PROYECTO DETECCIÓN DE CLUSTERS INTRODUCCIÓN AL RECONOCIMIENTO DE PATRONES

Diego Introini 4.379.988-6

Daniel Lena 4.870.247-2

INDICE

Introducción

Partimos de una base de datos de consumos mensuales de UTE, cuyos clientes se encuentran etiquetados dentro de algún rubro (etiquetas que son muy diversas, incluso dentro de un mismo rubro, ya que fueron generadas a criterio de cada funcionario sin ninguna clasificación global definida). Nuestro trabajo consistirá mediante dos técnicas distintas de clustering, intentar encontrar alguna correlación entre estas etiquetas y los clusters encontrados por los algoritmos. Para ello comenzaremos con un pequeño marco teórico orientado a cada una de las técnicas (K-means y Spectral Clustering), luego compararemos mediante la aplicación de los algoritmos a distintas bases de datos sintéticos la performance de ambas técnicas, para luego obtener alguna conclusión sobre sus diferencias (ventajas y desventajas) de cada una.

Finalmente, aplicaremos ambas técnicas a los datos extraídos y clasificados según las etiquetas más genéricas que obtengamos, para analizar los clusters que obtengamos y buscar alguna correlación entre estos y las etiquetas de los datos en cada uno de ellos, o algun comportamiento similar entre clusters, etc.

Marco Teórico

El proceso de clustering consiste en particionar los datos procurando que los objetos de un mismo cluster(grupo) tengan una similaridad alta, y baja con elementos de otros clusters. La medida de similaridad está basada en las características que describen los objetos. Por lo general se utilizan distancias: distancia euclídea, de Manhatan, de Mahalanobis, etc.

Clustering es una técnica más de Aprendizaje Automático, en la que el aprendizaje realizado es no supervisado. Desde un punto de vista práctico, el clustering juega un papel muy importante en aplicaciones de minería de datos, tales como exploración de datos científicos, recuperación de la información y minería de texto, aplicaciones sobre bases de datos espaciales (tales como GIS o datos procedentes de astronomía), aplicaciones web, marketing, diagnóstico médico, análisis de ADN en biología computacional, y muchas otras.

La mayoría de los métodos de clustering son de dos tipos:

- Particionales: buscan una partición en cierta cantidad de clases, que optimiza una función objetivo elegida a priori.

- Jerárquicos: jerarquía de particiones “anidadas”, cada nivel de la jerarquía es en sí misma una partición, obtenida por unión de clusters de la jerarquía inferior.

En particular utilizaremos dos algoritmos de clusters particionales: K-means y Spectral-Clustering.

K-means

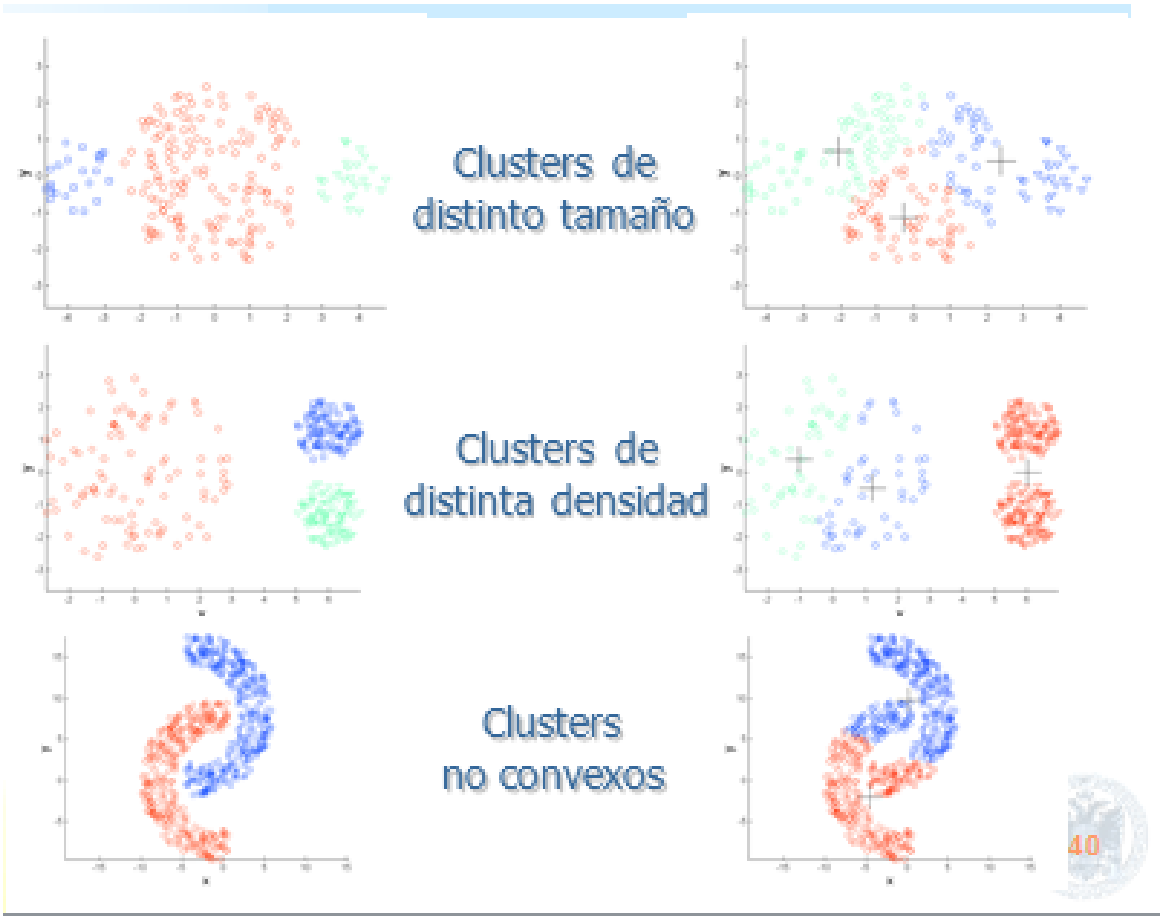
Creado por MacQueen en 1967, K-means es el algoritmo de clustering más utilizado y conocido debido a su eficacia y su muy simple aplicación. Sigue un proceso simple de clasificación de un conjunto de objetos en un determinado número K de clusteres, K determinado a priori.
Se le llama K-means porque cada uno de los cluster es representado por la media (o media ponderada) de sus puntos, es decir, por su centroide. La representación por centroides presenta la ventaja de tener un significado gráfico y estadístico inmediato. Se basa en la minimización de la distancia interna (suma de las distancias de los patrones asignados a un agrupamiento a su centroide). En realidad, se minimiza la suma de las distancias al cuadrado de cada patrón al centroide de su cluster.
El algoritmo del K-means se realiza en 4 etapas:

- **Etapa 1:** Elegir aleatoriamente K objetos formando así los K clusters iniciales. Para cada cluster, el valor inicial del centro es el valor del objeto asociado.
- **Etapa 2:** Cada objeto es reasignado al cluster de centroide más próximo a él según una medida de distancia, (habitualmente la medida euclidiana).
- **Etapa 3:** Una vez que todos los objetos son colocados, recalcular los centros de K cluster(los baricentros).
- **Etapa 4:** Repetir las etapas 2 y 3 hasta que no se hagan más reasignaciones.

Este algoritmo es simple, eficaz y solo depende de 1 parámetro, pero presenta las siguientes desventajas:

- No se garantiza el obtener la solución óptima. En efecto, el algoritmo es muy sensible a la elección aleatoria de los K centros iniciales. Esta es la razón por la que se utiliza el algoritmo del K-means numerosas veces sobre un mismo conjunto de datos para intentar minimizar este efecto, sabiendo que a centros iniciales lo más espaciados posibles dan mejores resultados.
- La necesidad de inicializar el número de prototipos al principio de la ejecución. Esto perjudica la eficacia del algoritmo ya que en la práctica, no se conoce a priori el número de clusters final. En general se debe elegir una cantidad de clusters tal que al agregar más clusters no se mejora la solución.
- Es susceptible a "outliers" porque distorsionan la distribución de los datos.
- Es importante normalizar los datos.

Además este algoritmo presenta dificultades con ciertas formas de soluciones, como se muestra en la siguiente imagen:



Spectral Clustering

Spectral clustering trata sobre un conjunto de técnicas que intentan hallar las clases desconocidas sobre un conjunto de datos (agruparlos en clusters), a través del uso de la similitud entre las muestras. Es decir, utilizando grafos de aristas ponderadas (grafo de similitud), donde la similitud de dos patrones dados j, i, viene dado por el valor que pondera la arista entre ambos patrones, y define que tan "similares" son. Un ejemplo común de función de similaridad es:

$$\exp(-\|x_i - x_j\|^2/(2\sigma^2))$$

(Además será la que utilizaremos en los algoritmos que construimos para este trabajo), aunque realmente la función de similaridad no es considerada como parte del problema teórico del clustering, al momento de llevarlo a problemas prácticos, puede depender muchísimo nuestro resultado según la función que se utilice. Matemáticamente la única restricción que le ponemos a la función de similaridad es que sea simétrica: $S(x_i,x_j)= S(x_j,x_i)$, y que $S(x_i,x_j) \geq 0$=Escriba aquí la ecuación. para todo par de patrones x_i,x_j de nuestro grafo. Obviamente, también le pedimos que represente de alguna forma la similitud entre cada par de datos.

Luego el problema de clustering, puesto en términos de un grafo de similitud, es la de obtener una partición del grafo de similaridad, en la cual, los patrones dentro de cada cluster tengan entre ellos altos valores ponderados en su representación dentro del grafo (lo que implica que son "similares"), mientras que los valores ponderados entre vértices de distintos cluster sea lo más bajo posible, lo que nos dice que los patrones de distintos clusters son lo suficientemente "distintos".

Existen varios criterios al momento de crear el grafo de similaridad:

The epsilon-neighborhood graph: En este grafo conectamos únicamente los pares de patrones que entre ellos su función de similaridad es mayor que cierta barrera ε que imponemos: $S(x_i,x_j) \geq \varepsilon$.

The k-nearest neighborhood: Con este criterio, se conecta al patrón xi, solo a los k patrones que tengan mayor similaridad con él. El problema de este criterio es que nuestro grafo debe ser no dirigido, esto se resuelve fácil con dos opciones: conectamos x_i con x_j solo si x_i esta en los k vecinos mas cercanos a x_j y viceversa, o se conecta todo el grafo con el criterio inicial y hacemos que todo arista sea bidireccional (esto genera que un vértice pueda estar conectado con k o mas vértices, mientras que la opción anterior genera que un vértice esté conectado a lo sumo con k vértices).

The fully connected graph: Solo conectamos todos los pares de vértices x_i, x_j ponderando su arista con cierta función $S(x_i,x_j)$ de similaridad.

Las técnicas de spectral clustering utilizan, como herramienta principal la matriz Laplaciana, que obtenemos a partir del grafo de similaridad, y que será muy útil para el algoritmo de spectral gracias a las propiedades de este tipo de matriz.

Definimos la unnormalized graph Laplacian matrix como:

$$L = D - W$$

Donde W es la matriz que tiene en la entrada i, j el valor ponderado de la arista para los patrones x_i, x_j en el grafo de similitud. W, por definición de la función de similaridad, es simetrica. La matriz L cumple las siguientes propiedades:

1) Para todo vector f que pertenece a Rn:
$$f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} wij\left(f_i - f_j\right)^2$$
2) L es simétrica y semi-definida positiva.
3) El menor valor propio de L es 0 que corresponde al vector **1**.
4) L tiene n valores propios reales positivos.
5) La multiplicidad k del valor propio 0 en L es igual al número de componentes conexas del grafo de similaridad.

Omitimos las demostraciones de estas propiedades que están en las notas "A Tutorial on Spectral Clustering".

Para este trabajo utilizaremos la técnica de espectral clustering que utiliza unnormalized graph Laplacian matrix, aunque existen otras matrices laplacianas definidas como, normalized graph Laplacians:

$$L_{sym} = D^{\frac{1}{2}}LD^{\frac{1}{2}}$$
$$L_{rw} = D^{-1}L$$

Estas matrices laplacianas también tienen sus propiedades análogas a las de unnormalized graph Laplacian matrix:

1) Para todo vector f que pertenece a Rn:
$$f'L_{Sym}f = \frac{1}{2}\sum_{i,j=1}^{n} wij\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2$$

2) λ es un valor propio de L_{rw} de vector propio u si y solo si es un valor propio de L_{sym} de vector propio w = $D^{\frac{1}{2}}$u.

3) λ es un valor propio de L_{rw} de vector propio u si y solo si λ y u resuelven el problema de vectores propios $Lu = \lambda Du$.

4) 0 es un valor propio de L_{rw} del vector constante **1** como vector propio. 0 es un valor propio de L_{sym} de vector propio w = $D^{\frac{1}{2}}$**1**.

5) L_{sym} y L_{rw} son positivas semi-definidas y tienen n valores propios reales positivos.

Unnormalized Spectral Clustering

Tenemos como entradas los datos y la cantidad k de clusters a generar.
Construimos el grafo de similaridad a partir de los datos, con alguno de los criterios descritos anteriormente. Obtenemos la matriz de pesos W, a partir de ella obtenemos D (como también esta definida anteriormente) y luego creamos la matriz Laplaciana L.
Computamos los primeros k vectores propios de L, y generamos la matriz U^{nxk} con los k vectores propios de L, u_i como columnas.
Luego utilizamos k-means para clasificar los n puntos y_i, siendo el i-esimo vector fila de la matriz U, en k clusters $C_1 \dots \dots C_k$.
Como salida obtenemos la clasificación de los datos, donde el dato i, pertenece al cluster j, si $y_i \in C_j$.

Normalized Spectral Clustering

Se pide como entrada los mismos parámetros que para Unnormalized.
Construimos el grafo de similaridad a partir de los datos, con alguno de los criterios descritos anteriormente y obtenemos L de la misma igual forma que en el anterior algoritmo.
Calculamos la matriz laplaciana normalizada L_{rw} y computamos sus primeros k vectores propios u_i. Luego generamos la matriz U^{nxk} con los k vectores propios como columnas.
Luego utilizamos k-means para clasificar los n puntos y_i, siendo el i-esimo vector fila de la matriz U, en k clusters $C_1 \dots \dots C_k$.
Como salida obtenemos la clasificación de los datos, donde el dato i, pertenece al cluster j, si $y_i \in C_j$.

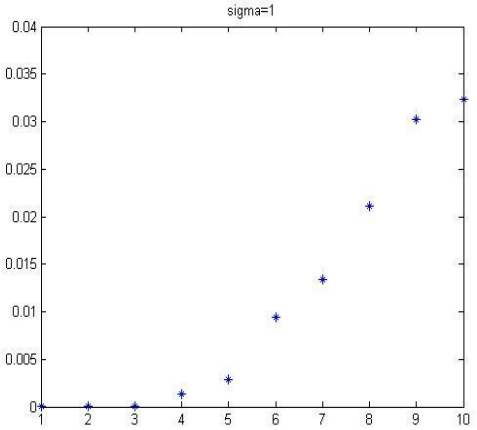
Estudio de valores propios Spectral Clustering

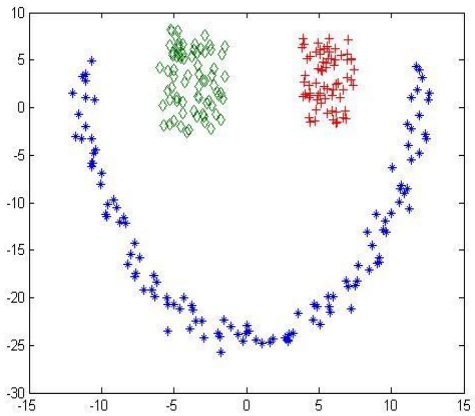
El mayor costo computacional de spectral, es sin lugar a dudas, el cálculo de los primeros i vectores propios de la Matriz Laplaciana. Dado que nuestra matriz nos puede quedar de dimensiones gigantes, el computo de estos vectores propios no se hace tan simple, en el caso de utilizar el grafo completamente conectado, esto se hace extremadamente costoso e ineficiente. Para resolver esto se suelten utilizar cualquiera de los otros dos métodos (la franja ϵ o el de los k vecinos más cercanos), ya que, al computar sus respectivas Laplacianas, obtendremos matrices sparsas para los cuales existen métodos eficientes para hallar sus vectores propios. Estos algoritmos dependen fuertemente de los saltos gap entre vectores propios, es decir, cuanto mayor sea $y_k = |\lambda_k - \lambda_{k+1}|$, mas rápido encontraran las soluciones de estos vectores.

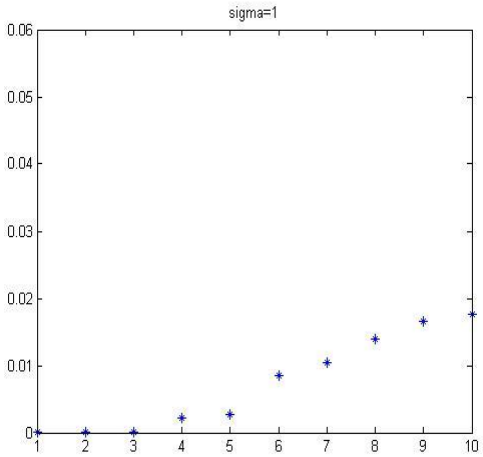
Es fácil notar, que en la situación ideal de que tengamos k componentes disconexas en nuestro grafo (que se traduce en por lo menos k clusters), el valor propio 0 tendrá una multiplicidad k en los valores propios de la matriz Laplaciana.

