,$$
implying $D^{-1} = \widetilde{D}$ in this case.
"$\Rightarrow$": Proof by contradiction: Let $d_i = 0$ for at least one i. Then the i-th row (and column) of D solely contains 0 entries. Thus for any $\widetilde{D} \in \mathbb{F}^{n\times n}$ we have that the i-th row of $D\widetilde{D}$ is necessarily a zero row. Thus there cannot be a matrix $\widetilde{D}$ so that $D\widetilde{D} = I$. In particular, there cannot be a matrix $\widetilde{D}$ satisfying the requirements of the inverse matrix for D.

Alternatively:
The invertibility statement also follows from:
$$D \in \mathrm{GL}_n(\mathbb{F}) \quad \Leftrightarrow \quad \det(D) = \prod_{i=1}^{n} d_i \neq 0 \quad \Leftrightarrow \quad \forall 1 \leq i \leq n \colon d_i \neq 0$$
Then with the first part above we can derive the explicit expression for the inverse D^{-1}.

2 Projections and Least Squares

Let $a, b \in \mathbb{R}^n \setminus \{0\}$ be two nonzero vectors. Consider the 1-dimensional optimization task
$$\min_{c\in\mathbb{R}} \frac{1}{2}\|ca - b\|_2^2 =: f(c),$$
where $\|x\|_2 := \left(\sum_{i=1}^n x_i^2\right)^{\frac{1}{2}}$ denotes the Euclidean norm of a vector $x \in \mathbb{R}^n$. Determine the parameter $c \in \mathbb{R}$ which minimizes f. Compare your results to the projection of b onto a, i.e., $\mathrm{proj}_a(b) := \frac{a^\top b}{\|a\|_2}\frac{a}{\|a\|_2}$.

Hint: As in high-school, compute the derivative f' of f with respect to c and solve the equation $f'(c) = 0$. *(4 Points)*

Solution:

First we note that
$$f(c) = \frac{1}{2}\|ca - b\|_2^2 = \frac{1}{2}\left(c^2 a^\top a - 2ca^\top b - b^\top b\right)$$
Thus, for the derivative with respect to the scalar c, we find
$$f'(c) = ca^T a - a^T b.$$
Since $a \neq 0$ and therefore $a^\top a \neq 0$, we find
$$f'(\hat{c}) = 0 \quad \Leftrightarrow \quad \hat{c} = \frac{a^T b}{a^T a}.$$
By convexity of f we can conclude that $\hat{c}$ is a minimizer (you will learn this in the course "Numerical Optimization").

Remark: We will later identity the equation $ca^T a - a^T b = 0$ as the **normal equation**. The vector on the line $\mathrm{span}(a)$ closest to b in terms of the Euclidean norm is given by
$$\hat{c}a = \frac{a^T b}{a^T a}a = \frac{a^T b}{\|a\|_2}\frac{a}{\|a\|_2} = \mathrm{proj}_a(b).$$

3 Rule of Sarrus

Derive the *Rule of Sarrus* for the determinant of a (3×3)-matrix by using the Laplace formula from the lecture with $n = 3$. Then compute the determinant of

$$A = \begin{pmatrix} 1 & 4 & 3 \\ 0 & 2 & 2 \\ 0 & 1 & 1 \end{pmatrix}.$$

What does it tell us about the columns?

(6 Points)

Solution:

Recall:

$$\begin{aligned} \det(A) &= \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \det(A_{ij}) \quad (i \in \{1, \ldots, n\}, \text{ fixed}) \\ \det(a) &:= a \end{aligned}$$

Now consider $n = 3$ and let us fix $i = 1$. We indicate the submatrices A_{ij} by colors:

$$j = 1: \quad \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$$j = 2: \quad \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$$j = 3: \quad \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

By Laplace formula we obtain

$$\begin{aligned} \det(A) &= \underbrace{(-1)^{1+1}}_{=1} a_{11} \det(A_{11}) + \underbrace{(-1)^{1+2}}_{=-1} a_{12} \det(A_{12}) + \underbrace{(-1)^{1+3}}_{=1} a_{13} \det(A_{13}) \\ &= a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{31}a_{23}) + a_{12}(a_{21}a_{32} - a_{31}a_{22}) \\ &= a_{11}a_{22}a_{33} - a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33} + a_{12}a_{31}a_{23} + a_{13}a_{21}a_{32} - a_{13}a_{31}a_{22}. \end{aligned}$$

Note that we have exploited also the Laplace formula for 2×2 matrices. For the example matrix this yields $\det(A) = 2 - 2 = 0$, so that we can conclude that the columns are linearly dependent.

