Facultad de Matemáticas
Departamento de Álgebra

Análisis topológico de datos

Javier Perera Lago

Doble Grado en Matemáticas
y Estadística

A.A. 2019-2020

Abstract

Topological data analysis is a branch of computational topology which uses algebra to obtain topological features from a data set. It has many applications in computer vision, shape description, time series analysis, biomedicine, drug design...

The first step to learn topological information from data is to build a filtration of nested simplicial complexes estimating the global structure of the data set at different scales of precision. The second step is to study the evolution of the simplicial homology along this filtration, using a tool from algebraic topology called persistent homology. When the filtration is indexed by a single parameter, we can describe discrete and complete invariants for persistent homology with coefficients over a field, such as barcodes, persistent diagrams or persistent landscapes. Unfortunately, we don't have a simple classification for persistent homology when the filtration is indexed by two or more parameters. The problem of classifying multiparameter persistent homology is being widely studied nowadays, and we present some proposals for it.

One of the big problems of large sets is the presence of noisy, wrong or incomplete data. Although persistent homology is stable under small perturbation, we still need to give a statistical frame in order to separate insignificant features from topological signal and build some hypothesis tests applied to topological features.

Another usual problem of large sets is the high dimensionality. Even if we have captured the main topological caracteristics of the data, we aren't usually able to fully understand its structure. In order to reduce dimensionality and obtain a better visualization we present the Mapper algorithm, which groups the data using a filter function and summarizes the set as a graph or a low dimensional simplicial complex.

Índice general

Capítulo 1

Complejos simpliciales

El punto de partida en el análisis topológico de datos es un conjunto de puntos $S = \{x_1, \cdots, x_N\} \in \mathbb{R}^d$, aunque también pueden estar en un espacio métrico general. Siempre se asume uno de los siguientes escenarios: o bien el conjunto S viene muestreado de una variedad topológica no observable $\mathbb{M}$, o bien es una muestra aleatoria simple de una distribución de probabilidad P cuyo soporte es $\mathbb{M}$. El primer escenario asume que nuestros datos son correctos y están medidos con precisión pero podrían ser más numerosos, mientras que el segundo asume que los datos tienen una componente aleatoria y pueden estar sujetos a error. En cualquier caso, nuestro objetivo es deducir las propiedades topológicas de $\mathbb{M}$ a partir del conjunto S. El primer paso para ello será construir un complejo simplicial sobre los puntos de S que reconstruya $\mathbb{M}$ de forma aproximada.

1.1. Primeras definiciones

Definición 1.1. *Un complejo simplicial abstracto es un conjunto no vacío K tal que todos sus elementos son conjuntos finitos y si $\sigma \in K$, y además $\tau \subset \sigma$, entonces se verifica que $\tau \in K$. En particular, $\emptyset \in K$. Cada elemento de un complejo simplicial se llama símplice.*

En otras palabras, es una generalización del concepto de grafo donde se permiten relaciones más complejas que las aristas, es decir, con tres elementos o más [16].

Definición 1.2. *Un símplice con $n+1$ elementos tiene dimensión n y se llama un n-símplice. En particular $\emptyset$ tiene dimensión -1. La dimensión de un complejo simplicial es el máximo de las dimensiones de sus símplices.*

Dados dos símplices $\sigma, \tau \in K$ tales que $\tau \subset \sigma$, se dirá que τ es una cara de σ y que σ es una cocara de τ. Un símplice se dirá maximal si no tiene cocaras propias en K.

Definición 1.3. *Un subcomplejo de K es un subconjunto $L \subset K$ que satisface la definición de complejo simplicial. Un tipo especial de subcomplejo es el n-esqueleto de K, formado por todos los símplices de K con dimensión n o menor.*

Existe una versión geométrica de la definición de complejo simplicial, que nos hará más fácil las construcciones sobre S [49].

Definición 1.4. *Un símplice geométrico σ de dimensión n es la envoltura convexa de $\{a_0, \cdots, a_n\} \subset \mathbb{R}^d$ afínmente independientes. La envoltura convexa de un subconjunto de $\{a_0, \cdots, a_n\}$ es una cara de σ. Un complejo simplicial geométrico K es una colección de símplices geométricos en $\mathbb{R}^d$ tal que cualquier cara de $\sigma \in K$ está en K y la intersección de dos símplices de K es o bien vacía o bien una cara común. La dimensión de un complejo simplicial geométrico es el máximo de las dimensiones de sus símplices.*

En esta versión geométrica, un símplice de dimensión 0 es un vértice, un símplice de dimensión 1 es un segmento o arista, uno de dimensión 2 es un triángulo, uno de dimensión 3 es un tetraedro y así sucesivamente. Esta nomenclatura geométrica se aplica también en los complejos abstractos.

1.2. Recubrimiento y nervio

Para construir algunos de los complejos simpliciales que definiremos más tarde necesitamos los conceptos de recubrimiento y nervio [16].

Definición 1.5. *Dado un conjunto finito S de puntos euclídeos, un recubrimiento de S es una familia de conjuntos no vacíos $U = \{U_i\}_{i\in I}$ tal que $S \subset \bigcup_{i\in I} U_i$. Si todos los U_i son abiertos (cerrados) no vacíos, se dice que U es un recubrimiento abierto (cerrado) de S.*

Definición 1.6. *Dado un recubrimiento $U = \{U_i\}_{i\in I}$, su nervio es un complejo simplicial abstracto N construido sobre el conjunto de índices I, de forma que $\emptyset \in N$ y cada subconjunto no vacío de índices $J \subset I$ pertenece a N si y solo si $\bigcap_{j\in J} U_j \neq \emptyset$. En particular, el símplice unitario i pertenece a N para todo $i \in I$.*

Dicho de otra forma, el nervio asigna un vértice a cada conjunto del recubrimiento y después crea una arista por cada pareja de conjuntos que se corten, un triángulo por cada terna de conjuntos que se corten y así sucesivamente.

Un motivo por el que esta construcción resulta tan útil es la existencia del siguiente teorema, que garantiza una relación de homotopía entre el recubrimiento y su nervio [45].

Teorema 1.1 (del nervio)**.** *Sea $\mathbb{M}$ un espacio topológico inmerso en $\mathbb{R}^d$ y sea $U = \{U_i\}_{i\in I}$ un recubrimiento abierto numerable para $\mathbb{M}$. Supongamos que para cada $\emptyset \neq J \subset I$ se tiene que $\bigcap_{j\in J} U_j$ es contráctil o vacío. Entonces el nervio N de U es homotópicamente equivalente a $\mathbb{M}$.*

Recordemos que dos espacios topológicos X e Y se dicen homotópicamente equivalentes si existen dos aplicaciones continuas $f : X \to Y$, $g : Y \to X$ tales que $g \circ f$ es homotópica (se puede deformar continuamente) a Id_X y $f \circ g$ es homotópica a Id_Y.

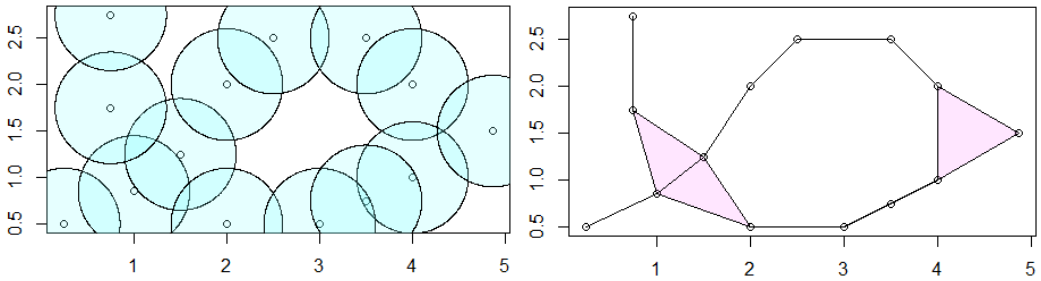
1.3. Complejo de Čech

El complejo de Čech es el complejo simplicial más elemental que se puede construir sobre S, y el que describe de una forma más natural cómo este conjunto recubre $\mathbb{M}$ [16].

Definición 1.7. *Dados S y un cierto $\varepsilon > 0$, sea el recubrimiento abierto $U_\varepsilon = \{B(x, \varepsilon)|x \in S\}$. Definimos el complejo de Čech $C(S, \varepsilon)$ como el nervio de U_ε. Definimos también $C(S, 0) = \emptyset$*

Los vértices de $C(S, \varepsilon)$ son los puntos de S, y dado un subconjunto de ellos, generan un símplice si la intersección de sus respectivas bolas es no vacía. Como estas bolas son convexas y contráctiles, el complejo de Čech es homotópicamente equivalente a U_ε según el teorema del nervio.

Es claro que si $0 < \delta < \varepsilon$, entonces $C(S, \delta) \subset C(S, \varepsilon)$. Además para un cierto $M > 0$ suficientemente grande, el complejo $C(S, M)$ será un $(N-1)$-símplice. Esta construcción es muy útil desde el punto de vista teórico gracias al siguiente resultado [16]:



(a) Recubrimiento U_ε de S para $\varepsilon = 0.6$ (b) Complejo de Čech $C(S, 0.6)$

Figura 1.1: Construcción de un complejo de Čech a partir de su recubrimiento. Nótese que en la parte inferior derecha hay tres puntos alineados, que definen un triángulo en el complejo aunque no quede representado visualmente.

Teorema 1.2. *Sea $\mathbb{M}$ una variedad Riemanniana compacta inmersa en $\mathbb{R}^d$. Entonces existe un cierto $e > 0$ tal que para todo $\varepsilon \leq e$ se tiene que $C(\mathbb{M}, \varepsilon)$ es homotópicamente equivalente a $\mathbb{M}$. Es más, existe un subconjunto finito $V \subset \mathbb{M}$ tal que $C(V, \varepsilon)$ también es homotópicamente equivalente a $\mathbb{M}$.*

Es decir, que bajo la hipótesis de que los puntos de S estén bien distribuidos podemos conseguir un complejo homotópicamente equivalente a $\mathbb{M}$. No obstante, su tamaño final desaconseja usarlo en la práctica.

En el caso extremo en que se obtiene un $(N-1)$-símplice, se obtienen un total de 2^N símplices de dimensiones entre -1 y $N-1$, que en general es mucho mayor que la dimensión del espacio donde se encuentran nuestros puntos.

Por otra parte, su computación no solo es compleja sino que además no es exacta sino aproximada. Comprobar si una pareja de puntos define una arista es sencillo pues basta ver si su distancia es menor que 2ε, lo cual es fácil de verificar usando la matriz de distancias de S. En cambio si queremos comprobar si tres o más vértices definen un símplice, debemos probar que la intersección de sus bolas con radio ε tenga al menos un punto en común. Esto es equivalente a probar que existe una bola con radio ε tal que contenga a dichos vértices. Lo cual nos obliga a calcular el problema de la mínima bola envolvente (o MEB por sus siglas en inglés de *"Minimum Enclosing Ball"*) para cada posible símplice. Los algoritmos que resuelven este problema en general no calculan el radio exacto de esta bola, sino una sobreestimación suya [71]. Por tanto, corremos el riesgo de obtener un complejo mayor que el real.

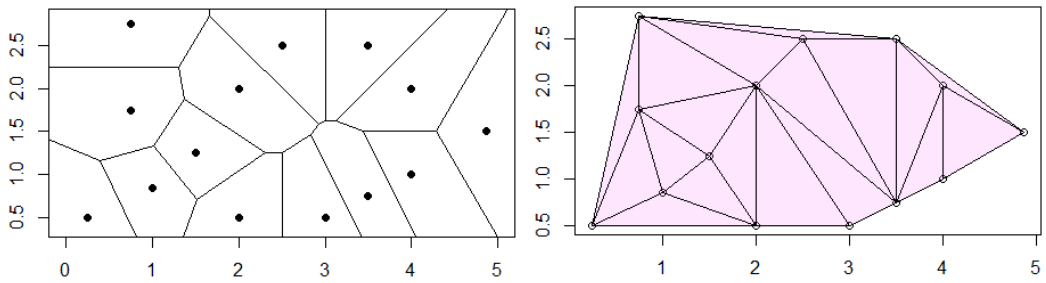
A continuación describiremos nuevos complejos que traten de mantener las buenas propiedades topológicas del complejo de Čech y a la vez resuelvan sus dificultades computacionales.

1.4. Complejo *alpha*

Esta estructura es una variante del complejo de Čech donde limitamos su dimensión a la del espacio ambiente. En primer lugar tomamos el conjunto de puntos S y construimos su diagrama de Voronoi.

Definición 1.8. *La región de Voronoi de un punto $x \in S$ es el conjunto $R(x) := \{y \in \mathbb{R}^d | d(x,y) \leq d(x',y), \forall x' \in S \setminus \{x\}\}$. Estas regiones definen un recubrimiento cerrado de S, y su nervio es el complejo de Delaunay $D(S)$.*

Las regiones de Voronoi nos sirven para limitar la expansión de las bolas de radio ε, dando lugar a un subcomplejo del complejo de Čech llamado el complejo *alpha* [38].



(a) Diagrama de Voronoi de S (b) Complejo de Delaunay $D(S)$

Figura 1.2: Construcción de un complejo de Delaunay a partir de su diagrama de Voronoi asociado

Definición 1.9. *Dados S y un cierto $\varepsilon > 0$, sea el recubrimiento $U_\varepsilon = \{B(x,\varepsilon) \cap R(x) | x \in S\}$. Definimos el complejo alpha $A(S,\varepsilon)$ como el nervio de este recubrimiento. Definimos también $A(S,0) = \emptyset$.*

Por construcción del recubrimiento, $A(S,\varepsilon) \subset C(S,\varepsilon)$. Es claro que si $0 < \delta < \varepsilon$, entonces $A(S,\delta) \subset A(S,\varepsilon) \subset D(S)$.

Aunque el complejo *alpha* tenga menos símplices que el de Čech, realmente no perdemos información porque $C(S,\varepsilon)$ y $A(S,\varepsilon)$ son homotópicamente equivalentes [4]. Pero el complejo *alpha* tiene su dimensión acotada por la del espacio ambiente siempre que los puntos se encuentren en posición general.

El complejo de Delaunay y por tanto el complejo *alpha* se pueden computar de forma eficiente para 2 y 3 dimensiones, mientras que en dimensiones superiores la computación es más difícil [10]. No obstante, la perspectiva de obtener un complejo de dimensión acotada hace factible asumir el coste computacional del algoritmo.

1.5. Complejo de Vietoris-Rips

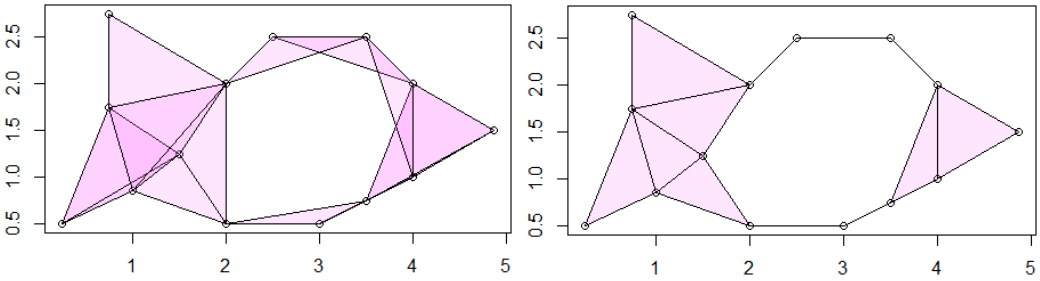
El complejo de Vietoris-Rips es una alternativa al complejo de Čech que no limita su dimensión ni conserva su homotopía pero sí aligera su tiempo de construcción. No se basa en un recubrimiento sino en un grafo. Para su construcción hemos de definir lo que es un complejo de cliques.

Definición 1.10. *Sea $G = (V,E)$ un grafo. El complejo de cliques de G es un complejo simplicial abstracto cuyos vértices son los de V y cuyos símplices son todos los subconjuntos $W \subset V$ tales que $(W, \{\{u,v\} \in E | u,v \in W\})$ es un subgrafo completo (o clique) de G.*

Dicho de otra forma, es un complejo cuyo 1-esqueleto coincide con G e incluye todos los posibles símplices que se pueden construir a partir del mismo [43]. La construcción de un complejo de cliques es muy veloz y eficiente [74], y está en la base de varios métodos diferentes.

Definición 1.11. *Dados S y un cierto ε, sea $G = (V,E)$ un grafo donde $E = S$ y $V = \{\{u,v\} | u,v \in S, u \neq v, d(u,v) \leq \varepsilon\}$. Definimos el complejo de Vietoris-Rips $VR(S,\varepsilon)$ como el complejo de cliques de G.*

La principal diferencia entre el complejo *VR* y el complejo de Čech está a la hora de determinar sus símplices de orden igual o superior a 2. En ambos casos las aristas se extraen analizando la matriz de distancias. Dado este conjunto de aristas, sus cliques definen sus símplices potenciales. Mientras que en el complejo de Čech debemos comprobar si estos cliques son en efecto



(a) Complejo $C(S, 0.8)$ (b) Complejo $A(S, 0.8)$

Figura 1.3: Comparativa entre los complejos de Čech y *alpha*.

símplices, en el complejo VR no es necesaria la comprobación y todos ellos entran en el complejo, simplificando mucho la computación. Es fácil ver que $C(S, \varepsilon) \subset VR(S, 2\varepsilon) \subset C(S, 2\varepsilon)$.

Este complejo es una aproximación del complejo de Čech que no conserva necesariamente sus propiedades topológicas exactas. Al igual que este, el complejo VR también puede alcanzar una dimensión igual a $N - 1$ en un caso extremo. Por ello también es posible limitar su dimensión de antemano y calcular solo un n-esqueleto suyo [4].

La sencillez de su construcción y la posibilidad de controlar su dimensión hacen que este complejo sea uno de los más usados, especialmente para conjuntos de datos con alta dimensionalidad.

1.6. Complejos *witness*

Esta familia de complejos está pensada para reducir sustancialmente el tamaño del objeto final. Se asume que si los puntos de S están distribuidos de forma homogénea sobre $\mathbb{M}$, no es necesario utilizar todos los puntos del conjunto de datos, sino que una fracción de ellos basta para reconstruir su topología.

Para ello hacemos una partición del conjunto S, quedando por un lado el conjunto L de marcadores y por otro el conjunto W de testigos (*witnesses* en inglés). Los marcadores de L serán los vértices del complejo resultante mientras que los testigos de W servirán para decidir qué símplices incluir [29].

Los marcadores se pueden seleccionar aleatoriamente o con el método *maxmin*. El método *maxmin* es inductivo, y comienza escogiendo un $l_1 \in S$ al azar. Si ya hemos escogido los marcadores $l_1, l_2, \ldots, l_i$, escogemos $l_{i+1} \in S \setminus \{l_1, l_2, \ldots, l_i\}$ como el punto que maximiza la función:

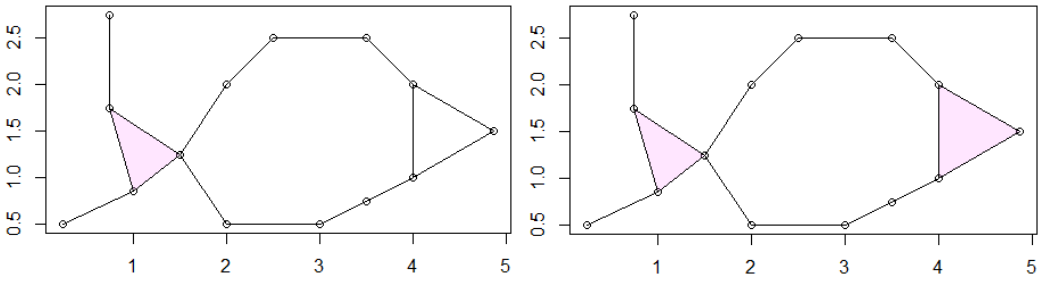
$$z \mapsto \min\{d(z, l_1), d(z, l_2), \ldots, d(z, l_i)\}$$

El método *maxmin* garantiza que los marcadores escogidos queden bien espaciados entre ellos, pero tiene un cierto sesgo hacia los puntos más alejados del resto. No existe una regla general para decir cuál es la cantidad perfecta de marcadores.

También hay varias formas de construir un complejo *witness*. En todas ellas hemos de construir una matriz D con $|L|$ filas y $|W|$ columnas, de modo que el elemento d_{ij} sea la distancia entre el marcador l_i y el testigo w_j.

Definición 1.12 (W_∞)**.** *Sea D la matriz de distancias entre L y W. El complejo witness estricto $W_\infty(D)$ se define de forma inductiva:*

- *Los marcadores l_i y l_j definen una arista si y solo si existe un testigo w_k tal que d_{ik} y d_{jk} sean las entradas más pequeñas de la columna k-ésima de D.*



(a) Complejo $C(S, 0.5)$ (b) Complejo $VR(S, 1)$

Figura 1.4: Comparativa entre los complejos de Čech y Vietoris-Rips. Nótese que la única diferencia entre ellos está en el triángulo equilátero con lado unidad de la derecha. Las tres bolas de sus vértices no tienen intersección común cuando $\varepsilon = 0.5$, y no puede entrar en $C(S, 0{,}5)$ pero sí lo hace en $VR(S, 0{,}5)$.

- *Por inducción, sea el p-símplice* $\sigma = \{l_{i_0}, l_{i_1}, \ldots, l_{i_p}\}$. *Supongamos que todas sus caras pertenecen a* $W_\infty(D)$. *Entonces* $\sigma \in W_\infty(D)$ *si y solo si existe un testigo* w_k *tal que* $d_{l_{i_0}k}, d_{l_{i_1}k}, \ldots, d_{l_{i_p}k}$ *sean las* $p+1$ *entradas más pequeñas de la columna k-ésima de D.*

La justificación teórica de esta construcción viene motivada por comparación con el complejo de Delaunay $D(L)$. El símplice $\sigma = \{l_{i_0}, l_{i_1}, \ldots, l_{i_p}\}$ entra en $D(L)$ si existe un punto $x \in \mathbb{R}^d$ equidistante de todos ellos y sin otro vecino más cercano. En estas condiciones, a x se le llama un testigo fuerte de σ. Dado que el conjunto de posibles testigos fuertes es discreto, no tiene sentido plantearse si algún elemento de W lo es. Por ello se introduce el concepto de testigo débil. Un $x \in \mathbb{R}^d$ es un testigo débil de $\sigma = \{l_{i_0}, l_{i_1}, \ldots, l_{i_p}\}$ si estos marcadores son los $p+1$ vecinos más cercanos a x. Podemos cambiar entonces la definición de $W_\infty(D)$ diciendo que un símplice σ está en $W_\infty(D)$ si y solo si tiene un testigo débil en W y además todas sus caras tienen también un testigo débil en W.

Esta construcción, aunque bien motivada teóricamente, es algo farragosa y difícil de calcular. Por ello definimos otro complejo más sencillo, basado en el método de cliques.

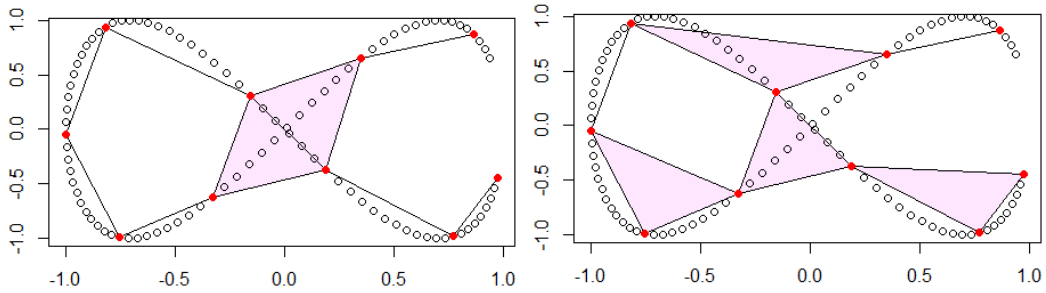
Definición 1.13 ($W_1(D)$)**.** *Sea D la matriz de distancias entre L y W. Sea* $G = (L, E)$, *donde* $\{l_i, l_j\} \in E$ *si y solo si existe un testigo* w_k *tal que* d_{ik} *y* d_{jk} *sean las entradas más pequeñas de la columna k-ésima de D. Definimos el complejo witness débil* $W_1(D)$ *como el complejo de cliques de G.*

Por construcción, $W_\infty(D) \subset W_1(D)$. Ninguno de estos dos complejos depende de un parámetro ε, lo cual nos impide examinar S al nivel de precisión que queramos. Para resolver este problema definimos una tercera clase de complejos *witness*.

