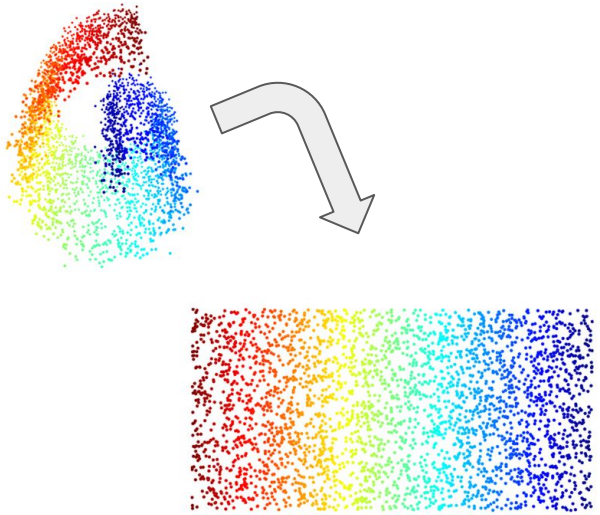
Reducción de dimensionalidad

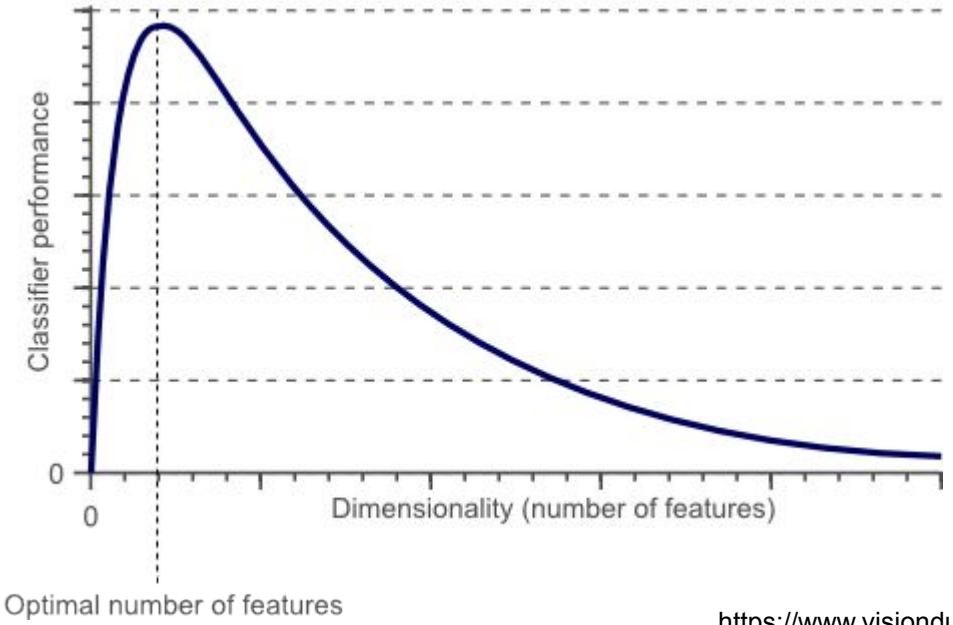
¿Qué es la reducción de dimensionalidad?

- Técnica utilizada para reducir el número de atributos (*features*) en un conjunto de datos, manteniendo la mayor cantidad de información posible de los datos originales.

- Proceso para transformar datos de alta dimensionalidad hacia un espacio de menor dimensionalidad que preserva la “esencia” de los datos originales.



Incluir más atributos no siempre mejora la performance



https://www.visiondummy.com/

La famosa “maldición de la dimensionalidad” (*curse of dimensionality*)

- Familia de fenómenos “problemáticos” en espacios de muy alta dimensionalidad.
- En un espacio de *f* atributos, el volumen en este espacio crece como potencia de *f* (orden O^f).
- Cuando la dimensionalidad aumenta, el volumen del espacio aumenta tan rápidamente que los datos representados quedan muy dispersos (*sparse*).
- La cantidad de datos necesarios para analizar estos espacios crece exponencialmente con la dimensionalidad.
- En espacios de alta dimensionalidad todos los puntos tienden a ser distantes (disímiles) entre sí, lo que dificulta encontrar organización o estructura en los datos.
- También aumenta la tendencia a la multicolinearidad (que algunos atributos sean correlacionados entre sí) dificultando regresiones.

