Departamento de Computación
Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Método adaptativo en tiempo real para la detección de anomalías mediante aprendizaje automático

Estudiante: David Novoa Paradela
Directores: Óscar Fontenla Romero
Bertha María Guijarro Berdiñas

A Coruña, 5 de septiembre de 2019.

A mi familia y amigos.

Agradecimientos

Quisiera agradecer a varias personas y entidades la ayuda que me han prestado para la realización de este Trabajo de Fin de Grado:

En primer lugar, a mis directores Óscar Fontenla y Bertha Guijarro, por todo lo que me han enseñado y el apoyo constante desde el primer día.

Al Laboratorio de Investigación y Desarrollo en Inteligencia Artificial [1] de la Universidad de A Coruña, por confiar en mí y ofrecerme mi primer empleo.

A la propia Facultad de Informática y a sus profesores, por estos años de esfuerzo y dedicación.

Finalmente, a mi familia y amigos, por alentarme y estar siempre a mi lado.

Resumen

La detección de anomalías es la subrama del aprendizaje automático encargada de construir modelos capaces de diferenciar entre datos normales y anómalos. Ya que los datos normales son los que prevalecen en estos escenarios y sus características suelen ser bien conocidas, el entrenamiento de los sistemas se realiza mayormente mediante estos conjuntos de datos normales, lo que diferencia a la detección de anomalías de otros problemas de clasificación estándar. Debido al habitual uso de estos sistemas en monitorización y a la inexistencia de métodos capaces de aprender en tiempo real, en este proyecto de investigación se presenta un nuevo método que proporciona dicha capacidad de adaptación *online*. El método desarrollado recibe el nombre de *OnlineS-DSCH* (*Online and Subdivisible Distributed Scaled Convex Hull*) y basa su funcionamiento en las propiedades de los cierres convexos. Tras su desarrollo, se ha evaluado y comparado su rendimiento con los principales algoritmos del área sobre diferentes conjuntos de datos, tanto reales como artificiales. Como consecuencia, se ha obtenido un algoritmo con la capacidad de aprendizaje *online*, fácilmente configurable y cuyas predicciones son justificables, todo ello sin que suponga una merma en su eficacia en relación a las otras soluciones disponibles. Por último, su ejecución se puede llevar a cabo de manera distribuida y en paralelo, lo que supone una ventaja interesante en el tratamiento de conjuntos de datos de alta dimensionalidad.

Abstract

In machine learning, anomaly detection is the branch responsible for building models capable of differentiating between normal and anomalous data. Normal data prevail in scenarios of this type, and their features are usually well known, so the training phase is largely done through these normal data sets, which differentiates the anomaly detection from other standard classification problems. Due to the usual use of these systems in monitoring and the lack of methods capable of learning in real time, this research project presents a new method that provides such online adaptability. The method developed is called *OnlineS-DSCH* (*Online and Subdivisible Distributed Scaled Convex Hull*) and bases its operation on the properties of convex hulls. After its development, its performance has been evaluated and compared with the main algorithms of the area on different real and artificial data sets. As a consequence, an algorithm with the online learning capacity, easily configurable and whose predictions are justifiable, has been obtained, all without diminishing its effectiveness in relation to the other available solutions. Finally, its execution can be carried out in a distributed and parallel way, which is an interesting advantage in the treatment of high dimensionality data sets.

Palabras clave:

- Aprendizaje Automático
- Detección de Anomalías
- Clasificación de Una Clase
- Aprendizaje Distribuido
- Cierre Convexo
- Monitorización
- Python

Keywords:

- Machine Learning
- Anomaly Detection
- One-Class Classification
- Distributed Learning
- Convex Hull
- Monitoring
- Python

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En la actualidad la Inteligencia Artificial se ha proclamado como un pilar fundamental para numerosos mercados y sectores, muchos de ellos hasta ahora inviables hasta la aplicación de esta tecnología. Gracias a la automatización de tareas, la gestión de los pedidos y *stocks* de gigantescas naves industriales nunca se habrían llevado a cabo de una manera tan rápida y eficiente (p. ej. gestión automatizada de almacenes de Amazon [2]), la predicción del consumo eléctrico de ciudades ha pasado a ser un problema fácilmente abordable permitiendo el ahorro de enormes cantidades de energía y dinero (p. ej. predicción del consumo eléctrico en Chipre [3]), e incluso controlar el estado de un paciente o el diagnóstico de enfermedades se ha visto reforzado siendo ahora más seguro (p. ej. diagnóstico de cáncer [4]).

No obstante, los sistemas inteligentes han sobrepasado los límites meramente industriales para llegar desde hace ya unos años a nuestras vidas como nuevas herramientas que facilitan nuestro día a día. Sin ir muy lejos, nuestros teléfonos inteligentes son un ejemplo de ello, incorporando asistentes virtuales capaces de reconocer nuestra voz, comprender nuestras órdenes y llevar a cabo pequeñas tareas [5]. Además de esto, incluyen sistemas de seguridad que basados en reconocimiento facial y de huellas dactilares [6] blindan el desbloqueo de nuestro dispositivo o aseguran el acceso a nuestra información bancaria.

El auge de la inteligencia artificial en los últimos años, y concretamente del aprendizaje automático, no es debido únicamente al desarrollo de nuevas técnicas y algoritmos más eficientes, sino también a la alta capacidad de cómputo que hemos alcanzado gracias a los avances en *hardware*. Este poder de procesamiento sumado a las mejoras logradas en la transmisión de la información y a la facilidad actual de capturar y tratar con grandes volúmenes de datos (*Big Data*), es lo que desemboca en un salto natural desde las tecnologías y herramientas convencionales hacia estas nuevas técnicas automáticas más apropiadas para el análisis y tratamiento de conjuntos de datos grandes, volátiles y heterogéneos.

Tratar con tremendos volúmenes de información, permite disponer en muchos casos de una fuente fiable y valiosa sobre la que desarrollar sistemas que modelan dichos conjuntos de datos, de forma que nos capacitan para encontrar singularidades, clasificar nuevos datos o predecir tendencias en comportamientos futuros. En muchos casos, es necesario desarrollar técnicas capaces de seguir aprendiendo tras su entrenamiento inicial debido a la naturaleza cambiante del problema. Estas nuevas formas de análisis son las que permiten a las empresas identificar problemas, utilizar tendencias mostradas en los datos para la toma de decisiones, reducir costes y crear nuevos productos y servicios con la capacidad de medir las necesidades y satisfacción de los clientes.

Ni en las ciencias de la computación, ni concretamente en el aprendizaje automático, existen algoritmos generales capaces de resolver todos los posibles problemas a los que se enfrentan. Los teoremas *No free lunch* [7] echan por tierra la idea de un gran algoritmo inteligente que resuelva todo tipo de problemas. Debido a la gran diversidad que existe, los algoritmos de aprendizaje automático se agrupan en taxonomías, por ejemplo, en función de la salida que emiten (discreta o continua), o en si conocemos *a priori* la clase de los datos con los que se entrenará el sistema. Podemos basarnos en la geometría de los datos, probabilidad o en la lógica, entre otros, para generar modelos que resuelvan nuestra tarea, según la naturaleza de la misma, de la forma más conveniente.