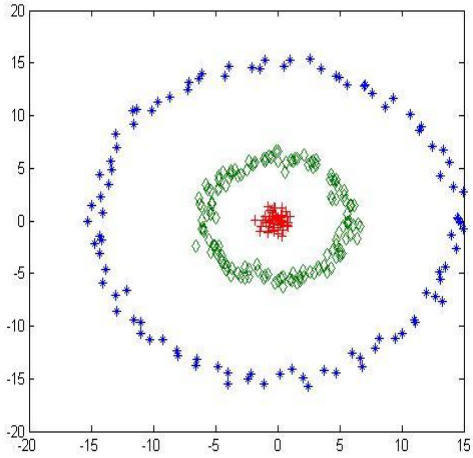
A continuación se muestran los primeros 10 vectores propios para dos distintas bases de datos sinteticos:

Valores Propios: Datos y clasificación:









En las figuras anteriores, aplicamos SC a ambas bases de datos, utilizando el grafo de los k vecino más cercanos (tomando k como uno de los criterios descritos más adelante) y σ =1. Utilizamos este valor de σ ya que nos resultó el más apropiado para el estudio de los valores propios, pero más adelante veremos que no es simple la elección del mismo y como puede implicar un gran cambio en el comportamiento de los valores propios, y por lo tanto afectar totalmente el resultado del algoritmo.

Criterios para definir el número de clusters para Spectral Clustering

En las figuras anteriores, se ve fácilmente que la multiplicidad del valor propio 0 es 3 para cada base de datos, en las cuales tenemos 3 clusters definidos. En los casos donde, nuestra cantidad de clusters está bien definida, de las graficas anteriores, podemos deducir que uno de los criterios a tomar a la hora de decidir en cuantos clusters particionar los datos, puede ser la multiplicidad del valor propio 0. Esto no ocurre normalmente cuando nos enfrentamos a datos reales, por lo que, si usamos este criterio, es probable que no consideremos clusters que no están tan definidos, pero aun así existen.

Existe otro criterio (el cual utilizaremos al aplicar SC a los datos reales) es la heurística en los saltos de los valores propios, en donde el objetivo es elegir k, de forma que todos los valores propios $\lambda_1 \ldots\ldots \lambda_k$ sean muy chicos y que λ_{k+1} sea relativamente mayor. Este criterio es útil ya que, llevado a la practica, la multiplicidad del valor propio 0 nos indicaría la cantidad de clusters bien definidos, la cantidad de valores propios cercanos a 0, la cantidad de clusters no tan definidos pero si existentes, y nos detenemos al ver un salto en los valores propios ya que esto implica que un cluster generado a partir de ese vector propio no cae en ninguno de los criterios anteriores.

Estimación de parámetros Spectral Clustering

Plantearemos técnicas para determinar el k, σ y ϵ óptimos frente a los distintos criterios sobre la construcción del grafo de similaridad.

Cuando trabajamos con k-vecinos o con el criterio de una franja ϵ, nuestro objetivo es que el grafo resultante sea conexo, o por lo menos que la cantidad de componentes conexas del grafo sea menor que la cantidad de clusters final a identificar.

Para el caso de los k-vecinos, si nuestro grafo es relativamente grande, es conocido que una buena aproximación a un grafo conexo la obtendremos si consideramos k como $log(n)$, si los datos provienen de alguna densidad con soporte conexo en R^d aunque en el caso de una base de datos sintéticos (pocos datos), en donde las pruebas no son costosas, se puede probar con una variedad de k. Estas consideraciones no son muy útiles a la hora de llevarlas a un problema práctico, por ende lo más recomendable es elegir la cantidad de vecinos mínima de forma tal que el grafo resultante sea conexo.

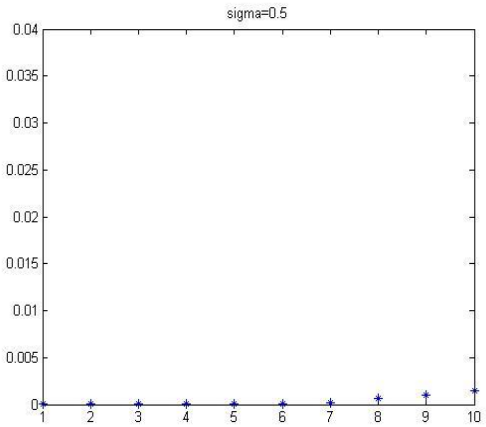
En el caso de los k vecinos mutuos no hay un criterio definido, en este caso el grafo conexo tiene la ventaja de no conectar componentes de distintas densidades, por lo que tendrá más componentes disconexas. Como la cantidad de lazos disminuye significativamente con respecto a k-vecinos, debemos elegir un k bastante mayor con respecto al k vecinos común, para amortiguar el efecto de los componentes disconexas.

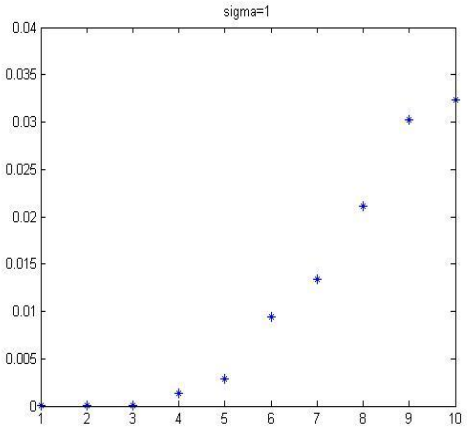
Para el caso de la franja ϵ, también pretendemos que el grafo resultante sea conexo, para eso utilizaremos el supuesto obtenidos de estudios prácticos, de que cuando n es relativamente grande, nuestro grafo será probablemente conexo si tomamos ϵ como $(log(n)/n)^d$, donde d representa la cantidad de dimensiones.

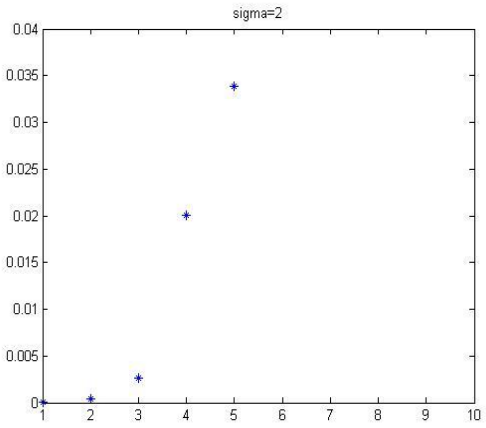
Nos resta, para el caso de utilizar la función de similaridad gaussiana para construir el grafo, setear el parámetro σ. Esto es decisivo en el caso del grafo de similitud completo, la idea es acotar por encima y por debajo con este parámetro, la cantidad de vecinos que tienen determinado nivel de similitud. Es común, tomar como criterio a priori, el usar σ como épsilon. Otro criterio es tomarlo como la distancia media de un punto al vecino k más cercano, tomando k como explicamos anteriormente. Estos criterios se desprenden de estudios prácticos y no son generalizables a todo conjuntos de datos, es decir, no garantizan buenos resultados. A continuación mostraremos el comportamiento de los 10 primeros valores propios de la Laplaciana creada a partir de distintos σ :

Nota: las bases de datos se corresponden con las mismas de la figura anterior, es decir, las primeros tres graficas se crean a partir de la primer base de datos (la que representa una "cara") y la 3 ultimas con la base de los círculos concéntricos.

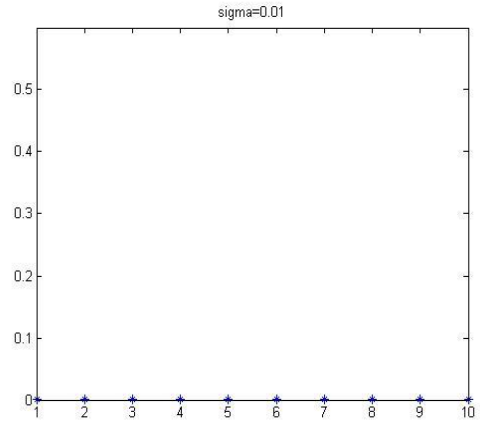
Primer base de datos:

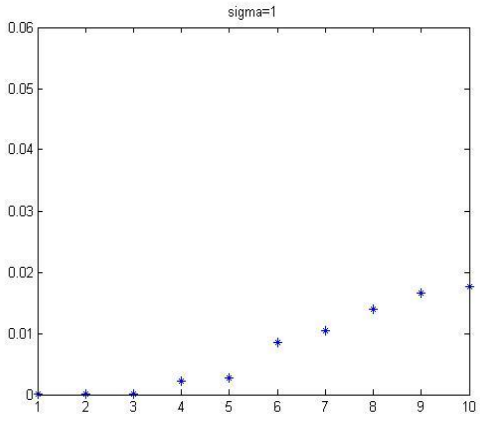


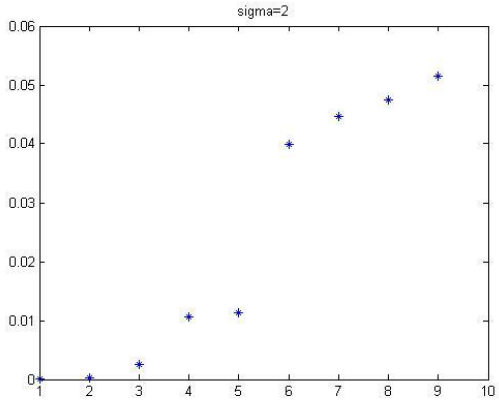




Segunda base de datos:







Como vemos, en ambos casos en los cuales tomamos el σ muy chico (0.5 y 0.01) respectivamente, los valores propios tienen a achicarse, de forma que la multiplicidad del valor propio 0 llego a 6 o más, cuando en realidad debería ser 3. Esto se explica ya que al tomar un σ tan pequeño, nuestra función de similitud considera como disimiles entre sí a casi todos los patrones y solo en un entorno muy pequeño a los similares, por lo tanto muchos de nuestros valores propios estarán muy cerca del 0. Por ende nuestro algoritmo encontrará mucho más clusters de los que realmente hay.

Esto no sucede cuando tomamos σ como 1 en ambos casos, donde vemos que la multiplicidad del 0 es 3, indicando los 3 clusters que realmente existen, y luego un salto en el cuarto valor propio que podemos considerar como la franja en el criterio para detenernos en la cantidad de clusters.

Luego vemos que ocurre totalmente lo opuesto que en el caso de considerar muy chico el σ, cuando lo consideramos demasiado grande. En este caso los valores propios se empiezan a alejar del 0, dejando este valor propio con multiplicidad uno, indicando que existe un solo cluster que engloba a los datos. Esto se explica como el efecto opuesto al σ pequeño, nuestra franja de similitud es mucho mas amplica y por ende todos nuestros datos tienden a ser similares entre sí, de forma que resulta en la clasificación de todos los datos dentro de un solo cluster.

Como conclusión final al estudio del parámetro σ, podemos decir que, el resultado del algoritmo es extremadamente dependiente de este parámetro, que actua de alguna forma como un ϵ. Es casi imposible definir un criterio global, y para encontrar el σ adecuado, se deben realizar muchísimas pruebas de spectral con la base que se quiere trabajar.

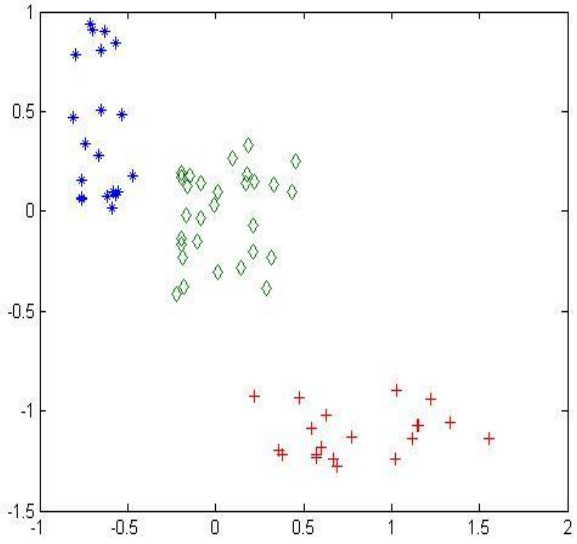
Comparación Spectral vs Kmeans

	Kmeans	Spectral
Ventajas	• Fácil de implementar. • Menores tiempos de ejecución. • Menores costos de memoria.	• Se adapta a clusters de formas más complejas. • No es tan susceptible a outliers.
Desventajas	• Funciona mal si los clusters tienen diferentes tamaños, densidad o no son convexos. • Busca clusters esféricos. • Problemas con outliers.	• Implementación más compleja. • Mayores tiempos de ejecución por complejidad en las operaciones con vectores propios de matrices grandes. • Mayores costos de memoria debido a la necesidad de almacenar la matriz de similaridad.

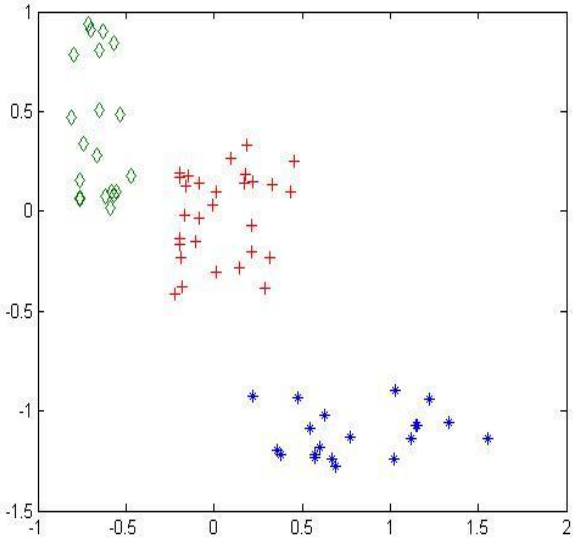
Datos Sintéticos

Programamos las funciones kmeandatos2d.m y SpectralUte.m en Matlab, las cuales implementan los algoritmos respectivamente. Luego los aplicamos a bases de datos sintéticos bidimensionales obtenidos de los prácticos, descargados de internet y algunas creadas por nosotros mismos, para ilustrar y comparar el funcionamiento de ambas técnicas. A continuación se muestran gráficamente los resultados:

K-means



Spectral

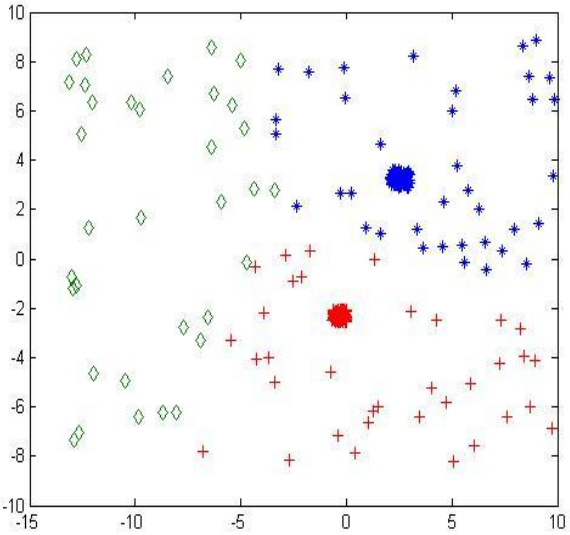


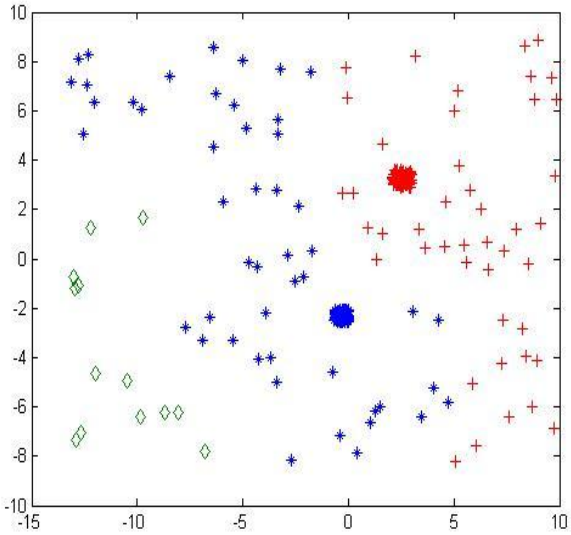
Tiempo de ejecución =0.00628s

Tiempo de ejecución =0.018619s

K-means

Spectral



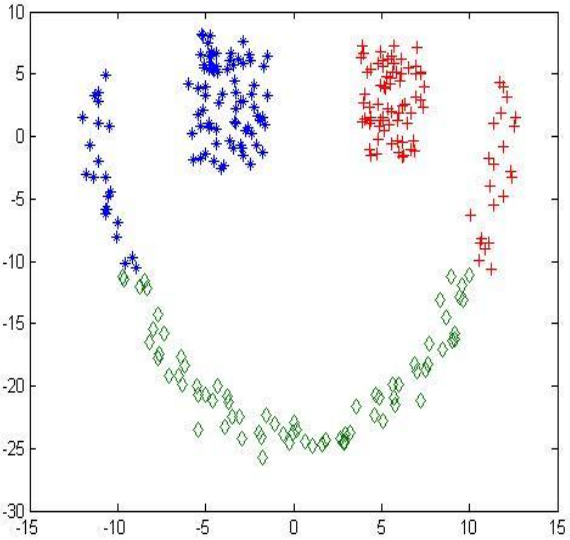


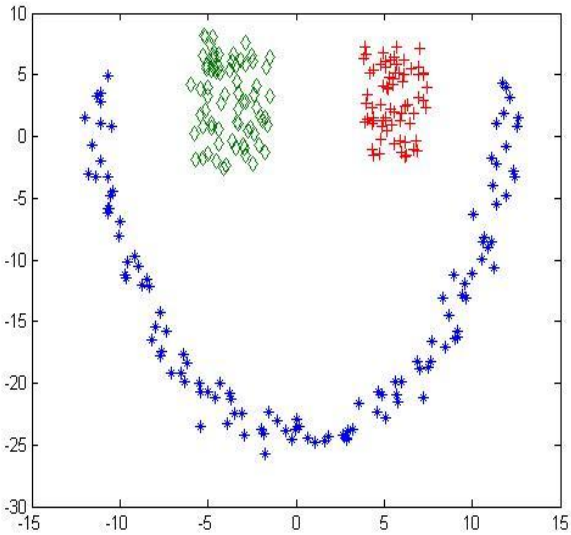
Tiempo de ejecución= 0.01559s

Tiempo de ejecución= 0.188067s

K-means

Spectral

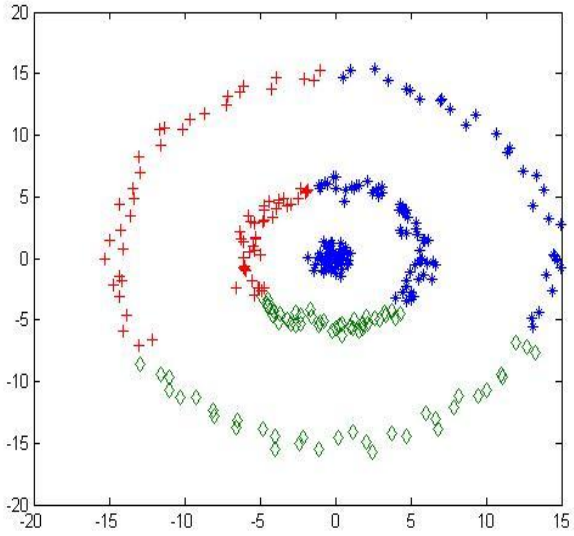




Tiempo de ejecución = 0.007943s

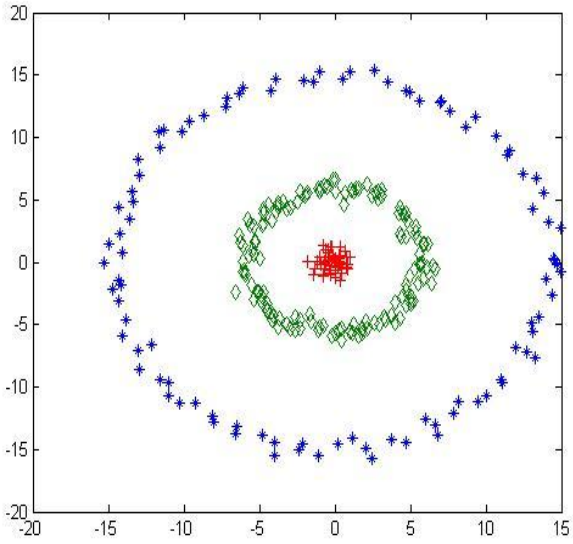
Tiempo de ejecución = 0.218917s

K-means



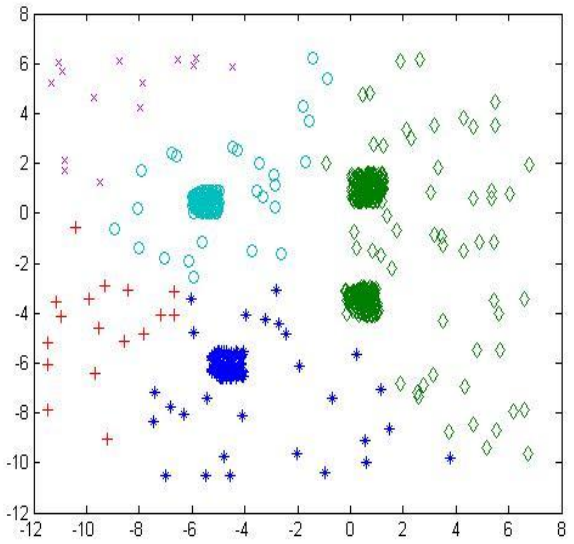
Tiempo de ejecución= 0.010062s

Spectral



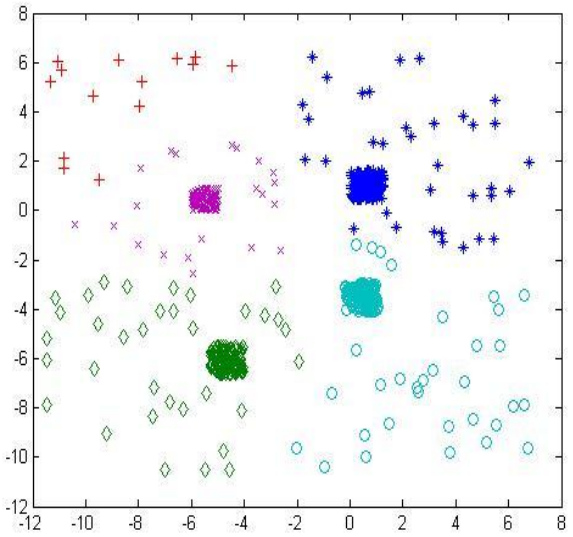
Tiempo de ejecución= 0.223489s

K-means



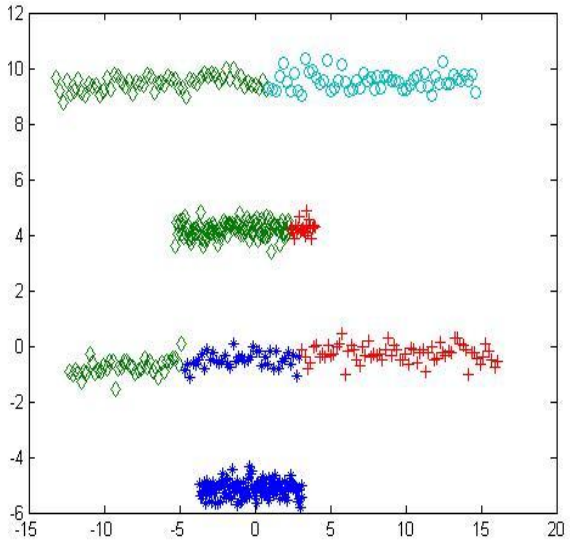
Tiempo de ejecución= 0.011628s

Spectral



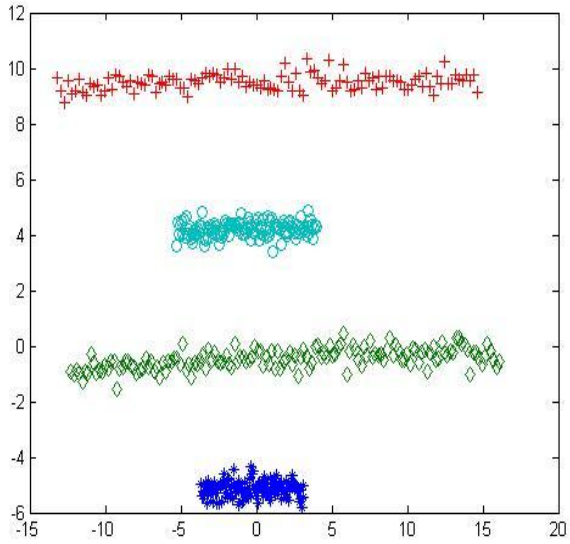
Tiempo de ejecución= 0.052090s

K-means



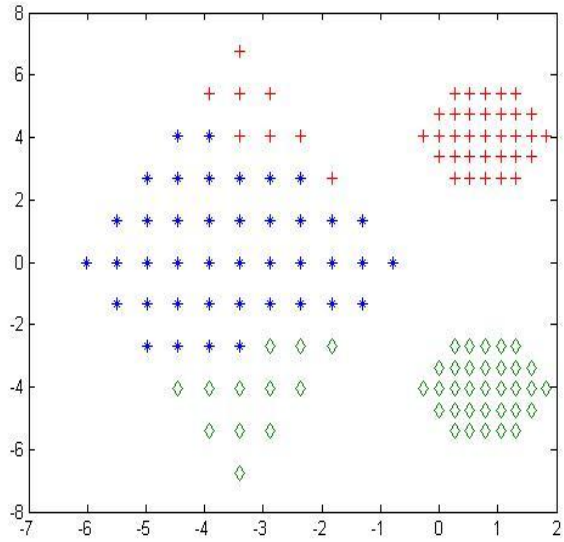
Tiempo de ejecución= 0.009579

Spectral



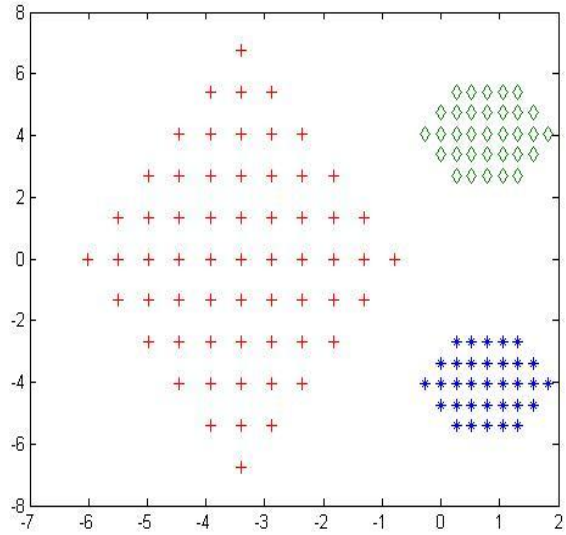
Tiempo de ejecución= 0.488634

K-means



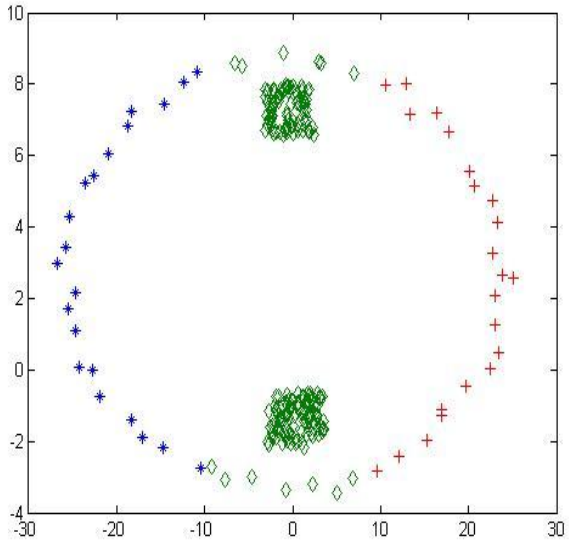
Tiempo de ejecución = 0.010062s

Spectral



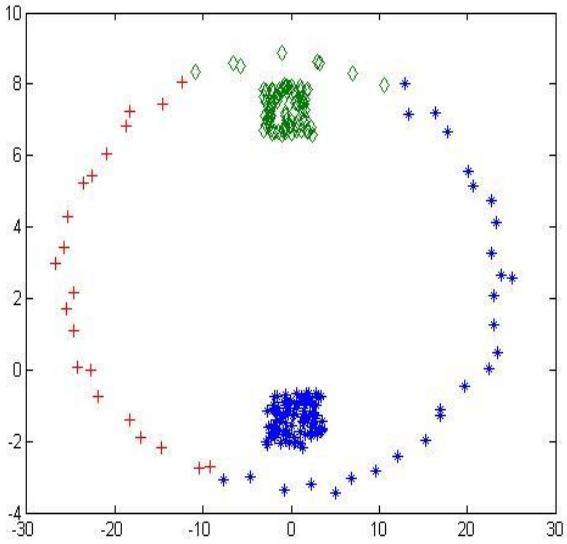
Tiempo de ejecución = 0.015177s

K-means



Tiempo de ejecución = 0.005511s

Spectral



Tiempo de ejecución = 0.066964

Observaciones: Comprobamos cierta superiodad de Spectral clustering para adaptarse a distintas formas de solución, obteniendo resultados aceptables en donde k-means falla. Aunque hay casos en los que ni siquiera spectral clustering funciona como nos gustaria. También verificamos que spectral tiene un tiempo de ejecución considerablemente mayor (del orden de 5 y 20 veces superior).

Técnicas de estimación de bondad de clusters

Las técnicas de estimación de bondad de clusters también llamadas validación de clustering, tienen como objetivo medir la calidad del resultado obtenido luego de aplicar un algoritmo de clustering. Este resultado puede ser una partición o una jerarquía de particiones, en el caso de nuestro trabajo será una partición, por lo que solo nos preocuparemos de la validación de particiones. Debido a la dificultad de conseguir una definición precisa de cluster y formular una base matemática sólida, la validación es una parte fundamental en el clustering. La dificultad de determinar el número correcto de clusters en unos datos determinados ha sido una de las razones que han provocado el desarrollo de esta área.
La validación de clustering se divide generalmente en dos tipos: interna y externa. Para utilizar validación externa se debe conocer previamente la partición correcta de los datos. En general, en un contexto real a priopi no se conoce la partición correcta, por lo cual este tipo de validación solo sirve en un contexto experimental para evaluar y comparar algoritmos de clustering. El problema de este tipo de técnicas es que no siempre se puede definir una única partición correcta, ya que los clusters pueden estar solapados.
La validación interna no precisa el conocimiento a priori de la partición correcta, ya que consiste en estudiar los datos y como se agrupan. Es decir, evalúa la partición en base a los datos y las distancias entre ellos. La ventaja principal de este tipo de técnicas es que no requiere el conocimiento de la partición correcta por lo que se puede utilizar en aplicaciones reales y no sólo en experimentos de laboratorio. Su desventaja principal es que evalúa la partición sobre los mismos datos que se han utilizado para construirla. Desafortunadamente,evaluar la adecuación de una partición sobre los datos de la misma partición evaluada es equivalente a evaluar los clusters bajo una diferente función objetivo. Por ejemplo, si utilizamos como medida de bondad de una partición el error cuadrático medio, es de suponer que un algoritmo como K-means obtenga mejores resultados que el single-linkage.
Tanto en validación externa como en interna, dada una partición se calculan índices que miden la calidad o bondad de dicha partición. Si el valor devuelto por el índice es inusualmente alto (o bajo), podemos deducir que la partición representa la estructura interna de los datos. El problema principal es determinar cuándo un valor es inusualmente alto (o bajo) y cuando no. Esto nos lleva a tratar el problema como un problema estadístico de contraste de hipótesis, lo cual en la práctica no es realizable. Lo que se utiliza es la comparación de índices de distintas particiones. Se utiliza, por tanto, una aproximación comparativa donde no se analiza el valor del índice en términos absolutos sino en términos relativos.

Validación externa

Los índices suelen ser medidas de similitud entre la partición resultante luego de aplicar el algoritmo y la correcta. Existen diversos tipos de medidas de similitud, dependiendo de la información en que se basan:

1-**Recuento de pares:** Este tipo de medidas basan la estimación de la similitud en una matriz de contingencia entre ambas particiones. Supongamos P = {C1,C2,…,CK} y P’ = {C’1,C’2,…,C’K’ } dos particiones de K y K0 clusters respectivamente. El valor de cada celda de la matriz de contingencia se define de la siguiente forma nij = |Ci C’j |. Es decir que el valor de la entrada ij de la matriz de contingencia es el número de datos que los clusters Ci y C’j tienen en común,
La matriz de contingencia correspondiente se muestra a continuación:

	C_1	C_2	$\cdots$	C_K	
C'_1	n_{11}	n_{12}	$\cdots$	n_{1K}	$n_{1.}$
C'_2	n_{21}	n_{22}	$\cdots$	n_{2K}	$n_{2.}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
$C'_{K'}$	$n_{K'1}$	$n_{K'2}$	$\cdots$	$n_{K'K}$	$n_{K'.}$
	$n_{.1}$	$n_{.2}$	$\cdots$	$n_{.K}$	$n_{..} = N$

Sin embargo,para calcular la similitud la mayoría de las técnicas se basan en una matriz 2x2. Se clasifican todos los posibles pares de datos (**A,B**) en:

- **A** y **B** están en el mismo cluster tanto en P como en P'.
- **A** y **B** están en el mismo cluster en P, pero no en P'.
- **A** y **B** están en el mismo cluster en P', pero no en P.
- **A** y **B** están en distintos clusters tanto en P como en P'.