Ventajas

- Menos atributos se traduce en menor complejidad
- Reducción de “ruido” o redundancias, lo que reduce el *overfitting*
- Reducción de espacio de almacenamiento (data compression)
- Menores tiempos de cómputo
- La precisión de los modelos puede aumentar al eliminar datos irrelevantes/engañosos
- Mejora la visualización de los datos

Desventajas

- Podemos perder información, potencialmente afectando el análisis
- Puede requerir mucho poder de cómputo, dependiendo de la técnica
- La interpretación de atributos transformados puede ser compleja o imposible

Usos típicos

- Visualización; compresión y/o filtrado de datos
- Reducción de dimensionalidad para comprimir cantidad de resultados de un clustering ya realizado
- Entender la estructura latente en datos (*manifold learning*)
- Regresión y clasificación de datos en espacio de menor dimensionalidad

Dos familias de técnicas de reducción de dimensionalidad

- Selección de atributos (*feature selection*): donde se selecciona un subconjunto de los atributos originales que resulten más relevantes para el problema a resolver. Una especie de “filtrado” que descarte atributos que no sean informativos.

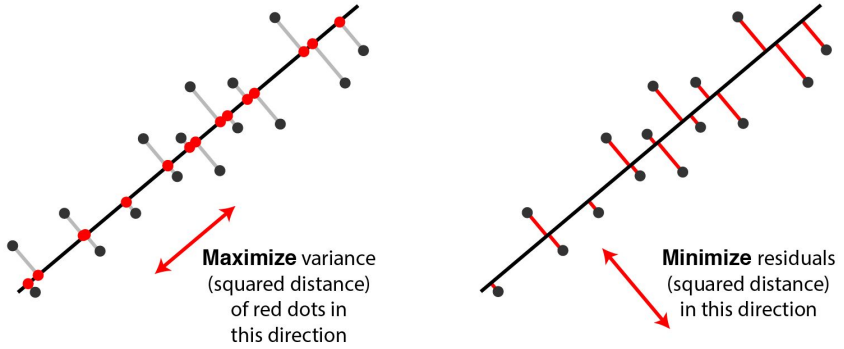
- Extracción de atributos (*feature extraction*): donde se crea un conjunto de nuevos atributos mediante la combinación o transformación de atributos originales, siempre que este nuevo conjunto sea de menor dimensionalidad que el original.

Modelos lineales: la piedra fundamental de la reducción de dimensionalidad

Dos visiones de PCA:

a) Rotación ortogonal de los datos hacia nuevos atributos (PCs) de máxima varianza

b) Mejor aproximación posible a los datos (cuadrados mínimos) en un subespacio de menor dimensión obtenido mediante una rotación ortogonal



- No distorsiona los datos. Se puede ir y volver entre las diferentes representaciones (sin perder info si mantuve todos los autovectores)
- Eficiente computacionalmente.

SVD es una generalización de PCA

Relación entre PCA y SVD

Consideremos la matriz de datos $\mathbf{X}$ de tamaño rectangular $n \times p$, donde n es el número de observaciones y p es el número de atributos. Asumimos que $\mathbf{X}$ está centrada (se le sustrajo las medias a cada columna). Entonces la matriz de covarianza $\mathbf{C}$ está dada por:

$$\mathbf{C} = \frac{\mathbf{X^T X}}{(n-1)}$$

es una matriz cuadrada $p \times p$ y es simétrica, por lo que puede ser diagonalizada:

$$\mathbf{C} = \mathbf{W D W^T}$$

donde $\mathbf{W}$ es una matriz de autovectores columna y $\mathbf{D}$ es una matriz diagonal de autovalores λ_i en orden decresciente a lo largo de la diagonal. Todas tienen tamaño $p \times p$. Los autovectores se denominan *ejes principales*. La matriz $\mathbf{Scores}$ de tamaño $n \times p$ que contiene la transformación de los datos sobre dichos componentes (una rotación ortogonal) se llama *componentes principales* (PCs) o *Scores de los componentes principales*:

