

Clase 7:

Reducción de dimensiones

Agenda

- Explicación: Reducción de dimensiones
- Hands-On
- Break
- Cierre

Reducción de dimensiones

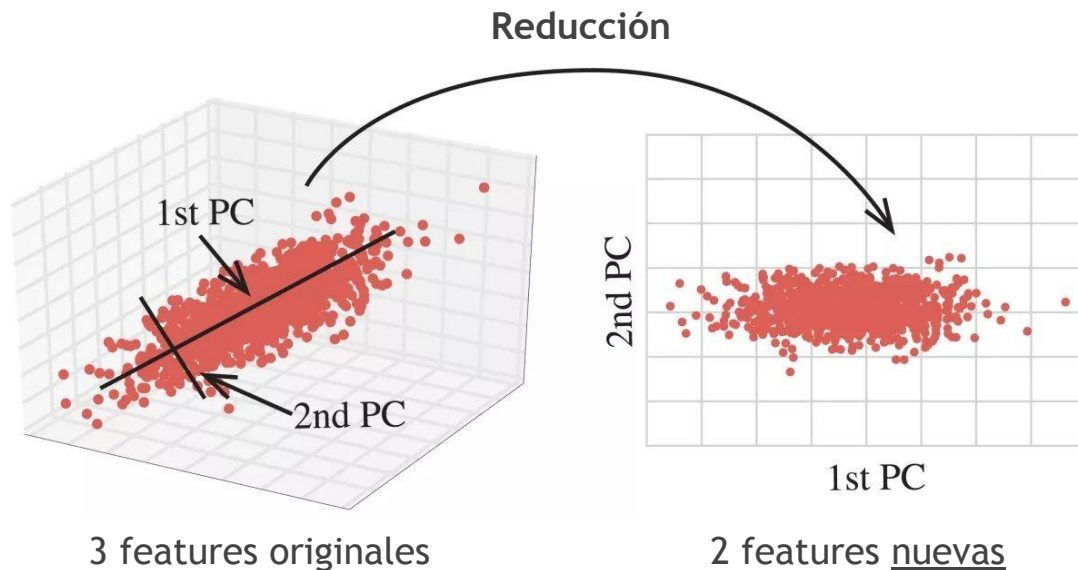
Repaso :

Reducción de la
dimensionalidad

- Clustering
- Reducción de dimensionalidad

Aprendizaje No Supervisado • Reducción de la dimensionalidad

Buscamos reducir la cantidad de features de un dataset, pero reteniendo la mayor cantidad de “información” posible.



Aprendizaje No Supervisado • Reducción de la dimensionalidad

¿Para qué sirve?

Reducir la cantidad de features en un dataset puede servir para:

- Reducir el input en un modelo de regresión o clasificación
- Compresión de archivos
- Visualización
- Detectar features relevantes en datasets
- Muchísimas mas cosas

Aprendizaje No Supervisado · Reducción de la dimensionalidad

¿Para qué sirve?

Reducir la cantidad de features en un dataset puede servir para:

- Reducir el input en un modelo de regresión o clasificación
- Compresión de archivos
- Visualización
- Detectar features relevantes en datasets
- Muchísimas mas cosas

¿Cómo se hace?

Algunos de los métodos de reducción de dimensionalidad son:

- PCA: Principal Component Analysis (usa SVD)
- MDS: Multidimensional scaling
- t-SNE: t-distributed Stochastic Neighbor Embedding
- Auto-Encoders (Se hace con Redes Neuronales)
- ●LDA: Linear Discriminant Analysis (si hay etiquetas de clases)

Aprendizaje No Supervisado · Reducción de la dimensionalidad

¿Para qué sirve?

Reducir la cantidad de features en un dataset puede servir para:

- Reducir el input en un modelo de regresión o clasificación
- Compresión de archivos
- Visualización
- Detectar features relevantes en datasets
- Muchísimas mas cosas

¿Cómo se hace?