Definición 1.14 ($W(D, \varepsilon, \nu)$)**.** *Sean* D, $\nu \in \mathbb{Z}_+$ *y* $\varepsilon > 0$.

- *Si* $\nu = 0$, *para cada* $k = 1, 2, \ldots, |W|$ *se define* $m_k = 0$. *Si en cambio* $v > 0$, m_k *se define como la* ν*-ésima entrada más pequeña de la columna k-ésima de D.*
- *Sea* $G = (L, E)$, *donde* $\{l_i, l_j\} \in E$ *si y solo si existe un testigo* w_k *tal que* $\max(d_{ik}, d_{jk}) \leq \varepsilon + m_k$. *Definimos* $W(D, \varepsilon, \nu)$ *como el complejo de cliques de G.*

Si $\delta < \varepsilon$, entonces $W(D, \delta, \nu) \subset W(D, \varepsilon, \nu)$. La familia de complejos que se obtiene al usar $\nu = 0$ está muy relacionada con la familia de complejos *VR*. En particular, se tiene que $W(D, \varepsilon, 0) \subset VR(L, 2\varepsilon) \subset W(D, 2\varepsilon, 0)$. Cuando $\nu = 1$ el complejo se puede interpretar como procedente de un recubrimiento mediante regiones del tipo Voronoi que se superponen al crecer



(a) Complejo $W(D, 0.6, 0)$ (b) Complejo $W(D, 0.3, 1)$

Figura 1.5: Dos ejemplos de complejos *witness* sobre un mismo conjunto de datos. Los marcadores son los puntos en rojo, mientras que los puntos en blanco son los testigos. Los dos marcadores situados más a la derecha quedan desconectados en ambos casos porque no hay testigos entre ellos dos que justifiquen la existencia de una arista.

ε. Para $\nu \geq 2$ no existe una motivación tan interesante, pero existe por ejemplo la relación $W(D, 0, 2) = W_1(D)$.

En la práctica casi siempre se aplicará esta última variante al permitir un análisis del conjunto de datos a diferentes escalas.

1.7. Filtraciones y multifiltraciones

Todos los métodos descritos hasta ahora dependen de un parámetro ε que hace crecer los complejos a medida que crece ε. Las filtraciones y multifiltraciones nos permitirán recoger esta sucesión creciente de complejos en un solo objeto de cara a hacer un análisis conjunto [4].

Definición 1.15. *Una multifiltración con p parámetros naturales (reales) es una colección de complejos simpliciales* $\{K_u\}_{u\in\mathbb{N}^p}$ *(*$\{K_u\}_{u\in\mathbb{R}^p}$*)tal que si* $u \leq v$*, entonces* $K_u \subset K_v$*. Tanto en* $\mathbb{N}^p$ *como en* $\mathbb{R}^p$ *diremos que* $u \leq v$ *si* $u_i \leq v_i$ *para todo* $i = 1, \cdots, p$*. Una multifiltración con un sólo parámetro se llama filtración.*

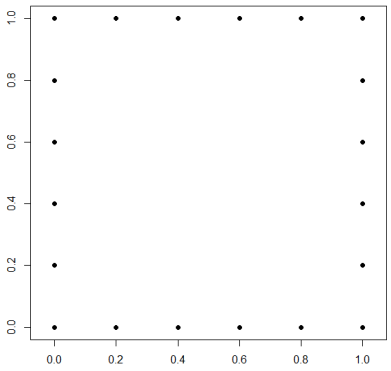
Definición 1.16. *Una filtración o una multifiltración se dicen de tipo finito si todos sus complejos simpliciales son finitos y existe una coordenada máxima* M *tal que si* $M \leq u$*, entonces* $K_M = K_u$*.*

Dado un símplice σ, se dice que u es una coordenada crítica para σ si $\sigma \in K_u$ pero $\sigma \notin K_v$ para cualquier $v \leq u$. Si todos los símplices de una multifiltración tienen una única coordenada crítica, se dice que la multifiltración es monocrítica [18]. Todas las filtraciones son monocríticas por construcción.

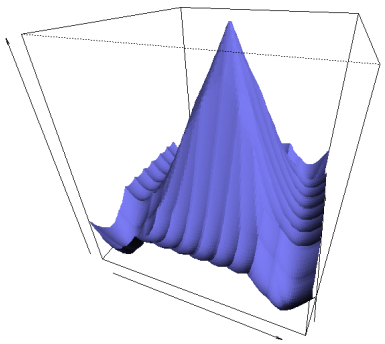
Los métodos de Čech, *alpha*, Vietoris-Rips o *witness* nos permiten construir filtraciones de tipo finito con parámetro real. Por cuestiones de economía computacional e interpretabilidad, en la práctica no se suele tomar la filtración completa sino que se escoge un M máximo y se impone que $K_\varepsilon = K_M$ para todo $\varepsilon \geq M$.

Existe otro método para construir filtraciones y multifiltraciones, basado en conjuntos de subnivel de una función $f : \mathbb{R}^d \to \mathbb{R}^p$, en una forma análoga a la que la teoría de Morse analiza los subconjuntos de nivel de una función sin puntos críticos degenerados [46]. Para cada $u \in \mathbb{R}^p$, se construye un complejo simplicial K_u que aproxime la estructura de $\{y \in \mathbb{R}^d | f(y) \leq u\}$ [36]. La unión de todos estos complejos nos da la multifiltración deseada.

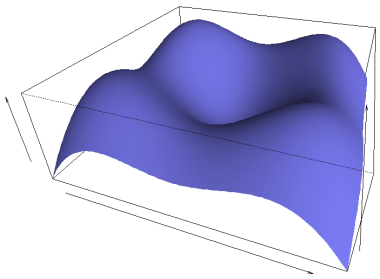
Algoritmo 1.1 (Conjuntos de subnivel)**.** *Sean* S *y una función* $f : \mathbb{R}^d \to \mathbb{R}^p$*. Los pasos para construir un complejo* K_u *que aproxime el conjunto de subnivel* $\{y \in \mathbb{R}^d | f(y) \leq u\}$ *son:*



(a) Muestra S



(b) Función DTM sobre S



(c) Función KDE sobre S

Figura 1.6: Gráficas de las funciones de distancia y densidad evaluadas sobre una malla en el cuadrado unidad

- *Se construye una malla cúbica C en torno a S.*
- *Dentro de cada cubo de C, además de sus aristas definimos unas diagonales de forma tal que el complejo de cliques del cubo, en su forma geométrica, lo "rellene". Dos puntos de C unidos por una arista o una diagonal se llamarán adyacentes.*
- *Se calcula f sobre cada punto $x \in C$, y decimos que $x \in V$ si y solo si $f(x) \leq u$.*
- *Si dos puntos $x_1, x_2 \in V$ son adyacentes en C, se define la arista $\{x_1, x_2\} \in E$.*
- *Dado $G = (V, E)$, se define K_u como el complejo de cliques de G.*

Este procedimiento se puede definir de forma análoga para analizar conjuntos de supernivel $\{y \in \mathbb{R}^d | f(y) \geq u\}$. Las funciones que se usen para filtrar S pueden ser de cualquier tipo, como estimadores de densidad, funciones de excentricidad u otras funciones con información geométrica adicional, como la curvatura escalar [16]. Un ejemplo es la función de distancia a una medida, que supone un suavizado de la distancia euclídea $\Delta(y) = \inf ||y - x||_{x \in S}$ [21].

Definición 1.17. *Sean S, $m_0 \in (0, 1)$ y $r \in [1, \infty)$. La función de distancia a una medida (DTM por sus siglas en inglés) es:*

$$\hat{d}_{m_0}(y) = \left(\frac{1}{k} \sum_{x_i \in N_k(y)} ||x - y||^r \right)^{1/r}, \tag{1.1}$$

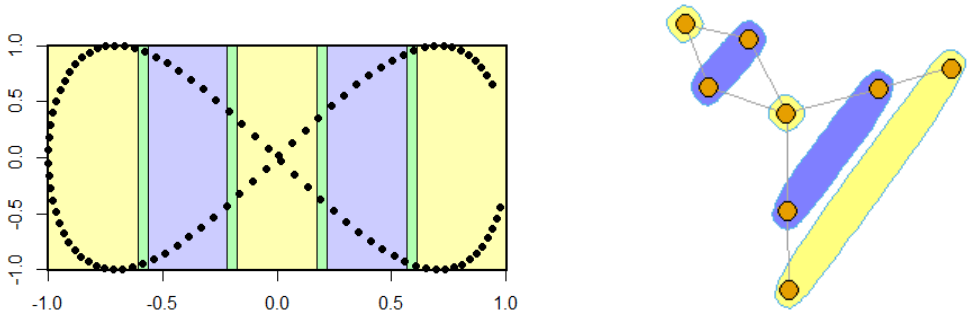
donde $k = \lceil m_0 \cdot N \rceil$ y $N_k(y)$ es el conjunto de los k vecinos de S más cercanos a y.

Según m_0 va creciendo, la función $\hat{d}_{m_0}$ se va haciendo más suave. En general siempre se toma $r = 2$ convirtiendo la expresión en una media cuadrática. Los conjuntos de subnivel $\{y \in \mathbb{R}^d | \hat{d}_{m_0}(y) \leq \varepsilon\}$ dan una buena aproximación del recubrimiento que define el complejo de Čech.

Para mitigar el efecto de posibles datos anormales o *outliers*, es posible definir estimadores de densidad [23], que dan más importancia a las regiones donde se acumulan más puntos de S.

Definición 1.18. *Dados S y $h > 0$, el estimador de densidad con núcleo gaussiano (KDE) es:*

$$\hat{p}_h(y) = \frac{1}{N(h\sqrt{2\pi})^d} \sum_{i=1}^{N} \exp\left(\frac{-||y - x_i||_2^2}{2h^2} \right) \tag{1.2}$$



(a) Recubrimiento mapper (b) Grafo mapper

Figura 1.7: Construcción de un grafo *Mapper* sobre un conjunto de puntos. La función de filtro aplicada es la proyección sobre la primera componente. Los rectángulos amarillos y azules son la antiimagen de los intervalos escogidos, siendo las franjas verdes la intersección entre ellos.

Esta función es el resultado de centrar en cada punto de S una gaussiana con varianza dependiente del parámetro h y hacer la media aritmética. Es continua, diferenciable y en particular es función de Morse.

Cuando se usa la función *DTM* se suelen tomar conjuntos de subnivel para aproximar el complejo de Čech, mientras que cuando se usa la función *KDE* u otras funciones de densidad como el estimador de los k vecinos más próximos, lo habitual es tomar conjuntos de supernivel.

También pueden definirse métodos mixtos para construir multifiltraciones, que combinen los complejos de Čech, *alpha*, Vietoris-Rips o *witness* para un parámetro y subconjuntos de subnivel o supernivel para otros [16].

1.8. Algoritmo *Mapper*

Este método es diferente a los anteriores en tanto que su objetivo no es sólo resumir información topológica sino además reducir la dimensionalidad y ofrecer una visualización útil y simple del conjunto de datos [59].

Algoritmo 1.2 (*Mapper*)**.** *Sean S y una función de filtro $f : S \to \mathbb{R}$.*

- *El rango de valores que toma f se recubre con un conjunto de intervalos de igual longitud que se solapen entre ellos. La longitud de esos intervalos y el porcentaje de solapamiento quedan a elección del usuario.*
- *A cada intervalo I se le calcula su antiimagen $f^{-1}(I) \subset S$.*
- *A cada $f^{-1}(I)$ se le aplica un algoritmo de clustering.*
- *Creamos un complejo simplicial abstracto K que tiene un vértice por cada cluster obtenido en el paso anterior.*
- *Los clusters $C_0, C_1, \cdots, C_p$ definen un símplice de K si y solo si tienen un punto de S en común.*

Es posible extender este algoritmo para funciones que tomen valores en $\mathbb{R}^n$. Dos ejemplos de funciones muy usadas como filtro son el estimador de densidad con núcleo gaussiano y la función de excentricidad, que identifica los puntos que se encuentran más lejos del centro sin calcularlo explícitamente. Su fórmula es:

$$E_p(y) = \left(\frac{\sum_{x \in S} ||y - x||^p}{N} \right)^{1/p}. \tag{1.3}$$

En general los complejos simpliciales que se obtengan con este algoritmo no se usarán para el análisis topológico que se desarrollará en los próximos capítulos, sino para el resumen y representación visual de conjuntos de datos muy numerosos [16]. No obstante, es ampliamente usado por un gran número de autores para ejemplos de aplicación práctica [51] y será en la última sección cuando veamos algunos de ellos.

Capítulo 2

Homología simplicial

El grupo fundamental $\pi_1(X)$, que estudia los lazos de un espacio topológico X, tiene una generalización en los grupos de homotopía $\pi_n(X)$, que se definen en términos de aplicaciones entre cubos n-dimensionales I^n y X y homotopías entre estas aplicaciones. Aunque tienen un gran interés teórico, en general estos grupos son difíciles de computar. En esta sección presentamos una alternativa: los grupos de homología. En particular estudiaremos la homología de un complejo simplicial K dado. Estos grupos son más fácilmente computables que los grupos de homotopía, pero tienen una definición menos natural y requieren resolver ciertas cuestiones técnicas para su cálculo [40].

Dado un complejo simplicial K, supondremos a lo largo de toda la sección que sus vértices están ordenados, y se escribirán como $x_1, x_2, \cdots, x_N$.

2.1. Grupos de cadenas

Siguiendo la definición 1.1 de complejo simplicial, un símplice σ viene determinado por un conjunto de vértices $\{x_{i_0}, x_{i_1}, \cdots, x_{i_p}\} \in K$, siendo irrelevante el orden en que se presenten. Por ejemplo, los conjuntos $\{x_1, x_2\}$ y $\{x_2, x_1\}$ definen el mismo símplice. Para definir la homología simplicial supondremos que los símplices están ordenados, y para remarcar que el orden importa cambiamos las llaves por corchetes, escribiendo $\sigma = [x_{i_0}, x_{i_1}, \cdots, x_{i_p}]$. Entonces, $[x_1, x_2]$ y $[x_2, x_1]$ son diferentes ordenaciones del mismo símplice.

Definición 2.1. *Decimos que un símplice ordenado $[x_{i_0}, x_{i_1}, \cdots, x_{i_p}]$ está bien ordenado si $i_0 < i_1 < \cdots < i_p$, es decir, si sus vértices siguen la ordenación que dimos al principio en K. Por otra parte, diremos que un símplice ordenado está orientado positivamente si existe una permutación par de sus vértices que lo convierta en un símplice bien ordenado. Si no existe esta permutación par, diremos que el símplice ordenado está orientado negativamente.*

Siguiendo con el ejemplo anterior, $[x_1, x_2]$ tendría orientación positiva porque está bien ordenado mientras que $[x_2, x_1]$ tendría orientación negativa. Una vez definidos estos términos, podemos definir los grupos de cadenas, que serán nuestro primer paso en la definición de la homología [49].

Definición 2.2. *Dado un complejo simplicial K, su grupo n-ésimo de cadenas $C_n(K)$ es un grupo abeliano libre que tiene por generadores a sus n-símplices bien ordenados, donde un reordenamiento de sus vértices es equivalente a un cambio de signo si la permutación es impar y no afecta nada si la permutación es par.*

En nuestro ejemplo, $[x_1, x_2] = -[x_2, x_1]$. Si un complejo simplicial no tiene n-símplices, entonces no hay generadores y $C_n(K) = \{0\}$. A partir de K, podemos crear el grupo $C_n(K)$ para todo $n \in \mathbb{Z}$.

Definición 2.3. *Un elemento $c \in C_n(k)$ es una n-cadena, y se puede representar como $c = \sum_i c_i \cdot \sigma_i$, donde los σ_i son n-símplices bien ordenados de K, y los c_i son coeficientes en $\mathbb{Z}$.*

En las próximas secciones veremos cómo se relacionan los grupos de cadenas entre sí dando lugar al concepto de homología simplicial.

2.2. Operador borde

Una vez definidos los grupos de cadenas, pasamos a estudiar cómo se relacionan entre ellos mediante los operadores borde, que asignan a cada símplice orientado sus caras orientadas [40]. A partir de ahora omitiremos el complejo simplicial K cuando nombremos a sus grupos de cadenas.

Definición 2.4. *El operador borde n-ésimo es un homomorfismo de grupos $\partial_n : C_n \to C_{n-1}$ que opera sobre sus símplices de la siguiente forma:*

$$\partial_n[v_0, \cdots, v_n] = \sum_{i=0}^{n} (-1)^i [v_0, \cdots, \hat{v}_i, \cdots, v_n], \tag{2.1}$$

donde $\hat{v}_i$ indica que el vértice v_i se ha eliminado del símplice. ∂_n se extiende al resto de cadenas compuestas por linealidad.

Definición 2.5. *Dada una n-cadena c, al elemento $\partial_n(c)$ se le llama borde de c. Una n-cadena con borde nulo se llama n-ciclo.*

Los operadores borde conectan los diferentes grupos de cadenas entre sí, dando lugar a un complejo de cadenas [40].

Definición 2.6. *Dado un complejo simplicial K, su complejo de cadenas C_* es un conjunto $\{\{C_n\}_{n\in\mathbb{N}}, \{\partial_n\}_{n\in\mathbb{N}}\}$ de grupos de cadenas y operadores borde formando el siguiente diagrama:*

$$C_* : \cdots \to C_{n+1} \xrightarrow{\partial_{n+1}} C_n \xrightarrow{\partial_n} C_{n-1} \to \cdots \tag{2.2}$$

A continuación probamos una característica importante de los complejos de cadenas [49].

Proposición 2.1. $\partial_{p-1} \circ \partial_p \equiv 0 \ \forall p \in \mathbb{Z}$.

Corolario 2.1. $\operatorname{im} \partial_p \subset \ker \partial_{p-1} \ \forall p = 1, \cdots, m+1$.

Esta proposición implica que $\operatorname{im} \partial_p \subset \ker \partial_{p-1}$. Dicho en lenguaje natural, el borde de cualquier n-cadena es un ciclo de C_{n-1}.

2.3. Grupos de homología

El corolario 2.1 afirma que $\operatorname{im} \partial_p \subset \ker \partial_{p-1}$. En particular, $\operatorname{im} \partial_p$ es un subgrupo de $\ker \partial_{p-1}$ y podemos dar la siguiente definición [40]:

Definición 2.7. *Sea un complejo simplicial K y sea C_* su complejo de cadenas. Definimos el grupo de homología n-ésimo de K (o de C_*) como el cociente:*

$$H_n = \frac{\ker \partial_n}{\operatorname{im} \partial_{n+1}} \tag{2.3}$$

Recordemos que $\ker \partial_n$ es el grupo de todos los n-ciclos de K, mientras que $\operatorname{im} \partial_{n+1}$ es el grupo de los n-ciclos que son borde de alguna $(n+1)$-cadena de K. Al hacer el cociente, H_n solo conserva los ciclos que no bordean ninguna cadena de dimensión mayor, es decir, conserva los ciclos que representan "agujeros vacíos" en K.

Como los grupos de homología son grupos abelianos con un número finito de generadores, en particular son $\mathbb{Z}$-módulos finitamente generados y les podemos aplicar el siguiente teorema [32] para dar una clasificación suya.

Teorema 2.1 (teorema fundamental de los R-módulos finitamente generados)**.** *Sea R un dominio de ideales principales (como $\mathbb{Z}$). Todo R-módulo finitamente generado se descompone de forma única salvo orden como la suma directa:*

$$\bigoplus_{i=1}^{\beta} R \oplus \bigoplus_{j=1}^{m} R/t_j R, \tag{2.4}$$

donde β es un entero no negativo y los t_j son elementos no nulos y no unidades de R tales que $t_j | t_{j+1}$.

El primer elemento de esta suma directa se llama submódulo libre y se caracteriza por su rango o número de Betti β. Cuando hablemos de un grupo de homología H_n, el número de Betti se escribirá β_n. El segundo elemento de la suma se llama submódulo de torsión y se caracteriza por sus coeficientes de torsión $t_1, \cdots, t_m$ (salvo producto por una unidad). En conjunto el R-módulo queda parametrizado por los $m+1$ elementos $\beta, t_1, \cdots, t_m$.

En complejos simpliciales de dimensión 3 sin torsión en su grupo de homología, los números de Betti tienen un significado intuitivo: β_0 es el número de componentes conexas, β_1 es el número de "túneles" y β_2 representa el número de "vacíos" o "burbujas" [4].

Aunque nuestra definición de homología se basa en grupos abelianos o $\mathbb{Z}$-módulos, se puede extender poniendo coeficientes en cualquier dominio de ideales principales en lugar de $\mathbb{Z}$. Por este motivo, de aquí en adelante no hablaremos de grupos de cadenas o de homología sino de módulos. En el caso particular en que usemos coeficientes en un cuerpo k, el submódulo de torsión desaparece, y H_n se reduce a un espacio vectorial de dimensión igual a su número de Betti β_n.

Una propiedad muy importante de los módulos de homología simplicial es que se mantienen invariantes bajo la acción de la homotopía [40].

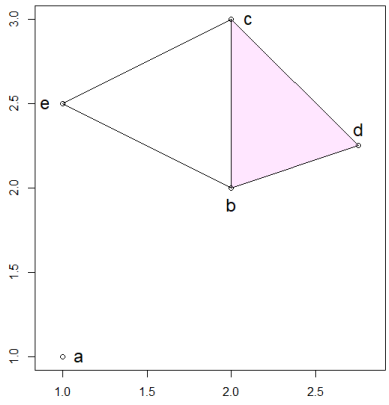
Proposición 2.2. *Sean K_1 y K_2 dos complejos simpliciales geométricos homotópicamente equivalentes. Entonces los R-módulos de homología $H_n(K_1)$ y $H_n(K_2)$ son isomorfos para todo n y para todo anillo R.*

Entonces dado S y un ε los complejos $C(S, \varepsilon)$ y $A(S, \varepsilon)$ no son en general iguales pero al ser homotópicamente equivalentes sus módulos de homología sí lo son.

Como veremos en la próxima sección, la elección de un anillo u otro puede simplificar o complicar la computación de las distintas homologías para un complejo K, pero modifica su estructura así que debemos ser cautelosos si no queremos perder información relevante.

2.4. Computación de la homología

Una vez definida la homología simplicial, procedemos a su computación efectiva. Para ello debemos calcular el núcleo y la imagen de cada operador borde ∂_n. Como son operadores lineales entre módulos, se pueden representar a través de una matriz M_n en términos de ciertas



(a) Complejo simplicial K

$$\left(\begin{array}{c|ccccc} & bc & bd & be & cd & ce \\ \hline a & 0 & 0 & 0 & 0 & 0 \\ b & -1 & -1 & -1 & 0 & 0 \\ c & 1 & 0 & 0 & -1 & -1 \\ d & 0 & 1 & 0 & 1 & 0 \\ e & 0 & 0 & 1 & 0 & 1 \end{array}\right)$$

(b) M_1

$$\left(\begin{array}{c|c} & bcd \\ \hline bc & 1 \\ bd & -1 \\ be & 0 \\ cd & 1 \\ ce & 0 \end{array}\right)$$

(c) M_2

Figura 2.1: Usaremos este complejo K para ilustrar los cálculos descritos en esta sección. Partimos de las matrices M_1 y M_2 respecto de las bases naturales de C_0, C_1 y C_2. Por comodidad cambiamos la notación con corchetes por la yuxtaposición para remarcar el orden de los vértices.

bases. Nuestra elección natural será la de tomar los n-símplices orientados de K como base de C_n. Calcular el núcleo y la imagen de ∂_n equivale a calcular el espacio nulo y el espacio de columnas de la matriz M_n respectivamente. La herramienta principal que usaremos para calcular la homología es la forma normal de Smith [5].

