week07-03–eigen

March 3, 2024

1 George McNinch Math 87 - Spring 2024

2 Week 7

3 Power-iteration & eigenvalues

4 Eigenvalues & power-iteration

Let $A \in \mathbb{R}^{n\times n}$ be a square matrix. Our goal is to understand the *eventual behavior* of powers of A; i.e. the matrices A^m for $m \to \infty$.

4.1 Example: Diagonal matrices

Let's look at a simple example. Consider the following matrix:

$$A = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 \\ 0 & 0 & 0 & \lambda_4 \end{bmatrix}$$

In this case, it is easy to understand the powers of A; indeed, we have

$$A^m = \begin{bmatrix} \lambda_1^m & 0 & 0 & 0 \\ 0 & \lambda_2^m & 0 & 0 \\ 0 & 0 & \lambda_3^m & 0 \\ 0 & 0 & 0 & \lambda_4^m \end{bmatrix}$$

4.2 example, continued

Observe that if $|\lambda| < 1$, then $\lambda^m \to 0$ as $m \to \infty$. So e.g. if $|\lambda_i| < 1$ for $i = 1, 2, 3, 4$, then

$$A^m \to \mathbf{0} \quad \text{as} \quad m \to \infty.$$

If $\lambda_1 = 1$ and $|\lambda_i| < 1$ for $i = 2, 3, 4$, then

$$A^m \to \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

On the other hand, if $|\lambda_i| > 1$ for some i, then $\lim_{m\to\infty} A^m$ doesn't exist, because $\lambda_i^m \to \pm\infty$ as $m \to \infty$.

Of course, "most" matrices aren't diagonal, or at least not *literally*.

4.3 Eigenvalues and eigenvectors

Recall that a number $\lambda \in \mathbb{R}$ is an *eigenvalue* of A if there is a non-zero vector $\mathbf{v} \in \mathbb{R}^n$ for which

$$A\mathbf{v} = \lambda\mathbf{v};$$

$\mathbf{v}$ is then called an *eigenvector*.

If A is diagonal – e.g. if

$$A = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 \\ 0 & 0 & 0 & \lambda_4 \end{bmatrix} = \mathrm{diag}(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$$

– it is easy to see that each standard basis vector $\mathbf{e}_i$ is an eigenvector, with corresponding eigenvalue λ_i (the (i,i)-the entry of A).

4.4 Eigenvectors

Now suppose that A is an $n \times n$ matrix, that $\mathbf{v}_1, \dots, \mathbf{v}_n$ are eigenvectors for A, and that $\lambda_1, \dots, \lambda_n$ are the corresponding eigenvalues. Write

$$P = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{bmatrix}$$

for the matrix whose columns are the $\mathbf{v}_i$.

Theorem 0: P is invertible if and only if the vectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ are linearly independent.

Theorem 1: If the eigenvalues $\lambda_1, \dots, \lambda_n$ are *distinct*, then the vectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ are linearly independent, and in particular, the matrix P is invertible.

4.5 Diagonalizable matrices

Theorem 2: If the eigenvectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ are linearly independent – equivalently, if the matrix P is invertible – then

$$P^{-1}AP = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_n \end{bmatrix} = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$$

i.e. $P^{-1}AP$ is the diagonal matrix $n \times n$ matrix whose diagonal entries are $\lambda_1, \dots, \lambda_n$.

Because of **Theorem 2**, one says that the $n \times n$ matrix A is *diagonalizable* if it has n linearly independent eigenvectors.

Thus if we are willing to replace our matrix by the *conjugate* matrix $P^{-1}AP$, then for A diagonalizable, for some purposes "we may as well suppose that A is diagonal" (though of course that statement is imprecise!).

4.6 Finding eigenvalues

One might wonder "how do I find eigenvalues"? The answer is: the eigenvalues of A are the roots of the *characteristic polynomial* $p_A(t)$ of A, where:

$$p_A(t) = \det(A - t \cdot \mathbf{I_n}).$$

Proposition: The characteristic polynomial $p_A(t)$ of the $n \times n$ matrix A has degree n, and thus A has no more than n distinct eigenvalues.

