DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Análisis de Redes Sociales

Métodos Numéricos

Autores

Integrante	LU	Correo electrónico
Battolla, Gianfranco	17/20	gianfrancobtl@gmail.com
Orfanos, Santiago	51/21	s.orfanos@hotmail.com
Vidal, Julian	681/21	juli.vidal14@gmail.com

Resumen

Análisis de la aplicación del método de la potencia con deflación para identificar autovalores y autovectores; aplicación de los mismos para la identificación de propiedades en los casos del Club de Karate y la matriz de atributos de Ego-Facebook.

Palabras clave

power iteration, eigenvalues, eigenvectors

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Introducción **3**

2. Desarrollo **6**
2.1. Los algoritmos 6
2.2. Los parámetros involucrados 6
2.3. Implementación y salida 6

3. Resultados y discusión **7**
3.1. Archivos de test utilizados 7
3.2. Análisis del algoritmo 7
3.3. Club de Karate 7
3.3.1. Centralidad del vector 7
3.3.2. Predicción de la separación de grupos 8
3.4. Ego-Facebook 9
3.4.1. PCA - análisis de los k componentes principales 10

4. Conclusiones **12**

1. Introducción

En este trabajo vamos a usar algunas técnicas de descomposición de matrices para analizar redes. Trabajaremos con dos casos de redes sociales representadas con matrices, las cuales descompondremos en autovalores y autovectores para poder caracterizarlas y analizar su estructura.

Un grafo finito se puede representar como una matriz, donde el elemento ij es 1 si el nodo i esta conectado al nodo j y 0 si no. Por este motivo se le llama matriz de adyacencia. En un grafo donde no se distingue el orden de la conexión de los nodos, la matriz resulta simétrica ya que el elemento ij siempre es igual a ji.

Se denominan autovalores y autovectores de un grafo a los autovalores y autovectores de su matriz de adyacencia. Recordemos que un autovector es un vector no nulo tal que al multiplicarlo por una matriz A cuadrada se obtiene:

$$Av = \lambda v$$

donde λ es el autovalor asociado a v, un escalar que sea solución no trivial del sistema. Si se piensa geométricamente, A está estirando o encogiendo a v.

La matriz de adyacencia nos permite obtener una medida de centralidad (ordenamiento de los nodos según importancia o influencia) bastante básica, donde se considera la cantidad de conexiones de cada nodo. Si consideramos además la importancia de los nodos a los que un nodo esta conectado podemos escribir la siguiente ecuación:

$$x_i = \frac{1}{\lambda} \sum_{j=1}^{n} A_{ij} x_j$$

Esta expresión es equivalente a la ecuación $Ax = \lambda x$, por eso se conoce como centralidad de autovector. Por el Teorema de Perron-Frobenius, si pedimos que la medida de centralidad tome valores sólo positivos, entonces λ es el mayor autovalor de la matriz de adyacencia.

A partir de la red, también se puede construir la matriz Laplaciana que es de gran utilidad para el análisis de la estructura de la red:

$$L = D - A$$

Aquí, D es una matriz diagonal cuyos valores indican la cantidad de conexiones que tiene cada nodo, expresada así:

$$D_{ii} = \sum_{j=1}^{n} A_{ij}$$

Al descomponer en autovalores y autovectores la matriz Laplaciana, obtenemos más información sobre la red:

$$Lv = \lambda v$$

Para redes no dirigidas, L es simétrica y semidefinida positiva. Por propiedad, sus autovalores son todos mayores o iguales que 0. El autovalor más pequeño distinto de cero es llamado conectividad algebraica. Este

autovalor y su correspondiente autovector resultarán de interés debido a que con ellos podremos establecer un criterio aproximado para poder cortar nuestra red en dos, resultando mínima la cantidad de aristas cortadas.

Vamos a estar aplicando estos conocimientos en la primera de nuestras redes. Se trata de un Club de Karate, donde los 34 nodos representan sus miembros y las aristas, sus interacciones fuera del club. Mientras se estudiaban estos datos, el instructor (nodo 1) y el administrador (nodo 34) se pelearon y separaron el club en dos. Analizaremos la centralidad de los miembros e intentaremos predecir qué bando tomó cada uno de acuerdo a la estructura de la red.