Por tanto, cada método de aprendizaje automático tiene una forma distinta de representar el conocimiento, de manera que cada uno es adecuado para un tipo de problemas, obteniendo así mayores rendimientos al tratarse de soluciones *ad hoc*. Este planteamiento, provoca que para resolver un problema específico dispongamos de una gran variedad de técnicas más o menos apropiadas para su resolución. Así, algunas optarán por una mayor velocidad de procesado a costa de perder precisión, o viceversa, dándose casos de complejidades inabordables o tiempos muy altos para su aplicación en sistemas de tiempo real, si la elección de la solución no es la apropiada. Estas ventajas y desventajas son a las que hemos de enfrentarnos a la hora de construir nuestro sistema, y por ello es importante tener conocimiento del tipo de problema a resolver antes de aplicar un algoritmo.

Una de las subramas del aprendizaje automático, y la que se va a estudiar en este proyecto, es la detección de anomalías. Esta se basada fundamentalmente en construir modelos capaces de diferenciar entre datos normales y anómalos, normalmente relacionados con el funcionamiento de un sistema. Como veremos, para solucionar este tipo de problemas existe una gran variedad de aproximaciones y su uso es habitual en sistemas de monitorización.

1.1 Motivación del proyecto y objetivos

La monitorización de una infraestructura o sistema consiste en controlar y supervisar su estado de forma constante (en tiempo real), analizando la información que genera de manera que se puedan localizar componentes o estados del sistema anormales. Por otro lado, como hemos comentado, en la actualidad el flujo de datos con el que tratan las empresas y por tanto sus sistemas ha crecido sustancialmente. Debido a ello y al uso extendido de la detección de anomalías en problemas de monitorización, en este proyecto de investigación se analizarán las técnicas actuales utilizadas en detección de anomalías para construir un nuevo algoritmo que proporcione la capacidad de adaptación en tiempo real en problemas de este tipo. Es importante que dicho algoritmo sea capaz de adaptarse ante la llegada de nuevos datos en tiempo real que puedan provocar reajustes en la forma del clasificador. El objetivo principal de este proyecto es, por tanto, el desarrollo de un método de aprendizaje automático adaptativo en tiempo real para la detección de anomalías. Para ello, se llevarán a cabo los siguientes subobjetivos:

- Se realizará un estudio completo de las propuestas actuales utilizadas en detección de anomalías.
- Se determinarán los tipos de anomalías a manejar (dimensionalidad, forma, etc).
- Se desarrollará el nuevo algoritmo adaptativo tomando como punto de partida los modelos de cierre convexo.
- Se realizará un estudio experimental del funcionamiento del algoritmo comparando sus resultados con los obtenidos por otras técnicas actuales.

1.2 Recursos tecnológicos

A continuación se detallan los recursos tecnológicos empleados durante el desarrollo de este proyecto de fin de grado.

1.2.1 Lenguaje de programación

Debido a la gran potencia de cálculo que posee, su sencillez, y la abundancia de librerías y recursos, el lenguaje de programación escogido es *Python* [8]. *Python* es un lenguaje interpretado que apuesta por la transparencia y un código legible. Además, es un lenguaje multiparadigma, ya que soporta orientación a objetos, programación imperativa y programación funcional. Es un lenguaje fuertemente tipado, dinámico y multiplataforma.

Debido a estas características y a una fuerte comunidad, se ha consolidado como un lenguaje idóneo para desarrollar sistemas basados en Inteligencia Artificial. Dispone de numerosas librerías que implementan algoritmos para aprendizaje automático, así como para el manejo y procesado de grandes cantidades de datos. Las librerías más importantes que se utilizarán en este proyecto son:

- **Numpy**: *Numpy* [9] es una librería de uso libre que facilita las operaciones y la manipulación de matrices. En nuestro proyecto, se operará con una gran cantidad de datos de forma constante, por lo que las funciones que *Numpy* proporciona son muy útiles.
- **Scikit-learn**: *Scikit-learn* [10] es una librería de aprendizaje automático de uso libre. Implementa algoritmos de clasificación, regresión y agrupamiento entre otros, y permite su uso de forma totalmente compatible con *Numpy*. Se utilizará para comparar los resultados de nuestro algoritmo frente a los métodos clásicos que *Scikit-learn* incorpora.

1.2.2 Herramientas

El desarrollo del proyecto se ha apoyado en las siguientes herramientas auxiliares:

- **PyCharm**: *PyCharm* [11] es un entorno de desarrollo integrado (IDE) creado específicamente para el lenguaje *Python*. Es multiplataforma y proporciona una gran variedad de herramientas y utilidades, como su integración con controladores de versiones (como *GitHub*). Su uso es gratuito.
- **Kaggle**: *Kaggle* [12] es una comunidad científica que permite a sus usuarios publicar sus propias bases de datos, modelos de aprendizaje o incluso resolver competiciones. Para probar la eficacia del algoritmo desarrollado en este proyecto, se han utilizado bases de datos obtenidas en esta plataforma. Su uso es gratuito.
- **IEEE Xplore**: *IEEE Xplore* [13] es una enorme biblioteca digital que alberga documentos de carácter científico, como artículos de revistas científicas o conferencias, todas ellas relacionadas con las ciencias de la computación, la ingeniería eléctrica y la electrónica. Los recursos teóricos utilizados para llevar a cabo este proyecto se han obtenido de esta plataforma. Su uso ha sido posible ya que la Facultad de Informática de A Coruña dispone de licencias para el alumnado.
- **Trello**: *Trello* [14] es un software de administración de proyectos con interfaz web. Mediante tarjetas virtuales, permite organizar tareas, agregar listas, adjuntar archivos o etiquetar eventos entre otros. Esta estructura en tablón virtual es versátil y fácil de usar, convirtiéndolo en una herramienta muy visual para organizar información. Su

uso es gratuito, y en este proyecto se ha utilizado para organizar las diferentes tareas y requisitos durante la implementación.

- **GitHub**: *GitHub* [15] es una plataforma de control de versiones para proyectos software. Su función principal es facilitar el desarrollo en grupo, permitiendo almacenar distintas versiones de los archivos, recuperar, y fusionar el trabajo realizado por distintos programadores. Su uso en la versión pública es gratuito.

- **Google Drive**: *Google Drive* [16] es un servicio de almacenamiento en la nube propiedad de *Google*. Su función en este proyecto es almacenar archivos no tan relacionados con el código como los ya alojados en *GitHub*. Un ejemplo son las hojas de cálculo dónde se almacenan los resultados obtenidos por el algoritmo, con sus correspondientes estadísticas, o los documentos de texto en los que durante el desarrollo se han ido describiendo los pasos y decisiones tomadas. Su uso mediante el plan de almacenamiento más básico es gratuito.

