Práctica 1 de Cálculo Numérico: Valores y vectores propios

Saúl Adrián Álvarez Tapia Carlos Eduardo Gil Mezta
Jorge Antonio Rotter Vallejo Sergio de Jesús Arnaud Gómez

Introducción

El problema de encontrar eigenvalores/eigenvectores (valores y vectores propios) de una matriz $A \in K^{n\times n}$ consiste en encontrar $\rho \in K$, $v \in K^n \setminus \{0\}$ tales que $Av = \rho v$. En este caso, se dice que (ρ, v) es una pareja eigenvalor/eigenvector de A. Toda matriz tiene a lo más n parejas de valores y vectores propios con sus vectores propios linealmente independientes, y en caso de que $K = \mathbb{C}$, toda matriz tiene al menos una pareja valor/vector propio.

El problema de encontrar valores y vectores propios es relevante ya que, al resolverlo, se puede comprimir información, hacer análisis de componentes principales, resolver ciertos sistemas de ecuaciones diferenciales y reducir la complejidad de varios problemas del álgebra lineal. Considerando que muchos problemas del mundo real se pueden simular usando una aproximación matricial, el encontrar los valores y vectores propios de una matriz es un problema muy importante en las matemáticas computacionales modernas.

Toda raíz del polinomio en λ dado por $\det(A - \lambda I)$ es un valor propio de A, con vector propio asociado cualquier vector no cero que resuelva la ecuación $(A - \lambda I)v = 0$. Sin embargo, este polinomio es de grado n, y por el teorema de Abel-Ruffini, no es resoluble analíticamente para $n \geq 5$. Además, obtener el polinomio característico de una matriz es un proceso computacional (al menos $O(n^3)$), y numéricamente inestable.

Por lo tanto, se vuelve necesario encontrar una forma de aproximar los valores y vectores propios de una manera numéricamente estable y con una complejidad aceptable. Para ello, estudiamos principalmente dos algoritmos: el método de la potencia y el algoritmo QR.

El método de la potencia funciona bajo el principio de que si hay un valor propio λ_1 dominante (es decir, que para cualquier otro valor propio μ de A, $|\lambda_1| > |\mu|$), entonces, para casi cualquier vector v, la secuencia

$$\frac{v}{||v||}, \frac{Av}{||Av||}, \ldots, \frac{A^k v}{||A^k v||}$$

converge a v_1, con $Av_1 = \lambda_1 v_1$. El método de la potencia consiste en generar esta secuencia hasta que se pueda decir que convergimos. El algoritmo converge linealmente con razón $\frac{|\lambda_2|}{|\lambda_1|}$, donde λ_2 es el valor propio de mayor magnitud después de λ_1.

Este proceso tiene algunos problemas. Por ejemplo, si no hay un valor propio dominante, la secuencia no convergerá. Si hay un valor propio dominante, pero no muy dominante (es decir, existe μ valor propio de A tal que $|\mu| \approx |\lambda_1|$), entonces el método de la potencia convergerá lentamente. Por otra parte, el método de la potencia sólo puede encontrar el valor propio de mayor magnitud, así que si queremos encontrar todas las parejas de valores y vectores propio, éste no es el camino.

Existen modificaciones que permiten encontrar valores propios distintos al de mayor magnitud. Estas se basan principalmente en que para toda matriz A invertible, si λ es valor propio de A, entonces $1/\lambda$ es valor propio de A^{-1}, y que para toda matriz, si λ es valor propio de A, entonces $\lambda - \mu$ es valor propio de $A - \mu I$. Usando estas propiedades, podemos encontrar más valores y vectores propios. Estas modificaciones generan el método de la potencia inversa con shift. Sin embargo, siguen teniendo convergencia lineal.

Usando el cociente de Rayleigh, se pueden buscar valores propios arbitrarios, con la ventaja de tener una convergencia cuadrática

En este laboratorio, implementamos el método de la potencia regular, el método de la potencia inversa con shift y el método de Rayleigh.

