Clustering con Incertidumbre

Víctor Muñoz Ramírez
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
vicmunram@us.es

Enrique Reina Gutiérrez
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
enrreigut@us.es

I. Investigación Kike

Según la Referencia 1:

Empieza con una pequeña introducción respecto a lo que que es Clustering en Machine Learning. Una técninca para poder separar un conjunto de puntos dados. Estos puntos deben de poseer algún tipo de caracteristica común que nos permita poder indentificar unos de otros y así poder agruparlos.

Habla de 5 técnicas de algoritmos de Clustering: K-Means, Mean-Shift, Density-Bases Spatial con Ruido, Expectation-Maximization (EM) usando mezcla de modelos Gaussianos (GMM) y Agglomerative Hierarchical Clustering. Cada uno presenta sus ventajas y desventajas, que ire desarollando a lo largo de los siguientes puntos:

K-Means:

Este es el algoritmo que es presentado en las transparecias de teoría. Según el artículo es uno de los métodos mas fácil de implementar y de entender. Lo primordialment destacable es que el input inicial es tuyo, es decir, debes primero observar los datos y tratar de indentificar el posible número o cantidad de grupos(clústeres) que puedes identificar según las densidades de los puntos.

Este algorimos consiste, en como ya se menciono anterioirmente, seleccionar tantos puntos considerados como centros de los clusters que considermaos que hay según lla representación dimensional de los datos. El algoritmo es iterativo, es decir, con cada iteración los centros se van desplazando de forma que se sitúen en el centro de cada clúster. Esto se consigue midiendo la distancia de los puntos con cada centro para ver a cual corresponden.

La ventaja de este método es que un algoritmo que es considerado bastante rápido con un complejidad de $\mathcal{O}(n)$. Sin embargo partes con la premisa de que debes seleccionar cuantos grupos de clúster quieres. Esta decisión no siempre es trivial. Una alternativa es realziar aproximaciones como en función de la cantidad de puntos, introducir x centros pero esto puede provocar resultados difrerentes, por ende resutlando en una falta de consistencia.

Otra posible optimización de este algoritmo es ahorrarnos el tener que situar los puntos manualmente y simplemente indicar cuantos queremos. Esto suele realizarse mediante la inicialización aleatoria de estos puntos, no obstante, volvemos a encontrarnos con el problemas de la falta de consistencia.

La Referencia ofrece una alternativa a K-Means conocida como K-Medians. La principal diferencia es que a la horta de recomputar los centros nos basamos en la mediana y no en la media. Este método es menos sensitivo a valroes atípicos sin emaego es más cosotoso de forma computacional ya que requiere *sorting* para computar la mediana del vector de los datos.

Mean-Shift:

Este algoritmo es un algoritmo basado en enventanado que trata de encontrar las areas mas densas de puntos con el fin de determinar el centro de los clusters de puntos dados. Este algoritmo es denominado en inglés como *centroid-base algorithym*. Esto nos describe la función de este algoritmo la cual es encontrar el centro de los clusters formado por el conjunto de puntos dados en funcion de la densidad de estos.

El funcionamiento de este algoritmo es sencillo. Lo primero que se hacer es inicializar un punto de forma aleatoria en el espacio de puntos con un radio de busqueda. La superficie cubierta por el círculo (ventana circular la cual se le conoce como *kernel*) se encarga de contar los puntos incluidos en este. El vector dirección en el que se mueve se hace en función del centro de masa, es decir, donde se vayan concentrando los puntos.

La forma en la que este algoritmo converge, es que en vez de inicializar una única ventana, se inicializan varias distribuidas de forma uniforme por el espacio de puntos. Vamos iterando hasta que las distancia entre el centro de las ventanas sean cubiertas por todas.

La ventanja de este algoritmo respecto a K-Means es que no requerimos de indicar el número de centros que vamos a querer, si no que se detectan solos basados en la desidad de puntos, sin embargo esta ventaja que presenta puede llegar a ser una desventaja. Este método es perfecto

para encontrar el centro de los puntos que formen parte de clusters que esten separados y sean densos. En el escenario de que los cluster conformen figuras, tipo el perímetro de un círculo, el algoritmo puede que no agrupe los puntos en único conjunto, sino que haga un subdivisión de estos. Otra desventaja que presenta es que se deben escoger la longitud del radio resultando factor determinista que puede provocar diferentes resultados.

Density-Based Spatial Clustering con ruidos (DBSCAN):

Este algoritmo también esta basado en la agrupación de los puntos en función de su densidad. La gran diferencia que presenta respecto a Mean-Shift es que no trata de localizar el centro del conjunto de los puntos, si no que va generando a agrupación al vuelo.