Remark: The eigenvalues of A are complex numbers which in general may fail to be real numbers, even when A has only real-number coefficients.

4.7 Tools for finding eigenvalues

`python` and `numpy` provides tools for finding eigenvalues. Let's look at the following example:

Example: Consider the matrix

$$A = \left(\frac{1}{10}\right) \cdot \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 3 & 3 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}.$$

[1]:
```
import numpy as np
import numpy.linalg as npl


float_formatter = "{:.2f}".format
np.set_printoptions(formatter={'float_kind':float_formatter})


A = (1/10)*np.array([[1,1,0,0],[0,2,2,0],[0,3,3,1],[0,0,1,2]])

A
```

[1]:
```
array([[0.10, 0.10, 0.00, 0.00],
       [0.00, 0.20, 0.20, 0.00],
       [0.00, 0.30, 0.30, 0.10],
       [0.00, 0.00, 0.10, 0.20]])
```

Let's find the eigenvectors/values of this 4×4 matrix A; we'll use the function `eig` found in the python module `numpy.linalg`:

[2]:
```
(e_vals,e_vecs) = npl.eig(A)
[e_vals,e_vecs]
```

[2]:
```
[array([0.10, 0.52, -0.02, 0.20]),
 array([[1.00, -0.12, -0.47, -0.30],
        [0.00, -0.51, 0.56, -0.30],
        [0.00, -0.81, -0.62, -0.00],
        [0.00, -0.25, 0.28, 0.90]])]
```

The function `eig` returns a "list of np arrays". This first array contains the eigenvalues, and the second contains the a matrix whose *columns* are the eigenvectors.

We've assigned the first component of the list to the variable `e_vals` and the second to `e_vecs`.

To get the individual eigenvectors, we need to slice the array `e_vecs`.

For example, to get the 0-th ("first"!) eigenvector, we can use

`e_vecs[:,0]`

Here, the argument `:` indicates that the full range should be used in the first index dimension, and the argument `0` indicates the the second index dimension of the slice is `0`. Thus `numpy` returns the array whose entries are `e_vecs[0,0]`, `e_vecs[1,0]`, `e_vecs[2,0]`, `e_vecs[3,0]`.

Let's confirm that this is really an eigenvector with the indicated eigenvalue:

[3]:
```
v = e_vecs[:,0]
[v,A @ v,e_vals[0]*v, (A @ v - e_vals[0] * v < 1e-7).all()]
```

[3]:
```
[array([1.00, 0.00, 0.00, 0.00]),
 array([0.10, 0.00, 0.00, 0.00]),
 array([0.10, 0.00, 0.00, 0.00]),
 True]
```

Let's check *all* of the eigenvalues:

[4]:
```
def check(A):
    e_vals,e_vecs = npl.eig(A)

    def check_i(i):
        lam = e_vals[i]
        v= e_vecs[:,i]
        return "\n".join([f"lambda    = {lam}",
                          f"v         = {v}",
                          f"Av        = {A @ v}",
                          f"lambda*v  = {lam * v}",
                          f"match?:    {(np.abs(A @ v - lam * v) < 1e-7).
 ↪all()}",
                          ""])
```

```
    return "\n".join([check_i(i) for i in range(len(e_vals))])
```

[5]:
```
print(check(A))
```

```
lambda   = 0.1
v        = [1.00 0.00 0.00 0.00]
Av       = [0.10 0.00 0.00 0.00]
lambda*v = [0.10 0.00 0.00 0.00]
match?:    True

lambda   = 0.5192582403567257
v        = [-0.12 -0.51 -0.81 -0.25]
Av       = [-0.06 -0.26 -0.42 -0.13]
lambda*v = [-0.06 -0.26 -0.42 -0.13]
match?:    True

lambda   = -0.019258240356725218
v        = [-0.47 0.56 -0.62 0.28]
Av       = [0.01 -0.01 0.01 -0.01]
lambda*v = [0.01 -0.01 0.01 -0.01]
match?:    True

lambda   = 0.19999999999999998
v        = [-0.30 -0.30 -0.00 0.90]
Av       = [-0.06 -0.06 -0.00 0.18]
lambda*v = [-0.06 -0.06 -0.00 0.18]
match?:    True
```

Let's observe that A has 4 distinct eigenvalues, and is thus diagonalizable. Moreover, every eigenvalue λ of A satisfies $|\lambda| < 1$. Thus, we conclude that $A^m \to \mathbf{0}$ as $m \to \infty$.