Algunos de los métodos de reducción de dimensionalidad son:

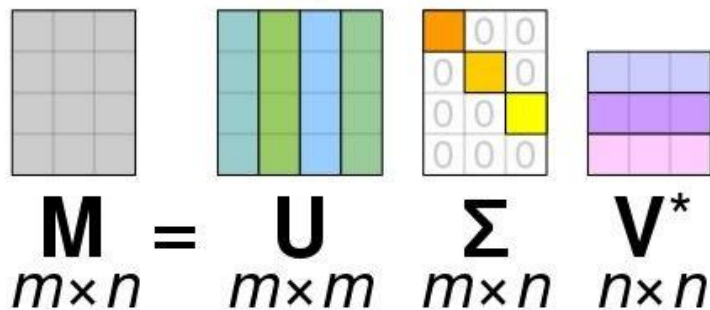
- **PCA: Principal Component Analysis (usa SVD)**
- MDS: Multidimensional scaling
- t-SNE: t-distributed Stochastic Neighbor Embedding
- Auto-Encoders (Se hace con Redes Neuronales)
- LDA: Linear Discriminant Analysis (si hay etiquetas de clases)

Aprendizaje No Supervisado

SVD (Singular Value Decomposition)

SVD • Definición

Es un método de álgebra lineal que nos permite representar cualquier matriz en términos de la multiplicación de otras 3 matrices.



The diagram illustrates the SVD decomposition of a matrix M into three matrices U , Σ , and V^* . Each matrix is represented by a grid of colored squares:

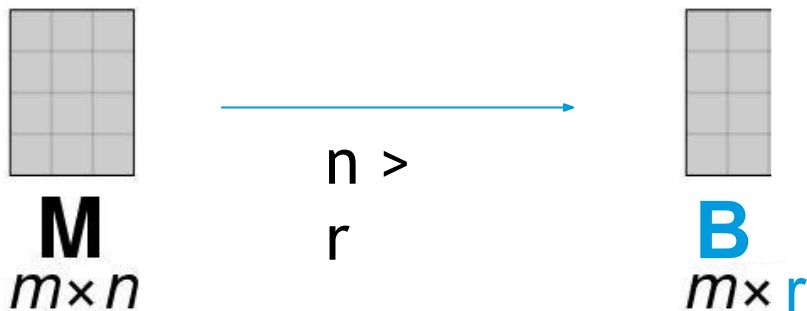
- M is a 4×4 matrix represented by a gray grid.
- U is a 4×4 matrix represented by a grid with four vertical columns of different colors: light blue, green, light blue, and green.
- Σ is a 4×4 matrix represented by a grid with a diagonal of colored squares (orange, yellow, yellow, and light blue) and zeros elsewhere.
- V^* is a 4×4 matrix represented by a grid with four horizontal rows of different colors: light blue, purple, purple, and light blue.

The equation is shown as:

$$\begin{matrix} \text{Grid} \\ \mathbf{M} \\ m \times n \end{matrix} = \begin{matrix} \text{Grid} \\ \mathbf{U} \\ m \times m \end{matrix} \begin{matrix} \text{Grid} \\ \mathbf{\Sigma} \\ m \times n \end{matrix} \begin{matrix} \text{Grid} \\ \mathbf{V}^* \\ n \times n \end{matrix}$$

SVD • ¿Para qué sirve?

Para MUCHAS COSAS. Es parte del corazón de muchos algoritmos numéricos (solución sis. lineal, pseudoinversa, etc.). En este contexto vamos a usarlo para “reducir” adecuadamente la matriz M (pasar de tener muchos features a tener menos, pero que sean buenos).



SVD · Álgebra

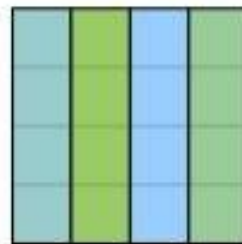
Se puede demostrar que a toda matriz M la podemos escribir como :

Matriz de Datos
(m instancias,
 n features)



$$\mathbf{M}_{m \times n} =$$

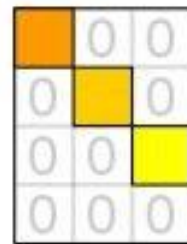
Matriz de
vectores
singulares por
izquierda



