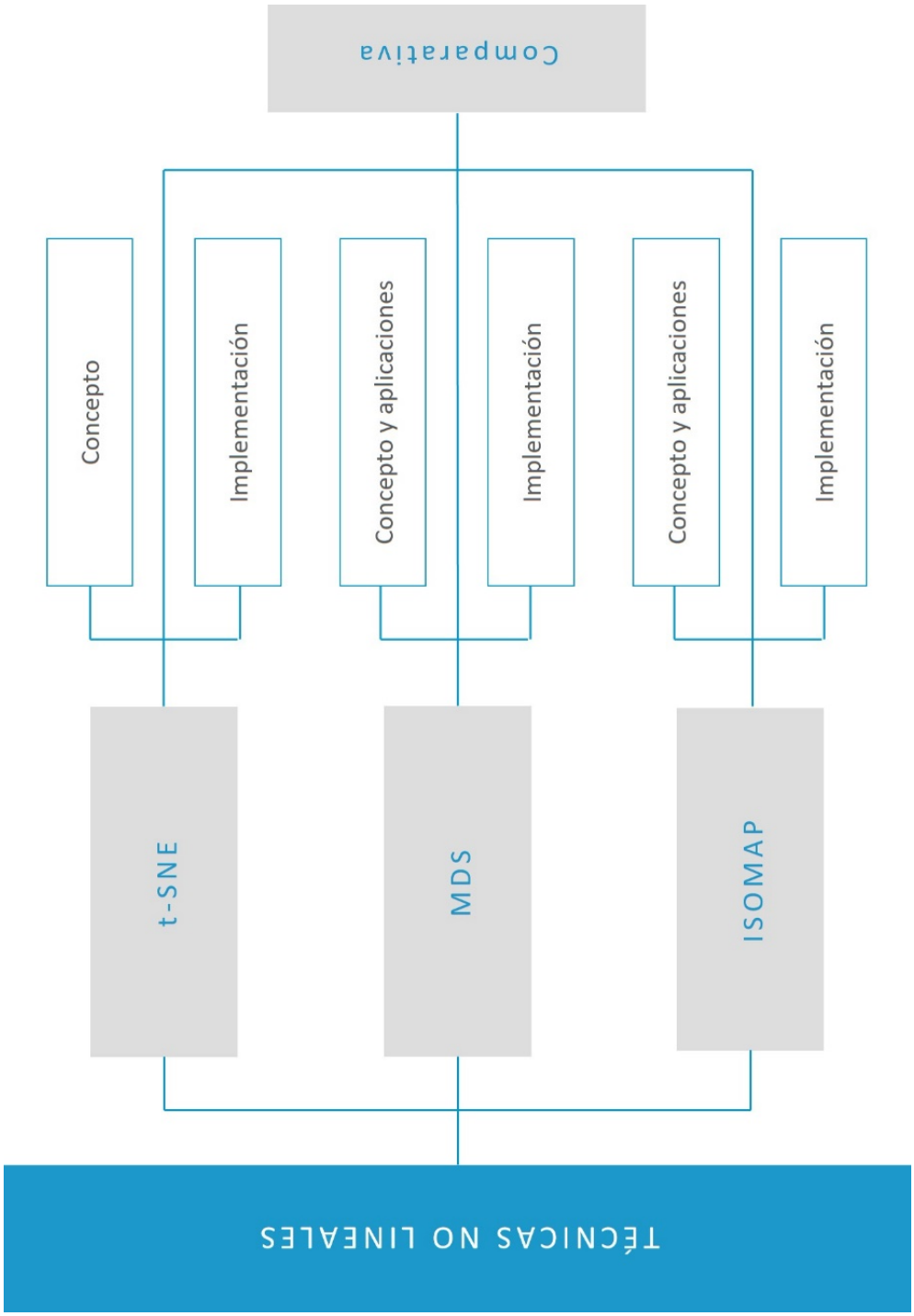
Aprendizaje Automático No Supervisado

Tema 7. Técnicas no lineales. t-SNE, MDS e ISOMAP

Índice

Esquema



7.1. Introducción y objetivos

Cuando vimos PCA nos dimos cuenta de que la visualización y el análisis de grandes cantidades de datos puede ser complejo, pero es importante para la comprensión y detección de patrones.

En este tema seguiremos explorando diversas técnicas de reducción de dimensionalidad, siendo el t-SNE (t-Distributed Stochastic Neighbor Embedding) una de las más destacadas (Belkina, Ciccolella, Anno, Halpert, Spidlen y Snyder-Cappione, 2019). El t-SNE permite transformar datos de múltiples dimensiones a representaciones bidimensionales o tridimensionales, facilitando la identificación de estructuras y relaciones intrínsecas que serían difíciles de observar en su espacio original. Su capacidad para preservar las relaciones locales entre los puntos de datos la convierte en una técnica invaluable para la exploración de conjuntos de datos complejos (Soni, Prabakar y Upadhyay, 2020; Cheng, Wang y Xia, 2021).

Además del t-SNE, exploraremos otras dos técnicas: MDS (Multi-Dimensional Scaling) e Isomap (Abdi, 2007; Tenenbaum, de Silva y Langford, 2000). Cada una de estas ofrece distintos enfoques y ventajas para la reducción de dimensionalidad y la visualización de datos. PCA es particularmente útil para datos lineales, mientras que t-SNE, MDS e Isomap son eficaces para descubrir estructuras no lineales. A través de la implementación práctica y comparación de estas técnicas, los estudiantes podrán entender mejor cada uno de los métodos.

Como **objetivos** de este tema nos planteamos:

- Comprender los fundamentos teóricos de las técnicas de reducción de dimensionalidad, explicando conceptos y diferencias entre t-SNE, MDS e Isomap.
- Aplicar los algoritmos de t-SNE, MDS e Isomap, implementando cada técnica en Python utilizando Scikit-learn.

