Para estudiantes de Educación Básica y Media
UNIVERSIDAD DE CHILE

Clase 16: Reducción de dimensionalidad

FM849 - Programación Científica para Proyectos de Inteligencia Artificial (IA)

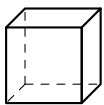
15 de enero de 2026

Motivación

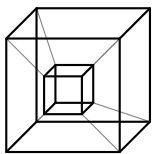
En muchos problemas de Machine Learning, los datos pueden tener una gran cantidad de dimensiones (columnas en un `pd.DataFrame`). Esto afecta principalmente en dos áreas:

- Maldición de la dimensionalidad: las distancias empiezan a ser parecidas en espacios de muchas dimensiones. Esto afecta a algoritmos basados en distancias, como k-NN y *K-Means*.
- Visualización de información: el ser humano, biológicamente, está limitado a ver en dos dimensiones y "construir" una sensación de profundidad, lo que añade una tercera dimensión. ¿Cómo interpretaríamos datos que pueden ser graficados únicamente en más de 3 dimensiones?

2D: cuadrado



3D: cubo



4D: ¿teseracto?

Maldición de la dimensionalidad

Veremos un ejemplo sencillo para entender este concepto.

Características sociales del curso

Queremos comparar características entre estudiantes de este curso. Si consideramos sólo una variable, como la `altura`, es relativamente sencillo encontrar estudiantes parecidos.

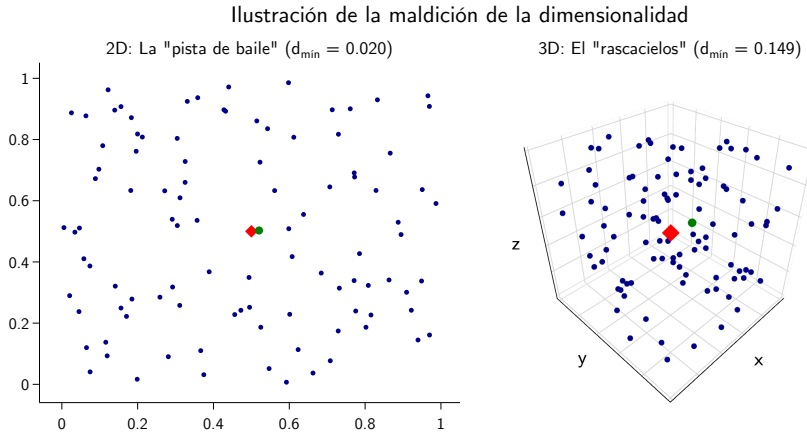
Si a la comparación le sumamos la `edad`, la probabilidad de encontrar estudiantes similares disminuye, pero aún es posible.

Si nos vamos al extremo, considerando 100 variables distintas, ya es casi imposible encontrar a dos estudiantes que sean similares en todas las dimensiones, incluso en varias de ellas.

Esto ilustra cómo, a medida que aumentamos las dimensiones, los datos tienden a dispersarse y las distancias se vuelven menos significativas.

Maldición de la dimensionalidad

Para entenderlo, ahora matemáticamente, veamos el siguiente diagrama, donde se indican los puntos centrales $(0.5, 0.5)$ y $(0.5, 0.5, 0.5)$ y el punto más cercano a ellos.



Maldición de la dimensionalidad

La razón detrás de que la distancia $d_{\text{mín}}$ aumente al pasar de un plano 2D a un espacio 3D radica en la fórmula que se ocupa para hacer esta medición.

- Para dos dimensiones, es $d = \sqrt{(\Delta x)^2 + (\Delta y)^2}$.
- Para tres dimensiones, es $d = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2}$.

En cualquier caso, el agregar la tercera componente $(\Delta z)^2$ sólo puede aumentar la distancia total d, porque $(\Delta z)^2 \geq 0$. La probabilidad de que este valor sea exactamente cero es muy baja.

Algoritmos de reducción de dimensionalidad

Enfoquémonos en los algoritmos que nos permiten reducir dimensiones. En esta clase, veremos tres de ellos:

- PCA (*Principal Component Analysis*).
- t-SNE (*t-distributed Stochastic Neighbor Embedding*).
- UMAP (*Uniform Manifold Approximation and Projection*).

Curiosidad

No es coincidencia que UMAP tenga "forzosamente" la palabra *map* en su nombre. En matemáticas, un mapa es una función que asigna elementos de un conjunto a otro, ¿y qué estamos haciendo acá?

PCA: *Principal Component Analysis*

Este algoritmo busca reducir dimensiones manteniendo la mayor cantidad de información explicada posible desde una perspectiva global.

Analogía del fotógrafo

Imaginen que están fotografiando un paisaje. El mundo real es tridimensional, pero las fotografías reposan sobre un plano bidimensional.

Un buen fotógrafo buscará el mejor ángulo para capturar la esencia del paisaje en su fotografía, que usualmente se obtendrá desde puntos altos, maximizando la cantidad de información visual que puede transmitir en dos dimensiones.

Podemos pensar que cada elemento del paisaje aporta información. Hay algunos que aportan más que otros. Por ejemplo, si sacamos la fotografía mirando a una pared, no capturaremos casi nada de la esencia original.

PCA: *Principal Component Analysis*

La idea que subyace tras este algoritmo es, en esencia, la misma. Hay variables que aportan más información que otras.

Ejemplo geométrico

Si tuviésemos variables de largo, anchura y altura, y además el área y el volumen, estas últimas no aportarían información adicional, pues pueden ser calculadas **directamente** a partir de las otras tres variables.

PCA y correlaciones

Naturalmente, variables que estén altamente correlacionadas entre sí aportan información que llamamos redundante. Una forma de identificar las variables que pueden ser eliminadas es analizando la matriz de correlación del problema.

Aplicación de PCA

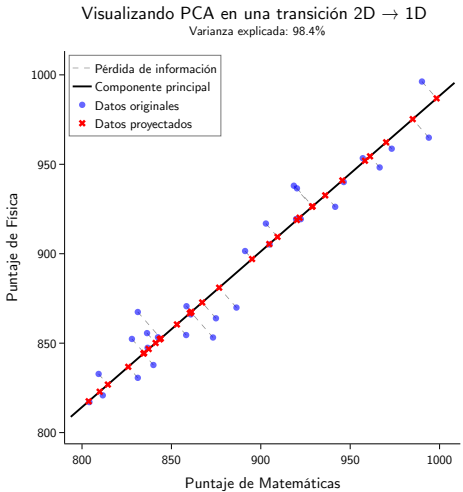
Usando la implementación de PCA de Scikit-Learn, podemos ver un ejemplo sencillo de cómo funciona este algoritmo.

```
>>> import pandas as pd
>>> from sklearn.decomposition import PCA

>>> data = pd.DataFrame({
...     'Math': [850, 871, 944, ...],
...     'Physics': [824, 880, 915, ...]
... })

>>> pca = PCA(n_components=1)
>>> pca_result = pca.fit_transform(data)

>>> pca.explained_variance_ratio_
```



t-SNE: *t-distributed Stochastic Neighbor Embedding*

Recordemos la analogía de la fotografía para PCA. Esta analogía sugiere que PCA reduce las dimensiones con un enfoque global, porque un buen ángulo de fotografía trataría de capturar la mayor parte del paisaje (p. ej., mirando desde arriba, sin detalles locales).

En contraste, *t*-SNE funciona preservando las relaciones locales de los puntos en altas dimensiones. Es decir, si dos puntos están cerca en el espacio original, deberían permanecer cerca en el espacio reducido.

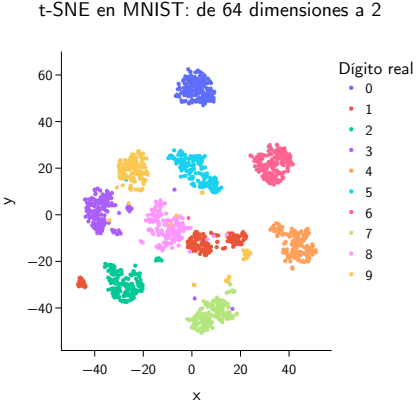
¿Por qué se parece tanto a una clusterización?

Es cierto que podemos vincular este algoritmo con la idea de clusterización, donde queremos que puntos similares (cercanos) permanezcan juntos. La diferencia radica en que *t*-SNE **no** asigna etiquetas (es decir, no dice tú eres del grupo 1, aquel del grupo 2, etc.), simplemente reduce las dimensiones.

Aplicación de *t*-SNE: MNIST

Veamos un ejemplo sencillo usando el conjunto de datos MNIST, que contiene imágenes de dígitos escritos a mano (0-9).

A la izquierda, se muestran algunas imágenes del conjunto de datos original, y a la derecha, el resultado de aplicar *t*-SNE (`from sklearn.manifold import TSNE`) para reducir las dimensiones a 2.



UMAP: *Uniform Manifold Approximation and Projection*

UMAP (`import umap`) es como el "híbrido" entre PCA y t-SNE. Busca preservar tanto las relaciones locales como las globales en los datos al reducir dimensiones.

Volviendo a la vista del paisaje...

Si lo pensamos como elementos que podemos identificar en el día a día:

- PCA sería la vista satelital: puede visualizar todo, pero sin enfocarse en detalles específicos.
- t-SNE sería el modo "Street View", que baja directamente a la calle y muestra detalles locales, sin preocuparse por la vista global.
- UMAP sería como una vista panorámica desde un dron, que es capaz de capturar, dada la altura que tiene, tanto la vista global como detalles locales.

Detalles importantes de estos algoritmos

- UMAP tiene hiperparámetros que permiten ajustar el equilibrio entre la preservación local y global. Esto permite incluso imitar el resultado que daría t-SNE o PCA dependiendo de los valores escogidos.
- PCA y UMAP son algoritmos que aprenden reglas: tal cual lo vimos en regresión lineal, se aprenden parámetros (que en ese caso eran coeficientes) que permiten transformar nuevos datos.
- t-SNE resuelve un acertijo único por ejecución: no permite transformar nuevos datos directamente, dado que corresponderían a "otro acertijo".