INTRODUCCIÓN
AL RECONOCIMIENTO DE PATRONES

SPECTRAL CLUSTERING

Natalia Pignataro
Guillermo Figueredo
2008

Indice

1. Objetivo

El objetivo del presente trabajo es el análisis teórico y de implementación práctica, de la técnica de agrupamiento Spectral Clustering (SC).

Se presentará un análisis de las ventajas de esta técnica respecto al algoritmo de clustering visto en el curso, k-means.

El análisis teórico del algoritmo SC está basado principalmente en [1], complementando este análisis con ejemplos sintéticos implementados en Matlab que evidencian las diferentes particularidades del SC.

La implementación en Matlab permite la comparación de la clasificación de datos por SC y k-means, resaltando con ejemplos sintéticos, las principales diferencias entre los mismos.

2. Introducción

En diferentes áreas como pueden ser la biología, estadística, ciencias sociales o medicina, se cuenta con un gran número de datos experimentales y se requiere en una primera etapa de análisis, identificar grupos de "comportamiento" similar. Para esto las mejores técnicas son las de clustering.

Si bien la técnica de Spectral Clustering data de 35 años atrás, ha tomado gran auge en estos últimos años. En 1973, Donath y Hoffman plantearon por primera vez la idea de construir particiones de un grafo basándose en los vectores propios de la matriz de adyacencia. Varios trabajos se subsiguieron a esta idea, aunque la técnica se populariza luego de la aplicación a la segmentación de imágenes, propuesta por Shi y Malik en [3] en el 2000.

Las principales ventajas de la técnica de SC son, su simple implementación, su resolución eficiente con un software estándar de álgebra lineal y en la mayoría de los casos presenta mejor performance que los algoritmos tradicionales de clustering, como con el que se compara en este documento, k-means.

Si bien las ventajas la hacen una técnica atractiva, tiene muchos puntos en los cuales deja cierta incertidumbre. El método utilizado para la implementación del algoritmo no tiene una demostración teórica de cuan alejado se puede estar de la solución correcta. Sumado a esto, el algoritmo tiene varios parámetros de ajuste, algunos de los cuales carecen de fundamentos teóricos que permitan optimizar su elección según el caso. Esto requiere de una implementación "supervisada" que lo hace menos automático y general que otros algoritmos. En [2] se plantea y testea sobre casos sintéticos e imágenes, un algoritmo de SC que no requiere ajustes.

En el siguiente punto se detalla el algoritmo de spectral clustering y los fundamentos teóricos que lo sustentan. En 4, se describe brevemente la implementación realizada en Matlab y las consideraciones prácticas involucradas. En el punto 5 se resumen los resultados obtenidos de clasificación para k-means y SC y finalmente en 6, se resumen las conclusiones derivadas del análisis realizado en el presente trabajo.

3. Técnica de Spectral Clustering

3.1. Grafos de similitud

Clustering implica dividir los datos en grupos de afinidad de forma que los datos dentro de un grupo sean "parecidos" entre si y "distintos" con los datos de los demás grupos. Si se dispone de n datos $x_1 \ldots x_n$ y de una cierta función de similitud entre los mismos $s_{ij}>=0$ (para una pareja de puntos i, j), una buena representación de los datos es mediante el *grafo de similitud G=(V,E). V* es el conjunto de los vértices donde cada punto se corresponde con un vértice, y *E* es el conjunto de las líneas que unen los vértices (aristas). Es usual representar con el grosor (o el color) de la línea que une dos vértices el grado de afinidad (peso) de los mismos.

Con esta representación en mente, el problema de clustering puede ser reformulado de la siguiente manera: *se quiere generar una partición del grafo de forma que las líneas que unen distintos grupos tengan "bajo" peso (puntos en grupos distintos son disímiles entre si) y que las líneas que unen puntos del mismo grupo tengan "alto" peso (puntos del mismo grupo son muy parecidos entre sí)*.

3.1.1 Notación usual de grafos

• Un grafo *G=(V,E)* es *no direccionado* si $w_{ij}= w_{ji}$, y contiene pesos (*sopesada*) si $w_{ij}>=0$ para todo i,j.

• *Matriz de adyacencia W* es la matriz cuyos elementos son los w_{ij}.

• *Grado de un vértice* v_i : $d_i = \sum_{j=1}^{n} w_{ij}$.

• *Matriz de grado* D: es una matriz diagonal con los elementos $a_{ii}= d_i$.

• *Adyacencia* entre A y B (A y B dos grupos o clusters): $W(A,B) = \sum_{i \in A, j \in B} w_{ij}$.

• *Tamaño* de A: $|A|$ = número de vértices pertenecientes a A; $vol(A) = \sum_{i \in A} d_i$.

• A *conectado* si para todo i, j perteneciente a A, las líneas o camino de líneas que conectan v_i con v_j están incluidas en A.

• A *componente conectado* si es *conectado* y no existen conexiones con vértices de $\overline{A}$ (complento de A).

• *Partición* de V: $A_1 \cup A_2 \cup A_3 \ldots \cup A_k = V$ y $A_i \cap A_j = \Phi$ (para todo i, j).

3.1.2 Diferentes Grafos de Similitud

Recordar que la meta de los grafos es la de modelar las relaciones de vecindad entre vértices (datos). A continuación se describen los tipos más usuales de grafos de similitud.

Grafo de vecindad-ε: conecta dos vértices v_i , v_j si la distancia $d_{ij}<\varepsilon$. Se hace notar que para ε pequeño, los vértices conectados tienen pesos de orden similar por lo cual los pesos no aportan información relevante de los datos al grafo. Es usual que este grafo no tenga pesos asociados.

Grafo de k-vecinos cercanos: conecta dos vértices v_i, v_j si v_j está entre los k-vecinos cercanos de v_i. Notar que la relación de k vecindad no es recíproca, por lo que existe un detalle de direccionalidad, para dar solución a este punto se tiene entonces:

Grafo de k-vecinos cercanos: conecta dos vértices v_i, v_j si v_j está entre los k-vecinos cercanos de v_i o si v_i está entre los k-vecinos cercanos de v_j.

Grafo de k-vecinos mutuos cercanos: conecta dos vértices v_i, v_j si v_j está entre los k-vecinos cercanos de v_i y si v_i está entre los k-vecinos cercanos de v_j.

En general en cualquier variante se trata de un grafo que incluye pesos.

Grafo totalmente conectado: conecta dos vértices v_i, v_j si $s_{ij}>0$ y los pesos w_{ij} se asignan según s_{ij}, s_{ij} deberá modelar la vecindad local (ejemplo: Similitud Gaussiana $\exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$ donde σ controla el grado de cercanía).

Es importante hacer notar que en [1] se indica que a la fecha del trabajo (2007) no había demostración teórica que explicase el tipo de influencia de un grafo de similitud u otro en el resultado final del SC.

3.2. Grafos Laplacianos y Propiedades Básicas

En los puntos siguientes se adoptan las siguientes consideraciones, se trabaja con grafos G no direccionados y con una matriz de pesos (de adyacencias) W ($w_{ij}=w_{ji}>0$). Los valores propios siempre aparecerán ordenados de menor a mayor respetando sus multiplicidades. Los vectores propios no necesariamente aparecerán normalizados y se hablará de los "primeros k vectores propios" correspondiendo esto a los k vectores propios asociados a los k valores propios más pequeños.

3.2.1 Grafo Laplaciano no normalizado

Definición: **L= D-W**

Proposición 1 (Propiedades de L)

1. $f \in \Re^n \Rightarrow f^{'}Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$
2. L simétrica y semidefinida positiva
3. Menor valor propio de L igual a cero con vector propio asociado constante
4. L tiene n valores propios reales positivos, $0=\lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \ldots\ldots \lambda_n$

Prueba:

(1)

$$f^{'}Lf \overset{def}{=} f^{'}(D-W)f \overset{linealidad}{=} f^{'}(D)f - f^{'}(W)f \overset{Ddiagonal}{=} \sum_{i=1}^{n} d_i f_i^2 - f^{'}Wf = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i=1}^{n} f_i \left(\sum_{j=1}^{n} w_{ij} f_j\right) =$$

$$= \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i w_{ij} f_j = \frac{1}{2}\left(2\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} f_i w_{ij} f_j\right) = \frac{1}{2}\left(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} f_i w_{ij} f_j + \sum_{i=1}^{n} d_i f_i^2\right) =$$

$$= \frac{1}{2}\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{n} w_{ij}\right) f_i^2 - 2\sum_{i,j=1}^{n} f_i w_{ij} f_j + \sum_{j=1}^{n}\left(\sum_{i=1}^{n} w_{ij}\right) f_j^2\right) = \frac{1}{2}\left(\sum_{i,j=1}^{n} w_{ij} f_i^2 - 2\sum_{i,j=1}^{n} f_i w_{ij} f_j + \sum_{i,j=1}^{n} w_{ij} f_j^2\right) =$$

$$= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(f_i - f_j\right)^2$$

(2)

$$L = D - W = \begin{pmatrix} d_1 & 0 & . & . & 0 \\ 0 & d_2 & & & . \\ . & & . & & . \\ . & & & . & 0 \\ 0 & . & . & 0 & d_n \end{pmatrix} - \begin{pmatrix} w_{11} & w_{12} & . & . & w_{1n} \\ w_{21} & w_{22} & & & . \\ . & & . & & . \\ . & & & . & . \\ w_{n1} & . & . & . & w_{nn} \end{pmatrix} = \begin{pmatrix} \sum_{\forall j \neq 1} w_{1j} & -w_{12} & . & . & -w_{1n} \\ -w_{21} & \sum_{\forall j \neq 2} w_{2j} & & & . \\ . & & . & & . \\ . & & & . & . \\ -w_{n1} & . & . & . & \sum_{\forall j \neq n} w_{nj} \end{pmatrix}$$

Esta última es simétrica por ser el grafo no direccional. A su vez por (1) $\forall f \in \Re^n, f^{'} L f \geq 0 \Rightarrow L$ semidefinida positiva.

(3) Sea v vector propio asociado al valor propio 0 $\Rightarrow Lv = \lambda v = 0 \Rightarrow v^{'} L v = v^{'} \lambda v = 0 = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(v_i - v_j \right)^2 \Leftrightarrow \underbrace{w_{ij}}_{\geq 0} \left(v_i - v_j \right) = 0 \Leftrightarrow v_i = v_j \Leftrightarrow v = cte$

(4) Como es simétrica semidefinida positiva tiene n valores propios reales positivos. Por (1), $\lambda_1 = 0$ lo que prueba (4).

Proposición 2 (Número de componentes conectados y espectro de L)

Grafo G no direccionado con pesos no-negativos $\Rightarrow$ La multiplicidad k del valor propio 0 de L es igual al número de componentes conectados $A_1, A_2, A_3, \ldots, A_k$ en el grafo. El espacio generado por el valor propio 0 es generado por los vectores propios indicatrices $1_{A_1}, \ldots, 1_{A_k}$ de cada componente conectada.

Prueba:

Caso k=1 $\Rightarrow$ un único cluster (un componente conectado) $\Rightarrow$ si 0 es valor propio con vector propio f: $Lf = 0 \Leftrightarrow f^{'} L f = 0 = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(f_i - f_j \right)^2 \Leftrightarrow w_{ij} \left(f_i - f_j \right)^2 = 0$. Si v_i, v_j conectados, entonces $w_{ij} > 0 \Rightarrow f_i = f_j$ (f constante). Si v_i, v_j no conectados ($w_{ij}=0$), existe un camino que los conecta (def. componente conectado) $\Rightarrow f_i = f_j$ (f resulta también constante). Entonces 1 es vector propio del valor propio 0.

Caso k>1. Reordenamos W de forma que colocamos los vértices v_i, v_j conectados ($w_{ij}>0$) juntos de forma de obtener bloques en L $\begin{pmatrix} L_1 & 0 & . & 0 \\ 0 & L_2 & & . \\ . & & . & 0 \\ 0 & . & 0 & L_k \end{pmatrix} \Rightarrow L_i$ es una matriz Laplaciana con vector propio 1_{A_i} como vector propio asociado al valor propio 0 de multiplicidad 1. Luego los valores propios de L son los de cada L_i que son 0 con multiplicidad 1 y lo mismo para los vectores propios $\Rightarrow$ L tiene valor propio 0 de multiplicidad k y vectores propios asociados que son indicatrices de los componentes conectados.

3.2.2 Grafos Laplacianos normalizados

$$L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \text{ (simétrica)}$$
$$L_{rw} = D^{-1} L = I - D^{-1} W \text{ (random-walk)}$$

Proposición 3 (Propiedades de Lsym y Lrw):

1. $f \in \Re^n \Rightarrow f^{'} L_{sym} f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2$
2. λ es valor propio de L_{rw} con vector propio $u \Leftrightarrow \lambda$ es valor propio de L_{sym} con vector propio $w = D^{\frac{1}{2}} u$
3. λ es valor propio de L_{rw} con vector propio $u \Leftrightarrow \lambda$ y u cumplen $Lu = \lambda D u$
4. 0 es valor propio de L_{rw} con vector cte 1 como vector propio. 0 es valor propio de L_{sym} con vector propio $D^{\frac{1}{2}} 1$
5. L_{sym} y L_{rw} son semidefinidas positivas y tienen n valores propios reales no negativos $0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \ldots\ldots \lambda_n$

Prueba:

$$f^{'} L_{sym} f \overset{def+linealidad}{=} f^{'} I f - f^{'} (D^{\frac{-1}{2}} W D^{\frac{-1}{2}}) f \overset{operando}{=} \sum_{i=1}^{n} f_i^2 - \sum_{i=1}^{n} f_i \left(\sum_{j=1}^{n} \frac{w_{ij}}{\sqrt{d_i d_j}} f_j \right) =$$

$$= \frac{1}{2} \left(\sum_{i=1}^{n} f_i^2 - 2 \sum_{i,j=1}^{n} f_i \frac{w_{ij}}{\sqrt{d_i d_j}} f_j + \sum_{j=1}^{n} f_j^2 \right) = \frac{1}{2} \left(\sum_{i=1}^{n} d_i \frac{f_i}{d_i}^2 - 2 \sum_{i,j=1}^{n} f_i \frac{w_{ij}}{\sqrt{d_i d_j}} f_j + \sum_{j=1}^{n} d_j \frac{f_j}{d_j}^2 \right) =$$

(1)

$$= \frac{1}{2} \left(\sum_{i=1}^{n} \left(\sum_{j=1}^{n} w_{ij} \right) \frac{f_i}{d_i}^2 - 2 \sum_{i,j=1}^{n} f_i \frac{w_{ij}}{\sqrt{d_i d_j}} f_j + \sum_{j=1}^{n} \left(\sum_{i=1}^{n} w_{ij} \right) \frac{f_j}{d_j}^2 \right) =$$

$$= \frac{1}{2} \left(\sum_{i,j=1}^{n} w_{ij} \frac{f_i^2}{d_i} - 2 \sum_{i,j=1}^{n} f_i \frac{w_{ij}}{\sqrt{d_i d_j}} f_j + \sum_{i,j=1}^{n} w_{ij} \frac{f_j^2}{d_j} \right) = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2$$

(2)

$$L_{sym} w = \lambda w \Leftrightarrow D^{\frac{-1}{2}} L \underbrace{D^{\frac{-1}{2}} w}_{u} = \lambda w$$
$$D^{\frac{-1}{2}} L u = \lambda w \Leftrightarrow \underbrace{D^{-1} L u}_{L_{rw}} = \lambda \underbrace{D^{\frac{-1}{2}} w}_{u}$$

(3) $\underbrace{L_{rw}}_{D^{-1}L} u = \lambda u \Leftrightarrow L u = \lambda D u$

(4) $L_{rw} 1 = 0 \Leftrightarrow D^{-1} L 1 = 0$ y como $L1 = 0 \Rightarrow D^{-1} L 1 = 0$ y por (2) 0 resulta valor propio de L_{sym} con $w = D^{\frac{1}{2}} 1$ vector propio asociado

(5) L_{sym} por (1) es semidefinida positiva y simétrica por definición con valores propios reales positivos, por ende, como los valores propios de L_{rw} son los mismos que los de L_{sym} por (2) se cumple lo mismo para L_{rw}

Proposición 4 (Número de componentes conectados y espectros de Lsym y Lrw):

Sea G grafo no direccionado con pesos no negativos. Luego la multiplicidad k del valor propio 0 de ambos L_{rw} y L_{sym} es igual al número de componentes conectados $A_2, A_3, \ldots, A_k$ en el grafo. Para L_{rw} el espacio de vectores propios de 0 es el generado por los vectores indicatrices $1_{A_1}, \ldots, 1_{A_k}$ de los componentes conectados. Para L_{sym} el espacio de vectores de 0 es el generado por los vectores $D^{\frac{1}{2}}1_{A_1}, \ldots, D^{\frac{1}{2}}1_{A_k}$.

