

UMAP y t-SNE

Lucca Frachelle

diciembre 2025

Introducción

En la era digital, el crecimiento exponencial en volumen y complejidad de los datos ha consolidado a la **reducción de dimensionalidad** como una técnica indispensable en el análisis de datos. Su objetivo es transformar datos de un espacio de alta dimensión a una representación de menor dimensión, preservando al máximo la información relevante y la estructura inherente. Esta tarea es crucial no solo para la visualización y la comprensión de patrones complejos, sino también para mejorar la eficiencia computacional de algoritmos de aprendizaje automático y para mitigar la “maldición de la dimensionalidad”.

Enfoques para la Reducción de Dimensionalidad

Existen dos enfoques principales para la reducción de dimensionalidad:

1. Factorización de Matrices:

- **PCA (Análisis de Componentes Principales):** Descompone la matriz de datos en componentes principales que maximizan la varianza.
- **SVD (Descomposición en Valores Singulares):** Factoriza la matriz en un producto de matrices ortogonales y una diagonal.
- **NMF (Non-negative Matrix Factorization):** Realiza una factorización con la restricción de que todos los elementos de las matrices deben ser no negativos.
- Características:
 - Basado en álgebra lineal.
 - Optimiza criterios globales como la varianza o el error de reconstrucción.
 - Computacionalmente eficiente.
 - Menos sensible al ruido.

2. Grafos de Vecindad:

- **t-SNE:** Construye un grafo de similitud basado en distribuciones de probabilidad.
- **UMAP:** Construye un grafo de vecindad difuso basado en fundamentos topológicos.
- **LLE (Local Linear Embedding):** Preserva las relaciones de vecindad locales asumiendo que la variedad es localmente lineal.
- Características:
 - Basado en teoría de grafos y topología.
 - Preserva la estructura local de los datos.
 - Especialmente útil para la visualización y el descubrimiento de *clusters*.
 - Puede ser computacionalmente más costoso.

Graph-based dimensionality reduction

La idea general de graph-based dimensionality reduction es la siguiente:

1. Se construye una métrica de similitud entre los datos en el espacio de alta dimensión.
2. Se busca reconstruir los datos en un espacio de baja dimensión, respetando la similitud entre los datos. Esto se hace mediante un algoritmo iterativo que busca optimizar la distancia entre los

datos en el espacio de baja dimensión para que coincida con la métrica de similitud en el espacio de alta dimensión.

Fundamentos Teóricos

La Función Softmax

La función softmax es una herramienta matemática fundamental, ampliamente utilizada en el campo del aprendizaje automático, especialmente en redes neuronales. Su propósito es transformar un vector de números reales en una distribución de probabilidad, es decir, un vector de valores entre 0 y 1 cuya suma es 1.

La fórmula de Softmax para un vector $z = [z_1, z_2, \dots, z_K]$ es:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

donde:

- z_i es el i -ésimo elemento del vector de entrada.
- K es el número de elementos en el vector de entrada.

En el contexto de t-SNE, la función softmax se utiliza para convertir las distancias euclidianas en probabilidades de similitud entre puntos.

Divergencia de Kullback-Leibler (KL)

La Divergencia de Kullback-Leibler (KL) es una medida de la diferencia entre dos distribuciones de probabilidad. Aunque a menudo se usa para cuantificar la “distancia” entre distribuciones, no es una métrica en el sentido matemático estricto, ya que no es simétrica ($KL(P||Q) \neq KL(Q||P)$) y no satisface la desigualdad triangular.

En t-SNE, la divergencia KL es la **función de costo** que el algoritmo busca minimizar. Mide la discrepancia entre las distribuciones de probabilidad de similitud en el espacio de alta dimensión (P) y en el de baja dimensión (Q). El objetivo es ajustar Q para que sea lo más parecida posible a P .

La fórmula de la divergencia KL de una distribución Q con respecto a una distribución de referencia P es:

$$KL(P||Q) = \sum_i P(i) \log \left(\frac{P(i)}{Q(i)} \right)$$

donde:

- $P(i)$ es la probabilidad del evento i en la distribución P .
- $Q(i)$ es la probabilidad del evento i en la distribución Q .
- La suma se realiza sobre todos los eventos posibles.

¿Cómo funciona?

La divergencia KL puede interpretarse como la esperanza del logaritmo del cociente de probabilidades entre las dos distribuciones, donde la esperanza se toma sobre la distribución de referencia P .

- **Si $P(i) \approx Q(i)$:** El cociente $\frac{P(i)}{Q(i)}$ estará cerca de 1, y $\log(1) = 0$. La contribución a la divergencia será mínima.
- **Si $P(i)$ es grande y $Q(i)$ es pequeña:** El cociente será grande, y su logaritmo será un número positivo grande. Esto penaliza fuertemente los casos en que Q subestima la probabilidad de un evento importante en P .

- **Si $P(i)$ es pequeña y $Q(i)$ es grande:** El cociente será pequeño, y su logaritmo será un número negativo. Sin embargo, al estar ponderado por un $P(i)$ pequeño, la contribución total al costo será también pequeña.
- **Si $P(i) \neq 0$ y $Q(i) = 0$:** La divergencia se vuelve infinita. Esta es una propiedad crucial en t-SNE, ya que impone una penalización infinita si la representación de baja dimensión asigna una probabilidad cero a una relación que era posible en el espacio original.

Observar que KL no es simétrica, es decir, $KL(P||Q) \neq KL(Q||P)$.

t-SNE: Teoría y Algoritmo

De SNE a t-SNE: Evolución del Algoritmo

Stochastic Neighbor Embedding (SNE) SNE fue el predecesor de t-SNE, desarrollado por Hinton y Roweis en 2002. El algoritmo se basa en convertir distancias euclidianas entre puntos en probabilidades de similitud. El proceso es el siguiente:

La idea de SNE es poder representar un espacio de alta dimensionalidad en un espacio de baja dimensionalidad de manera que las relaciones de vecindad se preserven. Para ello se usa la función softmax para convertir las distancias euclidianas en probabilidades de similitud. El objetivo es que las probabilidades de similitud en el espacio de baja dimensionalidad sean similares a las del espacio de alta dimensionalidad.

1. **Probabilidades en el espacio de alta dimensión:** Para cada par de puntos x_i y x_j en el espacio de alta dimensión, se calcula la probabilidad condicional $p_{j|i}$ de que x_i elija a x_j como su vecino:

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

donde σ_i es la varianza de la distribución gaussiana centrada en x_i . Esta varianza se ajusta para cada punto i de manera que la distribución de probabilidades P_i tenga una perplejidad fija.

2. **Probabilidades en el espacio de baja dimensión:** De manera similar, para los puntos y_i y y_j en el espacio de baja dimensión:

$$q_{j|i} = \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)}$$

Aquí se usa una distribución gaussiana con varianza fija (1/2) para simplificar.

3. **Función de costo:** SNE minimiza la suma de las divergencias KL entre las distribuciones P_i y Q_i :

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

La función de costo es lo que el algoritmo intenta minimizar, con la idea de que las probabilidades en el espacio de baja dimensionalidad sean similares a las del espacio de alta dimensionalidad.

Limitaciones de SNE SNE presentaba dos problemas principales:

1. **Asimetría en las probabilidades:**
 - Las probabilidades condicionales $p_{j|i}$ y $p_{i|j}$ no son iguales
 - Esto puede llevar a resultados inconsistentes
 - La función de costo es asimétrica respecto a P y Q

2. El problema de hacinamiento (crowding problem):

- En el espacio de baja dimensión, hay menos “espacio” disponible para que los puntos se separen y las probabilidades de vecindad sean similares a las del espacio de alta dimensionalidad.
- Los puntos tienden a aglomerarse en el centro
- La distribución gaussiana en baja dimensión no maneja bien este problema
- Las colas de la gaussiana decaen muy rápido (e^{-x^2})

La Solución: t-SNE t-SNE resuelve estos problemas mediante :

1. Probabilidades conjuntas simétricas:

- Reemplaza las probabilidades condicionales por probabilidades conjuntas
- $p_{ij} = \frac{p_{ji} + p_{ij}}{2N}$
- Esto asegura que $p_{ij} = p_{ji}$
- La función de costo se vuelve simétrica

2. Distribución t de Student en baja dimensión:

- Reemplaza la distribución gaussiana por una t de Student con un grado de libertad
- La fórmula para q_{ij} se convierte en:

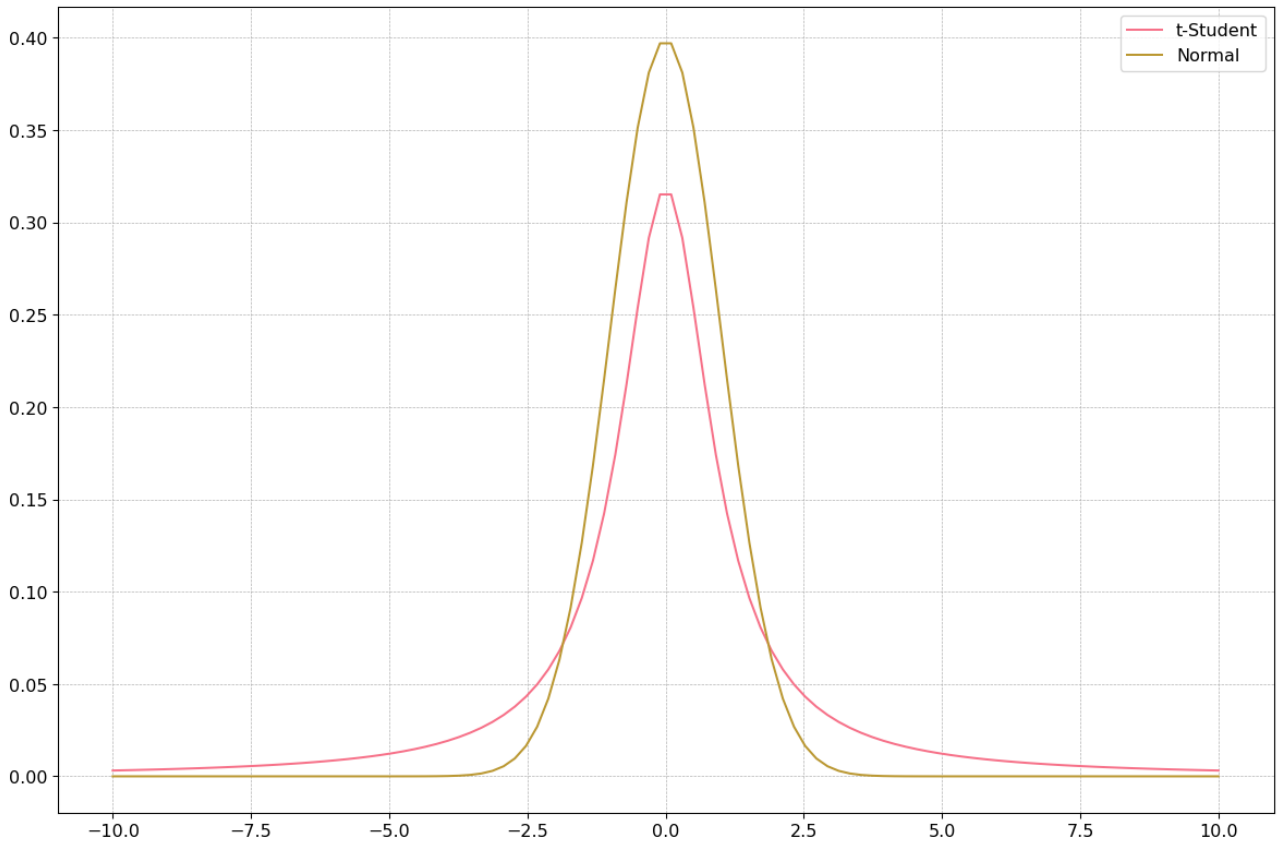
$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}}$$

La distribución t de Student tiene “colas pesadas” (heavy tails), lo que significa que: - Asigna una probabilidad significativa a valores lejos de la media - Permite que puntos moderadamente separados en alta dimensión se mapeen a distancias mayores en baja dimensión - Crea más “espacio” en el centro del mapa - Evita la aglomeración excesiva de puntos

La fórmula de la distribución t de Student con un grado de libertad es:

$$f(x) = \frac{1}{\pi(1 + x^2)}$$

Esta función tiene colas que decaen como $1/x^2$, lo que es más lento que la distribución gaussiana que decae como e^{-x^2} . Esta propiedad es clave para resolver el problema de hacinamiento.



