Elements of Mathematics

Exercise Sheet 5

Submission due date: **30.11.2021, 10:15h**

THEORY

1 Compute eigenvalues

Compute the eigenvalues of the following matrices.

1.
$$A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

2.
$$B = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -9 \\ 0 & 1 & 6 \end{bmatrix}$$

3.
$$C = \begin{bmatrix} \pi & 3 & i^2 & 6 \\ 0 & 4 & 1 & e^3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

(9 Points)

Solution:

1. By applying the determinant rule for 2×2 matrices we find
$$0 = \det(A - \lambda I) = \lambda^2 + 1 \quad \Leftrightarrow \quad \lambda \in \{-i, +i\}.$$

2. By applying the Sarrus rule for 3×3 matrices we find
$$0 \overset{!}{=} \det(A - \lambda I) = \det \begin{pmatrix} -\lambda & 0 & 0 \\ 1 & -\lambda & -9 \\ 0 & 1 & 6-\lambda \end{pmatrix} \begin{matrix} -\lambda & 0 \\ 1 & -\lambda \\ 0 & 1 \end{matrix} \overset{[\text{Sarrus}]}{=} \lambda^2(6-\lambda) - 9\lambda$$
$$\Leftrightarrow \quad 0 = \lambda(\lambda(6-\lambda) - 9) = \lambda(-\lambda^2 + 6\lambda - 9) = -\lambda(\lambda - 3)^2$$
$$\Leftrightarrow \quad \lambda = 0 \text{ or } \lambda = 3 \ (\sigma(A) = \{0, 3\})$$

3. By applying the determinant rule for triangular matrices we find
$$\det(A - \lambda I) = (\pi - \lambda)(4 - \lambda)(1 - \lambda)\left(\frac{1}{2} - \lambda\right)$$
$$\Rightarrow \quad \sigma(A) = \{\pi, 4, 1, \frac{1}{2}\}$$
(Short: A triangular $(\Rightarrow \ \sigma(A) = \ \text{diag}(A))$)

2 Eigendecomposition

Let

$$A := \begin{pmatrix} 2 & 3 \\ 3 & 2 \end{pmatrix} \in \mathbb{R}^{2\times 2}.$$

Why does this matrix possess an eigendecomposition $A = Q\Lambda Q^T$? Compute the matrices Λ and Q, by following this recipe:

1. Determine its eigenvalues λ_1 and λ_2 to find Λ by solving $\chi_A(\lambda) = \det(A - \lambda I) = 0$.
2. Determine the corresponding eigenvectors v_1 and v_2 by solving $(A - \lambda_i I)v = 0$.
3. Normalize the eigenvectors to find Q by setting $\tilde{v}_i := \frac{v_i}{\|v_i\|_2}$ and $Q := [\tilde{v}_1, \tilde{v}_2]$. Test if Q^TQ equals I_2.
4. Test if $Q\Lambda Q^T$ equals A.

(8 Points)

Solution:

First note that A is <u>symmetric</u> $(A = A^T)$. So Theorem on eigendecomposition implies the existence of an orthogonal matrix Q and a diagonal matrix Λ, with $A = Q\Lambda Q^T$, which we will now determine.

1. Eigenvalues:
   $0 = \det(A - \lambda I) = \det\left(\begin{pmatrix} 2-\lambda & 3 \\ 3 & 2-\lambda \end{pmatrix}\right) = (2-\lambda)^2 - 9$
   $\Leftrightarrow \quad (2-\lambda)^2 = 9 \quad \Leftrightarrow \quad 2 - \lambda = \pm 3 \quad \Leftrightarrow \quad \lambda = 2 \pm 3 \quad (\lambda_1 := 5, \lambda_2 := -1)$
   Set
   $$\Lambda := \operatorname{diag}(\lambda_1, \lambda_2) = \begin{pmatrix} 5 & 0 \\ 0 & -1 \end{pmatrix}.$$

2. Eigenvectors:

   1) Determine an eigenvector corresponding to $\lambda_1 = r$.
   $$\begin{aligned}(A - \lambda_1 I)v^1 = 0 &\Leftrightarrow \begin{pmatrix} -3 & 3 \\ 3 & -3 \end{pmatrix}\begin{pmatrix} v_1^1 \\ v_2^1 \end{pmatrix} = 0 \\ &\Leftrightarrow -3v_1^1 + 3v_2^1 = 0 \\ &\Leftrightarrow v_1^1 = v_2^1\end{aligned}$$
   Choose, e.g., $v^1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

   2) Determine an eigenvector corresponding to $\lambda_2 = r$.
   $$\begin{aligned}(A - \lambda_2 I)v^2 = 0 &\Leftrightarrow \begin{pmatrix} 3 & 3 \\ 3 & 3 \end{pmatrix}\begin{pmatrix} v_1^2 \\ v_2^2 \end{pmatrix} = 0 \\ &\Leftrightarrow 3v_1^2 + 3v_2^2 = 0 \\ &\Leftrightarrow v_1^2 = -v_2^2\end{aligned}$$
   Choose, e.g., $v^2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$.

