Chapter 2: EIGENVALUES AND EIGENVECTORS

2.1 Eigenvalues and eigenvectors problem statement

2.1.1 Eigenvalues and eigenvectors

We learned from the previous chapter that matrix A applied to a column vector x, that is, Ax, is a linear transformation of x. There is a special transform in the following form:

$$Ax = \lambda x,$$

where A is an $n \times n$ matrix, x is an $n \times 1$ column vector ($x \neq 0$), and λ is a scalar. Any λ that satisfies the above equation is known as an **eigenvalue** of the matrix A, while the associated vector x is called an **eigenvector** corresponding to λ.

2.1.2 The motivation behind eigenvalues and eigenvectors

The motivation behind eigenvalues and eigenvectors is that if we understand the characteristics of the linear transformation, it will help simplify the solutions to our problem. Say, we can multiply a vector A to another vector x, i.e., Ax. It essentially transforms the vector x into another vector, whereby the transformation represents a scale of the length of the vector and/or the rotation of the vector. The above equation points out that for some vectors, the effect of transformation Ax is only scaling (stretching, compressing, and flipping). The eigenvectors are the vectors having this property and the eigenvalues λ are the scale factors. Let us look at the following example.

Problem 2.1 Plot the vector $x = (0,1)$ and the vector $b = Ax$, where $A = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$

Ans:

Problem 2.2 Plot the vector x = (1,0) and the vector b = Ax where A = $\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$

Ans:

with x = (1,0) and $\lambda = 2$. The direction of the vector does not change at all (no rotation). You can also check that (0,1) is another eigenvector; try to verify this by yourself.

2.1.3 The characteristic equation

In order to get the eigenvalues and eigenvectors, from Ax = λx, we can use the following form:

$$(A - \lambda I)x = 0,$$

where I is the identify matrix with the same dimensions as A. If matrix $A - \lambda I$ has an inverse and both sides are multiplied by $(A - \lambda I)^{-1}$, we get a trivial solution x=0. Therefore, the only interesting case is when $A - \lambda I$ is singular (no inverse exists), and we have a nontrivial solution, which means that the determinant is zero:

$$\det(A - \lambda I) = 0.$$

This equation is called the **characteristic equation** that will lead to a polynomial equation for λ, which we can solve for the eigenvalues; see the example below.

Problem 2.3 Obtain the eigenvalues for matrix $\begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix}$

Ans:

Problem 2.4 Obtain the eigenvectors for the above two eigenvalues.

Ans:

2.2 The power method

2.2.1 Finding the largest eigenvalue

Some problems only require finding the largest dominant eigenvalue and its corresponding eigenvector. In this case, we can use the **power method**, which is an iterative method that will converge to the largest eigenvalue; see the example below.
Consider an $n \times n$ matrix A that has n real eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ and the corresponding linearly independent eigenvectors $v_1, v_2, \ldots, v_n$. Since the eigenvalues are scalars, we can rank them so that:

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|.$$

Note that we only require $|\lambda_1| > |\lambda_2|$; the other eigenvalues may be equal to each other.
Because the eigenvectors are assumed linearly 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 and can be written as:

$$x_0 = c_1v_1 + c_2v_2 + \cdots + c_nv_n$$

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 us multiply both sides by *A*:

$$Ax_0 = c_1Av_1 + c_2Av_2 + \cdots + c_nAv_n.$$

Since $Av_i = \lambda_i v_i$,

$$Ax_0 = c_1\lambda_1v_1 + c_2\lambda_2v_2 + \cdots + c_n\lambda_nv_n.$$

We can change the above equation to:

$$Ax_0 = c_1\lambda_1\left[v_1 + \frac{c_2}{c_1}\frac{\lambda_2}{\lambda_1}v_2 + \cdots + \frac{c_n}{c_1}\frac{\lambda_n}{\lambda_1}v_n\right] = c_1\lambda_1x_1.$$

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

This finishes the first iteration. For the second iteration, we apply *A* to x_1:

$$Ax_1 = \lambda_1v_1 + \frac{c_2}{c_1}\frac{\lambda_2^2}{\lambda_1}v_2 + \cdots + \frac{c_n}{c_1}\frac{\lambda_n^2}{\lambda_1}v_n.$$

Similarly, we can rearrange the above equation to get

$$Ax_1 = \lambda_1v_1 + \frac{c_2}{c_1}\frac{\lambda_2^2}{\lambda_1^2}v_2 + \cdots + \frac{c_n}{c_1}\frac{\lambda_n^2}{\lambda_1^2}v_n = \lambda_1x_2.$$

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

If we continue applying *A* to the new vector, we obtain from each iteration *k*:

$$Ax_{k-1} = \lambda_1\left[v_1 + \frac{c_2}{c_1}\frac{\lambda_2^k}{\lambda_1^k}v_2 + \cdots + \frac{c_n}{c_1}\frac{\lambda_n^k}{\lambda_1^k}v_n\right] = \lambda_1x_k.$$

Because λ_1 is the largest eigenvalue, the ratio $\frac{\lambda_i}{\lambda_1} < 1$ for all $i > 1$. When *k* is sufficiently large, the factor $\left(\frac{\lambda_n}{\lambda_1}\right)^k$ will be close to zero, so that all terms that contain this factor can be neglected as *k* increases:

$$Ax_{k-1} \sim \lambda_1v_1.$$

Essentially, if k is large enough, we will obtain the largest eigenvalue and its corresponding eigenvector. When implementing this power method, the resulting vector in each iteration is usually normalized. 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 provide the largest eigenvalue and its corresponding eigenvector at the same time; see the example below.