Prueba: Es análoga a la de la Proposición 2, usando Proposición 3.

3.3. Spectral Clustering (Algoritmos)

Spectral Clustering no normalizado

Entrada: matriz de similitud $S \in \Re^{n \times n}$ y número k de clusters a construir.

- Construir un grafo de similitud por alguno de los métodos vistos en 3.1. Sea W la matriz de adyacencia asociada a dicho grafo.
- Calcular la Laplaciana no normalizada L.
- **Hallar los k primeros vectores propios de L ($u_1 \ldots u_k$)**.
- Sea $U \in \Re^{n \times k}$ la matriz conteniendo los vectores propios $u_1 \ldots u_k$ como columnas.
- Con i de 1 a n, sea $y_i \in \Re^k$ el vector de la fila i de la matriz U.
- Clasificar los puntos $(y_i)_{i=1,\ldots,n}$ en $\Re^k$ con el algoritmo k-means en los clusters $C_1 \ldots C_k$.

Salida: Clusters $A_1 \ldots A_k$ con $A_i = \{j \mid y_j \in C_i\}$.

Spectral Clustering normalizado según Shi y Malik (2000)

Entrada: matriz de similitud $S \in \Re^{n \times n}$ y número k de clusters a construir.

- Construir un grafo de similitud por alguno de los métodos vistos en 3.1. Sea W la matriz de adyacencia asociada a dicho grafo.
- Calcular la Laplaciana no normalizada L.
- **Hallar los k primeros vectores propios generalizados ($u_1 \ldots u_k$) del problema de vectores propios generalizado** $Lu = \lambda Du$.
- Sea $U \in \Re^{n \times k}$ la matriz conteniendo los vectores propios $u_1 \ldots u_k$ como columnas.
- Con i de 1 a n, sea $y_i \in \Re^k$ el vector de la fila i de la matriz U.
- Clasificar los puntos $(y_i)_{i=1,\ldots,n}$ en $\Re^k$ con el algoritmo k-means en los clusters $C_1 \ldots C_k$.

Salida: Clusters $A_1 \ldots A_k$ con $A_i = \{j \mid y_i \in C_i\}$.

Spectral Clustering normalizado según Ng, Jordan, y Weiss (2002)

Entrada: matriz de similitud $S \in \Re^{n \times n}$ y número k de clusters a construir.

- Construir un grafo de similitud por alguno de los métodos vistos en 3.1. Sea W la matriz de adyacencia asociada a dicho grafo.
- Calcular la Laplaciana normalizada $\mathbf{L_{sym}}$.
- **Hallar los k primeros vectores propios de L_{sym} ($u_1 \ldots u_k$)**.
- Sea $U \in \Re^{n \times k}$ la matriz conteniendo los vectores propios $u_1 \ldots u_k$ como columnas.
- **Formar la matriz $T \in \Re^{n \times k}$ a partir de U normalizando sus filas, o sea** $t_{ij} = \dfrac{u_{ij}}{\left(\sum_k u_{ik}^2\right)^{\frac{1}{2}}}$.
- Con i de 1 a n, sea $y_i \in \Re^k$ el vector de la fila i de la matriz T.
- Clasificar los puntos $(y_i)_{i=1....n}$ en $\Re^k$ con el algoritmo k-means en los clusters $C_1 \ldots C_k$.

Salida: Clusters $A_1 \ldots A_k$ con $A_i = \{j \mid y_j \in C_i\}$.

La idea de los algoritmos es usar las Laplacianas de forma de transformar los puntos originales a un nuevo espacio donde, como se verá más adelante, se realzan los clusters y los puntos transformados se vuelven fácilmente separables. De esta forma, con la aplicación de algoritmos sencillos de clustering como por ejemplo k-means los puntos pueden clasificarse fácilmente.

A continuación se presenta un ejemplo sencillo para ilustrar los principios del algoritmo SC. Se trata de 500 puntos $\in \Re$ $x_1 \ldots x_{500}$ sorteados según una mezcla de 3 gaussianas con distinta probabilidad para cada clase y distintas varianzas para cada gaussiana. La figura 1 muestra el histograma resultante.

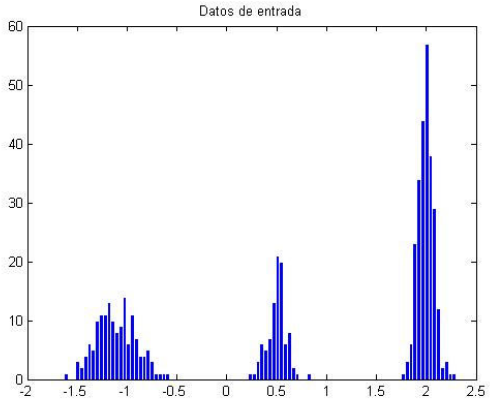

Figura 1: Histograma para las 3 gaussianas

Se eligió como función de similitud gaussiana $s(x_i, x_j) = \exp\left(\dfrac{-|x_i - x_j|^2}{2\sigma^2}\right)$ con $\sigma = 0.4$.

Como grafo de similitud se consideró k-vecinos cercanos (para k=10) y totalmente conectado, y se realiza SC con L y con L_{rw}.

La figura 2 muestra los 10 primeros valores propios de L_{rw} y los primeros 5 vectores propios asociados para el grafo 10-vecinos más cercanos. Se observa que los primeros 3 valores propios son cero y que los vectores propios asociados son indicadores de los clusters. Esto es así porque en el grafo 10-vecinos cercanos quedaron 3 componentes conectados, en cuyo caso los vectores propios vienen dados según la Proposición 2 y la Proposición 4 (la multiplicidad del valor propio 0 es 3 que es la misma que la cantidad de componentes conectados en el grafo).

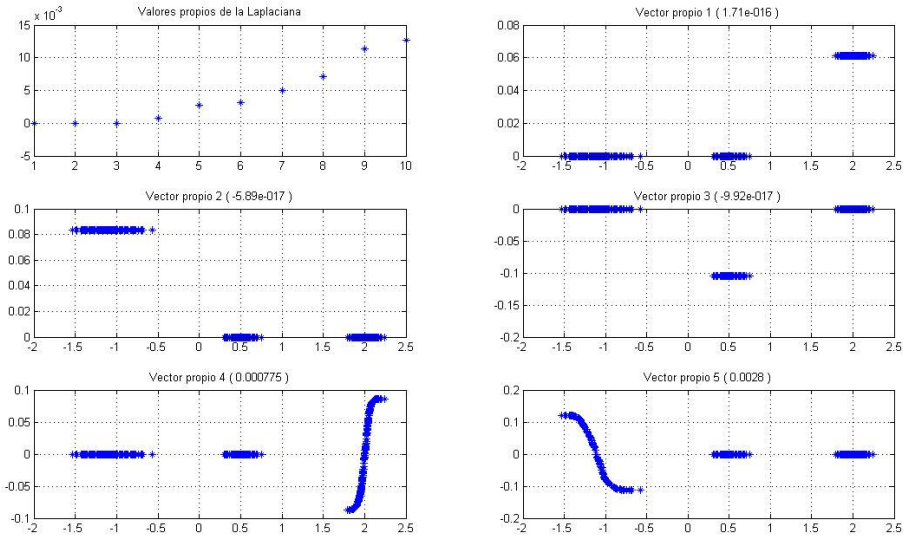

Figura 2: Valores propios de L_{rw} para k-vecinos (k=10) y vectores propios asociados.

La figura 3 muestra los 10 primeros valores propios de L y los primeros 5 vectores propios asociados para el grafo 10-vecinos más cercanos. Se observa el mismo comportamiento que en la figura 2 (idénticas razones).

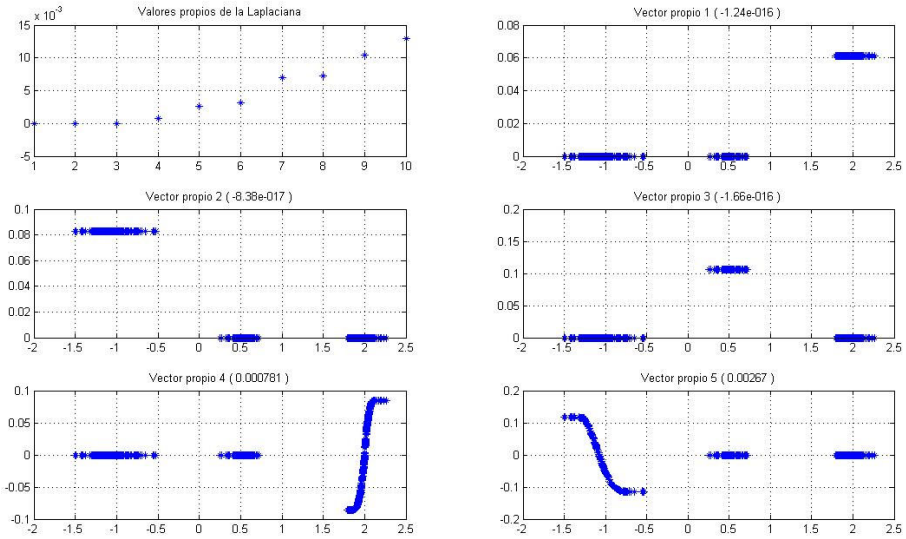

Figura 3: Valores propios de L para k-vecinos (k=10) y vectores propios asociados.

La figura 4 muestra los 10 primeros valores propios de L_{rw} y los primeros 5 vectores propios asociados para el grafo totalmente conectado. Como la función de similitud gaussiana es siempre positiva, el grafo contiene 1 solo componente conectado. Por este motivo el valor propio 0 tiene multiplicidad 1 y el vector asociado es constante. Son los siguientes vectores propios los que contienen la información acerca de los clusters (Observar que con un umbral en -0.05 el vector propio 2 separa la gaussiana centrada en -1 de las otras 2, y que con el mismo umbral, el vector propio 3 separa la gaussiana centrada en 0.5 de las otras 2 con lo cual es posible separar las 3 gaussianas).

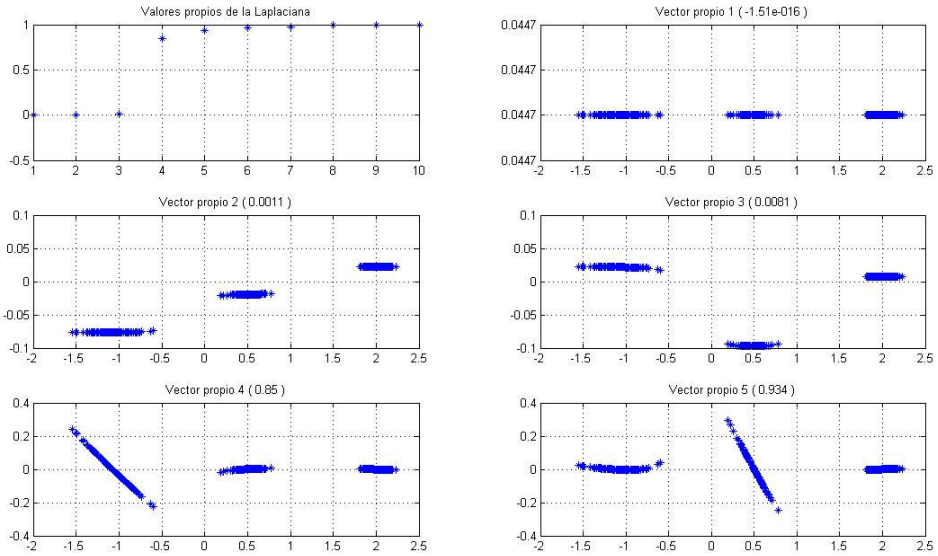

Figura 4: Valores propios de L_{rw} para totalmente conectado y vectores propios asociados.

La figura 5 muestra los 10 primeros valores propios de L y los primeros 5 vectores propios asociados para el grafo totalmente conectado. Se observa el mismo comportamiento que en la figura 4 (idénticas razones).

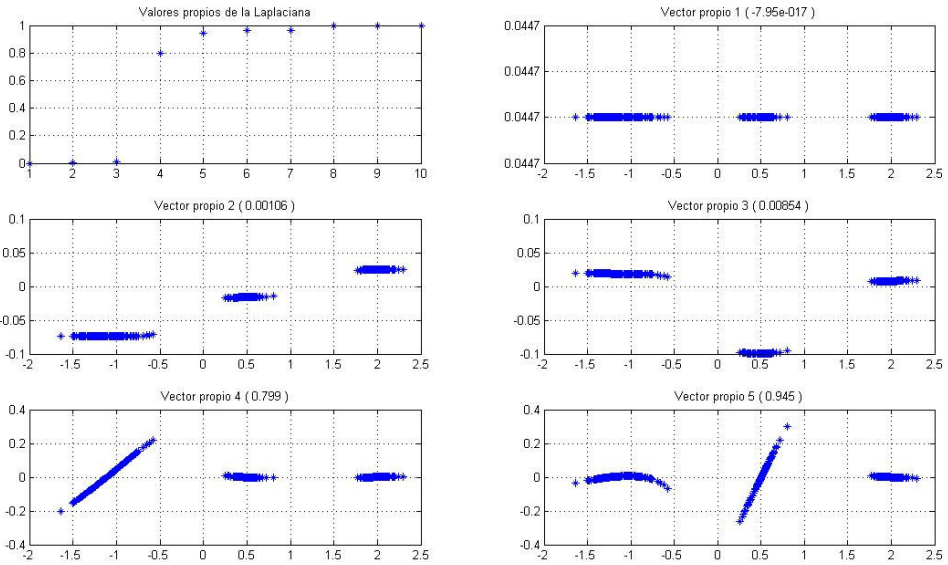

Figura 5: Valores propios de L para totalmente conectado y vectores propios asociados.

