SESIÓN CLUSTERIZACIÓN Y SUS PRINCIPALES ALGORITMOS

CONTENIDOS:

- ¿En qué consiste la técnica de clusterización?
- Aplicaciones de las técnicas de clusterización.
- Principales algoritmos de clusterización: ventajas y desventajas.
    - K-Means.
    - Clustering jerárquico.
    - DBSCAN.
    - Mezcla Gaussiana.
- Aplicaciones y casos que se pueden resolver con clusterización.

¿EN QUÉ CONSISTE LA TÉCNICA DE CLUSTERIZACIÓN?

La **técnica de clusterización** es un método de **aprendizaje de máquina no supervisado** que permite agrupar un conjunto de datos en subconjuntos o **clusters** (grupos). Su objetivo es organizar los datos de manera que los elementos dentro de un mismo cluster sean lo más similares posible entre sí y, al mismo tiempo, lo más diferentes posible de los elementos en otros clusters.

¿Cómo funciona la clusterización?

Cuando tenemos un gran volumen de datos sin etiquetas (es decir, sin una clasificación previa), la clusterización busca patrones o similitudes en ellos para agruparlos de forma **natural**. Este proceso se basa en la **distancia** o similitud entre los datos.

Por ejemplo, imagina que tienes un conjunto de fotos de diferentes animales y no sabes a qué especie pertenecen. Un algoritmo de clusterización analizaría características como el tamaño, color, forma, etc., y los agruparía en clusters de forma automática. Podría generar grupos como:

- **Grupo 1:** Perros
- **Grupo 2:** Gatos
- **Grupo 3:** Pájaros

Sin que nadie le haya dicho de antemano qué son los perros o los gatos, el algoritmo detecta similitudes y los organiza en grupos.

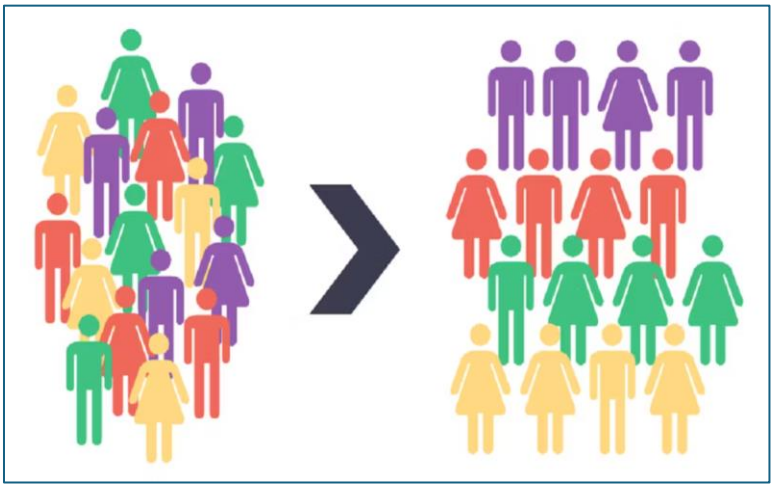

Ilustración 1 Muestra gráfica de la agrupación en clusters. Imagen de chehana

¿Cuándo se usa la clusterización?

Se usa cuando no tenemos etiquetas previas en los datos y queremos descubrir relaciones ocultas entre ellos. A diferencia del aprendizaje supervisado, donde entrenamos un modelo con ejemplos etiquetados, en la clusterización no le decimos al modelo qué buscar, sino que este encuentra estructuras o patrones ocultos por sí mismo.

APLICACIONES DE LAS TÉCNICAS DE CLUSTERIZACIÓN

La clusterización tiene aplicaciones en múltiples campos, ya que permite descubrir patrones ocultos en los datos sin necesidad de etiquetas previas. Algunas de las aplicaciones más importantes son:

1. Segmentación de Clientes (Marketing y Negocios):

Las empresas utilizan la clusterización para agrupar clientes con características similares y así personalizar estrategias de marketing.

Por ejemplo, un supermercado analiza datos de compra y encuentra que hay tres tipos principales de clientes:

- **Clientes A:** Compran productos económicos en grandes cantidades.
- **Clientes B:** Prefieren productos premium y exclusivos.
- **Clientes C:** Compran con poca frecuencia, pero en grandes montos.

Con esta información, la empresa puede ofrecer descuentos personalizados o diseñar campañas dirigidas a cada grupo.

2. Medicina y Biología:

En el área médica, la clusterización ayuda a identificar patrones en enfermedades y pacientes.

Considerando esto, un ejemplo de su uso podría ser un hospital analizando historiales médicos y descubriendo que los pacientes con diabetes se pueden dividir en subgrupos con diferentes síntomas y riesgos. Esto permite:

- Personalizar tratamientos según el grupo del paciente.
- Detectar patrones de enfermedades raras.
- Descubrir nuevos subtipos de enfermedades.

