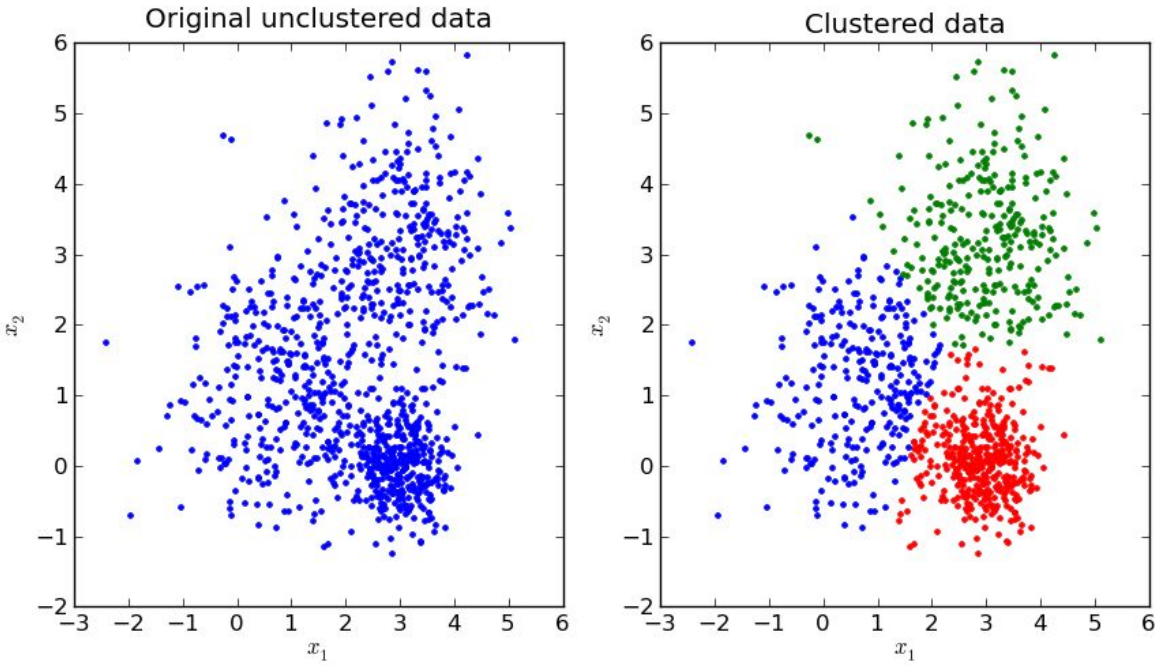
Clustering II

Curso de inteligencia artificial-Intermedio

Objetivo

Estudiar cómo evaluar el rendimiento de clustering por K Means y estudiar un método para optimizar la elección de K.



Evaluación rendimiento aprendizaje no supervisado

Aprendizaje no supervisado

Evaluar el rendimiento en modelos de aprendizaje no supervisado, como el **K-Means**, es más complicado que en el aprendizaje supervisado, ya que no contamos con etiquetas reales para comparar los resultados.

Evaluación rendimiento clustering K Means

Evaluación rendimiento K Means

1. **Suma de Errores Cuadráticos Dentro de los Clústeres (Within-Cluster Sum of Squares - WCSS)**

El WCSS mide la compactación de los clústeres formados. Se calcula como la suma de las distancias al cuadrado entre cada punto y su centroide dentro de cada clúster. El objetivo es minimizar este valor, lo que indica que los puntos dentro de cada clúster están cerca de su centroide.

$$WCSS = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2$$

Donde C_i son los puntos en el clúster i, y μ_i es el centroide del clúster i.

- **Inconveniente**: No nos da información sobre cuán separables son los clústeres entre sí, solo sobre la compactación dentro de los mismos.

Evaluación rendimiento K Means

2. **Coeficiente de Silueta (Silhouette Score)**

El coeficiente de silueta mide tanto la **cohesión** como la **separación** de los clústeres. La métrica evalúa cuán cerca están los puntos de su propio clúster en comparación con los puntos de otros clústeres. Su valor varía entre -1 y 1:

- **1**: El clúster es denso y bien separado.
- **0**: Los clústeres se solapan.
- **-1**: Los puntos están mal clasificados y deberían pertenecer a otro clúster.

