DBSCAN: Density-based spatial clustering of applications with noise.

I302 - Aprendizaje Automático y Aprendizaje Profundo

DBSCAN es un algoritmo de clustering que requiere dos parámetros clave: ϵ (el radio de la vecindad) y $minPts$ o K (el número mínimo de puntos necesarios para que una región se considere densa). El algoritmo funciona de la siguiente manera:

1. **Seleccionar un punto arbitrario:** El algoritmo comienza seleccionando un punto arbitrario p en el conjunto de datos.

2. **Examinar la ϵ-vecindad de p:** Se visita la vecindad de radio ϵ alrededor del punto p. Si esta vecindad contiene al menos $minPts$ puntos (incluyendo al propio punto p), se considera que p está en una región densa y se inicia un nuevo clúster C. Si no hay suficientes puntos en la vecindad, p se etiqueta como ruido. Es importante notar que un punto etiquetado inicialmente como ruido puede más tarde ser parte de un clúster si se encuentra en la vecindad de otro punto denso.

3. **Expandir el clúster:** Si p es parte de un clúster, todos los puntos en su ϵ-vecindad también se consideran parte del clúster. A continuación, se examina la vecindad de cada uno de estos puntos. Si alguno de estos puntos tiene al menos $minPts$ puntos en su ϵ-vecindad, estos puntos adicionales se añaden al clúster y su vecindad también se explora. Este proceso se repite de manera iterativa, expandiendo el clúster hasta que no se puedan añadir más puntos.

4. **Repetir el proceso:** Una vez que el clúster actual se ha expandido por completo, se selecciona un nuevo punto no visitado y se repite el proceso para descubrir nuevos clústeres o identificar más puntos de ruido.

Se recomienda consultar el paper de DBSCAN para una explicación más detallada del algoritmo y sus propiedades [1]. El pseudocódigo del algoritmo DBSCAN se muestra en el **Algoritmo 1**.

Algoritmo 1 DBSCAN

Input: Dataset D, distance function $dist$, minimum points $minPts$, radius ϵ
Output: Clusters C
$C \leftarrow \{\}$ ▷ Initialize an empty set of clusters
$visited \leftarrow \{\}$ ▷ Initialize an empty set of visited points
for each point $p \in D$ **do**
    if $p \in visited$ **then**
        continue ▷ Skip p if it has already been visited
    end if
    $visited \leftarrow visited \cup \{p\}$ ▷ Mark p as visited
    $neighbors \leftarrow$ REGIONQUERY(p, ϵ)
    if $|neighbors| < minPts$ **then**
        mark p **as noise**
    else
        $C \leftarrow C \cup \{$EXPANDCLUSTER(p, $neighbors$, C)$\}$
    end if
end for

function EXPANDCLUSTER(p, $neighbors$, C)
    $cluster \leftarrow \{p\}$ ▷ Initialize the new cluster with p
    for each point $n \in neighbors$ **do**
        if $n \notin visited$ **then**
            $visited \leftarrow visited \cup \{n\}$ ▷ Mark p as visited
            $newNeighbors \leftarrow$ REGIONQUERY(n, ϵ)
            if $|newNeighbors| \geq minPts$ **then**
                $neighbors \leftarrow neighbors \cup newNeighbors$ ▷ Add neighbors
            end if
        end if
        if $n \notin anyCluster$ **then** ▷ If neighbor is not part of any cluster
            $cluster \leftarrow cluster \cup \{n\}$ ▷ Add it to the current cluster
        end if
    end for
    return $cluster$
end function

function REGIONQUERY(p, ϵ)
    return $\{q \in D \mid dist(p, q) \leq \epsilon\}$
end function

Referencias

[1] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 1996, pp. 226-231.