Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Revisión de Software para Análisis Topológico de Datos

Autor: Gonzalo Turiel García
Tutor(a): Juan A. Fdez del Pozo

Madrid, [Junio 2024]

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática

Título: Revisión de Software para Análisis Topológico de Datos

[Junio 2024]

Autor: Gonzalo Turiel García
Tutor: Juan A. Fdez del Pozo
Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Resumen

Este trabajo consiste en una revisión de tres softwares dedicados al análisis topológico de datos: Topology ToolKit, Kepler Mapper y Perseus. La finalidad del trabajo consiste en dar un contexto del análisis topológico de datos mencionando tres de los métodos aplicados en este campo: clustering basado en densidad, homología persistente y algorimo Mapper con la finalidad de facilitar el entendimiento de los resultados obtenidos en el posterior uso del software.

El objetivo principal es mostrar los pasos de instalación, uso y comprensión de los resultados utilizando ejemplos existentes en la documentación oficial de los softwares mencionados, por lo que no se realizará un análisis de datos desde cero, pero permitirá contextualizar y facilitar el posterior uso de los mismos para cualquier usuario. A pesar de esto, se muestran gráficas de cada paso seguido para acompañar la parte técnica con la parte visual, explicando la interpretación de las mismas y facilitando así la lectura.

Debido a que es un campo cuyas bases matemáticas son complejas, la intención ha sido permitir una lectura lo más sencilla posible en la descripción de los tres métodos que se explican, sin dejar de lado la importancia que tiene comprender los fundamentos. Se explican detalladamente diferentes estudios que se han llevado a cabo para demostrar con usos reales de los métodos y de esta manera ser capaces de ver la magnitud de este campo y las aplicaciones que tiene en el mundo real, permitiendo a cualquier lector comprender la importancia de este campo que aún está en una etapa muy joven de su desarrollo, sin embargo, ya nos permite ver el potencial real que tiene.

Abstract

This work consists of a review of three software tools dedicated to topological data analysis: Topology ToolKit, Kepler Mapper, and Perseus. The purpose of this work is to provide a context for topological data analysis by mentioning three methods applied in this field: density-based clustering, persistent homology, and the mapper algorithm, with the aim of facilitating the understanding of the results obtained from the subsequent use of the software.

The main objective is to demonstrate the steps for installation, usage, and comprehension of the results using examples from the official documentation of the mentioned software. While no data analysis will be performed from scratch, this review will contextualize and facilitate the subsequent use of these tools for any user. Despite this, graphs of each step followed are presented to accompany the technical part with the visual part, explaining their interpretation and thus facilitating reading.

Given that this field has complex mathematical foundations, the intention has been to allow for the simplest possible reading in the description of the three explained methods, without neglecting the importance of understanding the fundamentals. Different studies that have been carried out using these methods are explained in detail to demonstrate their real-world applications. In this way, it is possible to appreciate the magnitude of this field and its applications in the real world, allowing any reader to understand the importance of a field that is still in a very early stage of development but already shows its real potential.

Tabla de contenidos

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Este Trabajo de Fin de Grado propone hacer una revisión de software para el análisis topológico de datos. El análisis topológico de datos es un método estadístico que encuentra estructuras en los datos. Por la reciente explosión en dimensiones, variedad y tamaño de los datos identificar, extraer y explotar la estructura subyacente es un problema importante para el análisis de datos y el aprendizaje estadístico. Para ello empleamos, como su propio nombre indica, ideas topológicas. Ejemplos de métodos que se usan en el TDA podrían ser la homología persistente, estimación en variedades, clustering, reducción de la dimensión no lineal, algoritmo Mapper, estimación de modas y estimación de ridge.

Para poner en contexto la magnitud de los datos que se generan a nivel mundial, según las últimas estimaciones se generan 328.77 millones de terabytes cada día[1], se estima que aproximadamente un 90 % de los datos totales generados a lo largo de la historia se han generado en los últimos dos años, en los últimos 13 años se ha incrementado entorno a 60 veces más la cantidad de datos generados pasando de 2 zettabytes (2^{41} gigabytes) en el año 2010, mientras que en 2023 se generaron 120 zettabytes (2^{47} gigabytes). Se espera que para 2025 la cifra aumente en un 150 % y alcance los 181 zettabytes (ver Figura1.1). [1], [2]

Esta situación, la cual solo va a seguir en aumento, necesita que se desarrollen métodos eficientes e innovadores de procesamiento de datos. Es importante considerar que el incremento en el volumen de datos va acompañado de una mayor complejidad. En determinadas circunstancias, esta complejidad puede generar ”ruido”. En el ámbito estadístico, se describe como ”ruidosos” aquellos datos que exhiben una variabilidad que no puede ser atribuida al modelo subyacente, lo que implica que los datos están desorganizados o resultan difíciles de interpretar. El ruido puede ser causado por errores en la recogida, medición o procesamiento de los datos, y puede ser aleatorio o sistemático. El ruido aleatorio es impredecible y suele estar fuera del control del investigador, mientras que el ruido sistemático sigue un patrón y puede ser predecible. El uso de técnicas de clustering y otras ideas procedentes de ámbitos como la informática, machine learning y la cuantificación de la incertidumbre, junto con modelos matemáticos y estadísticos, suelen ser muy útiles para el análisis de datos. Sin embargo, los

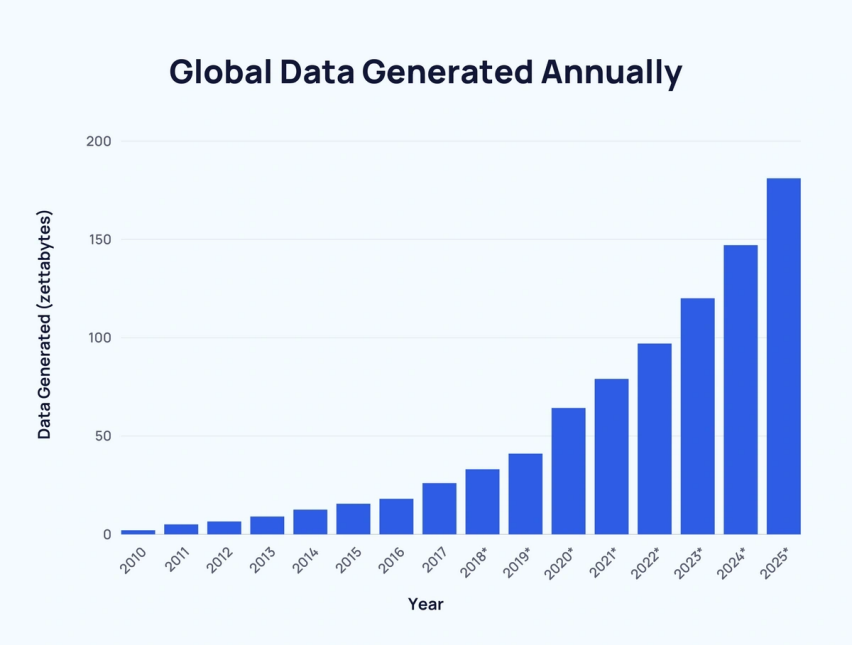

Figura 1.1: Gráfica de datos generados a nivel mundial.

recientes desarrollos matemáticos están arrojando nueva luz sobre dichas ideas "tradicionales", forjando nuevos enfoques propios y ayudando a las personas a descifrar mejor la estructura cada vez más complicada en los datos.

El campo del análisis de datos topológicos (TDA) ha intentado llenar este vacío produciendo una colección de técnicas que se derivan de la idea de que los datos tienen una forma que puede ser cuantificada rigurosamente para investigar los datos. Esta cuantificación toma la forma de una firma topológica: una representación de algún aspecto de la forma en una versión simplificada para su estudio. Estas técnicas han proporcionado una gran cantidad de nuevos conocimientos en el estudio de datos en un conjunto cada vez más diverso de aplicaciones, incluyendo la cobertura de redes de sensores, proteínas, la estructura tridimensional del ADN, el desarrollo de células, la estabilidad de las moléculas de fulereno, robótica, señales en imágenes, periodicidad en series temporales, cáncer, filogenética, imágenes naturales, la propagación de contagios, autosimilitud en geometría, ciencia de materiales, redes financieras, diversas aplicaciones en neurociencia, clasificación de redes ponderadas, redes de colaboración, análisis de datos de teléfonos móviles, comportamiento colectivo en biología, salida de series temporales de sistemas dinámicos, análisis de lenguaje natural, y más. [3].

Como con cualquier resumen, producir una firma topológica a partir de los datos es un proceso con pérdida; es decir, se perderá alguna información durante la creación del resumen. Sin embargo, el arte de utilizar TDA es colocar los datos en una forma que no solo se ajuste a los flujos de trabajo estándar de TDA, sino que la naturaleza con pérdida del método sirva para eliminar la alta dimensionalidad en lugar de la estructura importante. El usuario también necesita

comprender qué características de los datos pueden ser encontradas con (y cuáles son ignoradas por) los métodos disponibles, con el fin de elegir la herramienta adecuada para el problema de interés.

Antes de empezar a desarrollar los diferentes métodos quiero hablar de la historia del TDA. El análisis de datos topológicos se fundamenta en parte en la homología persistente. Contribuciones tempranas importantes incluyen a Frosini en 1992[4], Robins en 1999[5] y, en particular, Edelsbrunner y colaboradores en 2002[6], quienes introdujeron la idea de la persistencia de características a lo largo de filtraciones. Sin embargo, la primera aparición de la expresión "análisis de datos topológicos" parece no haber sido hasta el trabajo de De Silva & Carlsson en 2004[7] y Bremer y colaboradores en 2004[8]. Más tarde, Carlsson en 2009[9] fue clave en la popularización del análisis de datos topológicos, demostrando cómo las ideas de la topología podrían resolver muchos de los problemas enfrentados al aplicar métodos geométricos a datos complejos. Otros desarrollos son descritos por Perea en 2019[10].

Para el desarrollo del trabajo primero se establecerán una serie de objetivos, estos han sido propuestos antes de comenzar a escribir el trabajo con la intención de mantener una idea clara, y son los siguientes:

- Mostrar paquetes e implementar posibles pipelines de análisis: Se tratará de explorar y presentar diferentes paquetes de software, también trataremos de combinar los paquetes en pipelines para realizar tareas específicas de procesamiento y modelado de datos.
- Destacar los algoritmos más importantes: Voy a profundizar en los algoritmos fundamentales utilizados en el TDA, tratando sus principios, ventajas y desventajas.
- Explorar las características topológicas y sus invariantes: Analizaré las estructuras y funciones que extraen características topológicas de los datos, para lo que tendré que identificar invariantes asociadas a la geometría y forma de los datos, algunos conceptos que mencionaré serán la homología persistente o la forma de los espacios de datos.
- Enfatizar las diversas gráficas y diagramas específicos: Para lo que usaré representaciones gráficas para ilustrar las características topológicas y los resultados del análisis. Estas gráficas pueden incluir diagramas de dispersión o gráficos de barras, entre otros.
- Revisar aplicaciones y problemas específicos: Entre los que se puede encontrar la detección de anomalías o la clasificación de redes neuronales, discutiré cómo aplicar estas técnicas en situaciones del mundo real para resolver problemas complejos.

Para cumplir estos objetivos también se han definido unos requisitos con la idea de facilitar la comprensión y mejorar la experiencia de la lectura, estos van a ser:

- Definir las estructuras de datos para la exploración topológica de datos: Incluiré representaciones gráficas, matrices o grafos.

- Recoger la terminología en un glosario: Lo que nos permitirá comprender y comunicar los resultados de los algoritmos de manera más eficaz.
- Mostrar ejemplos sencillos para ilustrar el TDA: De esta manera mostraré cómo se extraen características topológicas y cómo se interpretarán los resultados.
- Aportar una bibliografía reciente y amplia sobre TDA.

El uso de gráficas va a ser crucial en el trabajo permitiendo ilustrar de forma clara los conceptos desarrollados, además cabe destacar que crearé desde cero un glosario en el que añadiré los conceptos clave que mencione para que mediante el uso del mismo pueda haber un nexo entre conceptos y de esta manera la explicación sea mucho más amena y clara.

Como he mencionado previamente, el análisis topológico de datos es un campo en desarrollo, el cual ya tiene aplicaciones en el mundo real, lo que nos permite ver de forma objetiva el interés que tiene y lo que aún queda por desarrollar. A lo largo del trabajo se definirán los métodos principales del TDA para dar un contexto general y luego se revisará el software en detalle, ya que es el tema principal del trabajo. El presente informe consta de 6 capítulos, bibliografía y anexos.

1. Capítulo 1: Una breve introducción.
2. Capítulo 2: Conceptos de TDA
3. Capítulo 3: Desarrollo del clustering basado en densidad, homología persistente y algoritmo Mapper.
4. Capítulo 4: Revisión de Software
5. Capítulo 5: Conclusiones, trabajo futuro, resumen de los resultados obtenidos e impacto.
6. Capítulo 6: Glosario
7. Bibliografía: Referencias usadas para la elaboración del trabajo.
8. Anexos: Código para ejecutar el Software.

Capítulo 2

Conceptos de TDA

El analisis topológico de datos (Topological Data Analysis, TDA) es una poderosa herramienta en el campo de la ciencia de datos, permitiendo explorar y comprender estrucuras complejas en conjuntos de datos multidimensionales. La motivación detras del uso del TDA nace en su capacidad para captruar características intrínsecas en los datos que otras técnicas de análisis podrían pasar por alto. Este capítulo tiene el objetivo de proporcionar una base sólida en los conceptos y técnicas fundamentales del TDA, preparando el terreno para su posterior aplicación práctica.

El análisis topológico esta basado en los principios de la topología, una rama de la matemática cuya finalidad es el estudio las propiedades de los espacios que permanecen invariantes bajo transformaciones continuas. Aplicado a la ciencia de datos, el TDA nos permite identificar patrones y formas en datos de alta dimensión, lo que nos ofrece una perspectiva nueva y complementaria a las técnicas estadísticas y de aprendizaje automático tradicionales.

2.1. Topología

La topología es una de las ramas más jóvenes de las matemáticas, aparece en el siglo XVII bajo el nombre análisis de la posición. Aparece por primera vez en el libro "Characteristica Geometrica" de G. Leibniz en 1679, tras esto L. Euler publica un artículo con la solución al famoso problema de los puentes de Königsberg, titulado "Solutio problematis ad geometriam situs pertinentis".

La topología se encarga de las propiedades de las figuras que permanecen invariantes cuando dichas figuras son plegadas, dilatadas, contraídas o deformadas, de forma que no aparezcan nuevos puntos o se hagan coincidir puntos diferentes. La transformación permitida presupone que hay una correspondencia biunívoca entre los puntos de la figura original y los de la transformada, y que la deformación hace corresponder puntos próximos a puntos próximos. Esta última propiedad se llama continuidad, la cual sirve para definir espacios topológicos, en los cuales se pueden considerar los límites de las secuencias. A veces se pueden definir distancias en estos espacios, en cuyo caso se llaman

espacios métricos; a veces no tiene sentido ningún concepto de distancia.[11], [12]. Tiene varios subcampos:

1. Topología General o Topología de Conjuntos Puntuales: Considera las propiedades locales de los espacios y está estrechamente relacionada con el análisis.
2. Topología Combinatoria: Considera las propiedades globales de los espacios, construidos a partir de una red de vértices, aristas y caras.
3. Topología Algebraica: Considera las propiedades globales de los espacios y utiliza objetos algebraicos como grupos y anillos para responder preguntas topológicas.
4. Topología Diferencial: Considera espacios con algún tipo de suavidad asociada a cada punto.

2.2. Espacios Métricos

Un espacio métrico es un conjunto de puntos donde cada par de puntos tiene una distancia definida que satisface ciertas propiedades: no-negatividad, identidad, simetría y desigualdad triangular. Formalmente, un espacio métrico es un par (X, d), donde X es un conjunto y $d : X \times X \to \mathbb{R}$ es una función que define la distancia entre dos puntos en X. Este concepto es fundamental en el TDA, ya que proporciona la base para medir similitudes y diferencias entre datos.[9]

- **No-negatividad**: $d(x, y) \geq 0$ para todos $x, y \in X$.
- **Identidad**: $d(x, y) = 0$ si y solo si $x = y$.
- **Simetría**: $d(x, y) = d(y, x)$ para todos $x, y \in X$.
- **Desigualdad triangular**: $d(x, z) \leq d(x, y) + d(y, z)$ para todos $x, y, z \in X$.

2.3. Homología

Supongamos que se tienen datos que se encuentran en un espacio métrico, como un subconjunto del espacio euclidiano con una función de distancia heredada. En muchas situaciones, no se está interesado en la geometría precisa de estos espacios, sino que se busca entender algunas características básicas, como el número de componentes o la existencia de agujeros y vacíos. La topología algebraica captura estas características básicas ya sea contándolas o asociándoles espacios vectoriales u otras estructuras algebraicas más sofisticadas.

Aquí nos interesa la homología, que es un concepto clásico en la topología algebraica, proporcionando una herramienta poderosa para formalizar y manejar la noción de las características topológicas de un espacio topológico o de un complejo simplicial de manera algebraica. Para cualquier dimensión k, los "agujeros" de dimensión k están representados por un espacio vectorial H_k, cuya dimensión es intuitivamente el número de tales características independientes. Por ejemplo, el grupo de homología de dimensión cero H_0 representa los componentes

conectados del complejo, el grupo de homología de dimensión uno H_1 representa los bucles unidimensionales, el grupo de homología de dimensión dos H_2 representa las cavidades bidimensionales, y así sucesivamente.

Una propiedad importante de estas estructuras algebraicas es que son robustas, ya que no cambian cuando el espacio subyacente se transforma mediante flexión, estiramiento u otras deformaciones. En términos técnicos, son invariantes de homotopía.

Puede ser muy difícil calcular la homología de espacios topológicos arbitrarios. Por lo tanto, aproximamos nuestros espacios mediante estructuras combinatorias llamadas 'complejos simpliciales', para los cuales la homología se puede calcular fácilmente de manera algorítmica. De hecho, a menudo no se da el espacio X, sino que se posee solo un conjunto de muestras discretas S a partir del cual construirlo. [3], [13]

2.4. Pipeline del Análisis Topológico de Datos

El pipeline del análisis topológico de datos es un proceso estructurado que guía desde la entrada de datos obtener información significativa. Este pipeline está compuesto de varias etapas que permiten extraer y analizar características topológicas y geométricas de un conjunto de datos. Los pasos principales de este proceso son los siguientes[13]:

1. Entrada de Datos: El proceso se supone que es un conjunto finito de puntos que tienen una noción de distancia o similitud entre ellos. Esta distancia puede ser la métrica del espacio ambiente (por ejemplo, la métrica euclidiana en $\mathbb{R}^d$) o una métrica intrínseca definida por una matriz de distancias por pares. La elección de la métrica es crucial para revelar características topológicas y geométricas interesantes de los datos.

2. Construcción de la Estructura: Se construye una "forma continua" sobre los datos para resaltar la topología subyacente o la geometría. Esto generalmente es un complejo simplicial o una familia anidada de complejos simpliciales, llamados filtración, que reflejan la estructura de los datos a diferentes escalas. El desafío es definir estructuras que reflejen información relevante sobre la estructura de los datos y que puedan ser construidas y manipuladas efectivamente en la práctica.

3. Extracción de Información Topológica o Geométrica: Se extrae de las estructuras construidas sobre los datos. Esto puede resultar en una reconstrucción completa (una triangulación) de la forma subyacente a los datos o en resúmenes aproximados que requieren métodos específicos para extraer información relevante, como la homología persistente. Es importante demostrar la relevancia y estabilidad de esta información con respecto a perturbaciones o ruido en los datos de entrada, y entender su comportamiento estadístico.

4. Uso de la Información Extraída: La información extraída proporciona nuevas familias de características y descriptores de los datos. Estos pueden ser

utilizados para comprender mejor los datos, especialmente a través de la visualización, o pueden ser combinados con otras características para análisis adicionales y tareas de aprendizaje automático. Es crucial demostrar el valor añadido y la complementariedad de la información proporcionada por las herramientas de TDA en esta etapa.

2.5. Coberturas y Complejos Simpliciales

Una cobertura de un espacio X es una colección de subconjuntos cuya unión es X. En el TDA, las coberturas se utilizan para simplificar la estructura del espacio, facilitando la construcción de complejos simpliciales. Un complejo simplicial es una estructura combinatoria que representa un espacio topológico de manera simplificada, utilizando vértices, aristas, triángulos y sus análogos de mayor dimensión. Estos complejos se utilizan para aproximar y analizar la forma y la conectividad del conjunto de datos.[14]

- **Vértice**: un punto del conjunto.
- **Arista**: un par de vértices conectados.
- **Simplicio**: una generalización de triángulos a cualquier dimensión.

2.6. Algoritmo Mapper

El algoritmo Mapper es una técnica en TDA que facilita la visualización y el análisis exploratorio de datos complejos. Mapper divide el conjunto de datos en regiones más manejables utilizando una función de filtrado y una cobertura, y luego construye un complejo simplicial (nervio) que captura la estructura topológica de los datos. Este método ayuda a descubrir patrones y estructuras en los datos que no son evidentes a simple vista.[15]

1. **Elección de una función de filtrado**: una función que proyecta los datos en un espacio de menor dimensión.
2. **Cobertura del espacio filtrado**: dividir el espacio en subconjuntos.
3. **Construcción del nervio**: crear un complejo simplicial a partir de las intersecciones de los subconjuntos de la cobertura.

2.7. Homología Persistente

Supongamos que se nos dan datos experimentales en forma de un espacio métrico finito S; hay puntos o vectores que representan mediciones junto con alguna función de distancia (por ejemplo, dada por una correlación o una medida de disimilitud) en el conjunto de puntos o vectores. Ya sea que el conjunto S sea una muestra de algún espacio topológico subyacente o no, es útil pensar en estos términos. Nuestro objetivo es recuperar las propiedades de dicho espacio subyacente de una manera que sea robusta a pequeñas perturbaciones en los datos S. En un sentido amplio, este es el tema de la inferencia topológica. Si S es

un subconjunto del espacio euclidiano, se puede considerar un 'engrosamiento' S dado por la unión de bolas de un cierto radio fijo ϵ alrededor de sus puntos y luego calcular el complejo de Čech. De esta manera, se puede intentar calcular características cualitativas del conjunto de datos S construyendo el complejo de Čech para un valor elegido de ϵ y luego calculando su homología simplicial. El problema con este enfoque es que a priori no hay una elección clara para el valor del parámetro ϵ.

La idea clave de la homología persistente es la siguiente: para extraer información cualitativa de los datos, se consideran varios (o incluso todos) los posibles valores del parámetro ϵ. A medida que el valor de ϵ aumenta, se añaden símplices a los complejos. La homología persistente entonces captura cómo cambia la homología de los complejos a medida que aumenta el valor del parámetro, y detecta qué características 'persisten' a través de los cambios en el valor del parámetro.

La homología persistente es una herramienta poderosa utilizada para calcular, estudiar y codificar de manera eficiente las características topológicas a múltiples escalas de familias anidadas de complejos simpliciales y espacios topológicos. No solo proporciona algoritmos eficientes para calcular los números de Betti de cada complejo en las familias consideradas, sino que también codifica la evolución de los grupos de homología de los complejos anidados a través de las escalas. Las ideas y resultados preliminares que subyacen a la teoría de la homología persistente se remontan al siglo XX.[3], [13]

2.8. Toplex

Un toplex es un complejo celular que puede ser construido a partir de una lista de celdas de dimensión superior. Así, un toplex cúbico de dimensión d es simplemente una lista de cubos de d dimensiones. Las caras y subcaras de estos cubos se construyen automáticamente a partir de la información del cubo superior proporcionada por los archivos de entrada. Una manera sencilla de representar un cubo de dimensión completa es proporcionar la coordenada de su vértice mínimo lexicográficamente, al cual llamaremos su ancla. Así, un cubo 3D anclado en el punto (a, b, c) con coordenadas enteras debe ser el cubo con vértices de la forma $(a+i, b+j, c+k)$ donde i, j y k solo pueden tomar los valores 0 y 1.[16]

2.9. Clustering

El clustering es el problema de encontrar un conjunto de grupos de objetos similares dentro de un conjunto de datos, al mismo tiempo que se mantienen los objetos disímiles separados en diferentes grupos o el grupo de ruido. En general, el clustering se considera una tarea de aprendizaje no supervisado. Dado un conjunto de datos como un conjunto de objetos de un espacio de datos dado $D \subset S$, y una función de disimilitud dis : $S \times S \to \mathbb{R}_0^+$, la tarea es aprender una agrupación significativa de los datos. A menudo, el conjunto de datos es

un conjunto de puntos reales de d-dimensiones, es decir, $D \subset S = \mathbb{R}^d$, que puede verse como una muestra de alguna densidad de probabilidad desconocida $p(x)$, con dis como la distancia euclidiana o alguna otra forma de distancia. El significado de la similitud dentro del grupo y la disimilitud entre grupos es diferente para diferentes enfoques de clustering. [17]

2.10. Conjunto de nivel

En matemáticas, un conjunto de nivel (level set en inglés) de una función real f de n variables reales es un conjunto donde la función toma un valor constante dado c, es decir:

$$L_c(f) = \{(x_1, \ldots, x_n) \mid f(x_1, \ldots, x_n) = c\} \ .$$

Cuando el número de variables independientes es dos, un conjunto de nivel se llama curva de nivel, también conocida como línea de contorno o isolínea; por lo tanto, una curva de nivel es el conjunto de todas las soluciones reales de una ecuación en dos variables x_1 y x_2. Cuando $n = 3$, un conjunto de nivel se llama superficie de nivel (o isosuperficie); por lo tanto, una superficie de nivel es el conjunto de todas las raíces reales de una ecuación en tres variables x_1, x_2 y x_3. Para valores más altos de n, el conjunto de nivel es una hipersuperficie de nivel, el conjunto de todas las raíces reales de una ecuación en $n > 3$ variables.

Relacionado con los conjuntos de nivel está el concepto de masa excesiva. Dada una clase de conjuntos $\mathcal{C}$, la funcional de masa excesiva se define como

$$E(t) = \sup\{P(C) - t\mu(C) : C \in \mathcal{C}\},$$

y cualquier conjunto $C \in \mathcal{C}$ tal que $P(C) - t\mu(C) = E(t)$ se llama un t-cluster generalizado. Si $\mathcal{C}$ incluye todos los conjuntos medibles y la densidad es acotada y continua, entonces el conjunto de nivel superior $\hat{L}_t$ es el único t-cluster.

2.11. Filtraciones

Una filtración de un complejo simplicial K es una familia anidada de subconjuntos $(K_r)_{r\in T}$, donde $T \subseteq \mathbb{R}$, tal que para cualquier $r, r' \in T$, si $r \leq r'$ entonces $K_r \subseteq K_{r'}$, y $K = \bigcup_{r\in T} K_r$. El subconjunto T puede ser finito o infinito. De manera más general, una filtración de un espacio topológico M es una familia anidada de subespacios $(M_r)_{r\in T}$, donde $T \subseteq \mathbb{R}$, tal que para cualquier $r, r' \in T$, si $r \leq r'$ entonces $M_r \subseteq M_{r'}$, y $M = \bigcup_{r\in T} M_r$. Por ejemplo, si $f : M \to \mathbb{R}$ es una función, entonces la familia $M_r = f^{-1}((-\infty, r])$, $r \in \mathbb{R}$, define una filtración llamada la filtración de conjuntos de subnivel de f.

En situaciones prácticas, el parámetro $r \in T$ a menudo puede interpretarse como un parámetro de escala y las filtraciones utilizadas clásicamente en TDA suelen pertenecer a filtraciones construidas sobre datos o filtraciones de conjuntos de subniveles.[13]

2.12. Complejo de Vietoris-Rips

El complejo Vietoris-Rips aproxima el complejo de Čech. Para un número real no negativo ε, el complejo Vietoris-Rips VR(S) a escala ε se define como:

$$VR(S) = \{\sigma \subseteq S \,|\, d(x,y) \leq 2\varepsilon \text{ para todo } x, y \in \sigma\}$$

El sentido en el que el complejo VR aproxima el complejo de Čech es que, cuando S es un subconjunto del espacio euclidiano, tenemos $\check{C}(S) \subseteq VR(S) \subseteq \check{C}(\sqrt{2}\varepsilon)(S)$. Decidir si un subconjunto $\sigma \subseteq S$ está en $VR(S)$ es equivalente a decidir si la distancia máxima entre cualquier par de vértices en σ es a lo sumo 2ε. Por lo tanto, se puede construir el complejo VR en dos pasos. Primero se calcula el grafo del vecindario de ε de S. Este es el grafo cuyos vértices son todos los puntos en S y cuyas aristas son:

$$(i,j) \in S \times S \,|\, i \neq j \text{ y } d(i,j) \leq 2\varepsilon$$

Segundo, se obtiene el complejo VR calculando el complejo de cliques del grafo del vecindario de ε. El complejo de cliques de un grafo es un complejo simplicial que se define de la siguiente manera: El subconjunto $\{x_0, ..., x_k\}$ es un k-simplejo si y solo si cada par de vértices en $\{x_0, ..., x_k\}$ está conectado por una arista. Tal colección de vértices se llama una clique.

Esta construcción hace que sea muy fácil calcular el complejo VR, porque para construir el complejo de cliques uno solo tiene que verificar las distancias entre pares; por esta razón, los complejos de cliques también se llaman "perezosos.[en] la literatura. Desafortunadamente, el complejo VR tiene la misma complejidad en el peor de los casos que el complejo de Čech. En el peor de los casos, puede tener hasta $2|S| - 1$ símplices y dimensión $|S| - 1$.

En aplicaciones, por lo tanto, generalmente solo se calcula el complejo VR hasta alguna dimensión $k \leq |S| - 1$. En nuestros estudios de referencia, a menudo elegimos $k = 2$ y $k = 3$. El artículo [95] revisa diferentes algoritmos para realizar ambos pasos para la construcción del complejo VR, e introduce algoritmos rápidos para construir el complejo de cliques.[3]

2.13. Números de Betti

Las números de Betti son objetos topológicos que fueron demostrados como invariantes por Poincaré, y utilizados por él para extender la fórmula poliédrica a espacios de dimensiones superiores. Informalmente, el número de Betti es el número máximo de cortes que se pueden hacer sin dividir una superficie en dos piezas separadas. Formalmente, el enésimo número de Betti es el rango del enésimo grupo de homología de un espacio topológico.

El primer número de Betti de un grafo es comúnmente conocido como su rango de circuito (o número cromático).

Sea p_r el rango del grupo de homología H_r de un espacio topológico K. Para una superficie cerrada y orientable de género g, los números de Betti son $p_0 = 1$, $p_1 = 2g$ y $p_2 = 1$. Para una superficie no orientable con k asas cruzadas, los números de Betti son $p_0 = 1$, $p_1 = k - 1$ y $p_2 = 0$.

El número de Betti de un grupo abeliano finitamente generado G es el número n (determinado de manera única) tal que

$$G = \mathbb{Z}^n \oplus G_1 \oplus \ldots \oplus G_s$$

donde $G_1, \ldots, G_s$ son grupos cíclicos finitos.

Los números de Betti de un módulo finitamente generado M sobre un anillo unitario local conmutativo noetheriano R son los números mínimos b_i para los cuales existe una secuencia exacta larga

$$\ldots \to R^{b_n} \xrightarrow{\phi_n} R^{b_{n-1}} \xrightarrow{\phi_{n-1}} \ldots \xrightarrow{\phi_1} R^{b_0} \xrightarrow{\phi_0} M \xrightarrow{\phi_{-1}} 0$$

que se llama una resolución libre mínima de M. Los números de Betti están determinados de manera única al requerir que b_i sea el número mínimo de generadores de $\mathrm{Ker}(\phi_{i-1})$ para todo $i \geq 0$. Estos números de Betti están definidos de la misma manera para módulos R-positivamente graduados finitamente generados si R es un anillo de polinomios sobre un campo.

Capítulo 3

Técnicas de TDA

En este capítulo se explicarán las tres diferentes técnicas principales del TDA y cuáles son sus principales aplicaciones en el mundo real. Se mostrarán ejemplos prácticos para visualizar cómo, mediante estos métodos, se puede obtener información de un conjunto de datos del cual, aparentemente, no podemos deducir nada. Se describirán los siguientes métodos: clustering basado en densidad, homología persistente y algoritmo Mapper, ya que son los más conocidos del TDA.