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Donde:

- $a(i)$: es la distancia promedio entre el punto i y los puntos de su propio clúster.
- $b(i)$: es la distancia promedio entre el punto i y los puntos del clúster más cercano que no contiene a i.

Evaluación rendimiento K Means

4. Índice de Davies-Bouldin

Este índice mide la relación entre la dispersión dentro de los clústeres y la separación entre ellos. Un valor más bajo indica mejor separación y cohesión de los clústeres. Se calcula como:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right)$$

Donde:

- σ_i es la dispersión del clúster i,
- c_i y c_j son los centroides de los clústeres i y j,
- $d(c_i, c_j)$ es la distancia entre los centroides de los clústeres i y j.

Evaluación rendimiento K Means

5. Índice de Calinski-Harabasz (Variance Ratio Criterion)

Este índice evalúa la proporción entre la dispersión entre los clústeres (separación) y la dispersión dentro de los clústeres (cohesión). Cuanto mayor es el valor, mejor es la formación de clústeres.

$$CH = \frac{Tr(B_k)}{Tr(W_k)} \times \frac{n-k}{k-1}$$

Donde:

- $Tr(B_k)$ es la dispersión entre los clústeres,
- $Tr(W_k)$ es la dispersión dentro de los clústeres,
- n es el número total de puntos,
- k es el número de clústeres.

Estimación de los clústeres

Estimación de los clústeres

El dominio del problema: Es importante explicar que a veces el número de clusters se puede conocer de antemano o se basa en el conocimiento del dominio del problema. En estos casos, las técnicas que veremos no son usadas.

Estimación de los clústeres

1. Método del Codo (Elbow Method)

El método del codo es uno de los más populares y sencillos de explicar. La idea detrás de este método es que mientras más clusters se utilicen, mejor será el ajuste del modelo, es decir, menor será la **inercia** o **suma de los errores al cuadrado (SSE)**. Sin embargo, a medida que se aumenta el número de clusters, se llega a un punto donde la reducción en la SSE se vuelve menos significativa, formando un "codo" en la gráfica.

Estimación de los clústeres

Pasos para aplicar el Método del Codo:

1. Entrena el algoritmo K-means para diferentes valores de *k* (número de clusters), por ejemplo, entre 1 y 10.
2. Para cada *k*, calcula el SSE.
3. Grafica el valor de *k* en el eje *x* y la SSE en el eje *y*.
4. Elige el valor de *k* en el que la curva empieza a aplanarse, es decir, donde se forma el "codo". Este valor indica el número óptimo de clusters.

Estimación de los clústeres

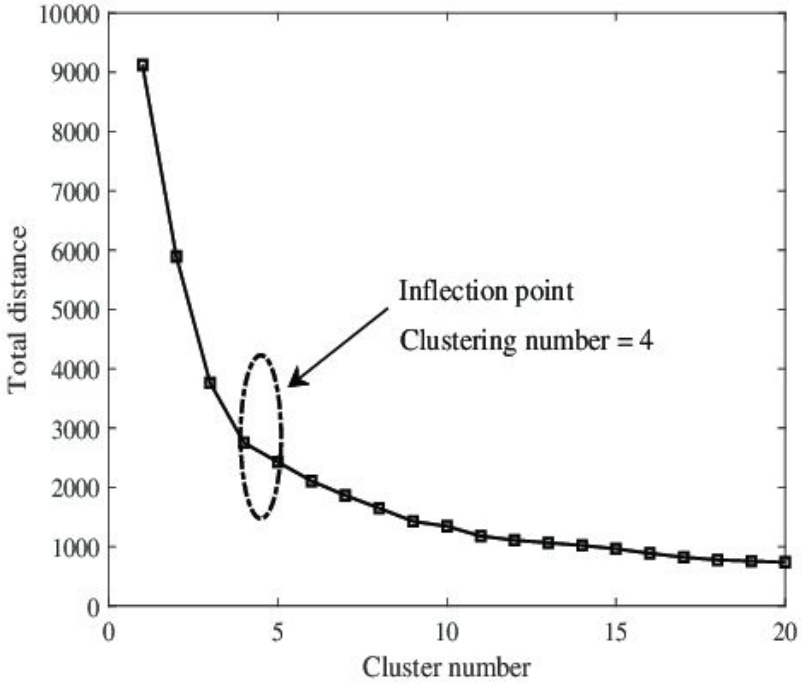
Ventajas: Es sencillo y visual, ideal para explicar a estudiantes que están empezando con Machine Learning.