$$\mathbf{Scores} = \mathbf{XW}$$

Por otro lado, si aplicamos Singular Value Decomposition (SVD) a la matriz de datos $\mathbf{X}$ la estamos decomponiendo de la siguiente manera:

$$\mathbf{X} = \mathbf{U S V^T}$$

donde $\mathbf{U}$ es una matriz unitaria de tamaño $n \times n$ (cuyas columnas se llaman *vectores singulares izquierdos*), $\mathbf{S}$ es la matriz diagonal rectangular $n \times p$ de valores singulares s_i, y $\mathbf{V}$ es una matriz unitaria $p \times p$ cuyas columnas se llaman *vectores singulares derechos*. De aqui se puede ver que:

$$\mathbf{C} = \frac{\mathbf{V S U^T U S V^T}}{(n-1)} = \mathbf{V} \frac{\mathbf{S^2}}{\mathbf{(n-1)}} \mathbf{V^T}$$

por lo que los vectores singulares derechos $\mathbf{V}$ son iguales a los ejes principales representados por la matriz $\mathbf{W}$:

$$\mathbf{V} = \mathbf{W}$$

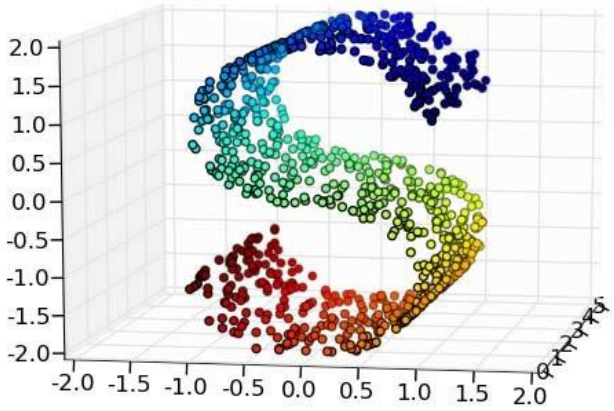
los valores singulares están relacionados a los autovalores de matriz de covarianza $\mathbf{C}$ via $\lambda_i = s_i^2/(n-1)$. Finalmente, los datos transformados via PCA, $\mathbf{Scores}$ quedan:

$$\mathbf{Scores} = \mathbf{XW} = \mathbf{U S V^T V} = \mathbf{US}$$

Corolario: al hacer SVD trabajamos directamente sobre la matriz $\mathbf{X}$ y obtenemos los PCs, los autovalores y los datos transformados $\mathbf{Scores}$ de PCA en una sola operación

Cuando los datos viven en variedades curvas las técnicas lineales fallan

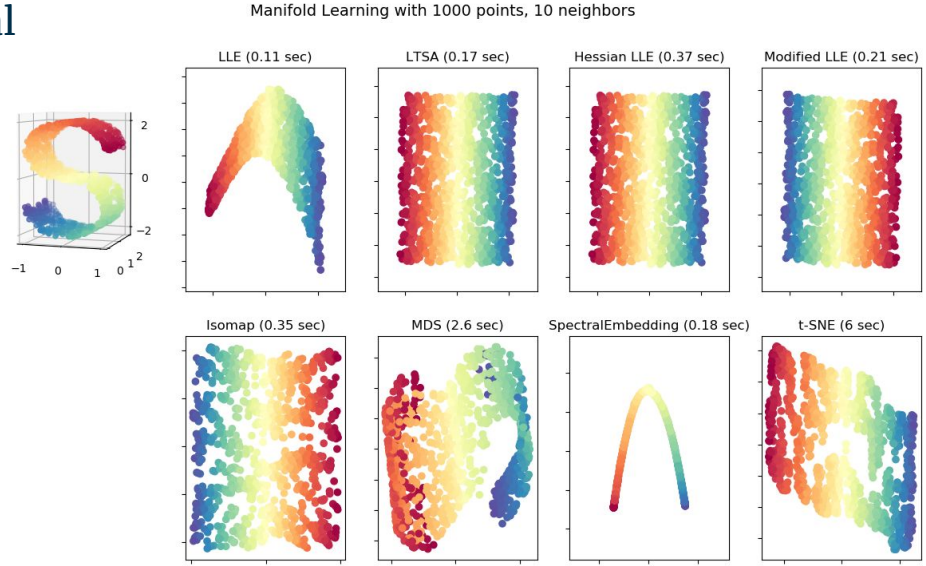
Variedad: objeto geométrico que generaliza la noción intuitiva de “curva” (1-variedad) y de “superficie” (2-variedad) a cualquier dimensión. Una variedad de *n* dimensiones (*n*-variedad) se parece localmente a un espacio euclideano de *n* dimensiones.