Teorema 2.2. *Sea A una matriz con entradas en el anillo R. Existen dos matrices P y Q productos de de matrices elementales tales que $A' = Q^{-1}AP$, donde A' tiene la forma:*

$$\left(\begin{array}{ccc|c} a_1 & & & \\ & \ddots & & 0 \\ & & a_r & \\ \hline & 0 & & 0 \end{array}\right), \tag{2.5}$$

y $a_1|a_2|\cdots|a_r$. En estas condiciones, se dice que A' es la forma normal de Smith de A.

La forma más sencilla de leer la estructura de H_n se basa en expresar la matriz del homomorfismo ∂_{n+1} con respecto a la base natural de C_{n+1} y una base de $\ker \partial_n$. Este cambio de base es trivial gracias al siguiente resultado:

Lema 2.1 (del cambio de base)**.** *Sea M_{k+1} la matriz de ∂_{k+1} con respecto a las bases naturales de C_{n+1} y C_n y sea M'_k la forma normal de Smith de M_k. Se puede obtener una matriz de ∂_{k+1} con respecto a la base natural de C_{k+1} y una base de $\ker \partial_k$ simplemente eliminando de M_{k+1} las filas que corresponden con columnas pivotales (es decir, no nulas) en M'_k.*

Demostración. Dado que $\partial_k \circ \partial_{k+1} \equiv 0$, entonces $M_k \cdot M_{k+1} = 0$. Para transformar M_k en M'_k solo se usan operaciones elementales por filas y columnas de los tipos 1 y 3. Para que el producto de las dos matrices siga siendo nulo durante la reducción de M_k, una operación en las columnas de M_k debe ir acompañada de una operación por filas en M_{k+1}. Las operaciones por filas en M_k no afectan a la matriz M_{k+1}.

Una transposición de columnas en M_k se corresponde con una transposición de sus filas respectivas en M_{k+1}. Si reemplazamos *(columna i)* por *(columna i) + q(columna j)* en M_k, debemos reemplazar *(fila j)* por *(fila j) $-q$(fila i)* en M_{k+1}.

Tras acabar la reducción, se ve que si una columna de M_k ha sido anulada, su correspondiente fila de M_{k+1} no ha sido alterada en ningún momento. Entonces tenemos:

$$0 = M_k \cdot M_{k+1} = M'_k \cdot \tilde{M}_{k+1} = (T|0) \cdot \begin{pmatrix} A \\ B \end{pmatrix} = T \cdot A + 0 \cdot B = T \cdot A,$$

donde T es una submatriz de $\tilde{M}_k$ con sus columnas pivotales, y 0 es una submatriz nula. Las filas de B son aquellas de M_{k+1} que no han sufrido alteración en el proceso, y se corresponden con una base de $\ker \partial_k$. Dado que $T \cdot A = 0$, y todas las columnas de T son linealmente independientes, $A = 0$ y podemos eliminar sus filas de M_{k+1}. □

Este lema y su prueba se basan en el artículo [75], que está referido a la homología persistente. En su versión original M'_k no es la forma normal de Smith sino la forma escalonada por columnas de M_k. Nuestra prueba es esencialmente la misma, pero también considera el efecto de las operaciones por filas en M_k. Por tanto hemos conseguido una mejora en el tiempo de computación reduciendo en buena parte el tamaño de las matrices implicadas.

Una vez que la matriz de ∂_{n+1} está escrita en función de las bases de C_{n+1} y $\ker \partial_n$, la llevamos a su forma normal de Smith mediante operaciones elementales por filas y columnas, como en (2.5). Sean $c_1, \cdots, c_s$ los ciclos correspondientes a las filas de esta matriz, o equivalentemente la base de $\ker \partial_n$. Entonces, el R-módulo n-ésimo de homología es:

$$H_n = \frac{\ker \partial_n}{\operatorname{im} \partial_{n+1}} \cong \frac{\langle c_1, \cdots, c_r, \cdots, c_s \rangle}{\langle a_1 c_1, \cdots, a_r c_r \rangle} \cong \bigoplus_{i=1}^{s-r} R \oplus \bigoplus_{j=1}^{r} R/a_j R \tag{2.6}$$

Si a_j es unidad de R para algún j, el cociente $R/a_j R$ se anula y no se incluye en la suma directa. Entonces, tenemos que $\beta_n = s - r$ y los elementos torsionarios de H_n son los a_j no unidades [32].

Con estas herramientas, podemos dar un algoritmo que nos ayude a calcular todos los grupos de homología en cadena [75].

Algoritmo 2.1 (cálculo en cadena de todos los módulos de homología)**.** *Sea K un complejo simplicial de dimensión m, y sean $M_1, \cdots, M_m$ las matrices de los operadores borde $\partial_1, \cdots, \partial_m$ respecto de las bases naturales de $C_0, C_1, \cdots, C_m$.*

- *Reducimos M_1 a su forma normal de Smith. Extraemos la estructura de H_0 como en la ecuación* (2.6)*, recordando que* $\ker \partial_0 = C_0$.
- *Tomamos M_2 y eliminamos las filas que se correspondan con columnas pivotales de M_1 en virtud del lema 2.1.*
- *Reducimos la matriz M_2 truncada a su forma normal de Smith y extraemos la estructura de H_1 como en (2.6).*
- *Repetimos los dos pasos anteriores con las matrices $M_3, \cdots, M_m$, calculando los módulos $H_2, \cdots, H_{m-1}$.*
- *Dado que* $\operatorname{im} \partial_{m+1} = 0$*, tenemos que $H_m = \ker \partial_m$.*
- *El resto de módulos de homología son triviales.*

Si solo quisiéramos calcular la estructura de H_n para un único n sin tener en cuenta qué ciclos generan $\ker \partial_n$ y $\operatorname{im} \partial_{n+1}$, bastaría con llevar M_{n+1} a su forma normal de Smith sin truncarla previamente. Es más, como los símplices de dimensión superior a $n+1$ no juegan ningún papel en el cálculo de H_n, no necesitaríamos calcular todo el complejo simplicial sino solo su $(n+1)$-esqueleto [40].

	bc^*	bd^*	be^*	cd $+bc$ $-bd$	ce $+bc$ $-be$
c	1	0	0	0	0
d	0	1	0	0	0
e	0	0	1	0	0
$b+c+d+e$	0	0	0	0	0
a	0	0	0	0	0

(a) Forma normal de Smith de M_1

	bcd
$cd+bc-bd$	1
$ce+bc-be$	0

(b) M_2 truncada

$$
\begin{aligned}
H_0 &\cong \mathbb{Z}^5/\mathbb{Z}^3 = \mathbb{Z}^2 \\
H_1 &\cong \mathbb{Z}^2/\mathbb{Z} = \mathbb{Z} \\
H_2 &\cong 0
\end{aligned}
$$

(c) Grupos de homología de K

Figura 2.2: Cálculo de los grupos de homología de K usando la forma normal de Smith. Véase que las columnas pivotales de $\tilde{M}_1$ (marcadas con asterisco) se han eliminado de M_2, quedando ésta de forma automática en forma normal de Smith.

Cuando $R = \mathbb{Z}$, el algoritmo de reducción a la forma normal de Smith tiene un orden superior al polinómico. Aunque existen técnicas más sofisticadas para obtener la reducción con menos operaciones [61, 62], en la práctica se suele preferir la reducción clásica. Si en cambio los coeficientes se toman dentro de un cuerpo k, no es necesario llevar las matrices a su forma normal de Smith, sino a su forma escalonada por columnas. En ese caso, sólo se deben calcular los números de Betti y éstos se extraen fácilmente a partir de los rangos de las matrices [66]. La computación es por tanto mucho más veloz cuando fijamos los coeficientes en un cuerpo.

Veamos con un pequeño ejemplo cómo afecta al resultado final la elección del anillo de coeficientes. Sea K un complejo simplicial homeomorfo a la botella de Klein. Los grupos de homología de K sobre $\mathbb{Z}$ son $H_0 = \mathbb{Z}$, $H_1 = \mathbb{Z} \oplus \mathbb{Z}/\mathbb{Z}2$ y $H_2 = \{0\}$. Tenemos entonces un 1-ciclo de torsión c tal que c no bordea ninguna 2-cadena, pero $2c$ si lo hace. Si tomamos coeficientes en el cuerpo $\mathbb{Z}/\mathbb{Z}2$, el ciclo $2c$ se consideraría nulo dentro de $C_1(K)$ y no aportaría nada a $\operatorname{im}\partial_2$. En ese caso, los números de Betti serían $\beta_0 = 1$, $\beta_1 = 2$, $\beta_2 = 1$. Si en cambio tomamos los coeficientes en $\mathbb{Z}/\mathbb{Z}p$ para cualquier $p \geq 2$ primo, o en otro cuerpo como $\mathbb{Q}$ o $\mathbb{R}$, el subespacio generado por $2c$ es el mismo que el generado por c, y entonces el ciclo c es parte del denominador de H_n. En ese caso, los números de Betti serían $\beta_0 = 1$, $\beta_1 = 1$, $\beta_2 = 0$. En otras palabras, al tomar los coeficientes sobre un cuerpo no podemos observar directamente la torsión de un complejo pero podemos deducir su existencia comparando los resultados obtenidos sobre cuerpos diferentes [75].

Cuando nuestro complejo K está inmerso en $\mathbb{R}^3$, sus $\mathbb{Z}$-módulos o grupos de homología no tienen torsión y se puede usar una técnica diferente para calcular sus números de Betti, llamada el algoritmo incremental [30], que no se apoya en cálculos sobre matrices sino en teoría de grafos.

Algoritmo 2.2 (algoritmo incremental para los números de Betti)**.** *Sea K un complejo simplicial inmerso en $\mathbb{R}^3$.*

- *Ordenamos los símplices de K según su dimensión, quedando $K = \{\sigma_1, \cdots, \sigma_n\}$.*
- *Definimos $\beta_j = 0$ para $j = 0, 1, 2, 3$*

- *Para todo* $i = 1, \cdots, n$

  $k = \dim \sigma_i$

  Si σ_i *pertenece a un* k*-ciclo de* $K_i = \{\sigma_1, \cdots, \sigma_i\}$

  entonces $\beta_k = \beta_k + 1$

  y si no, $\beta_{k-1} = \beta_{k-1} - 1$

En base a este algoritmo, se ve que cada vez que entra un nuevo símplice al complejo pasa una de dos cosas: o bien crea una nueva clase de homología k-ésima o bien destruye una clase $(k-1)$-ésima. Nótese que todos los vértices forman 0-ciclos al entrar en el complejo. Para detectar 1-ciclos y 2-ciclos se consideran el grafo de aristas y el grafo dual de los triángulos respectivamente y se aplican algoritmos eficientes de la teoría de grafos [27]. Por otra parte, no es posible que se forme ningún 3-ciclo porque entonces el conjunto de sus 3-símplices sería homeomorfo a $\mathbb{S}^4$, que no puede estar inmerso en $\mathbb{R}^3$. Entonces, cada tetraedro nuevo siempre destruye una clase de homología de dimensión 2.

Capítulo 3

Homología persistente

Sea una filtración de complejos simpliciales como en la definición 1.15. Cada uno de los complejos anidados que la componen es una aproximación de la topología de S a un cierto nivel de precisión. En la sección anterior hemos definido los módulos de homología para estudiar la topología de S mediante herramientas algebraicas que puedan calcularse computacionalmente. Esta herramienta opera sobre cada complejo de forma individual, pero nosotros no sabemos a priori cuál de estas diversas aproximaciones es la más acertada. Para resolver este problema nuestro enfoque consistirá en analizar la homología del complejo en todos los niveles de la filtración [33] para después resumir en una sola estructura algebraica la evolución de este proceso. Esta estructura que definiremos se llamará el módulo de homología persistente.

A lo largo de este capítulo supondremos que los módulos de cadenas y los módulos de homología tienen sus coeficientes sobre un anillo genérico R.

3.1. Complejos y módulos de persistencia

En una filtración $\mathcal{K}$, cada complejo simplicial K_u cuenta con su propio complejo de cadenas C_*^u. Los complejos de persistencia nos permitirán reunir todas estos complejos en uno solo [75].

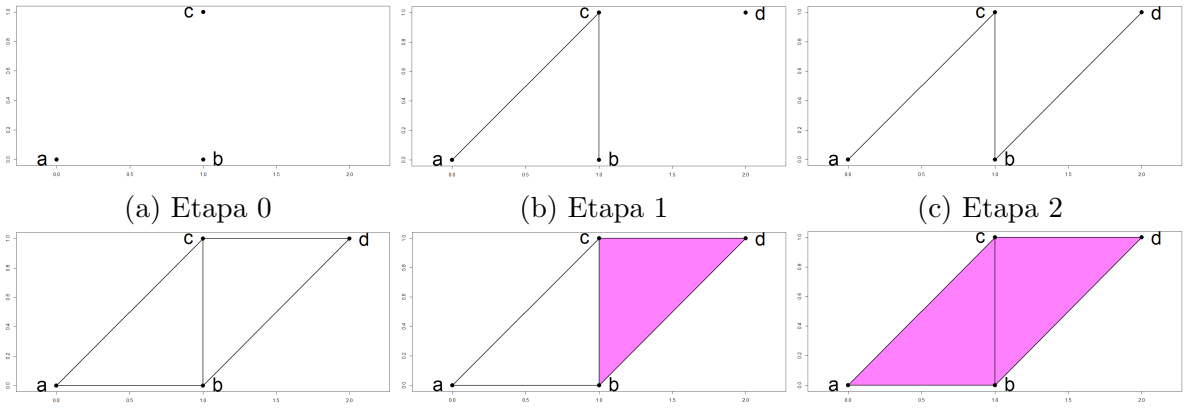
Definición 3.1. *Sea $\mathcal{K} = \{K_u\}_{u\in\mathbb{N}}$ ($\{K_u\}_{u\in\mathbb{R}}$) una filtración de tipo finito. Su complejo de persistencia $\mathcal{C}$ es la familia de complejos de cadenas $\{C_*^u\}_{u\in\mathbb{N}}$ ($\{C_*^u\}_{u\in\mathbb{R}}$) con coeficientes en un anillo R, junto con los homomorfismos de complejos de cadenas inducidos por la inclusión $i_{u,v} : C_*^u \to C_*^v$ por cada par $u \leq v$.*

Los complejos de persistencia de filtraciones con parámetro natural se pueden expresar más naturalmente en forma de diagrama como:

$$\mathcal{C} : C_*^0 \xrightarrow{i_{0,1}} C_*^1 \xrightarrow{i_{1,2}} C_*^2 \xrightarrow{i_{2,3}} \cdots \tag{3.1}$$

Si expandimos los complejos de cadenas, mostrando todos sus módulos de cadenas, el diagrama se puede ver como:

$$\begin{array}{ccccccc}
\partial_3 \downarrow & & \partial_3 \downarrow & & \partial_3 \downarrow & & \\
C_2^0 & \xrightarrow{i_{0,1}} & C_2^1 & \xrightarrow{i_{1,2}} & C_2^2 & \xrightarrow{i_{2,3}} & \cdots \\
\partial_2 \downarrow & & \partial_2 \downarrow & & \partial_2 \downarrow & & \\
C_1^0 & \xrightarrow{i_{0,1}} & C_1^1 & \xrightarrow{i_{1,2}} & C_1^2 & \xrightarrow{i_{2,3}} & \cdots \\
\partial_1 \downarrow & & \partial_1 \downarrow & & \partial_1 \downarrow & & \\
C_0^0 & \xrightarrow{i_{0,1}} & C_0^1 & \xrightarrow{i_{1,2}} & C_0^2 & \xrightarrow{i_{2,3}} & \cdots
\end{array} \tag{3.2}$$



(a) Etapa 0 (b) Etapa 1 (c) Etapa 2

(d) Etapa 3 (e) Etapa 4 (f) Etapa 5

Figura 3.1: Ejemplo de filtración de tipo finito con parámetro natural

Véase que en este diagrama cada operador borde ∂_n no actúa sobre un único módulo de cadenas, sino que lo hace simultáneamente sobre todos los módulos de una misma fila. De ahora en adelante omitiremos si el parámetro es natural o real.

Como según la definición 3.1 $\mathcal{K}$ es una filtración de tipo finito, $\mathcal{C}$ es un complejo de persistencia de tipo finito, donde todos sus R-módulos son finitamente generados y existe un parámetro máximo M tal que si $M \leq u \leq v$ entonces $i_{u,v}$ no es la inclusión sino la identidad.

Ahora definiremos una nueva estructura que agrupe todos los R-módulos de n-cadenas en uno solo.

Definición 3.2. *Un módulo de persistencia $\mathcal{M}$ es una familia de R-módulos $\{M^i\}$ junto con un homomorfismo $f_{u,v} : M^u \to M^v$ por cada par $u \leq v$ de forma tal que $f_{u,u} = Id_{M^u}$ y tal que $f_{v,w} \circ f_{u,v} = f_{u,w}$ si $u \leq v \leq w$.*

En el diagrama (3.2), cada fila es un módulo de persistencia con parámetro natural. El módulo de persistencia de la fila n-ésima se llama $\mathcal{M}_n$. Ésta será la herramienta principal para analizar la homología simplicial de una filtración.

3.2. Módulos de homología persistente

Sea un complejo de persistencia $\mathcal{C} = \{C^i_*\}$ dado por una filtración $\mathcal{K}$ de tipo finito. Para estudiar la evolución de la homología n-ésima a lo largo de la filtración, el primer paso consiste en calcular todos los R-módulos de homología $\{H^i_n\}$.

Como ya pudimos ver al analizar el algoritmo incremental 2.2 para los números de Betti, cuando incluimos un nuevo n-símplice en un complejo inmerso en $\mathbb{R}^3$ sin torsión pueden pasar dos cosas: o bien crea una clase de homología n-dimensional que antes no existía, o bien destruye una clase $(n-1)$-dimensional. Cada vez que se avanza un paso en la filtración, algunas clases de homología "nacen" y otras "mueren". Nuestro objetivo será el de determinar cómo es esta evolución para filtraciones más generales y usando coeficientes en un anillo general.

Definición 3.3 (primera definición de homología persistente)**.** *Sea $\mathcal{K}$ una filtración de tipo finito y sea $\mathcal{C}$ su complejo de persistencia. Definimos el módulo de homología persistente n-ésima PH_n como el módulo de persistencia dado por los R-módulos $\{H^u_n\}$ y los homomorfismos $\iota_{u,v} : H^u_n \to H^v_n$, donde $\iota_{u,v}$ lleva cada clase de homología de H^u_n a la clase de H^v_n que contenga a un representante suyo.*

Si la filtración tiene parámetro natural, el módulo PH_n se puede representar con el diagrama:

$$PH_n : H_n^0 \xrightarrow{\iota_{0,1}} H_n^1 \xrightarrow{\iota_{1,2}} H_n^2 \xrightarrow{\iota_{2,3}} \cdots . \tag{3.3}$$

Existe otra forma más breve y equivalente de introducir el módulo de persistencia PH_n. Al desarrollar el complejo de persistencia $\mathcal{C}$, vimos que los operadores borde ∂_n se pueden extender para actuar sobre todo un módulo de persistencia a la vez, de forma que el borde de una cadena se encuentra en la misma etapa que ésta. En base a este operador borde extendido, podemos dar la siguiente definición:

Definición 3.4 (segunda definición de homología persistente)**.** *Sea $\mathcal{K}$ una filtración y sea $\mathcal{C} = \{C_*^i\}$ su complejo de persistencia. Definimos el módulo de homología persistente n-ésima PH_n como el cociente:*

$$PH_n : \frac{\ker \partial_n}{\operatorname{im} \partial_{n+1}} \tag{3.4}$$

Ningún autor menciona explícitamente esta segunda definición, pero siempre se usa de forma implícita a la hora de su computación algorítmica.

Dada una filtración $\mathcal{K}$ de tipo finito, los módulos de persistencia PH_n son de tipo finito y por tanto existe un parámetro M tal que si $M \leq u \leq v$, entonces $\iota_{u,v}$ es la identidad entre R-módulos de homología.

Siguiendo un esquema análogo al de la sección anterior sobre los módulos de homología, debemos dar un teorema que nos permita la caracterización y parametrización de los módulos de persistencia de tipo finito. Para ello debemos introducir previamente el concepto de módulo graduado sobre un anillo graduado [12].

Definición 3.5. *Un anillo graduado es un anillo R que es suma directa de grupos abelianos $R = \oplus_{i\in\mathbb{N}} R_i$ (o bien $R = \oplus_{i\in\mathbb{R}} R_i$), donde el producto verifica que $R_n \cdot R_m \subset R_{n+m}$. Un elemento $e \in R_i$ se dice homogéneo de grado* $\deg e = i$*. El grado de un elemento no homogéneo es el mayor de los grados de sus componentes homogéneas. El* 1 *tiene grado* 0*, y el* 0 *tiene cualquier grado.*

Dado un anillo no graduado R, su anillo de polinomios $R[t]$ es un anillo graduado donde los monomios son los elementos homogéneos.

Definición 3.6. *Un módulo graduado M sobre un anillo graduado R es un módulo dado como suma directa $M = \oplus_{i\in\mathbb{N}} M_i$ (o bien $M = \oplus_{i\in\mathbb{R}} M_i$) donde la acción de R sobre M verifica que $R_n \cdot M_m$,$M_m \cdot R_n \subset M_{n+m}$. La definición de elemento homogéneo y de grado en un módulo graduado es análoga a la de un anillo graduado.*

Con estos conceptos ya definidos, podemos aportar el siguiente teorema de correspondencia [35]:

Teorema 3.1 (de correspondencia)**.** *Existe una equivalencia α entre la categoría de los módulos de persistencia de tipo finito sobre R y la categoría de los módulos graduados finitamente generados sobre el anillo de polinomios $R[t]$, dada por:*

$$\alpha(\mathcal{M}) = \bigoplus_u M^u, \tag{3.5}$$

donde el homomorfismo $f_{u,v}$ se corresponde con la multiplicación por el monomio t^{v-u}.

$$\left(\begin{array}{c|ccccc} & ac & bc & bd & ab & cd \\ \hline d & 0 & 0 & t & 0 & t^2 \\ a & -t & 0 & 0 & -t^3 & 0 \\ b & 0 & -t & -t^2 & t^3 & 0 \\ c & t & t & 0 & 0 & -t^3 \end{array}\right) \qquad \left(\begin{array}{c|cc} & bcd & abc \\ \hline cd & t & 0 \\ ab & 0 & t^2 \\ bd & -t^2 & 0 \\ bc & t^3 & t^4 \\ ac & 0 & -t^4 \end{array}\right)$$

(a) M_1 (b) M_2

Figura 3.2: Matrices de ∂_1 y ∂_2 para la filtración de la figura 3.1.

En una filtración, el grado de un elemento homogéneo se corresponde con la etapa en que lo estamos considerando, y la multiplicación por un monomio lo lleva a una etapa posterior.

La correspondencia de este teorema sugiere la no existencia de clasificaciones simples cuando usamos anillos que no son cuerpos. Por ejemplo, se sabe que la clasificación de módulos sobre $\mathbb{Z}[t]$ es muy compleja, y aunque puedan definirse invariantes de cierto interés, no existe una clasificación sencilla y completa de estos módulos. En cambio, nos ofrece una clasificación simple cuando usamos coeficientes sobre un cuerpo k. En este caso, el anillo graduado $k[t]$ es un dominio de ideales principales, y contamos con el siguiente teorema [75]:

Teorema 3.2 (de estructura de módulos graduados finitamente generados)**.** *Sea k un cuerpo, y sea el dominio de ideales principales $k[t]$. Todo módulo graduado finitamente generado sobre $k[t]$ se descompone de forma única salvo orden como la suma directa:*

$$\bigoplus_{i=1}^{n} \Sigma^{\alpha_i} k[t] \oplus \bigoplus_{j=1}^{m} \Sigma^{\gamma_j} k[t]/(t^{n_j}), \tag{3.6}$$

donde el símbolo Σ^d significa que el grado mínimo que puede tener un elemento en ese submódulo es d y (t^{n_j}) es el ideal de $k[t]$ generado por t^{n_j}.

El primer elemento de la suma directa es el submódulo libre. El segundo elemento de la suma directa es el submódulo de torsión. Para poder aplicar este teorema de aquí en adelante, siempre supondremos que los coeficientes de nuestros complejos de persistencia se tomarán sobre un cuerpo k y no hará falta explicitarlo. En ese caso cada módulo de homología H_n^u es un espacio vectorial y queda caracterizado por su dimensión o número de Betti β_n^u.