En el caso del grafo de los 10-vecinos cercanos los 3 primeros valores propios son cero, correspondientes a los 3 componentes conectados del grafo. Si bien los valores propios del grafo totalmente conectado son cercanos a cero, sólo el primero es exactamente cero (correspondiente a la única componente conectada, todo el grafo). A pesar de esta sutil diferencia ambos grafos dan un buen resultado en este caso, el grafo de 10-vecinos cercanos separa directamente los 3 clusters originales y si bien el grafo totalmente conectado tiene una sola componente conectada, el espectro de la matriz asociada logra separar los datos en los 3 clusters correctos. Esto se evidencia en que los 2 valores propios siguientes al cero son cercanos al mismo y la diferencia entre el tercero y el cuarto es más grande (eigengap para calculo de la cantidad de clusters, detallada más adelante).

3.4. Interpretación como corte del grafo

Objetivo: se quiere encontrar una partición del grafo en la cual los bordes entre diferentes grupos tengan muy poco peso y los bordes dentro de un mismo grupo tengan pesos altos. Lo que buscamos es la partición A_1, A_2, …. A_k que minimice el corte

$$cut(A_1, A_2, ..., A_k) = \frac{1}{2}\sum_{i=1}^{k} W(A_i, \overline{A_i})$$

Este problema es relativamente sencillo de resolver para k=2. Sin embargo, en la práctica a menudo no arriba a particiones satisfactorias ya que muchas veces presenta puntos aislados como clusters, lo cual no es deseable. Una manera de resolver esto es requerir que el tamaño de los clusters sea relativamente grande. Es entonces que se tiene RatioCut (Hagen y Kahng, 1992) y Ncut (Shi y Malik, 2000) siendo sus definiciones:

$$RatioCut(A_1, A_2, ..., A_k) = \sum_{i=1}^{k} \frac{W(A_i, \overline{A_i})}{|A_i|} = \sum_{i=1}^{k} \frac{cut(A_i, \overline{A_i})}{|A_i|}$$

$$Ncut(A_1, A_2, ..., A_k) = \sum_{i=1}^{k} \frac{W(A_i, \overline{A_i})}{vol(A_i)} = \sum_{i=1}^{k} \frac{cut(A_i, \overline{A_i})}{vol(A_i)}$$

Observar que ambas funciones objetivo $\frac{1}{|A_i|}$ y $\frac{1}{vol(A_i)}$ son mínimas cuando todas las particiones son iguales (esto refuerza la idea de balancear los clusters de forma de tener suficientes puntos en cada uno). Spectral Clustering resuelve una aproximación a estos problemas.

3.4.1 Aproximación de RatioCut para k=2

Se quiere resolver: $\underbrace{\min}_{A \subset V} RatioCut(A, \overline{A})$ (1).

Se reescribe el problema usando la siguiente función indicatriz:

$$A \subset V, f = (f_1, ...f_n)' \in \Re^n / f_i = \begin{cases} \sqrt{\frac{|\overline{A}|}{|A|}} \Leftrightarrow v_i \in A \\ -\sqrt{\frac{|A|}{|\overline{A}|}} \Leftrightarrow v_i \in \overline{A} \end{cases} \quad (2)$$

Observar entonces que:

$$f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 = \frac{1}{2}\sum_{i \in A, j \in \overline{A}}^{n} w_{ij}\left(\sqrt{\frac{|\overline{A}|}{|A|}} + \sqrt{\frac{|A|}{|\overline{A}|}}\right)^2 + \frac{1}{2}\sum_{i \in A, j \in A}^{n} w_{ij}\left(-\sqrt{\frac{|\overline{A}|}{|A|}} - \sqrt{\frac{|A|}{|\overline{A}|}}\right)^2 =$$

$$= cut(A, \overline{A})\left(\frac{|\overline{A}|}{|A|} + \frac{|A|}{|\overline{A}|} + 2\right) = cut(A, \overline{A})\left(\frac{|A| + |\overline{A}|}{|A|} + \frac{|A| + |\overline{A}|}{|\overline{A}|}\right) = |V| \cdot RatioCut(A, \overline{A})$$

y que:

$$\sum_{i=1}^{n} f_i = \sum_{i \in A}\sqrt{\frac{|\overline{A}|}{|A|}} - \sum_{i \in \overline{A}}\sqrt{\frac{|A|}{|\overline{A}|}} = |A|\sqrt{\frac{|\overline{A}|}{|A|}} - |\overline{A}|\sqrt{\frac{|A|}{|\overline{A}|}} = 0$$

por lo que $f_i \perp 1$, además: $\|f_i\|^2 = \sum_{i=1}^{n} f_i^2 = |A|\frac{|\overline{A}|}{|A|} + |\overline{A}|\frac{|A|}{|\overline{A}|} = |\overline{A}| + |A| = n$.

Todo lo anterior implica que (1) es equivalente a:

$$\underbrace{\min}_{A \subset V} f^{'} L f \text{ sujeto a } f \perp 1,\ f_i \text{ definido según (2) y } \|f\| = \sqrt{n}. \quad (3)$$

Esto es un problema discreto ya que f_i sólo puede tomar dos valores particulares. Aún así es un problema NP complejo. Entonces se procede a relajar (3) permitiendo que f_i tome valores en $\Re$ lo que lleva al problema de optimización relajado:

$$\underbrace{\min}_{f \in \Re^n} f^{'} L f \text{ sujeto a } f \perp 1 \text{ y } \|f\| = \sqrt{n}. \quad (4)$$

La solución a este problema corresponde al vector f que es el vector propio asociado al segundo menor valor propio de L (se usa teorema de Rayleigh-Ritz). Entonces, se aproxima la solución a (1) por medio del segundo vector propio de L. Para construir la partición, se re-transforma la solución f en los reales a valores discretos. La manera más sencilla es por medio de la función signo de f como indicador:

$$\left.\begin{matrix} v_i \in A \Leftrightarrow f_i \geq 0 \\ v_i \in \overline{A} \Leftrightarrow f_i < 0 \end{matrix}\right\}$$

Lo que en realidad hacen los algoritmos de SC es aplicar k-means a los valores de f en los reales y agrupar en clusters:

$$\left.\begin{matrix} v_i \in A \Leftrightarrow f_i \in C \\ v_i \in \overline{A} \Leftrightarrow f_i \in \overline{C} \end{matrix}\right\}$$

que es lo que hace el algoritmo de spectral clustering no normalizado para k=2.

3.4.2 Aproximación de RatioCut para k arbitrario

Sea una partición de V en k clusters $A_1,\ldots..A_k$, en la cual se definen k vectores indicatrices $h_j = \left(h_{1,j},...,h_{n,j}\right)^{'}$ según:

$$h_{i,j} = \left.\begin{cases} \dfrac{1}{\sqrt{|A_j|}} \Leftrightarrow v_i \in A_j \\ 0 \Leftrightarrow \textit{caso contrario} \end{cases}\right\} (i = 1,...,n;\ j = 1,...,k). \quad (5)$$

Considerar la matriz $H \in \Re^{n \times k}$ cuyas columnas son los vectores h_j. Observar que las columnas de H son ortonormales entre sí, o sea que $H^{'}H = I$. Puede verse de manera similar a lo expuesto antes que:

$$h_i^{'} L h_i = \frac{cut\left(A_i, \overline{A_i}\right)}{\overline{A_i}} \text{ y que } h_i^{'} L h_i = \left(H^{'} L H\right)_{ii}$$

Combinando esto se llega a que:

$$RatioCut\left(A_1,...,A_k\right) = \sum_{i=1}^{k} h_i^{'} L h_i = \sum_{i=1}^{k} \left(H^{'} L H\right)_{ii} = Tr\left(H^{'} L H\right)$$

por lo que el problema de minimizar RatioCut se puede reescribir como:

$$\underbrace{\min}_{A_1,...,A_k} Tr\left(H^{'} L H\right) \text{ sujeto a } H^{'}H = I \text{ y H definido como en (5).}$$

Nuevamente se relaja el problema permitiendo que los elementos de la matriz H tomen valores arbitrarios reales. Con esto el problema se vuelve:

$$\underbrace{\min}_{H \in \Re^{n \times k}} Tr\left(H^{'} L H\right) \text{ sujeto a } H^{'}H = I.$$

Volviendo a usar otra versión del teorema de Rayleigh-Ritz, la solución viene dada por la matriz H que contenga como columnas los primeros k vectores propios de la matriz L. En realidad H es la matriz U utilizada en el algoritmo no normalizado de SC (sección 3.3). Nuevamente es necesario reconvertir la matriz solución real a una matriz discreta. Como antes, la manera estándar es usar k-means sobre las filas de H (U) y esto lleva al algoritmo de SC no normalizado.

3.4.3 Aproximación de Ncut

Las ideas son muy similares a las expuestas para RatioCut. Para el caso k=2 se comienza por definirse el vector indicatriz f como:

$$f_i = \begin{cases} \sqrt{\frac{vol(\overline{A})}{vol(A)}} \Leftrightarrow v_i \in A \\ -\sqrt{\frac{vol(A)}{vol(\overline{A})}} \Leftrightarrow v_i \in \overline{A} \end{cases} \qquad (6)$$

Se puede chequear que $(Df)'1 = 0$, que $f'Df = vol(V)$ y que $f'Lf = vol(V)Ncut(A, \overline{A})$. Entonces puede reescribirse el problema de minimizar Ncut como:

$$\underbrace{\min}_{A} f'Lf \text{ sujeto a f definida como en (6), } Df \perp 1 \text{ y } f'Df = vol(V). \quad (7)$$

Nuevamente se relaja el problema permitiendo que f tome valores reales arbitrarios con lo cual el problema queda:

$$\underbrace{\min}_{A} f'Lf \text{ sujeto a } Df \perp 1 \text{ y } f'Df = vol(V). \qquad (8)$$

Luego, haciendo la sustitución $g = D^{\frac{1}{2}} f$ el problema queda reescrito como:

$$\underbrace{\min}_{g \in \Re^n} g' D^{\frac{-1}{2}} L D^{\frac{-1}{2}} g \text{ sujeto a } g \perp D^{\frac{1}{2}} 1 \text{ y } \|g\|^2 = vol(V). \qquad (9)$$

Observar que $D^{\frac{-1}{2}} L D^{\frac{-1}{2}} = L_{sym}$, $D^{\frac{1}{2}} 1$ es el primer vector propio de L_{sym} y vol(V) es una constante. Entonces aplicando una vez más el teorema de Rayleigh-Reitz, la solución viene dada por g igual al segundo vector propio de L_{sym}. Resustituyendo $f = D^{\frac{-1}{2}} g$ y usando la Proposición 3 se tiene que f es el segundo vector propio de L_{rw}, o lo que es lo mismo del problema generalizado $Lu = \lambda Du$.

Para k>2 clusters, se define $h_j = (h_{1,j}, ..., h_{n,j})'$ por:

$$h_{i,j} = \begin{cases} \frac{1}{\sqrt{vol(A_j)}} \Leftrightarrow v_i \in A_j \\ 0 \Leftrightarrow \textit{caso contrario} \end{cases} (i = 1,...,n; j = 1,...,k). \quad (10)$$

Nuevamente se forma la matriz H conteniendo como columnas los h_j y se observa que $H'DH = I$, $h_i' D h_i = 1$ y que $h_i' L h_i = \frac{cut(A_i, \overline{A_i})}{vol(A_i)}$ por lo que se puede reescribir el problema de minimizar Ncut como:

$$\underbrace{\min}_{A_1,...,A_k} Tr(H'LH) \text{ sujeto a } H'DH = I \text{ y H definida como en (10).}$$

Relajando el problema discreto, y substituyendo $T = D^{\frac{1}{2}} H$ se obtiene el problema relajado:

$$\underbrace{\min}_{T \in \Re^{n \times k}} Tr\left(T' D^{\frac{-1}{2}} L D^{\frac{-1}{2}} T \right) \text{ sujeto a } T'T = I. \qquad (11)$$

Nuevamente la solución a (11) viene dada por la matriz T cuyas columnas sean los primeros k vectores propios de L_{sym}. Resustituyendo $H = D^{\frac{-1}{2}}T$ y usando la Proposición 3 se ve que H contiene en sus columnas los primeros k vectores propios de L_{rw}. Esto hace el SC normalizado de Shi y Malik (2000).

3.4.4 Comentarios sobre la aproximación (problema relajado)

Lo más importante a destacar es que no hay garantía de que el resultado obtenido mediante SC (que es siempre una relajación de los problemas originales de minimización tanto de RatioCut como de Ncut) difiera poco de la solución al problema original.

3.5. Interpretación como una caminata aleatoria

Una caminata aleatoria (random walk – rw) es un proceso estocástico que aleatoreamente salta de un vértice del grafo a otro. La probabilidad de transición de v_i a v_j es proporcional al peso de la arista w_{ij} y viene dada por $p_{ij}=w_{ij}/d_i$. Sea entonces la matriz de transición $P=(p_{ij})_{ij=1,\ldots,n}$ de la caminata aleatoria definida como:

$$P = D^{-1}W .$$

Si el grafo es conectado y *no-bipartito*[1] la caminata aleatoria posee una única distribución estacionaria $\pi = (\pi_1,...,\pi_n)'$ donde $\pi_i = \frac{d_i}{vol(V)}$. Por su definición, las propiedades de P están relacionadas con las de L_{rw} (notar que L_{rw}=I-P).

Random walks y Ncut

Meila y Shi (2001) presentan una equivalencia formal entre Ncut y las probabilidades de transición de la caminata aleatoria.