3.1. Clustering basado en densidad

El término ”clustering” hace referencia a la tarea de identificar grupos o clusters en un conjunto de datos. En el clustering basado en la densidad, un cluster es un conjunto de objetos de datos distribuidos en el espacio de datos sobre una región contigua de alta densidad de objetos. Los clusters basados en densidad están separados entre sí por regiones contiguas de baja densidad de objetos. Los objetos de datos situados en regiones de baja densidad suelen considerarse ruido o valores atípicos. Este es uno de los métodos más antiguos de del TDA. Para ver la relación clara entre clustering y topología hay que centrarse en los métodos de clustering basado en densidad.[17]

Desde un punto de vista procedimental, muchos métodos de clustering buscan una partición del conjunto de datos en un número predefinido k de grupos, donde se minimiza la suma de las disimilitudes (al cuadrado) entre todos los objetos del cluster o la suma de las disimilitudes (al cuadrado) de todos los objetos del cluster con respecto a algún representante del cluster (por ejemplo, el valor medio), y se maximizan los valores respectivos entre diferentes clusters, dada alguna función de disimilitud dis. Estas suposiciones suelen dar como resultado clusters de forma convexa (o idealmente, esférica).

Desde un punto de vista estadístico, tales métodos corresponden a un enfoque paramétrico donde se asume que la densidad desconocida p(x) de los datos es una mezcla de k densidades pi(x), cada una correspondiente a uno de los k grupos en los datos. Se asume que los pi(x) pertenecen a alguna familia paramétrica (por ejemplo, distribuciones gaussianas) con parámetros desconocidos. Los

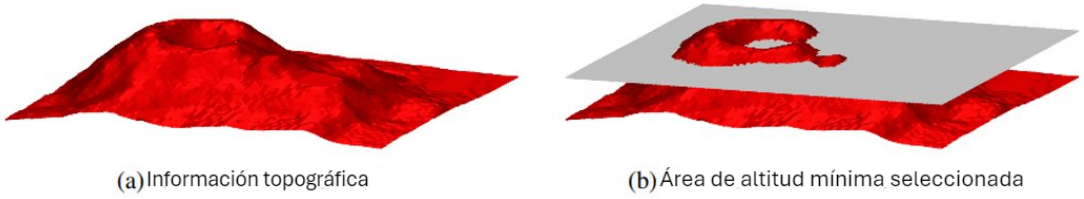

Figura 3.1: Volcán Maunga Whau[17]

parámetros se estiman en función de la muestra dada (el conjunto de datos).

En contraste, el clustering basado en densidad es un enfoque no paramétrico donde se considera que los clusters son áreas de alta densidad de la densidad p(x). Los métodos de clustering basados en densidad no requieren el número de clusters como parámetros de entrada, ni hacen suposiciones sobre la densidad subyacente p(x) o la varianza dentro de los clusters que pueden existir en el conjunto de datos. Como consecuencia, los clusters basados en densidad no son necesariamente grupos de puntos con una baja disimilitud dentro del cluster medida por una función de disimilitud dis, y por lo tanto, no necesariamente tienen una forma convexa, sino que pueden tener formas arbitrarias en el espacio de datos. Intuitivamente, un cluster basado en densidad es un conjunto de objetos de datos distribuidos en el espacio de datos sobre una región contigua de alta densidad de objetos, separada de otros clusters basados en densidad por regiones contiguas de baja densidad de objetos.[17]

3.1.1. Aplicaciones prácticas

1. Geografía y Cartografía: A veces, los clusters basados en densidad también se denominan 'clusters naturales' ya que son particularmente adecuados para ciertas aplicaciones inspiradas en la naturaleza.

   Por ejemplo, en datos espaciales, los clusters de puntos en el espacio pueden formarse a lo largo de estructuras naturales como ríos, carreteras, fallas sísmicas, etc. La Figura 3.1(a) muestra información topográfica del volcán Maunga Whau (Monte Edén) en Auckland. Seleccionar áreas por encima de cierta altitud se relaciona con el clustering basado en densidad, siempre que se seleccione un nivel de densidad acorde (Figura 3.1(b)).[17]

2. Zoología y Biología Evolutiva: Clasificación de especies: Este enfoque se utiliza para estudiar la divergencia de especies a partir de un ancestro común, permitiendo a los zoólogos definir familias de especies. Los miembros de una familia pueden no mostrar una alta similitud par a par, pero sí exhiben cierta similitud con al menos algunos otros miembros de la familia, lo que refleja una evolución natural de características.[17]-[19]

3. Análisis de datos fenotípicos: En estudios de biología, como el análisis del conjunto de datos de Iris de Fisher[20], los clusters naturales pueden ser

identificados en función de descriptores como la longitud y el ancho de los pétalos y sépalos, demostrando que los clusters biológicos típicamente no son esféricos.[17]

4. Las redes de suministro urbano de aguas son un importante activo subterráneo oculto. El clustering de las roturas y reventones de tuberías puede indicar problemas en ciernes. Mediante la herramienta de Clustering basado en densidad, los ingenieros pueden determinar dónde se encuentran esos clústeres y tomar medidas preventivas en las zonas de alto riesgo de sus redes de suministro de aguas.[21]

5. Otras aplicaciones:

   Suponga que dispone de datos de posición de todos los tiros con enceste y sin enceste de los jugadores de la NBA. La herramienta Clustering basado en densidad puede mostrarle los diferentes patrones de posiciones de tiro con enceste y sin enceste de cada jugador. A continuación, esta información se puede usar como base para desarrollar estrategias de juego.[21]

   Suponga que está estudiando una enfermedad concreta propagada por plagas y tiene un dataset de puntos que representa a los hogares de su área de estudio, algunas de ellas infestadas y otras no. Mediante la herramienta Clustering basado en densidad, puede determinar los mayores clústeres de hogares infestados para ayudar a delimitar un área en la que iniciar el tratamiento y la erradicación de las plagas.[21]

   La geolocalización de tuits después de una catástrofe natural o un ataque terrorista puede organizarse en clústeres para informar sobre las necesidades de rescate y evacuación en función del tamaño y la ubicación de los clústeres identificados. [21]

Dentro del clustering basado en densidad podemos encontrar tres métodos principales[22]:

- Clústeres de conjuntos de niveles.
- Árboles de densidad.
- Agrupamiento de modos y teoría de Morse.

3.1.2. Clústeres de conjuntos de niveles

Los level-set clusters (clústerers de conjuntos de nivel) son una técnica de análisis de datos topológico basada en identificar y agrupar regiones de alta densidad en un espacio de datos. Este método se apoya en la idea de que los datos a menudo se distribuyen de manera no uniforme y forman regiones con diferentes niveles de densidad. Los level-set clusters permiten identificar estas regiones y analizarlas como conjuntos separados.

Este método se fundamenta en el concepto de conjuntos de nivel de una función de densidad. Para entenderlo, consideremos una muestra aleatoria $X_1, X_2, \ldots, X_n$ extraída de una distribución P con densidad p. La densidad de una región es una

medida de cuántos puntos de datos se encuentran en esa región. Para entender este método tenemos que saber que los conjuntos de nivel superior (upper level sets) y sus componentes conectados llamados "density clusters" son fundamentales.[22]

1. **Definición de Conjuntos de Nivel**: Para cualquier umbral $t \geq 0$, el conjunto de nivel superior L_t se define como:

   $$L_t = \{x : p(x) > t\}$$

   Este conjunto contiene todos los puntos x cuya densidad es mayor que el umbral t.

2. **Clusters de Densidad**: Los clusters de densidad a un nivel t, denotados como C_t, son los componentes conexos de L_t. En otras palabras, cada cluster en C_t es una región continua de alta densidad donde todos los puntos tienen una densidad superior a t. El conjunto de todos los clusters de densidad se puede expresar como:

   $$C = \bigcup_{t \geq 0} C_t$$

   Esto representa la colección completa de clusters a todos los niveles posibles de densidad. Podemos apreciar en la Figura 3.2a una función de densidad, mientras que en la Figura 3.2b vemos los clústeres de densidad correspondientes a un valor concreto de t.

3. **Estimación del Conjunto de Nivel Superior**: Se puede estimar el conjunto de nivel superior usando la siguiente fórmula:

   $$\hat{L}_t = \{x : \hat{p}(x) > t\} \tag{3.1}$$

   donde $\hat{p}$ es el estimador de densidad.

4. **Estimación de Conjuntos de Nivel**: En la práctica, la densidad p no es conocida y debe ser estimada a partir de los datos. Un estimador común de densidad es el estimador de densidad con núcleo (KDE, por sus siglas en inglés), definido como:

   $$\hat{p}_h(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h^d} K\left(\frac{\|x - X_i\|}{h}\right)$$

   donde $h > 0$ es el parámetro de ancho de banda y K es el núcleo.

5. **Identificación de Clusters**: Necesitamos obtener los componentes conexos de $\hat{L}_t$. Sea $I_t = \{i : \hat{p_h}(X_i) > t\}$. Creamos un grafo cuyos nodos corresponden a $(X_i : i \in I_t)$ y ponemos una arista entre dos nodos X_i y X_j si $\|X_i - X_j\| \leq \epsilon$, donde $\epsilon > 0$ es un parámetro de ajuste. Los componentes conexos del grafo estiman los clusters al nivel t. El número de componentes conexos se denota por β_0, el número de Betti de orden cero.

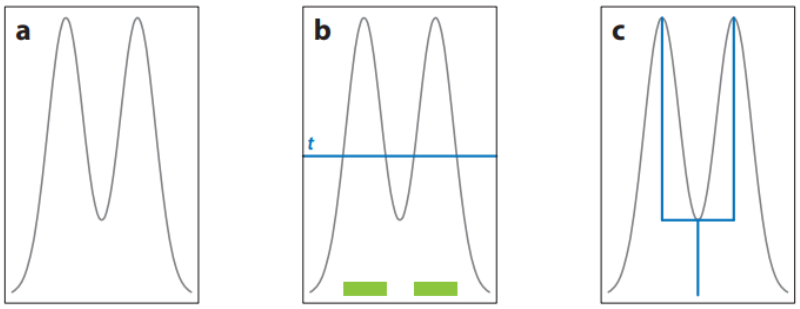

Figura 3.2: (a) Una función de densidad p. (b) Clusters de densidad correspondientes a $L_t = \{x : p(x) > t\}$. (c) El árbol de densidad correspondiente a p se muestra debajo de la densidad. Las hojas del árbol corresponden a los modos. Las ramas corresponden a los componentes conexos de los conjuntos de nivel.

Dominio de aplicación de clústeres de conjunto de niveles

1. Segmentación de imágenes médicas: Se usa para segmentar resonancias magnéticas o tomografías computarizadas, los clústeres de conjuntos de niveles pueden utilizarse para segmentar regiones de tejidos con densidades similares, esto nos va a permitir identificar y segmentar diferentes tipos de tejidos (músculo, grasa, hueso) en una imagen de resonancia magnética para ayudar en el diagnóstico médico[23]. También se puede combinar con algoritmos híbridos de machine learning para identificar y clasificar tumores, ya que esta técnica mejora la detección de ruido y ayuda a refinar la segmentación inicial[24].

2. Análisis de redes sociales: Es posible utilizar los clústeres de conjuntos de nivel para obtener datos sobre conexiones entre usuarios o comunidades. Estos nos ayudan a identificar grupos de usuarios con interacciones frecuentes entre ellos. Por ejemplo, identificar comunidades dentro de una red social basada en las interacciones y conexiones entre usuarios.[25][26][27]

3. Estudio de patrones de tráfico: En el análisis de datos de tráfico, usando datos de ubicación y velocidad de vehículos en una ciudad. Utilizando clústeres de conjuntos de niveles, podremos identificar regiones donde existirá congestión de tráfico, basándote en altas densidades de vehículos. Esto podría ayudar en la gestión del tráfico y la planificación urbana.[28][29][30][31]

3.1.3. Árboles de densidad

El concepto de Árboles de Densidad juega un papel crucial en el análisis topológico de datos (TDA), proporcionando una representación visual y estructural de los clusters de densidad dentro de un conjunto de datos. Estos árboles se fundamentan en la premisa de que si consideramos cualquier par de clusters de densidad A y B, uno está contenido en el otro o son disjuntos. Esta estructura jerárquica permite la representación de una densidad y sus clusters como un

árbol, denotado como $T(\rho)$, que aunque es técnicamente una colección de conjuntos de nivel, puede visualizarse como un árbol bidimensional. La representación del árbol bajo la función de densidad ρ muestra el número de conjuntos de nivel y cómo estos se fusionan.

Por ejemplo, al cortar el árbol en un nivel t, el número de ramas corresponde al número de componentes conectados del conjunto de nivel, y las hojas del árbol corresponden a los modos de la densidad. Los Árboles de Densidad, también conocidos como árboles de cluster, ofrecen una visualización conveniente de una densidad, independientemente de la dimensión d del espacio en el que residen los datos.

Dos árboles de densidad tienen la misma forma si su estructura de árbol es idéntica. La distancia entre dos puntos x e y en el árbol se define por la ecuación:

$$d_{T\rho}(x, y) = |\rho(x) + \rho(y) - 2m_\rho(x, y)|\,,$$

donde $m_\rho(x, y)$ es la altura de fusión, definida como el supremo de t para el cual existe un cluster C en el conjunto de nivel t que contiene tanto a x como a y. Esta definición se extiende para definir una distancia entre conjuntos de clusters en el árbol, induciendo una topología en $T(\rho)$. Dos árboles $T(\rho)$ y $T(q)$ son homeomórficos si existe un mapa bicontinuo entre ellos, lo que significa que tienen la misma forma o estructura de árbol.

El árbol de densidad puede estimarse utilizando cualquier estimador de densidad, como el estimador de densidad kernel, proporcionando una visualización efectiva de la estructura de cluster de los datos. Para estimar la forma del árbol de densidad, no es necesario que el ancho de banda h tienda a 0 a medida que aumenta n. Bajo condiciones débiles, existe un h_0 tal que para todo h menor que h_0, $T(\rho_h)$ es igual a $T(\rho)$, lo que tiene implicaciones prácticas importantes ya que $T(\rho_h)$ puede ser estimado con una tasa de $O(n^{-1/2})$ independientemente de las dimensiones d.

El bootstrap puede utilizarse para obtener conjuntos de confianza para el árbol de densidad. Por ejemplo, si P_n es la medida empírica que pone masa $1/n$ en cada punto de datos, se puede dibujar una muestra independiente y distribuida idénticamente $X_1, \ldots, X_n \sim P_n$ y calcular el estimador de densidad ρ_h. Repitiendo este proceso B veces, se obtienen estimaciones de densidad y se puede definir un conjunto de confianza asintótico para el árbol. El valor crítico t_a puede usarse para podar hojas y ramas no significativas del árbol. Un árbol de densidad es consistente con Hartigan si, con probabilidad tendiendo a 1, se recupera la estructura de cluster correcta. Generalmente, los árboles de densidad basados en estimadores de densidad consistentes serán consistentes con Hartigan.[22]

Dominio de aplicación de árboles de densidad

En este artículo de Cheng et al. en 2016 [32] se hace un estudio basandose en un conjunto de datos que contiene el registro del crimen organizado en el continente africano en el año 2008, al cual le aplican el método SCDOT (Spatial Clustering with Density-Ordered Tree) explicado en el mismo artículo y lo comparan con

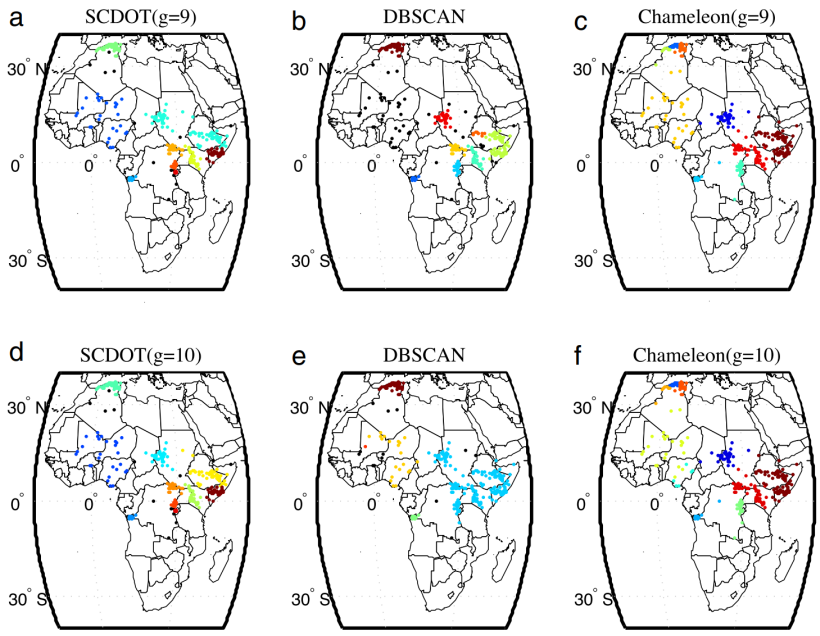

Figura 3.3: Clústeres descubiertos por SCDOT, DBSCAN y Chameleon

DBSCAN[33] y Chameleon[34]. Este estudio permite ver resultados precisos y coherentes de la distribución geográfica de los conflictos. El método aplicado permite identificar clústeres de diferentes densidades sin ajustar parámetros complejos, facilitando así su aplicación en situaciones prácticas como el análisis de conflictos y la planificación urbana. Para demostrar el uso de los Árboles de densidad podemos ver la Imagen3.3, en la cual:

- **Argelia del Norte (Cluster Verde)**: Esta área experimentó conflictos armados entre el gobierno argelino y AQIM.
- **Etiopía (Cluster Amarillo)**: Los conflictos involucraron al gobierno etíope y a grupos como ONLF y OLF.
- **Somalia del Sur (Cluster Marrón)**: Conflictos entre el gobierno somalí y Al-Shabaab.
- **Kenya Occidental (Cluster Verde Claro)**: Conflictos no estatales y violencia unilateral, notablemente entre los grupos Kalenjin y Kikuyu.
- **Sudán Occidental (Cluster Cian)**: Incluye violencia estatal y unilateral, involucrando a grupos como SLM/A y JEM.
- **República de Burundi (Cluster Rojo)**: Conflictos principalmente entre el gobierno de Burundi y Palipehutu-FNL.
- **Norte del Golfo de Guinea (Cluster Azul)**: Conflictos de menor densidad, incluyendo áreas como Mali, Níger y Nigeria, involucrando a varios grupos y fuerzas gubernamentales.

Por otro lado un reciente artículo de Menard et al en 2022[35] aborda la inves-

tigación sobre la comunidad académica italiana en el ámbito de la estadística y disciplinas similares. Se busca distinguir los grupos que emergen a partir de sus interconexiones, las cuales varían en tipo e intensidad, así como el principal mecanismo de agrupación. Comprender esta estructura es clave para fomentar colaboraciones y proyectos innovadores, o para reconocer a los líderes en áreas específicas de la comunidad.

El análisis se basó en la creación de una red multicapa ponderada, estructurada en cuatro niveles:

1. afiliación (AFF)
2. macro-sector (MS)
3. publicaciones conjuntas (PUBS)
4. palabras clave comunes (KW).

Se recogieron datos en noviembre de 2019, abarcando 1160 académicos e investigadores del registro del MIUR. La red global fue normalizada y se agregaron los pesos de las conexiones entre las capas para construir una red superpuesta.

La identificación de comunidades se llevó a cabo mediante el algoritmo de Louvain, diseñado para optimizar un concepto matemático denominado modularidad, una medida que se utiliza en la teoría de grafos para evaluar la calidad de una partición[36]. Se distinguieron 23 comunidades, con un rango de 13 a 102 miembros. La homogeneidad de los clústeres se midió en relación con los cuatro niveles mencionados. Los resultados mostraron que los grupos eran coherentes en cuanto a sector científico y afiliación, pero menos en publicaciones conjuntas y temas de investigación. Aunque la modularidad alcanzó su punto máximo, no se consideró excepcionalmente alta.

El estudio de los clústeres modales se efectuó considerando el grado de los actores, resultando en 499 clústeres en la opción AND y 139 grupos en la opción OR, cada uno con distintos niveles de homogeneidad y parsimonia. La interpretación de la generación de colaboración entre los investigadores se examinó mediante árboles de densidad, facilitando la visualización de la jerarquía de clústers y ofreciendo una perspectiva más detallada de las interacciones y colaboraciones dentro de la comunidad, resaltando la estructura y dinámica de los académicos en diversos niveles de detalle.

3.1.4. Agrupamiento de modos y teoría de Morse

Otro método de agrupación de densidades es la *agrupación de modos*. La idea es encontrar modos de la densidad y luego definir los grupos como las cuencas de atracción de los modos. Un punto m es un modo (local) si existe un vecindario abierto N de x tal que $p(x) > p(y)$ para cada $y \in N$ tal que $y \neq x$. Supongamos que p tiene k modos locales $M = \{m_1, \ldots, m_k\}$. Supongamos que p tiene un gradiente g y una Hessiana H.

Un punto x es un punto crítico si $g(x) = (0, \ldots, 0)^T$. La función p es una función de Morse si la Hessiana es no degenerada en cada punto crítico (ver [38]).

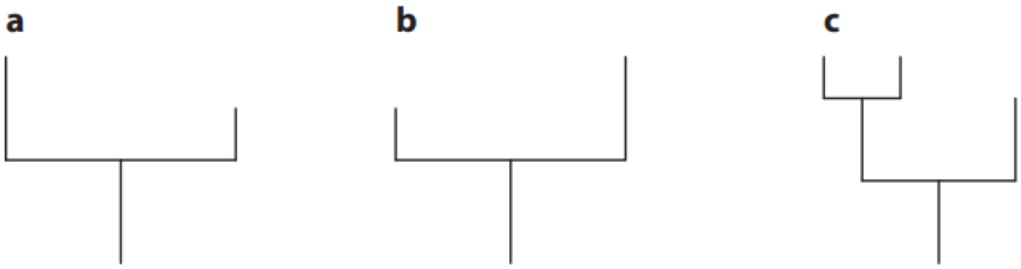

Figura 3.4: (a,b) Los árboles de densidad en los paneles a y b son homeomórficos; existe un mapa bicontinuo de un árbol al otro. (c) El tercer árbol no es homeomórfico a los otros dos. Así, los primeros dos árboles representan densidades con la misma forma.

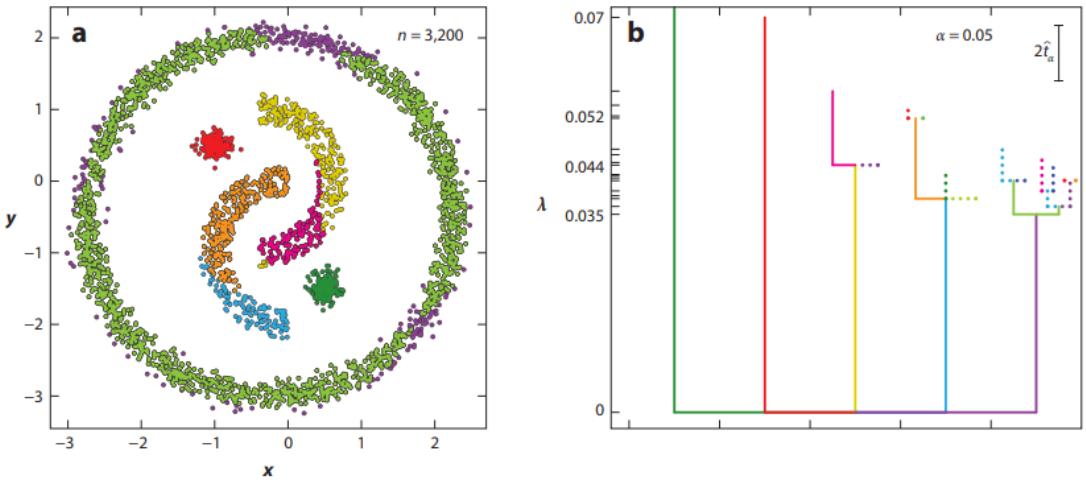

Figura 3.5: Ejemplo de árbol de densidad utilizando el bootstrap de Chen[37], mostrando (a) los datos y (b) el árbol. Las líneas sólidas son los árboles podados; las líneas discontinuas son hojas y ramas que se han podado porque son más pequeñas que el nivel de significancia del bootstrap $2t\alpha$ (indicado en la esquina superior derecha del panel b). Los colores a la izquierda corresponden a los grupos en el árbol a la derecha.

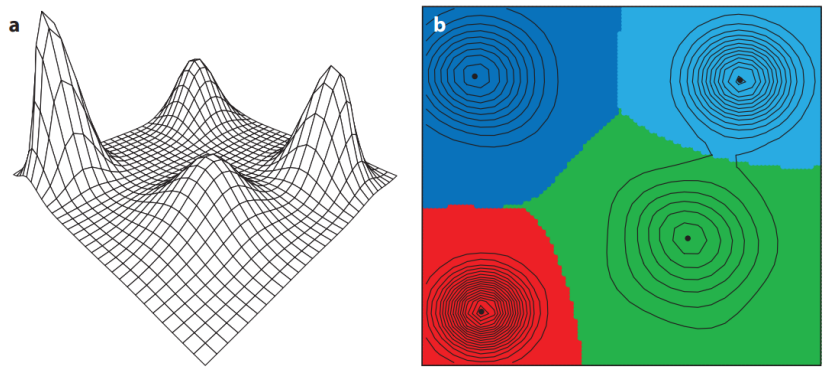

Figura 3.6: (a) Una densidad con cuatro modos. (b) La partición (cuencas de atracción) del espacio inducida por los modos. Estos son los clústeres de población.

Supondremos que p es de Morse. En este caso, m es un modo local si y solo si $g(m) = (0, \ldots, 0)^T$ y $\lambda_1(H(m)) < 0$, donde $\lambda_1(A)$ denota el mayor valor propio de la matriz A.

Ahora sea x un punto arbitrario. Si seguimos el camino de ascenso más empinado comenzando en x, eventualmente llegaremos a uno de los modos. Por lo tanto, cada punto x en el espacio de la muestra se asigna a un modo m_j. Decimos que m_j es el destino de x, lo que se escribe como $\text{dest}(x) = m_j$.

El camino $\pi_x : \mathbb{R} \to \mathbb{R}^d$ que conduce de x a un modo se define por la ecuación diferencial

$$\pi_x'(t) = \nabla p(\pi_x(t)), \quad \pi_x(0) = x$$

El conjunto de puntos asignados al modo m_j se llama la cuenca de atracción de m_j y se denota por C_j. Los conjuntos $C_1, \ldots, C_k$ son los grupos poblacionales. La Figura 3.6a muestra una densidad bivariada con cuatro modos, y la Figura 3.6b muestra la partición inducida por los modos.

Para estimar los grupos, encontramos los modos $\hat{M} = \{\hat{m}_1, \ldots, \hat{m}_r\}$ de la estimación de densidad. Un algoritmo simple llamado *algoritmo de desplazamiento medio* (ver [39]) se puede usar para encontrar los modos y encontrar el destino de cualquier punto x. Para cualquier x dado, definimos la iteración

$$x^{(j+1)} = \frac{\sum_i X_i K\left(\frac{||x^{(j)} - X_i||}{b}\right)}{\sum_i K\left(\frac{||x^{(j)} - X_i||}{b}\right)} \tag{3.2}$$

De esta manera obtenemos un gráfico como el que podemos ver en la Figura 3.7

Se puede demostrar bajo condiciones de regularidad adecuadas que los modos de la estimación de densidad del kernel son estimaciones consistentes de los modos de la verdadera densidad. Una vez más, sin embargo, no es necesario estimar bien la densidad para estimar bien los grupos de modos. Específicamente,

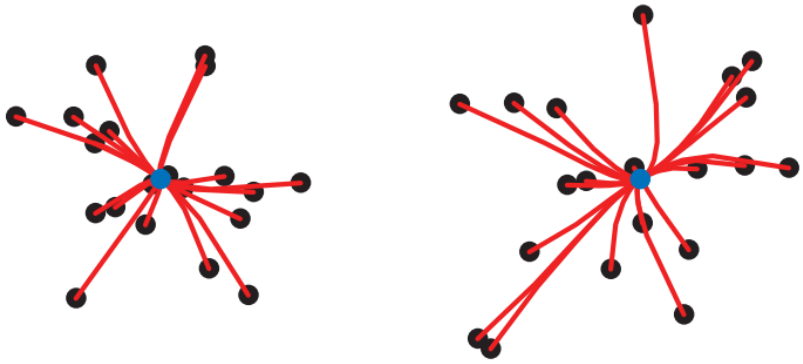

Figura 3.7: El algoritmo de desplazamiento medio. Los datos están representados por los puntos negros. Los dos modos de la estimación de densidad son los dos puntos azules. Las curvas rojas muestran las trayectorias de desplazamiento medio; cada punto de datos se mueve a lo largo de su trayectoria hacia un modo mientras iteramos el algoritmo.

definimos

$$c(x,y) = \begin{cases} 1 & \text{si dest}(x) = \text{dest}(y) \\ 0 & \text{si dest}(x) \neq \text{dest}(y) \end{cases} \tag{3.3}$$

Por lo tanto, $c(x,y) = 1$ si x e y están en el mismo grupo. De manera similar, los grupos estimados definen una función $\hat{c}$. Sea $C_1, \ldots, C_k$ los grupos del modelo. Sea $t_1, \ldots, t_k$ constantes y sea $C_j(t_j) = \{x \in C_j : p(x) > t_j\}$. Los conjuntos $C_1(t_1), \ldots, C(t_k)$ se llaman núcleos de grupo, y estos son los puntos de alta densidad dentro de los grupos. Sea Core $= \{X_i : X_i \in \bigcup_j C_j(t_j)\}$ los puntos de datos en los núcleos de grupo. En este artículo[40] muestran que, si $t_1, \ldots, t_k$ son suficientemente grandes, entonces

$$\mathbb{P}(\hat{c}(X_j, X_k) \neq c(X_j, X_k) \text{ para cualquier } X_j, X_k \in \text{Core}) \leq e^{-nb}$$

para algún $b > 0$, independiente de la dimensión. Esto significa que los puntos de alta densidad pueden agruparse con precisión utilizando la agrupación de modos. [22]

Dominio de aplicación de agrupamiento de modos y teoría de Morse

Aplicaciones de agrupamientos de modos:

Este método se ha usado para analizar imagenes como muestra este artículo de Cheng en 1995[39], en el cual se muestran tres posibles usos del método apoyados de ejemplos.

1. Clustering como proceso natural: Realiza el clustering moviendo iterativamente cada uno de los puntos de datos hacia áreas donde la densidad es mayor hasta que convergen en los modos de la distribución de densidad, de esta manera se generan clusters naturales sin necesidad de especificar cuantos clusters existían previamente. Este método ofrece gran adaptación

a diferentes formas y densidades de datos. En la Figura 3.8, se ha seleccionado aleatoriamente un conjunto S de datos de tamaño 100 en el cuadrado unitario (Fig. 3.8a) y aplicado cinco variaciones diferentes de ”mean shift” o desplazamiento medio. El kernel Gaussiano truncado se utilizó en las Figs. 3.8b, 3.8c y 3.8d, mientras que el kernel Gaussiano “no truncado,” se utilizó en las Figs. 3.8e y 3.8f. El proceso de desenfoque se utilizó en la Fig. 3.8b y el proceso sin desenfoque se utilizó en otros lugares.En (b), (c) y (e), los puntos de datos tenían el mismo peso, mientras que en (d) y (f) no.

2. Validación del clustering: La cual se centra en dos aspectos: estabilidad y separación. La estabilidad se comprueba introduciendo ruido y observando la robustez de los clusters. La separación se mide evaluando la distancia entre los modos de los clusters. Se utilizan visualizaciones y medidas cuantitativas, como la matriz de confusión y el análisis de varianza, para validar la calidad del clustering. Para este caso la Figura 3.9 es un gráfico del proceso de desenfoque aplicado a las velocidades de 82 galaxias[41]. Los mismos datos fueron ajustados con un modelo de mezcla normal bimodal por Roeder[42] y la conclusión fue que el universo observable contenía dos supercúmulos de galaxias rodeados por grandes vacíos. Sin embargo, al observar la Fig.3.9, se puede ver que el resultado más estable es de tres clústeres en lugar de dos. Esto muestra que mientras que la mezcla bimodal o el agrupamiento k-medias requieren cierto supuesto previo sobre el número de clústeres, el resultado del agrupamiento de modas es menos arbitrario.

3. Aplicación de la Transformada de Hough: Usando el agrupamiento de modos para mejorar la detección de formas geométricas en imágenes. En lugar de buscar picos en la matriz de acumulación, el agrupamiento de modos identifica los modos de densidad en el espacio de parámetros, mejorando la precisión y robustez de la detección de formas geométricas, especialmente en presencia de ruido y datos dispersos.

   La Figura 3.10 muestra una aplicación del agrupamiento de modos en la transformada generalizada de Hough. Se seleccionaron aleatoriamente 300 píxeles del borde de una imagen de 100 x 100 y cada par de ellos generó una línea recta que los atraviesa. Las intersecciones de estas líneas y los bordes de la imagen se redondean al entero más cercano (posiciones de píxeles) y se registran con acumuladores de 400 x 400 en el espacio de parámetros, similar a la transformada ”muff” sugerida por Wallace[43].