También se usa en análisis genético, agrupando genes con funciones similares para comprender mejor su impacto en la salud.

3. Detección de Fraudes (Finanzas y Seguridad):

Los bancos y aseguradoras utilizan la clusterización para detectar transacciones sospechosas o fraudulentas.

Ejemplo:
Un banco agrupa transacciones según su comportamiento y encuentra que:

- **Grupo 1:** Compras normales de clientes regulares.
- **Grupo 2:** Transacciones inusuales (monto alto, ubicación extraña, compras frecuentes en poco tiempo).

Si una transacción cae en el grupo de riesgo, el banco puede bloquearla o pedir verificación al usuario.

4. Análisis de Datos en Geografía y Climatología:

Se usa en el análisis de mapas y datos geoespaciales para identificar patrones en la distribución de datos ambientales o demográficos.

Ejemplo:

- Agrupar ciudades según niveles de contaminación para diseñar políticas ambientales.
- Identificar zonas de riesgo sísmico analizando datos históricos de terremotos.
- Agrupar regiones según patrones de temperatura y humedad para entender el cambio climático.

5. Recomendación de Contenido (Netflix, Spotify, YouTube):

Las plataformas de streaming usan la clusterización para agrupar usuarios con gustos similares y ofrecer recomendaciones personalizadas. Algunos ejemplos son:

- **Netflix** agrupa usuarios según sus preferencias de películas y series. Si muchos usuarios en un cluster ven "Stranger Things", el sistema recomienda esa serie a otros usuarios del mismo grupo.
- **Spotify** agrupa canciones con características similares y sugiere listas de reproducción personalizadas.

6. Agrupación de Imágenes y Videos (Visión por Computadora):

La clusterización se usa en reconocimiento de imágenes y clasificación automática. Por ejemplo:

- Google Fotos agrupa imágenes automáticamente según las caras que aparecen en ellas.
- Se usa en autos autónomos para identificar objetos en la carretera (coches, peatones, señales de tráfico).
- Los sistemas de vigilancia pueden agrupar actividades sospechosas en una multitud.

7. Optimización de Logística y Distribución:

Las empresas de logística usan clusterización para optimizar rutas de entrega y distribución de productos.

Por ejemplo:

- Amazon agrupa pedidos según la ubicación de los clientes y asigna rutas de entrega más eficientes.
- Empresas de transporte como Uber agrupan zonas con alta demanda para mejorar la disponibilidad de vehículos.

En conclusión, las clusterización no solo organiza datos, sino que permite a las empresas, científicos y gobiernos descubrir patrones ocultos y tomar mejores decisiones.

PRINCIPALES ALGORITMOS DE CLUSTERIZACIÓN: VENTAJAS Y DESVENTAJAS

Los algoritmos de clusterización se dividen en varias categorías, dependiendo de la forma en que agrupan los datos.

K-Means

K-Means es un algoritmo de clusterización basado en particiones. Se usa para dividir un conjunto de datos en K grupos (clusters**)**. El número **K** es elegido por el usuario.

¿Cómo funciona?

1. Se eligen **K centroides** (puntos centrales de cada grupo) al azar.
2. Se asigna cada dato al centroide más cercano.
3. Se recalculan los centroides como el promedio de los puntos asignados a cada grupo.
4. Se repiten los pasos hasta que los centroides no cambien (convergencia).

Ejemplo de uso en empresas:

- **Spotify** usa K-Means para agrupar canciones según características como ritmo, energía y popularidad, ayudando en las recomendaciones de playlists.
- **Uber** usa K-Means para detectar zonas con alta demanda de transporte y posicionar mejor a los conductores.

Ventajas y desventajas de K-Means

Ventajas	Desventajas
Es rápido y eficiente en grandes volúmenes de datos.	Debes definir el número de clusters **K** de antemano.
Fácil de entender e implementar.	No funciona bien si los clusters tienen formas irregulares.
Funciona bien cuando los grupos tienen formas circulares o esféricas.	Sensible a **outliers** (datos atípicos).