Proposición 5 (Ncut via probabilidades de transición):

Sea G conectado y no-bipartito. Asúmase que se corre una caminata aleatoria $(X_t)_{t\in N}$ comenzando con X_0 en la distribución estacionaria π. Para conjuntos disjuntos A, B $\subset$ V, se denota $P(B|A)=P(X_1 \in B|X_0 \in A)$. Entonces:

$$Ncut(A,\overline{A}) = P(\overline{A} \mid A) + P(A \mid \overline{A})$$

Prueba:

Se observa que:

$$P(X_0 \in A, X_1 \in B) = \sum_{i\in A, j\in B} P(X_0 = i, X_1 = j) = \sum_{i\in A, j\in B} \pi_i p_{ij} = \sum_{i\in A, j\in B} \frac{d_i}{vol(V)} \frac{w_{ij}}{d_i} = \frac{1}{vol(V)} \sum_{i\in A, j\in B} w_{ij}$$

Con esto se tiene que:

$$P(X_1 \in B \mid X_0 \in A) = \frac{P(X_0 \in A, X_1 \in B)}{P(X_0 \in A)} = \left(\frac{1}{vol(V)} \sum_{i\in A, j\in B} w_{ij}\right)\left(\frac{vol(A)}{vol(V)}\right)^{-1} = \frac{\sum_{i\in A, j\in B} w_{ij}}{vol(A)}$$

y la proposición queda demostrada usando la definición de Ncut.

[1] Grafo V bi-partito si existen dos conjuntos de vértices V_1 y V_2 tales que $V_1 \cup V_2$=V y $V_1 \cap V_2=\phi$ y dados dos vértices cualesquiera del mismo conjunto, no existe arista que los una.

Esta proposición cultiva la intuición de que cuando se minimiza Ncut (problema relajado es SC normalizado), en realidad se busca un corte en el grafo de forma tal que una caminata aleatoria rara vez salta de A a $\overline{A}$ y viceversa.

La distancia conmutada

La distancia conmutada (también llamada *distancia resistiva*) c_{ij} entre dos vértices v_i y v_j es el tiempo esperado que le lleva a una caminata aleatoria ir de v_i a v_j y volver. La distancia conmutada entre dos vértices decrece si hay varios caminos cortos de llegar de v_i a v_j, lo que implica que puntos conectado por un camino corto y que subyacen en el grafo en una misma área con alta densidad de puntos son considerados más cercanos, que otros dos puntos que también están conectados por un camino corto pero que subyacen en el grafo en distintas áreas con alta densidad. Esto último es una propiedad interesante para clustering.

Se demuestra que la distancia conmutada entre dos vértices del grafo es equivalente a la distancia euclídea entre puntos en $\Re^n$ (esto para una construcción particular usando la matriz inversa generalizada L^{Ψ} de L). Sin embargo aún no existe prueba de que el hecho de que SC construya clusters a partir de la distancia euclídea de y_i pueda ser visto de forma equivalente a construir clusters entre los vértices del grafo basándose en la distancia conmutada.

3.6. Interpretación como teoría de perturbación

La teoría de la perturbación estudia como varían los valores propios y vectores propios de una matriz A si los elementos de esta matriz sufren una perturbación. Existen teoremas que prueban que para cierta distancia, los valores y vectores propios de la nueva matriz perturbada $\widetilde{A} = A + H$ distan de los originales de A, una cierta constante multiplicada por una norma de H. Esta constante depende del valor propio, y en particular de cuan separado este éste del resto del espectro.

SC entonces se enlaza con esta teoría de manera que para el caso real, en el cual no tenemos distancia cero entre clusters pero si pequeña, los vectores y valores propios de L no distarán mucho de los originales para el caso ideal que son indicatrices de la pertenencia de los vértices a cada cluster. Por ende k-means debería poder clasificarlos siempre que las perturbaciones no sean demasiado grandes.

3.6.1 El argumento formal de perturbación

Ángulos canónicos: Sean $\mathcal{V}^1$ y $\mathcal{V}^2$ dos subespacios $\subset \Re^d$ y V_1 y V_2 matrices cuyas columnas son bases ortonormales de dichos subespacios, respectivamente. Entonces, $\cos\Theta_i$ del ángulo principal Θ_i son los valores singulares de $V_1^{'}V_2$.

Teorema 7 (Davis-Kahan):

Sean A, H $\in \Re^{n\times n}$ matrices simétricas y sea $\|\cdot\|$ la norma *Frobenius* o la norma dos para matrices. Considerar $\widetilde{A} = A + H$ una versión perturbada de A. Sea $S_1 \subset \Re$ un intervalo. Denotaremos $\sigma_{S_1}(A)$ al conjunto de valores propios de A contenidos en S_1, y V_1 el espacio propio asociado a esos valores propios. Denotaremos $\sigma_{S_1}(\widetilde{A})$ y $\widetilde{V}_1$ a las cantidades respectivas para $\widetilde{A}$. Definiremos la distancia entre S_1 y el espectro de A fuera de S_1 como

$$\delta = \min\{|\lambda - s|; \lambda - valor\ propio\ de\ A, \lambda \notin S_1, s \in S_1\}.$$

Entonces la distancia $d(V_1, \widetilde{V}_1) = \|\sin\Theta(V_1, \widetilde{V}_1)\|$ entre los dos subespacios V_1 y $\widetilde{V}_1$ está acotada por: $d(V_1, \widetilde{V}_1) \le \frac{\|H\|}{\delta}$.

La idea detrás de este teorema es que si vemos A como L y $\widetilde{A}$ como $\widetilde{L}$ notamos que la diferencia entre los vectores propios de una y otra estará acotada y la cota será de utilidad siempre que exista un gap importante entre el valor propio k y el k+1. Si esto no fuese así, las diferencias pueden ser grandes dado que la cota puede dispararse. Por supuesto también dependerá de qué tan grande sea la perturbación $\|H\|$.

3.6.2 Comentarios acerca de la aproximación por perturbación

Un comentario importante es el siguiente, es importante que los componentes de los vectores propios del caso perturbado estén alejados de cero. Si este no fuese el caso, podría suceder que se mal clasifiquen algunos datos.

Esto no sucederá nunca en el caso ideal para L y L_{rw} pero podría llegar a pasar para L_{sym} dado que los vectores propios vienen dados por $D^{\frac{1}{2}}1_{A_i}$ y podría suceder si los grados de los vértices difieren mucho o si hay vértices con grado bajo que las entradas asociadas en los vectores propios sean muy pequeñas. Es para esto que se usa el paso de normalización en el algoritmo de SC normalizado de Ng, Jordan y Weiss (2002). Este paso puede llegar incluso a fallar por ejemplo si tenemos $u_{i1} = \varepsilon_1$ y $u_{i2} = \varepsilon_2$. Luego de la normalización, estos componentes se multiplican por $\frac{1}{\sqrt{\varepsilon_1^2 + \varepsilon_2^2}}$ y se vuelven grandes. Esto aumentará la probabilidad que estos puntos queden clasificados en el mismo cluster cuando en realidad pertenecen a clusters distintos.

En general, SC con L y L_{rw} está bien justificado por la teoría de perturbación, mientras que para L_{sym} hay que tener más cuidado, especialmente si existen vértices en el grafo con muy bajo grado.

4. Consideraciones Prácticas

La implementación de la técnica SC como se mencionó anteriormente fue realizada en Matlab. Se reutilizó parte del código del Tutorial de Matthias Hein & Ulrike von Luxburg disponible en www.mlss.cc/tuebingen07/. El código se adjunta en formato digital junto con este informe.

El programa principal se denomina SC.m, este programa llama a las diferentes funciones donde se implementan: la generación de diferentes datos sintéticos, la construcción de la matriz de similitud con el grafo asociado y la aplicación del algoritmo de clasificación por agrupamiento espectral y kmeans.

La función GD_GenerateData permite la generación de datos en una, dos y tres dimensiones de formas variadas como ser las lunas, distribuciones gaussianas de distintos parámetros, círculos concéntricos y la adicción de ruido blanco a los datos.

En la implementación del algoritmo es necesario tomar ciertas definiciones como por ejemplo, que función o parámetros de la función de similitud se va a utilizar o que tipo de grafo es más conveniente. En los siguientes puntos se analizan algunas consideraciones prácticas referentes a la elección de los grafos y parámetros a utilizar según el caso.

4.1. Grafo de similitud

4.1.1 Función de similitud

La función de similitud es la función que describe cuan símiles o disímiles son una pareja de datos. En este sentido la función similitud va a depender fuertemente del tipo de datos que se estén manejando, por ejemplo si hablamos de texto la función similitud de alguna forma debe tomar como símiles textos de la misma categoría, para que esta tenga sentido. En los casos más comunes donde los datos se representan como vectores de R^d, los candidatos más lógicos son la distancia entre los puntos y la función gaussiana de similitud, ambas implementadas en el programa SC.m.

La función gaussiana de similitud está dada por la expresión $s(x_i, x_j) = \exp(-\|x_i - x_j\|^2 / (2\sigma^2))$, no es como la distancia, en este caso la elección del parámetro σ tiene particular importancia. Este parámetro nos da la magnitud de similitud de puntos que queramos imponer. Un sigma muy grande, comparable con la diferencia de distancias resultará en valores de similitud grandes y parecidos y por lo tanto todos los puntos pueden ser tomados como símiles aunque no lo sean. Por el contrario con valores de sigma muy pequeños puntos muy cercanos pueden ser tomados como disímiles y tender a aislar muchos clusters y no clasificar correctamente, como veremos en un ejemplo más adelante.

En definitiva la elección de la función de similitud depende del dominio de donde provienen los datos y no hay una sugerencia general acerca de la misma, aunque en casos de puntos en R^d la más utilizada es la función gaussiana de similitud.

4.1.2 Tipo de grafo de similitud

Luego de elegir la función de similitud se debe elegir el tipo de grafo a utilizar para armar la matriz de adyacencia. Respecto a esta elección, tenemos que tener en cuenta las consideraciones señaladas en este punto.

En la figura 6, se muestra con un ejemplo sencillo las diferentes características de los grafos de similitud tratados en este documento.

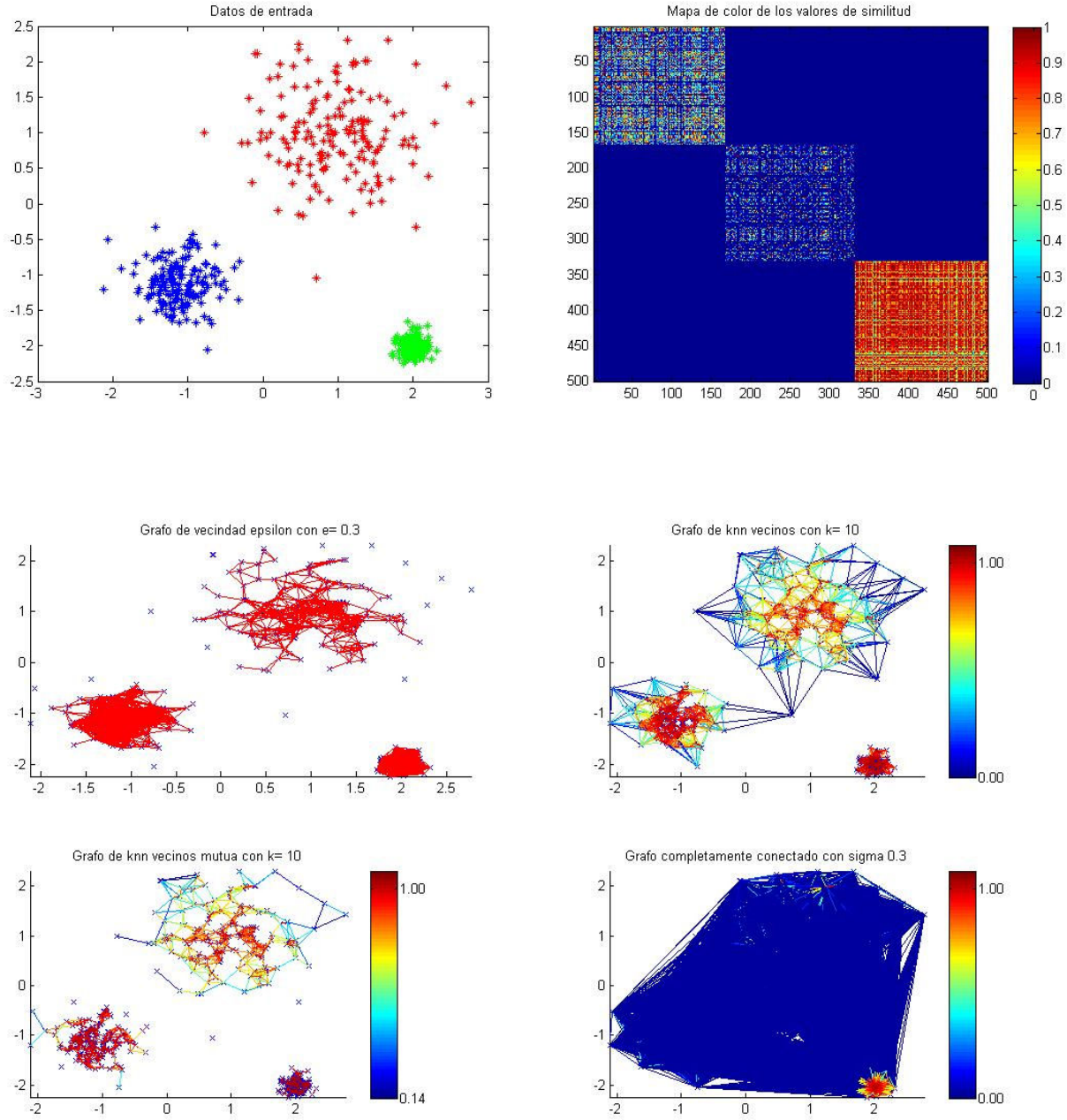

Figura 6. Arriba: Datos de entrada y matriz de similitud. Abajo: Diferentes grafos de similitud

Se consideraron en este ejemplo puntos en R^2 de 3 clases, los puntos de esas clases son generados por distribuciones gaussianas de diferentes varianzas. En la primera de las gráficas se muestran los puntos generados de esta forma.

En la segunda gráfica de la figura, se grafica la matriz de similitud generada para la función de similitud gaussiana, con σ=0.3, donde se notan claramente los tres bloques de mayor grado de similitud correspondientes a las tres clases generadas por las gaussianas.

En las siguientes cuatro gráficas se muestran los grafos de similitud generados a partir de estos datos y de la función de similitud gaussiana en el caso de los tres últimos y de la distancia en el caso de la gráfica de vecindad-ε.

La gráfica de vecindad-ε es un grafo sin pesos en este caso y como se puede observar en la gráfica es difícil encontrar un valor del parámetro que tenga utilidad. Es decir, cuando tenemos diferentes distancias entre puntos en diferentes regiones del espacio, este grafo no es útil para separar componentes conectadas. Por ejemplo, si tenemos datos con distintas "escalas" y elegimos un ε grande para unir los puntos más dispersos del grafo puede quedar completamente unido y si por el contrario, elegimos un ε más pequeño los puntos más dispersos quedarán desconectados, como es el caso de la figura.

El grafo de k-vecinos más cercanos no presenta el problema anterior, ya que puede trabajar con datos a diferentes escalas. Es decir, puede conectar dos clases de diferentes densidades como se ve en la figura entre dos de las gaussianas. Sin embargo como se ve también en la gráfica puede dejar componentes desconectadas en lugares alejados de alta densidad de puntos como pasa con la gaussiana de menor varianza.