Aprendizaje de variedades (*manifold learning*)

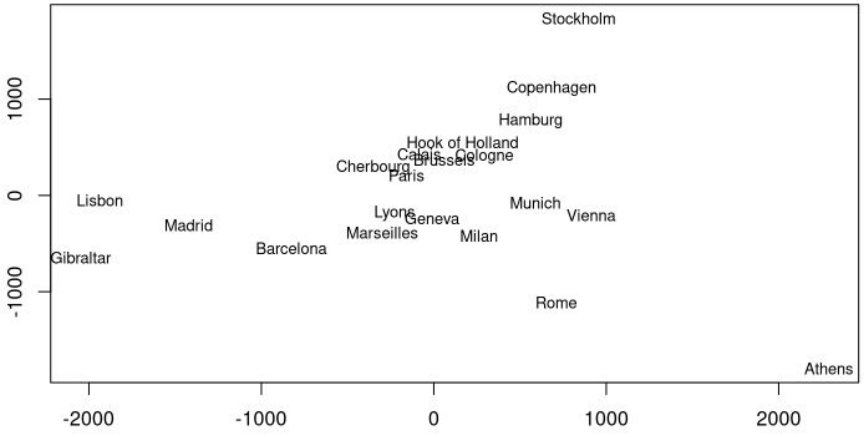
Intento de generalizar técnicas de reducción de dimensionalidad lineales para que sean sensibles a la estructura no lineal de los datos.

Típicamente es un problema no supervisado



Multidimensional Scaling (MDS)

- MDS busca una representación de menor dimensionalidad que respete la distancias originales entre puntos de la mejor manera posible.
- Trabaja sobre una serie de datos que distancias entre puntos (opuesto a similaridad).
- Uno pre-establece la dimensión del espacio al que queremos llegar.
- Hay datos que sólo se pueden interpretar naturalmente como distancias. Por ejemplo similaridad entre palabras o distancias entre ciudades.



Multidimensional Scaling (MDS)

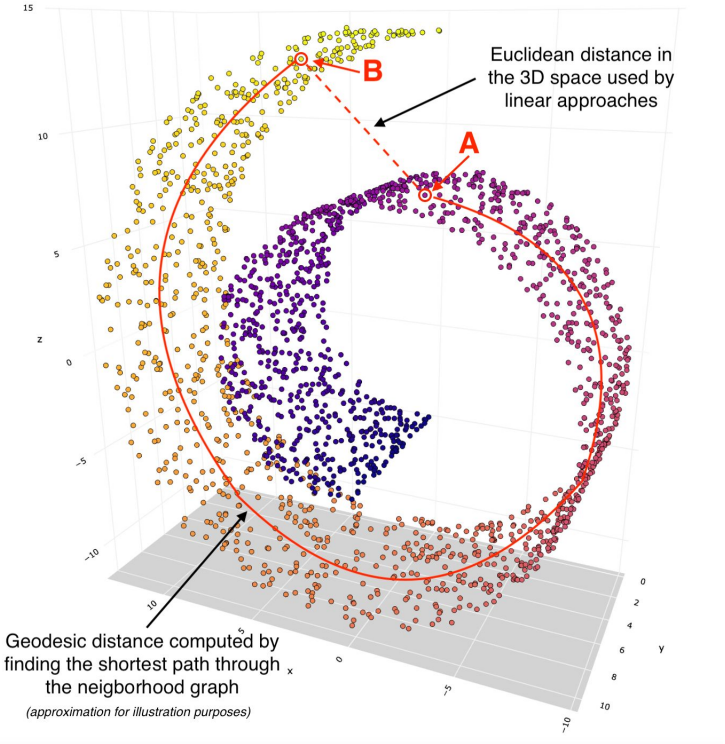
- Dos grandes ramas: la métrica (respeta identidad, simetría, y desigualdad triangular; por ej. distancias entre ciudades) y la no métrica (por ej. distancias afectivas). En el caso métrico se trata de preservar las distancias originales, mientras que en la no métrica el orden (ranking) de estas distancias.
- Para el caso más básico llamado Classic Euclidean (Torgerson') MDS tenemos resultado analítico, ya que es lo mismo que hacer PCA en matriz de disimilaridades.
- También hay soluciones iterativas que tratan de reducir el *stress*, y son generales para cualquier tipo de MDS. Es lo que usa scikitlearn.

