TESINA DE LICENCIATURA

TÍTULO: Agrupamiento dinámico para flujos de datos no estacionarios

AUTORES: Camila Onofri

DIRECTORA: Dra. Laura Lanzarini

CODIRECTOR: Esp. César Estrebou

CARRERA: Licenciatura en Sistemas

Resumen

El avance tecnológico permite generan datos a alta velocidad que deben procesarse sin ser almacenados. Estos datos analizados en línea son considerados flujos y su procesamiento en tiempo real debe lidiar con problemas tales como la capacidad limitada de memoria, el escaneo único de datos y la diferenciación entre anomalías y deriva de concepto. En esta tesina se estudian, implementan y aplican distintas técnicas de agrupamiento dinámico, principalmente aquellas abocadas al análisis dinámico de flujos de datos no estacionarios cuya distribución varía en el tiempo. Además, se emplean estas técnicas para la resolución de un problema del mundo real.

Palabras Clave

- *Agrupamiento dinámico.*
- *Flujos de datos*
- *Datos no estacionarios.*

Conclusiones

- *Esta tesina describe brevemente los algoritmos de agrupamiento convencionales y expone sus limitaciones al agrupar datos provenientes de un flujo.*
- *Como solución al punto anterior, se propone el uso de algoritmos de clustering dinámico a fin de realizar el agrupamiento de forma incremental.*
- *El agrupamiento se mantiene actualizado utilizando un factor de olvido.*
- *Los resultados obtenidos fueron satisfactorios tanto para los datos de repositorio como para el caso real.*

Trabajos Realizados

- *Se analizaron exhaustivamente distintos métodos de agrupamiento convencionales y se describieron algoritmos de agrupamiento dinámico basados en distancia y densidad.*
- *Se realizó un análisis comparativo de distintas métricas para evaluar particionamientos*
- *Se implementó el algoritmo Dyclee introduciendo variantes en sus hiperparámetros.*
- *Se analizaron datos de repositorio. Como problema real a resolver, se analizaron las trayectorias de taxis de Suecia durante un período dado.*

Trabajos Futuros

- *En relación al algoritmo DyClee, aún está pendiente la implementación de un módulo que, en base a un conjunto inicial de datos de entrada, determine cuál es la configuración inicial correcta de los hiperparámetros.*
- *En cuanto al clustering dinámico en general, sería interesante contar con una estrategia capaz de seleccionar los atributos más relevantes para el agrupamiento con capacidad de adaptarse a los cambios del flujo.*

Fecha de la presentación: Agosto 2023

FACULTAD DE INFORMATICA

UNIVERSIDAD
NACIONAL
DE LA PLATA

Agrupamiento dinámico para flujos de datos no estacionarios

Camila Onofri

Tesis presentada para obtener el grado de Licenciada en Sistemas

Directora: Prof. Laura Lanzarini
Codirector: Prof. César Estrebou

La Plata, 15 de agosto de 2023

Resumen

En la era digital en la que vivimos, estamos rodeados de aplicaciones y dispositivos de hardware que generan datos a alta velocidad, los cuáles crecen en términos exponenciales. Estos datos continuamente generados por diversas fuentes pueden no almacenarse, ya sea por un problema de espacio, o porque no es necesario hacerlo debido a que el patrón a detectar varía en el tiempo. En este caso, cuando los datos son procesados en línea, se denominan flujos de datos. Para la minería de datos tradicional, el conjunto de datos generalmente es de naturaleza estática y puede ser inspeccionado varias veces durante su análisis. Sin embargo, cuando se trata de flujos de datos, el procesamiento de los mismos se convierte en un nuevo desafío, ya que se deben satisfacer las demandas que implica el análisis en tiempo real: capacidad limitada de memoria, escaneo único de datos, y diferenciación entre anomalías y deriva de concepto. Es por esto, que las técnicas de minería convencionales son ineficientes para estos casos, ya que el comportamiento de los datos en sí se modifica.

En esta tesina se estudian, implementan y aplican distintas técnicas de agrupamiento, principalmente aquellas abocadas al análisis dinámico de flujos de datos no estacionarios, cuya distribución varía en el tiempo. Además, se emplean estas técnicas para la resolución de un problema del mundo real. Los resultados que se obtienen, son utilizados para analizar las modificaciones que ocurren en los agrupamientos a lo largo de todo el proceso. Así, siguiendo el lineamiento del agrupamiento dinámico, se mejora la capacidad descriptiva de los modelos, ya que se mantienen correctamente actualizados sin necesidad de recurrir a datos históricos, independientemente de la variabilidad que presenten los datos.

Palabras clave: Agrupamiento dinámico, flujos de datos, datos no estacionarios

Índice general

Índice de figuras

Índice de algoritmos

Glosario de términos y acrónimos

border	descriptor límite.
CF	vector de características.
core	descriptor central.
DB	Davies-Bouldin.
HCA	algoritmos de agrupamiento jerárquicos..
hiperparámetros	valores que deben ser definidos previamente para controlar el proceso de aprendizaje. La diferencia con los parámetros regulares radica en que estos últimos se derivan del proceso de entrenamiento, es decir, son aprendidos.
IoT	Internet de las cosas.
MST	árbol de expansión mínima.
outlier	valor extremo, por fuera del rango esperado, diferente del resto de los datos.
timestamps	marcas de tiempo utilizadas para hacer un seguimiento y tener una noción de orden de llegada. Indican una fecha y hora.

Capítulo 1

Introducción

1.1. Motivación

El avance tecnológico ha facilitado el acceso a la información e incrementado la capacidad de procesamiento. Esto ha impulsado el estudio y desarrollo de estrategias capaces de analizar la información disponible con el objeto de comprender su organización y generalizar los procesos utilizados, entendiendo que constituyen una herramienta sumamente útil para la toma de decisiones. La *Minería de Datos* es el área de la informática que reúne distintas técnicas capaces de resolver estas tareas, extrayendo relaciones válidas, novedosas, potencialmente útiles y comprensibles a partir de los datos.

Cuando se analiza información, existen básicamente dos tipos de tareas a resolver: descriptivas y predictivas según si interesa brindar información referida a cómo se distribuyen y organizan los datos, o predecir un nuevo resultado en base a experiencias anteriores, respectivamente.

Esta tesina se enmarca en la construcción de modelos capaces de resolver tareas descriptivas, es decir, modelos que buscan establecer similitudes y diferencias entre las distintas situaciones que pueden ocurrir, a fin de identificar cuáles son las características más importantes que deben ser tenidas en cuenta a la hora de explicar el comportamiento de un proceso.

Las técnicas de *agrupamiento* son las más utilizadas en estos casos.

Si se analiza la evolución de este tipo de técnicas puede verse que en un principio, los modelos han operado buscando establecer similitudes en un conjunto de datos conocido a priori. Utilizando el enfoque tradicional, dicho conjunto puede ser analizado todas las veces que sea necesario hasta determinar las características más representativas del problema. Esta forma de trabajo, si bien sigue siendo válida en numerosas situaciones, presenta problemas cuando deben procesarse volúmenes de información sumamente grandes. En estos casos puede

ocurrir que no se disponga de espacio suficiente para su almacenamiento o incluso que no resulte conveniente guardarlos, por la velocidad con la que la información se desactualiza.

Por lo antes expuesto, esta tesina se enmarca en las técnicas de agrupamiento para *flujos de datos*, es decir, técnicas capaces de establecer similitudes entre los datos analizándolos una única vez. Por lo tanto, la información no es almacenada sino que es procesada inmediatamente al momento de arribar.

Procesar un flujo de datos implica realizar modificaciones sobre el modelo de manera automática porque, como se dijo previamente, ya no se dispone de los datos originales para efectuar modificaciones, sino que deben ir efectuándose registros más genéricos a medida que se ingresa la información. Este tipo de entrada se ajusta perfectamente al procesamiento de *información temporal*.

1.2. Objetivos

El objetivo general de esta tesina es realizar una revisión de las técnicas de agrupamiento dinámico existentes en la literatura, haciendo énfasis en la modalidad de recolección de los datos, en la cantidad de veces que los mismos pueden ser utilizados, y en los parámetros de configuración que debe indicar el usuario.

El objetivo específico de esta tesina se basa en estudiar, implementar y aplicar distintas técnicas de agrupamiento dinámico para flujos de datos *no estacionarios*: aquellos cuya distribución varía en el tiempo. Para esto, se usaron conjuntos de datos artificiales, al igual que un conjunto de datos real, empleado para el monitoreo de taxis en Suecia durante octubre y noviembre de 2018. Los resultados obtenidos, permitieron analizar las modificaciones que ocurrieron en los agrupamientos a lo largo del proceso. Así, se mejoró la capacidad descriptiva del modelo, ya que de esta manera, se logró que se mantenga correctamente actualizado, sin necesidad de recurrir a datos históricos e independientemente de la variabilidad que presenten.

1.3. Resultados obtenidos

El contenido de la tesina, se puede separar en dos módulos principales:

- **Estudio y análisis de técnicas de Agrupamiento**
  - Estudio del contexto general de las técnicas de agrupamiento como herramientas para resolver tareas descriptivas.
  - Estudio de distintas técnicas de agrupamiento dinámico, basadas en distancia y/o densidad. Se consideraron soluciones tanto para datos almacenados como para flujos de datos no estacionarios.

  - Análisis comparativo de las distintas técnicas de agrupamiento.
  - Análisis comparativo de distintas métricas para evaluar particionamientos generados por diversos algoritmos, y posterior elección de la métrica a utilizar para los casos de prueba.

- **Resolución de casos de prueba concretos**

  - Se utilizaron conjuntos de datos disponibles en varios repositorios. Además, se construyeron visualizaciones que dejaron en evidencia la evolución de los distintos agrupamientos.
  - Utilizando información pública de coordenadas de taxis en Suecia, se analizó la evolución del recorrido de los vehículos a lo largo del tiempo, pudiendo identificar las zonas con mayor tráfico, en cada momento.

Cabe mencionar que, el método estudiado y aplicado en el problema real de agrupamiento de vehículos, podría emplearse para otros problemas de tráfico en donde se requiera agrupar posiciones por ejemplo de peatones, o bien en escenarios totalmente diferentes, como ser el análisis de datos emitidos por sensores de un dispositivo inteligente, la búsqueda de patrones de degradación de maquinaria dentro del ámbito de la industria, o sondeos de opiniones o tópicos en sitios web y redes sociales.

En lo que se refiere a los desarrollos realizados en el marco de esta tesina, se encuentran:

- Biblioteca de funciones que permite parametrizar la información a procesar y la técnica a utilizar para generar el modelo, trabajando con conjuntos de datos de 2 y 3 dimensiones, a fin de construir representaciones gráficas a lo largo del proceso de agrupamiento en un momento dato.
- Implementación de una técnica de agrupamiento dinámico para flujos de datos no estacionarios dada a conocer recientemente en la literatura.
- Construcción de gráficos que representan las modificaciones de los agrupamientos a medida que se fue procesando el flujo de datos, para los casos de prueba analizados.

El resultado central de esta tesina es el análisis de las técnicas de agrupamiento dinámico para flujos de datos no estacionarios. Se dispone entonces de una biblioteca de código capaz de ejemplificar el uso de este tipo de técnicas distinguiéndolas del enfoque convencional que procesa la misma información varias veces. Esto constituye un avance importante ya que se trata de soluciones que, sin requerir almacenamiento adicional (porque procesan los datos inmediatamente), generan un modelo que se mantiene permanentemente actualizado.

A su vez se hizo énfasis en el estudio de las modificaciones que se van produciendo sobre el agrupamiento a lo largo de todo el proceso, ya que esto genera información muy valiosa sobre cómo se va modificando la entrada de datos.

1.4. Estructura del documento

El documento se organiza de la siguiente manera. En el **Capítulo 2**, se realiza un análisis de las distintas técnicas de agrupamiento tradicionales. Se explican los conceptos básicos relacionados al agrupamiento en general. Luego, se hace una categorización según cómo se reparten los ejemplos en los grupos: se hace referencia al *agrupamiento partitivo* y *jerárquico*. También, se hace foco en diferenciar los algoritmos acorde al concepto utilizado a la hora de crear los grupos: se habla de *distancia* y *densidad*.

En el **Capítulo 3**, se introducen los*flujos de datos*. Se debaten los distintos desafíos que se presentan al trabajar con éstos, se explica por qué es necesario cambiar el enfoque tradicional para procesarlos, y se detallan los principales métodos utilizados hasta el momento para agruparlos. Este capítulo, introduce a DyClee, un algoritmo de agrupamiento dinámico que se dio a conocer recientemente en la literatura, el cual fue estudiado e implementado. A su vez, en esta parte del documento se realiza un análisis de las *métricas* útiles para evaluar el desempeño de los distintos algoritmos de agrupamiento, y el por qué de la métrica elegida para las pruebas subsiguientes.

En el **Capítulo 4**, se presentan los resultados obtenidos. Se realiza una *comparativa* entre los diferentes algoritmos presentados y detallados en las secciones anteriores, usando conjuntos de datos artificiales. Para esto, se explica particularmente cómo afecta a los resultados finales de cada algoritmo la configuración de los distintos valores que deben ser definidos previamente para controlar el proceso de aprendizaje. La diferencia con los parámetros regulares radica en que estos últimos se derivan del proceso de entrenamiento, es decir, son aprendidos (hiperparámetros). A su vez, se muestra el funcionamiento de DyClee a la hora de resolver un problema real: el algoritmo es aplicado para procesar el monitoreo de taxis en Suecia durante octubre y noviembre de 2018.

Por último, el **Capítulo 5** concluye el documento y presenta las direcciones a seguir para futuras investigaciones.

Capítulo 2

Técnicas de agrupamiento

El amplio uso del *agrupamiento de datos* como herramienta exploratoria, surge del hecho de que requiere pocas presunciones sobre los datos investigados [1], caracterizándose como una tarea *no supervisada* desde el punto de vista del aprendizaje automático [2]. En este capítulo, se presenta una revisión completa del *agrupamiento de datos*. Se señalan principalmente sus conceptos básicos en la **Sección 2.1**. Luego, en la **Sección 2.2**, se realiza una taxonomía para explicar así las variantes principales a la hora de elegir el paradigma a seguir. Todo esto, siguiendo el *enfoque tradicional*, es decir, asumiendo que se dispone del conjunto completo de ejemplos al momento de hacer el agrupamiento. Finalmente, en la **Sección 2.3**, se analizan en detalle algunos de los algoritmos de agrupamiento más conocidos, los cuales serán utilizados a lo largo de la tesina.

2.1. Conceptos básicos

Las técnicas de aprendizaje automático adaptan su respuesta en función de los datos de entrada. Los algoritmos de aprendizaje automático crean un *modelo*, el cual una vez entrenado, al proporcionarle una entrada, genera una salida. Dicho entrenamiento, es el proceso por el cuál se detectan patrones subyacentes en el conjunto de datos provistos, utilizados entonces para luego modelar la realidad. Este conjunto de datos, debe ser lo suficientemente representativo para que más tarde el modelo, al ser utilizado, pueda generalizar el conocimiento aprendido, y así procesar elementos nuevos, no vistos durante la etapa de entrenamiento, o bien expresar las relaciones actuales y eventualmente adaptarse a los cambios. El enfoque tradicional de aprendizaje automático, plantea entrenar el algoritmo elegido de forma *offline*, haciendo varias pasadas por el conjunto de entrenamiento, para así obtener un modelo, que pueda usarse más adelante de manera *online*, procesando ejemplos nuevos.

Cuando se analizan datos, existen básicamente dos tipos de tareas a resolver: *descriptivas* o *predictivas*. Si el conocimiento disponible permite etiquetar cada objeto de los datos de entrenamiento con una clase que representa cierto concepto, el objetivo es aprender la asociación entre los datos de entrada y la salida esperada. Esto se conoce como *aprendizaje supervisado*. En estos casos, interesa predecir un nuevo resultado en base a experiencias anteriores. Por otro lado, si el objetivo es, dado un conjunto de datos de entrenamiento, brindar información referida a cómo se distribuyen y organizan los elementos, esto se conoce como *aprendizaje no supervisado*. Para ello, las técnicas de *agrupamiento* son las más utilizadas.

Un concepto fundamental en lo que refiere a las técnicas de agrupamiento es el término de *cluster*. Aunque no existe una definición globalmente aceptada en la literatura [1], un *cluster* representa a un grupo o categoría a la que pertenecen datos de entrada, abstrayendo su estructura subyacente. El resultado de aplicar una técnica de agrupamiento es la organización de datos en un conjunto finito de estas categorías, ya sea agrupando objetos en una sola partición o construyendo una jerarquía de particiones para describir datos de acuerdo con similitudes o relaciones entre sus objetos [1][3][4]. Agrupar, consiste en establecer similitudes y diferencias entre las distintas situaciones que pueden ocurrir, a fin de identificar cuáles son las características más importantes que deben ser tenidas en cuenta a la hora de explicar el comportamiento de cierto proceso. El objetivo, es que los objetos dentro de un mismo grupo sean similares entre sí, y diferentes a objetos en otros grupos, siguiendo determinado criterio. Se espera que surjan relaciones previamente desconocidas de los datos como resultado del proceso de agrupación. Luego, la asignación de un concepto a cada grupo se deja como tarea a expertos en el dominio. Un *cluster*, entonces, puede definirse como un grupo de muestras similares, diferentes a las muestras pertenecientes a los demás *clusters* [5].

La Figura 2.1, detalla el procedimiento que se sigue cuando se trabaja bajo el enfoque de agrupamiento tradicional.

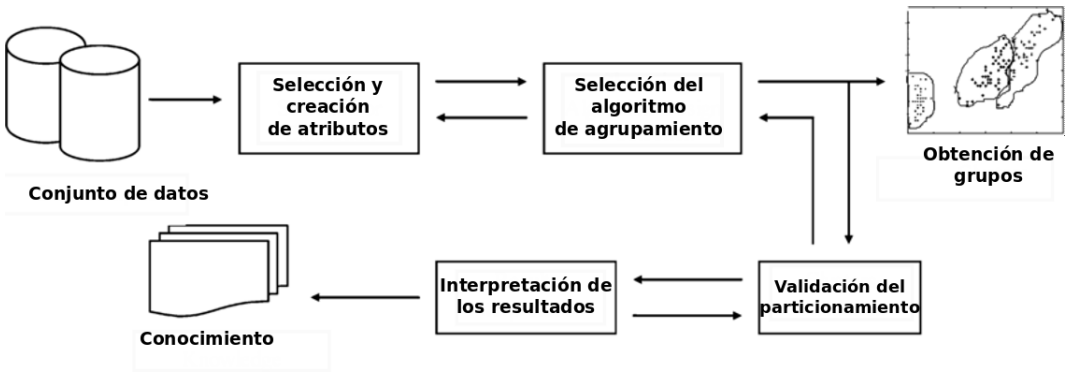

FIGURA 2.1: Procedimiento para agrupar datos siguiendo el enfoque tradicional [6].

Este enfoque, consiste en cuatro simples pasos, permitiendo a la salida del algoritmo cambiar la manera en que se ejecutan los pasos iniciales. Primero, los ejemplos se representan de alguna manera a través de la selección o extracción de atributos, proceso mediante el cual se

identifican las características más significativas de los ejemplos originales, pudiendo también producir nuevas características prominentes, respectivamente. El segundo paso, consiste en definir el algoritmo a utilizar y la medida de similitud que se evaluará a la hora considerar que tan parecidos son los ejemplos con aquellos representantes de los grupos que se van formando. Una de las medidas de similitud más utilizadas es la *distancia Euclídea*. El tercer paso, consiste efectivamente en la realización del agrupamiento. El cuarto paso, comprende la abstracción de los datos, es decir, la realización de una representación compacta de cada grupo obtenido, detectando qué es lo que identifica a cada uno. Típicamente, lo que se hace es determinar un representante de cada grupo: el *centroide*. Sin embargo, esto no es siempre útil ya que sólo sirve para describir grupos globulares. Se detallará este tópico más adelante en el documento. El último paso, refiere a la evaluación de los resultados obtenidos. Este paso es muy importante, ya que hay que tener en cuenta que cualquier algoritmo de agrupamiento producirá grupos, incluso aunque los datos no presenten ningún agrupamiento natural.

2.2. Medidas de similitud

Para poder determinar que los ejemplos en un grupo son más similares entre sí, en comparación con ejemplos de otros grupos, se necesita utilizar una *medida de similitud*. La forma más directa de definir la medida de similitud entre dos ejemplos es midiendo la distancia entre ellos. Una medida de distancia debería ser simétrica, es decir, para dos ejemplos $p, q \in R^n$, $d(p, q) = d(q, p)$. Además, el valor mínimo (idealmente cero) debería obtenerse cuando se calcula la distancia entre dos ejemplos idénticos. Las distancias más usadas para datos numéricos $p, q \in R^n$ son aquellas basadas en la *distancia Minkowski*, la cual está dada por la siguiente ecuación:

$$d\left(p, q\right) = \left(\sum_{i=1}^{n} |p_i - q_i|^x\right)^{\frac{1}{x}} \tag{2.1}$$

Estas distancias, son las normas $L1$, $L2$ y $L\infty$, y cada una define un valor específico para x dentro de la fórmula anteriormente expresada. Se procederá a detallar cada una y a explicar cómo se comportan.

2.2.1. Norma L2

La norma $L2$, corresponde a la *distancia Euclídea*, y viene definida por la siguiente fórmula:

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}. \tag{2.2}$$

Ésta, es la más simple de comprender ya que se deduce del teorema de Pitágoras, identificándose como la distancia 'ordinaria' entre puntos en el espacio.

Si bien se trata de la medida de similitud más popular y mayormente usada, cuando se agrupan datos de alta dimensionalidad utilizando la distancia Euclídea, generalmente se dificulta conseguir eficiencia. Esto se debe a que el comportamiento de las medidas de distancia cambia cuando se trabaja en un espacio de altas dimensiones [7]. En particular, al utilizar la norma $L2$ en estos espacios, la distancia entre los ejemplos aumenta con el aumento de la dimensionalidad dado que se incrementa la cantidad de términos de la sumatoria de la ecuación (2.2).Por lo tanto, los conceptos de *cercanía* y *lejanía* se debilitan: todos los objetos tienden a *alejarse*, perjudicando la determinación de similitudes [8]. Por otra parte, el atributo de mayor escala tiende a dominar los demás atributos. Esto último, puede resolverse normalizando los atributos a un rango común, para que todos incidan de igual manera [9]. Sin embargo, cabe mencionar que, más allá de que los atributos estén normalizados, si no tienen una desviación estándar similar, el atributo con mayor dispersión será el determinante en la métrica.

2.2.2. Norma L1

Para trabajar en espacios de altas dimensiones, la norma $L1$ es más apropiada que la norma $L2$ [10]. La misma, corresponde a la *distancia Manhattan* y está definida por la siguiente fórmula:

$$d(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\| = \sum_{i=1}^{n} |p_i - q_i| \tag{2.3}$$

Comparando las ecuaciones (2.2) y (2.3) puede afirmarse que, si dos ejemplos son cercanos para la gran mayoría de los atributos, pero lejanos en alguno de ellos, la distancia Euclídea va a exagerar esa discrepancia. Por otra parte, la distancia Manhattan será mayormente influenciada por la cercanía de los otros atributos, y acortará la diferencia entre los ejemplos. Así, se puede notar que la distancia Euclídea es más sensible a valores extremos o ruido, más allá de que se haya aplicado normalización a los datos.

2.2.3. Norma $L\infty$

Por último, la norma $L\infty$ refiere a la *distancia Chebyshev* y corresponde a la distancia de Minkowski con exponente infinito. La misma, está dada por:

$$d_{\text{Chebyshev}}(p, q) := \max_i(|p_i - q_i|). \tag{2.4}$$

y equivale al límite:

$$\lim_{x \to \infty} \left(\sum_{i=1}^{n} |p_i - q_i|^x \right)^{1/x} \tag{2.5}$$

Por lo tanto, la norma $L\infty$ representa a la mayor diferencia entre los atributos de los ejemplos en cuestión. Se puede apreciar que si se trabaja con múltiples dimensiones, esta medida ignora la dimensionalidad, midiendo la distancia en función de un solo atributo.

2.2.4. Elección de la medida de similitud adecuada

Cuando se quiere hacer un agrupamiento de un conjunto de datos, pueden aparecer diversos resultados usando diferentes medidas de distancia, por lo que es muy importante tener un cierto cuidado al determinar la distancia a utilizar, ya que quizás no tenga sentido aplicarla en el problema a resolver. Encontrar la medida de distancia 'correcta' no es una tarea fácil. Aún más, para cualquier aplicación dada, distintas medidas de distancia pueden producir respuestas similares e igualmente correctas.

De la clasificación anterior, se puede ver que cuando se trabaja con altas dimensiones, reducir el exponente hace que todas las características jueguen un papel importante en el cálculo de la distancia. Cuanto menor sea el exponente, menos relevante será una gran diferencia en alguna dimensión dada.

Cabe mencionar que también es importante considerar la naturaleza del problema a la hora de elegir una medida de similitud. Tiene más sentido en ciertos casos considerar la distancia geométrica y usar así la norma $L2$, mientras que en otros casos, donde los elementos están dispuestos en forma de cuadrícula, sería más conveniente usar la norma $L1$.

La Figura 2.2 ilustra cómo deben interpretarse las medidas de similitud presentadas en esta sección, cuando se trabaja en dos dimensiones.

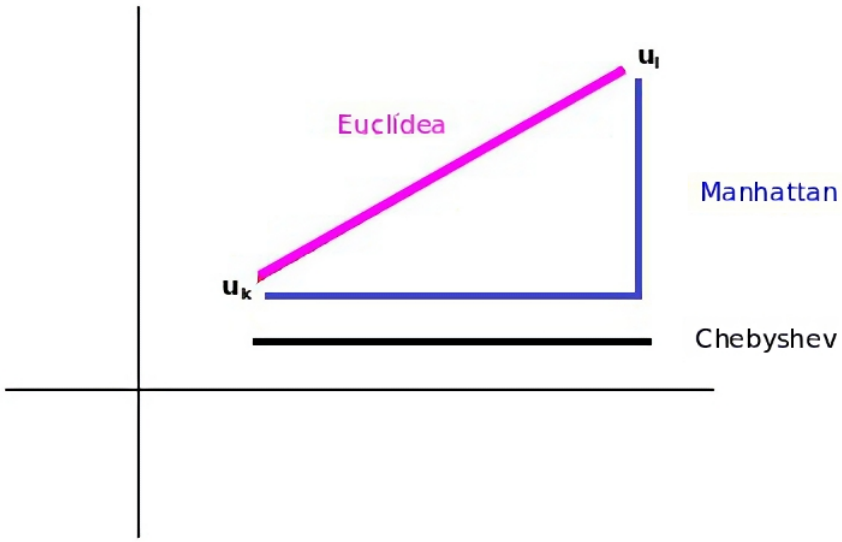

FIGURA 2.2: Interpretación de las normas $L1$, $L2$, $L\infty$ para dos vectores $u_k, u_i \in R^2$

2.3. Tipos de agrupamiento

Durante las últimas décadas, las diferentes definiciones de agrupamientos han dado lugar a una serie de algoritmos, demostrando un desarrollo significativo dentro del campo. Desde sus comienzos, el agrupamiento de datos ha sido aplicado a las más diversas áreas de especialización [11], siendo utilizado para diagnóstico médico [12], recomendaciones en marketing [13], análisis web [14], entre otros.

Se han propuesto en la literatura varios enfoques para realizar agrupamiento [15]. En la práctica, elegir el algoritmo de agrupamiento adecuado, se convierte en una tarea difícil, poco trivial. La elección depende tanto de los datos que se disponen, como del problema en sí y el resultado al que se espera llegar.

En la literatura, se presentan varias técnicas de agrupamiento, y a menudo pueden superponerse unas con otras, por lo que es difícil proporcionar una taxonomía única que los caracterice en su totalidad [16][17].