No pasa lo mismo en la gráfica de k-vecinos mutua. Como se ve en la gráfica correspondiente, para el mismo valor de k deja los tres clusters desconectados ya que este tipo de grafo tiene la característica de no conectar regiones de diferentes densidades y tiende a conectar componentes de densidad similar. Este grafo es un subgrafo del grafo de k-vecinos y es un término medio entre éste y el de vecindad-ε. Es apropiado para detectar clusters de diferentes densidades, pero puede dejar puntos aislados, lo cual no es beneficioso para la clasificación.

El grafo completamente conectado usualmente se utiliza con la función de similitud gaussiana. En este grafo el parámetro que juega un rol fundamental es σ de la función de similitud gaussiana. Puntos en la vecindad local son conectados con pesos relativamente altos (líneas en rojo), mientras que las conexiones entre puntos lejanos tienen pesos positivos pero muy pequeños (degrade de azules).

La matriz de adyacencia que genera este grafo no es una matriz con bloques (como puede ser la matriz generada por los otros grafos donde cada componente conectado es un bloque de la matriz) ya que tiene una sola componente conectada, por lo tanto el primer vector propio será constante, asociado al valor propio 0 (teorema visto en el punto 3.2.1).

En la siguiente figura se muestran las matrices de adyacencia generadas a partir de los datos del ejemplo anterior por los distintos grafos. Se pueden observar bloques en cada una correspondientes a los componentes conectados. Como se observa en la figura 7, la primera matriz de adyacencia, generada por la vecindad-ε es una matriz que tiene dos colores, ya que no es un grafo sopesado y los puntos que están unidos son los que distan menos que ε (puntos rojos en la matriz), en otro caso están desconectados (puntos azules). Como se observa tanto en el grafo como en la matriz de adyacencia asociada, el parámetro elegido no es útil para clasificar los datos pues puntos de un mismo bloque aparecen aislados.

En cambio si se observan las otras 3 matrices de adyacencia se evidencian los tres bloques dentro de los cuales se nota la diferencia de pesos de las uniones entre sus puntos.

Respecto a la cantidad de componentes conectadas, según se puede observar en los diferentes grafos, así como en la gráfica de la figura 8, la única que tiene un solo componente conectado en este caso es la totalmente conectada ya que el grafo de k-vecinos tiene 2 componentes conectadas y tanto la de vecindad mutua como la de vecindad-ε tienen una cantidad mayor de componentes conectados. En el caso de la vecindad-ε se debe a que deja puntos aislados pero en el caso del grafo de k-vecinos mutuo es porque además de separar los tres componentes conectados deja puntos aislados. Recordemos que la matriz Laplaciana tendrá tantos valores propios en 0 como componentes conectados. Es decir, que la matriz Laplaciana del grafo totalmente conectado tendrá un único valor propio cero mientras que la de k-vecinos más cercanos tendrá dos. En la figura 9, se grafican los valores propios de las matrices Laplacianas correspondientes a cada tipo de grafo.

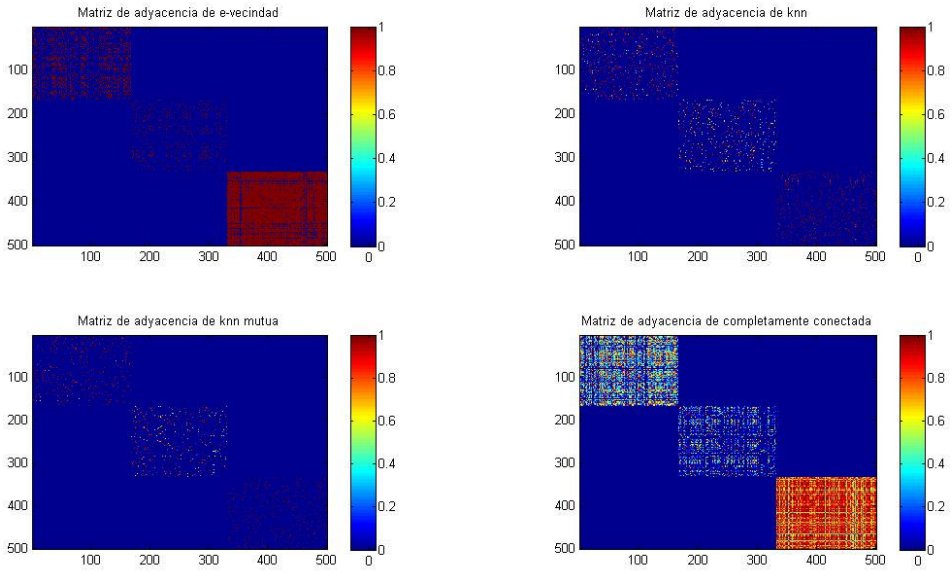

Figura 7. Matrices de adyacencia generadas con cada tipo de grafo

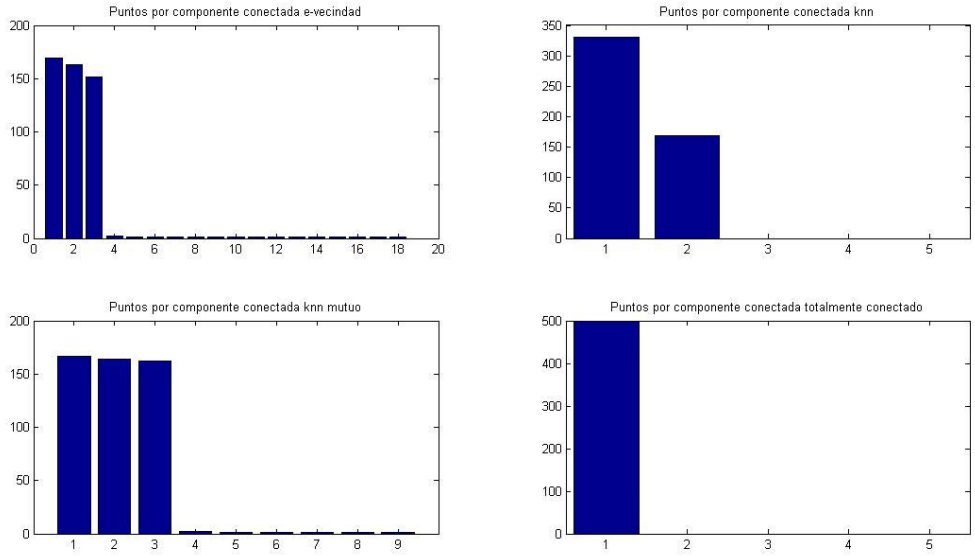

Figura 8. Puntos por componente conectada con cada tipo de grafo

Como recomendación general se sugiere la utilización del grafo de k-vecinos o totalmente conectado. K-vecinos tiene la ventaja de que la matriz W asociada es más de a bloques, lo que la vuelve computacionalmente más sencilla de manejar pero puede llegar a perder información relevante del grafo. Por el contrario, la matriz W del grafo totalmente conectado puede ser más compleja pero conserva toda la información asociada al grafo. Lo importante es que la cantidad de componentes conectadas del grafo sea menor o igual que la cantidad de clusters a clasificar y algunas características más que veremos en los siguientes puntos.

4.2. Cómputo de valores y vectores propios

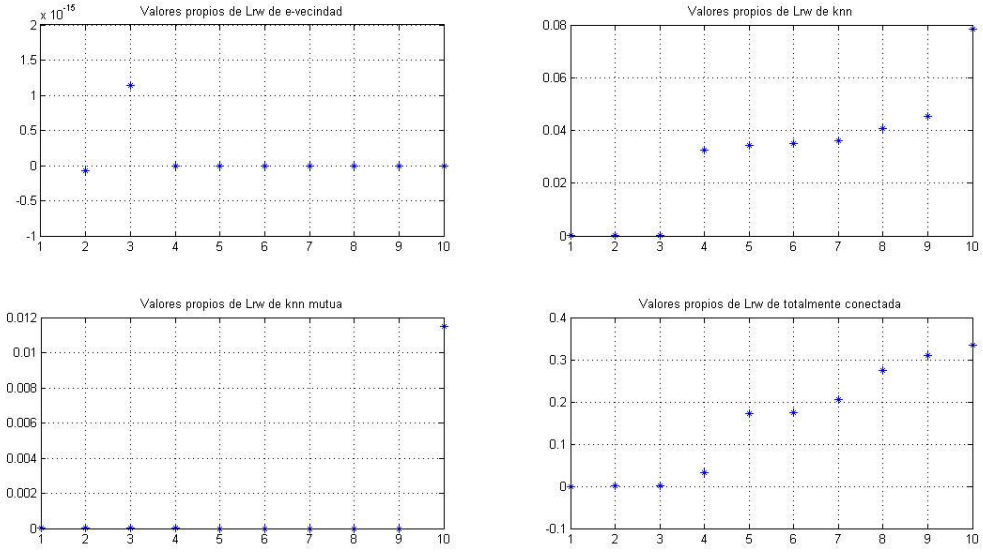

Figura 9. Valores propios para cada grafo

Como se muestra en la figura, para el caso simple que estamos analizando los tres valores propios más pequeños, tanto del grafo de k vecinos como del grafo totalmente conectado tienen valores muy cercanos a cero. Esto hace que usando esos grafos los tres vectores propios asociados nos den una partición correcta de los datos, alcanzando los resultados mostrados en las figura 10. Observar como los tres primeros vectores alcanzan para clasificar las tres gaussianas.

En cuanto al cálculo de los valores y vectores propios, hay algoritmos eficientes para matrices sparse cuya velocidad de convergencia depende del tamaño del salto de los valores propios, es decir $\gamma_k = |\lambda_k - \lambda_{k+1}|$, cuanto mayor sea el gap mayor será la velocidad de convergencia de los algoritmos.

Como se puede ver en las gráficas de la figura 9 de k-vecinos y totalmente conectado este gap es notorio a partir del tercer valor propio.

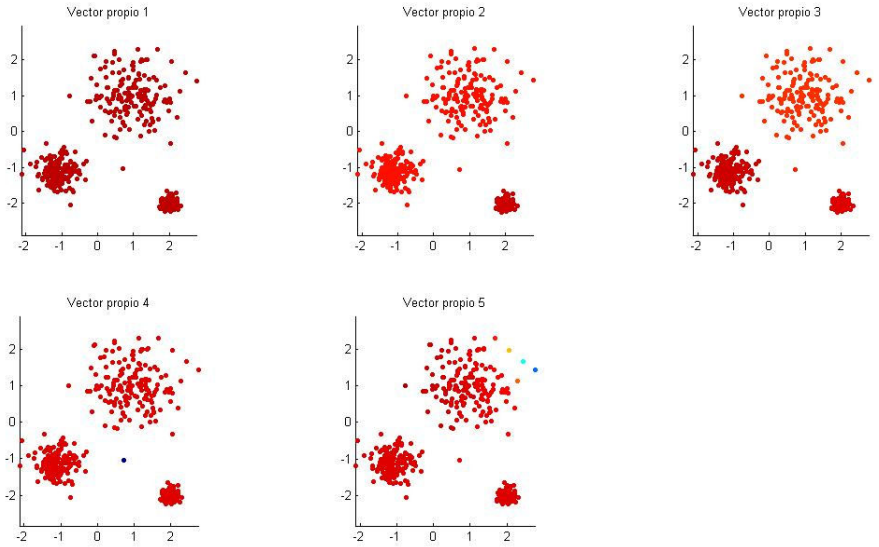

Figura 10. Separabilidad de los cinco primeros vectores propios (grafo totalmente conectado)

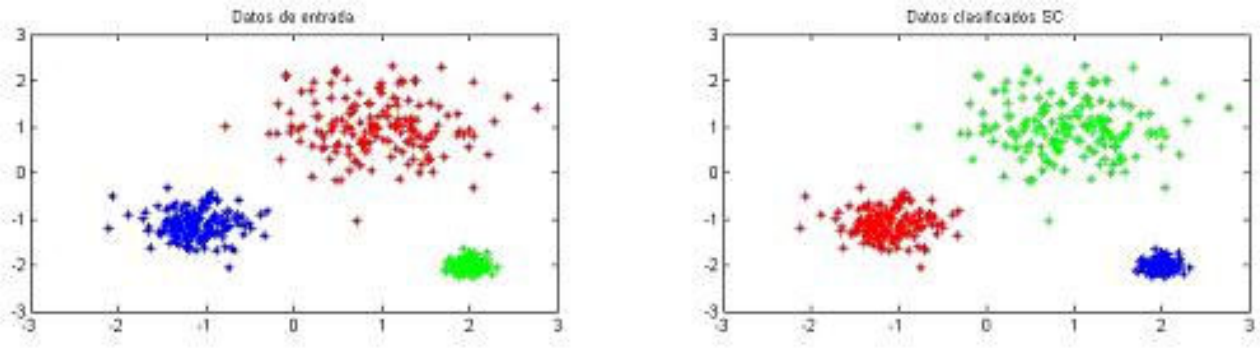

Figura 11. Izquierda: Datos. Derecha: Clasificación por Spectral Clustering

Para calcular los valores propios de la matriz Laplaciana, el programa utiliza la matriz Lsym por facilidad de cálculo. Sabemos por lo demostrado en el punto 3.2.2 (proposición 3) que los valores propios de Lsym son los mismos que los de Lrw y los vectores propios se convierten por la fórmula vista en dicha proposición. Para este ejemplo se utiliza la matriz Laplaciana normalizada L_{rw}, más adelante veremos por que se recomienda su utilización.

4.3. Parámetros de los grafos de similitud

Una vez elegido el tipo de grafo hay que elegir el parámetro de vecindad tanto ε ,σ como k. Desafortunadamente no hay ningún estudio teórico que defina cual es el mejor ε y k pero veremos más adelante algunas propuestas para la elección de los mismos.

La realidad es que hay muchos estudios teóricos relacionados con el tema que prueban la conectividad de grafos aleatorios cuando $n\to\infty$. Por ejemplo, es conocido que para n puntos, datos i.i.d de alguna densidad con soporte conexo en R^d, el grafo de k-vecinos y el de k-vecinos mutuo estará conectado si se elige k del orden de log(n) (e.g., Brito,Chavez, Quiroz and Yukich,1997). Argumentos similares muestran que el parámetro ε del grafo de vecindad-ε debe ser elegido como $(\log(n)/n)^d$ para garantizar conectividad en el límite (Penrose, 1999). Aunque estos resultados tengan significancia teórica, en la práctica no son realmente útiles para elegir los parámetros. Algunas recomendaciones prácticas son las siguientes.

Cuando trabajamos con k-vecinos el parámetro de conectividad debe de ser elegido tal que el grafo quede conectado, o por lo menos tenga menos componentes conectados que clusters a detectar. Para datos sintéticos, esto puede irse probando sistemáticamente pero cuando los grafos se vuelven más grandes una buena aproximación puede ser log(n).