- **TexMaker**: *TexMaker* [17] es un editor gratuito multiplataforma para la escritura de documentos de texto mediante *LaTeX*. Es de gran utilidad ya que reúne una gran variedad de funciones en una sola aplicación. Esta herramienta ha sido la empleada para la redacción de la memoria del trabajo.

1.3 Metodología del desarrollo y costes del proyecto

En esta sección se describen las diferentes etapas de la metodología escogida para el desarrollo de este Trabajo de Fin de Grado. También se exponen las estimaciones de tiempo de cada fase y el coste total del proyecto.

1.3.1 Fases

Debido al carácter investigador del proyecto, su desarrollo se ha llevado a cabo mediante una metodología basada en el método científico. Las etapas que conforman el desarrollo del proyecto son las siguientes:

- **Fase 1: Exploración**. En esta fase se realiza un estudio del estado actual del campo de la detección de anomalías. Tras el análisis y entendimiento de sus conceptos teóricos fundamentales, se analizan las técnicas más empleadas en este tipo de problemas, concretamente las basadas en cierre convexo.

- **Fase 2: Diseño**. Tras analizar las posibles vías de intervención, se concretan los objetivos concretos del proyecto. Se diseña la solución al problema planteado y las diferentes pruebas y conjuntos de datos que evaluarán su cumplimiento.

- **Fase 3: Implementación**. Se lleva a cabo la implementación de la solución en el lenguaje escogido. Se implementan las diferentes funcionalidades de forma progresiva hasta cumplir todos los requisitos y alcanzar una solución final.

- **Fase 4: Pruebas y evaluación**. En esta fase se mide la eficacia y eficiencia de la solución final en comparación con otros algoritmos de referencia en el campo de la detección de anomalías determinados en la Fase 1. Estas pruebas se llevan a cabo sobre los distintos conjuntos de datos escogidos en la segunda fase. De esta forma, se comprueba si el método cumple los objetivos establecidos en la fase de diseño, y se buscan carencias en el método que podrían implicar un refinamiento de la implementación y, por tanto, una vuelta a la Fase 2.

- **Fase 5: Redacción de la memoria**. Esta fase se lleva a cabo de forma paralela a todas las demás desde el comienzo de la fase de diseño. En ella, se lleva a cabo la redacción de la memoria de este Trabajo de Fin de Grado, en la que se recoge toda la información relevante.

1.3.2 Costes

Para representar la disposición de las diferentes fases del proyecto a lo largo del tiempo y sus dependencias, se ha generado un diagrama de *Gantt* (véase Figura 1.1). Para calcular el coste estimado del proyecto se ha tenido en cuenta únicamente los días laborables. La fecha de inicio del trabajo fue el 1/02/2019 y finalizó el 4/09/2019, lo que suma un total de 149 días laborables. Suponiendo una retribución de 16€/hora [18], y una jornada laboral a tiempo parcial (4h/día), el coste total del proyecto ascendería a 9.536€ + IVA.

El origen de este coste proviene en su totalidad del pago por recursos humanos. No se han incluido costes provenientes de recursos materiales ya que no ha sido necesario adquirir ningún tipo de elemento *hardware* y el utilizado ya estaba amortizado. Además, todo el *software* utilizado es completamente gratuito, o en caso contrario, la Facultad de Informática dispone de licencias para su uso educativo.

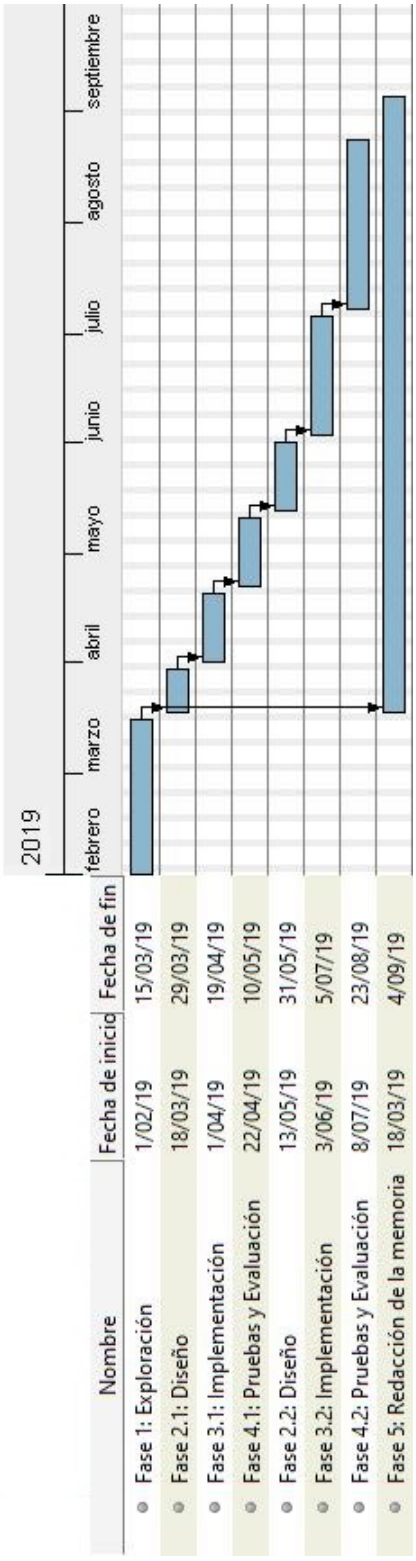

Figura 1.1: Diagrama de Gantt del desarrollo de este Trabajo de Fin de Grado.

1.4 Organización del proyecto

La memoria de este Trabajo de Fin de Grado se estructurará de la siguiente manera:

- **Contextualización y estado del arte**: En el Capítulo 2 se introducirá y explicarán los conceptos básicos en aprendizaje automático, se profundizará en el campo de la detección de anomalías y se estudiarán las técnicas principales en la actualidad.
- **Fundamentos teóricos**: En el Capítulo 3 se explicarán los fundamentos teóricos de las técnicas basadas en los cierres convexos.
- **Propuesta de un algoritmo adaptativo en tiempo real**: En el Capítulo 4 se analizará el desarrollo de un nuevo algoritmo adaptativo en tiempo real para la detección de anomalías.
- **Estudio de rendimiento del algoritmo desarrollado**: En el Capítulo 5 se estudiarán los resultados obtenidos por el algoritmo desarrollado en comparación con otras técnicas clásicas para evaluar su rendimiento.
- **Conclusiones y trabajo futuro**: En el Capítulo 6 se expondrán las conclusiones obtenidas y las posibles líneas de trabajo futuro.

Capítulo 2

Contextualización y estado del arte

El aprendizaje automático es una rama de la inteligencia artificial con más de medio siglo de existencia y desarrollo. Aunque su etapa triunfal se esté dando en estos momentos, muchos de los planteamientos utilizados contienen y se basan en ideas propuestas hace décadas. A modo de contextualización, se va a realizar una síntesis de los conceptos clave en aprendizaje automático, de forma que luego se pueda comprender de una manera más precisa la detección de anomalías y la propuesta sobre la que se construirá nuestro algoritmo.