Función de Costo de t-SNE La función de costo de t-SNE es una divergencia KL entre las probabilidades conjuntas P y Q :

$$C = \text{KL}(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Gradiente de la Función de Costo El gradiente de la función de costo con respecto a y_i es:

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}$$

Este gradiente tiene una interpretación física intuitiva: - Si $p_{ij} > q_{ij}$: Los puntos i y j están más cerca en alta dimensión que en baja dimensión * Se crea una fuerza atractiva que los acerca - Si $p_{ij} < q_{ij}$: Los puntos i y j están más lejos en alta dimensión que en baja dimensión * Se crea una fuerza repulsiva que los aleja * La fuerza se modera por el término $(1 + \|y_i - y_j\|^2)^{-1}$

El término $(1 + \|y_i - y_j\|^2)^{-1}$ es crucial porque: - Modera la fuerza de repulsión entre puntos ya separados - Permite que los puntos se separen más fácilmente - Contribuye a la formación de clusters bien definidos

Algoritmo de Optimización

El algoritmo de t-SNE utiliza un algoritmo de descenso de gradiente para minimizar la función de costo. Se aplican técnicas como:

- **Momentum:** Para acelerar la convergencia y evitar mínimos locales. La actualización de las coordenadas y_i en cada iteración t es:

$$Y^{(t)} = Y^{(t-1)} + \eta \frac{\partial C}{\partial Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)})$$

donde η es la tasa de aprendizaje y $\alpha(t)$ es el término de momentum.

- **Ajuste del learning rate:** Se suele usar un learning rate que se incrementa en las primeras iteraciones y luego se mantiene constante.
- **Inicialización:** Los puntos y_i se inicializan aleatoriamente de una distribución normal con una pequeña varianza.

Algoritmo de t-SNE

El algoritmo de t-SNE se puede describir de la siguiente manera:

Algoritmo 1: Versión Simple de t-Distributed Stochastic Neighbor Embedding.

Datos: conjunto de datos $X = \{x_1, x_2, \dots, x_n\}$, parámetros de la función de costo: perplejidad $Perp$, parámetros de optimización: número de iteraciones T , tasa de aprendizaje η , momento $\alpha(t)$.

Resultado: representación de baja dimensión $Y^{(T)} = \{y_1, y_2, \dots, y_n\}$.

Inicio:

1. calcular afinidades por pares $p_{j|i}$ con perplejidad $Perp$ (usando Ecuación 1)
2. establecer $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
3. muestrear solución inicial $Y^{(0)} = \{y_1, y_2, \dots, y_n\}$ desde $\mathcal{N}(0, 10^{-4}I)$
4. para $t = 1$ hasta T hacer
 - a. calcular afinidades de baja dimensión q_{ij} (usando Ecuación 4)
 - b. calcular gradiente $\frac{\partial C}{\partial Y}$ (usando Ecuación 5)
 - c. establecer $Y^{(t)} = Y^{(t-1)} + \eta \frac{\partial C}{\partial Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)})$
5. Fin

La actualización de los puntos en cada iteración t se realiza mediante la ecuación:

$$Y^{(t)} = Y^{(t-1)} + \eta \frac{\partial C}{\partial Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)})$$

Donde:

1. $Y^{(t)}$: Es la nueva posición de los puntos en la iteración t
2. $Y^{(t-1)}$: Es la posición actual de los puntos
3. η : Es la tasa de aprendizaje (learning rate)
4. $\frac{\partial C}{\partial Y}$: Es el gradiente de la función de costo
5. $\alpha(t)$: Es el término de momento
6. $(Y^{(t-1)} - Y^{(t-2)})$: Es el cambio en la posición del paso anterior

El movimiento de los puntos está determinado por dos componentes principales:

1. **Término del gradiente** ($\eta \frac{\partial C}{\partial Y}$):
 - Mueve los puntos en la dirección que minimiza la divergencia KL
 - Los puntos se mueven para preservar la estructura de vecindad
 - Si dos puntos están cerca en el espacio original, tenderán a moverse uno hacia el otro
 - Si dos puntos están lejos en el espacio original, tenderán a alejarse
2. **Término de momento** ($\alpha(t)(Y^{(t-1)} - Y^{(t-2)})$):
 - Ayuda a evitar mínimos locales
 - Mantiene cierta “inercia” del movimiento anterior
 - Ayuda a que la optimización sea más estable
 - El factor $\alpha(t)$ controla cuánto del movimiento anterior se mantiene

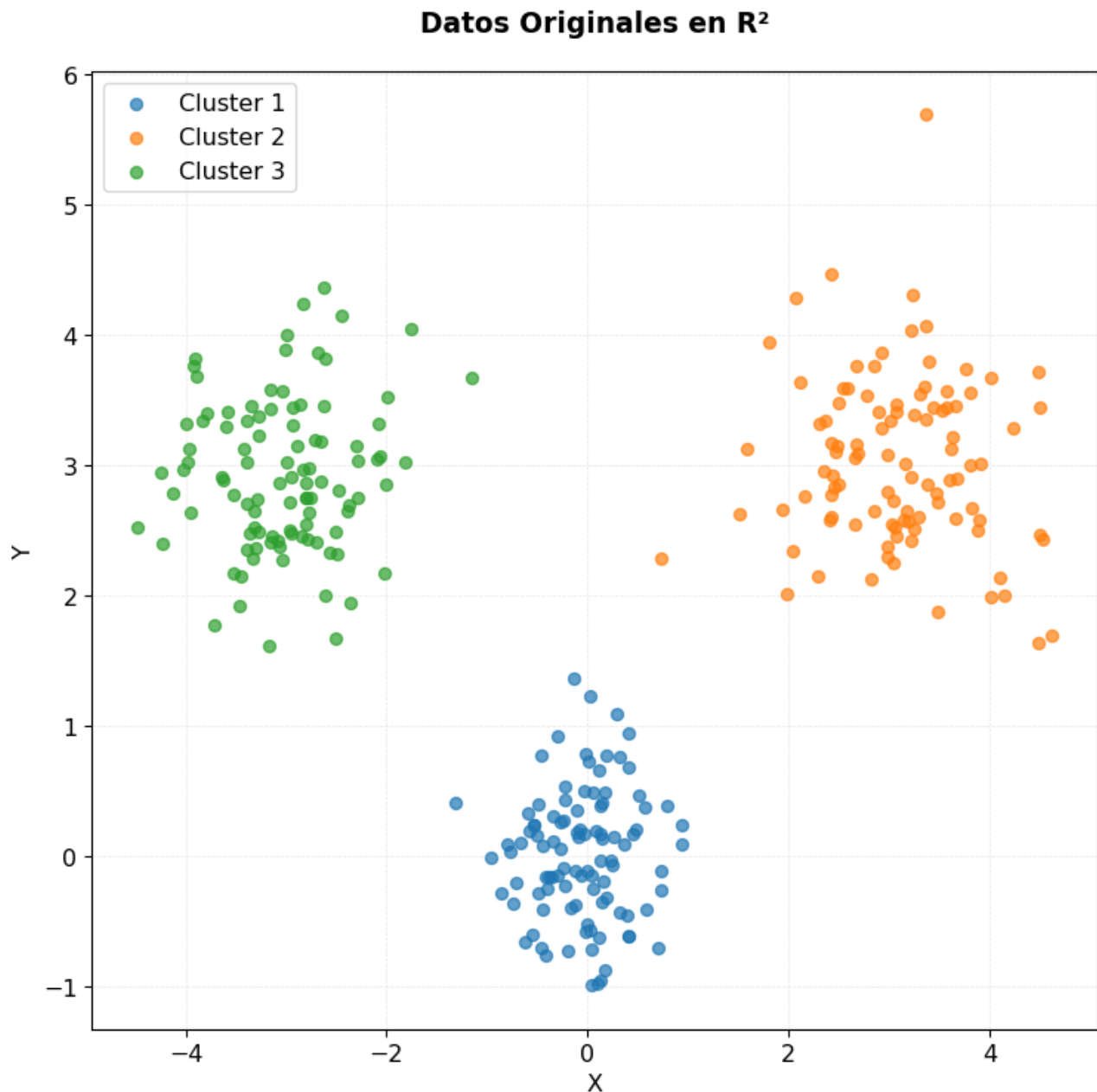
Variante con Inicialización PCA Una variante común del algoritmo t-SNE utiliza PCA (Análisis de Componentes Principales) para inicializar los puntos en lugar de usar una distribución normal aleatoria. Esta variante tiene las siguientes características:

1. **Inicialización con PCA:**

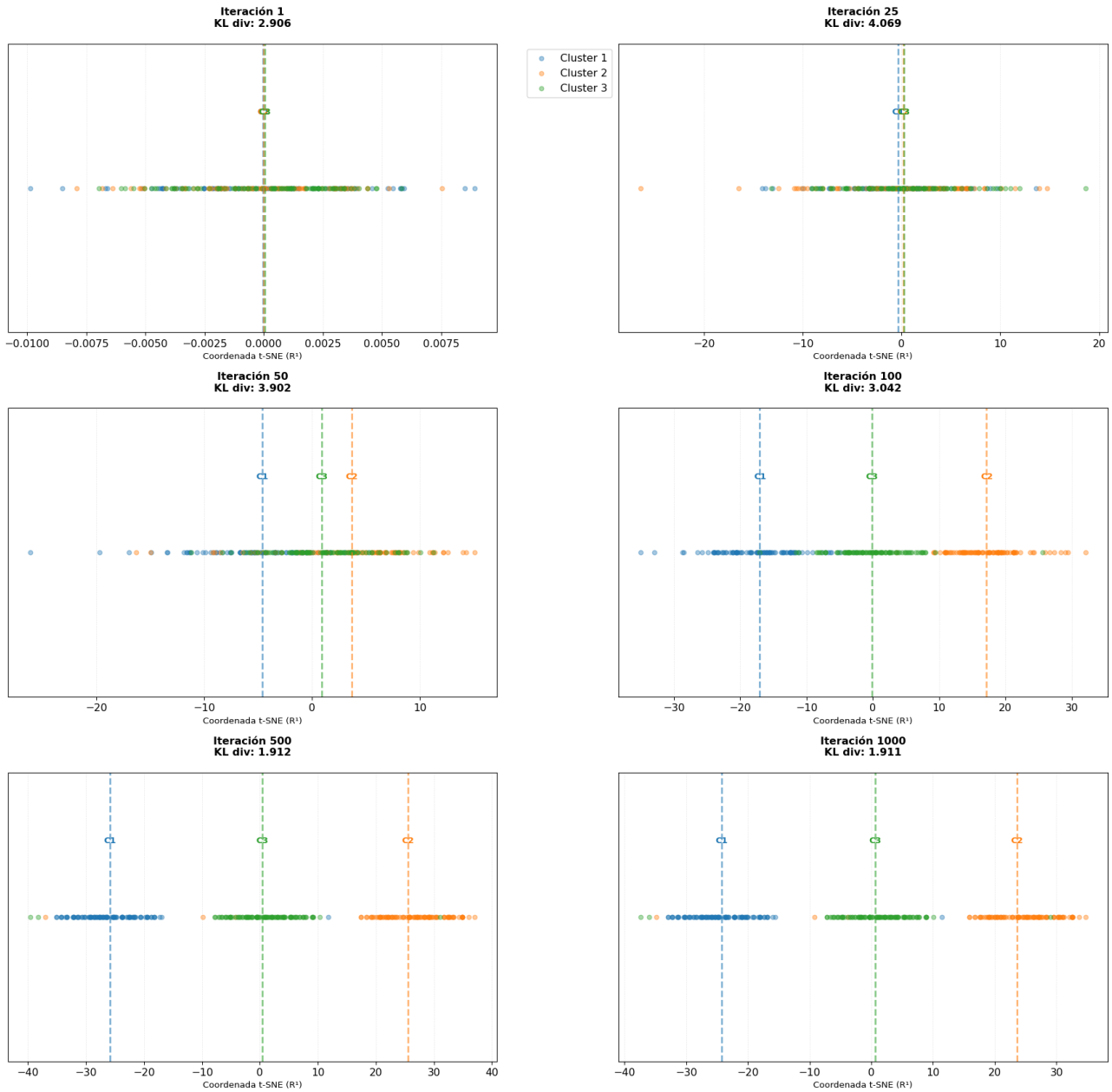
- En lugar de $Y^{(0)} \sim \mathcal{N}(0, 10^{-4}I)$, se usa $Y^{(0)} = \text{PCA}(X)$
- Esto proporciona una mejor posición inicial que preserva la estructura global de los datos
- Puede acelerar la convergencia del algoritmo
- Ayuda a evitar mínimos locales pobres

Ejemplo del algoritmo

A continuación, implementaremos t-SNE paso a paso para proyectar datos de $\mathbb{R}^2$ a $\mathbb{R}^1$. Generaremos tres distribuciones normales con diferentes medias y varianzas, y seguiremos el proceso de optimización en diferentes iteraciones.