Una forma de clasificar los métodos de agrupamiento, es dividiéndolos de acuerdo con el tipo de manejo que realizan con el conjunto de datos [18]. En este sentido, los distintos métodos de agrupamiento se dividen en dos categorías principales: agrupamiento *hard* y agrupamiento *difuso* (o bien *soft*). Para el caso de agrupamientos del tipo *hard*, un elemento pertenece a uno y solo un grupo. Por otra parte, para aquellos métodos dentro de la categoría *difuso*, al agrupar los elementos, estos pueden pertenecer a más de un *cluster*.

Entre los tipos de agrupamiento *hard* más estudiados, se encuentran el agrupamiento *partitivo* y el agrupamiento *jerárquico* [15]. A su vez, el agrupamiento jerárquico se puede subdividir en *aglomerativo* (ascendente) y *divisivo* (descendente). La Figura 2.3 muestra la clasificación detallada anteriormente.

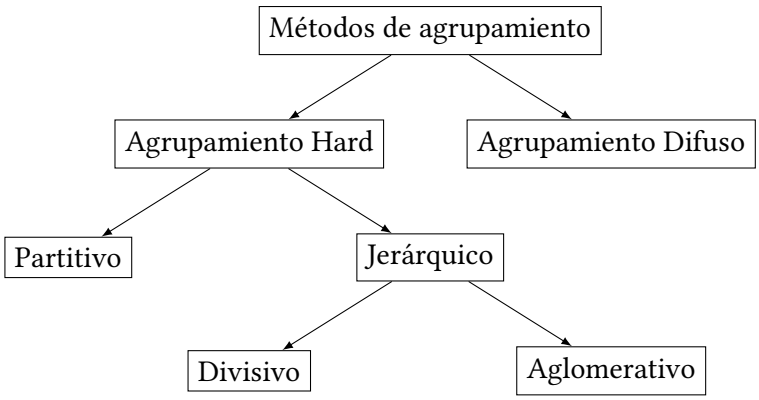

FIGURA 2.3: Clasificación tradicional de los distintos algoritmos de agrupamiento.

Por otra parte, un *cluster* también puede ser definido como una región caracterizada por una alta densidad de elementos, separada de otras regiones densas (otros grupos), por una zona con una baja densidad de elementos [1]. Esta definición, es la base de los algoritmos de agrupamiento que siguen el paradigma *basado en densidad*.

Es necesario tener en cuenta que las definiciones de agrupamientos que se proporcionaron anteriormente son bastante generales, es decir, no definen un algoritmo específico para la obtención de los grupos, dejando espacio para la definición de diferentes mecanismos. Los algoritmos en sí, surgen de la adopción de diferentes sesgos a la hora de particionar los datos.

En las siguientes secciones, se detallan cada uno de los enfoques planteados a la hora de describir la taxonomía elegida. En esta tesina, se hará foco particularmente en los agrupamientos *partitivos* y en los *basados en densidad*.

2.3.1. Agrupamiento difuso

Según este enfoque, cada objeto pertenece a todos los k *clusters*, pero con diferente grado de pertenencia, los cuáles varían del 0 (pertenencia más baja) al 1 (pertenencia más alta).

Formalmente, dado un conjunto de objetos, $\{o_1, ..., o_n\}$, un agrupamiento de k *clusters difusos*, $C_1, ..., C_k$, se puede representar usando una matriz de partición:

$M = [w_{ij}](1 \leq i \leq n, 1 \leq j \leq k)$, donde w_{ij} es el grado de pertenencia del objeto o_i en el grupo difuso C_j. La matriz de partición debe cumplir los siguientes tres requisitos:

- Para cada objeto o_i y grupo C_j, $0 \leq w_{ij} \leq 1$.

- Para cada objeto o_i, $\sum_{j=1}^{k} w_{ij} = 1$.
- Para cada *cluster* C_j, $0 < \sum_{i=1}^{n} w_{ij} < n$. Este requerimiento asegura que por cada cluster, haya al menos un objeto para el cual el nivel de pertenencia es distinto de 0.

Los agrupamientos difusos, entonces, permiten que un objeto participe en múltiples grupos al mismo tiempo, pero de diferente manera.

Dado que el objetivo de esta tesina está puesto en algoritmos dentro de la categoría de agrupamiento *hard*, se remite al lector a los documentos [19] [20], los cuales se enfocan en los diversos trabajos de Bezdek, y a su vez se recomiendan los siguientes [21] [22] para obtener más información sobre los métodos de agrupamiento *difuso*.

2.3.2. Agrupamiento jerárquico

Los algoritmos de agrupamiento jerárquicos. (HCA) no producen como resultado una sola partición, sino un conjunto de particiones anidadas: una *jerarquía de particiones*. Así, mediante esta estructura jerárquica, denominada *dendrograma*, se provee una forma de ver las particiones generadas a partir de los datos, con distinta granularidad.

Los HCA se pueden clasificar en dos categorías principales, según cómo se obtiene el resultado o bien la jerarquía final. Estas dos subclases de algoritmos jerárquicos, son:

- divisivos
- aglomerativos

El flujo de trabajo general para los HCA *divisivos* es el siguiente:

I dados n objetos, asignar todos los objetos a un solo grupo

II dividir el grupo inicial en dos grupos de acuerdo con un criterio dado

III aplicar recursivamente el paso (ii) a los dos grupos generados por la división inicial, hasta que cada grupo tenga un solo objeto, es decir hasta que $k = n$

Por otra parte, para los HCA *aglomerativos*, el procedimiento es el siguiente:

I dados n objetos, asignar cada objeto a un grupo particular, es decir $k = n$

II fusionar los dos grupos más similares en un nuevo grupo

III aplicar el paso (ii) hasta que todos los objetos pertenezcan a un único *cluster*, es decir, hasta que $k = 1$

La Figura 2.4 ilustra las diferencias a la hora de formar la jerarquía de particiones, según se trabaje con un enfoque u otro.

Como se mencionó anteriormente, el resultado de un HCA es representado en forma de *dendrograma*, lo cual se ilustra en la Figura 2.5.

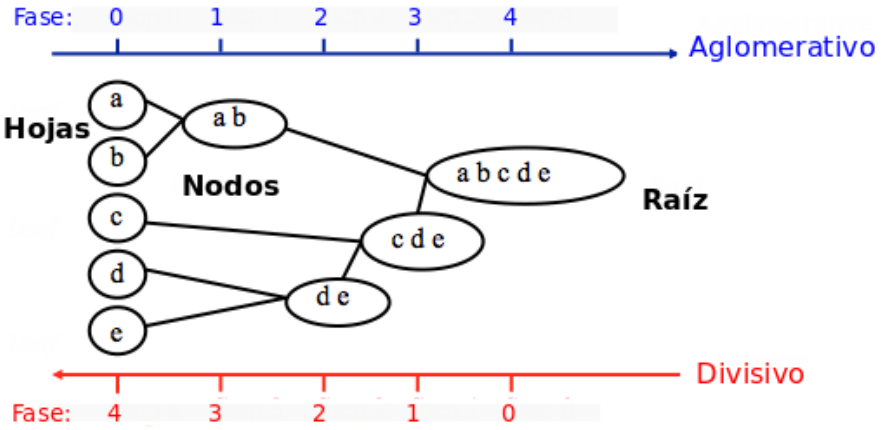

FIGURA 2.4: Diferencia entre HCA aglomerativos y divisivos [23]

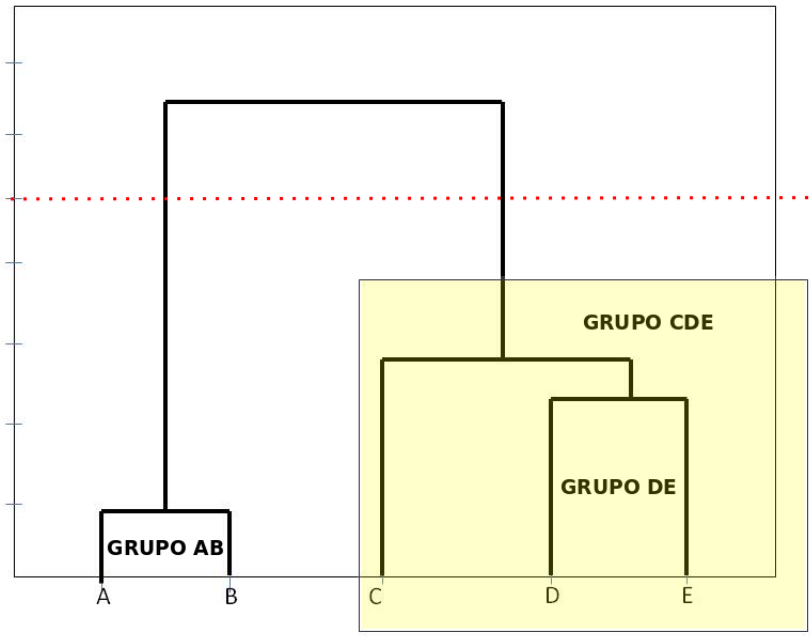

FIGURA 2.5: Agrupación jerárquica de cinco observaciones. La línea punteada genera una partición con dos clusters

Éste, es un diagrama especial que muestra la relación jerárquica entre los diferentes elementos. Es una estructura costosa desde el punto de vista de la memoria necesaria para almacenarlo, y permite principalmente poder seleccionar la mejor manera de ubicar a los elementos en *clusters*. Cada nodo o bien hoja del dendrograma, representa un elemento en particular. A

la hora de interpretar un dendrograma, es necesario enfocarse en la altura en la cual los objetos se unen. Las muestras más similares entre sí, serán aquellas que se unan primeramente, aquellas cuya diferencia sea menor. En el eje y, se ubican los distintos valores para la medida de similitud utilizada. Así, se indican los puntos en los que se forma un grupo (en el caso de HCA aglomerativos) o bien se disuelve (en el caso de HCA divisivos).

Particularmente, en la Figura 2.5, se puede observar que los elementos A y B son los más parecidos entre sí. De esta manera, a partir del dendrograma, se pueden obtener distintas particiones (estructuras de *clusters*) de los datos, y evaluar la elección de la partición más óptima. Vale la pena notar que la figura se presenta con fines ilustrativos, y que otros criterios pueden ser implementados para fusionar o dividir grupos, como es el caso de HDBSCAN [24].

Algunos de los algoritmos de agrupamiento jerárquico tradicionales son BIRCH [25], CHAMELEON [26], y CURE [27].

2.3.3. Agrupamiento partitivo

La versión más simple y fundamental del análisis de *clusters*, está dada por el *agrupamiento partitivo*, el cuál organiza los elementos de un conjunto de datos en k grupos exclusivos. Usualmente, el número de *clusters* que se desean construir, se debe proveer de antemano, por cierto conocimiento del dominio. Este parámetro, que determina la cantidad de grupos a formar, es el punto de partida para la mayoría de los métodos partitivos, basados en distancia.

Formalmente, dado un conjunto D, de n ejemplos, y dado k, el número de grupos a formar, un algoritmo partitivo organiza los elementos en k grupos (donde $k \leq n$). Los grupos se forman tratando de minimizar cierto criterio, como pueden ser las medidas de distancia definidas en la Sección 2.2), de modo que los elementos dentro de un grupo sean 'similares' entre sí y 'diferentes' a los elementos en otros grupos, en términos de los atributos del conjunto de datos. Así, el agrupamiento partitivo es tratado como un problema de optimización.

Por lo comentado anteriormente, entonces, los métodos de agrupamiento partitivos son útiles para las aplicaciones donde se requiere un número fijo de grupos. Además, estos grupos deben cumplir los siguientes requisitos:

- cada grupo debe contener al menos un ejemplo, y
- cada ejemplo debe pertenecer exactamente a un grupo

Un algoritmo estándar y tradicional, que proporciona una representación efectiva de los algoritmos partitivos, es el K-Means [28] [29] [30]. El objetivo de K-Means es agrupar los datos en k grupos, de tal manera que la distancia entre los ejemplos de cada grupo y su centroide sea mínima. Cada grupo es representado por su centro: el promedio de todos los ejemplos, la media aritmética. Se detallará dicho algoritmo en la Sección 2.3.3.1.

Otro algoritmo partitivo muy conocido es el K-Medoids. K-Medoids es una variación del K-Means que, en lugar de utilizar el punto medio para representar a los grupos, utiliza ejemplos reales, uno por *cluster*: el ejemplo más cercano al centro del grupo. Como resultado, este método es más robusto respecto al ruido o bien valores atípicos, ya que en K-Means, valores extremos pueden distorsionar fuertemente el valor del centro del *cluster*. El algoritmo PAM [15] es una implementación popular de este enfoque.

La calidad de la solución obtenida con dichos algoritmos, depende totalmente de la partición inicial realizada con los datos. Una heurística para obtener una mejor (aunque no necesariamente óptima) solución, es repetir el algoritmo, a partir de diferentes soluciones iniciales, y devolver la solución con el valor mínimo para la función elegida como criterio de similitud.

Entre los algoritmos partitivos más populares, se encuentran a su vez CLARA [15] y CLARANS [31].

Se hará énfasis a continuación particularmente en el algoritmo base, tradicional, más frecuentemente utilizado: K-Means.

2.3.3.1. K-means

El algoritmo K-Means, brevemente introducido previamente, es uno de los algoritmos de agrupamiento más conocidos y simples, aplicado mayoritariamente a la hora de resolver un problema de agrupamiento. Es un algoritmo partitivo basado en el concepto de *centroide*. K-Means, define al centroide de un grupo como el valor medio de los ejemplos dentro del grupo. Conceptualmente, el centroide de un grupo es su elemento central, y representa o bien identifica al mismo.

El objetivo de K-Means, es agrupar los datos en k grupos, de tal manera que la distancia entre los elementos en cada grupo y su centroide sea mínima. El algoritmo, realiza una minimización codiciosa de la siguiente función objetivo:

$$\sum_{i=1}^{k} \sum_{\mathbf{p} \in C_i} \|\mathbf{p} - \boldsymbol{c}_i\|^2$$

donde,

- D representa un conjunto de observaciones $D = \{p_1, p_2, ..., p_n\}$, donde cada observación es un vector real de d dimensiones.
- c_i es el centroide del cluster C_i
- $||p - c_i||$ es la distancia Euclídea entre una observación p y el centroide c_i del grupo al cual pertenece.

Así, partiendo de n observaciones, K-Means construye k conjuntos ($k \leq n$), a fin de minimizar la suma de las diferencias al cuadrado dentro de cada grupo.

Para cada elemento en cada grupo, la distancia al centro del *cluster* al cual pertenece es elevada al cuadrado, y esos términos se suman. Esta función objetivo, lo que intenta hacer, es que los k *clusters* resultantes sean lo más compactos y se encuentren lo más separados posible del resto de grupos.

En lo que respecta al procedimiento en sí, el algoritmo K-Means comienza con k centroides, siendo los valores iniciales para los centroides seleccionados al azar o bien derivados de cierta información previa. Entonces, cada ejemplo en el conjunto de datos se asigna al grupo más cercano, es decir, el centroide más cercano. Finalmente, los centroides se recalculan de acuerdo a los ejemplos asociados: se realiza la media aritmética. El algoritmo K-Means, mejora iterativamente la variación dentro de cada *cluster*. Para cada grupo, calcula los nuevos centroides considerando los elementos asignados al grupo en la iteración anterior. Todos los elementos se reasignan utilizando los centroides actualizados, como los nuevos representantes de cada *cluster*. Se puede ver que, los k centroides cambian su ubicación paso a paso, hasta que llegue el momento en el que no se realicen más cambios en los grupos. En otras palabras, hasta que los centroides ya no varíen. Las iteraciones continúan, entonces, hasta que se logre la convergencia; hasta que la asignación de los elementos en los grupos sea la misma en dos iteraciones consecutivas, es decir, los grupos formados en la ronda actual sean los mismos que los formados en la ronda pasada, o bien se alcance el límite de iteraciones especificadas. El funcionamiento del K-Means se resume en la Figura 2.6, donde es aplicado a un conjunto de datos de 2 dimensiones.

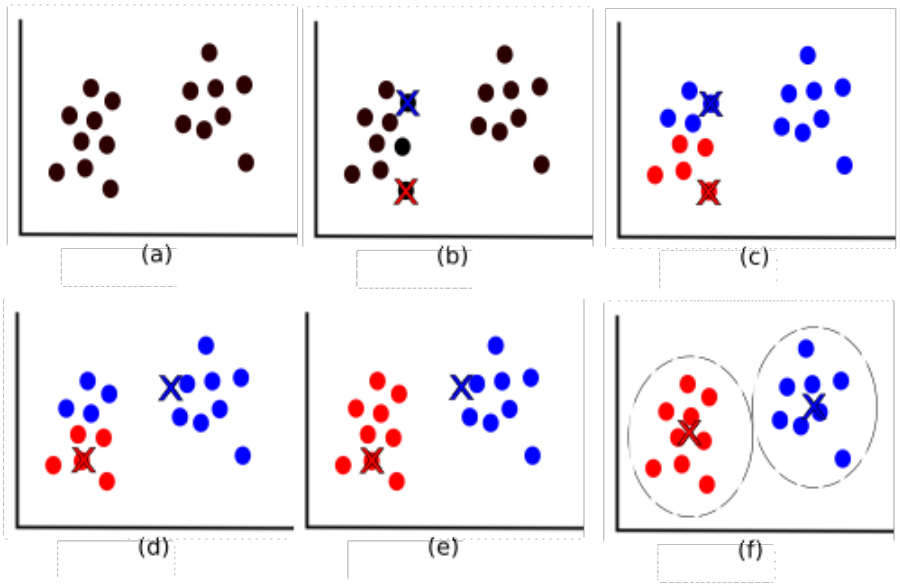

FIGURA 2.6: Funcionamiento del K-Means. (a) Conjunto de datos original. (b) Centroides iniciales aleatorios. (c-e) Iteraciones de K-means, con la reasignación de ejemplos a los grupos y la respectiva actualización de centroides. (f) Agrupamiento final.

Los ejemplos de entrenamiento se muestran ilustrados como círculos, y los centroides como cruces. Partiendo del conjunto de datos original, se seleccionan los centroides iniciales. En cada iteración, se asigna cada ejemplo de entrenamiento al grupo más cercano, considerando los representantes de cada grupo en ese momento. Se puede notar como se recalculan y actualizan los centroides en cada corrida, ya que van variando los ejemplos asignados a cada grupo. Se visualizan los agrupamientos intermedios y el resultado final.

El Algoritmo 2.1 contiene el pseudocódigo correspondiente.

Algoritmo 2.1 K-means

1: **entrada**: k // número de grupos a formar
2: **entrada**: $D = \{p_1, p_2, ..., p_n\}$ // conjunto de datos
3: **salida**: particionamiento final
4:
5: **método**:
6: Elegir arbitrariamente k elementos de D como los centros iniciales de los clusters
7: **mientras** no haya convergencia **hacer:**
8: asignar cada elemento al cluster cuyo centroide es más similar/cercano, basándose en la distancia Euclídea
9: actualizar los centroides de los clusters: calcular el valor promedio de los elementos dentro de cada cluster
10: **fin mientras**

Cabe destacar que K-Means no necesariamente encuentra la partición óptima, entendiendo por tal, aquella que permita alcanzar al mínimo global de la función objetivo. La calidad de la solución obtenida depende intrínsecamente de la partición inicial realizada con los datos. Una heurística para obtener una mejor solución, es repetir el algoritmo, a partir de diferentes centroides iniciales, y así devolver la solución con el valor mínimo para la función objetivo, de todas aquellas que fueron obtenidas.

K-Means presenta ciertas ventajas que lo convierten en el algoritmo más popular en lo que refiere a agrupamiento en general:

1. Es relativamente eficiente: $O(t * k * n * d)$, donde n es el número de elementos, k es el número de grupos, d es la cantidad de atributos de cada objeto, y t es el número de iteraciones ejecutadas. Normalmente, $k, t, d << n$. De esta manera, es factible utilizarlo con grandes conjuntos de datos.

2. Es fácil de entender e implementar.

Sin embargo, a pesar de su simplicidad y fácil implementación, K-Means también presenta algunas limitaciones:

1. Es significativamente sensible a los centros seleccionados inicialmente, ya que estos serán determinantes en lo que respecta al resultado final del algoritmo. Puede ejecutarse

varias veces para reducir este efecto, o bien considerar cierto criterio [32] [33] [34] que determine la ubicación inicial de los centroides, pudiendo de esta manera conseguir resultados determiníscos para cuando se ejecute con los mismos parámetros, evitando las ejecuciones con elecciones aleatorias. Sin embargo, no hay un método eficiente, estándar y universal para identificar los centros iniciales más óptimos

2. Es sumamente sensible a valores atípicos. Aunque un objeto se encuentre lo suficientemente lejos del centroide perteneciente al *cluster* más cercano, está forzado a pertenecer a un grupo y, por lo tanto, distorsiona la forma del grupo al cual se asigna. No existe el concepto de *ruido*.

3. K-Means solo se permite trabajar con conjuntos de datos numéricos y continuos. Muchas de las aplicaciones de minería de datos, constan de datos categóricos. Por lo tanto, convertir estos datos en numéricos para aplicar el algoritmo convencional de K-Means, a menudo revela resultados sin sentido. Cabe destacar que, para realizar *clustering* partitivo con variables categóricas, se presentan 2 variantes del K-Means: K-Modes, para datos categóricos, y K-Prototypes para datos mixtos, aquellos con variables tanto categóricas como continuas. Se recomienda al lector la lectura de los trabajos de Huang [35]. En esta tesina, se hará foco en el agrupamiento de datos numéricos continuos.

4. El usuario necesita indicar el número de grupos a formar de antemano. Si bien existen estrategias para determinar un valor adecuado de k, resulta sumamente difícil garantizar la identificación del número de grupos k óptimo. Los resultados dependen del valor de k.

5. Sólo funciona cuando los grupos naturales que surgen a partir del análisis de los datos son *globulares*. Por su definición, falla a la hora de encontrar grupos con formas arbitrarias.

Este último ítem es particularmente problemático, y resulta de la utilización de un único representante a la hora de caracterizar a los grupos. Por otra parte, en lo que respecta a problemas del mundo real, rara vez se puede encontrar con grupos esféricos subyacentes en los datos. Los algoritmos de agrupamiento basados en densidad salvan esta limitación. Se procederá a detallarlos en la siguiente sección.

2.3.4. Agrupamiento basado en densidad

La aplicación de los algoritmos de agrupamiento para derivar información a partir de grandes bases de datos, deja en evidencia la necesidad de que se cumplan los siguientes requisitos:

1. mínimo conocimiento del dominio para poder determinar los parámetros de entrada (los valores apropiados a menudo no se conocen de antemano cuando se trata de grandes bases de datos),

2. capacidad de descubrir grupos con formas arbitrarias,

3. buena eficiencia

Los conocidos algoritmos de agrupamiento presentados anteriormente no ofrecen solución alguna a la combinación de estos requisitos.

Particularmente, los métodos partitivos que utilizan un único elemento para representar los grupos encontrados presentan dificultades para encontrar grupos de formas arbitrarias como los ilustrados en la Figura 2.7. Dados tales datos, probablemente identificarían de manera inexacta regiones convexas, estando el ruido o bien los valores atípicos incluidos en los grupos. Para encontrar grupos de forma arbitraria, alternativamente, se pueden describir grupos como regiones densas en el espacio de los datos, separadas por regiones dispersas. Esta es la principal estrategia detrás de los métodos de agrupamiento basados en la densidad, los cuáles son capaces de descubrir grupos que no necesariamente tengan una forma esférica.

Figura 2.7: Grupos con formas arbitrarias y presencia de ruido [36]

Estos algoritmos pueden gestionar eficientemente el ruido y la presencia de ejemplos anómalos, no siendo necesario saber de antemano el número de *clusters* a ser identificados, y por consiguiente disminuyendo el grado necesario de conocimiento del dominio.

Algunos de los algoritmos de agrupamiento basados en densidad más populares son: DBSCAN [37], OPTICS [38], y DENCLUE [39]. Entre ellos, el más popular es DBSCAN que, según la literatura, es considerado el primer algoritmo basado en el concepto de densidad. En la actualidad, es ampliamente utilizado, y sigue siendo estudiado y empleado para el desarrollo de nuevos métodos [40] [41] [42]. En la siguiente sección, se procederá a explicarlo.

2.3.4.1. DBSCAN

DBSCAN es un algoritmo de agrupamiento basado en densidad, que reúne a los ejemplos en *clusters* arbitrarios altamente densos, separados por áreas dispersas. DBSCAN es considerado el primer algoritmo a través del cual se introdujo el concepto de *densidad* en relación al agrupamiento de datos.

La idea principal es que, para cada elemento en un grupo, el *vecindario*, considerando un radio dado, debe contener al menos un número mínimo de vecinos. En otras palabras, la densidad en el vecindario debe exceder un cierto umbral. La forma de determinar el vecindario de un elemento, es mediante la elección de una función de distancia. Por ejemplo, cuando se usa la distancia de *Manhattan* en el espacio $2D$, la forma del vecindario es rectangular. Siguiendo este enfoque, el mecanismo opera con cualquier función de distancia para que así, dada una aplicación determinada, se pueda elegir la función adecuada. Cabe mencionar que, con el propósito de obtener mejores visualizaciones, los ejemplos presentados a continuación, serán en el espacio $2D$, utilizando la distancia *Euclídea*.

Entonces, según DBSCAN, la densidad viene dada por la relación entre el área en un círculo de radio Eps, cuyo centro se encuentra en un elemento p_0, y posee en su interior el número mínimo de vecinos $MinPts$.

Hay dos tipos de elementos en un grupo. Un elemento puede ser tanto descriptor central (core), como descriptor límite (border). En general, el vecindario que se obtiene para un *border* considerando Eps, contiene significativamente menos elementos que un vecindario de un *core*.

Se presentan a continuación algunas definiciones esenciales para poder proceder a detallar el algoritmo:

Definición 2.1. Directamente Alcanzable*: Un elemento p es* directamente alcanzable *desde un elemento q considerando Eps y $MinPts$, si:*

1. *p se encuentra en el vecindario de q, de radio Eps*
2. *el vecindario de q tiene al menos $MinPts$ elementos (condición para que q sea* core*)*

Definición 2.2. Alcanzable*: Un elemento p es* alcanzable *desde un elemento q considerando Eps y $MinPts$, si hay una cadena de elementos p_i ... p_n, $p_i = q, p_n = p$ de manera tal que p_{i+1} es directamente alcanzable por densidad desde p_i.*

Definición 2.3. Conectado*: Un elemento p está* conectado *a un elemento q considerando Eps y $MinPts$, si hay un elemento o de manera tal que tanto p como q son* alcanzables *desde o*

Por otra parte, también se encuentran los elementos considerados como ruido: aquellos que no se incluyen en ningún *cluster*. La Figura 2.8 ilustra los distintos tipos de elementos que se pueden encontrar al ejecutar el algoritmo, junto con las definiciones detalladas anteriormente. Se puede ver cómo es definido el concepto de densidad para el algoritmo DBSCAN, la cual está dada por el hyperparámetro $MinPts$, y por un radio que está representado por Eps.

Así, basado en el concepto de densidad explicado anteriormente, DBSCAN realiza un agrupamiento a través de dos procesos principales:

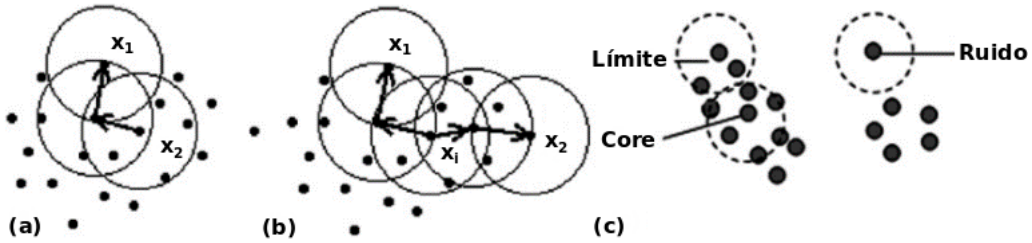

FIGURA 2.8: a) Propiedad de *alcanzable*. b) Propiedad de *conectado*. c) Distintos tipos de elementos [43]

- Escanear(): se encarga de recorrer todo el listado de elementos para identificar así cada elemento *core*
- ExpandirCluster(): es utilizado por el proceso anterior, y se encarga de expandir el *cluster* creado por el elemento *core* en cuestión, mediante la búsqueda de elementos que posean las propiedades de *conectado* y *alcanzable*.