Ideas clave

- Evaluar las ventajas y limitaciones de cada una de las técnicas de reducción de dimensionalidad, identificando fortalezas y analizando las ventajas y desafíos del uso de cada técnica.

7.2. Entendiendo el t-SNE

t-SNE o incrustación de vecinos estocásticos distribuidos en t es un método estadístico que permite visualizar datos de alta dimensionalidad en un espacio de menor dimensionalidad, generalmente dos o tres dimensiones.

La idea central de t-SNE es mapear los puntos de datos de alta dimensionalidad en un espacio de menor dimensionalidad, preservando las relaciones locales entre los puntos. Se mide la similitud entre puntos de datos en el espacio de dimensión alta y se representa dicha similitud en función de sus probabilidades. Una vez se tienen las distancias se construye una distribución de probabilidad similar en el espacio de dimensiones inferiores y se minimiza la diferencia entre las dos distribuciones utilizando la técnica del descenso del gradiente. De esta forma se captura de manera efectiva la estructura local de los datos, lo que hace particularmente útil la visualización de datos complejos y el descubrimiento de patrones.

Cuando se habla de preservar las relaciones entre los puntos, se refiere a mantener las distancias relativas y las similitudes entre puntos de datos vecinos cuando se asignan desde un espacio de alta dimensionalidad a un espacio de menor dimensión.

Imaginemos que tenemos un conjunto de datos que se encuentran muy cercanos unos de otros en el espacio. Una vez que se reduce la dimensionalidad, los puntos siguen estando muy juntos en el espacio 2D o 3D. De igual forma, si dos puntos están lejos uno del otro en el espacio original, deberían permanecer alejados en un espacio de 2D o 3D.

El concepto base del t-SNE es calcular las similitudes entre cada punto de datos con todos los demás puntos del conjunto de datos. Una vez se tienen estas puntuaciones, el algoritmo reduce la dimensionalidad a 2D o 3D, mientras se preservan las relaciones locales, encontrando los vecinos más cercanos entre

puntos.

Algoritmo

Vamos a ver el paso a paso del algoritmo:

- **Selección** del conjunto de **datos.**
- **Preprocesamiento y normalización** de los **datos.**
- **Cálculo de distancias en la dimensión original:** calcular las distancias entre puntos no es muy confiable debido a la «maldición de la dimensionalidad», por lo que t-SNE utiliza probabilidades. Para cada punto de datos, se traza una distribución gaussiana a su alrededor con una media cero y una desviación estándar que se determina en función de la densidad de los puntos cercanos alrededor de un punto $x1$.

- En el eje X (x-axis), se van a representar las distancias desde un punto de referencia, como $x1$ (imaginemos que es un punto específico en nuestro conjunto de datos). Para cada punto en nuestro conjunto de datos, calculamos la distancia a $x1$. Las distancias se trazan a lo largo del eje X. Esto quiere decir que para cada punto del conjunto de datos se tendrá una posición en el eje X basada en qué tan lejos está de $x1$.

- En el eje Y (y-axis), vamos a representar la densidad de probabilidad. La idea es representar qué tan probable es que un punto de datos se encuentre a cierta distancia de $x1$. Esto se calcula utilizando la función de densidad de probabilidad, que nos permite determinar la probabilidad de cada punto de datos en relación con $x1$. Esta probabilidad actúa como una puntuación de similitud, indicando qué tan similar es un punto al punto de referencia $x1$.

- De manera similar, se aplican los dos pasos anteriores a cada punto del conjunto de datos, lo que nos da como resultado una matriz de nxn. Donde la puntuación de similitud para cada punto se registra en relación con todos los demás puntos de

datos. La distribución es única alrededor de cada punto, la probabilidad del punto $x1$ dado un punto $x2$ es $P(x1 \vee x2)$ y no es necesariamente igual a la probabilidad de $x2$ dado un punto $x1 P(x2 \vee x1)$. Un valor alto de P entre dos puntos significa que son vecinos entre sí, y un valor bajo indica que están lejos el uno del otro.

- **Distribución de probabilidades t de Student:** para reducir la dimensionalidad, se distribuyen puntos aleatoriamente en el eje x. Se calcula la puntuación de similitud de cada punto con relación a los demás, obteniendo otra matriz de nxn.

- Con las dos **matrices de** nxn**,** donde una representa las puntuaciones de similitud en la dimensión superior y la otra representa las puntuaciones de similitud en una dimensión inferior, se define una función de coste basada en la divergencia de Kullback-Leibler entre las distribuciones de alta y baja dimensión. Se alinean la matriz de dimensión inferior con la de dimensión superior, utilizando técnicas de optimización como el descenso del gradiente, para minimizar la función de coste. Esto implica ajustar la posición de los puntos de forma iterativa hasta que la matriz de similitud en la dimensión inferior se parezca lo máximo posible a la de dimensión superior. Dicho de otra forma, se itera hasta que las posiciones de los puntos en el espacio de baja dimensión converjan a una configuración que minimice la función de coste.

- **Visualización y análisis:** se generan visualizaciones de los datos en el espacio de baja dimensión, interpretando los resultados y analizando las agrupaciones formadas.

Formulación matemática del t-SNE

Para cada punto de datos en el espacio de alta dimensión calculamos su similitud con todos los demás puntos utilizando una distribución gaussiana. De igual manera, en el espacio de baja dimensionalidad, se calculan las similitudes usando una distribución t.