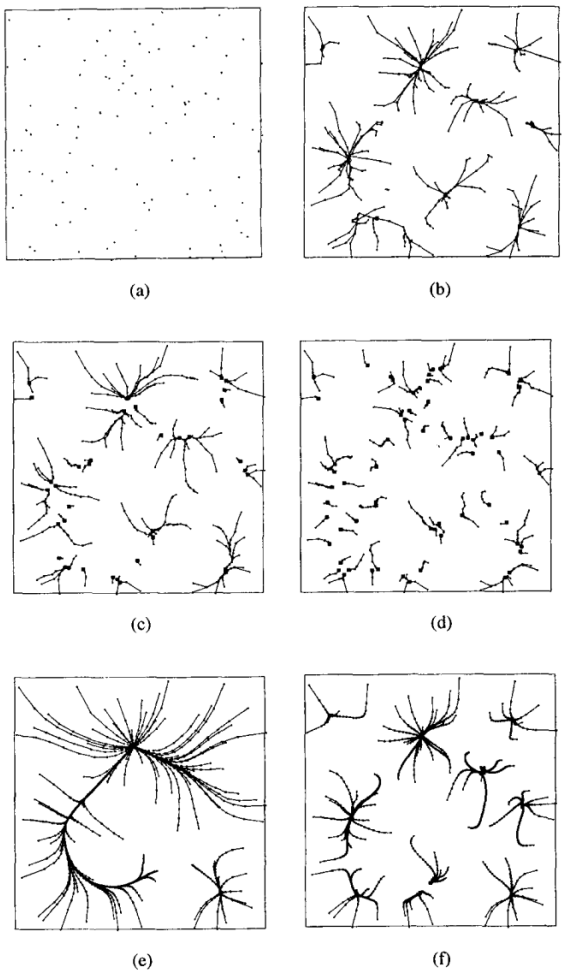

Figura 3.8: Trayectorias de diferentes variaciones del agrupamiento de modas en el mismo conjunto de datos. (a) El conjunto de datos S, también el conjunto inicial T para los procesos sin desenfoque. (b) Un proceso de desenfoque con 10 iteraciones y nueve clústeres. (c) Un proceso de agrupamiento de modas sin desenfoque con un kernel Gaussiano truncado y pesos uniformes. Terminó en 20 iteraciones con 37 clústeres. (d) agrupamiento de modos sin desenfoque con un kernel Gaussiano truncado y una función de peso adaptativa. Terminó en 20 iteraciones con 64 clústeres. (e) Sin desenfoque con un kernel Gaussiano no truncado y pesos uniformes. Terminó en 274 iteraciones con dos clústeres. (f) Sin desenfoque con un kernel Gaussiano no truncado y pesos adaptativos. Terminó en 127 iteraciones con 13 clústeres.

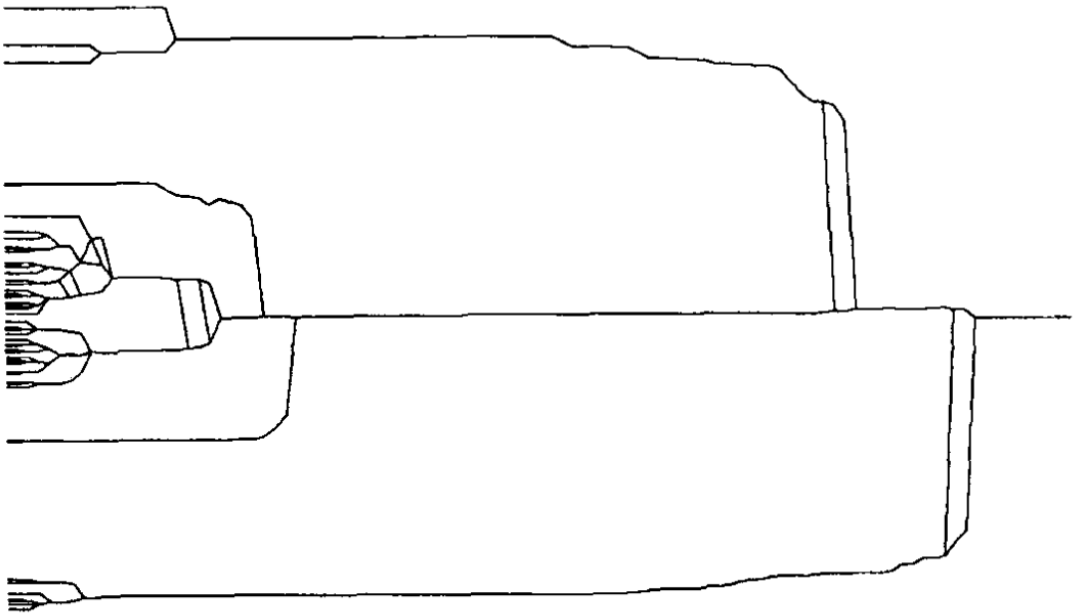

Figura 3.9: Agrupamiento de 82 galaxias basado en sus velocidades. El diagrama en forma de árbol muestra la validez relativa de los resultados del agrupamiento con el kernel $(GF)_\lambda$ de diferentes valores de λ. Se utilizaron valores de λ más grandes cerca de la raíz del árbol, mientras que se utilizaron valores de λ más pequeños cerca de las hojas del árbol, donde la dimensión a lo largo del árbol indica las velocidades.

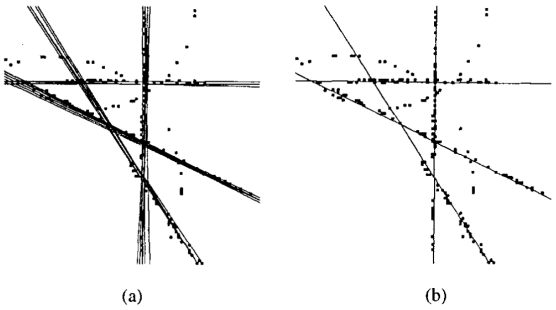

Figura 3.10: Detección de picos de desplazamiento medio en el espacio de parámetros de la transformada de Hough (muff). La imagen contiene 300 píxeles de borde. Los pares de píxeles de borde hacen 44,850 sugerencias de posibles líneas que se codifican como puntos en el espacio de parámetros. Cada línea se codifica con dos enteros entre 0 y 400, que etiquetan las intersecciones de la línea con los bordes de la imagen. (a) muestra la detección de líneas basada en el número de sugerencias que caen en cada punto en el espacio de parámetros. Se utilizó un umbral de 30. (b) es el resultado después de aplicar el desplazamiento medio o "mean shift" a estas líneas en el espacio de parámetros. Los pesos fueron el número de sugerencias, y surgieron cuatro clústeres como las cuatro líneas mostradas

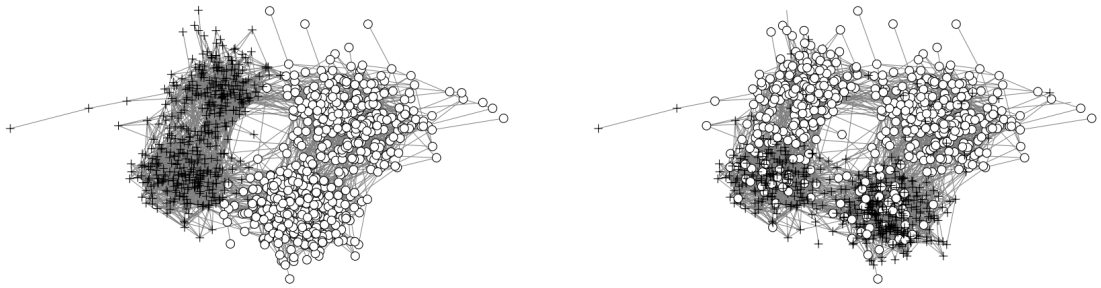

Figura 3.11: Red social para "Countryside High School" y su escuela intermedia hermana. (Izquierda) Los vértices en forma de cruz son estudiantes de la escuela intermedia y los vértices en forma de círculo son estudiantes de la escuela secundaria. (Derecha) Los vértices en forma de cruz son estudiantes blancos y los vértices en forma de círculo son estudiantes no blancos. Un puñado de estudiantes no tienen datos raciales y sus vértices no están marcados.

Aplicaciones de Teoría de Morse:

Aplicaciones para redes sociales:

El "National Longitudinal Study of Adolescent Health" o Estudio Longitudinal Nacional de Salud del Adolescente es un estudio basado en escuelas de jóvenes estadounidenses. En 1994-95 se eligió una muestra de "high schools" o escuelas secundarias y "middle schools" o escuelas intermedias, y cuando fue posible, cada escuela secundaria se emparejó con una escuela intermedia hermana de la misma comunidad. A los estudiantes de cada escuela se les pidió que enumeraran hasta cinco de sus amigas más cercanas que asisten a su escuela o a su escuela hermana, y lo mismo para sus amigos varones.

Moody[44] crea gráficos de redes a partir de los datos de la encuesta. Cada estudiante está representado por un vértice, y el borde entre dos estudiantes existe si cada estudiante enumeró al otro como amigo cercano. Analizamos el gráfico para los estudiantes de "Countryside High School", un seudónimo utilizado por Moody, y su escuela intermedia hermana. De los 1,147 estudiantes de esta comunidad que participaron en la encuesta, 20 no compartieron conexiones con otros y fueron eliminados del gráfico.

La Figura 3.11 ilustra la siguiente estructura interesante en este gráfico. La mayoría de los estudiantes pueden ubicarse en uno de los cuatro grupos, que contienen una mayoría de estudiantes de secundaria blancos, estudiantes de secundaria no blancos, estudiantes de escuela intermedia blancos o estudiantes de escuela intermedia no blancos. Hay muchas amistades entre estudiantes del mismo grupo. Además, hay un número significativo de amistades entre grupos de la misma escuela o categoría racial. Sin embargo, hay muy pocas amistades entre grupos que no comparten ninguna categoría en común.

Su objetivo es utilizar su enfoque de teoría de Morse para recuperar esta estructura. Utilizan la mayorización de estrés, una estrategia de optimización en el escalado multidimensional[45], para embeber los vértices del gráfico como un

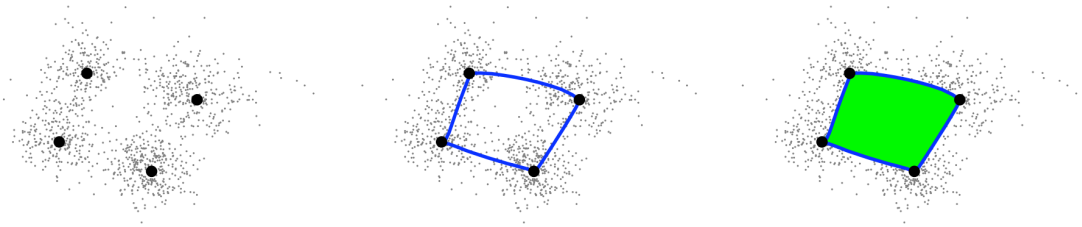

Figura 3.12: Datos de red social, proyectados a un plano utilizando análisis de componentes principales (PCA). El complejo Z^a crece a medida que el umbral de densidad a disminuye. De izquierda a derecha, tenemos cuatro 0-celdas, un bucle cuadrado y un disco.

conjunto de datos $X \subset \mathbb{R}^5$ de manera que distorsione el métrica de camino más corto lo menos posible. La distancia de camino más corto entre los vértices v y w es el menor número de bordes que se deben cruzar para viajar de v a w. Cada punto $x \in X$ corresponde a un estudiante, y consideramos el conjunto resultante X como nuestro conjunto de datos de entrada.

Su método encuentra cuatro 0-celdas, que corresponden a los cuatro grupos de estudiantes. Encuentran cuatro 1-celdas que forman un bucle cuadrado. Las 0-celdas tienen densidades en $[1{,}69 \times 10^{-2}, 2{,}02 \times 10^{-2}]$ y las 1-celdas en $[1{,}07 \times 10^{-2}, 1{,}54 \times 10^{-2}]$. Se encuentra una 2-celda que llena el bucle con densidad $0{,}79 \times 10^{-2}$.

Recuerde que el complejo CW Z^a se define como la unión de las celdas con densidad mayor o igual a a. Para $a \in (1{,}54 \times 10^{-2}, 1{,}69 \times 10^{-2})$, el modelo Z^a consiste en cuatro 0-celdas (Figura 3.12). Por lo tanto, recuperan las agrupaciones de estudiantes basadas en la escuela y la raza. Para $a \in (0{,}79 \times 10^{-2}, 1{,}07 \times 10^{-2})$, el modelo Z^a es un cuadrado (Figura 3.12). El cuadrado revela que los grupos que comparten la escuela o la categoría racial están más estrechamente vinculados que los grupos que comparten ninguna. Esto sugiere que es más difícil cruzar dos barreras culturales que una. Para $a < 0{,}79 \times 10^{-2}$, el modelo Z^a se llena hasta un disco (Figura 3.12), que es una representación apropiada de los datos a una escala lo suficientemente gruesa. Tenga en cuenta que esta secuencia creciente de modelos de complejo CW proporciona una mejor comprensión del conjunto de datos que cualquier modelo único por sí solo.

Datos de expresión génetica:

En tres estudios anteriores de Spellman et al en 1998[46], Alter et al en 2000[47] y Whitfield et al en 2002[48] se identifican genes asociados con el ciclo celular utilizando análisis de microarrays, y los niveles de expresión de estos genes son cíclicos. Una forma de modelar el comportamiento cíclico matemáticamente es con una función periódica. Sin embargo, mientras una función periódica tiene una amplitud y un período fijos, el comportamiento cíclico en biología no es tan rígido. La amplitud y el período de las expresiones génicas del ciclo celular cambian con el tiempo. En particular, la amplitud tiende a disminuir, lo que puede interpretarse como la desincronización de células inicialmente sincronizadas. Otra forma de modelar el comportamiento cíclico matemáticamente es con un

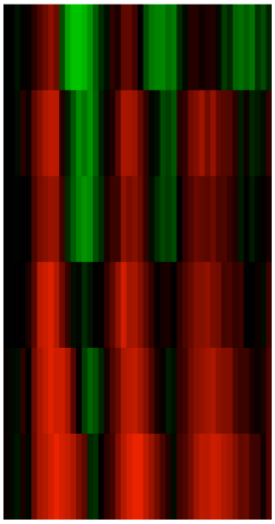

Figura 3.13: Datos de microarrays para los seis genes CDC2, UBE2C, TOP2A, CCNF, AURKA y PLK1. Cada fila es un gen y cada columna es un punto de tiempo. El color rojo corresponde a una alta expresión y el color verde a una baja expresión. Aproximadamente se pueden observar tres ciclos celulares.

círculo, y esta representación topológica es robusta a los cambios en amplitud y frecuencia.

En este articulo de Whitfield et al en 2002[48] miden los niveles de expresión génica a medida que las células HeLa pasan por el ciclo celular. Seleccionamos genes cuyas mediciones pasan varios umbrales de calidad y que se desvían significativamente de la expresión promedio. Su umbral es estricto ya que el propósito de este ejercicio no es descubrir nueva biología sino probar su método en datos de series temporales. Normalizan y agrupan los genes resultantes. Un grupo consta de seis genes que son bien conocidos por ser parte del ciclo celular y que tienen niveles de expresión periódicos, como se muestra en la Figura 3.13. Para el punto de tiempo $t \in \{0, 1, \ldots, 46\}$, sea w_t el vector vertical de longitud seis que contiene los niveles de expresión de estos genes en el tiempo t. Usan bloques de cinco puntos de tiempo para formar el conjunto de datos. Definimos $X = \{x_0, \ldots, x_{42}\} \subset \mathbb{R}^{30}$, donde:

$$x_0 = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_4 \end{pmatrix}, \quad x_1 = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_5 \end{pmatrix}, \quad \ldots, \quad x_{42} = \begin{pmatrix} w_{42} \\ w_{43} \\ \vdots \\ w_{46} \end{pmatrix}.$$

Aplican a su método a X y encuentran tres 0-celdas y tres 1-celdas que forman un círculo. Estas celdas tienen densidades en el rango $[9{,}4 \cdot 10^{-18}, 12{,}2 \cdot 10^{-18}]$. Encuentran una 2-celda llenando el círculo con densidad $4{,}0 \cdot 10^{-18}$. El modelo Z_a para $a \in (4{,}0 \cdot 10^{-18}, 9{,}4 \cdot 10^{-18})$ es un círculo que recupera la naturaleza cíclica de las expresiones génicas (Figura 3.14).

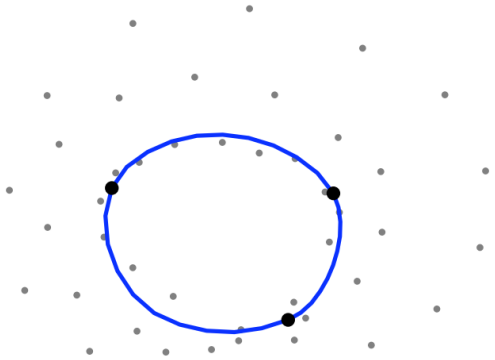

Figura 3.14: Conjunto de datos de expresión génica X, proyectado a un plano usando PCA. El complejo Z^a es un círculo para $a \in (4{,}0 \cdot 10^{-18}, 9{,}4 \cdot 10^{-18})$.

3.2. Homología persistente

La homología persistente es una extensión de la homología al contexto de complejos de cadenas filtradas. Las cadenas filtradas son una estructura matemática que consiste en una secuencia de complejos de cadenas $C_{\varepsilon_0} \subset C_{\varepsilon_1} \subset C_{\varepsilon_2} \subset \ldots$, donde cada complejo en la secuencia incluye al siguiente. Estos complejos surgen en situaciones donde tenemos una secuencia de espacios $X_{\varepsilon_0} \subset X_{\varepsilon_1} \subset X_{\varepsilon_2} \subset \ldots$.

Por ejemplo, podemos construir estos complejos considerando conjuntos de nivel de una función de filtración $f : X \to \mathbb{R}$ o conjuntos de bolas abiertas alrededor de una nube de puntos en un espacio métrico (M, d). Una propiedad fundamental de estos complejos es que las aplicaciones de inclusión entre ellos inducen aplicaciones en los grupos de homología correspondientes. Este concepto, conocido como 'functorialidad', establece una conexión esencial entre la estructura de los complejos de cadenas y los grupos de homología, lo que resulta fundamental en el análisis topológico de datos.

La homología persistente, que se basa en este principio, se utiliza para analizar cómo las características topológicas de los datos evolucionan a medida que cambia un parámetro. La Figura 3.15 ilustra un ejemplo de la construcción de uno de estos complejos, llamado complejo de Vietoris-Rips, que se fundamenta en una filtración de distancia y es comúnmente utilizado en el análisis topológico de datos.

Los **grupos de homología persistente** de grado k son las imágenes de estas inclusiones. En otras palabras, consisten en las clases de homología de grado k de C_{ε_i} que aún existen después de aplicar la inclusión $H_k(\iota_{i,j})$. Una clase de homología α se dice que nace en C_{ε_i} si no está en la imagen de $H_k(\iota_{i-1,i})$. Si α nace en C_{ε_i}, se dice que muere en C_{ε_j} si no está en la imagen inversa de $H^k_{i-1,j-1}$ pero sí lo está en la imagen inversa de $H^k_{i-1,j}$. La persistencia de α se calcula como la diferencia entre ε_j y ε_i, y se considera infinita si nunca muere.

Los **números de Betti** persistentes proporcionan información sobre cómo cambia la topología a lo largo de la filtración. Este cambio se visualiza en un diagrama de persistencia, donde cada punto representa el nacimiento y la muerte de una característica topológica a diferentes escalas. Cada eje del diagrama representa los valores de ε en los que se crean y destruyen las características topológicas, respectivamente.

El **diagrama de persistencia** se compone de puntos en el plano $\mathbb{R}^2$, donde la coordenada x representa el valor de ε en el que una característica topológica nace, y la coordenada y representa el valor de ε en el que muere. La multiplicidad de un punto en el diagrama de persistencia indica cuántas clases de homología mueren en ese mismo intervalo de persistencia. Es decir, cuántas características topológicas aparecen y desaparecen en el mismo rango de escalas. Por ejemplo, si hay un punto con multiplicidad dos en $(3, 5)$, significa que dos clases de homología nacen en $\varepsilon = 3$ y mueren en $\varepsilon = 5$.

La homología persistente es valiosa en análisis de datos debido a su estabilidad frente a perturbaciones en la función de filtración. Esto significa que es robusta ante el ruido y codifica propiedades topológicas intrínsecas de los datos. El espacio de diagramas de persistencia puede ser dotado de una métrica inducida por la **bottleneck** (o las distancias de **Wasserstein**) [49].

Un teorema de estabilidad célebre establece que la distancia L^∞ de dos funciones reales f y g es una cota superior para la distancia de botella W^∞ de sus respectivos diagramas de persistencia D_f y D_g, es decir, $W^\infty(D_f, D_g) \leq \|f - g\|_\infty$. Este teorema de estabilidad y sus variantes son altamente relevantes para aplicaciones, ya que implican que el comportamiento de la homología persistente bajo ruido es conocido; los descriptores como los diagramas de persistencia cambian continuamente a medida que la función de entrada varía, y la "amplitud" de su cambio está acotada por encima mediante el teorema de estabilidad [50], [51].

3.2.1. Dominio de aplicación de Homología Persistente

1. *Redes:* Se puede interpretar una red no dirigida como un complejo simplicial de una dimensión. Si la red tiene peso en las aristas, entonces filtrar por aumento o disminución de peso produce un complejo simplicial filtrado de una dimensión. Para obtener información más refinada sobre la red, es deseable construir símplices de dimensiones superiores. Existen varios métodos para hacer esto. El método más simple, llamado filtración de cliques de rango de peso (WRCF)[52], consiste en construir un complejo de cliques en cada subred. Otro método para estudiar redes con PH consiste en asignar los nodos de la red a puntos de un espacio métrico finito. Hay varias formas de calcular distancias entre nodos de una red; una de ellas consiste en calcular un camino más corto entre nodos. Para que esta distancia esté bien definida, se necesita que la red esté conectada (aunque convencionalmente se considera que la distancia entre nodos en diferentes componentes es infinita). Hay muchos métodos para asociar un complejo simplicial no filtrado tanto a redes no dirigidas como dirigidas. [3]

Figura 3.15: Diferentes etapas de una filtración de Vietoris-Rips para una nube de puntos 'círculo' simple. De izquierda a derecha, la conectividad del complejo simplicial subyacente aumenta a medida que ϵ aumenta.

2. *Imágenes digitales:* Las imágenes digitales tienen una estructura cúbica natural: las imágenes digitales bidimensionales están hechas de píxeles, y las imágenes tridimensionales están hechas de voxels. Por lo tanto, para estudiar imágenes digitales, los complejos cúbicos son más apropiados que los complejos simpliciales. A grandes rasgos, los complejos cúbicos son espacios construidos a partir de una unión de vértices, aristas, cuadrados, cubos, etc. Se puede calcular PH para complejos cúbicos de manera similar a como se hace para complejos simpliciales. Otros enfoques para estudiar imágenes digitales también son útiles. En general, dado una imagen digital que consiste en N píxeles o voxels, se puede considerar esta imagen como un punto en un espacio dimensional c × N, con cada coordenada almacenando un vector de longitud c que representa el color de un píxel o voxel. Definir una función de distancia adecuada en dicho espacio permite considerar una colección de imágenes (cada una de las cuales tiene N píxeles o voxels) como un espacio métrico finito. [3]

3. *Espacios métricos finitos:* Como mencionamos en las dos aplicaciones anteriores, tanto las redes no dirigidas como los datos de imágenes se pueden interpretar como espacios métricos finitos. Por lo tanto, los métodos para estudiar espacios métricos finitos con PH se aplican al estudio de redes y conjuntos de datos de imágenes.

   En algunas aplicaciones, los puntos de un espacio métrico tienen 'pesos' asociados. Por ejemplo, en el estudio de moléculas, se puede representar una molécula como una unión de bolas en el espacio euclidiano. Para tales conjuntos de datos, también se consideraría un valor de filtración mínimo. [3]

3.3. Algoritmo Mapper

Este método se basa en ideas topológicas, lo que significa que preserva una noción de cercanía, pero puede distorsionar distancias a gran escala. Esta propiedad es a menudo deseable, ya que si bien las funciones de distancia a menudo codifican una noción de similitud o cercanía, las distancias a gran escala a

menudo carecen de significado.

El método comienza con un conjunto de datos X y una función de valor real $f : X \to \mathbb{R}$, para producir un grafo. Esta función puede ser una función que refleje propiedades geométricas del conjunto de datos, como el resultado de un estimador de densidad, o puede ser una función definida por el usuario, que refleje propiedades del conjunto de datos en estudio. En el primer caso, se intenta obtener información sobre las propiedades cualitativas del conjunto de datos en sí, y en el segundo caso, se intenta comprender cómo estas propiedades interactúan con funciones interesantes en el conjunto de datos.

Las funciones determinan el espacio al que producimos un mapa. El método puede modificarse fácilmente para lidiar con mapas a espacios de parámetros distintos de $\mathbb{R}$, como $\mathbb{R}^2$ o el círculo unitario S^1 en el plano. En el primer caso, se produce un complejo simplicial bidimensional, junto con un mapa natural desde el conjunto de datos hacia él. En el segundo caso, se construye un grafo con un mapa desde el grafo hacia un círculo.

En el caso donde el espacio de parámetros objetivo es $\mathbb{R}$, nuestra construcción equivale a una versión estocástica del grafo de Reeb[15] asociado con la función de filtro. Si la cobertura de $\mathbb{R}$ es demasiado gruesa, estaremos construyendo una imagen del grafo de Reeb[53] de la función, mientras que si es lo suficientemente fina, recuperaremos el grafo de Reeb precisamente.

La idea básica puede denominarse agrupamiento parcial[15], en el sentido de que un paso clave es aplicar algoritmos de agrupamiento estándar a subconjuntos del conjunto de datos original, y luego comprender la interacción de los clústeres parciales formados de esta manera entre sí. Es decir, si U y V son subconjuntos del conjunto de datos, y $U \cap V$ no es vacío, entonces los clústeres obtenidos de U y V respectivamente pueden tener intersecciones no vacías, y estas intersecciones se utilizan para construir un complejo simplicial.

Esta construcción produce una imagen "multiresolución" o "multiescala" del conjunto de datos. En realidad, se puede construir una familia de complejos simpliciales (grafos en el caso de un espacio de parámetros unidimensional), que se ven como imágenes en niveles variables de grosor, y mapas entre ellos que van desde un complejo a una resolución más fina a uno de resolución más gruesa. Este hecho permite evaluar hasta qué punto las características son "reales" en lugar de "artefactos", ya que las características que persisten sobre un rango de valores de grosor se verían como menos propensas a ser artefactos.

No intentamos obtener una representación completamente precisa de un conjunto de datos, sino más bien una imagen de baja dimensión que sea fácil de entender y que pueda señalar áreas de interés. Tengamos en cuenta que implícito en el método está el hecho de que se fija un espacio de parámetros, y su dimensión será una cota superior de la dimensión del complejo simplicial que se estudia. Como tal, es de cierta manera análogo a la idea de una torre de Postnikov o la filtración cosesquelética en topología algebraica.

La construcción en este método está motivada por la siguiente idea. Dada una cobertura finita $U = \{U_\alpha\}_{\alpha \in A}$ de un espacio X, definimos el nervio de la cobertura

U como el complejo simplicial $N(U)$ cuyo conjunto de vértices es el conjunto de índices A. Una familia $\{\alpha_0, \alpha_1, \ldots, \alpha_k\}$ forma un k-simplejo en $N(U)$ si y solo si $U_{\alpha_0} \cap U_{\alpha_1} \cap \ldots \cap U_{\alpha_k} \neq \emptyset$. Con información adicional, una partición de la unidad, se puede obtener un mapa de X a $N(U)$. Una partición de la unidad subordinada a la cobertura abierta finita U es una familia de funciones reales $\{\phi_\alpha \in A\}_{\alpha \in A}$ con ciertas propiedades.

Recordamos que si $\{v_0, v_1, \ldots, v_k\}$ son los vértices de un simplejo, entonces los puntos v en el simplejo corresponden de manera biunívoca al conjunto de k-tuplas ordenadas de números reales $(r_0, r_1, \ldots, r_k)$ que satisfacen ciertas condiciones.

Luego, para cualquier punto x en X, dejamos que $T(x) \subseteq A$ sea el conjunto de todos los α tal que x está en U_α. Definimos $\rho(x)$ en $N(U)$ como el punto en el simplejo generado por los vértices α en $T(x)$, cuyas coordenadas baricéntricas son ciertos valores determinados por las funciones $\phi_\alpha(x)$. La continuidad de ρ se puede verificar fácilmente, y proporciona una especie de coordinatización parcial de X en el complejo simplicial $N(U)$.

Supongamos ahora que se nos da un espacio equipado con un mapa continuo $f : X \to Z$ a un espacio de parámetros Z, y que Z está equipado con una cobertura $U = \{U_\alpha\}_{\alpha \in A}$ para algún conjunto finito de índices A. Dado que f es continuo, los conjuntos $f^{-1}(U_\alpha)$ también forman una cobertura abierta de X. Para cada α, podemos considerar la descomposición de $f^{-1}(U_\alpha)$ en sus componentes conexas por caminos, denotadas como $\{V(\alpha, i)\}_{i=1}^{j_\alpha}$, donde j_α es el número de componentes conexas en $f^{-1}(U_\alpha)$. Llamamos U a la cobertura de X obtenida de esta manera a partir de la cobertura U de Z.[15]

3.3.1. Dominio de aplicación del algoritmo Mapper

1. *El estudio de la diabetes de Miller-Reaven:* En el articulo[54], G. M. Reaven y R. G. Miller describen los resultados obtenidos al aplicar el método de búsqueda de proyección [55] a datos [56] obtenidos de un estudio realizado en la Universidad de Stanford en la década de 1970. Participaron en el estudio 145 pacientes que tenían diabetes, antecedentes familiares de diabetes, que deseaban un examen físico o participar en un estudio científico. Se midieron seis cantidades para cada paciente: edad, peso relativo, glucosa plasmática en ayunas, área bajo la curva de glucosa plasmática durante la prueba de tolerancia a la glucosa de tres horas (OGTT), área bajo la curva de insulina plasmática para la (OGTT) y respuesta de glucosa plasmática en estado estable. Esto creó un conjunto de datos de 6 dimensiones, que se estudió utilizando métodos de búsqueda de proyección, obteniendo una proyección en un espacio euclidiano tridimensional, en la cual el conjunto de datos aparece como en la Figura 3.16.

   Miller y Reaven observaron que el conjunto de datos consistía en un núcleo central y dos "flares" que emanaban de él. Los pacientes en cada uno de los flares se consideraban como sufriendo de enfermedades esencialmente diferentes, que corresponden a la división de la diabetes en las formas de

inicio en adultos y juvenil. Una manera en la que deseamos utilizar Mapper es como una herramienta automática para detectar tales flares en los datos, incluso en situaciones donde las proyecciones en espacios bidimensionales o tridimensionales no proporcionan una imagen clara. La Figura 3.17 muestra los resultados obtenidos al aplicar Mapper a este mismo conjunto de datos, utilizando densidad estimada por un estimador de núcleo. Mostramos dos resoluciones diferentes.[15]

2. *Algoritmo Mapper en una base de datos con formas 3D:* Aplicamos Mapper a una colección de formas tridimensionales de una base de datos pública de objetos [57]. Estas formas corresponden a siete clases diferentes de objetos (camellos, gatos, elefantes, caras, cabezas, caballos y leones). Para cada clase hay entre 9 y 11 objetos (ver Figura 3.18). Para utilizar estas formas como entrada de nube de puntos en Mapper, preprocesamos las formas seleccionando 4000 puntos de referencia de un conjunto de puntos P, utilizando un procedimiento de máximo mínimo euclidiano. Llamamos a este conjunto Y, donde L es el conjunto de índices de los puntos de referencia. Calculamos las distancias entre los puntos en Y mediante una matriz de adyacencia A calculada a partir de la información de la malla de puntos P, y luego la matriz de distancias D entre puntos de Y usando el algoritmo de Dijkstra en el grafo especificado por A.

   Para aplicar Mapper a este conjunto de formas, elegimos usar $E_1(x)$ como nuestra función de filtro. Para minimizar el sesgo debido a la distribución de características locales, utilizamos un umbral global para la agrupación dentro de todos los intervalos, determinado mediante la heurística del histograma y la mediana de estos umbrales como el umbral global.