El algoritmo QR se basa en los mismos principios que el método de la potencia. Sin embargo, con este método podemos encontrar todos los valores y vectores propios de una matriz simultáneamente ya que el algoritmo QR usa el hecho de que los vectores propios de distintos valores propios son linealmente independientes.

El método QR consiste en que a partir de la matriz inicial A, obtener su factorización QR, donde Q es ortonormal y R triangular superior. Luego, se genera la matriz $RQ = Q * AQ$, que es similar a A. Por un teorema visto en clase, al repetir este proceso esta matriz converge a una matriz triangular superior, cuyos valores propios son las entradas en la diagonal.

La convergencia del algoritmo QR es lineal, y su complejidad no es muy buena. Sin embargo, haciendo unos pequeños trucos, como empezar usando la factorización de Hessenberg de A o usando shifts apropiados durante la QR, su convergencia se acelera radicalmente.

Por otra parte, el algoritmo QR es numéricamente muy estable, ya que sólo manipula matrices ortonormales.

El algoritmo QR es el método predilecto para los problemas actuales de valores y vectores propios.

Nosotros implementamos tanto el método QR clásico como el método QR dinámico.

Implementación

El repositorio `https://github.com/jarotter/calculo_numerico.git` contiene los seis ejercicios indicados en la práctica 1 de Cálculo Numérico en el ITAM (otoño 2017).

Para el proyecto utilizamos Julia (versión 0.6.0), con sólo dos librerías adicionales a la base: Distritbutions en el ejercicio 6 y MatrixDepot en el ejercicio 5. En ambos casos, la primera línea de cada ejercicio (que dejamos comentada) instala la librería adecuada.

Fuera del contenido estándar de la clase, queremos recalcar las siguientes particularidades de nuestro proyecto:

- El método de la potencia-Rayleigh tiene como condición de finalización alternativa a superar el número de iteraciones el que la matriz $A - \rho I$, donde ρ es el shift de Rayleigh actual, sea singular.
- Para los métodos QR, usamos factorización de Hessenberg.
- Para el método QR dinámico, usamos shift de Wilkinson.

Todas las funciones auxiliares utilizadas están en la carpeta lib, y están documentadas en Julia. Para acceder a su documentación, puede usarse ? func desde la terminal de Julia, donde func es el nombre de la función. Cada ejercicio tiene comentarios destinados a que los ejercicios sean autocontenidos, para evitar en la medida de lo posible cambiar entre el código y el trabajo escrito.

Resultados y discusión

Ejercicio 1

a) Después de haber aplicado el método de la potencia a la matriz

$$A = \begin{bmatrix} 1 & 1 & 2 \\ -1 & 9 & 3 \\ 0 & -1 & 3 \end{bmatrix}$$

con el vector inicial $\mathbf{q}_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$, se obtuvieron como valor y vector propios:

$$\lambda = 8.354544835511888 \qquad \text{y} \qquad \mathbf{v} = \begin{bmatrix} 0.08518345321780828 \\ 1.00 \\ -0.18675723715906348 \end{bmatrix}$$

b) El método de Julia para calcular valores y vectores propios dio como resultados:

$$\lambda_1 = 8.35454483516155 \qquad \text{y} \qquad \mathbf{v}_1 = \begin{bmatrix} 0.08344366245939693 \\ 0.9795759546471593 \\ -0.18294289893971968 \end{bmatrix}$$

c) Normalizando $\mathbf{v}_1$, obtenemos:

$$\frac{\mathbf{v}1}{||\mathbf{v}_1||\infty} = \begin{bmatrix} 0.08518345317025786 \\ 1.00 \\ -0.1867572372227279 \end{bmatrix}$$

Notemos que, salvo por un factor constante, el vector propio $\mathbf{v}$ calculado con el método de la potencia es igual al vector propio $\mathbf{v}_1$ calculado por Julia.