Estimación de los clústeres

Ventajas: Es sencillo y visual, ideal para explicar a estudiantes que están empezando con Machine Learning.

Desventajas: A veces el "codo" no es claro o no existe, lo que dificulta la identificación del número óptimo de clusters.

Estimación de los clústeres



Estimación de los clústeres

2. Método de la Silueta (Silhouette Method)

El coeficiente de silueta mide qué tan bien está asignado un punto a su cluster y cómo de similar es ese punto a otros clusters. El coeficiente varía entre -1 y 1:

- Un valor cercano a 1 indica que los puntos están bien agrupados.
- Un valor cercano a 0 indica que los puntos están en el borde de dos clusters.
- Un valor negativo indica que los puntos están probablemente en el cluster equivocado

Estimación de los clústeres

Pasos para aplicar el Método de la Silueta:

1. Entrena K-means para diferentes valores de *k*.
2. Calcula el **coeficiente de silueta** para cada punto.
3. Promedia los coeficientes de silueta para cada valor de *k*.
4. Elige el valor de *k* que maximice el coeficiente promedio de silueta.

Estimación de los clústeres

Ventajas: No solo considera la distancia entre puntos dentro del mismo cluster, sino también cómo los clusters están separados entre sí.

Estimación de los clústeres

Ventajas: No solo considera la distancia entre puntos dentro del mismo cluster, sino también cómo los clusters están separados entre sí.

Desventajas: Computacionalmente más costoso que el método del codo, ya que se evalúa cada punto respecto a todos los clusters.

Implementación en Python

Explicación código

Generación de Datos Sintéticos

Antes de implementar los métodos, generamos un conjunto de datos de ejemplo usando `make_blobs`, lo cual nos permite tener un dataset simulado para aplicar los métodos de clustering.

Nuevas librerías

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.datasets import make_blobs
```

Importaciones nuevas:

- **`make_blobs`**: Esta función genera datos distribuidos en varios clusters (en este caso, 4 centros). Los datos generados (`X`) contienen 1000 muestras, con una desviación estándar de 0.60 entre los puntos de cada cluster.

Generación datos sintéticos

```
# Generar datos sintéticos
X, y = make_blobs(n_samples=1000, centers=4, cluster_std=0.60, random_state=0)
# Visualización de los datos generados
plt.scatter(X[:, 0], X[:, 1], s=50, cmap='viridis') plt.title('Datos de ejemplo
para clustering') plt.show()
```

Método del codo

```
# Lista para almacenar la suma de los errores al cuadrado (SSE)
sse = []
# Probar KMeans con k desde 1 hasta 10
k_values = range(1, 11)
for k in k_values: kmeans = KMeans(n_clusters=k, random_state=0) kmeans.fit(X)
sse.append(kmeans.inertia_) # inercia es la SSE
```

Aquí se inicializa el modelo de **K-Means**:

- `kmeans.inertia_` devuelve la inercia, que es el valor de SSE para el modelo entrenado.

Bucle `for`:

- Probamos el algoritmo KMeans para un número de clusters (k) que varía entre 1 y 10.
- Para cada valor de *k*, entrenamos el modelo KMeans (`kmeans.fit(X)`) en los datos y almacenamos el valor de la inercia (SSE) en la lista `sse`.

Método de la silueta

```
# Lista para almacenar las puntuaciones de silueta
silhouette_scores = []
# Probar KMeans con k desde 2 hasta 10 (Nota: la silueta no está definida para
k=1)
for k in range(2, 11): kmeans = KMeans(n_clusters=k, random_state=0) labels =
kmeans.fit_predict(X) silhouette_avg = silhouette_score(X, labels)
silhouette_scores.append(silhouette_avg)
```

Bucle `for`:

- Ejecutamos KMeans para valores de *k* entre 2 y 10, ya que la métrica de silueta no se define para *k* = 1 (no se puede calcular la separación entre clusters si solo hay uno).
- Después de entrenar el modelo KMeans, obtenemos las etiquetas (`labels`) que indican a qué cluster pertenece cada punto.
- Luego calculamos el **coeficiente de silueta** utilizando `silhouette_score(X, labels)` y almacenamos el valor en la lista `silhouette_scores`.