   La salida de Mapper en este caso es un grafo. Algunas observaciones de los resultados incluyen la capacidad de Mapper para recuperar el esqueleto de la forma con precisión aceptable, mostrando resultados cualitativamente similares para poses diferentes de la misma forma y resultados significativamente diferentes para formas diferentes, lo que sugiere la retención de cierta información intrínseca invariante a la pose. Este comportamiento sugiere la utilidad de Mapper como herramienta para simplificar formas y realizar consultas en bases de datos y tareas de comparación de formas invariantes.

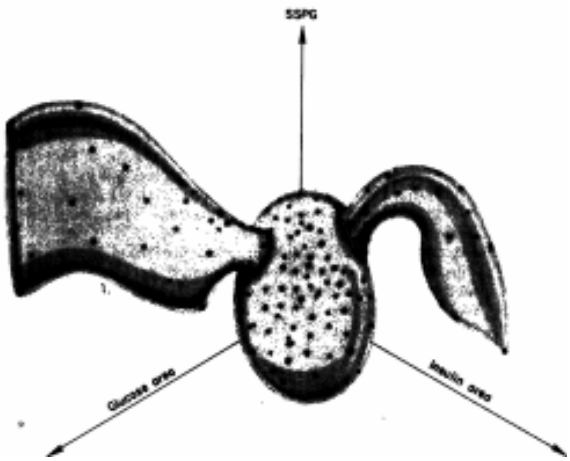

Figura 3.16: Proyección tridimensional de los datos de la diabetes obtenida utilizando proyección y búsqueda.

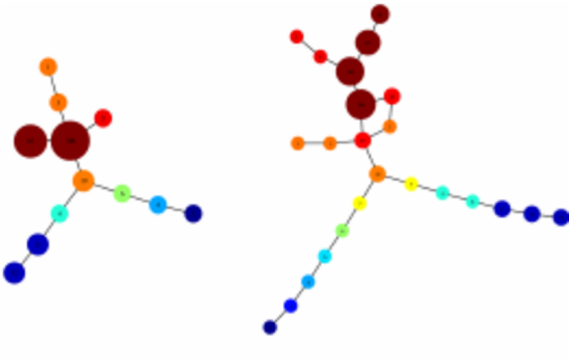

Figura 3.17: A la izquierda se muestra una salida de Mapper de "baja resolución", que se calculó utilizando 3 intervalos en el rango del filtro con un solapamiento del 50 %. A la derecha se muestra una salida de Mapper de "alta resolución" calculada utilizando 4 intervalos en el rango del filtro con un solapamiento del 50 %. Los colores codifican los valores de densidad, con rojo indicando alta densidad y azul baja densidad. El tamaño del nodo y el número en él indican el tamaño del clúster. Los extremos de baja densidad reflejan la diabetes tipo I y tipo II. Las ramificaciones que ocurren en la Figura 3.16 ocurren aquí como ramificaciones con extremos azules.

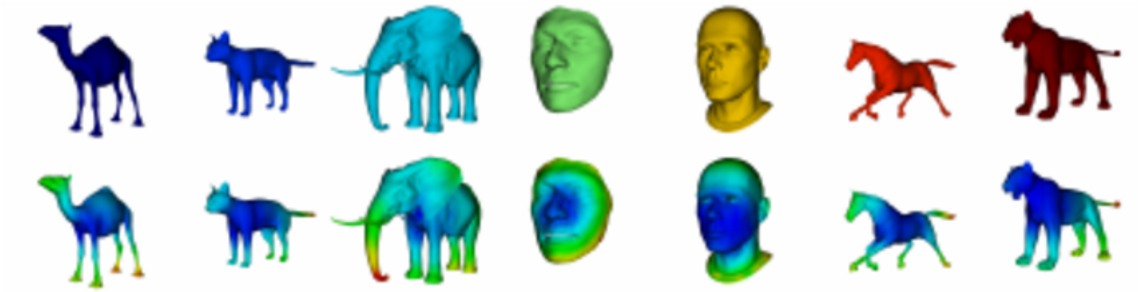

Figura 3.18: La fila superior muestra la representación de un modelo de cada una de las 7 clases. La fila inferior muestra el mismo modelo coloreado por la función.

Capítulo 4

Herramientas de TDA

Ahora que ya tenemos una noción básica de alguno de los múltiples métodos que existen para aplicar el análisis topológico de datos podemos dar paso a introducir algunos de los softwares más actuales que nos permitan aplicar estos métodos. La mayoría de estos softwares están escritos en C++[58] por lo que se necesitan ciertos conocimientos de este lenguaje para entender plenamente que hace el código pero trataré de explicarlo de tal forma que sea lo más fácil de comprender para todos los lectores. Por el momento comenzaré hablando del Topology Toolkit[59]

4.1. Topology ToolKit

The Topology ToolKit (TTK) [59] es una biblioteca de código abierto para TDA que fue lanzada en 2017 bajo la licencia permisiva BSD[60]. Cuenta con una colección sustancial de implementaciones genéricas y eficientes de algoritmos de TDA de referencia. TTK está principalmente escrito en C++ (~110k líneas de código) para ofrecer el mejor rendimiento posible. Hasta la fecha, 15 instituciones han contribuido con código a TTK, incluidas 12 organizaciones académicas (Arizona State University[61], CNRS[62], Heidelberg University[63], INRIA[64], Linkoping University[65], Los Alamos National Laboratory[66], Sorbonne Universite[67], TU Kaiserslautern[68], University of Arizona[69], University of Leeds[70], University of Utah[70], Zuse Institute Berlin[71]) y 3 empresas (Kitware[72], Total[73], ShapeShift3D[74]).

TTK es accesible para los desarrolladores a través de varias APIs: C++, VTK/C++ o Python[75]. Para los usuarios finales, TTK es accesible directamente en forma de un complemento para ParaView [76] y un paquete anaconda [77]. Los datos pueden ser proporcionados a TTK en múltiples formas: pueden ser muestreados a lo largo de rejillas regulares 1D, 2D o 3D (incluyendo rejillas periódicas), o mallas 1D, 2D o 3D (complejos simpliciales). También pueden ser proporcionados como nubes de puntos de dimensión arbitraria.

Para mostrar el funcionamiento del software mostraré un ejemplo de uso del mismo, incluiré el código necesario para llevarlo acabo el cual no explicaré en

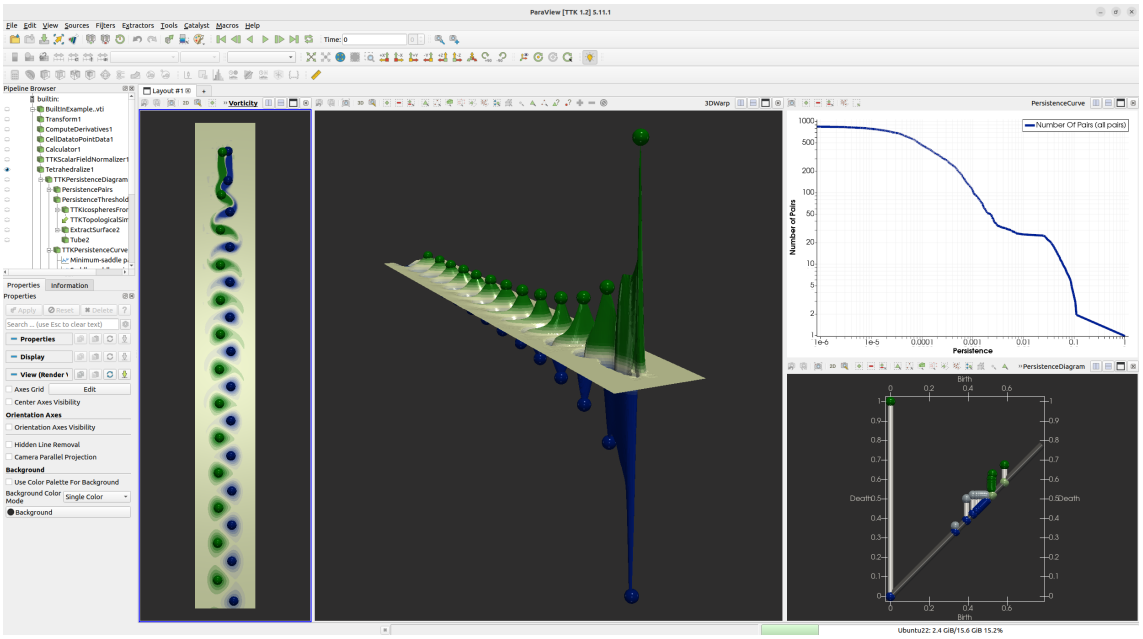

Figura 4.1: Visualización del ejemplo integrado ”Scalar data example”

detalle sino que mencionaré de forma general como se ha llevado acabo. Para hacer uso de esta herramienta necesitamos utilizar una distribución de Ubuntu[78] 22.04, 20.04 o 18.04. En mi caso voy a utilizar Ubuntu 22.04.4, para llevar acabo la instalación hay que seguir los pasos mencionados en su página oficial en la pestaña ”Installation”[59]. Si todo ha salido correctamente, al arrancar Paraview deberíamos ser capaces de abrir uno de los ejemplos que vienen integrados en el software, lo que nos debería permitir ver la visualización.

Ahora que todo funciona como esperado puedo mostrar un ejemplo de como utilizar la herramienta. Para ello vamos a usar el dataset de TTK, el cual podemos descargar desde su página de descargas[79]. Vamos a escoger uno de los ejemplos integrados y sobre este voy a explicar cómo podemos trabajar sobre él modificando parámetros sobre el ParaView. Para empezar vamos a centrarnos en los datos escalares y más en concreto en ”Morse Persistence” ya que utiliza uno de los métodos comentados anteriormente en este trabajo, la teoría Morse.

4.1.1. Morse Persistence

Para este ejemplo debemos tener descargado el paquete de datos de TTK-data. La manera de usar esta herramienta es simple al ser un ejemplo cuyo código ya está escrito, tan solo tenemos que acceder al directorio donde se encuentra el TTK-data y ejecutar el comando ”paraview states/morsePersistence.pvsm”. Una vez ejecutado se nos abrirá ParaView y podremos visualizar el ejemplo con el que vamos a trabajar (Figura 4.2). Podemos apreciar una superficie triangulada con algunos datos de esqueleto adjuntos a ella, estos son una forma de resaltar las características topológicas importantes, como los puntos críticos y las líneas de gradiente que conectan estos puntos, en este caso estos datos son la

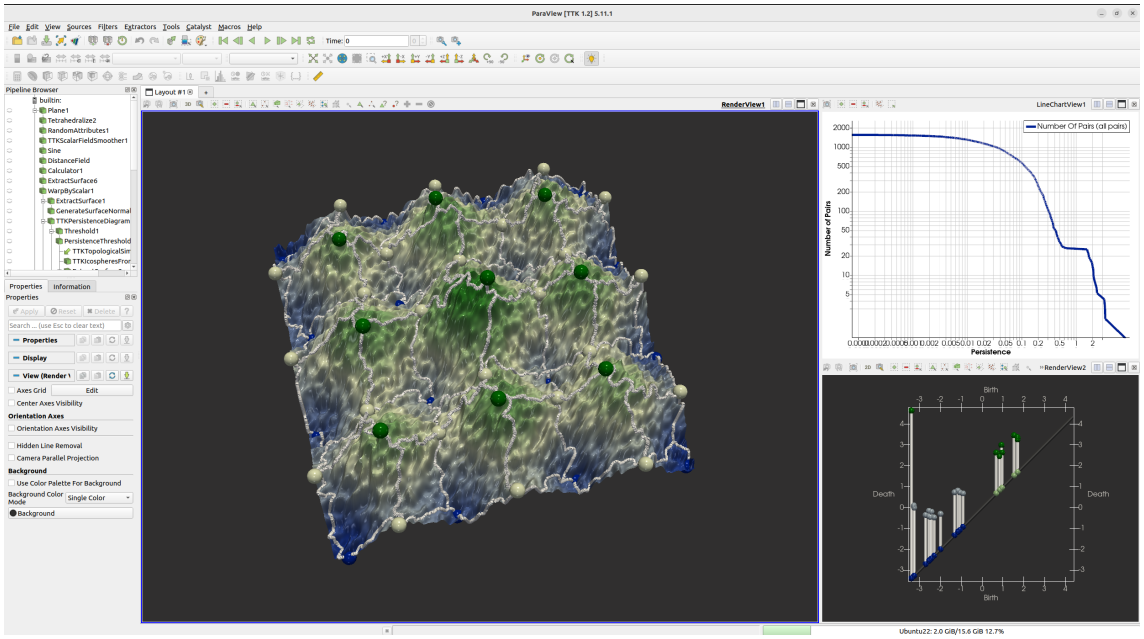

Figura 4.2: Captura de pantalla de ParaView mostrando lo primero que vemos al hacer la demo de persitencia Morse.

elevación, la cual se representa en azul para valores bajos y verde para valores altos. Podemos apreciar que los datos son ruidosos ya que se observan muchas protuberancias muy pequeñas. En cuanto a las esferas de color blanco, verde y azul corresponden a los a los puntos críticos más persistentes de los datos, esto quiere decir que son los puntos con mayor estabilidad o importancia en relación con la estructura de los datos, los puntos críticos en el ”smooth setting” son los puntos en los que las derivadas se anulan, las esferas azules representan un mínimo local y las verdes un máximo local, mientras que las líneas blancas que que conectan dichos puntos son las sendas separadas del complejo Morse, las cuales voy a explicar más adelante.

En la esquina superior derecha (ver Figura4.2) podemos ver la curva de persistencia que describe la distribución de los pares de puntos críticos en función de su persistencia. Esta curva se lee de izquierda a derecha, significa que a medida que aumentas cierto umbral de persistencia, tienes menos pares de puntos críticos que son más persistentes que tu umbral. Habitualmente podemos observar un tipo de comportamiento de doble disminución, donde tienes una primera pendiente la cual indica la eliminación de las características ruidosa y una segunda pendiente que indica la eliminación de las características de gran escala. Podemos apreciar que cuando el eje y, que representa el número de pares, tiene un valor cercano a 30 en este ejemplo y habitualmente tiene este mismo comportamiento donde hay un claro corte entre señal y ruido, lo que nos permite a los usuarios elegir la cantidad correcta de simplificación topológica para los datos, normalmente poner un umbral de persistencia en este valor es una buena idea.

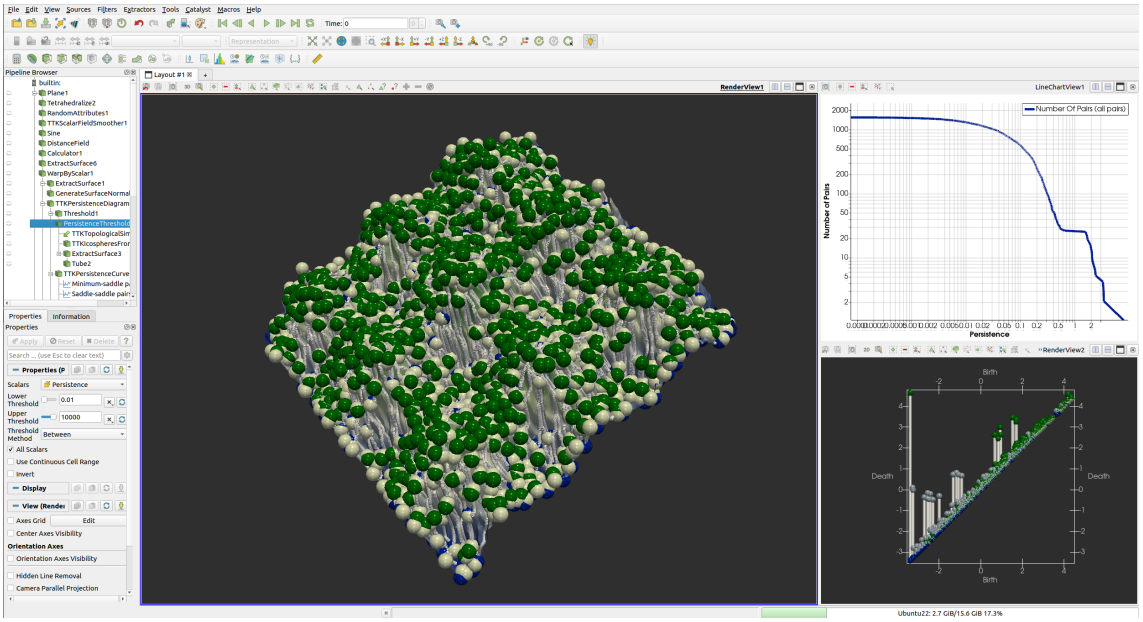

Figura 4.3: Situación de la demostración tras modificar el ”PersistenceThreshold”a 0.01

En la esquina inferior derecha tenemos el diagrama de persistencia de los datos, es un diagrama 2D que muestra la distribución de pares de puntos críticos en los datos. En este gráfico cada par esta representado por una barra vertical, la coordenada X de esta barra cooresponde al valor de la función del punto crítico que creó el par, mientras que la coordenada Y del punto equivale al valor escalar de del punto crítico que destruyó el par. La codificación por colores corresponde a esos puntos críticos del diagrama.

Lo interesante de esta demostración es que puedes hacer click en el apartado de ”PersistenceThreshold” o umbral de persistencia del pipeline que aparece en el ”Pipeline Browser” en la esquina superior izquierda y podemos modificar el umbral de persistencia con el que podemos realizar alguna simplificación lógica. En nuestro caso los datos ya han sido simplificados ya que tenemos muchas protuberancias suaves en el diagrama y podemos deducir que debe haber algunas características topológicas. Para mostrar como sería el modificar el ”PersistenceThreshold” voy a cambiar su valor de 0.7 que tenía inicialmente la demonstración por 0.01 (ver Figura4.3).

Lo que sucede ahora es que, incluso para las protuberancias más leves, aparecen puntos críticos y parece muy desordenado el layout, el diagrama de persistencia desaparece debido a que hay muchas barras muy pequeñas cerca de la diagonal. En la práctica queremos dar un valor al umbral de persistencia debemos fijarnos en la curva de persistencia (arriba derecha), en este caso podemos ver los valores que hay en la ”meseta” de la curva, como el valor para el eje X es de entorno a 0.67 ponemos como mínimo del umbral de persistencia el valor 0.07 que teníamos en un inicio.

Si nos fijamos bien en el ejemplo podemos ver que la colina del centro es mucho

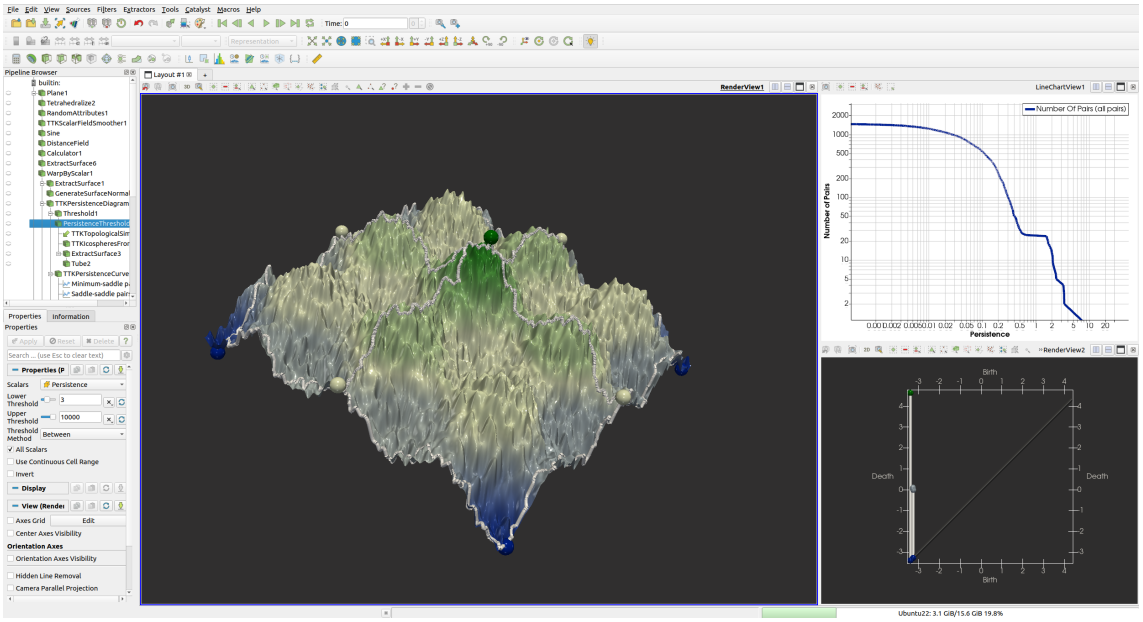

Figura 4.4: Layout con un valor mínimo de 3 en el umbral de persistencia.

mayor al resto por lo que podemos ir más allá en la simplificación y darle un valor de 3 al mínimo del umbral de persistencia para obtener un layout como el siguiente (Figura 4.4), podemos ver que tan solo se muestra la bola verde de la colina del centro. Esta técnica es muy útil cuando hacemos segmentación de datos. Para demostrarlo hay que seguir los siguientes pasos:

- Dejamos el mínimo del umbral de persistencia en 0.7
- En el ”Pipeline Browser” nos desplazamos abajo donde pone ”TTKTopologicalSimplification1”, lo extendemos, clicamos ”segmentation” y marcamos las casillas de ”Ascending Segmentation”, ”Descending Segmentation” y ”Morse-Smale Complex Segmentation”.
- En ”GenerateSurfaceNormals2” en el apartado de ”Coloring” elegimos la opción ”MorseSmale Manifold” y en la opción ”edit” debajo de esta escogemos la opción ”Rainbow Uniform”

Esto nos permite ver una representación del complejo Morse del dataset (ver Figura4.5), lo que significa que para cualquier punto ubicado en la misma parcela, si integras hacia arriba en pendiente ascendente o integras hacia abajo en pendiente descendente terminaras en el mismo par de puntos críticos, en el mismo máximo y mínimo. Sin embargo, si vas al otro lado de estos parámetros, integrarás cuesta arriba a diferentes máximos, mientras sigues integrando cuesta abajo al mismo mínimo. El complejo Morse divide nuestros datos en celdas donde la integración hacia adelante y hacia atrás termina en las mismas extremidades. Esto es útil en varios escenarios, en astrofísica utilizan las sendas que conectan las monturas a los máximos para extraer estructuras de filamentos en sus simulaciones, a las que llaman la red cósmica, y tener la posibilidad de observar varias escalas de importancia es muy interesante para fines de análisis de datos.

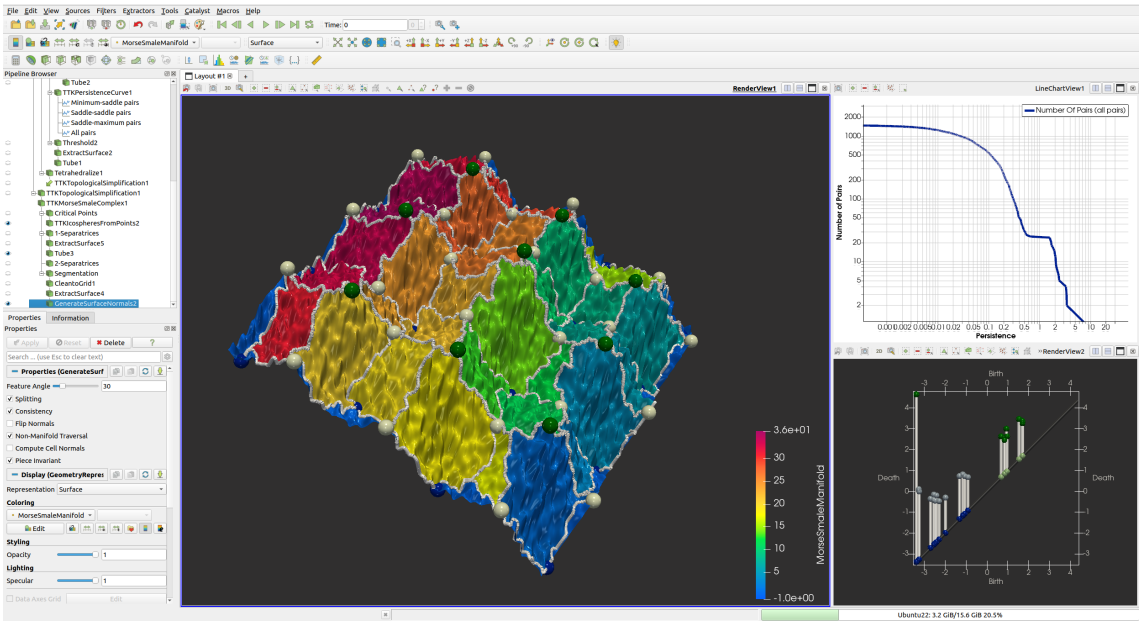

Figura 4.5: Representación de la segmentación de datos siguiendo los pasos mencionados.

4.1.2. Salidas de TTK

En el primer anexo se muestra el código Python necesario para llevar a cabo el análisis de los datos utilizando la biblioteca Topology ToolKit (TTK) y ParaView:

En cuanto a los outputs del código tenemos 5 como podemos ver en los últimas líneas de código y son los siguientes:

- PersistenceDiagram.vtu: el diagrama de persistencia de salida en formato de archivo VTK.
- PersistenceCurve.csv: la curva de persistencia de salida.
- MorseComplexeCriticalPoints.vtp: los puntos críticos de salida (o elementos de 0 dimensiones) del complejo de Morse Smale en formato de archivo VTK.
- MorseComplexe1Separatrices.vtp: cilindros, representando los bordes (o elementos de 1 dimensión) del complejo de Morse Smale de salida en formato de archivo VTK.
- MorseComplexeSegmentation.vtp: superficies, representando la segmentación del complejo de Morse Smale de salida en formato de archivo VTK (vista central, arriba en la captura de pantalla).

Como aclaración, eres libre de cambiar la extensión de archivo vtu a la de cualquier otro formato de archivo compatible (por ejemplo, csv) en el script de Python para todos los outputs.

4.2. KeplerMapper

Esta biblioteca, Kepler Mapper, es una implementación en Python[75] del algoritmo Mapper tal como se describe por primera vez en el artículo "Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition"[15]. Kepler Mapper presenta una interfaz intuitiva para el algoritmo Mapper y proporciona múltiples métodos de comprensión para visualizar el grafo de red que Mapper produce. Aprovecha algoritmos de agrupamiento y escalado compatibles con la API de Scikit-Learn[80] para construir grafos de red de manera flexible y fácil de usar. También ofrece una extensa suite de tutoriales que detallan el uso de Kepler Mapper para casos de uso simples y complejos.

Mapper es una forma de construir un grafo (o complejo simplicial) a partir de datos de manera que revele algunas de las características topológicas del espacio. Aunque no es exacto, Mapper es capaz de estimar aspectos de conectividad importantes del espacio subyacente de los datos para que pueda ser explorado en un formato visual. Este es un método no supervisado para generar una representación visual de los datos que a menudo puede revelar nuevas ideas de los datos que otros métodos no pueden.

Informalmente, el algoritmo Mapper funciona realizando un agrupamiento local guiado por una función de proyección. Los pasos son los siguientes:

1. Proyecta un conjunto de datos (por ejemplo: la media de x para x en X, o `t-SNE(2d).fit_transform(X)`). Puedes usar cualquier función de proyección de matemáticas, estadísticas, econometría o machine learning.
2. Cubre esta proyección con intervalos/híper-cubos superpuestos. En teoría, puedes usar cualquier forma de intervalo, pero KeplerMapper actualmente soporta híper-cubos n-dimensionales.
3. Agrupa los puntos dentro de un intervalo (ya sea aplicando agrupamiento en la proyección y sufriendo pérdida de proyección, o agrupando en la imagen inversa/datos originales). Puedes usar cualquier algoritmo de agrupamiento (jerárquico, basado en densidad, etc.) y métrica de distancia (no necesita ser una métrica adecuada que satisfaga la desigualdad triangular).
4. Los clusters se convierten en nodos en un grafo.
5. Debido a la superposición, un solo punto puede aparecer en múltiples nodos. Cuando hay una intersección de miembros de este tipo, dibuja un borde entre estos nodos.
6. Ahora, para ayudar en la exploración visual:
    - Poner un tamaño de los nodos del grafo según una función de interés (por ejemplo: número de miembros dentro del grupo).
    - Colorea los nodos del grafo por una función de interés (por ejemplo: gasto promedio del cliente).
    - Forma los nodos del grafo por una función de interés (por ejemplo: cuadrados para avg_timestamp <2015, círculos para avg_timestamp

>= 2015).

- Tamaño de los bordes del grafo por una función de interés (por ejemplo: número de miembros intersectantes entre nodos).
- Colorea los bordes del grafo por una función de interés (por ejemplo: color promedio de nodos conectados).
- Forma los bordes del grafo por una función de interés (por ejemplo: línea punteada cuando la intersección de miembros <3, de lo contrario, línea sólida).
- Proporciona estadísticas descriptivas sobre los nodos y el grafo (por ejemplo: prueba de Kolmogorov-Smirnov entre variables de nodo y variables de conjunto de datos).

Para instalar la librería es muy simple pero previamente necesitamos instalar una serie de dependencias:

- Python (>= 2.7 o >= 3.3)
- NumPy
- Scikit-learn
- matplotlib
- bokeh
- PIL
- Roboto Webfont (Google)
- D3.js (Mike Bostock)

Una vez tenemos estas dependencias instaladas tenemos que instalar Keppler-Mapper con el siguiente comando:

```
pip install kmapper
```

Con esto ya podríamos empezar a usar la herramienta, al igual que con el Topology ToolKit[59] mostraré un ejemplo ya construido que se puede encontrar en su página oficial. Para esto vamos a usar el ejemplo llamado ”Breast Cancer” para el cual primero hemos de decargar el dataset que se puede encontrar aquí [81], tenemos que almacenarlo en una carpeta llamada ”data”.

Para la demostración debemos comprender el término lente o ”lens” en inglés, el cual en el contexto del TDA se refiere a una función aplicada a los datos con la finalidad de crear una representación simplificada. Estas lentes proyectan datos de alta dimensión en espacios de menor dimensión, de esta manera se facilita el análisis y visualización de los datos.

La razón detrás de la elección de lentes en la demostración a continuación es:

- **Lens1**: Lentes que tienen sentido biológico; en otras palabras, lentes que resaltan características especiales en los datos. Se crea utilizando el algoritmo Isolation Forest, el cual es adecuado para detectar anomalías.
- **Lens2**: Lentes que dispersan los datos, en contraposición a agrupar muchos puntos juntos.

En el caso de estos datos particulares, el uso de un puntaje de anomalía (en este caso, calculado usando IsolationForest de sklearn) tiene sentido biológico ya que las células cancerosas son anómalas. Para el segundo lente, usamos la norma l^2 ((distancia euclidiana) esto dispersa los datos en lugar de agrupar muchos puntos juntos

Este software también permite establecer múltiples funciones de color para puntos de datos y funciones de color para nodos en sus visualizaciones html. El código de ejemplo que se encuentra en el segundo anexo de demuestra tres formas en que esto podría hacerse.