Tradicionalmente la cantidad de pares de cada tipo se ha denominado a, b, c y D respectivamente. A continuación se muestra la matriz correspondiente:

		P'	
		Mismo cluster	Distinto cluster
P	Mismo cluster	a	b
	Distinto cluster	c	d

Se concluirá que dos particiones son similares si tienen valores altos de a y d, y valores bajos de b y c. Para calcular estos valores se utilizan las siguientes fórmulas:

$$a = (1/2)\sum_{i=1}^{K}\sum_{j=1}^{K'} n_{ij}^2 - (N/2) \qquad b = (1/2)\sum_{j=1}^{K'} n_{.j}^2 - (1/2)\sum_{i=1}^{K}\sum_{j=1}^{K'} n_{ij}^2$$

$$c = (1/2)\sum_{i=1}^{K} n_{i.}^2 - (1/2)\sum_{i=1}^{K}\sum_{j=1}^{K'} n_{ij}^2$$

$$d = \frac{N(N+1)}{2} - (1/2)\left[\sum_{i=1}^{K} n_{i.}^2 + \sum_{j=1}^{K'} n_{.j}^2\right]$$

Se han definido multitud de medidas de similitud basados en el recuento
de pares. A continuación se describen algunas de ellas, las cuales son las más utilizadas y se utilizarán en nuestro trabajo:

- **Rand: (a+d)/(a+b+c+d).** Es la más utilizada, el coeficiente corresponde a la fracción de pares de casos que caen en el mismo estado en ambas particiones.
- **Jaccard: a/(a+b+c).** Es similar a Rand, pero difiere en que no se toman en cuenta los pares con sus dos elementos clasificados en distintos clusters en ambas particiones. Se utiliza cuando hay muchos clusters, donde d puede ser muy alto.
- **Fowlkes-Mallows**: $a/\sqrt{(a+b)(a+c)}$. También ignora el valor de d.
- **Adjusted Rand:**

$$\frac{\sum_{i=1}^{K}\sum_{j=1}^{K'}\binom{n_{ij}}{2} - \left[1/\binom{N}{2}\right]\sum_{i=1}^{K}\binom{n_{i.}}{2}\sum_{j=1}^{K'}\binom{n_{.j}}{2}}{(1/2)\left[\sum_{i=1}^{K}\binom{n_{i.}}{2} + \sum_{j=1}^{K'}\binom{n_{.j}}{2}\right] - \left[1/\binom{N}{2}\right]\sum_{i=1}^{K}\binom{n_{i.}}{2}\sum_{j=1}^{K'}\binom{n_{.j}}{2}}$$

  Es una versión ajustada de Rand para que el resultado sea casi nulo cuando se comparan particiones generadas aleatoriamente.

2) **Teoría de la información**: Este es un grupo de medidas basado en conceptos de la teoría de la información. Las medidas de similitud usadas son:

- Entropía conjunta.
- Información mútua.
- Variation of Information.
- Normalized Mutual Information 1.
- Normalized Mutual Information 1.

No vamos a especificar como se calculan, debido a que no utilizaremos este grupo de medidas para este trabajo.

3) **Número de ediciones**: El índice propuesto por Charon es una distancia de edición. Calcula el número mínimo de casos que hay que mover de un cluster a otro para convertir una partición en otra.

A pesar de la gran cantidad de medidas de similitud propuesta no hay un criterio claro establecido para saber cuál medida se debe seleccionar. Además parece que hay una gama amplia de medidas, pero muchas de ellas son demasiado similares o incluso equivalentes.

Validación interna

Existe una gran cantidad de índices para la validación interna, la mayoría se basa en la cohesión de los clusters y la separación entre los mismos. Como no existe una base teórica matemática fuerte sobre esta área, no hay un criterio claro para determinar que índice se comporta mejor para cada caso. Los índices se dividen en dos tipos: tipo ratio y tipo suma. A continuación se mostrarán los índices que utilizaremos en nuestro trabajo:

- **Calinski-Harabasz:** Este índice está definido como la razón entre la dispersión interior de los clusters y la dispersión entre los clusters. El objetivo es maximizar el valor de la función CH defina de la siguiente forma:

$$\mathrm{CH}(P) = \frac{(N - |P|)\,\mathrm{inter}_{CH}(P)}{(|P| - 1)\,\mathrm{intra}_{CH}(P)}$$

$$\mathrm{inter}_{CH}(P) = \sum_{C\in P} |C|\,\mathrm{d}(\bar{C}, \bar{X}) \text{ e } \mathrm{intra}_{CH}(P) = \sum_{C\in P}\sum_{x\in C}\mathrm{d}(x, \bar{C}).$$

- **Davies-Bouldin**: emplea como medida de compacidad de un cluster la media de las distancias de sus puntos a su centroide, mientras que como medida de separabilidad utiliza la distancia entre los clusters, que puede ser, por ejemplo, la distancia euclidea entre los centros. Para escoger el número de clusters adecuado se toma el valor k que minimiza el índice de Davies Bouldin porque eso significa que los clusters son más compactos y están más separados.
  Este índice tiene como inconveniente que considera clusters compactos y
  bien separados, por lo que al calcular la compacidad según la distancia de los puntos a los centros, no detecta bien la forma de los clusters y si, además, éstos están solapados, no es posible determinar las fronteras entre los clusters, dando resultados erróneos acerca del número de grupos. Se calcula de la siguiente manera:

$$\mathrm{DB}(P) = \frac{1}{|P|} \sum_{C_k \in P} \max_{C_l \in P/C_k} \left\{ \frac{\mathrm{S}(C_k) + \mathrm{S}(C_l)}{\mathrm{d}(\bar{C}_k, \bar{C}_l)} \right\}$$
$$\mathrm{S}(C) = 1/|C| \sum_{x \in C} \mathrm{d}(x, \bar{C}).$$

- **Dunn:** El índice *Dunn* corresponde al ratio de la distancia más pequeña entre las observaciones de diferentes clústers y la distancia inter-cluster más grande . El número de clusters que maximiza Dunn(P) se considerará el correcto, se calcula del siguiente modo:

$$\mathrm{Dunn}(P) = \frac{\mathrm{inter}_{Dunn}(P)}{\mathrm{intra}_{Dunn}(P)}$$

$$\mathrm{inter}_{Dunn}(P) = \min_{C_k \in P} \left\{ \min_{C_l \in P/C_k} \{\delta(C_k, C_l)\} \right\}$$

$$\delta(C_k, C_l) = \min_{x_i \in C_k} \left\{ \min_{x_j \in C_l} \mathrm{d}(x_i, x_j) \right\}$$

$$\mathrm{intra}_{Dunn}(P) = \max_{C \in P} \left\{ \max_{x_i, x_j \in C} \mathrm{d}(x_i, x_j) \right\}$$

- **Silhouette:** El índice *silhouette width* , al igual que índice Dunn, mide cuan compactos y separados están los clústers. Este índice corresponde al promedio del valor silhouette de cada observación y mide el grado de confianza en la asignación de clústers de una observación en particular. Los valores próximos a 1 gozarán de una mayor confianza en la agrupación realizada, por el contrario, los valores próximos a -1 significan que la agrupación no es fiable. El intervalo de valores es [-1,1] . Para una observación *i* se define así:

$$S(i) = \frac{b_i - a_i}{max(b_i, a_i)}$$

Donde:
- ai es la distancia promedio entre i y el resto de las observaciones en el mismo cluster.
- bi es la distancia promedio entre i y el resto de las observaciones del cluster vecino más cercano.

Para un número de clusters dado *K*, el ancho de silueta promedio de la configuracion de conglomerados será simplemente el promedio de sil_i sobre todas las observaciones. Es decir:

$$\bar{s} = \frac{\sum sil_i}{n}$$

Kaufman y Rousseeuw (1990) sugirieron estimar el número óptimo de cluster K para el cual el ancho de silueta promedio sea la mayor posible.

- **R-Squared:** Es una medida de similitud de clusters tipo ratio. Toma valores entre 0 y 1, valores cercanos a 1 representan que hay una diferencia significante entre los clusters, y valores cercanos a 0 representan lo contrario.

$$RS = \frac{SS_t - SS_w}{SS_t}$$

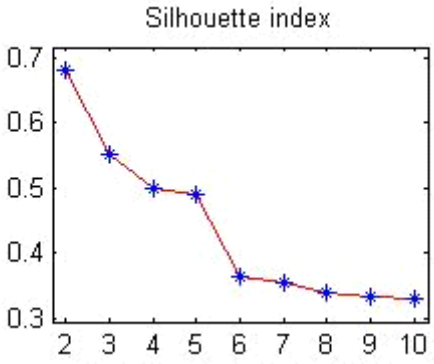
$$SS_t = \sum_{j=1}^{d}\sum_{k=1}^{n_j}\left(x_k - \overline{x_j}\right)^2,\ SS_w = \sum_{\substack{i=1\ldots nc\\ j=1\ldots d}}\sum_{k=1}^{n_{ij}}\left(x_k - \overline{x_j}\right)^2$$

Validación con datos sintéticos

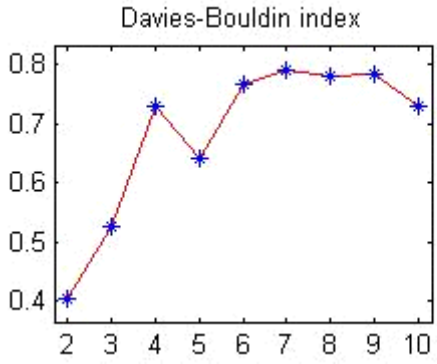
A continuación se utilizaran técnicas de validación interna para estimar el número de clusters en: datos9 (Iris), datos3(carita), datos4(círculos). En resumen para calcular la cantidad de clusters más bondadosa se debe buscar la cantidad de agrupaciones que minimizen DaviesBoulding, que maximizen Dunn, Calinski y la silueta promedio; y que al aumentar la cantidad de cluster no aumente considerablemente R-Squared.

Datos9: "Iris"

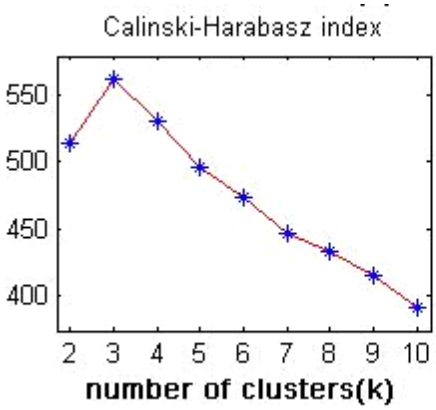
K-means:

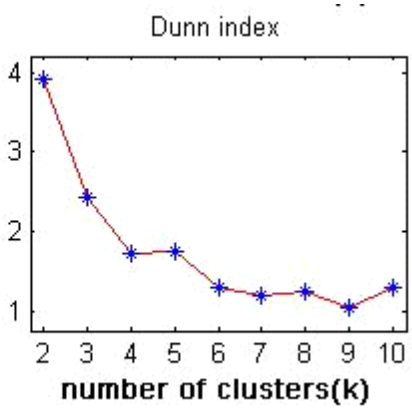


Maximiza en 2 clusters.



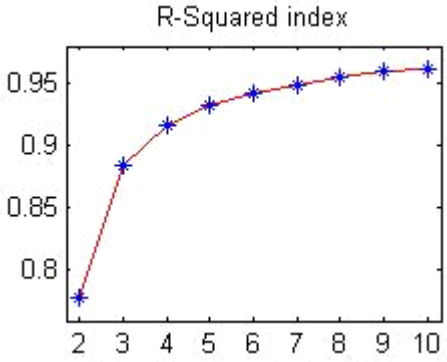
Minimiza en 2 clusters.





Maximiza en 3 clusters.

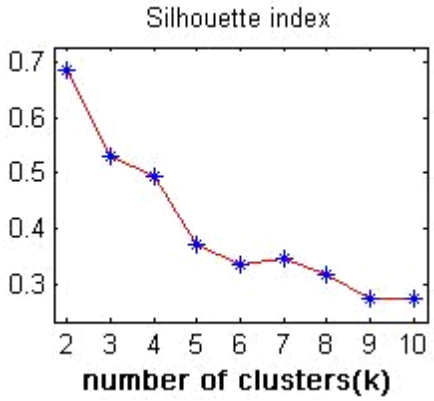
Maximiza en 2 clusters.



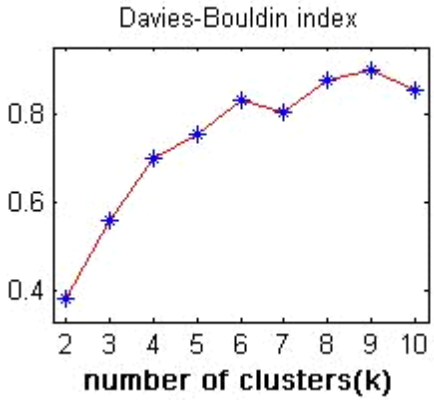
El mayor aumento se da desde 2 a 3, y luego de 3 no aumenta considerablemente, por lo cual se tomarían 3 clusters.

Calinski-Harabatz y R-Squared acertaron la cantidad de clusters, mientras Dunn, Davies-Boulding y Silohuette erróneamente dijeron que hay 2 clusters.

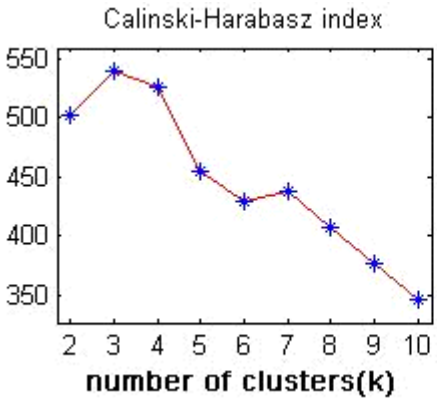
Spectral



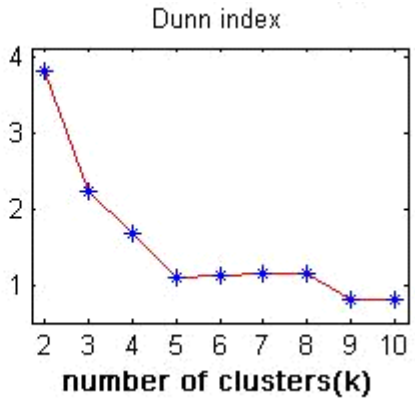
Maximiza en 2 clusters.



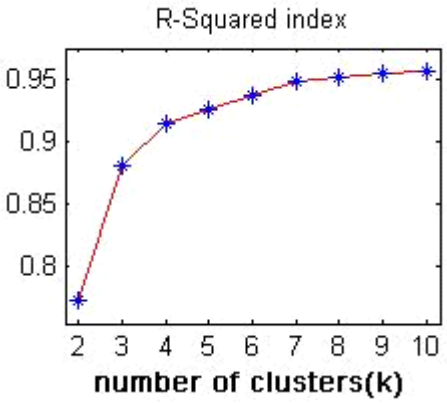
Minimiza en 2 clusters.



Maximiza en 3 clusters.



Maximiza en 2 clusters.

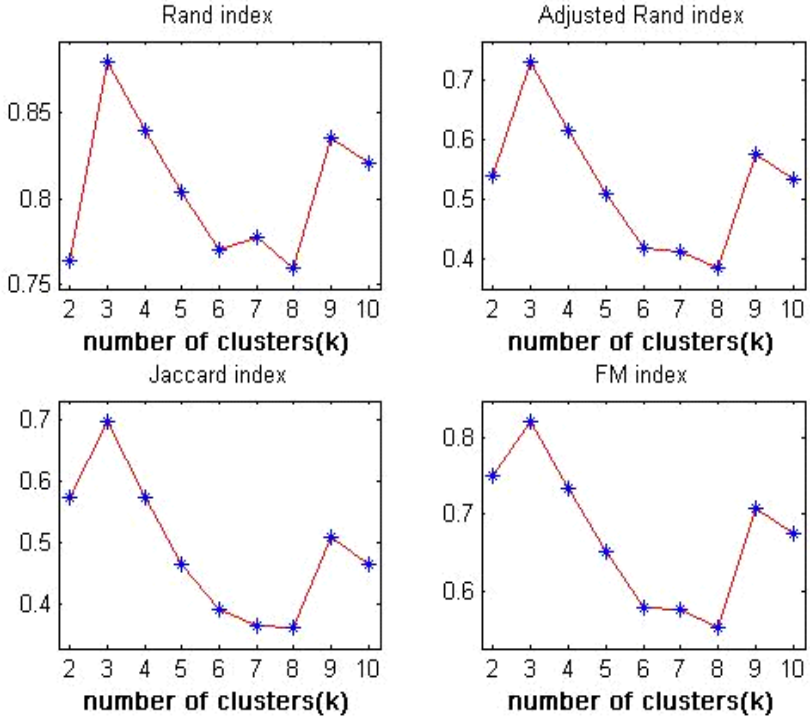


El mayor aumento se da desde 2 a 3, y luego de 3 no aumenta considerablemente, por lo cual se tomarían 3 clusters.