Similitud en alta dimensionalidad

Dado un conjunto de datos $X = \{x_1, x_2, \ldots, x_N\}$ con una similitud por pares p_{ij}, donde p_{ij} representa la probabilidad condicional de elegir x_j como vecino de x_i bajo una distribución gaussiana centrada en x_i, definida como:

$$p_{j \vee i} = \frac{\exp\left(\frac{-\|x_i - x_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(\frac{-\|x_i - x_k\|^2}{2\sigma_i^2}\right)}$$

Similitud en baja dimensionalidad

Para el espacio bajo-dimensional $Y = \{y_1, y_2, \ldots, y_N\}$ la similitud q_{ij} está definida por la similitud en el espacio bajo-dimensional usando la distribución t-Student con un grado de libertad:

$$q_{j \vee i} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq i} \left(\left(1 + \|y_i - y_k\|^2\right)^{-1}\right)}$$

La función objetivo

Deseamos minimizar la diferencia entre las similitudes de puntos en el espacio de alta dimensión y sus contrapartes en el espacio de baja dimensión. Se mide esta diferencia utilizando la divergencia de Kullback-Leibler (KL). La divergencia KL mide qué tanto diverge una distribución de probabilidad de otra. En este caso, vamos a medir qué tanto divergen los espacios de alta y de baja dimensionalidad.

La función objetivo C está definida como la divergencia Kullback-Leibler entre la distribución p_{ij} y la p_{ij}, ponderado por la constante Perplexity que controla el número efectivo de vecinos.

$$C = \sum_i KL(P_i \vee Q_i) = \sum_i \sum_j p_{j \vee i} \log \frac{p_{j \vee i}}{q_{j \vee i}}$$

Descenso del gradiente

Para minimizar la divergencia de KL, utilizamos el descenso del gradiente. Lo que hace es ajustar las posiciones de los puntos en el espacio de baja dimensionalidad. En cada iteración, se calcula el gradiente con respecto a las posiciones de los puntos en el espacio de baja dimensionalidad. El gradiente lo que nos indica es la dirección en la que debemos movernos en cada punto para reducir la diferencia entre similitudes entre alta y baja dimensión. Una vez actualizada la posición de los puntos en función del gradiente, converge gradualmente hacia una configuración donde las similitudes de baja dimensión coinciden con las del espacio de alta dimensionalidad.

El gradiente de la función objetivo con respecto a los puntos Y de baja dimensión se calcula utilizando las posiciones de los puntos en el espacio de baja dimensión de forma iterativa.

Gradiente de la función de coste $\frac{\partial C}{\partial y_i}$ con respecto a los puntos y_i está dado por:

$$\frac{\partial C}{\partial y_i} = 4 \sum_j \left(p_{j \vee i} - q_{j \vee i}\right) \left(y_i - y_j\right) \left(1 + \|y_i - y_j\|^2\right)^{-1}$$

En lugar de calcular el gradiente de la función objetivo usando todos los puntos de datos a la vez (lo cual sería muy costoso computacionalmente), el SGD actualiza las posiciones de los puntos de manera iterativa y estocástica. Esto significa que, en cada iteración, se selecciona un subconjunto aleatorio (mini-batch) de los datos para calcular una aproximación del gradiente.

Aplicaciones del t-SNE (Awan, 2023)

- **Agrupación y clasificación:** agrupar puntos de datos similares en dimensiones inferiores. Se suele utilizar para encontrar patrones en los datos.

Ideas clave

- **Detección de anomalías:** se pueden identificar valores atípicos y anomalías en los datos.
- **Procesamiento del lenguaje natural:** visualiza incrustaciones de palabras generadas a partir de un gran corpus de texto que facilita la identificación de similitudes y relaciones entre palabras.
- **Seguridad informática:** se pueden visualizar patrones de tráfico de red y detectar anomalías.
- **Investigación del cáncer:** para visualizar perfiles moleculares de muestras de tumores e identificar subtipos de cáncer.
- **Interpretaciones geológicas:** visualización de atributos sísmicos e identificación de anomalías geológicas.
- **Procesamiento de señales biomédicas:** para visualizar electroencefalogramas y detectar patrones de actividad cerebral.

Ventajas y desventajas del t-SNE (Sachinsoni, 2024)

- Si se aplica correctamente, puede ofrecer visualizaciones muy intuitivas preservando la estructura local de los datos en las dimensiones inferiores.
- Es costoso computacionalmente.
- No es muy bueno preservando la estructura global.
- Sensible a hiperparámetros.
- Puede quedarse atascado en los mínimos locales.
- La interpretación puede ser un desafío.

7.3. t-SNE vs. PCA

Tanto t-SNE como PCA son **técnicas de reducción de dimensionalidad.** PCA funciona mejor con datos lineales; busca identificar los componentes principales subyacentes en los datos proyectándolos en dimensiones inferiores, minimizando la variación y preservando grandes distancias por pares.

A t-SNE no le interesa si los datos son lineales o no, se centra en preservar las similitudes por pares entre puntos de datos en un espacio de dimensiones inferiores. t-SNE se preocupa por preservar pequeñas distancias por pares, mientras que PCA se enfoca en mantener grandes distancias por pares para maximizar la varianza. En la Figura 1, se muestra un ejemplo de reducción de dimensionalidad del conjunto de datos MNIST utilizando PCA y t-SNE.

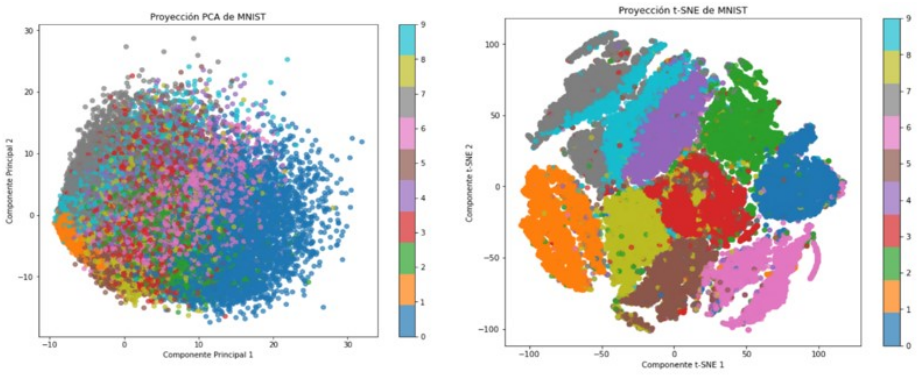


Figura 1. PCA vs. t-SNE conjunto de datos MNIST. Fuente: elaboración propia.