Para la gráfica de k-vecinos mutua hay menos reglas prácticas. Recordemos que la ventaja de este grafo frente al de k-vecinos es que no une componentes de distintas densidades, por lo que para aprovechar esta característica, vamos a tener que permitirle más componentes desconectadas. Además el grafo de k-vecinos mutuo tiene menos conexiones que el de k-vecinos por lo que se recomienda la elección de un k significativamente mayor. De todas formas hay que tener idea si la no unión de regiones conectadas tiene sentido y para esto no hay ninguna regla probada que lo determine.

En el caso del grafo de vecindad-ε se sugiere elegir ε de forma que el grafo quede totalmente conectado. Para determinar esto hace falta elegir el ε tal que su magnitud coincida con la arista más larga del mínimo spanning tree del grafo totalmente conectado de los puntos. En el caso de que existan outliers o que los datos contengan muchos clusters densos alejados entre ellos, la elección del ε será mayor a la escala de los datos más importantes.

Para el grafo totalmente conectado, con función de similitud gaussiana el parámetro a setear es σ. La idea de este parámetro es que hay que asegurar que la mayoría de los puntos que están en la

vecindad con similitud significativamente mayor que 0 “no sean tan pocos pero tampoco muchos”. Para la elección del σ algunas reglas prácticas son; elegirlo como la distancia media de un punto a su k-ésimo vecino más cercano, donde k se elige como explicamos antes y otra forma podría ser elegir σ=ε. Estas son reglas que dependen fuertemente de los datos, lo que hace que pueden funcionar para algunos casos y para otros no.

En las siguientes figuras se muestran algunos ejemplos de la variación de sigma en la clasificación de los datos. Con σ=0.07, en la figura 12 vemos que la clasificación de los círculos concéntricos no es correcta. Este efecto se da al considerar un sigma muy pequeño que hace que tanto los datos muy cercanos como los lejanos se vuelvan disímiles y esto se ve reflejado en que los valores propios de la matriz L_{rw} son todos prácticamente cero. Esto significa que toma puntos aislados como clusters, no clasificando de forma correcta.

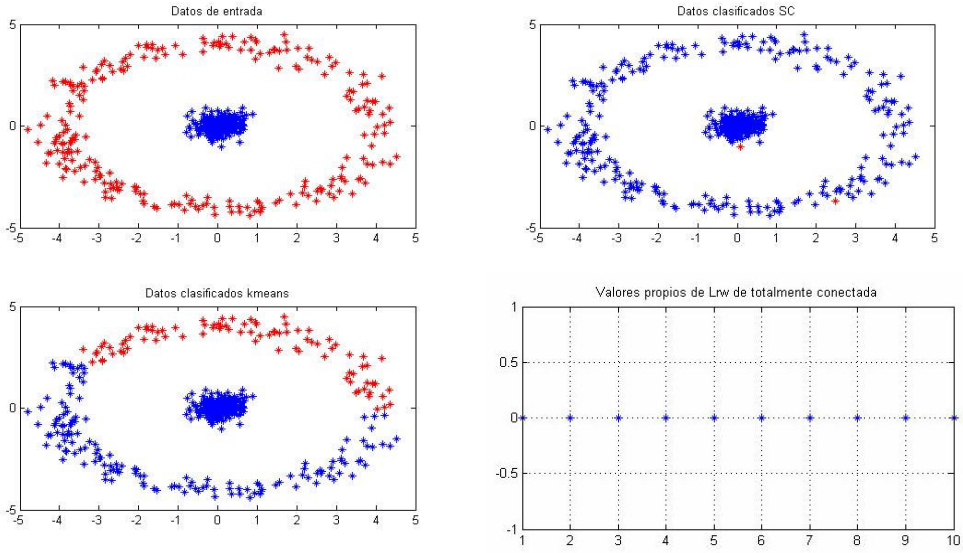

Figura 12. Datos y clasificación por SC y kmeans. Valores propios de la matriz Lrw, para σ=0.07

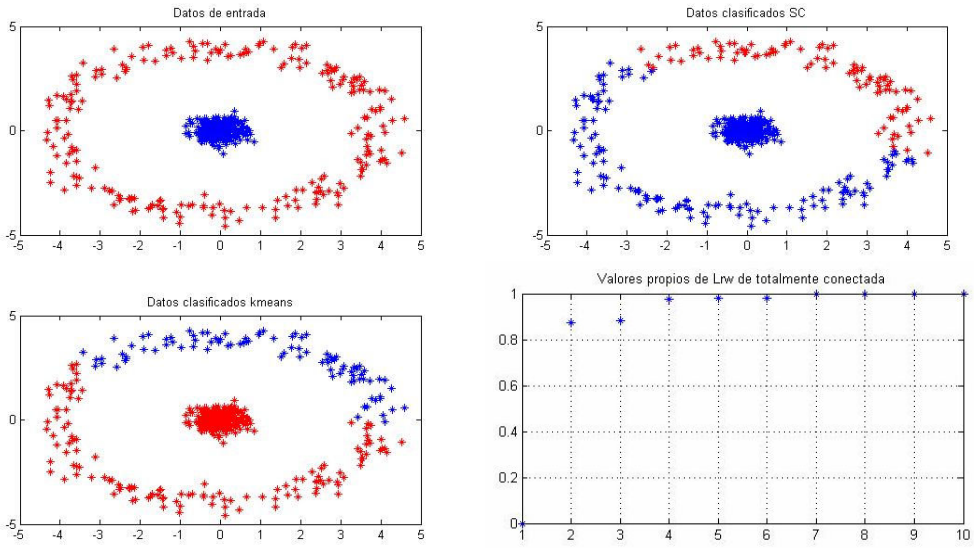

Figura 13. Datos y clasificación por SC y kmeans. Valores propios de la matriz Lrw, para σ=5

En la figura 13, se observa el resultado de la clasificación para un valor de sigma más grande (σ=5). Para este caso, la información de similitud no aporta nada extra a la clasificación. Todos los puntos son considerados cuasi símiles, lo que se nota en los valores propios de la matriz L_{rw}. Luego del primer valor propio cero hay un gap importante, esto quiere decir que el componente conectado principal es uno sólo (multiplicidad del valor propio cero), por esta razón la clasificación es muy similar al k-means y la aplicación del algoritmo de SC no mejora la performance del k-means.

En la siguiente figura se muestra un ejemplo con un valor de sigma adecuado, en el cual se puede observar que la clasificación es correcta y que los valores propios de la matriz presentan valores esperables.

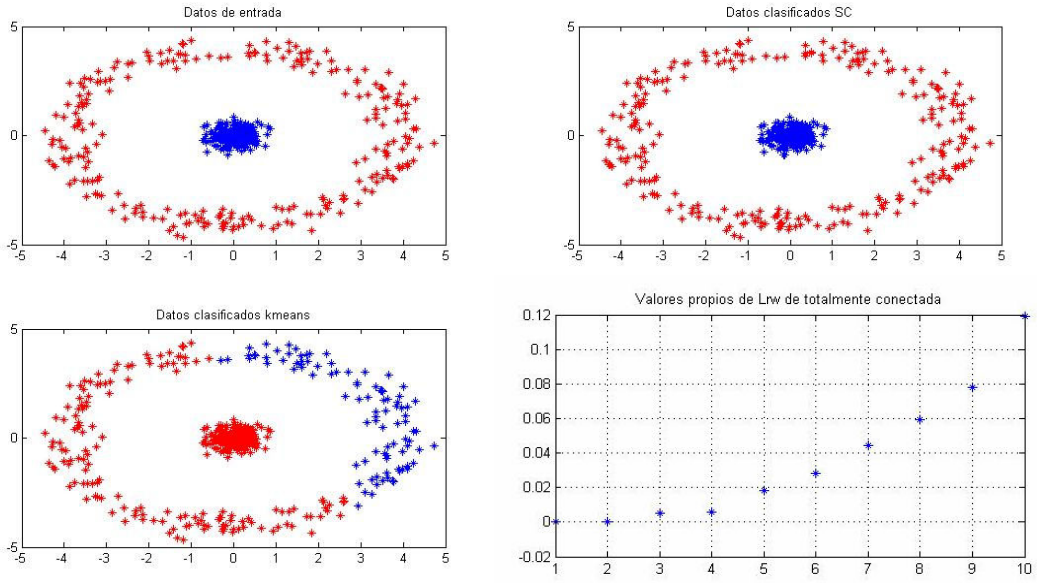

Figura 14. Datos y clasificación por SC y kmeans. Valores propios de la matriz Lrw, para σ=0.5

Una forma de automatizar la elección de σ, puede ser establecer un intervalo de variación del parámetro y ver que sigma provee la menor distorsión de clusters de las filas de la matriz de vectores propios. Esto aumenta significativamente el tiempo de cómputo y el rango de testeo de σ se debe setear manualmente. Sin embargo, hay algunas situaciones en que un valor de sigma fijo no puede resolver la clasificación. Este es el caso en que los datos incluyan diferentes estadísticas locales (distintas escalas) en el que puede no existir un único σ que resuelva el problema para todos los datos. Un ejemplo de este último comentario es el que se muestra en la figura 15.

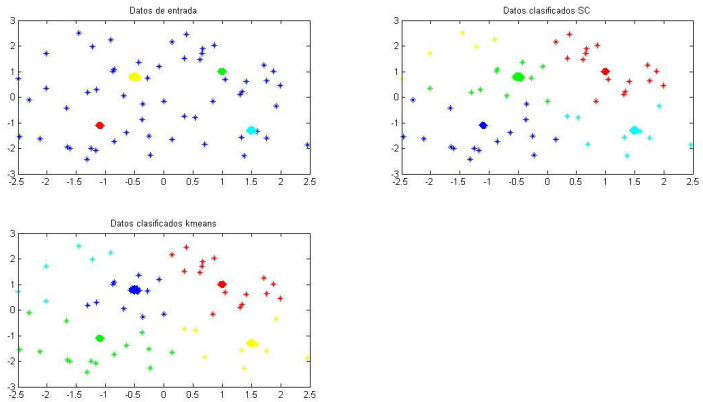

Figura 15. Datos y clasificación por SC y kmeans. Valores propios de la matriz Lrw, para σ=0.6

Al igual que en muchas simulaciones, el valor de sigma en la figura no resuelve bien la clasificación de este tipo de datos. No es particularidad de este grafo, sino que ningún otro tipo de grafos (k-vecinos ni vecindad-ε) resuelve la clasificación propuesta.

El problema es que según se consideren los puntos de "fondo" o los círculos pequeños densos del centro, el parámetro de similitud apropiado será significativamente diferente. Esto se resuelve con la propuesta de sigma auto ajustable en [2].

[2] propone que en vez de elegir un único parámetro de escala σ, se calcule un parámetro local de escala σ_i para cada punto s_i. La distancia de s_i a s_j vista por s_i es $d(s_i, s_j)/\sigma_i$ mientras que vista por s_j es $d(s_i, s_j)/\sigma_j$. Por lo cual la similitud entre ambos puntos se puede generalizar como $A_{ij} = \exp(-\frac{d^2(s_i, s_j)}{\sigma_i \sigma_j})$.

Usando un parámetro de escala específico para cada punto permite el autoajuste de las distancias acorde a la estadística local de los puntos vecinos i y j. La selección de la escala local debe ser realizada estudiando la estadística local de la vecindad del punto s_i. Una elección sencilla usada en los siguientes ejemplos es $\sigma_i = d(s_i, s_k)$, donde s_k es le k-ésimo vecino del punto.

La elección del valor de k es independiente de la escala y es función de la dimensión del espacio de datos. En [2] recomiendan usar k=7, en base de la experiencia, con datos sintéticos e imágenes.

En la siguiente figura, vemos como con el parámetro auto ajustable de vecindad local propuesto por [2] el ejemplo que no resolvía adecuadamente con σ fijo se resuelve.

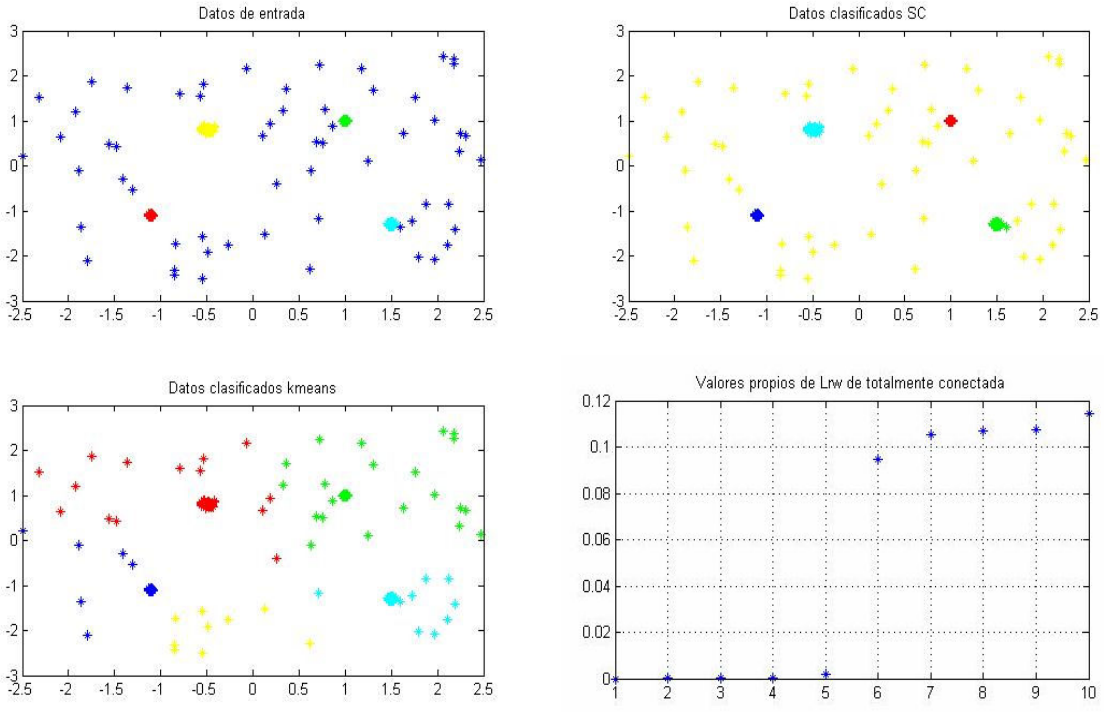

Figura 16. Datos y clasificación por SC y kmeans. Valores propios de la matriz Lrw, para σ autoadjustable

En la figura 16 se puede observar la correcta clasificación de los puntos. Se ve además en la gráfica de los valores propios de la matriz L_{rw} como los primeros cinco son valores muy cercanos a cero y luego hay un gap pronunciado entre estos y el resto. En la siguiente figura se muestra la separabilidad de los cinco vectores propios correspondientes.

