Introducción a Algoritmos No Supervisados

Los algoritmos no supervisados son una categoría de métodos de aprendizaje automático que operan sin etiquetas predefinidas. Identifican patrones y estructuras en los datos de entrada por sí mismos.

4/26/2024

¿Qué son los Algoritmos No Supervisados?

- **Sin supervisor**: No hay respuestas correctas proporcionadas durante el entrenamiento.
- **Identificación autónoma**: Descubren patrones y regularidades en los datos sin intervención.

Tipos de Algoritmos No Supervisados

1. Clustering (Agrupamiento)
2. Reducción de dimensionalidad
3. Asociación

Clustering (Agrupamiento)

Definición

Agrupa objetos basados en su similitud, haciendo que los miembros de un grupo sean más similares entre sí que con los de otros grupos.

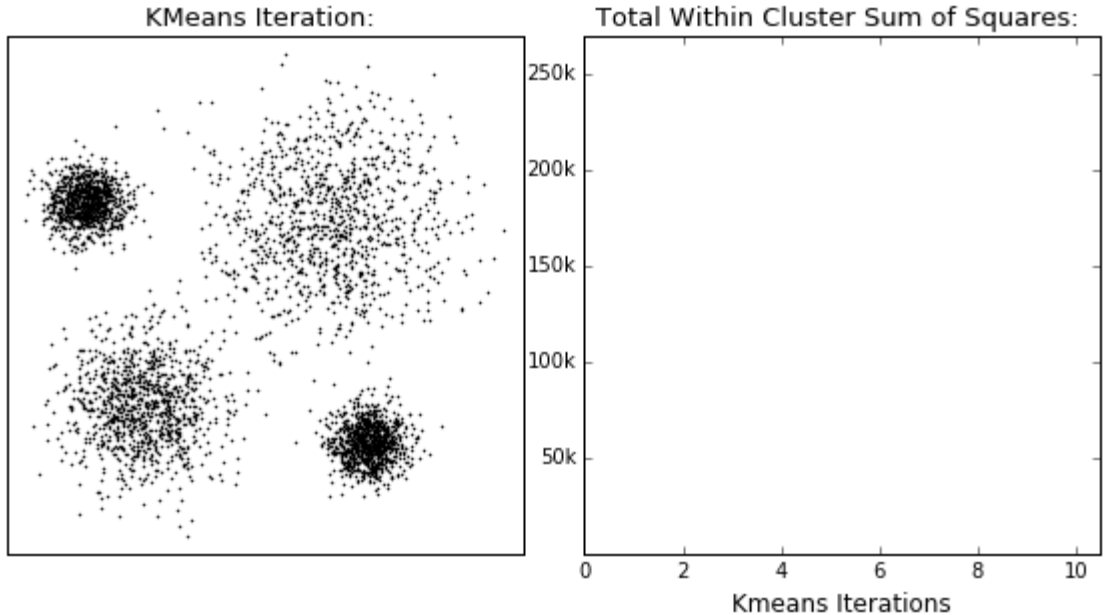
Ejemplos de Clustering

- **K-means**: Particiona los datos en K grupos centrados en centroides.
- **DBSCAN**: Agrupa puntos según densidades, ideal para formas irregulares.
- **Agrupamiento jerárquico**: Visualiza relaciones con un dendrograma.

Introducción a K-means

K-means es un algoritmo de clustering popular en el aprendizaje automático que agrupa datos en K grupos basados en sus características. Los grupos se forman minimizando la varianza dentro de cada grupo, lo cual es conceptualmente similar a maximizar la distancia entre los centros de los grupos.

- **Objetivo**: Minimizar la suma de las distancias cuadradas entre cada punto y el centroide de su cluster.
- **Aplicaciones**: Segmentación de clientes, organización de datos, y compresión de imágenes.


KMeans Iteration:
Total Within Cluster Sum of Squares:
250k
200k
150k
100k
50k
0
2
4
6
8
10
Kmeans Iterations

Selección de Parámetros en K-means

La ejecución del algoritmo K-means requiere la selección de K, el número de clusters, que es uno de los parámetros más importantes y también uno de los más desafiantes de determinar.