4 Compute Determinants

Compute the determinants of the following matrices.

$$A = \begin{pmatrix} 1 & \pi & 2 & 12 \\ 0 & \frac{1}{5} & \frac{1}{\sqrt{2}} & 17 \\ 0 & 0 & 5 & \frac{1}{3} \\ 0 & 0 & 0 & 4 \end{pmatrix}, \qquad B = \begin{pmatrix} 5 & 3 \\ 1 & -2 \end{pmatrix}, \qquad C = \begin{pmatrix} 0 & \frac{1}{2} & 2 \\ -\frac{1}{2} & 0 & 7 \\ -2 & -7 & 0 \end{pmatrix}.$$

(6 Points)

Solution:

1. $\det(A) \overset{[A \text{ is upper triangular}]}{=} 1 \cdot \frac{1}{5} \cdot 5 \cdot 4 = 4$

2. $\det(B) \;=\; 5\cdot(-2) - 3\cdot 1 \;=\; -13$

3.

$$\det(C) = \det(\begin{pmatrix} 0 & \frac{1}{2} & 2 \\ -\frac{1}{2} & 0 & 7 \\ -2 & -7 & 0 \end{pmatrix}) \overset{\text{[Sarrus' Rule]}}{=} 0+0+0+\frac{1}{2}+7+(-2)+2+\left(-\frac{1}{2}\right)+(-7)$$
$$-(-2+0+2)-(-7+7+0)-\left(0+\left(-\frac{1}{2}\right)+\frac{1}{2}\right) \;=\; 0$$

PROGRAMMING

---

5 Gram-Schmidt algorithm

The Gram-Schmidt algorithm is an algorithm that can be used to compute a reduced (sometimes also called "thin"or "economic") QR-decomposition of a matrix $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ and $\mathrm{rank}(A) = n$, i.e., of a matrix whose columns are linearly independent.
The basic idea is to successively built up an orthogonal system from a given set of linearly independent vectors; in our case the columns of $A = [a_1, \dots, a_n] \in \mathbb{R}^{m\times n}$. We choose the first column as starting point for the algorithm and set $\widetilde{q}_1 := a_1$. Of course, in order to generate an orthogonal matrix Q we have to rescale the vector and set $q_1 := \frac{\widetilde{q}_1}{\|\widetilde{q}_1\|}$. The successive vectors $\widetilde{q}_k$ are generated by subtracting all the "shares" $a_k^\top q_\ell$ (=$\mathrm{proj}_{q_\ell}(a_k)$) of the previous vectors q_ℓ from the column a_k, i.e.,

$$\widetilde{q}_k := a_k - \sum_{\ell=1}^{k-1} a_k^\top q_\ell \, q_\ell.$$

The following algorithm computes a reduced QR-decomposition of some matrix $A \in \mathbb{R}^{m\times n}$.

$r_{11} \leftarrow \|a_1\|$
$q_1 \leftarrow \frac{a_1}{r_{11}}$

for $k = 2, \dots, n$
  for $\ell = 1, \dots, k-1$
    $r_{\ell k} \leftarrow a_k^\top q_\ell$

  $\widetilde{q}_k \leftarrow a_k - \sum_{\ell=1}^{k-1} r_{\ell k}\, q_\ell$
  $r_{kk} \leftarrow \|\widetilde{q}_k\|$
  $q_k \leftarrow \frac{\widetilde{q}_k}{r_{kk}}$

Task:

- Implement the Gram-Schmidt algorithm as a function `QR(A)`, which takes a matrix A as input and returns the matrices Q and R.
- Run your algorithm on an example matrix (e.g., `numpy.random.rand(m,n)`) and test your result by computing $Q^\top Q$ and $QR - A$, where the first should yield the identity and the latter a zero-matrix.
- Find a SciPy routine to compute the QR decomposition (for the reduced QR you may need to set the parameters accordingly).

Hint: You can use `numpy.allclose()` to check whether two `numpy.ndarray`'s are equal up to a certain tolerance. *(10 Points)*

Solution:

```python
import numpy as np
import scipy.linalg as la

def QR(A):
    """
    Computes a (reduced) QR-decomposition of a (mxn)-matrix with m>=n
    via Gram-Schmidt Algorithm.

    Parameters
    ----------
    A : (mxn) matrix with m>=n

    Returns
    -------
    Q : (mxn) with orthonormal columns
    R : (nxn) upper triangular matrix
    """
    m, n = A.shape
    R = np.zeros((n, n))
    Q = np.zeros((m, n))

    R[0, 0] = np.linalg.norm(A[:, 0])
    Q[:, 0] = A[:, 0] / R[0, 0]

    for k in range(1, n):
        for l in range(0, k):
            R[l, k] = A[:, k] @ Q[:, l]
        q = A[:, k] - Q @ R[:, k]
        R[k, k] = np.linalg.norm(q)
        Q[:, k] = q / R[k, k]

    return Q, R


if __name__ == "__main__":
    # Example
    m, n = 4, 2
    A = np.random.rand(m, n)
    print("A = \n", A)
#    A[:, -3] = A[:, 0]

    Q, R = QR(A)
    Q2, R2 = la.qr(A)#, mode='economic')  # Compare to SciPy
    print("Ours:\n-------\nQ = \n", np.round(Q, 2), "\nR = \n", np.round(R, 2))
    print("Sc:\n-------\nQ = \n", np.round(Q2, 2), "\nR = \n", np.round(R2, 2))
    print("\nTest 1: Q^TQ = I is", np.allclose(Q.transpose()@Q, np.eye(n)))
    print("\nTest 2: QR = A is", np.allclose(Q@R, A))
```

Total Number of Points = 34 (T:24, P:10)