2.1 Aprendizaje automático

El aprendizaje automático, aprendizaje máquina o *machine learning* es un subcampo de la ciencia de la computación y una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras obtener conocimientos (aprender) a partir de datos de manera similar a la del ser humano. Esta filosofía surge como una nueva forma de programación automática, en la que los sistemas se desarrollan solos a base de ejemplos, induciendo el conocimiento. Esta visión es muy beneficiosa cuando nos encontramos ante problemas mal definidos o para los cuales no se puede programar una solución mediante métodos clásicos, como por ejemplo, personalizar las noticias que se muestran a los usuarios según sus intereses [19].

Los diferentes algoritmos de aprendizaje automático se agrupan por la naturaleza del proceso de aprendizaje empleado. Algunos tipos son:

- **Aprendizaje supervisado**: El comportamiento deseado se representa por medio de un conjunto de ejemplos, de los cuales conocemos *a priori* cuál debería ser la respuesta correcta del sistema. El algoritmo a partir de estos ejemplos produce una función que establece una correspondencia entre las entradas y las salidas deseadas. Por tanto, la base de conocimiento del sistema está formada por ejemplos etiquetados, que permiten

definir un criterio para evaluar el comportamiento real del sistema. Su aplicación es común en problemas de clasificación (e.g clasificación de imágenes de comida [20]) y regresión.

- **Aprendizaje no supervisado**: Todo el proceso de modelado se lleva a cabo sobre un conjunto de ejemplos formado tan solo por entradas del sistema, de las que desconocemos sus categorías (etiquetas). No se define ningún tipo de señal que indique un comportamiento deseado para el sistema. El criterio de evaluación se basa en la regularidad de los grupos de datos, es decir, el sistema tiene que ser capaz de reconocer patrones para poder etiquetar las nuevas entradas.

- **Aprendizaje semisupervisado**: Utiliza datos de entrenamiento tanto etiquetados como no etiquetados, normalmente una pequeña cantidad de datos etiquetados junto a una gran cantidad de datos no etiquetados. El etiquetado de los datos suele llevarse a cabo por un agente humano capacitado para clasificar de forma manual dichos ejemplos de entrenamiento. El coste asociado a este proceso puede hacer que un conjunto de entrenamiento totalmente etiquetado sea inviable, mientras que la adquisición de datos sin etiquetar es relativamente poco costosa. En estos casos la utilización de una aproximación híbrida puede ser muy ventajosa.

- **Aprendizaje por refuerzo**: El comportamiento deseado no se representa mediante ejemplos sino mediante la evaluación de los resultados que genera el sistema con su entorno. La información de entrada del sistema es la retroalimentación o *feedback* que obtiene del mundo exterior como respuesta a sus acciones, por lo tanto, el sistema aprende a base de ensayo y error. El agente inteligente debe aprender cómo se comporta el entorno mediante recompensas (refuerzos) o castigos, derivados del éxito o del fracaso respectivamente. El objetivo principal es aprender la función de valor que le ayude al agente inteligente a maximizar la señal de recompensa. Su uso es común en robótica autónoma (p. ej. navegación de robots mediante aprendizaje por refuerzo [21]).

Además de esto, podemos agrupar los problemas de aprendizaje automático en cuatro tipos según su naturaleza:

- **Problemas de regresión**: Se caracterizan por que las variables de salida del sistema son continuas en lugar de encontrarse restringidas por un grupo limitado de valores. Su objetivo es minimizar el error entre la función aproximada y el valor de la aproximación. Existen diferentes modelos que pueden ser usados para modelar dicha relación entre una variable dependiente Y y unas independientes X_i, siendo la regresión lineal la más simple (p. ej. predicción del abandono de clientes en una aerolínea mediante regresión [22]). El aprendizaje en estos problemas, por tanto, es de tipo supervisado.

- **Problemas de clasificación**: Se caracterizan por trabajar con variables de salida discretas o cualitativas, a veces recibiendo el nombre de categorías. En estos problemas el objetivo es determinar a qué clase pertenece el dato a evaluar, generalmente calculando la probabilidad de pertenencia a cada clase. El aprendizaje en estos problemas también es de tipo supervisado.

- **Problemas de agrupamiento o *clustering***: en estos problemas el objetivo es obtener información sobre el conjunto de datos para encontrar grupos o *clusters*. Buscamos particiones que dividan los datos en grupos distintos pero homogéneos, siendo dichos grupos lo más distintos posibles entre ellos pero muy similares internamente. El aprendizaje en estos problemas es no supervisado y las salidas son de tipo discreto (p. ej. división del problema del viajero a gran escala en subproblemas más simples mediante clustering [23]).

- **Problemas de reducción de la dimensionalidad**: el objetivo de estos problemas es reducir la dimensión o número de atributos que caracterizan a los datos. Esto es útil si se tiene una gran cantidad de variables difíciles de tratar o si existen variables poco representativas o redundantes. El aprendizaje en estos problemas es de tipo no supervisado (p. ej. reducción del número de características para la clasificación de gases [24]).

El tipo de problema hacia el que queremos orientar la técnica a desarrollar en este proyecto es la detección de anomalías. Esta se incluye dentro de los problemas de clasificación ya que se basa en llevar a cabo una clasificación en dos únicas clases, datos normales y datos anómalos. El aprendizaje es supervisado y usa datos de una única clase para entrenamiento, de ahí que se la conozca como clasificación de una clase (*one-class classification*).

2.2 Detección de anomalías

La detección de anomalías [25] se puede considerar como un caso particular de los problemas de clasificación en el que existen únicamente dos posibles clases, datos normales y anomalías. La idea principal en la detección de anomalías es que un sistema durante su correcto funcionamiento presenta patrones de comportamiento similares. Este estado de normalidad es el que predomina y puede invertirse si el sistema falla inesperadamente o entra en un estado poco habitual que lo puede conducir a fallar.

La detección de anomalías se aprovecha de que estos estados anómalos no encajan con los patrones de comportamiento habituales para entrenar clasificadores con datos normales, que modelan el estado natural del sistema y dotan al clasificador del poder de predecir a qué clase pertenecen nuevos datos. Datos que no se ajusten al comportamiento normal del sistema

aprendido por el clasificador serán etiquetados como anomalías y datos que sí lo hagan como normales.

El motivo de que estos clasificadores se entrenen únicamente con datos normales y no aprendan a modelar también el otro conjunto de datos es que las anomalías son muy poco comunes, heterogéneas y no siempre presentan comportamientos previsibles, por lo que no se puede saber *a priori* todas las posibles formas que pueden tomar. Por tanto, una solución más factible es modelar con precisión el conjunto de datos normales del cual se tiene un número considerable de ejemplos, y emplear este modelo para clasificar nuevos datos como normales o anómalos.

