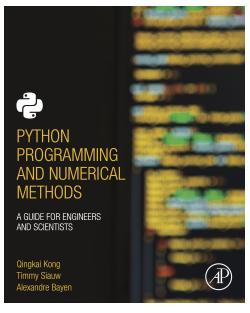

Contents

Find the largest eigenvalue

The inverse power method

The shifted power method

< 15.1 Mathematical Characteristics of Eigen-problems | Contents | 15.3 The QR Method >

The Power Method

Find the largest eigenvalue

In some problems, we only need to find the largest dominant eigenvalue and its corresponding eigenvector. In this case, we can use the power method - a iterative method that will converge to the largest eigenvalue. Let's see the following how the power method works.

Consider an $n \times n$ matrix A that has n linearly independent real eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ and the corresponding eigenvectors $v_1, v_2, \ldots, v_n$. Since the eigenvalues are scalars, we can rank them so that $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$ (actually, we only require $|\lambda_1| > |\lambda_2|$, other eigenvalues may be equal to each other).

Because the eigenvectors are independent, they are a set of basis vectors, which means that any vector that is in the same space can be written as a linear combination of the basis vectors. That is, for any vector x_0, it can be written as:

$$x_0 = c_1 v_1 + c_2 v_2 + \cdots + c_n v_n$$

where $c_1 \neq 0$ is the constraint. If it is zero, then we need to choose another initial vector so that $c_1 \neq 0$.

Now let's multiply both sides by A:

$$Ax_0 = c_1 A v_1 + c_2 A v_2 + \cdots + c_n A v_n$$

Since $Av_i = \lambda v_i$, we will have:

$$Ax_0 = c_1 \lambda_1 v_1 + c_2 \lambda_2 v_2 + \cdots + c_n \lambda_n v_n$$

We can change the above equation to:

$$Ax_0 = c_1 \lambda_1 [v_1 + \frac{c_2}{c_1}\frac{\lambda_2}{\lambda_1} v_2 + \cdots + \frac{c_n}{c_1}\frac{\lambda_n}{\lambda_1} v_n] = c_1 \lambda_1 x_1$$

where x_1 is a new vector and $x_1 = v_1 + \frac{c_2}{c_1}\frac{\lambda_2}{\lambda_1} v_2 + \cdots + \frac{c_n}{c_1}\frac{\lambda_n}{\lambda_1} v_n$.

This finishes the first iteration. And we can multiply A to x_1 to start the 2nd iteration:

$$Ax_1 = \lambda_1 v_1 + \frac{c_2}{c_1}\frac{\lambda_2^2}{\lambda_1} v_2 + \cdots + \frac{c_n}{c_1}\frac{\lambda_n^2}{\lambda_1} v_n$$

Similarly, we can rearrange the above equation to:

$$Ax_1 = \lambda_1 [v_1 + \frac{c_2}{c_1}\frac{\lambda_2^2}{\lambda_1^2} v_2 + \cdots + \frac{c_n}{c_1}\frac{\lambda_n^2}{\lambda_1^2} v_n] = \lambda_1 x_2$$

where x_2 is another new vector and $x_2 = v_1 + \frac{c_2}{c_1}\frac{\lambda_2^2}{\lambda_1^2} v_2 + \cdots + \frac{c_n}{c_1}\frac{\lambda_n^2}{\lambda_1^2} v_n$.

We can continue multiply A with the new vector we get from each iteration k times:

$$Ax_{k-1} = \lambda_1[v_1 + \frac{c_2}{c_1}\frac{\lambda_2^k}{\lambda_1^k}v_2 + \cdots + \frac{c_n}{c_1}\frac{\lambda_n^k}{\lambda_1^k}v_n] = \lambda_1 x_k$$

Because λ_1 is the largest eigenvalue, therefore, the ratio $\frac{\lambda_i}{\lambda_1} < 1$ for all $i > 1$. Thus when we increase k to sufficient large, the ratio of $(\frac{\lambda_n}{\lambda_1})^k$ will be close to 0. So that all the terms that contain this ratio can be neglected as k grows:

$$Ax_{k-1} = \lambda_1 v_1$$

Essentially, as k is large enough, we will get the largest eigenvalue and its corresponding eigenvector. When implementing this power method, we usually normalize the resulting vector in each iteration. This can be done by factoring out the largest element in the vector, which will make the largest element in the vector equal to 1. This normalization will get us the largest eigenvalue and its corresponding eigenvector at the same time. Let's take a look of the following example.

You may ask when should we stop the iteration? The basic stopping criteria should be one of the three: in the consecutive iterations, (1) the difference between eigenvalues is less than some specified tolerance; (2) the angle between eigenvectors is smaller than a threshold ; or the norm of the residual vector is small enough.

TRY IT! We know from last section that the largest eigenvalue is 4 for matrix $A = \begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix}$, now use the power method to find the largest eigenvalue and the associated eigenvector. You can use the initial vector [1, 1] to start the iteration.

