**INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO**

*Algoritmos Numéricos por Computadora*

# *Eigenvalores y Eigenvectores*


Profesor:

Doctor Ángel Fernando Kuri Morales

Alumnos:

José Miguel Fernández Cardoso 181332

Francesca Perrone Jimenez 179682

Introducción:

Los eigenvectores (vectores característicos) de una transformación lineal son vectores, distintos al vector cero, que solamente cambian según un escalar cuando dicha transformación se aplica. Es decir: son vectores que, al ser transformados, solamente cambian en magnitud (son multiplicados por un valor escalar). Estos vectores tienen valores correspondientes (los eigenvalores/ valores propios) denotados casi siempre por λ. Este es el factor por el cual el eigenvector es escalado.

La siguiente igualdad muestra la propiedad de la que estamos hablando.

$$A\overline{v} = \lambda\overline{v}$$

El producto de la matriz A por el vector propio $\overline{v}$ da exactamente el mismo resultado que daría escalar al vector $\overline{v}$ por un escalar λ.

Para encontrar cuales son los eigenvalores y los eigenvectores tradicionalmente lo que se hace es utilizar a la matriz identidad. Multiplicamos a nuestro escalar λ con la matriz identidad para así poder restar ambas matrices y factorizar al eigenvector $\overline{v}$.

$$(A - \lambda I)\,\overline{v} = \overline{0}$$

Como establecimos anteriormente, el eigenvector es un vector distinto al vector cero, por lo tanto tenemos que encontrar un vector $\overline{v}$ que multiplicado por la matriz $(A - \lambda I)$ nos de al vector cero. Para poder lograr esto vamos a necesitar sacar la determinante de $(A - \lambda I)$ e igualarlo a cero. Al conseguir el $det(A - \lambda I)$ lo que obtenemos es al polinomio característico de la matriz A.

El determinante es cero intuitivamente si se colapsan las dimensiones del espacio que puedes generar con la matriz (lo que indica dependencia lineal). El determinante (al aplicarse a una matriz) nos dice como se distorsiono el espacio con la transformación aplicada. El determinante, si los vectores son linealmente dependientes, es cero porque no se genera el espacio que generaría si fueran linealmente independientes(se colapsa el espacio a una dimensión menor y ya no tiene sentido describirla con un número (por eso el cero)). Es decir, deja de existir lo que puedes generar con esos vectores si pasa a una dimensión menor. Cómo una matriz con eigenvectores siempre va a tener un determinante distinto de cero,

restarle lambda y encontrar la raíz del polinomio característico nos va decir los espacios en los que aplicar una transformación no distorsionara al espacio.

Es por esto que buscamos las raíces del polinomio característico, pues sabemos que necesitamos que nuestro determinante sea cero. Estas raíces son los valores de lambda.

Para encontrar los vectores propios correspondientes a los valores propios, evaluamos la matriz $(A-\lambda I)$ sustituyendo el valor de lambda por uno de los eigenvalores y luego encontramos para que valore de la matriz modificada se llega al vector cero (con eliminación Gauss-Jordan).

Ejemplo de eigenvalores:
Sea $A$ la siguiente matriz de 3x3.

$$A=\begin{pmatrix}1 & -3 & 3\\ 0 & -5 & 6\\ 0 & -3 & 4\end{pmatrix}$$

Al sacar el determinante de $(A-\lambda I)$ obtenemos el polinomio característico.

$$|A-\lambda I|=\begin{pmatrix}1-\lambda & -3 & 3\\ 0 & -5-\lambda & 6\\ 0 & -3 & 4-\lambda\end{pmatrix}$$

$$el\det(|A-\lambda I|)=-\lambda^3+3\lambda-2$$

$$el\det(|A-\lambda I|)=(\lambda-1)(\lambda+2)(\lambda-1)$$

Al igualar el polinomio característico a cero obtenemos los eigenvalores.

$$(\lambda-1)(\lambda+2)(\lambda-1)=0$$

$$\lambda_1=\lambda_2=1$$
$$\lambda_3=-2$$

A partir de estos eigenvalores podemos sustituir $\lambda$ en $A$ y resolver el Gauss-Jordan.

$$\begin{pmatrix} 1-\lambda & -3 & 3 \\ 0 & -5-\lambda & 6 \\ 0 & -3 & 4-\lambda \end{pmatrix} \begin{vmatrix} 0 \\ 0 \\ 0 \end{vmatrix} = \begin{pmatrix} 1-1 & -3 & 3 \\ 0 & -5-1 & 6 \\ 0 & -3 & 4-1 \end{pmatrix} \begin{vmatrix} 0 \\ 0 \\ 0 \end{vmatrix}$$

Esta forma de calcular los eigenvalores (y de ahí obtener los eigenvectores) es relativamente sencilla para matrices cuadradas de tamaño pequeño. Conforme se incrementa el tamaño de la matriz, la dificultad para calcular los eigenvalores incrementa.