3. Normalize eigenvectors to define Q:
   $\tilde{v}_1 := \frac{v_1}{\|v_1\|} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \tilde{v}_2 := \frac{v_2}{\|v_2\|} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}$
   Set $Q := [\tilde{v}_1, \tilde{v}_2] = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.
   We find that A is orthogonal, more precisely,
   $$Q^TQ = \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} = I.$$

4. Test:

$$Q\Lambda Q^T = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\underbrace{\begin{pmatrix}5 & 0\\ 0 & -1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}}_{=\frac{1}{\sqrt{2}}\begin{pmatrix}5 & 5\\ -1 & 1\end{pmatrix}}$$

$$= \frac{1}{2}\underbrace{\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}5 & 5\\ -1 & 1\end{pmatrix}}_{=\begin{pmatrix}4 & 6\\ 6 & 4\end{pmatrix}}$$

$$= A \quad (\checkmark)$$

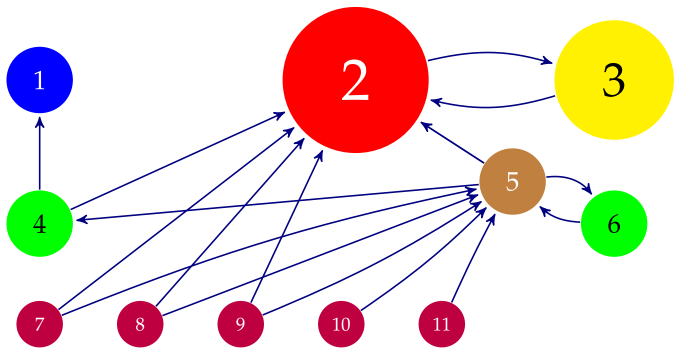
3 A Motivation for Eigenvectors: The *PageRank*

Aim: To rank results of web search enginge (such as Google) according to the *"importance"* of the web pages.

1998: For this purpose, Larry Page and Sergei Brin develop the PageRank algorithm as the basis of the Google empire.

Assumption: *"important"* means more links from other (important) web pages.

Idea: Let us think of the web as a directed graph, i.e., web pages are nodes and links from one page to another, i.e, from one node to another, are modeled as directed edges. For example a web structure consisting of 11 web pages could look as follows:



We now randomly place a random surfer according to the initial probability distribution $x^0 = (x_1^0, \dots, x_n^0)$ on this graph. Here, n (in the example above $n = 11$) denotes the number of web pages and x_i^0 denotes the probability that the random surfer *starts* at web page i. Further let $e := (1, \dots, 1)^T$, then the fact that x^0 is a probability distribution (i.e., probabilities sum up to 1) translates into $e^T x^0 = x_1^0 + \dots + x_n^0 = 1$. Now we make the assumption that the random surfer moves...

(1) ...with probability $\alpha \in (0, 1)$ according to the link structure (with equal preferences to outgoing links)

(2) ...with probability $(1 - \alpha)$ he can teleport to a random page (with equal probability) to prevent stranding in deadlocks

$\rightarrow$ Pages, where the random surfer is more likely to appear in the long run based on the web's structure are considered more important.

These two movements can be described by multiplying the current probability distribution with the two following matrices:

$$P_1 = \left(\begin{array}{cccccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 1 & 1 & & & 1/2 & & & & & & & \\ 2 & & & 1 & 1/2 & 1/3 & & 1/2 & 1/2 & 1/2 & & \\ 3 & & 1 & & & & & & & & & \\ 4 & & & & & 1/3 & & & & & & \\ 5 & & & & & & 1 & 1/2 & 1/2 & 1/2 & 1 & 1 \\ 6 & & & & & 1/3 & & & & & & \\ 7 & & & & & & & & & & & \\ 8 & & & & & & & & & & & \\ 9 & & & & & & & & & & & \\ 10 & & & & & & & & & & & \\ 11 & & & & & & & & & & & \end{array}\right), \quad P_2 := \frac{1}{n} e e^T = \left(\frac{1}{n}\right)_{ij}$$