En este punto, se recomienda al lector dirigirse al Algoritmo 2.2.

En lo que refiere al algoritmo en sí, para encontrar un *cluster*, DBSCAN comienza con un elemento arbitrario p y recupera todos los elementos *alcanzables* desde p, considerando Eps y $MinPts$. Si p es un elemento *core*, se forma un grupo. Si p es un elemento *border*, no hay elementos *alcanzables* a través de p, y DBSCAN procede a visitar el siguiente elemento de la base de datos. Dado que se utilizan valores globales para Eps y $MinPts$, DBSCAN puede combinar dos grupos en uno solo, si éstos grupos de densidad diferente se encuentran lo suficientemente 'cerca' uno del otro. Luego, dos conjuntos de elementos que tengan al menos la densidad del *cluster* más disperso, serán separados uno del otro solo si la distancia entre los dos conjuntos es mayor que Eps. En consecuencia, una llamada recursiva de DBSCAN puede ser necesaria para los grupos detectados con un valor muy alto para $MinPts$. Sin embargo, esto no es una desventaja porque la aplicación recursiva de DBSCAN produce un algoritmo básico, elegante, y muy eficiente. Además, la agrupación recursiva de los elementos de un grupo solo es necesaria en condiciones que se puedan detectar fácilmente.

Se listan a continuación las ventajas de DBSCAN, las cuáles hacen que éste sea popularmente utilizado:

- No es necesario saber de antemano la cantidad de grupos en los cuales se dividirán los puntos.
- Habilidad para identificar ruido o valores extremos.

Por último, entre las limitaciones del algoritmo, se encuentran:

- La incapacidad de descubrir grupos de diferentes densidades, ya que tanto Eps como $MinPts$ se definen globalmente, y

Algoritmo 2.2 DBSCAN

```
1: entrada: eps                                                    // radio vecindario
2: entrada: D = {p_1, p_2, ..., p_n}                               // conjunto de datos
3: entrada: MinPts        // mínimo de elementos necesarios para que un elemento sea core
4: salida: particionamiento final
5:
6:
7: proceso ESCANEAR(eps, D, MinPts)                                // proceso principal
8:     C ← 0
9:     por cada elemento p no visitado en el conjunto de dato D hacer:
10:        marcar p como visitado
11:        vecinos ← consultarRegión(p, eps)
12:        si cantidad_elementos(vecinos) < MinPts entonces
13:            marcar p como ruido
14:        en otro caso
15:            C ← próximo cluster
16:            expandirCluster(p, vecinos, C, eps, MinPts)
17:        fin si
18:    fin por
19: fin proceso
20:
21:
22: proceso EXPANDIRCLUSTER(p, vecinos, C, eps, MinPts)
23:    agregar p al cluster C
24:    por cada elemento p' en vecinos hacer:
25:        si p' no fue visitado entonces
26:            marcar p' como visitado
27:            vecinos' ← consultarRegión(p', eps)
28:            si cantidad_elementos(vecinos') ≥ MinPts entonces
29:                vecinos ← vecinos combinado con vecinos'
30:            fin si
31:        fin si
32:        si p' todavia no pertenece a ningún cluster entonces
33:            agregar p' al cluster C
34:        fin si
35:    fin por
36: fin proceso
37:
38:
39: proceso CONSULTARREGIÓN(p, eps)
40:    retornar todas los elementos dentro del vecindario de p' (incluyendo p)
41: fin proceso
```

- La dificultad al elegir la combinación más óptima de los hiperparámetros Eps y $MinPts$

2.4. Conclusión

En este capítulo, se han detallado los principales métodos de agrupamiento tradicionales: *partitivos* y *jerárquicos*, junto con los algoritmos más populares dentro de cada categoría. Particularmente se debatieron los beneficios y las limitaciones del K-Means, algoritmo mayormente utilizado a la hora de aplicar agrupamiento a un conjunto de datos. Esto se debe ante todo por su simpleza, lo cual hace que sea fácil tanto de interpretar como de implementar. Las limitaciones que presentan los métodos partitivos basados en distancia, y en el concepto de centroide, referente a la imposibilidad de captar grupos de formas no convexas, junto con la sensibilidad ante valores extremos, hicieron que se desarrollen otros enfoques, como es el enfoque *basado en densidad*. Se habló especialmente del algoritmo DBSCAN, destacado por ser el primero en incorporar el concepto de *densidad* en el área de *clustering*, y por su capacidad para descubrir grupos con formas arbitrarias, sin la necesidad de indicar de antemano la cantidad de grupos a formar, información que generalmente no se tiene a la hora de resolver un problema del mundo real.

Por otra parte, los enfoques detallados, pese a las diferencias que presentan, coinciden en el hecho de que necesitan para su aplicación la totalidad del conjunto de datos desde un principio, recorriéndolo iterativamente para realizar el análisis y obtener el agrupamiento final.

En la era digital en la que vivimos, abundan las aplicaciones y dispositivos que generan datos a muy alta velocidad, los cuáles crecen en términos exponenciales. Estos datos continuamente generados por diversas fuentes, se denominan *flujos de datos*. Cuando se trata de flujos de datos, el procesamiento de los mismos se convierte en un nuevo desafío, ya que se deben satisfacer las demandas que implica el análisis en tiempo real: capacidad limitada de memoria, escaneo único de datos, y diferenciación entre anomalías y deriva de concepto. Es por este motivo, que las técnicas de agrupamiento convencionales, detalladas anteriormente, resultan ineficientes para estos casos, ya que el comportamiento de los datos en sí ha cambiado.

Se introduce de esta manera el *agrupamiento de flujos de datos*, el cuál se tratará en detalle en el siguiente capítulo.

Capítulo 3

Flujos de Datos

Los flujos de datos han ido cobrando cada vez mayor importancia con el avance en el campo de Internet de las cosas (IoT). De hecho, objetos que se utilizan en el día a día están constantemente volviéndose más inteligentes, proporcionando así muchísimos datos. Estos datos generados no se registran en bases de datos sino que se transmiten por *flujos* para que se puedan tener en todo momento los valores actuales. Tratar con flujos de datos, agrega cierta complejidad al proceso de agrupamiento, ya que se añaden inconvenientes adicionales al problema que se quiere resolver:

1. Los flujos de datos son virtualmente infinitos. Eso implica que el algoritmo de agrupamiento no pueda acceder a cada elemento del conjunto en cualquier momento. Cada elemento, debe *sumarizarse* de alguna manera al momento en que aparece, debido a la limitación de memoria. De esta manera, los resultados finales son aproximados, utilizando herramientas estadísticas.

2. En algunos casos, el flujo de datos puede cambiar con el tiempo. Por ejemplo, el entorno de cierto sensor puede modificarse debido a cambios climáticos. Es por esto, que el algoritmo de agrupamiento necesita manejar lo que se conoce como *deriva del concepto*, referente a la evolución por la naturaleza de la propia data o bien por la ocurrencia de un evento inesperado durante el tiempo de ejecución.

3. Manejo de ruido y valores atípicos. Esto es sumamente importante ya que los valores extremos pueden llegar a distorsionar los resultados.

4. Dependiendo de la ventana de tiempo de ejecución en la se esté ubicado, los resultados serán diferentes. El objetivo es asegurarse de obtener el mejores resultados en cada momento.

A la hora de desarrollar un algoritmo de agrupamiento de flujos de datos, entonces, se ve la necesidad de satisfacer los siguientes tres requisitos [44]:

1. Utilización de cierta estructura de datos compacta para lograr así una mayor eficiencia de memoria, considerando las limitaciones referentes a la disponibilidad.
2. Desarrollar un proceso rápido, para mejorar el tiempo de ejecución.
3. Manejo apropiado de valores atípicos, para mayor precisión.

Para poder adaptarse a los posibles cambios en las distribuciones de los diferentes grupos que se van formando a partir de un flujo de datos, es deseable que los *clusters* tengan los siguientes comportamientos:

1. Aparición: se crea un nuevo *cluster* cuando un grupo de datos atípicos se vuelve lo suficientemente denso como para ser considerados representativos. Cuando un nuevo dato no pertenece a ningún *cluster* existente, es considerado como un caso atípico.
2. Desaparición: un grupo previamente formado se va desvaneciendo con el tiempo, hasta un cierto punto donde se desvanece por completo: el grupo desaparece.
3. Autoevolución: la posición o el tamaño de un *cluster* evolucionan. La solidez de la autoevolución depende del peso que se le de a los elementos nuevos.
4. Fusión: dos grupos se vuelven lo suficientemente similares como para ser considerados como uno solo.
5. División: un *cluster* se puede dividir en dos.

Es gracias a dichos comportamientos que puede observarse y modelarse la evolución en los datos.

Es importante destacar que este tipo de algoritmos son incrementales, y comparten una idea común: resumir el flujo en *microclusters*, para luego agrupar éstos en una fase *offine*. El concepto de *microcluster*, fue introducido junto con el CluStream [45]. Un *microcluster*, corresponde a una extensión temporal de la noción de *vector de características*, presentada en [25]:

Definición 3.1. vector de características (CF)*: estructura de sumarización que se mantiene sobre un* cluster*, suponiendo que los elementos del conjunto de datos son vectores compuestos por* d *números reales. Es un vector triple, dado por:*

$$CF = (N, LS, SS) \tag{3.1}$$

Donde:

1. *N: número de elementos en el CF*

2. *LS: suma d-dimensional de los N elementos*
3. *SS: suma de cuadrados d-dimensionales de los N elementos*

La característica más importante de los CF, es que éstos son aditivos:

$$CF_1 + CF_2 = (N_1 + N_2, LS_1 + LS_2, SS_1 + SS_2) \tag{3.2}$$

El resultado final sería equivalente a fusionar dos clusters *disjuntos,* CF_1 *y* CF_2.

Partiendo de los CF, se pueden calcular fácilmente tanto la distancia desde un elemento a un cluster*, como la distancia promedio entre* clusters *y la distancia promedio dentro de cada grupo.*

Definición 3.2. Microcluster*: Al agrupar flujos de datos, no resulta práctico guardar todos los datos entrantes. Los* microclusters *son popularmente utilizados en el agrupamiento de flujos de datos, ya que permiten mantener una representación compacta de los datos recibidos hasta el momento. Considerando un conjunto de datos formado por elementos d-dimensionales, un* microcluster *queda definido de la siguiente manera:*

$$MC = (N, LS, SS, LST, SST) \tag{3.3}$$

Donde:

1. *LST: suma de los marcas de tiempo utilizadas para hacer un seguimiento y tener una noción de orden de llegada. Indican una fecha y hora (timestamps) de los elementos agregados*
2. *SST: suma de los cuadrados de los timestamps de los elementos agregados*

Cabe mencionar que en este caso se extiende la definición de CF, agregando marcas de tiempo.

En esta sección, serán detallados 2 de los algoritmos más populares a la hora de hacer un agrupamiento de flujos de datos, CluStream y DenStream [46]. A su vez, se explicará un nuevo algoritmo introducido recientemente en la literatura: DyClee [47], el cuál será utilizado principalmente para el desarrollo de la tesina.

3.1. CluStream

Como se ha mencionado anteriormente, los algoritmos de agrupamiento de flujos de datos están restringidos por la condición de una pasada: dado que se está utilizando como conjunto de datos un flujo, se puede acceder a la información referente a un elemento dado solo una única vez, cuando éste arriba. Esto se debe a que el almacenamiento de los datos generaría altos requisitos de disco, por lo cual no es factible. Los algoritmos de agrupamiento de flujos que se desarrollaron en un principio [25] [48] [49], son 'ciegos' a la evolución de los datos. Esto da como resultado un agrupamiento deficiente, cuando se tienen datos que evolucionan en el tiempo.

Unos de los algoritmos diseñados para manejar la *deriva del concepto* o bien evolución de los datos, es CluStream. Mientras que los primeros algoritmos introducidos para la tarea de *clustering* de flujos de datos ejecutan una única instancia destinada a capturar el flujo indefinidamente, CluStream utiliza otro enfoque, empleando el uso de *microclusters* y resúmenes almacenados.

CluStream se divide en dos etapas. La primera, llamada *'online'*, escanea el flujo de datos continuamente y crea *microclusters* usando un vector de características o CF que representa los elementos englobados en cada *microcluster*. Luego, periódicamente, se crean y almacenan representaciones del estado actual de los *microclusters* a modo de resumen, en una *estructura piramidal*. Estos resúmenes permiten al usuario obtener un agrupamiento del flujo en diferentes ventanas de tiempo, durante la segunda parte del algoritmo. El usuario debe elegir una ventana de tiempo a partir de la cuál cargar dichos resúmenes. Así, los *microclusters* que se obtienen de la operación anterior, se usan como base para el agrupamiento *macro* o agrupamiento final que se realiza en la segunda fase, llamada *'offline'*. Para la ejecución de esta etapa, el usuario debe definir de antemano cuántos *clusters* espera para el agrupamiento, y finalmente, los *microclusters*, que representan los elementos reales del flujo, que fueron recibidos y resumidos en la etapa *'online'*, se agrupan de acuerdo con el número de *macroclusters* especificado, ejecutando el algoritmo K-Means 2.3.3.1.

Más en detalle, la fase *online* tiene en cuenta la actividad de los *microclusters*, considerando que sólo se mantiene un número máximo de q *microclusters* en memoria (ver la Figura 3.1).

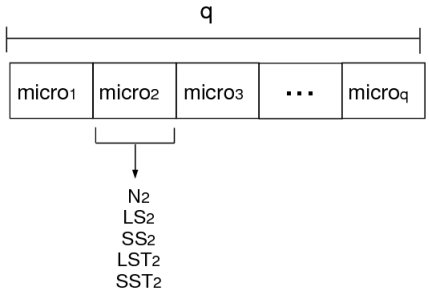

FIGURA 3.1: Estructura de *microclusters* usada en el algoritmo CluStream [50]

Si un elemento requiere crear un nuevo *microcluster* y no hay más espacio disponible, el algoritmo elimina el *microcluster* más antiguo o fusiona dos de los *microclusters* más antiguos para permitir así la creación del nuevo *microcluster*. La noción del *microcluster* 'más antiguo' se define teniendo en cuenta si ha habido una actividad reciente significativa en el *microcluster*. Más precisamente, los valores LST y SST se utilizan para calcular la desviación promedio y estándar de sus marcas de tiempo. Suponiendo una distribución normal de las marcas de tiempo, el algoritmo puede usarlas para decidir si una fracción deseada de los elementos en el *microcluster* es más antigua que un umbral de antigüedad definido.

Para un mayor detalle, se provee el pseudocódigo del mismo en el Algoritmo 3.1. Así mismo, se recomienda al lector dirigirse a la Figura 3.2 para visualizar cómo se comportan las etapas *online* y *offline*.

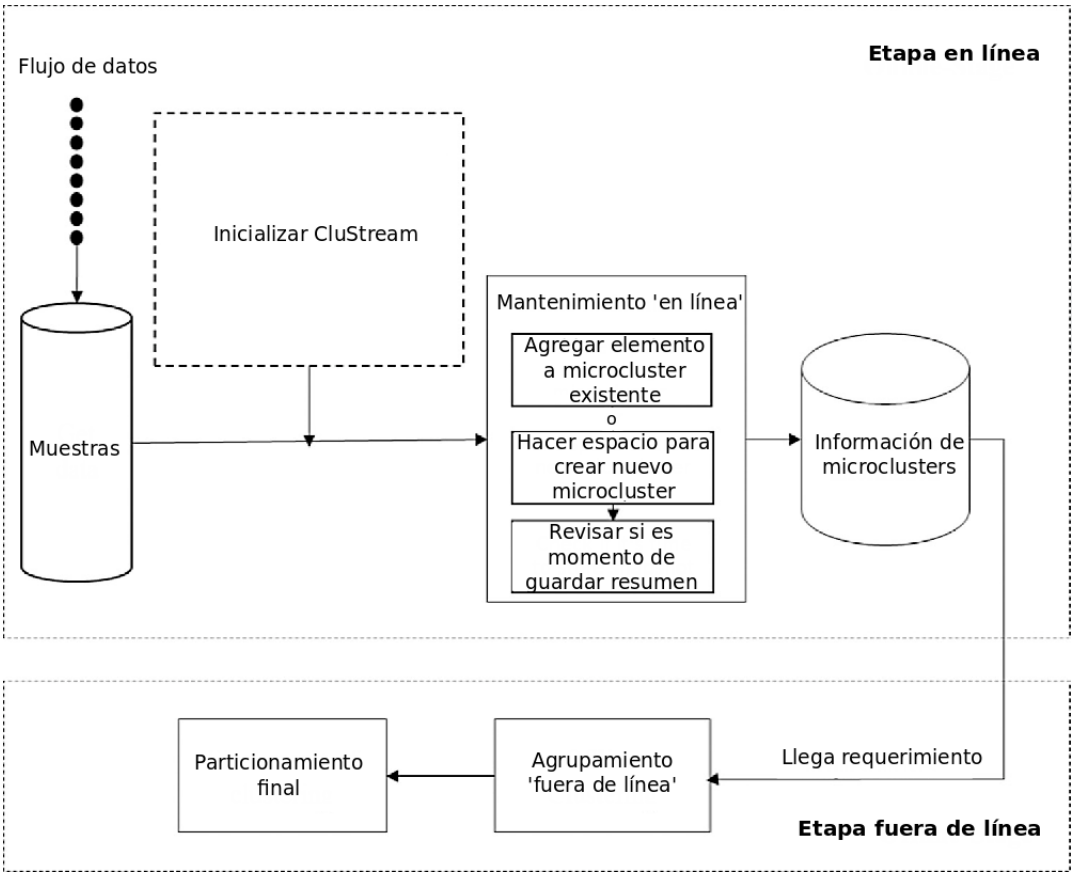

FIGURA 3.2: Funcionamiento del algoritmo CluStream, considerando las fases online y offline [51]

En lo que respecta al método en sí, es realmente ventajoso por dos razones. Primero, se brinda la posibilidad de elegir el horizonte temporal para el cual se requiere que se ejecute el agrupamiento final. Esto, permite enmarcar diferentes conceptos originados por una eventual deriva de concepto. Resulta interesante tener la posibilidad de visualizar el flujo como un proceso cambiante a lo largo del tiempo. Segundo, a pesar de que se utiliza un parámetro k en el proceso de agrupamiento final, considerando la restricción de 'una sola pasada' para

Algoritmo 3.1 CluStream

```
1: Entrada: DS                                          // referencia al flujo de datos
2: Entrada: n_elementos_iniciales   // cantidad de elementos que cebe contener el búfer para
   inicializar
3: Entrada: q                                  // límite de microcluster a crear en memoria
4: Entrada: k                                  // límite de clusters finales a generar
5:
6:
7: proceso EtapaOnline
8:     mientras haya una nueva muestra p perteneciente a un flujo de datos hacer:
9:         si no se ha inicializado entonces
10:            guardar p en el búfer
11:            si el búfer alcanzó n_elementos_iniciales entonces
12:                formar q microclusters iniciales corriendo un K-Means sobre los elementos en el búfer
13:                setear el flag de algoritmo inicializado
14:            fin si
15:            cortar iteración
16:        fin si
17:        computar la distancia entre p y cada uno de los centros de los q microclusters
18:        C ← microcluster más cercano a p
19:        si p cae en los límites de C entonces
20:            C absorbe p y las estadísticas de C son actualizadas          // p es similar a C
21:        en otro caso
22:            si no hay espacio en memoria entonces
23:                borrar el microcluster más obsoleto o combinar los dos más cercanos si eso no es
                   posible
24:            fin si
25:            crear nuevo microcluster para representar a p
26:        fin si
27:        si es tiempo de almacenar resumen de los microclusters actuales entonces
28:            guardar resumen
29:        fin si
30:    fin mientras
31: fin proceso
32:
33:
34: proceso EtapaOffline(h, k)
35:    tomar los microclusters a partir de un determinado período de tiempo u horizonte h
       (resumen)
36:    aplicar K-Means a los microclusters en h para derivar los k clusters finales
37: fin proceso
38:
```

cada elemento, no se genera ningún problema ya que, al no estar limitados en la etapa final por un mecanismo *online*, se puede generar varias veces un agrupamiento, para una ventana de tiempo particular, usando diferentes parámetros cada vez para comparar los resultados.

Desafortunadamente, CluStream presenta ciertos inconvenientes. Por un lado, es necesario especificar de antemano la cantidad de *microclusters* q que se mantendrán en el tiempo. Es importante remarcar que q suele ser un valor mucho mayor a k, y mucho menor que la cantidad total de elementos que arriban. Luego, debido a la complejidad del algoritmo, la calidad del agrupamiento disminuye cuando el flujo de datos es altamente multidimensional. Además, como se detalló anteriormente en la Sección 2.2, una dimensionalidad demasiado alta conduce a problemas en la interpretación de *distancia*. Al ser CluStream un algoritmo basado en distancia, cuanto aumenta la dimensionalidad de los datos, los ejemplos están sujetos a estar demasiado lejos el uno del otro en al menos una dimensión. En este caso, si el umbral que determina la incorporación de un dato dentro de un *microcluster* es demasiado bajo, es probable que se rechacen los datos. Mientras que, si es demasiado alto, se espera que se acepten valores atípicos, lo que termina resultando en una calidad de agrupamiento no aceptable.

Por otra parte, puede decirse que CluStream hereda las limitaciones que trae K-Means, principalmente en lo relacionado con la sensibilidad ante valores atípicos y la necesidad de especificar el valor del parámetro k que fija a priori la cantidad de grupos a formar.

3.2. DenStream

Los algoritmos para flujos de datos propuestos en un principio, como ser CluStream, a menudo producen *clusters* esféricos, ya que se utiliza la distancia para poder determinar la similitud o diferencia entre elementos. Por otra parte, en el caso particular de CluStream, al utilizarse una variante del algoritmo K-Means para obtener el resultado final, un *cluster* natural puede llegar a dividirse en por ejemplo dos partes, por el mismo motivo de que se utiliza la distancia, particularmente Euclídea, como medida de similitud.

Más aún, debido a la naturaleza dinámica de los flujos de datos que evolucionan, el papel de los valores atípicos y los *clusters*, con frecuencia se intercambian y, en consecuencia, nuevos grupos suelen emerger, mientras que los viejos grupos se desvanecen. Se vuelve todavía más complejo cuando existe ruido. Por ejemplo, en CluStream, el algoritmo mantiene continuamente un número fijo de q *microclusters*. Tal enfoque es especialmente arriesgado cuando el flujo de datos contiene valores atípicos, ya que de esa manera se crearán nuevos *microclusters* para valores que no son representativos, y para ello se eliminarán o fusionarán *microclusters* existentes. Así, diferentes *grupos naturales* pueden llegar a tener que fusionarse. Se ve clara la necesidad de que un algoritmo de agrupamiento para flujos de datos, proporcione cierto mecanismo para distinguir los elementos que cumplen rol de *semillas* para la formación de nuevos grupos, de los valores atípicos [46].

De esta manera, se introduce DenStream [46], un algoritmo de agrupamiento de flujos de datos basado en *densidad*, el cuál extiende al algoritmo tradicional DBSCAN, detallado anteriormente en la Sección 2.3.4.1. Al igual que CluStream, DenStream utiliza *microclusters* para capturar información a medida que llegan datos de un flujo, a modo de 'resumen'. Por otra parte, también utiliza el enfoque de aprendizaje basado en 2 etapas, incorporando un mecanismo de olvido. Su componente *online*, actualiza continuamente la colección de *microclusters*. Cada *microcluster* tiene un centro y un radio que se derivan de su respectivo CF. Al aplicar DenStream un *componente de olvido*, los elementos de los CF disminuyen en la medida que los respectivos *microclusters* se mantienen inactivos por cierto tiempo.

A partir de los valores de umbral para el *peso* y el *radio* que determinan la cantidad de elementos y el tamaño del vecindario respectivamente, se definen tres tipos de *microclusters*:

1. *core* microclusters, aquellos cuya cantidad de elementos, indicada a través de su peso w, es mayor a $MinPts$
2. *potenciales core* microclusters *(p-microclusters)*, donde $w \geq \beta * MinPts$, siendo β un parámetro que determina el umbral para considerar un *microcluster* como valor extremo, por fuera del rango esperado, diferente del resto de los datos (outlier). Notar $0 < \beta < 1$
3. *outlier* microclusters *(o-microclusters)*, $w < \beta * MinPts$

Las estructuras de *p-microclusters* y *o-microclusters* se introducen para mantener la diferencia entre los potenciales grupos y los valores atípicos. Esto se debe a que, como se mencionó anteriormente, en los flujos de datos que evolucionan, el papel de los *clusters* y los valores atípicos se intercambian a lo largo del tiempo. La principal diferencia entre ellos radica en sus diferentes restricciones en cuanto a los pesos. Además, se define un buffer para separar el procesamiento de los *p-microclusters* de los *o-microclusters*. Se recomienda al lector dirigirse a la Figura 3.3.

Entonces, resumiendo, DenStream utiliza un enfoque de vector de características para resumir los datos, y una variante del algoritmo DBSCAN para realizar el agrupamiento final bajo demanda del usuario. Esta última etapa, recibe como entrada los *p-microclusters* representados por vectores de características, y dos parámetros - Eps y $MinPts$ - para así poder particionar los datos. Un *cluster,* entonces, se crea como un grupo de *microclusters*, lo cuáles son densos y se encuentran relativamente cerca unos de otros. A pesar de que el usuario no necesita especificar explícitamente el número de *clusters* a obtener, la definición de Eps y $MinPts$ tiene una fuerte influencia en la partición de datos resultante.

Como se mencionó anteriormente, DenStream divide el proceso de agrupamiento en dos partes. Primero, se lleva a cabo un paso de mantenimiento *online* de *microclusters*, seguido de una fase *offline* que genera los grupos finales, utilizando el algoritmo de agrupamiento DBSCAN.

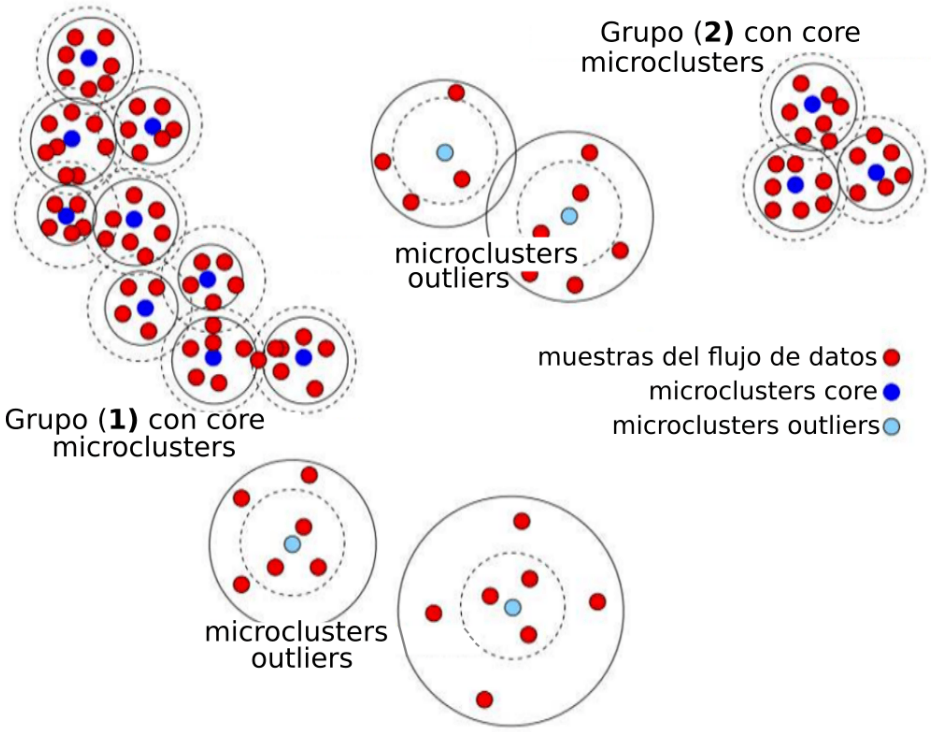

FIGURA 3.3: Estructura de *microclusters* usada en el algoritmo DenStream [52]

Se recomienda dirigirse al Algoritmo 3.2.

A continuación se procederá a detallar particularmente cada fase.

1. Fase *online* (mantenimiento de *microclusters*): Los *p-microclusters* y *o-microclusters* se mantienen en memoria, considerando los *o-microclusters* se ubican en un espacio separado, el cual se conoce como *outlier-buffer*. Cuando un nuevo elemento llega, se fusiona con alguno de los *microclusters* existentes de la siguiente manera:

   a) Se combina con el *p-microcluster* más cercano

      1) sólo si el radio final del *p-microcluster*, obtenido después de agregar al nuevo punto, está por debajo o bien es igual al umbral que se considera para el radio de un *c-microcluster*.

      2) De lo contrario, se combina el nuevo punto con el *o-microcluster* más cercano.

   b) Si es combinado con el *o-microcluster* más cercano, se verifica el peso resultante del *o-microcluster*,

      1) si éste es más alto que el umbral, significa que creció lo suficiente como para convertirse en *p-microcluster*, por lo que se eliminará del *outlier-buffer* y pasará a ser un *p-microcluster*.

      2) De lo contrario, el *microcluster* permanece en el búfer de valores atípicos.

Algoritmo 3.2 DenStream

Entrada: DS // referencia al flujo de datos
Entrada: eps // radio vecindario
Entrada: β // umbral para determinar si un microcluster representa valores atípicos
Entrada: $MinPts$ // mínimo de elementos necesarios para que un elemento sea core
Entrada: λ // utilizada para el componente de olvido

proceso DenStream($eps, DS, MinPts, \lambda, \beta$) // main
    $T_p = \frac{1}{\lambda} \ln(\frac{\beta * MinPts}{\beta * MinPts - 1})$
    tomar el siguiente elemento p al momento t desde el flujo de datos DS
    Combinar(p)
    si $t \bmod T_p = 0$ **entonces** // actualizar las estructuras de microclusters
        por cada p-microcluster c_p **hacer:**
            si w_p (el peso de c_p) $< \beta * MinPts$ **entonces**
                eliminar c_p
            fin si
        fin por
    fin si
    por cada o-microcluster c_o **hacer:**
        $aux = \frac{2^{-\lambda(t-t_o+T_p)} - 1}{2^{-\lambda T_p} - 1}$
        si w_o (el peso de c_o) $< aux$ **entonces**
            eliminar c_o
        fin si
    fin por
    si llega un requerimiento para agrupar **entonces** // agrupamiento actual
        correr DBSCAN y generar los grupos finales
    fin si
fin proceso

proceso Combinar(p)
    $c \leftarrow$ p-microcluster más cercano a p
    si r_p (nuevo radio de c) $\leq eps$ **entonces**
        combinar p con c
    en otro caso
        $c \leftarrow$ o-microcluster más cercano a p
        si r_p (nuevo radio de c) $\leq eps$ **entonces**
            combinar p con c
            si w (nuevo peso de c) $\geq \beta * MinPts$ **entonces**
                remover c del búfer de valores atípicos y crear un nuevo p-microcluster
            fin si
        en otro caso
            crear un nuevo o-microcluster para representar p e insertarlo en el búfer de valores atípicos
        fin si
    fin si
fin proceso

c) En caso de que el nuevo elemento no sea similar a ningún *microcluster* existente hasta el momento, un nuevo *o-microcluster* es creado para representar al objeto en cuestión. Así, el algoritmo coloca cada nuevo elemento de entrada en el *outlier-buffer*, cuando éstos no pueden incoporarse en ninguno de los *microclusters* existentes. Conceptualmente, estos datos pueden representar un valor atípico, o bien cumplir el rol de *semilla*, a partir de la cual surja un nuevo *p-microcluster*.

2. Fase *offline* (generación de grupos): después de capturar los *p-microclusters* en la fase *online*, se aplica el algoritmo DBSCAN sobre los mismos, para obtener de esta manera el resultado final de agrupamiento, en un momento dado, bajo demanda del usuario. Cada *p-microcluster* se considera como un elemento virtual a agrupar, cuando en realidad resume a n cantidad de elementos. El peso es un parámetro importante en esta variante. Para cualquier decisión, el peso es consultado. El mismo, deberá ser más alto que el umbral determinado. Además, el algoritmo presenta una estrategia para distinguir entre los valores atípicos reales y aquellos que posiblemente pasarán a ser *p-microclusters*. En intervalos de tiempo especiales, el algoritmo comprueba los pesos tanto de los *p-microclusters*, como de los *o-microclusters*, y toma una decisión basada en ellos:

   a) si el peso actual de un *p-microcluster* es inferior al umbral, pasará a considerarse como *outlier*. Eso significa que el *microcluster* se mantuvo activo por cierto tiempo, pero luego dejó de serlo: no incorporó ningún dato nuevo durante cierto período de tiempo

   b) si el peso actual de un *o-microcluster* está por encima del umbral, el *o-microcluster* pasa a considerarse como un *p-microcluster*.

Finalmente, en lo que respecta al método en general, resulta interesante por dos razones. Al utilizar una variante de DBSCAN, primero, no requiere especificar el número de *clusters* a obtener de antemano, y puede desempeñarse bien con grupos de formas arbitrarias. Además, es robusto: tiene la capacidad de detectar valores atípicos.

Por otra parte, DenStream presenta ciertos inconvenientes: hereda las limitaciones que trae DBSCAN. Por un lado, no tiene capacidad de detectar grupos con diferentes densidades. Esto se debe a que los parámetros asociados a la densidad, se definen globalmente. Luego, determinar una distancia adecuada de vecindario (Eps), junto con el hiperparámetro $MinPts$, no es fácil y requiere conocimiento de dominio.

3.3. DyClee

Las principales desventajas de muchos de los enfoques presentados para flujos de datos, consisten en altos requerimientos en términos de memoria y/o potencia de cómputo, los cuáles son sumamente necesarios para realizar un agrupamiento de *streams*. También, se observa

en los algoritmos presentados, una imposibilidad de hacer frente a agrupamientos complejos (es el caso de CluStream), o bien incapacidad de lidiar con grupos de densidades múltiples, subyacentes en los datos (es el caso de DenStream).

En esta sección, se introduce a DyClee [47], un algoritmo para el agrupamiento de flujos de datos no estacionarios, presentado recientemente en la literatura que contempla las limitaciones presentadas anteriormente alcanzando propiedades interesantes. En particular, el manejo de grupos no convexos, y la capacidad de formar grupos de diferentes densidades, pudiendo además rechazar *outliers*, y manteniendo una adaptabilidad completa, lo cual permite llevar un seguimiento continuo de la evolución de los *clusters*.

Las técnicas de agrupamiento para flujos de datos, han surgido generalmente como algoritmos de dos etapas. Esto se detalló en las secciones anteriores. En la primera etapa, las muestras de datos se recopilan, preprocesan y comprimen en *microclusters*, siguiendo un enfoque de agrupamiento basado en la distancia. En la segunda etapa, los *microclusters* se agrupan en grupos reales, que pueden vincularse a conceptos del dominio. DyClee, sigue a su vez ese camino.

Partiendo de lo anterior, se procederá a mencionar dos distinciones importantes. En la etapa basada en distancia, DyClee usa la norma L_1, conocida como la distancia de Manhattan. La misma, fue detallada en la Sección 2.2.2. Esta elección, se debe a que la medida de distancia seleccionada, es más apropiada para espacios de alta dimensionalidad, comparando con la norma L_2 . En la etapa basada en densidad, DyClee logra formar grupos de diversas densidades: no se rechazan los grupos de baja densidad como valores atípicos, cosa que sí hacen algoritmos como DenStream, que sólo presentan una configuración global respecto a la densidad. Se recomienda al lector dirigirse a la Figura 3.4.

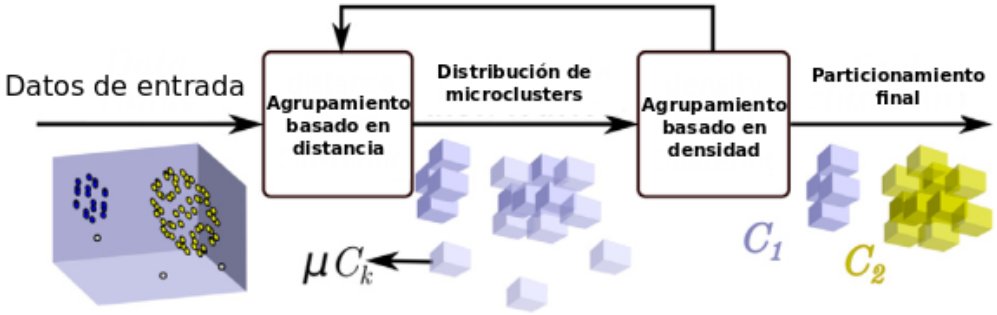

FIGURA 3.4: Forma de trabajo del algoritmo DyClee [47]

Más en detalle, la primera etapa de DyClee, opera a la velocidad del flujo de datos y crea *microclusters* que reúnen muestras de datos 'cercanas', según la norma L_1. Los *microclusters*

se almacenan en forma de representaciones a modo de 'resumen', incluyendo información estadística y temporal.

Luego, la segunda etapa de DyClee, tiene lugar a una frecuencia más baja, y analiza la distribución de los *microclusters* obtenidos hasta el momento. El hiperparámetro t_{global}, especifica el número de muestras que han sido procesadas por el algoritmo de la primera etapa, y es el utilizado para determinar cuándo debe ejecutarse la etapa *offline*. La densidad de un *microcluster* se considera baja, media o alta, y es utilizada para crear los grupos finales. De manera similar a DenStream, un grupo se define como el conjunto de *microclusters* conectados, donde cada *microcluster* interno presenta alta densidad, y cada *microcluster* ubicado en los límites, muestra una densidad media o alta. Ambas etapas funcionan como hilos paralelos e intercambian información mutuamente para mejorar el rendimiento.

Se enfatizan en este punto, las propiedades interesantes reunidas en el algoritmo presentado. Es por esto, que DyClee resultó relevante para el desarrollo de la actual tesina. Cabe mencionar, además, que dado que aún no se ha dado a conocer implementación alguna del algoritmo, se llevó a cabo el desarrollo de la misma para poner a prueba las características y aptitudes enunciadas, y así poder compararlo con otras técnicas, como alternativa a la hora de resolver diferentes problemas.

Se procederá entonces a explicar en detalle DyClee, haciendo foco en cada una de las etapas con mayor detenimiento.

3.3.1. Etapa basada en distancia

La primera etapa de agrupamiento tiene como objetivo formar grupos de objetos de tamaño pequeño y controlado: *microclusters*, basados en la similitud entre elementos.

Dado un elemento $X = [x_1, ..., x_d]$ con una marca de tiempo t_X y calificado por d características, un *microcluster* μC_k, es la representación de un grupo de objetos cercanos en todas las dimensiones, cuya ubicación se identifica mediante un índice i_{dk} y cuya información se resume en un vector de características CF_k de la siguiente forma:

$$CF_k = (n_k, LS_k, SS_k, tl_k, ts_k, D_k, Clase_k) \tag{3.4}$$

Donde $n_k \in N$ es el número de elementos en el microcluster k, $LS_k \in R^d$ es el vector que contiene la suma de cada atributo de los n_k objetos, $SS_k \in R^d$ es la suma cuadrada de atributos de los n_k objetos, $tl_k \in R$ es el momento en que se asignó el último objeto al *microcluster* en cuestión, $ts_k \in R$ representa el momento en que se creó el *microcluster*, D_k es la densidad *microcluster* y $Clase_k$ es la etiqueta del *microcluster* si se conoce. Usando LS_k,

SS_k y n_k se puede calcular la varianza y el promedio del grupo de elementos asignados al *microcluster* μC_k.

Los *microclusters* tienen forma de '*boxes*(cajas) d-dimensionales', ya que se usa la norma $L1$ como medida de distancia. El tamaño S^i a lo largo de cada dimensión i, se define como una fracción del rango de valores para el atributo correspondiente. El tamaño de cada atributo, se encuentra de acuerdo con la siguiente fórmula:

$$S^i = \phi^i |max^i - min^i|, \forall i = 1, ...d. \tag{3.5}$$

donde ϕ^i es una constante que establece la fracción.

Por otra parte, la densidad D_k de un *microcluster* μC_k se calcula utilizando el número actual de objetos n_k y el *hipervolumen* del tamaño delimitador $V = \prod_{i=1}^{d} S^i$, como se muestra en 3.6:

$$D_k = \frac{n_k}{V} \tag{3.6}$$

La estructura de DyClee comienza vacía y se construye mediante sucesivas llegadas de muestras. La primera muestra que llega se convierte en el centro del primer *microcluster* y, a partir de ese momento, se aplica el mecanismo de inserción que se describe a continuación. Este mecanismo utiliza la noción de *microcluster alcanzable*:

Definición 3.3. Microcluster alcanzable*: Un microcluster μC_x de una muestra de datos $X = (x^1, ..., x^d)$ es considerado como* alcanzable *para un elemento x si*

$$L\infty(x, c_k) \equiv max(x^i - c_k^i) < \frac{S^i}{2} \forall i = 1, ..., d \tag{3.7}$$

donde $c_k = [c_k^1, ..., c_k^d]^T$ es el centro de μC_k y los c_k^i se calculan a partir de LS_k como $c_k^i = \frac{LS_k^i}{n_k}$.

De esta manera, pensando la propiedad gráficamente, ante un elemento nuevo, se mira para cada microcluster *la dimensión para la cual la diferencia con el centro es mayor. Cabe mencionar que el elemento debería ubicarse a lo sumo en el borde del* microcluster *para considerar a éste como* alcanzable*. A modo de ejemplo, en la Figura 3.5 se observa cómo, para un elemento x, el microcluster graficado es considerado como* 'alcanzable' *ya que la mayor diferencia entre el centro del mismo y x, está dada en el eje x^1, y cumple con la restricción de ser menor a $S^1/2$.*

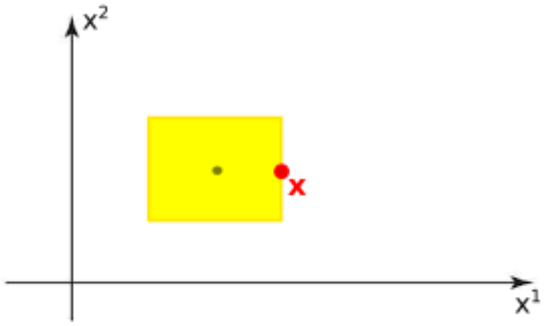

FIGURA 3.5: Propiedad de *Microcluster alcanzable* para un elemento x.

Así, las muestras de datos se insertan en el *microcluster alcanzable* más cercano con respecto a la distancia Manhattan. Una vez encontrado, el vector de características del *microcluster* seleccionado se actualiza con la información del nuevo elemento. Para acelerar la recuperación del *microcluster* más cercano al momento en el que llega una nueva muestra, los *microclusters* se almacenan en una de las siguientes dos listas. La primera lista es la lista de *microclusters Activos*, la lista *A*, en la que se guardan los *microclusters* de densidad media y alta: aquellos en los que los elementos entrantes se asignan con mayor frecuencia. Esta lista, es la primera fuente consultada para encontrar los *microclusters* alcanzables, dando prioridad en la búsqueda de los *microclusters* de alta y media densidad por sobre aquellos de baja densidad. Por lo tanto, siguiendo con lo antes mencionado, la segunda lista, la lista *O*, contiene a los *microclusters* de baja densidad: *Outliers*. Esta lista, se consulta solo si el nuevo dato no puede ser incorporado por ninguno de los *microclusters* en la lista A. En caso de no poder incorporarse a ninguno de los *microclusters* existentes en las dos listas, se crea un nuevo *microcluster* con la información del *data point*, utilizando su *timestamp* de llegada como el tiempo de creación del mismo. Se adjunta así a la lista O.

El Algoritmo 3.3 proporciona el pseudocódigo para la etapa de formación de *microclusters*, es decir, para la etapa basada en distancia.

3.3.2. Etapa basada en densidad

En esta segunda etapa, se ejecuta un agrupamiento basado en densidad sobre los *microclusters* creados en la etapa anterior, lo que permite que el algoritmo encuentre grupos de formas arbitrarias.

Como se indicó al comienzo, un grupo se define como un conjunto de *microclusters* conectados, donde cada *microcluster* interno es denso y cada *microcluster* 'límite' es de baja densidad.

A continuación, se incluyen algunas definiciones:

Algoritmo 3.3 DyClee - Etapa 1

mientras x en cola **hacer:**
    si hay mensaje de la etapa basada en densidad **entonces**
        actualizar la lista de microclusters con el mensaje
    fin si
    $alcanzables$ = microclusters en la lista A que son alcanzables para x
    si $alcanzables$ no es nula **entonces**
        encontrar el microcluster más cercano en la lista de alcanzables
        actualizar al más cercano con x
    en otro caso
        $alcanzables$ = microclusters en la lista O que son alcanzables desde x
        si $alcanzables$ no es nula **entonces**
            encontrar el microcluster más cercano en la lista de alcanzables
            actualizar al más cercano con la información de x
        en otro caso
            crear microcluster con x
            agregar microcluster a la lista O
        fin si
    fin si
    escribir la tendencia en un log
    si $actual\_t - t\_ult\_mensaje \geq t\_global$ **entonces**
        enviar las listas de microclusters a la etapa basada en densidad
        $t\_ult\_mensaje = actual\_t$
    fin si
fin mientras

Definición 3.4. Microclusters directamente conectados*: Los* microclusters μC_j *directamente conectados al* microcluster μC_k*, son aquellos cuyo centro* c_j *satisface la siguiente condición:*

$$\exists I_\phi = \{i_1, ..., i_{(d-\phi)}\} \subseteq \{1, ..., d\} \text{ tal que } max_i|c_j^i - c_k^i| < \frac{S^i}{2}, \forall i \in I_\phi \tag{3.8}$$

La idea es similar a aquella presentada en la propiedad de Microclusters alcanzables*, sólo que al tratarse de dos* microclusters*, se comparan los centros de los mismos, y entonces, en la dimensión para la cual la diferencia es mayor, debe cumplirse que ésta sea menor a la mitad del tamaño de los* microclusters*, para ese atributo. Si se visualiza esto gráficamente, se tiene que los* microclusters *deben estar superpuestos para poder estar así directamente conectados.*

Se recomienda al lector dirigirse a la Figura 3.6. En ella se ilustran dos *microclusters* los cuáles están *directamente conectados*, ya que la mayor diferencia entre los respectivos centros se da en el eje x^1, y cumple con la restricción de ser menor a $S^1/2$.

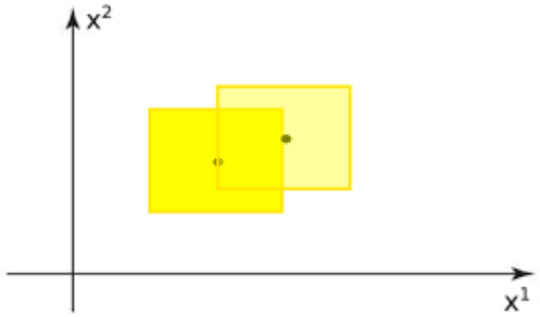

FIGURA 3.6: *Microclusters directamente conectados.*

Definición 3.5. Microclusters Conectados*: Un* microcluster μC_{k_1} *está conectado al* microcluster μC_{k_n}*, si hay una cadena de* microclusters $\mu C_{k_1}, \mu C_{k_2}, ..., \mu C_{k_n}$ *de forma que* μC_{k_i} *está directamente conectado con* $\mu C_{k_{i+1}} \forall i = 1, 2, 3, ..., n$

Definición 3.6. Cluster*: conjunto de* microclusters *que se encuentran conectados, donde cada uno de los* microclusters *internos es denso, mientras que aquellos que se encuentran en los límites son densos o semi-densos.*

A la hora de generar los clusters *finales, se tiene en cuenta entonces el carácter 'denso' de los* microclusters *formados hasta el momento, junto con sus conexiones.*

DyClee implementa dos enfoques diferentes para establecer el tipo de *microcluster*, considerando la densidad. Se brinda al usuario la posibilidad de utilizar tanto un enfoque global, como un enfoque local. El primero, permite detectar *clusters* de acuerdo con sólo dos niveles de densidad, mientras que el último, permite la detección de grupos con diferentes densidades. Para el desarrollo de este trabajo, se utilizó el primer enfoque, ya que el segundo excede

el alcance de la tesina. Así, la densidad de un *microcluster* queda definida como:

$$D_k = \frac{n_k}{V} \tag{3.9}$$

donde $V = \prod_{i=1}^{d} S^i$

La *densidad* en sí, es una magnitud escalar referida a la cantidad de masa en un determinado volumen de un objeto. Por esto, considerando n_k, a mayor cantidad de elementos, mayor densidad (ver Figura 3.7).

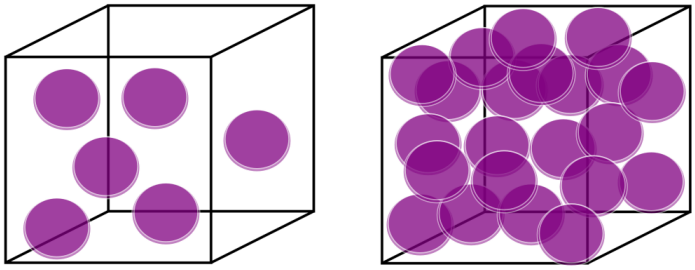

FIGURA 3.7: *Densidad*

Por último, la categoría en cuanto a densidad, para cada *microcluster*, se determina de la siguiente manera:

μC_k dense $\Leftrightarrow D_k \geq \text{median}\,(D_{G_1}) \wedge D_k \geq \text{avg}\,(D_{G_1})$
μC_k semi-dense $\Leftrightarrow D_k \geq \text{median}\,(D_{G_1}) \vee D_k \geq avg\,(D_{G_1})$
μC_k outlier $\Leftrightarrow D_k < \text{median}\,(D_{G_1}) \wedge D_k < \text{avg}\,(D_{G_1})$

Se puede notar que se utilizan tanto la media como la mediana, bajo la intuición de que en un conjunto de datos con distribución uniforme, estas dos medidas son aproximadamente iguales.

La Figura 3.8 muestra como difieren los resultados alcanzados siguiendo el enfoque local, de aquellos obtenidos por el global: se logra identificar 3 *clusters*, cuando de otra manera se identificarían solo 2, catalogando el último como ruido. A la izquierda se ve el primer *cluster* identificado. En el centro, se puede ver que se identifica el segundo *cluster*. Por último, a la derecha, el *cluster* de baja densidad es encontrado, finalizando la identificación de los tres *clusters*

Por otra parte, el Algoritmo 3.4 proporciona el pseudocódigo para la etapa de armado de *clusters*, es decir, para la etapa basada en densidad.

Resumiendo, entonces, DyClee resulta interesante, ya que es capaz de encontrar grupos no convexos, de densidad múltiple, pudiendo rechazar valores atípicos, implementando además un procedimiento de detección de novedades. El uso de un algoritmo basado en dos etapas, enfocadas cada una en la distancia y en la densidad, que corren a diferentes frecuencias,

Algoritmo 3.4 DyClee - Etapa 2

```
1: DMC ← microclusters tal que D_k > D_avg y D_k > D_mediana
2: vistos ← []
3: por cada microcluster en DMC hacer:
4:     si microcluster no se encuentra en vistos entonces
5:         agregar microcluster a la lista vistos
6:         si la etiqueta del microcluster figura como 'Sin clase' entonces
7:             cid ← tomar la siguiente etiqueta de cluster
8:             asignar cid como la etiqueta de microcluster vigente
9:         fin si
10:        microclusters_conectados ← todos los microclusters en el vecindario que se sola-
           pan con el microcluster actual
11:        mientras microclusters_conectados no esté vacía hacer:
12:            vecino ← quitar elemento de microclusters_conectados
13:            si vecino es denso entonces
14:                vecino.etiqueta ← cid
15:                nuevos_conectados ← todos los microclusters superpuestos en el vecindario de
                   vecino
16:            fin si
17:            por cada vecino en nuevos_conectados hacer:
18:                si vecino es denso entonces
19:                    agregar vecino a microclusters_conectados
20:                fin si
21:                vecino.etiqueta ← cid
22:            fin por
23:        fin mientras
24:    fin si
25: fin por
```

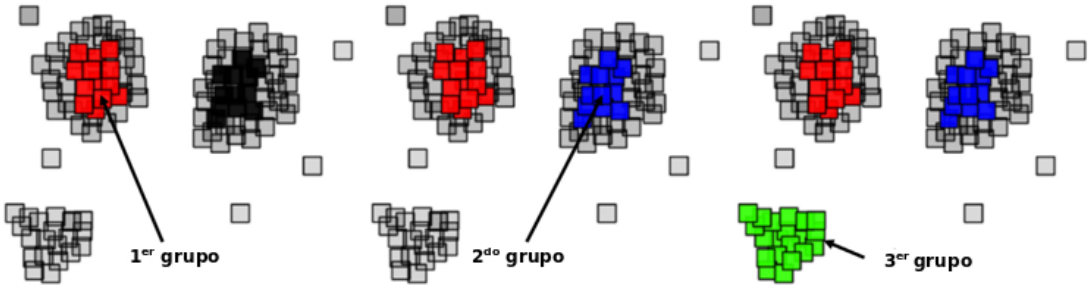

FIGURA 3.8: Enfoque local de DyClee: permite encontrar clusters de diversas densidades. Se debe leer de izquierda a derecha. [47]

permite agrupar los datos que llegan a alta frecuencia, ya que las tareas pesadas del algoritmo basado en la densidad, no se ejecutan a la velocidad de datos de entrada.

Se concluye así que DyClee logra un conjunto de propiedades valiosas que generalmente no se encuentran juntas en un mismo algoritmo. Es por esto, que es elegido y utilizado particularmente para el desarrollo de esta tesina.

3.4. Métricas para validar agrupamientos

No resulta trivial elegir el algoritmo de agrupamiento a utilizar cuando se desea agrupar un conjunto de datos. Esto se debe tanto a la variedad de algoritmos de agrupamiento existentes, y los diferentes enfoques presentes en la literatura. Una estrategia sencilla que suele emplearse, es simplemente ejecutar diversos algoritmos, con varias configuraciones diferentes de hiperparámetros, y luego seleccionar el mejor agrupamiento del conjunto de soluciones generado. Sin embargo, tampoco es sencillo seleccionar el mejor.

Seleccionar una 'buena' solución de *clustering* es una de las principales dificultades al agrupar datos, ya que hay muchas posibles soluciones de agrupamiento para un problema en particular. Por un lado, existen variados algoritmos de agrupamiento, que pueden producir particiones muy distintas para un mismo conjunto de datos. Más aún, un solo algoritmo de agrupamiento puede producir resultados diferentes dependiendo de los hiperparámetros elegidos.

La evaluación de la calidad de un modelo no supervisado, como es el caso de cualquier técnica de agrupamiento, resulta algo compleja ya que no cuenta con información adicional a la hora de formar los grupos. Esto no ocurre cuando se trata de medir el desempeño de un clasificador, generado mediante aprendizaje supervisado. En el caso de los modelos predictivos se utiliza la respuesta esperada de cada ejemplo para calcular métricas que determinen cómo se está

comportando el modelo entrenado. La más utilizada suele ser el porcentaje de aciertos o precisión de las predicciones. Estas métricas tienen por finalidad determinar si el modelo logró aprender la asociación entre los datos de entrada y la salida esperada.

En cambio, en problemas de agrupamiento, seleccionar la solución con el número óptimo de grupos, y la disposición óptima de los elementos de entrada en los grupos finales, es un problema abierto, ya que a menudo no existe un conocimiento de *ground truth*, o conocimiento de dominio; no se dispone de tales etiquetas, solo es posible basar las estimaciones de calidad en los datos y la partición bajo consideración.

Así, para conjuntos de datos de baja dimensionalidad, la visualización de los mismos puede ayudar a seleccionar la mejor solución de agrupamiento. Sin embargo, la mayoría de los conjuntos de datos contienen una gran cantidad de atributos, y la proyección en dimensiones más bajas para evaluar la calidad del *clustering* puede ser engañosa. Por lo tanto, es deseable el uso de técnicas cuantitativas para identificar el 'mejor' agrupamiento. Éstas, lo que hacen es mapear un agrupamiento a un valor numérico que indica su calidad.

El objetivo, entonces, es seleccionar una métrica que de lugar a las 'mejores' soluciones de agrupamiento, para que así el usuario pueda seleccionar una solución de compromiso.

Existe una amplia gama de métricas para evaluar la calidad de una solución de *clustering* dada. Las métricas *internas* determinan la calidad de la solución utilizando solo propiedades de la estructura de los grupos que se han formado; por ejemplo, cohesión de elementos dentro de un *cluster*, o separación entre distintos *clusters*. Por otro lado, las medidas de calidad *externas* hacen referencia a una solución de agrupamiento conocida u óptima en la evaluación.

Dado que en problemas de agrupamiento reales, una solución de agrupamiento óptima a menudo no está disponible, resulta interesante utilizar solo medidas de calidad internas. Para que esta estrategia sea válida, se requiere una métrica interna que permita una comparación justa entre los algoritmos de agrupamiento a utilizar. Además, la métrica empleada debe ser robusta. Es decir, hay que asegurarse de que el aumento o bien decremento del valor de la función objetivo esté altamente correlacionado con los cambios en la calidad de la solución.

Entonces, retomando, las métricas internas solo se basan en propiedades intrínsecas al conjunto de datos. Estas métricas, son formulaciones matemáticas que capturan algunas ideas sobre cómo debería ser una buena partición de los datos. Éstas, deben permitir la comparación de particiones con un número diferente de *clusters*. Por ejemplo, la suma de diferencias elevadas al cuadrado considerando cada elemento y el centoide dentro de determinado *cluster*, función minimizada cuando se utiliza el K-Means, no se puede emplear para comparar particiones, dado que su valor disminuye a medida que aumenta el número de *clusters*, alcanzando el valor ideal de 0, para una solución en la que cada *cluster* está formado por un único elemento.

En lo que queda de esta sección, se detallarán diferentes métricas frecuentemente utilizadas para problemas de agrupamiento, listando sus ventajas y desventajas. A su vez, se explicará

por qué es elegida la métrica DBCV para evaluar los resultados obtenidos con los distintos algoritmos especificados anteriormente en el documento.

3.4.1. Métricas basadas en distancia

Índice Silhouette El índice SWC [53] es una métrica que evalúa qué tan bien están asignados los objetos a los grupos de los que actualmente son miembros. Primero, debe calcularse la distancia promedio entre cada elemento i y los otros miembros de su grupo, C_i:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j) \tag{3.10}$$

Se puede interpretar $a(i)$ como una medida de qué tan bien está asignado el elemento i a su grupo. Cuanto menor es su valor, mejor es la asignación.

En segundo lugar, se calcula la distancia promedio 'más pequeña' entre cada elemento i y los elementos en su grupo vecino más cercano:

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j) \tag{3.11}$$

El valor de $b(i)$ mide la distancia promedio entre el elemento i y los elementos del grupo más cercano. A mayor valor, más distinto es el elemento i con respecto a los de su 'grupo vecino'.

Entonces, el índice SWC de un objeto, mide qué tan bien se encuentra asignado cada elemento en su *cluster*, teniendo en cuenta su distancia promedio a sus compañeros de grupo y la distancia promedio a los elementos del *cluster* vecino. Está dado por la siguiente fórmula:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \qquad si |C_i| > 1 \tag{3.12}$$

De la definición anterior puede notarse que $-1 \leq s(i) \leq 1$.

Entonces, si el valor que toma s(i) es cercano a 1, se deduce que el ejemplo i está bien clasificado. Si por otro lado, el valor es negativo, entonces se puede ver que el ejemplo i está mal clasificado, y debería pertenecer al *cluster* vecino. Luego, si el valor obtenido es cercano a 0, no está claro si i debería pertenecer al *cluster* vecino o al *cluster* actual; se tiene un indicio de que existe superposición de *clusters*.

El SWC total, para calificar la solución de agrupamiento en general, queda definido como el promedio de los valores individuales que toma el índice para cada elemento:

$$SWC(P) = \frac{\sum_{i=1}^{n} s(i)}{n} \tag{3.13}$$

Para esta métrica, entonces, son deseables los valores más altos, es decir, positivos y cercanos a 1.

Entre las desventajas que trae el índice, se encuentra el hecho de que devuelve valores mayores para grupos convexos, no siendo útil aplicarlo para la evaluación de *clusters* que se forman siguiendo un esquema por densidad, por ejemplo.

Luego, en cuanto a las ventajas que presenta, se destaca el hecho de que es simple de computar, y que es sumamente intuitivo: otorgará valores más altos cuando los grupos sean densos, cohesivos, y estén bien separados, lo cual se relaciona al concepto tradicional de *clusters*.

Índice Davies-Bouldin El índice Davies-Bouldin (DB) [54] se basa en la relación entre una medida de distancia *intra-cluster*, y una *entre-clusters*. En lo que respecta a la distancia *intra-cluster*, para un grupo C_i, considerando que x es un elemento n-dimensional asigando al mismo, se tiene:

$$S_i = \left(\frac{1}{|C_i|} \sum_{j=1}^{|C_i|} |x_j - A_i|^p \right)^{1/p} \tag{3.14}$$

Donde A_i es el centroide de C_i, y $|C_i|$ es el tamaño de éste. Usualmente, p toma el valor de 2, lo cual hace que la función de distancia entre el centroide del *cluster* y cada elemento perteneciente al mismo, sea la distancia Euclídea.

Luego, en lo que refiere a una medida de distancia entre grupos, para dos *clusters* dados, C_i y C_j, se define como:

$$M_{i,j} = ||A_i - A_j||_p = \left(\sum_{k=1}^{n} |a_{k,i} - a_{k,j}|^p \right)^{\frac{1}{p}} \tag{3.15}$$

Donde $a_{k,i}$ es el k-ésimo elemento de A_i, considerando n dimensiones.

Luego, se calcula una medida de similitud $R(C_i, C_j)$, teniendo en cuenta las distancias previamente detalladas:

$$R_{i,j} = \frac{S_i + S_j}{M_{i,j}} \tag{3.16}$$

Esto, se utiliza para definir así D_i:

$$D_i \equiv \max_{j \neq i} R_{i,j} \tag{3.17}$$

De esta manera, si N es el número total de *clusters*, el índice DB queda definido por la siguiente fórmula:

$$DB \equiv \frac{1}{N} \sum_{i=1}^{N} D_i \tag{3.18}$$

D_i representa el peor de los casos. Toma un valor igual a $R_(i, j)$, considerando el grupo más similar a i.

Dicho de otra manera, el índice DB determina para cada grupo, con qué otro grupo es más similar, y mide esta similitud, utilizando una medida como la que se acaba de presentar. El índice proporciona un promedio de estas similitudes máximas de *cluster*.

En lo que respecta al entendimiento del resultado obtenido, si el valor de DB es bajo, entonces eso significa que los grupos son muy diferentes entre sí. Es un indicativo de que los grupos están bien separados y son cohesivos.

Entre las ventajas que se presentan a la hora de utilizar el índice, se destaca el hecho de que es considerablemente simple de computar, y también que el mismo es calculado usando sólo atributos inherentes al conjunto de datos. Luego, en cuanto a las desventajas, cabe mencionar que, como sucede con el índice presentado anteriormente, el resultado obtenido con DB es generalmente mejor para *clusters* convexos, globulares, por cómo está definido el mismo.

Por último, un buen valor reportado por esta métrica, no implica necesariamente que se logró la mejor extracción de información a partir de los datos de entrada.

3.4.2. Métricas basadas en densidad

Las métricas discutidas hasta ahora tienen un sesgo hacia la validación de grupos con forma esférica. En un sentido amplio, se pueden caracterizar por favorecer particiones con una mayor 'similitud dentro de cada grupo', que la 'similitud entre grupos', por supuesto, en base a diferentes formulaciones de tales conceptos. Dado este sesgo particular, estas medidas no pueden emplearse, al menos no directamente, para la validación de los resultados de agrupamiento basados en la densidad, como los generados por DBSCAN.

La gran mayoría de las métricas se basan en la idea de calcular la relación entre la dispersión dentro del grupo (compactibilidad) y la separación entre grupos.

Como se mencionó anteriormente, las métricas que siguen esta definición se han diseñado para la evaluación de grupos convexos (grupos globulares), y fallan cuando se aplican para validar grupos no convexos, de forma arbitraria. Tampoco están definidas para considerar ruido. Estas métricas son principalmente adecuadas para particiones con grupos esféricos, que presentan una variación mínima dentro de cada uno de ellos, mientras que una separación máxima entre ellos, en términos de la definición del criterio elegido. En este sentido, tales medidas son más apropiadas cuando los grupos en un conjunto de datos se extraen de k distribuciones gaussianas, en las que cada distribución está asociada con uno de los k grupos en los datos.

Son pocos intentos hasta ahora que se hicieron para desarrollar métricas en el contexto del agrupamiento considerando formas arbitrarias. Sin embargo, muchas de las métricas propuestas no tienen en cuenta directamente los conceptos basados en densidad. Proporcionan solo estimaciones aproximadas de la calidad del *clustering*, basadas en la distribución de ciertos elementos representativos dentro de los grupos finales, como ser las medidas de cohesión y separación, e introducen diversos parámetros que no son deseables para la tarea de validación. Por ejemplo, el índice de validación CDbw [55], que se propuso para evaluar grupos de formas arbitrarias, define la 'compactibilidad' y 'separación' de grupos, de forma similar a como lo hacen los índices de validación de grupos que se usan para grupos esféricos. La única diferencia, es que simplemente utiliza múltiples elementos en lugar de un solo punto como representantes en cada grupo.

Los trabajos de Maria Halkidi y sus colegas son probablemente los más conocidos en el ámbito de la validación de agrupamientos basados en densidad. Con ese fin, los autores introdujeron tres medidas diferentes destinadas a la validación de los resultados: SD [56], SDbw [57] y la anteriormente mencionada CDbw.

El índice *SD* se basa en dos conceptos: *dispersión* promedio dentro de los grupos y *separación* entre grupos, considerando la varianza y la distancia entre los centroides de los *clusters*. La métrica se define como la suma de estos dos términos.

Motivados por tener en cuenta las variaciones de densidad entre los grupos, *SDbw* tiene una definición similar al índice SD, difiriendo el concepto de dispersión dentro de los grupos,

y separación entre grupos. Este último índice se define también como la suma de esos dos términos. Se puede notar que ninguna de estas métricas puede tratar con grupos de formas arbitrarias, dado que consideran el centro de los grupos dentro de sus definiciones, el cuál no es un punto representativo en los grupos no convexos, basados en la densidad.

La métrica *CDbw* refleja un intento de superar varios de los inconvenientes encontrados en SD y Dbw. CDbw es frecuentemente empleada para la validación basada en densidad. Similar a SD y SDbw, también evalúa la compactibilidad y separación de grupos obtenidos por un algoritmo de agrupamiento. Sin embargo, el enfoque adoptado por CDbw, consiste en considerar múltiples elementos representativos por grupo en lugar de uno solo, y luego capturar la forma arbitraria de un cluster basándose en la distribución espacial de dichos elementos. CDbw tiene, por otro lado, varios inconvenientes importantes relacionados con los múltiples representantes que emplea. El primero de estos inconvenientes, es que la métrica no especifica cómo determinar el número de elementos representativos para cada grupo. Dado que grupos de diferentes tamaños, densidades y formas están bajo evaluación, definir un número fijo de representantes para todos los grupos no parece ser el mejor enfoque. Incluso, si se emplea un solo número de objetos representativos para todos los grupos, como sugieren los autores, este número aún puede ser crítico para el rendimiento de la métrica por ser un parámetro que genera variabilidad, lo cual no es deseable. La adopción de diferentes enfoques para encontrar elementos representativos se puede ver no solo como otro parámetro, sino como una fuente importante de inestabilidad, dado que dos grupos diferentes de representantes para un mismo *cluster*, los cuáles contienen el mismo número de muestras pero fueron generados por diferentes técnicas, pueden conducir a diferentes resultados. Esto añade una complejidad extra a la hora de realizar una evaluación de resultados.

En 2014 se introdujo en la literatura una nueva métrica basada en el concepto de densidad, denominada DBCV[58]. La misma, es considerada la más óptima para evaluar agrupamientos con *clusters* de formas arbitrarias, y teniendo en cuenta la incapacidad de las métricas presentadas anteriormente como intento de solución al problema planteado, se optó por elegir a ésta como aquella a utilizar en nuestras evaluaciones, principalmente para poder evaluar grupos no convexos. Se procederá a detallarla.

3.4.3. DBCV

Como se mencionó previamente en el documento, DBCV [58] es bastante diferente a aquellas métricas discutidas con anterioridad. DBCV tiene la capacidad de manejar *clusters* con diferentes densidades y formas, ya que por un lado se basa en la noción de una distancia llamada *all-points-core-distance*, que permite capturar propiedades de densidad, y por otro lado hace uso de un árbol de expansión mínima (MST) por *cluster* para captar la forma de los mismos. Por esto último, es capaz de manejar formas arbitrarias. La *all-points-core-distance* de un elemento x que pertenece al grupo C_i se define de la siguiente manera:

$$a_{pts}\ \text{coredist}\,(x) = \left(\frac{\sum_{j \in C_i, j \neq x}^{|c_i|} \left(\frac{1}{dist(x,j)} \right)^d}{|c_i| - 1} \right)^{\frac{-1}{d}} \tag{3.19}$$

Donde para $x \in C_i$, $dist(x, j)$ es la distancia Euclídea al vecino j, y d es la dimensionalidad de x. La *all-points-core-distance* de todos los elementos puede verse como la inversa de la densidad de un objeto en su grupo. Usando esta distancia en particular, se define la *reachability-distance* entre dos objetos para incorporar sus propiedades de densidad:

$$\begin{aligned} d_{\text{reach}}\,(x_i, x_j) = \text{máx}\,\{&a_{\text{pts}}\ \text{coredist}\,(x_i) \\ &a_{pts}\,\text{coredist}\,(x_j)\,, d\,(x_i, x_j)\} \end{aligned} \tag{3.20}$$

Para cada grupo, se construye un grafo de *reachability-distance*, con los elementos del *cluster* como vértices, y las distancias como pesos. Así, se crea un MST para cada uno de estos grafos; un MST para cada grupo. La *dispersión* de un grupo se define entonces como el peso máximo de los bordes internos del correspondiente MST. La *separación* entre un par de *clusters*, teniendo en cuenta la densidad, se define como la *reachability-distance* mínima entre los nodos internos de los MST correspondientes a cada *cluster*. Teniendo en cuenta la menor densidad dentro de un grupo (DSC) y la densidad de separación entre pares de grupos (DSPC), se define el índice para un solo grupo de la siguiente manera:

$$V_c\,(c_i) = \frac{\text{mín}_{1 \leq j \leq |C|, j \neq i}\,(DSPC\,(c_i, c_j)) - \text{DSC}\,(c_i)}{\text{máx}\left(\text{mín}_{1 \leq j \leq |C|, j \neq i}\,(\text{DSPC}\,(c_i, c_j))\,, \text{DSC}\,(c_i)\right)} \tag{3.21}$$

Luego, el índice total, para evaluar una agrupación, queda definido como el promedio ponderado de los índices individuales obtenidos para cada *cluster*:

$$DBCV(C) = \sum_{i=1}^{K} \frac{|c_i|}{N} V_c\,(c_i) \tag{3.22}$$

Donde N es el número de elementos en evaluación.

Nuevamente, esta medida combina la compactibilidad y la separación de los grupos, pero estos conceptos ahora se definen como propiedades de los grafos construidos en el espacio transformado de *reachability-distances*.

Su complejidad es $O(N^2)$, y los valores que otorga están en el intervalo $[-1, 1]$, considerando el índice debe maximizarse.

Es importante remarcar que, al utilizar DBCV para evaluar la validez de una partición, se realiza una suma ponderada de los resultados individuales obtenidos para cada grupo, siendo los pesos proporcionales al tamaño de los *clusters*. Los elementos identificados como *ruido*, simplemente no contribuyen a esta suma. Esto, reduce la puntuación máxima que se puede

obtener para una partición. Por ejemplo, si el $50\,\%$ de los elementos se identifican como ruido, la puntuación DBCV estará en el intervalo $[-0{,}5, 0{,}5]$ y estaría entre $[-1, 1]$ si no se identificara ruido.

Como principal ventaja, se destaca el hecho de que por cómo está definida la métrica, DBCV resulta ideal a la hora de evaluar la calidad de particiones obtenidas con algoritmos que no se limitan a la generación de *clusters* globulares. Al estar basada en el concepto de densidad, es la mejor métrica a utilizar cuando se tienen grupos de formas arbitrarias.

Como desventaja, puede verse que no es tan simple de computar. Además, si se van a comparar soluciones provistas por algoritmos que no contemplan el concepto de ruido, con aquellas generadas por algoritmos que sí lo hacen, puede notarse cómo es que la métrica penaliza a éstos últimos.

3.5. Conclusiones

En este capítulo se ha introducido el concepto de *flujos de datos*, y por qué resulta un desafío el poder agruparlos. Se mencionó que la gran cantidad de información que se genera constantemente, hace que no sea factible persistir los datos, ni tampoco mantenerlos en memoria para poder procesarlos varias veces; se requiere ir resumiendo los datos a medida que van llegando, y trabajar en base a estos resúmenes. Además, se introdujo un *componente de olvido*, el cual permite mantener la información actualizada, olvidando conceptos o nociones que se quedan en el pasado. Así, es posible notar las tendencias en tiempo real, por lo que se puede trabajar con datos no estacionarios: se es capaz de seguir su evolución.

Se planteó un enfoque totalmente diferente al agrupamiento en lotes, o *batch*, que caracteriza al *clustering* tradicional, donde los datos se encuentran disponibles desde un principio, pudiendo recorrer el conjunto de datos original las veces que sean necesarias.

Seguido a esto, se detallaron distintos algoritmos que tratan de resolver la problemática descrita previamente, planteando ventajas y desventajas de cada uno. CluStream, extiende el conocido K-Means, por lo que hereda las características de éste: es simple de comprender, pero sólo encuentra grupos globulares, convexos. Requiere que se le parametrice la cantidad de grupos a formar y, además, mantiene un número fijo de q *microclusters* en memoria. DenStream se basa en DBSCAN. Al seguir un enfoque basado en densidad, permite obtener grupos con formas arbitrarias. No requiere la especificación previa del número de *clusters* a identificar, pero sí requiere que se establezcan parámetros de densidad globales, que permitan definir cuándo dos elementos están lo suficientemente 'juntos' como para pertenecer al mismo grupo, y cuándo se está en presencia de ruido.

Luego, se introdujo DyClee, un algoritmo recientemente incorporado en la literatura, que presenta características destacables que hacen que sea interesante la elección del mismo por sobre el resto de algoritmos capaces de lidiar con flujos. Este algoritmo, es capaz de definir

la densidad de los elementos teniendo una noción global o local, con lo que, a diferencia del algoritmo anterior, puede encontrar grupos de densidades diferentes.

Por otra parte, también se debatieron las métricas que suelen utilizarse para problemas de agrupamiento, detallando por qué aquellas tradicionales no sirven para evaluar particiones con grupos de formas arbitrarias. Se explicaron distintas métricas que aspiraron a solucionar dicha problemática, y se planteó el por qué de la elección de DBCV como índice a utilizar en las posteriores evaluaciones: al estar definida en base al concepto de *densidad*, resulta ideal cuando se tienen grupos no convexos. Además, no hay forma de que se genere variabilidad, teniendo en cuenta los resultados, ya que no requiere que se ajuste ningún hiperparámetro.

En el capítulo siguiente, se realizará una comparativa práctica de los algoritmos detallados a lo largo de todo el documento. Se hará hincapié en los hiperparámetros que éstos utilizan, y cómo es que afectan al resultado final. Además, se realizará el análisis de un problema del mundo real, donde se ve clara la necesidad de adaptación *en línea* del algoritmo a utilizar. Así, quedará demostrada una forma sencilla de aplicar los algoritmos de agrupamiento de flujos de datos en un caso puntual.

Capítulo 4

Resultados obtenidos

En este capítulo, se ilustran diferentes resultados de agrupamiento obtenidos para diversos casos de prueba, usando los algoritmos detallados en las secciones anteriores: tanto aquellos dedicados al análisis de *flujos de datos*, como aquellos pertenecientes a la categoría de *clustering tradicional*, los cuáles se aplican sobre un conjunto de datos completo. El capítulo, está dividido en dos partes: por un lado, se utiliza una sección para explicar con profundo nivel de detalle, los hiperparámetros a ajustar de cada algoritmo explicado a lo largo del documento. Por otro lado, se dispone de una sección particular para analizar la *performance* de los algoritmos mencionados aplicados en diversos escenarios.

Para los experimentos, se utilizan conjuntos de datos de prueba presentes en la literatura, ampliamente utilizados en el sector, para así dejar en evidencia la forma en que trabaja cada algoritmo, y el tipo de resultado que genera cada uno. Se hace hincapié en los hiperparámetros que éstos utilizan, y cómo es que la configuración de éstos afecta al resultado final.

Por último, se realiza el análisis de un problema real, utilizando un conjunto de datos que contiene coordenadas GPS de taxis en Suecia. El mismo, es empleado para simular un flujo de datos, pudiendo así mostrar una aplicación de la técnica de *clustering dinámico*, particularmente utilizando DyClee, sobre un caso real y concreto, en dónde queda demostrada la utilidad de poder realizar un agrupamiento en línea y dinámico, que se mantiene actualizado y se adapta a los cambios en la distribución de los datos de entrada.

4.1. Detalle de hiperparámetros

En esta sección, se detallan específicamente cuáles son los hiperparámetros que toma cada algoritmo, y qué efecto produce cada uno de estos en el resultado final. Se brinda un indicio de cómo se debe ajustar cada uno de acuerdo al efecto que se quiera causar en el agrupamiento final.

4.1.1. K-Means

k
Número de *clusters* a formar.

max_iter
Número máximo de iteraciones del algoritmo K-Means para una sola ejecución.

El hiperparámetro más importante del algoritmo K-Means es k por ser el que determina la cantidad de grupos a generar. Es así que, si se considera el caso extremo donde no hay ningún grupo natural subyacente en los datos, el algoritmo de todas formas genera la cantidad de grupos o *clusters* preestablecida.

En cuanto a max_iter, su valor se utiliza para limitar la cantidad de iteraciones realizadas por el algoritmo.

Es importante considerar también el algoritmo a utilizar a la hora de elegir los k centroides iniciales. Pueden hallarse en la literatura distintas alternativas para esto [32] [33] [34].

En la Figura 4.1 se ilustra cómo, cuando en realidad hay dos grupos globulares bien formados, si al algoritmo se le indica $k = 3$, se fuerza la generación de 3 *clusters*.

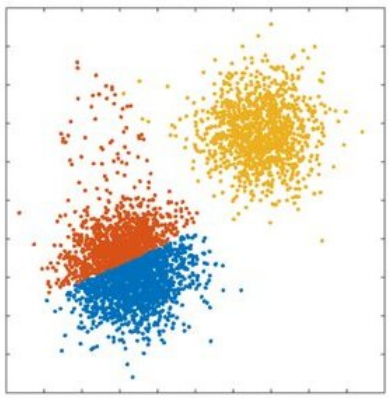

FIGURA 4.1: Resultado de aplicar el algoritmo K-means utilizando $k = 3$ [59]

Esta es una gran desventaja del método, ya que se requiere un conocimiento del dominio, o *ground truth*, previo a realizar el agrupamiento. Si este es el caso, la gran ventaja de utilizarlo está dada por la simplicidad del mismo.

Como se mencionó anteriormente, un enfoque que suele tomarse es probar el algoritmo con diferentes valores de k, y quedarse con aquel que provee la mejor solución, basándose en una métrica en particular, como ser aquellas detalladas en el Capítulo 2 .

4.1.2. DBSCAN

eps

La distancia máxima entre dos muestras para que una se considere lo suficientemente próxima a la otra, y así de manera que forme parte de su vecindario.

MinPts

Número mínimo de muestras requeridas en un vecindario para que el elemento en cuestión sea considerado como un objeto *core*. Incluye al elemento en sí.

El valor agregado que presenta DBSCAN por sobre K-Means, es el hecho de que tiene la capacidad de poder obtener grupos no convexos, de formas arbitrarias, como aquellas presentadas en la Figura 4.2, sin necesidad de establecer de ante mano la cantidad de grupos a encontrar.

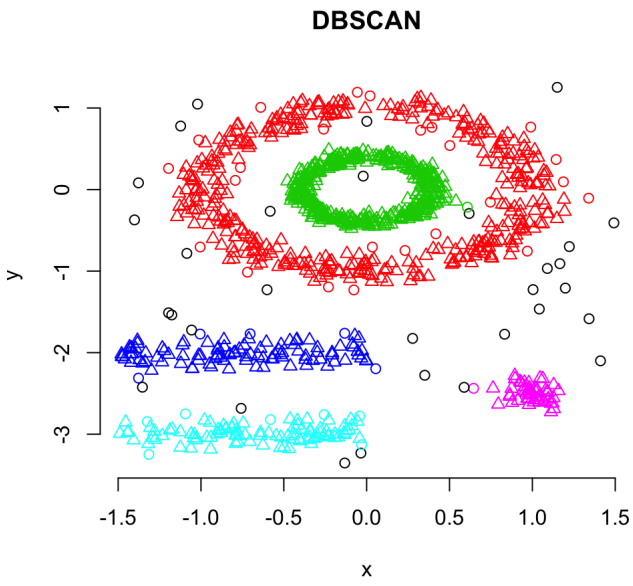

FIGURA 4.2: Agrupamiento de ejemplos por densidad [60]

En lo que respecta a sus hiperparámetros, se puede observar que, si se elige un valor demasiado alto para eps, se obtendrán grupos muy grandes, que en su interior incluyen varios grupos naturales. Por otro lado, si se opta por un valor sumamente chico para eps, se generarán múltiples grupos, que no representarán la forma en la que se distribuyen los elementos originales. La Figura 4.3 [1], ilustra lo mencionado, mostrando cómo diversas configuraciones de eps afectan al resultado final.