- **Método del codo**: Consiste en variar K y calcular el costo (suma de las distancias cuadradas dentro del cluster) para cada valor de K, y buscar el punto donde la mejora deja de ser significativa (punto de codo).
- **Silhouette Score**: Evalúa la calidad de los clusters basándose en cómo de cerca están los puntos dentro de un cluster y lo separados que están los clusters entre sí.

Funcionamiento del Algoritmo K-means

El algoritmo K-means sigue un proceso iterativo simple para clasificar un conjunto de datos en K clusters:

1. **Inicialización**: Seleccionar aleatoriamente K puntos del conjunto de datos como centros iniciales de los clusters.
2. **Asignación**: Asignar cada punto al centroide más cercano.
3. **Actualización**: Recalcular el centroide de cada cluster tomando el promedio de todos los puntos asignados a ese cluster.
4. **Repetición**: Repetir los pasos 2 y 3 hasta que la posición de los centroides no cambie significativamente, lo que indica la convergencia del algoritmo.

Algoritmo K-means: Descripción Paso a Paso

```
Algoritmo K-means:
1. Inicializar k centroides seleccionándolos aleatoriamente de los puntos de datos.
2. Repetir hasta la convergencia:
   a. Asignar cada punto al centroide más cercano.
   b. Actualizar cada centroide para que sea el promedio de los puntos asignados a él.
3. Finalizar cuando no haya cambios significativos en los centroides entre dos iteraciones consecutivas.
```

Matemáticas detrás de K-means

K-means minimiza la suma de las distancias cuadradas entre los puntos y el centroide de su cluster, conocida como la inercia o la suma de los errores cuadráticos dentro del cluster:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \|x_i^{(j)} - c_j\|^2$$

- **(J)**: Función de coste total
- **(k)**: Número de clusters
- **(n)**: Número de puntos en cada cluster
- **(x_i^{(j)})**: i-ésimo punto en el cluster j
- **(c_j)**: Centroide del cluster j

Complejidad del Algoritmo K-means

La complejidad temporal de K-means es generalmente $O(n \cdot k \cdot I \cdot d)$:

- n: Número total de puntos.
- k: Número de clusters.
- I: Número de iteraciones hasta la convergencia.
- d: Número de dimensiones de cada punto.

Aunque eficiente para datos de baja a media dimensionalidad, puede ser computacionalmente costoso para conjuntos de datos muy grandes o de alta dimensionalidad.

Desafíos y Limitaciones de K-means

Aunque K-means es muy útil, tiene varias limitaciones que deben considerarse:

- **Sensibilidad a los valores iniciales**: Los resultados pueden variar significativamente según los centroides iniciales.
- **No adecuado para clusters de formas no esféricas o tamaños variados**: K-means asume que los clusters son esféricos y de tamaño similar, lo que no siempre es el caso.
- **Necesidad de especificar K de antemano**: La elección de K puede ser arbitraria y no siempre es intuitiva sin un análisis previo.

Aplicaciones Prácticas de K-means

K-means es ampliamente utilizado en una variedad de aplicaciones, gracias a su simplicidad y eficacia:

- **Segmentación de mercado**: Agrupar clientes con preferencias o comportamientos similares para marketing dirigido.
- **Organización de inventarios**: Clasificar productos en categorías basadas en características de venta o consumo.
- **Procesamiento de imágenes**: Reducción de colores en imágenes para compresión o para mejorar el procesamiento de imágenes.
- **Agrupación de documentos**: Clasificar documentos o textos en categorías temáticas para una fácil navegación o recuperación de información.

Conclusión sobre K-means

K-means es un algoritmo de clustering poderoso y ampliamente utilizado debido a su simplicidad y eficacia en una variedad de aplicaciones. A pesar de sus limitaciones, como la sensibilidad a los centroides iniciales y la suposición de clusters esféricos, sigue siendo una herramienta fundamental en el análisis de datos y la ciencia de datos para la segmentación y agrupación rápida y eficiente de datos.

DBSCAN

- **DBSCAN**(Density-Based Spatial Clustering of Applications with Noise) es un algoritmo de clustering popular que se basa en la densidad de los datos para formar clusters, permitiendo identificar zonas de alta densidad separadas por zonas de baja densidad.