La implementación de los eigenvalores en la vida real es más compleja. En muchos casos se trabaja con matrices enormes (como de 200 x 200) y si queremos encontrar los valores de estos eigenvalores nos tardaríamos muchísimo. Es por esto que existen algoritmos para aproximarnos a los valores de los eigenvalores.

Hay varias formas para calcular eigenvalores y los eigenvectores. El más sencillo es el del método de la potencia. También existe el método QR, el cual es el más estable y el más eficiente hasta ahora.

<u>Cálculo de los eigenvalores mediante el método de la potencia.</u>

El método de la potencia es un algoritmo que funciona con matrices diagonalizables. El algoritmo regresa un número $\lambda$, el cual es el valor más grande de los eigenvalores en la matriz y también regresa el eigenvector correspondiente ($\overline{v}$) de $\lambda$.

El problema con este algoritmo es que la convergencia de los valores es lenta (en especial en el paso de la multiplicación de las matrices) y por eso su implementación es más efectiva con matrices dispersas.

Sabemos que una matriz de $n \times n$ $A$ tiene $n$ eigenvectores linealmente independientes,

$$\overline{v_1},\ \overline{v_2},\ \ldots,\ \overline{v_n}$$

y estos eigenvectores tienen $n$ eigenvalores correspondientes.

$$\lambda_1,\ \lambda_n,\ \ldots,\ \lambda_n$$

Sabemos que cualquier vector $x$ se puede expresar como la combinación lineal de sus $n$ eigenvectores linealmente independientes. (Así es como en el espacio podemos llegar a cualquier punto, digamos en el espacio $R^2$ con cualquier 2 vectores LI).

$$x = c_1 v_1 + c_2 v_2 + c_3 v_3 + ... + c_n v_n$$

Lo que hacemos en el método de la potencia es multiplicar ambos lados de la ecuación anterior por la matriz $A$.

$$Ax = c_1 A v_1 + c_2 A v_2 + c_3 A v_3 + ... + c_n A v_n$$

Lo cual, por la igualdad $A\overline{v} = \lambda\overline{v}$, sabemos que es lo mismo que escalar a los vectores por las respectivas $\lambda$

$$Ax = c_1 \lambda_1 v_1 + c_2 \lambda_2 v_2 + c_3 \lambda_3 v_3 + ... + c_n \lambda_n v_n$$

Ahora, si multiplicamos por la matriz $A$ repetidas veces llegamos a lo siguiente.

$$\begin{aligned}
Ax &= c_1 \lambda_1 v_1 + c_2 \lambda_2 v_2 + c_3 \lambda_3 v_3 + ... + c_n \lambda_n v_n \\
A^2 x &= c_1 \lambda_1^2 v_1 + c_2 \lambda_2^2 v_2 + c_3 \lambda_3^2 v_3 + ... + c_n \lambda_n^2 v_n \\
A^3 x &= c_1 \lambda_1^3 v_1 + c_2 \lambda_2^3 v_2 + c_3 \lambda_3^3 v_3 + ... + c_n \lambda_n^3 v_n \\
&\vdots \\
A^k x &= c_1 \lambda_1^k v_1 + c_2 \lambda_2^k v_2 + c_3 \lambda_3^k v_3 + ... + c_n \lambda_n^k v_n
\end{aligned}$$

(esto sucede porque podemos escribir cualquier vector como la suma de eigenvectores escalados)

El eigenvalor dominante es el eigenvalor que tiene la magnitud más grande de todos los eigenvalores de la matriz $A$.

Asumamos que $\lambda_1$ en este caso es el eigenvalor dominante y factoricemos.

$$= \lambda_1^k \left( c_1 v_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k v_2 + c_3 \left(\frac{\lambda_3}{\lambda_1}\right)^k v_3 + ... + c_n \left(\frac{\lambda_n}{\lambda_1}\right)^k v_n \right)$$

Como $\lambda_1$ es el eigenvalor dominante (es decir, el que tiene la magnitud más grande) todas las cocientes $\frac{\lambda_2}{\lambda_1}, \frac{\lambda_3}{\lambda_1}, ..., \frac{\lambda_n}{\lambda_1}$ van a tener magnitudes menores a 1, y sabemos que al elevar números menores a 1 con potencias cada vez más grandes, estos números tienden a cero. Así que si iteramos lo suficiente ($A^k$, $A^{k+1}$, $A^{k+2}$, ...) los cocientes eventualmente desaparecerán (pues se acercan a 0). El resultado en convergencia, por lo tanto, es la siguiente ecuación.

$$A^k x = c_1 \lambda_1^k v_1$$

Así es como, tomando un vector cualquiera $x$, eventualmente nos aproximamos al eigenvector de la matriz $A$. una vez que obtenemos el eigenvector, el eigenvalor es fácil de obtener.

