La idea sobre la que se basa la técnica de bootstrapping es que la inferencia sobre una población a partir de una muestra de datos puede ser modelada por medio de re-muestrear los datos de la muestra y llevar a cabo inferencias sobre un remuestro de la muestra. Como la población es desconocida el error verdadero en una muestra estadística sobre el valor de la población es desconocido. En el remuestreo en bootstrapping, la población es en realidad la muestra, la cual es conocida. Por tanto, la calidad de la inferencia de la muestra verdadera a partir de los datos muestreados es medible.

Como ejemplo, supongamos que nos interesa conocer la media de la altura de la población mundial. Debido a que es complejo medir la altura de todas las personas en el mundo, en su lugar podemos muestrear una parte de ellas y medir la altura en ellas. Supongamos que el tamaño de esta muestra es N, entonces tenemos la altura de N personas. Con esta muestra solo podemos tener una estimación de la altura media de la población. Para obtener una mejor imagen de la población necesitamos obtener algún tipo de variabilidad sobre los datos obtenidos. El método de bootstrap es el más sencillo para obtener está variabilidad y consiste en muestrear con remplazo una muestra de tamaño N. Por ejemplo, si muestreamos sobre [1,2,3,4,5] podríamos obtener [2,5,3,3,1]. Cuando el tamaño N es suficientemente grande para todos los efectos prácticos existe una probabilidad casi cero de que sea una muestra idéntica a la muestra inicial. Si se repite este proceso miles de veces y para cada muestra se obtiene la media, se obtiene un histograma de medias obtenido con bootstrap.

El método bagging, descrito a continuación, consiste en un meta-algoritmo para ponderar los resultados de múltiples muestras de bootstrapping.

Método bagging

El método de bagging es una clase de algoritmos de ensemble o combinación de clasificadores. El término bagging proviene de la contracción de bootstrap aggregation. Se trata de un procedimiento general que permite reducir la varianza de un método de machine learning. Es un método para combinar varias instancias de estimadores de caja negra que se han construido sobre muestras aleatorias del conjunto de entrenamiento original y que agregan las predicciones individuales para obtener una predicción única.

Dado un conjunto de n observaciones independientes Z1…Zn, cada una con una varianza , la varianza de la media Z de las observaciones es $/n$.

Es decir, la media de un conjunto de observaciones reduce la varianza. Sin embargo, como lo habitual es no tener muchas observaciones se suele hacer bootstrap tomando muestras repetidas del data set de entrenamiento.

Para ello, se entrenan B conjuntos de entrenamiento distintos y se obtiene la media de las predicciones:

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{b}\hat{f}^{*b}(x)$$

Estos métodos se utilizan como una forma para reducir la varianza de un estimador base (ejemplo: árboles de decisión) por medio de introducir aleatoriedad en el procedimiento de construcción y a continuación construir el ensemble.

En muchos casos, los métodos de bagging son un método sencillo de mejorar, un único modelo sin la necesidad de tener que modificar el algoritmo de entrenamiento base. Como se trata de un método para reducir el overfitting, los métodos de bagging funcionan mejor cuando se utilizan con modelos complejos (ejemplo: árboles de decisión profundos), en contraste con los métodos de boosting que funcionan mejor con modelos simples. (ejemplo: shallow decisión trees)

Existen diferentes variantes de los métodos de bagging, pero dependen principalmente de la forma en que obtienen las muestras del conjunto de entrenamiento:

- Cuando se obtienen muestras aleatorias del conjunto de datos como subconjuntos aleatorios de las muestras, el algoritmo se conoce como pasting (Breiman, 1999).
- Cuando se obtienen los conjuntos para entrenar los algoritmos por medio de muestreo con remplazo del conjunto de entrenamiento, el método se conoce como bagging (Breiman, 1998).
- Cuando se obtienen subconjuntos aleatorios como subconjuntos del espacio de las variables, el método se conoce como random subspaces (Kam, 1998).
- Por último, cuando los estimadores base se construyen con subconjuntos de muestras y variables, el método se conoce como random patches (Louppe. Y Geurts, 2012).

Out-of-bag (OOB) error

Existe una forma sencilla de estimar el error de prueba en un modelo bagged, por medio del conjunto de datos que se queda fuera de la muestra. Este conjunto de datos fuera de la muestra, por lo general, suele ser 1/3 del conjunto de datos total. Por tanto, para este 1/3 del conjunto total se puede predecir la respuesta que se hubiera obtenido. Cuando el valor B (número de conjuntos de entrenamiento distintos) es grande, la estimación del out-of-bag error es equivalente al leave-one-out cross validation.

Método de boosting

Al igual que bagging, se trata de un método de combinación de modelos que se puede aplicar a los modelos de regresión y de clasificación.

En bagging se crean múltiples copias del conjunto de entrenamiento original utilizando el muestreo bootstrap y entrenando un modelo en cada copia, para posteriormente combinar todos los modelos en uno solo. En bagging cada modelo se construye en un bootstrap data set independiente de los otros. Sin embargo, en el caso de boosting los modelos se generan de forma secuencial, es decir, cada modelo utiliza información de los modelos anteriores.

Boosting es la técnica seguida por los modelos de generalized boosted models (GBMs). En estos modelos, en lugar de entrenar un gran árbol de decisión sobre los datos, lo cual es complicado y puede dar lugar a overfitting, se utiliza el enfoque boosting para aprender poco a poco. Cada uno de los árboles puede ser bastante pequeño, con pocos nodos terminales, pero todos ellos se combinan.

Por tanto, el método boosting puede verse como un meta-algoritmo de ensemble para reducir sesgo y varianza el cual se aplica sobre algoritmos de machine learning supervisados con el objetivo de convertir modelos simples en modelos más precisos.

Este método se basa en una pregunta formulada por Kearns y Valiant (año 1989): ¿Puede un conjunto de modelos débiles crear un modelo fuerte?

En este contexto un modelo débil se define como un clasificador que está poco correlacionado con la clase real. Se trata de ejemplos ligeramente mejor que el acierto aleatorio. Por otro lado, un modelo fuerte se define como un clasificador que está bien correlacionado con la clase verdadera de la clasificación.

La respuesta a la pregunta de Kearns y Valiant ha tenido una gran repercusión en el mundo de machine learning dando lugar al desarrollo de los modelos de boosting.

Cuando se introdujo esta hipótesis de problemas boosting dio lugar a que los algoritmos que consiguieran mejorar por medio de la combinación de modelos débiles fueran llamados boosting.

A pesar de que los algoritmos boosting no tienen ninguna restricción algorítmica, la mayoría de estos algoritmos consisten en ir aprendiendo modelos débiles iterativamente con respecto a una distribución. Cada vez que se añade un nuevo modelo débil, los datos son re-calibrados: los ejemplos que estaban correctamente clasificados pierden peso y los ejemplos incorrectamente clasificados ganan peso.