Ahora, veamos las razones

$$r_j = \frac{||\mathbf{q}j + 1 - \mathbf{v}||\infty}{||\mathbf{q}j - \mathbf{v}||\infty}$$

$r_1 = 0.3723858933535865$ $r_8 = 0.4093844091160062$ $r_{15} = 0.4094196574874508$
$r_2 = 0.3947898421559911$ $r_9 = 0.4094240984438118$ $r_{16} = 0.4093735364467985$
$r_3 = 0.40353317920115317$ $r_{10} = 0.4094401618698333$ $r_{17} = 0.4092607095659459$
$r_4 = 0.40704312628481176$ $r_{11} = 0.40944629465903026$ $r_{18} = 0.4089849081976997$
$r_5 = 0.4084682723923035$ $r_{12} = 0.4094477245034639$ $r_{19} = 0.4083102111249005$
$r_6 = 0.4090496328317853$ $r_{13} = 0.4094456706115635$ $r_{20} = 0.40665590279039$
$r_7 = 0.40928724073553063$ $r_{14} = 0.4094383839364173$

d) La razón teórica es:

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

y vemos que las razones en los incisos c) y d) son esencialmente iguales.

Ejercicio 2

Método de la potencia inversa con shift $\rho = 0$ aplicado a A con vector inicial $\mathbf{q}_0$.

a) Se obtuvieron como valor y vector propios:

$$\lambda = 1.224671629205953 \qquad \text{y} \qquad \mathbf{v} = \begin{bmatrix} 1.00 \\ 0.10565065554749963 \\ 0.05951048682922665 \end{bmatrix}$$

b) El método de Julia para calcular valores y vectores propios dio como resultados:

$$\lambda_1 = 1.2246716291515263 \qquad \text{y} \qquad \mathbf{v}_1 = \begin{bmatrix} -0.9927283126924615 \\ -0.10488239703499058 \\ -0.059077745141235156 \end{bmatrix}$$

c) Normalizando $\mathbf{v}_1$, obtenemos:

$$\frac{\mathbf{v}1}{||\mathbf{v}_1||\infty} = \begin{bmatrix} -1.00 \\ -0.10565065556610374 \\ -0.05951048679271115 \end{bmatrix}$$

Notemos que, salvo por un factor constante, el vector propio $\mathbf{v}$ calculado con el método de la potencia es igual al vector propio $\mathbf{v}_1$ calculado por Julia.

Ahora, veamos las razones

$$r_j = \frac{||\mathbf{q}j + 1 - \mathbf{v}||\infty}{||\mathbf{q}j - \mathbf{v}||\infty}$$

$r_1 = 1.0$
$r_2 = 0.8542917968590314$
$r_3 = 0.565767866528141$
$r_4 = 0.4496157392942056$
$r_5 = 0.3938455853242537$
$r_6 = 0.37129726345621406$
$r_7 = 0.3628356745142426$
$r_8 = 0.359749421095197$
$r_9 = 0.3586351319070328$
$r_{10} = 0.3582341677250352$
$r_{11} = 0.3580899932546206$
$r_{12} = 0.3580380882567092$
$r_{13} = 0.35801921682548565$
$r_{14} = 0.3580118561159328$
$r_{15} = 0.35800761393024155$
$r_{16} = 0.35800163976493404$
$r_{17} = 0.35798706917444434$
$r_{18} = 0.3579471303382953$
$r_{19} = 0.3578358216405784$
$r_{20} = 0.3575248065699232$

d) La razón teórica es:

$$\left|\frac{\lambda_n}{\lambda_{n-1}}\right| = 0.35800909831776423$$

y vemos que las razones en los incisos c) y d) son esencialmente iguales.

