Practica 1

Gonzalez Borja, Miguel
Illescas Arizti, Rodrigo
Meyer Mañon, Juan Carlos
Rodriguez Orozco, Alejandro

11 / Marzo / 2018

Introducción

A continuacion los ejercicios de la practica. Todos fueron realizados en Python 3.6 utilizando los paquetes *numpy* y *scipy.linalg*. Tambien se incluye un cuaderno de IPython para cada ejercicio dentro de la carpeta Ejercicios, junto con los scripts *.py*. Cada uno de estos utiliza los siguientes imports:

```
In [1]: import sys
        sys.path.append('../')
        from IPython.display import Latex
        import latexStrings as ls
        import numpy as np
        import scipy.linalg as linear
        import eigenvalues as ev
```

Se puede encontrar un repositorio del proyecto aqui.

1 Ejercicio 1

Tenemos la matriz A y el vector q_0 definidos como:

```
In [1]: A = np.array([[1,1,2],[-1,9,3],[0,-1,3]])
        q = np.array([1,1,1])
```

$$A = \begin{pmatrix} 1 & 1 & 2 \\ -1 & 9 & 3 \\ 0 & -1 & 3 \end{pmatrix} \qquad \vec{q_0} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Queremos calcular 10 iteraciones del metodo de la potencia. Esto nos da como resultado:

```
In [2]: [q10, l10 ,iterations]=ev.powerMethod(A,q,1e-6,10)
```

Numero de Iteraciones: 10

$$\vec{q_{10}} = \begin{pmatrix} 0.083519 \\ 0.979588 \\ -0.182845 \end{pmatrix} \qquad \lambda = 8.35525106702442$$

Despues comparemos los resultados con los valores 'exactos' calculados por el paquete *scipy.linalg*:

```
In [3]: [L,V] = linear.eig(A)
```

$$\vec{\lambda} = \begin{pmatrix} 8.354545 \\ 1.224672 \\ 3.420784 \end{pmatrix} \qquad V = \begin{pmatrix} 0.083444 & -0.992728 & 0.515311 \\ 0.979576 & -0.104882 & -0.332386 \\ -0.182943 & -0.059078 & 0.789921 \end{pmatrix}$$

Vemos que en efecto el metodo de la potencia calculo el eigenvector dominante de la matriz. Esto se debe a que se cumplen las condiciones del metodo, es decir:

1. A tiene un eigenvalor dominante (8.3545)
2. Nuestro vector inicial q_0 puede ser escrito como combinacion lineal de los eigenvectores de A con coeficientes todos distintos de 0

Ahora, tomemos el eigenvector asociado al eigenvalos dominante, dado por:

```
In [4]: v=V[:,0]
```

$$\vec{v} = \begin{pmatrix} 0.083444 \\ 0.979576 \\ -0.182943 \end{pmatrix}$$

Y con esto queremos calcular las razones de convergencia en cada paso de la iteracion, dadas por:

$$\widetilde{r} = \{r_i\}, \qquad r_i = \frac{\|q_i - v\|_\infty}{\|q_{i-1} - v\|_\infty}, \qquad i \in \{1, 2, ..., 10\}$$

```
In [5]: ratios=[]
        prevq=q
        for i in range(1,11):
            [currentq,_,_] = ev.powerMethod(A,q,1e-6,i)
            ratio = linear.norm(currentq-v,np.inf)/linear.norm(prevq-v,np.inf)
            ratios.append(ratio)
            prevq = currentq
```

$\widetilde{r} = \{0.297033, 0.382353, 0.390998, 0.400885, 0.405792, 0.407929, 0.408825, 0.409194, 0.409346, 0.409409\}$