Aplicando K-Means

Vamos a usar la librería **sklearn** para clusterizar puntos de datos aleatorios.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Generamos datos aleatorios
np.random.seed(0)
X = np.random.rand(100, 2) * 10  # 100 puntos en un espacio 2D

# Aplicamos K-Means con 3 clusters
kmeans = KMeans(n_clusters=3, random_state=0)
kmeans.fit(X)
labels = kmeans.labels_
centroids = kmeans.cluster_centers_

# Graficamos los clusters
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', marker='o')
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='X', s=200, label="Centroides")
plt.title("Clusterización con K-Means")
plt.legend()
plt.show()
```

Ilustración 2 Ejemplo de la aplicación de K-Means para clusterizar datos aleatoreos

Clustering Jerárquico

El **clustering jerárquico** agrupa los datos en una estructura en forma de árbol (dendrograma). No requiere especificar el número de clusters desde el principio.

Tipos de clustering jerárquico:

1. **Aglomerativo** (bottom-up): Empieza con cada punto como un cluster separado y los une progresivamente.
2. **Divisivo** (top-down): Empieza con un único cluster y lo divide hasta obtener los grupos finales.

Ejemplo de uso en empresas:

- **Google News** usa clustering jerárquico para agrupar noticias similares en categorías.
- **Medicina**: Se usa para identificar nuevas categorías de enfermedades a partir de registros clínicos.

Ventajas y Desventajas del clustering jerárquico

Ventajas	Desventajas
No requiere definir el número de clusters de antemano.	Computacionalmente más costoso que K-Means.
Permite visualizar la relación entre los datos con un dendrograma.	Difícil de aplicar en grandes volúmenes de datos.

Aplicando Clustering Jerárquico:

A continuación, veremos un pequeño ejemplo en Python, de un código que implementa un análisis de clusterización jerárquica en un conjunto de datos X.

```python
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering

# Creamos el dendrograma
plt.figure(figsize=(8, 5))
dendrogram = sch.dendrogram(sch.linkage(X, method='ward'))
plt.title("Dendrograma - Clustering Jerárquico")
plt.show()

# Aplicamos el clustering jerárquico con 3 clusters
hc = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='ward')
labels_hc = hc.fit_predict(X)

# Graficamos los clusters
plt.scatter(X[:, 0], X[:, 1], c=labels_hc, cmap='plasma', marker='o')
plt.title("Clusterización Jerárquica")
plt.show()
```

Ilustración 3 Ejemplo de implementación de clusterización jerárquica

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

DBSCAN es un algoritmo basado en densidad. Identifica clusters como regiones densas de puntos y deja fuera los puntos aislados (**outliers**).

¿Cómo funciona?

1. Encuentra puntos densamente conectados.
2. Expande clusters desde los puntos de mayor densidad.
3. Ignora puntos ruidosos o dispersos.

Ejemplo de uso en empresas:

- **Tesla** usa DBSCAN para agrupar vehículos en análisis de tráfico y conducción autónoma.
- **Sistemas de detección de fraudes** lo aplican para identificar transacciones inusuales que no encajan en patrones normales.

Ventajas y Desventajas del DBSCAN

Ventajas	Desventajas
No requiere especificar el número de clusters.	No funciona bien con clusters de densidad variable.
Funciona bien con formas de clusters irregulares.	Es más lento en grandes volúmenes de datos.

Aplicando DBSCAN:

En este ejemplo vemos una clusterización basada en densidad en un conjunto de datos X.

```
from sklearn.cluster import DBSCAN

# Aplicamos DBSCAN
dbscan = DBSCAN(eps=1.0, min_samples=5)
labels_dbscan = dbscan.fit_predict(X)

# Graficamos los clusters
plt.scatter(X[:, 0], X[:, 1], c=labels_dbscan, cmap='coolwarm', marker='o')
plt.title("Clusterización con DBSCAN")
plt.show()
```

Ilustración 4 Ejemplo de aplicación de DBSCAN

Mezcla Gaussiana (Gaussian Mixture Model, GMM)

GMM es un algoritmo basado en estadística que asume que los datos provienen de varias distribuciones normales (gaussianas) y las combina para encontrar los clusters.

Ejemplo de uso en empresas:

- **Reconocimiento facial** en cámaras de seguridad.
- **Agrupación de clientes en tiendas online** para analizar su comportamiento de compra.

Ventajas y Desventajas de GMM

Ventajas	Desventajas
Más flexible que K-Means, ya que no asume clusters esféricos.	Puede ser lento en grandes conjuntos de datos.
Funciona bien con clusters superpuestos.	Es más difícil de interpretar.

Aplicando GMM:

GMM es similar a K-Means, pero en lugar de asignar cada punto a un solo cluster, calcula la probabilidad de que un punto pertenezca a cada cluter. Esto permite manejar datos con formas más complejas.