Evolución de t-SNE Personalizado (Iteraciones Clave)



Este ejemplo muestra:

- **Generación de datos:**
 - Tres clusters gaussianos en $\mathbb{R}^2$
 - Cada cluster tiene 100 puntos
 - Diferentes medias y varianzas para cada cluster

Extensiones y Variaciones

El artículo también discute extensiones como:

- **t-SNE para conjuntos de datos grandes:** Se propone una técnica de random walks en grafos de vecindad para manejar grandes volúmenes de datos, donde solo un subconjunto de puntos se visualiza directamente, pero la estructura global influye en el embedding.
- **t-SNE con costos de incrustación tempranos (early exaggeration):** Multiplicar los p_{ij} por un factor (ej. 4 o 12) en las primeras iteraciones para crear clústeres más compactos y evitar

que se formen “mini-clústeres” que no se pueden separar más tarde.

$$p'_{ij} = p_{ij} \times \text{exaggeration_factor}$$

para las primeras T iteraciones.

UMAP: Teoría y Algoritmo

UMAP (Uniform Manifold Approximation and Projection) es una técnica de reducción de dimensionalidad no lineal desarrollada por Leland McInnes, John Healy y James Melville. Su propósito central es **proyectar datos de alta dimensión en un espacio de menor dimensión** (comúnmente 2D o 3D) para **visualización** o como un paso de **preprocesamiento** en pipelines de aprendizaje automático.

A diferencia de técnicas como t-SNE, que se basan en probabilidades de similitud y optimización de la divergencia KL, UMAP se asienta sobre un **marco teórico robusto derivado de la geometría riemanniana y la topología algebraica**. Esta base le confiere propiedades distintivas en términos de **velocidad, escalabilidad y una superior preservación de la estructura global** de los datos.

Idea general

La potencia de UMAP radica en su suposición fundamental de que los datos de alta dimensión se encuentran aproximadamente sobre una **variedad (manifold) riemanniana** de baja dimensión. El objetivo principal del algoritmo es **preservar la estructura topológica** de esta variedad subyacente al proyectar los datos a un espacio de menor dimensionalidad. La teoría subyacente se expresa más fácilmente en el lenguaje de la topología y la teoría de categorías.

A un alto nivel, UMAP logra esto utilizando **aproximaciones locales de la variedad**. Estas aproximaciones se “parchean” o unen a través de representaciones de **conjuntos simpliciales difusos** (fuzzy simplicial sets) para construir una representación topológica coherente de los datos en alta dimensión. Posteriormente, se construye una representación topológica similar en el espacio de baja dimensión. El algoritmo optimiza entonces la disposición de los puntos en el espacio de baja dimensión para **minimizar la entropía cruzada entre ambas representaciones topológicas**.

La construcción de estas representaciones topológicas difusas se desglosa en dos problemas: 1. **Aproximar la variedad** sobre la que se asume que se encuentran los datos. 2. **Construir una representación de conjunto simplicial difuso** de la variedad aproximada.

En última instancia, UMAP se puede entender, desde una perspectiva computacional, como la **construcción y operación sobre grafos ponderados**, situándolo en la clase de algoritmos de aprendizaje basados en k -vecinos (como Laplacian Eigenmaps, Isomap y t-SNE). Las diferencias entre estos algoritmos residen en los detalles de cómo se construye el grafo y cómo se calcula su disposición en baja dimensión.

Distribución Uniforme de Datos en una Variedad y Aproximación Geodésica El primer paso crucial en UMAP es aproximar la variedad subyacente. Se asume que los datos están **distribuidos uniformemente** en esta variedad con respecto a una **métrica riemanniana** g intrínseca a la misma, no necesariamente la métrica euclidiana del espacio ambiente. Aunque los datos reales rara vez se comportan así, UMAP aborda esto de manera ingeniosa: **adapta la métrica localmente** para que los datos *parezcan* uniformemente distribuidos.

Normalmente, para cada punto x_i , UMAP normaliza las distancias a sus vecinos con respecto a la distancia a su k -ésimo vecino más cercano, denotada ρ_i . Esto se basa en el **Lema 1** (discutido en el paper), que establece que, bajo ciertas condiciones de un *manifold* Riemanniano (M, g) , en una bola B con respecto a g centrada en un punto $p \in M$, la distancia geodésica de p a cualquier $q \in B$ es proporcional a la distancia euclidiana en el espacio ambiente $d_{\mathbb{R}^n}(p, q)$, escalada por el radio de la bola r :

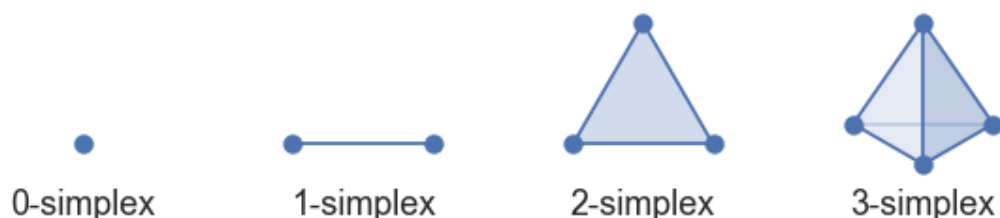
$$\frac{1}{r}d_{\mathbb{R}^n}(p, q)$$

La implicación práctica es que, al crear una **distancia personalizada para cada x_i** (normalizando por ρ_i), se asegura que la suposición de distribución uniforme en la variedad sea válida localmente. El “costo” es que se generan **nociones de distancia locales e independientes** para cada punto, que pueden no ser directamente compatibles entre sí. La tarea siguiente es fusionar estos “espacios métricos discretos” locales en una estructura global consistente.

Representación de la Variedad con Conjuntos Simpliciales Difusos La forma en que UMAP unifica estas nociones locales es a través de los **conjuntos simpliciales difusos**.

- Un **simplex** es una generalización de elementos geométricos básicos: un punto es un 0-simplex, una arista es un 1-simplex, un triángulo es un 2-simplex, etc. Un **conjunto simplicial** es una colección de estos simplices que se conectan entre sí para formar una representación discreta de una forma o espacio topológico.
- El término “**difuso**” (fuzzy) significa que las conexiones entre los simplices no son binarias (existe/no existe), sino que tienen un **grado de membresía o probabilidad** (un valor en el intervalo $[0, 1]$). Esto permite capturar la incertidumbre y la gradualidad en las relaciones de vecindad de los datos, lo que es crucial para manejar la complejidad de la alta dimensionalidad y permite la combinación de las métricas locales inconsistentes en una estructura global.

Computacionalmente, aunque la teoría habla de conjuntos simpliciales difusos completos, en la práctica UMAP trabaja con el “**one skeleton**” de estos conjuntos, lo que se traduce directamente en un **grafo ponderado**. En este grafo: * Cada punto de datos x_i es un **nodo**. * Cada **arista** entre x_i y x_j tiene un **peso** que representa la “fuerza” o “probabilidad” de conexión entre ellos.