Tipos de algoritmos boosting

Tema 3. Algoritmos de aprendizaje automático supervisado

Existen diversos algoritmos de tipo boosting. El algoritmo original fue propuesto por Yoav Freund y Robert Schapire en 1997, el cual era no adaptativo y no explotaba de forma completa la ventaja de los modelos débiles. Posteriormente, Freund y Schapire desarrollaron el algoritmo AdaBoost, un algoritmo de boosting adaptativo que ganó el prestigioso premio Gödel.

La principal variación entre los algoritmos de boosting es el método de ponderar los datos de entrenamiento y las hipótesis. AdaBoost es muy popular e históricamente el primero que fue capaz de adaptarse a los modelos débiles. Sin embargo, hay más algoritmos de boosting como: el LPBoost, BrownBoost, TotalBoost, LogitBoost y xgboost.

Hemos hecho hasta aquí, un recorrido por los principales enfoques y sus algoritmos en lo referente al aprendizaje automático supervisado, sin embargo a cualquiera medianamente vinculado con el área se daría cuenta de la ausencia de uno de los iconos de estos tiempos, la estrella del gran show del aprendizaje automático, me refiero a las redes neuronales. Bueno en esta lección no la vamos a tocar por la simple razón que hemos preparado dos lecciones para tratar las redes neuronales clásicas y las redes de aprendizaje profundo.

Conclusiones

Para concluir, solo recomendarles profundizar todo lo que puedan en los temas de álgebra si así lo consideran.

Hemos tocado los principales enfoques, aunque el tema es basto y se han desarrollado cientos de algoritmos y nos es imposible cubrir todo lo que pensamos que es importante. Aun así creemos que hay material para que inicien este fascinante viaje por el mundo de aprendizaje automático.

Tema 4. Algoritmos de aprendizaje automático no supervisado

4.1. Algoritmos de aprendizaje automático no supervisado I

Esta unidad formativa está compuesta por dos lecciones que tratan el tema del aprendizaje automático no supervisado.

En la primera lección se trata de manera inicial las métricas de distancia, concepto que es la piedra angular de los métodos de aprendizajes no supervisados.

Luego se tratan los algoritmos de clustering jerárquicos para continuar con métodos como K-Means, Affinity Propagation, Mean Shift, Spectral Clustering, DBSCAN y por ultimo BIRCH.

La segunda lección la dedicamos exclusivamente a las métricas de validez de los resultados de los algoritmos de aprendizaje automáticos no supervisados, como son: Calinski-Harabasz Index. Silhouette Coefficient y Davies-Bouldin Index conocidos como criterios de validez interna y luego se tratan los conocidos como criterios externos: Rand Index, Medidas basadas en Información Mutua, Homogeneidad, Completitud y V-Measure, ¡Error! Marcador no definido.

En esta lección trataremos todo lo referente a los métodos de aprendizaje automático no supervisado. En la lección anterior tratamos las métricas de evaluación de los métodos de aprendizaje automáticos supervisados de regresión y los métodos supervisados para clasificación y en esta, siguiendo ese mismo estilo, inicialmente veremos las métricas de evaluación y a continuación los algoritmos.

El objetivo es, que al final de estas dos lecciones, usted esté en condiciones de aplicar los métodos de aprendizaje no supervisado a problemas prácticos y domine los conceptos esenciales que les permita profundizar luego en la teoría y práctica de estos métodos.

Tema 4. Algoritmos de aprendizaje automático no supervisado

Los objetivos que se persiguen son:

- Conocer y aplicar las principales métricas de distancia.
- Conocer y aplicar métodos de agrupamientos como K-Means, Affinity Propagation, Mean Shift, Spectral Clustering, DBSCAN y por ultimo BIRCH.
- Conocer y aplicar los criterios de evaluación de los resultados de los algoritmos de agrupamientos.

Introducción

En el aprendizaje supervisado dada una serie de etiquetas se “aprendía” un mapeo para obtenerlas. En el caso del aprendizaje no supervisado el problema consiste en probar y determinar la estructura existente en los datos, pero sin utilizar una etiqueta previa. Estos algoritmos, también se conocen con el nombre de algoritmos de agrupamiento o clustering, puesto que agrupan instancias de los datos en función de las variables de los conjuntos de datos. Es decir, este tipo de algoritmos buscan patrones en los datos con el objetivo de encontrar agrupaciones o clúster.

Estas técnicas se utilizan cuando se desconoce la estructura de los datos puesto que no se tiene la variable objetivo de los datos. Por ejemplo, cuando se desconoce cuántos grupos de usuarios similares existen.

Estos métodos dividen los datos en grupos similares. Pero esta división se lleva a cabo sin la necesidad de indicar las características de cada uno de estos grupos. Para este objetivo, las instancias de un grupo deben de ser muy similares entre sí, pero muy distintas a las de otros grupos. Por tanto, para establecer esta distinción entre casos y grupos es imprescindible que definamos unas métricas de similaridad entre objetos y cómo usarla en el contexto de la solución del problema.

La clave de estos algoritmos consiste en buscar buenas variables capaces de distinguir entre las diferentes instancias o registros.

Tema 4. Algoritmos de aprendizaje automático no supervisado

Los algoritmos de agrupamiento se pueden utilizar para diversos objetivos, tales como:

- **El aprendizaje estadístico semisupervisado:** es aquel que compagina los problemas de clasificación estadística junto con el análisis de grupos.
- **Filtrado colaborativo:** El agrupamiento proporciona un resumen de usuarios con una mentalidad parecida. Las calificaciones dadas por cada individuo para los demás son usadas para realizar dicho proceso, obteniéndose así sistemas de recomendación.
- **Segmentación de clientes para agrupamiento de atributos arbitrarios:** Se establece como objetivo para adecuar las características de los productos, a las necesidades de los clientes, utilizándose en el Marketing, la segmentación de mercados, el diseño de producto y la toma de decisiones sobre la tecnología de fabricación.
- **Minería de datos:** Extrae conocimiento de grandes bases de datos ya sean relacionales o no estructuradas, como la información de las páginas web. Puede detectar tendencias y patrones, así como la detección de datos atípicos.
- **Resumen de datos y reducción de la dimensionalidad:** El primero permite crear representaciones de datos compactos que se procesan e interpretan más fácilmente. El segundo elimina datos irrelevantes y atributos redundantes. La extracción de características proyecta los datos en un espacio de dimensionalidad más reducida, como ejemplos están el Análisis de Componentes Principales, la Descomposición del Valor y el Análisis Lineal Discriminante. La selección de características selecciona un subconjunto reducido de aquellas, que minimiza la redundancia y maximiza la relevancia del atributo. Destacan entre otros la divergencia de Kullback-Leibler y el Lasso.