Observemos que en efecto, las razones de cada iteracion rapidamente al valor teorico, dado por:

$$r = \left|\frac{\lambda_2}{\lambda_1}\right| = 0.40945181373495726$$

2 Ejercicio 2.1

Utilizando la matriz A y el vector q_0 del ejercicio anterior definidos como sigue:

```
In [1]: A = np.array([[1,1,2],[-1,9,3],[0,-1,3]])
        q = np.array([1,1,1])
```

$$A = \begin{pmatrix} 1 & 1 & 2 \\ -1 & 9 & 3 \\ 0 & -1 & 3 \end{pmatrix} \quad \vec{q_0} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Se calcularán 10 iteraciones del método de la potencia con shift ρ_1 y ρ_2 y donde $\vec{q}_{10}$ es el vector que se aporxima al eigenvector y σ_{10} el eigenvalor aproximado después de 10 iteraciones. Para $\rho_1 = 0$, que es simplemente aplicar el método de la potencia inversa, se obtiene:

```
In [2]: [q10, l10 ,iterations]=ev.inversePowerShift(A,q,0,1e-6,10)
```

Numero de Iteraciones: 10

$$\vec{q_{10}} = \begin{pmatrix} 0.992719 \\ 0.104174 \\ 0.060466 \end{pmatrix} \quad \sigma_{10} = 1.2267894261411831$$

Comparemos los resultados anteriores con los valores "exactos" calculados por el paquete *scipy.linalg*:

```
In [3]: [L,V] = linear.eig(A)
        [L,V] = ev.pairSort(L,V)
```

$$\vec{\lambda} = \begin{pmatrix} 8.354545 \\ 3.420784 \\ 1.224672 \end{pmatrix} \quad V = \begin{pmatrix} 0.083444 & 0.515311 & -0.992728 \\ 0.979576 & -0.332386 & -0.104882 \\ -0.182943 & 0.789921 & -0.059078 \end{pmatrix}$$

$\vec{\lambda}$ es el vector que contiene a los tres eigenvalores "exactos" de la matriz A ordenados por magnitud y las columnas de la matriz V son los eigenvectores "exactos" de A.

Se observa que $\sigma_{10} = 1.22678942614$ tiene dos decimales iguales a $\lambda_3 = 1.224672$, la tercera entrada de $\vec{\lambda}$. El método de la potencia inversa converge teóricamente al menor eigenpar de A bajo las siguientes condiciones:

1. A tiene un menor eigenvalor, es decir $|\lambda_1| > |\lambda_2| > |\lambda_3|$ (En este caso 1.2246)

2. $\vec{q}_0$ puede ser escrito como combinacion lineal de los eigenvectores de A con los coeficientes de $\vec{v}_2$ y $\vec{v}_3$ distintos de 0. En este caso, $\vec{q}_0$ claramente no es ortogonal a los vectores de la matriz V, por lo tanto ninguno de los coeficientes de la combinacion lineal sera 0.

Ahora comparemos $\vec{v}_3$, el eigenvector asignado a λ_3, con $\vec{q}_{10}$.

```
In [4]: v=V[:,2]
```

$$\vec{v_3} = \begin{pmatrix} -0.992728 \\ -0.104882 \\ -0.059078 \end{pmatrix} \quad \vec{q_{10}} = \begin{pmatrix} 0.992719 \\ 0.104174 \\ 0.060466 \end{pmatrix}$$

Observamos que $\vec{q}_{10}$ es apximadamente $-\vec{v}_3$. Esto se debe a que el método de la potencia inversa puede converger a $\pm\vec{v}_3$. Entonces de ahora en adelante nos referiremos a $\vec{v}_3$ por $-\vec{v}_3$:

```
In [5]: v=-v
```

Out[6]:

$$\vec{v_3} = \begin{pmatrix} 0.992728 \\ 0.104882 \\ 0.059078 \end{pmatrix}$$

Calculamos las razones de convergencia en cada paso de la iteracion, dadas por:

$$\widetilde{r} = \{r_i\}, \qquad r_i = \frac{\|q_i - v\|_\infty}{\|q_{i-1} - v\|_\infty}, \qquad i \in \{1, 2, ..., 10\}$$

```
In [6]: ratios=[]
        prevq=q
        for i in range(1,10):
            [currentq,_,_] = ev.inversePowerShift(A,q,0,1e-6,i)
            ratio = linear.norm(currentq-v,np.inf)/linear.norm(prevq-v,np.inf)
            ratios.append(ratio)
            prevq = currentq
```

$$\widetilde{r} = \{0.752001, 0.966084, 0.853759, 0.678343, 0.495284, 0.404372, 0.373393, 0.363308, 0.359880\}$$