3.3. Computación de la homología persistente

La computación de la homología persistente sigue un esquema análogo al de la homología simplicial visto en el capítulo anterior, pero con ciertas diferencias tanto en la construcción de las matrices como en el algoritmo de reducción.

Para la computación de PH_n usaremos la definición 3.4, que requiere calcular el núcleo y la imagen de cada operador borde extendido ∂_n. La base natural del módulo de persistencia $\mathcal{M}_n$ está formada por los n-símplices que aparecen en la filtración con su respectivo grado. Dado un símplice σ de la filtración, definimos su grado como la etapa en la que σ apareció por primera vez.

En las distintas matrices M_n las entradas serán polinomios (monomios en particular). Sus coeficientes se asignan de igual forma que en el cálculo de la homología simplicial. Para asignar el grado del monomio seguimos la regla: $\deg M_n(i,j) = \deg e_j - \deg \hat{e}_i$, donde e_j pertenece a la base del módulo $\mathcal{M}_n$ y $\hat{e}_i$ pertenece a la base del módulo $\mathcal{M}_{n-1}$. Para agilizar los cálculos sobre

$$\left(\begin{array}{c|cccc} & & & & ab & cd \\ & & & & -t^2\cdot ac & -t\cdot bd \\ & bd^* & ac^* & bc^* & +t^2\cdot bc & +t^2\cdot bc \\ \hline d & t & 0 & 0 & 0 & 0 \\ a & 0 & -t & 0 & 0 & 0 \\ b & -t^2 & 0 & -t & 0 & 0 \\ c & 0 & t & t & 0 & 0 \end{array}\right)$$

(a) M_1 escalonada por columnas

$$\left(\begin{array}{c|cc} & bcd & abc \\ \hline cd - t\cdot bd + t^2\cdot bc & t & 0 \\ ab - t^2\cdot ac + t^2\cdot bc & 0 & t^2 \end{array}\right)$$

(b) M_2 truncada

$$\begin{aligned} PH_0 &\cong \Sigma^0\mathbb{R}[t] \oplus \Sigma^0\mathbb{R}[t]/(t) \oplus \Sigma^0\mathbb{R}[t]/(t) \oplus \Sigma^1\mathbb{R}[t]/(t) \\ PH_1 &\cong \Sigma^3\mathbb{R}[t]/(t^2) \oplus \Sigma^3\mathbb{R}[t]/(t) \\ PH_2 &\cong 0 \end{aligned}$$

(c) Homología persistente de la filtración

Figura 3.3: Cálculo de la homología persistente de la filtración usando la forma escalonada por columnas.

la matriz M_n, las columnas deben ir en orden creciente de grado y las filas en orden decreciente, como se hace en la figura 3.2.

Para ver la estructura de PH_n, debemos expresar la matriz de ∂_{n+1} en términos de una base de $\mathcal{M}_{n+1}$ y una base de $\ker \partial_n$. Para hacer este cambio de base en M_{n+1} usamos la versión original del lema de cambio de base 2.1. Simplemente debemos llevar la matriz M_n a su forma escalonada por columnas y eliminar en M_{n+1} las filas correspondientes a columnas pivotales (es decir, no nulas) en M_n [75]. Esta reducción en el tamaño de las matrices supone una mejora en el tiempo de computación.

Una vez hecho el cambio de base, debemos llevar esta matriz truncada a su forma normal de Smith. No obstante, no es necesario calcular la reducción completa gracias al siguiente resultado [75]:

Lema 3.1 (de la forma escalonada)**.** *Los pivotes de la forma escalonada por columnas de M_n son los mismos que los elementos diagonales en su forma normal de Smith. Es más, el grado de los elementos de las filas pivotales también es el mismo en ambas formas.*

Gracias a la ordenación de las columnas por grados, estas matrices se pueden llevar a su forma escalonada por columnas usando sólo operaciones elementales del tipo 3.

Veamos ahora cómo leer la estructura de PH_n a partir de esta matriz. Supongamos que hemos truncado M_{n+1} y la hemos llevado a su forma escalonada por columnas, quedando de la forma:

$$\left(\begin{array}{c|cccc} & e_1 & e_2 & \cdots & e_t \\ \hline \hat{e}_1 & t^{n_1} & 0 & 0 & 0 \\ \hat{e}_2 & * & t^{n_2} & 0 & 0 \\ \hat{e}_3 & * & * & 0 & 0 \\ \vdots & * & * & \vdots & \vdots \end{array}\right) \tag{3.7}$$

Recordemos que cada fila es un elemento de la base de $\ker \partial_n$. Si la fila i tiene por pivote $\tilde{M}_{n+1}(i,j) = t^n$ (el escalar no importa), entonces se añade el submódulo $\Sigma^{\deg \hat{e}_i} k[t]/(t^n)$ a PH_n. En el caso particular de que el pivote sea unitario no se añade nada. Si la fila i no tiene pivote, entonces se añade $\Sigma^{\deg \hat{e}_i} k[t]$ a PH_n.

Combinando todas estas herramientas, podemos dar un algoritmo análogo al algoritmo 2.1 para homología simplicial que calcule en cadena todos los módulos de homología persistente de una filtración [75].

Algoritmo 3.1 (cálculo en cadena de todos los módulos de homología persistente)**.** *Sea $\mathcal{K}$ una filtración de dimensión m, y sean $M_1, \cdots, M_m$ las matrices de los operadores borde $\partial_1, \cdots, \partial_m$ respecto de las bases naturales de $\mathcal{M}_0, \mathcal{M}_1, \cdots, \mathcal{M}_m$.*

- *Llevamos M_1 a su forma escalonada por columnas. Extraemos la estructura de PH_0, recordando que $\ker \partial_0 = \mathcal{M}_0$.*
- *Tomamos M_2 y eliminamos las filas que se correspondan con columnas pivotales de M_1 en virtud del lema 2.1.*
- *Llevamos la matriz M_2 truncada a su forma escalonada por columnas y extraemos la estructura de PH_1.*
- *Repetimos los dos pasos anteriores con las matrices $M_3, \cdots, M_m$, calculando los módulos $PH_2, \cdots, PH_{m-1}$.*
- *Dado que $\operatorname{im} \partial_{m+1} = 0$, tenemos que $PH_m = \ker \partial_m$.*
- *Todos los demás módulos de homología persistente son triviales.*

Si solo quisiéramos calcular el módulo de homología persistente PH_n para un único n sin tener en cuenta cuáles son los generadores de $\ker \partial_n$ y $\operatorname{im} \partial_{n+1}$, bastaría con llevar M_{n+1} a su forma escalonada por columnas sin truncarla previamente. Y como ya comentamos en el capítulo anterior, no necesitaríamos filtrar todo el complejo simplicial sino solo su $(n+1)$-esqueleto [40].

Este algoritmo se puede llevar a la práctica de una forma muy eficiente, sin almacenar explícitamente las matrices de los operadores borde y sin operar sobre los polinomios de $k[t]$, solo sobre sus coeficientes [33]. Este algoritmo está basado en el algoritmo incremental 2.2 y solo requiere saber los símplices de la filtración y sus respectivos grados. En el peor de los casos, este algoritmo tiene el mismo tiempo de computación que la eliminación gaussiana sobre cuerpos, $\mathcal{O}(m^3)$, donde m es el número de símplices de la filtración. Sin embargo, estas matrices suelen tener muchas entradas nulas y en la práctica se suele observar un desarrollo lineal en el algoritmo.

Como ya comentamos en la sección 2.4, la elección de un cuerpo u otro para los coeficientes produce diferencias en el resultado final y nos puede ayudar a deducir la existencia de ciertos elementos torsionarios en un complejo de persistencia [75].

3.4. Códigos de barras y curvas de Betti

La clasificación del teorema de estructura 3.2 nos da n submódulos libres y m submódulos de torsión. Como se puede ver en la figura 3.3, la presentación de los módulos de homología es poco visual. Para resolver este problema, definimos los códigos de barras [26].

Definición 3.7. *Sea $\mathcal{M} = \bigoplus_{i=1}^{n} \Sigma^{\alpha_i} k[t] \oplus \bigoplus_{j=1}^{m} \Sigma^{\gamma_j} k[t]/(t^{n_j})$ un módulo de persistencia finitamente generado sobre $k[t]$. Por cada submódulo $\Sigma^{\alpha_i} k[t]$ consideramos el intervalo semiinfinito $[\alpha_i, \infty)$ y por cada submódulo $\Sigma^{\gamma_j} k[t]/(t^{n_j})$ consideramos el intervalo finito $[\gamma_j, \gamma_j + n_j)$. Definimos el código de barras de $\mathcal{M}$ como el multiconjunto de estos $n + m$ intervalos.*

Gracias al teorema de estructura, podemos decir que el código de barras es un invariante completo para los módulos de persistencia y aportan una parametrización que es única salvo isomorfismo.

Los intervalos del código de barras tienen una interpretación natural. Un intervalo semi-infinito $[\alpha_i, \infty)$ representa una clase de homología que se crea en la etapa α_i y no desaparece en ninguna etapa de la filtración (se dice que esta clase "persiste" infinitamente). Un intervalo finito $[\gamma_j, \gamma_j + n_j)$ representa una clase que nace en la etapa γ_j pero sólo persiste n_j etapas, desapareciendo después.

Los códigos de barras tienen una representación visual muy sencilla y útil de los módulos de persistencia, como se puede ver en la figura 3.4. Las barras más largas representan clases de homología que persisten durante muchas etapas o infinitamente, y muy probablemente se corresponden con una característica topológica real del espacio $\mathbb{M}$ que queremos describir. En cambio, las barras más cortas representan clases de corta duración, y que podrían corresponderse con errores o ruido debido a la forma en que se muestrean los puntos de S en el espacio $\mathbb{M}$. No obstante, no siempre es conveniente asociar todas las clases de corta persistencia como un ruido, porque pueden aportar igualmente buena información topológica sobre $\mathbb{M}$.

Otros invariantes que se pueden extraer de los módulos de homología persistente PH_n son los números persistentes de Betti [33] o las curvas de Betti.

Definición 3.8. *Dada una filtración $\mathcal{K}$ y dado un par de parámetros $u \leq v$, se define el número persistente de Betti $\beta_n^{u,v}$ como $\beta_n^{u,v} = \operatorname{rank}(\iota_{u,v} : H_n^u \to H_n^v) = \dim(\operatorname{im}(\iota_{u,v}))$.*

Como hemos fijado los coeficientes en un cuerpo, tanto H_n^u como H_n^v son espacios vectoriales y tiene sentido hablar de la dimensión de la imagen de $\iota_{u,v}$. Si decidiéramos usar coeficientes en un dominio de ideales principales más general, aunque no tengamos teorema de estructura también podemos definir los números persistentes de Betti, donde el operador rank denota la dimensión del submódulo libre de la imagen de $\iota_{u,v}$.

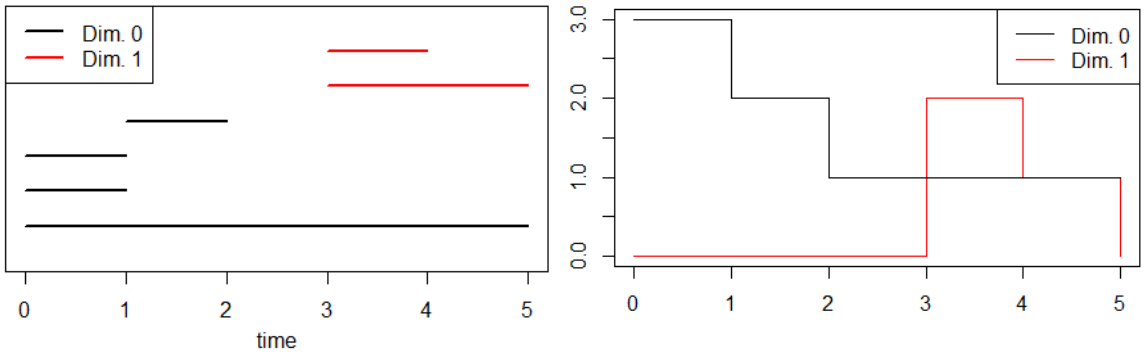
El número $\beta_n^{u,v}$ cuenta las clases de homología n-ésima que existían en la etapa u y persisten como mínimo hasta la etapa v. Si el contexto lo permite, podemos escribir simplemente $\beta^{u,v}$. El número $\beta^{u,u}$ se escribe β^u. El conjunto de todos los $\beta_n^{u,v}$ es un invariante para $\mathcal{M}_n$, pero es difícil de manejar. En la práctica se puede usar un subconjunto suyo llamado la curva de Betti [56].

Definición 3.9. *Dada una filtración $\mathcal{K}$, su curva de Betti n-ésima es la aplicación β_n que a cada parámetro u le asigna $\beta_n(u) = \beta_n^u$.*

Esta curva mide cómo evoluciona el número de clases de homología n-ésima a lo largo de la filtración. Se obtiene rápidamente a partir del código de barras contando cuántos de sus intervalos contienen a cada u. Este invariante es sencillo de calcular y visualizar, pero no es completo puesto que dos módulos de persistencia no isomorfos pueden generar la misma curva de Betti. Por tanto no se profundizará más en este aspecto, y centraremos nuestra atención en los códigos de barras, que sí son únicos salvo isomorfismo.

3.5. Diagramas de persistencia

Una alternativa equivalente a los códigos de barras son los llamados diagramas de persistencia [33]. Esta representación visual de los módulos de persistencia, más geométrica que los códigos de barras, nos permite definir de una forma sencilla una medida de similitud entre módulos que en las próximas secciones nos será de ayuda para poder aplicar inferencia estadística a la homología persistente.



(a) Código de barras (b) Curvas de Betti

Figura 3.4: Representación visual de los códigos de barras y curvas de Betti de la filtración de la figura 3.1.

Definición 3.10. *Sea $B_n = \{[b_i, d_i)\}_i$ el código de barras del módulo de homología persistente PH_n. Por cada intervalo $[b_i, d_i) \in B$, dibujamos el punto (b_i, d_i) en el plano real extendido $[0, \infty]^2$. Definimos el diagrama de persistencia D_n como $D_n = \{(b_i, d_i) | [b_i, d_i) \in B_n\} \cup d_+$, donde d_+ es la diagonal $d_+ = \{(x, x) | x \in \mathbb{R}_+\}$.*

Por construcción, todos los puntos no diagonales del diagrama de persistencia se encuentran en el primer cuadrante del plano real extendido, y por encima de d_+. En esta representación, los puntos más alejados de la diagonal se corresponden con las clases de homología más persistentes, mientras que los puntos cercanos a ella se corresponden con clases de corta duración.

Estos diagramas son muy fáciles de representar y visualizar, y por ello es habitual dibujar los diagramas de varias dimensiones diferentes simultáneamente en una misma gráfica, como se puede ver en la figura 3.5a.

A continuación presentaremos un resultado que relaciona los diagramas de persistencia con los números persistentes de Betti, dándole a éstos una representación visual [33].

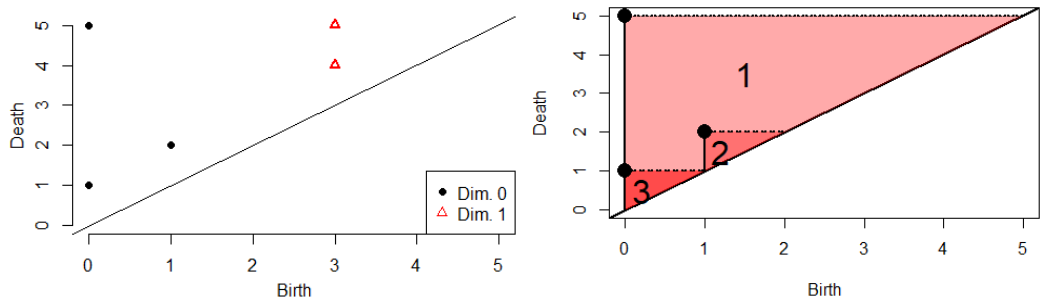
Definición 3.11. *Dado un punto (b, d) de un diagrama de persistencia D, definimos $\mathcal{T}_{b,d}$ como el triángulo dado por las desigualdades:*

$$\mathcal{T}_{b,d} = \{(x, y) \in [-\infty, \infty]^2 | x \geq b,\ y < d,\ y \geq x\} \tag{3.8}$$

Lema 3.2 (de los triángulos)**.** *Sea el módulo de homología persistente n-ésima PH_n y sea T el multiconjunto de todos los triángulos $\mathcal{T}_{b,d}$ de su diagrama de persistencia D_n. El número de triángulos de T que se superponen sobre el punto (l, p) coincide con $\beta_n^{l,p}$.*

Gracias a este lema podemos decir que el diagrama de persistencia no sólo caracteriza el nacimiento y muerte de todas las clases de homología, sino que además aporta información sobre el rango de todos los homomorfismos $\iota_{l,p}$ dentro del módulo PH_n. Recíprocamente, conociendo el valor de cada $\beta_n^{l,p}$ podemos reconstruir todos los triángulos $\mathcal{T}_{b,d}$ y por tanto el diagrama D_n. Es decir, ambos invariantes son equivalentes y como además son equivalentes al código de barras, son completos. Por ello existen algunos autores que definen la homología persistente n-ésima como la imagen de todos los homomorfismos $\iota_{l,p} : H_n^l \to H_n^p$ [4].

Por último daremos un resultado sobre la estabilidad de los diagramas de persistencia frente a pequeñas perturbaciones en el conjunto de puntos S. Para medir la perturbación de un diagrama de persistencia, antes debemos dar una medida de similaridad entre ellos [48, 25].



(a) Diagramas de persistencia (b) Lema de los triángulos

Figura 3.5: Diagrama de persistencia asociado a nuestra filtración. A la derecha vemos cómo es el lema de los triángulos sobre el diagrama de dimensión 0. Véase que $\mathcal{T}_{0,1}$ es un triángulo doble porque el código de barras contiene dos veces al intervalo $[0, 1)$

Definición 3.12. *Sean dos diagramas de persistencia D y D'.*

- *La distancia p de Wasserstein entre D y D' es:*

$$W_p(D, D') = \inf_{\phi \in \Phi} \left[\sum_{x \in D} ||x - \phi(x)||_\infty^p \right]^{1/p}, \tag{3.9}$$

  donde Φ es el conjunto de todas las biyecciones entre D y D'.

- *La distancia bottleneck entre D y D' es:*

$$d_B(D, D') = \inf_{\phi \in \Phi} \sup_{x \in D} ||x - \phi(x)||_\infty \tag{3.10}$$

En ambos casos, si no existe el ínfimo, se dice que la distancia entre D y D' es infinita.

La distancia *bottleneck* d_B es el límite de W_p cuando p tiende a infinito, y por ello algunos autores la denotan como W_∞. A la hora de calcular estas biyecciones, hay que tener en cuenta que d_+ también es parte de D, que algunos puntos pueden ser múltiples y otros pueden estar en la recta del infinito.

En base a esta distancia *bottleneck* , podemos dar el siguiente resultado [25]:

Teorema 3.3 (de estabilidad para diagramas de persistencia)**.** *Sea $\mathbb{X}$ un espacio triangulable y sean $f, g : \mathbb{X} \to \mathbb{R}$ dos funciones de Morse. Supongamos que se construyen sendas filtraciones por conjuntos de subnivel de f y g y sean $D_n(f)$ y $D_n(g)$ sus diagramas de persistencia de dimensión n asociados. Entonces los diagramas de persistencia satisfacen:*

$$d_B(D_n(f), D_n(g)) \leq ||f - g||_\infty \tag{3.11}$$

Es decir, que una pequeña perturbación en la función de filtro induce un cambio pequeño en la estructura de nuestros módulos de homología. Mientras que una pequeña perturbación en el conjunto de puntos S puede cambiar sustancialmente la homología simplicial de un complejo, la homología persistente se mantiene estable. Este es otro motivo por el cual es preferible estudiar una base de datos mediante filtraciones de complejos simpliciales antes que hacerlo a un único nivel de precisión.

Capítulo 4

Homología multipersistente

En el capítulo anterior hemos definido la homología persistente de una filtración de complejos simpliciales, hemos caracterizado su estructura algebraica y hemos encontrado una parametrización discreta de la misma. En esta sección trataremos de seguir un desarrollo similar para el caso de filtraciones multiparamétricas o multifiltraciones.

La definición 3.1 de complejo de persistencia de una filtración, la definición 3.2 de módulo de persistencia y la definición 3.3 de módulo de homología persistente se extienden fácilmente al caso multiparamétrico cambiando $u \in \mathbb{N}$ ($u \in \mathbb{R}$) por $u \in \mathbb{N}^p$ ($u \in \mathbb{R}^p$) y recordando que $u \leq v$ significa que $u_i \leq v_i$ para todo $i = 1, \cdots, p$. Igualmente, los conceptos de anillo graduado y módulo graduado se extienden de forma natural teniendo en cuenta que la suma directa que los define pasa a estar indexada sobre $\mathbb{N}^p$ ($\mathbb{R}^p$). Dado un anillo no graduado R, el anillo de polinomios con p variables $R[x_1, \cdots, x_p]$ es un anillo p-graduado donde los monomios son los elementos homogéneos. Cuando R es un cuerpo, es habitual denotar $A_p = R[x_1, \cdots, x_p]$.

En analogía con el caso uniparamétrico, también contamos con un teorema de correspondencia similar al teorema 3.1 [19]:

Teorema 4.1 (de correspondencia)**.** *Existe una equivalencia α entre la categoría de los módulos de multipersistencia de tipo finito sobre R y la categoría de los módulos graduados finitamente generados sobre el anillo de polinomios $R[x_1, \cdots, x_p]$, dada por:*

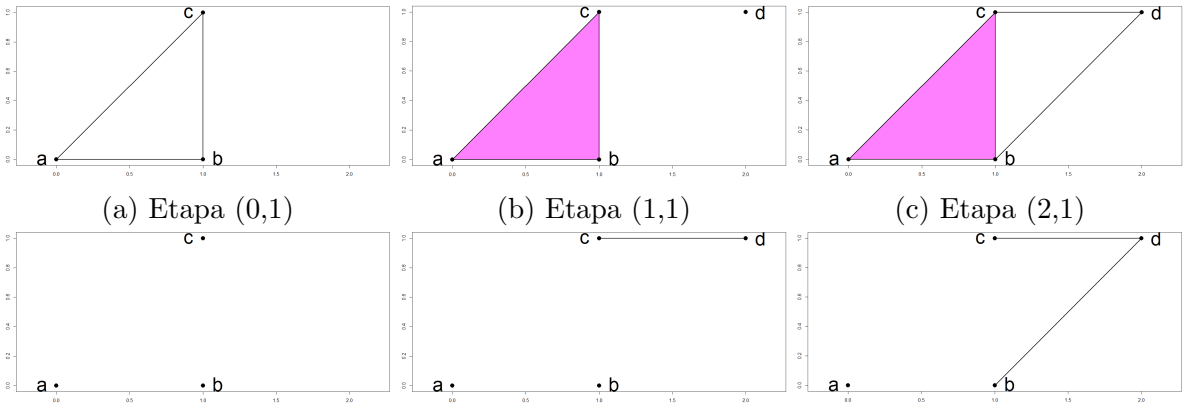
$$\alpha(\mathcal{M}) = \bigoplus_u M^u, \tag{4.1}$$

donde el homomorfismo $f_{u,v}$ se corresponde con la multiplicación por el monomio $x^{v-u} = x_1^{v_1-u_1} \cdots x_p^{v_p-u_p}$.

Al contrario que en el caso uniparamétrico, $R[x_1, \cdots, x_p]$ no es un dominio de ideales principales ni siquiera cuando R es un cuerpo, y por tanto este teorema de correspondencia no nos conduce a un teorema de estructura similar al teorema 3.2. Tampoco podremos definir un invariante similar al código de barras o al diagrama de persistencia que caracterice completamente y de forma única los módulos de multipersistencia [19].