- **Detección de tendencias dinámicas para datos disponibles en streaming:** Con la creciente utilización de toma de datos en tiempo real a través de sensores, cámaras y datos tomados de internet, pueden detectarse tendencias que se produzcan en el tráfico o climatología, por ejemplo. El algoritmo ha de ser adaptable y de fácil escaneo.

- **Análisis de datos multimedia:** Las imágenes de resonancia magnética permiten conocer la estructura interna de objetos y organismos biológicos. Por ejemplo, determinar la existencia de segmentos similares en las imágenes y realizar la segmentación de las mismas. Esto puede hacerse, por ejemplo mediante un agrupamiento jerárquico. También pueden emplearse las k medias, o también algoritmos borrosos o Fuzzy.

- **Agrupamiento de documentos, basado en su contenido:** Se utiliza en el almacenamiento y búsqueda automática de documentos. Separa los documentos en grupos que reflejan el contenido de cada documento. Estos algoritmos, han de ser computacionalmente eficientes, dado el elevado volumen de datos manejado.

- **Análisis de datos biológicos:** Se ha aplicado en la secuenciación del genoma humano. También se han agrupado secuencias lineales de genes y proteínas con niveles similares de expresiones, estableciendo taxonomías biológicas.

- **Análisis de las redes sociales:** Se utiliza la estructura de la red social para determinar las comunidades importantes de la red subyacente. Esto permite una buena comprensión de la estructura de la comunidad en la red. El agrupamiento también permite resumir las características de la red social.

Los algoritmos de aprendizaje no supervisado se pueden dividir de muchas maneras y es un área de investigación activa, sin embargo de manera general existe consenso en que estos se pueden agrupar en:

- **Particionales:** tienen definidos de antemano un número de grupos. Son algoritmos iterativos que comienzan con una asignación inicial y se van modificando siguiendo

un criterio de optimización.

- **Jerárquicos:** En cada iteración solo un objeto cambia de grupo y los grupos están anidados en los de los pasos anteriores. Si un objeto ha sido asignado a un grupo ya no vuelve a cambiar.

- **Bayesianos:** No los veremos en esta lección.

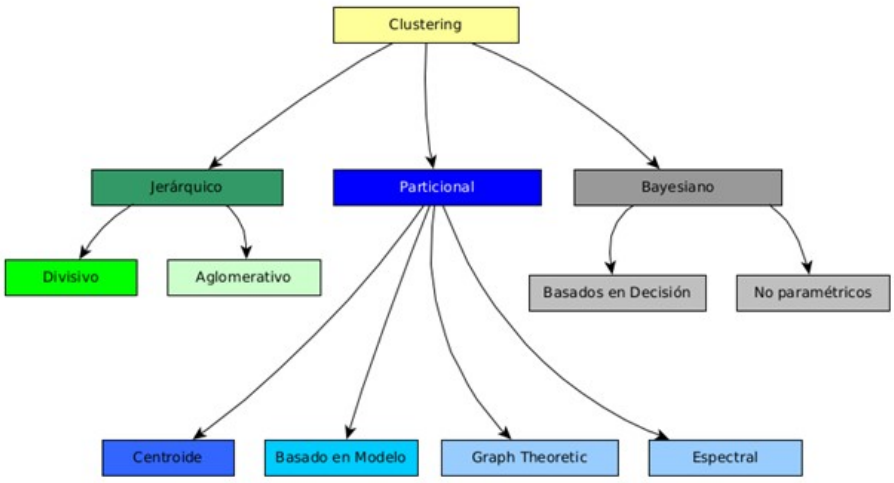


Figura 1. Taxonomía del área de Clustering.

En esta lección y en la que sigue, solo tocaremos los métodos que están en colores en la figura anterior, no tocaremos los que están en color gris.

Aunque esta taxonomía no es completa y hay muchos métodos nuevos, creemos que es importante en una lección de introducción al tema, tocar aquellos métodos que son considerados el núcleo del área de clustering.

Matemáticamente podemos definir el problema de Clustering de la siguiente manera:

Dado un conjunto de datos X de tamaño n y un número positivo k, tal que:

$$k \leq n$$

Encontrar una partición de X en k subconjuntos tal que:

$$S_1 \cup S_2 \ldots \cup S_k = X$$

Cumpliéndose que:

$$k = 1 k S_k = \emptyset$$

Es decir que los casos solo deben pertenecer a un solo subconjunto de la partición. A este tipo de modelo se les llama crisp, sin embargo los modelos basados en lógica difusa, como Fuzzy C-Mean, los casos pueden pertenecer a más de un conjunto con cierto grado de pertenencia.

En muchas ocasiones este número k es imposible de conocer a priori y no se tiene información sobre la cantidad de grupos que puede contener el conjunto X. La única forma en que podemos atacar el problema del agrupamiento es definiendo alguna medida que nos permita comparar cada uno de los casos en X con la esperanza de encontrar grupos que nos expliquen la estructura de X.

Para eso, se han desarrollado métricas, sobre las que descansan los métodos de agrupamientos desarrollados.

Métricas de distancia y similaridad

Entonces, sabemos que para poder encontrar grupos o determinar la estructura de un conjunto de datos X, es necesario definir de menara clara, la forma en que comparamos cada $x \in X$. De manera general podemos definir una métrica sobre el conjunto X como $d : X * X \rightarrow R^+$, tal que d cumpla las siguientes propiedades:

- No negatividad: $d(p,q) \geq 0, \forall p,q \in X y \forall p \in X : d(p,p) = 0$
- Sea simétrica: $d(p,q) = d(q,p), \forall p,q \in X$
- Desigualdad Triangular: $d(p,r) \leq d(p,q) + d(q,r), \forall p,q,r \in X$

Entre las medidas de similaridad más comunes tenemos:

Distancia Euclidiana: Es la raíz cuadrada de la suma de las diferencias al cuadrado entre los componentes de los dos vectores o casos que pertenecen a X.

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

Distancia Manhattan: Es la suma de los cuadrados de los valores absolutos entre las diferencias de los componentes del vector o caso que pertenecen a X.

$$d(p,q) = \sum_{i=1}^{n} |p_i - q_i|^2$$

Distancia máxima:

$$d(p,q) = max\,|p_i - q_i|$$

Distancia Minkowski: Esta distancia es una generalización de la distancia euclidiana y depende de un parámetro r, que es un real positivo. Si r=2, entonces esta distancia es equivalente a la euclidiana. Si r=1 entonces esta métrica es equivalente a la distancia Manhattan.

$$d(p,q) = \left(\sum_{k=1}^{n} |p_k - q_k|^r \right)^{1/r}$$

Distancia Coseno: Básicamente esta métrica es el producto escalar de dos vectores, normalizada con el producto de sus módulos que se resta de 1. Sería 1 menos la similaridad coseno es la distancia coseno.

$$d(p,q) = 1 - \frac{p * q}{\|p\| \, \|q\|} = 1 - \frac{\sum_{i=1}^{n} (pi * q_i)}{\sqrt{\sum_{i=1}^{n} p_i} \sqrt{\sum_{i=1}^{n} q_i}} = 1 - \cos\theta$$

El coseno del ángulo entre dos vectores p y q, es una medida de similaridad. Por tanto para convertirla en una medida de distancia se hace 1-cos(p,q).