En la figura 1, podemos ver como t-SNE nos da una mejor visualización de los grupos de datos que perteneces a los diferentes dígitos en comparación con PCA. Los datos del MNIST son no lineales y t-SNE captura mejor los datos que PCA.

7.4. Implementación del t-SNE en Python

Veamos el ejemplo con el conjunto de datos MNIST; un conjunto de datos que contiene imágenes de dígitos del 0 al 9 escrito a mano.

```
#Importar las librerías necesarias
import numpy as np
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
from sklearn.datasets import fetch_openml
from sklearn.preprocessing import StandardScaler

# Descargar el conjunto de datos MNIST
mnist = fetch_openml('mnist_784', version=1)
X = mnist.data
y = mnist.target.astype(int)

# Normalizar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Reducir la dimensionalidad a 2 componentes usando t-SNE (puede tardar
varios minutos)
tsne = TSNE(n_components=2, random_state=42)
X_tsne = tsne.fit_transform(X_scaled)

# Crear una gráfica de los datos proyectados
plt.figure(figsize=(10, 8))
scatter = plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=y, cmap='tab10',
alpha=0.7)
plt.colorbar(scatter)
plt.xlabel('Componente t-SNE 1')
plt.ylabel('Componente t-SNE 2')
plt.title('Proyección t-SNE de MNIST')
plt.show()
```

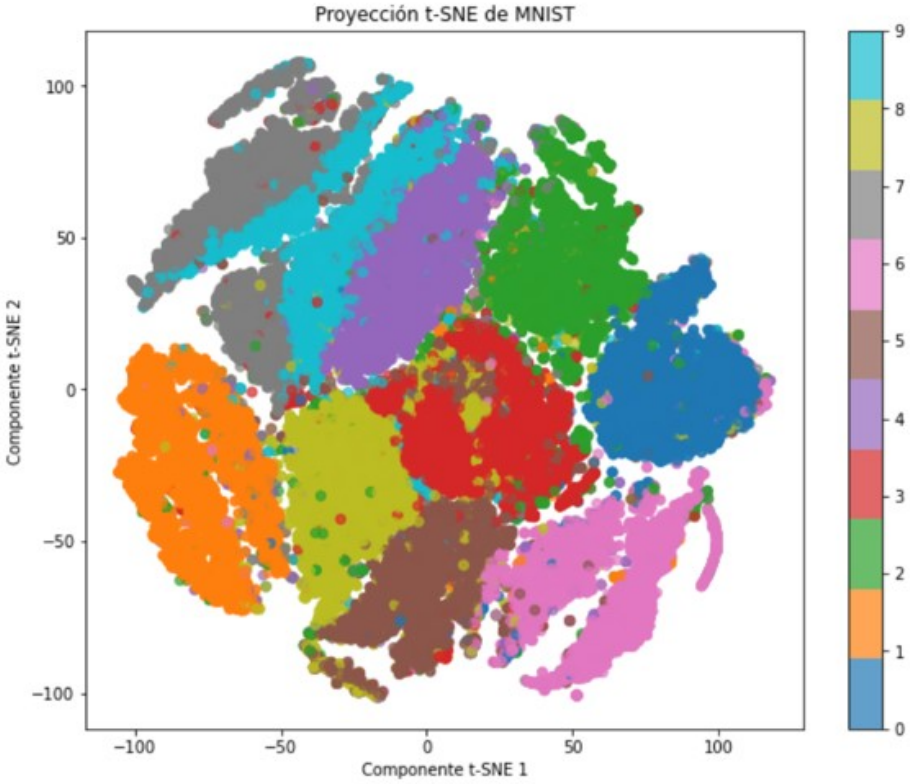
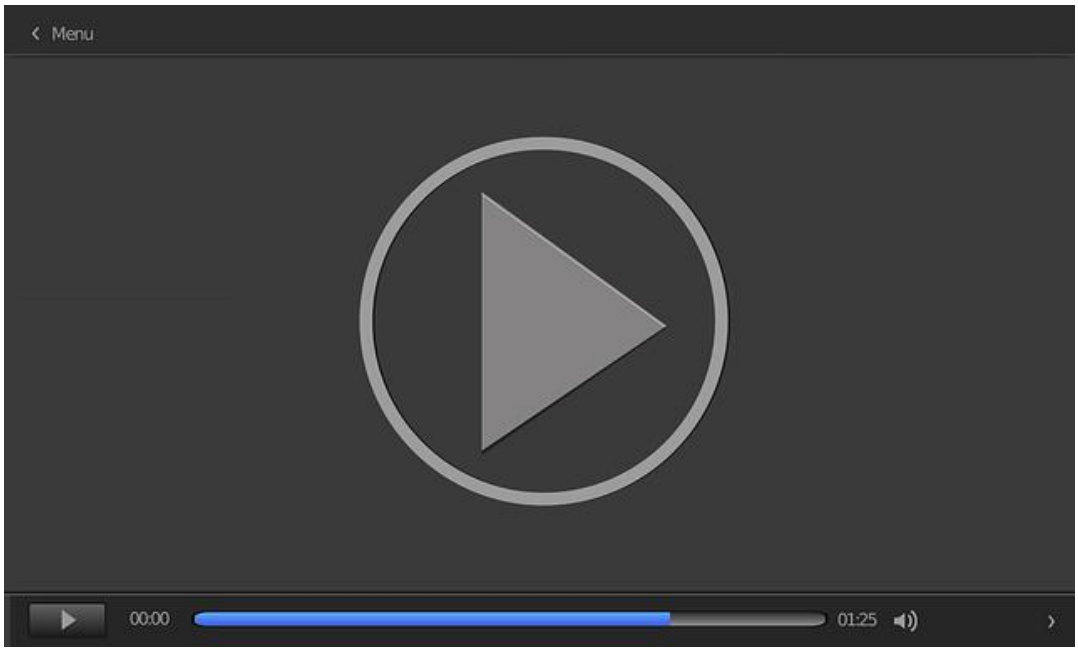