Al ejecutar dicho código en nuestra máquina debemos obtener este output:

```
KeplerMapper(verbose=3)
..Composing projection pipeline of length 1:
        Projections: l2norm
        Distance matrices: False
        Scalers: MinMaxScaler()
..Projecting on data shaped (569, 31)

..Projecting data using: l2norm

..Scaling with: MinMaxScaler()

Mapping on data shaped (569, 31) using lens shaped (569, 2)

Minimal points in hypercube before clustering: 2
Creating 225 hypercubes.
Cube_0 is empty.
Cube_1 is empty.
Cube_2 is empty.
Cube_3 is empty.
Cube_4 is empty.
Cube_5 is empty.
Cube_6 is empty.
Cube_7 is empty.
Cube_8 is empty.
Cube_9 is empty.
Cube_10 is empty.
Cube_11 is empty.
Cube_12 is empty.
Cube_13 is empty.
   > Found 2 clusters in hypercube 14.
```

```
Cube_15 is empty.
Cube_16 is empty.
Cube_17 is empty.
Cube_18 is empty.
   > Found 2 clusters in hypercube 19.
Cube_20 is empty.
   > Found 2 clusters in hypercube 21.
Cube_22 is empty.
Cube_23 is empty.
Cube_24 is empty.
   > Found 2 clusters in hypercube 25.
   > Found 2 clusters in hypercube 26.
   > Found 2 clusters in hypercube 27.
   > Found 2 clusters in hypercube 28.
Cube_29 is empty.
Cube_30 is empty.
Cube_31 is empty.
   > Found 2 clusters in hypercube 32.
Cube_33 is empty.
   > Found 2 clusters in hypercube 34.
Cube_35 is empty.
Cube_36 is empty.
   > Found 2 clusters in hypercube 37.
   > Found 2 clusters in hypercube 38.
   > Found 2 clusters in hypercube 39.
   > Found 2 clusters in hypercube 40.
Cube_41 is empty.
   > Found 2 clusters in hypercube 42.
   > Found 2 clusters in hypercube 43.
   > Found 2 clusters in hypercube 44.
Cube_45 is empty.
Cube_46 is empty.
   > Found 2 clusters in hypercube 47.
Cube_48 is empty.
   > Found 2 clusters in hypercube 49.
   > Found 2 clusters in hypercube 50.
   > Found 2 clusters in hypercube 51.
   > Found 2 clusters in hypercube 52.
   > Found 2 clusters in hypercube 53.
   > Found 2 clusters in hypercube 54.
   > Found 2 clusters in hypercube 55.
   > Found 2 clusters in hypercube 56.
   > Found 2 clusters in hypercube 57.
   > Found 2 clusters in hypercube 58.
   > Found 2 clusters in hypercube 59.
   > Found 2 clusters in hypercube 60.
   > Found 2 clusters in hypercube 61.
```

```
   > Found 2 clusters in hypercube 62.
   > Found 2 clusters in hypercube 63.
   > Found 2 clusters in hypercube 64.
   > Found 2 clusters in hypercube 65.
   > Found 2 clusters in hypercube 66.
   > Found 2 clusters in hypercube 67.
   > Found 2 clusters in hypercube 68.
   > Found 2 clusters in hypercube 69.
   > Found 2 clusters in hypercube 70.
Cube_71 is empty.
   > Found 2 clusters in hypercube 72.
   > Found 2 clusters in hypercube 73.
   > Found 2 clusters in hypercube 74.
   > Found 2 clusters in hypercube 75.
   > Found 2 clusters in hypercube 76.
   > Found 2 clusters in hypercube 77.
   > Found 2 clusters in hypercube 78.
   > Found 2 clusters in hypercube 79.
   > Found 2 clusters in hypercube 80.
   > Found 2 clusters in hypercube 81.
   > Found 2 clusters in hypercube 82.
   > Found 2 clusters in hypercube 83.
   > Found 2 clusters in hypercube 84.
   > Found 2 clusters in hypercube 85.
   > Found 2 clusters in hypercube 86.
   > Found 2 clusters in hypercube 87.
   > Found 2 clusters in hypercube 88.
   > Found 2 clusters in hypercube 89.
   > Found 2 clusters in hypercube 90.
   > Found 2 clusters in hypercube 91.
   > Found 2 clusters in hypercube 92.
   > Found 2 clusters in hypercube 93.
   > Found 2 clusters in hypercube 94.
   > Found 2 clusters in hypercube 95.
Cube_96 is empty.
   > Found 2 clusters in hypercube 97.
   > Found 2 clusters in hypercube 98.
   > Found 2 clusters in hypercube 99.
   > Found 2 clusters in hypercube 100.
   > Found 2 clusters in hypercube 101.
   > Found 2 clusters in hypercube 102.
   > Found 2 clusters in hypercube 103.
   > Found 2 clusters in hypercube 104.
   > Found 2 clusters in hypercube 105.
   > Found 2 clusters in hypercube 106.
   > Found 2 clusters in hypercube 107.
   > Found 2 clusters in hypercube 108.
```

```
Cube_109 is empty.
   > Found 2 clusters in hypercube 110.
   > Found 2 clusters in hypercube 111.
   > Found 2 clusters in hypercube 112.
   > Found 2 clusters in hypercube 113.

Created 304 edges and 158 nodes in 0:00:00.672039.
Wrote visualization to: output/breast-cancer.html
Wrote visualization to: output/breast-cancer-multiple
-color-functions.html
Wrote visualization to: output/breast-cancer-multiple-
node-color-functions.html
Wrote visualization to: output/breast-cancer-multiple
-color-functions-and-multiple-node-color-functions.html
no display found. Using non-interactive Agg backend
```

Este código carga un conjunto de datos de cáncer de mama de Wisconsin[82] y lo preprocesa para eliminar columnas no deseadas y convertir la variable objetivo en valores numéricos. Luego, crea las lentes unidimensionales personalizadas que se combinan para formar una representación bidimensional del conjunto de datos. Utilizando KeplerMapper, se construye un complejo simplicial utilizando las lentes creadas, especificando un método de cobertura y un algoritmo de agrupamiento. Se generan visualizaciones del complejo simplicial con diferentes funciones de color para los puntos de datos y los nodos, así como visualizaciones adicionales con múltiples funciones de color para los nodos y los puntos de datos. Finalmente, se muestra una visualización final utilizando Matplotlib para representar el grafo del complejo simplicial.

Después de adentrarnos en el código y entender cómo las lentes personalizadas, como lens1 y lens2, transforman los datos complejos en representaciones más manejables, podemos ver cómo estas abstracciones numéricas se convierten en visualizaciones vívidas gracias a KeplerMapper. Estas visualizaciones no son simples gráficos; representan el resultado de un proceso analítico complejo que revela la estructura oculta dentro de nuestros datos. Al explorar los mapas topológicos, empezamos a descubrir patrones y relaciones que antes pasaban desapercibidos. Cada nodo y conexión en estos mapas cuenta una historia sobre la proximidad y la relación entre los datos, ofreciéndonos una perspectiva única que va más allá de los números y las ecuaciones, y nos brinda una comprensión más profunda de los datos que estamos analizando.

Previo a explicar las visualizaciones cabe destacar que en la parte superior izquierda de estas vamos a poder observer tres botones interactivos:

1. Cluster details: Esta sección nos proporciona detalles sobre los clústeres individuales que encontramos en la visualización. De arriba abajo podemos distinguir:

    a) Actions: Esto nos permite centrar la vista en el nodo que hemos seleccionado.

b) Cluster Details: Muestra el id del nodo.

c) Member Distribution: Muestra la distribución de los miembros dentro de un clúster específico. Proporciona un histograma que representa cómo se distribuyen los miembros del clúster a través de diferentes intervalos o categorías. Esto nos puede ayudar a entender la densidad y la dispersión de los puntos dentro del clúster.

d) Cluster Statistics Size: Representa el tamaño del clúster, normalmente es el número de puntos o miembros que contiene. Esta estadística es importante ya que puede indicar la prominencia de ciertas características dentro del conjunto de datos, un clúster más grande puede significar una tendencia más común o un grupo más significativo dentro de los datos.

e) Members: Lista de los miembros individuales del clúster. Cada miembro puede ser un punto de datos o una entidad dentro del conjunto de datos que ha sido agrupado en ese clúster particular. Esta lista permite examinar y explorar los elementos específicos que componen el clúster, lo cual es útil para realizar análisis detallados y para entender qué datos están contribuyendo a las tendencias observadas en la visualización.

2. Mapper Summary: Es un resumen de toda la visualización. Está compuesto por varios elementos:

a) PROJECTION: Indica la lente utilizada para proyectar los datos.

b) N_CUBES: Indica el número de hiper-cubos o intervalos que se utilizan para dividir el rango de la proyección. Un número mayor de cubos puede resultar en una visualización más detallada, pero también más compleja.

c) PERC_OVERLAP: Muestra el porcentaje de solapamiento entre los hiper-cubos adyacentes. Un solapamiento mayor permite capturar más conexiones entre los datos, pero también puede aumentar la redundancia.

d) CLUSTERER: Es el algoritmo de agrupamiento utilizado para identificar clústeres dentro de cada hiper-cubo.

e) SCALER: Se refiere al método de escalado aplicado a los datos antes de la proyección. El escalado puede mejorar la calidad de la proyección al normalizar las características de los datos.

f) NERVE_MIN_INTERSECTION: Es el mínimo número de intersecciones requerido para conectar clústeres en la visualización final. Esto afecta cómo se construye el "nervio" del complejo simplicial.

g) NODES: El número total de nodos en la visualización, cada nodo representa un clúster encontrado en los datos.

h) EDGES: El número total de aristas en la visualización, que conectan nodos si sus clústeres correspondientes se solapan significativamente.

i) TOTAL SAMPLES: El número total de muestras o puntos de datos que se han utilizado para crear la visualización.

j) UNIQUE SAMPLES: El número de muestras únicas presentes en la visualización, lo que puede ser menor que el total de muestras debido a la superposición en los clústeres.

k) COLOR FUNCTIONS: Las funciones de color utilizadas para colorear los nodos, lo que puede ayudar a identificar características interesantes como gradientes o densidades dentro de los datos.

l) NODE COLOR FUNCTIONS: Funciones específicas utilizadas para determinar el color de cada nodo individualmente, basadas en las características de los datos dentro de ese clúster.

m) NODE DISTRIBUTION: La distribución de los nodos a través de la visualización, lo que puede dar una idea de la distribución de los clústeres y la densidad de los datos.

3. Help: Proporciona instrucciones sobre cómo interactuar con la visualización.

A continuación se mostrarán varias visualizaciones del mapeador de cáncer de mama utilizando múltiples funciones de color para:

1. Puntos de datos.
2. Nodos.
3. Puntos de datos y nodos.

La distribución de nodos va a ser la misma para las tres visualizaciones (ver Figura 4.6) lo que varían son los ”cluster details” y el ”Mapper Summary”.

4.2.1. Visualización para los puntos de datos.

Esta la visualización (Figura 4.6) es como un mapa detallado que nos muestra la ubicación exacta de cada punto de datos según las características seleccionadas. Imagina que cada punto de datos es una persona en una clase, y el color de su camiseta representa algo único sobre ella, como su bebida favorita o el género de música que prefiere.

Al usar diferentes colores para representar distintas métricas, esta visualización nos permite ver rápidamente si hay patrones o si ciertas características tienden a agruparse. Por ejemplo, podríamos descubrir que la mayoría de los puntos de datos con alta expresión de un gen particular están todos en una esquina del gráfico, lo que sugiere una relación entre esa expresión génica y otras características de los datos.

En esta representación podemos cambiar el valor de ”Color Function” entre ”Isolation Forest” y ”L2-norm”, esto nos va a permitir:

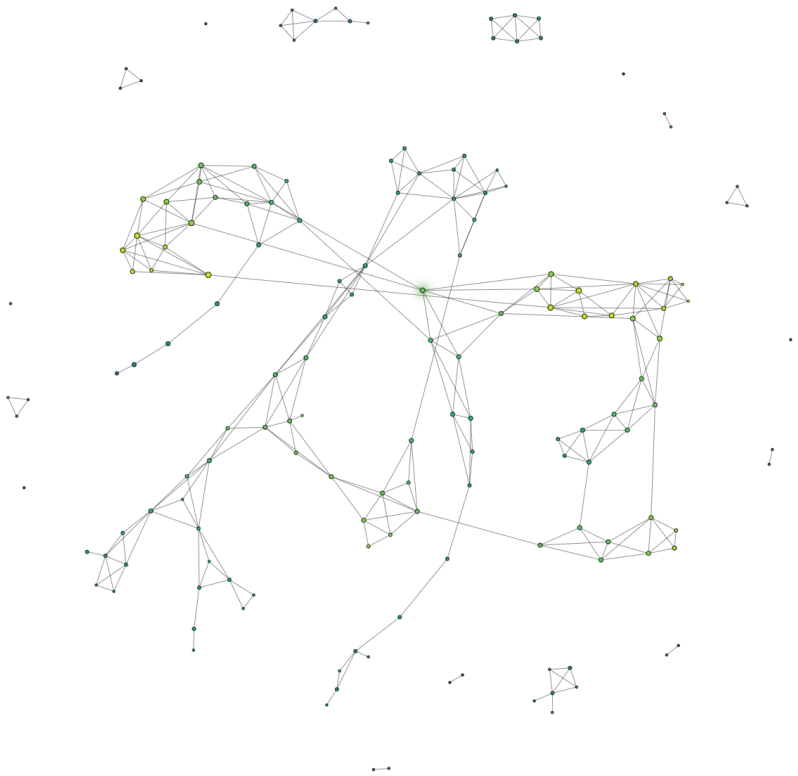

Figura 4.6: Proyección de datos del dataset de cáncer de mama de Wisconsin.

- En el caso de elegir la lente "Isolation Forest", esta una función que se basa en un algoritmo de detección de anomalías, que se van a destacar en la visualización. Esto nos puede permitir ver los puntos que se devían significativamente del resto, lo cual nos será útil para detectar posibles errores o para centrar estudios en anomalías. Si seleccionamos un nodo sobre la proyección de la Figura 4.6 podremos ver los siguientes datos en los apartados "Cluster details" (ver Figura 4.7) y en "Mapper summary" (ver Figura 4.8), de esta manera podremos estudiar cada nodo por separado.

- En el caso de elegir la lente "L2-norm", esta función mide la distanca de cada punto de datos al origen, utilizarala supone que resaltará la magnitud de los puntos de datos en el espacio de características. Nos permite ver cómo se distribuyen los puntos de datos en términos de su distancia al origen, lo que puede ser indicativo de la varianza o la concentración de los datos.Si seleccionamos un nodo sobre la proyección de la Figura 4.6 podremos ver los siguientes datos en los apartados "Cluster details" (ver Figura 4.9) y en "Mapper summary" (ver Figura 4.10), de esta manera podremos estudiar cada nodo por separado.

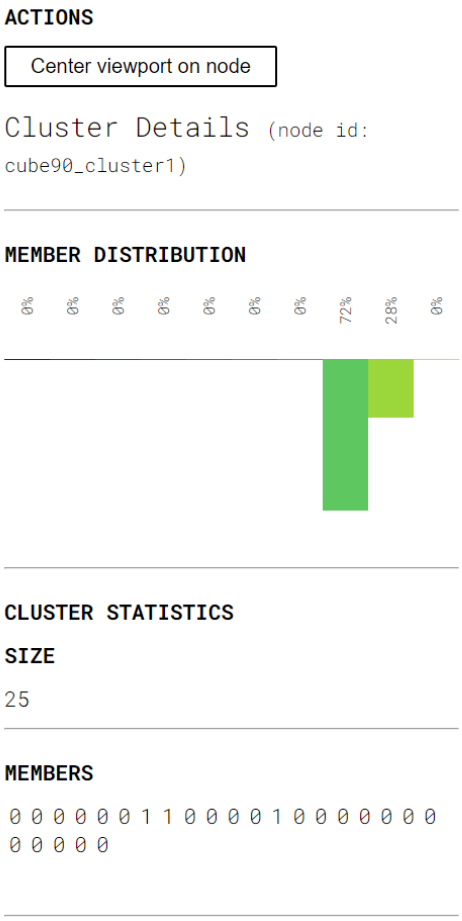

Figura 4.7: Cluster details de "output/breast-cancer-multiple-color-functions.html" con la lente "Isolation Forest".

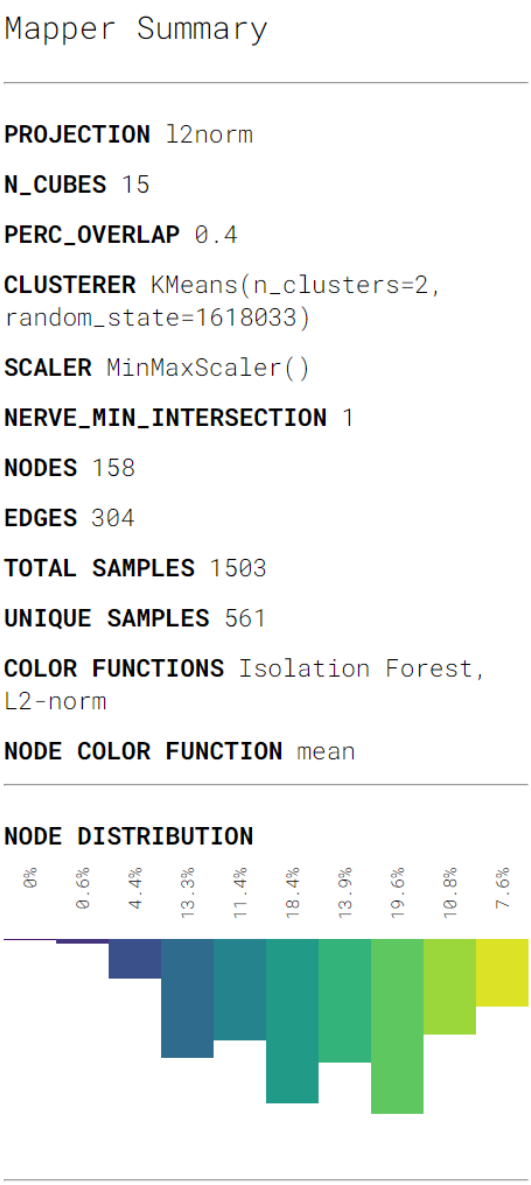

Figura 4.8: Mapper summary de "output/breast-cancer-multiple-color-functions.html" con la lente "Isolation Forest".

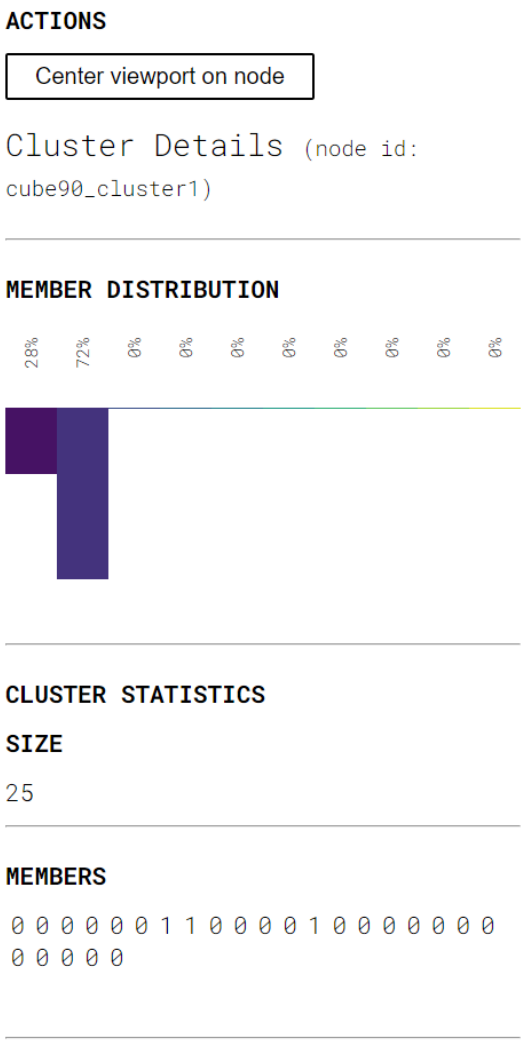

Figura 4.9: Cluster details de "output/breast-cancer-multiple-color-functions.html" con la lente "L2-norm".

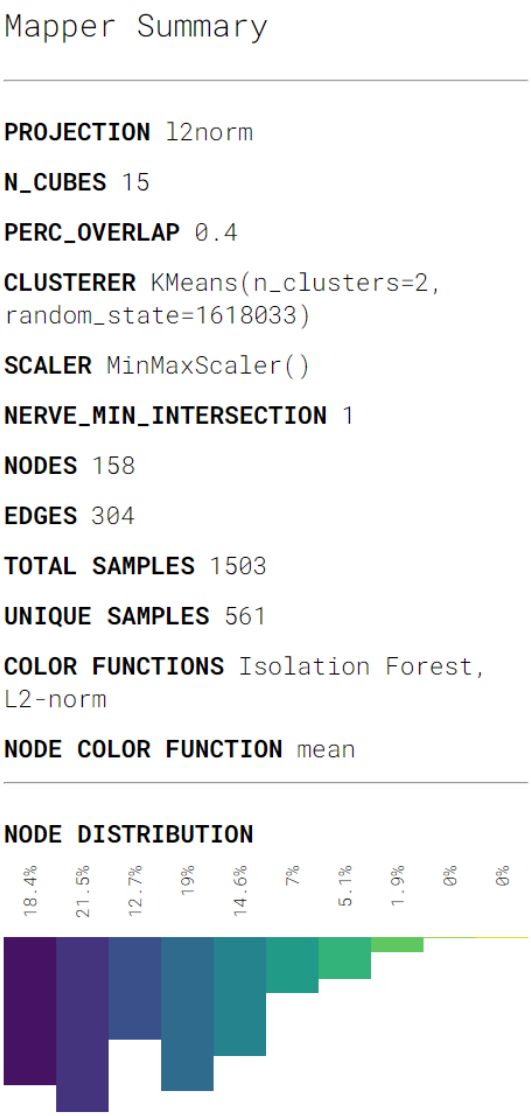

Figura 4.10: Mapper summary de "output/breast-cancer-multiple-color-functions.html" con la lente "L2-norm".

4.2.2. Visualización para los nodos.

En esta visualización (Figura 4.6) cada nodo es como un resumen de un grupo de personas en la clase. El color del nodo podría decirnos la bebida más popular entre ese grupo o el género musical predominante. Esto nos da una idea general de lo que está pasando en diferentes áreas del gráfico.

Los colores de los nodos pueden indicar la media de una métrica específica dentro de ese clúster, la desviación estándar, o incluso la presencia de valores extremos. Esto es útil para identificar áreas de interés o preocupación, como un nodo que representa un grupo de pacientes con características de alto riesgo.

En esta representación podemos modificar el valor de ”node color function” entre mean, std, median o max, sin embargo al no poder cambiar la lente el campo de ”cluster details” no se va a ver afectado por ello :

- Mean o Media: Es el promedio de los valore dentro de cada nodo, nos puede ayudar a identificar los nodos que tienen un valor promedio alto o bajo una característica específica, esto puede ser indicativo de un patrón o una tendencia general en el conjunto de datos (ver Figura 4.11)
- Std o Desviación estándard: Mide la cantidad de variación o disperción de los valores dentro de un nodo, esto puede resaltar los nodos con alta variabilidad, lo que podría sugerir heterogeneidad en los datos o la presencia de subgrupos (ver Figura 4.12).
- Median o Mediana: Es el valor medio que divide los valores dentro de un nodo en dos mitades iguales, puede ser útil para entender la distribución de los datos, especialmente en casos donde la media no es representativa debido a la presencia de valores atípicos (ver Figura 4.13).
- Max o Máximo: Es el valor más alto dentro de un nodo, esto puede ayudar a identificar los nodos que contienen valores extremos, lo que puede ser relevante para estudios de casos máximos o para la identificación de posibles valores atípicos (ver Figura 4.14).

4.2.3. Visualización para los puntos de datos y los nodos.

Es visualización (Figura 4.6) es la clase en todo su esplendor, con información detallada sobre cada alumno y resúmenes de los grupos al mismo tiempo. Es una combinación poderosa porque nos permite ver no solo cómo se agrupan los datos, sino también cómo las características individuales se reflejan en el nivel del grupo.

Por ejemplo, si un nodo tiene un color que indica un alto riesgo y los puntos de datos dentro de ese nodo también tienen colores que representan características de alto riesgo, tenemos una confirmación visual fuerte de que ese clúster es significativo y merece una atención especial.

En este caso podemos modificar ambas funciones, lo cual nos va a permitir obtener una representación más rica y detallada del conjunto de datos, comprender

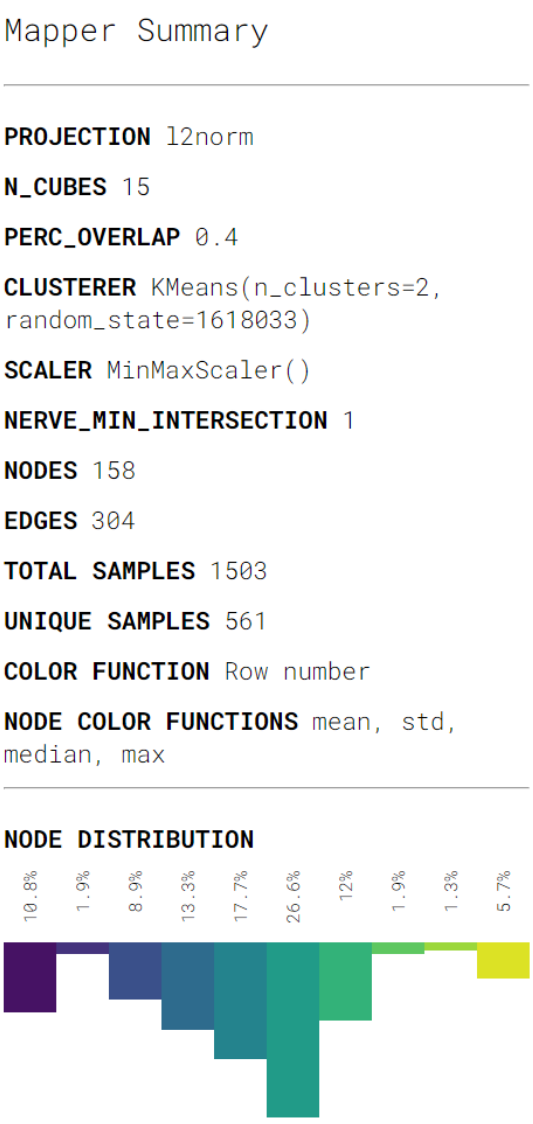

Figura 4.11: Visualización con ”mean” como ”node color function”.

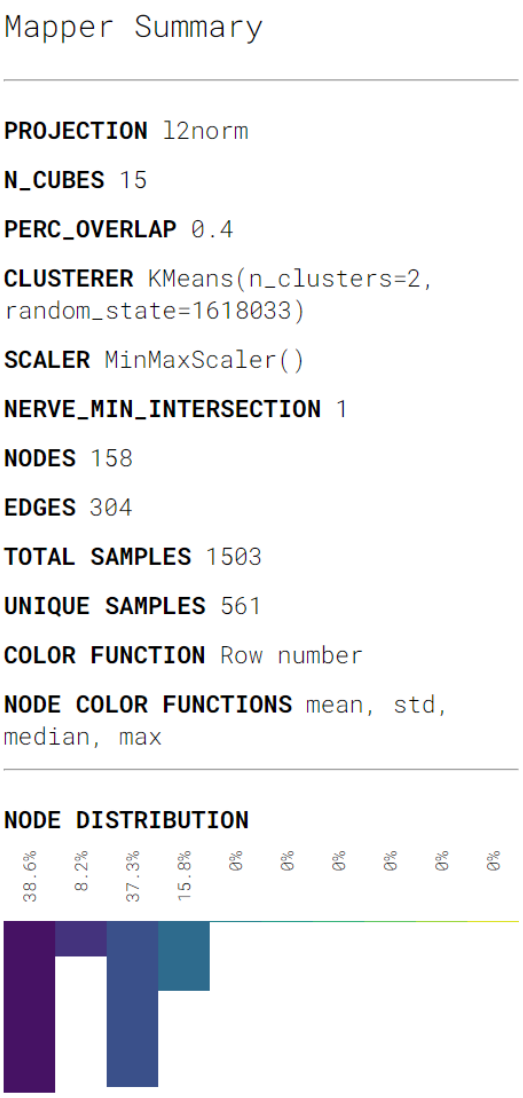

Figura 4.12: Visualización con ”std” como ”node color function”.

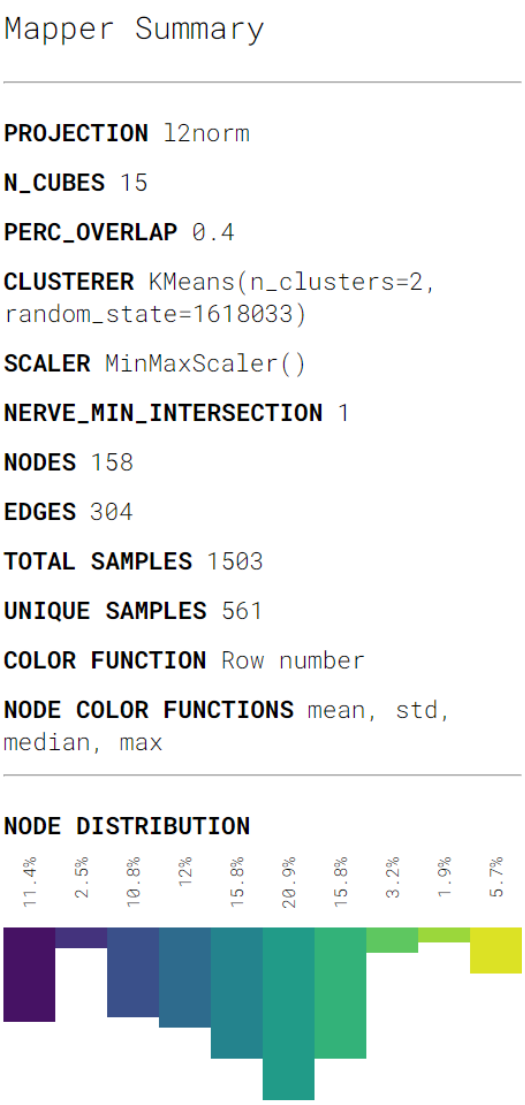

Figura 4.13: Visualización con ”median” como ”node color function”.

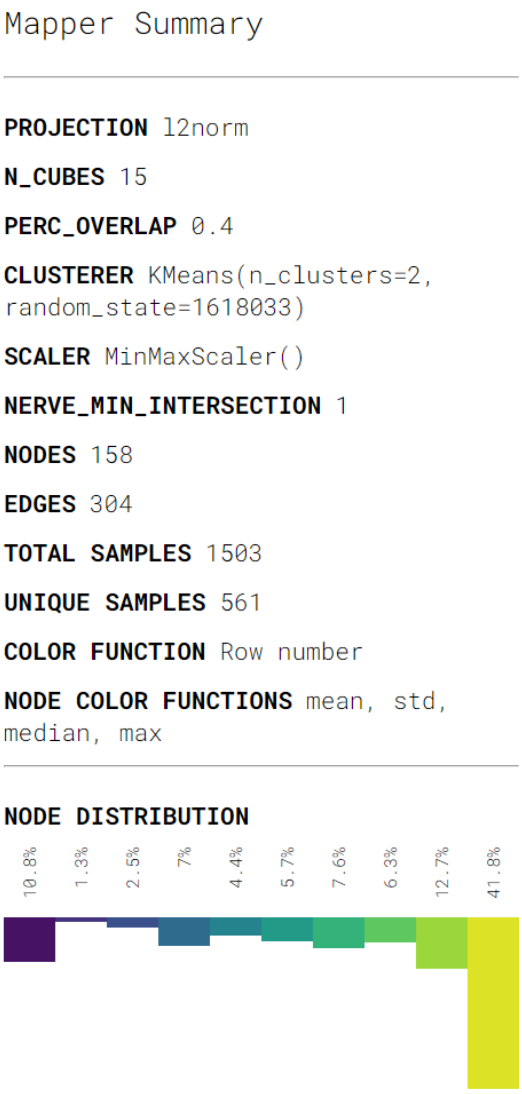

Figura 4.14: Visualización con ”max” como ”node color function”.

más profundamente la estructura de los datos, identificar patrones, tendencias y relaciones significativas.

4.3. Perseus

Este software esta desarrollado sobre C++[58] por Vidit Nanda[16] calcula la homología persistente de muchos tipos diferentes de complejos celulares filtrados después de realizar ciertas reducciones Morse teóricas que preservan la homología. En todos los casos, los usuarios debemos preparar la filtración de entrada como un archivo de texto con el formato correcto y luego leer los intervalos de homología persistente de salida, nuevamente presentados como archivos de texto.

Dado que Perseus se basa en la teoría discreta de Morse, no depende de las idiosincrasias de un tipo particular de estructura compleja o dimensión para su eficiencia. Dicho esto, ciertos tipos de complejos son mucho más comunes que otros en el ámbito del análisis de datos. Para lidiar con películas e imágenes, es mejor trabajar con estructuras de datos cúbicas. Por otro lado, si la fuente de datos es una triangulación de una variedad, entonces la representación apropiada consiste en información de celda superior en un complejo simplicial. Los datos de nube de puntos generalmente se manejan de manera efectiva con complejos de Vietoris-Rips construidos alrededor de esos puntos.

Para calcular la homología persistente de nuestro complejo filtrado de entrada, deberemos ingresar los siguientes argumentos de línea de comandos (sin paréntesis) en el orden dado, ya sea directamente en un terminal o utilizando nuestro entorno de desarrollo integrado.

(ruta al ejecutable de Perseus) (tipo de complejo) (nombre de archivo de entrada) (cadena de archivo de salida)

La (cadena de archivo de salida) es completamente opcional, simplemente proporciona un prefijo para los archivos de salida para que sean fáciles de ubicar e identificar. Por ejemplo, si estamos trabajando en el proyecto Madrid, hacer que la cadena de salida sea Madrid produce archivos de salida que tienen un prefijo Madrid seguido de otros elementos.

La salida principal de Perseus es una colección de archivos de texto que contienen intervalos de persistencia para cada dimensión. Estos archivos se nombran según la (cadena de archivo de salida) ingresada en la línea de comando al llamar al software. Debemos tener en cuenta que es posible incluir una ruta de archivo completa en la cadena de archivo de salida. Por ejemplo, si usa:

```
"C:\usuarios\miNombre\miSalida\resultado"
```

en Windows o

```
"/home/MiNombre/miSalida/resultado"
```

en sistemas tipo Mac o Unix, entonces los archivos de salida se crearán en los directorios correspondientes `miSalida` y todos tendrán nombres de archivo

precedidos por `resultado`. Por supuesto, es importante que tengamos los privilegios de escritura apropiados en el directorio de salida deseado. En caso de no usar ninguna cadena de archivo de salida, la elección predeterminada es `output` y los archivos de salida se crearán con este prefijo en el directorio de trabajo actual. Debemos tener en cuenta que si ya existe un archivo con ese nombre en este directorio, será sobrescrito.