Estas medidas de distancia, asumen que los componentes de cada uno de los vectores son cuantitativos, es decir, cada caso está compuesto por rasgos numéricos. Sin embargo, esto no siempre es así y podemos encontrarnos con muchos conjuntos de datos que tienen rasgos que son más como etiquetas o con rasgos mesclados.

En el caso de datos con rasgos cualitativos, podemos usar la medida de disimilaridad como la medida de distancia de Hamming generalizada igual al número de estos rasgos cuyos valores para los casos comparados son diferentes.

Cuando tenemos que tratar con datos mesclados, es decir, donde tenemos rasgos cualitativos y cuantitativos se puede usar el coeficiente de Gower.

Un análisis completo de las medidas de distancia, similaridad o disimilaridad, se nos escapa, por ser esta una lección de introducción al tema del clustering.

En la práctica no hay un criterio establecido de cuando escoger una medida de distancia u otra. Y la variedad es grande y sigue creciendo. Una de las medidas más usadas en la Euclidiana, aunque es muy sensible a ruidos y tiene dificultades cundo se usa en conjuntos de datos con una alta dimensionalidad.

Métodos de Clustering Jerárquicos

Estos métodos están entre los métodos tradicionales de clustering y consisten en agregaciones o separaciones sucesivas de los casos. El resultado de este tipo de método es una estructura de tipo árbol que se le denomina dendograma.

En el caso de las técnicas aglomerativas, inician viendo cada uno de los casos como un clúster y sucesivamente va encontrando los más cercanos y los va agrupando hasta que no queda más que agrupar.

Algoritmo Clustering Aglomerativo:

Entrada: $\boldsymbol{X = (x_1, x_2, \dots x_n)^T}$

Salida: $\boldsymbol{C = \{c_1, c_2, \dots, c_{2n-1}\}}$ **Dendograma.**

1. **Inicialización: Establecer a cada caso n de X como un clúster y medir la distancia entre cada uno de ellos. Construir la matriz de distancia** $\boldsymbol{D = d|x_{ij}|}$
2. **Encontrar un par de** $\boldsymbol{C_i, C_j}$ **que sean los más cercanos de todos.**
3. **Formar un nuevo clúster** $\boldsymbol{C_k = C_i \cup C_j}$ **.Esto corresponde a añadir un nuevo nodo en el dendograma y conectarlo con los otros nodos correspondientes a** $\boldsymbol{C_i, C_j}$**.**
4. **Actualizar la matriz de distancias, calculando ahora la distancia de** $\boldsymbol{C_k}$ **al resto de los nodos, excepto a** $\boldsymbol{C_i, C_j}$**.**
5. **Eliminar de la matriz de distancia las filas y columnas de** $\boldsymbol{C_i, C_j}$ **y añadir las correspondientes a** $\boldsymbol{C_k}$**.**
6. **Repetir los pasos del 2 al 5 hasta que quede un solo clúster.**

Este seudocódigo nos permite definir a muy groso modo como se procede con este tipo de método, sin embargo el paso 3 esconde detalles especiales y es en donde se concentras las principales diferencias entre los algoritmos que pertenecen a esta clase.

Este paso 3, es implementado de diferentes formas, veamos:

- En el método conocido como de enlace simple (single Linkage) la distancia entre dos clúster es igual a la distancia entre los dos elementos más cercanos que pertenezcan a los clúster. (este método se conoce como de los vecinos cercanos)

- En el método conocido como enlace completo (Complete Linkage) o método de los vecinos lejanos, la distancia entre dos clúster se toma como la distancia entre los dos casos más lejanos pertenecientes a clúster diferentes.

- En el método de enlace promedio (Average Linkage) la distancia entre dos clúster se toma como la distancia promedio de todos los casos que pertenecen a ambos clusters.

- En el método Promedio ponderado del par-grupo (weighted pair-group average, WPGA) la distancia es calculada similar a la anterior; pero los cálculos se hacen con los pesos igual al número de casos en cada clúster.

- En el método de los Centroides (unweighted pair-group centroid, UPGC) la distancia es igual a la distancia entre sus centroides o centros de gravedad.
- En el método conocido como de centroides ponderados (weighted pair-group centroid, WPGC) la distancia se calcula desde los centroides de los clúster aunque en este se usa el número de casos en cada clúster como peso.
- Este método conocido como el de mínima varianza, se minimiza la suma de los cuadrados entre el objeto y el centro del clúster al que pertenece.

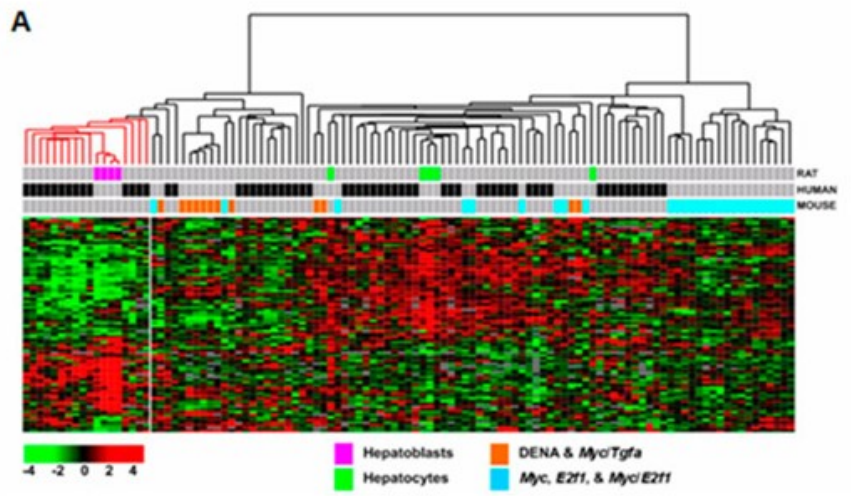


Figura 2. Ejemplo de dendograma y su mapa de calor para un particionamiento jerárquico.

