Topological Methods in Machine Learning

Análisis Topológico de Datos para ML

Curso corto de 6 horas

Carlos Ramírez

Escuela Politécnica Nacional

3 de diciembre de 2025

Agenda general

1. Motivación y panorama
2. Topología básica para TDA
3. Homología (visión intuitiva)
4. Homología persistente: idea y filtraciones
5. Diagramas de persistencia y distancias
6. De diagramas a features para ML
7. Aplicaciones y cierre

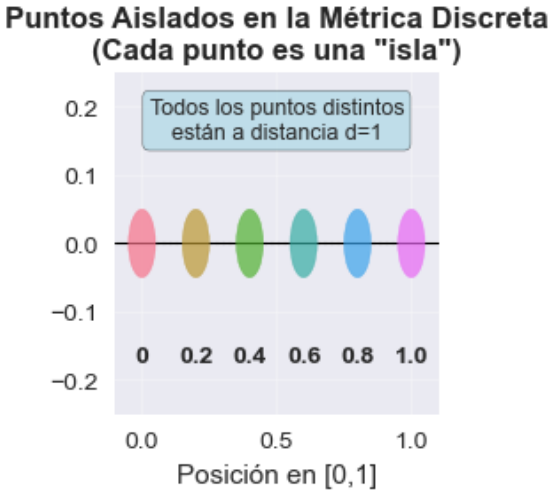
¿Por qué topología en Machine Learning?

- Datos modernos: altos dimensionales, no lineales, con estructura compleja.
- Los métodos clásicos (PCA, clustering estándar, etc.) capturan geometría local, pero no siempre la estructura global.
- La Topological Data Analysis (TDA) estudia:
  - Componentes conexas (0-huecos),
  - Bucles (1-huecos),
  - Cavidades (2-huecos y más).
- Objetivo del curso:
  - Entender homología persistente y Mapper.
  - Ver cómo se integran en tareas de ML.

Espacio topológico y métrica

- Pensamos un conjunto de datos X como un conjunto de puntos con noción de cercanía.
- Un **espacio métrico** (X, d) permite definir vecindades y continuidad.
- Ejemplos en TDA:
  - Nube de puntos en $\mathbb{R}^N$ (distancia euclidiana u otra).
  - Grafos: nodos vecinos $\Rightarrow$ relación de cercanía natural.
  - Imágenes: píxeles vecinos.
- La **topología** codifica la información de vecindad, no el tamaño exacto.

Caso: Métrica discreta



Complejos simpliciales

- Un **símplice** generaliza el concepto de triángulo:
  - 0-símplice: vértice.
  - 1-símplice: arista.
  - 2-símplice: triángulo lleno.
  - 3-símplice: tetraedro lleno, etc.
- Un **complejo simplicial** es una unión de símplices que se intersectan solo en símplices completos comunes.
- Son la estructura discreta sobre la que calculamos homología.

Ejemplo

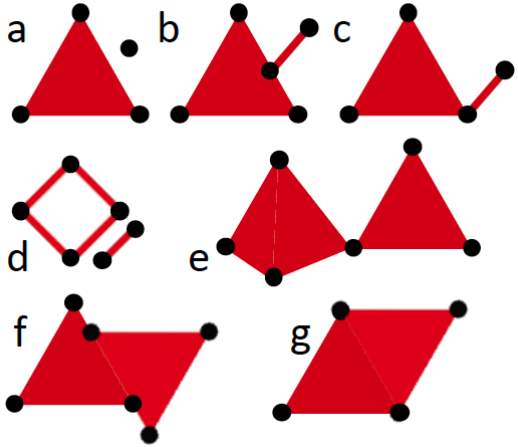


Figura: Algunos Símplices y otros que no lo son

¿Por qué la dimensión es sutil en topología?

- En topología trabajamos con una familia especial de espacios: las **variedades** (*manifolds*).
- La idea clave:

Variedad k–dimensional

Un espacio topológico donde, alrededor de cada punto, existe un vecindario que se parece (es homeomorfo) a una **bola** en $\mathbb{R}^k$.

- Importante: la dimensión se decide mirando la estructura *local*, no cómo se ve el objeto en el espacio que lo contiene.