Método de la potencia inversa con shift $\rho = 3.5$ aplicado a A con vector inicial $\mathbf{q}_0$.

a) Se obtuvieron como valor y vector propios:

$$\lambda = 3.4207835356870624 \qquad \text{y} \qquad \mathbf{v} = \begin{bmatrix} 0.6523575697850283 \\ -0.4207835356870626 \\ 1.00 \end{bmatrix}$$

b) El método de Julia para calcular valores y vectores propios dio como resultados:

$$\lambda_1 = 3.4207835356869154 \qquad \text{y} \qquad \mathbf{v}_1 = \begin{bmatrix} 0.5153107267324352 \\ -0.3323856112281586 \\ 0.7899206671324489 \end{bmatrix}$$

c) Normalizando $\mathbf{v}_1$, obtenemos:

$$\frac{\mathbf{v}1}{||\mathbf{v}_1||\infty} = \begin{bmatrix} 0.6523575697836895 \\ -0.42078353568691507 \\ 1.00 \end{bmatrix}$$

Notemos que, salvo por un factor constante, el vector propio $\mathbf{v}$ calculado con el método de la potencia es igual al vector propio $\mathbf{v}_1$ calculado por Julia.

Ahora, veamos las razones

$$r_j = \frac{||\mathbf{q}j + 1 - \mathbf{v}||\infty}{||\mathbf{q}j - \mathbf{v}||\infty}$$

$r_1 = 0.016689323935299996$ $r_4 = 0.013498249340114093$ $r_7 = \text{Inf}$
$r_2 = 0.01567021398656163$ $r_5 = 0.039384710051032855$
$r_3 = 0.017755141369851894$ $r_6 = 0.0$

d) La norma teórica es:

$$\left|\frac{\lambda_n}{\lambda_{n-1}}\right| = 0.03481539865981832$$

Aquí recalcamos que por la convergencia rápida del algoritmo, en la séptima iteración ya no hay diferencia (a 16 cifras) entre ambos vectores. De aquí el cero y luego el valor Inf (división entre cero). No incluímos las razones siguientes porque ya no dan información útil.

Ejercicio 3
Al aplicarle el método de la potencia con shift del cociente de Rayleigh a la matriz A y con distintos vectores iniciales se obtuvo lo siguiente:

$$D = P^{-1}AP \qquad \text{y} \qquad \text{con}$$

$$D = \begin{bmatrix} 3.420784 & 0 & 0 \\ 0 & 8.354545 & 0 \\ 0 & 0 & 1.224672 \end{bmatrix} \qquad \text{y} \qquad P = \begin{bmatrix} 0.652358 & 0.0851835 & 1.0 \\ -0.420784 & 1.0 & 0.105651 \\ 1.0 & -0.186757 & 0.0595105 \end{bmatrix}$$

Donde los elementos de la diagonal de D son las aproximaciones a los valores propios de A y las columnas de P son los vectores propios linealmente independientes de A.

Para verificar la convergencia cuadrática del primer vector propio, veamos las razones calculadas como en los incisos c) del ejercicio 1.

$r_1 = 0.08635778444044981$ $\quad r_4 = 0.6746997204377666$ $\quad r_7 = \text{NaN}$
$r_2 = 1.584419192227463$ $\quad r_5 = 0.684807202146323$
$r_3 = 0.49289362960482147$ $\quad r_6 = 0.0$

Observemos que desde la sexta razón tenemos cero. Es decir, el algoritmo ya logró convergencia hacia un valor propio. Aunque la razón teórica sea

$$\left| \frac{\lambda_n}{\lambda_{n-1}} \right| = 0.4094518137349568$$

la convergencia es tan rápida que no alcanza estabilizarse en la lista de arriba antes de detenerse. Aquí queremos aclarar que los resultados varían entre ejecución y ejecución porque el vector inicial es aleatorio.