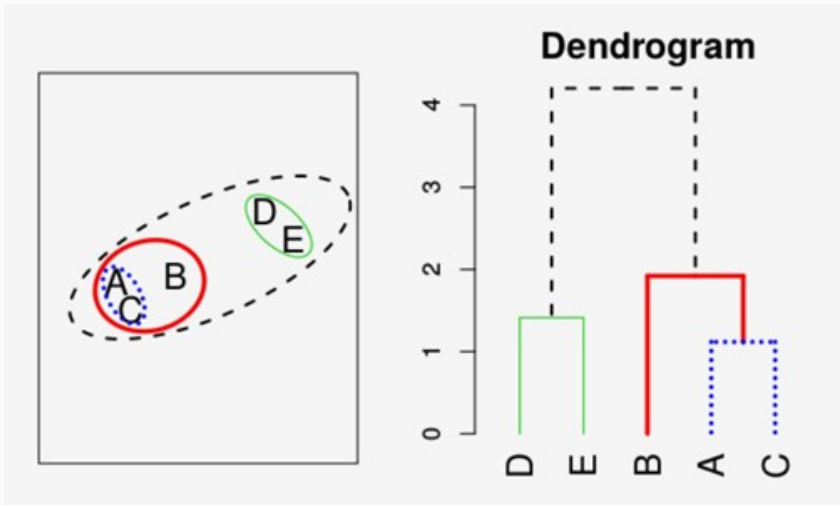


Figura 3. Ejemplo de un dendograma y su correspondiente agrupamiento.

A diferencia del método anterior, las técnicas divisivas el análisis comienza con un solo clúster formado por todos los casos de X, el cual se va dividiendo sucesivamente hasta que casa caso está contenido en su propio clúster o se llega al criterio de parada establecido.

Algoritmo de Clustering Divisivo

Entrada: $X = (x_1, x_2, \dots x_n)^T$

Salida: $S_1, S_2, \dots S_n$

1. Calcular la matriz de distancia basado en una métrica.
2. Dividir el clúster en dos, encontrando el caso más disimilar en cuanto a disimilaridad promedio y poniéndolo en el nuevo clúster, luego se recalcula las métricas de disimilaridad promedio y aquellos casos con valores positivos de mueven también.
3. Se calcula el diámetro de los clusters y se selecciona el de mayor valor para dividirlo.
4. Se repiten los pasos del 2 al 3 hasta que cada clúster contiene un solo caso.

Este seudocódigo pertenece al algoritmo DIANA o DIvisive ANAlisys. Estos métodos son muy poco tratado en la literatura y cuando la encuentras el algoritmo no está bien detallado o con ejemplo. Con vistas a ayudar a aclarar cómo se hace este tipo de procesamiento veremos una explicación más o menos detallada basándonos en el ejemplo dado por LEONARD KAUFMAN y PETER J. ROUSSEEUW en el libro

donde introducen el método: Finding Groups in Data: An Introduction to Cluster Analysis.

El ejemplo que pondremos es el mismo del libro antes mencionado y por tanto reproducimos aquí los detalles.

EL ejemplo comienza con una matriz de disimilaridad entre los casos {a,b,c,d,e}.

La matriz es la que sigue:

	a	b	c	d	e
a	0.0	2.0	6.0	10.0	9.0
b	2.0	0.0	5.0	9.0	8.0
c	6.0	5.0	0.0	4.0	5.0
d	10.0	9.0	4.0	0.0	3.0
e	9.0	8.0	5.0	3.0	0.0

Como se trata de algoritmo divisivo, asume que todos los datos están inicialmente en el mismo clúster {a,b,c,d,e}.

En el primer paso, como vimos en la descripción del método, el algoritmo tiene que dividir el clúster en dos. Para hacerlo, se busca el caso con la mayor disimilaridad promedio. Cuando hay dos o más objetos con la misma, toma uno al azahar. Para poder hacerlo es necesario calcular las disimilaridad promedio entre todos los casos.

	Disimilaridad uno vs el resto
a	(2.0+6.0+10.0+9.0)/4 = 6.75
b	(2.0+5.0+9.0+8.0)/4 = 6.0
c	(6.0+5.0+4.0+5.0)/4 = 5.0
d	(10.0+9.0+4.0+3.0)/4 =6.50
e	(9.0+8.0+5.0+3.0)/4 = 6.25

Es evidente que el caso con la disimilaridad promedio más alta es a, entonces se añade a un grupo que llaman Splinter. Por tanto en esta etapa ya tenemos los dos primeros grupos: {a} y {b,c,d,e}. Ahora hay que seguir añadiendo elementos al nuevo conjunto. Para ello se calcula de nuevo la disimilaridad promedio de los objetos del grupo que contiene más elementos y se compara con las disimilaridad del grupo Splinter, que aún es {a}. Veamos cómo queda la tabla.

Objeto	Disimilaridad promedio	Disimilaridad promedio del grupo. Splinter	Diferencia
b	(5.0+9.0+8.0)/3=7.33	2.0	5.33
c	(5.0+4.0+5.0)/3=4.67	6.0	-1.33
d	(9.0+4.0+3.0)/3=5.33	10.0	-4.67
e	(8.0+5.0+3.0)/3=5.33	9.0	-3.67

Teniendo en cuenta los nuevos valores recalculados son el caso a y comparado con las disimilaridades de la tabla original, entonces el único que tiene un valor positivo es b, por tanto lo añadimos al Splinter. Ahora nos queda dos grupos: {a,b} y {c,d,e}.

El próximo paso es encontrar cuál de los grupos hay que separar de nuevo y para eso se calcula el diámetro de cada clúster. El diámetro de un conjunto es igual a la

disimilaridad más grande entre dos elementos del clúster. Teniendo en cuenta esto:

D({a,b}) = 2.0

D({c,d,e}) = 5.0

¿De dónde sale esto?, pues de las disimilaridades de la matriz de disimilaridad comparando {c,d,e}.

Esto nos dice que el clúster a dividir es {c,d,e}. De nuevo se repite el paso anterior. Es decir, se escoge el caso con la mayor disimilaridad promedio entre c,d,e:

Objeto	Disimilaridad Promedio
c	(4.0+5.0)/2=4.5
d	(4.0+3.0)/2=3.5
e	(5.0+3.0)/2=4.0

Teniendo en cuenta el resultado del cálculo de la disimilaridad promedio entonces el caso c, es el de mayor disimilaridad, entonces se hace otro clúster con él y el Splinter nos quedaría: {a,b},{c,d,e},{c}. Pero aún no terminamos, hay que ver que otros casos pueden añadirse a este nuevo clúster.

Para hacerlo, como hicimos en el paso anterior, se comparan las disimilaridades de los casos del clúster con los del Splinter.

Objeto	Disimilaridad promedio	Disimilaridad promedio del grupo. Splinter	Diferencia
d	3.0	4.0	-1.0
e	3.0	5.0	-2.0

Estas medidas salen de comparar d y e en la matriz de disimilaridad y contra los casos en el Splinter.

Como todos los casos tienen diferencia negativa, entonces no se puede hacer nada y

terminamos con el proceso y nos quedas los clúster de la siguiente forma:{a,b},{c} y {d,e}.