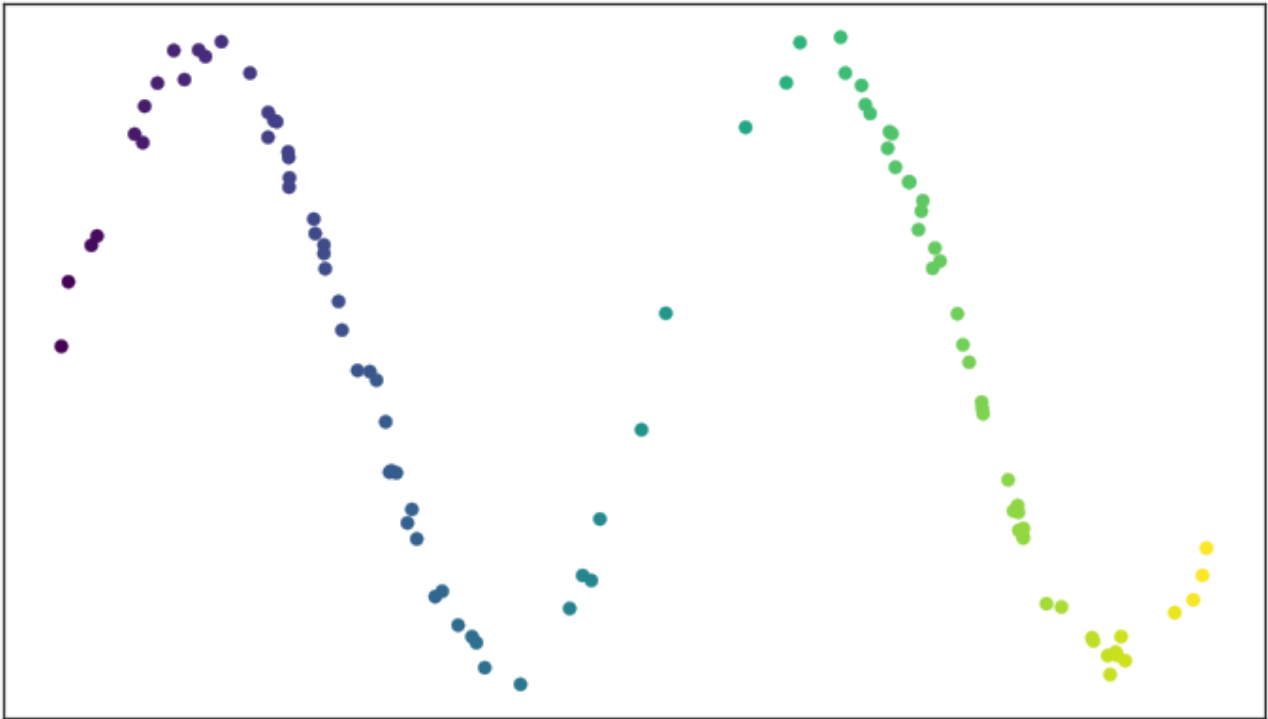


Tipos de símplex

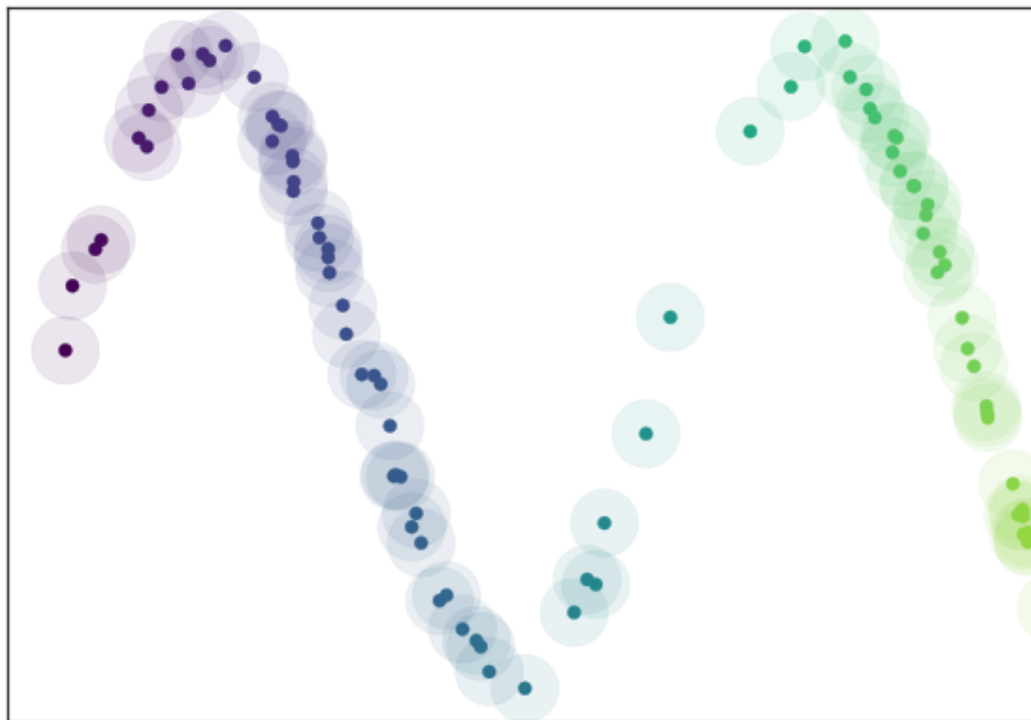
De los Datos Discretos a una Cubierta Adaptativa

Consideremos un conjunto de datos en un espacio de alta dimensión. La hipótesis fundamental de UMAP es que estos datos son un muestreo de una variedad (manifold) subyacente de menor dimensionalidad. El primer paso computacional es construir una “cubierta” de esta variedad mediante conjuntos abiertos, que pueden ser visualizados como “bolas” alrededor de cada punto.

Observemos un conjunto de datos bidimensional de ejemplo, una “onda sinusoidal ruidosa”:



A diferencia de métodos que emplean un radio fijo para estos conjuntos abiertos, UMAP **adapta el tamaño de cada conjunto abierto** basándose en la densidad local de los datos.



Cubierta con Radios Fijos

Una cubierta con un radio globalmente fijo puede llevar a una representación imprecisa de la topología local de la variedad.

Adaptación de la Métrica: Densidad y Uniformidad Local

Para garantizar que todas las vecindades locales muestren una **uniformidad de muestreo aparente**, UMAP implementa una re-escalación adaptativa de la métrica en cada vecindad. Esto se conceptualiza como un ajuste del “volumen” efectivo de la vecindad de k puntos alrededor de cada dato.

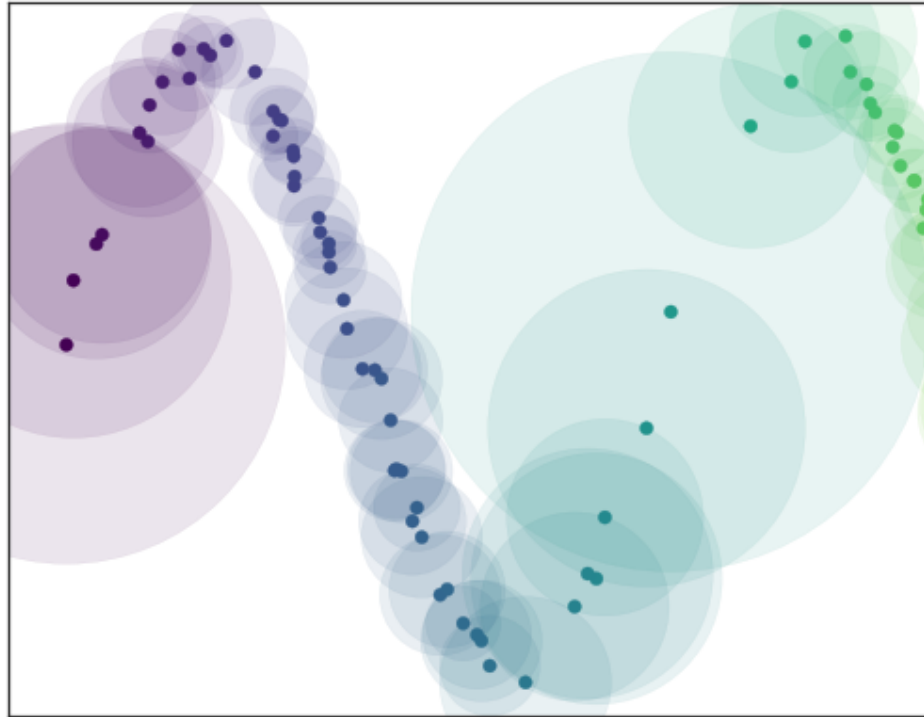
Este ajuste métrico se logra calculando un valor σ_i (sigma local) para cada punto X_i . Este σ_i actúa como un factor de escala. Se determina iterativamente para satisfacer la siguiente condición:

$$\sum_{j=1}^k \exp \left(-\frac{d(X_i, X_j) - \rho_i}{\sigma_i} \right) = \log_2(k)$$

Desglosemos los términos:

- X_i : El **punto de datos focal** bajo consideración.
- X_j : Los k **vecinos más cercanos** a X_i .
- $d(X_i, X_j)$: La **distancia euclidiana** (o la métrica seleccionada) entre X_i y cada vecino X_j en el espacio de alta dimensión.
- ρ_i (rho): La **distancia al primer vecino no cero** de X_i , es decir, la distancia más pequeña a cualquier otro punto.
- σ_i (sigma): El **parámetro de escala local** calculado para cada X_i . Un σ_i pequeño indica una compresión efectiva de las distancias (región densa), mientras que un σ_i grande indica un estiramiento (región dispersa).
- $\log_2(k)$: Un **valor objetivo constante**. Asegura que, en promedio, la suma de las probabilidades de conexión de los k vecinos de X_i sea $\log_2(k)$, normalizando la “conectividad efectiva” de cada punto.

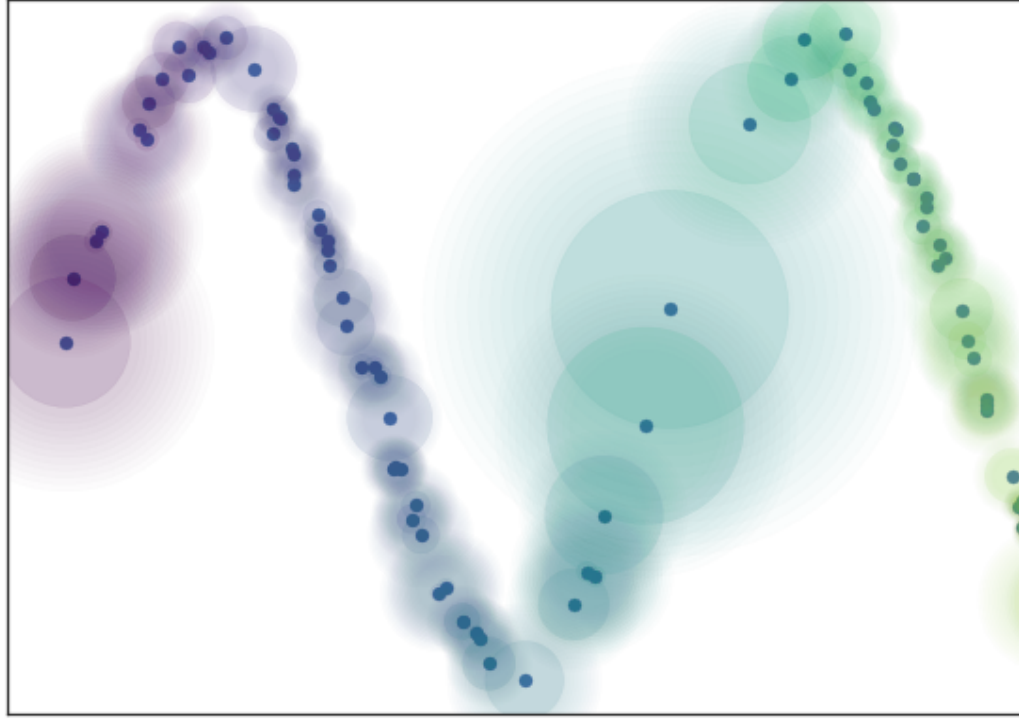
Impacto: Este procedimiento resulta en una **distribución uniforme local simulada**, garantizando que las vecindades de los puntos se comporten de manera consistente, independientemente de la densidad original de los datos.



Cubierta Adaptada por Densidad

La Métrica Adaptada: Visualizando la Uniformización Local

El ajuste dinámico de σ_i y ρ_i produce una métrica localmente adaptada que hace que los puntos se “sientan” uniformemente distribuidos. Conceptualmente, esto puede visualizarse como una deformación del espacio alrededor de cada punto, donde las bolas de radios fijos se “estiran” o “comprimen” para lograr una densidad efectiva constante.



Espacio Métrico Adaptado

Este ajuste es fundamental para que el “Fuzzy Simplicial Set” final refleje con precisión la topología subyacente, independientemente de las variaciones de densidad intrínsecas del dataset original.

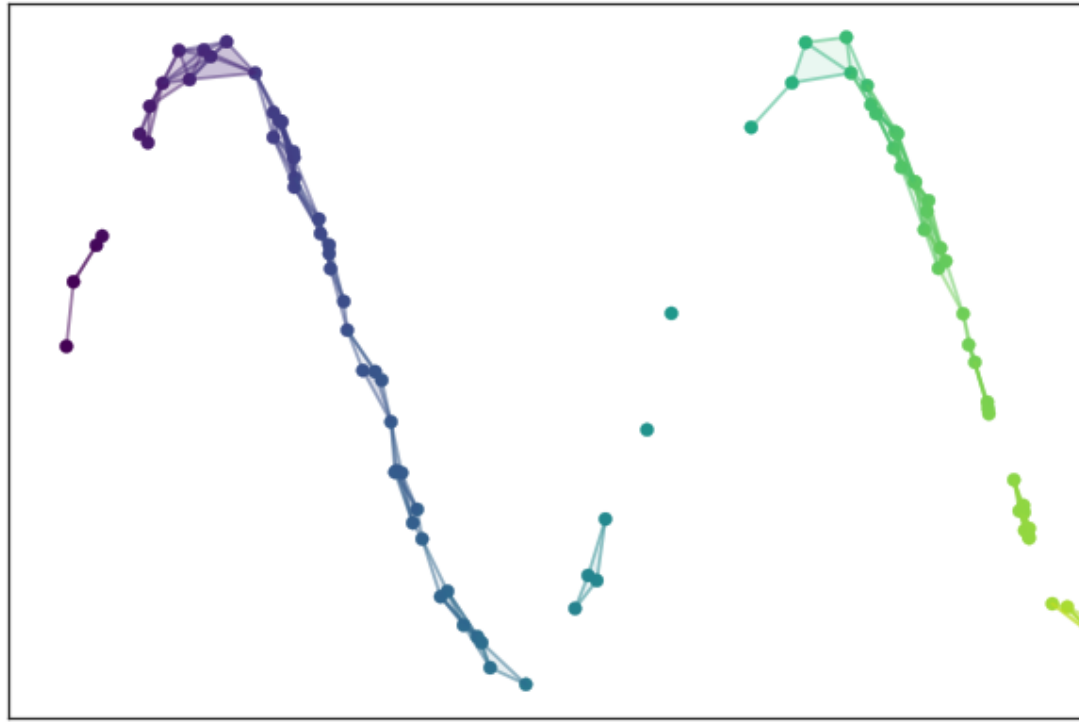
Fuzzy Simplicial Set

Una vez que los parámetros de escala local (σ_i y ρ_i) han sido determinados para cada punto, UMAP calcula los “scores de similaridad” entre cada punto X_i y sus vecinos X_j . Estos scores representan la **fuerza de conexión** probabilística entre los puntos:

$$s_{ij} = \exp \left(-\frac{d(X_i, X_j) - \rho_i}{\sigma_i} \right)$$

Aquí, s_{ij} es el **score de similaridad** (o “probabilidad de conexión”) de X_i a X_j . Un valor cercano a 1 indica una conexión fuerte.

