4. Aprendizaje no supervisado

🎯 **OBJETIVO GENERAL**

Comprender los fundamentos del aprendizaje no supervisado, explorando técnicas como **clustering** y **reducción de dimensionalidad**. Implementar algoritmos clave como **k-means** y **PCA** para visualizar y analizar datos sin etiquetas.

---

1. Introducción al Aprendizaje No Supervisado

El aprendizaje automático se divide en dos grandes categorías: **supervisado** y **no supervisado**. La diferencia clave radica en la disponibilidad de etiquetas en los datos de entrenamiento.

Diferencia con el Aprendizaje Supervisado

Característica	Aprendizaje Supervisado	Aprendizaje No Supervisado
Datos de entrenamiento	Contienen **etiquetas** (salidas esperadas).	No tienen etiquetas, solo variables de entrada.
Objetivo	Aprender una función de mapeo entre entrada y salida.	Descubrir patrones, estructuras o relaciones ocultas en los datos.
Ejemplos de algoritmos	Regresión Lineal, Random Forest, Redes Neuronales.	k-Means, PCA, DBSCAN, Autoencoders.
Ejemplo práctico	Clasificar correos como spam o no spam.	Agrupar clientes con comportamientos similares en el mercado.

El aprendizaje supervisado se usa cuando sabemos qué queremos predecir, mientras que en el no supervisado queremos descubrir estructuras ocultas en los datos.

Casos de Uso del Aprendizaje No Supervisado

1. Segmentación de clientes

- Empresas como Amazon o Netflix agrupan a sus usuarios según su comportamiento.
- Ejemplo: Clasificación de clientes según patrones de compra para personalizar ofertas.

2. Detección de anomalías

- Útil en **ciberseguridad** (detección de ataques inusuales).
- Aplicaciones en **banca** para identificar fraudes en transacciones.
- Se analizan patrones normales y se identifican datos que se alejan de la norma.

3. Análisis de imágenes y compresión de datos

- **Reducción de dimensionalidad** en imágenes de alta resolución.
- **Agrupamiento de imágenes similares** sin necesidad de etiquetas.
- **Ejemplo:** Google Photos agrupa fotos similares automáticamente.

4. Biología y Genética

- Clasificación de células en imágenes de microscopio.
- Análisis de secuencias de ADN para identificar similitudes entre especies.

5. Motores de Recomendación

- Empresas como **Spotify, Netflix o YouTube** agrupan usuarios según sus preferencias sin conocer de antemano qué tipo de usuario es cada uno.
- **Ejemplo:** Un usuario que escucha reguetón y salsa es clasificado en un grupo que recibe recomendaciones de esos géneros.

Conclusión

El aprendizaje no supervisado es una herramienta poderosa para encontrar **patrones ocultos** en los datos. A diferencia del aprendizaje supervisado, no requiere etiquetas y es útil para análisis exploratorio, reducción de dimensionalidad y agrupamiento de datos sin información previa.

2. Clustering

¿Qué es el clustering?

El **clustering** es una técnica de aprendizaje no supervisado que agrupa datos en subconjuntos llamados **clusters**. Los datos dentro de un mismo cluster son más **similares** entre sí que con los de otros clusters. Esta técnica permite descubrir patrones en datos sin etiquetar.

Ejemplo visual:

Imagina un conjunto de puntos en un plano sin categorías asignadas. Un algoritmo de clustering puede identificar **agrupaciones naturales** en los datos.

Aplicaciones Reales del Clustering

El clustering tiene diversas aplicaciones en el mundo real:

1. Segmentación de clientes

- Agrupación de usuarios en **e-commerce** para personalizar ofertas.
- Bancos usan clustering para diferenciar clientes según su **nivel de riesgo financiero**.

2. Detección de anomalías

- En **ciberseguridad**, para detectar patrones de comportamiento inusuales en la red.
- En **banca**, para identificar fraudes en transacciones.

3. Análisis de imágenes y compresión de datos

- Reducción de colores en imágenes mediante **k-means**.
- Agrupamiento de imágenes en categorías sin etiquetas previas.

4. Biología y Medicina

- Clasificación de tumores según similitudes en imágenes médicas.
- Agrupación de genes con funciones similares.

5. Motores de Recomendación

- Servicios como **Spotify, Netflix o YouTube** agrupan usuarios con gustos similares para sugerir contenido.

Introducción al Algoritmo k-means

Uno de los algoritmos más usados en clustering es **k-means**, que busca dividir los datos en **k grupos** basándose en la similitud entre los puntos.

Idea Principal: Asignación de puntos a centroides

1. Se seleccionan **k centroides** al azar.
2. Cada punto se asigna al centroide más cercano.

3. Se recalculan los centroides como la media de los puntos asignados a cada cluster.
4. El proceso se repite hasta que los centroides dejan de cambiar significativamente.

