SESIÓN REDUCCIÓN DIMENSIONAL

CONTENIDOS:

- Qué es la reducción dimensional.
- Principales algoritmos de reducción de dimensionalidad, ventajas y desventajas.
    - PCA.
    - T-SNE.
- Aplicaciones y casos que se pueden resolver con reducción de dimensionalidad.

QUÉ ES LA REDUCCIÓN DIMENSIONAL

La reducción dimensional es una técnica utilizada en el procesamiento de datos y el aprendizaje de máquina para reducir la cantidad de variables o dimensiones en un conjunto de datos mientras se conserva la mayor cantidad posible de información relevante. Esto es crucial en problemas con datos de alta dimensionalidad donde el almacenamiento, procesamiento y análisis pueden volverse computacionalmente costosos y propensos a la maldición de la dimensionalidad.

¿Para qué se usa la reducción dimensional?

- **Mejora del rendimiento** de los modelos de aprendizaje de máquina al eliminar ruido y redundancia en los datos.
- **Reducción de los tiempos de cálculo**, optimizando el uso de recursos computacionales.
- **Visualización de datos** en 2D o 3D para facilitar el análisis.
- **Eliminación de la multicolinealidad** en los datos para mejorar la interpretabilidad.

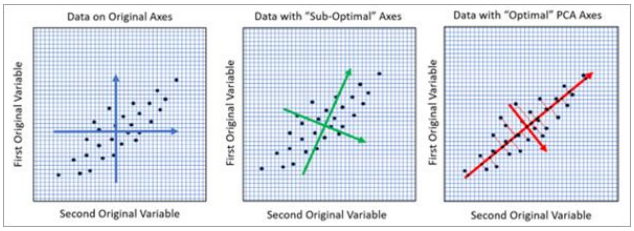

Ilustración 1 Esquema Funcionamiento Reducción de dimensión. ArcGIS

PRINCIPALES ALGORITMOS DE REDUCCIÓN DE DIMENSIONALIDAD, VENTAJAS Y DESVENTAJAS

Existen dos enfoques principales:

1. **Reducción de dimensionalidad basada en selección de características**:
    - En este enfoque, se eligen un subconjunto de variables o características que son las más relevantes para el problema en cuestión.
    - Ejemplos de técnicas incluyen:
        - ***Métodos de filtrado***, que seleccionan características en función de métricas estadísticas como correlación o información mutua.
        - ***Métodos de envoltura (wrapper)***, que utilizan algoritmos de aprendizaje automático para evaluar la calidad de las características seleccionadas.
        - ***Métodos basados en embebido*****,** que integran la selección de características dentro del proceso de entrenamiento del modelo, como el LASSO.
2. **Reducción de dimensionalidad basada en transformaciones**:
    - Este enfoque implica transformar los datos originales en un nuevo espacio de menor dimensionalidad, de manera que se preserve la mayor cantidad de información posible.
    - Ejemplos comunes incluyen:
        - **PCA (Análisis de Componentes Principales)**: Reduce la dimensionalidad al encontrar nuevas variables llamadas componentes principales que maximizan la varianza.
        - **t-SNE (t-Distributed Stochastic Neighbor Embedding)**: Utilizado para la visualización de datos, especialmente en espacios de alta dimensionalidad.
        - **Autoencoders**: Redes neuronales utilizadas para aprender una representación comprimida de los datos.

Ambos enfoques son útiles en diferentes escenarios. La selección de uno depende de factores como el tipo de datos, el propósito del análisis y los requisitos del modelo de aprendizaje automático.

A continuación, veremos en detalle las técnicas PAC, t-SNE y Autoencoders.

PCA: Análisis de Componentes Principales

El PCA *(Principal Component Analysis)* es una técnica estadística ampliamente utilizada para reducir la dimensionalidad de los datos. La idea principal es identificar una nueva representación del conjunto de datos en términos de componentes principales. Cada componente principal es una combinación lineal de las variables originales y está diseñado para:

- Maximizar la varianza de los datos (capturar la mayor cantidad de información posible).
- Minimizar redundancias entre variables (colinearidad).

Esto es útil para simplificar conjuntos de datos complejos, eliminar el "ruido" y visualizar datos de alta dimensionalidad.

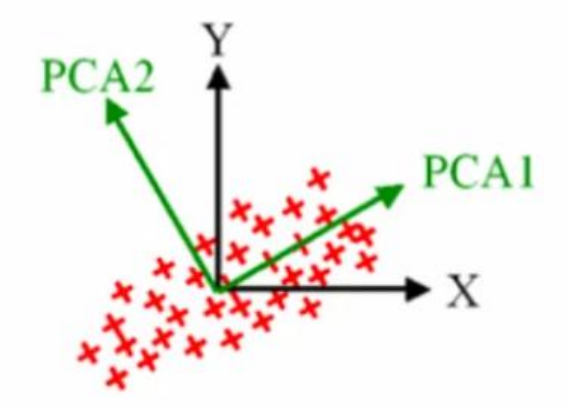

Ilustración 2 Intuición de PCA. complex-system.ai.com

¿Cómo funciona PCA?