Principios Básicos de DBSCAN

- **Densidad de puntos**: La clave de DBSCAN es la noción de densidad, que se mide por el número de puntos dentro de un radio específico (ε).
- **Puntos core**: Un punto es un punto core si tiene un número mínimo de otros puntos (MinPts) dentro de su radio ε.
- **Puntos frontera y ruido**: Los puntos que no son core pero están cerca de un punto core son frontera. Puntos que no son ni core ni frontera son considerados ruido.

Funcionamiento de DBSCAN

El algoritmo DBSCAN procede de la siguiente manera:

1. **Selección de Punto Inicial**: Elige un punto al azar que no haya sido evaluado.
2. **Vecindad de Punto**: Encuentra todos los puntos dentro del radio ϵ del punto seleccionado. Estos puntos son los vecinos ϵ.
3. **Evaluación de Punto Central**: Si un punto tiene MinPts o más puntos en su vecindad ϵ, se considera un punto central, y se forma un cluster alrededor de este punto.
4. **Expansión del Cluster**: Todos los puntos vecinos ϵ se agregan al cluster. Luego, para cada uno de estos puntos vecinos, el algoritmo repite el proceso de encontrar más vecinos que cumplan los criterios. Este proceso se repite recursivamente, lo que puede hacer que el cluster crezca.
5. **Procesamiento de Puntos de Ruido y No Centrales**: Si un punto no tiene suficientes vecinos para ser un punto central pero está dentro de la vecindad ϵ de un punto central, se considera un punto frontera. Los puntos que no cumplen ninguno de estos criterios se consideran ruido.
6. **Iteración a través de todos los puntos**: Este proceso se repite para todos los puntos del conjunto de datos hasta que cada punto haya sido evaluado como parte de un cluster o como

Algoritmo DBSCAN: Descripción Paso a Paso

A continuación se presenta el pseudocódigo para el algoritmo DBSCAN, ilustrando cómo opera de manera estructurada:

```
Algoritmo DBSCAN:
1. Para cada punto en el conjunto de datos:
   a. Si el punto ya ha sido evaluado, continúa al siguiente punto.
   b. Recupera todos los puntos vecinos dentro del radio ε (vecindad ε).
   c. Si el número de vecinos en la vecindad ε es menor que MinPts:
      - Marca el punto como ruido (potencialmente temporalmente si es un punto frontera más tarde).
   d. Si el punto tiene al menos MinPts vecinos:
      - Crea un nuevo cluster, y añade el punto a este cluster.
      - Expande el cluster:
        i. Para cada punto en la vecindad ε:
           - Si el punto no ha sido visitado:
             - Marca como visitado.
             - Recupera sus vecinos, y si tiene al menos MinPts vecinos, añade estos vecinos al cluster.
           - Si el punto no es miembro de ningún cluster, añádelo al cluster actual.
2. Repite hasta que todos los puntos hayan sido evaluados y clasificados.
```

Algoritmo DBSCAN: Diagrama de Flujo

La Matemática Detrás de DBSCAN

La matemática detrás de DBSCAN se centra en conceptos de densidad y distancia para identificar y agrupar puntos de datos.

Distancia entre Puntos en DBSCAN

DBSCAN utiliza la distancia euclidiana, que se define como:

$$d(p, q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

donde p y q son puntos en el espacio $\mathbb{R}^n$ y p_i, q_i son las coordenadas de esos puntos.

Epsilon (ϵ) en DBSCAN

Este parámetro define el radio de la vecindad alrededor de cada punto. Cualquier punto a distancia $\leq \epsilon$ de otro es considerado parte de su vecindad.

MinPts (Minimum Points) en DBSCAN

El número mínimo de puntos requeridos para formar un "vecindario denso". Un punto es un punto central si, y sólo si, el número de puntos dentro de su vecindad ϵ (incluido él mismo) es al menos MinPts.

Definiciones Clave en DBSCAN

- **Punto central**: Tiene al menos MinPts dentro de su vecindad ϵ.
- **Punto frontera**: No es central, pero está dentro de la vecindad de un punto central.
- **Punto de ruido**: No es ni central ni frontera.