Suponiendo que estamos utilizando la cadena predeterminada `output`, los archivos de salida creados se llamarán output_0.txt, output_1.txt,... y así sucesivamente. La cantidad de archivos creados depende de cuántas dimensiones tenga realmente el complejo reducido de Morse discreto; esto siempre es menor o igual a la dimensión del complejo original que se ha utilizado como entrada. Es posible que la versión reducida de un complejo tridimensional sea solo bidimensional, en cuyo caso no se creará output_3.txt.

Cada archivo del tipo output_n.txt contiene los intervalos de persistencia correspondientes a generadores n-dimensionales de homología. Estos intervalos están dispuestos en dos columnas de enteros, donde la primera entrada indica el tiempo de nacimiento y la segunda entrada indica la muerte. Tenga en cuenta que un tiempo de muerte de -1 corresponde a persistencia infinita, lo que significa que el ciclo generador correspondiente persistió más allá del final de la filtración y no se llenó como un límite incluso cuando se agregaron todas las celdas del complejo subyacente.

Finalmente, el archivo llamado "output_betti.txt" contiene los números de Betti en cada paso de la filtración. Una línea típica de este archivo se ve así:

```
12 14 4 7 0
```

lo cual indica que cuando se incluyen todas las celdas con tiempo de nacimiento menor o igual a 12, entonces hay 14 componentes conectadas, 4 túneles, 7 cavidades y ningún generador de homología de dimensiones superiores. Los números 14, 4 y 7 en este contexto se llaman los números de Betti cero, uno y dos del subcomplejo número 12 en la filtración de persistencia.

Para ayudar en la visualización, se incluye un sencillo script de Matlab llamado `persdia` junto con el código fuente de Perseus. Este script se puede llamar desde el prompt de comandos de Matlab para graficar el archivo de salida de Perseus como un diagrama de persistencia de la siguiente manera:

```
>> persdia('output_1.txt');
```

Por supuesto, puede que necesitemos cambiar el argumento de cadena `'output_1.txt'` para apuntar a la ruta donde se almacenan los archivos de salida de Perseus en nuestro ordenador. Un ejemplo de un diagrama de persistencia creado por `persdia` se vería así (ver Figura 4.15).

Cada gráfica generada por el script `persdia` representa un diagrama de persistencia. Las características principales de estas gráficas incluyen:

- **Puntos azules:** Cada punto azul en la gráfica corresponde a una característica topológica (ciclo) que se detecta en el complejo filtrado.

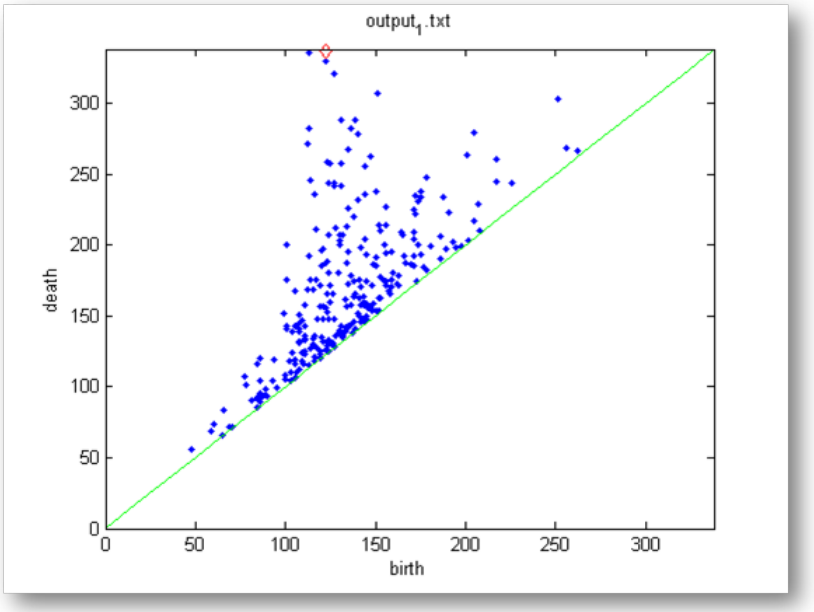

Figura 4.15: Ejemplo de diagrama de persistencia creado por persdia.

  - El **eje** x (birth) indica el tiempo de nacimiento del ciclo.
  - El **eje** y (death) indica el tiempo de muerte del ciclo.
- **Línea diagonal verde** $y = x$**:** Los puntos que caen sobre esta línea diagonal representan ciclos cuyos tiempos de nacimiento y muerte son muy cercanos o iguales, considerados como insignificantes o ruido. Los puntos por encima de la diagonal representan ciclos significativos que persisten durante un rango de filtración más largo. La distancia vertical entre el punto y la línea diagonal indica la duración de la persistencia del ciclo.
- **Puntos rojos:** Los puntos rojos representan ciclos que persisten indefinidamente (mueren en -1). En la gráfica, estos puntos se dibujan por encima del rango de muerte máxima observada, con una marca distintiva (como un diamante) para diferenciarlos de los ciclos finitos.

4.3.1. Homología Persistente de Toplexes Cúbicos

Perseus actualmente puede calcular la homología persistente de los siguientes tipos de datos cúbicos:

Rejillas Cúbicas Densas

Una rejilla cúbica densa de dimensión d consiste en cubos con vértices enteros en el espacio euclidiano d-dimensional. Cada cubo recibe un tiempo de nacimiento, pero si estamos interesados en calcular la homología estándar en lugar de la persistencia, podemos establecer todos los tiempos de nacimiento de los

cubos que aparecen en 1. Por convención, un tiempo de nacimiento de -1 se reserva para los cubos que faltan en la rejilla.

El formato del archivo es sencillo. La primera línea contiene la dimensión de la rejilla cúbica, digamos d. Las siguientes d líneas del archivo indican el tamaño de la rejilla a lo largo de esa dimensión. Por supuesto, estos deben ser números no negativos. Entonces, para ingresar información sobre una rejilla de cuatro dimensiones de 100 por 30 por 50 por 12 cubos, las primeras cinco líneas de nuestro archivo deben ser 4, 100, 30, 50 y 12 en ese orden. Las líneas posteriores contienen los tiempos de nacimiento enteros de todos los cubos en orden lexicográfico de puntos de anclaje, con preferencia dada a las dimensiones en su orden elegido.

Para la demostración vamos a crear un archivo al que vamos a llamar 'grid1.txt' que contiene una rejilla 2D de 3 por 4 donde el tiempo de nacimiento de cada cubo 2D es la suma de las coordenadas de anclaje, excepto en el caso del cubo anclado en (1,1) que nunca aparece en el complejo y por lo tanto recibe un tiempo de nacimiento de -1. Nota que todos los índices comienzan con 0 en lugar de 1. El contenido de este archivo es el siguiente:

```
2
3
4
1
2
1
2
-1
2
1
5
1
2
2
2
```

A continuación se presenta una explicación detallada de cada número:

- **2**: Número de dimensiones de la rejilla cúbica (es una rejilla 2D).
- **3**: Número de cubos a lo largo de la primera dimensión de la rejilla (hay 3 cubos en la primera dimensión).
- **4**: Número de cubos a lo largo de la segunda dimensión de la rejilla (hay 4 cubos en la segunda dimensión).

Los siguientes números representan los tiempos de nacimiento de cada cubo 2D, ordenados lexicográficamente según sus puntos de anclaje:

- **1**: Tiempo de nacimiento del cubo 2D anclado en $(0, 0)$.
- **2**: Tiempo de nacimiento del cubo 2D anclado en $(1, 0)$.

- **1**: Tiempo de nacimiento del cubo 2D anclado en $(2, 0)$.
- **2**: Tiempo de nacimiento del cubo 2D anclado en $(0, 1)$.
- **-1**: Esto indica que no hay un cubo anclado en $(1, 1)$.
- **2**: Tiempo de nacimiento del cubo 2D anclado en $(2, 1)$.
- **1**: Tiempo de nacimiento del cubo 2D anclado en $(0, 2)$.
- **5**: Tiempo de nacimiento del cubo 2D anclado en $(1, 2)$.
- **1**: Tiempo de nacimiento del cubo 2D anclado en $(2, 2)$.
- **2**: Tiempo de nacimiento del cubo 2D anclado en $(0, 3)$.
- **2**: Tiempo de nacimiento del cubo 2D anclado en $(1, 3)$.
- **2**: Tiempo de nacimiento del cubo 2D anclado en $(2, 3)$.

La representacion de la rejilla 2D de 3x4 con los tiempos de nacimiento sería:

Coordenada (i, j)	**Tiempo de nacimiento**
(0,0)	1
(1,0)	2
(2,0)	1
(0,1)	2
(1,1)	-1
(2,1)	2
(0,2)	1
(1,2)	5
(2,2)	1
(0,3)	2
(1,3)	2
(2,3)	2

Cuadro 4.1: Tiempos de nacimiento de cada cubo en la rejilla 2D de 3x4.

Cada cubo de menor dimensión hereda automáticamente el tiempo de nacimiento mínimo de todos los cubos de mayor dimensión en su coborde. Una vez que tienes un archivo de entrada en el formato anterior, así es como invocamos el ejecutable compilado, que supongo se llama ”perseus”. La clave aquí es ”cubtop”, que es abreviatura de ”cubical toplex”:

```
./perseusLin cubtop (ruta archivo entrada) (cadena archivo salida)
```

Dado que esta estructura de datos se almacena internamente como una matriz contigua para el acceso en tiempo constante, cada cubo superior que nunca aparece (es decir, se le da un tiempo de nacimiento de -1) desperdicia tanto memoria como tiempo. Si hay muchos cubos que nunca aparecen, entonces es mejor usar el formato de entrada de rejilla cúbica dispersa que se describe más tarde.

Demostración del software

Primero, debemos descargar el archivo ejecutable desde la página de Perseus[16] correspondiente a nuestro sistema operativo. En mi caso, utilizo una máquina Linux y el ejecutable se llama `perseusLin`. En el caso de Windows, el ejecutable se llama `perseusWin`. Para poder visualizar los resultados obtenidos, es necesario tener Matlab[83] instalado. El código fuente contiene el script `persdia.m`, que se ha explicado previamente.

Desde el directorio donde se encuentra nuestro ejecutable, ejecutaremos el siguiente comando:

```
./perseusLin cubtop grid1.txt output
```

Esto generará la siguiente salida en la terminal:

```
Read 11 birth times from Input File
Written to Cell Complex! Complex stored with 62 cells!
+++coreductions:  62 --> 8, fraction removed 0.870968 at height 1
+++reductions:    8 --> 8, fraction removed 0 at height 2
+++coreductions:  8 --> 8, fraction removed 0 at height 3
Computing Persistence Intervals!
Linearly ordered 8 cells...
Done!!! Please consult [output*.txt] for results.
```

Se generarán tres archivos .txt:

- **output_0.txt**: Este archivo contiene los intervalos de persistencia para los H_0 (componentes conexas) o generador 0-dimensional de homología[84]. Cada línea en este archivo representa un intervalo de persistencia en la forma $[b_i, d_i)$, donde b_i es el tiempo de nacimiento y d_i es el tiempo de muerte de una componente conexa.

  El contenido de este archivo es el siguiente:

  ```
  1 2
  1 2
  1 2
  1 -1
  ```

- **output_1.txt**: Este archivo contiene los intervalos de persistencia para los H_1 (agujeros) o generador 1-dimensional de homología. Similar al archivo anterior, cada línea representa un intervalo de persistencia en la forma $[b_i, d_i)$, donde b_i es el tiempo de nacimiento y d_i es el tiempo de muerte de un agujero.

  El contenido de este archivo es el siguiente:

  ```
  2 -1
  ```

- **output_betti.txt**: Este archivo contiene los números de Betti en cada paso de la filtración. Los números de Betti son invariantes topológicos que

cuentan:

- El primer valor indica el tiempo en la filtración (cuando todas las celdas con tiempo de nacimiento menor o igual al número que contenga esta columna están incluidas).
- El segundo valor es el 0-ésimo número de Betti (β_0), que cuenta el número de componentes conectados.
- El tercer valor es el 1-ésimo número de Betti (β_1), que cuenta el número de túneles o agujeros 1-dimensionales.
- El cuarto valor es el 2-ésimo número de Betti (β_2), que cuenta el número de cavidades o vacíos 2-dimensionales.
- El resto de valores representan los números de Betti para dimensiones superiores, normalmente su valor será 0 a excepción de que haya generadores significativos en dimensiones más altas.

Los números de Betti son una secuencia de enteros $\beta_0, \beta_1, \ldots$ donde cada β_i cuenta el número de i-ciclos independientes en el complejo simplicial, y se pueden derivar de estos intervalos de persistencia.

El contenido de este archivo es el siguiente:

```
1 4 0
2 1 1
```

Para visualizar los resultados, debemos estar en la carpeta que contenga el archivo `persdia.m`, los outputs recién obtenidos y ejecutar el siguiente comando en Matlab:

```
>> persdia('output_1.txt');
```

La Figura 4.16, generada a través de Matlab, representa los intervalos de persistencia para los H_1 (agujeros) en el complejo simplicial generado a partir del archivo `grid1.txt`. En el eje x se encuentran los tiempos de nacimiento (*birth*) y en el eje y los tiempos de muerte (*death*) de los agujeros.

Cada punto en la gráfica corresponde a un ciclo 1-dimensional (agujero) detectado en el complejo:

- Punto rojo en la coordenada $(2, 3{,}05)$, indicando que hay un agujero que nace alrededor del tiempo 2 y y persiste hasta el final del rango de filtración observada, esto puede indicar una característica significativa de los datos.

Mediante el uso del mismo comando pero cambiando el argumento por 'output_0.txt' o 'output_betti.txt' obtenemos otras dos gráficas diferentes.

- Gráfica con 'output_0.txt': Esta gráfica (ver Figura4.17) muestra la persistencia de las características topológicas en el complejo.
  - Hay tres puntos azules en $(1, 2)$ indicando que hay tres características que nacen en el tiempo 1 y muere en el tiempo 2.

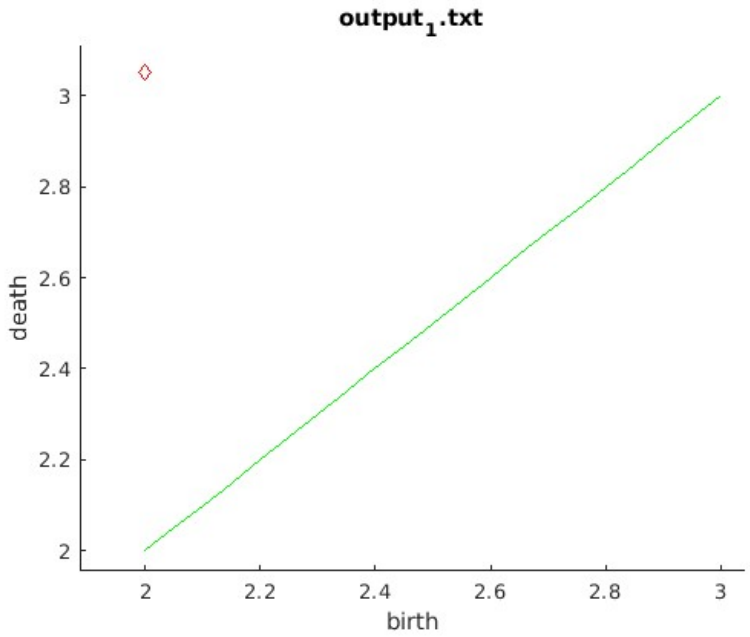

Figura 4.16: Gráfico generado por el script 'persdia' pasándole 'output_1.txt' como argumento.

- El punto rojo en $(1, 2{,}05)$ indica que nace en el tiempo 1 y persiste hasta el final del rango de filtración observada, esto puede indicar una característica significativa de los datos.

- Gráfica con 'output_betti.txt': Esta gráfica (ver Figura4.18) muestra los números de Betti en diferentes tiempos de la filtración, indicando el número de componentes conectados y agujeros.
  - El punto azul en $(1, 4)$ indica que en el tiempo 1 hay 4 componentes conectados (β_0) y 0 agujeros (β_1). Representa el estado topológico del complejo cuando se incluyen todas las celdas con tiempo de nacimiento menor o igual a 1.
  - El punto azul en $(2, 1)$ indica que en el tiempo 2 hay 1 componente conectado (β_0) y 1 agujero (β_1). Representa el estado topológico del complejo cuando se incluyen todas las celdas con tiempo de nacimiento menor o igual a 2.

Rejillas Cúbicas Dispersas

Este formato complejo es apropiado para una estructura de cuadrícula cúbica donde una gran fracción de los cubos están ausentes. Por ejemplo, si tenemos una gran esfera (hueca) o toro aproximado por cubos, no deseamos tener cubos cubriendo las porciones huecas, ya que sería muy desperdiciado — un archivo de entrada casi lleno de -1's — si usáramos la estructura de cuadrícula cúbica densa mencionada anteriormente. Como heurística, digamos que deseas calcular la homología de la región negra en alguna imagen pixelada en blanco y negro

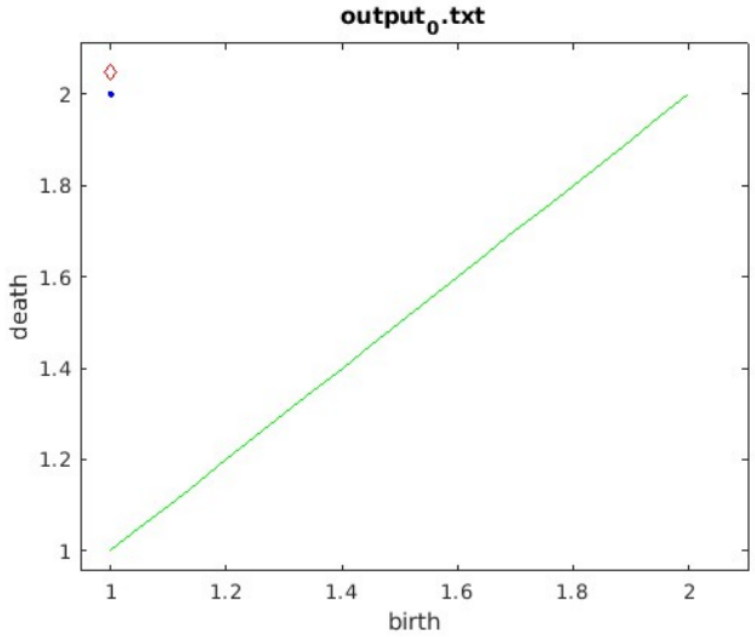

Figura 4.17: Gráfico generado por el script 'persdia' pasándole 'output_0.txt' como argumento.

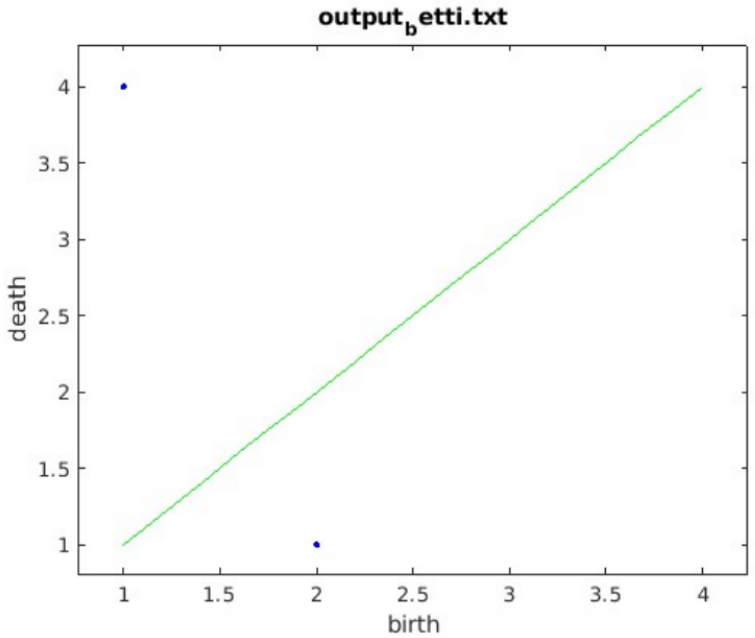

Figura 4.18: Salida del script 'persdia' pasándole 'output_betti.txt' como argumento.

de tamaño 1000 por 1000. Si más del 70 por ciento del total de 1 millón de píxeles son negros, entonces no estará mal usar el formato de cuadrícula cúbica densa anterior. Pero si el número de píxeles negros es menor al 70 por ciento, probablemente deberíamos usar la siguiente implementación de cuadrícula cúbica dispersa.

El formato del archivo de entrada es el siguiente. El primer número es la dimensión deseada d de los cubos que se están describiendo. Después de esto, cada cubo está representado por $d + 1$ números separados por espacios: los primeros d números deben ser las coordenadas del ancla del cubo (es decir, el vértice minimal lexicográficamente) y el número final debe ser el tiempo de nacimiento deseado para ese cubo. Nuevamente, evitamos usar -1 como tiempo de nacimiento a menos que deseemos que ese cubo sea completamente ignorado, en cuyo caso no hay necesidad de escribirlo en el archivo de entrada en primer lugar. Usaremos un pequeño archivo de ejemplo al que llamaremos "grid2.txt" que crea 4 cubos en dimensión 3, y aquí está una explicación de los números en el ejemplo:

- **3**: esta es la dimensión de la cuadrícula cúbica.
- **1 3 4 2**: este es el cubo 3D anclado en (1,3,4) con tiempo de nacimiento 2.
- **2 12 -4 7**: este es el cubo 3D anclado en (2, 12, -4) con tiempo de nacimiento 7.
- **0 2 3 1**: este es el cubo 3D anclado en (0,2,3) con tiempo de nacimiento 1.

Por supuesto, podemos ingresar muchos más cubos. Si ingresamos múltiples cubos con los mismos puntos de ancla y diferentes tiempos de nacimiento, solo se considerará el primero y los demás se ignorarán. En este ejemplo, podemos ver que el punto (1,3,4) aparece como una cara del primer cubo, así como del tercer cubo. En este caso, se le asignará el tiempo de nacimiento más bajo de 1 del tercer cubo. Si estamos interesados en calcular la homología estándar en lugar de la persistencia, simplemente podemos dar a todos los cubos un tiempo de nacimiento igual a 1.

Una vez que tengamos un archivo de entrada en el formato deseado, así es como le pedimos a Perseus que calcule su homología persistente. Nota el uso de la palabra clave `scubtop` que significa "sparse cubical toplex" o "rejillas cúbicas dispersas":

```
./perseusLin scubtop (ruta archivo entrada) (cadena archivo salida)
```

En el caso que explico se llamará de la siguiente manera:

```
./perseusLin scubtop grid2.txt sparse
```

Esto nos va a generar 2 archivos:

- **sparse_0.txt**: Que genera la siguiente salida:

```
1 -1
7 -1
```

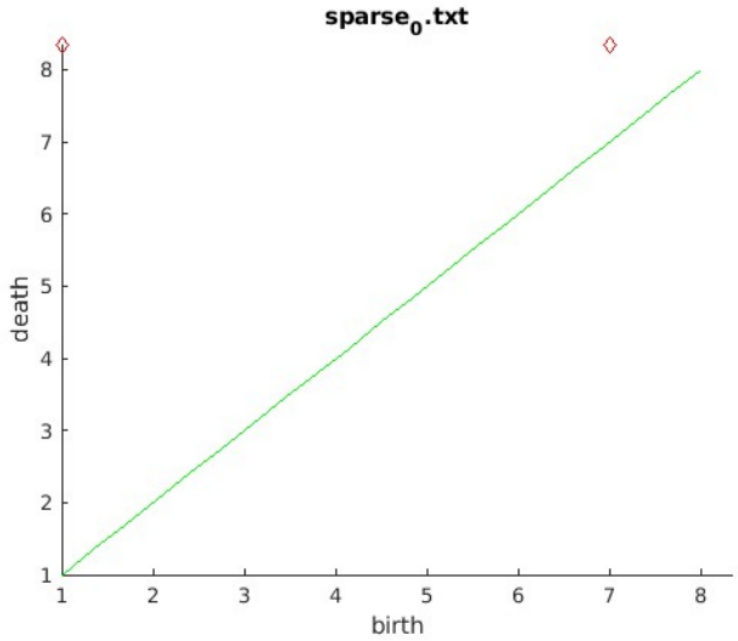

Figura 4.19: Gráfico generado por el script 'persdia' pasándole 'sparse_0.txt' como argumento.

- **sparse_betti.txt**: Que genera la siguiente salida:

```
1 1
7 2
```

Al ejecutar el script "persdia" en matlab obtendremos las siguientes representaciones para **sparse_0.txt** (ver Figura4.19) y **sparse_betti.txt** (ver Figura4.20), de las cuales podemos deducir:

- **sparse_0.txt**: Hay dos puntos rojos en las posiciones (1, 8.35) y (7, 8.35) los cuales nos permiten saber que que ambos ciclos persisten hasta el final del rango de filtración observado. Es decir, estos ciclos no 'mueren' dentro del rango de la filtración, lo que indica características topológicas que permanecen persistentes a lo largo de todo el proceso. La persistencia de estos ciclos puede señalar componentes conectados significativos en la estructura de datos analizada.
- **sparse_betti.txt**: Hay un punto azul en la posición (1, 1) que indica que hay un componente conectado (β_0=1). Este ciclo no persiste significativamente, sugiriendo que podría ser ruido ya que está en la línea diagonal (nace en el tiempo 1 y muere en el tiempo 1). Además, hay un punto azul en la posición (7, 2) que representa un ciclo 1-dimensional que nace en el tiempo 7 y muere en el tiempo 2 que indica que hay un componente conectado (β_0=2). Este punto indica un ciclo significativo que persiste durante un rango de filtración más largo, señalando una característica topológica relevante en los datos analizados.

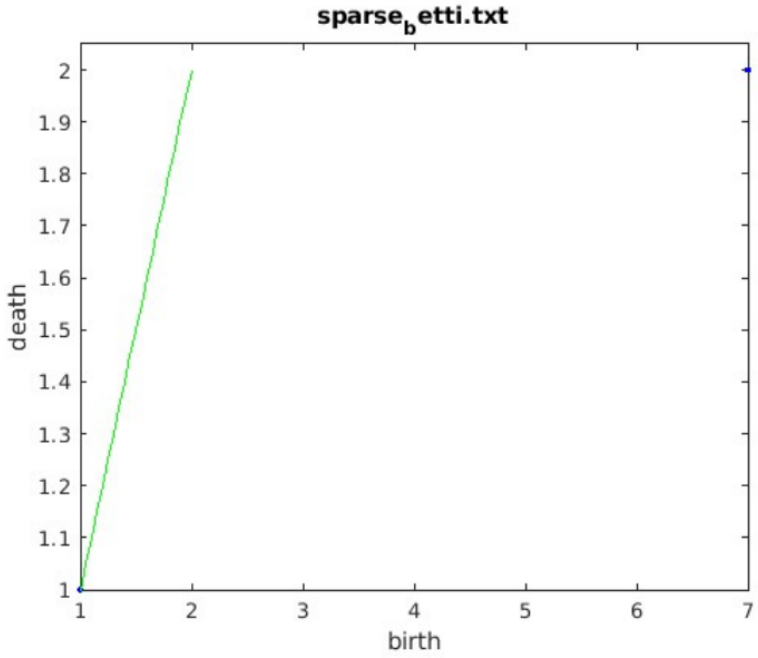

Figura 4.20: Gráfico generado por el script 'persdia' pasándole 'sparse_betti.txt' como argumento.

4.3.2. Homología Persistente de Toplexes Simpliciales

Un *toplex simplicial* es una lista de símplices de dimensión máxima a partir de los cuales se extrae automáticamente la información de las caras para construir un complejo simplicial usual. Existen dos posibles casos de toplex simpliciales: uno en el que hay una dimensión máxima uniforme para todos los símplices y otro en el que hay diferentes dimensiones máximas.

Triangular una variedad de dimensión d siempre produce un complejo simplicial con dimensión máxima uniforme d. Por otro lado, un complejo simplicial general no necesita tener una dimensión uniforme. Por ejemplo, todo el complejo podría consistir en un 1-símplice que comparte un vértice con un 3-símplice.

Representación de Símplices

Para preparar nuestro complejo simplicial para Perseus, primero debemos proporcionar la dimensión ambiente de los vértices, que denotaremos aquí por n. Recomiendo que simplemente indexemos todos los vértices presentes con números naturales en cualquier orden y luego usemos "1" como la dimensión ambiente. Este no es un paso necesario, pero hará que las cosas sean muy eficientes en términos de tiempo y memoria. En cualquier caso, un punto de dimensión ambiente n se representa de forma única por sus coordenadas, que son simplemente una colección ordenada de n números de punto flotante. Si estamos usando la idea de indexar los vértices por números naturales, entonces cada punto es solo un número natural y no es necesario proporcionar información de coordenadas.

Un símplice de dimensión d se representa por una colección de $d+1$ puntos, donde cada punto es solo una colección de n coordenadas como se describió anteriormente. Entonces, realmente necesitamos $n(d+1)$ números por símplice. Aquí, por ejemplo, está el 2-símplice determinado por los vértices (0.1, 3.4, 7.1), (2.4, -1.6, 0.4) y (13.5, -7.9, 1.0):

$$0{,}1\ 3{,}4\ 7{,}1\ 2{,}4\ -1{,}6\ 0{,}4\ 13{,}5\ -7{,}9\ 1{,}0$$

Podemos reorganizar el orden de los vértices a nuestro gusto, pero por supuesto es una mala idea cambiar el orden de las coordenadas de un solo vértice. Por otro lado, si indexamos los vértices por orden lexicográfico, entonces este símplice se representaría en el formato mucho más conveniente:

$$1\ 2\ 3$$

Perseus puede usarse para calcular la homología persistente de los siguientes tipos de complejos simpliciales:

Triangulaciones Uniformes

El formato del archivo para un toplex simplicial con dimensión máxima uniforme d es el siguiente. El primer número del archivo es d, la dimensión máxima uniforme. El siguiente número es n, el número de coordenadas para cada vértice. Nuevamente, recomiendo etiquetar los puntos con números naturales y establecer $n = 1$ como se describió anteriormente. Luego, cada línea subsiguiente contiene $n(d+1)+1$ números. Los primeros $n(d+1)$ de estos son números de punto flotante que representan los vértices de un solo símplice en cualquier orden como se describió anteriormente. El último número es un entero y corresponde al tiempo de nacimiento de ese símplice. Nuevamente, tenemos en cuenta que los símplices dados con un tiempo de nacimiento de -1 serán ignorados. Por otro lado, si solo deseamos calcular la homología ordinaria no persistente, debemos dar a todos los símplices un tiempo de nacimiento de 1.

Para la demostración crearemos un archivo de texto simple con nombre "grid3.txt" que consiste en cuatro símplices de 3 dimensiones cuyos vértices tienen dos coordenadas cada uno. Aquí hay una explicación de los números en este archivo:

- **3**: esta es la dimensión de los símplices, por lo que cada símplice tiene 3+1 = 4 vértices.
- **2**: este es el número de coordenadas por vértice.
- **0 0 0 1 1 0 1 1 1**: este es el 3-símplice con vértices (0,0), (0,1), (1,0), (1,1) y tiempo de nacimiento 1.

Para que Perseus calcule la homología persistente de nuestro archivo de entrada una vez que esté preparado, solo lo llamamos usando la palabra clave `simtop` de la siguiente manera:

```
./perseusLin simtop (ruta archivo entrada) (cadena archivo salida)
```

En el caso que explico se llamará de la siguiente manera:

```
./perseusLin simtop grid3.txt uniform
```

Al ejecutar el comando en nuestra máquina obtendremos la siguiente respuesta:

```
Read 4 top simplices from Input File
Writing Cell Complex From Simplicial Complex
Done! Complex stored with 45 cells!
+++coreductions:  45 --> 1, fraction removed 0.977778 at height 1
+++reductions:    1 --> 1, fraction removed 0 at height 2
+++coreductions:  1 --> 1, fraction removed 0 at height 3
Computing Persistence Intervals!
Linearly ordered 1 cells...
Done!!! Please consult [uniform*.txt] for results.
```

Lo que nos generará los siguientes archivos:

- **uniform_0.txt**: El contenido de este archivo es el siguiente:

  ```
  1 -1
  ```

  En este caso, el intervalo `1 -1` indica que hay un generador de 0-homología que nace en el tiempo 1 y persiste indefinidamente.