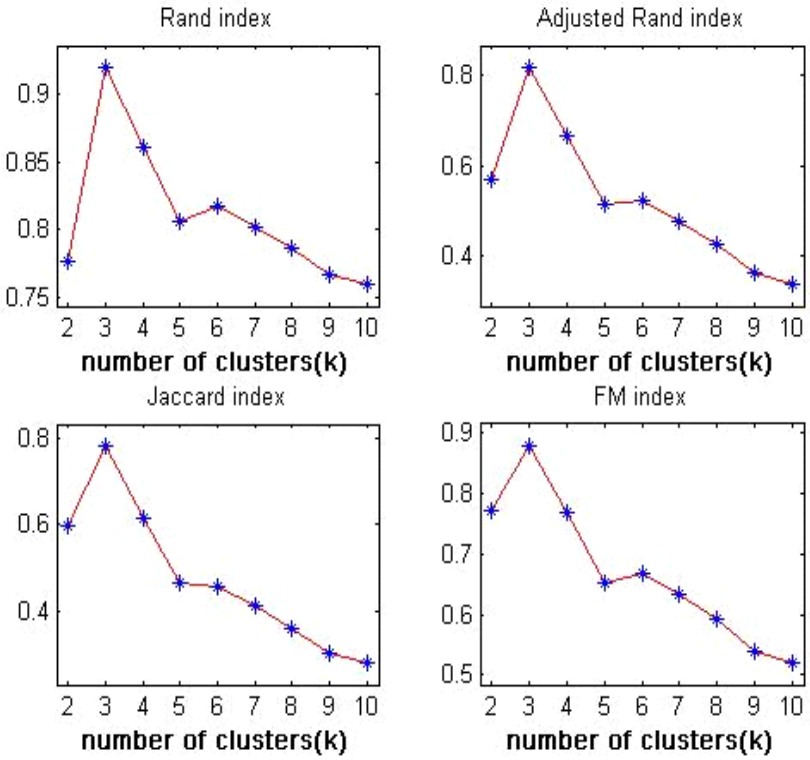
Calinski-Harabatz y R-Squared acertaron la cantidad de clusters, mientras Dunn, Davies-Boulding y Silohuette erróneamente dijeron que hay 2 clusters, al igual que en kmeans.

Validación externa para iris

k-means:



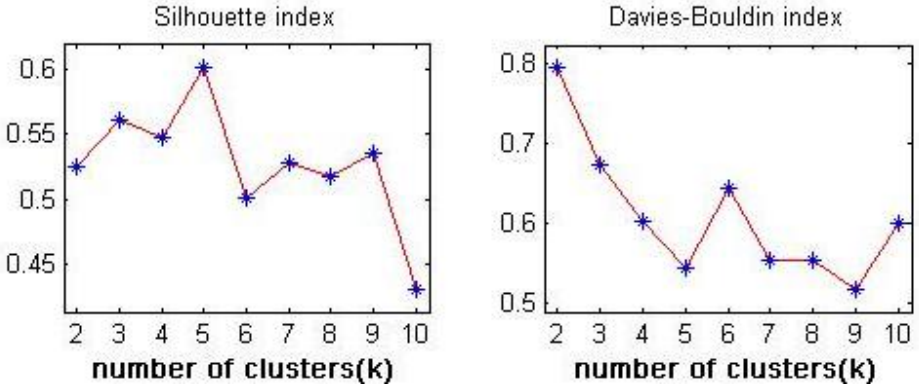
Spectral



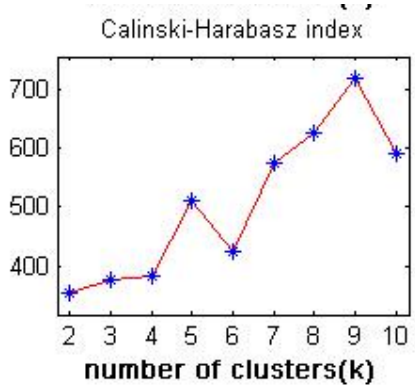
Luego de aplicar validación externa se verifica que la cantidad de clusters apropiadas para estos algoritmos es 3. Para esta cantidad de clusters se observa una superioridad leve en spectral clustering en los 4 índices de validación.

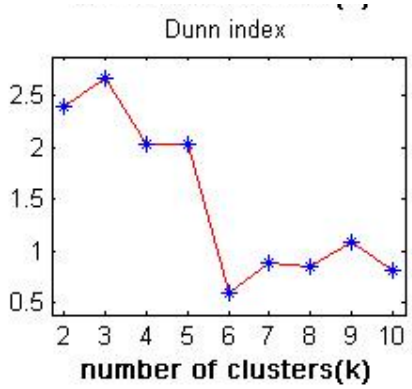
Datos3: “carita”

Kmeans:

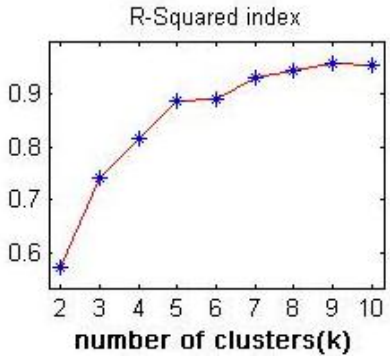


Maximiza en 5 Minimiza en 9





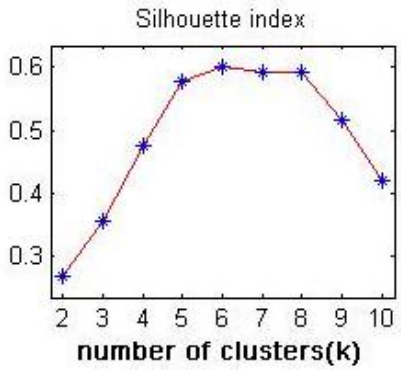
Maximiza en 9 Maximiza en 3

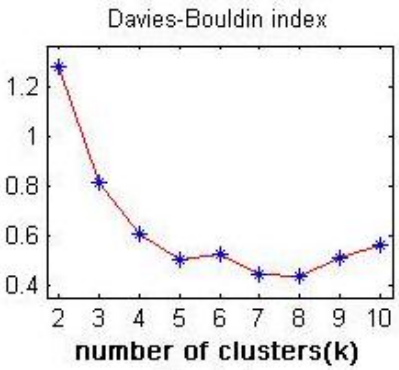


Mirando el gráfico se observa que luego de 5 clusters la medida R-Squared aumeta poco, por lo tanto parecería razonable concluir que habrían 5 clusters.

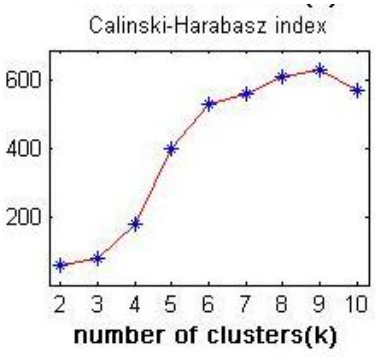
Solo Dunn estimo bien el número de clusters, además el resto de los algoritmos dieron resultados distintos entre sí, esto se debe a que en realidad el algoritmo k-means no está capacitado para agrupar esta base de datos.

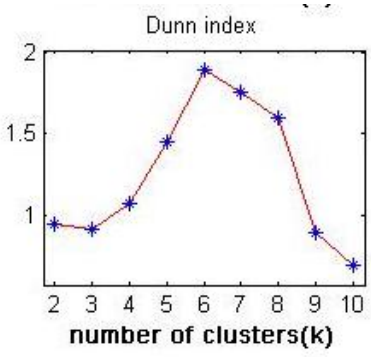
Spectral:





Maximiza en 6 Minimiza en 8



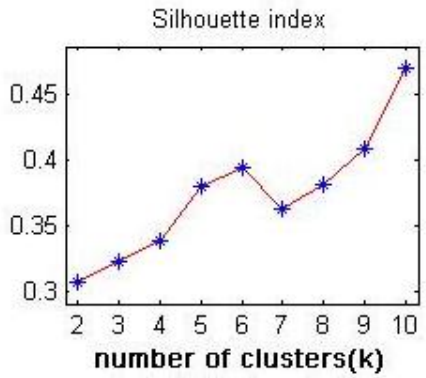


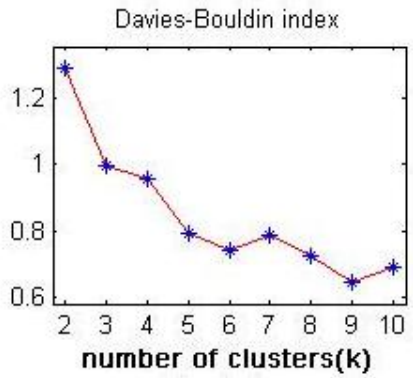
Maximiza en 9 Maximiza en 6

A pesar de que el algoritmo hace una clasificación perfecta en 3 clusters, ninguno de los índices es capaz de notarlo.

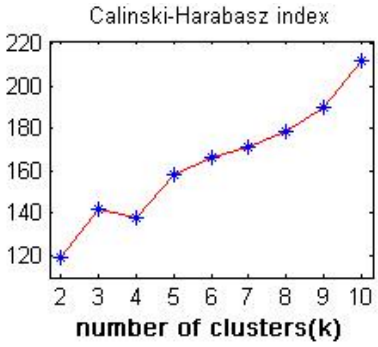
Datos4: “Círculos concéntricos”

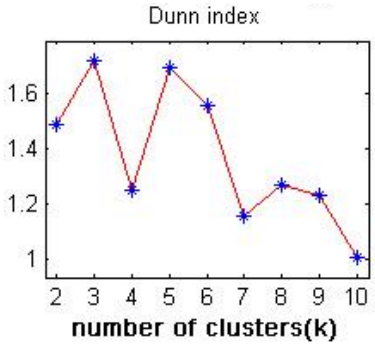
kmeans



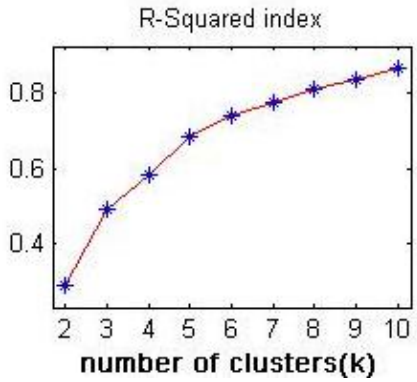


Maximiza en 10 o más clusters. Minimiza en 9 clusters.





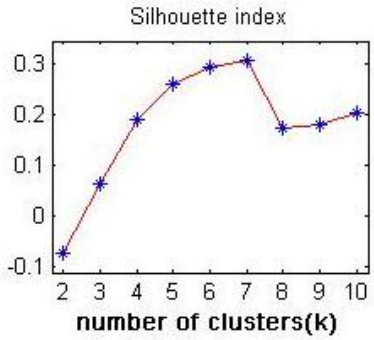
Maximiza en 10 o más clusters. Maximiza en 3 clusters.



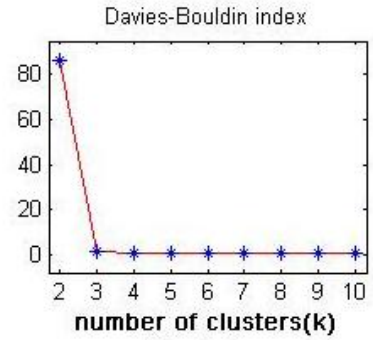
El gráfico no es muy concluyente ya que no se verifica una cantidad de clusters óptima tal que luego de aumentarla no aumente significativamente el índice R-Squared.

Solo Dunn estimo bien el número de clusters, además el resto de los algoritmos dieron resultados distintos entre sí, esto se debe a que en realidad el algoritmo k-means no está capacitado para agrupar esta base de datos.

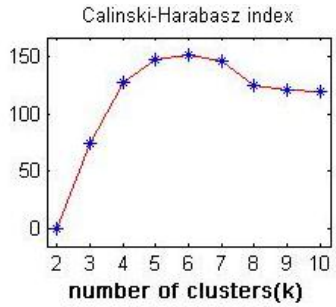
Spectral:



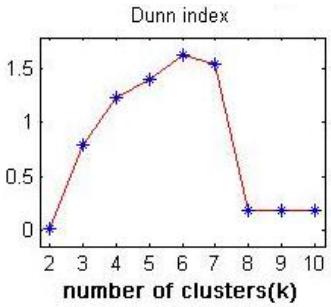
Maximiza en 7.



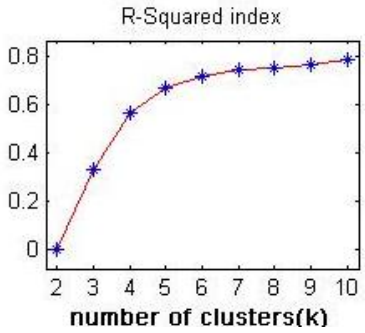
Se acerca a 0 entre 3 y 10 clusters.



Maximiza en 6.



Maximiza en 6.

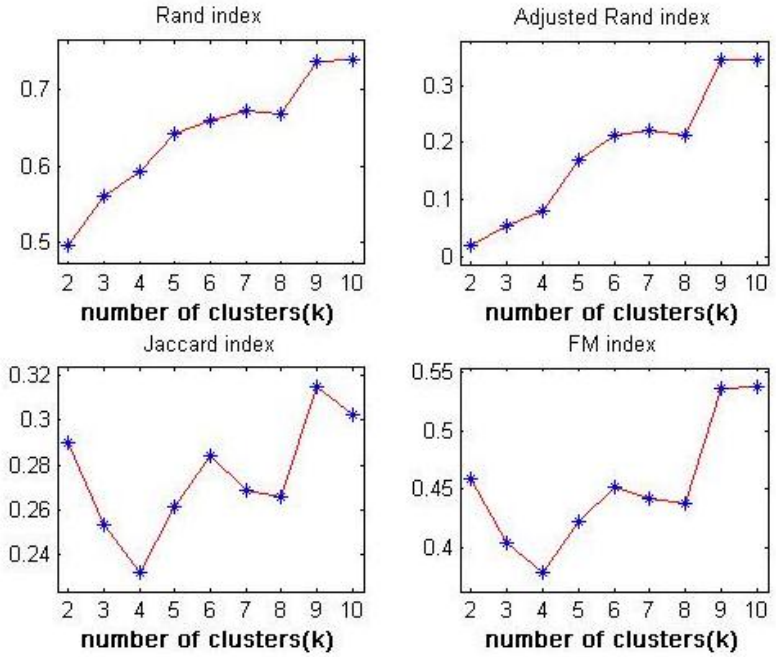


El gráfico no es muy concluyente ya que no se verifica una cantidad de clusters óptima tal que luego de aumentarla no aumente significativamente el índice R-Squared.

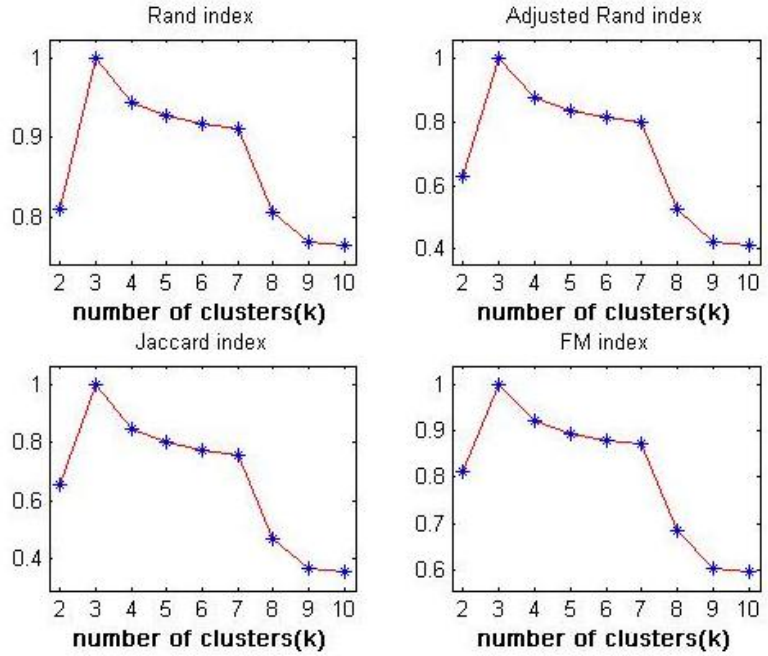
A pesar de que el algoritmo clasifica bien esta base de datos para 3 clusters, al aplicar validación interna solo Davies-Bouldin tuvo un resultado medianamente apropiado.

Validacion externa para círculos

Kmeans:



Spectral:



Se observa que kmeans no tiene valores aceptables en ninguno de los índices, esto se debe a que agrupa indebidamente los clusters en esta base de datos. En cambio spectral clustering muestra una perfomance perfecta para 3 clusters porque es capaz que de etiquetarlos perfectamente en 3 clusters.