Ahora veamos la razon de convergencia teórica, dada por:

$$r = \left|\frac{\lambda_3}{\lambda_2}\right| = 0.35800909831776$$

Se oberva que, en efecto, $r_{10} = 0.359880$ es aproximademente la razón teórica $r = 0.35800909831776$

3 Ejercicio 2.2

Para la misma matriz A y vector q_0 defininidos anteriormente, se realizarán las mismas pruebas para un shift $\rho_2 = 3.3$. Se encontrarán nuevos: vector $\vec{q}_{10}$, la aproximación de eigenvector, y σ_{10} la aproximación del eigenvalor.

```
In [1]: [q10, l10 ,iterations]=ev.inversePowerShift(A,q,3.3,1e-6,10)
```

Numero de Iteraciones: 4

$$\vec{q_{10}} = \begin{pmatrix} 0.515311 \\ -0.332385 \\ 0.789921 \end{pmatrix} \quad \sigma_{10} = 3.420782623785237$$

La primera observación es que el proceso sólo hizo 4 iteraciones, esto quiere decir que el método alcanzó el criterio de error relativo sin necesidad de hacer las 10 iteraciones. Comparemos los resultados anteriores con los valores "exactos" calculados por el paquete *scipy.lingalg* calculados anteriormente

$$\vec{\lambda} = \begin{pmatrix} 8.354545 \\ 3.420784 \\ 1.224672 \end{pmatrix} \quad V = \begin{pmatrix} 0.083444 & 0.515311 & -0.992728 \\ 0.979576 & -0.332386 & -0.104882 \\ -0.182943 & 0.789921 & -0.059078 \end{pmatrix}$$

El vector $\vec{q}_{10}$ es aproximadamente al eigenvector "exacto" $\vec{v}_2$, segunda columna de la matriz V. Ademas, el valor σ_{10} tiene 5 cifras decimales iguales al eigenvalor "exacto" λ_2.

Observamos lo siguiente:

$$|\lambda_2 - \rho_2| < |\lambda_3 - \rho_2| < |\lambda_1 - \rho_2| \Rightarrow |\lambda_2 - \rho_2|^{-1} > |\lambda_3 - \rho_2|^{-1} > |\lambda_1 - \rho_2|^{-1}$$

Con un razonamiento similar al ejercicio anterior, al tener la desigualdad anterior, el método de la potencia inversa con shift $\rho_2 = 3.3$ cumple una de las hipótesis y el converge al valor $(\lambda_2 - \rho_2)^{-1}$ y el eigenvector asignado $\vec{v}_2$. Luego con un simple despeje obtenemos λ_2 y con ello el eigenpar de A $(\lambda_2, \vec{v}_2)$.

Nuevamente, el vector inicial $\vec{q}_0$ no es ortogonal a ninguno de los vectores de la matriz V, por lo tanto $\vec{q}_0$ se puede escribir como combinanción lineal de los vectores de V con coeficientes todos distintos de 0, cumpliendo la segunda hipótesis del metodo, justificando asi la convergencia.

Calculamos las razones de convergencia en cada paso de la iteracion, dadas por:

$$\widetilde{r} = \{r_i\}, \qquad r_i = \frac{\|q_i - v\|_\infty}{\|q_{i-1} - v\|_\infty}, \qquad i \in \{1, 2, ..., 10\}$$

```
In [2]: ratios=[]
        v=V[:,1]
        prevq=q
        for i in range(1,11):
            [currentq,_,_] = ev.inversePowerShift(A,q,3.3,0,i)
            ratio = linear.norm(currentq-v,np.inf)/linear.norm(prevq-v,np.inf)
            ratios.append(ratio)
            prevq = currentq
```

$$\widetilde{r} = \{0.014461, 0.024879, 0.020834, 0.032431, 0.021112,$$
$$0.085590, 0.050552, 0.061824, 0.056628, 0.062165\}$$