La segunda red que analizaremos es una red-ego de Facebook. Esta consiste en nodos representando las amistades de un usuario (ego), que no esta representado, y aristas que muestran las amistades entre los nodos. En el dataset también tenemos una matriz de atributos donde hay un dato para cada usuario y una serie de atributos relacionados a categorías como trabajo, educación, ciudad natal, etc. Veremos si es posible predecir las amistades según los atributos compartidos. Para esto, necesitamos plantear algunos conceptos más.

Dados x e y, dos vectores, vamos a definir la covarianza como:

$$Cov(x, y) = \frac{(x - \mu_x) * (y - \mu_y)}{n - 1}$$

donde μ es el valor medio del vector. Podemos interpretar esto como el producto interno de los vectores "centrados" y normalizada por la dimensión del vector menos uno.

Después, vamos a definir la correlación como:

$$Corr(x, y) = \frac{(x - \mu_x) * (y - \mu_y)}{\sqrt{(x - \mu_x) * (x - \mu_x) * (y - \mu_y) * (y - \mu_y)}}$$

que es una covarianza normalizada para que el rango sea entre -1 y 1. Esto resulta equivalente al coseno del ángulo de los vectores $cos\theta_{xy}$. Los vectores son paralelos si hay alta correlación, positiva para vectores en la misma dirección y negativa en direcciones opuesta y perpendiculares si la $Corr$ es nula.

Si consideramos los vectores $x_i - \mu_{x_i}$ (como columnas) de la matriz X, obtenemos la matriz de covariancia, que es simétrica y semidefinida positiva:

$$C = \frac{X^t X}{n - 1}$$

Por lo tanto, vale la propiedad como en la matriz Laplaciana de que sus autovalores son no negativos.

Consideremos ahora, una matriz de datos $X \in R^{mxn}$. PCA o Analisis de Componentes Principales busca hallar un cambio de base tal que las dimensiones se ordenen en componentes que expliquen los datos de menor a mayor. Análogamente, los ordena según su varianza de mayor a menor. Descomponemos la matriz de covarianza:

$$C = VDV^t$$

V tiene como columnas a los autovectores y D es una matriz diagonal con los autovalores, que representan la varianza capturada por cada nueva dirección dada por los autovectores. Podemos transformar los datos X con $V^t X$. Se puede reducir la dimensionalidad de los datos usando sólo los primeros k vectores columna de la matriz V.

Dado un conjunto de datos $X \in R^{mxm}$ con m datos y n atributos, definimos una matriz de similaridad como una matriz de R^{mxm} que computa una función sobre cada par de datos ij, por ejemplo, el producto interno:

$$D_{ij} = f(X_i,X_j)$$

$$D = XX^t$$

Con todo esto, ya podemos continuar con el desarrollo del trabajo.

2. Desarrollo

2.1. Los algoritmos

A continuación, expresamos ambos con un pseudocódigo que permita una primera aproximación a los mismos.

Algorithm 1 Método de la potencia(matrix A, int niter, double eps)

$b \leftarrow vector\_random(A.cols)$

$i \leftarrow 0$
while $i < niter$ **do**
  $old \leftarrow b$
  $b \leftarrow normalizar(A@b)$
  $cosAngle \leftarrow dot\_product(b, old)$
  $If\ (1 - eps) < cos\_angle <= 1\ then\ break$
  $i \leftarrow i + 1$
end while
$eigenvalue \leftarrow dot\_product(b, A@b)$

return eigenvalue, b

Algorithm 2 Eigen(matrix A, int niter, double eps)

$A \leftarrow copia(A)$
$eigenvalues \leftarrow []$
$eigenvectors \leftarrow matriz\_de\_ceros\_dimensiones\_de\_A$

$i \leftarrow 0$
while $i \leqslant filas(A)$ **do**
  $l, v \leftarrow power\_iteration(A, niter, eps)$
  $eigenvalues.push\_back(l)$
  $eigenvectors[:, i] \leftarrow v$
  $A \leftarrow A - l * v * v.T$
  $i \leftarrow i + 1$
end while

return eigenvalues, eigenvectors

2.2. Los parámetros involucrados

El método de la potencia con deflación recibe tres parámetros: en primer lugar, una matriz de adyacencia con unos y ceros que indica si dos nodos están relacionados o no. En segundo lugar, recibimos un entero positivo llamado *niter*, el cual servirá como un máximo de iteraciones para el método de la potencia a la hora de encontrar cada autovector y autovalor. Finalmente, tenemos un *epsilon*, el cual será nuestro criterio de parada; cuando el vector encontrado forme un ángulo de epsilon grados con el vector encontrado previamente, esto significará que no habrá cambios significativos para la siguiente iteración del método. Luego no será necesario realizarla y podremos quedarnos con este resultado.