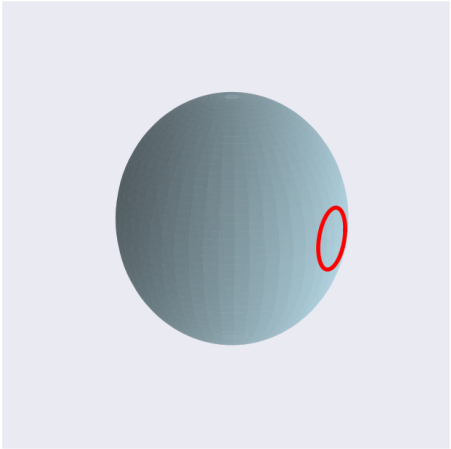
Dimensión intrínseca vs espacio ambiente

- No nos interesa el espacio ambiente (por ejemplo, $\mathbb{R}^2$ o $\mathbb{R}^3$), sino las propiedades *intrínsecas* del objeto: **el "material" del que está hecho**.
- Ejemplo: el **círculo**.
  - Lo dibujamos como curva cerrada en $\mathbb{R}^2$.
  - Pero topológicamente es 1–dimensional: cualquier vecindario pequeño de un punto del círculo se parece a un segmento abierto de recta.
- Un modo de pensarlo:
  - Una hormiga que camina sobre el círculo sólo tiene dos opciones: avanzar o retroceder; vive en un mundo 1–dimensional.

Ejemplo: la esfera como superficie 2–dimensional

- La **esfera** (la superficie de una pelota) se dibuja en $\mathbb{R}^3$, pero es una variedad de dimensión 2.
- Razón: un vecindario pequeño de cualquier punto de la esfera se parece a un pedazo de plano en $\mathbb{R}^2$.
- De nuevo, piensa en los "habitantes":
  - Una hormiga que vive en la superficie de la esfera puede moverse libremente en dos direcciones independientes.
- Otros ejemplos de variedades 2–dimensionales:
  - el **toro** (un donut),
  - una superficie de género 2 (dos "agujeros").
- Existen variedades de dimensión mayor que 3, pero es difícil visualizarlas.

Ejemplo



Variedades con frontera

- Hasta ahora, las variedades no tenían borde.
- Una **variedad k–dimensional con frontera** es un espacio en el que cada punto tiene un vecindario que se parece a:
  - una bola abierta en $\mathbb{R}^k$ (puntos interiores), o
  - una parte del **semiespacio** $\mathbb{R}^k_+ = \{x \in \mathbb{R}^k \mid x_k \geq 0\}$ (puntos de la frontera).
- La **frontera** ∂M está formada precisamente por los puntos del segundo tipo.
- Localmente, cerca de la frontera, el espacio se parece a "media bola".

Ejemplos de frontera

- **Esfera** (superficie):
  - Es una variedad 2–dimensional *sin frontera*.
  - Una hormiga puede moverse indefinidamente sin encontrar un borde.
- **Disco unitario** $D \subset \mathbb{R}^2$:
  - Es una superficie 2–dimensional *con frontera*.
  - Su frontera es la circunferencia unitaria C.
  - Una hormiga que vive dentro del disco eventualmente llega a C, más allá no puede continuar.
- **Bola sólida** $B \subset \mathbb{R}^3$:
  - Es una variedad 3–dimensional con frontera.
  - Su frontera es la esfera S (la superficie de la pelota).

Interpretación “desde dentro”

- La frase “la esfera no tiene frontera, pero la bola sólida sí” puede parecer contraintuitiva al comienzo.
- Vista desde dentro:
  - Los habitantes de la **esfera** viven en su superficie; nunca encuentran un borde, sólo más superficie.
  - Un ratón que vive dentro de la **bola** sí encuentra un límite: la superficie es la frontera de su mundo.
- La frontera es, por tanto, un concepto intrínseco: depende del espacio donde se vive, no de cómo lo incrustamos en otro.

Propiedades de la frontera

- Principio fundamental:

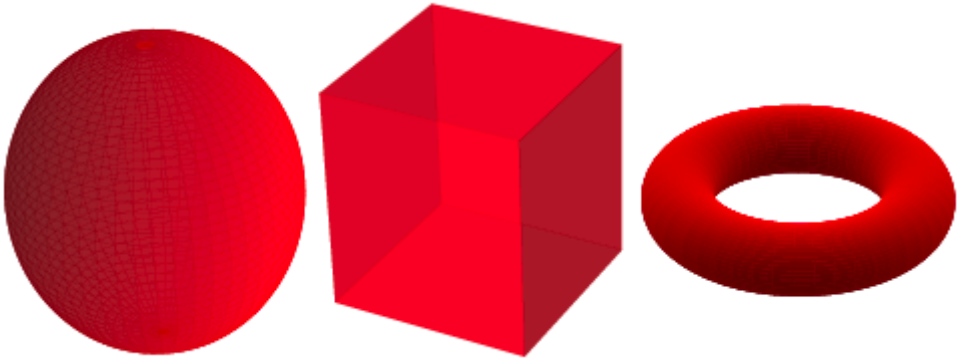
La frontera de la frontera es vacía

$$\partial(\partial X) = \varnothing.$$

- Además, si M es una variedad k–dimensional con frontera, entonces su frontera ∂M es, típicamente, una variedad $(k-1)$–dimensional *sin frontera*.
- Ejemplos:
  - Frontera del disco D (2D con frontera): la circunferencia C, que es una variedad 1D sin frontera.
  - Frontera de la bola sólida B (3D con frontera): la esfera S, que es una superficie 2D sin frontera.
- Esta relación entre dimensión y frontera será crucial al hablar de homología.

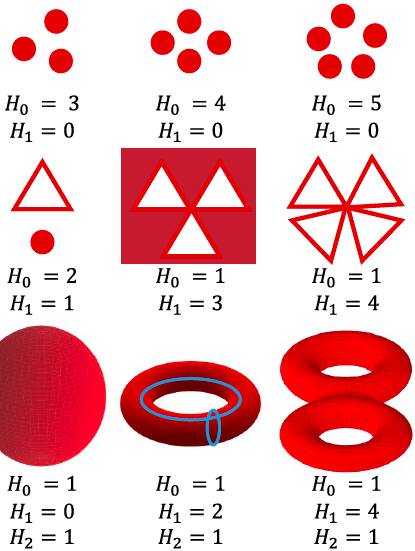
Equivalencia topológica e invariantes

- Dos espacios son **equivalentes topológicamente** si uno se deforma continuamente en el otro (homeomorfismo).
- Ejemplo:
  - Esfera y cubo: mismos "huecos".
  - Esfera vs toro: el toro tiene un "agujero" esencial adicional.
- **Invariantes topológicos**:
  - Número de componentes, números de Betti.
  - Característica de Euler, grupos de homología.
- La TDA usa invariantes computables sobre representaciones discretas (complejos simpliciales).



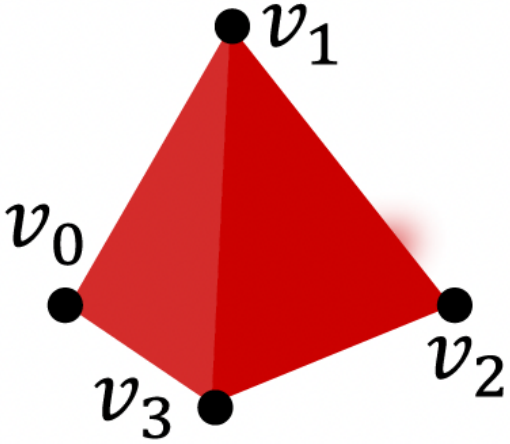
Huecos y números de Betti

- Homología cuenta "huecos" en distintas dimensiones.
- Para un espacio X:
  - β_0: número de componentes conexas.
  - β_1: número de bucles esenciales (no contraíbles).
  - β_2: número de cavidades tridimensionales, etc.
- Ejemplos:
  - Esfera: $\beta_0 = 1$, $\beta_1 = 0$, $\beta_2 = 1$.
  - Toro: $\beta_0 = 1$, $\beta_1 = 2$, $\beta_2 = 1$.
- Estos números son invariantes topológicos.


$H_0 = 3$
$H_1 = 0$
$H_0 = 4$
$H_1 = 0$
$H_0 = 5$
$H_1 = 0$
$H_0 = 2$
$H_1 = 1$
$H_0 = 1$
$H_1 = 3$
$H_0 = 1$
$H_1 = 4$
$H_0 = 1$
$H_1 = 0$
$H_2 = 1$
$H_0 = 1$
$H_1 = 2$
$H_2 = 1$
$H_0 = 1$
$H_1 = 4$
$H_2 = 1$

Cadenas, ciclos y bordes (idea)

- Representamos subestructuras como combinaciones formales de símplices:
  - **cadenas** k-dimensionales: sumas de k-símplices.
- El operador borde ∂_k lleva cadenas k-dimensionales a su frontera $(k-1)$-dimensional.
- **Ciclos**: cadenas con borde cero ($\partial_k c = 0$).
- **Bordes**: cadenas que son borde de una cadena de dimensión superior.
- Homología $H_k = Z_k / B_k$ distingue ciclos "reales" de los que son frontera de algo.


v_1
v_0
v_2
v_3

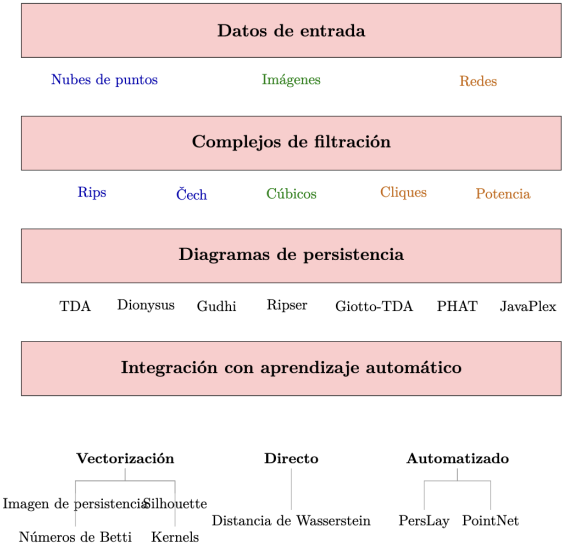
¿Qué es la homología persistente?

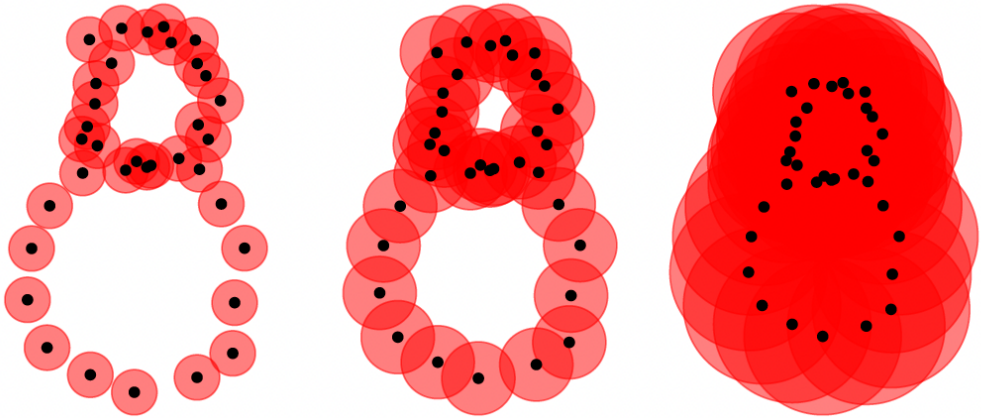
- En lugar de un solo complejo, consideramos una **filtración**:

$$K_1 \subset K_2 \subset \cdots \subset K_n.$$

- A medida que avanzamos en la filtración, aparecen y desaparecen:
  - Componentes,
  - Bucles,
  - Cavidades.
- Homología persistente registra el "tiempo de vida" de cada rasgo topológico.
- Resultado: un resumen multiescala de la forma de los datos.

Pipeline PH


Datos de entrada
Nubes de puntos
Imágenes
Redes
Complejos de filtración
Rips
Čech
Cúbicos
Cliques
Potencia
Diagramas de persistencia
TDA
Dionysus
Gudhi
Ripser
Giotto-TDA
PHAT
JavaPlex
Integración con aprendizaje automático
Vectorización
Imagen de persistencia
Silhouette
Números de Betti
Kernels
Directo
Distancia de Wasserstein
Automatizado
PersLay
PointNet



(a) $\epsilon = 0.4$ (b) $\epsilon = 0.7$ (c) $\epsilon = 2$

Filtraciones para nubes de puntos

- Datos: nube de puntos $X = \{x_1, \ldots, x_m\}$ en un espacio métrico.
- Definimos una familia de radios $\varepsilon_1 < \varepsilon_2 < \cdots < \varepsilon_n$.
- Para cada ε construimos un complejo:
  - **Čech**: intersección no vacía de bolas.
  - **Rips**: todos los puntos de un símplice a distancia $\leq \varepsilon$ entre sí.
- Obtenemos una filtración de complejos simpliciales que aproxima las vecindades $N_\varepsilon(X)$.

Método 1 – Complejos de Rips

Para un conjunto de puntos $X \subset \mathbb{R}^N$ y un parámetro $r > 0$, el complejo de Rips (también llamado complejo de Vietoris–Rips) es el complejo simplicial abstracto $R_r(X)$, donde un k–símplice

$$\sigma = [x_{i_0}, x_{i_1}, \ldots, x_{i_k}] \in R_r(X)$$

si y sólo si

$$d(x_{i_m}, x_{i_n}) < r \quad \text{para todo } 0 \leq m, n \leq k.$$

En otras palabras, para $r > 0$, si $k+1$ puntos son mutuamente r–cercanos entre sí, entonces forman un k–símplice en $R_r(X)$.

Método 2 – Complejos de Čech

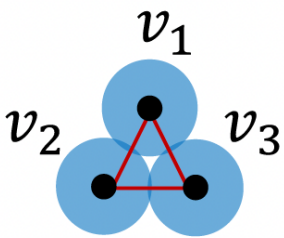
De forma similar, para un conjunto de puntos $X \subset \mathbb{R}^N$ y un parámetro $r > 0$, el complejo de Čech es el complejo simplicial abstracto $\check{C}_r(X)$, donde un k–símplice

$$\sigma = [x_{i_0}, x_{i_1}, \ldots, x_{i_k}] \in \check{C}_r(X)$$

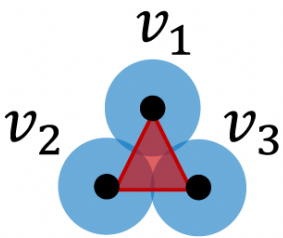
si y sólo si

$$\bigcap_{j=0}^{k} B_r(x_{i_j}) \neq \emptyset.$$

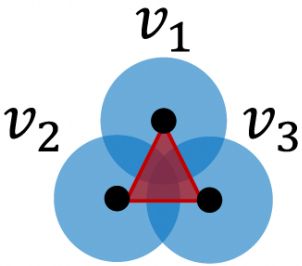
Aquí, la condición para construir un k–símplice es diferente: se exige que la intersección de las bolas de radio r de los $k+1$ puntos sea no trivial (es decir, no vacía).



(a) Cech Complex



(b) Rips Complex



(c) Cech Complex

Teorema del nervio

Enunciado

Sea X un espacio topológico y $\mathcal{U} = \{U_i\}_{i \in I}$ una familia de subconjuntos abiertos de X que forma un recubrimiento bueno, es decir, toda intersección finita no vacía de elementos de $\mathcal{U}$ es contráctil.

Entonces el complejo simplicial nervio $N(\mathcal{U})$ tiene el mismo tipo de homotopía que X. En particular, X y $N(\mathcal{U})$ son homotópicamente equivalentes.

Definition

El *nervio* $N(\mathcal{U})$ del recubrimiento $\mathcal{U}$ es el complejo simplicial cuyas vértices son los conjuntos U_i, e incluye un símplex $[U_{i_0}, \ldots, U_{i_k}]$ siempre que la intersección

$$U_{i_0} \cap \cdots \cap U_{i_k} \neq \varnothing.$$

Maquinaria de HP para nubes de puntos con complejos de Rips

Algorithm 1 Maquinaria de homología persistente para nubes de puntos con complejos de Rips

Input: Nube de puntos P, métrica de distancia d, conjunto de umbrales $\{\varepsilon_i\}_{i=0}^{n}$

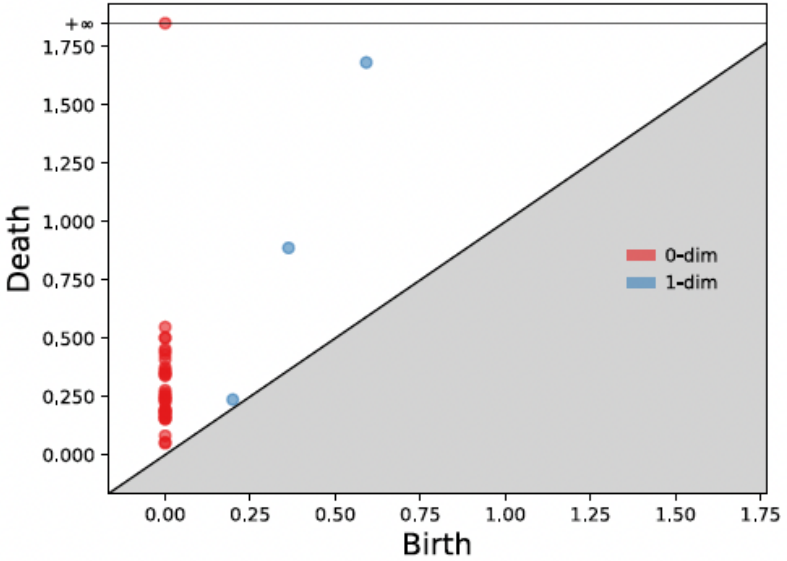
Output: Vector topológico $\vec{\beta}(P, d, \{\varepsilon_i\})$
$V \leftarrow \{v \in P\}$ ▷ el conjunto de vértices no cambia con la filtración
for $i = 0$ **hasta** n **do** ▷ índice de filtración
    $E_i \leftarrow \{(u, v) \mid u, v \in P \text{ y } d(u, v) \leq \varepsilon_i\}$ ▷ aristas con distancia $\leq \varepsilon_i$
    $\mathcal{R}_i \leftarrow (V, E_i)$ ▷ complejo de Rips en ε_i
end for
for $k = 0$ **hasta** 1 **do** ▷ dimensiones topológicas
    Calcular el diagrama de persistencia $PD_k(\{\mathcal{R}_i\}_{i=0}^{n})$
    Obtener la vectorización $\vec{\beta}_k$ a partir de $PD_k(\{\mathcal{R}_i\}_{i=0}^{n})$
end for
return Concatenación de $\vec{\beta}_0$ y $\vec{\beta}_1$, denotada como $\vec{\beta}(P, d, \{\varepsilon_i\}) = \vec{\beta}_0 \parallel \vec{\beta}_1$

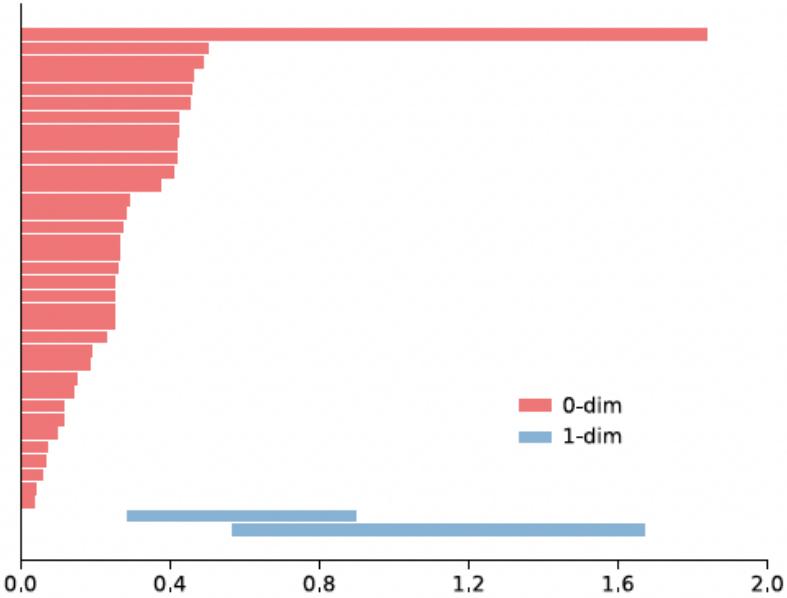
Filtraciones para imágenes y grafos

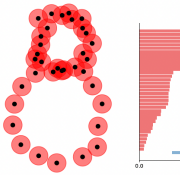
- **Imágenes**:
  - Complejos cúbicos o subnivel de una función de intensidad.
  - Umbrales crecientes: aparecen y desaparecen componentes y agujeros.
- **Grafos**:
  - Complejos de cliques (clique complexes) a partir de un grafo ponderado.
  - Filtraciones por peso: se añaden aristas y cliques conforme baja un umbral.
- En todos los casos: filtración = *versión multiescala* de los datos.

Diagramas de persistencia

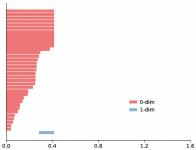
- Para cada clase de homología registramos:
  - **nacimiento**: nivel de la filtración donde aparece.
  - **muerte**: nivel donde deja de existir.
- Representación gráfica:
  - Plano nacimiento–muerte, cada punto es un "hueco".
  - Puntos cerca de la diagonal: ruido.
  - Puntos lejos de la diagonal: estructura relevante.
- Un diagrama por dimensión k (componentes, bucles, cavidades).


+∞
1,750
1,500
1,250
1,000
0,750
0,500
0,250
0,000
Death
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Birth
0-dim
1-dim

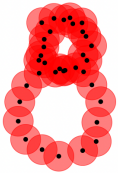

0-dim
1-dim
0,0
0,4
0,8
1,2
1,6
2,0



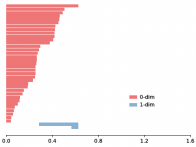
(a) $\epsilon = 0.4$



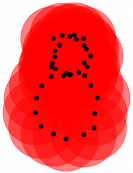
(b) PB up to $\epsilon = 0.4$



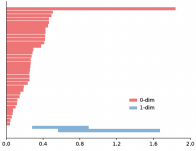
(c) $\epsilon = 0.7$



(d) PB up to $\epsilon = 0.7$



(e) $\epsilon = 2$



(f) Persistence barcode

Distancias entre diagramas

- Para comparar datasets usamos distancias entre diagramas.
- Distancias comunes:
  - **bottleneck**,
  - **Wasserstein**.
- Intuición:
  - Se busca un "emparejamiento" entre puntos de dos diagramas que minimice el costo global.
- Propiedad importante: estabilidad frente a perturbaciones pequeñas en los datos.

Distancia de Wasserstein entre diagramas de persistencia

Sean $PD(X^+)$ y $PD(X^-)$ los diagramas de persistencia de X^+ y X^- (omitimos dimensiones). Denotamos

$$PD(X^+) = \{q_j^+\} \cup \Delta, \qquad PD(X^-) = \{q_l^-\} \cup \Delta,$$

donde Δ es la diagonal (ciclos triviales) con multiplicidad infinita, y $q_j^\pm = (b_j^\pm, d_j^\pm)$.
Sea $\varphi : PD(X^+) \to PD(X^-)$ una biyección. La presencia de Δ garantiza que existan biyecciones incluso si $|\{q_j^+\}| \neq |\{q_l^-\}|$.

La **distancia de Wasserstein de orden** p, $\mathcal{W}_p$, se define como

$$\mathcal{W}_p\big(PD(X^+), PD(X^-)\big) = \min_{\varphi} \left(\sum_j \left\| q_j^+ - \varphi(q_j^+) \right\|_p^p \right)^{1/p}, \qquad p \in \mathbb{Z}^+.$$

Donde:

$$\big\|(x_1, y_1) - (x_2, y_2)\big\|_\infty = \max\big\{|x_1 - x_2|,\ |y_1 - y_2|\big\}$$

es la norma supremo. Cuando $p = \infty$, obtenemos la **distancia bottleneck**:

$$\mathcal{W}_\infty\big(PD(X^+), PD(X^-)\big) = \min_{\varphi} \max_j \left\| q_j^+ - \varphi(q_j^+) \right\|_\infty.$$

Vectorizaciones clásicas

- Para usar TDA en ML necesitamos vectores de dimensión fija.
- Estrategias:
  - **Imágenes de persistencia**: discretizar el plano nacimiento–muerte en píxeles.
  - **Landscapes y silhouettes**: funciones que resumen el diagrama.
  - **Kernels** sobre diagramas: entradas para SVM u otros métodos kernel.
- Elección de parámetros (resolución, rango de ejes) afecta el desempeño.

Integración con modelos de ML

- **Pipeline clásico**:

$$\text{Datos} \to \text{Filtración} \to \text{Diagramas} \to \text{Vectorización} \to \text{Modelo ML}$$

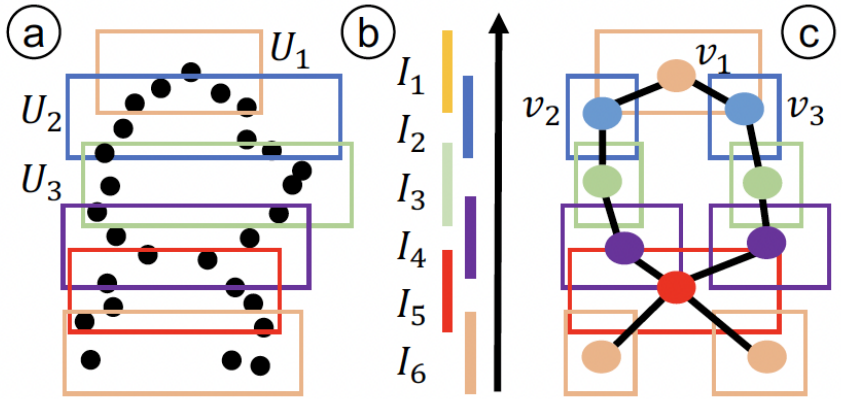
- Uso con:
  - Clasificación, regresión, clustering.
  - Redes neuronales (capas especializadas para diagramas).
- Ventaja: capturar estructura global que otros métodos no ven.

Idea del algoritmo Mapper

- Entrada: nubes de puntos (o datos con métrica) y una función “filtro”.
- Pasos básicos:
  1. Elegir un filtro (por ejemplo, proyección PCA, densidad, etc.).
  2. Cubrir el rango del filtro con intervalos solapados.
  3. Para cada intervalo:
     - tomar los puntos que caen en la preimagen,
     - aplicar un método de clustering local.
  4. Construir un grafo: nodos = clusters, aristas = intersección no vacía de clusters.
- Resultado: un grafo que resume la forma global de los datos.