Un problema que el algoritmo tiene, es que si el eigenvalor dominante tiene un valor más grande que 1, el término $c_1 \lambda_1^k v_1$ se podría salir de proporción y nuestro eigenvector terminaría siendo enorme. Es por esto que en cada iteración el vector $x$ es multiplicado por la matriz $A$ y luego normalizado.

Es importante saber que si asumimos que la matriz $A$ tiene un eigenvalor dominante (que es estrictamente mayor en magnitud a comparación de los otros eigenvalores) y el vector $x$ es distinto de cero, entonces la secuencia del método convergerá a un eigenvector asociado al eigenvalor dominante. Sin asumir esto puede que la secuencia no necesariamente converja.

Conclusión:

Hay muchas usos para los eigenvalores aparte de la diagonalización de matrices, tales como análisis de estabilidad, análisis de vibraciones, análisis de orbitales y reconocimiento facial. El método de la potencia es uno de los métodos iterativos más fáciles para obtener los eigenvalores y los eigenvectores dominantes, pero aún hay métodos más complejos que logran aproximarse a todos los eigenvalores y los eigenvectores de una matriz. El método más completo hasta hoy es el método QR, que descompone una matriz $A$ (QR decomposition) y la vuelve un producto de una matriz ortogonal $Q$ y una matriz triangular superior $R$. A lo largo de las iteraciones de $A$ se observa que las matrices resultantes $A_1, ..., A_k$ son matrices similares, lo que hace que el algoritmo sea numéricamente más estable (pues funciona a través de “orthogonal similarity transformations”). Esto no le resta importancia al método de la potencia, pues en muchos cálculos el resultado importante es el del eigenvalor y el eigenvector dominante. En particular, Google lo emplea para calcular el PageRank de los documentos en su motor de búsqueda.

Código:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
```

```
@author: administrador
"""

# Power Method to Find Largest Eigen Value and Eigen Vector
# Importing NumPy Library
import numpy as np
import sys

def NumCols(Linea,Linelen):
    cols=1
    for i in range(0,LineLen):
        if (Linea[i]=="\t"):
            cols=cols+1
        #endIf
    #endFor
    return cols
#endFunction

def getNum(Linea,numVar):
    countVar=-1
    first=0
    for i in range(0,numVar+1):
        x=""
        for j in range(first,len(Linea)):
            if (Linea[j]=="\t" or Linea[j]=="\n"):
                first=j+1
                countVar=countVar+1
                break
            else:
                x=x+Linea[j]
            #endif
        #endFor
        if (countVar==numVar):
            return float(x)
        #endIf
    #endFor
#endFunction


Arch=input("Deme el nombre del ARCHIVO DE DATOS A ANALIZAR: \t")
try:
    FN = open(Arch,"r")
    print("\t*** Se abrió el archivo \""+Arch+"\"" )
except:
```

```
    print("\t*** No se encontró el archivo \""+Arch+"\"")
    sys.exit("**** Fin de programa ****")
lista = list(range(0))
lista=FN.readlines()
Filas=len(lista)
print("Filas: "+str(Filas))
FN = open(Arch,"r")
Lineas=list(range(Filas))
Lineas[0]=FN.readline()
LineLen=len(Lineas[0])
Columnas=NumCols(Lineas[0],LineLen)
print("Columnas: "+str(Columnas))
if (Columnas!=Filas):
    print("El número de filas de ser igual al número de columnas")
    print("**** \tFin de Programa ****\n\n\n")
    sys.exit()
FN = open(Arch,"r")
A=np.zeros((Filas,Filas))
for i in range(0,Filas):
    A[i]=list(range(Columnas))
for i in range(0,Filas):
    Lineas[i]=FN.readline()
    for j in range(0,Columnas):
        L=Lineas[i]
        Num=getNum(L,j)     #j-th variablesin i-th line
        A[i][j]=Num
    #endFor
#endFor


# Making numpy array n x 1 size and initializing to zero
# for storing initial guess vector
x = np.zeros((Filas))

# Reading initial guess vector
print('Ingresa el vector: ')
for i in range(Filas):
    x[i] = float(input( 'x['+str(i)+']='))