$$Stress(X) = \sum_{i<j\leq n} w_{ij}(d_{ij}(X) - \delta_{ij})^2$$

Donde δ_{ij} son las distancias en el espacio original, $d_{ij}(X)$ las distancias en el nuevo espacio en base a los punto X y w_{ij} un conjunto de pesos que sirven para penalizar información sobre pares de puntos.

Isomap

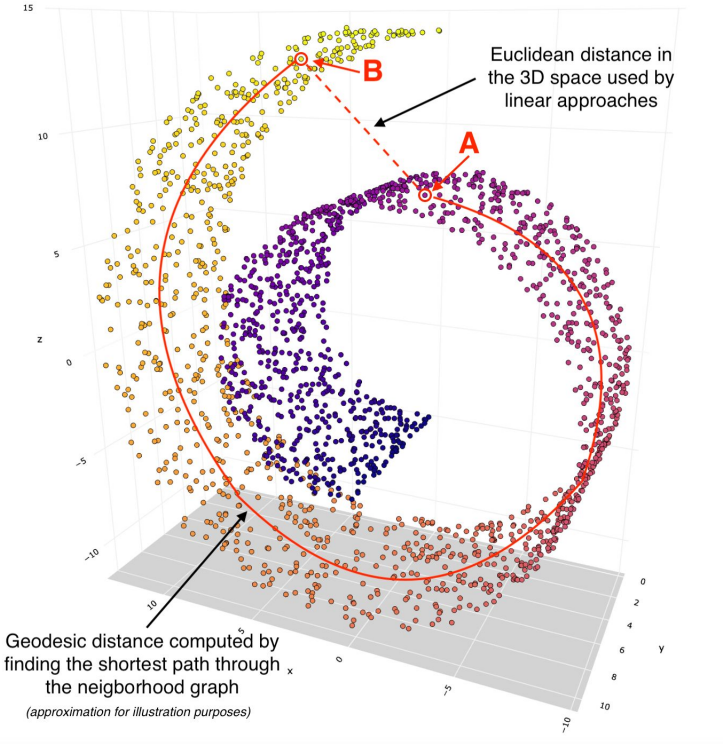
- Se basa en una representación de los datos en forma de grafos
- A cada punto se lo conecta con sus vecinos más próximos, y así se arma un grafo.
- Distancia entre dos puntos no se la define en el espacio original de datos, sino como el largo del camino más corto que conecta esos dos puntos en el grafo (la *geodésica*).



Isomap

- De esta manera recuperamos una información de la geometría de la variedad sobre la que yacen los puntos: Recorremos “caminos” sobre la variedad.