Hiperparámetros en Mapper

- Elecciones importantes:
  - Filtro: qué función resalta mejor la estructura de interés.
  - Número de intervalos y grado de solapamiento.
  - Algoritmo de clustering y su escala (por ejemplo, k-means, DBSCAN).
- Interpretación:
  - Componentes del grafo, ramas, ciclos.
  - Relacionar regiones del grafo con etiquetas o variables externas.


a
U_1
U_2
U_3
b
I_1
I_2
I_3
I_4
I_5
I_6
c
v_1
v_2
v_3

Algoritmo Mapper - Parte 1/2

Algorithm 2 Construcción de nodos

Require: X, $f : X \to \mathbb{R}$, cubrimiento $\mathcal{I} = \{I_k\}$

Ensure: Conjunto de nodos V

1: Inicializar $V \leftarrow \emptyset$
2: Calcular $f(X)$
3: **for** cada $I_k \in \mathcal{I}$ **do**
4: $\quad U_k \leftarrow \{x \in X \mid f(x) \in I_k\}$
5: $\quad$ Clusters $\{C_i\} \leftarrow$ agrupar(U_k)
6: $\quad$ Crear nodo v_i para cada cluster C_i
7: $\quad V \leftarrow V \cup \{v_i\}$
8: **end for**
9: **return** V

Algoritmo Mapper - Parte 2/2

Algorithm 3 Construcción de aristas

Require: Conjunto de nodos V con clusters asociados $\{C_i\}$
Ensure: Grafo $G = (V, E)$

```
Inicializar E ← ∅
for cada par v_i, v_j ∈ V do
    if C_i ∩ C_j ≠ ∅ then
        E ← E ∪ {(v_i, v_j)}
    end if
end for
G ← (V, E)
return G
```

Aplicaciones destacadas

- **Nubes de puntos**: reconocimiento de formas y análisis de geometría.
- **Grafos de transacciones**: detección de anomalías en criptomonedas.
- **Imágenes médicas**: diagnóstico de cáncer en imágenes histopatológicas.
- **Descubrimiento de fármacos**: análisis topológico de espacios químicos.
- **Genómica**: genotipado de cáncer con Mapper sobre datos de RNA-seq.

Bibliotecas de software para homología persistente

En la columna **Datos**, **P** indica soporte para nubes de puntos (Point cloud), **I** para datos de imagen (Image) y **N** para datos de red (Network/grafo).

Biblioteca	Datos	Lenguaje	Característica notable	URL del código
Giotto-TDA	P,N,I	Python	Integración con Scikit	`https://github.com/giotto-ai/giotto-tda`
Gudhi	P,I	C++/Python	Bien establecida	`https://github.com/GUDHI/gudhi-devel`
Dionysus2	P	C++/Python	Tipos de persistencia	`https://github.com/mrzv/dionysus`
Ripser.py	P	Python	Integración con Scikit-TDA	`https://github.com/scikit-tda/ripser.py`
Ripser	P	C++	Ejecutable independiente	`https://github.com/Ripser/ripser`
JavaPLEX	P	Matlab/Java	Bien establecida	`https://github.com/appliedtopology/javaplex`
TDA	P	R	Integración con el ecosistema de R	`https://cran.r-project.org/package=TDA`
PHAT	P	Python/C++	Bien establecida	`https://bitbucket.org/`

Resumen y discusión

- TDA aporta una visión global de la estructura de los datos.
- Homología persistente y Mapper son herramientas clave.
- Integración natural con ML: extracción de características topológicas + modelos clásicos o profundos.
- Preguntas abiertas: escalabilidad, automatización de hiperparámetros, interpretabilidad.

Pueden consultar el código y diapositivas aquí: github.com/cero1979/Repositorio

Bibliografía notable en Análisis Topológico de Datos

- G. Carlsson, "Topology and data", *Bull. Amer. Math. Soc.* **46** (2009), 255–308.
- H. Edelsbrunner, J. L. Harer, *Computational Topology: An Introduction*, American Mathematical Society, 2010.
- A. Zomorodian, G. Carlsson, "Computing persistent homology", *Discrete & Computational Geometry* **33** (2005), 249–274.
- R. Ghrist, "Barcodes: The persistent topology of data", *Bull. Amer. Math. Soc.* **45** (2008), 61–75.
- F. Chazal, V. de Silva, M. Glisse, S. Oudot, *The Structure and Stability of Persistence Modules*, Springer, 2016.
- S. Oudot, *Persistence Theory: From Quiver Representations to Data Analysis*, American Mathematical Society, 2015.
- T. K. Dey, Y. Wang, *Computational Topology for Data Analysis*, Cambridge University Press, 2022.