Estos scores se utilizan para construir un **grafo de conectividad ponderado global**. Cada punto de datos se convierte en un nodo del grafo, y las conexiones entre X_i y X_j son aristas con un peso W_{ij} . Este grafo se conceptualiza como un **Conjunto Simplicial Difuso (Fuzzy Simplicial Set)**.



Grafo de Conectividad

Unión Difusa

La característica “difusa” (fuzzy) de los Conjuntos Simpliciales radica en que las conexiones no son binarias (sí/no), sino que poseen un **grado de pertenencia** (un valor continuo entre 0 y 1). Para consolidar los scores s_{ij} y s_{ji} (ya que la similaridad de X_i a X_j puede diferir de X_j a X_i debido a sus σ y ρ individuales), UMAP emplea la siguiente fórmula:

$$W_{ij} = s_{ij} + s_{ji} - (s_{ij} \cdot s_{ji})$$

Análisis de la fórmula:

- W_{ij} : El **peso final de la arista** entre X_i y X_j en el grafo de conectividad global.
- s_{ij} : El score de similaridad calculado de X_i a X_j .
- s_{ji} : El score de similaridad calculado de X_j a X_i .

Esta operación es una “**unión difusa**”. Es fundamental comprender que esta no es una operación de promediado. Promediar diluiría la información crucial inherente a una conexión fuerte. Si X_i percibe a X_j como un vecino muy cercano (alto s_{ij}), o si X_j percibe a X_i como un vecino muy cercano (alto s_{ji}), es deseable que esta fuerte conexión se mantenga en la representación topológica final. La unión difusa garantiza que la existencia de una conexión significativa en **al menos una de las direcciones** (ida o vuelta) contribuya de manera robusta al peso combinado. Conceptualmente, si X_i está conectado a X_j , o X_j está conectado a X_i , entonces X_i y X_j están considerados conectados en el grafo global.

Este grafo ponderado constituye la **aproximación topológica de la variedad** subyacente en el espacio de alta dimensión, capturando tanto las relaciones de proximidad como las características geométricas locales de una manera flexible y probabilística.

Grafo de Conectividad en Alta Dimensión

Como resultado de todos los pasos previos (adaptación métrica local, cálculo de scores de similaridad y unión difusa), el proceso de UMAP culmina en la construcción de un **grafo de conectividad ponderado** en el espacio de alta dimensión.

En este grafo:

- Cada **nodo** representa un punto de dato original.
- Cada **arista** entre dos nodos indica una relación de similaridad o conectividad entre esos puntos.
- El **peso** de cada arista (W_{ij}) cuantifica la fuerza de esa similaridad, calculada a través de la unión difusa.

Este grafo ponderado puede ser convenientemente representado como una **matriz de similaridad dispersa**. En esta matriz, las filas y columnas corresponden a los puntos de datos, y cada entrada (i, j) contiene el peso W_{ij} de la arista que conecta el punto i con el punto j . Dado que UMAP se enfoca en los k vecinos más cercanos, la mayoría de las entradas de esta matriz serán cero, de ahí su naturaleza dispersa.

Ejemplo Conceptual de Matriz de Similaridad

Punto	P_1	P_2	P_3	P_4	P_5
P_1	1.00	0.95	0.00	0.00	0.00
P_2	0.95	1.00	0.82	0.00	0.00
P_3	0.00	0.82	1.00	0.71	0.00
P_4	0.00	0.00	0.71	1.00	0.90
P_5	0.00	0.00	0.00	0.90	1.00

Representación en Baja Dimensión

Una vez que UMAP ha construido el **Conjunto Simplicial Difuso (Fuzzy Simplicial Set)** en el espacio de alta dimensión, que encapsula la estructura topológica y las similaridades probabilísticas (W_{ij}), el siguiente paso es encontrar una representación de estos datos en un espacio de menor dimensionalidad (e.g., $\mathbb{R}^2$ o $\mathbb{R}^3$).

El objetivo es generar una incrustación $Y = \{y_1, \dots, y_N\} \subset \mathbb{R}^d$ (donde d es la dimensión objetivo, típicamente 2 o 3) que sea topológicamente similar al grafo de alta dimensión. Esto se logra formulando un problema de optimización para preservar las **conectividades probabilísticas** establecidas en el espacio original. La meta es que si dos puntos X_i y X_j tienen una alta probabilidad de conexión W_{ij} en alta dimensión, sus correspondientes incrustaciones y_i y y_j deben estar cerca en baja dimensión, y viceversa.

Para cuantificar la “similaridad” o “conectividad” entre los puntos y_i y y_j en el espacio de baja dimensión, UMAP emplea una función de kernel específica, elegida por sus propiedades deseables para la optimización y la visualización:

$$w_{ij} = \frac{1}{1 + a(d(y_i, y_j))^{2b}}$$

Analicemos los componentes de esta fórmula y su propósito:

- w_{ij} : Representa la **similaridad o probabilidad de conexión** entre las proyecciones y_i y y_j en el espacio de baja dimensión. Este es el homólogo de W_{ij} pero definido en el espacio de salida.
- $d(y_i, y_j)$: Es la **distancia euclidiana** entre y_i y y_j en el espacio de baja dimensión.
- a y b : Son **parámetros de UMAP** derivados de los hiperparámetros `min_dist` y `spread`.
 - **min_dist**: Este parámetro controla la distancia mínima que los puntos pueden tener en el espacio de baja dimensión. Un `min_dist` pequeño (cercano a 0) permite que los puntos se agrupen densamente, mientras que un `min_dist` mayor los fuerza a separarse más, resultando en agrupaciones más difusas.
 - **spread**: Este parámetro controla la dispersión de los puntos, influenciando la escala a la que se interpretan las distancias. Un `spread` más alto permite que los puntos se dispersen más ampliamente.

- Los valores de a y b se ajustan para que la curva de decaimiento de la conectividad en baja dimensión (`min_dist` y `spread`) se alinee lo mejor posible con la curva de decaimiento del espacio de alta dimensión (σ_i y ρ_i), creando una correspondencia métrica.

Minimización de la Divergencia de Entropía Cruzada

El corazón de la incrustación de UMAP reside en la minimización de la “divergencia” entre el grafo de conectividad en alta dimensión (W_{ij}) y el grafo en baja dimensión (w_{ij}). Esto se formula como la minimización de la **divergencia de la sección cruzada de la entropía (Cross-Entropy)**, una medida estándar para la diferencia entre dos distribuciones de probabilidad.

La función objetivo que UMAP busca minimizar es:

$$L(Y) = \sum_{i \neq j} \left[W_{ij} \log \left(\frac{W_{ij}}{w_{ij}} \right) + (1 - W_{ij}) \log \left(\frac{1 - W_{ij}}{1 - w_{ij}} \right) \right]$$

Donde:

- W_{ij} : La **probabilidad de conexión** entre X_i y X_j en el espacio de **alta dimensión** (un valor entre 0 y 1).
- w_{ij} : La **probabilidad de conexión** entre y_i y y_j en el espacio de **baja dimensión** (calculada con el kernel de Cauchy).

Interpretación de los Términos de la Función de Pérdida y sus Gradientes:

1. **Término de Atracción (primer término):** $W_{ij} \log \left(\frac{W_{ij}}{w_{ij}} \right)$
 - Cuando W_{ij} es alto (los puntos están conectados en alta dimensión) y w_{ij} es bajo (están lejos en baja dimensión), este término se vuelve grande y positivo, generando una penalización significativa.
 - El gradiente de este término tiende a **acercar** los puntos y_i y y_j .
2. **Término de Repulsión (segundo término):** $(1 - W_{ij}) \log \left(\frac{1 - W_{ij}}{1 - w_{ij}} \right)$
 - Cuando W_{ij} es bajo (los puntos no están conectados en alta dimensión) y w_{ij} es alto (están demasiado cerca en baja dimensión), este término también se vuelve grande, penalizando la proximidad.
 - El gradiente de este término tiende a **alejar** los puntos y_i y y_j .

La minimización de esta función de costo guía el proceso de ajuste de las posiciones de los puntos en el espacio de baja dimensión, buscando una configuración óptima que preserve la topología probabilística del espacio original.

Descenso de Gradiente Estocástico

La minimización de la función de pérdida $L(Y)$ se realiza mediante un proceso iterativo de **Descenso de Gradiente Estocástico (Stochastic Gradient Descent - SGD)**.

Las etapas clave son:

1. **Inicialización del Embedding:**
 - Para acelerar la convergencia y proporcionar un punto de partida razonable, los puntos y_i en el espacio de baja dimensión se inicializan comúnmente utilizando una **incrustación espectral** (e.g., basada en los vectores propios del Laplaciano normalizado del grafo de conectividad de alta dimensión). Alternativamente, se puede usar una inicialización aleatoria.
2. **Muestreo de Aristas y No-Aristas:**
 - En cada iteración del SGD, se selecciona aleatoriamente una arista (un par (i, j) con $W_{ij} > 0$) de la matriz de similaridad en alta dimensión. Para este par, se aplica una **fuerza de atracción** a y_i y y_j .

- Para la repulsión, UMAP emplea **muestreo negativo**. En lugar de considerar todos los pares no conectados (lo que sería computacionalmente inviable para grandes datasets, $O(N^2)$), se muestrean aleatoriamente varios puntos y_k que no están conectados a y_i . Para estos pares (i, k) , se aplica una **fuerza de repulsión**. La proporción de muestras negativas por muestra positiva es un parámetro configurable.
3. **Cálculo de Gradientes y Actualización de Posiciones:**
- Se calculan las derivadas parciales de la función de pérdida L con respecto a las coordenadas de los puntos y_i y y_j .
 - Estas gradientes determinan la dirección y magnitud del ajuste de las posiciones de los puntos.
 - Las posiciones de los puntos se actualizan iterativamente: $y_k \leftarrow y_k - \eta \nabla_{y_k} L$, donde η es la tasa de aprendizaje.
4. **Annealing y Convergencia:**
- El proceso de optimización se ejecuta durante un número predefinido de épocas (**n_epochs**).
 - A menudo, las fuerzas repulsivas se ponderan más fuertemente al inicio de la optimización para asegurar que los puntos se dispersen adecuadamente y no queden atrapados en mínimos locales apretados. A medida que avanza la optimización, el equilibrio entre atracción y repulsión se ajusta para permitir que las agrupaciones se refinan.