El funcionamiento del algoritmo es muy simple. Sen encarga de ir recorriendo todos los puntos del conjunto de puntos y mira en un radio alrededor de este. Si se encuentra un punto dentro de este radio, se le asigna como parte del clúster. Los puntos que se leen, se marcan como visitados. En el momento que ha acabdo de ”agrupar un conjunto de puntos” pasa al siguiente en la lista de puntos .

Este algoritmo presenta ventajas respecto a K-Means ya que como en Mean-Shift no es necesario indicar de antemano el número de agrupaciones que identificamos o vamos a querer, además de que a difrencia de Mean-Shift es capaz dde identificar figuras. Sin embargo, también tiene su desventaja la cual es que requiere cierta distancia mínima entre los puntos para poder agrupar todo en la misma figura, ya que la distancia a la que miramos, conocida como Epsilon, varía en función de la densidad de puntos en esa zona.

El artículo habla de otros dos algoritmos los cuales no consideranmos relevantes para el objetivo que se nos propone pero consideramos que son muy interesantes y presentan alternativas a la hora de agrupar diferentes grupos de puntos.

De momento las técnicas de *clustering* presentadas, como hemos redactado, tienen sus ventajas y desventajas en las cuales nos basaremos para tratar el tema de nuestro objetivo. El principal porblema encontrado es que en función de la distribución de puntos, conviene más una técnica que otra de forma que la dificultad se encuentra en automatizar la clasificación de los puntos de forma que siempre se use la técnica más efectiva para agrupar puntos.

Contando con la desventaja presentada anteriormente, igualemente debemos de elaborar un algoritmo que ataje el problema propuesto. Tras la investigación realziada hasta el momento, creemos que DBSCAN es la opción más factible, sin embargo tenemos en mente el problema de que requiere cierta densidad de puntos que conformen la circunferencia. Por ello, la siguiente idea propuesta es realizar una mezcla entre K-Means y Mean-Shift. El objetivo es inicializar una cierta cantidad de centro de forma que estos cubran todo el espacio de puntos y luego vamos aproximando los centros por iteración como en K-Means. La condición de parada sería cuano estos centros convergan.

Creemos que K-Means es la segunda mejor opción ya que el algoritmo como su nombre indica, no esta basado en la debsidad de puntos, sino en la media de la posición de esto. Esto nos interesa ya que las circufenrencias generalmente lo puntos estarán distribuidos de forma que en el centro de esta no exista ningún punto salvo que se colapse con otra circunferencia. Estos caso específicos serán estudiados con mayor precisión en el estudio del problema.

II. Estructura del proyecto

Generación y representación de ejemplos:

En primer lugar, se decidió implementar las clases y métodos que fueran necesarios para la generación de ejemplos y representación gráfica de estos mismos. El objetivo era poder comprobar, durante el desarrollo de las distintas partes del algoritmo, la eficacia de lo implementado para una buena variedad de casos de prueba y así ver los fallos que podrían haberse cometido y subsanarlos lo antes posible.
De este modo, se implementaron las clases Point y Circunference, que representan los puntos y circunferencias del problema; y las clases DataSet y Canvas, que nos permiten, respectivamente, realizar la carga de datos de un ejemplo a partir de un archivo y elaborar estos de forma aleatoria añadiendo ruido o eliminando puntos.

A continuación, se realizó la implementación de los objetivos específicos relativos al código siguiendo el orden en el que aparecían en el documento; ya que iban de menos a más en la elaboración del algoritmo final.

De esta manera, se creó una estructura de datos formada por dos arrays:

- Threshold: que almacena arrays con los grados de pertenencia de cada punto a cada cluster. El cual se recalcula al inicio de cada iteración.
- Clusters: que almacena objetos de la clase Circle, los cuales representan los clusters. Este se recalcula después que el anterior ya que utiliza esos datos para saber que puntos usar para recalcular cada uno.

Cálculo de circunferencias:

Tras esto, se elaboró un método para calcular centro y radio de una circunferencia a partir de un listado de puntos.
La primera versión de este método calculaba el centro como el baricentro del conjunto de puntos y el radio como la distancia media de este a todos los puntos del conjunto. Este método, aunque muy eficiente incluso con ruido, fallaba cuando se le proporcionaba un arco en lugar de una circunferencia; ya que como el baricentro se sitúa en función de la densidad, el centro queda muy cerca del arco resultando en un radio menor al debido, y quedando los puntos de los extremos muy alejados de la circunferencia.
Por ello, se decidió implementar un segundo método que calculara el centro como el circuncentro del conjunto, usando para ello tres puntos de este. La selección de estos tres puntos inicialmente consistía en tomar un punto aleatorio, buscar el más lejano a este y por úlitmo buscar uno a una distancia intermedia de ambos. Esta forma requería cierto coste computacional adicional ya que había que recorrer el listado de puntos más de una vez, pero reaccionaba bien incluso con arcos. Sin embargo, en situaciones con mucho ruido el resultado se desviaba.
Finalmente, se dejaron ambos métodos para poder comprobar con el algoritmo final, como reaccionaba cada uno y cual era más eficiente.