En el nuevo paso, chequeamos si podemos dividir de nuevo, los clúster, de la misma forma que lo habíamos hecho en los pasos anteriores. Buscamos el clúster con el diámetro mayor, veamos cómo lo hacemos:

Obviamente el clúster {c}, no puede dividirse, mientras que el clúster {a,b} tiene diámetro 2, mientras que {d,e} tiene diámetro 3. Por tanto hay que dividirlo.

La matriz de similaridad de estos dos casos es:

	d	e
d	0.0	3.0
e	3.0	0.0

Como las disimilaridades son las mismas podemos escoger cualquiera de los casos. Por tanto podemos escoger el caso d para formar el otro clúster, entonces ahora el Splinter queda compuesto por:{a,b},{c},{d} y {e}.

Por tanto, en el próximo paso, de nuevo calculamos los diémetros de los clúster y como tres de ellos son singletons, nos queda solo dividir {a,b} en dos clúster singletons más, y al terminar este paso nos quedamos con el Splinter igual a: {a},{b}, {c},{d} y {e}.

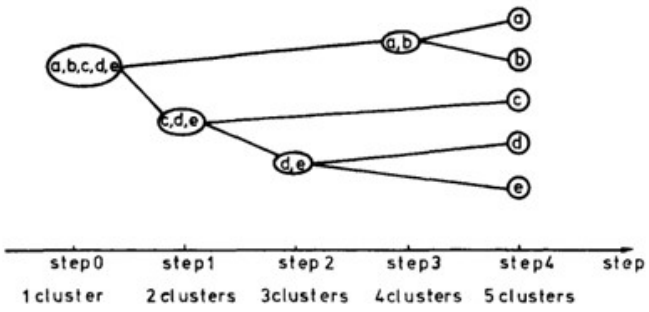


Figura 4. Dendograma para el ejemplo anterior.

Es importante destacar que el ejemplo que acabamos de exponer pertenece a los llamados métodos top-down y que AGNES otro algoritmo que puede considerarse gemelo de DIANA pero que trabaja de la forma opuesta, es considerado como un modelo Botton-up.

AGNES es sinónimo de HAC, mientras que DIANA los HDC. Que significan Hierarchical Agglomerative clustering y Hierarchical Divisive Clustering.

Algoritmos de Clustering

En esta sección iremos explicando, aunque de manera muy superficial el funcionamiento de algunos de los algoritmos y veremos ejemplos de su uso en Python.

Los métodos que veremos en esta sección son los siguientes:

- K-Means.
- Affinity Propagation.
- Mean Shift.
- Spectral clustering.
- DBSCAN.
- BIRCH.

Comenzaremos con el análisis de estos algoritmos y en la próxima lección veremos los métodos de validación de los resultados de estos.

K-Means

Este es uno de los algoritmos de clustering más usado y del cual abunda documentación, por eso lo tocamos primero.

Este algoritmo se puede clasificar como basado en centroide. La idea detrás de este método es minimizar la suma cuadrada de las distancias de los casos dentro del clúster.

Esto sería: Dado un conjunto de casos $X = (x_1, x_2, \ldots x_n)^T$ encontrar una partición $S_1, S_2, \ldots S_n \, tal \, que \, s_n \, S_{n-1} = \emptyset$ que minimice la inercia.

Entonces el papel del algoritmo es encontrar la distribución de todos los casos entre los clúster que minimice la suma de los cuadrados de las distancias entre cada caso y centroide del clúster al que pertenece.

Digámoslo de otro modo. K-Mean divide el conjunto de N casos en K clúster disjuntos Ck, cada uno descrito por la media mj de los casos en el clúster. Estas medias son conocidas como «centroides»; Es importante notar que estos centroides no son casos concretos de X.

Entonces, K-mean elije los centroides de forma que minimicen la «Inercia» o criterio de la suma cuadrada intra-clúster.

El algoritmo consiste en asignar cada uno de los, n casos a uno de los k clusters, donde k es un número definido previamente. El objetivo es minimizar las diferencias entre los grupos de cada clúster y maximizar las diferencias entre clusters.

A menos que k y n sean extremadamente pequeños no es factible calcular los grupos óptimos entre todas las combinaciones posibles de ejemplos. En su lugar, el algoritmo utiliza un proceso heurístico para calcular la solución óptima.

El algoritmo comprende dos fases:

- Asigna ejemplos a un conjunto inicial de k clusters.
- Después actualiza las asignaciones ajustando los límites de los grupos de acuerdo con los ejemplos de cada clúster.

- Este proceso de asignación y actualización ocurre varias veces hasta que los cambios no proporcionan mejoras en los clusters.

Debido a la naturaleza heurística del algoritmo los resultados pueden ser distintos en función de la inicialización del algoritmo. Sin embargo, si los resultados difieren mucho los unos de los otros puede indicar un problema. Por ejemplo, puede ocurrir que los datos no se puedan agrupar bien en k clusters.

El algoritmo k-medias considera que los valores de las variables son coordenadas en un espacio multidimensional.

Ejemplo de ejecución

El algoritmo empieza eligiendo los k puntos iniciales.

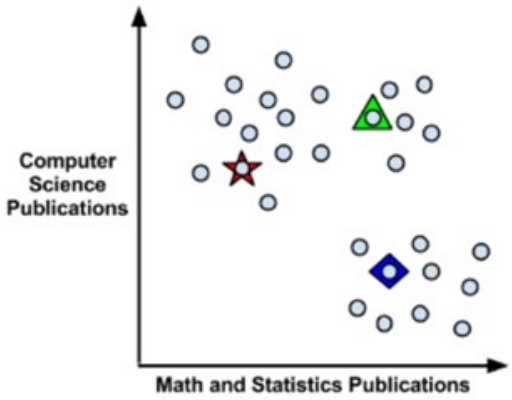


Figura 5. Elección inicial de tres puntos como centroides.

Los puntos iniciales se suelen elegir al azar, en este caso se eligen tres ejemplos al azar. Otras opciones son elegir puntos que pueden ocurrir en cualquier intervalo del espacio de las variables de entrada. Otra opción es asignar inicialmente de forma aleatoria cada punto a un clúster.

Después de elegir los puntos iniciales, los otros ejemplos se asignan al centroide del clúster más cercano de acuerdo a la función de distancia, esta función suele ser la

distancia euclidiana.

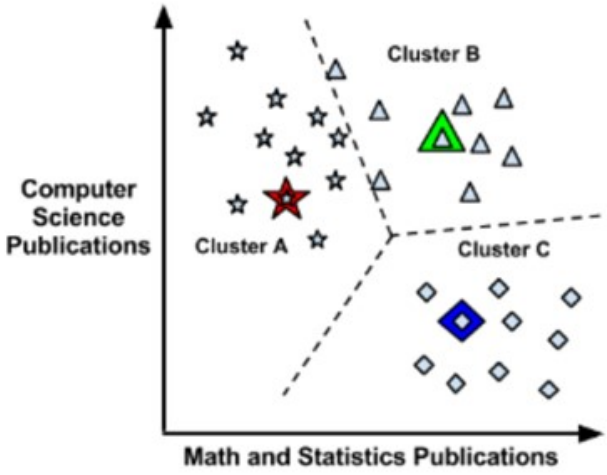


Figura 6. Asignación del resto de los puntos a los centroides.