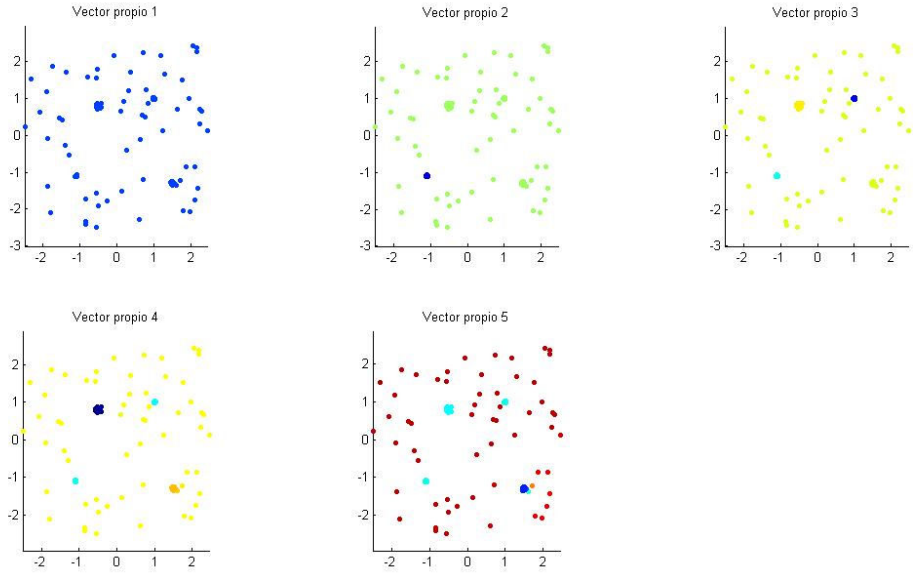

Figura 17. Separabilidad de los cinco primeros vectores propios (grafo totalmente conectado)

La propuesta de automatización en la elección de sigma, depende del caso particular de datos que estemos considerando, depende de la elección de la vecindad local, lo cual también requiere cierto conocimiento de los datos.

En la siguiente figura veremos un ejemplo donde con sigma fijo el algoritmo clasifica mejor que el con sigma auto ajustable, con los parámetros sugeridos en [2]. Hay que considerar además que este ejemplo no tiene multiescalas, es más las lunas son igualmente densas, simétricas y muy cercanas.

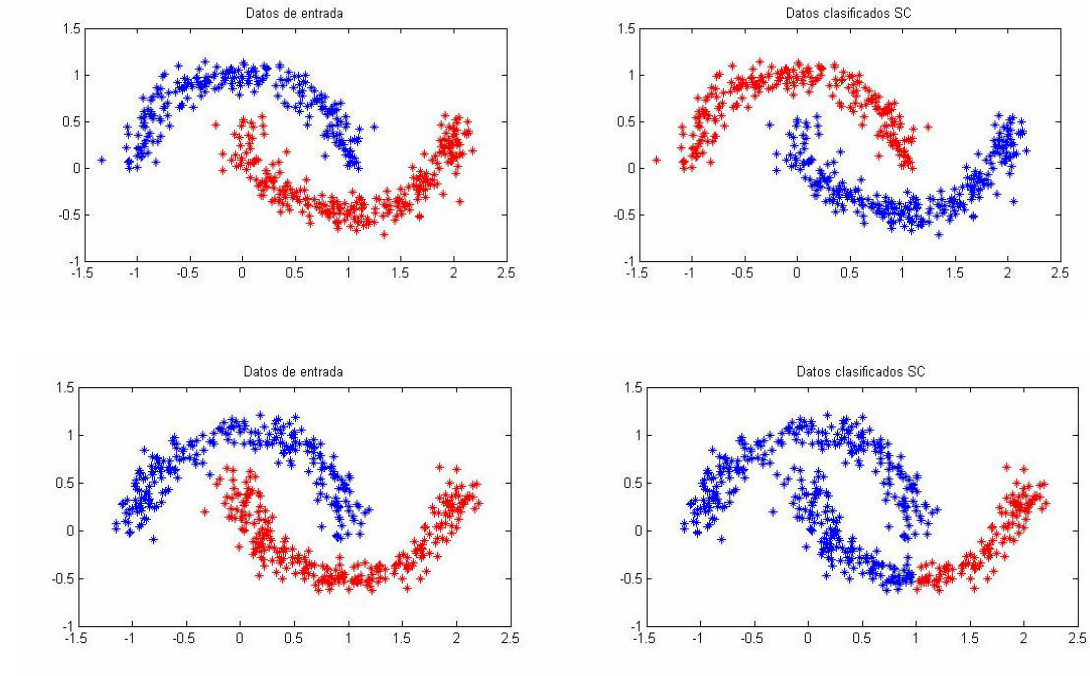

Figura 18. Arriba: Clasificación con σ fijo = 0.1. Abajo: Clasificación con sigma autoajustable (k=7)

En general, lo que hemos constatado es que esta técnica es sensible a los parámetros de las funciones de similitud y de la elección de los grafos. Desafortunadamente no hay una receta general

para todos los casos, sino que es importante ver algunas características de los grafos y la matriz Laplaciana para poder elegir el grafo que mejor resuelva un problema determinado. Esto requiere de una implementación supervisada conociendo en profundidad como funciona el algoritmo.

4.4. Número de clusters

La elección del número de clusters es un problema que aparece en todos los algoritmos de clustering. Una forma de ver el número de clusters especial para la técnica de SC, es la heurística de gap de valores propios, la cual puede ser usada para las tres Laplacianas. El objetivo será encontrar el número de clusters c que cumpla que $\lambda_1 \ldots . \lambda_c$ sean muy pequeños pero λ_{c+1} relativamente grande. Hay varias justificaciones para este procedimiento, una de las cuales está basada en la teoría de perturbaciones. La misma observa que en el caso ideal de c componentes conectados, el valor propio cero tendrá multiplicidad c, y habrá un gap con el primer valor propio $\lambda_{c+1}>0$.

Para ilustrar lo antedicho veamos la figura 19.

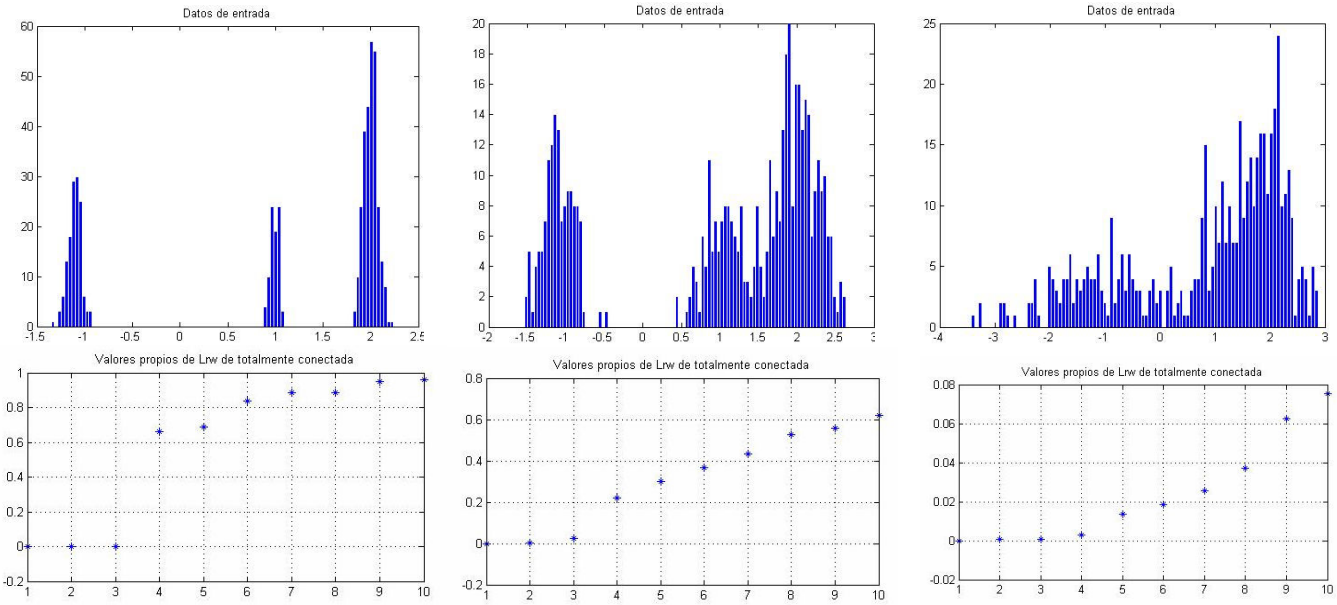

Figura 19. Tres grupos de datos y los primeros 10 valores propios asociados de Lrw

Como vemos en la figura 19, los datos son producidos por tres gaussianas a las cuales se les va agregando ruido. La primera fila muestra los histogramas de las gaussianas desde las más separadas hasta la situación con mayor ruido donde se confunden las tres. En la fila de abajo se muestran los valores de los diez primeros valores propios asociados a la matriz L_{rw} construida a partir del grafo totalmente conectado con sigma fijo. Como se puede observar en la primera gráfica, cuando los clusters están bien definidos (tres gaussianas con poco ruido) los tres primeros valores propios de la matriz son prácticamente cero y el salto entre el tercer y cuarto valor propio es pronunciado, esto indica que la heurística del gap funciona en estos casos. Veamos ahora los otros casos, a medida que vamos aumentando el ruido en las gaussianas, las mismas se empiezan a mezclar y los valores propios ya no presentan el gap pronunciado que presentaban antes. En estos casos esta heurística no es tan clara, este método funciona cuando los clusters de los datos están bien pronunciados, cuanto más solapamiento de clusters haya más complicado se vuelve identificar un gap claro entre el tercer y cuarto valor propio. Pero para el último caso "patológico", en el cual se solapen mucho los clusters, puede no haber algoritmo que lo resuelva correctamente, mirando los datos originales, donde están las tres gaussianas????.

Como conclusión el estimar la cantidad de clusters por el gap entre valores propios es útil cuando en los datos tenemos clusters pronunciados, pero en casos ambiguos dan resultados ambiguos también.

Otra consideración en este punto es que la elección de los parámetros de similitud del grafo y el número de clusters estan relacionados. Cuando la elección del parámetro es tal que deja c componentes conectados, la elección de c clusters puede ser válida. Cuando todo el grafo está conectado la relación entre el número de clusters puede no ser clara, como vimos en el ejemplo.

Otro método de automatización planteado por [2] para la elección de la cantidad de clusters, se basa en la estructura de los vectores propios. La idea es que en el caso ideal donde hay c clusters, la matriz L es una matriz con c bloques donde los valores propios de esos bloques, son los valores propios de L y los vectores propios asociados tienen valores de 1 en las componentes pertenecientes a cada bloque y cero en el resto. Como la cantidad de bloques es c, el valor propio cero tendrá multiplicidad c y habrán c vectores propios asociados constantes ortogonales entre si.

Como en el caso real estos vectores propios no son exactamente constantes, dado que la matriz tampoco es idealmente con bloques separados, el espacio generado por los vectores propios está afectado por alguna rotación R. Lo que en [2] se plantea es encontrar esta rotación R tal que minimice la distancia entre los vectores propios obtenidos por el algoritmo y los de la base canónica.

De esta forma se construyen una función "costo" a minimizar por gradiente descendente que da el mínimo apartamiento a la base canónica y luego se elige el c que de ese mínimo. Así se consigue calcular el número c de clusters. Tal vez se pueda clasificar con los vectores rotados ya que están muy cerca de la base canónica y se podrá elegir la máxima coordenada para cada punto sin tener que utilizar el algoritmo de kmeans. Por más detalles de la implementación ver [2].

De hecho el uso de kmeans en el último paso de los algoritmos es también un tema en discusión, se podría utilizar otro algoritmo en su lugar. La explicación es que es un algoritmo sencillo que puede clasificar correctamente si los datos están bien separados en distintos clusters, pero de hecho otros autores aplican otras técnicas para construir la solución final de la representación de los datos. En todos estos ejemplos presentados se utiliza k-means en la etapa final de los algoritmos.

4.5. Elección del grafo Laplaciano

Hasta ahora no hemos comentado nada acerca de la elección del tipo de Laplaciano a utilizar. Un tema importante a considerar en el algoritmo de SC es cual de los tres Laplacianos utilizar a la hora del cálculo de los vectores propios. En el caso que la gráfica de similitud sea muy regular, es decir que la mayoría de los vértices tengan el mismo grado, las tres Laplacianas serán similares y su performance por ende también será similar.

La primera razón a nuestro juicio más importante de por que utilizar Laplacianas normalizadas (L_{rw} y L_{sym}) y no una no normalizada es la siguiente: los objetivos de los algoritmos de clustering son dos:

1. Encontrar una partición que minimice la similitud entre clusters. Es decir, que los clusters que clasifique sean lo más disímiles posibles entre ellos.
2. Encontrar una partición que maximice la similitud dentro de los clusters. Es decir, que no sólo separe clusters disímiles sino que los que una sean realmente similares.

Por lo visto antes, tanto RatioCut como Ncut implementan directamente el primer objetivo planteado, ya que incorporan en la función objetivo a minimizar $cut(A, \overline{A})$, la cual minimiza la similitud entre A y complemento de A. Pero para el segundo objetivo ambos algoritmos tienen

distinto comportamiento. Sea $W(A,A) = W(A,V) - W(A,A) = vol(A) - cut(A,\overline{A})$, para maximizar la similitud intracluster tenemos que $cut(A,\overline{A})$ sea pequeño y que vol(A) grande. Esto último es exactamente lo que implementamos minimizando con el criterio Ncut, por lo que además Ncut cumple el segundo objetivo también.

Por el contrario, RatioCut busca maximizar $|A|$ y $|\overline{A}|$ en vez de vol(A) y en realidad la cantidad de puntos no siempre tiene que ver con la similitud entre los mismos. Por lo tanto minizando RatioCut no cumplimos los dos objetivos propuestos sino sólo el 1. Por esta razón la relajación del criterio de RatioCut usada para L no normalizado tampoco buscará el segundo objetivo planteado.

Por lo tanto es importante usar los grafos Laplacianos normalizados (L_{rw} y L_{sym}) ya que buscan los 2 objetivos impuestos en la clasificación, mientras que la no normalizada busca sólo el primero de ellos.

Sumado a esto, hay otra razón por la cual utilizar los grafos normalizados y es que para cantidades grandes de datos , la consistencia de los grafos normalizados L_{rw} y L_{sym} está matemáticamente probada, mientras que para grafos no normalizados en algunas situaciones falla la convergencia. Podemos ver el ejemplo sintético en la siguiente figura.