- El parámetro crítico es el número de vecinos: que no permita “saltos” en la variedad.



t-distributed Stochastic Neighbor Embedding (t-SNE)

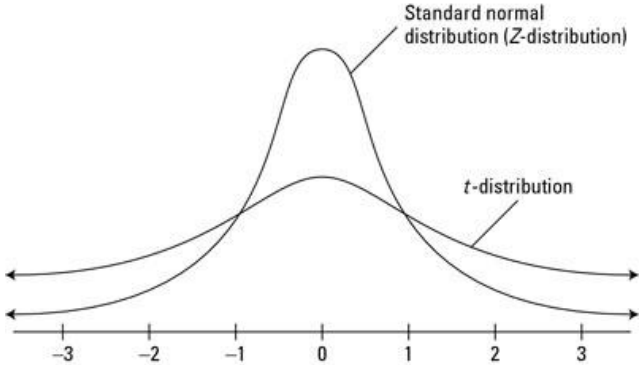
- t-SNE busca crear una proyección a un espacio de dimensión menor tratando de preservar la distribución de puntos en el espacio original, manteniendo los agrupamientos locales.

- A diferencia de MDS que trabaja en distancia de pares, t-SNE trabaja con distribuciones de distancia y preserva mejor la estructura global de los datos

Previene que se me solape todo, forzando que lo dissimilar se aleje

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)},$$

$$q_{ij} = \frac{(1+\|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k\neq i}(1+\|\mathbf{y}_i - \mathbf{y}_k\|^2)^{-1}}$$



t-distributed Stochastic Neighbor Embedding (t-SNE)

Se hacen simétricas las distribuciones $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$ y se mide la divergencia KL

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

El parámetro clave es la *perplexity*, que está relacionada al σ

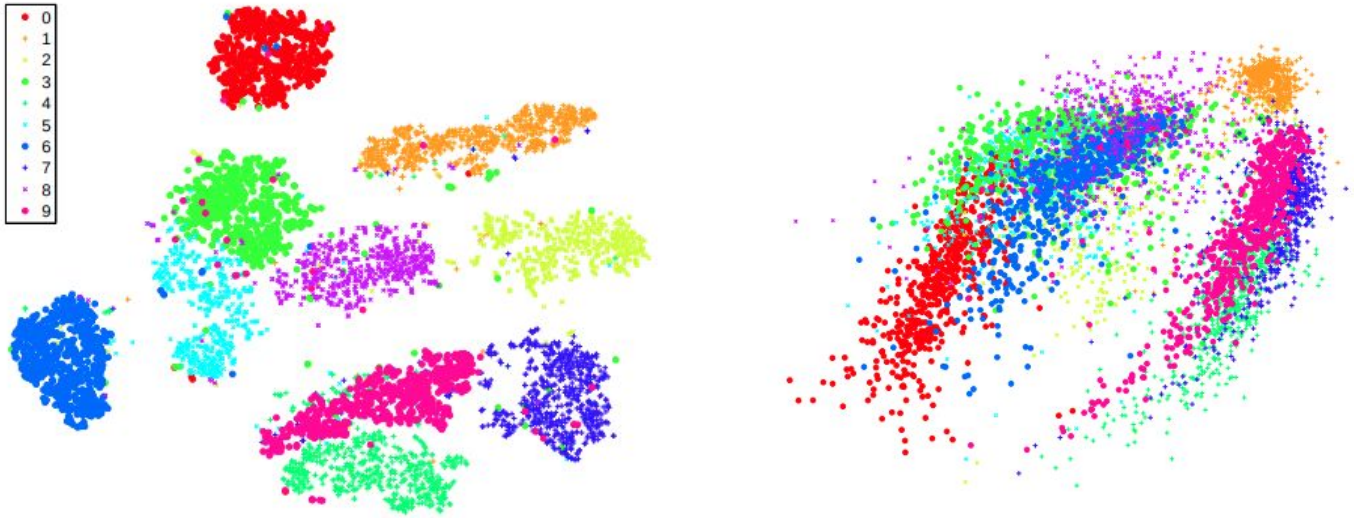
Uno de los parámetros libres de TSNE es la *perplexity* que mide en cierta forma la cantidad de vecinos que se consideran. Sirve para calcular para cada punto la desviación σ en base a la fórmula: $P = 2^{-\sum_x p(x) \log_2 p(x)}$

Tira los puntos en baja dimensión con una inicialización determinada, y va moviendo de a un punto, viendo si KL disminuye y continúa esta iteración por descenso por gradiente.