Figura 2. t-SNE sobre el conjunto de datos MNIST. Fuente: elaboración propia.

A continuación, veremos el vídeo ***t-SNE utilizando Python.***



Ideas clave

Accede al vídeo:

https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=0d3c083c-0cb4-4bdd-9d37-b1bf00149af2

7.5. Multi-Dimensional Scaling (MDS)

El **escalado multidimensional** (MDS) es una herramienta estadística que ayuda a descubrir conexiones entre puntos en un espacio de dimensiones más bajo que las dimensiones originales. Utiliza técnicas de análisis de datos de similitud o disimilitud canónica.

El objetivo principal es mantener las proximidades originales entre conjuntos de datos; los datos más similares se ubican muy cerca unos de otros, mientras que los datos más separados se colocan separados en el espacio de dimensión menor.

Utiliza algoritmos de optimización avanzados para minimizar las discrepancias entre las distancias originales en el espacio de altas dimensiones y el espacio de bajas dimensiones. Para poder lograrlo, el algoritmo ajusta las posiciones de los puntos para que las distancias en las dimensiones más bajas sean lo más cercanas posibles a los datos originales.

Sus aplicaciones se han dado especialmente en áreas como la psicología, sociología, marketing, geografía y biología.

Existen **tres tipos** de escala multidimensional:

- **Algoritmo clásico:** toma una matriz de entrada que representa diferencias entre pares de elementos y produce una matriz de coordenadas que minimiza la dimensionalidad.
- **Escalamiento multidimensional métrico** que generaliza el procedimiento de optimización a varias funciones de pérdida y matrices de entrada con distancias y pesos desconocidos. Minimiza una función de coste llamada «stress».
- **Escalado multidimensional no métrico:** encuentra la relación monótona no paramétrica entre las diferencias y distancias euclidianas entre elementos, junto con

la ubicación de cada elemento en el espacio de baja dimensionalidad. Se define una función de «stress» a optimizar y se considera que es monótonamente creciente.

Veamos un ejemplo de implementación en Python:

```
#Cargamos la librería de Isomap
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.datasets import fetch_openml
from sklearn.preprocessing import StandardScaler
from sklearn.manifold import MDS

# Descargar el conjunto de datos MNIST
mnist = fetch_openml('mnist_784', version=1)
X = mnist.data
y = mnist.target.astype(int)

# Usar solo una muestra del conjunto de datos para reducir la carga de
memoria
sample_size = 10000 # Puedes ajustar este número según sea necesario
X_sample = X[:sample_size]
y_sample = y[:sample_size]

# Normalizar los datos
scaler = StandardScaler()
X_sample_scaled = scaler.fit_transform(X_sample)

# Reducir la dimensionalidad a 50 componentes usando PCA
pca = PCA(n_components=50)
X_sample_pca = pca.fit_transform(X_sample_scaled)

# Reducir la dimensionalidad a 2 componentes usando MDS
mds = MDS(n_components=2, random_state=42)
X_sample_mds = mds.fit_transform(X_sample_pca)

# Crear una gráfica de los datos proyectados
plt.figure(figsize=(10, 8))
scatter = plt.scatter(X_sample_mds[:, 0], X_sample_mds[:, 1], c=y_sample,
cmap='tab10', alpha=0.7)
plt.colorbar(scatter)
plt.xlabel('Componente MDS 1')
plt.ylabel('Componente MDS 2')
plt.title('Proyección MDS de una muestra de MNIST')
plt.show()
```

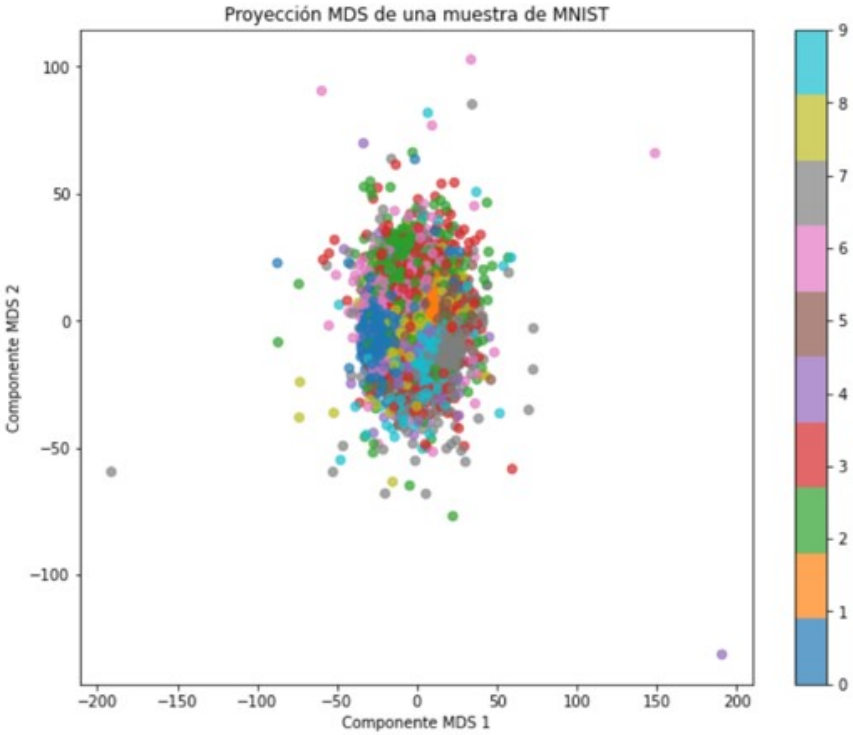


Figura 3. MDS sobre el conjunto de datos MNIST. Fuente: elaboración propia.