1. **Cálculo de la matriz de covarianza**:
    - La matriz de covarianza resume cómo las variables originales están relacionadas entre sí.
    - Cada elemento de la matriz mide la covarianza entre dos variables específicas, lo que refleja la correlación en términos de su magnitud y dirección.

2. **Obtención de autovalores y autovectores**:
    - Los **autovalores** indican cuánta varianza explica cada componente principal. Componentes con mayores autovalores aportan más información.
    - Los **autovectores** corresponden a los "ejes" o "direcciones" del nuevo espacio transformado. Son las direcciones donde los datos tienen más variabilidad.
3. **Selección de componentes principales**:
    - Se ordenan los componentes principales según su varianza explicada (es decir, sus autovalores).
    - El número de componentes seleccionados se basa típicamente en el porcentaje de varianza total que queremos conservar (por ejemplo, el 90% o el 95%).
4. **Proyección de los datos**:
    - Los datos originales se transforman al nuevo espacio definido por los componentes principales seleccionados. Esto reduce su dimensionalidad mientras se mantiene la mayor cantidad de información relevante.

Ejemplo práctico de PCA:

Imagina un conjunto de datos con tres variables altamente correlacionadas, como altura, peso e índice de masa corporal. PCA podría transformar estos tres factores en dos componentes principales:

1. Un componente que refleje la tendencia general de tamaño (altura + peso).
2. Otro componente menor que capture diferencias sutiles no explicadas en el primero.

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

# Crear un conjunto de datos simulado con altura, peso e IMC
np.random.seed(42)
n_muestras = 100

altura = np.random.normal(170, 10, n_muestras)  # cm
peso = np.random.normal(70, 15, n_muestras)     # kg
imc = peso / (altura / 100) ** 2                # kg/m²
```

Ilustración 3 Primera parte del algoritmo

```python
# Construcción del DataFrame
df = pd.DataFrame({'Altura': altura, 'Peso': peso, 'IMC': imc})

# Normalizar los datos antes de aplicar PCA
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df)

# Aplicar PCA para reducir de 3 a 2 dimensiones
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Convertir los componentes en un DataFrame
df_pca = pd.DataFrame(X_pca, columns=['Componente 1', 'Componente 2'])

# Visualizar los resultados
plt.figure(figsize=(8, 6))
plt.scatter(df_pca['Componente 1'], df_pca['Componente 2'], alpha=0.7, color='blue')
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.title('PCA aplicado a datos de Altura, Peso e IMC')
plt.grid()
plt.show()

# Explicación de la varianza explicada por cada componente
print("Varianza explicada por cada componente:", pca.explained_variance_ratio_)
```

Ilustración 4 Segunda parte del algoritmo de ejemplificación

t-SNE: t-Distributed Strochastic Embedding

t-SNE (t-Distributed Stochastic Neighbor Embedding) es una técnica de reducción de dimensionalidad no lineal, diseñada específicamente para preservar la estructura **local** de los datos. A diferencia de métodos lineales como PCA, t-SNE es particularmente efectivo cuando los datos tienen relaciones complejas o no lineales.

Es especialmente conocida por su uso en **visualización de datos de alta dimensionalidad** en 2 o 3 dimensiones, lo que facilita la interpretación y análisis de patrones. Es comúnmente utilizada en áreas como aprendizaje automático, bioinformática y procesamiento de imágenes.

¿Cómo funciona t-SNE?

El algoritmo opera en varias etapas principales para transformar datos a un espacio de menor dimensionalidad:

1. **Modela las relaciones locales**:
    - t-SNE mide las similitudes entre puntos de datos en el espacio de alta dimensionalidad utilizando probabilidades condicionales. Esto se basa en la

distancia entre puntos cercanos: cuanto más cercanos, mayor probabilidad de estar relacionados.

- En esencia, intenta capturar cómo de "cercanos" o similares son los datos en su espacio original.

2. **Define un espacio de menor dimensionalidad**:
   - En el espacio de menor dimensión (por ejemplo, 2D o 3D para visualización), el algoritmo asigna puntos de manera que las relaciones de similitud (estructuras locales) entre ellos se parezcan lo más posible a las del espacio original.
3. **Minimiza la divergencia de similitud (KL-Divergence)**:
   - t-SNE ajusta las posiciones de los puntos en el espacio de menor dimensionalidad iterativamente para minimizar la diferencia entre las similitudes originales y las proyectadas. Este paso es clave para garantizar que los patrones locales se preserven.

Ejemplo práctico de t.SNE:

Imagina un conjunto de datos con descripciones de música, como género, tempo, tonalidad e intensidad. t-SNE podría proyectar estas características en un espacio 2D, agrupando canciones similares (por ejemplo, jazz o rock) en clústeres cercanos que preservan relaciones locales.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
from sklearn.preprocessing import StandardScaler

# Crear un conjunto de datos simulado con características musicales
np.random.seed(42)
n_samples = 200

generos = np.random.choice(['Rock', 'Jazz', 'Clásica'], n_samples)
tempo = np.random.normal(120, 20, n_samples)  # BPM
intensidad = np.random.normal(0.7, 0.1, n_samples)  # Volumen normalizado
tonalidad = np.random.randint(0, 12, n_samples)  # Do=0, Re=1, ..., Si=11
```