2.3. Implementación y salida

Por un lado, para implementar los algoritmos mostrados en pseudocódigo hicimos uso de la librería Eigen de C++ junto con todas las operaciones vectoriales y matriciales que ofrece.

Por otro lado, el algoritmo realizado tiene dos archivos de texto de salida para cada ejecución que se realice del mismo, los cuales son: 1) los autovalores de la matriz de adyacencia dada como parámetro; 2) los autovectores correspondientes ordenados por columna.

3. Resultados y discusión

Para chequear la precisión de los resultados de nuestro algoritmo, realizamos variados tests, los cuales serán expuestos en esta sección.

3.1. Archivos de test utilizados

Para realizar las pruebas del algoritmo, utilizamos los siguientes archivos .txt:

1. ego-facebook.feat: un archivo con 792 usuarios de la red de facebook identificados con un número aleatorio y anónimo. Cada usuario tiene 320 atributos relacionados.
2. karateclub_matriz.txt: un archivo que contiene una matriz de adyacencia que representa a 34 personas del club de karate y muestra para cada coordenada ij si i es amigo de j.

3.2. Análisis del algoritmo

Para chequear que el cálculo de autovalores y autovectores de nuestro algoritmo era correcto, realizamos pruebas con matrices cuyos autovalores y autovectores eran conocidos por nosotros.

En este proceso, para asegurarnos de la correctitud, comparamos con dos métodos: por un lado, lo hicimos con el método de la potencia programado en el taller de la cátedra. Notamos que los autovalores y autovectores eran los mismos y estaban en orden. Por otro lado, comparamos con los resultados que obtuvimos con el método facilitado por numpy. En este caso, notamos que los autovalores y autovectores salían en otro orden. Pero, si los ordenamos de mayor a menor (como salen en nuestro algoritmo), el resultado es correcto.

Realizamos tests para una matriz de Householder, una matriz diagonal, una matriz simétrica y otra de la forma $A * A^t$. Por ejemplo, veamos qué sucede para el caso de la siguiente matriz simétrica:

$$\begin{bmatrix} 7 & 2 & -3 \\ 2 & 2 & -3 \\ -3 & -2 & -2 \end{bmatrix} \tag{3.1}$$

Al calcular los autovalores con nuestro método, obtuvimos los siguientes: 8.83835; -3.2849522; 1.446602. Los autovectores respectivos resultaron: [0.88455707 0.22057717 -0.41097993]; [0.34915668 0.27109465 0.89699346]; [-0.30927074 0.9369383 -0.16278271].

Para ver si el resultado era correcto, multiplicamos la matriz por el primer autovector y el primer autovalor por el primer autovector y nos fijamos que los resultados se acercasen a, al menos, una diferencia de 1e-6. Esto fue así. Luego probamos con el segundo y tercer autovalor y autovector y los resultados fueron nuevamente satisfactorios. Con las otras matrices seguimos pasos similares y llegamos, también, a lo que esperábamos.

En la experimentación, surgieron algunos puntos que nos gustaría resaltar: por ejemplo, hubo casos en los que los resultados obtenidos distaban demasiado de los esperados. A raíz de esto, notamos que estábamos trabajando con una cantidad de iteraciones baja o un epsilon (cota del error) demasiado elevado).

Por esta razón, terminamos eligiendo un epsilon genérico de 1e-20, el cual nos permitió obtener resultados satisfactorios para cada una de las matrices utilizadas. Además, a la hora de evaluar los resultados, tomamos por correcto todo aquel que nos diese a lo sumo una diferencia de 1e-6 con el esperado. Para esto, utilizamos la función de *numpy* llamada AllClose y le pasamos este número como segundo parámetro.

3.3. Club de Karate

Para analizar la matriz de adyacencia (a partir de ahora, A) del club de karate, realizamos diferentes experimentos para encontrar la centralidad de autovector y encontrar los nodos más importantes. Luego, tratamos de visualizar si la matriz laplaciana basada en A nos permite encontrar cómo se espera que los grupos queden divididos tras la pelea ocurrida.