Comparación de algoritmos con validación

A continuación se muestra una tabla en donde se calculan los índices de validación interna para todas las bases de datos utilizando los dos algoritmos:

Datos/Algoritmo	Kmeans	Spectral(sigama=1.3)	Conclusiones
Datos1(3clusters) "DATOS2D DEL PRACTICO 5"	Silhouette: 0.55 Davies-Bouldin: 0.44 Calinski-Harabasz:94.9 Dunn: 1.54 R-squared: 0.97	Silhouette: : 0.57 Davies-Bouldin: 0.51 Calinski-Harabasz: 190.8- Dunn: 2.4 R-squared:0.89	Se asemejan Sil y DB, spectral es un pocomejor en CH y Dunn, y kmeans un poco mejor en R-Squared. No hay muchas diferencias debido a que no difieren mucho en la forma de agrupar para este caso.
Datos2(3clusters) "DOS AGRUPACIONES Y EL RESTO DISPERSOS"	Silhouette: 0.63 Davies-Bouldin: 0.60 Calinski-Harabasz: 309.26 Dunn: 1.82 R-squared: 0.73	Silhouette: 0.60 Davies-Bouldin: 0.91 Calinski-Harabasz:214.4 Dunn: 1.09 R-squared:0.79	A pesar de que spectral clasifica mejor, k means presenta mejores resultados de validación.
Datos3(3 clusters) "CARITA"	Silhouette: 0.56 Davies-Bouldin:0.67 Calinski-Harabasz:376.3 Dunn: 2.67 R-squared: 0.84	Silhouette: 0.36 Davies-Bouldin: 1.25 Calinski-Harabasz:79.0 Dunn: 0.91 R-squared:0.38	A pesar de que spectral clasifica mejor, k means presenta mejores resultados de validación.
Datos4(3 clusters) "CIRCULOS CONCENTRICOS"	Silhouette: 0.33 Davies-Bouldin: 1.06 Calinski-Harabasz:142.0 Dunn: 1.78 R-squared:0.73	Silhouette: -0.01 Davies-Bouldin: 81.9 Calinski-Harabasz:0.02 Dunn: 0.008 R-squared:0.17	A pesar de que spectral clasifica mejor, k means presenta mejores resultados de validación.
Datos5(5clusters) "CUATRO AGRUPACIONES Y EL RESTO DISPERSOS"	Silhouette: 0.67 Davies-Bouldin: 0.53 Calinski-Harabasz:870 Dunn: 2.07 R-squared:0.94	Silhouette: 0.68 Davies-Bouldin: 0.51 Calinski-Harabasz:876.9 Dunn: 1.67 R-squared:0.94	Se asemejan en todos los índices, esto se debe a que clasifican parecido.
Datos6(4clusters) "LINEAS"	Silhouette: 0.43 Davies-Bouldin:0.72 Calinski-Harabasz:407.6 Dunn: 1.23 R-squared:0.82	Silhouette: 0.39 Davies-Bouldin: 0.99 Calinski-Harabasz:241.7 Dunn: 0.95 R-squared:0.618	A pesar de que spectral clasifica mejor, k means presenta mejores resultados de validación.
Datos7(3 clusters) "DENSIDADES Y	Silhouette: 0.64 Davies-Bouldin: 0.48 Calinski-	Silhouette: 0.61 Davies-Bouldin: 0.43 Calinski-Harabasz:227.4 Dunn: 2.0	Se asemejan en todos los índices, por más que hay diferencias significantes en la forma de etiquetar.

TAMAÑOS DIFERENTES ”	Harabasz:230.7 Dunn: 2.0 R-squared:0.80	R-squared:0.80	
Datos8(3clusters) “DOS AGRUPACIONES Y CIRCULO”	Silhouette: 0.74 Davies-Bouldin: 0.48 Calinski-Harabasz:631.1 Dunn: 2.98 R-squared:0.86	Silhouette: 0.67 Davies-Bouldin:0.68 Calinski-Harabasz:352.9 Dunn: 1.94 R-squared:0.94	K-means gana en casi todos los índices menos en RS, a pesar de que Spectral etiqueta adecuadamente.
Datos9(3 clusters) “IRIS”	Silhouette: 0.58 Davies-Bouldin: 0.57 Calinski-Harabasz:584.95 Dunn: 2.42 R-squared:0.94	Silhouette: 0.56 Davies-Bouldin: 0.59 Calinski-Harabasz:560.37 Dunn: 2.33 R-squared:0.95	K-means recibe una validación levemente superior.

Se observa que los índices de validación son en general más “generosos” con k-means, además este algoritmo gana casi siempre en RS debido a que es su función objetivo. Por lo tanto no parece muy razonable usar validación interna para hacer comparaciones entre los resultados de estos algoritmos.

A continuación se muestra una tabla de validación externa para los dos algoritmos:

Datos/Algoritmo	Kmeans	Spectral
Datos9(3 clusters) “IRIS”	Rand:0.88 Adjusted Rand:0.73 Jaccard:0.70 Fowlkes-Mallows:0.82	Rand:0.89 Adjusted Rand:0.76 Jaccard:0.72 Fowlkes-Mallows:0.84
Datos4(3 clusters) “CIRCULOS CONCENTRICOS”	Rand:0.56 Adjusted Rand:0.052 Jaccard:0.25 Fowlkes-Mallows:0.40	Rand:1 Adjusted Rand:1 Jaccard:1 Fowlkes-Mallows:1

Se observa una leve superioridad de spectral respecto a k-means para Iris, y una superioridad extrema también para Spectral clustering en el caso de círculos concentricos. Además los índices de kmeans para esta base de datos son muy pequeños, por lo cual se considera una mala clasificación. Todas estas conclusiones se corresponden perfectamente con lo visto en las gráficas de los datos sintéticos, por lo cual concluímos que validación externa es una buena técnica de validación y es eficaz para comparar particiones.

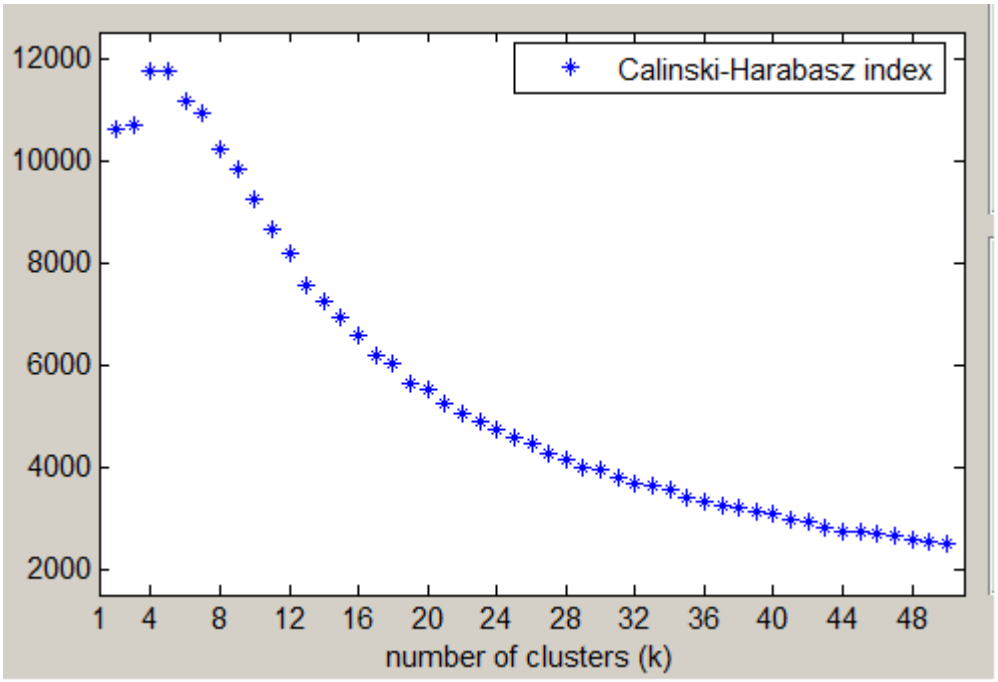
Preprocesamiento de datos

Se construyó un programa en java llamado CSVconsumo, en donde a partir de los archivos de texto Consumos.txt y Suministros.txt proporcionados por UTE, se crea el archivo **CSV.csv** tomando como características los datos de consumo mensuales de UTE desde enero del 2008 hasta octubre del 2010. Se descartaron los clientes que tenían consumos con reajustes (consumos negativos), y a los que les faltaba el registro de al menos dos consumos mensuales (es decir aquellos clientes que tienen menos de 33 de los consumos seleccionados como características). En caso de que falte solo un consumo interpolamos el consumo faltante linealmente. De esta forma logramos obtener 25986 registros de los 36000 aproximados disponibles en la base. De forma paralela, etiquetamos estos patrones, utilizando expresiones regulares, en 15 grupos según las etiquetas de "rubro" que a nuestro criterio agrupaban más clientes, logrando clasificar según este criterio 8355 de los 25986 clientes. Los rubros seleccionados fueron: vivienda, fábrica, almacén, oficina, restaurante, sgalum, pensión, tienda, panadería, peluquería, kiosco, imprenta, estacionamiento, farmacia y pizzería.

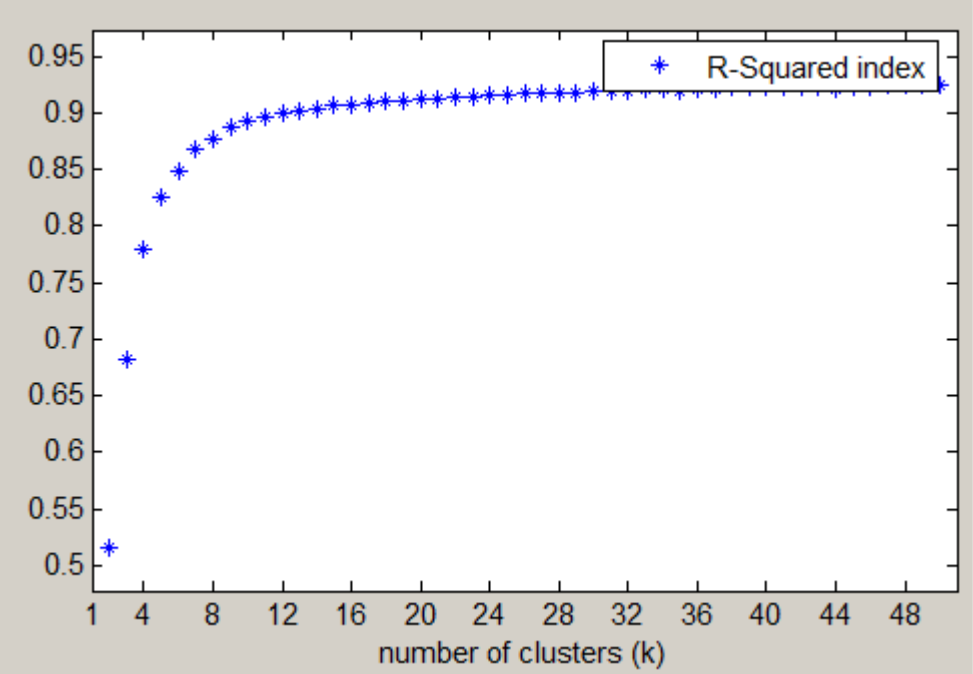
Procesamiento con Kmeans

Se programó en Matlab un script llamado procesamiento.m, en el mismo se cargan los datos del archivo CSV.csv generado en el preprocesamiento de datos. Luego, se normalizan los mismos, se aplica el algoritmo kmeans y se calculan al fin los porcentajes de muestras de cada rubro que caen en cada cluster. Este script recibe como parámetro de entrada la cantidad de clusters con los que se va a correr kmeans.

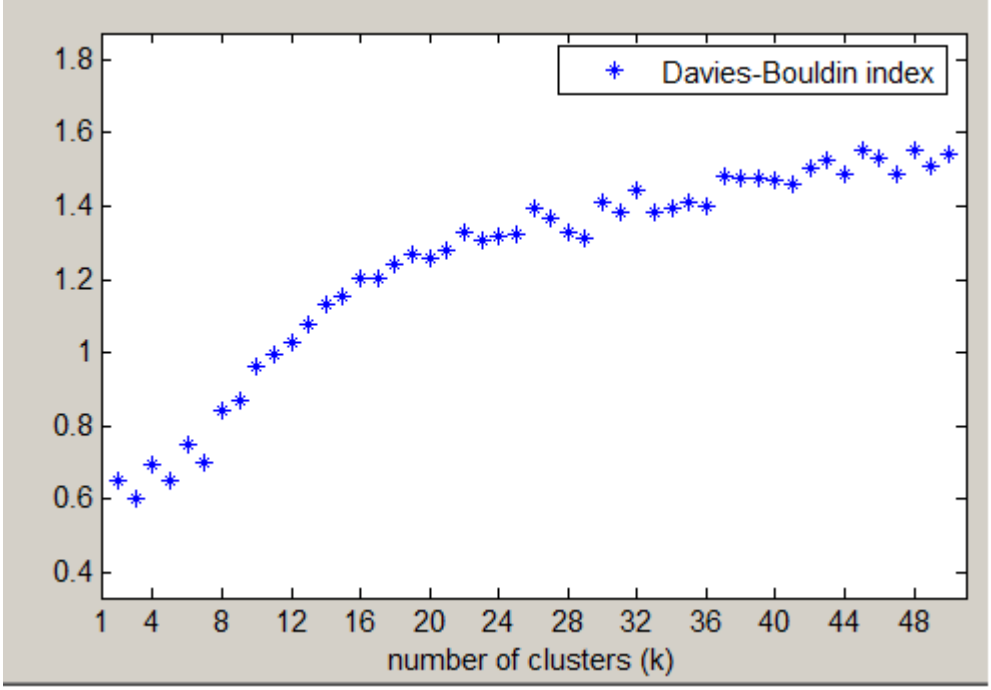
Corrimos el script mencionado en el párrafo anterior variando la cantidad de clusters entre 2 y 50 clusters. Para cada una de estas cantidades calculamos los índices de validación interna Davies-Bouldin, R-Squared, Calinski-Harabasz y Dunn. Luego graficamos índice vs cantidad de clustes. Los resultados fueron los siguientes:



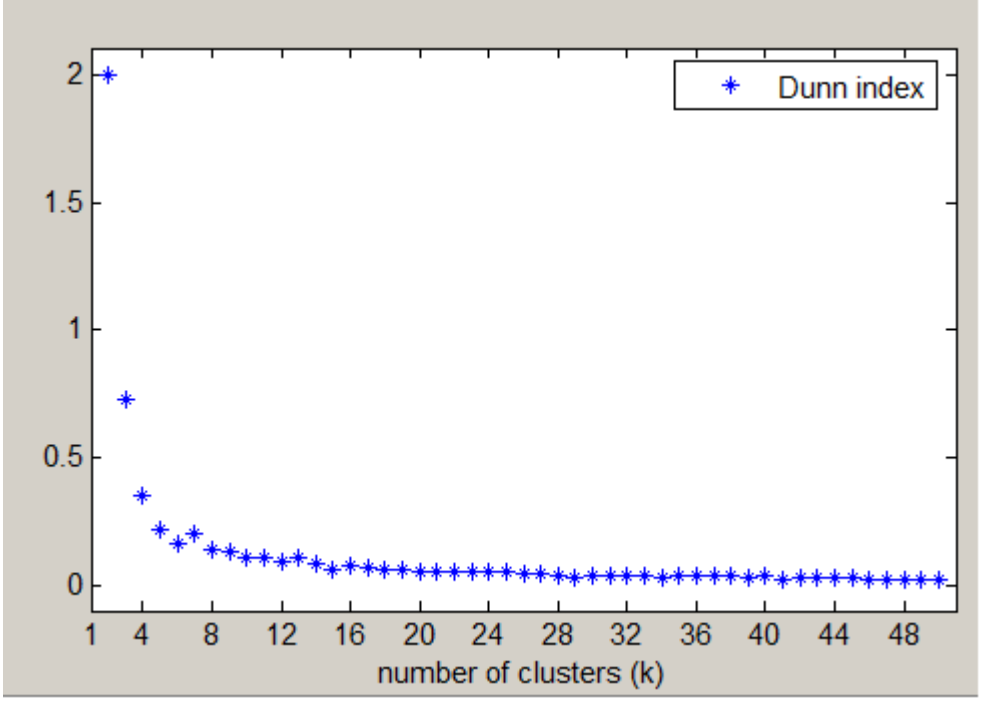
Maximiza en 4 o 5 clusters.



Observando la gráfica se aprecia que luego de los 7 clusters no se aumenta significativamente R-Squared.



Minimiza en 3 clusters.



Maximiza para 2 clusters.