Sí que existe una clasificación completa para los módulos de multipersistencia, que depende de dos invariantes discretos y un tercer invariante que en general es continuo [19]. Esta clasificación no nos resulta útil porque en la práctica estaría sujeta a la precisión que seamos capaces de dar dentro del rango continuo de opciones. Por este motivo la clasificación completa se descartará y tendremos que renunciar a las buenas propiedades que tienen los invariantes del caso uniparamétrico.

El problema de clasificar los módulos graduados sobre A_p está siendo ampliamente estudiado en la actualidad, y existen diferentes propuestas para atajarlo. Dedicaremos las restantes secciones de este capítulo para hacer una breve introducción a algunas de ellas.



(a) Etapa (0,1) (b) Etapa (1,1) (c) Etapa (2,1)

(d) Etapa (0,0) (e) Etapa (1,0) (f) Etapa (2,0)

Figura 4.1: Ejemplo de bifiltración monocrítica

4.1. Computación de la homología multipersistente

Una vía para el estudio de la homología multipersistente abandona la búsqueda de invariantes y trata de calcular MH_n de una forma más directa, suponiendo que la multifiltración $\mathcal{K}$ es monocrítica y por tanto los operadores borde ∂_n se pueden expresar en forma de matriz. Toda la teoría de esta sección se desarrolla en el artículo [18].

El primer paso es calcular una base para el submódulo de los ciclos $\ker \partial_n$. La matriz M_n tiene m_n columnas, a las que llamaremos $F = \{f_1, f_2, \cdots, f_{m_n}\}$. El conjunto de vectores polinómicos $(q_1, q_2, \cdots, q_{m_n}) \in A_p^{m_n}$ tales que

$$\sum_{i=1}^{m_n} q_i f_i = 0$$

forman el módulo de sicigias de $\langle F \rangle$, denotado $\mathrm{Syz}(f_1, f_2, \cdots, f_{m_n})$. Nuestro objetivo será encontrar una base de generadores para $\mathrm{Syz}(f_1, f_2, \cdots, f_{m_n})$. Para ello debemos calcular una base G de Gröbner para $\langle F \rangle$ y a partir de ella usar el algoritmo de Schreyer, basado principalmente sobre el algoritmo de división de polinomios, para obtener una base de Gröbner para el submódulo $\mathrm{Syz}(f_1, f_2, \cdots, f_{m_n})$ [28].

En segundo lugar hemos de calcular una base para el submódulo de los bordes $\operatorname{im} \partial_{n+1}$. La matriz M_{n+1} tiene m_{n+1} columnas, a las que llamaremos $F = \{f_1, f_2, \cdots, f_{m_{n+1}}\}$. F forma una base del submódulo $\operatorname{im} \partial_{n+1}$.

El tercer paso sería comprobar si los vectores de G pertenecen al submódulo $\langle F \rangle$. Para comprobar si $f \in \langle F \rangle$, aplicamos el algoritmo de división ordenadamente entre los elementos de F. Al final del proceso tenemos que $f = (\sum_{i=1}^{m_{n+1}} q_i f_i) + r$. Si el resto r es igual a 0, entonces $f \in \langle F \rangle$. En cambio, el recíproco no siempre es cierto en módulos sobre anillos de polinomios en varias variables y podría darse que $r \neq 0$ aunque $f \in \langle F \rangle$.

Para solventar este problema, debemos cambiar la base F por una base de Gröbner. Si F es una base de Gröbner y $f \in \langle F \rangle$, siempre obtenemos $r = 0$ al aplicar el algoritmo de división, y el problema de pertenencia a un submódulo queda resuelto. Dada una base F de un submódulo M, siempre podemos calcular una base de Gröbner de M a partir de F mediante el algoritmo de Buchberger [28].

Una vez que tenemos una base de Gröbner para $\operatorname{im} \partial_{n+1}$, este tercer paso es sencillo. Basta tomar cada generador de $\ker \partial_n$ y dividirlo por dicha base. Si el resto r es nulo, ese generador es un borde y no genera ninguna clase de MH_n. Si en cambio $r \neq 0$, podemos añadir r tanto a $\operatorname{im} \partial_{n+1}$ como a MH_n.

$$\left(\begin{array}{c|ccccc} & ac & ac & bc & bd & cd \\ \hline a & -x_2 & -x_2 & 0 & 0 & 0 \\ b & x_2 & 0 & -x_2 & -x_1^2 & 0 \\ c & 0 & x_2 & x_2 & 0 & -x_1 \\ d & 0 & 0 & 0 & x_1 & 1 \end{array}\right) \qquad \left(\begin{array}{c|c} & abc \\ \hline ab & x_1 \\ ac & -x_1 \\ bc & x_1 \\ bd & 0 \\ cd & 0 \end{array}\right)$$

(a) M_1 (b) M_2

Figura 4.2: Matrices de ∂_1 y ∂_2 para la bifiltración de la figura 4.1.

Los algoritmos que resuelven estos problemas en general requieren un tiempo de computación y una memoria exponencial respecto al número de generadores, pero los módulos procedentes de multifiltraciones monocríticas gozan de unas propiedades de homogeneidad que nos permiten reducir el tiempo y la memoria hasta un orden polinómico.

La solución computada por este método para la homología multipersistente es compacta y completa, pero no es un invariante dado que las bases de Gröbner calculadas son diferentes según la base con la que se inicie el algoritmo de Buchberger y la relación de orden que se defina entre monomios. No obstante, una base de Gröebner siempre se puede convertir en una base reducida de Gröbner, que sí es única para un orden fijado entre los monomios [28].

4.2. Invariante del rango

Dada una multifiltración de tipo finito cualquiera, podemos dar un invariante que compare la homología de todas las parejas de complejos posibles, al que llamaremos el invariante del rango. La descripción de este invariante se detalla en el artículo [19]. En lo que resta de capítulo supondremos que la multifiltración tiene parámetros naturales, pero también es posible considerar parámetros reales.

Definición 4.1. *Sea* $\bar{\mathbb{N}} = \mathbb{N} \cup \{\infty\}$ *y sea* $\mathbb{D}^p = \{(u, v) | u \in \mathbb{N}^p, v \in \bar{\mathbb{N}}^p, u \leq v\}$. *Dados dos pares* (u, v), $(u', v') \in \mathbb{D}^p$, *definimos* $(u, v) \preceq (u', v')$ *si y solo si* $u \leq u'$ *y* $v' \leq v$.

En el caso uniparamétrico, $\mathbb{D}^1$ es el espacio donde podemos situar los puntos de un diagrama de persistencia y $(b, d) \preceq (b', d')$ si y solo si $\mathcal{T}_{b',d'} \subset \mathcal{T}_{b,d}$.

Definición 4.2. *Sea* MH_n *un módulo de homología multipersistente. Definimos el invariante del rango de* MH_n *como la aplicación* $\rho_n : \mathbb{D}^p \to \mathbb{N}$ *dada por* $\rho_n(u, v) = \operatorname{rank}(\iota_{(u,v)} : H_n^u \to H_n^v)$.

El invariante del rango es la versión multiparamétrica de los números persistentes de Betti. Una propiedad importante del invariante del rango es que conserva el orden $\preceq$ [19].

Lema 4.1. *Si* $(u, v) \preceq (u', v')$, *entonces* $\rho_n(u, v) \leq \rho_n(u', v')$.

Demostración. Si $(u, v) \preceq (u', v')$, entonces se tiene la cadena de desigualdades $u \leq u' \leq v' \leq v$. Por tanto se tiene que $\iota_{(u,v)} = \iota_{(v',v)} \circ \iota_{(u',v')} \circ \iota_{(u,u')}$. La prueba concluye trivialmente sabiendo que dadas dos transformaciones lineales f y g, se tiene que $\operatorname{rank}(g \circ f) \leq \operatorname{rank}(f), \operatorname{rank}(g)$. □

Mientras que en el caso uniparamétrico el invariante del rango es completo, para $p \geq 2$ no lo es y podrían haber dos módulos de persistencia no isomorfos que compartan la misma ρ. El invariante del rango cuenta con su propio teorema de estabilidad, que nos permite saber que es robusto frente a pequeñas perturbaciones en S [20].

Teorema 4.2 (de estabilidad del invariante del rango)**.** *Sea $\mathbb{X}$ un espacio triangulable y sean $f, g : \mathbb{X} \to \mathbb{R}^p$ dos funciones continuas. Supongamos que se construyen sendas filtraciones por conjuntos de subnivel de f y g y sean $\rho_n(\mathbb{X}, f)$ y $\rho_n(\mathbb{X}, g)$ sus invariantes del rango de dimensión n asociados. Entonces existe una función de distancia d_{match} dentro del conjunto $\{\rho_n(\mathbb{X}, \phi) | \phi : \mathbb{X} \to \mathbb{R}^p \text{continua}\}$ tal que:*

$$d_{match}(\rho_n(\mathbb{X}, f), \rho_n(\mathbb{X}, g)) \leq \max_{x \in \mathbb{X}} ||f(x) - g(x)||_\infty \tag{4.2}$$

Para calcular el invariante del rango, debemos considerar todos los pares $(u, v) \in \mathbb{D}^d$ y calcular $\rho_n(u, v)$ para cada uno. Para ello podemos considerar la filtración que contiene únicamente a los complejos K_u y K_v y aplicar el algoritmo de persistencia 3.1. El rango $\rho_n(u, v)$ coincidiría con el número de intervalos en su código de barras que pasen a la vez por 0 y 1 [18].

Otra vía para calcular el invariante del rango consiste en calcular la homología multipersistente con los algoritmos de la sección 4.1 y calcular cada rango $\rho_n(u, v)$ con un algoritmo específico que toma como valores de entrada una base de Gröbner G de $\operatorname{im} \partial_{n+1}$ y una base Z de sicigias de $\ker \partial_n$ [18].

En ambos casos, el invariante del rango completo requiere de un gran espacio de almacenaje y un gran tiempo de cálculo. En un caso extremo donde cada coordenada crítica de nuestra multifiltración solo aporta un símplice nuevo y hay m símplices en total, pueden existir hasta un total de m^p complejos simpiciales distintos dentro de la multifiltración. En total hay $\mathcal{O}(m^{2p})$ pares de coordenadas comparables y el invariante del rango requiere un espacio de almacenaje exponencial. Por tanto es preferible solo calcularlo para algún subconjunto de pares de $\mathbb{D}^p$, como el conjunto de los pares triviales (u, u), que extiende las curvas de Betti al caso multiparamétrico.

4.3. Diagrama de persistencia multiparamétrico

Los diagramas de persistencia multiparamétricos se construirán como una especie de "derivada" del invariante del rango ρ_n, de forma tal que el diagrama indica en qué pares de $\mathbb{D}^p$ la función ρ_n cambia. Esta sección se basa en el artículo [47], y para su desarrollo cambiaremos la definición 4.1 para $\mathbb{D}^p$, eliminando los puntos infinitos.

Definición 4.3. *Definimos el orden parcial $\leq$ en $\mathbb{D}^p$ de forma tal que $(u, v) \leq (u', v')$ si existen $\varepsilon, \delta \geq 0$ tales que $u' = u + \varepsilon \cdot \vec{1}$ y $v' = v + \delta \cdot \vec{1}$*

Definición 4.4. *Sea $\mathbb{A}(\mathbb{D}^p)$ el conjunto de todas las matrices $\alpha : \mathbb{D}^p \times \mathbb{D}^p \to \mathbb{Z}$ tales que $\alpha(I, J) = 0$ si $I \not\leq J$. La multiplicación de dos matrices α y β tiene por fórmula:*

$$\alpha\beta(I, K) = \sum_{I \leq J \leq K} \alpha(I, J) \cdot \beta(J, K) \tag{4.3}$$

El conjunto $\mathbb{A}(\mathbb{D}^p)$ es cerrado para la suma, el producto por un escalar y la multiplicación.

Definición 4.5. *Hay dos matrices de $\mathbb{A}(\mathbb{D}^p)$ particularmente interesantes.*

- *La función zeta, dada por*

$$\zeta(I, K) = \begin{cases} 1 & si \quad I \leq K \\ 0 & cc. \end{cases} \tag{4.4}$$

- *La matriz de Möbius μ, dada de forma inductiva por*

$$\mu(I, K) = \begin{cases} 1 & si \quad I = K \\ -\sum_{I \leq J \leq K} \mu(I, J) & si \quad I \leq K \\ 0 & cc. \end{cases} \tag{4.5}$$

Ambas funciones son invertibles, y de hecho $\zeta\mu = \mu\zeta = 1$.

Definición 4.6. *Sea MH_n un módulo de homología multipersistente y sea ρ_n su invariante del rango. Definimos el diagrama de persistencia de MH_n como la inversión de Möbius de ρ_n, a la que denotaremos $\partial\rho_n$, y que tiene por fórmula:*

$$\partial\rho_n(K) = \sum_{J \leq K} \mu(J, K)\rho_n(J) \tag{4.6}$$

El diagrama de persistencia es nulo salvo para un número finito de elementos de $\mathbb{D}^p$, y estaremos interesados en encontrar el soporte de $\partial\rho_n$. En el caso uniparamétrico, la función $\partial\rho_n$ vale 1 en los puntos simples del diagrama D, 2 en los puntos dobles y así sucesivamente. En el caso general, $\partial\rho_n$ es no nula en los pares $(u, v) \in \mathbb{D}^p$ donde u es una coordenada de nacimiento de una clase y v es una coordenada de muerte, aunque no necesariamente estén ambas referidas a la misma clase.

El diagrama de persistencia $\partial\rho_n$ es una derivada de ρ_n en tanto que ρ_n se puede reconstruir a partir de $\partial\rho_n$ mediante sumas, y de hecho se tiene que $\rho_n(K) = \sum_{J \leq K} \partial\rho_n(J)$. Esta función también es un invariante incompleto para MH_n, y al igual que el invariante del rango, también requiere en el peor de los casos un tiempo de cálculo y una memoria exponenciales. No obstante, como solo nos interesan sus valores no nulos que en general son una minoría, el espacio de almacenamiento puede ser mucho menor.

Capítulo 5

Inferencia estadística aplicada a la homología persistente

En el capítulo 3 vimos la persistencia como una herramienta para analizar la evolución de la homología de un complejo simplicial a lo largo de una filtración. Su estructura algebraica se correspondía con un módulo graduado, y al calcular sus coeficientes sobre el dominio de ideales principales $k[t]$, encontramos dos parametrizaciones discretas equivalentes: el código de barras y el diagrama de persistencia.

Nuestro objetivo ahora será aplicar técnicas de inferencia estadística sobre los módulos de persistencia. La principal desventaja de los códigos de barras o los diagramas de persistencia a la hora de hacer estadística con ellos es que no están sujetos a nociones de aditividad o producto por un escalar, haciendo difícil incluso el cálculo de una media aritmética entre diagramas. Algunos autores propusieron usar la media de Fréchet para resolver este problema.

Definición 5.1. *Dados los diagramas de persistencia $D_1, D_2, \cdots, D_p$, se dice que el diagrama $\bar{D}$ es una media de Fréchet de este conjunto si verifica:*

$$\bar{D} = \arg\min_{Y} \frac{1}{p} \sum_{i=1}^{p} d_B(D_i, Y) \tag{5.1}$$

La búsqueda de la media de Fréchet requiere aplicar un algoritmo de descenso del gradiente, que no siempre alcanza su óptimo. Además, esta definición de media es inestable frente a pequeñas perturbaciones y en general no es única [65]. Por ello definiremos en esta sección nuevos objetos que nos permitan analizar la homología persistente desde un punto de vista estadístico.

5.1. Paisajes de persistencia y siluetas

Un paisaje de persistencia es un conjunto de funciones lineales a trozos que presenta una parametrización alternativa a los diagramas de persistencia. Su definición [24] se basa en unas ciertas funciones triangulares extraídas de un diagrama de persistencia.

Definición 5.2. *Sea D un diagrama de persistencia y sea $p = (b, d) \in D$.*

- *Si p es un punto finito no diagonal, definimos su función triangular Λ_p como:*

$$\Lambda_p(t) = \begin{cases} t - b & si \quad t \in [b, (b+d)/2] \\ d - t & si \quad t \in [(b+d)/2, d] \\ 0 & c.c. \end{cases} \tag{5.2}$$

- *Si p está en la recta del infinito, Λ_p se define como:*

$$\Lambda_p(t) = \begin{cases} t-b & si \quad t \in [b, \infty) \\ 0 & c.c. \end{cases} \tag{5.3}$$

- *Si $p \in d_+$, $\Lambda_p \equiv 0$*

Intuitivamente, estos funciones son el resultado de tomar los triángulos $\mathcal{T}_{b,d}$ de la definición 3.11, apoyar sus bases sobre la recta $y = 0$ mediante la transformación $(b, d) \mapsto (\frac{b+d}{2}, \frac{d-b}{2})$ y tomar sus otros lados como su gráfica. Los triángulos más altos se corresponden con clases de homología con mayor persistencia, mientras que los triángulos de menor altura se corresponden con clases de duración más corta.

Consideremos ahora todas las funciones $\Lambda_p(t)$. El paisaje de persistencia es un resumen de todas ellas en forma de una única función.

Definición 5.3. *Sea D un diagrama de persistencia. Definimos su paisaje de persistencia como la función $\lambda : \mathbb{N} \times \mathbb{R}_+ \to \mathbb{R}$ dada por:*

$$\lambda(k, t) = \underset{p \in D}{\text{kmax}} \Lambda_p(t), \tag{5.4}$$

donde kmax *es el k-ésimo valor más alto del conjunto.*

También es habitual presentar el paisaje de persistencia a partir de sus funciones componente $\lambda_k(t),\ k \in \mathbb{N}$ dadas como $\lambda_k(t) = \lambda(k, t)$ [13]. A cada función componente $\lambda_k(t)$ se le llama k-ésimo paisaje de persistencia. Para todo k, se tiene que $\lambda_k(t) \geq 0$, $\lambda_k(t) \geq \lambda_{k+1}(t)$ y $\lambda_k(t)$ es una función 1-Lipschitziana. Además, se puede relacionar con los números persistentes de Betti mediante el siguiente resultado:

Proposición 5.1. *Sea D un diagrama y sea λ su paisaje de persistencia. Entonces para todo $k \in \mathbb{N}$ y para todo $t \in \mathbb{R}_+$, se tiene que:*

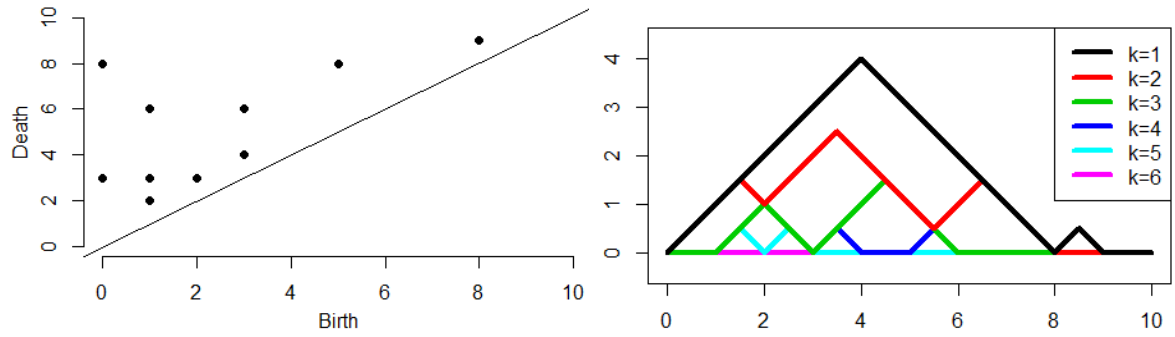
$$\lambda_k(t) = \sup\{m \geq 0 | \beta^{t-m,t+m} \geq k\}, \tag{5.5}$$

y si el supremo no existe entonces $\lambda_k(t) = 0$.

Demostración. Sea $t \in \mathbb{R}_+$. Notemos que la transformación $(b, d) \mapsto (\frac{b+d}{2}, \frac{d-b}{2})$ hace que $(t - m, t + m) \mapsto (t, m)$ Por el lema de los triángulos 3.2, el número de Betti $\beta^{t-m,t+m}$ coincide con el número de triángulos $\mathcal{T}_{b,d}$ que se superponen sobre $(t - m, t + m)$, y por tanto coincide con el número de funciones triangulares tales que $\Lambda_p(t) \geq m$.

Fijado t, la función $m \mapsto \beta^{t-m,t+m}$ es escalonada. Además si $m < m'$, entonces $(t - m', t + m') \preceq (t - m, t + m)$ en $\mathbb{D}^1$ y, en virtud del lema 4.1, $\beta^{t-m',t+m'} \leq \beta^{t-m,t+m}$. Es decir, $m \mapsto \beta^{t-m,t+m}$ es decreciente.

Fijado t, sea $n := \beta^{t,t}$. Si $n < k$, tanto $\lambda_k(t)$ como $\sup\{m \geq 0 | \beta^{t-m,t+m} \geq k\}$ valen 0. Supongamos ahora que $n \geq k$. En este caso, el supremo sí existe. Cada vez que la función $m \mapsto \beta^{t-m,t+m}$ baja un escalón, la recta $(t - m, t + m)$ sale de un triángulo $\mathcal{T}_{b,d}$, y por tanto la recta (t, m) cruza la gráfica de un triángulo Λ_p. Cuando se llega al valor m que marca el supremo, $(t-m, t+m)$ está en en el borde del k-ésimo triángulo $\mathcal{T}_{b,d}$ más alejado de la diagonal, y por tanto el punto (t, m) está en la gráfica de la k-ésima función Λ_p más elevada sobre el valor t. Entonces $\underset{p \in D}{\text{kmax}} \Lambda_p(t) = m$ y esto concluye la prueba. □



(a) Diagrama de persistencia (b) Paisaje de persistencia

Figura 5.1: Construcción del paisaje de persistencia a partir de un diagrama de persistencia

En virtud de este resultado, (5.5) puede ser usado como definición de los paisajes de persistencia. De hecho es la definición original que se dio en [13]. La prueba de la equivalencia entre las dos definiciones es de elaboración propia.

Esta función λ es equivalente a los diagramas de persistencia y los códigos de barras, y por tanto parametriza completamente la homología persistente sin perder información [13]. También cuenta con su propio teorema de estabilidad y tiene su versión multiparamétrica [68]. Además, al estar dentro del espacio vectorial de las funciones continuas sobre $\mathbb{N} \times \mathbb{R}_+$, ya se puede someter a operaciones como la suma o el producto por un escalar y cuenta con una norma.

Definición 5.4. *Sea un diagrama de persistencia D sin puntos infinitos y sea λ su paisaje de persistencia (acotado).*

- *Dado un $p \in [1, \infty)$, definimos la norma $||\lambda||_p$ como:*

$$||\lambda||_p = \left(\sum_{k=1}^{\infty} ||\lambda_k||_p^p \right)^{1/p}, \quad \textit{donde} \quad ||\lambda_k||_p^p = \int_{\mathbb{R}_+} |\lambda_k(t)|^p dt \tag{5.6}$$

- *La norma infinito se define como $||\lambda||_\infty = ||\lambda_1||_\infty = \sup_{t \in \mathbb{R}} \lambda_1(t)$.*

Si nuestro paisaje de persistencia λ proviene de un diagrama con un número finito de puntos finitos, entonces $\lambda \in \mathcal{L}^p(\mathbb{N} \times \mathbb{R}_+)$ para todo $p \geq 1$. En particular pertenece al espacio de Lebesgue $L^p(\mathbb{N} \times \mathbb{R}_+) = \mathcal{L}^p(\mathbb{N} \times \mathbb{R}_+)/\sim$, donde $f \sim g$ si y solo si $||f - g||_p = 0$. Los espacios $L^p(\mathbb{N} \times \mathbb{R}_+)$ son espacios de Banach, y por tanto podemos aplicar en los próximos apartados la teoría de variables aleatorias con valores en espacios de Banach. De ahora en adelante omitiremos el dominio de las funciones al nombrar a estos espacios.

Los paisajes de persistencia pueden ser un poco incómodos porque tienen un número infinito de componentes, aunque a partir de un cierto entero m todas las componentes son idénticamente nulas. Por eso definimos otra forma de resumir un diagrama de persistencia, llamada silueta ponderada, que sólo tiene una variable e integra información de todas las componentes del paisaje λ [24].