3.3.1. Centralidad del vector

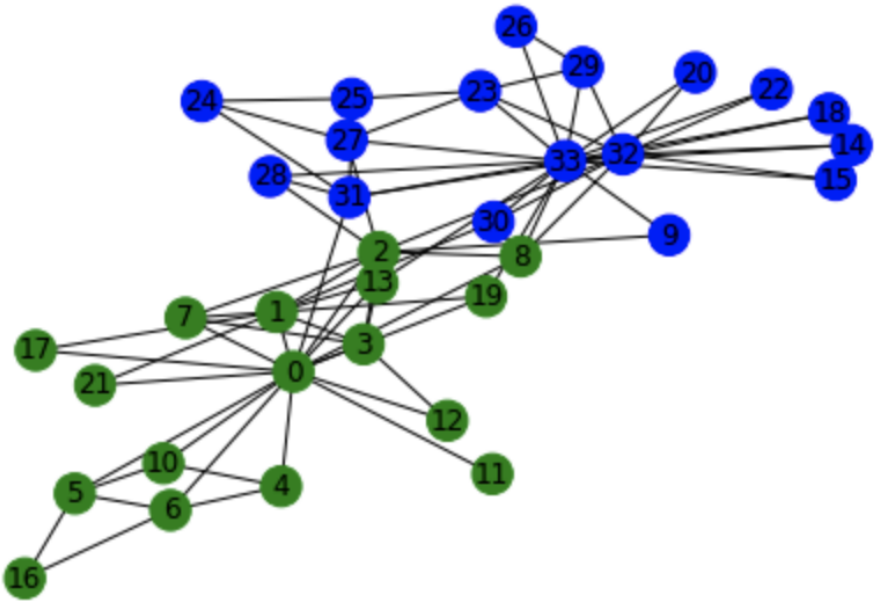
Calculamos la centralidad del vector para encontrar cuáles son los nodos (o las personas, en este caso) que tienen más importancia dentro del club.

Para esto, calculamos los autovalores y autovectores de la matriz A con el método de la potencia con deflación implementado previamente en C++. Luego, tomamos el mayor autovalor y tomamos el autovector correspondiente al mismo. Finalmente, normalizamos el autovector obtenido. Los resultados fueron los siguientes:

Por un lado el vector normalizado (para que la suma de 1) obtenido fue el que puede verse en la Figura 1 (izq.), donde la columna en negrita representa la persona y la otra su centralidad normalizada.

0	0.071393	**17**	0.018573
1	0.053421	**18**	0.020376
2	0.063713	**19**	0.029730
3	0.042430	**20**	0.020376
4	0.015269	**21**	0.018573
5	0.015976	**22**	0.020376
6	0.015976	**23**	0.030154
7	0.034356	**24**	0.011451
8	0.045691	**25**	0.011882
9	0.020630	**26**	0.015182
10	0.015269	**27**	0.026813
11	0.010626	**28**	0.026329
12	0.016935	**29**	0.027108
13	0.045513	**30**	0.035110
14	0.020376	**31**	0.038388
15	0.020376	**32**	0.061953
16	0.004748	**33**	0.074930

(a) Centralidad normalizada para cada nodo del club de karate.



(b) Grafo obtenido a partir de la matriz de adyacencia de la matriz de karate.

Figura 1: Grafos de rankings que dan paridad de puntaje a todos los nodos

Por otro lado, el grafo del club obtenido es el que puede verse en la Figura 1 (b). Como se aprecia, y en consonancia con el ranking de la Figura 1 (a), los nodos 0 y 33 son los que más conexiones presentan. A raíz de esto, podemos inferir que tenían gran relevancia dentro del grupo y tuvieron mucha influencia para que los grupos se dividieran de la manera que finalmente se dividieron.

3.3.2. Predicción de la separación de grupos

A partir de la matriz A puede obtenerse la matriz Laplaciana L, la cual ayuda a identificar cómo se separarían los nodos en caso de haber una pelea.

Para esto: en primer lugar, calculamos la matriz L = D - A, donde D es una matriz diagonal cuyos valores indican la cantidad de conexiones que tiene cada nodo. Para ahorrar tiempo, calculamos L con la función laplacian_matrix que provee el módulo de *networkx* partiendo de la matriz de adyacencia.