And indeed, we confirm that:

[6]:
```
res=[(npl.matrix_power(A,j) - np.zeros((4,4)) < 1e-7*np.ones((4,4))).all() for␣
 ↪j in range(50)]

j = res.index(True)   ## find the first instance in the list of results

print(f"A^{j} == 0")
```

```
A^24 == 0
```

4.8 Eigenvalues and power iteration.

Theorem 3: Let A be a diagonalizable $n \times n$, with n linearly independent eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$. As before, write

$$P = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{bmatrix}.$$

a) Suppose $|\lambda_i| < 1$ for all i. Then $A^m \to \mathbf{0}$ as $m \to \infty$.

b) Suppose that $\lambda_1 = 1$, and $|\lambda_i| < 1$ for $2 \leq i \leq n$. Any vector $\mathbf{v} \in \mathbb{R}^n$ may be written

$$\mathbf{v} = \sum_{i=1}^{n} c_i \mathbf{v}_i.$$

If $c_1 \neq 0$, then

$$A^m \mathbf{v} = c_1 \mathbf{v}_1 \quad \text{as} \quad m \to \infty.$$

If $c_1 = 0$ then

$$A^m \mathbf{v} = \mathbf{0} \quad \text{as} \quad m \to \infty.$$

4.9 Proof:

For **a)**, note that $P^{-1}AP = \text{diag}(\lambda_1, \dots, \lambda_n)$. Which shows that

$$(P^{-1}AP)^m = \text{diag}(\lambda_1, \dots, \lambda_n)^m = \text{diag}(\lambda_1^m, \dots, \lambda_n^m) \to \mathbf{0} \quad \text{as } m \to \infty.$$

Let's now notice that

$$(P^{-1}AP)^2 = (P^{-1}AP)(P^{-1}AP) = P^{-1}AAP = P^{-1}A^2P$$

and more generally

$$(P^{-1}AP)^m = P^{-1}A^mP \quad \text{for } m \geq 0.$$

We now see that

$$P^{-1}A^mP \to \mathbf{0} \quad \text{as } m \to \infty$$

so that

$$A^m \to P \cdot \mathbf{0} \cdot P^{-1} = \mathbf{0} \quad \text{as } m \to \infty$$

4.10 Proof of b):

Recall that $\mathbf{v} = \sum_{i=1}^{n} c_i \mathbf{v}_i$.

For $i > 1$, **a)** shows that

$$A^m \mathbf{v}_i \to \mathbf{0} \quad \text{as } m \to \infty.$$

while

$$A^m \mathbf{v}_1 = \mathbf{v}_1 \quad \text{for all } m.$$

The preceding discussion now shows that

$$A^m \mathbf{v} = \sum_{i=1}^{n} c_i A^m \mathbf{v}_i \mapsto c_1 \mathbf{v_1}$$

and **b)** follows at once.

4.11 Corollary

Suppose that A is diagonalizable with eigenvalues $\lambda_1, \ldots, \lambda_n$, that $\lambda_1 = 1$, and that $|\lambda_i| < 1$ for $i = 2, \ldots, n$. Let $\mathbf{v_1}$ be a 1-eigenvector for A.

Then

$$A^m \to B \quad \text{as } m \to \infty$$

for a matrix B with the property that each column of B is either $\mathbf{0}$ or some multiple of $\mathbf{v_1}$.

Indeed: the ith column of B can be found by computing

$$(\heartsuit) \qquad \lim_{m\to\infty} A^m \mathbf{e}_i$$

where $\mathbf{e}_i$ is the ith standard basis vector.