Otros conceptos como ruido y datos atípicos (*outliers*) también están presentes en esta clase de problemas pero difieren del significado de anomalía. Los datos del mundo real suelen presentarse de una forma ruidosa, este ruido en los datos se ve tradicionalmente como un error en las variables que se están midiendo. Estos errores si se encuentran en el conjunto de datos normal pueden empeorar el aprendizaje del modelo limitando su precisión en futuras predicciones, por ello es habitual limpiar el conjunto de datos para suavizar el efecto del ruido. Los datos *outliers* se diferencian de las anomalías en que estas llegan a formar un conjunto o clase diferenciable del resto y por tanto pueden ser modeladas y proporcionar información del estado del sistema, como por ejemplo un uso inusual de una tarjeta de crédito tras su robo. Los *outliers*, en cambio, solo son valores atípicos que pueden deberse al ruido presente en los datos pero no presentan una frecuencia de repetición suficiente para indicar un comportamiento crítico.

Un aspecto importante es la naturaleza de las anomalías, las cuales se pueden clasificar en las siguientes tres categorías:

- **Anomalías puntuales**: Es el tipo de anomalía más simple, si un dato se puede considerar anómalo respecto al resto de datos, el dato se denomina anomalía puntual. En la Figura 2.1, los puntos O_1 y O_2, así como los puntos de la región A se encuentran fuera de las regiones normales N_1 y N_2, por ello son anomalías puntuales.

- **Anomalías contextuales**: Si un dato es anómalo en un contexto específico (pero no para el todo) se denomina anomalía contextual. La definición de contexto es dependiente del conjunto de datos y debe especificarse en la formulación del problema. En la Figura 2.2 podemos ver un caso de anomalía contextual, correspondiente a los valores de la temperatura en una serie temporal. En cada periodo de tiempo T los valores de las temperaturas presentan crecimientos similares, de manera que en un instante t_i y

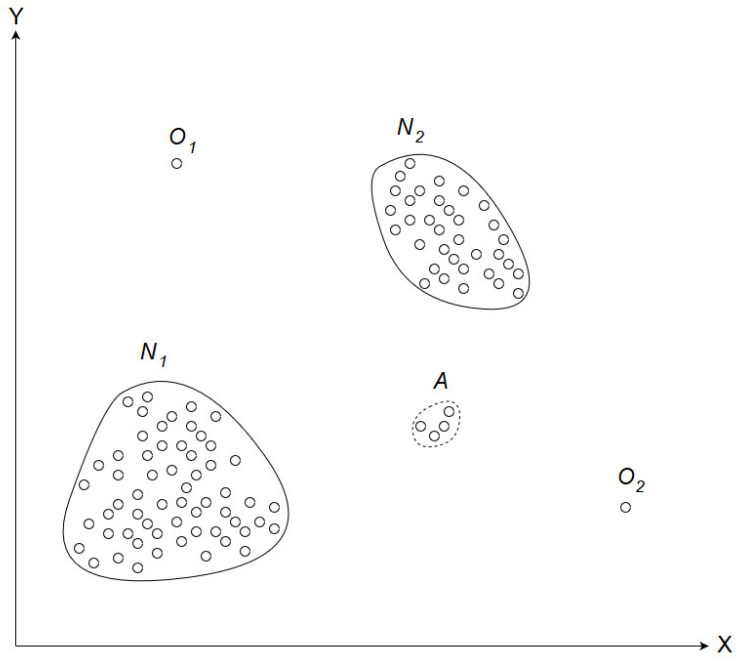

Figura 2.1: Anomalías puntuales en un conjunto de datos bidimensional.

t_{i+T} los valores de la temperatura serán muy parecidos. Esto ocurre en los instantes t_1 y t_2, en cambio en el instante t_3 no debería ocurrir ya que las temperaturas se espera que tomen valores mucho más grandes. La temperatura en t_3 no es irrealista, en otros instantes de tiempo se llega a alcanzar ese valor pero nunca en el contexto en el que se ha producido, de ahí que se les denomine anomalías contextuales.

- **Anomalías colectivas**: Si una colección de datos relacionados es anómala respecto al total de los datos, estamos ante una anomalía colectiva. Los datos individuales en una anomalía colectiva pueden no ser anomalías por sí mismas, pero su aparición formando una colección es anómala. En la Figura 2.3 se puede ver una aproximación de la forma que tendría la salida de un electrocardiograma humano. Las pulsaciones en los instantes t_1 y t_2 son normales, en cambio la región circundante al instante t_3 denota una anomalía porque existe el mismo valor bajo durante un tiempo excesivamente largo.

Debemos tener en cuenta que mientras que las anomalías puntuales pueden ocurrir en cualquier conjunto de datos, las anomalías colectivas solamente pueden hacerlo en conjuntos en los que los datos están relacionados. Por contra, la ocurrencia de anomalías contextuales depende de la disponibilidad de atributos de contexto en los datos. Anomalías puntuales o colectivas también pueden ser anomalías contextuales si se analizan respecto a un contexto, por lo tanto, un problema de detección de anomalías puntuales o un problema de detección

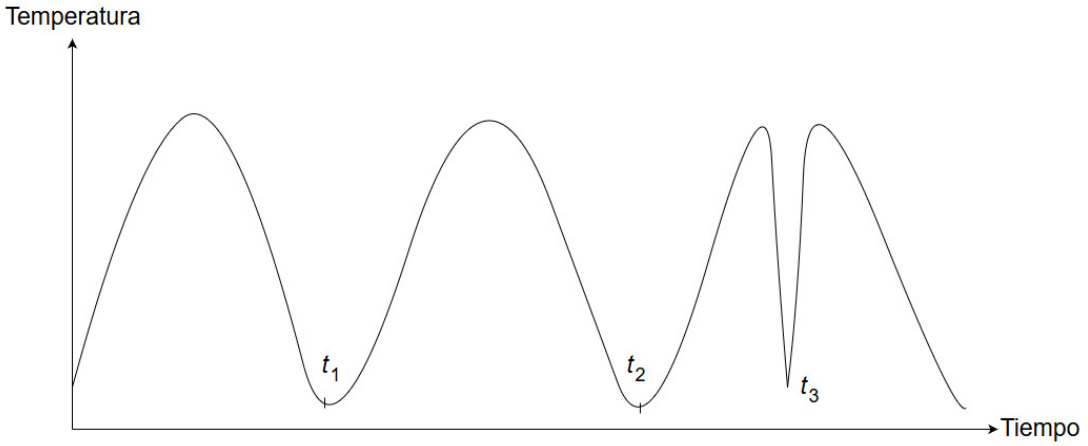

Figura 2.2: Anomalía contextual en temperatura.

de anomalías colectivas se puede transformar en un problema de detección de anomalías contextuales incorporando dicha información de contexto.

2.3 Estado actual de las técnicas

Las técnicas de detección de anomalías basadas en la clasificación trabajan en dos fases, una fase de entrenamiento y una de prueba o *test*. Durante la fase de entrenamiento el sistema aprende de los ejemplos disponibles generando un clasificador que, durante la fase de prueba, clasificará nuevos datos como normales o anómalos. Según las etiquetas disponibles para la fase de entrenamiento, las técnicas de detección de anomalías basadas en la clasificación se pueden agrupar en dos categorías:

- **Técnicas de detección de anomalías basadas en la clasificación *multi-class* o de clases múltiples**: Suponen que los datos de entrenamiento contienen datos etiquetados que pertenecen a múltiples clases normales o que pueden existir diferentes clases de anomalías. Desarrollan un clasificador para distinguir entre cada clase normal y el resto de las clases, de forma que un nuevo dato se considera anómalo si ninguno de los clasificadores lo considera normal. Algunas técnicas de esta subcategoría asocian una puntuación o *score* de confianza a la predicción obtenida.