Impacto de Parámetros Clave en la Embedificación

La calidad y la interpretación de la incrustación de UMAP están significativamente influenciadas por varios hiperparámetros, que afectan directamente la función de pérdida y el proceso de optimización:

- **n_neighbors (Número de vecinos):** Este parámetro (usado en la fase de alta dimensión para construir el k-NN graph) tiene un impacto crucial en el balance entre la preservación de la estructura local y global en la incrustación final.
 - **Valores pequeños:** Enfatizan la estructura local, lo que puede llevar a una fragmentación de la incrustación y a la aparición de muchos clústeres pequeños y separados.
 - **Valores grandes:** Permiten que UMAP considere más la estructura global de los datos, lo que puede resultar en una vista más cohesiva, pero quizás menos detallada de las relaciones locales.
- **min_dist (Distancia Mínima):** Directamente relacionado con el parámetro **a** del kernel de Cauchy.
 - Controla cuán cerca se pueden agrupar los puntos en la incrustación de baja dimensión.
 - **min_dist pequeño (cercano a 0):** Los puntos pueden agruparse de forma muy densa, lo que es útil para visualizar clústeres bien separados y compactos.
 - **min_dist grande:** Los puntos son forzados a estar más dispersos, lo que puede revelar una estructura más continua y fluida, o separar clústeres que de otra manera se solaparían visualmente.
- **spread (Dispersión):** Directamente relacionado con el parámetro **b** del kernel de Cauchy.
 - Controla la dispersión general del embedding.
 - **spread pequeño:** Resulta en una incrustación más compacta.
 - **spread grande:** Permite que la incrustación se extienda más, útil para visualizar la jerarquía global.
- **n_epochs (Número de Épocas):**
 - Define el número de iteraciones del algoritmo de optimización SGD.
 - Más épocas (**n_epochs** más alto) generalmente conducen a una incrustación más optimizada y estable, pero a costa de un mayor tiempo de cómputo.
 - Para datasets grandes o embeddings de alta calidad, un **n_epochs** elevado es recomendable.

Estos parámetros permiten al usuario ajustar el equilibrio entre la preservación de la estructura local y global, y la densidad visual de la incrustación resultante, adaptando UMAP a diversas necesidades de análisis y visualización.

Ventajas Clave de UMAP

UMAP se compara favorablemente con t-SNE y otros algoritmos de la misma clase (basados en grafos de k -vecinos). Su principal ventaja es que sus elecciones algorítmicas, tanto en la construcción del grafo como en el diseño de la incrustación, tienen una **justificación teórica clara** derivada de sus axiomas sobre la variedad y la topología, lo que le permite ser:

- **Más Rápido y Escalable:** Su optimización en grafos dispersos y el muestreo eficiente de no-vecinos lo hacen significativamente más rápido que t-SNE, siendo aplicable a datasets de millones de puntos con una complejidad computacional que se aproxima a $O(N \log N)$.
- **Mejor Preservación de la Estructura Global:** La naturaleza de su función objetivo y la forma en que maneja las distancias adaptativas le permiten mantener de manera más consistente las relaciones a gran escala entre los clusters, además de la estructura local.
- **Flexible y Controlable:** Los parámetros `min_dist` y `spread` ofrecen un control intuitivo sobre la densidades y separación de los *clusters* en la visualización final.
- **Capacidad de Transformación:** UMAP puede aprender una transformación desde el espacio de alta a baja dimensión, lo que permite proyectar nuevos datos en un *embedding* existente sin recalcularlo todo.

Comparativa t-SNE vs UMAP

Ambas t-SNE y UMAP son herramientas poderosas para la reducción de dimensionalidad no lineal y la visualización, buscando revelar la estructura intrínseca de los datos. Sin embargo, sus fundamentos matemáticos y sus enfoques computacionales les otorgan características y rendimientos distintivos.

Característica	t-SNE	UMAP
Fundamentos Teóricos	Teoría de probabilidades, divergencia Kullback-Leibler. Busca preservar las probabilidades de similitud (vecindad).	Topología algebraica, teoría de la geometría riemanniana, conjuntos simpliciales difusos. Busca preservar la estructura topológica (conectividad).
Concepto de Similitud	Probabilidades de similitud p_{ij} (basadas en Gaussianas) en alta dimensión y q_{ij} (basadas en t-Student) en baja dimensión.	Conectividad en un grafo ponderado (fuzzy simplicial set) que representa la proximidad. La noción de “vecino” es localmente adaptable.
Preservación de Estructura	Excelente para la estructura local (clusters). Tiende a aglomerar puntos cercanos y separarlos bien. A menudo distorsiona la estructura global (las distancias entre clusters no son fiables).	Excelente para estructura local Y global. Busca preservar tanto los clusters como las relaciones entre ellos. Es más probable que las distancias entre clusters tengan significado.
Función de Costo	Divergencia Kullback-Leibler (KL) asimétrica: $KL(P Q)$. Penaliza fuertemente la pérdida de vecinos cercanos.	Inspirada en la entropía cruzada binaria, optimizada para la correspondencia de grafos: $\sum [w \log(w/q) + (1 - w) \log((1 - w)/(1 - q))]$.
Distribución en Baja Dimensión	Distribución t-Student con 1 grado de libertad (Cauchy). Sus colas pesadas resuelven el “problema de hacinamiento”.	Función de conectividad personalizada $(1 + a y_i - y_j ^{2b})^{-1}$. Parametros <code>min_dist</code> y <code>spread</code> controlan su forma para influir en la dispersión.

Característica	t-SNE	UMAP
Velocidad y Escalabilidad	Generalmente lento. Su complejidad es $O(N^2)$ (o $O(N \log N)$ con optimizaciones como la del árbol de Barnes-Hut). No es ideal para datasets con más de ~50,000 puntos.	Significativamente más rápido. Su complejidad es $O(N \log N)$ para la construcción del grafo y luego $O(N)$ para la optimización. Puede manejar datasets de millones de puntos.
Reproductibilidad	Muy sensible a los parámetros (perplexity , learning_rate , init). Ejecuciones repetidas con diferentes random_state pueden producir visualizaciones notablemente diferentes.	Generalmente más robusto a los cambios de parámetros y más consistente en la estructura global resultante entre ejecuciones.
Parámetros Clave	perplexity (número de “vecinos efectivos”), learning_rate (tasa de aprendizaje).	n_neighbors (número de vecinos para construir el grafo inicial), min_dist (separación mínima entre puntos proyectados), spread (escala de los clusters), metric (métrica de distancia).
Forma de los Clusters	Tiende a producir clusters más densos y circulares, a veces con puntos aglomerados en el centro del mapa.	Puede producir clusters con formas más diversas y una mejor separación visual, reflejando mejor la estructura intrínseca de los datos.
Aplicaciones Típicas	Excelente para la visualización de datos de transcriptómica (ej. single-cell RNA-seq), imágenes, texto. Muy bueno para identificar subpoblaciones distintas.	Ampliamente utilizado en biología (ej. single-cell), visualización de embeddings de procesamiento de lenguaje natural (NLP), análisis de imágenes grandes. A menudo es la opción preferida por su equilibrio entre velocidad y calidad del embedding.

Implementación en el dataset MNIST

En esta sección, aplicaremos tanto t-SNE como UMAP al dataset de dígitos de Scikit-learn y analizaremos los resultados, comparando las visualizaciones y las métricas de calidad de las proyecciones.

Representación de Imágenes como Matrices

Para procesar imágenes con algoritmos de machine learning, es fundamental entender cómo se representan numéricamente. Las imágenes, en esencia, son rejillas de píxeles, donde cada píxel tiene un valor numérico que representa su intensidad o color.

Imágenes en Escala de Grises Una imagen en escala de grises se representa como una matriz 2D, donde cada elemento de la matriz corresponde a un píxel y su valor representa la intensidad del gris (típicamente entre 0 para negro y 255 para blanco, u otros rangos dependiendo del formato).

En nuestro ejemplo, el dataset de dígitos de Scikit-learn consta de imágenes de 8x8 píxeles en escala de grises. Cada imagen individual se puede visualizar como una matriz de 8x8:

$$\begin{pmatrix} valor_{0,0} & valor_{0,1} & \dots & valor_{0,7} \\ valor_{1,0} & valor_{1,1} & \dots & valor_{1,7} \\ \vdots & \vdots & \ddots & \vdots \\ valor_{7,0} & valor_{7,1} & \dots & valor_{7,7} \end{pmatrix}$$

Imágenes a Color (RGB) Las imágenes a color, comúnmente en formato RGB (Rojo, Verde, Azul), se representan como tensores 3D. Tienen dimensiones de altura, ancho y canales de color. Para una imagen RGB, hay 3 canales:

$$\text{Imagen RGB} \in \mathbb{R}^{\text{altura} \times \text{ancho} \times 3}$$

Cada canal es una matriz 2D que representa la intensidad de ese color específico para cada píxel. La combinación de los valores en los tres canales para un píxel dado determina su color final.

Aplanamiento (Flattening) para Algoritmos Lineales Algoritmos como t-SNE o PCA trabajan con vectores de características de una sola dimensión. Por lo tanto, es necesario “aplanar” la representación matricial o tensorial de cada imagen en un único vector largo.

Para una imagen de 8x8 como en el dataset de dígitos, la matriz 2D de 8x8 (64 píxeles) se aplanan en un vector 1D de 64 elementos:

$$\begin{pmatrix} valor_{0,0} \\ valor_{0,1} \\ \vdots \\ valor_{7,7} \end{pmatrix}_{64 \times 1}$$

Para una imagen RGB de altura x ancho x 3 canales, el tensor 3D se aplanan en un vector 1D de altura × ancho × 3 elementos. Por ejemplo, una imagen de 100x100 píxeles RGB se convertiría en un vector de 100 × 100 × 3 = 30,000 elementos.

Este vector aplanado se convierte en la “muestra” o “instancia” de datos en el conjunto de datos de entrada para t-SNE, donde cada elemento del vector es una “característica” (un valor de píxel).

Visualización Tabular de los Datos (DataFrame)

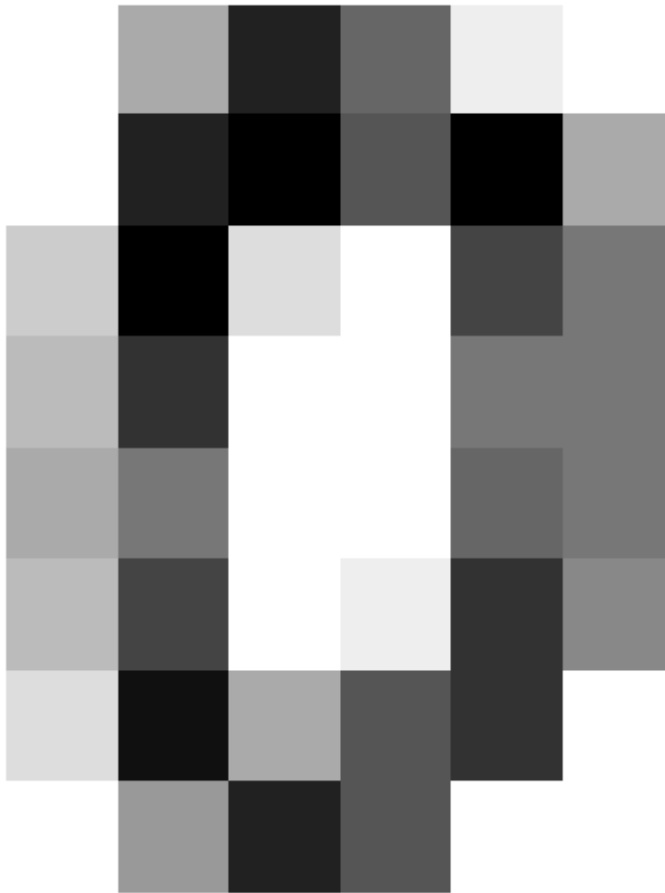
Para comprender mejor la estructura de los datos aplanados antes de aplicar los algoritmos de reducción de dimensionalidad, podemos visualizarlos en un formato tabular, como un DataFrame de Pandas. Cada fila representará una imagen aplanada (una muestra), y cada columna representará un píxel específico (una característica).

En este DataFrame, cada una de las primeras 64 columnas (`pixel_0` a `pixel_63`) representan la intensidad aplanada de un píxel de la imagen de 8x8. La última columna (`digito`) contiene la etiqueta de la clase (el dígito que representa la imagen).

Esta vista nos permite ver los valores numéricos exactos que alimentan los algoritmos.