Por lo tanto es lento, y escalea mal con el número de datos.

t-distributed Stochastic Neighbor Embedding (t-SNE)

6000 dígitos de MNIST



t-SNE

Isomap

Van der Maaten & Hinton, 2008

t-distributed Stochastic Neighbor Embedding (t-SNE)

https://distill.pub/2016/misread-tsne

2 clusters, 50 puntos por cluster en 50 dimensiones reducidos a 2 dimensiones

Rango “ideal” de perplexity 5-50

perplexity < número total de puntos

Perplexity mide el balance entre la atención a lo local y a lo global, es proporcional al número de vecinos que tiene un punto

t-distributed Stochastic Neighbor Embedding (t-SNE)

https://distill.pub/2016/misread-tsne

2 clusters gaussianos, uno 10x más compacto que el otro

Ecualización de densidad local. Tiende a homogeneizar tamaño de clusters

de puntos local. En t-SNE la noción de distancia se adapta a la densidad

El tamaño del cluster en t-SNE no significa nada

t-distributed Stochastic Neighbor Embedding (t-SNE)

3 clusters gaussianos, uno más lejos que los otros

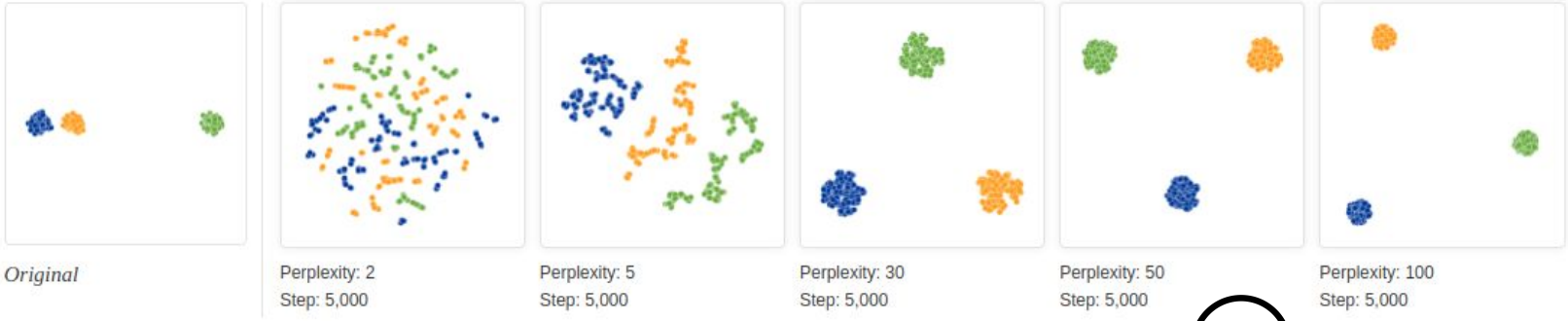
https://distill.pub/2016/misread-tsne

¿será por acá?

t-distributed Stochastic Neighbor Embedding (t-SNE)

https://distill.pub/2016/misread-tsne

Aumentamos de 50 a 200 puntos por cluster → tenemos que aumentar perplexity



Dificultad de balancear entre global y local: perplexity es un único parámetro así que si tengo clusters con diferente densidad y forma, la geometría global se rompe

t-distributed Stochastic Neighbor Embedding (t-SNE)

500 puntos en 100 dimensiones describiendo una gaussiana

Puede “alucinar estructura” que no existe. Cuidado con las granulosidades.