Luego, calculamos con nuestro algoritmo los autovalores y autovectores. Después, calculamos la conectividad algebraica. Para esto, tomamos el autovalor más chico mayor que cero y su respectivo autovector (llamémoslo avec_l). Comprobamos que avec_l presenta la mayor correlación con el vector de amistades original pues da un valor absoluto de 0.576.

Geométricamente, que el valor absoluto de la correlación sea 1 significa que los vectores son paralelos. Si bien en este caso el resultado no es tan cercano a 1 podemos afirmar que el autovector elegido es el "más paralelo" al de las conexiones reales.

Finalmente, comparamos componente a componente avec_l con el vector de amistades original. Esto nos permitió vislumbrar que el autovector correspondiente a la conectividad algebraica permite predecir correctamente a qué grupo se irán 20 nodos pero no lo hace con 14.

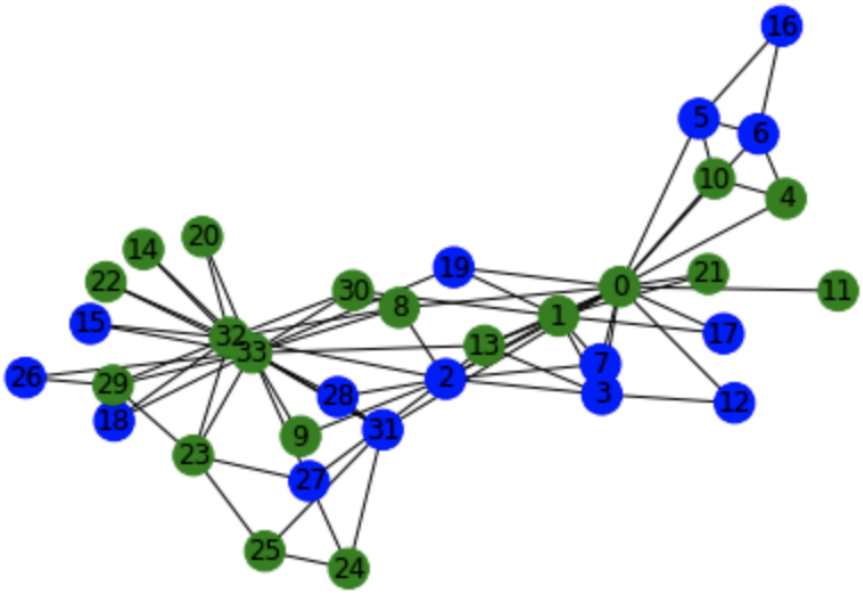


Figura 2: Grafo obtenido a través de la predicción.

3.4. Ego-Facebook

Tenemos la matriz de atributos X. Con la misma, primero computamos la matriz de similaridad D, realizando el producto interno entre X y X traspuesta. De esta manera, obtuvimos una matriz simétrica y semidefinida positiva con todos sus autovalores no negativos y dimensión n = 792. Es decir, nuestra nueva matriz tiene tantas filas y columnas como cantidad de usuarios.

Con esta matriz, es posible construir una matriz de adyacencia que intente aproximar la matriz de amistades original. Para esto, es preciso elegir un umbral u acorde. Tras probar con varios umbrales, nos encontramos con resultados interesantes. Al elegir u = 3, la cantidad de conexiones fue de 176123. ¿Por qué? Esto se debe a que, al elegir un menor umbral, estamos pidiendo que no haya tanto en común entre los usuarios para que estén relacionados. Luego, con este umbral muchas más personas resultan conectadas. Algo similar ocurre al elegir u = 4; en este caso hay 87324 conexiones.

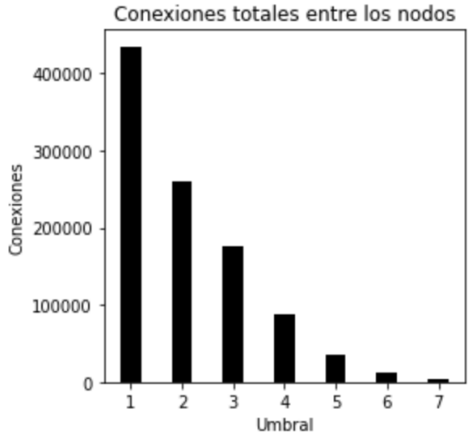


Figura 3: Variación de las conexiones según el umbral elegido.