- **Técnicas de detección de anomalías basadas en la clasificación *one-class* o de clase única**: Asumen que todos los datos de entrenamiento pertenecen solo a una clase. Estas técnicas modelan un límite en torno a los datos normales utilizando un algoritmo de clasificación de una clase, por ejemplo, *One-class Support Vector Machines* [26].

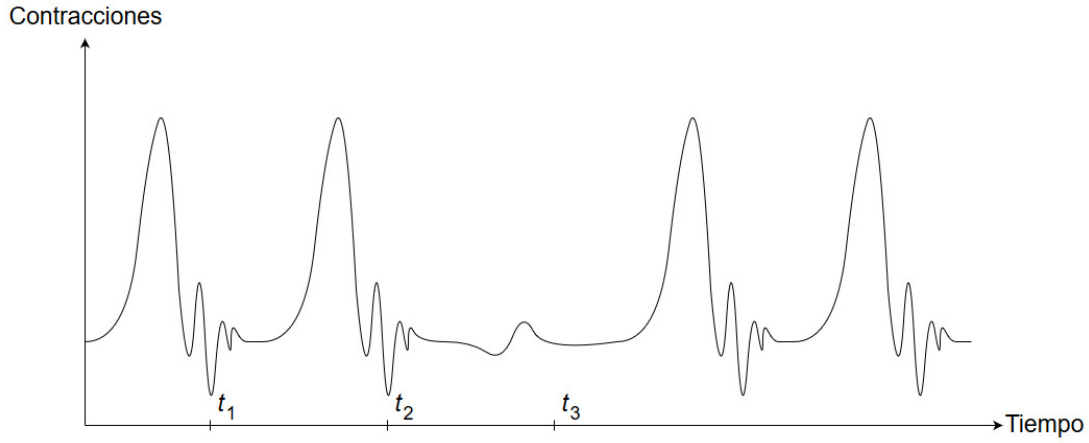

Figura 2.3: Anomalías colectivas en la monitorización del ritmo cardíaco.

Métodos probabilísticos	Aproximaciones paramétricas	Basados en modelos Gaussianos
		Basados en modelos de regresión
	Aproximaciones no paramétricas	Basados en histogramas
		Basados en kernels
Métodos basados en distancias	Vecino más cercano	kNN, LOF
	Agrupamiento	
Métodos basados en reconstrucción	Basados en redes neuronales	RNN
	Basados en subespacios	PCA
Métodos basados en el dominio	SVM, OSVM, SVDD	

Tabla 2.1: Esquema de los principales métodos en detección de anomalías.

Cualquier nuevo dato que no se encuentre dentro del límite aprendido se declara como anómalo.

En base a las clasificaciones de las diferentes técnicas realizadas por Chandola et al. [27] y Pimentel et al. [28], se van a estudiar las ideas principales de cada grupo de métodos, sus campos de aplicación y las ventajas y desventajas que presentan. En la Tabla 2.1 se muestra un resumen de los diferentes tipos de técnicas que se tratarán.

2.3.1 Métodos probabilísticos

El objetivo principal de los métodos basados en aproximaciones probabilísticas es la estimación de la función de densidad que genera los datos. Se asume que los datos del entrena-

miento son generados por una distribución de probabilidad D, la cual puede ser estimada mediante esos mismos datos. Esta distribución estimada $\hat{D}$ habitualmente representa un modelo de normalidad, los datos normales se darán en regiones con probabilidades altas del modelo y las anomalías en las regiones con probabilidades más bajas. De esta forma la clasificación de un nuevo dato mediante estos métodos se basa en determinar la probabilidad de que dicho dato pueda haber sido generado por $\hat{D}$, clasificándose como un dato normal si esta probabilidad supera un umbral establecido (el umbral será una probabilidad) o como un valor anómalo en caso contrario. Para modelar estas funciones de densidad se utilizan técnicas paramétricas y no paramétricas.

Aproximaciones paramétricas

Las aproximaciones paramétricas asumen conocimiento sobre la distribución y estiman sus parámetros. Parten de que los datos normales fueron generados por una distribución con parámetros Θ y una función de densidad de probabilidad $f(x, \Theta)$. Los parámetros Θ son estimados a partir del conjunto de datos de entrenamiento, de forma que se puede predecir sobre nuevos datos x. En función del tipo de distribución asumida, las técnicas paramétricas más habituales se pueden dividir en las siguientes clases:

- **Basadas en modelos Gaussianos**: Asumen que los datos fueron generados por una distribución Gaussiana (Figura 2.4). Los parámetros se calculan mediante una estimación por máxima verosimilitud o *Maximum Likelihood Estimation* (MLE), de forma que la puntuación o *score* obtenido por un dato a clasificar es su distancia respecto a la media estimada. Debido a sus propiedades analíticas, la distribución gaussiana se utiliza a menudo para determinar el *score* umbral.
- **Basadas en modelos de regresión**: Se basan en que toda variable aleatoria está caracterizada por su distribución de probabilidad, que no es sino el conjunto de valores posibles de la variable aleatoria, acompañados de sus respectivas probabilidades. Utilizados frecuentemente en series temporales, su funcionamiento suele dividirse en dos etapas. Una primera en la que un modelo de regresión se ajusta a los datos, y una segunda en la que para cada dato a predecir se asignan como *scores* los valores residuales. Un valor residual es la parte del valor calculado que no es explicada por el modelo de regresión, de ahí su uso como medida de anormalidad.

Aproximaciones no paramétricas

Las aproximaciones no paramétricas, a diferencia de las anteriores, no asumen conocimiento, la estructura del modelo no está definida *a priori*, pero se determina a partir de los datos.

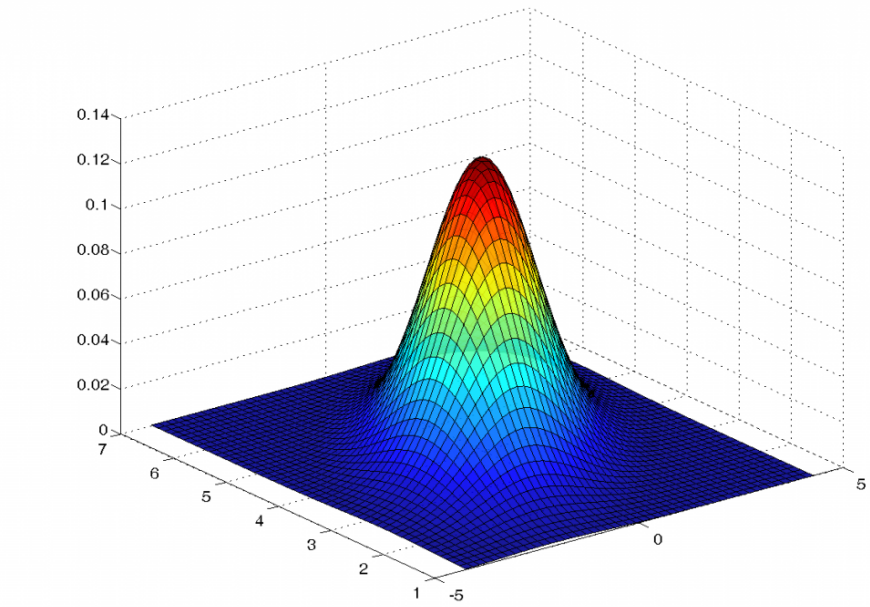

Figura 2.4: Distribucción Gaussiana multivariante en un espacio tridimensional.