Uniform Manifold Approximation and Projection (UMAP)

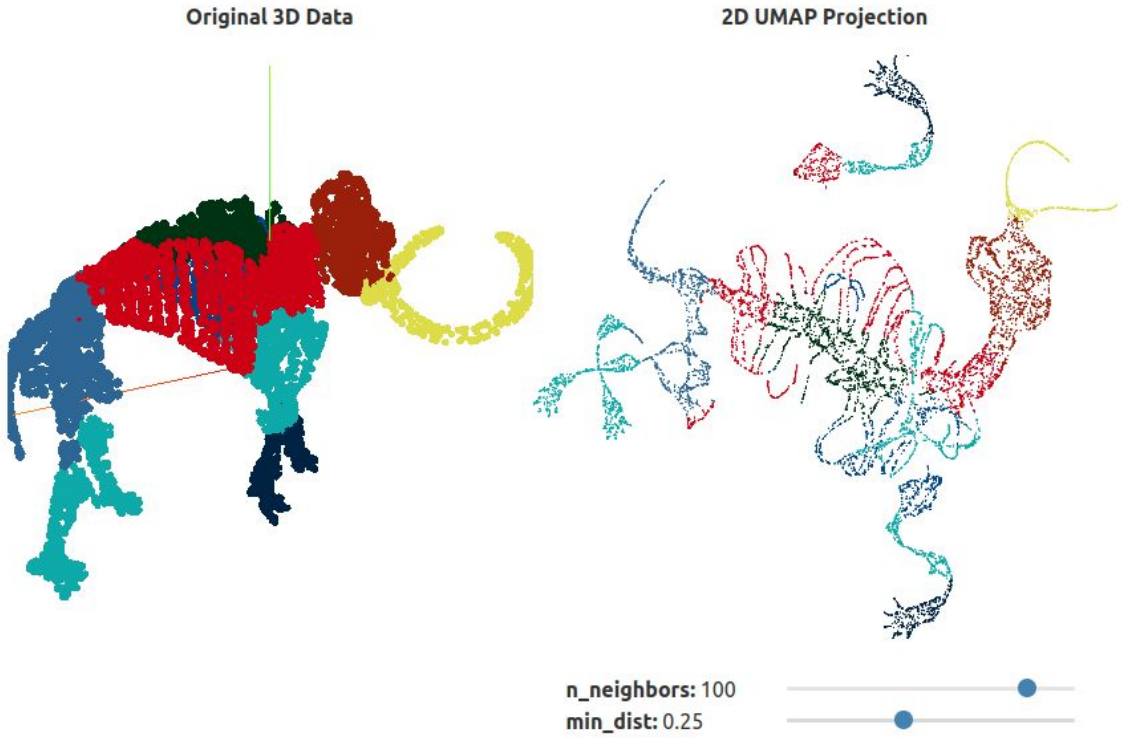
- UMAP es similar a t-SNE pero **más eficiente** computacionalmente, y puede resolver mejor **geometrías globales**.
- Construye un grafo de los datos en alta dimensión, y luego optimiza una representación en baja dimensión cuyo grafo sea lo más similar posible al de alta dimensión.
- En vez de usar perplexity para controlar la extensión de la vecindad **fijamos el número de vecinos de antemano**
- El grafo en alta dimensión es un **grafo pesado**, donde el peso del vértice representa la probabilidad de que dos puntos estén conectados
- Para establecer la conectividad, de cada punto hace crecer una esfera, y conecta puntos cuando sus esferas se tocan. El radio de esta esfera está determinado por el número de primeros vecinos que voy a elegir. De esta forma, el radio se define localmente.
- El peso del vértice disminuye a medida que necesito una esfera más grande.
- Cada punto tiene aunque sea un primer vecino, lo que **fuerza conexión global de la red**.

https://pair-code.github.io/understanding-umap/

Uniform Manifold Approximation and Projection (UMAP)

- Luego construye un **grafo de baja dimensión** de una manera similar a t-SNE, pero con trucos que lo hacen **más rápido**.
- La parte importante es entender el primer grafo de alta dimensión.
- El **parámetro clave es el número de primeros vecinos** que fijamos. Controla el balance entre local y global.
- El otro parámetro es min_dist, que es la distancia mínima de los puntos en baja dimensionalidad. Controla cuán solapadas quedan las representaciones de baja dimensión.
- A pesar de ser estocástico es **más confiable** (resultados se reproducen más que con t-SNE) por el énfasis en lo global.
- Como en t-SNE, el tamaño de los clusters puede no significar nada.

Uniform Manifold Approximation and Projection (UMAP)



Uniform Manifold Approximation and Projection (UMAP)