$$\mathbf{U}_{m \times m}$$

Matriz
Unitaria

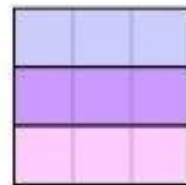
Matriz de
los valores
singulares



$$\mathbf{\Sigma}_{m \times n}$$

Matriz
Diagonal

Matriz de
vectores
singulares por
derecha



$$\mathbf{V}^*_{n \times n}$$

Matriz
Unitaria

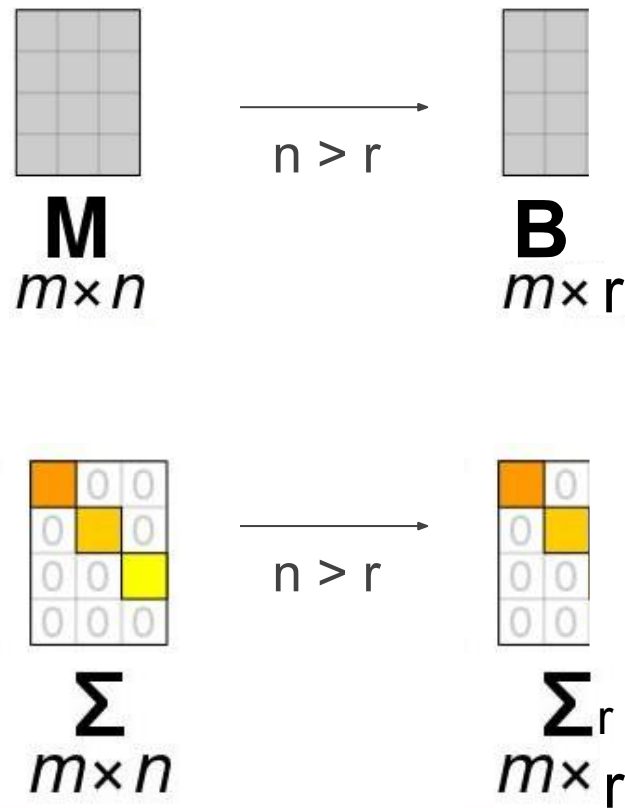
Aprendizaje No Supervisado

SVD truncado

SVD truncado

Objetivo: queremos una nueva matriz B que reemplace a M , que tenga menos columnas (menos features).

Idea de cómo lograrlo: si tomamos solo los r valores principales (elementos en la diagonal de Sigma) de valor más grande, podemos construir una matriz B que sea una “buena” reducción de M .



SVD truncado

Matriz completa: es la M original, tiene toda la información.

$$\begin{matrix} \text{4x4} & & \text{4x4} & & \text{4x4} & & \text{4x4} \\ \mathbf{M} & = & \mathbf{U} & \mathbf{\Sigma} & \mathbf{V}^* \\ m \times n & & m \times m & m \times n & n \times n \end{matrix}$$

Matriz truncada: perdimos información. Pero si tomamos un valor de r adecuado, \tilde{M} es muy parecida a M . Construimos una matriz B mas chica que M , esta es la matriz con la que vamos a trabajar.

$\downarrow n > r$

$$\begin{matrix} \text{4x4} & & \text{4x4} & & \text{4x3} & & \text{3x4} \\ \tilde{\mathbf{M}} & = & \mathbf{U} & \mathbf{\Sigma}_r & \mathbf{V}_r^* \\ m \times n & & m \times m & m \times r & r \times n \end{matrix}$$

SVD truncado

Parecidas

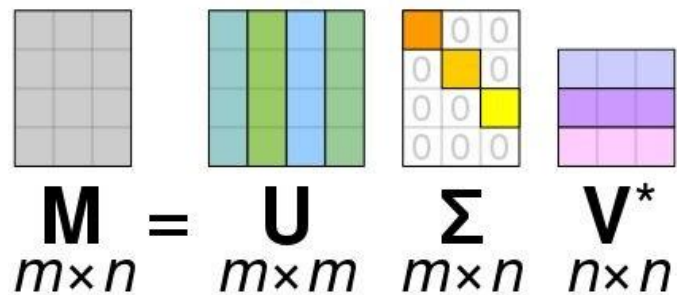


Diagram illustrating the full SVD decomposition of matrix M into three matrices: U , Σ , and V^* .

Matrix M is shown as a 4x4 grid. Matrix U is shown as a 4x4 grid with colored columns. Matrix Σ is shown as a 4x4 grid with a diagonal of colored squares and zeros elsewhere. Matrix V^* is shown as a 4x4 grid with colored rows.

$$\begin{matrix} \text{4x4} \\ \mathbf{M} \end{matrix} = \begin{matrix} \text{4x4} \\ \mathbf{U} \end{matrix} \begin{matrix} \text{4x4} \\ \mathbf{\Sigma} \end{matrix} \begin{matrix} \text{4x4} \\ \mathbf{V}^* \end{matrix}$$

Dimensions: $m \times n$, $m \times m$, $m \times n$, $n \times n$

$n > r$

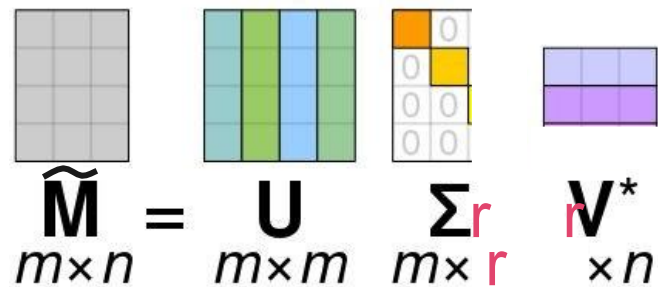


Diagram illustrating the truncated SVD decomposition of matrix M into three matrices: U , Σ_r , and V^* .

Matrix \tilde{M} is shown as a 4x4 grid. Matrix U is shown as a 4x4 grid with colored columns. Matrix Σ_r is shown as a 4x4 grid with a diagonal of colored squares and zeros elsewhere. Matrix V^* is shown as a 4x4 grid with colored rows.

$$\begin{matrix} \text{4x4} \\ \tilde{\mathbf{M}} \end{matrix} = \begin{matrix} \text{4x4} \\ \mathbf{U} \end{matrix} \begin{matrix} \text{4x4} \\ \mathbf{\Sigma}_r \end{matrix} \begin{matrix} \text{4x4} \\ \mathbf{V}^* \end{matrix}$$

Dimensions: $m \times n$, $m \times m$, $m \times r$, $r \times n$

r

SVD truncado

$$\hat{\mathbf{M}}_{m \times n} = \mathbf{U}_{m \times m} \Sigma_r_{m \times r} \mathbf{V}^*_{r \times n}$$

B
 $m \times r$

Matriz con la que vamos a trabajar en vez de \mathbf{M} , tiene la misma información que \mathbf{M} moño.

Esta matriz funciona como un diccionario para pasar del mundo de \mathbf{B} al mundo de \mathbf{M} .

SVD • Hiperparámetro r

¿Cómo podríamos elegir el valor de r ?

Una posibilidad es mirar la distancia entre M y M moño.

$$\|M - \tilde{M}\|_F = \sqrt{\sum_{ij} (M_{ij} - \tilde{M}_{ij})^2}$$

El método de SVD nos GARANTIZA que elegimos los mejores r vectores (combinaciones de features) para minimizar esta norma!

Full-Rank Dog



Rank 200 Dog



Rank 100 Dog



Rank 50 Dog



Rank 30 Dog



Rank 20 Dog



Rank 10 Dog



Rank 3 Dog

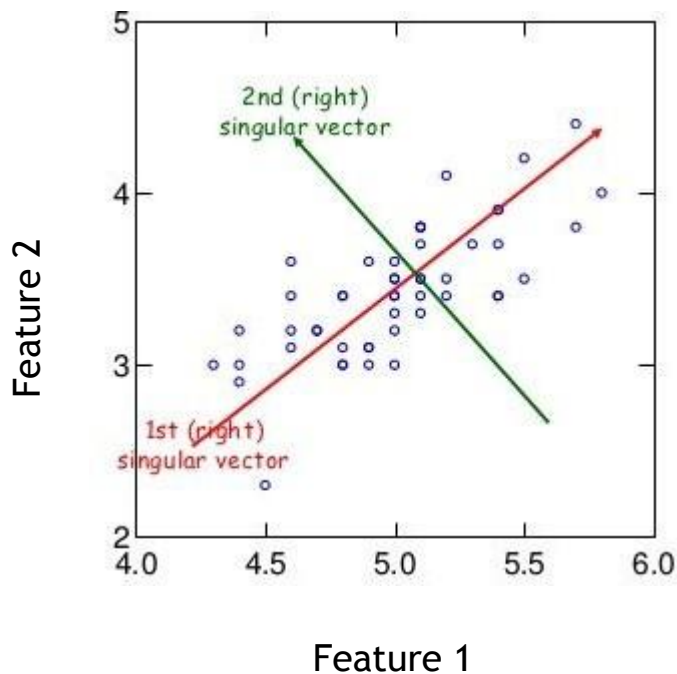


Aprendizaje No Supervisado

Representación

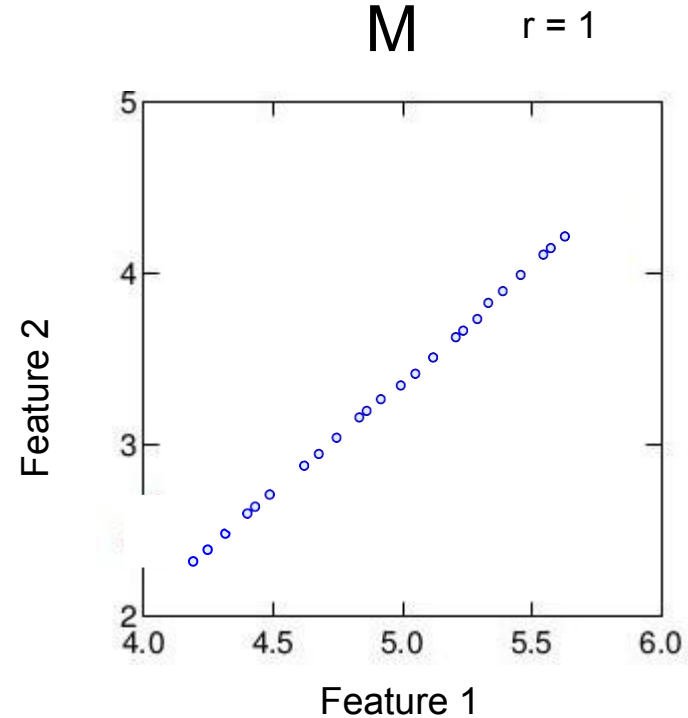
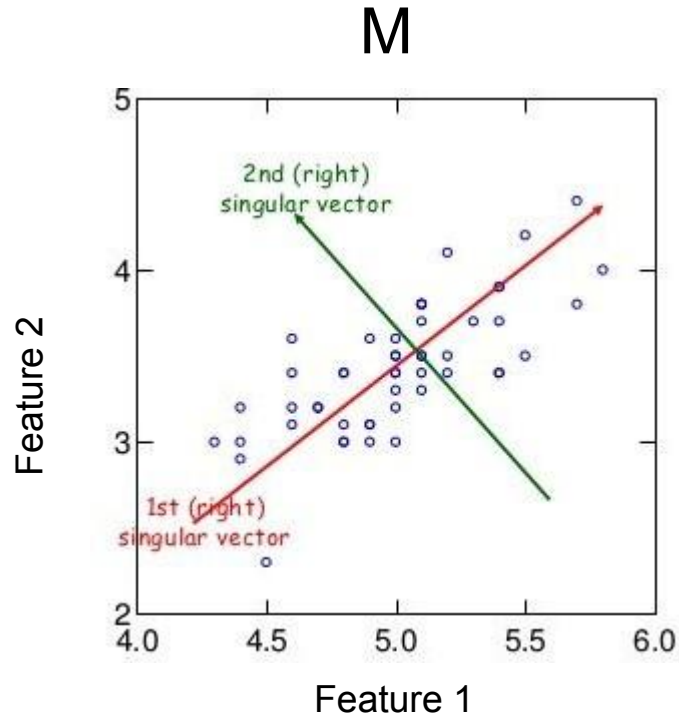
gráfica SVD

SVD • Representación gráfica



- El espacio original tiene 2 coordenadas, 2 features. Esto sirve para definir la posición de todas las instancias del dataset (cada punto azul).
- SVD nos da dos nuevos vectores, el 1er y 2do vector singular. Si usamos ambos como coordenadas, podemos definir perfecto la posición de cada punto.
- Veamos qué pasa si ahora sólo usamos el primer vector singular para definir los puntos.

SVD • Representación gráfica





Aprendizaje No Supervisado

PCA (Principal Component Analysis)

“Por lo que vimos en clase y lo que vi en los videos, SVD y PCA parecen ser lo mismo...”

Anónimo



“Casi, pero no, no son lo mismo.”

(otro) Anónimo



PCA • Definición

PCA es el método de reducción de dimensionalidad más utilizado.

¿Y PCA es muy distinto a usar
un SVD truncado?



PCA • Definición

PCA y SVD truncado son casi iguales, solo que existe una diferencia:

PCA = Centrar datos + SVD truncado

PCA • Definición

PCA y SVD truncado son casi iguales, solo que existe una diferencia:

PCA = Centrar datos + SVD truncado



Debemos sustraer la media de cada columna de Features antes de aplicar SVD truncado.