Hay que tener en cuenta que, debido a que estamos usando distancias, todos los datos deben de ser numéricos y normalizados antes de utilizarlos. El primer paso de la actualización conlleva a reubicar los centroides iniciales a una nueva posición calculada como la media de los puntos asignados al clúster. Como los límites de los centroides se han modificado es muy posible que haya que reasignar instancias a otros clusters.

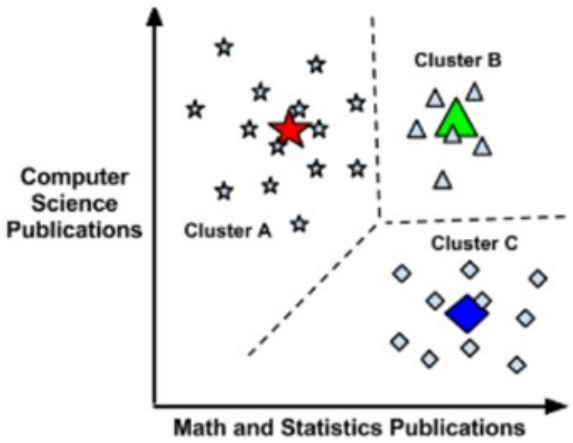


Figura 7. Reajuste de los centroides en función de la asignación del resto de puntos.

Elegir el valor de K

Elegir un buen valor de K requiere de cierto balance. Un número muy grande mejora la homogeneidad, pero a la vez sobre-ajusta los datos.

Idealmente existe un conocimiento a priori sobre el número de grupos apropiado. Por ejemplo, si estamos agrupando películas se puede elegir el valor de k que concuerde con los géneros existentes. Otras veces el número de clusters viene dado por los requisitos de negocio.

Sin conocimiento a priori se suele elegir un valor de $k = \sqrt{(n/2)}$ También se puede usar métricas para medir la homogeneidad versus la heterogeneidad.

A continuación mostramos un ejemplo de cómo se usa KMeans en Python usando Scikit-Learn. Aunque en scipy.cluster.vq también hay una implementación de este.

```python
#Clustering con K-Means
from numpy import unique
from numpy import where
from sklearn.cluster import KMeans
from matplotlib import pyplot
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
# Cargamos el Iris Dataset. Tres clases, 150 casos
# 50 casos por clase, todos los atributos numéricos
idf= datasets.load_iris()
sc = StandardScaler()
X = sc.fit_transform(idf.data)
# Definimos el modelo, le pasamos como parámetro
# el número de clústers que queremos
model = KMeans(3)
# Hacemos el clustering y retornamos el clúster
# asignado a cada caso
yhat = model.fit_predict(X)
# Recuperamos los lusters
clusters = unique(yhat)
# Vamos a mostrar un Scatterplot para cada clúster
for cluster in clusters:
    print("Cluster:{0} cantidad de casos:{1}".format(cluster,len(yhat[yhat==cluster])))
    row_ix = where(yhat == cluster)
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# show the plot
pyplot.show()
```

Figura 8. Ejemplo de uso de K-Means en Python, con Scikit-Learn.

El resultado de este programa se da en la siguiente imagen.

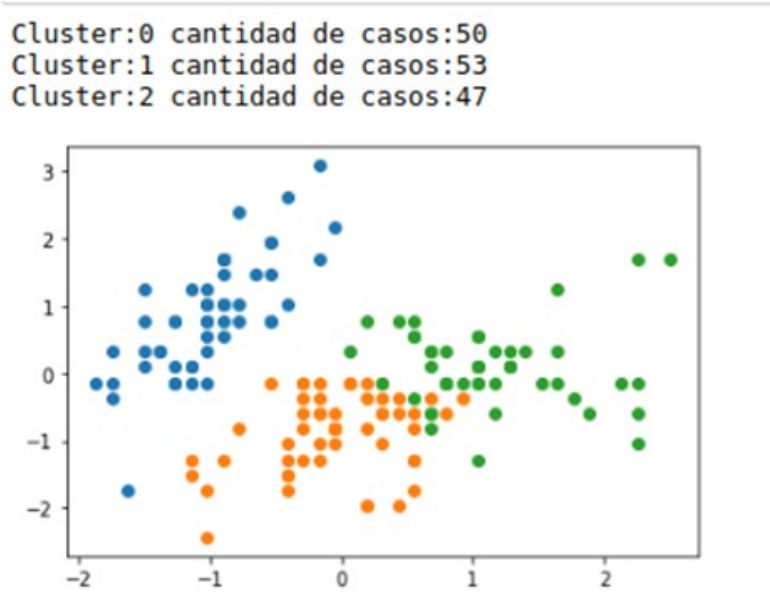


Figura 9. Resultados de la corrida de K-Means sobre Iris DataSet.

Como puede verse, este método se acerca bastante bien a la realidad, aunque hay que pasarle el número de clúster. Esta es una de las desventajas de usar este tipo

de método, pues la mayoría de las veces no conocemos esa cantidad.

También en este mismo módulo se implementa una variante de KMeans llamada MiniBatchKMeans. Revisar la documentación para los detalles.

Affinity Propagation

En este algoritmo, el conjunto de datos puede verse como una red, donde los casos se mandan mensajes entre unos y otros. La idea detrás de esta comunicación es tratar de encontrar aquellos casos que tienen las características adecuadas para representar al resto de los casos de un posible clúster como ejemplar. Los ejemplares, son casos que pueden representar mejor a otros puntos y son más significativos para el grupo. Esta ejemplaridad se evalúa por medio de dos criterios, la responsabilidad y la disponibilidad *(availability)*.

Estos criterios se obtienen por medio de los mensajes que se pasan entre todos los casos y el resultado se almacenados en dos matrices:

- La matriz de responsabilidad R: En esta matriz se almacena un valor $r(i,k)$ que dice cuan capacitado está el caso k para representar, como ejemplar, al caso i.
- La matriz de «availability» A: Este criterio, $a(i,k)$, explica cuan apropiado para i, sería, elegir a k como ejemplar.

Estos criterios se calculan a partir de una medida de similaridad. En este caso los desarrolladores del algoritmo decidieron usar una variante de la distancia euclidiana.

Específicamente la distancia euclidiana negativa al cuadrado.

$$s(i,k) = -\|x_i - x_k\|^2$$

Entonces la responsabilidad se puede calcular como:

$$r(i,k) = s(i,k) - max\{a\left(i,k^{'}\right) + s(i,k^{'})\}$$

Mientras más alto es el valor de r, mejor preparado está k para representar a i como ejemplar.