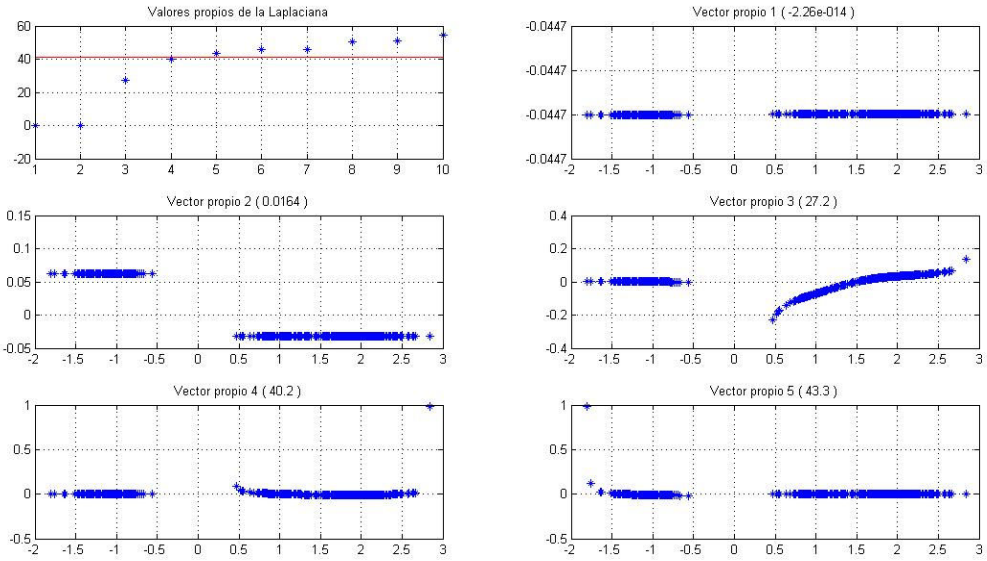

Figura 20. Valores y vectores propios para L no normalizado con σ=0.5

En la figura los datos son las tres gaussianas unidimensionales espaciadas usadas en el punto anterior. Se utilizó un grafo totalmente conectado con σ=0.5 pero construyendo el grafo laplaciano no normalizado. Como vemos los primeros tres vectores propios logran discriminar las tres gaussianas. En este caso esos tres valores propios son menores que una línea roja que indica el mínimo grado de los vértices del grafo. Veamos en la figura 21, el comportamiento de los vectores propios si algunos de los tres primeros valores propios queda por debajo de este mínimo.

El caso es igual al anterior con la diferencia que el valor de σ es 5.

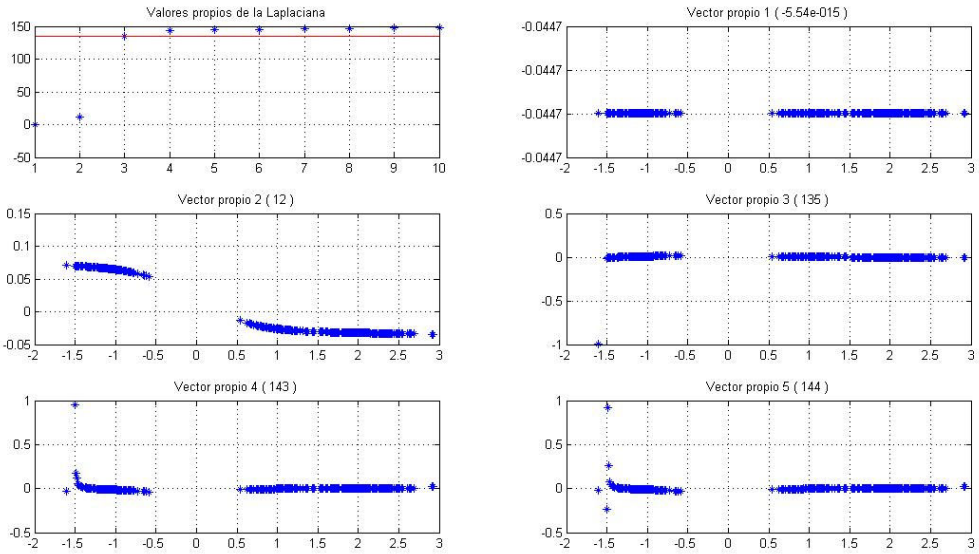

Figura 21. Valores y vectores propios para L no normalizado con σ=5

En este caso los vectores propios correspondientes a valores propios mayores que el mínimo grado de los vértices, tienen formas de diracs y como el tercer vector propio entra dentro de esta categoría la clasificación no es correcta. Por lo tanto nos deberíamos asegurar que la cantidad de valores propios igual a la cantidad de clusters, sea menor al mínimo grado de los vértices del grafo.

Con valores de n más grandes, todos los vectores propios son deltas de diracs por lo cual no son apropiados para la clasificación en clusters de los datos.

Esto sólo sucede con los valores propios de L, no sucede ni con L_{rw} ni con L_{sym} por lo que es otra de las razonas por las cuales es mejor utilizar el grafo Laplaciano normalizado.

Respecto al Laplaciano normalizado a usar, la respuesta es L_{rw} y la justificación es bastante simple. Si pensamos en que los vectores propios de L_{rw} son los conjuntos indicatrices de cluster y que los de L_{sym} están multiplicados por un factor dependiente del grado de los vértices ($D^{1/2}$). Si tenemos puntos cuyos grados sean realmente pequeños, por más que el vector propio sea muy parecido al vector indicatriz del cluster, se va a ver alterado por un factor pequeño que puede alterar la clasificación de ese punto por el algoritmo k-means.

Por las razones anteriormente expuestas en general se utilizó para todos los ejemplos L_{rw}.

5. Simulaciones

En el punto anterior hemos discutido al detalle los diferentes parámetros seteables del algoritmo SC. En este punto vamos a comparar la performance del mismo versus la performance del algoritmo k-means en distintos casos sintéticos.

En la siguiente figura se muestran dos líneas horizontales, separadas por una distancia de uno con diferentes valores de ruido gaussiano. Compararemos para este ejemplo la perfomance del algoritmo de clasificación de k-means y el SC con sigma autoajustable y Laplaciana L_{rw}.

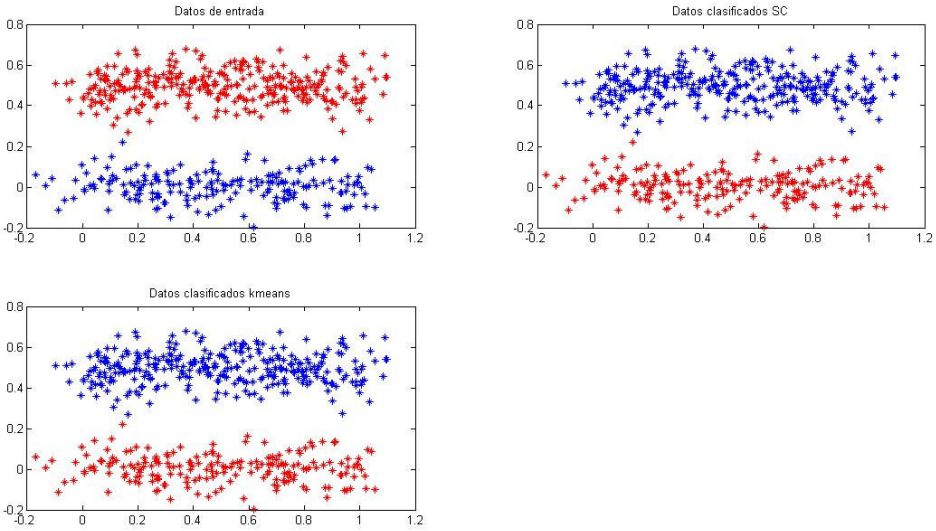

Figura 22- Clasificación por SC y K-means para un ruido gaussiano de varianza 0.05

Como se observa en la figura ambos algoritmos clasifican bien con ruido bajo, si bien el algoritmo k-means al ser un algoritmo iterativo depende de la semilla de inicialización que se le pasa al mismo. En el siguiente ejemplo se aumenta el ruido y el algoritmo SC clasifica mejor que el de k-means.

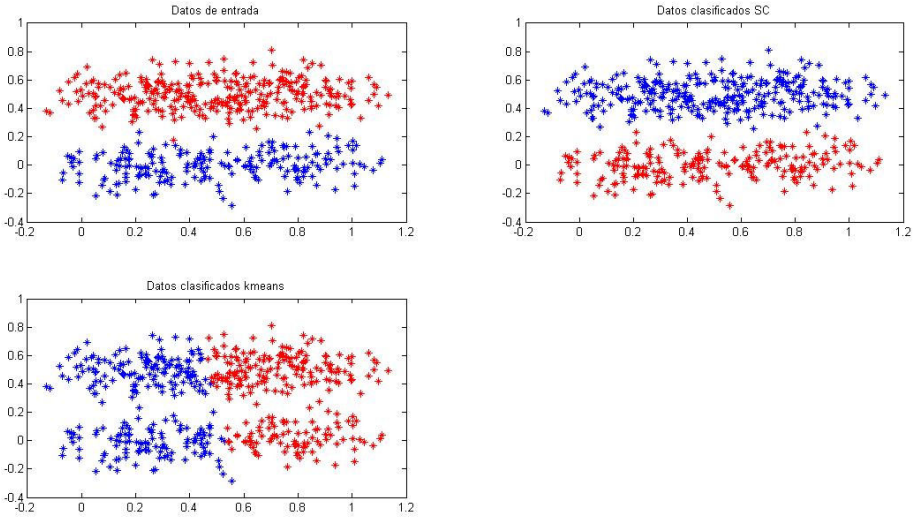

Figura 23- Clasificación por SC y K-means para un ruido gaussiano de varianza 0.01

Otros ejemplos sintéticos donde el k-means no funciona bien y el SC resuelve la clasificación correctamente son por ejemplo el caso de las dos lunas, figuras alargadas y casos como por ejemplo círculos concéntricos. En la siguiente figura se muestra la clasificación de las dos lunas con sigma=0.1 para el Laplaciono normalizado totalmente conectado y círculos concéntricos en la figura 25.

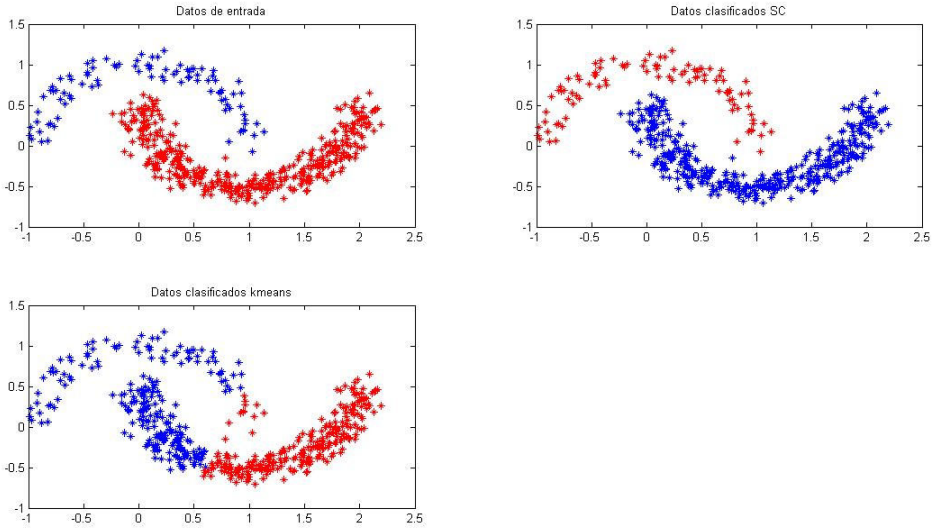

Figura 24- Clasificación por SC con sigma 0.1 y K-means

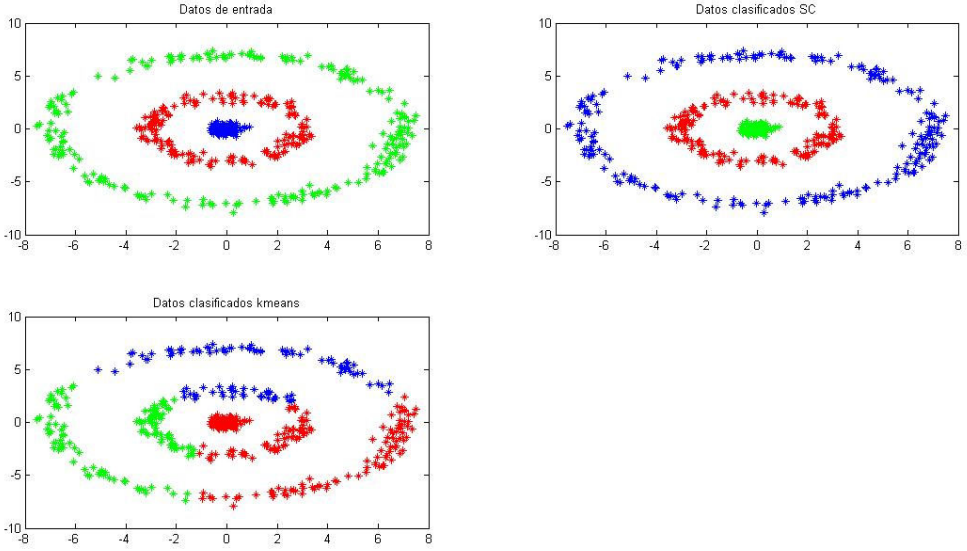

Figura 25- Clasificación por SC con sigma autoajustable y K-means

En general Spectral Clustering resuelve algunos problemas que no logra resolver el algoritmo k-means por si solo. Otras veces ambas clasificaciones fallan. Para la clasificación de puntos de distintas densidades con el de la figura 16, sin duda que el mejor algoritmo es SC con sigma autoadjustable pero también hay que considerar que la forma de que SC funcione correctamente es setear adecuadamente todos los parámetros involucrados y muchas veces no es tarea sencilla.

6. Conclusiones

Spectral Clustering puede lograr una correcta clasificación en casos que kmeans no es capaz de resolver. Es un algoritmo de simple implementación pero que muchas veces deja dudas de su funcionamiento.

Otra ventaja que presenta este algoritmo frente a kmeans es que no es iterativo. Por esta razón no existe incertidumbre acerca de la inicialización del mismo y el tiempo de cómputo puede ser menor.

La principal desventaja de esta técnica es que la solución alcanzada puede diferir significativamente de la solución real. Esto se debe a que el algoritmo implementa condiciones de relajación sobre el problema original para que la resolución del problema sea alcanzable.

Como gran desventaja presenta una fuerte dependencia de la elección de los parámetros respecto a los datos. Aun está lejos de ser un algoritmo tipo caja negra, por el contrario requiere de un profundo conocimiento de la técnica para obtener resultados confiables.

El automatizar la elección de los parámetros (como el sigma auto ajustable y la elección de la cantidad de clusters) como otros temas, son puntos abiertos de discusión en la actualidad.

En nuestra opinión, se trata de un tema interesante el cual disfrutamos estudiar. El hecho de que queden cosas sin explicación formal da una motivación adicional.

7. Referencias

[1] Ulrike von Luxburg. *A Tutorial on Spectral Clustering.* Versión original en línea www.springer.com.

[2] Lihi Zelnik – Manor Pietro Perona. *Self-Tuning Spectral Clustering.* Versión original en línea www.vision.caltech.edu/lihi/Demos/SelfTuningClustering.html.

[3] Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation.