# Reducción de dimensiones

**Organización de datos, Argerich. FIUBA**
**Fecha: 23-06-22**

# Introducción

La reducción de dimensiones se encarga de buscar una representación más compacta de un set de datos no solo para optimizar el uso de recursos sino también para otros usos: visualización de datos, compresión de imágenes, feature engineering, etc.

# SVD

# SVD

Surge de realizar una descomposición de una matriz. La SVD está íntimamente relacionada con el hecho de diagonalizar una matriz simétrica.

# Descomposición de matrices

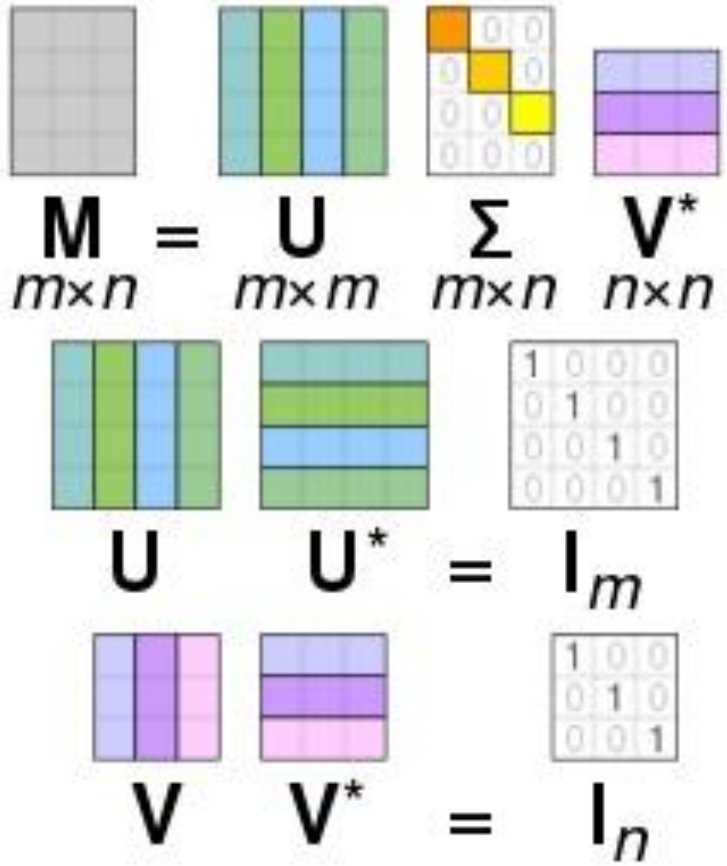
$$M = U \Sigma V$$

Las columnas de **U** son los autovectores de M por M traspuesto.

**Sigma** es la matriz más importante, es una matriz diagonal y sus valores son los llamados valores singulares, que son la raíz de los autovalores de M traspuesto por M, ordenado decrecientemente.

Las columnas de **V** son los autovectores de M traspuesto por M.

**Objetivo:** queremos lograr con una descomposición la mejor aproximación de la matriz M, pero almacenando la menor cantidad de datos.



# PCA

# PCA

Para muchas aplicaciones es de utilidad **tener la posibilidad de visualizar la información, en particular en 2 o 3 dimensiones.** Uno de los usos que se le puede dar a PCA es reducir dimensiones para visualizar datos.

Es un método lineal, utiliza planos.

**Objetivo:** encontrar las componentes que maximicen la varianza de los datos.

La varianza hace referencia a la dispersión del conjunto de datos respecto de la media. Utiliza la matriz de covarianza.

# SVD vs PCA

La SVD es en general más eficiente y numéricamente más estable que PCA dado que no hace falta construir la matriz de covarianza y es por este motivo que como método genérico lineal para reduccíon de dimensiones se prefiere siempre usar la SVD.

# MDS

# MDS

Este algoritmo busca resolver el siguiente problema: Dada una matriz con las distancias entre ciudades que se desconocen, se quieren encontrar coordenadas en el plano para estas ciudades de forma tal que las distancias entre las mismas se aproximen lo más posible a las distancias de nuestra matriz.

# ISOMAP

# ISOMAP

Es un algoritmo de Manifold Learning no-lineal.

Puede usarse para visualización de datos, como preprocesamiento o como features a agregar a los datos mismos.

El algoritmo consiste en construir un grafo a partir de los datos, calcular las distancias en dicho grafo y luego aplicar MDS para obtener una representación de los datos en pocas dimensiones.

# ISOMAP

1° Paso: crear un grafo no-dirigido en donde cada punto del set de datos va a estar conectado a sus k vecinos más cercanos.

2° Paso: una vez obtenida la matriz de distancias, se construye una nueva matriz de distancias de todos los puntos contra todos. Se aplica Floyd-Warshall o Dijkstra (devuelve la matriz de distancias mínimas).

3° Paso: se aplica MDS con k = 2 o k = 3 para obtener la representación de nuestros puntos en dos o tres dimensiones respectivamente.

# T-SNE

# T-SNE

Es un algoritmo para la representación de datos en dos y tres dimensiones.

**Objetivo:** a intentar que los puntos que estaban cerca en n dimensiones se mantengan cercanos en k dimensiones.

# T-SNE

Este algoritmo comienza calculando una probabilidad de que j sea vecino de i.

El objetivo de T-SNE es lograr que si pij es un valor cercano a 1 entonces {qij} sea un valor cercano a 1 y si {pij} es un valor cercano a cero entonces {qij} puede quedar libre.

Siendo pij la probabilidad de que j e i sean cercanos en la dimensión original y qij la probabilidad de que j e i sean cercanos en 2 o 3 dimensiones.

Esto se logra mediante la divergencia de Kullback-Leibler. El objetivo es minimizar esta divergencia.

# UMAP

# UMAP

Es una técnica de aprendizaje para la reducción de dimensiones.

UMAP se construye a partir de un marco teórico basado en geometría y topología algebraica.

Se puede aplicar a cualquier cantidad de dimensiones.

Soporta nuevos puntos, se puede usar para predecir.