¿Por qué PCA es tan relevante?
¿Por qué no se lo ve solo como
un caso particular de SVD?



PCA • Importancia

- Matemáticamente, se puede llegar por otro camino (Matriz de covarianza)
- Tiene una interpretación muy intuitiva:

Componentes
Principales

► Direcciones de
máxima varianza

La primer componente principal está en la dirección donde los datos presentan varianza máxima.

La segunda componente principal está la segunda dirección en términos de la varianza, y así sucesivamente.



¿Cuándo uso PCA y cuándo SVD?



Comparación • PCA vs. SVD

Para muchos casos, el resultado de usar uno u otro método va a ser muy parecido. Dependiendo del problema, puede que restar la media y trabajar con la distancia de cada feature a la media en el dataset sea mejor (si no saben, lo más común es usar PCA).

Analogías	PCA	SVD
	Numero de componentes	Rango R
	Componentes principales	Vectores singulares por derecha
	Autovalores	Valores singulares
	Maximiza Varianza	Minimiza Distancia

t-SNE

- **t-SNE** (*T-distributed Stochastic Neighbor Embedding*) es un algoritmo diseñado para la visualización de conjuntos de datos de alta dimensionalidad.
- Si el número de dimensiones es muy alto, Scikit-Learn recomienda en su documentación utilizar un método de reducción de dimensionalidad previo (como PCA) para reducir el conjunto de datos a un número de dimensiones razonable (por ejemplo 50), lo que reducirá el ruido y aligerará la ejecución de t-SNE.

t-SNE

- t-SNE se ejecuta en dos pasos: en primer lugar construye una distribución de probabilidad sobre parejas de muestras en el espacio original, de forma tal que las muestras semejantes reciben alta probabilidad de ser escogidas, mientras que las muestras muy diferentes reciben baja probabilidad de ser escogidas.
- El concepto de "semejanza" se basa en la distancia entre puntos y densidad en las proximidades de un punto. Tal y como lo describen los autores:

La similaridad entre el punto x_j y el punto x_i es la probabilidad condicional de que x_i escogiese a x_j como su vecino si los vecinos fuesen escogidos proporcionalmente a su densidad de probabilidad bajo una curva gaussiana centrada en x_i .

t-SNE

- En segundo lugar, t-SNE lleva los puntos del espacio de alta dimensionalidad al espacio de baja dimensionalidad de forma aleatoria, define una distribución de probabilidad semejante a la vista en el espacio destino (el espacio de baja dimensionalidad), y minimiza la denominada **divergencia Kullback-Leibler** entre las dos distribuciones con respecto a las posiciones de los puntos en el mapa (la divergencia de Kullback-Leibler mide la similitud o diferencia entre dos funciones de distribución de probabilidad).
- Dicho con otras palabras: t-SNE intenta reproducir la distribución que existía en el espacio original en el espacio final.

t-SNE

Scikit-Learn implementa este algoritmo en [sklearn.manifold.TSNE](#).

Esta clase incluye varios parámetros que definen el comportamiento del algoritmo:

- **n_components** (2 por defecto): Dimensiones del conjunto transformado.
- **perplexity** (30 por defecto): Según la documentación de Scikit-Learn, se recomienda un valor entre 5 y 50 (mayor cuanto mayor sea el dataset), aunque se indica que el algoritmo no es muy sensible a este valor.
- **early_exaggeration** (12 por defecto): Este parámetro controla la distancia entre bloques semejantes en el espacio final. La elección de este valor no es crítico.
- **learning_rate** (200 por defecto): Habitualmente en el rango (10-1000). Si es muy elevado, los datos transformados estarán formados por una conjunto de puntos equidistantes unos de otros. Si es muy bajo, los puntos se mostrarán comprimidos en una densa nube con algunos outliers.

t-SNE

- **n_iter** (1000 por defecto): Número máximo de iteraciones para la optimización. Debería ser, por lo menos, 250.
- **metric**: métrica para la medición de las distancias.
- **method**: algoritmo a usar para el cálculo del gradiente.

A pesar de los comentarios de Scikit-Learn sobre la poca sensibilidad de los algoritmos a ciertos parámetros, lo cierto es que cambiar en una única unidad parámetros como *perplexity*, *early_exaggeration* o *learning_rate* da lugar a visualizaciones completamente diferentes.