7.6. Isometric mapping (Isomap)

Mapeo isométrico o **Isomap** combina varios algoritmos que le permiten utilizar una forma no lineal para reducir las dimensiones preservando las estructuras locales.

Algoritmo

- Utiliza KNN para encontrar los K vecinos más cercanos de cada uno de los puntos. K es un número elegido arbitrariamente.
- Una vez se han encontrado los vecinos, se construye un gráfico de vecindad donde los puntos están conectados entre sí si son vecinos. Los demás puntos permanecen desconectados.
- Se calcula el camino más corto entre cada par de puntos, utilizando el algoritmo Floyd-Warshall o también el de Dijkstra. Este paso halla la distancia geodésica entre puntos. Este es el secreto de la capacidad que tiene el algoritmo para reducir la dimensionalidad no lineal y al mismo tiempo equilibrar la relación entre las estructuras locales y globales.
- Utiliza el escalado multidimensional (MDS) para calcular la incrustación de dimensiones inferiores. Como se conocen las distancias entre puntos, MDS coloca cada objeto en el espacio N-dimensional de tal manera que las distancias entre puntos se conservan bastante bien.

Veamos cómo implementarlo en Python utilizando el mismo conjunto de datos que venimos trabajando en los ejemplos anteriores, el de los dígitos escritos a mano-MNIST.

```
#Cargamos la librería de Isomap
from sklearn.manifold import Isomap

# Reducir la dimensionalidad a 2 componentes usando Isomap
isomap = Isomap(n_components=2)
X_isomap = isomap.fit_transform(X_scaled)
```

```
# Crear una gráfica de los datos proyectados
plt.figure(figsize=(10, 8))
scatter = plt.scatter(X_isomap[:, 0], X_isomap[:, 1], c=y, cmap='tab10',
alpha=0.7)
plt.colorbar(scatter)
plt.xlabel('Componente Isomap 1')
plt.ylabel('Componente Isomap 2')
plt.title('Proyección Isomap de MNIST')
plt.show()
```

El código anterior tiene un problema requiere gran cantidad de memoria. Para poder ejecutarlo en Python es necesario trabajar con una muestra más pequeña.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.datasets import fetch_openml
from sklearn.preprocessing import StandardScaler
from sklearn.manifold import Isomap

# Descargar el conjunto de datos MNIST
mnist = fetch_openml('mnist_784', version=1)
X = mnist.data
y = mnist.target.astype(int)

# Usar solo una muestra del conjunto de datos para reducir la carga de
memoria
sample_size = 10000 # Se puede ajustar este número
X_sample = X[:sample_size]
y_sample = y[:sample_size]

# Normalizar los datos
scaler = StandardScaler()
X_sample_scaled = scaler.fit_transform(X_sample)

# Reducir la dimensionalidad a 2 componentes usando Isomap
isomap = Isomap(n_components=2)
X_isomap = isomap.fit_transform(X_sample_scaled)

# Crear una gráfica de los datos proyectados
plt.figure(figsize=(10, 8))
scatter = plt.scatter(X_isomap[:, 0], X_isomap[:, 1], c=y_sample,
cmap='tab10', alpha=0.7)
plt.colorbar(scatter)
plt.xlabel('Componente Isomap 1')
```