# Reading tolerable error
tolerable_error = float(input('Ingrresa el error tolerable: '))

# Reading maximum number of steps
max_iteration = int(input('Ingresa el máximo número de iteraciones: '))
```

```
# Power Method Implementation
lambda_old = 1.0
condition =  True
step = 1
while condition:
    # Multiplying a and x
    x = np.matmul(A,x)

    # Finding new Eigen value and Eigen vector
    lambda_new = max(abs(x))

    x = x/lambda_new

    # Displaying Eigen value and Eigen Vector
    print('\nIteración %d' %(step))
    print('----------')
    print('Eigen Value = %0.4f' %(lambda_new))
    print('Eigen Vector: ')
    for i in range(Filas):
        print('%0.3f\t' % (x[i]))

    # Checking maximum iteration
    step = step + 1
    if step > max_iteration:
        print('No convergió en el número máximo de iteraciones!')
        break

    # Calculating error
    error = abs(lambda_new - lambda_old)
    print('errror='+ str(error))
    lambda_old = lambda_new
    condition = error > tolerable_error
```

---

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""

@author: administrador
"""

import numpy as np
import sys
```

```
from numpy import linalg as LA


def NumCols(Linea,Linelen):
    cols=1
    for i in range(0,LineLen):
        if (Linea[i]=="\t"):
            cols=cols+1
        #endIf
    #endFor
    return cols
#endFunction

def getNum(Linea,numVar):
    countVar=-1
    first=0
    for i in range(0,numVar+1):
        x=""
        for j in range(first,len(Linea)):
            if (Linea[j]=="\t" or Linea[j]=="\n"):
                first=j+1
                countVar=countVar+1
                break
            else:
                x=x+Linea[j]
            #endif
        #endFor
        if (countVar==numVar):
            return float(x)
        #endIf
    #endFor
#endFunction

Arch=input("Deme el nombre del ARCHIVO DE DATOS A ANALIZAR: \t")
try:
    FN = open(Arch,"r")
    print("\t*** Se abrió el archivo \""+Arch+"\"" )
except:
    print("\t*** No se encontró el archivo \""+Arch+"\"")
    sys.exit("**** Fin de programa ****")

lista = list(range(0))
lista=FN.readlines()
Filas=len(lista)
```

```
print("Filas: "+str(Filas))
FN = open(Arch,"r")
Lineas=list(range(Filas))
Lineas[0]=FN.readline()
LineLen=len(Lineas[0])
Columnas=NumCols(Lineas[0],LineLen)
print("Columnas: "+str(Columnas))
if (Columnas!=Filas):
   print("El número de filas de ser igual al número de columnas")
   print("**** \tFin de Programa ****\n\n\n")
   sys.exit()

FN = open(Arch,"r")
A=np.zeros((Filas,Filas))
for i in range(0,Filas):
   A[i]=list(range(Columnas))
for i in range(0,Filas):
   Lineas[i]=FN.readline()
   for j in range(0,Columnas):
      L=Lineas[i]
      Num=getNum(L,j)     #j-th variablesin i-th line
      A[i][j]=Num
   #endFor
#endFor

print("\tMatriz de datos:")
print(A)
w,v=LA.eig(A)
print("\nEigenvalores = n")
print(w)
print("\nEigenvectores = u")
print(v)
print("\n")

print("comprobamos que si se cumpla la ecuación An = nu")
for i in range(0,Filas):
   u=v[:,i]
   lam=w[i]
   print("\nEigenvalor en la casilla "+str(i+1)+":  "+str(lam)+"\n")
   print(" Matriz por el Eigenvector (An):")
   print(np.dot(A,u))
   print(" Eigenvalor por el Eigenvector (nu)")
   print(lam*u)
   print("------------------------")
```

Páginas web utilizadas y para complementar:

- https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors
- http://www.physics.utah.edu/~detar/lessons/python/numpy_eigen/node1.html
- https://numpy.org/doc/stable/reference/generated/numpy.linalg.eig.html
- https://blog.nekomath.com/algebra-lineal-i-eigenvalores-y-eigenvectores-de-transformaciones-y-matrices/
- https://www.youtube.com/watch?v=Gx0PaWI9eYo
- https://www.youtube.com/watch?v=TS32rBteHh4
- https://en.wikipedia.org/wiki/Power_iteration
- http://mlwiki.org/index.php/Power_Iteration
- https://www.quantstart.com/articles/QR-Decomposition-with-Python-and-NumPy/