Por otro lado, al probar con umbrales más grandes ocurrió lo contrario: al pedir que haya más características en común entre los usuarios, menos de ellos resultaron estar conectados. Por, ejemplo, al elegir un umbral de 6, obtenemos 11980 conexiones, y al elegir uno de 7, tenemos 3471.

Tras analizarlo, decidimos elegir un umbral que, al realizar la ejecución, nos diera una cantidad de de amistades similar a las amistades originales, que eran alrededor de 28000. Con esta premisa, llegamos a la conclusión que un umbral de 5, que da alrededor de 34369 conexiones, o sea 17184 amistades, es el más indicado para comparar los atributos con la matriz original.

Pero, ¿alcanza con pedir esto? No. ¿Qué pasaría si, por ejemplo, la cantidad de amistades son las mismas pero se producen entre nodos totalmente diferentes?

Para poder comparar las amistades originales con las obtenidas mediante la matriz de similar, seguimos los siguientes pasos:

Primero, creamos la matriz de adyacencia basándonos en las amistades entre los nodos (ego-Facebook.edges). Como vimos que había nodos que figuraban en este archivo pero no en la matriz de atributos, borramos todas las conexiones que incluían a nodos con ID menor a 2661. Llamemos a esta matriz B, la cual tiene 27652 conexiones totales (o 13826 relaciones de amistad).

Posteriormente, comparamos la correlación entre B y D luego de haberles aplicado la función de numpy para aplanarlas (flatten). Sin embargo, notamos que medir esta correlación era engañoso. Por ejemplo, si el umbral fuese 100, no habría conexiones entre los nodos. Sin embargo, y como hay un porcentaje pequeño porcentaje de nodos conectados - un 0.0045 por ciento-, la correlación será alta puesto que coincidirán muchos valores 0 de la matriz D con la B.

Luego, calculamos la correlación entre los autovectores de la matriz D y los de la matriz B para intentar ver qué umbral era el apropiado. En este sentido, identificamos que la correlación más alta, de 0.63, se alcanza con un umbral de u=7 (donde $D >= u$).

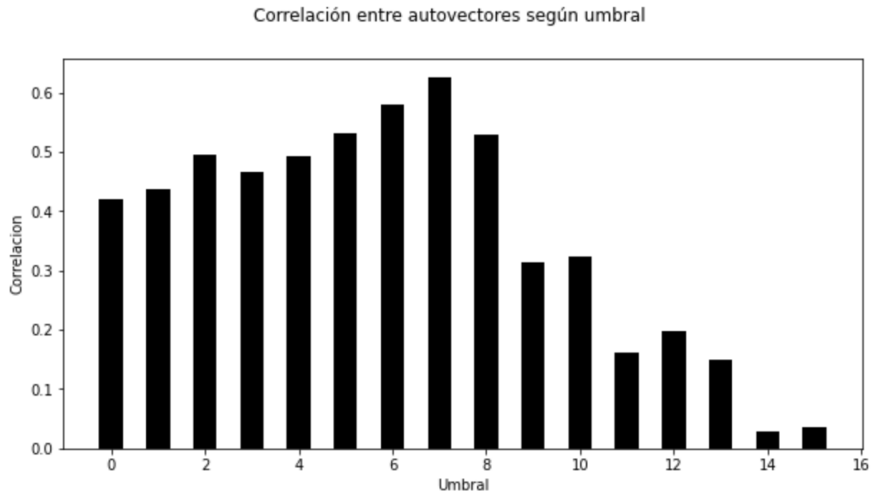


Figura 4: Variación de la correlación entre autovectores según el umbral elegido.

¿Qué pasa con otros umbrales? Como podemos ver en la Figura 4, la correlación entre los autovectores mantiene un sostenido ascenso hasta u = 7, luego del cual comienza a bajar hasta transformarse en, prácticamente, 0.

De esta manera, dejamos de lado nuestra hipótesis previa de que el mejor umbral era 5 y la reemplazamos por la idea de que u = 7 es el mejor umbral para que la matriz de amigos de Facebook original se parezca a la modelada con la matriz de atributos.

3.4.1. PCA - análisis de los k componentes principales

Generalmente, en una matriz de gran cantidad de atributos, sucede que hay unos pocos que, solos, explican gran parte de las conexiones que luego suceden entre los nodos.