```
plt.ylabel('Componente Isomap 2')
plt.title('Proyección Isomap de una muestra de MNIST')
plt.show()
```

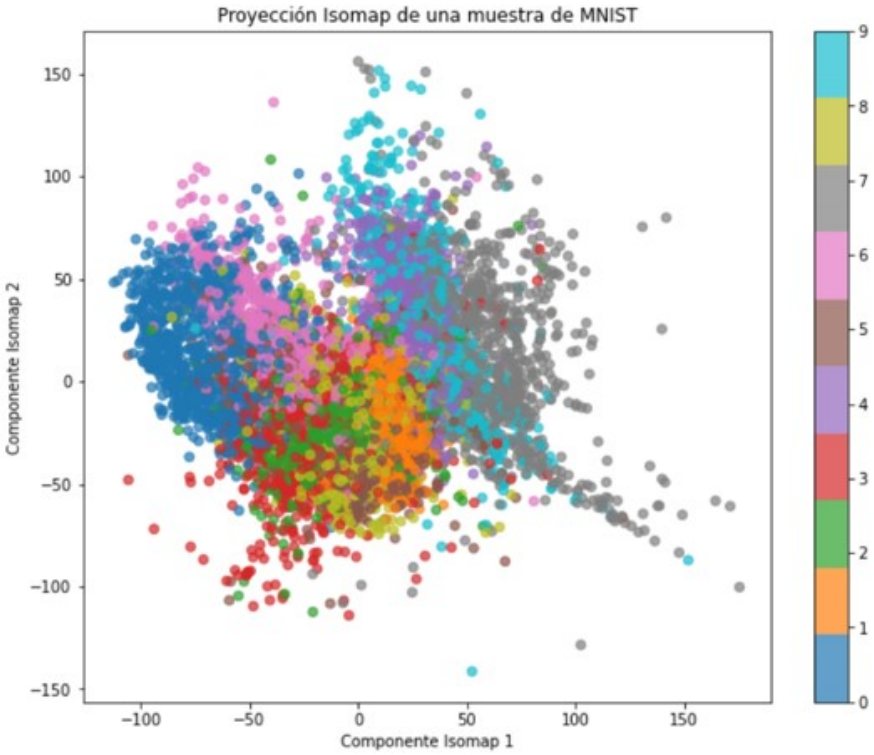


Figura 4. Isomap sobre el conjunto de datos MNIST. Fuente: elaboración propia.

Isoamp también se utiliza como parte del análisis de procesamiento del lenguaje natural para reducir la dimensionalidad de los datos de texto antes de entrenar un modelo de clasificación.

7.7. Tabla comparativa entre las diferentes técnicas de reducción de dimensiones

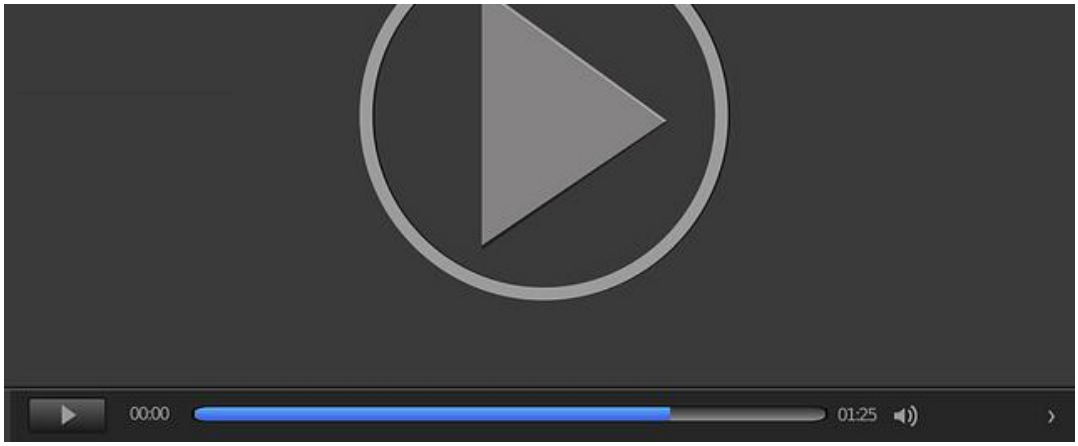
Veamos las diferencias y características de las técnicas que hemos visto en este tema a través de la Tabla 1:

Técnica	Objetivo	Visualización	Aplicabilidad	Interpretación
MDS	Conservar las distancias o disimilitudes originales por pares.	Proporciona visualizaciones intuitivas de similitudes y diferencias.	Adecuado para datos con diferencias o similitudes conocidas, aplicable en varios dominios	Enfatiza la preservación de las relaciones, facilitando la interpretación cualitativa.
PCA	Maximiza la variación a lo largo de ejes ortogonales.	Eficiente para capturar la estructura global, pero es posible que no preserve las distancias por pares.	Adecuado para transformaciones de datos lineales. A menudo utilizado para extracción de características.	Se centra en capturar la varianza. Útil para la reducción de dimensionalidad en datos de alta dimensión.
t-SNE	Enfatiza las similitudes locales al asignar datos de alta dimensión a un espacio de baja dimensión.	Crea grupos densos para puntos de datos similares, pero las distancias no se conservan.	Efectivo para visualizar datos de alta dimensión con estructuras complejas.	Se utiliza principalmente para visualización, menos énfasis en preservar las relaciones globales.
Isomap	Preserva distancias geodésicas para descubrir la estructura múltiple subyacente.	Captura relaciones no lineales. Útil para datos con dimensionalidad intrínseca.	Efectivo para datos con estructuras no lineales, como imágenes o redes de sensores.	Se centra en descubrir la estructura intrínseca. Útil para comprender relaciones no lineales.

Tabla 1. Características de las técnicas de reducción de dimensiones. Fuente: basado en GeeksforGeeks, 2024.

A continuación, veremos el vídeo ***Comparación de técnicas de reducción de dimensionalidad en Python.***

Ideas clave



Accede al vídeo:

https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=e84afe29-0480-4b59-aa21-b1bf000dd216

7.8. Cuaderno de ejercicios

Ejercicio 1. Conceptos básicos

Explica brevemente en qué se diferencian las técnicas de PCA y t-SNE en términos de sus objetivos y aplicaciones.

Solución

PCA (Principal Component Analysis) busca maximizar la variación de los datos a lo largo de ejes ortogonales y es particularmente eficaz para transformaciones lineales, capturando la estructura global. t-SNE, por otro lado, se centra en preservar las relaciones locales entre puntos de datos en un espacio de menor dimensionalidad, siendo más adecuado para la visualización de datos no lineales y complejos, como en el caso del conjunto de datos MNIST.

Ejercicio 2. Implementación t-SNE

Describe los pasos básicos necesarios para implementar t-SNE en Python utilizando un conjunto de datos de ejemplo.

Solución

Para implementar t-SNE en Python, primero se importan las librerías necesarias (numpy, matplotlib, sklearn). Luego se descarga y normaliza el conjunto de datos. Posteriormente, se aplica t-SNE con `TSNE(n_components=2 o 3)` y se transforma el conjunto de datos. Finalmente, se visualizan los datos proyectados en un gráfico 2D usando `plt.scatter()`.

Ejercicio 3. Aplicaciones de MDS

Menciona dos aplicaciones del escalado multidimensional (MDS) y explica brevemente cómo esta técnica es útil en esos contextos.

Solución

MDS se utiliza en psicología para descubrir patrones en respuestas de encuestas y en marketing para analizar preferencias de consumidores. En ambos casos, MDS ayuda a visualizar y entender las similitudes y diferencias entre elementos (como individuos o productos) en un espacio de menor dimensionalidad, manteniendo las distancias relativas entre los datos.

Ejercicio 4. Ventajas y desventajas de Isomap

Menciona dos ventajas y dos desventajas del uso de Isomap en la reducción de dimensionalidad.

Solución

Dos ventajas de Isomap son su capacidad para preservar relaciones no lineales y descubrir la estructura intrínseca de los datos, lo que es útil para datos complejos como imágenes o redes de sensores. Dos desventajas son su alta demanda computacional y la sensibilidad a la elección del número de vecinos más cercanos (K), lo cual puede afectar significativamente los resultados.