Por otra parte, en lo que respecta al hiperparámetro $MinPts$, si se elige un valor muy elevado, cada elemento en particular necesitará tener demasiados elementos 'vecinos' (considerando

[1] `https://github.com/onofricamila/ClusteringPlotterAndDBCVCalc/blob/master/notebooks/dbscan_eps_plotter.ipynb`

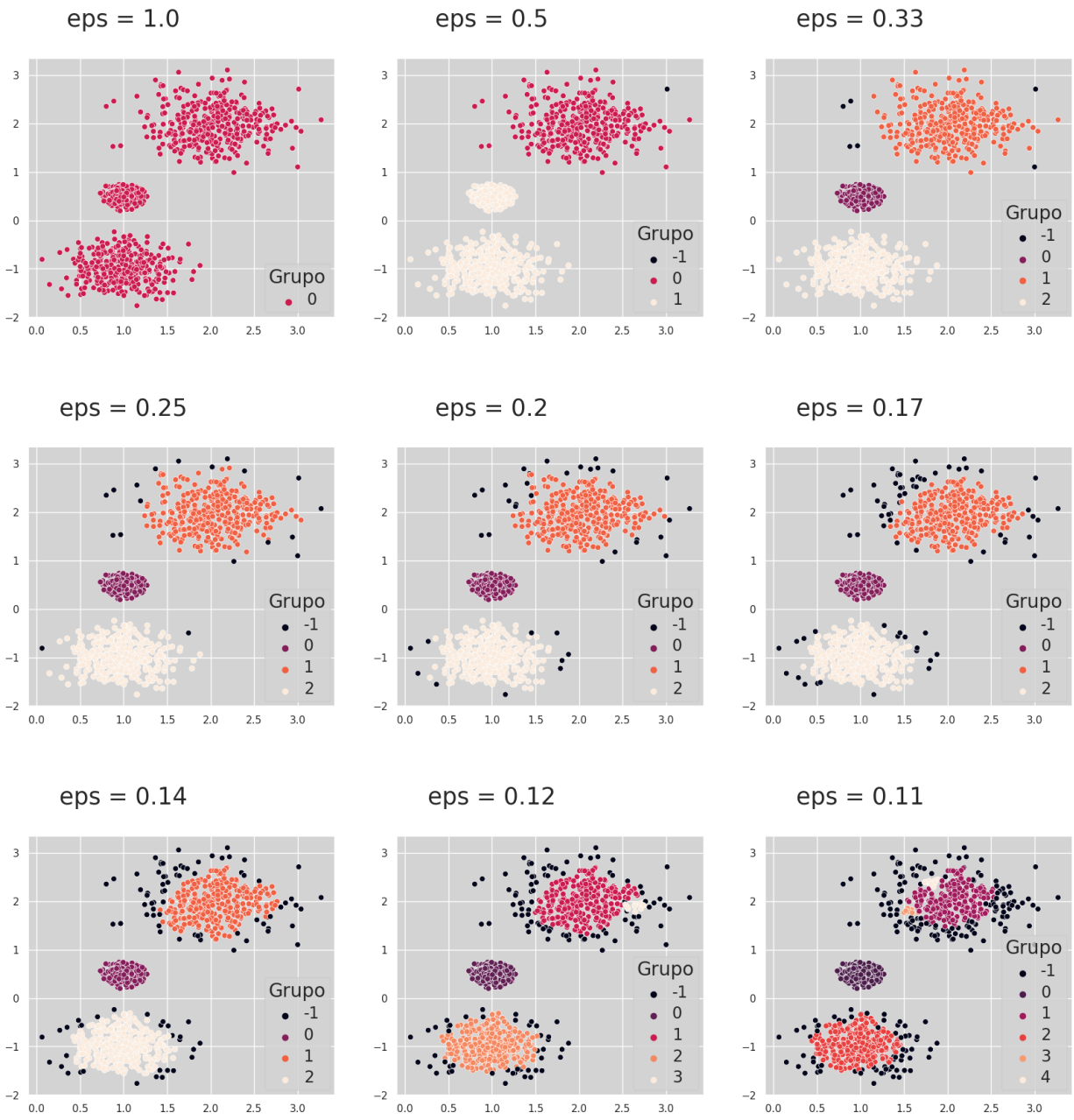

FIGURA 4.3: Distintas configuraciones de eps y su influencia en el resultado obtenido al utilizar DBSCAN.

un vecindario delimitado por eps), para poder hacer que a partir de él crezca un *cluster*. Ésto, resultaría muy restrictivo, ya que produce un efecto negativo en el resultado final, haciendo que se obtengan pocos *clusters* finales y que la gran mayoría de muestras sean consideradas como ruido. Por el contrario, si se elige un valor muy chico para este hiperparámetro, se pierde la noción de valores atípicos, ya que casi cualquier muestra podría ser considerada como *core*; *clusters* naturales estarían incorporando ruido en sus grupos.

4.1.3. CluStream

Horizonte (h)
: controla el tamaño de la ventana de tiempo en segundos, utilizada para determinar cuáles son los *microclusters* que expiran, o bien pierden relevancia.

Cluster radius multiplier (r)
: controla qué tan grande es el radio de un *microcluster*, para permitir o no que se agreguen nuevos elementos. Toma el valor 2 por defecto.

Búfer inicial (b)
: controla el tamaño de *buffer* de elementos iniciales, a partir del cual se crean luego los primeros *microclusters*.

Número de microclusters (m)
: controla cuántos *microclusters* serán mantenidos en memoria para resumir la información entrante. Suele tomar un valor menor a la cantidad de elementos que se espera recibir, pero mayor a la cantidad de *clusters* que se van a formar.

Número de clusters (k)
: controla cuántos grupos se generarán mediante la aplicación del algoritmo tradicional K-Means por sobre los *microclusters* obtenidos hasta el momento, de forma *offline*.

Para el algoritmo CluStream, entra en juego el *factor de olvido*. Relacionado a este componente, se encuentra el hiperparámetro $horizonte$. A modo de ejemplo, si se elige un valor muy alto para éste, no se aplicará olvido alguno, y no podrá aprovecharse la adaptabilidad que el algoritmo permite.

La Figura 4.4 ilustra el resultado de elegir un valor muy alto para el hiperparámetro $horizonte$ [2] [3]. En ella puede verse que no se alcanza nunca a la cantidad requerida de elementos procesados para comenzar a olvidar: los *microclusters* nunca 'envejecen', por lo que se termina modelando constantemente la información total acumulada, perdiendo la posibilidad de que el resultado se muestre actualizado, solo con la información relevante.

Si por el contrario se opta por un valor muy chico para este hiperparámetro, entonces el algoritmo procede a olvidar casi toda la información pasada, descartando datos que pueden ser importantes.

Por otra parte, en lo que respecta al hiperparámetro *Búfer Inicial*, el tamaño del mismo determinará en qué momento se empiezan a formar los *microclusters* iniciales, y luego, a medida

[2] `https://github.com/onofricamila/MOAOnlineClustering/blob/master/src/utils/clusterers/ClustreamClusterer.java`

[3] `https://github.com/onofricamila/ClusteringPlotterAndDBCVCalc/blob/master/utils/clustering_managers/time_series_mngrs/clustream_clustering_manager.py`

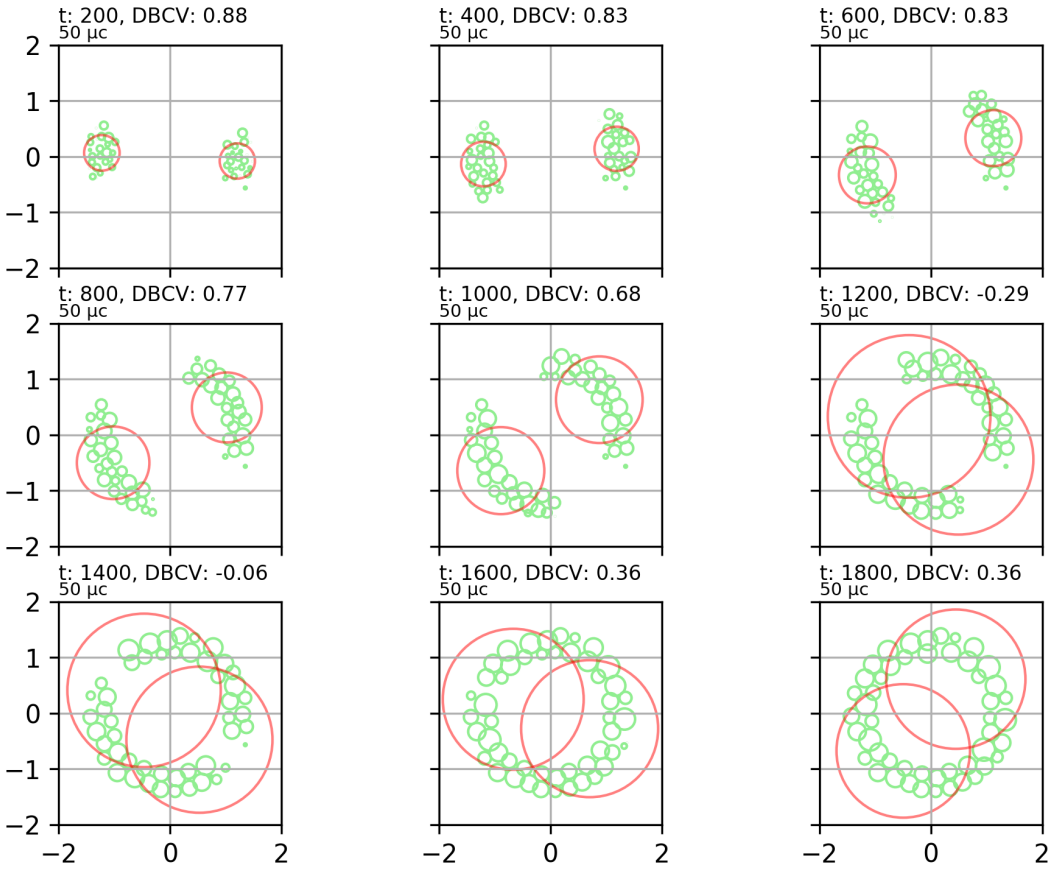

FIGURA 4.4: Resultados obtenidos para el algoritmo CluStream, donde t indica la marca de tiempo, y el hiperparámetro $horizonte$ toma un valor sumamente alto

que lleguen elementos nuevos, se evaluará si los mismos pueden incorporarse dentro de los *microclusters* actuales o deben crearse nuevos *microclusters*, que representen esa información entrante.

Adicionalmente, el hiperparámetro *Cluster radius multiplier*, afecta a la forma en que se organizan los *microclusters*. Si se opta por un valor muy chico, entonces prácticamente se necesitaría un *microcluster* por cada elemento entrante, ya que la condición para que una muestra sea incluida en un *microcluster* existente resultaría muy restrictiva. En cambio, si se elije un valor grande para *Cluster radius multiplier*, se necesitarían pocos *microclusters* para modelar la información entrante, ya que la condición de pertenencia sería muy laxa, y un mismo *microcluster* estaría incluyendo elementos no tan parecidos entre sí.

Por último, los hiperparámteros que determinan la cantidad máxima de *microclusters* a formar y la cantidad final de *clusters* a generar, están relacionados totalmente con la etapa del algoritmo que utiliza K-Means. Cabe mencionar que en el primer caso, si se opta por un valor chico para la cantidad de *microclusters* entonces se tendrá que intentar combinar o bien eliminar con más frecuencia *microclusters* existentes para comenzar a modelar la información nueva que ingresa y no se parece a la información resumida hasta el momento como para ser incluida en alguno de los *microclusters* vigentes. Si el valor que toma este hiperparámetro por el contrario es muy alto, entonces se podrán formar libremente más *microclusters* (de acuerdo al hiperparámetro *Cluster radius multiplier*), corriendo el riesgo de mantener resumida la información que ya perdió relevancia y debería ser descartada.

4.1.4. DenStream

epsilon

define el vecindario, el radio máximo de *microclusters* ($r <= epsilon$). Toma valores en el rango $[0, 1]$.

MinPts

cantidad de elementos mínima que requiere un *microcluster* para que éste sea considerado *core* ($w >= MinPts$). Definida como el peso w. Rango $[0, +inf]$.

beta

multiplicador para $MinPts$, el cuál es utilizado para detectar *microclusters* que modelan ruido a partir de los pesos w ($w < beta * MinPts$). Rango de valores $[0, 1]$.

initPoints

número de muestras a utilizar para la inicialización del algoritmo, utilizando DBSCAN.

offline

multiplicador para $epsilon$. Es utilizado durante el proceso de *reachability reclustering*: DenStream aplica *reachability* (como *dbscan*) entre *microclusters* para reagrupar utilizando *epsilon * offline*.

lambda

constante de decaimiento. Se utiliza para determinar 'la porción' a olvidar de cierto atributo, y también al determinar si un *microcluster* debe eliminarse o no. Rango $[0, 25, 1]$.

speed

cantidad de elementos entrantes por unidad de tiempo (importante para el decaimiento). Rol: establecer el valor del *timestamp* (marca de tiempo).

En lo que respecta al algoritmo DenStream, $epsilon$ y $MinPts$ son los hiperparámetros utilizados para aplicar el tradicional algoritmo de *clustering* en lotes DBSCAN, mientras que $lambda$ y $speed$ son puntualmente aquellos relacionados a lo que es el componente de olvido.

Si se elige un valor muy alto para $lambda$, será muy fuerte el descarte de información pasada. Además, en lo que respecta a la determinación del momento en que debe aplicarse la función de decaimiento, el olvido en sí, que tendrá un mayor o menor impacto de acuerdo al valor de $lambda$, influye el valor que toma el hiperparámetro $speed$. En caso de que este valor sea bajo, se aplicará el componente de olvido con mayor frecuencia; por otra parte, si contrariamente éste valor es demasiado alto, se produce un efecto negativo ya que no aumentaría nunca el *timestamp*, y toda la información recibida hasta el momento se consideraría como 'reciente' o 'relevante'.

Así mismo, igual que en el algoritmo anterior, el hiperparámetro $initPoints$ indica la cantidad inicial de elementos a utilizar. Así, se forman los primeros *microclusters*, en este caso, aplicando el algoritmo DBSCAN). Éstos, empezarán a resumir la información entrante. Cuanto mayor sea el valor que tome, más se demorará en esta inicialización, aunque se tendrá mayor información de la distribución de puntos para la formación de los *microclusters* base.

Luego, el hiperparámetro $beta$ será clave para la detección de *outliers*, noción que carece el algoritmo anterior, de la misma manera que el algoritmo tradicional en el cual se basa. Representa a una proporción a tomar sobre $MinPts$. Como consideración, si se le da un valor muy grande, entonces prácticamente la mayor cantidad de *microclusters* serán considerados como ruido, valores atípicos. Si por el contrario se establece un valor muy chico para este, entonces es probable que los valores atípicos pasen desapercibidos.

Por último, el hiperparámetro *offline*, es utilizado para el agrupamiento final, junto con $epsilon$. Determina cuál será el vecindario a tomar para cada *microcluster*, y permite así hacer crecer los grupos finales. Si éste toma un valor alto, entonces cada grupo final agrupará más *microclusters*. Por otra parte, si se elije un valor bajo para dicho hiperparámetro, entonces se obtendrán más grupos finales, con pocos *microclusters* cada uno, ya que resultará más exigente la determinación de cada vecindario.

4.1.5. DyClee

Finalmente, para el algoritmo DyClee, se optó por personalizar los hyperparámetros requeridos, originalmente definidos en el documento presentado por los autores del mismo. Esta decisión fue tomada con el fin de acotar el desarrollo de la implementación [4] al alcance de la tesina. Por este motivo, por ejemplo, se quitó la posibilidad de generar *clusters* de diferentes densidades, utilizando un enfoque por defecto de *densidad global*, descartando así la posibilidad de agrupar de forma consistente datos con grupos de diferentes densidades, considerando a las regiones menos densas directamente como ruido.

A su vez, se decidió modificar el conjunto de hiperparámetros inicial, para poder realizar ciertas mejoras al algoritmo, producto de un detallado análisis realizado por sobre las implementaciones de algoritmos que siguen el mismo enfoque, explicados previamente en el documento: CluStream y DenStream. Es por esto que, por ejemplo, no se tomó $tGlobal$ como *hiperparámetro*, sino que finalmente se decidió enviar un mensaje al objeto encargado de realizar el agrupamiento, cuando se requiera generar el *clustering* final. Así, quedó por fuera de la funcionalidad del algoritmo, el hecho de determinar en qué momento se debe devolver el agrupamiento con la información recibida hasta el momento. Más aún, se incorporaron nuevos hiperparámetros, cuya necesidad quedó en evidencia luego de examinar los algoritmos anteriores.

A continuación se detallan los hiperparámetros elegidos de aquellos listados en el documento original, y luego los hiperparámetros propuestos, los cuales fueron ideados al momento del desarrollo de la implementación del algoritmo utilizada en esta tesina.

relativeSize
: determina el tamaño de los hiper-rectángulos que dan forma a los *microclusters*. Define para cada atributo, la porción a tomar dentro del rango total. Acepta valores en el intervalo $[0, 1]$.

uncommonDimensions
: Valor entero. Permite al usuario seleccionar la cantidad de atributos que serán tenidos en cuenta para determinar si se parecen o no entre sí los *microclusters*.

lambda
: determina cuánta información se olvidará, al aplicar el componente de olvido.

dataContext
: El usuario puede optar por proporcionar una *matriz* para brindar información de contexto. Ésta se utiliza solamente con fines de normalización de los datos.

[4] `https://github.com/onofricamila/DyClee/blob/master/utils/dyclee.py`

processingSpeed
: define en qué momento se debe aumentar la marca de tiempo o bien *timestamp* actual; relacionado con el proceso de olvido.

timeWindow
: define si la función de decaimiento debe aplicarse a un determinado *microcluster*, comprobando tanto la marca de tiempo actual, como el valor de tl. El tl indica en qué momento se realizó la última incorporación de un dato de entrada al *micro cluster* en cuestión.

periodicUpdateAt
: define el momento en el que se verifica el valor de tl de los *miscroclusters*; representa una proporción del hiperparámetro $processingSpeed$

periodicRemovalAt
: define el momento en que los *microclusters* que modelan valores atípicos, son llevados a la lista de *outliers*; representa una proporción del hiperparámetro $processingSpeed$

closenessThreshold
: utilizado al construir los *clusters* finales, al momento de unir diferentes *microclusters*

El hiperparámetro *relative size* determina que tan exigente se es a la hora de examinar si un elemento de entrada es lo suficientemente similar a un *microcluster* existente, como para ser incorporado y representado por éste. También, determina qué *microclusters* se encuentran *directamente conectados* como para formar parte del mismo *cluster* final. Cuánto más chico sea su valor, más restrictiva serán dichas comparaciones. Cabe mencionar que, si se setea un valor demasiado bajo, consecuentemente se crearán más *clusters* finales debido a cambios de densidad dentro de cada grupo natural. En otras palabras, se terminarán formando distintos subgrupos, debido a regiones de baja densidad que se encuentran entre regiones de mayor densidad, como se puede ver en la imágen de la izquierda de la Figura 4.5.

Por otro lado, si el valor que toma dicho hyperparámetro es demasiado grande, grupos grandes pueden terminar fusionándose incorrectamente como se puede ver en la Figura 4.6, debido a la proximidad entre *microclusters* de alta densidad.

Uno de los cambios que se realizó a la hora de desarrollar la implementación, que involucra a dicho hiperparámetro, es el siguiente: al momento de evaluar si dos *microclusters* están *directamente conectados*, siendo la definición de esta propiedad detallada en el Capítulo 3, e ilustrada en la Figura 3.5, se optó por tomar simplemente el valor de *relative size* y no la mitad de éste, de manera tal que la evaluación sea menos restrictiva, y se permita conectar a más *microclusters*. Habiendo actualizado la restricción, se logró que se conecten *microclusters* que, para la dimensión donde la diferencia es mayor, estén superpuestos, o bien uno junto al otro. La Figura 4.7 ilustra lo comentado.

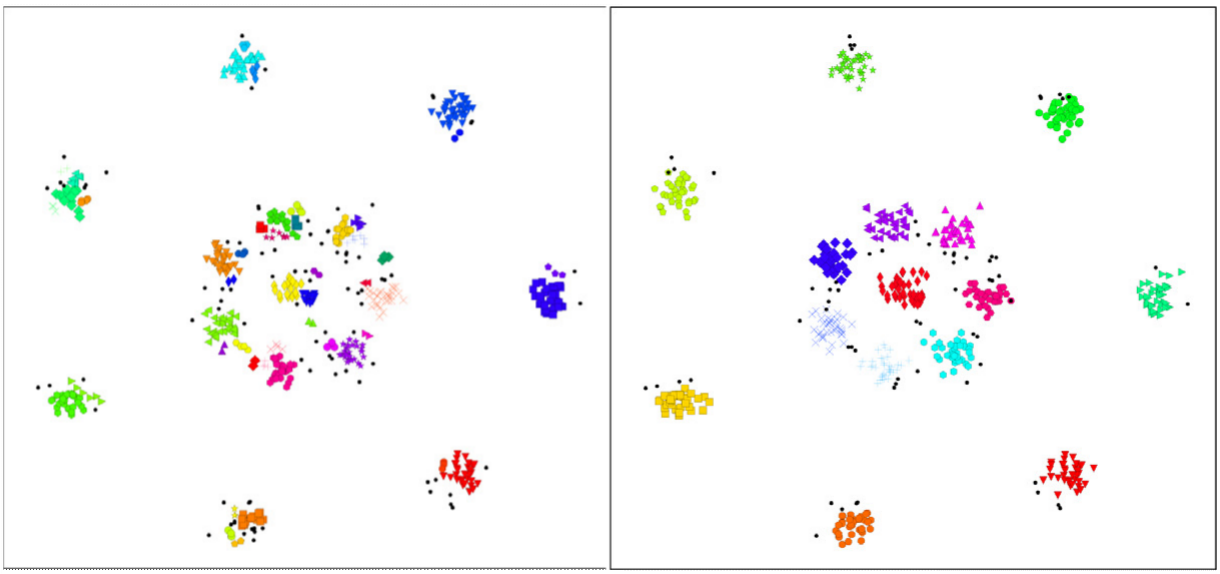

FIGURA 4.5: Resultados de agrupamiento de DyClee para *microclusters* con un *relative size* de 0,01 (izquierda) y 0,03 (derecha) [47].

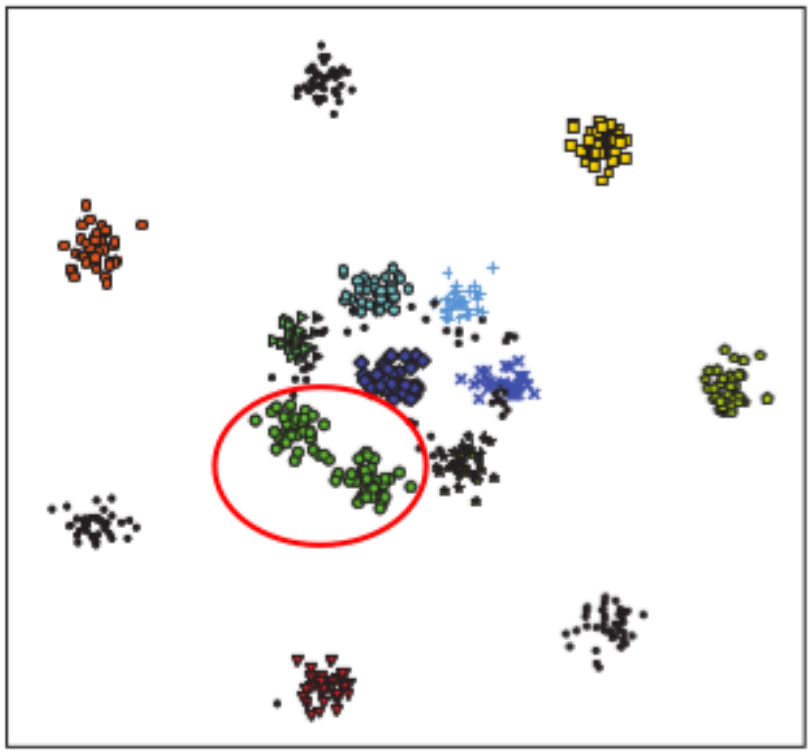

FIGURA 4.6: Resultados de agrupamiento de DyClee para *microclusters* con un *relative size* de 0,06 [47].

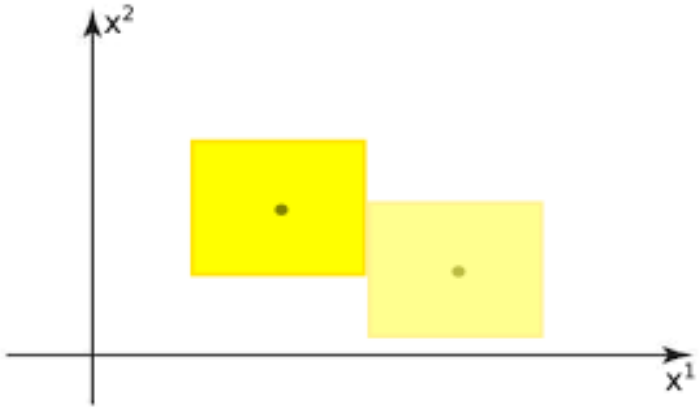

FIGURA 4.7: Redefinición de la propiedad de *microclusters directmente conectados*. La mayor diferencia está en el eje x, y cumple con la condición de ser menor o igual a S.

Por otra parte, en lo que respecta al hiperparámetro *uncommonDimensions*, este toma un papel importante a su vez a la hora de ver si dos *microclusters* están directamente conectados. Cuanto más atributos se tengan en cuenta, más restrictiva será dicha evaluación. Si por ejemplo este hiperparámetro toma el valor 0, la propiedad de *Directamente conectados* se cumple sólo cuando los *microclusters* se superponen en todas las dimensiones.

Otra de las modificaciones realizadas, tiene que ver con la formación de grupos finales. Debido a que tras la realización de diversos experimentos, se notó al utilizar el algoritmo con datos reales y de prueba, que era muy restrictivo tener en cuenta sólo la propiedad de *clusters conectados* para unir *microclusters*, más allá de la modificación realizada a la propiedad de *directamente conectados*, se decidió incorporar el hiperparámetro *closenessThreshold*. El mismo, da un indicio de que tan cerca deben estar dos *microclusters* como para que, por más de que no cumplan la propiedad anterior, pasen a formar parte de un mismo grupo final. Este hiperparámetro, representa la proporción a tomar de la desviación estándar calculada por sobre las distancias entre todos los *microclusters*. Cuanto más grande sea el valor que tome el hiperparámetro, más laxa será la condición, por lo que una mayor cantidad de *microclusters* pasarán a formar parte de un mismo grupo final. El rol de *closenessThreshold* queda detallado de la siguiente manera:

Los microclusters μC_j y μC_k, formarán parte del mismo grupo final, si están conectados o bien

$$d(\mu C_j, \mu C_k) < avg_d - stddev_d * closenessThreshold \tag{4.1}$$

donde $d(\mu C_j, \mu C_k)$ representa a la distancia *Manhattan* entre ambos *microclusters*, y avg_d la distancia promedio entre los *microclusters* en cuestión y los demás existentes.

Además, en lo que respecta al componente de olvido, diversos hiperparámetros influyen de diferente manera. Del lado de aquellos que marcan un tiempo, o bien interactuan con el *timestamp*, como ser *periodicUpdateAt, periodicRemovalAt, processingSpeed, timeWindow*, cuanto más chico sea el valor que tomen, más rápida o recurrente será la aplicación de la función de decaimiento, olvidando con mayor fuerza la información pasada. Cabe mencionar que, *periodicRemovalAt* es utilizado para eliminar los *outlier microclusters* que no cumplen con la densidad promedio, brindando una nueva funcionalidad.

Luego, de la misma manera que se realiza en algoritmos que siguen el mismo enfoque que DyClee, el hiperparámetro *lambda* es el que indica qué porcentaje de la información pasada será descartada, considerando que a medida que toma valores más grandes, mayor es la cantidad de información que se considera irrelevante y por lo tanto se procede a olvidar.

Como último comentario, el hiperparámetro *dataContext* es totalmente opcional, y puede utilizarse en caso de tener el nivel de conocimiento del dominio o *ground truth* necesario como para proveer para cada atributo valores de normalización.

4.2. Análisis de los experimentos realizados

En esta sección, se analizan los resultados obtenidos al aplicar los distintos algoritmos de agrupamiento detallados previamente, utilizándolos en diferentes escenarios. Los *clusters* generados por cada uno de éstos son representados mediante colores: las muestras pertenecientes al mismo grupo son ilustradas con un mismo color. Los valores atípicos detectados se representan con color negro.

Los casos de prueba dejan en evidencia el comportamiento de cada algoritmo, permitiendo visualizar de forma práctica lo explicado teóricamente en las secciones anteriores.

Los experimentos también son utilizados para ilustrar cómo afectan las distintas configuraciones de hiperparámetros a los resultados finales, habiendo descrito cada uno de ellos previamente en el documento.

Por último, considerando que la inspección visual no es suficiente para evaluar la calidad del agrupamiento y comprender los resultados generados, se indica en cada caso el valor de la métrica DBCV obtenido. Se recuerda al lector que en el Capítulo 3 se detalla el por qué de la elección de dicha métrica.

Se presentan a continuación los resultados logrados con los distintos algoritmos, a la hora de resolver un conjunto de problemas puntuales. Se describe en cada caso el escenario donde se aplica el agrupamiento, los algoritmos utilizados, y el por qué de los resultados obtenidos por cada uno.