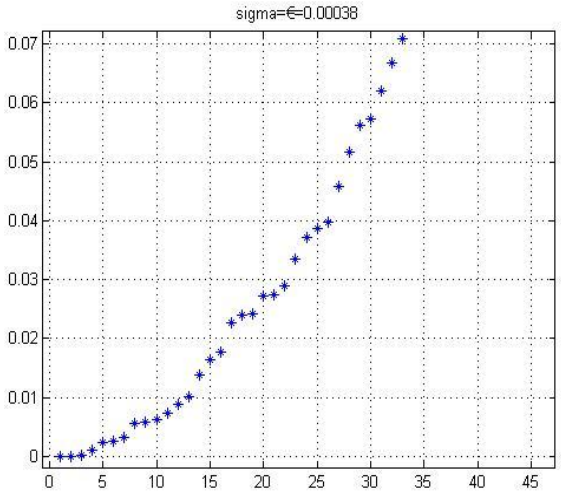
En base a estos resultados, decidimos realizar el estudio de las correspondencias con las etiquetas para 3 y 7 clusters, y creamos para cada test, una tabla, en donde cada fila indica el “rubro”, la primer columna la cantidad dentro de cada rubro (constante para todas las tablas) y el resto de las columnas los clusters que encontró el algoritmo. De esta forma, el casillero de fila i, columna j indica la cantidad de patrones (porcentual) en ese rubro que cayeron dentro del cluster j (excluyendo la primer columna).

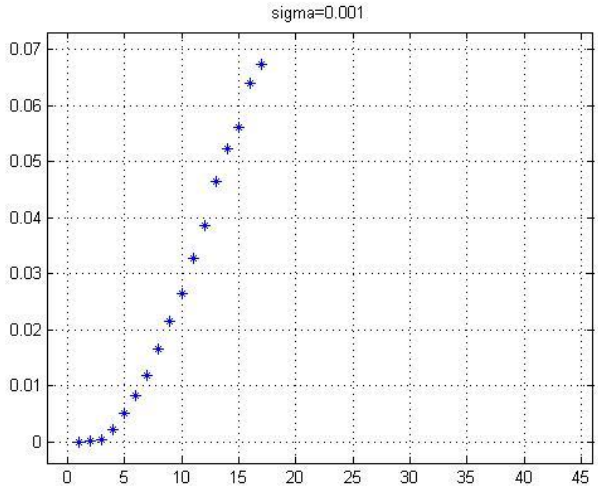
Procesamiento con Spectral Clustering

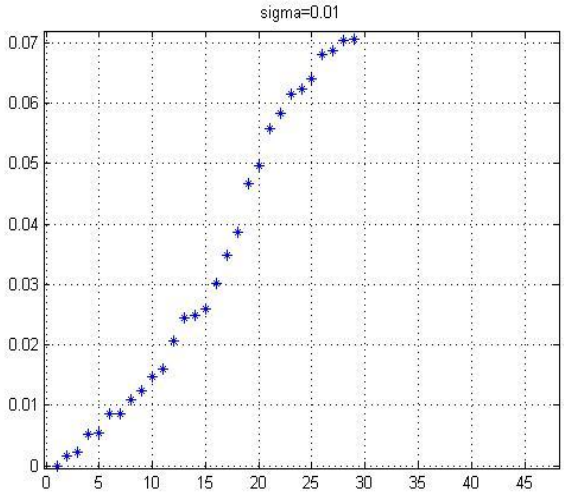
Comenzamos normalizando los datos y obtenemos la matriz de similaridad. Esta matriz la creamos utilizando el criterio de “The k-nearest neighborhood”, tomando los k vecinos más cercanos para construir el grafo, en donde k lo calculamos como log(n) con n la cantidad de datos, de forma que obtenemos un k aproximado a 10. A su vez, utilizamos solo los primeros 10 mil datos de de los 25986 disponibles, ya que por temas de tiempo de ejecución, y principalmente de memoria, no pudimos utilizarlos todos (la matriz de 26mil por 26mil, necesaria para representar los pesos correspondientes, necesitaba demasiada memoria).

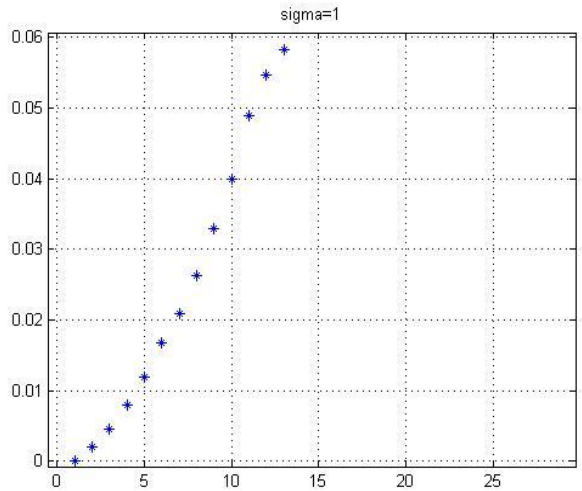
Parametrización cantidad de clusters y σ

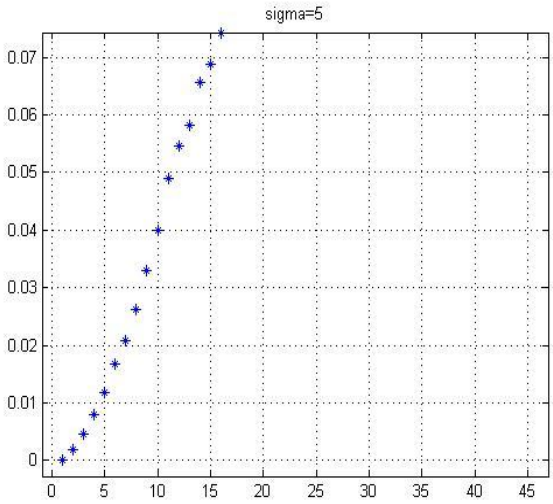
La parametrización de la cantidad de clusters a encontrar y del σ a utilizar las realizamos a través del estudio de los valores propios según lo visto en la parte teorica, aplicando SC a distintos σ para estudiar su comportamiento:









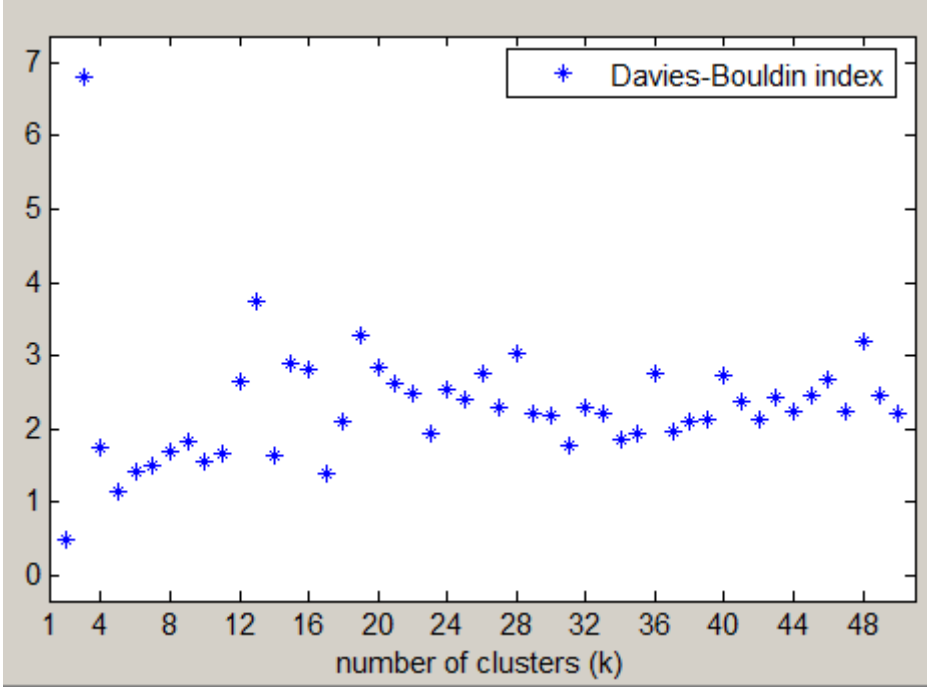


De las gráficas anteriores podemos deducimos que el σ más apropiado resultó ser el planteado como una opción teórica, es decir $\sigma = \epsilon = 0.00038$, ya que este es el que, teniendo multiplicidad 3 del valor propio 0, encuentra 3 clusters definidos, y a su vez tiene saltos en los valores propios definidos, lo que marca una pauta para encontrar las mejores opciones en cantidad de clusters. A diferencia de estos, en el resto de las gráficas (salvo quizás en la de 0.001, relativamente cercano al seleccionado como optimo 0.00038) obtuvimos multiplicidad del valor propio 0 como 1, lo que indica que posiblemente estamos considerando a muchos datos como "muy similares" y por lo tanto agrupándolos a la mayoría en un único cluster. Ademas la primer gráfica nos sugiere que las posibles cantidades de clusters son:

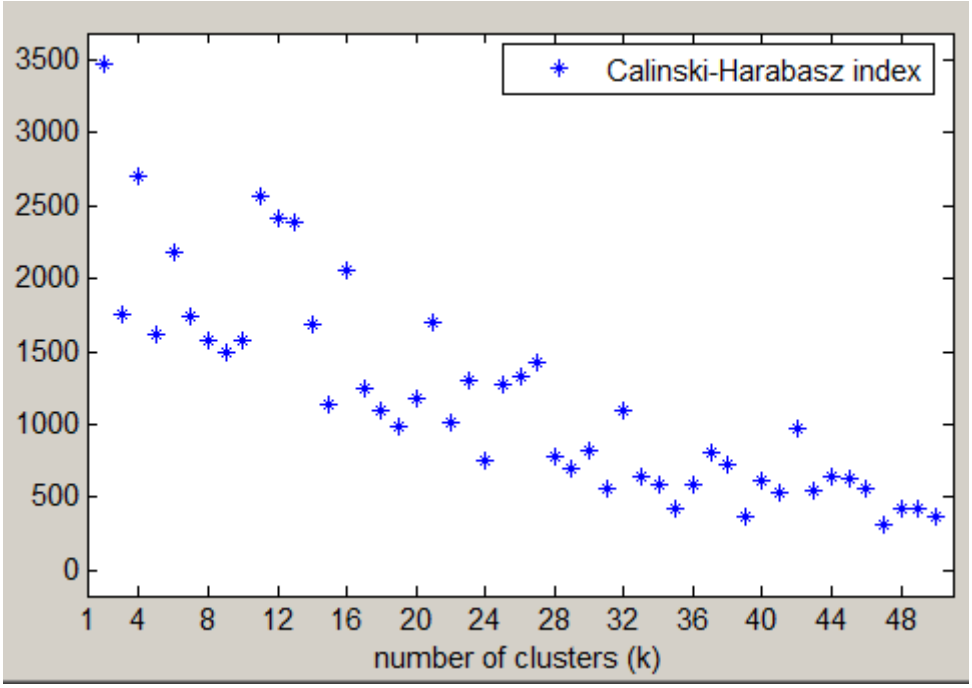
- 3 : es la multiplicidad del valor propio 0 y por lo tanto la cantidad de clusters básicas a encontrar.
- 7 : es el primer salto significativo entre las magnitudes del séptimo y el octavo valor propio, pudiendo considerarse como una opción de parada según los criterios vistos en el teorico.
- 13 : es el siguiente salto significativo más grande aun que el del séptimo y octavo valor propio

Estudio de bondades σ=0.00038

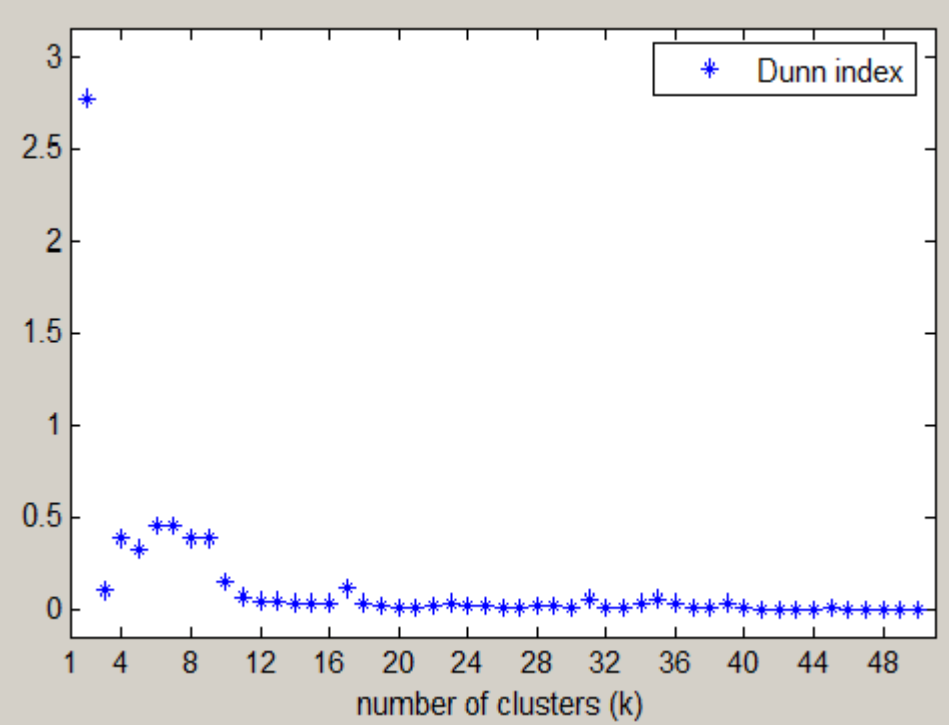
Luego, con el σ óptimo encontrado, aplicamos SC a los datos desde 2 a 50 clusters y le realizamos el estudio de bondad de clusters, para determinar la cantidad optima de clusters a seleccionar, previo a realizar el estudio de correspondencia de los clusters con los rubros. Además, realizamos una comparación entre la cantidad de clusters óptima surgida a partir del estudio de los valores propios, y la que se desprenderá del estudio de las bondades de los clusters.



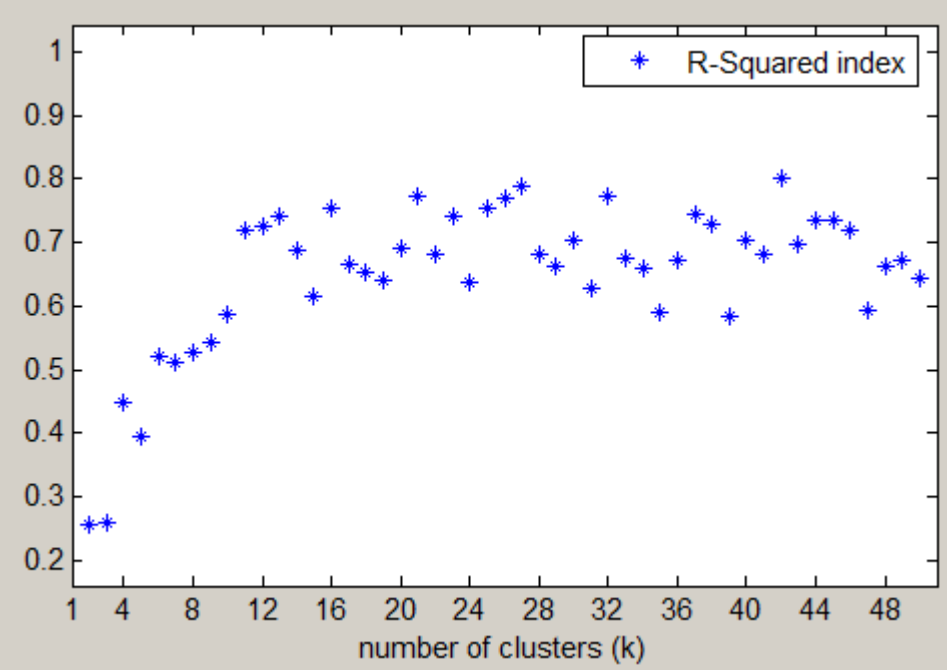
Minimiza para 2 clusters.



Maximiza para 2 clusters



Maximiza para 2 clusters



Observando la gráfica una buen elección sería 12 clusters.

Debido a que estas medidas de validación interna se basan en general en distancias y son muy mezquinas para Spectral clustering, decidimos basar la elección de la cantidad de clusters en el estudio de las gráficas de los valores propios, es decir realizaremos el estudio de correspondencia con las etiquetas para 3 y 7 clusters.