La disponibilidad o «availability» se calcula de la siguiente forma:

$$a(i,k) \leftarrow min\left(0, r(k,k) + \sum_{i' \notin \{i,k\}} max\left(0, r\left(i', k\right)\right)\right) \forall i \neq k$$

$$a(k,k) \leftarrow \sum_{i' \neq k} max\left(0, r\left(i', k\right)\right)$$

De esta forma, un caso k, es electo como ejemplar de un clúster basado en: que es suficientemente similar a muchos casos y es elegido por muchos casos como ejemplar.

Una de las ventajas de usar este método es que no hay que especificar de antemano la cantidad de clúster en el dataset, sin embargo es necesario especificar la preferencia, por medio del parámetro “preference” y este está ligado a la cantidad de ejemplares que se escogen y por tanto a la cantidad de grupos que retorna en método.

Veamos un ejemplo de cómo se usa en Python y los resultados que se obtiene sobre el dataset iris.

En la figura 11 se puede observar el resultado de la aplicación de Affinity Propagation al Iris Dataset.

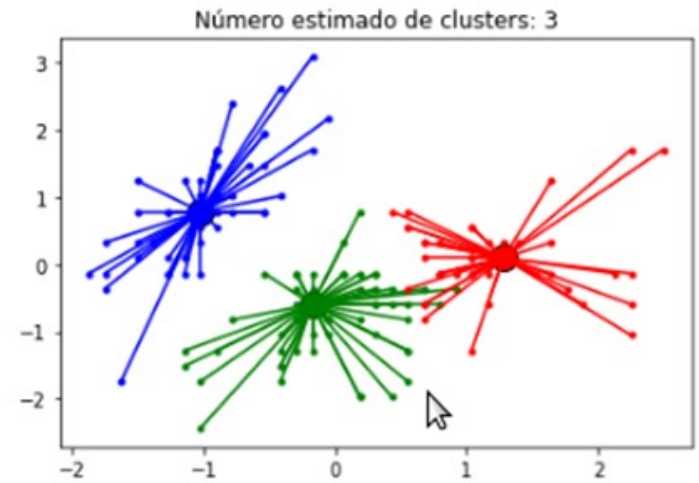
En la figura 11, se puede ver claramente cuantos casos se han asignado a cada uno de los clústers. Si recordamos que el iris dataset tiene 150 casos, tres clases de flores y 50 casos en cada clase, entonces mirando la cantidad de casos que asignó a cada una de las clúster, es un buen síntoma. Sin embargo, no veremos las métricas de evaluación de estos métodos hasta la próxima lección.

Es momento de que veamos los parámetros o argumentos que toma el método a la

hora de construirlo.

```python
#Clustering con Propagación por afinidad(AfinitiPropagation)
from sklearn.cluster import AffinityPropagation
import matplotlib.pyplot as plt
from sklearn import datasets
from itertools import cycle
from sklearn.preprocessing import StandardScaler
# Cargamos el Iris Dataset. Tres clases, 150 casos
# 50 casos por clase, todos los atributos numéricos
idf= datasets.load_iris()
sc = StandardScaler()
X = sc.fit_transform(idf.data)
# Definimos el modelo, le pasamos como parámetro
# preference y damping
model = AffinityPropagation(preference=-50,damping=0.9,random_state=0)
af = model.fit(X)
cluster_centers_indices = af.cluster_centers_indices_
labels = af.labels_
n_clusters_ = len(cluster_centers_indices)
plt.close("all")
plt.figure(1)
plt.clf()
colors = cycle("bgrcmykbgrcmykbgrcmykbgrcmyk")
for k, col in zip(range(n_clusters_), colors):
    class_members = labels == k
    cluster_center = X[cluster_centers_indices[k]]
    plt.plot(X[class_members, 0], X[class_members, 1], col + ".")
    plt.plot(
        cluster_center[0],
        cluster_center[1],
        "o",
        markerfacecolor=col,
        markeredgecolor="k",
        markersize=14,
    )
    for x in X[class_members]:
        plt.plot([cluster_center[0], x[0]], [cluster_center[1], x[1]], col)
plt.title("Número estimado de clusters: %d" % n_clusters_)
plt.show()
```

Figura 10. Ejemplo de uso de clustering con Affinity Propagation en Python.




```
Número de Cluster:0  Número de casos:50
Número de Cluster:1  Número de casos:56
Número de Cluster:2  Número de casos:44
```

Figura 11. Resultados del uso de clustering con Affinity Propagation en Python usando Iris Dataset.

Parámetros:

- damping: Este parámetro es de tipo float y su valor debe estar entre 0.5 y 1. Se denomina factor de damping, su función es evitar las oscilaciones numéricas y su valor por defecto es 0.5.
- max_iter: Especifica el número máximo de iteraciones, espera valores enteros y el valor por defecto es 200.
- preference: Puede tomar forma de arreglo de n_samples. Cuando se pasa como preferencia un arreglo de valores, estos son interpretados como los valores de preferencia para cada uno de los casos. Entre más grande es el valor para un individuo, más seleccionable como ejemplar es. También puede ser un float y el valor por defecto es None. Cuando se pasa un único valor es equivalente a pasar el mismo valor de preferencia a todos los casos. Mientras más alto es este valor, más clúster genera este algoritmo.

Yo propongo que para ajustar este valor se use validación cruzada. Sin embargo

cuando el dataset no es muy grande se pueden probar algunas combinaciones, aunque su valor depende del conjunto de datos.

Affinity: Este parámetro define el tipo de medida de similaridad a usar, los valores que se pueden usar son: “euclidean” o “precomputed”. El valor por defecto es “euclidean” y especifica que se usará la distancia euclidiana negativa cuadrada.

Estos son los parámetros más significativos, veremos ahora que retorna este algoritmo luego de ser ajustado a los datos.

- cluster_centers_indices_: Luego de usar el método fit(), esta variable contiene un arreglo de tamaño igual al número de clústers que contiene los valores correspondientes a los índices de los centros de cada clúster.
- cluster_centers_: Contiene el valor de los centros de los clústers, si la distancia usada no es “precomputed”.
- labels_: Retorna un ndarray con las etiquetas asignadas a cada caso.

Para conocer detalles adicionales de la implementación de este método, es necesario consultar la documentación de scikit-learn.

Mean Shift

Este algoritmo puede considerarse dentro de la familia de los algoritmos basados en centroide. También se en la literatura se le clasifica dentro de los métodos de los algoritmos de búsqueda de la moda de un conjunto de datos.

Este método, no hace asunción alguna sobre la naturaleza de la distribución estadística de los datos y por tanto también puede clasificarse como no paramétrico. Tampoco pone restricciones sobre la forma de los clúster.

Otra de las ventajas de este es que no es necesario pasarle de antemano la cantidad de clústers en que se pretende agrupar nuestros datos.