4.2.1. Detección de distribuciones convexas y no convexas

Este apartado tiene como objetivo mostrar cómo se comportan los algoritmos K-Means, DBSCAN y DyClee al separar *clusters* 'vecinos' de forma no convexa, dejando en evidencia, a su vez, de qué forma accionan cuando se está en presencia de ruido o bien valores atípicos. Para exponer este comportamiento, se optó por utilizar algunos conjuntos de datos de prueba, conocidos ampliamente en el campo de la Ciencia de Datos, los cuáles están disponibles en el módulo de Python *scikitlearn* [61].

Como al día de la fecha los autores del algoritmo DyClee no han hecho pública una implementación del mismo, se ha desarrollado una implementación propia para poder llevar a cabo las distintas pruebas. Dicho algoritmo se comparará con las implementaciones de K-Means y DBSCAN proporcionadas por el módulo *scikitlearn*.

Para este experimento, se realizó la evaluación de 5 conjuntos de datos: $blobs$, $noisy\_moons$, $noisy\_circles$, $varied$ y $aniso$. Cada uno de estos conjuntos se compone de 1500 muestras generadas mediante una distribución a la que se le incorporó un porcentaje de $0{,}05$ de ruido [5]. Los grupos subyacentes en los datos no varían ni evolucionan en el tiempo. Debido a esto último, se deshabilitó de DyClee el componente de olvido al momento de efectuar la prueba.

Luego de realizar 24 ejecuciones, se determinó cuáles fueron los valores de los hyperparámetros para cada algoritmo con los que se llegó a los mejores resultados. Estos fueron configurados de la siguiente manera:

- K-Means (número real de clusters para cada caso),
- DBSCAN ($eps = 0{,}2$, $min\_pts = 10$),
- DyClee ($relative\_size = 0,06$).

Los *clusters* generados, utilizando dicha configuración de hiperparámetros, se muestran en la Figura 4.8. Se recuerda al lector en este punto, que un valor cercano a 1 representa un buen valor para la métrica DBCV. Los valores que otorga esta métrica se encuentran en el intervalo $[-1, 1]$, considerando el índice debe maximizarse. Para más detalle acerca de la métrica, dirigirse a la Sección 3.4.3

Como se ha mencionado anteriormente, K-Means requiere el número de *clusters* a formar como parámetro inicial, lo cual es bastante restrictivo, dado que es necesario tener cierto conocimiento del dominio que permita de antemano setear cuántos grupos se van a querer formar. Volviendo a la Figura 4.8 [6] [7], se puede notar que, cuando los datos se organizan

[5] https://github.com/onofricamila/ToyDatasetsPersistorIntoCsvs
[6] https://github.com/onofricamila/ClusteringWTraditionalAlgorithms
[7] https://github.com/onofricamila/ClusteringPlotterAndDBCVCalc/tree/master/utils/clustering_managers/non_time_series_clustering_mngr.py

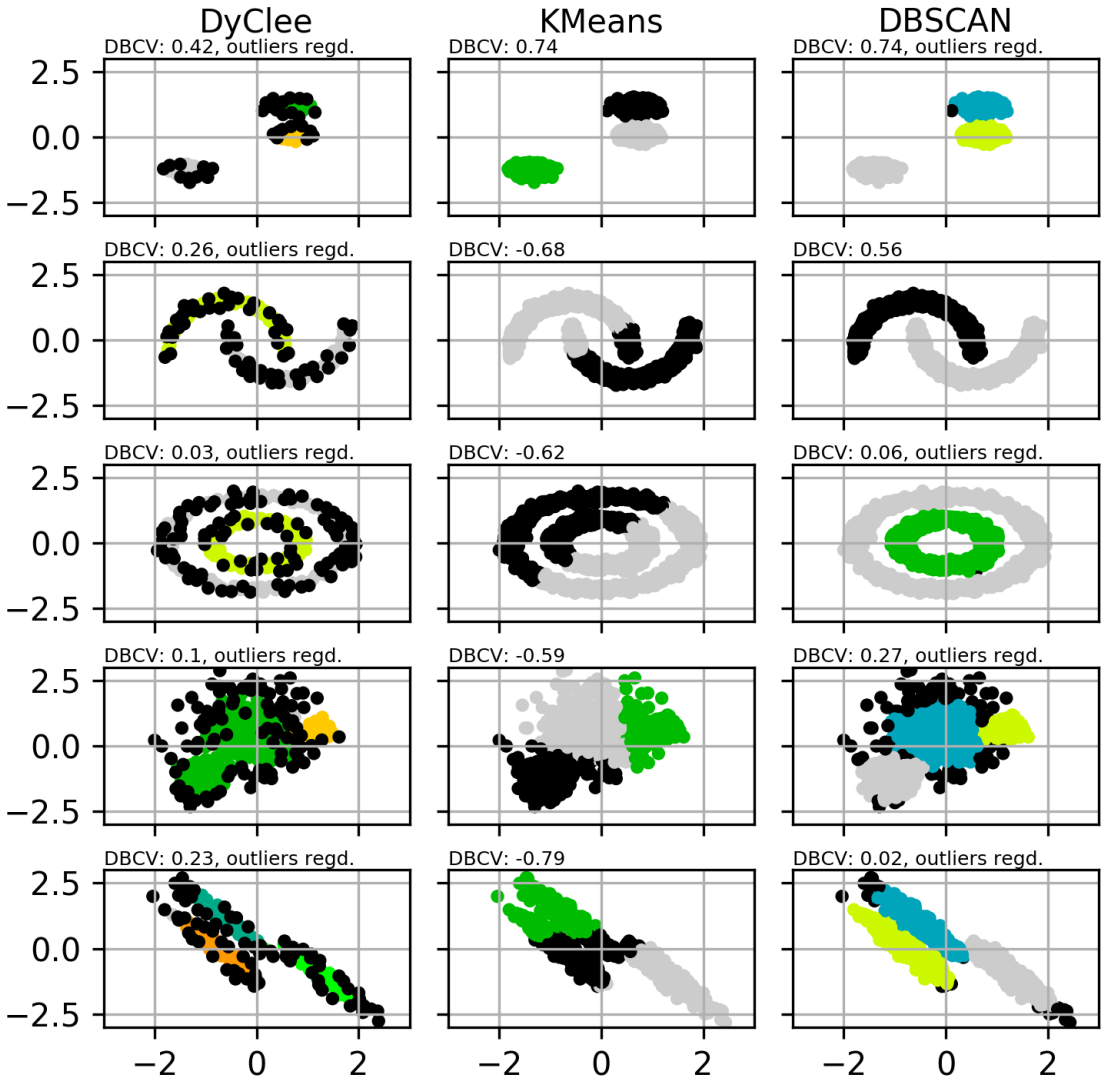

Figura 4.8: Comportamiento de DyClee, DBSCAN y Kmeans en distribuciones convexas y no convexas.

en conjuntos no convexos, el algoritmo K-Means tiende a fallar al querer agruparlos; esto ocurrirá incluso si se le da como entrada el número real de grupos presentes en los datos. Además, K-Means agrega los valores atípicos al grupo cuyo centroide es más cercano, ya que carece de la capacidad para identificarlos y segregarlos.

Por otra parte, cabe mencionar que DBSCAN y DyClee son capaces de detectar estas distribuciones no convexas sin problema alguno. En este escenario, entonces, ambos algoritmos funcionan correctamente.

En cuanto a la métrica en sí, se puede observar que DBSCAN obtiene los mejores resultados para todos los conjuntos de datos excepto el último, más allá de que en éste se captan correctamente los grupos subyacentes en los datos. Con la configuración actual de hyperparámetros, DyClee encuentra una mayor cantidad de outliers, valores atípicos. Ésto, permite que obtenga un valor mayor para DBCV para el último conjunto de datos, pero hace que el valor de la métrica sea más bajo en el resto. De todas formas todos los valores que obtiene para DBCV son positivos, o cercanos a 0. Por contraparte, K-Means obtiene un buen valor para la métrica únicamente cuando los grupos son convexos, como puede observarse en el primer caso. Para aquellos con formas arbitrarias, se obtienen valores cercanos a -1, lo cual indica que el algoritmo no logra captar correctamente los grupos.

4.2.2. Deriva de concepto: utilidad del componente de olvido

En esta sección, DyClee, DenStream y CluStream se enfrentan a distribuciones variables en el tiempo; son utilizados para identificar *clusters* que evolucionan. Se generó para este caso un escenario con *deriva de concepto*, en el cual se ve clara la necesidad de *aprendizaje en línea*.

En la sección anterior, el comportamiento de DyClee, DBSCAN y K-Means se demostró tomando como entrada distribuciones estáticas, con diferentes características. Sin embargo, en gran parte de las aplicaciones del mundo real, no se suele estar en presencia de datos estacionarios, invariables en el tiempo; por el contrario, usualmente se espera que los datos evolucionen, es decir, que la distribución subyacente cambie en cierto aspecto. Esto mismo, se conoce como la *deriva de concepto*, término que se utiliza para referenciar algún cambio sutil de la distribución de datos.

Para esta prueba, se han utilizado tres distribuciones claramente diferenciadas, para formar así tres grupos. Dos de estas distribuciones, fueron manipuladas para que se desvíen en el tiempo: se generaron muestras sintéticas cambiando el centro de ambas distribuciones cada 100 elementos. En consecuencia, las posiciones de los grupos correspondientes fueron rotando significativamente, como se puede ver en la Figura 4.9, en la que cada imagen corresponde al estado del conjunto de datos en un determinado momento.

Dado que las implementaciones de DBSCAN y K-Means no consideran la posible evolución de los datos en el tiempo, y toman todas las muestras de datos disponibles como datos de

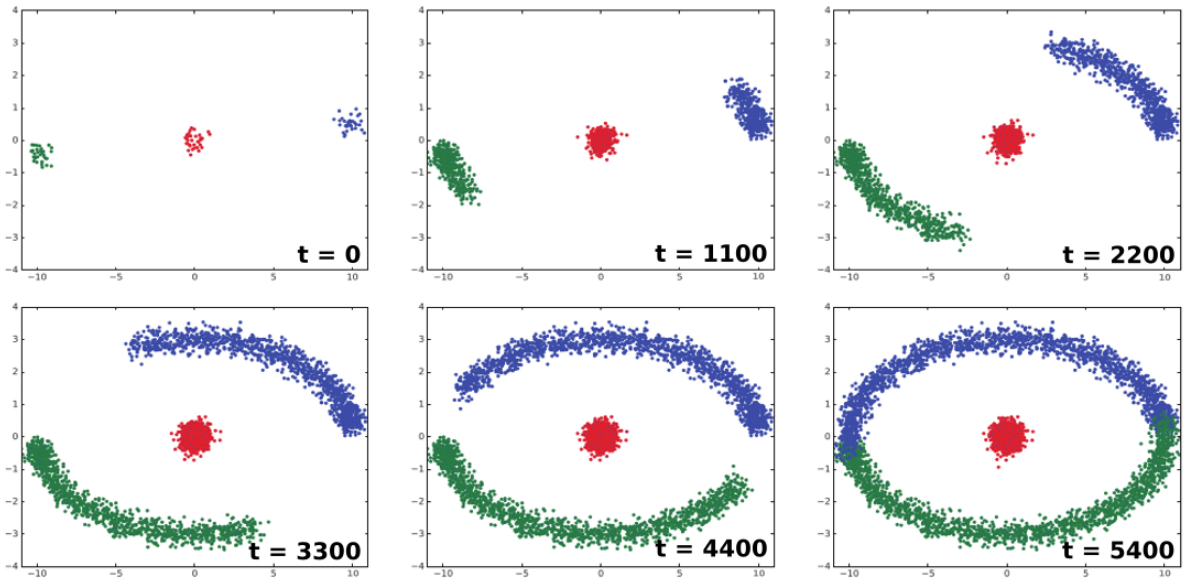

FIGURA 4.9: Conjunto de datos generado para ilustrar la deriva de concepto. Capturas cronológicas en las que se indica la cantidad de elementos generados hasta el momento en cada cuadro.

entrada, para este caso puntual, si se quisiera utilizar dichos algoritmos, no se notaría evolución alguna, se use o no su versión incremental (es decir, aquella versión que realiza un agrupamiento acumulado a medida que ingresan más datos). Esto se debe, a que dichos algoritmos, no presentan ningún componente que permita olvidar en cierto punto la información pasada, o bien tener una noción de relevancia que pondere a los datos más recientes.

Si se vuelve a visualizar el conjunto de datos generado para este experimento, se puede notar que, si se utilizara DBSCAN a la hora de hacer el agrupamiento, por la forma en que funciona el algoritmo, los dos grupos que notablemente evolucionan, en el conjunto de datos propuesto, terminarían fusionándose en un solo *cluster*, siguiendo el enfoque por densidad.

Otra alternativa para poder utilizar estos algoritmos tradicionales, sería proceder a descartar toda la información acumulada cada cierto período de tiempo (o bien cada cierta cantidad de elementos procesados), y empezar así a agrupar desde cero, descartando esa información pasada. Sin embargo, si se sigue este enfoque, no es posible que el modelo se adapte. Esto se debe, a que se agrupa de forma independiente en cada momento, ignorando los estados previos. Además, de esta manera, existe la posibilidad de descartar información relevante, la cuál debería ser mantenida. Esto último, se debe a que con este método, se elimina toda la información pasada, sin considerar que *clusters* con una marca de tiempo de creación 'vieja' pueden mantenerse activos, incorporando elementos nuevos, manteniendo su relevancia.

Debido a todos los puntos mencionados anteriormente, se decidió para el caso de prueba propuesto utilzar DenStream, CluStream y DyClee, algoritmos ideados para resolver este tipo de problemas donde los datos de entrada son no estacionarios.

En lo que respecta a la prueba en sí, la Figura 4.10 muestra los grupos obtenidos en cada momento por DenStream [8] [9]. En cada cuadro se visualiza la marca de tiempo t, el valor para la métrica elegida, y la cantidad de *microclusters* μc generados hasta el momento.

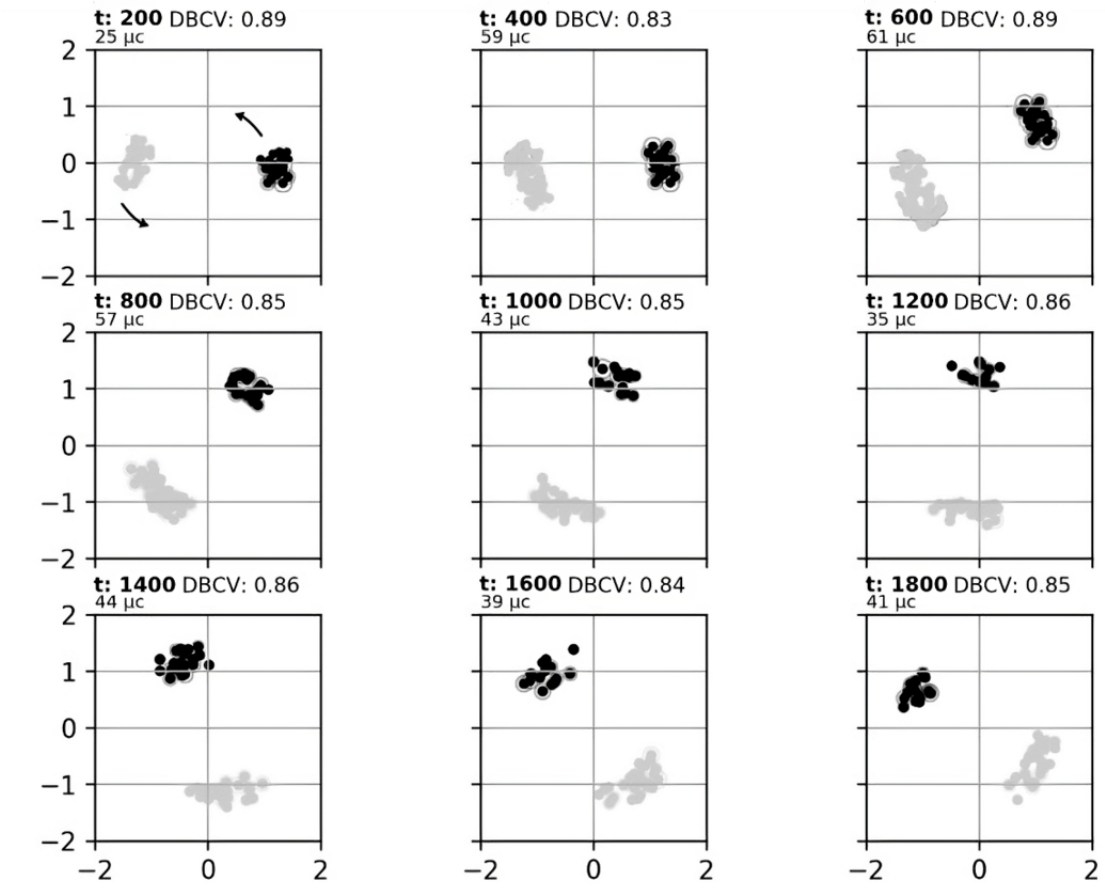

FIGURA 4.10: Resultado obtenido con DenStream para el conjunto de datos de prueba de Deriva de concepto. Debe leerse de izquierda a derecha, de arriba hacia abajo, siguiendo la marca de tiempo t.

Como se puede ver, el algoritmo no detecta valores atípicos, sino simplemente dos grupos que evolucionan. A su vez, se puede notar que, los valores obtenidos para la métrica DBCV, son muy buenos. El algoritmo se comporta correctamente.

Por otra parte, la Figura 4.11, muestra los resultados generados por el algoritmo CluStream. Como antes, se muestran tanto los valores de t como μc y DBCV.

Igual que en el caso anterior, CluStream logra detectar efectivamente dos grupos, y los valores que toma DBCV son muy altos.

[8] `https://github.com/onofricamila/MOAOnlineClustering/blob/master/src/utils/clusterers/DenstreamClusterer.java`

[9] `https://github.com/onofricamila/ClusteringPlotterAndDBCVCalc/blob/master/utils/clustering_managers/time_series_mngrs/denstream_clustering_manager.py`

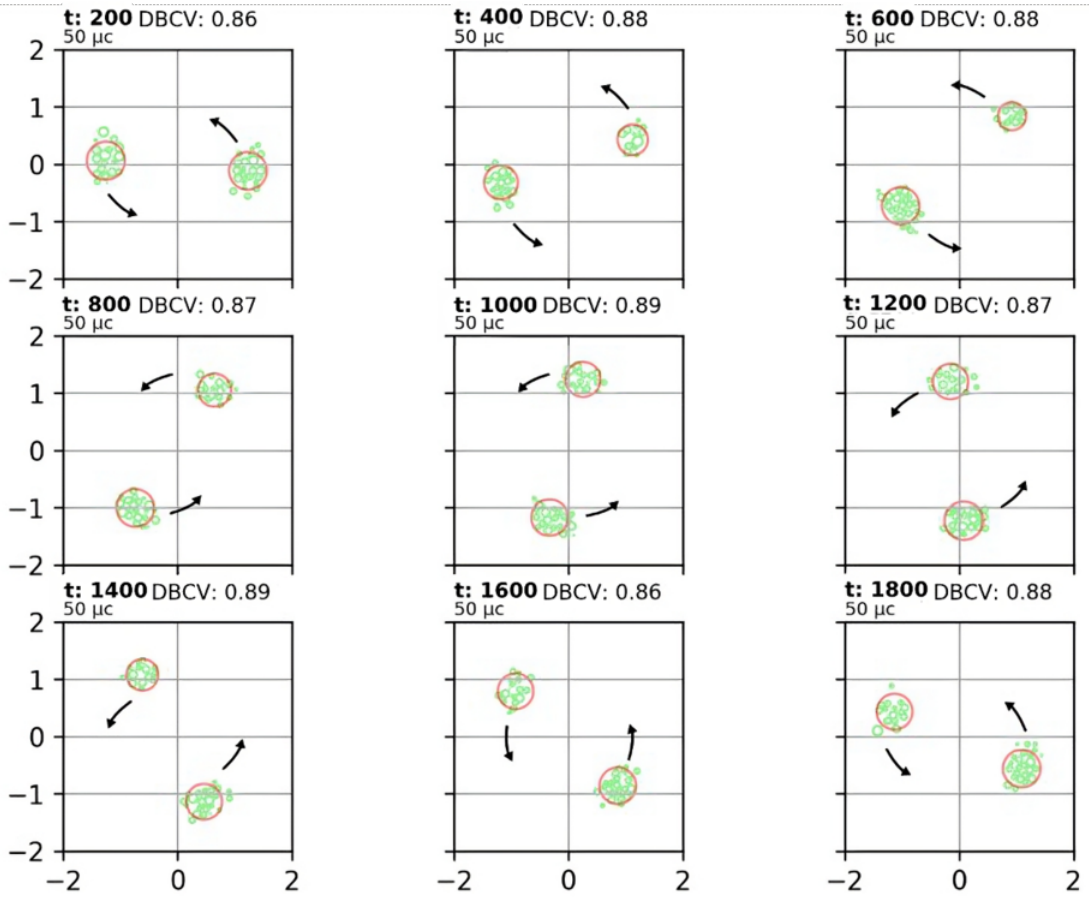

FIGURA 4.11: Resultado obtenido con CluStream para el conjunto de datos de prueba de Deriva de concepto. Debe leerse de izquierda a derecha, de arriba hacia abajo, siguiendo la marca de tiempo t.

Por último, la Figura 4.12 [10], muestra aquellos resultados obtenidos cuando se utilizó DyClee, visualizando en cada momento la marca de tiempo, el valor para la métrica elegida, y la cantidad de *microclusters* generados.

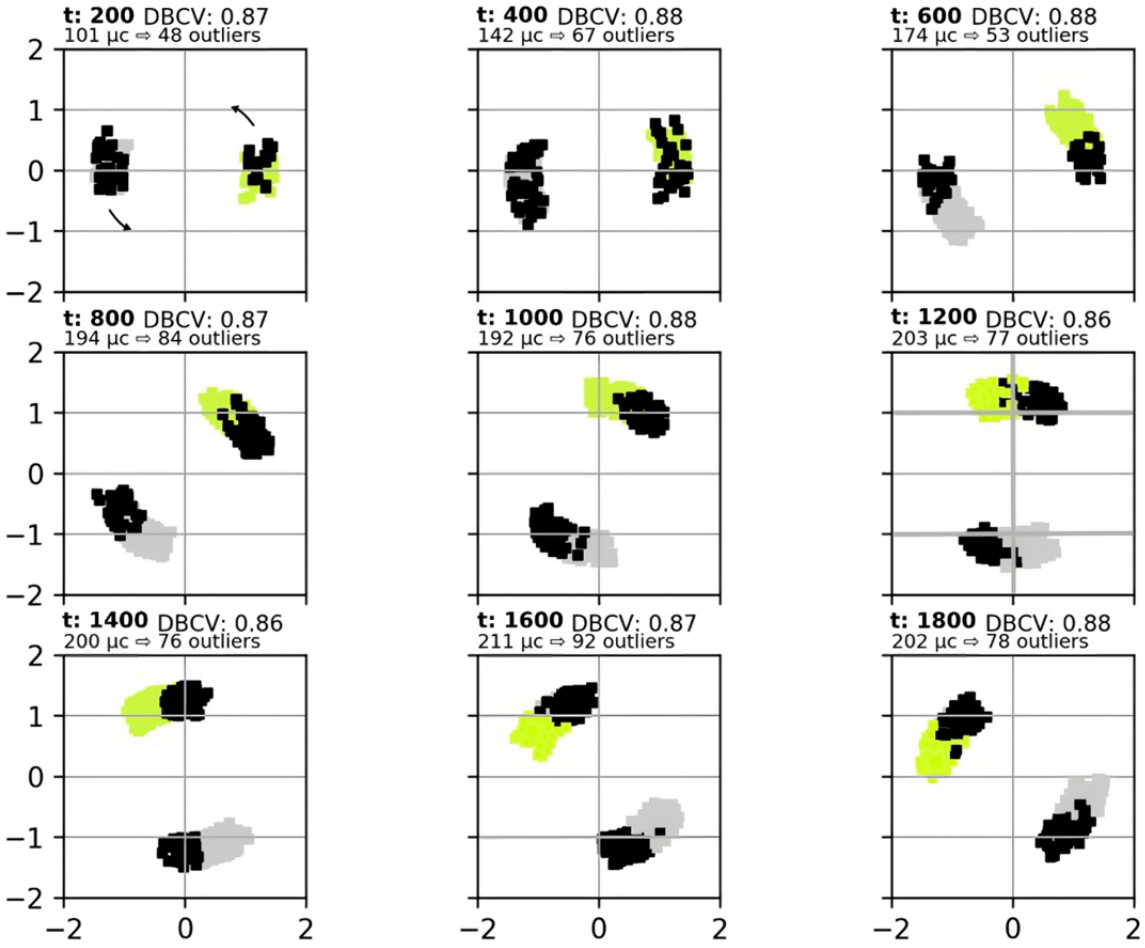

FIGURA 4.12: Resultado obtenido con DyClee para el conjunto de datos de prueba de Deriva de concepto. Debe leerse de izquierda a derecha, de arriba hacia abajo, siguiendo la marca de tiempo t.

En este caso, se puede observar cómo es que más allá de que se detectaron perfectamente dos grupos, también se encontraron valores atípicos. Esta capacidad, marca un diferencial respecto a los dos algoritmos anteriores: permite entender en qué sentido se da la evolución, según la ubicación de los *microclusters outliers* que se van formando. Más aún, para este problema con datos en dos dimensiones, visualmente, en un solo cuadro se puede apreciar cómo es que se dio la evolución de cada grupo, sin mirar el cuadro anterior; se da un indicio de en qué sentido evolucionaron los *clusters*. Esta característica, hace que el algoritmo se destaque por sobre el resto. Además, cabe mencionar que, los resultados que se obtuvieron para la métrica DBCV, fueron muy buenos.

[10] `https://github.com/onofricamila/ClusteringPlotterAndDBCVCalc/blob/master/utils/clustering_managers/time_series_mngrs/dyclee_clustering_manager.py`

Concluimos entonces, que DyClee es el algoritmo que mejor se desempeñó en este caso.

4.2.3. Resolviendo un problema real con DyClee

Luego de haber realizado las diferentes comparativas, y dejando así en evidencia el buen *desempeño* de DyClee, en este apartado se muestran los resultados obtenidos al emplear el algoritmo para realizar un análisis experimental sobre un conjunto de datos real.

Los datos utilizados, fueron obtenidos de la ampliamente reconocida plataforma de Ciencia de Datos llamada *Kaggle* [11]. Los mismos, corresponden a datos de ubicación de taxis en Suecia, durante octubre y noviembre de 2018 [12]. Representan coordenadas GPS, muestreadas aproximadamente una vez por minuto.

Resulta interesante mencionar, que el conjunto de datos fue elegido por la peculiaridad que simula perfectamente un escenario de *clustering dinámico*, donde los vehículos emiten constantemente un reporte de su ubicación geográfica en forma de flujo, y el algoritmo de agrupamiento recibe y resume la información en línea, realizando luego el agrupamiento con los datos recibidos hasta el momento, a medida que esto sea solicitado.

La utilidad de analizar estos datos en particular, está dada por el hecho de que obtener un indicio de cómo se encuentran agrupados los diferentes vehículos de la zona, en cada instante del día, permite entender qué zonas son las más afectadas en cuánto al tráfico, y en qué momento. Esto, posibilita mejorar la asignación de taxis, perfeccionar el flujo de circulación de los mismos, y también facilita la determinación de qué lugares son claves a la hora de construir infraestructura de carga, necesaria para brindar soporte a los vehículos.

Al momento de realizar la prueba, se configuró DyClee con los siguientes hiperparámetros:

- relativeSize = 0.05,
- speed = 5000,
- lambda = 0.4 ,
- periodicUpdateAt = 3,
- timeWindow = 3,
- periodicRemovalAt = 3,
- closenessThreshold = 0.85,

El agrupamiento generado por DyClee, se muestra en la Figura 4.13 [13].