Cálculo de grados de pertenencia:

Después, se implementó el método correspondiente al cálculo de los grados de pertenencia.
Como primer paso se calculan todas las distancias del punto a los clusters, tras lo cual se realiza un sumatorio de todas ellas. A este se le restan las distancias por separado, para que los grados de pertenencia sean inversamente proporcionales a la distancia. Y para que dichos valores esten normalizados, se dividen los valores de la operación anterior entre el sumatorio.
Cabe destacar que durante la experimentación con el algoritmo final, en ocasiones al intentar normalizar se producían divisiones por cero; lo cual fue solventado instanciando el sumatorio de los puntos con una cantidad despreciable.
Posteriormente, se diseñó una modificación de este método que calcula los grados de pertenencia para todos los puntos de un problema, devolviendo el array correspondiente a ThresholdValues, ya mencionado antes. Además de esta información, este método devuelve otro array formado por tuplas compuestas por un punto y un índice que corresponde al cluster con el que el punto tiene un mayor grado de pertenencia.

Implementación del esquema general:

Finalmente, se implementó la clase ClusteringSolver en la cual encontramos el método learn, el cual sigue el esquema general planteado.
Para la inicialización, desde un inicio se decidió instanciar las circunferencias de forma aleatoria, concretamente, se toma como centro un punto aleatorio del listado y se le proporciona un radio también aleatorio.
En cuanto al criterio de parada, primero se implementó con un número fijo de iteraciones ya que era más sencillo y permitía comprobar el comportamiento del algoritmo con facilidad; aunque finalmente se acabó implementando un criterio de parada basado en que los centros y radios de los clusters no variaran en una cantidad determinada por el usuario. Sin embargo, es necesario aportar un número máximo de iteraciones pues pueden producirse casos en los que en la búsqueda por alcanzar la precisión deseada genere que el centro alterne entre dos puntos de forma indefinida.
Se decidió permitir al usuario elegir si utilizar baricentro o circuncentro para recalcular las circunferencias. Aunque es cierto, que si el conjunto de puntos suministrado para un cluster es menor de tres, se utiliza el baricentro ya que el circuncentro necesita como minimo esa cantidad.
Por último, para la presentación final de los datos, se creó la clase Solution; la cual utiliza la información contenida en un

objeto ClusteringSolver que ya ha realizado el método learn. Nos permite eliminar puntos cuyo grado de pertenencia final sea menor al elegido por el usuario, además de representar los datos de forma gráfica mostrando las circunferencias junto con los puntos coloreados en función del cluster al que pertenecen. Actualmente, esta representación tiene como límite siete clusters, ya que a partir de octavo los colores se repetirán y no se podrá saber con certeza si la asignación es correcta. Aún así, es facilmente escacalable.
Una vez implementado todo, realizando la experimentación, se observó que la inicialización de los clusters al ser aleatoria condicionaba mucho el resultado final del algoritmo. De modo, que se decidió investigar sobre ello y se encontró que una técnica bastante utilizada era el lanzamiento del algoritmo con múltiples instancias haciendo uso de una heurística para obtener el mejor resultado de todas ellas. En nuestro caso, la heurística es que se considera un mejor resultado aquel en el que el sumatorio de las distancias de los puntos a los clusters es menor.

Mejoras:

Como mejora, se implementó una interfaz gráfica dentro de Jupyter Notebook, que nos permite crear ejemplos aleatorios, eliminar de estos puntos, insertar ruido y guardarlo como fichero Excel en la ruta deseada; así como la resolución de estos indicando el número de instancias, clusters, precisión y máximo de iteraciones.
Además, se aporta otra interfaz gráfica en la que se carga un archivo Excel, se muestra y se resuelve con los parámetros antes mencionados.
Los archivos Excel mencionados están formados por una primera fila con las cabeceras ”Coord X, Coord Y, Circulo”, en la que ”Circulo” es opcional; y por cada punto del ejemplo una nueva fila con las coordenadas x e y del punto así como el círculo al que pertenece separado por comas.

REFERENCIAS

[1] Seif, G., 2020. The 5 Clustering Algorithms Data Scientists Need To Know. [online] Medium. Available at: https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68 [Accessed 18 April 2020].