Definición 5.5. *Sea D un diagrama de persistencia y sean $\Lambda_1, \cdots, \Lambda_m$ sus funciones triangulares asociadas. Dado un número $p \in (0, \infty)$, definimos la silueta ponderada como una función $\phi^p : \mathbb{R}_+ \to \mathbb{R}$ dada por:*

$$\phi^p(t) = \frac{\sum_{j=1}^m |d_j - b_j|^p \Lambda_j(t)}{\sum_{j=1}^m |d_j - b_j|^p} \tag{5.7}$$

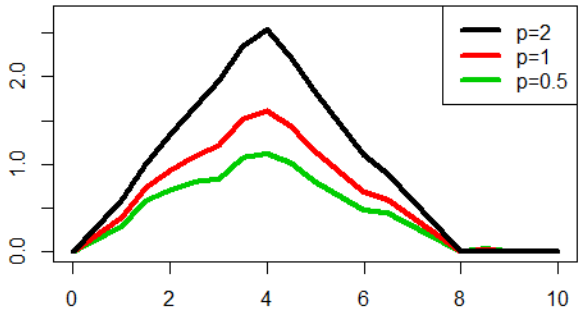


Figura 5.2: Siluetas ponderadas del diagrama de la figura 5.1

Esta silueta es pues una media ponderada de las funciones triangulares Λ_j. Cuando p es un valor bajo, la silueta está dominada por los triángulos de baja persistencia, mientras que si p es alto, está dominada por los triángulos más persistentes. Si el diagrama de persistencia D tiene un punto infinito, no podemos aplicar esta fórmula. Sí podríamos aplicarla si cambiamos la definición, cambiando los pesos $|d_j - b_j|^p$ por otros más generales.

Las siluetas ponderadas no dan una descripción completa de los módulos de homología, pero hacen un buen resumen de su estructura. Además, al igual que el paisaje de persistencia λ, son funciones 1-Lipschitzianas y si están acotadas pertenecen al espacio de Lebesgue L^p para todo $p \geq 1$. Entonces, toda la teoría de probabilidad que desarrollaremos en la próxima sección para paisajes también será válida para las siluetas de persistencia.

5.2. Teoría de probabilidad sobre paisajes y siluetas

Sea el espacio probabilístico $(\Omega, \mathcal{A}, P)$. Sobre él consideramos una variable aleatoria X tal que para cada $\omega \in \Omega$, $X(\omega)$ es un conjunto de datos en $\mathbb{R}^d$, y una variable aleatoria Λ tal que $\Lambda(\omega) = \lambda(X(\omega)) = \lambda$ es el paisaje de persistencia asociado al conjunto $X(\omega)$, suponiendo que hemos fijado un método para construir filtraciones. Λ es entonces una variable aleatoria que toma valores en el espacio de Banach separable L^p. Toda la teoría que describimos a continuación se desarrolla en el artículo [13].

Definición 5.6. *Sean $X_1, \cdots, X_n$ n variables independientes e idénticamente distribuidas a X, y sean $\Lambda^1, \cdots, \Lambda^n$ sus paisajes respectivos. Usando la estructura de espacio vectorial de L^p, definimos el paisaje medio $\bar{\Lambda}^n$ como la media puntual, es decir, $\bar{\Lambda}^n(\omega) = \bar{\lambda}^n$, donde*

$$\bar{\lambda}^n(k,t) = \frac{1}{n}\sum_{i=1}^{n} \lambda^i(k,t) \tag{5.8}$$

Nótese que la media aritmética de n paisajes de persistencia no es un paisaje de persistencia en general, pero al igual que ellos satisface que $\bar{\lambda}^n \geq 0$, $\bar{\lambda}^n_k \geq \bar{\lambda}^n_{k+1}$ y es 1-Lipschitziana.

Ahora introduciremos unos conceptos generales de variables aleatorias sobre espacios de Banach $\mathfrak{B}$ con norma $||\cdot||$. Consideremos de nuevo nuestro espacio probabilístico y una variable aleatoria $V : (\Omega, \mathcal{A}, P) \to \mathfrak{B}$. La variable compuesta $||V|| : (\Omega, \mathcal{A}, P) \to \mathbb{R}$ dada por $||V||(\omega) = ||V(\omega)||$ es una variable aleatoria real. Igualmente, dada una función $f \in \mathfrak{B}^*$, la composición $f(V) : (\Omega, \mathcal{A}, P) \to \mathbb{R}$ es también una variable aleatoria real.

Ya sabemos que para una variable aleatoria real Y, su media o valor esperado se calcula como $E[Y] = \int_\Omega Y(\omega) dP(\omega)$. En el caso de una variable aleatoria V con valores sobre $\mathfrak{B}$, el concepto de valor esperado se cambia por el de integral de Pettis. Un elemento $E[V] \in \mathfrak{B}$ se dice que es la integral de Pettis de V si se verifica que $E[f(V)] = f(E[V])$ para toda función

$f : \mathfrak{B} \to \mathbb{R}$ lineal y continua. En general, una variable V tiene una integral de Pettis si y sólo si $E||V|| < \infty$.

Las variables aleatorias sobre espacios de Banach tienen su propia versión de la ley de los grandes números y el teorema central del límite [41], que se pueden aplicar a los paisajes de persistencia.

Teorema 5.1 (ley fuerte de los grandes números para paisajes)**.** *Sea $\{\Lambda^n\}_{n\in\mathbb{N}}$ una sucesión de variables independientes e idénticamente distribuidas a Λ. La sucesión de los paisajes medios $\bar{\Lambda}^n$ converge casi seguro a $E[\Lambda]$ si y sólo si $E||\Lambda||_p < \infty$.*

Véase que este resultado es cierto independientemente de la norma $||\cdot||_p$ de que estemos dotando a nuestros paisajes de persistencia. El teorema central del límite, en cambio, será algo más restrictivo.

Teorema 5.2 (teorema central del límite para paisajes)**.** *Sea $\{\Lambda^n\}_{n\in\mathbb{N}}$ y supongamos que $p \geq 2$. Si $E||\Lambda||_p < \infty$ y $E(||\Lambda||_p^2) < \infty$, entonces la sucesión $\sqrt{n}[\bar{\Lambda}^n - E[\Lambda]]$ converge débilmente a una variable gaussiana en el espacio L^p con media nula y la misma estructura de covarianzas que Λ.*

El teorema central del límite no nos será útil en esta versión, sino que necesitaremos aplicar un funcional a los paisajes de persistencia para obtener una variable real que satisfaga el teorema central del límite en su versión más clásica. Esto nos lleva al siguiente corolario [13]:

Corolario 5.1. *Supongamos que $p \geq 2$ y que $E||\Lambda||_p < \infty$ y $E(||\Lambda||_p^2) < \infty$. Dada una función $f \in L^q$, con $\frac{1}{p} + \frac{1}{q} = 1$, sea la variable aleatoria real*

$$Y = \int_{\mathbb{N}\times\mathbb{R}_+} f\Lambda = ||f\Lambda||_1 \tag{5.9}$$

Entonces, $\sqrt{n}[\bar{Y}^n - E[Y]]$ converge en ley a una variable normal $N(0, Var(Y))$.

Gracias a este corolario podemos crear nuestro primer contraste de hipótesis. Sean $X_1, \cdots, X_n$ copias idénticas de la variable X, y sean $X'_1, \cdots, X'_{n'}$ copias idénticas de la variable X'. Supongamos que sus correspondientes paisajes Λ, Λ' están en L^p, con $p \geq 2$. Sea un funcional $f \in L^q$ con $\frac{1}{p} + \frac{1}{q} = 1$, y definamos Y, Y' como lo hicimos en (5.9). Sean $\mu = E[Y]$ y $\mu' = E[Y']$. Si queremos contrastar la hipótesis nula $H_0 : \mu = \mu'$, podemos usar para ello los contrastes de hipotesis sobre poblaciones normales. Por ejemplo, podemos considerar el estadístico

$$z = \frac{\bar{Y} - \bar{Y}'}{\sqrt{\frac{S_Y^2}{n} + \frac{S_{Y'}^2}{n'}}}, \tag{5.10}$$

donde S^2 denota la cuasivarianza muestral. Este estadístico z bajo H_0 tiene asintóticamente la misma distribución que una normal $N(0, 1)$.

Para poder aplicar estos resultados, debemos escoger un funcional $f \in L^q$. La opción más simple pasa por usar una función f que sea constante e igual a 1 en un subconjunto de $\mathbb{N} \times \mathbb{R}_+$ que contenga a los soportes de $\Lambda(\omega)$ y $\Lambda'(\omega)$ para todo $\omega \in \Omega$, y nula en el resto del dominio. Si preferimos que el funcional sólo afecte a las K primeras componentes de cada paisaje, podemos restringir esta función para que se anule a partir de la componente $K + 1$. Para aumentar la potencia de estos tests, podemos usar todo un vector de funcionales $(f_1, \cdots, f_r)$ y aplicar un test T^2 de Hotelling, que también tiene su equivalente en la teoría de variables sobre espacios de Banach.

Como ya adelantamos en la sección previa, toda esta teoría de la probabilidad con variables sobre los espacios de Banach L^p aplica también en el contexto de las siluetas ponderadas. Por tanto, contamos con una ley fuerte de los grandes números para siluetas y un teorema central del límite para siluetas. Considerando un funcional $f \in L^q$ adecuado, también se les puede aplicar contrastes de hipótesis como el que acabamos de explicar.

5.3. Bandas de confianza basadas en *bootstrap*

Sea P una variable aleatoria d-dimensional con un soporte $\mathbb{M}$ no observable. Supondremos en todo momento que nuestro conjunto de datos S es una muestra de realizaciones independientes de P. Nuestro objetivo será aproximar $\mathbb{M}$ usando S. Consideraremos que siempre se hacen filtraciones basadas en funciones. En particular nos centraremos en dos casos: filtraciones por conjuntos de subnivel de la función *DTM* (véase la definición 1.17) y filtraciones por conjuntos de supernivel de un estimador de densidad con núcleo gaussiano (véase la definición 1.18).

En ambos casos la función sobre la que construimos la filtración depende de un subconjunto de $\mathbb{R}^d$ y de un parámetro. Fijado el parámetro, llamaremos f a la función definida con respecto a $\mathbb{M}$. Esta función f no es observable, pero podemos construir un estimador $\hat{f}$ definido con S. Nuestro objetivo en esta sección será dar una banda de confianza al nivel $(1-\alpha)$ para $\hat{f}$, de forma que si otra función se mantiene dentro de ella en todo su dominio, no haya evidencia significativa para afirmar que son funciones diferentes.

Para calcular esta banda de confianza, usaremos el algoritmo *bootstrap* de remuestreo con reemplazamiento [34].

Algoritmo 5.1 (banda *bootstrap* de confianza para una función)**.** *Sea $S = \{x_1, \cdots, x_N\}$ y sea $\hat{f}$ el estimador de f calculado sobre S.*

- *Tomamos una submuestra $x_1^*, \cdots, x_N^*$ de S con reemplazamiento y sobre ella calculamos el estimador $\hat{f}^*$.*
- *Teniendo $\hat{f}^*$, calculamos $\theta^* = \sqrt{N}||\hat{f} - \hat{f}^*||_\infty$.*
- *Repetimos los dos primeros pasos B veces, obteniendo $\hat{f}_1^*, \cdots, \hat{f}_B^*$ y $\theta_1^*, \cdots, \theta_B^*$.*
- *Calculamos el cuantil $q_\alpha = \text{ínf}\{q : \frac{1}{B}\sum_{j=1}^{B} I(\theta_j^* \geq q) \leq \alpha\}$.*
- *La banda de confianza es $C(\hat{f}) = [\hat{f} - \frac{q_\alpha}{\sqrt{N}}, \hat{f} + \frac{q_\alpha}{\sqrt{N}}]$*

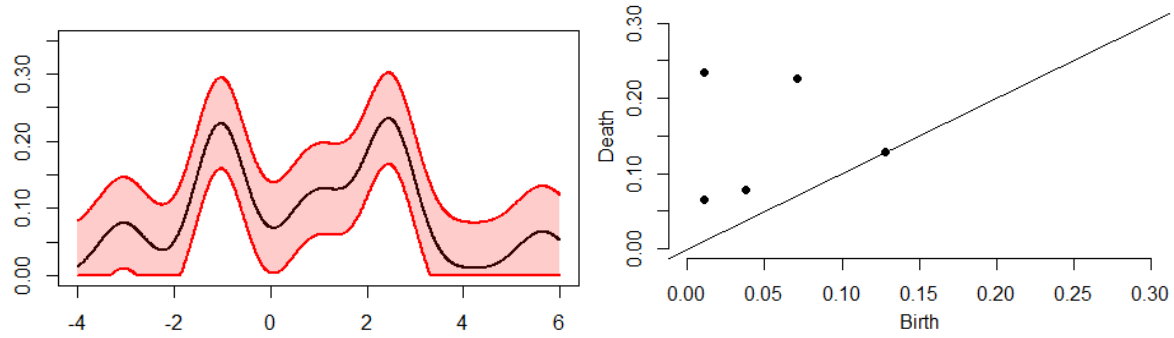
La validez de este método se prueba en [37] cuando f es un estimador de densidad cualquiera, y en [22] cuando f es una función *DTM*. La validez del método *bootstrap* significa que, asintóticamente,

$$P\left(\sqrt{N} \cdot ||f - \hat{f}||_\infty > q_\alpha\right) \to \alpha, \tag{5.11}$$

y podríamos usar estas bandas de confianza para hacer contrastes de hipótesis sobre f.

5.4. Bandas *bootstrap* de confianza para paisajes y siluetas

Supongamos que se fija un $k \in \mathbb{N}$ y cuando digamos λ nos referiremos a λ_k.El objetivo será ahora encontrar una banda de confianza para un paisaje o una silueta. La forma más sencilla



(a) Banda *bootstrap* (b) Diagrama asociado a esta función

Figura 5.3: Banda *bootstrap* para un estimador de densidad. El estimador se ha construido con 100 puntos tomados de la mixtura de cinco normales distintas.

de calcularla es aplicando el algoritmo *bootstrap* 5.1 pero poniendo $\theta^* = \sqrt{n}||\hat{\lambda} - \hat{\lambda}^*||_\infty$, donde cada paisaje $\hat{\lambda}^*$ se calcula a partir de $\hat{f}^*$ [23].

Si tenemos una muestra de paisajes $\lambda_1, \cdots, \lambda_n$ tomada de una distribución Λ y queremos hacer una banda de confianza para su paisaje medio $\bar{\lambda}_n$, podemos hacer una variante del algoritmo *bootstrap*, llamada *bootstrap* multiplicativo [24].

Algoritmo 5.2 (*bootstrap* multiplicativo para paisajes)**.** *Sean los paisajes de persistencia k-ésimos* $\lambda_1, \cdots, \lambda_n$.

- *Calculamos el paisaje medio* $\bar{\lambda}_n = \frac{1}{n}\sum_{i=1}^n \lambda_i$
- *Generamos* $\xi_1, \cdots, \xi_n \sim N(0,1)$ *independientes.*
- *Calculamos* $\theta^* = \frac{1}{\sqrt{n}}||\sum_{i=1}^n \xi_i(\lambda_i - \bar{\lambda}_n)||_\infty$.
- *Repetimos los dos pasos anteriores B veces, obteniendo* $\theta_1^*, \cdots, \theta_B^*$.
- *Calculamos el cuantil* $q_\alpha = \text{ínf}\{q : \frac{1}{B}\sum_{j=1}^B I(\theta_j^* \geq q) \leq \alpha\}$.
- *La banda de confianza es* $C(\bar{\lambda}_n) = [\bar{\lambda}_n - \frac{q_\alpha}{\sqrt{n}}, \bar{\lambda}_n + \frac{q_\alpha}{\sqrt{n}}]$

La validez del *bootstrap* multiplicativo se prueba en [24] e implica que, asintóticamente,

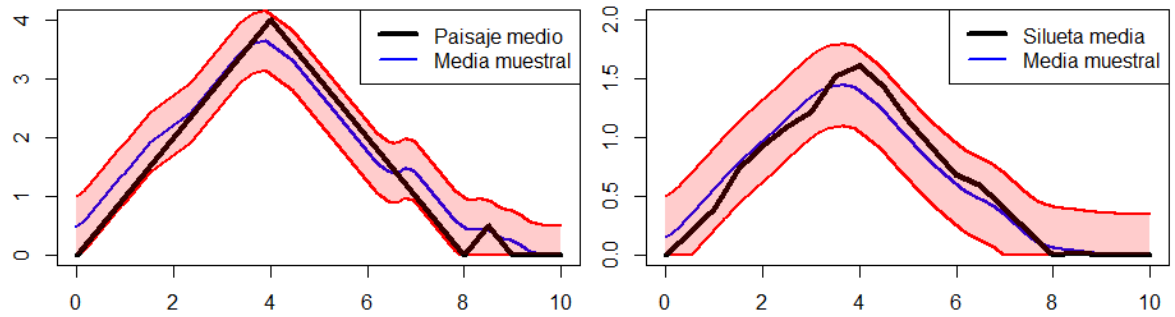
$$P\left(\sqrt{n} \cdot ||E[\Lambda] - \bar{\lambda}_n||_\infty > q_\alpha\right) \to \alpha$$

Por tanto, esta banda de confianza se puede aplicar para contrastar la hipótesis $H_0 : E[\Lambda] = \lambda$. Este método multiplicativo y la prueba de su validez también aplican en el caso de siluetas ponderadas.

Esta banda de confianza es igual de ancha en todo el dominio del paisaje medio $\bar{\lambda}_n$, pero podría ser más precisa para algunos valores t que para otros. Podemos resolver este problema definiendo una banda de confianza adaptativa, cuyo ancho varía en función de t. Para calcularla se usa una pequeña variante del algoritmo 5.2.

Algoritmo 5.3 (banda *bootstrap* adaptativa)**.** *Sean los paisajes de persistencia k-ésimos* $\lambda_1, \cdots, \lambda_n$.

- *Calculamos el paisaje medio* $\bar{\lambda}_n = \frac{1}{n}\sum_{i=1}^n \lambda_i$
- *Estimamos la desviación típica como:* $\hat{\sigma}(t) = \sqrt{\frac{1}{n}\sum_{i=1}^n (\lambda_i^2(t) - \bar{\lambda}_n^2(t))}$



(a) Banda de confianza para un paisaje (b) Banda de confianza para una silueta

Figura 5.4: Bandas de confianza basadas en *bootstrap* multiplicativo. La muestra de paisajes y siluetas se ha obtenido añadiendo pequeñas perturbaciones al diagrama de la figura 5.1. Véase que en ambos casos la banda de confianza contiene completamente a la función esperada.

- *Generamos* $\xi_1, \cdots, \xi_n \sim N(0,1)$ *independientes.*
- *Calculamos* $\theta^* = \frac{1}{\sqrt{n}} || \sum_{i=1}^{n} \xi_i \frac{\lambda_i(t) - \bar{\lambda}_n(t)}{\hat{\sigma}(t)} ||_\infty$.
- *Repetimos los dos pasos anteriores* B *veces, obteniendo* $\theta_1^*, \cdots, \theta_B^*$.
- *Calculamos el cuantil* $q_\alpha = \inf\{q : \frac{1}{B} \sum_{j=1}^{B} I(\theta_j^* \geq q) \leq \alpha\}$.
- *La banda de confianza es* $C(\bar{\lambda}_n) = [\bar{\lambda}_n - \frac{q_\alpha \hat{\sigma}(t)}{\sqrt{n}}, \bar{\lambda}_n + \frac{q_\alpha \hat{\sigma}(t)}{\sqrt{n}}]$

Esta banda de confianza adaptativa también es asintóticamente correcta [24], y se ajusta más a la variabilidad de los paisajes en cada t del dominio.

5.5. Conjuntos de confianza

En este apartado trataremos el problema de la detección de ruido en un diagrama de persistencia. Para ello debemos encontrar el equivalente a un intervalo de confianza para diagramas de persistencia, al que llamaremos un conjunto de confianza al nivel $(1-\alpha)$.

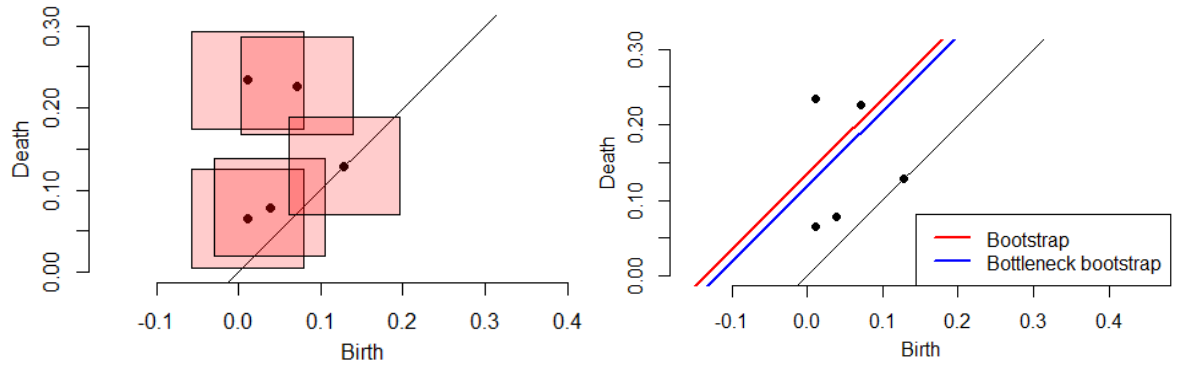
Supongamos que los diagramas provienen de filtraciones basadas en funciones como en las secciones anteriores. Sea D_f el diagrama de persistencia no observable asociado a la filtración por la función f. Nuestro conjunto S nos da unos estimadores $\hat{f}$ y $\hat{D}$. Aplicando el algoritmo *bootstrap* 5.1 para funciones, encontramos un valor q_α tal que asintóticamente $P(\sqrt{N} \cdot ||f - \hat{f}||_\infty > q_\alpha) \to \alpha$. Gracias al teorema de estabilidad 3.3 para diagramas, tenemos que:

$$P\left(\sqrt{N} \cdot d_B(D_f, \hat{D}) > q_\alpha\right) \leq P\left(\sqrt{N} \cdot ||f - \hat{f}||_\infty > q_\alpha\right) \to \alpha, \tag{5.12}$$

y hemos encontrado un conjunto de confianza asintóticamente correcto para $\hat{D}$ [37]. Si llamamos $c_\alpha = q_\alpha / \sqrt{N}$, el conjunto de confianza $C(c_\alpha)$ es la bola cerrada

$$C(c_\alpha) = \{D : d_B(\hat{D}, D) \leq c_\alpha\} \tag{5.13}$$

El conjunto $C(c_\alpha)$ se puede visualizar centrando en cada punto de $\hat{D}$ una caja de lado $2c_\alpha$, como se hace en la figura 5.5a. Un diagrama D pertenece a $C(c_\alpha)$ si tiene un punto en cada caja. Si la caja de un punto concreto interseca a la diagonal, entonces existe un diagrama en $C(c_\alpha)$ donde este punto se integra en la diagonal y desaparece. Entonces, dado nuestro diagrama $\hat{D}$, podemos eliminar todos los puntos cuyas cajas corten con la diagonal, y por la construcción



(a) Conjunto de confianza al 95 %

(b) Bandas de confianza al 95 %

Figura 5.5: Conjunto y bandas de confianza para el diagrama de la figura 5.3.

que hemos hecho sabemos que el nuevo diagrama no es significativamente distinto al original porque está dentro de $C(c_\alpha)$. Una alternativa a las cajas es dibujar sobre la diagonal una banda con altura $2c_\alpha$, como se hace en la figura 5.5b, y eliminar los puntos que caigan sobre esta banda, es decir, que tengan una persistencia menor que $2c_\alpha$.

Podemos hacer un conjunto de confianza más preciso si en vez de usar el algoritmo *bootstrap* 5.1 sobre las funciones lo usamos directamente sobre la distancia *bottleneck* . Este algoritmo se llama *bottleneck bootstrap*.

