Aprendizaje No Supervisado

Analítica de Datos, Universidad de San Andrés

Si encuentran algún error en el documento o hay alguna duda, mandenmé un mail a rodriguezf@udesa.edu.ar y lo revisamos.

1. Intro. al Aprendizaje No Supervisado

El aprendizaje no supervisado es una rama del machine learning que se enfoca en encontrar patrones y estructuras en datos no etiquetados. A diferencia del aprendizaje supervisado, donde tenemos una variable objetivo, acá buscamos descubrir relaciones y estructuras inherentes en los datos.

Los principales objetivos del aprendizaje no supervisado son:

- Descubrir grupos naturales en los datos (clustering)
- Reducir la dimensionalidad de los datos
- Detectar anomalías
- Encontrar reglas de asociación

2. Clustering

El clustering es una técnica que agrupa observaciones similares en clusters o grupos. La idea es que los puntos parecidos se ponen en el mismo grupo porque tienen características similares. Lo podemos usar para agrupar clientes según comportamiento de compras, agrupar documentos según su contenido, etc.

2.1. K-Means

K-Means es uno de los algoritmos más populares y también de los más simples. Imaginá que tenés datos de jugadores de tenis con información como la cantidad de aces por partido y el porcentaje de primeros saques. K-Means va a intentar agruparlos en tres grupos, por ejemplo: jugadores con buen saque, jugadores con bajo rendimiento en el saque, y un grupo intermedio.

¿Cómo funciona K-Means?

- Se especifica el número K de clusters deseados
- El algoritmo inicializa K centroides aleatorios
- Iterativamente:
  - Asigna cada punto al centroide más cercano
  - Recalcula los centroides como el promedio de los puntos asignados
- El proceso continúa hasta la convergencia

2.1.1. Implementación en Python

```
from sklearn.cluster import KMeans

# Crear y entrenar el modelo
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X)

# Obtener centroides
centroids = kmeans.cluster_centers_
```

3. Reducción de Dimensionalidad

Imaginemos que tenemos un dataset con muchas variables, por ejemplo 100 o 1000. Esto puede ser extremadamente complicado para analizar o visualizar, mucho más para después entrenar un modelo. No quisiera perder variables porque todas aportan cierta información, y me interesa poder conservar la mayor cantidad de información sin perder nada. Reducir la dimensionalidad de los datos nos ayuda a simplificar esto manteniendo la información más importante.

3.1. PCA (Principal Component Analysis)

PCA es la técnica más popular para reducir la dimensionalidad de los datos. Imaginemos que tenemos datos de personas: altura, peso, talle de ropa. Las tres variables están bastante relacionadas seguramente (la gente alta suele

pesar más, y usar ropa más grande). PCA va a intentar fusionarlas en nuevas variables que sean menos, contengan la mayor cantidad de la información original y que sean independientes entre sí.

¿Qué hace PCA?

- Toma todas las variables y las combina de una manera especial
- Crea nuevas variables (llamadas componentes principales) que capturan la mayor variación posible
- El primer componente captura la mayor variación, el segundo la segunda mayor, y así sucesivamente

¿Con cuántos componentes principales nos quedamos?

El estándar es querer conservar el 95 % de la varianza. Vamos entonces a quedarnos con los componentes cuya suma de varianza explicada sea 0.95 o más.

3.1.1. Implementación en Python

```python
from sklearn.decomposition import PCA

# Crear y aplicar PCA
pca = PCA(n_components=2)  # Reducimos a 2 dimensiones
X_reduced = pca.fit_transform(X)

# Ver cuanta varianza explica cada componente
expl_variance = pca.explained_variance_ratio_
print(f"var. explicada por cada componente: {expl_variance}")
print(f"var. total explicada: {sum(expl_variance):.2f}")
```

3.2. ¿Cuándo usar PCA?

- **Visualización:** Cuando queres ver patrones en datos con muchas variables
- **Ruido:** Para limpiar datos eliminando variaciones no importantes
- **Colinealidad:** Cuando tenes variables muy correlacionadas entre sí
- **Curse of Dimensionality:** Para evitar problemas cuando tenes muchas variables

3.3. Consideraciones importantes

- PCA funciona mejor con datos normalizados
- La interpretación de los componentes puede ser complicada
- No siempre es la mejor solución - depende del problema
- Es importante verificar cuanta varianza se mantiene

4. Evaluación de Modelos No Supervisados

- **Clustering:**
  - Silhouette Score: Mide qué tan similar es un punto a su propio cluster vs. otros clusters
  - Calinski-Harabasz Index: Ratio entre varianza intra-cluster y entre-clusters
  - Davies-Bouldin Index: Mide la separación entre clusters
- **Reducción de Dimensionalidad:**
  - Varianza explicada
  - Reconstrucción del error
  - Preservación de distancias

```
from sklearn.metrics import silhouette_score,
    calinski_harabasz_score

# Evaluar clustering
silhouette_avg = silhouette_score(X, clusters)
calinski_score = calinski_harabasz_score(X, clusters)
```

4.1. Recursos Extra

Video de Youtube que explica K-Means de una gran manera:
`https://www.youtube.com/watch?v=4b5d3muPQmA`

Video de Youtube que explica PCA de una gran manera:
`https://www.youtube.com/watch?v=FgakZw6K1QQ`