Resultados

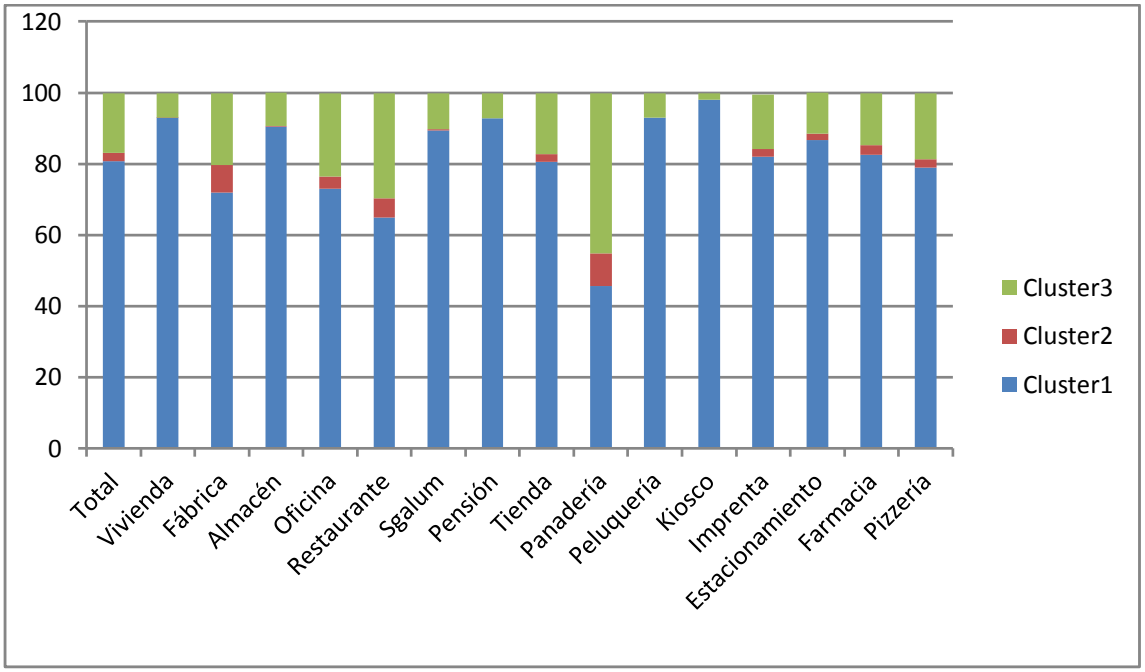
Tabla para 3 clusters (kmeans)

Rubro/Cluster	Total Datos	Cl1	Cl2	Cl3
Total	25986	80,8	2,3	16,7
Vivienda	3210	92,9	0,1	6,9
Fábrica	114	71,9	7,8	20,1
Almacén	580	90,5	0,17	9,3
Oficina	1341	73,0	3,5	23,3
Restaurante	475	65,0	5,4	29,4
Sgalum	761	89,3	0,5	10,1
Pensión	210	92,8	0	7,1
Tienda	170	80,5	2,3	17,0
Panadería	420	45,7	9,2	45,0
Peluquería	381	92,9	0	7,0
Kiosco	164	98,1	0	1,8
Imprenta	144	82,1	2,0	15,3
Estacionamiento	164	86,6	1,8	11,6
Farmacia	178	82,5	2,8	14,6
Pizzería	43	79,0	2,3	18,6

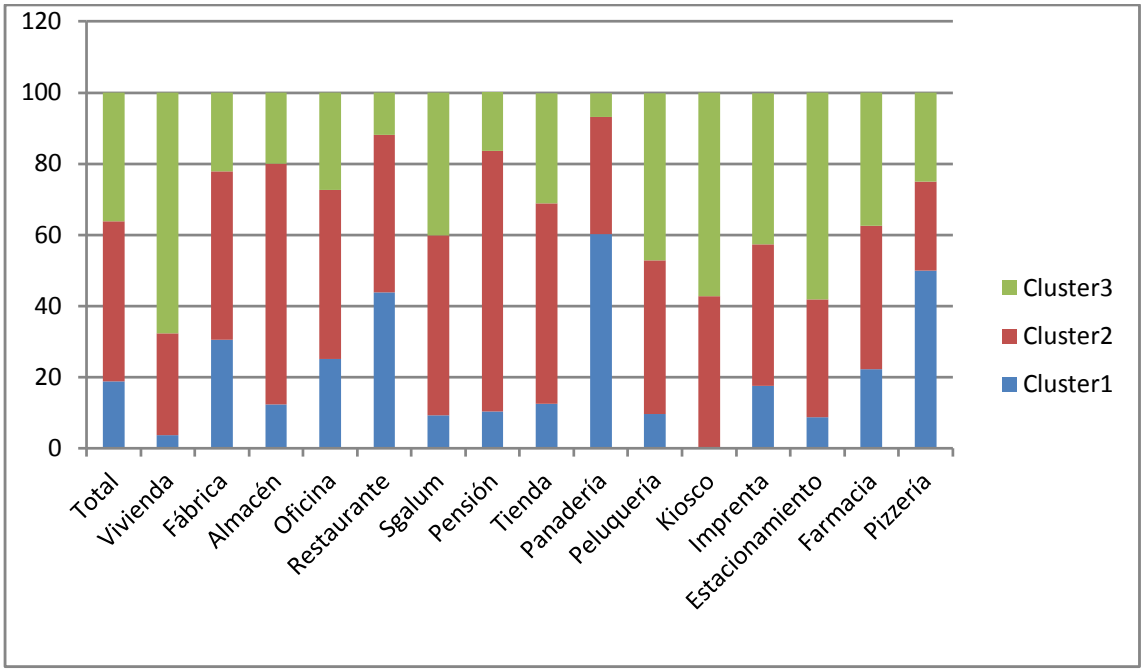
Tabla para 3 clusters (spectral)

Rubro /Cluster	Total Datos	Cl1	Cl2	Cl3
Total	10000	18,8	45,0	36,2
Vivienda	899	3,8	28,5	67,7
Fábrica	36	30,6	47,2	22,2
Almacén	170	12,4	67,6	20,0
Oficina	661	25,1	47,5	27,4
Restaurante	221	43,9	44,3	11,8
Sgalum	396	9,3	50,5	40,2
Pensión	164	10,4	73,2	16,5
Tienda	87	12,6	56,3	31,0
Panadería	161	60,2	32,9	6,8
Peluquería	187	9,6	43,3	47,0
Kiosco	89	0	42,7	57,3
Imprenta	68	17,6	39,7	42,6
Estacionamiento	91	8,8	33,0	58,2
Farmacia	72	22,2	40,3	37,5
Pizzería	8	50	25	25

Grafica 3 clusters(kmeans)



Grafica 3 clusters(spectral)



Observaciones: De la gráfica de k-means vemos que el porcentaje de muestras de cada rubro que cae en cada cluster es proporcional al tamaño de los mismos. Es decir que ninguna etiqueta se corresponde completamente con un cluster específico, salvo “kiosco” que concentra el 98.1% de las muestras en el mismo cluster. Lo que si podemos observar es que ciertos conjuntos de rubros se comportan de manera similar. Por un lado tenemos a “Sgalum”, “vivienda” y “almacen” en donde la distribución en clusters se asemeja a 90%-0.3%-9.7%. Por otro lado tenemos a “imprenta”, “pizzería” y “tienda” cuya distribución se acerca a 80%-2%-18%, que es muy parecida a la distribución total. También se destaca la igualdad de distribución entre “peluquería” y “pensión”, las cuales rondan alrededor de 92.8%-0%-7.2%.

A partir de la gráfica de spectral clustering también vemos que las distribuciones de los rubros no se diferencian significativamente de la distribución total en clusters. Solo se observa una similitud de distribuciones entre “fábrica” y “sgalum” las cuales rondan cerca de 28%-47,2%-22,2%.

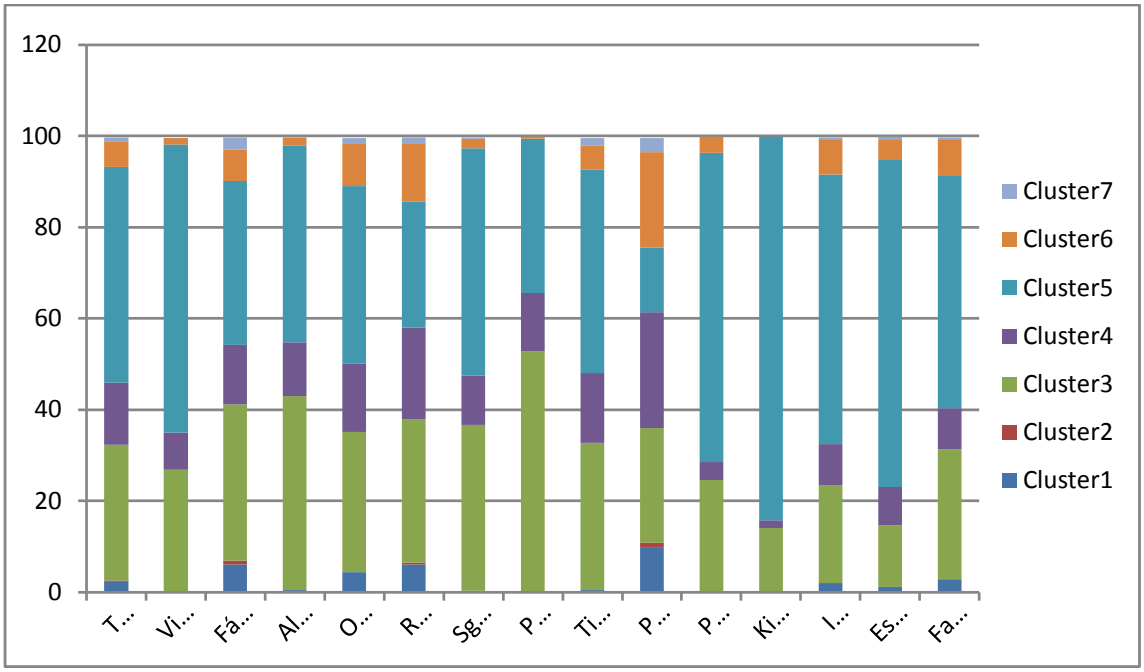
Tabla para 7 clusters(k-means)

	Total Datos	Cluster1	Cluster2	Cluster3	Cluster4	Cluster5	Cluster6	Cluster7
Total	25986	2.4	0.1	29.8	13.6	47.3	5.6	0.9
Vivienda	3210	0.1	0	26.8	8.1	63.1	1.5	0.0
Fábrica	114	6.1	0.8	34.2	13.1	35.9	7.0	2.6
Almacén	580	0.5	0	42.5	11.8	43.1	1.8	0
Oficina	1341	4.4	0.0	30.7	15.0	39.0	9.3	1.1
Restaurante	475	6.1	0.4	31.5	20.0	27.7	12.8	1.2
Sgalum	761	0.3	0	36.3	10.9	49.8	2.1	0.3
Pensión	210	0	0	52.8	12.8	33.8	0.4	0
Tienda	170	0.5	0	32.3	15.2	44.7	5.2	1.7
Panadería	420	9.7	1.1	25.2	25.4	14.2	20.9	3.0
Peluquería	381	0.2	0	24.4	3.9	67.9	3.4	0
Kiosco	164	0	0	14.0	1.8	84.1	0	0
Imprenta	144	2.0	0	21.5	9.0	59.0	7.6	0.6
Estacionamiento	164	1.2	0	13.4	8.5	71.9	4.2	0.6
Farmacia	178	2.8	0	28.6	8.9	51.1	7.8	0.5
Pizzería	43	2.3	0	23.2	16.2	48.8	6.9	2.3

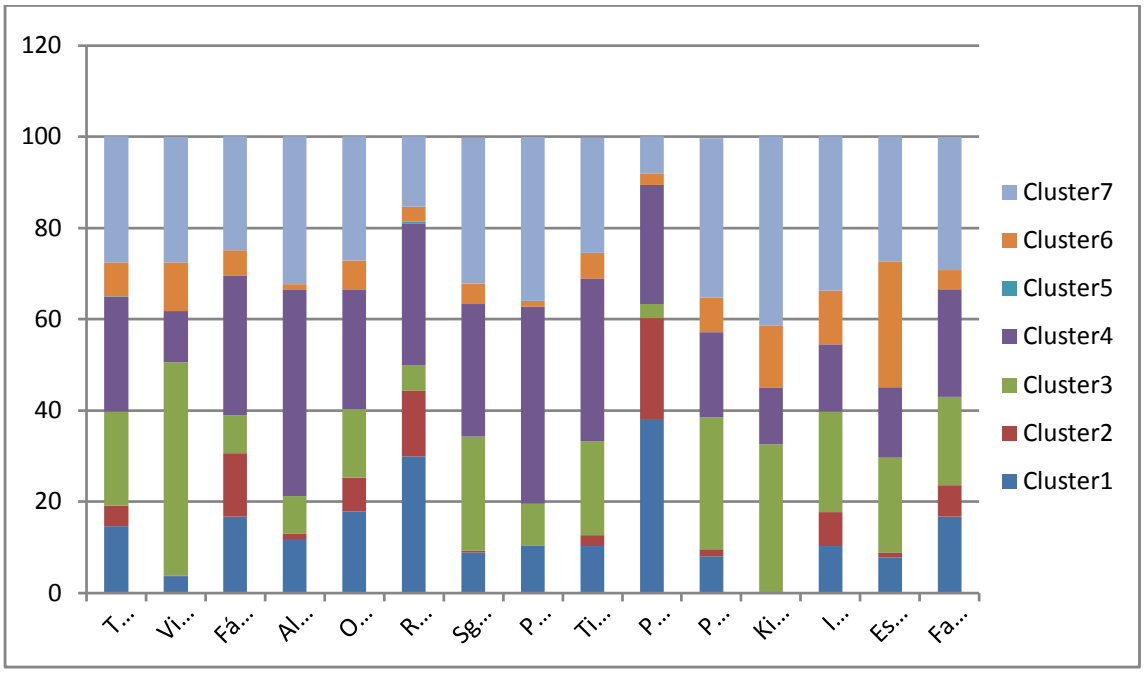
Tabla para 7 clusters (spectral)

Rubro /Cluster	Total Datos	Cluster1	Cluster2	Cluster3	Cluster4	Cluster5	Cluster6	Cluster7
Total	10000	14,6	4,5	20,7	25,1	0,1	7,4	27,7
Vivienda	899	3,7	0,1	46,7	11,3	0	10,7	27,5
Fábrica	36	16,7	13,9	8,3	30,6	0	5,6	25
Almacén	170	11,8	1,2	8,2	45,3	0	1,2	32,4
Oficina	661	17,9	7,4	15,0	26,2	0	6,4	27,2
Restaurante	221	29,9	14,5	5,4	31,2	0,5	3,2	15,4
Sgalum	396	8,8	0,5	25,0	29,0	0	4,5	32,1
Pensión	164	10,4	0	9,1	43,3	0	1,2	36,0
Tienda	87	10,3	2,3	20,7	35,6	0	5,7	25,3
Panadería	161	37,9	22,4	3,1	26,1	0	2,5	8,1
Peluquería	187	8,0	1,6	28,9	18,7	0	7,5	35,0
Kiosco	89	0	0	32,6	12,4	0	13,5	41,6
Imprenta	68	10,3	7,4	22,1	14,7	0	11,8	33,8
Estacionamiento	91	7,7	1,1	20,9	15,4	0	27,5	27,5
Farmacia	72	16,7	6,9	19,4	23,6	0	4,2	29,2
Pizzería	8	50	0	12,5	12,5	0	12,5	12,5

Grafica 7 clusters(kmeans)



Grafica 7 clusters (spectral)



Se observa que al aumentar la cantidad de clusters todos los rubros parecen distribuirse de diferente manera luego de aplicar cualquiera de los dos algoritmos. “Kiosco” para el caso de kmeans se corresponde en su 84.1% con un cluster, lo que no es tan destacable si tenemos en cuenta que ese cluster agrupa el 47.3% de las muestras totales. Cabe destacar también que el caso de spectral, la similitud de la distribución de “Almacén” y “Pensión” que rondan por los 10.7%-0.5%-9.0%-43.0%0.5%-1.2%-36%.

Conclusiones

Luego de visualizar el contenido de las tablas, no logramos establecer una correspondencia directa entre los grupos etiquetados y los clusters encontrados por kmeans y por Spectral. En prácticamente todos los rubros la cantidad de muestras que caen en cada clusters son casi proporcionales al tamaño de los mismos. Hubo algunas excepciones (panadería en Spectral y kiosco en kmeans). Para panadería luego de usar Spectral Clustering con 3 clusters, el 61% de las muestras se clasifican en un cluster que contiene solo el 18.8% de las muestras. Utilizando kmeans al clasificar con 20 clusters, en el rubro kiosco se agruparon el 75% de las muestras en dos clusters y el 77.4% de los estacionamientos se agruparon en 3 clusters. A pesar de la superioridad de desempeño que presentó Spectral clustering con los datos sintéticos, en este caso no logramos registrar mejores resultados al aplicarlo a la base de consumos de UTE, tampoco registramos una mejora luego de realizar un estudio para obtener la parametrización óptima. Los dos algoritmos planteados para este trabajo, generaron resultados considerablemente distintos, lo cual, nos muestra la magnitud de la diferencia en la forma de trabajar de ambas técnicas. Esto no demuestra una mejor performance de uno u otro, para estos datos en particular (aunque si vimos una clara mejora de spectral frente a kmeans con los datos sintéticos). Además al obtener resultados tan diferentes con los dos algoritmos, los resultados de los mismo no sean concluyentes, es decir, no encontramos ninguna correlación fuerte entre etiquetas y clusters definidos. Las medidas de validación interna de kmeans son mejores que las de Spectral clustering, pero esto no es fiable ya que se basan en cocientes de distancias y en general tienden a favorecer a kmeans tal como observamos con los datos sintéticos.

Referencias

- Trabajos sobre spectral clustering de Ulrike von Luxburg
  http://www.informatik.uni-hamburg.de/ML/contents/people/luxburg/
- Para k-means se utilizaron las diapositivas de clase
- Algunos datos sinteticos los obtuvimos de practicos del curso y otros de
  http://www.ima.umn.edu/~iwen/REU/REU_cluster.html