TIC

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

TIC

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) es un algoritmo de agrupamiento que se basa en la densidad de los puntos en el espacio de características. Su funcionamiento se centra en la identificación de regiones de alta densidad, que se consideran como clusters, separados por regiones de baja densidad. Para ello, utiliza dos parámetros clave: epsilon (ε), que especifica la distancia máxima entre dos puntos para que se consideren vecinos, y minPts, que establece el número mínimo de puntos dentro de un vecindario para que un punto se considere central.

DBSCAN clasifica los puntos como centrales, de frontera o de ruido, lo que permite identificar clusters de diversas formas y tamaños, así como puntos atípicos. Su aplicación abarca desde la segmentación de datos espaciales hasta la detección de anomalías en conjuntos de datos de alta dimensionalidad.

DBSCAN es un algoritmo de agrupamiento que se basa en la densidad de los puntos en el espacio de características. Su funcionamiento se puede resumir en los siguientes pasos:

1

Principio de funcionamiento: DBSCAN divide el conjunto de datos en regiones de alta densidad separadas por regiones de baja densidad. Para ello, utiliza dos parámetros principales:

TIC

Epsilon (ε): Especifica la distancia máxima entre dos puntos para que se consideren vecinos.

MinPts: Establece el número mínimo de puntos dentro de un vecindario para que un punto se considere central.

2

Definición de parámetros: Epsilon y MinPts son dos parámetros fundamentales en DBSCAN. Epsilon determina el tamaño del vecindario alrededor de cada punto, mientras que MinPts define la densidad mínima requerida para formar un cluster.

3

Identificación de puntos: DBSCAN clasifica los puntos en tres categorías:

Puntos centrales: Son aquellos que tienen al menos MinPts puntos dentro de su vecindario de distancia ε.

Puntos de frontera: No son puntos centrales, pero están dentro del vecindario de un punto central.

Puntos de ruido: No son centrales ni de frontera.

4

Ejemplos de aplicación: DBSCAN se utiliza en diversas aplicaciones, como la detección de anomalías y el clustering de datos. Por ejemplo, en la detección de anomalías, los puntos de ruido pueden considerarse como anomalías, mientras que en el clustering de datos, DBSCAN puede identificar clusters de diferentes formas y tamaños de manera automática.

TIC

Implementación de DBSCAN

Escribe una función en Python que implemente el algoritmo DBSCAN. La función debe aceptar los siguientes parámetros de entrada:

data: La matriz de datos para agrupar.

epsilon: El valor de epsilon para definir el radio del vecindario.

inPts: El número mínimo de puntos dentro de un vecindario para que un punto se considere central.

La función debe devolver una lista de etiquetas de clúster para cada punto en los datos.

```
from sklearn.cluster import DBSCAN

def custom_dbscan(data, epsilon, minPts):
    dbscan = DBSCAN(eps=epsilon, min_samples=minPts)
    labels = dbscan.fit_predict(data)
    return labels
```

Identificación de Puntos

Dado un conjunto de datos y los resultados del agrupamiento utilizando DBSCAN, escribe una función para identificar y contar el número de puntos centrales, puntos de frontera y puntos de ruido.

```
def identify_points(labels):
    unique_labels = set(labels)
    core_points = sum(1 for label in labels if label != -1)
    border_points = sum(1 for label in labels if label == -1)
    noise_points = sum(1 for label in labels if label == 0)
    return core_points, border_points, noise_points
```

```
# Ejemplo de uso
labels = [0, 1, -1, 1, 2, 2, 0, -1]
core, border, noise = identify_points(labels)
print("Puntos centrales:", core)
print("Puntos de frontera:", border)
print("Puntos de ruido:", noise)
```

Ejemplo de Aplicación

Utiliza DBSCAN para agrupar un conjunto de datos sintéticos y visualiza los resultados utilizando un gráfico de dispersión.

```
import numpy as np
import matplotlib.pyplot as plt

# Generar datos sintéticos
np.random.seed(0)
X, _ = make_moons(n_samples=200, noise=0.1)

# Aplicar DBSCAN
epsilon = 0.2
minPts = 5
labels = custom_dbscan(X, epsilon, minPts)

# Visualizar los resultados
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.title('Agrupamiento con DBSCAN')
plt.xlabel('Característica 1')
plt.ylabel('Característica 2')
plt.show()
```