More precisely,

(1) **Link structure:** P_1 is the probability matrix (column stochastic) defined by
$P_1^{ij} :=$ Probability that random surfer moves from page j to page i defined by the link structure

(2) **Jumps:** P_2 is the probability matrix (column stochastic) defined by
$P_2^{ij} := \frac{1}{n} =$ Probability that random surfer jumps from page j to page i

The movement of the random surfer is then completely defined by the probability matrix

$$P = \alpha P_1 + (1-\alpha) P_2.$$

This matrix is also known as the ***Google Matrix***. For the next time instances we therefore obtain

$$\begin{aligned} x^1 &= \alpha P_1 x^0 + (1-\alpha) P_2 x^0 = P x^0 \\ x^2 &= \alpha P_1 x^1 + (1-\alpha) P_2 x^1 = P x^1 \\ x^{k+1} &= \alpha P_1 x^k + (1-\alpha) P_2 x^k = P x^k = P^{k+1} x^0 \\ x^* &= \lim_{k\to\infty} x^k =: \textit{\textbf{PageRank}} \end{aligned}$$

Observations:

- One can easily show that P_1, P_2 and thus P are column stochastic (i.e., $e^T P = e^T$)
- Consequently, since x^0 is a probability distribution (i.e., $e^T x^0 = 1$), also $e^T x^k = 1$ for all k and $e^T x^* = 1$

Question:
Does this sequence $\{x^k\}_{k\in\mathbb{N}}$ of vectors converge (to a steady state)? More precisely, is there a $x^* = \lim_{k\to\infty} x^k = \lim_{k\to\infty} P^k x^0$, so that

$$P x^* = 1 x^*. \tag{1}$$

$\rightarrow$ With other words, is there an ***eigenvector*** x^* to the ***eigenvalue*** 1 of the matrix P?

$\rightarrow$ ***Eigenvalue algorithms*** are developed to solve such problems. One of them is the ***Power method***, which, applied to the eigenvalue problem above, produces precisely the sequence

$$x^k = P^k x^0.$$

Intuitively, in the limit most of the "mass" would be located at web pages that have many incoming links and would therefore be ranked as being more important. In fact, the i-th component of x^* is called the *PageRank* of the web page i.

Solution:

4 Eigenvalue Algorithm 1: The Power Iteration and the *PageRank*

The **power method** or **power iteration** is an eigenvalue algorithm, which, given a matrix A, produces a sequence of numbers converging to the largest (in absolute value) eigenvalue of A and a sequence of vectors converging to the corresponding eigenvector. More precisely, the iteration rule is given as follows:

$$x^{k+1} := \frac{Ax^k}{\|Ax^k\|_p}, \quad \text{and} \quad \mu^k := \frac{(x^k, Ax^k)_2}{(x^k, x^k)_2}. \tag{2}$$

Let $\lambda_1 \in \sigma(A)$ be the largest eigenvalue (in absolute value). Then (under certain conditions) we obtain

$$\mu_k \xrightarrow{k\to\infty} \lambda_1 \quad \text{and} \quad x_k \xrightarrow{k\to\infty} v_1 \quad \text{with} \quad Av_1 = \lambda_1 v_1.$$

Remark: For the normalization step $\frac{1}{\|Ax^k\|_p}$ in the equation above one can choose any p-norm $\|x\| := (\sum_{i=1}^n |x_i|^p)^{\frac{1}{p}}$ with $p \geq 1$; typical choices are $p \in \{1, 2, \infty\}$. Note that the choice $p = 2$ corresponds to the Euclidean norm introduced in the lecture.

Task:

1. Implement a function `power_iteration(A,m,p=1)` which takes as input a matrix $A \in \mathbb{R}^{n\times n}$, a maximum iteration number $m \in \mathbb{N}$ and an *optional* parameter p which determines the order of the p-norm and is set to $p = 1$ by default. This function shall then return the m-th iterates x_m and μ_m of the power iteration (2).

   Hints:

   - You can use a random distribution x_0 as initial guess by calling for example the numpy function

     ```
     x = numpy.random.dirichlet(np.ones(n),size=1).reshape(n)
     ```

     or simply choose

     ```
     x = 1./n * np.ones(n).
     ```

   - For the normalization step use the numpy function

     ```
     numpy.linalg.norm(x, ord=p),
     ```

     which allows the choices $p \in \{1, 2, \infty\}$ (among others).