- **Basadas en histogramas**: La técnica más simple es usar histogramas para generar un perfil de datos normales. También se conocen como técnicas basadas en la frecuencia o en el conteo. Su uso es habitual en detección de intrusiones y fraudes debido a que en estos problemas los datos están basados en un perfil (por ejemplo, el titular de la cuenta bancaria) modelable mediante histogramas. La detección se basa en dos etapas, siendo la primera la construcción de un histograma basado en los datos de entrenamiento. La segunda etapa consiste en la clasificación de nuevos datos. En este caso, se comprobará si los valores del nuevo dato coinciden con una de las columnas del histograma. Si coincide se considerará como normal, de no ser así como anómalo. La frecuencia de la columna en la que ha caído se puede utilizar como un *score*. El número de *bins* es clave a la hora de construir el histograma ya que valores muy grandes pueden dificultar la caída de nuevos datos en zonas vacías, lo que empeora la clasificación de anomalías.

- **Basadas en *kernels***: Estas técnicas se basan en funciones de *kernel* para estimar la función de distribución de probabilidad del conjunto normal de datos. Un ejemplo de técnica no paramétrica basada en *kernel* es la estimación de ventanas de Parzen [29].

Principales ventajas de las técnicas probabilísticas

- Si las suposiciones con respecto a la distribución de datos son ciertas, brindan una solución justificable estadísticamente.
- Los *scores* están asociados a un intervalo de confianza, por lo que proporcionan una información adicional que puede ser muy beneficiosa a la hora de clasificar nuevos datos.
- Si la etapa de estimación de la distribución es afín a las anomalías en los datos, este tipo de técnicas permiten operar en un entorno no supervisado con datos no etiquetados.
- Si las propiedades del conjunto de datos normal es aproximable mediante una función de densidad, trabajar directamente con dicha función y no con el gran conjunto de datos de entrenamiento es beneficioso para el sistema.
- Técnicas como la basada en histogramas pueden modelar con gran precisión perfiles de usuario de manera muy personalizada.

Principales desventajas de las técnicas probabilísticas

- Las aproximaciones paramétricas se basan en la asunción de que los datos se generan a partir de una distribución particular. Esto no siempre es así, sobre todo al trabajar con conjuntos de datos grandes con un alto número de variables.
- Elegir el mejor criterio estadístico para detectar anomalías en distribuciones de alta dimensionalidad no es una tarea trivial.
- Las técnicas basadas en histogramas no son suficientes para modelar la relación entre atributos en conjuntos de datos multivariables.
- La presencia de anomalías durante el entrenamiento puede impedir que las técnicas basadas en modelos de regresión obtengan buenos resultados.

2.3.2 Métodos basados en distancias

Estos métodos se basan en métricas de distancia para calcular la similitud entre dos datos. Las dos clases principales son las basadas en el vecino más cercano y las basadas en agrupamiento o *clustering*.

Vecino más cercano

Son los métodos más utilizados en detección de anomalías y asumen que los datos normales se dan en forma de vecindarios densos, mientras que las anomalías se encuentran lejos de dichos vecindarios. Para conjuntos de datos con atributos continuos, la distancia euclídea es la opción más habitual. Para atributos categóricos, se suele utilizar un coeficiente de coincidencia, aunque en ambos casos se pueden utilizar otras medidas. Para atributos multivariantes las distancias o similaridades se computan para cada atributo y se combinan.

La aproximación conocida como k-vecino más cercano o *k-nearest neighbour* (k-NN) [30] representa al grupo de técnicas más usado. Fundamenta su funcionamiento en que el *score* de un dato viene dado por la distancia a sus k vecinos más cercanos, de modo que se pueden clasificar nuevos datos en base a su *score* y un umbral, o al tipo de sus k vecinos (por ejemplo, mediante un esquema de votación por mayoría). En la Figura 2.5 se muestra un ejemplo de clasificación mediante este método.

Otro grupo importante es el de las técnicas basadas en la densidad relativa en lugar de las distancias. Estas técnicas estiman la densidad de los vecindarios de forma que un dato que se encuentre en una vecindad con baja densidad será anómalo, mientras que un dato que se encuentre en una vecindad con alta densidad se considerará normal. Un ejemplo perteneciente a este grupo es la técnica *Local Outlier Factor* (LOF) [31].

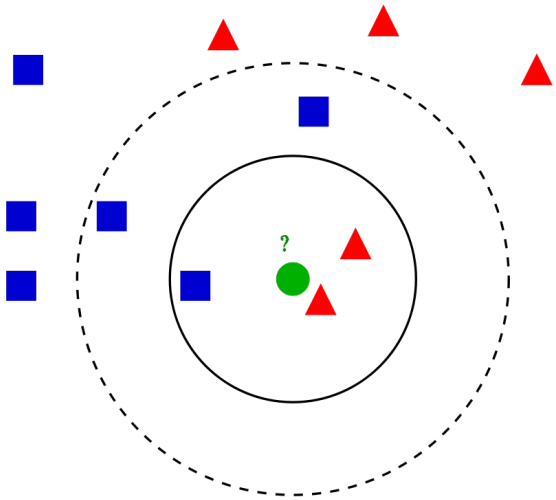

Figura 2.5: Ejemplo del algoritmo k-NN. El dato que se desea clasificar es el círculo verde. Para $k = 3$ es clasificado como clase triángulo ya que es la clase mayoritaria dentro del círculo que contiene a sus 3 vecinos más próximos. Con $k = 5$ en cambio es clasificado como clase cuadrado.

Agrupamiento o clustering

El objetivo principal de estos métodos es agrupar datos similares en grupos o *clusters* como explicamos anteriormente. El aprendizaje suele llevarse a cabo de forma no supervisada y sus técnicas se pueden clasificar en tres grupos.