Entonces, hicimos lo siguiente: 1) calculamos la matriz de covarianza, la traspasamos a un archivo .txt y calculamos sus autovectores y autovalores con nuestro algoritmo. Así, nos construimos una matriz V con los autovalores de la matriz de covarianza (M_x) en las columnas. De esta manera, pudimos diagonalizar M_x y representarla como VM_xV^T.

En segundo lugar, vimos cuántas componentes son las más importantes dentro de los datos que teníamos. De esta manera, apreciamos que si tomamos 67 atributos, lograremos explicar el 90 por ciento de la información

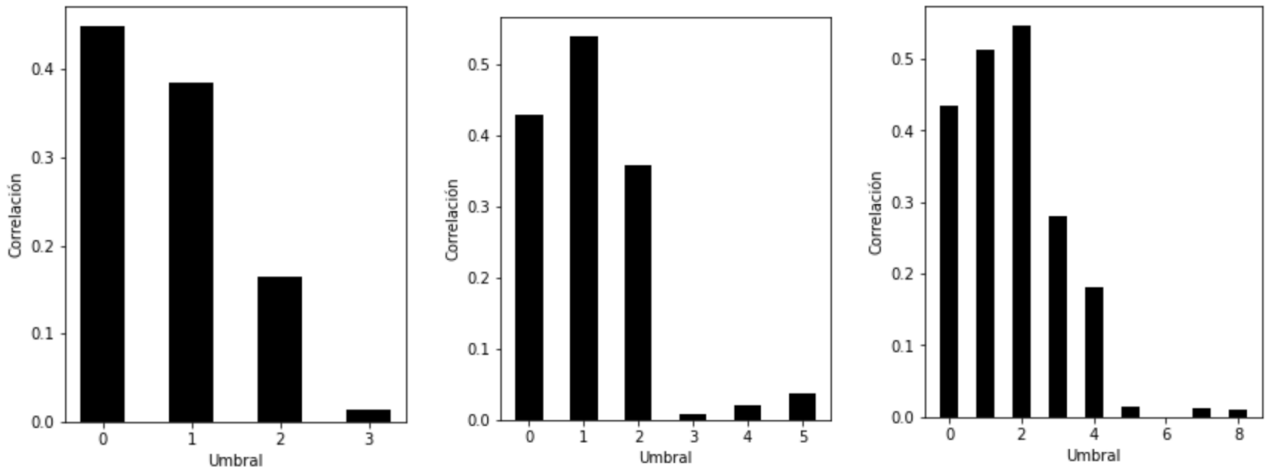
brindada por la totalidad de los mismos; si tomamos 37, el 80 por ciento; si tomamos 18, el 70 por ciento; si tomamos 9, el 60 por ciento; y, si tomamos 4, el 50 por ciento. Vemos que, llegado un punto, las componentes que agregamos no aportan a la fiabilidad del resultado.

Veamos ahora qué pasa con el umbral u elegido anteriormente si limitamos nuestra matriz de similaridad a k componentes principales. Para hacer esto, nos construimos una nueva matriz llamémosla X_{pca}, que será el producto punto entre X y los primeros k autovectores que elijamos. Con esta, nos construimos otra matriz (llamémosla D') la cual será el producto de X_{pca} y X_{pca}^T.

A continuación, se muestran los resultados a los cuales arribamos:

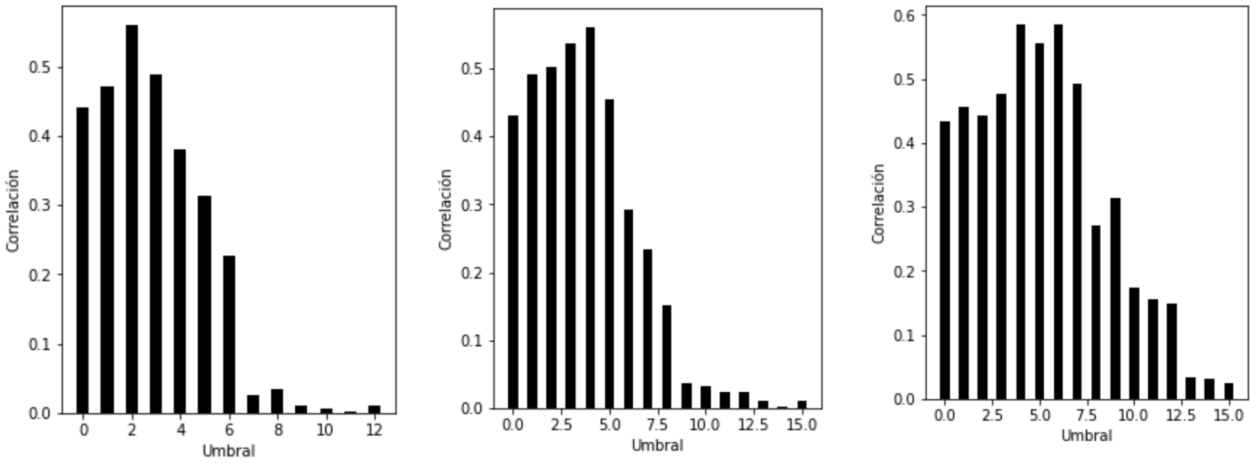
1) Para k = 4 (explica el 50 por ciento de los datos), k = 9 (60) y k = 18 (70) solo obtenemos datos razonables de la covarianza para u = 1. La covarianza entre la matriz de similaridad obtenida y la matriz original fue de un promedio de 0.5.