When should we stop the iteration? The basic stopping criterion should be one of these the following: (1) the difference between eigenvalues is less than some specified tolerance; (2) the angle between eigenvectors is smaller than a threshold; or (3) the norm of the residual vector is small enough.

Problem 2.5 We know from the last section that the largest eigenvalue is 4 for the matrix $A = \begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix}$

Ans:

Problem 2.6 Implement the power method in Python.

Ans:

2.2.2 The inverse power method

The eigenvalues of the inverse matrix A^{-1} are the reciprocals of the eigenvalues of A. By taking advantage of this feature, as well as the power method, we are able to obtain the smallest eigenvalue of A; this will be basis of the **inverse power method**. The steps are very simple: instead of applying A as described above, we just apply 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. In practice, we can use the methods we covered in the previous chapter to calculate the inverse of the matrix. We will not go to greater detail here, but we present an example below.

Problem 2.7 Find the smallest eigenvalue and eigenvector for $A = \begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix}$

Ans:

2.2.3 The shifted power method

In some cases, it is necessary to find all the eigenvalues and eigenvectors instead of just the largest and smallest. One simple, but inefficient way is to use the **shifted power method**; we will introduce you a more efficient method in the next section.
Given $Ax = \lambda_1 x$, and λ_1 being the largest eigenvalue obtained by the power method, we 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, $\lambda_2 - \lambda_1$, $\lambda_3 - \lambda_1$,...,$-\lambda_n - \lambda_1$.

Now if we apply the power method to the shifted matrix, we can determine the largest eigenvalue of the shifted matrix, i.e., α_k. Since $\alpha_k = \lambda_k - \lambda_1$, we can obtain the eigenvalue λ_k easily. Repeating this process many times will find the all the other eigenvalues, but you can see it is very labor intensive. A better method for finding all the eigenvalues is to use the QR method, which we will introduced next.

2.3 The QR method

The **QR method** is the preferred iterative method to find all the eigenvalues of a matrix (but not the eigenvectors at the same time). The idea is based on the following two concepts:

1.Similar matrices will have the same eigenvalues and associated eigenvectors. Two square ma- trices A and B are similar if

$$A = C^{-1}BC$$

where C is an invertible matrix.

2.The QR method is a way to decompose a matrix into two matrices Q and R, where Q is an orthogonal matrix, and R is an upper triangular matrix. An orthogonal matrix satisfies Q−1 = QT , which means Q−1Q = QT Q = I .

How do we link these two concepts to find the eigenvalues? Say, we have a matrix A_0 whose eigenvalues must be determined. At the kth step (starting with $k = 0$), we can perform the QR decomposition and obtain

$$A_k = Q_k R_k$$

where Qk is an orthogonal matrix, and R_k is an upper triangular matrix. We then form

$$A_{k+1} = R_k Q_k$$

to obtain

$$A_{k+1} = R_k Q_k = Q^{-1} Q_k R_k Q_k = Q_k{}^{-1} A_k Q_k .$$

Because all the A_k are similar, as we discussed above, they all have the same eigenvalues. As the iteration continues, we will eventually converge to an upper triangular matrix with the form:

$$A_k = R_k Q_k = \begin{bmatrix} \lambda_1 & X & \dots & X \\ 0 & \lambda_2 & \dots & X \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix},$$

where the diagonal values are the eigenvalues of the matrix. In each iteration of the QR method, factoring a matrix into an orthogonal and an upper triangular matrix can be done by using a special matrix called **Householder matrix**. We will not go into the mathematical details how you get the *Q* and *R* from the matrix. Instead, we will use the Python function to obtain the two matrices directly.

Problem 2.8 Use the `qr` function in `numpy.linalg` to decompose matrix $A = \begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix}$. Verify the results.

Ans:

Problem 2.9 Use the QR method to get the eigenvalues of matrix A = $\begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix}$. Do 20 iterations, and print out the first, fifth, 10th, and 20th iteration.

Ans:

2.4 Eigenvalues and eigenvectors in python

The methods introduced above are fairly complicated to execute. The calculation of eigenvalues and eigenvectors in Python is fairly easy. The main built-in function in Python to solve the eigenvalue/eigenvector problem for a square array is the `eig` function in `numpy.linalg`; see below for an example in how to execute it.

Problem 2.10 Calculate the eigenvalues and eigenvectors for matrix $A = \begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix}$.

Ans:

Problem 2.11 Compute the eigenvalues and eigenvectors for the matrix $A = \begin{bmatrix} 2 & 2 & 4 \\ 1 & 3 & 5 \\ 2 & 3 & 4 \end{bmatrix}$

Ans:

2.5 Summary

2.5.1 Summary

1. Eigenvalues and eigenvectors help us understand the characteristics of a linear transformation.
2. Eigenvectors of a matrix are the vectors that can only be scaled lengthwise without rotation after applying the matrix transformation; the eigenvalues are the factors of the scaling.
3. We can use power method to get the largest eigenvalue and corresponding eigenvector of a matrix.
4. The inverse power method can help us get the smallest eigenvalue and corresponding eigenvector of a matrix.
5. The shifted power method can get all the other eigenvectors/eigenvectors of a matrix.
6. The preferred method to get all the eigenvalues is the QR method.