Algoritmo 5.4 (*bottleneck bootstrap* para diagramas)**.** *Sea* $S = \{x_1, \cdots, x_N\}$ *y sean* $\hat{f}$ *y* $\hat{D}$ *los estimadores de* f *y* D_f *calculados sobre* S*.*

- *Tomamos una submuestra* $x_1^*, \cdots, x_N^*$ *de* S *con reemplazamiento y sobre ella calculamos el estimador* $\hat{f}^*$*. A partir de* $\hat{f}^*$ *calculamos el diagrama de persistencia* D^**.*
- *Teniendo* D^**, calculamos* $\theta^* = \sqrt{N} \cdot d_B(D^*, \hat{D})$*.*
- *Repetimos los dos primeros pasos* B *veces, obteniendo* $D_1^*, \cdots, D_B^*$ *y* $\theta_1^*, \cdots, \theta_B^*$*.*
- *Calculamos el cuantil* $q_\alpha = \text{ínf}\{z : \frac{1}{B}\sum_{j=1}^{B} I(\theta_j^* \geq q) \leq \alpha\}$*, y* $c_\alpha = q_\alpha/\sqrt{N}$*.*
- *El conjunto de confianza es* $C(c_\alpha) = \{D : d_B(\hat{D}, D) \leq c_\alpha\}$

Este algoritmo es correcto independientemente de que f sea una función DTM o un estimador de densidad, y da un conjunto de confianza más preciso porque el algoritmo *bootstrap* 5.1 para funciones aprovecha que gracias al teorema 3.3 de estabilidad $d_B(D_f, \hat{D}) \leq ||f - \hat{f}||_\infty$ y encuentra una cota superior para $||f - \hat{f}||_\infty$, lo cual da un conjunto de confianza muy conservativo [22].

5.6. Análisis de máxima persistencia

A lo largo de todo este capítulo hemos supuesto que las filtraciones se construían a través de funciones. En particular nos hemos centrado en la función DTM, que depende de un parámetro m_0, y del estimador de densidad con núcleo gaussiano, que depende de un parámetro h que define su varianza. La elección de un buen parámetro para estas funciones es aún un problema sin resolver.

Definición 5.7. *Sea una filtración dada por una función f que depende de un parámetro m. Para cada m, sean $l_1(m), l_2(m), \cdots$ las persistencias de los diferentes puntos de su diagrama, y sea $c(m)$ la altura de su banda de confianza. Definimos:*

- $N(m) = \#\{i : l(i) > c(m)\}$
- $S(m) = \sum_i [l(i) - c(m)]_+$

$N(m)$ cuenta el número de clases de homología significativas, mientras que $S(m)$ nos indica cuánto persisten estas clases. En el artículo [22] se propone escoger el parámetro m que maximice $N(m)$, $S(m)$ o bien la vida significativa media $S(m)/N(m)$.

Capítulo 6

Aplicaciones del análisis topológico de datos

A pesar de lo reciente de esta rama de estudio, el análisis topológico de datos ya ha sido aplicado en todo tipo de áreas, tales como el reconocimiento de formas [31], análisis de redes [44], análisis del lenguaje natural [72], análisis de imágenes [17], biología computacional [42], diseño de fármacos [69], etc. En este capítulo haremos una breve introducción a algunas de sus aplicaciones.

6.1. *Machine learning* basado en homología persistente

En los capítulos anteriores hemos descrito diferentes objetos que representan la homología persistente, tales como los códigos de barras, las curvas de Betti, los diagramas de persistencia, los paisajes y las siluetas. Estos objetos pueden ser utilizados como variable de entrada en diversos algoritmos de aprendizaje automático, bien en su forma original o bien tras aplicarles alguna transformación.

La forma más simple de extraer características a partir de la homología persistente es hacer un resumen estadístico del código de barras. Por ejemplo extraer el máximo, el mínimo, la media y la varianza de los tiempos de nacimiento, los tiempos de muerte o las persistencias de cada intervalo [14].

Otros autores proponen construir vectores numéricos a partir de nuestros objetos con el método del empaquetado o *binning*. La idea es discretizar el rango de la filtración en varios intervalos del mismo tamaño y asignar a cada uno un valor que resuma una propiedad suya. Para un código de barras, podemos dividir su dominio en n intervalos y crear tres vectores con n componentes que cuenten respectivamente cuántos intervalos nacen, cuántos mueren y cuántos hay en total en cada intervalo. Para la curva de Betti, el *binning* es tan simple como almacenar sus valores para una serie de puntos equidistantes. Para un paisaje se puede repetir este proceso una vez por cada componente, obteniendo un vector distinto para cada una y para siluetas se puede aplicar también. Para un diagrama de persistencia rotado, podemos dar una malla con $n \times n$ cuadrados y crear un vector de n^2 componentes que cuente cuántos puntos hay en cada cuadrado. Estos vectores numéricos se pueden usar para algoritmos de *clustering*, máquinas de vector soporte, árboles de decisión, redes neuronales, etc [56].

En el caso de una bifiltración o una trifiltración, podemos crear matrices en dos y tres dimensiones si almacenamos el número de Betti para cada pareja o terna de parámetros. Al contrario que las representaciones vectoriales antes explicadas, estas matrices pueden ser tratadas como imágenes y se pueden combinar con redes neuronales convolucionales [15].

Otro enfoque para poder aplicar máquinas de vector soporte a diagramas o paisajes de persistencia pasa por definir sobre ellos diferentes medidas de semejanza o funciones núcleo cuyos valores se puedan almacenar en una matriz [56]. Las medidas de semejanza más conocidas son la distancia p de Wasserstein y la distancia *bottleneck* entre diagramas, cuyas fórmulas se describen en la definición 3.12. También podemos definir núcleos entre diagramas de persistencia.

Definición 6.1. *Sean dos diagramas de persistencia D y D'.*

- *El núcleo PSSK (persistent scale space kernel [58]) tiene por fórmula:*

$$\kappa_{PSSK}(D, D', \sigma) = \frac{1}{8\pi\sigma} \sum_{x \in D, x' \in D'} e^{-\frac{||x-x'||_\infty^2}{\sigma}} - e^{-\frac{||\bar{x}-x'||_\infty^2}{\sigma}}, \tag{6.1}$$

  donde $\bar{x}$ es el punto simétrico de x respecto a la diagonal.

- *El núcleo GTK (geodesic topological kernel [52]) tiene por fórmula:*

$$\kappa_{GTK}(D, D', \sigma) = e^{\{\frac{1}{h} W_2(D,D')^2\}} \tag{6.2}$$

- *El núcleo GLK (geodesic laplacian kernel [52]) tiene por fórmula:*

$$\kappa_{GLK}(D, D', \sigma) = e^{\{\frac{1}{h} W_2(D,D')\}} \tag{6.3}$$

También es posible definir núcleos para paisajes o siluetas.

Definición 6.2. *Sean dos paisajes de persistencia m-ésimos λ_m y λ'_m. El núcleo GPHK (global persistent homology kernel [56]) tiene por fórmula:*

$$\kappa_{GPHK}(\lambda_m, \lambda'_m) = \langle \lambda_m, \lambda'_m \rangle = \int_{\mathbb{R}} \lambda_m(x) \cdot \lambda'_m(x) dx \tag{6.4}$$

Utilizando herramientas como éstas, podemos combinar nuestros resultados topológicos con cualquier algoritmo de aprendizaje automático que necesitemos.

6.2. Análisis de formas

En los últimos años se están creando bases de datos de formas tridimensionales cada vez más grandes, siendo necesario desarrollar nuevos algoritmos para su procesamiento. En el artículo [11], se emplea el método de las bolsas de palabras para la clasificación de estas formas.

Supongamos que tenemos una base con M formas $3D$ almacenadas como complejos simpliciales de dimensión 2 con vértices V_i. El primer paso consiste en definir unas ciertas funciones $f_i : V_i \to \mathbb{R}^n$ que describan la geometría local de la forma en torno a cada vértice. Después estas funciones se proyectan sobre un conjunto de puntos $W = \{w_k | k = 1, \cdots, K\}$ al que llamaremos diccionario o *codebook*. La composición de las f_i con la proyección sobre W son las llamadas funciones de palabras $C_i : V_i \to \mathbb{R}^K$, y a los puntos de $\mathbb{R}^K$ que forman su imagen se les llama simplemente palabras.

El último paso de este proceso consiste en encontrar un buen resumen de todas estas funciones C_i en forma de un único vector compacto $P = (P_i)_{i=1,\cdots,M}$ que pueda ser usado por ejemplo por una máquina de vector soporte. Este paso se conoce como *pooling* y se puede realizar de varias formas. Dos muy comunes son el *pooling* de la suma y el del máximo [70], cuyas componentes P_i vienen dadas por:

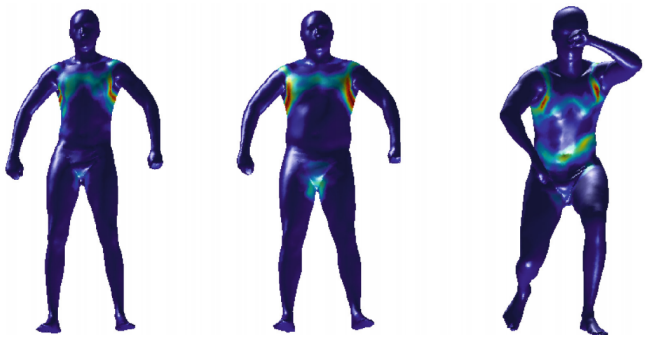


Figura 6.1: Ejemplo de una función de palabras sobre tres formas distintas de la base SHREC 2014. Imagen extraída de [11]

$$P_i = (SumPool(C_{i,1}), \cdots, SumPool(C_{i,K}) = \left(\sum_{x \in V_i} C_i(x)_1, \cdots, \sum_{x \in V_i} C_i(x)_K \right)$$

$$P_i = (MaxPool(C_{i,1}), \cdots, MaxPool(C_{i,K}) = \left(\max_{x \in V_i} C_i(x)_1, \cdots, \max_{x \in V_i} C_i(x)_K \right)$$

El *pooling* del máximo es más robusto que el de la suma, y hace un resumen de los máximos locales de las f_i, sin importar la "prominencia" del pico que representan. Los autores del artículo [11] proponen un nuevo método más robusto que los anteriores basado en homología persistente de dimensión 0, llamado *pooling* topológico, que sí tiene en cuenta la prominencia de estos máximos.

Definición 6.3. *Dada una función f sobre un grafo G, definimos:*

$$TopoPool(f)_i = p_i(D_f),$$

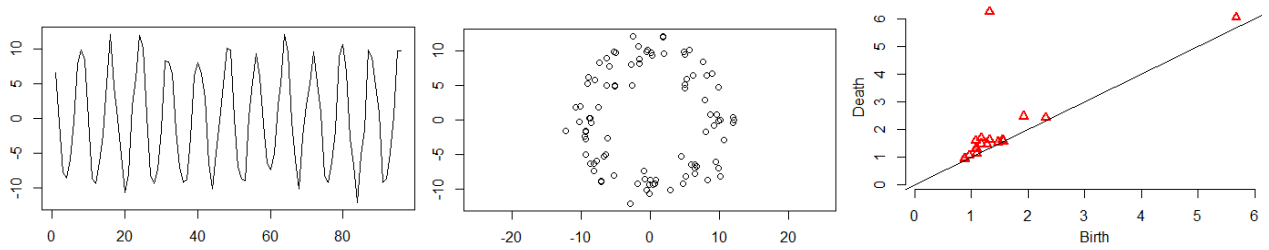
donde D_f es el diagrama de persistencia de dimensión 0 asociado a los conjuntos de supernivel de f y p_i es la persistencia de su i-ésimo punto más alejado de la diagonal. Si D_f no tuviera i puntos, se asigna el valor 0.

Gracias al teorema de estabilidad 3.3 de diagramas, el *pooling* topológico es estable y ofrece mejores prestaciones que los métodos convencionales. La comparación entre los tres métodos de *pooling* se hizo usando la base de datos SHREC 2014 [55], que contiene 400 formas de 40 personas en 10 poses distintas.

6.3. Análisis de series temporales

Existen diferentes vías para usar la homología persistente y sus representaciones en el estudio de series temporales tanto numéricas como categóricas [57]. Los datos de una serie temporal no pueden ser usados directamente para construir una filtración, sino que deben ser transformados previamente en una nube de puntos. Para ello contamos con dos métodos distintos: la inmersión de retrasos de Takens y el metodo de ventanas deslizantes.

Sea una serie temporal $\{x_t | t = 1, 2, \cdots, T\}$. La inmersión de retrasos de Takens [63] convierte la serie en una nube de puntos en $\mathbb{R}^d$ dados por $v_i = (x_i, x_{i+\tau}, x_{i+2\tau}, \cdots, x_{i+(d-1)\tau})$, donde d indica la dimensión de los puntos creados y τ es un parámetro de retraso. Existen multitud de propuestas para escoger unos valores d, τ apropiados.



(a) Serie temporal (b) Nube de puntos (c) Diagrama de persistencia

Figura 6.2: Detección de señales periódicas con homología persistente. La nube de puntos se ha construido con el método de retrasos de Takens.

El método de las ventanas deslizantes es una alternativa al método de Takens para detectar periodicidades en series con ruido, con demasiada separación temporal o con separaciones desiguales [54]. El primer paso, opcional, es eliminar ruido aplicando por ejemplo una media móvil. El segundo es interpolar esta serie suavizada con un *spline* cúbico g. El tercero es seleccionar N puntos equidistantes $1 = i_1 < i_2 < \cdots < i_N = T$, dando lugar a una nueva serie $\{g(i_1), \cdots, g(i_N)\}$. El cuarto paso es aplicar el método de Takens a esta nueva serie con unos d, τ adecuados. El último paso, también opcional, es estandarizar la nube de puntos obtenida.

Si nuestra serie temporal es categórica o contiene fuertes discontinuidades, podemos aplicarle una transformada de Walsh-Fourier antes de aplicarle alguno de los métodos anteriores [60].

Una vez construida una nube de puntos, podemos calcular una filtración sobre ella con cualquiera de los métodos explicados en el capítulo 1 y a partir de ella obtener un diagrama de persistencia, un paisaje, una silueta o cualquier otra representación que nos pueda ser útil.

La homología persistente ha resultado ser una herramienta muy interesante en el estudio de series temporales, dando buenos resultados en la detección de señales periódicas ocultas, *clustering*, clasificación de series temporales y detección temprana de cambios estructurales. Por ejemplo, un ciclo de dimensión 1 de larga persistencia indica la existencia de una señal periódica en la serie, como se puede ver en la figura 6.2. Se puede encontrar un resumen de las otras aplicaciones en el artículo [57], que sirve de introducción a toda esta teoría.

6.4. Algoritmo *Mapper* en biomedicina

El análisis topológico de datos fue usado en [51] para el estudio de información genómica en células tumorales de pacientes con cáncer de mama, obteniendo resultados nuevos que por ejemplo el *clustering* no fue capaz de detectar.

Este estudio se realizó con una base de datos de 295 pacientes con cáncer de mama [67]. A cada una de ellas se les analizaron 262 genes de sus células tumorales, obteniendo los vectores $T_1, \cdots, T_{295}$. Cada vector se comparó con un modelo lineal que representa una situación de salud, obteniendo la descomposición $T_i = N_i + D_i$, donde N_i es la componente normal de T_i y D_i es su componente de enfermedad. Unos D_i alejados del vector nulo se corresponden con fuertes desviaciones con respecto al estado normal de salud. Una vez separados los datos en componente normal y componente enferma, se descartaron los genes que no presentaban desviaciones significativas del estado de salud. Por último se aplicó el algoritmo *Mapper* 1.2 a los datos usando como funciones de filtro potencias k-ésimas de la norma de L^p. Para calcular los *clusters* del algoritmo *Mapper*, usamos la distancia de correlación entre los datos.

Podemos ver el grafo resultante al usar $p = 2$ y $k = 4$ en la figura 6.3. Este grafo tiene tres ramas principales. La rama de la izquierda, con los puntos en azul, se corresponde con los tumores que tienen un tipo genómico parecido al estado normal de salud. La rama superior

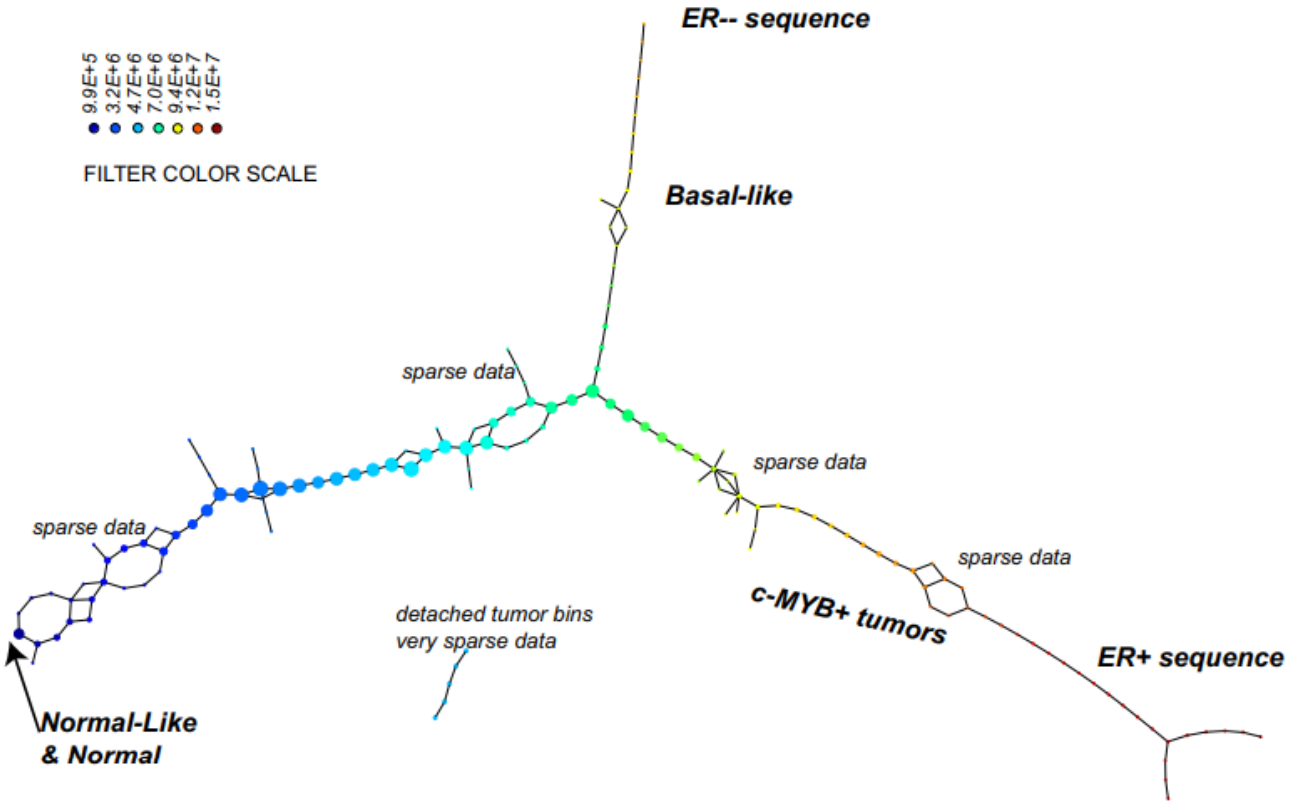


Figura 6.3: Realización del algoritmo *Mapper* sobre la base de datos del cáncer de mama. Imagen extraída de [51]

corresponde con los tumores de la secuencia ER^-, con una gran desviación con respecto al estado normal y de los que la mayoría son tumores basales. Existe una tercera secuencia larga de tumores con una gran desviación con respecto al estado normal, la secuencia ER^+. Se puede observar dentro de la secuencia ER^+ un subgrupo de puntos rodeado por dos áreas más dispersas, al que los autores llamaron $c - MYB^+$.

El subgrupo $c - MYB^+$, una vez detectado, ha sido estudiado y ha demostrado tener bastante uniformidad en su estructura molecular, un 100 % de supervivencia y un riesgo nulo de reaparición. También se puede detectar usando otros parámetros para el algoritmo *Mapper* y se ha validado aplicando el mismo procedimiento a otros conjuntos de datos de cáncer de mama. Es decir, estos tumores realmente forman un subgrupo totalmente diferenciado de los ya conocidos con anterioridad.

En cambio, el subgrupo $c-MYB^+$ no puede ser distinguido con técnicas más antiguas como los algoritmos de *clustering*. En la figura 6.4, se puede ver cómo el *clustering* dispersa los tumores del subgrupo $c-MYB^+$ en diversos *clusters* sin encontrar una relación significativa entre ellos. Esto nos da a entender que el análisis topológico de datos puede aportar información realmente novedosa y vale la pena profundizar en este tipo de técnicas para otros estudios similares.

6.5. *Software* para el análisis topológico de datos

Terminamos este último capítulo haciendo un repaso a los distintos *softwares* que se pueden usar para el cálculo de las herramientas que hemos definido a lo largo de todo este trabajo. Entre ellos se encuentran Javaplex [3], Perseus [50], Dipha [7], Dionysus [1], jHoles [9], GUDHI [64], Ripser [6], PHAT [8], DIPHA [7] y R. Centraremos nuestra atención en el *software* R, enfocado a usuarios con conocimientos de estadística y extensamente estudiado en este doble

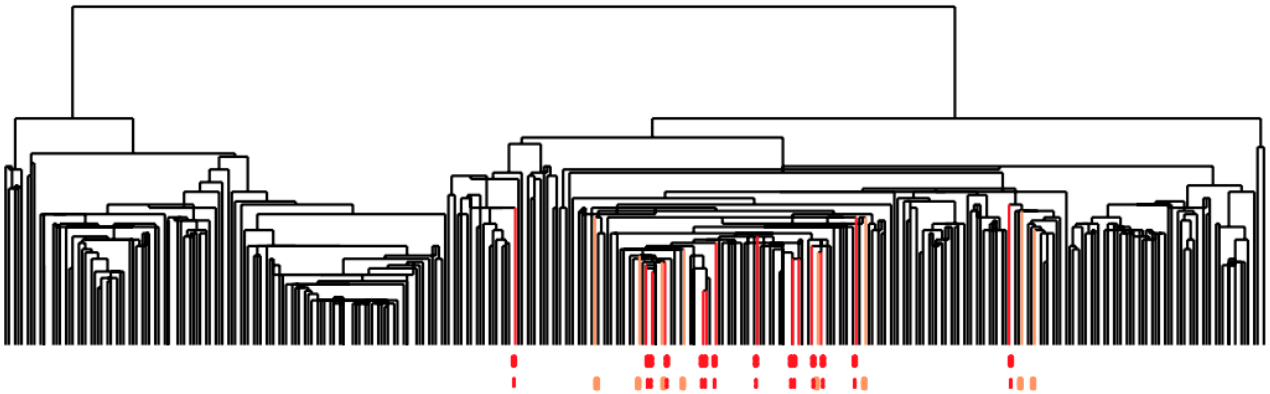


Figura 6.4: Dendrograma de la base de datos tras aplicar un algoritmo de *clustering*. Las barras rojas representan tumores del subgrupo $c - MYB^+$. Imagen extraída de [51]

grado.

La librería TDA de R fue desarrollada con vistas al tratamiento estadístico de la homología persistente [36]. Con ella se pueden calcular filtraciones basadas en el complejo *alpha* y Vietoris-Rips, filtraciones basadas en funciones, homología persistente con coeficientes en $\mathbb{Z}/\mathbb{Z}2$, códigos de barras y diagramas de persistencia, distancias *bottleneck* y de Wasserstein, paisajes, siluetas, bandas de confianza para funciones, bandas para paisajes y siluetas con el *bootstrap* multiplicativo, conjuntos de confianza con el *bottleneck bootstrap* y se puede hacer análisis de máxima persistencia. Casi todas las imágenes que figuran en este trabajo han sido creadas con las funciones que ofrece esta librería.

La librería TDAmapper [53] sirve para crear el grafo *Mapper* de un conjunto de datos para una o dos funciones de filtro. Los resultados de estas funciones se pueden combinar con las funciones de la librería igraph para obtener una representación visual del grafo.

Existen algunas funcionalidades que aún no han sido implementadas en este lenguaje de programación, como el cálculo de un complejo de Čech o un complejo *witness*, el cálculo de la homología simplicial con coeficientes arbitrarios o el cálculo de multifiltraciones. Para el cálculo de los complejos de Čech y *witness* podemos usar la librería GUDHI de C++, o bien la librería JavaPlex de Java. Para el cálculo de la homología simplicial con coeficientes en un anillo cualquiera podemos usar la librería CHomP de C++ o el *software* SageMath [73]. Para el cálculo de la homología multipersistente tal y como se hace en la sección 4.1 podemos usar la librería CoCoA de C++ [2] y el *software* Macaulay2 [39].