Proceso Iterativo de k-means

Paso 1: Inicialización

- Se eligen **k centroides** al azar dentro del conjunto de datos.

Paso 2: Asignación de puntos

- Cada punto se asigna al centroide más cercano utilizando la **distancia euclidiana**.

Paso 3: Actualización de centroides

- Se calculan nuevos centroides tomando la media de los puntos asignados a cada cluster.

Paso 4: Repetición

- Se repiten los pasos 2 y 3 hasta que los centroides se estabilicen.

Ejemplo gráfico:

Si comenzamos con 3 centroides en un conjunto de puntos dispersos, cada iteración de k-means moverá los centroides hasta encontrar una configuración estable.

Limitaciones del Algoritmo k-means

1. **Número de clusters predefinido:**
   - Hay que elegir **k** antes de ejecutar el algoritmo.
   - Técnicas como el **método del codo** ayudan a seleccionar un número adecuado de clusters.
2. **Sensibilidad a la inicialización:**
   - Dependiendo de los centroides iniciales, el resultado puede variar.
   - Para mitigar esto, se usa **k-means++**, que selecciona mejores puntos iniciales.
3. **No maneja bien datos con formas complejas:**
   - Si los clusters tienen formas no esféricas (como en el clustering jerárquico o DBSCAN), k-means puede no ser efectivo.

Conclusión

k-means es un algoritmo potente y eficiente para clustering, pero debe utilizarse con cuidado al seleccionar **k** y al tratar conjuntos de datos con formas complejas. Se puede combinar con otras técnicas como **PCA** para mejorar su rendimiento en datos de alta dimensión.

Ejemplo paso a paso de K-Means

Supongamos que tenemos los siguientes puntos en un plano 2D:

Punto	X	Y
A	2	3
B	3	3
C	3	4
D	5	6
E	8	8
F	9	8

Queremos agrupar estos puntos en **K = 2** clusters usando K-Means.

Paso 1: Inicialización de centroides

Seleccionamos dos puntos aleatoriamente como centroides iniciales. Supongamos que elegimos:

- **Centroide 1 (C1)**: (2,3) → Coincide con el punto A.
- **Centroide 2 (C2)**: (8,8) → Coincide con el punto E.

Paso 2: Asignación de cada punto al centroide más cercano

Calculamos la distancia euclidiana entre cada punto y los centroides:

$$d(A, C1) = \sqrt{(2-2)^2 + (3-3)^2} = 0$$

$$d(A, C2) = \sqrt{(2-8)^2 + (3-8)^2} = \sqrt{36+25} = \sqrt{61} = 7.81$$

Como $d(A, C1) < d(A, C2)$, asignamos A al cluster de **C1**.

Hacemos esto para todos los puntos:

Punto	Distancia a C1 (2,3)	Distancia a C2 (8,8)	Asignación
A (2,3)	0.00	7.81	C1
B (3,3)	1.00	7.21	C1
C (3,4)	1.41	6.40	C1
D (5,6)	4.24	4.24	C1
E (8,8)	7.81	0.00	C2
F (9,8)	8.49	1.00	C2

Después de la primera asignación:

- **Cluster 1 (C1)**: {A, B, C, D}
- **Cluster 2 (C2)**: {E, F}

Paso 3: Recalcular centroides

Calculamos el nuevo centroide de cada cluster promediando las coordenadas de sus puntos.

- **Nuevo C1**:

$$X = \frac{(2+3+3+5)}{4} = 3.25, \quad Y = \frac{(3+3+4+6)}{4} = 4$$

Nuevo **C1** = (3.25, 4)

- **Nuevo C2**:

$$X = \frac{(8+9)}{2} = 8.5, \quad Y = \frac{(8+8)}{2} = 8$$

Nuevo **C2** = (8.5, 8)

Paso 4: Repetir asignación y recalcular centroides

Volvemos a calcular las distancias de cada punto a los nuevos centroides y reasignamos si es necesario.

Punto	Distancia a C1 (3.25,4)	Distancia a C2 (8.5,8)	Asignación
A (2,3)	1.60	7.91	C1
B (3,3)	1.03	7.21	C1
C (3,4)	0.25	6.40	C1
D (5,6)	2.50	4.03	C1
E (8,8)	6.40	0.50	C2
F (9,8)	7.07	0.50	C2

Las asignaciones se mantienen, por lo que el algoritmo converge.

Paso 5: Condición de parada

El algoritmo se detiene cuando los centroides no cambian o cuando se alcanza el número máximo de iteraciones.