```
from sklearn.mixture import GaussianMixture

# Aplicamos el modelo GMM con 3 clusters
gmm = GaussianMixture(n_components=3, random_state=0)
gmm.fit(X)
labels_gmm = gmm.predict(X)

# Graficamos los clusters
plt.scatter(X[:, 0], X[:, 1], c=labels_gmm, cmap='coolwarm', marker='o')
plt.title("Clusterización con Mezcla Gaussiana (GMM)")
plt.show()
```

Ilustración 5 Ejemplo de Aplicación de GMM

APLICACIONES Y CASOS QUE SE PUEDEN RESOLVER CON CLUSTERIZACIÓN

La clusterización se ha aplicado en numerosos sectores para resolver problemas complejos en la toma de decisiones, la segmentación de datos y la optimización de procesos. A continuación, exploramos casos prácticos donde la clusterización ha demostrado su impacto.

Caso 1: Segmentación de clientes en e-commerce (Amazon, Mercado Libre)

- **Problema:** Las tiendas en línea necesitan entender el comportamiento de sus clientes para ofrecer productos relevantes y mejorar las recomendaciones.
- **Solución:** Algoritmos como **K-Means** agrupan clientes según su historial de compras, permitiendo personalizar promociones y mejorar la experiencia de usuario.

Caso 2: Optimización de rutas de entrega (DHL, FedEx, Uber Eats)

- **Problema:** Empresas de logística deben minimizar costos y tiempos de entrega.
- **Solución: DBSCAN** agrupa pedidos por cercanía geográfica para asignar rutas más eficientes a los repartidores.

Caso 3: Diagnóstico y personalización de tratamientos médicos

- **Problema:** Los hospitales deben clasificar a los pacientes según características médicas para mejorar los tratamientos.
- **Solución: GMM y Clustering Jerárquico** han sido utilizados para descubrir subgrupos de pacientes con respuestas similares a tratamientos, optimizando la medicina personalizada.

Recomendación de contenido en plataformas de streaming (Netflix, Spotify, YouTube)

- **Problema:** Las plataformas deben sugerir contenido relevante sin etiquetas previas.
- **Solución: K-Means y GMM** agrupan a los usuarios según sus patrones de consumo, personalizando las recomendaciones.

Seguridad y detección de fraudes en bancos (Visa, Mastercard, PayPal)

- **Problema:** Los fraudes financieros deben ser detectados antes de causar pérdidas.
- **Solución: DBSCAN** detecta transacciones inusuales al agrupar transacciones legítimas y destacar las que no encajan en ningún cluster.

ACTIVIDAD PRÁCTICA GUIADA: SEGMENTACIÓN DE CLIENTES CON K-MEANS

Objetivo:

Aplicar K-Means para agrupar clientes según su comportamiento de compra.

Requisitos:

- Tener instalado Python y Jupyter Notebook (o Google Colab).
- Instalar numpy, matplotlib y sklearn si aún no los tienes (pip install numpy matplotlib scikit-learn).

1. Importar Librerías

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import pandas as pd
```

2. Crear Datos Simulados:

Simularemos 100 clientes con dos variables:

- Monto gastado en compras ($UD)
- Frecuencia de compra (veces al mes)

```
np.random.seed(0)
monto_gastado = np.random.randint(100, 1000, 100)   # Gasto entre $100 y $1000
frecuencia_compra = np.random.randint(1, 30, 100)   # Compran entre 1 y 30 veces al mes

# Convertimos los datos en un DataFrame
clientes = pd.DataFrame({'Monto Gastado': monto_gastado, 'Frecuencia Compra': frecuencia_compra})

# Visualizamos los datos
plt.scatter(clientes['Monto Gastado'], clientes['Frecuencia Compra'])
plt.xlabel('Monto Gastado ($USD)')
plt.ylabel('Frecuencia de Compra (veces al mes)')
plt.title('Distribución de Clientes')
plt.show()
```

3. Aplicar K-Means

```
# Aplicamos K-Means con 3 clusters
kmeans = KMeans(n_clusters=3, random_state=0)
clientes['Cluster'] = kmeans.fit_predict(clientes[['Monto Gastado', 'Frecuencia Compra']])

# Obtenemos los centroides
centroides = kmeans.cluster_centers_

# Graficamos los clusters
plt.scatter(clientes['Monto Gastado'], clientes['Frecuencia Compra'], c=clientes['Cluster'], cmap='viridis', marker='o')
plt.scatter(centroides[:, 0], centroides[:, 1], c='red', marker='X', s=200, label="Centroides")
plt.xlabel('Monto Gastado ($USD)')
plt.ylabel('Frecuencia de Compra (veces al mes)')
plt.title('Segmentación de Clientes con K-Means')
plt.legend()
plt.show()
```

4. **Analizar los Clusters**

Podemos describir los segmentos identificados:

```
clientes.groupby('Cluster').mean()
```

Posible interpretación:

- **Grupo 0:** Clientes que compran con poca frecuencia y gastan poco.
- **Grupo 1:** Clientes frecuentes con gasto moderado.
- **Grupo 2:** Compradores VIP que gastan mucho y compran con frecuencia.