- **uniform_betti.txt**: Este archivo resume la información de los números de Betti en cada paso de la filtración. El contenido es el siguiente:

  ```
  1 1
  ```

  En este caso, `1 1` indica que cuando todas las celdas con tiempo de nacimiento menor o igual a 1 están incluidas, hay un componente conectado en la homología 0 (0-homología).

Podemos visualizar esto con el uso del script 'persdia.m' obtendremos las siguientes gráficas:

- Gráfica de uniform_0.txt (ver Figura4.21): Podemos ver que hay un punto rojo en (1, 2.05).
- Gráfica de uniform_betti.txt (ver Figura4.22): Podemos ver que hay un punto azul en (1, 1).

Triangulaciones no uniformes

Aquí consideraremos el caso general en el que el toplex simplicial no tiene una dimensión máxima uniforme. Por supuesto, la dimensión ambiente de los puntos vértice sigue siendo constante en todo momento y la llamaremos n. Este será el primer número en el archivo de entrada. El resto de las líneas están dedica-

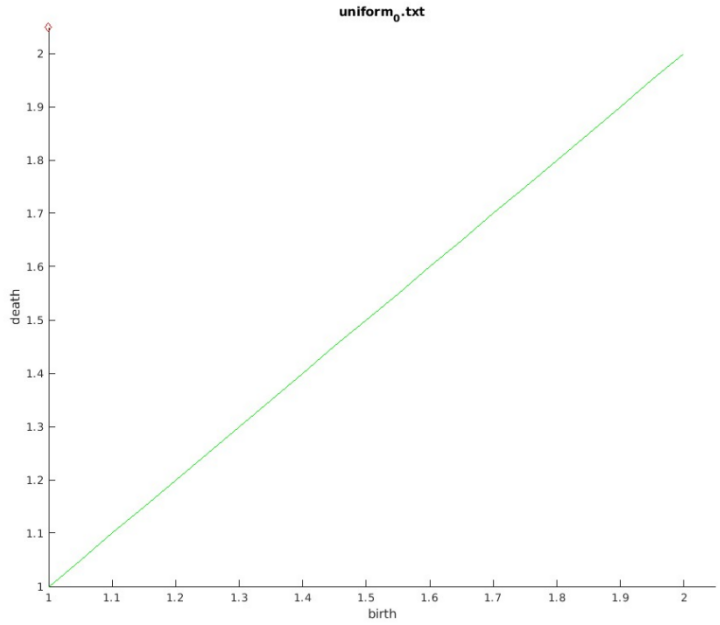

Figura 4.21: Gráfico generado por el script 'persdia' pasándole 'uniform_0.txt' como argumento.

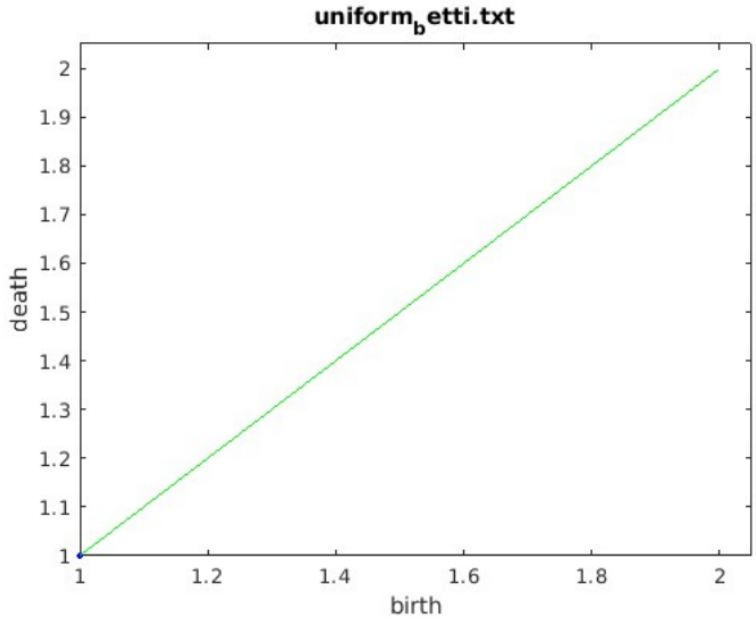

Figura 4.22: Gráfico generado por el script 'persdia' pasándole 'uniform_betti.txt' como argumento.

das a los vértices en el símplice. Nuevamente, recomiendo establecer n en 1 y simplemente etiquetar los vértices con números enteros.

Dado que no hay una dimensión uniforme en la lista de símplices, tendremos que proporcionar a cada símplice su propia dimensión. La forma de hacer esto es bastante sencilla: antes de ingresar las coordenadas de los vértices en nuestro símplice, solo ponemos un número adicional m que indique la dimensión de ese símplice; de esta manera, el software sabe que los próximos $n(m+1)$ números en el archivo son las coordenadas de los vértices de ese símplice en particular. Aquí hay un ejemplo (con la dimensión del punto n igual a 3) de un símplice 2D con vértices (0.1, 0.3, 7), (-2, 8.1, 0) y (8.9, -3, 9.2). Asumimos que este símplice tiene un tiempo de nacimiento de 5, que es el último número como de costumbre. Nota el "2" al comienzo de esta línea para indicar la dimensión.

```
2 0.1 0.3 7 -2 8.1 0 8.9 -3 9.2 5
```

Usaremos un archivo de texto que consta de algunos símplices en varias dimensiones al que he llamado "grid4.txt". Los vértices están indexados con números naturales ($n = 1$), como se indica en la parte superior del archivo. Aquí hay una explicación de algunos de los números en este archivo:

- **1**: este es el número de coordenadas por vértice.
- **2 1 3 5 1**: este es el símplice 2D con vértices 1, 3 y 5; el tiempo de nacimiento es 1.
- **3 1 2 4 6 2**: este es el símplice 3D con vértices 1, 2, 4 y 6; el tiempo de nacimiento es 2.
- **6 1 2 3 4 5 6 7 4**: símplice 6D, vértices 1 al 7.

Tenemos en cuenta que es posible introducir varias inconsistencias con este formato. Por ejemplo, podemos declarar que un símplice tiene un tiempo de nacimiento de 3 y luego declarar que una de sus caras tiene un tiempo de nacimiento de 10. En este caso, el tiempo de nacimiento de la cara se cambiará sin piedad a 3 para que se preserve la estructura de la filtración. No se permite que ninguna cara nazca después de su símplice padre. Como de costumbre, si no te importa la persistencia y simplemente queremos calcular la homología, solo establecemos todos los tiempos de nacimiento en 1 explícitamente.

Para que Perseus calcule la homología persistente de nuestro complejo una vez que haya sido codificado en un archivo de texto, usamos la palabra clave del toplex simplicial no uniforme `nmfsimtop` de la siguiente manera:

```
./perseusLin nmfsimtop (ruta archivo entrada) (cadena archivo salida)
```

En el caso que explico se llamará de la siguiente manera:

```
./perseusLin nmfsimtop grid4.txt nonuniform
```

Al ejecutar el comando en nuestra máquina obtendremos la siguiente respuesta:

```
Read 4 top simplices from Input File
Writing Cell Complex From Non-manifold simplicial Complex
```

```
Done! Complex stored with 127 cells!
+++coreductions:  127 --> 1, fraction removed 0.992126 at height 1
+++reductions:    1 --> 1, fraction removed 0 at height 2
+++coreductions:  1 --> 1, fraction removed 0 at height 3
Computing Persistence Intervals!
Linearly ordered 1 cells...
Done!!! Please consult [nonuniform*.txt] for results.
```

Lo que nos generará los siguientes archivos:

- **nonuniform_0.txt**: El contenido de este archivo es el siguiente:

  ```
  1 -1
  ```

  En este caso, el intervalo `1 -1` indica que hay un generador de 0-homología que nace en el tiempo 1 y persiste indefinidamente.

- **nonuniform_betti.txt**: Este archivo resume la información de los números de Betti en cada paso de la filtración. El contenido es el siguiente:

  ```
  1 1
  ```

  En este caso, `1 1` indica que cuando todas las celdas con tiempo de nacimiento menor o igual a 1 están incluidas, hay un componente conectado en la homología 0 (0-homología).

Las gráficas van a ser exactamente igual que en el en caso de las triangulaciones uniformes.

4.3.3. Homología Persistente de Complejos de Vietoris-Rips

El complejo de Vietoris está completamente determinado por el 1-esqueleto subyacente. Este esqueleto se puede representar como una matriz de distancias simétrica donde las entradas provienen de distancias por pares entre puntos en una nube de puntos, o de correlaciones y una variedad de otras fuentes. Perseus puede calcular la homología persistente de complejos de Vietoris generados alrededor de tres tipos diferentes de datos: puntos de nacimiento uniformes, puntos de nacimiento no uniformes y matrices de distancia.

Puntos con Tiempos de Nacimiento Uniformes

Este es el tipo más común de complejo de Vietoris-Rips. La entrada es una lista de vértices (es decir, puntos) embebidos en algún espacio euclidiano. Para cada vértice hay un radio inicial r. El radio para cada vértice se incrementa N veces por algún tamaño de paso universal s para darnos la secuencia creciente $r, r+s, r+2s, r+3s, \ldots, r+Ns$ de radios para cada punto (generalmente, se usa $r = 0$ para todos los puntos, pero esto no es necesario). La imagen correcta aquí es imaginar bolas creciendo con un radio cada vez mayor alrededor de cada vértice. Dados dos vértices, la arista entre ellos se le asigna un tiempo de nacimiento igual al primer número de paso entre 1 y N donde sus bolas

asociadas se intersectan. Por supuesto, es posible que los puntos estén tan lejos y sus radios iniciales sean tan pequeños que las bolas a su alrededor no se intersecten incluso cuando el radio se incrementa por un factor aditivo de Ns, y en este caso no se introduce ninguna arista entre estos vértices.

El archivo de entrada para este tipo de complejo es el siguiente. La primera línea es la dimensión ambiente d donde viven los vértices, es decir, el número de coordenadas para cada vértice. Los siguientes tres números son, en orden: un multiplicador de radio inicial k, el tamaño de paso universal s y el número total de pasos N. Lo único nuevo aquí es k: esta es una manera rápida de escalar los radios asociados a cada vértice por un factor uniforme. Casi siempre, deberíamos simplemente establecer $k = 1$. Si k es igual a 0.5, por ejemplo, los radios iniciales de todos los vértices se reducirán a la mitad. Después de k, s y N puedemos comenzar a ingresar la información del vértice y del radio inicial en el siguiente formato.

Introducimos $d + 1$ números de punto flotante: los primeros d dan las coordenadas del vértice y el último (que debe ser no negativo, de lo contrario el vértice se ignora completamente) es el radio inicial r para ese vértice. Por ejemplo, cuando $n = 4$, entonces la línea $2{,}3\ \ 4{,}5\ \ -3{,}7\ \ 2{,}8\ \ 0{,}15$ corresponde al vértice $(2{,}3, 4{,}5, -3{,}7, 2{,}8)$ con el radio inicial $0{,}15$. Usaremos este un archivo llamado "grid5.txt" de ejemplo. Los números en este archivo se explican a continuación:

- **3**: la dimensión ambiente, es decir, el número de coordenadas por vértice.
- **1 0.01 100**: el factor de escalado del radio $k = 1$, el tamaño de paso $s = 0{,}01$, el número de pasos $N = 100$
- **1.2 3.4 -0.9 0.5**: el vértice $(1{,}2, 3{,}4, -0{,}9)$ con radio asociado $r = 0{,}5$
- **2.0 -6.6 4.1 0.3**: el vértice $(2{,}0, -6{,}6, 4{,}1)$ con radio asociado $r = 0{,}3$

Y así sucesivamente.

Finalmente, para calcular la homología persistente de este complejo, usamos la palabra clave `brips`, de la siguiente manera:

```
./perseusLin brips (ruta archivo entrada) (cadena archivo salida)
```

En el caso que explico se llamará de la siguiente manera:

```
./perseusLin brips grid5.txt unibirth
```

Al ejecutar el comando en nuestra máquina obtendremos la siguiente respuesta:

```
#points read: 3
nbr matrix made:
Read 3 point/radius pairs and birth times!
Writing Cell Complex From RIPS Complex
Done! Complex stored with 4 cells!
+++coreductions:  4 --> 2, fraction removed 0.5 at height 1
+++reductions:    2 --> 2, fraction removed 0 at height 2
+++coreductions:  2 --> 2, fraction removed 0 at height 3
Computing Persistence Intervals!
```

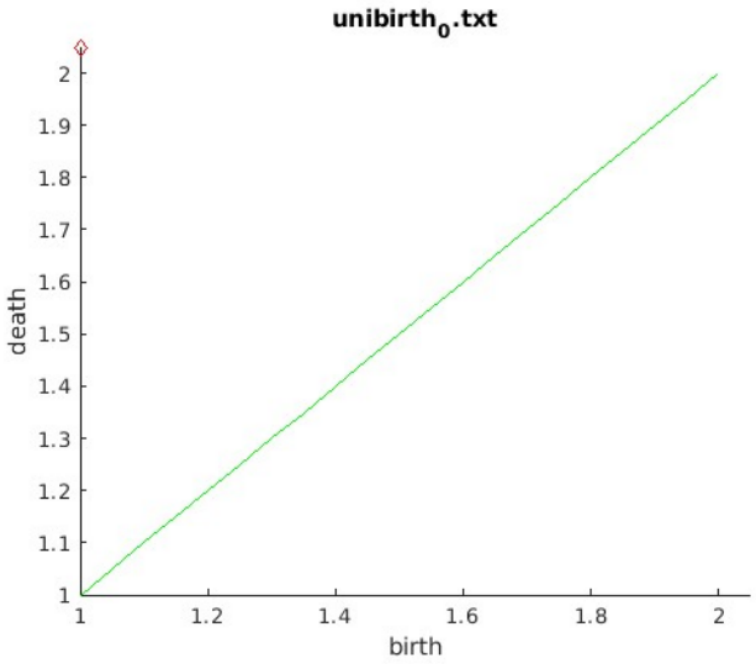

Figura 4.23: Gráfico generado por el script 'persdia' pasándole 'unibirth_0.txt' como argumento.

```
Linearly ordered 2 cells...
Done!!! Please consult [unibirth*.txt] for results.
```

Lo que nos generará los siguientes archivos:

- **unibirth_0.txt**: El contenido de este archivo es el siguiente:

  ```
  1 -1
  1 -1
  ```

- **nonuniform_betti.txt**: Este archivo resume la información de los números de Betti en cada paso de la filtración. El contenido es el siguiente:

  ```
  1 2
  ```

Podemos visualizar esto con el uso del script 'persdia.m' obtendremos las siguientes gráficas:

- Gráfica de unibirth_0.txt (ver Figura4.23): Podemos ver que hay un punto rojo en (1, 2.05), sin embargo viendo el archivo .txt podemos ver que realmente serán dos superpuestos con persistencia infinita.
- Gráfica de unibirth_betti.txt (ver Figura4.24): Podemos ver que hay un punto azul en (2, 1). El cual indica que en el tiempo de filtración 2, el complejo topológico tiene 1 componente conexa.

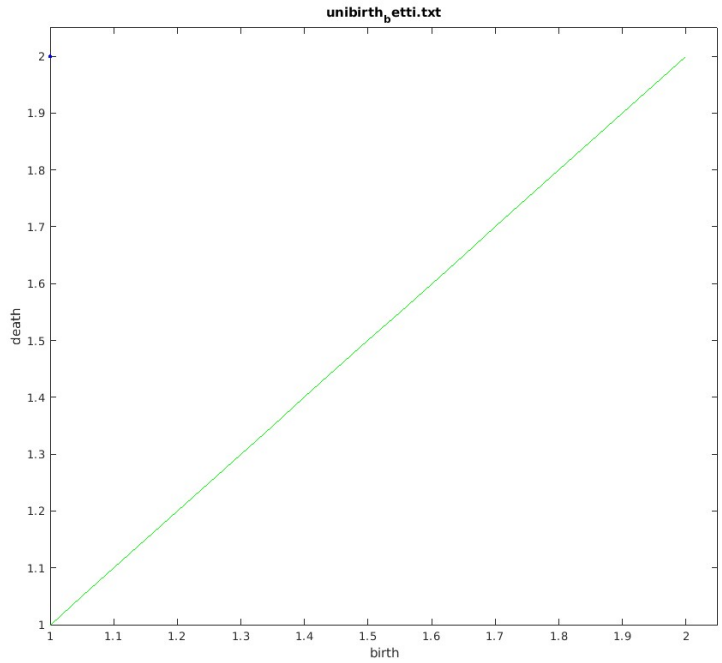

Figura 4.24: Gráfico generado por el script 'persdia' pasándole 'unibirth_betti.txt' como argumento.

Puntos con Diferentes Tiempos de Nacimiento

Este tipo de complejo de Vietoris se crea con triples vértice-radio-nacimiento. Cada vértice tiene su propio tiempo de nacimiento y un radio asociado. Dos vértices obtienen una arista entre ellos si las bolas correspondientes de un radio dado se intersectan. Esta arista obtiene el máximo de los tiempos de nacimiento de los vértices. Por ejemplo, supongamos que tenemos los triples vértice-radio-nacimiento $[(0,0,0);1;2]$ y $[(0,1,1);1;3]$. La distancia entre los vértices es aproximadamente 1.414, y la suma de sus radios es 2, por lo que sabemos que habrá una arista entre ellos. Esta arista obtiene el máximo tiempo de nacimiento de estos dos vértices, que en este caso es 3. De manera similar, los símplices 2D obtienen los tiempos de nacimiento máximos de las aristas en sus fronteras, y así sucesivamente.

El formato del archivo es sencillo, como se muestra a continuación. El primer número n es la dimensión de embebido de los vértices, es decir, el número de coordenadas por vértice. Cada línea subsiguiente consta de n coordenadas de punto flotante para un vértice, seguidas de un radio de punto flotante no negativo y un entero de nacimiento. Como de costumbre, los vértices con nacimiento igual a -1 serán ignorados. Usaremos un archivo llamado "grid6.txt" donde los números están organizados como se explica a continuación:

- **2**: esta es la dimensión ambiente, es decir, el número de coordenadas por vértice.

- **0.1 0.1 0.52 1**: el vértice $(0{,}1, 0{,}1)$ con radio 0.52 y nacimiento 1.
- **-0.2 0.1 0.51 3**: el vértice $(-0{,}2, 0{,}1)$ con radio 0.51 y nacimiento 3.

Para calcular la homología persistente de este complejo, se debe usar la palabra clave `rips` para el tipo de complejo:

```
./perseusLin rips (ruta archivo entrada) (cadena archivo salida)
```

En el caso que explico se llamará de la siguiente manera:

```
./perseusLin rips grid6.txt difbirth
```

Al ejecutar el comando en nuestra máquina obtendremos la siguiente respuesta:

```
Read 3 point/radius pairs and birth times!
Writing Cell Complex From RIPS Complex
Done! Complex stored with 4 cells!
+++coreductions:  4 --> 2, fraction removed 0.5 at height 1
+++reductions:    2 --> 2, fraction removed 0 at height 2
+++coreductions:  2 --> 2, fraction removed 0 at height 3
Computing Persistence Intervals!
Linearly ordered 2 cells...
Done!!! Please consult [difbirth*.txt] for results.
```

A partir de aquí es exactamente igual que en el apartado anterior.

Matrices de Distancia y Correlación

A veces no es factible obtener una muestra de puntos embebida en algún espacio euclidiano. En su lugar, podemos tener una matriz de distancias simétrica donde la entrada (i, j) registra la distancia entre los puntos i y j directamente sin proporcionar ninguna información sobre la ubicación de dichos puntos. Otro enfoque relacionado es registrar las correlaciones de pares de eventos, en cuyo caso la noción de puntos y distancias entre ellos ni siquiera tiene sentido.

En cualquier caso, es posible construir un complejo de Vietoris-Rips directamente sin ninguna información sobre los puntos. De hecho, una de las mayores ventajas del complejo de Vietoris-Rips sobre sus rivales, los complejos de Čech y Alpha —además de su eficiente computabilidad— es el hecho de que se puede construir a partir de una matriz de distancias en lugar de ubicaciones de puntos. Estas ventajas tienen un precio: no existe un análogo del lema del nervio para los complejos de Vietoris-Rips.

Para crear un archivo de entrada adecuado para Perseus, primero proporcionamos el tamaño M de nuestra matriz de distancias. La matriz debe ser cuadrada, por lo que solo es necesario un número: simplemente escribimos 10 para una matriz de 10 por 10. Luego, proporcionamos una distancia de umbral mínima g que tiene el siguiente significado: cualesquier dos puntos que estén a una distancia menor que g obtienen una arista entre ellos nacida en el tiempo cero. Entonces, para desconectar todos los puntos distintos inicialmente, debemos

establecer g en cero. Los siguientes números son el tamaño de paso s como antes, seguido de N, el número de pasos. Así, una arista nace en el paso n entre dos vértices si n es el número más pequeño entre 1 y N de modo que el umbral aumentado $ns + g$ en el n-ésimo paso exceda la distancia por pares entre estos vértices según lo codificado en la matriz de entrada. Por supuesto, si n es mayor que el número total de pasos N, entonces esa arista nunca se crea. Finalmente, debemos ingresar un número natural positivo C que limite la dimensión total del complejo. No se construirán símplices de dimensión estrictamente mayor que C. Esto se hace por razones computacionales: si las características de interés son de baja dimensión, entonces no hay necesidad de desperdiciar tiempo y memoria construyendo complejos de mayor dimensión.

Finalmente, insertamos M-cuadrado números de punto flotante que indiquen las entradas de una matriz de distancias. Las entradas diagonales deben ser cero, ya que representan la distancia de cada punto a sí mismo. Usaremos un archivo llamado "grid7.txt" que contiene un pequeño ejemplo (con $M = 3$) junto con una explicación de los números.

- **3**: este es el número de filas/columnas en la matriz de distancias simétrica.
- **0.1 0.2 5 2**: distancia de umbral inicial $g = 0{,}1$, tamaño de paso $s = 0{,}2$, número de pasos $N = 5$ y límite de dimensión $C = 2$.
- **0 0.26 0.4**: distancia de la entrada 1 a sí misma, entrada 2, entrada 3.
- **0.26 0 2.1**: distancia de la entrada 2 a la entrada 1, a sí misma y a la entrada 3.
- **0.4 2.1 0**: etc.

Si en lugar de distancias estuviéramos usando correlaciones entre pares de eventos, entonces la "distancia" efectiva sería $1 -$ correlación, donde correlación es el número leído del archivo. No es necesario cambiar explícitamente tus entradas de datos en este caso, Perseus también acepta entradas de correlación directamente y realiza el cálculo $1 -$ valor como se explica a continuación.

Para calcular la homología persistente para dicha entrada, deben usarse las palabras clave `distmat` o `corrmat` como tipo de complejo:

```
./perseusLin distmat (ruta al archivo de la matriz de distancias)
(cadena de archivo de salida)

./perseusLin corrmat (ruta al archivo de la matriz de correlación)
(cadena de archivo de salida)
```

En el caso que explico se llamará de la siguiente manera:

```
./perseusLin distmat grid7.txt dist

./perseusLin corrmat grid7.txt cor
```

Al ejecutar el comando con 'distmat' en nuestra máquina obtendremos la siguiente respuesta:

```
Read 3 point/distance pairs!
Done! Complex stored with 5 cells!
+++coreductions:  5 --> 5, fraction removed 0 at height 1
+++reductions:    5 --> 5, fraction removed 0 at height 2
Computing Persistence Intervals!
Linearly ordered 5 cells...
Done!!! Please consult [dist*.txt] for results.
```

Lo que nos generará los siguientes archivos:

- **dist_0.txt**: El contenido de este archivo es el siguiente:

```
0 3
0 4
0 -1
```

- **dist_1.txt**: El cual esta vacío, por lo que no hay clicos de dimensión 1 (agujeros) en la complejidad Vietoris-Rips para los datos proporcionados en el archivo de matriz de distancias.
- **dist_betti.txt**: Este archivo resume la información de los números de Betti en cada paso de la filtración. El contenido es el siguiente:

```
0 3 0
3 2 0
4 1 0
```

Al ejecutar el comando con 'corrmat' en nuestra máquina obtendremos la siguiente respuesta:

```
Read 3 point/distance pairs!
Done! Complex stored with 4 cells!
+++coreductions:  4 --> 2, fraction removed 0.5 at height 1
+++reductions:    2 --> 2, fraction removed 0 at height 2
+++coreductions:  2 --> 2, fraction removed 0 at height 3
Computing Persistence Intervals!
Linearly ordered 2 cells...
Done!!! Please consult [cor*.txt] for results.
```

Lo que nos generará los siguientes archivos:

- **cor_0.txt**: El contenido de este archivo es el siguiente:

```
0 -1
0 -1
```

- **cor_betti.txt**: Este archivo resume la información de los números de Betti en cada paso de la filtración. El contenido es el siguiente:

```
0 2
```

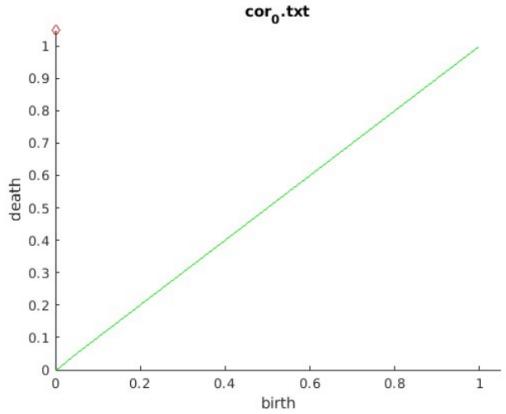

Figura 4.25: Gráfico generado por el script 'persdia' pasándole 'cor_0.txt' como argumento.

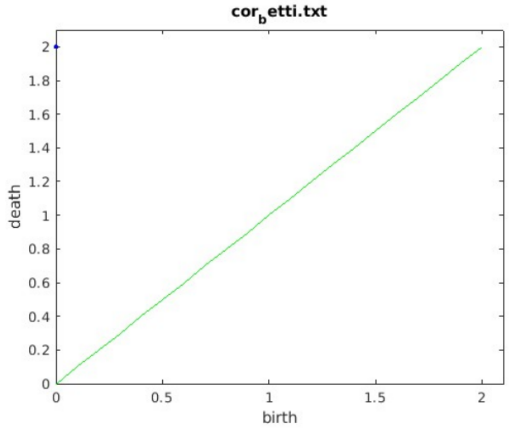

Figura 4.26: Gráfico generado por el script 'persdia' pasándole 'cor_betti.txt' como argumento.

Esto nos va a generar las siguientes gráficas (ver Figuras 4.25, 4.26, 4.27 y 4.28), las cuales hemos de interpretar siguiendo la misma lógica que en el resto de casos explicados previamente.

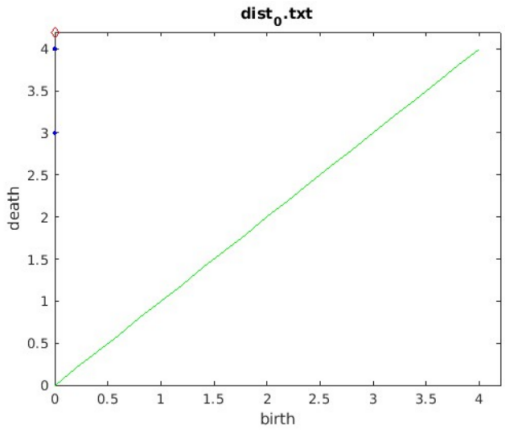

Figura 4.27: Gráfico generado por el script 'persdia' pasándole 'dist_0.txt' como argumento.

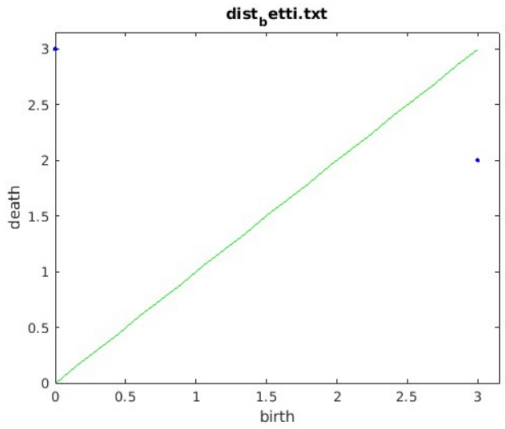

Figura 4.28: Gráfico generado por el script 'persdia' pasándole 'dist_betti.txt' como argumento.

Capítulo 5

Conclusiones y trabajo futuro

En este capítulo se expondrá una valoración del resultado obtenido del trabajo, impacto, conclusión y trabajo futuro.

5.1. Resumen de resultados obtenidos

Los resultados obtenidos del trabajo han sido todos satisfactorios al hacer uso de Software existente y no tener que desarrollar líneas de código. A pesar de esto me ha permitido visualizar los principales métodos del análisis topológico de datos y en casos como el software KeplerMapper haciendo uso de datasets de cáncer de mama de Wisconsin me ha permitido contextualizar el uso que pueden llegar a tener realmente estos métodos en el día a día o el desarrollo de ciertos campos como la biología o las ciencias sociales.

Cada uno de los sofwares ha permitido generar una representación diferente:

- TTK ha facilitado la generación de diagramas de persistencia para destacar características significativas como protuberancias y estructuras persistentes que reflejan la conectividad y forma del espacio de datos.
- KepplerMapper por otro lado ofrece visualizaciones detalladas que revelan patrones ocultos en datasets complejos como por ejemplo agrupamientos y conexiones dentro del dataset lo que nos ayuda a comprender de manera visual las características que poseen los datos.
- Perseus nos permite calcular la homología persistente de diferentes tipos de complejos incluyendo cúbicos, simpliciales y de Vietoris-Rips lo que nos ayuda a entender la estructura y los componentes significativos de los mismos ya que facilitan la identificación de ciclos y cavidades persistentes.

En conjunto estos softwares revisados en el trabajo permiten explorar y comprender mejor los datasets, haciendolo de manera rápida y efectiva. Facilitan la labor de encontrar características significativas de la estructura topológica de los datos y aportan representaciones visuales claras que facilitan la interpretación de los datos estudiados.

En cuanto a los objetivos marcados en la introducción, se han cumplido todos a lo largo del trabajo:

- Mostrar paquetes e implementar posibles pipelines de análisis: A lo largo del capítulo llamado "Herramientas de TDA" se ha descrito la forma de uso y comprensión de tres softwares diferentes.
- Destacar los algoritmos más importantes: En el capítulo "Técnicas de TDA" se muestran tres de las principales técnicas usadas en el análisis topológico de datos.
- Explorar las características topológicas y sus invariantes: Este objetivo se ha cumplido a la par que el primero debido a que se han explicado las salidas de los softwares mientras se exlpicaba cómo usarlo.
- Enfatizar las diversas gráficas y diagramas específicos: Tras finalizar el trabajo considero que este objetivo y el anterior son similares, ya que estudiamos las cracterísticas en las gráficas o diagramas que producen los softwares.
- Revisar aplicaciones y problemas específicos: Uno de los objetivos más importantes a mi parecer era buscar aplicaciones en el mundo real para estas técnicas, para ello se han mostrado al menos dos usos de cada técnica.

5.2. Evaluación del proceso de desarrollo del trabajo

Considero que al ser un área aún muy joven y estar en desarrollo las posibilidades son muy amplias y ser capaces de aprovechar estos métodos y herramientas nos permitirá obtener perspectivas completamente innovadoras en el campo del análisis de datos. Combinar diferentes enfoques y el uso de estas nuevas tecnologías nos va a abrir nuevas formas de estudiar y entender los datos, lo que con ayuda del desarrollo tecnológico con el que contamos y el uso de IA generativa creo que aún no se puede medir la magnitud de estos métodos.

A nivel personal ha sido un trabajo muy enriquecedor me ha permitido estudiar una rama del estudio de datos que desconocía y para mí ha sido como un "iceberg" comencé estudiando las bases matemáticas de los métodos sin intención de sumergirme demasiado en ellas y centrarme en el software y cuando comencé a buscar información ví que no era tan fácil dar con lo que buscaba, hasta que fui enlazando artículos que me han permitido recopilar toda la información que he incluido en el trabajo. En cuanto a la parte de software solo he tratado ejemplos simples y previamente documentados ya que tan solo buscaba hacer un análisis de los softwares y explicar su funcionamiento y cómo interpretar los resultados, algo que no ha sido fácil ya que tengo una base muy ligera de análisis de datos. A pesar de ello he descubierto un campo que como he mencionado en el primer párrafo considero que tiene una proyección inmensa y el haber dedicado tiempo a su estudio y aplicación a pesar de que haya sido a baji nivel creo que me va a ayudar en un futuro a comprender los desarrollos que se hagan en este precioso área que es el análisis de datos.