Visualización de Datos Originales

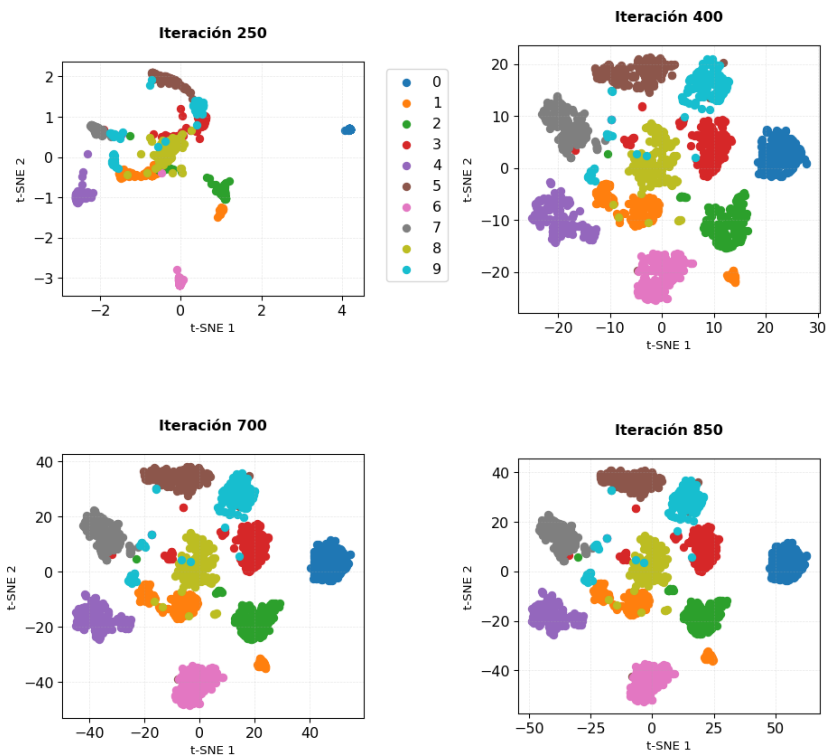
Ejemplo de Dígito (0)



Implementación y Análisis de t-SNE

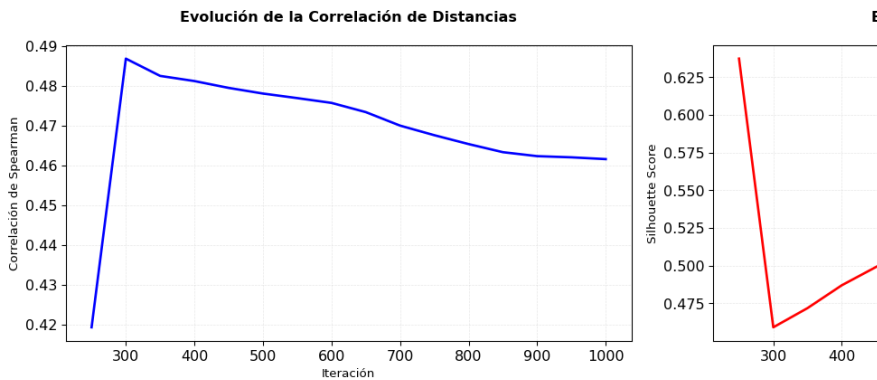
En esta subsección, aplicaremos t-SNE al dataset de dígitos y exploraremos el proceso de optimización y el efecto de sus hiperparámetros.

Evolución de la Proyección t-SNE



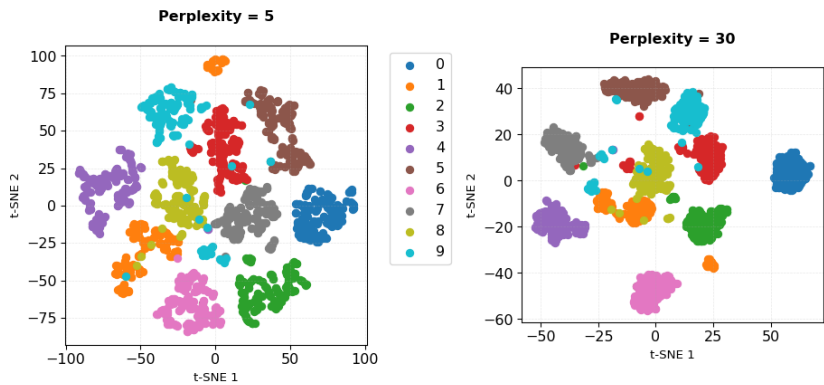
Proceso de Optimización de t-SNE

Métricas de Convergencia de t-SNE



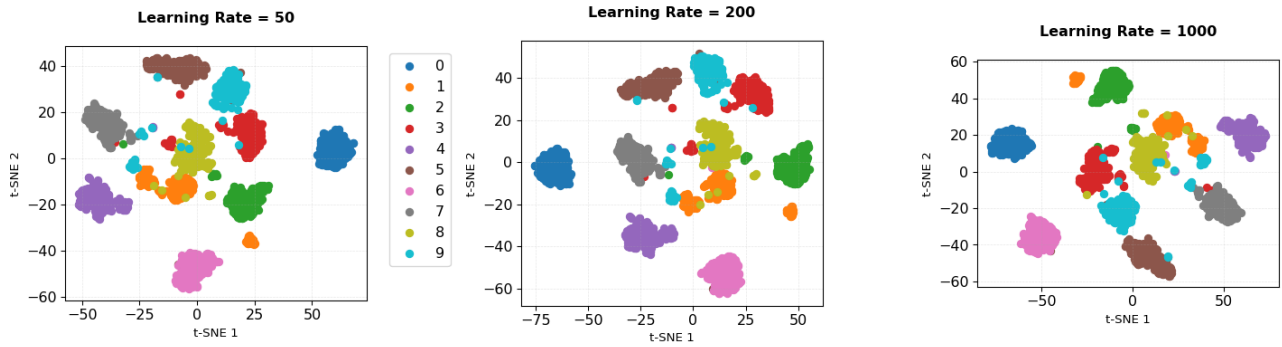
Análisis de la Convergencia de t-SNE

Efecto de la Perplejidad en t-SNE



Efecto de los Hiperparámetros de t-SNE

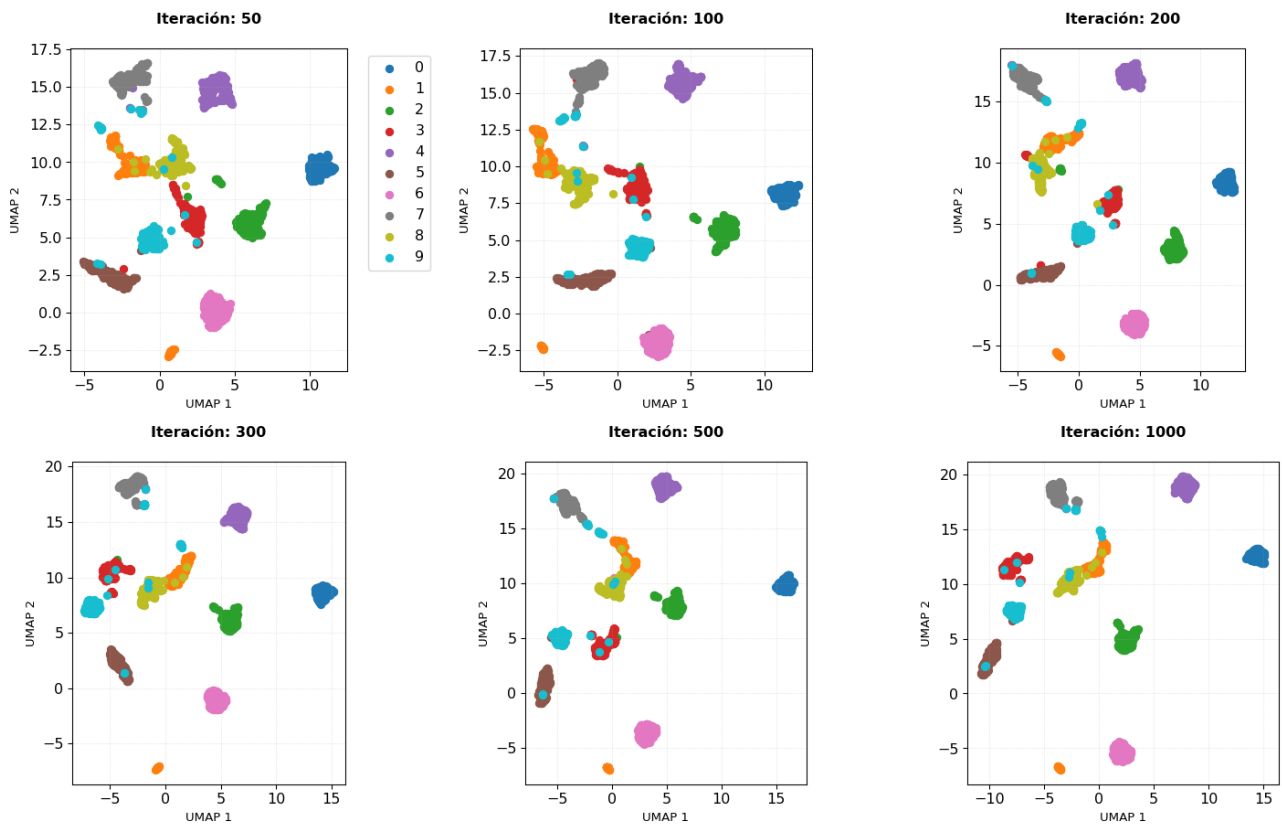
Efecto del Learning Rate en t-SNE



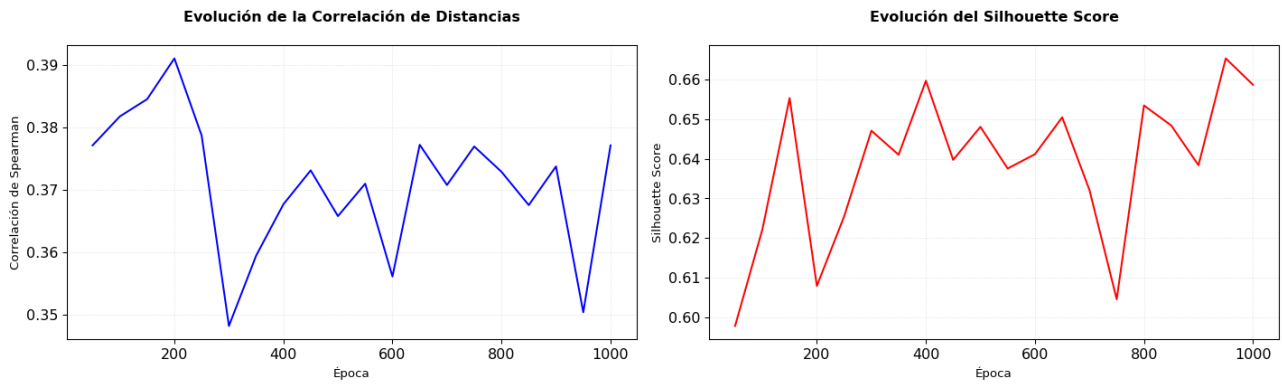
Implementación y Análisis de UMAP en MNIST

En esta sección, aplicaremos UMAP al dataset de dígitos MNIST usando scikit-learn y analizaremos su comportamiento y parámetros.

