TIC

K-Means y agrupamiento jerárquico

TIC

K-Means

K-Means es un algoritmo de agrupamiento que divide un conjunto de datos en k grupos o clusters. El proceso ocurre de la siguiente manera:

1. Comienza con la selección aleatoria de k centroides, que son los puntos representativos de cada clúster.
2. Se asigna cada punto de datos al centroide más cercano, formando así los clústeres.
3. Se recalculan los centroides como el promedio de los puntos dentro de cada clúster.
4. Este proceso se repite iterativamente hasta que los centroides no cambian significativamente o se alcanza un número máximo de iteraciones.

Proceso de agrupamiento utilizando K-Means:

K-Means es un algoritmo de agrupamiento que divide un conjunto de datos en k grupos o clusters. El proceso ocurre de la siguiente manera:

1. **Inicialización de centroides:** Seleccionar k centroides al azar del conjunto de datos.
2. **Asignación de puntos a clústers:** Asignar cada punto de datos al centroide más cercano.
3. **Actualización de centroides:** Recalcular los centroides como el promedio de los puntos en cada clúster.
4. **Iteración:** Repetir los pasos 2 y 3 hasta que los centroides converjan o se alcance el número máximo de iteraciones.

TIC

Consideraciones sobre la inicialización de centroides y selección del número de clústers

Inicialización de centroides: La selección inicial de centroides puede afectar los resultados del agrupamiento. Diferentes métodos de inicialización pueden conducir a resultados diferentes. Algunas técnicas comunes incluyen la inicialización aleatoria, la inicialización basada en heurísticas y la inicialización inteligente utilizando **técnicas como KMeans++.**

Selección del número de clusters (k): Determinar el número óptimo de clusters es crucial para obtener resultados significativos. Se pueden utilizar técnicas como el método del codo (Elbow Method), el criterio de información bayesiano (BIC), o la validación cruzada para seleccionar el valor óptimo de k.

Ejemplos prácticos de aplicación de K-Means en Python

```
from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

import numpy as np

# Generar datos de ejemplo

np.random.seed(0)

X = np.random.rand(100, 2)

# Inicializar y ajustar el modelo K-Means

kmeans = KMeans(n_clusters=3)

kmeans.fit(X)
```

```
# Visualizar los clusters y los centroides

plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_, cmap='viridis')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,
1], marker='x', color='red', s=200)

plt.title("Clustering con K-Means")

plt.xlabel("Feature 1")

plt.ylabel("Feature 2")

plt.show()
```

TIC

Agrupamiento jerárquico

El agrupamiento jerárquico es un enfoque de agrupamiento en el que los datos se organizan en una estructura jerárquica de clusters. Este método no requiere que se especifique el número de clusters de antemano y puede representarse como un árbol o dendrograma. Hay dos enfoques principales en el agrupamiento jerárquico: aglomerativo y divisivo.

1. **Aglomerativo:** Comienza con cada punto de datos como un clúster individual y fusiona iterativamente los clusters más cercanos hasta que todos los puntos estén en un solo cluster.

2. **Divisivo:** Comienza con un único clúster que contiene todos los puntos de datos y divide iterativamente el clúster en clusters más pequeños hasta que cada punto esté en un cluster individual.

Métodos de enlace

Los métodos de enlace se utilizan para calcular la distancia entre dos clústers durante el proceso de agrupamiento. Algunos de los métodos de enlace más comunes incluyen:

Single-linkage (o mínimo): La distancia entre dos clusters se define como la distancia mínima entre cualquier par de puntos, uno de cada cluster.

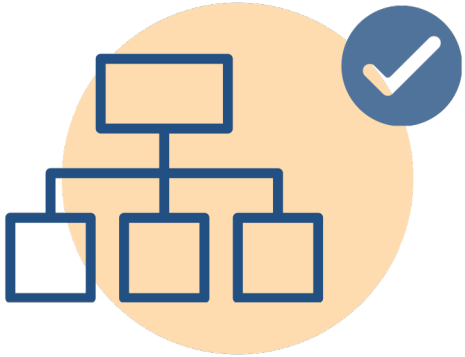
Complete-linkage (o máximo): La distancia entre dos clusters se define como la distancia máxima entre cualquier par de puntos, uno de cada cluster.

Average-linkage: La distancia entre dos clusters se define como el promedio de todas las distancias entre pares de puntos, uno de cada clúster.

TIC

Visualización de dendrogramas y selección del número de clusters

Los dendrogramas son representaciones gráficas de la jerarquía de clusters generados durante el agrupamiento jerárquico. Permiten visualizar la estructura jerárquica y facilitan la selección del número óptimo de clusters al observar la altura a la que se realizan las fusiones en el dendrograma.

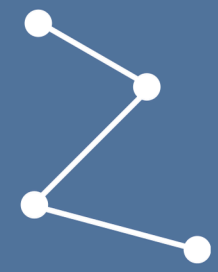


Implementación de agrupamiento jerárquico en Python

En Python, se puede implementar el agrupamiento jerárquico utilizando bibliotecas como scikit-learn o scipy. Estas bibliotecas ofrecen funciones y clases para realizar agrupamiento jerárquico con diferentes métodos de enlace y criterios de distancia. Una vez agrupados los datos, se pueden visualizar los dendrogramas y seleccionar el número óptimo de clusters según los requisitos del problema.

Métodos de enlace

Single-linkage (enlace simple): Calcula la distancia mínima entre cualquier par de puntos pertenecientes a diferentes grupos y considera esta distancia como la distancia entre los grupos. Es sensible a los efectos de cadena o alargamiento, donde grupos pueden fusionarse debido a un solo par de elementos muy cercanos.



Complete-linkage (enlace completo): Calcula la distancia máxima entre cualquier par de puntos pertenecientes a diferentes grupos y considera esta distancia como la distancia entre los grupos. Es menos sensible a los efectos de cadena que el enlace simple.

Average-linkage (enlace promedio): Calcula el promedio de las distancias entre todos los puntos en los dos grupos y lo considera como la distancia entre los grupos. Es más robusto frente a los efectos de cadena que el enlace simple, pero puede ser afectado por la presencia de outliers.

Los métodos de enlace determinan cómo se calcula la distancia entre clusters. El método single-linkage considera la distancia mínima entre los puntos de los clusters. El complete-linkage considera la distancia máxima. El average-linkage considera el promedio de las distancias.

Visualización de dendrogramas

```
# Visualización de dendrogramas

import numpy as np

import matplotlib.pyplot as plt

from scipy.cluster.hierarchy import dendrogram, linkage


# Generar datos aleatorios

np.random.seed(0)

X = np.random.randn(20, 2)
```

```
# Visualizar dendrograma
plt.figure(figsize=(10, 5))
dendrogram(Z)
plt.title('Dendrograma')
plt.xlabel('Índices de la muestra')
plt.ylabel('Distancia')
plt.show()
```

Implementación de agrupamiento jerárquico

```
# Implementación de agrupamiento jerárquico en Python
from sklearn.cluster import AgglomerativeClustering

# Generar datos aleatorios
np.random.seed(0)
X = np.random.randn(20, 2)

# Crear una instancia del modelo de agrupamiento jerárquico
model = AgglomerativeClustering(n_clusters=3, linkage='ward')

# Ajustar el modelo a los datos
clusters = model.fit_predict(X)

# Imprimir resultados
print("Clusters:", clusters)
```