5.3. Trabajo futuro

Queriendo establecer unas líneas futuras a este Trabajo de Fin de Grado, considero que como he mencionado esto es la punta del "iceberg", tan solo he sido capaz de sintetizar, descurbrir y probar ciertos softwares, aún quedan muchos por estudiar, entre ellos: giotto-tda[85] (Python), Dionysus[86] (C++ con bindings en Python), GUDHI[87] (C++ con interfaz en Python), Javaplex[88] (Java y Matlab), PHAT[89] (C++), Ripser[90] (C++), Ripser++[91] (C++), Rivet[92], UMAP[93] (Python) y Ayasdi[94] (nuevo SymphonyAI). Los cuales no he podido probar por falta de tiempo.

Sobre el actual trabajo considero que debería estudiar ejemplos más complejos para cada software, por ejemplo TTK tiene un ejemplo llamado "Tectonic Puzzle" el cual procesa un modelo geofísico bidimensional de la superficie terrestre para segmentarlo en función de las placas tectónicas por mencionar uno de los muchos ejemplos que tiene que aplican al mundo real que considero que es lo más atractivo de los datos o el software KepplerMapper nos permite hacer una detección de tráfico TOR cifrado con análisis de datos topológicos y de refuerzo. Y también hacer mi propia investigación utilizando estos softwares con potencial prácticamente desconocido.

5.4. Impacto

Cuando hablamos de impacto voy a valorar 6 apartados por separado: personal, empresarial, social, económico, medioambiental y cultural.

A nivel personal, enfocándolo desde un punto en el que este artículo sea leído por un posible usuario de estos softwares considero que aprender a utilizar estos softwares sería realmente útil para el análisis de grandes bases de datos lo que complementandose con otros métodos o herramientas tendría un gran valor a nivel profesional.

A nivel empresarial, el uso de estos softwares nos puede ayudar a obtener insights de los datos, lo que nos podría ayudar a la toma de decisiones, por ejemplo en el campo de marketing podrían hacerse segemntaciones de clientes y patrones de comportamiento. Por otra parte las empresas que aprovechen estos softwares podrían obtener ventaja competitiva e impulsar la innovación

A nivel social, el uso de KepplerMapper se se utiliza para el análisis de datasets médicos, permitiendonos mejorar diagnósticos y tratamientos. También tienen uso desde la biología hasta las ciencias sociales, lo que nos puede permitir encontrar nuevos descubrimientos o avances.

A nivel económico, el uso de dichos softwares no quieren de un desarrollo de un software propio lo cual asociado a la investigación es una ventaja. También el análisis es eficiente y está optimizado lo que reduce los recursoso y por tanto tendrán un rendimiento más económico

A nivel medioambiental, podemos analizar modelos climáticos y datos acerca de la biodiversidad por lo que podrían ser útiles para estudiar cambios me-

dioambientales o sus impactos, esto nos permitiría tomar decisiones sobre la sostenibilidad o la gestión ambiental.

A nivel cultural, se puede aplicar para preservación del arte y patrimonios culturales analizando patrones y estructuras en obras de arte, también pueden ayudar en política y estrategias culturales ya que nos pueden revelar patrones o tendencias de grupos.

Acerca del alineamiento con Objetivos de Desarrollo Sostenible (ODS) de la Agenda 2030 de la ONU considero que son relevantes:

- Objetivo 3: Garantizar una vida sana y promover el bienestar para todos en todas las edades, como he mencionado antes nos permitiría mejorar estudios de enfermedades, diagnósticos y tratamientos.
- Objetivo 4: Garantizar una educación inclusiva, equitativa y de calidad y promover oportunidades de aprendizaje durante toda la vida para todos, son herramientas valiosas para campos como la informática o biología, lo que mejoraría las competencias técnicas (4.4).
- Objetivo 9: Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación, por lo que he mencionado en el apartado de "Impacto".
- Objetivo 11: Lograr que las ciudades sean más inclusivas, seguras, resilientes y sostenibles, estos softwares nos pueden ayudar a planificar y gestionar ciudades de manera más sostenible.
- Objetivo 12: Garantizar modalidades de consumo y producción sostenibles, se pueden estudiar datos de producción y consumo para optimizar recursos mediante el estudio de patrones de consumo y produccion.
- Objetivo 13: Adoptar medidas urgentes para combatir el cambio climático y sus efectos, analizando datos climáticos y ambientales.
- Objetivo 15: Gestionar sosteniblemente los bosques, luchar contra la desertificación, detener e invertir la degradación de las tierras, detener la pérdida de biodiversidad, analizando datos de biodiversidad y ecosistemas.

Capítulo 6

Glosario

Agujero (ciclo 1-dimensional) Característica topológica que representa un ciclo cerrado en el espacio.

Aprendizaje no supervisado Tipo de aprendizaje automático para encontrar patrones en datos no etiquetados.

Bicontinuo Propiedad de una función que es continua y tiene una inversa continua.

Bootstrap Técnica estadística para estimar la distribución de un estadístico mediante muestreo con reemplazo.

Clúster basado en densidad Grupo de objetos de datos distribuidos en una región contigua de alta densidad.

Cluster Agrupación de objetos similares en conjuntos.

Componente conexa Parte de un espacio topológico en la que cualquier par de puntos puede conectarse por una curva.

Complejidad Medida de la dificultad de resolver un problema o describir una estructura.

Complejo simplicial Estructura matemática formada por simples (vértices, aristas, triángulos, etc.) que se utilizan para estudiar las propiedades topológicas de un espacio.

Conjuntos de nivel Regiones donde una función adquiere un valor particular.

Consistencia Propiedad de un estimador que converge al valor verdadero a medida que aumenta el tamaño de la muestra.

Cuadrícula cúbica dispersa Formato de entrada para representar estructuras de cuadrícula cúbica donde una gran fracción de los cubos están ausentes.

Cuencas de atracción Regiones del espacio de datos que convergen hacia un modo de densidad.

Densidad de probabilidad Distribución de la probabilidad de ocurrencia de una variable aleatoria continua.

Desplazamiento medio Algoritmo utilizado para encontrar modos de densidad en conjuntos de datos.

Detección de anomalías Identificación de patrones inusuales en los datos.

Diagramas de dispersión Gráficos que muestran la relación entre dos variables.

Distancia euclidiana Medida de distancia entre dos puntos en un espacio euclidiano.

Estimación consistente Propiedad de un estimador cuyos valores se aproximan al verdadero a medida que aumenta el tamaño de la muestra.

Estimación de modas Técnica para identificar los picos en la distribución de densidad de datos.

Estimación de ridge Método de regresión lineal que incluye una penalización para evitar el sobreajuste.

Filtración Proceso de añadir sucesivamente simples a un complejo simplicial para estudiar cambios en sus propiedades topológicas.

Firma topológica Conjunto de invariantes topológicos que caracterizan una forma o estructura.

Función de disimilitud Medida de la diferencia entre dos objetos en un espacio de características.

Función de Morse Función cuyos puntos críticos tienen una estructura específica.

Generadores n-dimensionales de homología Elementos que representan clases de homología en una dimensión específica.

Hesiana Matriz de segundas derivadas parciales de una función; describe la curvatura de una función en un punto.

Hiper-cubo Generalización de un cubo en dimensiones superiores.

Homología persistente Técnica que estudia las características topológicas de un espacio a través de diferentes escalas.

Invariancia topológica Propiedad de un objeto que no cambia bajo transformaciones continuas.

Intervalo de persistencia Par (b_i, d_i) que representa el tiempo de nacimiento (b_i) y el tiempo de muerte (d_i) de una característica topológica en un complejo simplicial.

Método no paramétrico Enfoque de modelado estadístico que no asume una forma específica para la distribución de los datos.

Método paramétrico Enfoque de modelado estadístico que asume una forma específica para la distribución de los datos.

Modos de densidad Puntos donde la densidad alcanza un máximo local.

Nervio del complejo simplicial Conjunto simplicial que captura la intersección de cubos o otros conjuntos.

Número de Betti Entero β_i que cuenta el número de i-ciclos independientes en un complejo simplicial.

Núcleos de grupo Puntos de alta densidad dentro de los grupos identificados por el agrupamiento de modas.

Parámetro desconocido Valor que influye en el comportamiento de una distribución estadística, pero cuyo valor exacto no se conoce.

Partición División de un conjunto de datos en subconjuntos más pequeños y distintos.

Perseus Software para calcular la homología persistente de un complejo simplicial.

Persistencia de características Duración de características topológicas a través de una filtración.

Pipeline de análisis Secuencia de pasos de procesamiento de datos y análisis.

Punto crítico Punto donde el gradiente de una función es cero.

Redes neuronales Modelos de aprendizaje automático inspirados en el cerebro humano.

Reducción de la dimensión no lineal Métodos para reducir la cantidad de variables en los datos sin asumir relaciones lineales.

Regularidad Propiedad de una función o conjunto de datos que satisface ciertas condiciones matemáticas.

Ruido de los datos Perturbaciones aleatorias en los datos que pueden afectar a los resultados de un análisis.

Script 'persdia' Script de Matlab para visualizar los intervalos de persistencia generados por Perseus.

Teoría topográfica Estudio de la forma y las características físicas de la superficie de la Tierra o de otros objetos.

Topología Rama de la matemática que estudia las propiedades de los espacios que permanecen invariables bajo deformaciones continuas.

Toplex cúbico disperso (scubtop) Formato de entrada específico utilizado en Perseus para representar cuadrículas cúbicas dispersas.

Bibliografía

[1] E. Topics, «Data Generated Per Day», *Exploding Topics Blog*, 2024. dirección: https://explodingtopics.com/blog/data-generated-per-day.

[2] TechJury, *How Much Data Is Created Every Day?*, 2024. dirección: https://techjury.net/blog/how-much-data-is-created-every-day/.

[3] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod y H. A. Harrington, «A roadmap for the computation of persistent homology», *EPJ Data Science*, vol. 6, págs. 1-38, 2017.

[4] P. Frosini, «Measuring shapes by size functions», *Proc. of SPIE*, vol. 1610, págs. 122-133, 1992.

[5] V. Robins, «Towards computing homology from finite approximations», Tesis doct., University of Colorado at Boulder, 1999.

[6] H. Edelsbrunner, D. Letscher y A. Zomorodian, «Topological persistence and simplification», *Discrete and Computational Geometry*, vol. 28, n.º 4, págs. 511-533, 2002.

[7] V. de Silva y G. Carlsson, «Topological estimation using witness complexes», en *Eurographics Symposium on Point-Based Graphics*, 2004, págs. 157-166.

[8] P.-T. Bremer, H. Edelsbrunner, B. Hamann y V. Pascucci, «A topological hierarchy for functions on triangulated surfaces», en *IEEE Visualization*, IEEE, 2004, págs. 597-604.

[9] G. Carlsson, «Topology and data», *Bulletin of the American Mathematical Society*, vol. 46, n.º 2, págs. 255-308, 2009.

[10] J. A. Perea, «A brief history of persistence», *Journal of Artificial Intelligence Research*, vol. 69, págs. 393-407, 2019.

[11] Universidad del País Vasco, *Topología*, Accedido el 29 de mayo de 2024. dirección: https://www.ehu.eus/~mtwmastm/sigma20.pdf.

[12] Universidad de Waterloo, *What is topology?*, Accedido el 29 de mayo de 2024. dirección: https://uwaterloo.ca/pure-mathematics/about-pure-math/what-is-pure-math/what-is-topology#:~:text=Topology%20studies%20properties%20of%20spaces,rubber%2C%20but%20cannot%20be%20broken.

[13] F. Chazal y B. Michel, «An introduction to topological data analysis: fundamental and practical aspects for data scientists», *Frontiers in artificial intelligence*, vol. 4, pág. 108, 2021.

[14] J. L. Edelsbrunner Herbert y Harer, *Computational Topology: An Introduction*. American Mathematical Soc., 2010.

[15] G. Singh, F. Memoli y G. Carlsson, «Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition», en *Eurographics Symposium on Point-Based Graphics*, M. Botsch, R. Pajarola, B. Chen y M. Zwicker, eds., The Eurographics Association, 2007, ISBN: 978-3-905673-51-7. DOI: 10.2312/SPBG/SPBG07/091-100.

[16] V. Nanda. «Perseus, the Persistent Homology Software». (), dirección: http://www.sas.upenn.edu/~vnanda/perseus.

[17] H.-P. Kriegel, P. Kröger, J. Sander y A. Zimek, «Density-based clustering», *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, n.º 3, págs. 231-240, 2011.

[18] J. W. Carmichael y R. S. Julius, «Finding Natural Clusters», *Systematic Biology*, vol. 17, n.º 2, págs. 144-150, jun. de 1968, ISSN: 1063-5157. DOI: 10.1093/sysbio/17.2.144. eprint: https://academic.oup.com/sysbio/article-pdf/17/2/144/4615930/17-2-144.pdf. dirección: https://doi.org/10.1093/sysbio/17.2.144.

[19] J. A. Hartigan, «Direct clustering of a data matrix», *Journal of the american statistical association*, vol. 67, n.º 337, págs. 123-129, 1972.

[20] R. A. Fisher, «The use of multiple measurements in taxonomic problems», *Annals of eugenics*, vol. 7, n.º 2, págs. 179-188, 1936.

[21] Esri, *Cómo funcionan los algoritmos de agrupamiento basados en densidad*, https://doc.arcgis.com/es/allsource/latest/analysis/geoprocessing-tools/spatial-statistics/how-density-based-clustering-works.htm.

[22] L. Wasserman, «Topological data analysis», *Annual Review of Statistics and Its Application*, vol. 5, págs. 501-532, 2018.

[23] Q. Chen y C. He, «Integrating clustering with level set method for piecewise constant Mumford-Shah model», *EURASIP Journal on Image and Video Processing*, vol. 2014, n.º 1, pág. 1, 2014. DOI: 10.1186/1687-5281-2014-1. dirección: https://doi.org/10.1186/1687-5281-2014-1.

[24] L. He y H. Wu, «Efficient Clustering of Brain Tumor Segments using Level-set Hybrid Machine Learning Algorithms», *Scalable Computing: Practice and Experience*, vol. 24, n.º 4, págs. 327-338, 2023. DOI: 10.12694/scpe.v24i4.2141. dirección: https://www.scpe.org/index.php/scpe/article/view/2141.

[25] P. Anderson y L. Kim, «Community Detection Using Level Set Clustering», *Social Network Analysis and Mining*, vol. 10, n.º 1, págs. 34-45, 2020.

[26] H. Wang y G. Lee, «Social Network Analysis with Level Sets», *IEEE Transactions on Network Science and Engineering*, vol. 8, n.º 3, págs. 567-578, 2021.

[27] D. Easley y J. Kleinberg, *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010, ISBN: 9780521195331.

[28] M. Gonzalez y C. Rivera, «Traffic Pattern Analysis Using Level Set Clustering», *Transportation Research Part C: Emerging Technologies*, vol. 105, págs. 123-134, 2019.

[29] W. Chen y L. Zhang, «Level Set Methods in Urban Traffic Management», *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, n.º 4, págs. 1567-1578, 2020.

[30] S. P. Hoogendoorn y P. H. L. Bovy, *Urban Traffic Networks: Dynamic Flow Modeling and Control*. Springer, 2001, ISBN: 9783540412090.

[31] X. Li y X. Guan, *Data-Driven Traffic Engineering*. Springer, 2016.

[32] Q. Cheng, X. Lu, Z. Liu, J. Huang y G. Cheng, «Spatial clustering with density-ordered tree», *Physica A: Statistical Mechanics and its Applications*, vol. 460, págs. 188-200, 2016.

[33] T. Preethi, *DBSCAN Algorithm: Density Based Spatial Clustering of Application with Noise*, Accessed: 2024-05-27, 2018. dirección: https://medium.com/@tpreethi/dbscan-algorithm-density-based-spatial-clustering-of-application-with-noise-a826538dcb42.

[34] G. Karypis, E. Han y V. Kumar, «A hierarchical clustering algorithm using dynamic modeling», 1999.

[35] G. Menardi y D. De Stefano, «Density-Based Clustering of Social Networks», *Journal of the Royal Statistical Society Series A: Statistics in Society*, vol. 185, n.º 3, págs. 1004-1029, mar. de 2022, ISSN: 0964-1998. DOI: 10.1111/rssa.12796. eprint: https://academic.oup.com/jrsssa/article-pdf/185/3/1004/49416014/jrsssa_185_3_1004.pdf. dirección: https://doi.org/10.1111/rssa.12796.

[36] C. de Datos, *El algoritmo de Louvain es uno de los algoritmos más utilizados para la detección de comunidades*, Accessed: 2024-05-27, 2023. dirección: https://cienciadedatos.net/documentos/pygml02-detecion-comunidades-grafos-redes-python.

[37] J. Kim, Y.-C. Chen, S. Balakrishnan, A. Rinaldo y L. Wasserman, «Statistical inference for cluster trees», *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[38] J. W. Milnor, *Morse theory*. Princeton university press, 1963.

[39] Y. Cheng, «Mean shift, mode seeking, and clustering», *IEEE transactions on pattern analysis and machine intelligence*, vol. 17, n.º 8, págs. 790-799, 1995.

[40] M. Azizyan, Y.-C. Chen, A. Singh y L. Wasserman, «Risk bounds for mode clustering», *arXiv preprint arXiv:1505.00482*, 2015.

[41] B. S. Everitt, *Cluster Analysis, 3rd edn., Edward Arnold*, 1993.

[42] K. Roeder, «Density estimation with confidence sets exemplified by superclusters and voids in the galaxies», *Journal of the American Statistical Association*, vol. 85, n.º 411, págs. 617-624, 1990.

[43] R. Wallace, «A modified Hough transform for lines», *Proc. CVPR, 1985*, págs. 665-667, 1985.

[44] J. Moody, «Race, school integration, and friendship segregation in America», *American journal of Sociology*, vol. 107, n.º 3, págs. 679-716, 2001.

[45] C.-h. Chen, W. Härdle, A. Unwin, M. A. Cox y T. F. Cox, *Multidimensional scaling*. Springer, 2008.

[46] P. T. Spellman, G. Sherlock, M. Q. Zhang et al., «Comprehensive identification of cell cycle–regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization», *Molecular biology of the cell*, vol. 9, n.º 12, págs. 3273-3297, 1998.

[47] O. Alter, P. O. Brown y D. Botstein, «Singular value decomposition for genome-wide expression data processing and modeling», *Proceedings of the National Academy of Sciences*, vol. 97, n.º 18, págs. 10 101-10 106, 2000.

[48] M. L. Whitfield, G. Sherlock, A. J. Saldanha et al., «Identification of genes periodically expressed in the human cell cycle and their expression in tumors», *Molecular biology of the cell*, vol. 13, n.º 6, págs. 1977-2000, 2002.

[49] D. Cohen-Steiner, H. Edelsbrunner y J. Harer, «Extending Persistence Using Poincaré and Lefschetz Duality», *Foundations of Computational Mathematics*, vol. 9, n.º 1, págs. 79-103, 2009. DOI: 10.1007/s10208-008-9027-z. dirección: https://doi.org/10.1007/s10208-008-9027-z.

[50] D. Cohen-Steiner, H. Edelsbrunner y J. Harer, «Stability of Persistence Diagrams», *Discrete & Computational Geometry*, vol. 37, n.º 1, págs. 103-120, 2007. DOI: 10.1007/s00454-006-1276-5. dirección: https://doi.org/10.1007/s00454-006-1276-5.

[51] P. Skraba y K. Turner, *Wasserstein Stability for Persistence Diagrams*, 2023. arXiv: 2006.16824 [math.AT].

[52] B. Murayama, M. Kobayashi, M. Aoki et al., «Characterizing reaction route map of realistic molecular reactions based on weight rank clique filtration of persistent homology», *Journal of Chemical Theory and Computation*, vol. 19, n.º 15, págs. 5007-5023, 2023.

[53] S. Biasotti, D. Giorgi, M. Spagnuolo y B. Falcidieno, «Reeb graphs for shape analysis and applications», *Theoretical computer science*, vol. 392, n.º 1-3, págs. 5-22, 2008.

[54] R. G. Miller, «Discussion: Projection Pursuit», *The Annals of Statistics*, vol. 13, n.º 2, págs. 510-513, 1985.

[55] P. J. Huber, «Projection pursuit», *The annals of Statistics*, págs. 435-475, 1985.

[56] D. F. Andrews y A. M. Herzberg, *Data: a collection of problems from many fields for the student and research worker*. Springer Science & Business Media, 2012.

[57] R. W. Sumner y J. Popović, «Deformation transfer for triangle meshes», *ACM Transactions on graphics (TOG)*, vol. 23, n.º 3, págs. 399-405, 2004.

[58] ISO/IEC, *ISO/IEC 14882:2020: Information Technology – Programming Languages – C++*, https://isocpp.org/std/the-standard, Accessed: 2024-04-20, 2020.

[59] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet y M. Michaux, «The Topology ToolKit», *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2017, https://topology-tool-kit.github.io/.

[60] Burlington School District, *Burlington School District*, https://www.bsd.org/, Accessed: 2024-04-22.

[61] Arizona State University, *Arizona State University*, https://www.asu.edu, Accessed: 2024-04-20.

[62] CNRS, *Centre National de la Recherche Scientifique (CNRS)*, https://www.cnrs.fr, Accessed: 2024-04-20.

[63] Heidelberg University, *Heidelberg University*, https://www.uni-heidelberg.de, Accessed: 2024-04-20.

[64] INRIA, *INRIA*, https://www.inria.fr, Accessed: 2024-04-20.

[65] Linkoping University, *Linkoping University*, https://www.liu.se, Accessed: 2024-04-20.

[66] Los Alamos National Laboratory, *Los Alamos National Laboratory*, https://www.lanl.gov, Accessed: 2024-04-20.

[67] Sorbonne Universite, *Sorbonne Universite*, https://www.sorbonne-universite.fr, Accessed: 2024-04-20.

[68] TU Kaiserslautern, *Technische Universität Kaiserslautern (TU Kaiserslautern)*, https://www.uni-kl.de, Accessed: 2024-04-20.

[69] University of Arizona, *University of Arizona*, https://www.arizona.edu, Accessed: 2024-04-20.

[70] University of Leeds, *University of Leeds*, https://www.leeds.ac.uk, Accessed: 2024-04-20.

[71] Zuse Institute Berlin, *Zuse Institute Berlin (ZIB)*, https://www.zib.de, Accessed: 2024-04-20.

[72] Kitware, *Kitware*, https://www.kitware.com, Accessed: 2024-04-20.

[73] Total, *Total*, https://www.total.com, Accessed: 2024-04-20.

[74] ShapeShift3D, *ShapeShift3D*, https://www.shapeshift3d.com, Accessed: 2024-04-20.

[75] Python Software Foundation, *Python Programming Language – Official Website*, https://www.python.org/, Accessed: 2024-04-20.

[76] J. Ahrens, B. Geveci, C. Law, C. Hansen y C. Johnson, «36-paraview: An end-user tool for large-data visualization», *The visualization handbook*, vol. 717, págs. 50 038-1, 2005.

[77] *TopologyToolKit on Anaconda*, https://anaconda.org/conda-forge/topologytoolkit, Accessed: 05/05/2024.

[78] Canonical Ltd., *Ubuntu – The leading operating system for PCs, IoT devices, servers and the cloud*, https://ubuntu.com/, Accessed: 2024-04-20.

[79] Topology ToolKit, *Topology ToolKit – Downloads*, https://topology-tool-kit.github.io/downloads.html, Accessed: 2024-05-22.

[80] F. Pedregosa, G. Varoquaux, A. Gramfort et al., «Scikit-learn: Machine Learning in Python», *Journal of Machine Learning Research*, vol. 12, págs. 2825-2830, 2011.

[81] UCI Machine Learning Repository, *Wisconsin Breast Cancer Dataset*, https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data, 2016.

[82] UCI Machine Learning Repository, *Breast Cancer Wisconsin (Diagnostic) Data Set*, https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data, 2016.

[83] T. M. Inc., *MATLAB version: 9.13.0 (R2022b)*, Natick, Massachusetts, United States, 2022. dirección: https://www.mathworks.com.

[84] A. Hatcher, *Algebraic topology*. 2005.

[85] T. giotto-tda contributors, *giotto-tda: A Python library for Topological Data Analysis*, https://giotto-ai.github.io/gtda/, Accedido: 2024-05-29, 2020.

[86] D. Morozov, *Dionysus*, https://mrzv.org/software/dionysus2/, Accedido: 2024-05-29.

[87] T. G. Project, *GUDHI: A Geometric Understanding of Higher Dimensional Geometry Library*, https://gudhi.inria.fr/, Accedido: 2024-05-29, 2021.

[88] S. University, *Javaplex*, http://appliedtopology.github.io/javaplex/, Accedido: 2024-05-29.

[89] J. R. Ulrich Bauer Michael Kerber, *PHAT: Persistent Homology Algorithm Toolbox*, https://github.com/phat/phat, Accedido: 2024-05-29, 2014.

[90] U. Bauer, *Ripser: Efficient computation of Vietoris-Rips persistence barcodes*, https://github.com/Ripser/ripser, Accedido: 2024-05-29.

[91] e. a. Zhe Chen, *Ripser++: A GPU-accelerated version of Ripser*, https://github.com/simonzhang00/ripser-plusplus, Accedido: 2024-05-29.

[92] e. a. Brittany Terese Fasy, *Rivet: Rank Invariant Visualization and Exploration Tool*, https://rivet.online/, Accedido: 2024-05-29.

[93] J. M. Leland McInnes John Healy, *UMAP: Uniform Manifold Approximation and Projection*, https://umap-learn.readthedocs.io/, Accedido: 2024-05-29, 2018.

[94] *Ayasdi*, https://www.ayasdi.com, Accedido: 2024-05-29.

Anexos

Apéndice A

Primer anexo

Código en Python del ejemplo de ”Morse Persistence” utilizado para mostrar el uso del software llamado ”Topology ToolKit”.

```
#!/usr/bin/env python

from paraview.simple import *

# create a new 'Plane'
plane1 = Plane()
plane1.XResolution = 300
plane1.YResolution = 300

# create a new 'Tetrahedralize'
tetrahedralize2 = Tetrahedralize(Input=plane1)

# create a new 'Random Attributes'
randomAttributes1 = RandomAttributes(Input=tetrahedralize2)
randomAttributes1.DataType = "Float"
randomAttributes1.ComponentRange = [0.0, 1.0]
randomAttributes1.GeneratePointScalars = 1

# create a new 'TTK ScalarFieldSmoother'
tTKScalarFieldSmoother1 =
    ↪ TTKScalarFieldSmoother(Input=randomAttributes1)
tTKScalarFieldSmoother1.ScalarField = ["POINTS", "RandomPointScalars"]
tTKScalarFieldSmoother1.IterationNumber = 7

# create a new 'Calculator'
sine = Calculator(Input=tTKScalarFieldSmoother1)
sine.ResultArrayName = "Sine"
sine.Function = "sin(20*coordsX+1.5)+sin(20*coordsY+1.5)"

# create a new 'Calculator'
distanceField = Calculator(Input=sine)
distanceField.ResultArrayName = "DistanceField"
distanceField.Function = "-sqrt(coordsX*coordsX+coordsY*coordsY)"

```

```
# create a new 'Calculator'
calculator1 = Calculator(Input=distanceField)
calculator1.ResultArrayName = "Blend"
calculator1.Function = "Sine+5*DistanceField+5*RandomPointScalars"

# create a new 'Extract Surface'
extractSurface6 = ExtractSurface(Input=calculator1)

# create a new 'Warp By Scalar'
warpByScalar1 = WarpByScalar(Input=extractSurface6)
warpByScalar1.Scalars = ["POINTS", "Blend"]
warpByScalar1.ScaleFactor = 0.05

# create a new 'TTK PersistenceDiagram'
tTKPersistenceDiagram1 = TTKPersistenceDiagram(Input=warpByScalar1)
tTKPersistenceDiagram1.ScalarField = ["POINTS", "Blend"]
tTKPersistenceDiagram1.IgnoreBoundary = False

# create a new 'TTK PersistenceCurve'
tTKPersistenceCurve1 =
    ↪ TTKPersistenceCurve(Input=tTKPersistenceDiagram1)

# create a new 'Threshold'
threshold1 = Threshold(Input=tTKPersistenceDiagram1)
threshold1.Scalars = ["CELLS", "PairIdentifier"]
threshold1.ThresholdMethod = "Between"
threshold1.LowerThreshold = 0.0
threshold1.UpperThreshold = 100000.0

# create a new 'Threshold'
persistenceThreshold = Threshold(Input=threshold1)
persistenceThreshold.Scalars = ["CELLS", "Persistence"]
persistenceThreshold.ThresholdMethod = "Between"
persistenceThreshold.LowerThreshold = 0.7
persistenceThreshold.UpperThreshold = 10000.0

# create a new 'Tetrehedralize'
tetrahedralize1 = Tetrahedralize(Input=warpByScalar1)

# create a new 'TTK TopologicalSimplification'
tTKTopologicalSimplification1 = TTKTopologicalSimplification(
    Domain=tetrahedralize1, Constraints=persistenceThreshold
)
tTKTopologicalSimplification1.ScalarField = ["POINTS", "Blend"]

# create a new 'TTK MorseSmaleComplex'
tTKMorseSmaleComplex1 =
    ↪ TTKMorseSmaleComplex(Input=tTKTopologicalSimplification1)
tTKMorseSmaleComplex1.ScalarField = ["POINTS", "Blend"]

# save the ouput
SaveData("PersistenceDiagram.vtu", tTKPersistenceDiagram1)
SaveData("PersistenceCurve.csv", OutputPort(tTKPersistenceCurve1, 3))
```

```
85 SaveData("MorseComplexeCriticalPoints.vtp",
       ↪ OutputPort(tTKMorseSmaleComplex1, 0))
86 SaveData("MorseComplexe1Separatrices.vtp",
       ↪ OutputPort(tTKMorseSmaleComplex1, 1))
87 SaveData("MorseComplexeSegmentation.vtu",
       ↪ OutputPort(tTKMorseSmaleComplex1, 3))
```

Apéndice B

Segundo anexo

Código en Python del ejemplo usado para demostrar el funcionamiento del software "KepplerMapper".

```
import sys

try:
    import pandas as pd
except ImportError as e:
    print(
        "pandas is required for this example. Please install with `pip
            ↪ install pandas` and then try again."
    )
    sys.exit()

import numpy as np
import kmapper as km
import sklearn
from sklearn import ensemble

# For data we use the Wisconsin Breast Cancer Dataset
# Via:
df = pd.read_csv("data/breast-cancer.csv")
feature_names = [c for c in df.columns if c not in ["id", "diagnosis"]]
df["diagnosis"] = df["diagnosis"].apply(lambda x: 1 if x == "M" else 0)
X = np.array(df[feature_names].fillna(0))  # quick and dirty imputation
y = np.array(df["diagnosis"])

# We create a custom 1-D lens with Isolation Forest
model = ensemble.IsolationForest(random_state=1729)
model.fit(X)
lens1 = model.decision_function(X).reshape((X.shape[0], 1))

# We create another 1-D lens with L2-norm
mapper = km.KeplerMapper(verbose=3)
lens2 = mapper.fit_transform(X, projection="l2norm")

# Combine both lenses to create a 2-D [Isolation Forest, L^2-Norm] lens
```

```
lens = np.c_[lens1, lens2]

# Create the simplicial complex
graph = mapper.map(
    lens,
    X,
    cover=km.Cover(n_cubes=15, perc_overlap=0.4),
    clusterer=sklearn.cluster.KMeans(n_clusters=2,
        ↪ random_state=1618033),
)

# Visualization
mapper.visualize(
    graph,
    path_html="output/breast-cancer.html",
    title="Wisconsin Breast Cancer Dataset",
    custom_tooltips=y,
)


# Visualization with multiple color functions
mapper.visualize(
    graph,
    path_html="output/breast-cancer-multiple-color-functions.html",
    title="Wisconsin Breast Cancer Dataset",
    custom_tooltips=y,
    color_values=lens,
    color_function_name=["Isolation Forest", "L2-norm"],
)


# Visualization with multiple node color functions
mapper.visualize(
    graph,
    path_html="output/breast-cancer-multiple-node-color-functions.html",
    title="Wisconsin Breast Cancer Dataset",
    custom_tooltips=y,
    node_color_function=["mean", "std", "median", "max"],
)

# Visualization showing both multiple color functions, and also
    ↪ multiple node color functions
mapper.visualize(
    graph,
    path_html="output/breast-cancer-multiple-color-functions-and-
    multiple-node-color-functions.html",
    title="Wisconsin Breast Cancer Dataset",
    custom_tooltips=y,
    color_values=lens,
    color_function_name=["Isolation Forest", "L2-norm"],
    node_color_function=["mean", "std", "median", "max"],
)

```

```
85
86 import matplotlib.pyplot as plt
87
88 km.draw_matplotlib(graph)
89 plt.show()
```

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Mon Jun 03 17:54:00 CEST 2024
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)