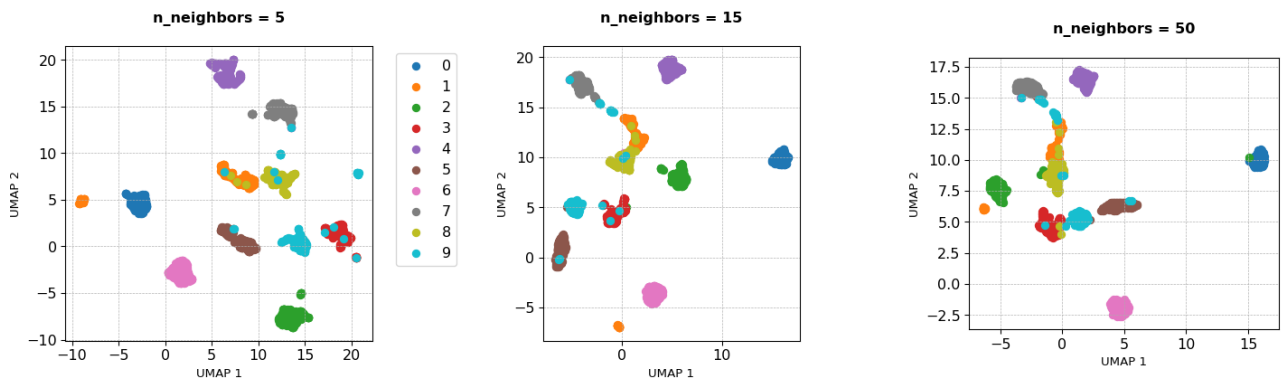
Evolución de la Proyección UMAP por Épocas



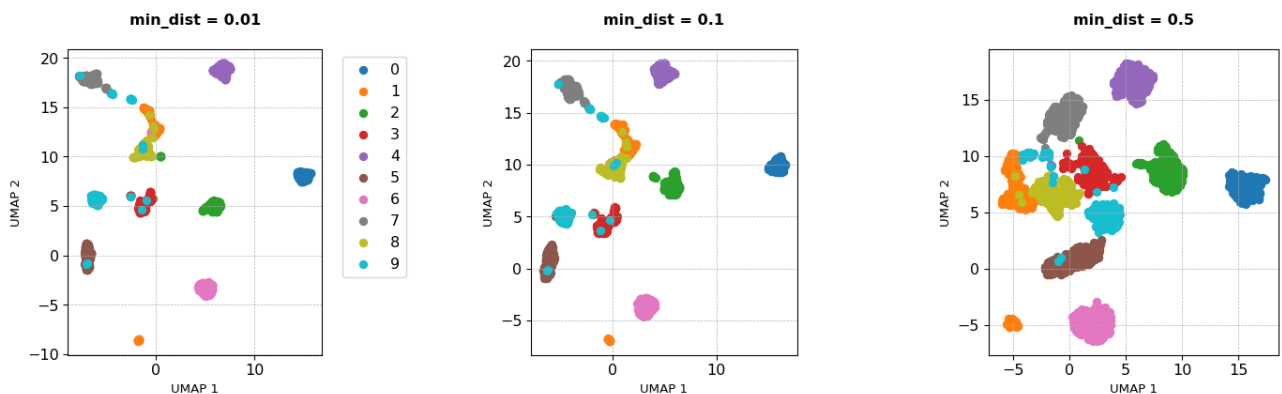
Métricas de Convergencia de UMAP



Efecto del Parámetro $n_neighbors$



Efecto del Parámetro min_dist



Predicción con UMAP

Una de las ventajas significativas de UMAP, en contraste con t-SNE, es su capacidad para **aprender una función de mapeo** desde el espacio de alta dimensión al espacio de baja dimensión. Esto significa que, una vez que un modelo UMAP ha sido entrenado con un conjunto de datos, se puede utilizar para **transformar nuevos datos** (es decir, datos no vistos durante el entrenamiento) en el mismo espacio de baja dimensión sin necesidad de reentrenar el modelo completo.

Esta capacidad permite:

- **Evaluación realista:** Permite evaluar el rendimiento de los modelos de aprendizaje automático en el espacio de baja dimensión utilizando datos no vistos, lo cual es fundamental para una evaluación imparcial.
- **Implementación en producción:** Una vez entrenado, el modelo UMAP puede ser guardado y reutilizado para proyectar nuevos datos en tiempo real, lo que lo hace adecuado para sistemas de producción.

Proyección de Datos No Vistos en UMAP

La capacidad de UMAP para proyectar nuevos datos se deriva de su proceso de entrenamiento, donde no solo se reduce la dimensionalidad de los datos de entrada, sino que también se construye un **grafo de vecindad** y se aprende una **función de transformación implícita**. Cuando se invoca el método `transform()` en un nuevo conjunto de datos X_{new} , UMAP realiza los siguientes pasos:

1. **Cálculo de Distancias a los Vecinos más Cercanos Aprendidos:** Para cada nuevo punto $x_{\text{new},i} \in X_{\text{new}}$, UMAP identifica sus k vecinos más cercanos dentro del conjunto de datos de entrenamiento original X_{train} . Se calculan las distancias $d(x_{\text{new},i}, x_j)$ a estos vecinos.
2. **Inferencia de Probabilidades de Conexión:** Utilizando los parámetros σ_j y ρ_j aprendidos de los vecinos del conjunto de entrenamiento, se infieren las probabilidades de conexión para el nuevo punto con sus vecinos más cercanos. Esto se asemeja al cálculo de s_{ij} en la fase de construcción del grafo:

$$s_{\text{new},ij} = \exp \left(-\frac{d(x_{\text{new},i}, x_j) - \rho_j}{\sigma_j} \right)$$

donde x_j son los vecinos de $x_{\text{new},i}$ en X_{train} .

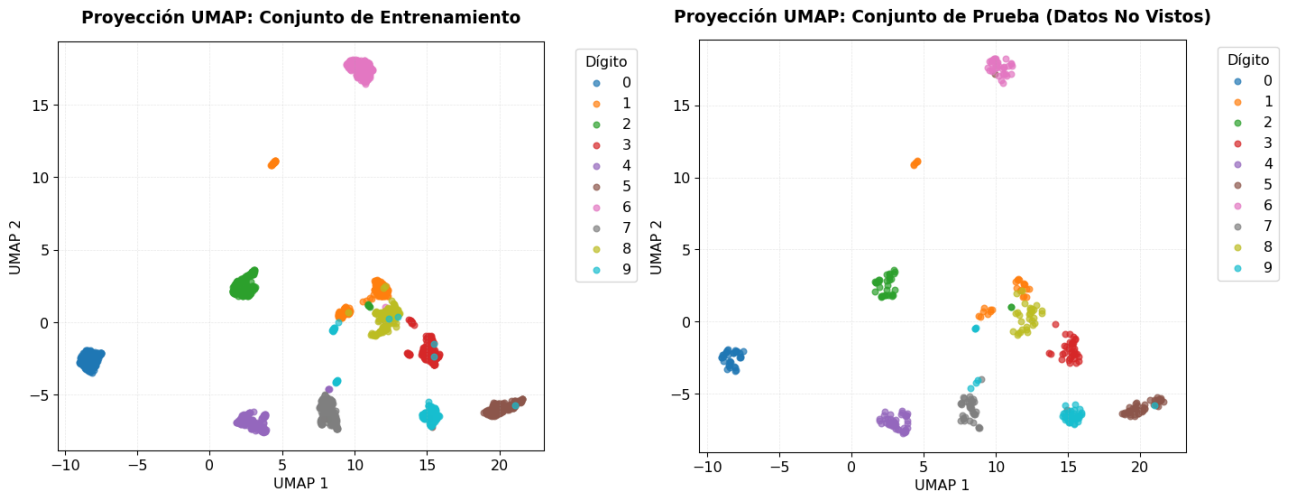
3. **Localización en el Espacio de Baja Dimensión:** El nuevo punto $y_{\text{new},i}$ se inserta en el espacio de baja dimensión de tal manera que sus probabilidades de conexión con los puntos de entrenamiento cercanos y_j (ya incrustados) sean lo más similares posible a las probabilidades inferidas en alta dimensión. Esto se logra minimizando una función de costo similar a la utilizada en la fase de entrenamiento, pero manteniendo fijos los puntos del conjunto de entrenamiento y_j .

$$L(y_{\text{new},i}) = \sum_{j \in \text{vecinos}(x_{\text{new},i})} \left[s_{\text{new},ij} \log \left(\frac{s_{\text{new},ij}}{w_{\text{new},ij}} \right) + (1 - s_{\text{new},ij}) \log \left(\frac{1 - s_{\text{new},ij}}{1 - w_{\text{new},ij}} \right) \right]$$

donde $w_{\text{new},ij}$ es la probabilidad de conexión en baja dimensión entre $y_{\text{new},i}$ y y_j , calculada con el kernel de Cauchy:

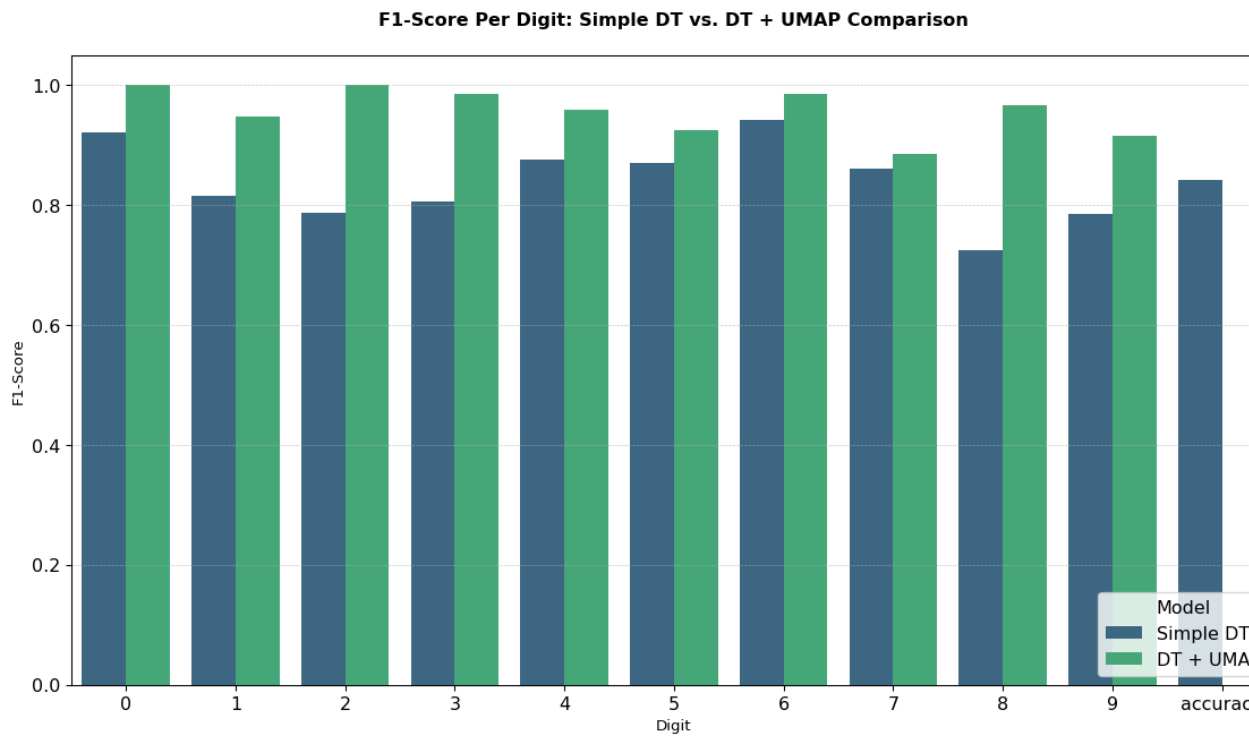
$$w_{\text{new},ij} = \frac{1}{1 + a(d(y_{\text{new},i}, y_j))^{2b}}$$

Este proceso es computacionalmente eficiente porque solo se optimiza la posición de los nuevos puntos, sin alterar el embedding ya existente del conjunto de entrenamiento. Esto permite una proyección consistente y escalable de datos no vistos.



Predicción de Dígitos con Árboles de Decisión

Para complementar el análisis de reducción de dimensionalidad, es útil demostrar cómo los datos proyectados o el dataset original pueden ser utilizados en tareas de clasificación.



Resultados

Se observa que el modelo con UMAP tiene un mejor rendimiento que el modelo simple. Esto se debe a que UMAP al ser un algoritmo de reducción de dimensionalidad no lineal, es capaz de capturar las relaciones no lineales entre los datos, lo que es útil para la clasificación.