[11] `https://www.kaggle.com`
[12] `https://www.kaggle.com/henrikengdahl/taximovementconcatenated`
[13] `https://github.com/onofricamila/DyClee/blob/master/main3.py`

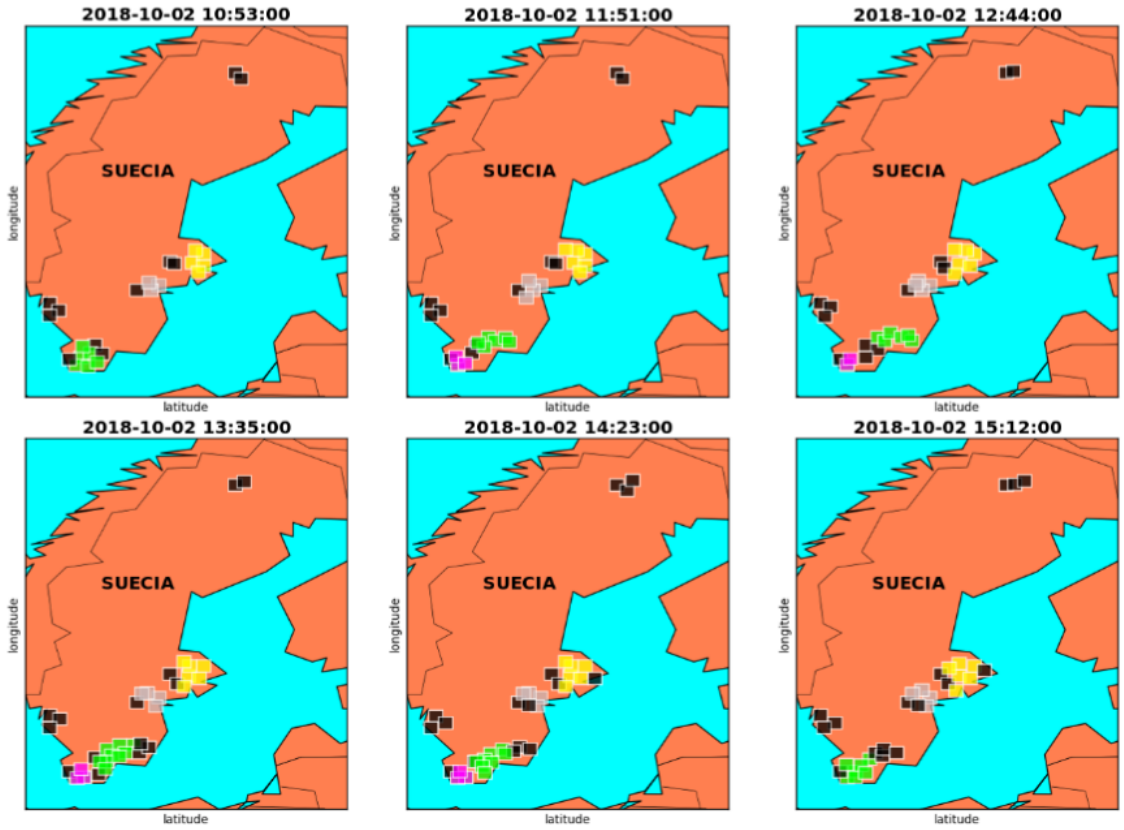

FIGURA 4.13: Resultado obtenido con DyClee para el conjunto de datos real de coordenadas de taxis en Suecia. Debe leerse de arriba hacia abajo, de izquierda a derecha.

En ella se puede observar que el algoritmo identifica cuatro grupos principales y a su vez diversos *microclusters outliers* en el mapa que representa a todo el país.

El *agrupamiento* es consistente con el comportamiento de los vehículos en cada momento. Los *microclusters* densos representan el aglomerado de automóviles en la zona. Los movimientos de taxis de una zona a otra son identificados gracias a los *microclusters outliers*.

Si se observa la Figura 4.14, los *microclusters outliers* que rodean al grupo que se encuentra ubicado en el extremo inferior del mapa, denotan la evolución en la circulación de los vehículos.

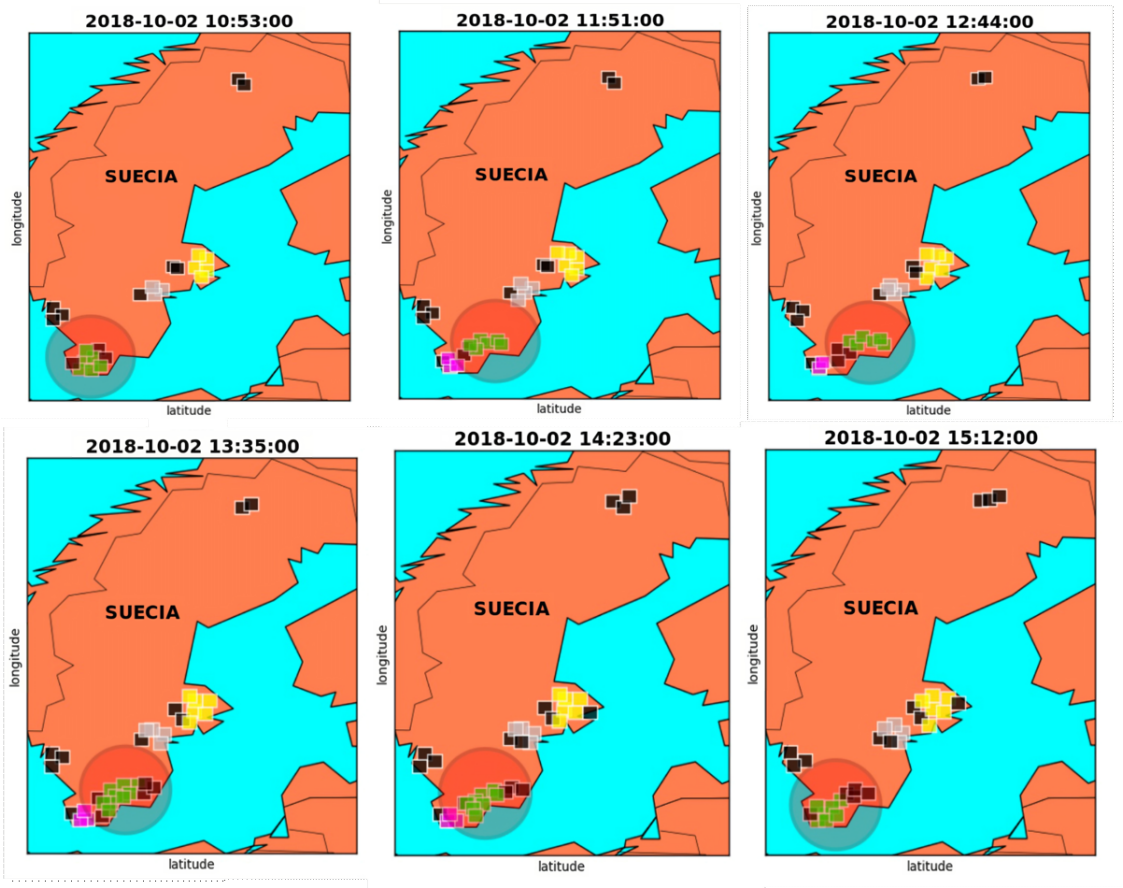

FIGURA 4.14: Foco en la evolución para el resultado obtenido con DyClee sobre el conjunto de datos real de coordenadas de taxis en Suecia.

Diversos automóviles ubicados en la zona densa, se mueven hacia otro punto en el mapa en sentido este, y luego viceversa. Cabe señalar que, la cantidad de puntos que contiene cada uno de estos *microclusters ouliers*, no es suficiente para que sean considerados densos, y pasen por consiguiente a formar parte de un grupo. Es por esto que se mantienen como valores atípicos. Sin embargo, estos *microclusters*, cumplen la función de representar el cambio en el tráfico, como se ha mencionado previamente. Además, gracias a la función de olvido, el estado total se mantiene constantemente actualizado, pudiendo de esta manera ilustrar hacia donde es

la evolución en cada momento, considerando de qué estado se venía en el agrupamiento anterior.

Luego, si se observan las figuras 4.15 4.16, éstas ilustran qué tipo de información podría verse, con un simple gráfico de los elementos de entrada, ya sea mostrando un acumulado de todos los objetos recibidos hasta el momento, o graficando de a una las ventanas de tiempo, descartando la información pasada.

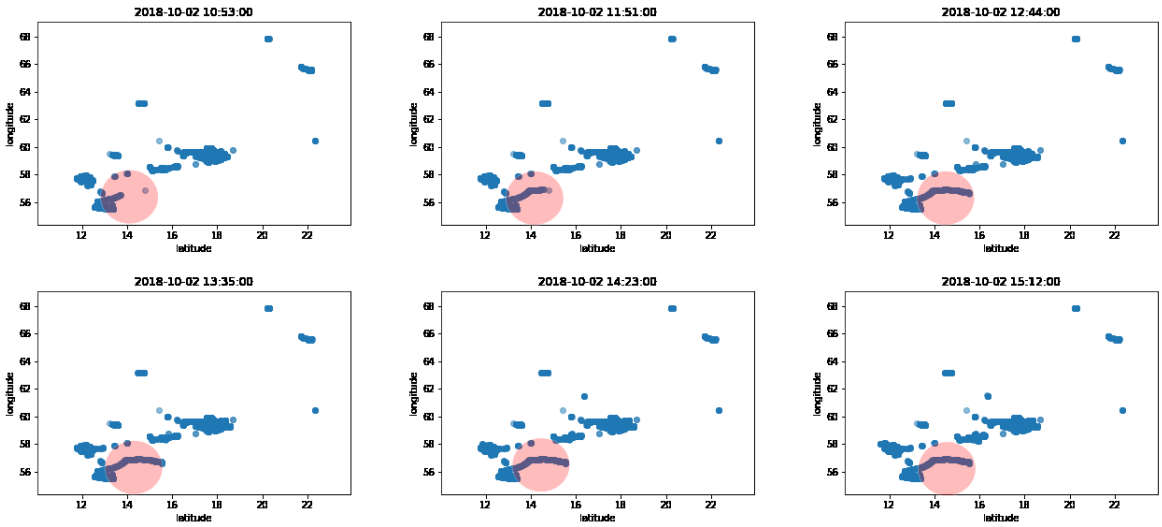

FIGURA 4.15: Distribución de datos acumulados para el conjunto de datos de taxis en Suecia.

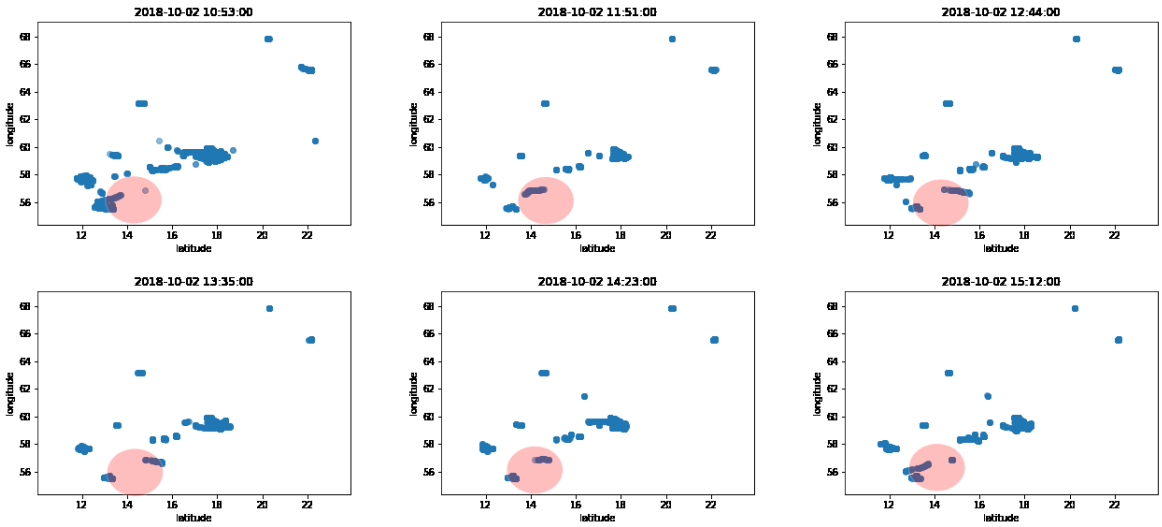

FIGURA 4.16: Distribución de datos en distintas ventanas de tiempo para el conjunto de datos de taxis en Suecia.

La ventaja de utilizar un algoritmo de agrupamiento de la naturaleza de DyClee por sobre estos tipos de gráficos, está dada por el hecho de que al resumir la información a medida que va llegando, en línea, y luego aplicar un componente de olvido, se tiene en todo momento

la información actualizada, representando el estado actual, y teniendo sólo los *microclusteres* relevantes.

En un problema como el actual en dos dimensiones, si se opta por utilizar un gráfico incremental para inspeccionar los datos de entrada, se pierde la noción de cuál es la información importante y cuál aquella que dejó de serlo. En el ejemplo propuesto, de agrupamiento de vehículos, si se vuelve a visualizar la Figura 4.15, se puede notar que no es posible percibir que hay una tendencia de que los vehículos se dirijan hacia el este y luego regresen, sino que se ve simplemente el acumulado de elementos recibidos hasta el momento. Además, este método implica grandes requerimientos en términos de memoria y potencia de cómputo, por lo que no es adecuado para manejar grandes cantidades de datos, como aquellos que llegan en forma de flujo.

Por otra parte, si se elige graficar por períodos de tiempo, borrando toda la información pasada, se corre el riesgo de estar eliminando información relevante, ya que es muy difícil determinar el tamaño perfecto de la ventada de tiempo; esto puede notarse en la Figura 4.16, donde se percibe levemente la evolución, pero el hecho de descartar toda la información pasada hace que no se pueda comprender en un cuadro dado cuáles son los *microclusters* que vienen manteniendo su relevancia a lo largo del tiempo, y cuáles son más recientes. No es posible identificar cuáles son los grupos de muestras que tienen más peso o relevancia en un momento dado. Además, se pierden otras nociones, como es el poder distinguir la forma en que se da la evolución en los datos.

Finalmente se concluye entonces, por los puntos mencionados previamente, que queda clara la ventaja del uso de un algoritmo de agrupamiento dinámico como DyClee para poder analizar este tipo de problemas, donde la información que va llegando es tan voluminosa que se ve la necesidad de resumirla, y los grupos subyacentes en los datos evolucionan; cambia la distribución de las muestras a medida que pasa el tiempo.

Capítulo 5

Conclusiones y líneas de trabajo futuras

En esta tesina se hizo un análisis exhaustivo sobre distintos métodos de agrupamiento tradicionales, detallando particularmente K-Means y DBSCAN, y también se describieron algoritmos de *clustering* dinámico, como CluStream, DenStream y DyClee, realizando un análisis especial sobre este último. DyClee es un algoritmo recientemente introducido en la literatura y fue implementado con las variantes de hiperparámetros descritas en el **Capítulo 4**, para ser analizado en esta tesina. Esta implementación, puede ser descargada de GitHub [1].

Se explicó por qué un enfoque tradicional no es útil a la hora de agrupar datos provenientes de un flujo, considerando un contexto no estacionario.

A su vez, se hizo foco en las cualidades que permiten a los algoritmos detectar comportamientos atípicos pero también novedades, pensando en capturar y representar la posible evolución en los datos, en un entorno cambiante.

Se analizaron los algoritmos dinámicos basados en dos etapas: una *online*, y otra *offline*. Éstos permiten agrupar datos que llegan a gran velocidad gracias a que resumen los datos de entrada, ejecutando luego las tareas pesadas de agrupamiento final, con diferente frecuencia, en el momento en que se solicita.

El desempeño de DyClee a la hora de generar un agrupamiento se probó en varios casos de prueba, utilizando conjuntos de datos ampliamente conocidos por la comunidad dentro de la Ciencia de Datos.

Se comparó al algoritmo DyClee con otros algoritmos presentes en la literatura, y los resultados obtenidos fueron muy buenos, utilizando como parámetro para medirlos la métrica DBCV.

[1] `https://github.com/onofricamila/DyClee`

Así, quedó demostrado que DyClee logra reunir un conjunto de características deseables, que generalmente no se encuentran juntas en un sólo algoritmo, considerando además la originalidad del enfoque del mismo. Se menciona particularmente esto ya que el algoritmo no se basa en ningún algoritmo de agrupamiento tradicional, como sí lo hacen CluStream y DenStream, los cuáles heredan las ventajas y complicaciones de ellos. La métrica en cuestión fue elegida tras un arduo análisis, teniendo en cuenta que la misma es la única que demuestra comportarse correctamente a la hora de ponderar resultados con *clusters* con formas arbitrarias, en presencia de ruido.

Por otra parte, quedó demostrado que DyClee encuentra más valores atípicos que su comparable DenStream, lo cuál permite seguir mejor el cambio de estado en los *microclusters*, posibilitando comprender cómo viene dada la evolución. Es por esto que DyClee fue elegido para resolver un problema real.

Se optó por utilizar un conjunto de datos que corresponde a coordenadas de ubicación de taxis en Suecia, durante octubre y noviembre de 2018 [2], muestreados aproximadamente una vez por minuto.

Quedó en evidencia la utilidad de este tipo de algoritmos para la realización de agrupamientos de forma incremental, a medida que llega la información, utilizando un componente de olvido, argumentando que si se opta por tomar la información acumulada o en ventanas de tiempo, se pierde la noción de relevancia de los datos analizados hasta el momento y se puede correr el riesgo de descartar información importante.

En lo que respecta al trabajo futuro en relación al algoritmo DyClee, resulta interesante implementar un módulo que en base a un conjunto inicial de datos de entrada permita determinar cuál es una configuración inicial correcta para los hiperparámetros del algoritmo. La necesidad de implementar el módulo en cuestión surge dada la gran cantidad de diferentes hiperparámetros que posee el algoritmo, lo que hace que se requiera un esfuerzo extra a la hora de elegir el valor que tomará cada uno. Es importante tener en cuenta que esta etapa es crucial para luego obtener buenos resultados al agrupar.

Ya culminando, otro trabajo interesante, vinculado al *clustering* dinámico en general, es proveer un mecanismo de selección de atributos, dirigido cien por ciento a la agrupación dinámica, capaz de ordenar los diferentes atributos disponibles de forma *online*, de acuerdo con su capacidad para poder separar los datos, la cuál a su vez puede ir variando en el tiempo.

La idea detrás de este trabajo es cambiar los atributos elegidos para determinar si un elemento entrante se parece o no a otros previamente recibidos como para ubicarse dentro de un mismo *microcluster*, y luego pasar a representar cierto concepto. Así, el algoritmo en cuestión lograría adaptarse de forma *online*, interpretando la característica separadora de cada atributo, pudiendo desempeñar una mejor tarea a la hora de evaluar la cercanía entre muestras y *microclusters* existentes, considerando un espacio de múltiples dimensiones.

[2] `https://www.kaggle.com/henrikengdahl/taximovementconcatenated`

Bibliografía

[1] C. Aggarwal y C. Reddy: *Data Clustering: Algorithms and Applications*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press, (2018).

[2] P.-N. Tan y col.: *Introduction to Data Mining (2nd Edition)*. 2nd. Pearson, (2018).

[3] B. Everitt y col.: *Cluster Analysis*. Wiley Series in Probability and Statistics. Wiley, (2011).

[4] P. Giordani, M. Ferraro y F. Martella: *An Introduction to Clustering with R*. Behaviormetrics: Quantitative Approaches to Human Behavior. Springer Nature Singapore, (2020).

[5] B. S. Everitt, S. Landau y M. Leese: *Cluster Analysis*. 4th. Wiley Publishing, (2009).

[6] I. Witten y col.: *Data Mining: Practical Machine Learning Tools and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, (2016).

[7] D. Pandove, S. Goel y R. Rani: Systematic Review of Clustering High-Dimensional and Large Datasets. En: *ACM Trans. Knowl. Discov. Data* 12.2 (2018).

[8] G. Hamerly: Learning Structure and Concepts in Data using Data Clustering, PhD Thesis. (2003).

[9] F. G. Ahmatshin y L. A. Kazakotsev: Impact of data normalization methods and clustering model in the problem of automatic grouping of industrial products. En: *Journal of Physics: Conference Series* 1679.3 (2020).

[10] C. Aggarwal, A. Hinneburg y D. Keim: On the Surprising Behavior of Distance Metric in High-Dimensional Space. En: *Database Theory — ICDT 2001. ICDT 2001. Lecture Notes in Computer Science* 1973 (2002).

[11] P. A. Gore: 11 - Cluster Analysis. En: *Handbook of Applied Multivariate Statistics and Mathematical Modeling*. Ed. por H. E. Tinsley y S. D. Brown. San Diego: Academic Press, (2000).

[12] T. J. Loftus y col.: Phenotype clustering in health care: A narrative review for clinicians. En: *Frontiers in artificial intelligence* 5.842306 (2022).

[13] L. Ližbetinová y col.: Application of cluster analysis in marketing communications in small and medium-sized enterprises: An empirical study in the Slovak Republic. En: *Sustainability* 11.8 (2019).

[14] T. Liu, C. Rosenberg y H. A. Rowley: Clustering Billions of Images with Large Scale Nearest Neighbor Search. En: *2007 IEEE Workshop on Applications of Computer Vision (WACV '07)*. (2007).

[15] L. Kaufman y P. Rousseeuw: *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Series in Probability and Statistics. Wiley, (2009).

[16] J. Han, J. Pei y M. Kamber: *Data mining: concepts and techniques*. Elsevier, (2011).

[17] D. Monaco: Recursive data clustering through finding vague solutions. (2019).

[18] C. Ma y J. Wu: *Data Clustering: Theory, Algorithms, and Applications*. Vol. 20. (2007).

[19] V. H: *Fuzzy C-mean Clustering using Data Mining*. BookRix, (2019).

[20] E. H. Ruspini, J. C. Bezdek y J. M. Keller: Fuzzy Clustering: A Historical Perspective. En: *IEEE Computational Intelligence Magazine* 14.1 (2019).

[21] T. Ren y col.: Study on the improved fuzzy clustering algorithm and its application in brain image segmentation. En: *Applied Soft Computing* 81 (2019).

[22] W. Jang, Y. Park y H. Seol: Identifying emerging technologies using expert opinions on the future: A topic modeling and fuzzy clustering approach. En: *Scientometrics* (2021).

[23] S. Kamande, K. Miriti y E. Ahishakiye: Consumer Segmentation and Profiling using Demographic Data and Spending Habits Obtained through Daily Mobile Conversations. En: *International Journal of Computer Applications* 181 (2018).

[24] R. Campello, D. Moulavi y J. Sander: Density-Based Clustering Based on Hierarchical Density Estimates. En: vol. 7819. (2013).

[25] T. Zhang, R. Ramakrishnan y M. Livny: BIRCH: an efficient data clustering method for very large databases. En: *ACM Sigmod Record* 25.2 (1996).

[26] G. Karypis, E.-H. Han y V. Kumar: Chameleon: Hierarchical clustering using dynamic modeling. En: *Computer* 32.8 (1999).

[27] S. Guha, R. Rastogi y K. Shim: CURE: an efficient clustering algorithm for large databases. En: *ACM Sigmod record* 27.2 (1998).

[28] K. Peng, V. C. M. Leung y Q. Huang: Clustering Approach Based on Mini Batch Kmeans for Intrusion Detection System Over Big Data. En: *IEEE Access* 6 (2018).

[29] C. Yuan y H. Yang: Research on K-Value Selection Method of K-Means Clustering Algorithm. (2019).

[30] K. P. Sinaga y M.-S. Yang: Unsupervised K-Means Clustering Algorithm. En: *IEEE Access* 8 (2020).

[31] R. T. Ng y J. Han: CLARANS: A method for clustering objects for spatial data mining. En: *IEEE transactions on knowledge and data engineering* 14.5 (2002).

[32] H. Kim, H. K. Kim y S. Cho: Improving spherical k-means for document clustering: Fast initialization, sparse centroid projection, and efficient cluster labeling. En: *Expert Systems with Applications* 150 (2020).

[33] Z. Jiang: A New Initialization Method for K-means Algorithm Based on Clustering Coefficient. En: *Journal of Physics: Conference Series* 1992.4 (2021).

[34] P. Fränti y S. Sieranoja: How much can k-means be improved by using better initialization and repeats? En: *Pattern Recognition* 93 (2019).

[35] Z. Huang: Extensions to the k-means algorithm for clustering large data sets with categorical values. En: *Data mining and knowledge discovery* 2.3 (1998).

[36] S. Thilagamani, A. Jayanthiladevi y N. Arunkumar: Data mining algorithms, fog computing. (2018).

[37] M. Ester y col.: A density-based algorithm for discovering clusters in large spatial databases with noise. En: *Knowledge Discovery and Data Mining*. Vol. 96. 34. (1996).

[38] M. Ankerst y col.: OPTICS: ordering points to identify the clustering structure. En: *ACM Sigmod record* 28.2 (1999).

[39] A. Hinneburg y D. A. Keim: A general approach to clustering in large databases with noise. En: *Knowledge and Information Systems* 5.4 (2003).

[40] S.-S. Li: An improved DBSCAN algorithm based on the neighbor similarity and fast nearest neighbor query. En: *Ieee Access* 8 (2020).

[41] Y. Chen y col.: BLOCK-DBSCAN: Fast clustering for large scale data. En: *Pattern Recognition* 109 (2021).

[42] Y. Chen y col.: A fast clustering algorithm based on pruning unnecessary distance computations in DBSCAN for high-dimensional data. En: *Pattern Recognition* 83 (2018).

[43] L. Loong y col.: Self-organized Population Segmentation for Geosocial Network Neighborhood. En: *International Journal of Advanced Computer Science and Applications* 9 (2018).

[44] D. Barbará: Requirements for clustering data streams. En: *ACM Newsletter of the Special Interest Group on Knowledge Discovery and Data Mining* 3.2 (2002).

[45] M. Carnein: *Machine Learning on Data Streams: Improving the Applicability and Performance of Stream Clustering Algorithms*. Westfälische Wilhelms-Universität Münster, (2019).

[46] F. Cao y col.: Density-based clustering over an evolving data stream with noise. En: *Proceedings of the 2006 SIAM international conference on data mining*. SIAM. (2006).

[47] N. B. Roa, L. Travé-Massuyès y V. H. Grisales-Palacio: DyClee: Dynamic clustering for tracking evolving environments. En: *Pattern Recognition* 94 (2019).

[48] L. O'callaghan y col.: Streaming-data algorithms for high-quality clustering. En: *Proceedings 18th International Conference on Data Engineering*. IEEE. (2002).

[49] D. Barbará y P. Chen: Using the fractal dimension to cluster datasets. En: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. (2000).

[50] J. Silva y col.: Data Stream Clustering: A Survey. En: *ACM Computing Surveys* 46 (2014).

[51] J.-Y. Chen y H.-H. He: A fast density-based data stream clustering algorithm with cluster centers self-determined for mixed data. En: *Information Sciences* 345 (2016).

[52] A. Amini y T. Y. Wah: Density Micro-Clustering Algorithms on Data Streams: A Review. (2011).

[53] M. Shutaywi y N. Kachouie: Silhouette Analysis for Performance Evaluation in Machine Learning with Applications to Clustering. En: *Entropy* 23.6 (2021).

[54] D. L. Davies y D. W. Bouldin: A cluster separation measure. En: *IEEE transactions on pattern analysis and machine intelligence* 2 (1979).

[55] M. Halkidi y M. Vazirgiannis: A density-based cluster validity approach using multi-representatives. En: *Pattern Recognition Letters* 29.6 (2008).

[56] M. Halkidi, M. Vazirgiannis e Y. Batistakis: Quality scheme assessment in the clustering process. En: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer. (2000).

[57] D. J. Hand y R. J. Till: A simple generalisation of the area under the ROC curve for multiple class classification problems. En: *Machine learning* 45.2 (2001).

[58] D. Moulavi y col.: Density-based clustering validation. En: *Proceedings of the 2014 SIAM international conference on data mining*. SIAM. (2014).

[59] Y. Raykov y col.: What to Do When K-Means Clustering Fails: A Simple yet Principled Alternative Algorithm. En: *PLOS ONE* 11 (2016).

[60] S. Chandrasekaran y A. Kumar: A Clustering Approach for Customer Billing Prediction in Mall: A Machine Learning Mechanism. En: *Journal of Computer and Communications* 07 (2019).

[61] F. Pedregosa y col.: Scikit-learn: Machine learning in Python. En: *the Journal of machine Learning research* 12 (2011).