Formación de Clusters en DBSCAN

Los clusters se forman mediante la exploración y expansión de puntos centrales, siguiendo un proceso iterativo de agregación de puntos que cumplen con los criterios de densidad a un cluster en crecimiento.

Selección de Parámetros en DBSCAN

DBSCAN depende crucialmente de los parámetros ε (epsilon) y MinPts (Minimum Points). Veamos cómo seleccionarlos efectivamente:

- **Epsilon (ε)**: Define el radio de la vecindad alrededor de cada punto. Usar el gráfico de la distancia k-ésima para encontrar un ε adecuado en el punto de máxima curvatura.
- **MinPts**: Determina el número mínimo de puntos en la vecindad ε. Generalmente, MinPts debería ser igual a la dimensión del conjunto de datos más uno o dos.

Manejo de Datos de Alta Dimensionalidad con DBSCAN

Los desafíos de la "maldición de la dimensionalidad" pueden abordarse con técnicas específicas:

- **Reducción de Dimensionalidad**: Utilizar PCA o t-SNE para reducir la dimensionalidad antes de aplicar DBSCAN.
- **Ajuste de Parámetros**: En altas dimensiones, ε y MinPts pueden requerir ajustes y pruebas exhaustivas para optimizar el clustering.

Aplicaciones Prácticas de DBSCAN

DBSCAN es utilizado en varios campos debido a su flexibilidad y eficacia:

- **Segmentación de Mercado**: Identificar grupos de consumidores para estrategias de marketing personalizadas.
- **Biología Computacional**: Agrupar expresiones genéticas o proteínas con funciones similares.
- **Sistemas de Recomendación**: Mejorar las recomendaciones agrupando productos o usuarios con patrones similares.
- **Detección de Anomalías**: Usar la identificación de ruido para detectar actividades anómalas o fraudulentas.

Complejidad del Algoritmo DBSCAN

Dependiendo de la implementación, la complejidad temporal puede ser $O(n^2)$, pero puede mejorarse con estructuras de datos como árboles KD o árboles de búsqueda de rango.

Conclusión sobre DBSCAN

DBSCAN ofrece una herramienta robusta y versátil para el análisis de datos, capaz de manejar diversos tipos de datos y aplicaciones. Su habilidad para identificar clusters de cualquier forma y manejar el ruido lo hace invaluable en muchos contextos prácticos.

Ventajas de DBSCAN

- **No requiere especificar el número de clusters**: A diferencia de K-means, DBSCAN no requiere que el número de clusters sea definido a priori.
- **Maneja clusters de formas irregulares**: Puede identificar clusters de formas no esféricas, lo que es útil en muchos escenarios reales.
- **Robusto ante el ruido**: Distingue efectivamente los puntos de ruido de los puntos de clusters.

Limitaciones de DBSCAN

- **Sensibilidad a los parámetros**: La selección de ε y MinPts puede afectar significativamente los resultados, y su ajuste óptimo no siempre es intuitivo.
- **Dificultad con variaciones de densidad**: DBSCAN puede tener dificultades cuando los clusters varían en densidad, ya que un único valor de ε y MinPts puede no ser adecuado para todos los clusters.
- **Costo computacional en grandes datasets**: Aunque es eficiente, puede ser computacionalmente costoso para datasets muy grandes.

Reducción de Dimensionalidad

Objetivo

Reducir el número de variables consideradas para mejorar la visualización y rendimiento de algoritmos.

Ejemplos de Reducción de Dimensionalidad

- **PCA (Análisis de Componentes Principales)**: Transforma y reduce dimensiones manteniendo la máxima varianza.
- **t-SNE**: Excelente para visualizar datos de alta dimensión manteniendo la estructura local.

Asociación

Propósito

Identificar reglas que expliquen patrones en grandes volúmenes de datos, como en transacciones comerciales.

Ejemplo de Asociación

- **Apriori**: Detecta ítems frecuentemente comprados juntos en transacciones.

Aplicaciones de Algoritmos No Supervisados

- Segmentación de mercado
- Detección de anomalías
- Sistemas de recomendación

Conclusión

Los algoritmos no supervisados ofrecen herramientas poderosas para análisis sin etiquetas predefinidas, facilitando el descubrimiento de conocimiento en grandes datasets.