Concluimos presentando en el anexo una propuesta de elaboración propia para la implementación en R del cálculo de complejos de Čech, complejos *witness* y una función para representar complejos simpiciales cuyos puntos están en el plano.

Bibliografía

[1] Dionysus, the persistent homology software. Available at `https://www.mrzv.org/software/dionysus/`.

[2] J. Abbott, A. M. Bigatti, and L. Robbiano. CoCoA: a system for doing Computations in Commutative Algebra. Available at `http://cocoa.dima.unige.it`.

[3] H. Adams, A. Tausz, and M. Vejdemo-Johansson. Javaplex: A research software package for persistent (co) homology. In *International Congress on Mathematical Software*, pages 129–136. Springer, 2014.

[4] American Mathematical Society and A. J. Zomorodian, editors. *Advances in applied and computational topology: AMS Short Course on Computational Topology, January 4-5, 2011, New Orleans, Louisiana*. Number v. 70 in Proceedings of symposia in applied mathematics. American Mathematical Society, Providence, R.I, 2012.

[5] M. Artin. *Algebra*. Pearson, 2 edition, 2018.

[6] U. Bauer. Ripser: a lean c++ code for the computation of vietoris–rips persistence barcodes. *Software available at https://github. com/Ripser/ripser*, 436, 2017.

[7] U. Bauer, M. Kerber, and J. Reininghaus. Distributed computation of persistent homology. In *2014 proceedings of the sixteenth workshop on algorithm engineering and experiments (ALENEX)*, pages 31–38. SIAM, 2014.

[8] U. Bauer, M. Kerber, J. Reininghaus, and H. Wagner. Phat–persistent homology algorithms toolbox. *Journal of symbolic computation*, 78:76–90, 2017.

[9] J. Binchi, E. Merelli, M. Rucco, G. Petri, and F. Vaccarino. jholes: A tool for understanding biological complex networks via clique weight rank persistent homology. *Electron. Notes Theor. Comput. Sci.*, 306:5–18, 2014.

[10] J.-D. Boissonnat, O. Devillers, and S. Hornus. Incremental construction of the delaunay triangulation and the delaunay graph in medium dimension. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 208–216, 2009.

[11] T. Bonis, M. Ovsjanikov, S. Oudot, and F. Chazal. Persistence-based pooling for shape pose recognition. In *International workshop on computational topology in image context*, pages 19–29. Springer, 2016.

[12] N. Bourbaki. Elements of mathematics. algebra, part i: Chapters 1-3. *Translated from the French, Hermann, Paris*, 1974.

[13] P. Bubenik. Statistical topological data analysis using persistence landscapes. *The Journal of Machine Learning Research*, 16(1):77–102, 2015.

[14] Z. Cang, L. Mu, K. Wu, K. Opron, K. Xia, and G.-W. Wei. A topological approach for protein classification. *Computational and Mathematical Biophysics*, 1(open-issue), 2015.

[15] Z. Cang and G.-W. Wei. Topologynet: Topology based deep convolutional and multi-task neural networks for biomolecular property predictions. *PLoS computational biology*, 13(7):e1005690, 2017.

[16] G. Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, Jan. 2009.

[17] G. Carlsson, T. Ishkhanov, V. De Silva, and A. Zomorodian. On the local behavior of spaces of natural images. *International journal of computer vision*, 76(1):1–12, 2008.

[18] G. Carlsson, G. Singh, and A. Zomorodian. Computing multidimensional persistence. In *International Symposium on Algorithms and Computation*, pages 730–739. Springer, 2009.

[19] G. Carlsson and A. Zomorodian. The Theory of Multidimensional Persistence. *Discrete & Computational Geometry*, 42(1):71–93, July 2009.

[20] A. Cerri, B. D. Fabio, M. Ferri, P. Frosini, and C. Landi. Betti numbers in multidimensional persistent homology are stable functions. *Mathematical Methods in the Applied Sciences*, 36(12):1543–1557, Aug. 2013.

[21] F. Chazal, D. Cohen-Steiner, and Q. Merigot. Geometric inference for probability measures. *Foundations of Computational Mathematics*, 11:733–751, 12 2011.

[22] F. Chazal, B. Fasy, F. Lecci, B. Michel, A. Rinaldo, A. Rinaldo, and L. Wasserman. Robust topological inference: Distance to a measure and kernel distance. *The Journal of Machine Learning Research*, 18(1):5845–5884, 2017.

[23] F. Chazal, B. Fasy, F. Lecci, A. Rinaldo, A. Singh, and L. Wasserman. On the Bootstrap for Persistence Diagrams and Landscapes. *Modeling and Analysis of Information Systems*, 20(6):111–120, Mar. 2015.

[24] F. Chazal, B. T. Fasy, F. Lecci, A. Rinaldo, and L. Wasserman. Stochastic Convergence of Persistence Landscapes and Silhouettes. In *Annual Symposium on Computational Geometry - SOCG'14*, pages 474–483, Kyoto, Japan, 2014. ACM Press.

[25] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of Persistence Diagrams. *Discrete & Computational Geometry*, 37(1):103–120, Jan. 2007.

[26] A. Collins, A. Zomorodian, G. Carlsson, and L. Guibas. A barcode shape descriptor for curve point cloud data.

[27] T. Cormen and E. Leiserson. Charles, and rl rivest, introduction to algorithms, ser, 1990.

[28] D. A. Cox, J. Little, and D. O'shea. *Using algebraic geometry*, volume 185. Springer Science & Business Media, 2006.

[29] V. De Silva and G. E. Carlsson. Topological estimation using witness complexes. *SPBG*, 4:157–166, 2004.

[30] C. J. A. Delfinado and H. Edelsbrunner. An incremental algorithm for betti numbers of simplicial complexes. In *Proceedings of the ninth annual symposium on Computational geometry*, pages 232–239, 1993.

[31] B. Di Fabio and C. Landi. A mayer–vietoris formula for persistent homology with an application to shape recognition in the presence of occlusions. *Foundations of Computational Mathematics*, 11(5):499, 2011.

[32] D. S. Dummit and R. M. Foote. *Abstract algebra*, volume 3. Wiley Hoboken, 2004.

[33] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. In *Proceedings 41st annual symposium on foundations of computer science*, pages 454–463. IEEE, 2000.

[34] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[35] D. Eisenbud. Commutative algebra, with a view toward algebraic geometry grad. *Texts in Math*, 150:145–153, 1995.

[36] B. T. Fasy, J. Kim, F. Lecci, C. Maria, D. L. Millman, V. R. T. included GUDHI is authored by Clement Maria, D. by Dmitriy Morozov, P. by Ulrich Bauer, M. Kerber, and J. Reininghaus. *TDA: Statistical Tools for Topological Data Analysis*, 2019. R package version 1.6.9.

[37] B. T. Fasy, F. Lecci, A. Rinaldo, L. Wasserman, S. Balakrishnan, and A. Singh. Confidence sets for persistence diagrams. *The Annals of Statistics*, 42(6):2301–2339, Dec. 2014.

[38] J. Giesen, F. Cazals, M. Pauly, and A. Zomorodian. The conformal alpha shape filtration. *The Visual Computer*, 22:531–540, 08 2006.

[39] D. R. Grayson and M. E. Stillman. Macaulay2, a software system for research in algebraic geometry. Available at `https://faculty.math.illinois.edu/Macaulay2/`.

[40] A. Hatcher. *Algebraic topology*. Cambridge University Press, Cambridge ; New York, 2002.

[41] J. Hoffmann-Jorgensen, G. Pisier, et al. The law of large numbers and the central limit theorem in banach spaces. *The Annals of Probability*, 4(4):587–599, 1976.

[42] P. M. Kasson, A. Zomorodian, S. Park, N. Singhal, L. J. Guibas, and V. S. Pande. Persistent voids: a new structural metric for membrane fusion. *Bioinformatics*, 23(14):1753–1759, 2007.

[43] D. Kozlov. *Combinatorial algebraic topology*, volume 21. Springer Science & Business Media, 2007.

[44] H. Lee, H. Kang, M. K. Chung, B.-N. Kim, and D. S. Lee. Persistent brain network homology from the perspective of dendrogram. *IEEE transactions on medical imaging*, 31(12):2267–2277, 2012.

[45] L. Lovász. Topological methods in combinatorics. *Lecture notes. http://www.cs.elte.hu/lovasz/kurzusok/topol16.pdf*, 2016.

[46] Y. Matsumoto. *An introduction to Morse theory*. Number v. 208 in Translations of mathematical monographs. American Mathematical Society, Providence, R.I, 2002.

[47] A. McCleary and A. Patel. Multiparameter persistence diagrams. *arXiv preprint arXiv:1905.13220*, 2019.

[48] Y. Mileyko, S. Mukherjee, and J. Harer. Probability measures on the space of persistence diagrams. *Inverse Problems*, 27(12):124007, 2011.

[49] J. R. Munkres. *Elements of algebraic topology*. CRC Press, 2018.

[50] V. Nanda. Perseus: the persistent homology software. *Software available at http://www. sas. upenn. edu/˜ vnanda/perseus*, 2012.

[51] M. Nicolau, A. J. Levine, and G. Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, 108(17):7265–7270, Apr. 2011.

[52] T. Padellini and P. Brutti. Supervised learning with indefinite topological kernels. *arXiv preprint arXiv:1709.07100*, 2017.

[53] P. Pearson, D. Muellner, and G. Singh. *TDAmapper: Analyze High-Dimensional Data Using Discrete Morse Theory*, 2015. R package version 1.0.

[54] J. A. Perea and J. Harer. Sliding Windows and Persistence: An Application of Topological Methods to Signal Analysis. *Foundations of Computational Mathematics*, 15(3):799–838, June 2015.

[55] D. Pickup, X. Sun, P. L. Rosin, R. Martin, Z. Cheng, Z. Lian, M. Aono, A. B. Hamza, A. Bronstein, M. Bronstein, et al. Shrec14 track: Shape retrieval of non-rigid 3d human models. In *Proceedings of the 7th Eurographics workshop on 3D Object Retrieval*, volume 1, page 6. Eurographics Association, 2014.

[56] C. S. Pun, K. Xia, and S. X. Lee. Persistent-Homology-Based Machine Learning and Its Applications – A Survey. *SSRN Electronic Journal*, 2018.

[57] N. Ravishanker and R. Chen. Topological data analysis (tda) for time series. *arXiv preprint arXiv:1909.10604*, 2019.

[58] J. Reininghaus, S. Huber, U. Bauer, and R. Kwitt. A stable multi-scale kernel for topological machine learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4741–4748, 2015.

[59] G. Singh, F. Mémoli, and G. E. Carlsson. Topological methods for the analysis of high dimensional data sets and 3d object recognition. *SPBG*, 91:100, 2007.

[60] D. S. Stoffer. Walsh-fourier analysis and its statistical applications. *Journal of the American Statistical Association*, 86(414):461–479, 1991.

[61] A. Storjohann. Near optimal algorithms for computing Smith normal forms of integer matrices. In *Proceedings of the 1996 international symposium on Symbolic and algebraic computation - ISSAC '96*, pages 267–274, Zurich, Switzerland, 1996. ACM Press.

[62] A. Storjohann. Computing Hermite and Smith normal forms of triangular integer matrices. *Linear Algebra and its Applications*, 282(1-3):25–45, Oct. 1998.

[63] F. Takens. *Detecting Strange Attractors in Turbulence. Lecture Notes in Mathematics*, volume 898, pages 366–381. 11 2006.

[64] The GUDHI Project. *GUDHI User and Reference Manual*. GUDHI Editorial Board, 3.3.0 edition, 2020.

[65] K. Turner, Y. Mileyko, S. Mukherjee, and J. Harer. Frechet Means for Distributions of Persistence Diagrams. *Discrete & Computational Geometry*, 52(1):44–70, July 2014.

[66] F. Uhlig. *Transform linear algebra*. Prentice Hall, 2002.

[67] M. J. Van De Vijver, Y. D. He, L. J. Van't Veer, H. Dai, A. A. Hart, D. W. Voskuil, G. J. Schreiber, J. L. Peterse, C. Roberts, M. J. Marton, et al. A gene-expression signature as a predictor of survival in breast cancer. *New England Journal of Medicine*, 347(25):1999–2009, 2002.

[68] O. Vipond. Multiparameter persistence landscapes. *Journal of Machine Learning Research*, 21(61):1–38, 2020.

[69] K. Wu and G.-W. Wei. Quantitative toxicity prediction using topology based multitask deep neural networks. *Journal of chemical information and modeling*, 58(2):520–531, 2018.

[70] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *2009 IEEE Conference on computer vision and pattern recognition*, pages 1794–1801. IEEE, 2009.

[71] E. A. Yildirim. Two Algorithms for the Minimum Enclosing Ball Problem. *SIAM Journal on Optimization*, 19(3):1368–1391, Jan. 2008.

[72] X. Zhu. Persistent homology: An introduction and a new text representation for natural language processing. In *IJCAI*, pages 1953–1959, 2013.

[73] P. Zimmermann, A. Casamayou, N. Cohen, G. Connan, T. Dumont, L. Fousse, F. Maltey, M. Meulien, M. Mezzarobba, C. Pernet, et al. *Computational mathematics with SageMath*. SIAM, 2018.

[74] A. Zomorodian. Fast construction of the Vietoris-Rips complex. *Computers & Graphics*, 34(3):263–271, June 2010.

[75] A. Zomorodian and G. Carlsson. Computing Persistent Homology. *Discrete & Computational Geometry*, 33(2):249–274, Feb. 2005.

Apéndice A

Funciones de R

plot.complex Dada una filtración con puntos bidimensionales, esta función dibuja el complejo simplicial a la escala deseada.

```
plot.complex=function(filtration,scale,color=rgb(1,0.5,1,0.2),
    main="",sub="",xlab="",ylab="",asp=NULL){

    pts=filtration$coordinates
    plot(x=c(min(pts[,1]),max(pts[,1])),y=c(min(pts[,2]),max(pts[,2])),
        main=main, sub=sub, xlab=xlab, ylab=ylab, asp=asp,type="n")

    val=filtration$values[filtration$values<=scale]
    len=length(val)
    if (len==0) break
    for (i in 1:len){
        if (length(filtration$cmplx[[i]])==1) {
            point=pts[filtration$cmplx[[i]],]
            points(x=point[1],y=point[2])}
        else polygon(pts[filtration$cmplx[[i]],],col=color)
        }
    }
```

meb Dado un conjunto A de puntos en $\mathbb{R}^n$ y un nivel de precisión, esta función te da el centro y el radio aproximados de la menor bola euclídea que envuelve al conjunto A. A debe ir en forma de matriz donde cada fila representa un punto. El algoritmo está basado en [71].

```
meb = function(A,epsilon=0.0001){

    ref=rep(0,dim(A)[2])
    sqdist=function(u) sum((u-ref)^2)

    alpha=which.max(apply(A,1,sqdist))
    ref=A[alpha,]
    beta=which.max(apply(A,1,sqdist))
    center=(A[alpha,]+A[beta,])/2
    gamma= sqdist(A[beta,])

    ref=center
    cappa=which.max(apply(A,1,sqdist))
    delta=(sqdist(A[cappa,])/gamma) -1

    while(delta > (1+epsilon)^2-1){
        lambda=delta/(2+2*delta)
        gamma=(1-lambda)*gamma+lambda*(1-lambda)*(1+delta)*gamma
        center=(1-lambda)*center+lambda*A[cappa,]
        ref=center
        cappa=which.max(apply(A,1,sqdist))
        delta=sqdist(A[cappa,])/gamma -1
        }

    result=list(center=center,radius=sqrt(gamma*(1+delta)))
    result
    }
```

cechFiltration Dado un conjunto de puntos X, un ε máximo y una dimensión máxima para los símplices, esta función te calcula la filtración de Čech en un formato compatible con las funciones de la librería TDA. Está basado en los algoritmos de [74]

```
cechFiltration = function(X, maxdimension, maxscale){

    #En primer lugar definimos una serie de funciones auxiliares
    # 1. Aristas del complejo
    cech.edges=function(X, max.scale){

        edges=c()
        l=dim(X)[1]
        for (i in 1:(l-1)){
            for (j in (i+1):l){
                d=sqrt(sum((X[i,]-X[j,])^2))
                if (d<=2*max.scale){
                    edges=rbind(edges,c(j,i,d/2))
                    }
                }
            }
        if (!is.null(edges)){edges=matrix(edges,ncol=3)
        edges=edges[order(edges[,3]),]}
        edges
    }

    # 2. Vecinos menores
    get_lower_nbrs=function(edgemat,num){edgemat[edgemat[,1]==num,2]}

    # 3. Engrandecer (encontrar símplices de una dimensión mayor)
    cech.enlarge= function(simplices,edges){
        d=dim(simplices)
        result=c()
        for (i in 1:d[1]){
            near=get_lower_nbrs(edges,simplices[i,1])
            for (j in 2:(d[2]-1)){
                near =intersect(near,get_lower_nbrs(edges,simplices[i,j]))
                if (length(near)==0) break
            }
            for (n in near) {
                r=meb(X[c(simplices[i,-d[2]],n),])$radius
                if(r<=maxscale){
                    result=rbind(result,c(simplices[i,-d[2]],n,r))
                }
            }
        }
        if (!is.null(result))result=matrix(result,ncol=d[2]+1)
        result
    }

    # 4. Generar todos los símplices a partir de las aristas
    fill_cliques= function(edges,max.dim){

        if(is.null(edges)) simplices=NULL
```

```
        else{
        simplices=list(edges)
        if (max.dim==0) return(simplices)
        else{
            matrix_to_enlarge=edges
            for (i in 1:max.dim){
                new_simplices=cech.enlarge(matrix_to_enlarge,edges)
                if (length(new_simplices)==0) break
                simplices=c(simplices,list(new_simplices))
                matrix_to_enlarge=new_simplices
            }
        }}
        simplices
    }

    #Ahora podemos crear una lista de matrices, que almacenan los
    #vértices de cada símplice y la escala a la que entran
    matrices.list=fill_cliques(cech.edges(X,max.scale=maxscale),
        max.dim=maxdimension)

    #Por último almacenamos esta información en un formato compatible
    #con las funciones de la librería TDA
    cmplx=c(1:dim(X)[1],list())
    values=rep(0,dim(X)[1])
    if(!is.null(matrices.list)){
    for (l in 1:length(matrices.list)){
        for (i in 1:dim(matrices.list[[l]])[1]) {
            s=matrices.list[[l]][i,]
            values=c(values,s[l+2])
            cmplx=c(cmplx,list(s[-(l+2)]))
        }
    }}
    cmplx=cmplx[order(values)]
    values=sort(values)
    result=list(cmplx=cmplx,values=values,increasing=T,coordinates=X)
}
```

choose.witness Dado un conjunto de puntos y una proporción entre 0 y 1, esta función te separa los puntos en marcadores y testigos con el método *maxmin* y te da la matriz de distancias entre un subconjunto y otro. Basado en [29].

```
choose.witness = function(set,prop){

    #Definimos una función auxiliar que nos dé una matriz de distancias

    distmat= function(set1,set2){
        a=dim(set1)
        b=dim(set2)
        if(a[2]!=b[2]) {warning("Points␣dimension␣differ.")}
        m=matrix(0,nrow=a[1],ncol=b[1])
        for (i in 1:a[1]){
            for (j in 1:b[1]){
                m[i,j]=sqrt(sum((set1[i,]-set2[j,])^2))
            }
        }
        m
    }

    #Ya podemos usar el método maxmin

    d=dim(set)
    n=floor(d[1]*prop)
    r=sample(d[1],1)
    L=matrix(set[r,],nrow=1)
    W=set[-r,]
    dm=distmat(L,W)
    nW=dim(W)[1]

    for (i in 2:n){
        mini=c()
        for (j in 1:nW){
            mini[j]=min(dm[,j])
        }
        t=which.max(mini)
        newrow=distFct(matrix(W[t,],nrow=1),W[-t,])
        L=rbind(L,W[t,])
        W=W[-t,]
        nW=nW-1
        dm=dm[,-t]
        dm=rbind(dm,newrow)
    }
    rownames(dm)=c()
    result = list(L=L, W=W, distmatrix=dm)
}
```

witnessFiltration Dada una matriz como las obtenidas con **choose.witness**, una dimensión máxima para los símplices y una escala máxima, esta función nos calcula la filtración *witness* con $\nu = 0$ o $\nu = 1$. Basado en las ideas de [29] y los algoritmos de [74].

```
witnessFiltration= function(wlist, maxdimension, maxscale,v){
    if(v!=0 & v!=1) warning("v should be either 0 or 1")

    #En primer lugar definimos una serie de funciones auxiliares
    # 1. Aristas del complejo
    witness.edges= function(wlist, max.scale, v){
        distmat=wlist$distmatrix
        d=dim(distmat)
        if(v==1){
            for (j in 1:d[2]){
                m=min(distmat[,j])
                distmat[,j]=distmat[,j]-m}}
        edges=c()
        for (j in 1:d[2]){
            verticesj=which(distmat[,j]<=max.scale)
            if (length(verticesj)<2) next
            for (k in 1:(length(verticesj)-1)){
                for (l in (k+1):(length(verticesj))){
                    edges=rbind(edges,c(verticesj[l],verticesj[k]))}}}
        edges=unique(edges)
        if(!is.null(edges)){
        values=c()
        for (n in 1:dim(edges)[1]){
            values=c(values,min(apply(distmat[edges[n,],],2,max)))}
        edges=cbind(edges,values)
        edges=edges[order(values),]
        colnames(edges)=c()
        edges = matrix(edges, ncol=3)}
        edges
        }

    # 2. Vecinos menores
    get_lower_nbrs=function(edgemat,num){edgemat[edgemat[,1]==num,2]}

    # 3. Engrandecer (encontrar símplices de una dimensión mayor)
    enlarge=function(simplices,edges){
        d=dim(simplices)
        result=c()
        values=c()

        for (i in 1:d[1]){
            near=get_lower_nbrs(edges,simplices[i,1])
            for (j in 2:(d[2]-1)){
                near =intersect(near,get_lower_nbrs(edges,simplices[i,j]))
                if (length(near)==0) break
            }
             for (n in near) {r=simplices[i,d[2]]
           m2=c()
```

```
        for (j in 1:(d[2]-1)) {m2=c(m2,edges[edges[,1]==simplices[i,j]&edges
        r=max(r,max(m2))
        result=rbind(result,c(simplices[i,-d[2]],n,r))}}
        if (!is.null(result)) result=matrix(result,ncol=d[2]+1)
        result
}

 # 4. Generar todos los símplices a partir de las aristas
 fill_cliques= function(edges,max.dim){
     if(is.null(edges)) simplices=NULL
     else{simplices=list(edges)
     if (max.dim==0) return(simplices)
     else{
         matrix_to_enlarge=edges
         for (i in 1:max.dim){
             new_simplices=enlarge(matrix_to_enlarge,edges)
             if (length(new_simplices)==0) break
             simplices=c(simplices,list(new_simplices))
             matrix_to_enlarge=new_simplices
         }
     }}
     simplices
 }

 #Ahora podemos crear una lista de matrices, que almacenan los
 #vértices de cada símplice y la escala a la que entran

 matrices.list=fill_cliques(witness.edges(wlist,max.scale=maxscale,v),
     max.dim=maxdimension)

 #Por último almacenamos esta información en un formato compatible
 #con las funciones de la librería TDA
 cmplx=c(1:dim(wlist$L)[1],list())
 values=rep(0,dim(wlist$L)[1])
 if(!is.null(matrices.list)){
 for (l in 1:length(matrices.list)){
     for (i in 1:dim(matrices.list[[l]])[1]) {
         s=matrices.list[[l]][i,]
         values=c(values,s[l+2])
         cmplx=c(cmplx,list(s[-(l+2)]))
     }
 }}
 cmplx=cmplx[order(values)]
 values=sort(values)
 result=list(cmplx=cmplx,values=values,increasing=T,coordinates=wlist$L)}
```