7.9. Referencias bibliográficas

Abdi, H. (2007). Metric multidimensional scaling (MDS): analyzing distance matrices. *Encyclopedia of measurement and statistics,* 1-13.

Awan, A. (2023, march). *Introduction to t-SNE.* Datacamp. https://www.datacamp.com/tutorial/introduction-t-sne?dc_referrer=https%3A%2F%2Fwww.google.com%2F

Belkina, A. C., Ciccolella, C. O., Anno, R., Halpert, R., Spidlen, J. y Snyder-Cappione, J. E. (2019). Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. *Nature communications, 10*(1), 5415.

Cheng, Y., Wang, X. y Xia, Y. (2021). Supervised t-distributed stochastic neighbor embedding for data visualization and classification. *INFORMS journal on computing, 33*(2), 566-585.

GeeksforGeeks. (2024, Mayo 19). *What is Multidimensional Scaling?* https://www.geeksforgeeks.org/what-is-multidimensional-scaling/

Sachinsoni. (2024, febrero 11). *Mastering t-SNE (t-distributed stochastic neighbor embedding).* Medium. https://medium.com/@sachinsoni600517/mastering-t-sne-t-distributed-stochastic-neighbor-embedding-0e365ee898ea

Soni, J., Prabakar, N. y Upadhyay, H. (2020). Visualizing high-dimensional data using t-distributed stochastic neighbor embedding algorithm. *Principles of data science,* 189-206.

Tenenbaum, J. B., de Silva, V. y Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science, 290*(5500), 2319-2323. https://doi.org/10.1126/science.290.5500.2319

Ideas clave

Reducción de dimensiones: t-SNE

Organización de datos. (2020, junio 6). *Reducción de dimensiones: t-SNE* [Vídeo]. YouTube. https://www.youtube.com/watch?v=NtvFoZORPB8&t=620s

Explica de forma detallada el paso a paso del algoritmo con base en un artículo de investigación.

Entendiendo cómo trabaja la reducción de dimensionalidad

Wang, Y., Huang, H., Rudin, C. y Shaposhnik, Y. (2021). Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMap, and PaCMAP for Data Visualization. *Journal of Machine Learning Research, 22,* 1-73. https://www.jmlr.org/papers/volume22/20-1061/20-1061.pdf

El objetivo del *paper* es que el lector comprenda los aspectos más importantes de algunas técnicas de reducción de dimensiones. A partir de las técnicas estudiadas, los autores diseñan un nuevo algoritmo denominado proyección de aproximación de colector controlado por pares (PaMAP), que preserva tanto la estructura local como la global.

Test

1. ¿Cuál es el objetivo principal del t-SNE?

A. Maximizar la variación global en los datos.

B. Preservar las relaciones locales entre puntos de datos.

C. Minimizar el uso de recursos computacionales.

D. Garantizar la linealidad de los datos.

2. ¿Cuál de las siguientes técnicas es más adecuada para datos lineales?

A. t-SNE.

B. PCA.

C. MDS.

D. Isomap.

3. ¿Cuál es una de las desventajas del t-SNE?

A. Es muy eficiente computacionalmente.

B. Preserva la estructura global de los datos.

C. Puede quedarse atascado en mínimos locales.

D. Es muy sensible a los hiperparámetros.

4. ¿Qué técnica utiliza distribuciones gaussianas para calcular similitudes en alta dimensionalidad?

A. PCA.

B. MDS.

C. t-SNE.

D. Isomap.

5. ¿Qué significa MDS en el contexto de la reducción de dimensionalidad?

A. Minimum Dimension Search.

B. Multi-Dimensional Scaling.

C. Maximum Distance Sorting.

D. Multi-Data Scaling.

6. ¿Cuál de las siguientes técnicas es especialmente útil para datos no lineales?

A. Isomap.

B. t-SNE

C. MDS.

D. Todas las anteriores.

7. ¿Qué técnica de reducción de dimensionalidad utiliza la divergencia de Kullback-Leibler?

A. PCA.

B. MDS.

C. t-SNE.

D. Isomap.

8. ¿Cuál de las siguientes aplicaciones es adecuada para t-SNE?

A. Reducción de dimensionalidad de datos lineales.

B. Visualización de datos complejos y no lineales.

C. Análisis de encuestas de preferencias.

D. Cálculo de distancias euclidianas entre puntos.

9. ¿Qué técnica utiliza el algoritmo de Floyd-Warshall para calcular el camino más corto entre pares de puntos?

A. PCA.

B. MDS.

C. t-SNE.

D. Isomap.

10. ¿Cuál de las siguientes afirmaciones sobre las técnicas de reducción de dimensionalidad es correcta?

A. MDS es más adecuado que t-SNE para datos con estructuras no lineales.

B. PCA puede preservar mejor las relaciones locales entre los datos que t-SNE.

C. t-SNE es computacionalmente más eficiente que PCA para grandes conjuntos de datos.

D. Isomap utiliza el cálculo de distancias geodésicas para preservar las relaciones no lineales en los datos.