Resultado final

- **Cluster 1 (C1 = (3.25, 4))**: {A, B, C, D}
- **Cluster 2 (C2 = (8.5, 8))**: {E, F}

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

# Datos de ejemplo (coordenadas X, Y)
X = np.array([
    [2, 3],  # A
    [3, 3],  # B
    [3, 4],  # C
    [5, 6],  # D
    [8, 8],  # E
    [9, 8],  # F
])

# Aplicar K-Means con k=2
kmeans = KMeans(n_clusters=2, random_state=42, n_init=10)
kmeans.fit(X)

# Obtener los centroides y etiquetas de cada punto
centroids = kmeans.cluster_centers_
labels = kmeans.labels_

# Graficar los puntos con sus clusters asignados
plt.figure(figsize=(8, 6))
colors = ['blue', 'red']
for i in range(len(X)):
    plt.scatter(X[i, 0], X[i, 1], c=colors[labels[i]], s=100, label=f"Punto {chr(65+i)}")

# Graficar centroides
plt.scatter(centroids[:, 0], centroids[:, 1], c='black', marker='X', s=200, label="Centroides")

plt.xlabel("X")
plt.ylabel("Y")
plt.title("Clustering K-Means con K=2")
plt.legend()
plt.grid()
plt.show()
```

3. Reducción de Dimensionalidad

¿Por qué reducir la dimensionalidad?

En muchos problemas de ciencia de datos y aprendizaje automático, los conjuntos de datos contienen un gran número de características (columnas). Si bien esto puede parecer beneficioso, en realidad **demasiada dimensionalidad puede ser perjudicial**.

Razones para reducir la dimensionalidad:

1. **Evitar la maldición de la dimensionalidad**
   - A medida que el número de dimensiones aumenta, los datos se dispersan y los modelos tienen más dificultades para encontrar patrones.
2. **Eliminar ruido y redundancia**
   - Muchas características pueden contener información irrelevante o ser altamente correlacionadas entre sí. Reducirlas ayuda a mejorar el rendimiento del modelo.
3. **Facilitar la visualización**
   - En datasets con más de 3 dimensiones, es imposible representarlos gráficamente de manera directa. Métodos de reducción de dimensionalidad como **PCA (Análisis de Componentes Principales)** permiten visualizar datos en 2D o 3D.
4. **Acelerar el entrenamiento de modelos**
   - Menos dimensiones significan menos cálculos y un entrenamiento más rápido para modelos de aprendizaje automático.

Eliminación de ruido y redundancia en los datos

La reducción de dimensionalidad puede **mejorar la calidad del aprendizaje automático** al enfocarse solo en la información más relevante.

- **Ejemplo:** En un dataset con 100 características, si 20 de ellas contienen información redundante, podemos reducirlas sin perder información valiosa.
- **Método:** PCA selecciona las características más importantes preservando la mayor cantidad de varianza posible.

Visualización de datos en 2D o 3D

Cuando trabajamos con datos de muchas dimensiones, es difícil comprender sus patrones. **PCA** y otras técnicas permiten proyectar datos en un espacio de 2 o 3 dimensiones para su análisis visual.

- **Ejemplo:** En análisis de clientes, podemos reducir cientos de variables a solo **dos dimensiones** y visualizar **grupos naturales** dentro de los datos.

PCA (Análisis de Componentes Principales)

Concepto de Transformación Lineal

PCA es un método matemático que transforma un conjunto de datos en **nuevas variables (componentes principales)** que:

- Son combinaciones lineales de las variables originales.
- Capturan la mayor cantidad posible de variabilidad en los datos.

Ejemplo intuitivo:

Si tenemos un dataset de 3 dimensiones, PCA puede transformarlo en un **nuevo espacio** con **ejes mejor orientados** para capturar la estructura de los datos con menos dimensiones.

Cálculo de Autovalores y Autovectores

PCA utiliza la descomposición de valores propios para encontrar los **componentes principales**.

1. **Paso 1:** Normalizar los datos (restar la media y dividir por la desviación estándar).
2. **Paso 2:** Calcular la matriz de covarianza.
3. **Paso 3:** Obtener **autovalores** y **autovectores** de la matriz de covarianza.
   - Los **autovalores** indican la cantidad de varianza capturada por cada componente.
   - Los **autovectores** representan las direcciones de los nuevos ejes (componentes principales).
4. **Paso 4:** Seleccionar los componentes principales más relevantes (aquellos con los autovalores más grandes).

Interpretación de los Componentes Principales