1st iteration: $$

$$\begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{8}$$

=\begin{bmatrix} 2\5\ \end{bmatrix} =5\begin{bmatrix} 0.4\1\ \end{bmatrix} $$

2nd iteration: $$

$$\begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix} \tag{9}$$

$$\begin{bmatrix} 0.4 \\ 1 \end{bmatrix} \tag{10}$$

=\begin{bmatrix} 2\3.8\ \end{bmatrix} =3.8\begin{bmatrix} 0.5263\1\ \end{bmatrix} $$

3rd iteration: $$

$$\begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix} \tag{11}$$

$$\begin{bmatrix} 0.5263 \\ 1 \end{bmatrix} \tag{12}$$

=\begin{bmatrix} 2\ 4.0526\ \end{bmatrix} = 4.0526\begin{bmatrix} 0.4935\1\ \end{bmatrix} $$

4th iteration: $$

$$\begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix} \tag{13}$$

$$\begin{bmatrix} 0.4935 \\ 1 \end{bmatrix} \tag{14}$$

=\begin{bmatrix} 2\ 3.987\ \end{bmatrix} = 3.987\begin{bmatrix} 0.5016\1\ \end{bmatrix} $$

5th iteration: $$

$$\begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix} \tag{15}$$

$$\begin{bmatrix} 0.5016 \\ 1 \end{bmatrix} \tag{16}$$

=\begin{bmatrix} 2\ 4.0032\ \end{bmatrix} = 4.0032\begin{bmatrix} 0.4996\1\ \end{bmatrix} $$

6th iteration: $$

$$\begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix} \tag{17}$$

$$\begin{bmatrix} 0.4996 \\ 1 \end{bmatrix} \tag{18}$$

=\begin{bmatrix} 2\ 3.9992\ \end{bmatrix} = 3.9992\begin{bmatrix} 0.5001\1\ \end{bmatrix} $$

7th iteration: $$

$$\begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix} \tag{19}$$

$$\begin{bmatrix} 0.5001 \\ 1 \end{bmatrix} \tag{20}$$

=\begin{bmatrix} 2\ 4.0002\ \end{bmatrix} = 4.0002\begin{bmatrix} 0.5000\1\ \end{bmatrix} $$

We can see after 7 iterations, the eigenvalue converged to 4 with [0.5, 1] as the corresponding eigenvector.

TRY IT! Implement the power method in Python

```
import numpy as np
```

```
def normalize(x):
    fac = abs(x).max()
    x_n = x / x.max()
    return fac, x_n
```

```
x = np.array([1, 1])
a = np.array([[0, 2],
              [2, 3]])

for i in range(8):
    x = np.dot(a, x)
    lambda_1, x = normalize(x)

print('Eigenvalue:', lambda_1)
print('Eigenvector:', x)
```

```
Eigenvalue: 3.999949137887188
Eigenvector: [0.50000636 1.        ]
```

The inverse power method

The eigenvalues of the inverse matrix A^{-1} are the reciprocals of the eigenvalues of A. We can take advantage of this feature as well as the power method to get the smallest eigenvalue of A, this will be basis of the inverse power method. The steps are very simple, instead of multiplying A as described above, we just multiply A^{-1} for our iteration to find the largest value of $\frac{1}{\lambda_1}$, which will be the smallest value of the eigenvalues for A. As for the inverse of the matrix, in practice, we can use the methods we covered in the previous chapter to calculate it. We won't got to the details here, but let's see an example.

TRY IT! Find the smallest eigenvalue and eigenvector for $A = \begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix}$.

```
from numpy.linalg import inv
```

```
a_inv = inv(a)

for i in range(8):
    x = np.dot(a_inv, x)
    lambda_1, x = normalize(x)

print('Eigenvalue:', lambda_1)
print('Eigenvector:', x)
```

```
Eigenvalue: 0.20000000000003912
Eigenvector: [1. 1.]
```

The shifted power method

In some cases, we need to find all the eigenvalues and eigenvectors instead of the largest and smallest. One simple but inefficient way is to use the shifted power method (we will introduce you an efficient way in next section). Given $Ax = \lambda x$, and λ_1 is the largest eigenvalue obtained by the power method, then we can have:

$$[A - \lambda_1 I]x = \alpha x$$

where α's are the eigenvalues of the shifted matrix $A - \lambda_1 I$, which will be $0, \lambda_2 - \lambda_1, \lambda_3 - \lambda_1, \ldots, \lambda_n - \lambda_1$.

Now if we apply the power method to the shifted matrix, then we can determine the largest eigenvalue of the shifted matrix, i.e. α_k. Since $\alpha_k = \lambda_k - \lambda_1$, we can get the eigenvalue λ_k easily. We can repeat this process many times to find the all the other eigenvalues. But you can see that, it involves a lot of work! A better method for finding all the eigenvalues is to use the QR method, let's see the next section how it works!

< 15.1 Mathematical Characteristics of Eigen-problems | Contents | 15.3 The QR Method >