We've seen above that $(\heartsuit)$ is either 0 or a multiple of $\mathbf{v}$, depending on whether or not the coefficient c_1 in the expression

$$\mathbf{e}_i = \sum_{j=1}^{n} c_j \mathbf{v}_j$$

is zero.

4.12 Examples revisited: population growth & aging

Recall from last week our finite-state machine describing population & aging.

We considered a population of organisms described by:

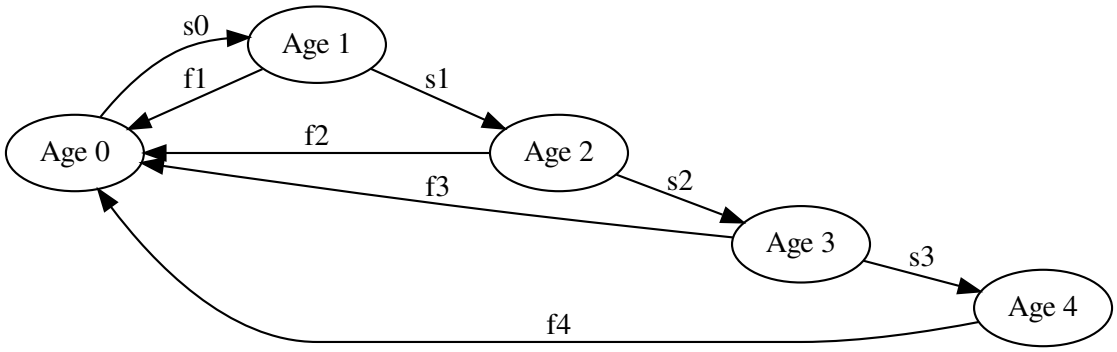
[30]:
```
from graphviz import Digraph
pop = Digraph("pop")
pop.attr(rankdir='LR')

p = list(range(5))
with pop.subgraph() as c:
#     c.attr(rank='same')
    for i in p:
        c.node(f"Age {i}")

for i in p:
    if i+1 in p:
        pop.edge(f"Age {i}",f"Age {i+1}",f"s{i}")
    if i != 0:
        pop.edge(f"Age {i}","Age 0",f"f{i}")

pop
```

[30]:



We suppose that $s_7 = 0$, so that the life-span of the organism in question is ≤ 8 time units.

If the population at time t is described by $\mathbf{p}^{(t)} = \begin{bmatrix} p_0 & p_1 & \cdots & p_7 \end{bmatrix}^T$ then the population at time $t+1$ is given by

$$\mathbf{p}^{(t+1)} = \begin{bmatrix} \sum_{i=0}^{7} f_i p_i & s_0 p_0 & \cdots & s_6 p_6 \end{bmatrix}^T = A\mathbf{p}^{(t)}$$

where

$$A = \begin{bmatrix} f_0 & f_1 & f_2 & \cdots & f_6 & f_7 \\ s_0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & s_1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & s_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & s_6 & 0 \end{bmatrix}.$$

4.13 parameters

Previously, we considered this model for two different sets of parameters:

```
fA = [.30,.50,.35,.25,.25,.15,.15,.5]
sA = [.30,.60,.55,.50,.30,.15,.05,0]
```

and

```
fB = [.50,.70,.55,.35,.35,.15,.15,.5]
sB = [.40,.70,.55,.50,.35,.15,.05,0]
```

[76]:
```
import numpy as np


float_formatter = "{:.2f}".format
np.set_printoptions(formatter={'float_kind':float_formatter})

def bv(ind,list):
    return np.array([1.0 if i == list.index(ind) else 0.0 for i in␣
 ↪range(len(list))])

## note
```

```
## bv("b",["a","b","c"])
## >> np.array([0,1,0])

ones = np.ones(8)

def A(f=[],s=[]):
    return np.array([np.array(f)]
                    +
                    [x*bv(x,s) for x in s[:-1]])

def display(f,s):
    e_val,e_vec = npl.eig(A(f,s))
    s = [f"Eigenvalues lambda of A when",
         f" f = {f}",
         f" s = {s}",
        "are as follows:"]
    return "\n".join(s+[f" |lambda| = {np.abs(e):.5f}" for e in e_val])
```