- **Primer componente principal (PC1):** Captura la **mayor varianza posible** en los datos.
- **Segundo componente principal (PC2):** Es perpendicular a PC1 y captura la **segunda mayor varianza**.
- **Tercer componente principal (PC3), etc.:** Siguen el mismo principio, pero capturan menos varianza.

Ejemplo en la práctica:

Si aplicamos PCA en un dataset con variables económicas (ingreso, gasto, deuda, etc.), el **primer componente** podría representar una combinación de todas esas variables reflejando el "nivel económico" de una persona.

Ejemplo de PCA paso a paso

1. Definir un conjunto de datos

Supongamos que tenemos el siguiente conjunto de datos con dos variables:

x1	x2
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2.0	1.6
1.0	1.1
1.5	1.6
1.1	0.9

2. Centrar los datos

El primer paso en PCA es centrar los datos restando la media de cada variable.

$$\mu_1 = \frac{1}{n}\sum x_1, \quad \mu_2 = \frac{1}{n}\sum x_2$$

Calculamos la media de cada variable:

$$\mu_1 = 1.81, \quad \mu_2 = 1.91$$

Restamos la media a cada valor:

$$x1' = x1 - \mu_1, \quad x2' = x2 - \mu_2$$

Ahora, los datos centrados serían:

x1'	x2'
0.69	0.49
-1.31	-1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01

3. Calcular la matriz de covarianza

La matriz de covarianza se calcula como:

$$\Sigma = \begin{bmatrix} \text{Var}(x1') & \text{Cov}(x1', x2') \\ \text{Cov}(x1', x2') & \text{Var}(x2') \end{bmatrix}$$

Usamos las fórmulas:

$$\text{Var}(X) = \frac{1}{n-1} \sum (X_i - \bar{X})^2$$

$$\text{Cov}(X, Y) = \frac{1}{n-1} \sum (X_i - \bar{X})(Y_i - \bar{Y})$$

Calculamos:

$$\Sigma = \begin{bmatrix} 0.616 & 0.615 \\ 0.615 & 0.716 \end{bmatrix}$$

4. Calcular los valores propios y vectores propios

Los valores propios (λ) se obtienen resolviendo:

$$\det(\Sigma - \lambda I) = 0$$

Los valores propios son:

$$\lambda_1 = 1.284, \quad \lambda_2 = 0.049$$

Los vectores propios correspondientes son:

$$v_1 = \begin{bmatrix} 0.677 \\ 0.736 \end{bmatrix}, \quad v_2 = \begin{bmatrix} -0.736 \\ 0.677 \end{bmatrix}$$

5. Elegir los componentes principales

El primer componente principal ($PC1$) es el vector propio asociado con el mayor valor propio. En este caso, v1v_1v1 representa la dirección de mayor varianza en los datos.

Podemos reducir la dimensión usando solo $PC1$.

6. Transformar los datos

Los datos transformados (nuevas coordenadas en el espacio PCA) se obtienen mediante:

$$Z = X' \cdot V$$

Calculamos la nueva representación:

PC1
0.83
-1.78
0.99
0.29
1.38
0.86
-0.17
-1.14
-0.48
-1.22

Estos valores son la nueva representación de los datos en un solo eje, reduciendo la dimensión de 2D a 1D.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

# Datos originales
data = np.array([
    [2.5, 2.4],
    [0.5, 0.7],
    [2.2, 2.9],
    [1.9, 2.2],
    [3.1, 3.0],
    [2.3, 2.7],
    [2.0, 1.6],
    [1.0, 1.1],
    [1.5, 1.6],
    [1.1, 0.9]
])

# Paso 1: Centrar los datos (Restar la media de cada columna)
mean = np.mean(data, axis=0)
centered_data = data - mean

# Paso 2: Calcular la matriz de covarianza
cov_matrix = np.cov(centered_data, rowvar=False)

# Paso 3: Calcular los valores propios y vectores propios
eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)

# Ordenar por valores propios en orden descendente
idx = np.argsort(eigenvalues)[::-1]
eigenvalues = eigenvalues[idx]
eigenvectors = eigenvectors[:, idx]

# Paso 4: Transformar los datos al nuevo espacio
pca_data = centered_data @ eigenvectors

# Crear un DataFrame para mostrar los datos transformados
df_pca = pd.DataFrame(pca_data, columns=["PC1", "PC2"])
print(df_pca)

# Paso 5: Visualización de los datos en el nuevo espacio
plt.figure(figsize=(8, 6))
plt.scatter(pca_data[:, 0], pca_data[:, 1], color='blue', alpha=0.7)
plt.axhline(0, color='gray', linestyle='--')
plt.axvline(0, color='gray', linestyle='--')
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.title("Proyección de los Datos en el Nuevo Espacio PCA")
plt.grid()
plt.show()
```

Conclusión

PCA es una técnica poderosa para **reducir dimensiones sin perder demasiada información**. Se usa para eliminar ruido, mejorar la eficiencia de modelos y facilitar la visualización. Aunque PCA es un método lineal, hay alternativas como **t-SNE** y **UMAP** que pueden capturar relaciones más complejas.

==========

PRÁCTICAS