2. Determine the **PageRank** of the web structure given above. Therefore apply your function `power_iteration(A,m,p=1)` to the PageRank matrix

   $$A := P = \alpha P_1 + (1-\alpha)P_2,$$

   where

   $$P_1 = \left(\begin{array}{cccccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 1 & 1 & & & 1/2 & & & & & & & \\ 2 & & & 1 & 1/2 & 1/3 & & 1/2 & 1/2 & 1/2 & & \\ 3 & & 1 & & & & & & & & & \\ 4 & & & & & 1/3 & & & & & & \\ 5 & & & & & & 1 & 1/2 & 1/2 & 1/2 & 1 & 1 \\ 6 & & & & & 1/3 & & & & & & \\ 7 & & & & & & & & & & & \\ 8 & & & & & & & & & & & \\ 9 & & & & & & & & & & & \\ 10 & & & & & & & & & & & \\ 11 & & & & & & & & & & & \end{array}\right), \quad P_2 := \frac{1}{n}ee^T = \left(\frac{1}{n}\right)_{ij}.$$

   Play around with the damping factor α. What do you observe?

   Hint: For implementing P_1 and P_2, and thus P, you can use the code snippet

```
import numpy as np
def P(alpha):
    P_1=np.array([[1,0,0,1.0/2,0,0,0,0,0,0,0],
                  [0,0,1.0,1.0/2,1.0/3,0,1.0/2,1.0/2,1.0/2,0,0],
                  [0,1.0,0,0,0,0,0,0,0,0,0],
                  [0,0,0,0,1.0/3,0,0,0,0,0,0],
                  [0,0,0,0,0,1.0,1.0/2,1.0/2,1.0/2,1.0,1.0],
                  [0,0,0,0,1.0/3,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0,0,0,0]])

    P_2 = (1.0 / 11.0) * np.ones((11,11))

    return alpha * P_1 + (1-alpha) * P_2
```

(10 Points)

Solution:

```
import numpy as np


def power_iteration(A, m, p=1):
    """
    Solves eigenvalue problem via Power Method
    Expects the largerst eigenvalue of A to be scritly larger

    Parameters
    ----------
    A : (n,n) ndarray
        matrix
    m : int
        number of iterations
    p : int or numpy.inf, optional
        specifying the order of the p-Norm used for normalization

    Returns
    -------
    x : (n,1) ndarray
        normalized (with p-Norm) eigenvector for largest eigenvalue
    mu : float
         largest eigenvalue
    """
    n = A.shape[1]
    # x = np.random.dirichlet(np.ones(n), size=1).reshape(n)
    x = 1./n * np.ones(n)
    for k in range(m):
        z = A.dot(x)
        x = z / np.linalg.norm(z, ord=p)
        mu = x.dot(z) / (x.dot(x))
    return x, mu


def P(alpha):
    P_1 = np.array([[1, 0, 0, 1.0/2, 0, 0, 0, 0, 0, 0, 0],
                    [0,0,1.0,1.0/2,1.0/3,0,1.0/2,1.0/2,1.0/2,0,0],
                    [0,1.0,0,0,0,0,0,0,0,0,0],
                    [0,0,0,0,1.0/3,0,0,0,0,0,0],
                  [0,0,0,0,0,1.0,1.0/2,1.0/2,1.0/2,1.0,1.0],
                  [0,0,0,0,1.0/3,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0,0,0,0],
                  [0,0,0,0,0,0,0,0,0,0,0]])

    P_2 = (1.0 / 11.0) * np.ones((11, 11))

    return alpha * P_1 + (1-alpha) * P_2

if __name__ == "__main__":
    m = 20
    k = 10
    # run with different damping factors
    for alpha in np.linspace(0, 1, k, endpoint=False):
        print("\n-----------------\nalpha =",
              np.round(alpha, 4),
              "\n-----------------")
        eigvec, mu_max = power_iteration(P(alpha), m, p=1)
        print("Maximal eigenvalue = ", np.around(mu_max, 10))
        print("\nEigenvector = \n", np.around(eigvec, 5))

        lbdmax_np = np.max(np.linalg.eigvals(P(alpha)))
        print("\nMaximal eigenvalue from numpy = ",
              np.around(np.real(lbdmax_np), 10))
```

Total Number of Points = 27 (T:17, P:10)