Ahora veamos la razon de convergencia teórica, dada por:

$$r = \left| \frac{\lambda_2 - \rho_2}{\lambda_3 - \rho_2} \right| = 0.05819972269618957$$

Observamos que las razones r_i comienzan a oscilar alrededor de $r = 0.05819972269618957$ (razón teórica) desde $i = 7$. Esto se debe a que como el método converge rápidamente y con más iteraciones la aproximación se vuelve prácticamente igual al eigenpar busacado. De hecho, si aumentamos las razones calculadas, veremos que:

```
In [3]: for i in range(11,21):
            [currentq,_,_] = ev.inversePowerShift(A,q,3.3,0,i)
            ratio = linear.norm(currentq-v,np.inf)/linear.norm(prevq-v,np.inf)
            ratios.append(ratio)
            prevq = currentq
```

$$\widetilde{r}_{10-20} = \{0.062165, 0.172414, 1.100000, 1.045455, 0.913043, 1.095238, \\ 0.913043, 1.095238, 0.913043, 1.095238, 0.913043\}$$

La razones de la práctica empiezan a acercarse a 1. Esto se debe a que $\vec{q}_i - \vec{v}$ es casi igual a $\vec{q}_{i-1} - \vec{v}$ por lo que la razón se acerca a 1.

4 Ejercicio 3

Tomando A y q_0 como los definimos anteriormente:

```
In [1]: A = np.array([[1,1,2],[-1,9,3],[0,-1,3]])
        q = np.array([1,1,1])
```

$$A = \begin{pmatrix} 1 & 1 & 2 \\ -1 & 9 & 3 \\ 0 & -1 & 3 \end{pmatrix} \quad \vec{q_0} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Se calcularán 10 iteraciones del método de la potencia con shift $\rho = 3.6$, donde q_{10} es el vector que se aporxima al eigenvector y σ_{10} el eigenvalor aproximado después de 10 iteraciones

```
In [2]: [q10, l10 ,iterations]=ev.inversePowerShift(A,q,3.6,1e-15)
```

Numero de Iteraciones: 12

$$\vec{q_{10}} = \begin{pmatrix} 0.5153107267324357 \\ -0.33238561122815846 \\ 0.7899206671324485 \end{pmatrix} \quad \sigma_{10} = 3.4207835356869145$$

El método requiere 12 iteraciones para cumplir con el criterio de error relativo igual a 10^{-15}, es decir

$$\frac{\|A\vec{q}_{10} - \sigma_{10}\vec{q}_{10}\|}{\|A\vec{q}_{10}\|} \leq 10^{-15}$$

Comparemos esto con los valores exactos, dados por:

```
In [3]: [L,V] = linear.eig(A)
```

$$\vec{v} = \begin{pmatrix} 0.515310726732435 \\ -0.3323856112281598 \\ 0.7899206671324484 \end{pmatrix} \quad \lambda = 3.420783535686916$$

En efecto, vemos que los valores tienen alrededor de 15 cifras correctas con respecto a los calculados por *scipy*

¿Por qué converge tan rapido el metodo? Veamos todos los eigenvalores de la matriz con el shift $\rho = 3.6$:

```
In [4]: [L,V] = linear.eig(A-3.6*np.identity(len(A)))
```

$$\vec{\lambda} = \begin{pmatrix} 4.754545 \\ -2.375328 \\ -0.179216 \end{pmatrix} \quad V = \begin{pmatrix} 0.083444 & -0.992728 & 0.515311 \\ 0.979576 & -0.104882 & -0.332386 \\ -0.182943 & -0.059078 & 0.789921 \end{pmatrix}$$

El método converge "rápido" porque $|\lambda_3 - \rho_3| = |\lambda_3 - 3.6| = 0.179216464313$ es cercano a 0 y la razón de convergencia teórica r es tal que:

$$r = \left| \frac{\lambda_3 - \rho_3}{\lambda_2 - \rho_2} \right| = 0.07544913221790368$$