Ilustración 5 Primera parte del Algoritmo

```python
# Construcción del DataFrame
df = pd.DataFrame({'Género': generos, 'Tempo': tempo, 'Intensidad': intensidad, 'Tonalidad': tonalidad})

# Convertir variables categóricas a numéricas
df['Género'] = df['Género'].astype('category').cat.codes

# Normalizar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df[['Género', 'Tempo', 'Intensidad', 'Tonalidad']])

# Aplicar t-SNE
tsne = TSNE(n_components=2, perplexity=30, random_state=42)
X_tsne = tsne.fit_transform(X_scaled)

# Convertir los resultados en un DataFrame
df_tsne = pd.DataFrame(X_tsne, columns=['Componente t-SNE 1', 'Componente t-SNE 2'])
df_tsne['Género'] = generos

# Graficar resultados
plt.figure(figsize=(8, 6))
for genero in ['Rock', 'Jazz', 'Clásica']:
    subset = df_tsne[df_tsne['Género'] == genero]
    plt.scatter(subset['Componente t-SNE 1'], subset['Componente t-SNE 2'], label=genero, alpha=0.7)

plt.xlabel('Componente t-SNE 1')
plt.ylabel('Componente t-SNE 2')
plt.title('t-SNE aplicado a características de música')
plt.legend()
plt.grid()
plt.show()
```

Ilustración 6 Segunda parte del algoritmo de ejemplificación

Autoencoder

Un **Autoencoder** es un tipo de red neuronal utilizada para la reducción de dimensionalidad y la representación comprimida de datos.

¿Cómo funcionan?

1. **Codificación**: Los datos de entrada se transforman en una representación comprimida (codificada) en una capa intermedia más pequeña.
2. **Decodificación**: Desde esa representación comprimida, los datos se reconstruyen lo más parecido posible a los originales.

Objetivo principal:

El modelo aprende a identificar patrones clave al minimizar la pérdida entre los datos originales y los reconstruidos. Esto permite representar los datos de forma eficiente y detectar características importantes.

Ejemplo práctico:

Imagina que tienes imágenes en alta resolución. Un autoencoder puede reducir la información a un conjunto más pequeño de números (como "resúmenes" de las imágenes) y luego reconstruirlas para usarlas en tareas como compresión o detección de anomalías.

Son especialmente útiles en aplicaciones como compresión de datos, detección de fraudes o generación de contenido.

Ventajas y Desventajas de PCA, t-SNE y Autoencoders:

Método	Ventajas	Desventajas
PCA	Simple y rápido de implementar.	Solo captura relaciones lineales entre variables.
	Útil para datos donde la varianza explica patrones significativos.	Puede perder información si los datos tienen relaciones no lineales.
t-SNE	Excelente para la visualización en 2D o 3D.	Escalabilidad limitada: no es eficiente con grandes volúmenes de datos.
	Preserva la estructura local de los datos, revelando patrones complejos.	Los resultados no son fácilmente interpretables ni reproducibles (pueden variar entre ejecuciones).
Autoencoders	Capturan relaciones tanto lineales como no lineales.	Requieren una cantidad considerado de datos para entrenar adecuadamente.
	Útiles para tareas específicas como comprensión, detección de anomalías y generación de datos.	Su interpretación es menos intuitiva compara con PCA o t-SNE.

APLICACIONES Y CASOS QUE SE PUEDEN RESOLVER CON REDUCCIÓN DE DIMENSIONALIDAD

1. Análisis de Datos Genéticos

La reducción de dimensionalidad se usa para procesar datos genéticos, donde se analizan miles de genes para encontrar patrones clave asociados a enfermedades o características hereditarias. PCA y t-SNE pueden identificar agrupaciones en perfiles genéticos y simplificar su análisis.

2. Reconocimiento de Imágenes

En visión por computadora, técnicas como PCA se utilizan para reducir el tamaño de las imágenes eliminando redundancias sin perder información relevante. Esto es útil en reconocimiento facial y compresión de imágenes.

3. Análisis de Sentimientos y Procesamiento de Lenguaje Natural (NLP)

Los modelos de NLP suelen manejar grandes volúmenes de texto representados en vectores de alta dimensión. La reducción de dimensionalidad ayuda a reducir la complejidad computacional, permitiendo entrenar modelos más eficientes.

4. Sistemas de Recomendación

En plataformas de streaming o comercio electrónico, PCA y t-SNE pueden ayudar a encontrar patrones en el comportamiento de los usuarios y mejorar la personalización de recomendaciones.

5. Finanzas y Análisis de Mercado

La reducción de dimensionalidad se emplea en el análisis de datos financieros para identificar tendencias ocultas y relaciones entre múltiples variables económicas, facilitando la toma de decisiones en inversiones y predicciones del mercado.