Ejercicio 4
El método de la potencia que implementamos no funciona con la matriz

$$A = \begin{bmatrix} 1 & -4 & -6 \\ -12 & -8 & -6 \\ 11 & 10 & 10 \end{bmatrix}$$

porque, como se muestra a continuación, no tiene un valor propio dominante, dado que el de mayor magnitud es complejo y, por tanto, viene con su conjugado, que tiene la misma magnitud. Los siguientes valores propios se calcularon con el método de Julia para aproximar valores propios.

$$\begin{bmatrix} -2.76311 & 0 & 0 \\ 0 & 2.88155 - 2.76057i & 0 \\ 0 & 0 & 2.88155 + 2.76057i \end{bmatrix}$$

Ejercicio 5

A continuación se comparan los métodos QR simple y QR con shifts dinámicos implementados por nosotros con el método de Julia para calcular valores propios. Los vectores $\mathbf{v}$ tienen como entradas a los valores propios de A y n es el número de iteraciones:

El método QR simple dio como resultados:

$$\mathbf{v} = \begin{bmatrix} 216.73194654281997 \\ -126.81827789682255 \\ -40.0301574197931 \\ -14.2402443490463 \\ -8.499822670072632 \\ -5.236067977499783 \\ -3.766822029130399 \\ -2.7580423738097717 \\ -2.177499542899788 \\ -1.7414906590978931 \\ -1.4560662464312706 \\ -1.2305894732043927 \\ -1.0708938888510213 \\ -0.9409191949161145 \\ -0.8440426859973454 \\ -0.7639320225002095 \\ -0.702278178498368 \\ -0.6511150888037952 \\ -0.6111494859682925 \\ -0.5783794940752632 \\ -0.552973671851485 \\ -0.5329619142946431 \\ -0.518255221991412 \\ -0.507979555928666 \\ -0.5019855013356324 \end{bmatrix}$$

$n = 1000$

El método QR con shifts dinámicos calculó:

$$\mathbf{v} = \begin{bmatrix} 216.73194654282025 \\ -126.81827789682264 \\ -40.03015741979313 \\ -14.240244349046273 \\ -8.499822670072643 \\ -5.236067977499788 \\ -3.766822029130398 \\ -2.7580423738097704 \\ -2.177499542899792 \\ -1.7414906590978905 \\ -1.456066246431275 \\ -1.2305894732043925 \\ -1.07089388885102 \\ -0.9409191949161126 \\ -0.8440426859973459 \\ -0.7639320225002109 \\ -0.7022781784983658 \\ -0.6511150888037956 \\ -0.6111494859682942 \\ -0.5783794940752622 \\ -0.5529736718514865 \\ -0.5329619142946429 \\ -0.518255221991411 \\ -0.50198550133501 \\ -0.5079795559292898 \end{bmatrix}$$

$n = 30$

El método de Julia dio como resultado:

$$\mathbf{v} = \begin{bmatrix} 216.7319465428201 \\ -126.8182778968222 \\ -40.03015741979337 \\ -14.240244349046577 \\ -8.499822670072746 \\ -5.236067977499999 \\ -3.7668220291304237 \\ -2.758042373809843 \\ -2.1774995428995965 \\ -1.741490659097849 \\ -1.4560662464313339 \\ -1.230589473204317 \\ -1.0708938888509496 \\ -0.9409191949160169 \\ -0.844042685997301 \\ -0.7639320225001289 \\ -0.702278178498337 \\ -0.6511150888037579 \\ -0.611149485968286 \\ -0.5783794940751732 \\ -0.5529736718513584 \\ -0.5329619142945015 \\ -0.5182552219913622 \\ -0.507979555929172 \\ -0.5019855013348997 \end{bmatrix}$$

Ejercicio 6

Con las aristas unidireccionales, el ranking de Google (interpretado en norma 1 para ser probabilidades) converge a 0.3. Añadiendo más aristas (haciéndolas bidireccionales) no estamos haciendo cambios significativos cuando n se va a infinito, pues de cualquier manera el nodo cero tiene más conexiones que todos los demás y es más probable acabar en él. La matriz de Google es la representación de una cadena de Markov homogénea, donde la entrada (i, j) puede interpretarse como la probabilidad de pasar de la página i a la j a través de un hipervínculo. En su construcción se compensa el hecho de que uno puede cambiar a cualquier otra página que no esté vinculada en la actual. Aunque el código sólo muestra n = 100, repetimos el proceso para n = 3, n = 10, n = 100 y n = 1000 (1000 tarda bastante tiempo) y en todos los casos se da la convergencia.