2) Veamos que pasa para k = 37 (80 por ciento) en adelante: como puede apreciarse en los gráficos de las figuras 5 y 6, a medida que aumenta el k, el umbral con mayor correlación. Esto sucede debido a lo siguiente: si hay pocas componentes, luego no podemos pedir que los nodos tengan tantas en común. Por eso vemos que para un k = 37, u = 1 es óptimo. Sin embargo, con k = 67 precisamos u = 2, con k = 100 precisamos u = 3, etcétera, hasta que llegamos a 300 componentes (un caso muy parecido al original) en el cual precisamos un u = 6. Podemos apreciar cómo la correlación entre la matriz original y la nuestra de similaridad también crece a medida que aumentamos el k.



(a) k = 37 **(b)** k = 67 **(c)** k = 100

Figura 5: Evolución de la correlación según los umbrales.



(a) k = 150 **(b)** k = 200 **(c)** k = 300

Figura 6: Evolución de la correlación según los umbrales.

4. Conclusiones

Durante el desarrollo de este trabajo práctico, trabajamos sobre cómo los autovalores y autovectores pueden ser aplicados para resolver diferentes problemas.

En primer lugar, creemos que el algoritmo de la potencia con deflación que desarrollamos es un método eficiente para el cálculo de autovalores y autovectores.

En segundo lugar, vimos que nos permitió identificar cuáles son los nodos más influyentes en la red del club de karate. Además, pudimos ver cómo, según las conexiones de cada persona, iría hacia uno u otro grupo tras la pelea ocurrida entre los integrantes. En este análisis, consideramos que la selección con el autovector más grande sirvió bastante bien para identificar la centralidad de los nodos. Sin embargo, al analizar el autovector correspondiente a la conectividad algebraica (el cual nos debería permitir encontrar la mejor separación entre ambos grupos) vimos que este tenía una correlación de "solo" un 53 por ciento y predecía correctamente la ubicación de 20 nodos tras la pelea. Consideramos que esta no es una predicción aceptable. De todas maneras, cabe aclarar que no existe una separación demasiado clara en el grafo del club de karate y esto puede llevar a confusiones sobre dónde debería pertenecer cada persona. Aun así, creemos que, para que la correlación sea aceptable, debería tener un valor del 80 por ciento o más.

En tercer lugar, analizamos la matriz de atributos de la matriz de Facebook, con la cual construimos una matriz de similaridad que nos permitió predecir cómo se unirían las personas en función de su lugar de trabajo, estudio, lugar de residencia, etc. En este caso, luego de probar con varios umbrales para la matriz de similaridad, logramos aproximar los resultados con un 90 por ciento de efectividad lo cual calificamos de manera positiva.

Por otro lado, nos pareció interesante el análisis de las k componentes principales y cómo solo 67 componentes explican el 90 por ciento de la información total que brindaban las 319 en su conjunto. Creemos que es una herramienta matemática que puede ser de mucho provecho, sobre todo a la hora de evaluar casos como el presente en el cual hay una gran cantidad de atributos mas muchos no aporten a la hora de buscar uniones entre las personas del set de datos.

Finalmente, concluímos que el análisis de autovectores y autovalores fue para nosotros sumamente provechoso para aproximarnos al tema de manera más práctica con matrices que explican casos de la vida real. Creemos que aplicar la teoría en casos de este estilo es por demás interesante.