[72]:
```
fA = [.30,.50,.35,.25,.25,.15,.15,.5]
sA = [.30,.60,.55,.50,.30,.15,.05,0]
print(display(fA,sA))
```

```
Eigenvalues lambda of A when
 f = [0.3, 0.5, 0.35, 0.25, 0.25, 0.15, 0.15, 0.5]
 s = [0.3, 0.6, 0.55, 0.5, 0.3, 0.15, 0.05, 0]
are as follows:
 |lambda| = 0.7665
 |lambda| = 0.3648
 |lambda| = 0.3648
 |lambda| = 0.3483
 |lambda| = 0.3483
 |lambda| = 0.0000
 |lambda| = 0.0000
 |lambda| = 0.0000
```

[71]:
```
fB = [.50,.70,.55,.35,.35,.15,.15,.5]
sB = [.40,.70,.55,.50,.35,.15,.05,0]
print(display(fB,sB))
```

```
Eigenvalues lambda of A when
 f = [0.5, 0.7, 0.55, 0.35, 0.35, 0.15, 0.15, 0.5]
 s = [0.4, 0.7, 0.55, 0.5, 0.35, 0.15, 0.05, 0]
are as follows:
 |lambda| = 1.0105
 |lambda| = 0.3650
 |lambda| = 0.3650
 |lambda| = 0.3749
 |lambda| = 0.3749
```

```
 |lambda| = 0.1788
 |lambda| = 0.1729
 |lambda| = 0.1729
```

Let's look at one more example, now where the organisms have a max life-span of 4 time units (for simplicity!)

Let's consider

```
fC = [0, .2, .49559, 0.4]
sC = [.98, .96, .9, 0]
```

[134]:
```
fC = [0.000, .2, .49559, 0.399]
sC = [.9799, .96, .9, 0]
print(display(fC,sC))

e_vals,e_vecs = npl.eig(A(fC,sC))
print(f"\nIn fact, the largest eigenvalue is lambda = {e_vals[0]}")
print(f"\n& the corresponding eigenvector is {e_vecs[:,0]}")
```

```
Eigenvalues lambda of A when
 f = [0.0, 0.2, 0.49559, 0.399]
 s = [0.9799, 0.96, 0.9, 0]
are as follows:
 |lambda| = 1.00000
 |lambda| = 0.75089
 |lambda| = 0.75089
 |lambda| = 0.59912

In fact, the largest eigenvalue is lambda = (0.9999969131763253+0j)

& the corresponding eigenvector is [0.5298541 +0.j 0.51920563+0.j 0.49843895+0.j
0.44859644+0.j]
```

4.14 Explainer

In each case, the matrix A has distinct eigenvalues (in case `C` there are two eigenvalues with the same absolute value, but they are complex and distinct from one another!) Thus A is diagonalizable in each case.

For the parameters `fA,sA` all eigenvalues of A have absolute value < 1. This confirms our previous conclusion that

$$A^m \to \mathbf{0} \quad \text{as } m \to \infty$$

For the parameters `fB,sB` there is an eigenvalue of A which has abolute value $1.01 > 1$ (actually, this 1.01 *is* the eigenvalue). Thus A^m has no limiting value as $m \to \infty$.

Finally, the parameters `fC,sC` yield an eigenvalue of A which is very close to 1.

4.15 fC,sC

In this setting, note that the corresponding 1-eigenvector is

`w=[0.5298541, 0.51920563, 0.49843895, 0.44859644]`

Let's normalize the vector `w` by dividing by the sum of its components:

[131]:
```
w=np.array([0.5298541, 0.51920563, 0.49843895, 0.44859644])
ww = (1/sum(w,0))*w
ww
```

[131]:
```
array([0.27, 0.26, 0.25, 0.22])
```

Thus the components of `ww` sum to 1. They represent *probabilities*.

We conclude that the expected longterm population distribution in this case is:

Age 0	Age 1	Age 2	Age 3
27 %	26 %	25 %	22 %