- Técnicas que asumen que los datos normales pertenecen a *clusters* y los anómalos no. En base a este supuesto aplican métodos tradicionales de *clustering* al conjunto de datos y declaran cualquier dato que no pertenezca a ningún *cluster* como anómalo.
- Técnicas que asumen que los datos normales se encuentran cerca del centroide de sus *clusters* más próximos, mientras que las anomalías se situarán en zonas alejadas. Tras calcular los *clusters*, el *score* de un dato será por tanto su distancia al centroide del *cluster* más cercano. Se establece una distancia umbral para realizar la clasificación de nuevos datos.
- Técnicas que asumen que los datos normales pertenecen a grupos grandes y densos, mientras que las anomalías pertenecen a grupos pequeños o dispersos. En este caso la clasificación se realizará en base a un umbral de tamaño o densidad y no de distancias.

Principales ventajas de las técnicas basadas en distancia

- Pueden operar en modo no supervisado
- Se pueden adaptar a conjuntos de datos complejos simplemente modificando el algoritmo de agrupamiento utilizado.
- Las predicciones suelen ser rápidas sobre conjuntos de datos sencillos ya que la cantidad de agrupaciones con las que se debe comparar el nuevo dato es una pequeña constante establecida por el usuario.

Principales desventajas de las técnicas basadas en distancia

- Su rendimiento depende en gran medida de la efectividad del algoritmo de agrupación utilizado.
- Ciertos algoritmos de *clustering* asignan todos los datos a algún *cluster*. Esto puede provocar que las anomalías formen un grupo de tamaño considerable, lo que choca con el funcionamiento de las técnicas que trabajan bajo la asunción de que las anomalías no pertenecen a ningún grupo.
- Algunas técnicas por tanto son efectivas solamente cuando las anomalías no forman agrupaciones significativas entre sí.

- La complejidad computacional para agrupar los datos suele ser un cuello de botella.

- *k*-NN no es muy eficiente al trabajar con grandes conjuntos de datos o altas dimensionalidades debido a las operaciones de búsqueda de vecinos que utiliza.

2.3.3 Métodos basados en reconstrucción

Los métodos basados en reconstrucción se utilizan a menudo en problemas de regresión y clasificación. Estos métodos pueden modelar de forma autónoma el conjunto de datos de entrenamiento, y predecir un nuevo dato calculando su error de reconstrucción. Este error de reconstrucción se define como la distancia entre el valor proporcionado por el modelo y el valor objetivo, sirviendo así de *score*. Los métodos basados en reconstrucción se pueden clasificar a su vez en dos grupos, los basados en redes neuronales y los basados en subespacios.

El comportamiento de las redes neuronales se inspira en su homólogo biológico y los métodos que las utilizan suelen trabajar en dos pasos. Primero, una red neuronal es entrenada con el conjunto normal de datos. En segundo lugar, cada dato de prueba se trata como un vector de entrada en la red neuronal. La red a partir de este vector de entrada calculará su salida correspondiente, la cual valdrá para clasificar al dato como normal o anómalo. En base a estos conceptos se han desarrollado una gran cantidad de técnicas y variantes. En detección de anomalías es habitual el uso de las RNN o *Replicator Neural Networks* [32]. Esta red se caracteriza por ser multicapa y disponer del mismo número de neuronas en las capas de entrada y salida. Esta cantidad de neuronas debe coincidir con el número de variables o atributos que componen un dato. Entre las capas de entrada y salida se dispone de tres capas ocultas con un número inferior de neuronas, en la Figura 2.6 se puede observar un ejemplo de esta arquitectura. En estas capas de menor dimensionalidad se comprimirá el conjunto de datos normal durante la fase de entrenamiento, obteniéndose una red ajustada a los datos normales. La fase de *test* consiste en introducir un dato a la red y medir el error de reconstrucción que presenta tras su paso por las cinco capas. Este error se podrá utilizar como *score* de los nuevos datos y nos permitirá clasificarlos como anomalías si presentan errores altos.

Los métodos basados en subespacios por otra parte, se basan en proyectar los atributos independientes y diferenciadores de los datos en subespacios de menor dimensionalidad. De esta forma se pueden observar las diferencias entre datos normales y anómalos de una manera más clara. Comparando los valores de proyección de las muestras a clasificar frente a las utilizadas durante el entrenamiento se puede determinar la clase de los datos. Si su estructura es semejante a la de las muestras el valor de correlación será bajo, y por tanto se clasificarán como normales, en cambio si el valor de proyección es alto se clasificarán como anomalías.

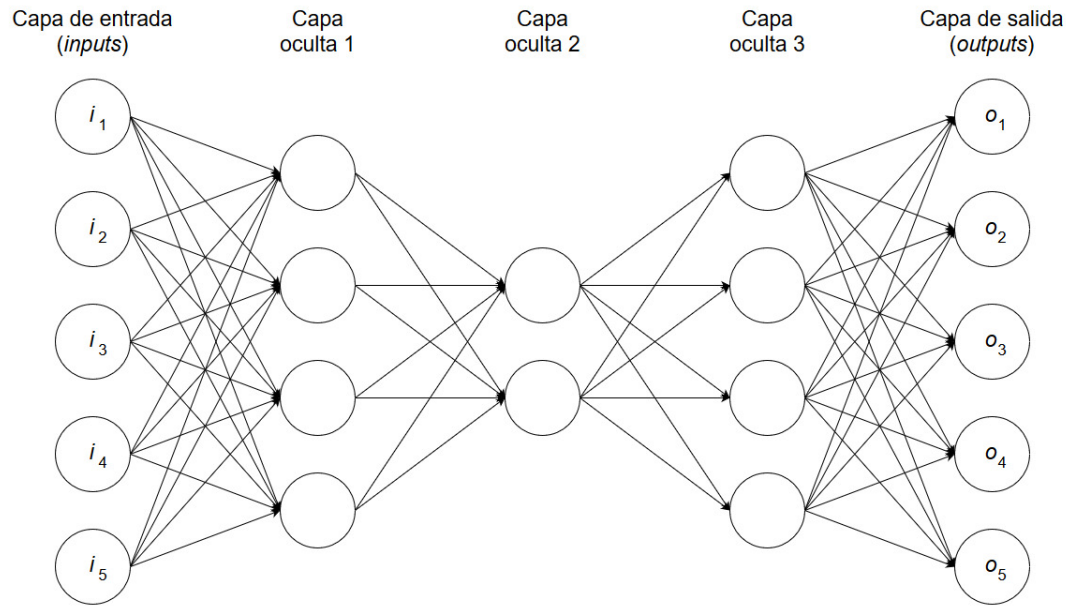

Figura 2.6: Ejemplo de arquitectura en cinco capas de una red RNN.

Una de las técnicas principales en esta categoría es el Análisis de Componentes Principales o PCA [33].

Principales ventajas de las técnicas basadas en reconstrucción

- Pueden entrenarse de forma supervisada y no supervisada.
- No precisan información *a priori* sobre los datos.
- En el caso de las redes neuronales solo es necesario conservar los pesos de las neuronas, lo que hace del modelo de predicción un elemento poco pesado y que además no expone los datos usados durante el entrenamiento.

Principales desventajas de las técnicas basadas en reconstrucción

- En las redes neuronales el tiempo invertido en la fase de aprendizaje es elevado.
- Las técnicas PCA habitúan ser costosas computacionalmente y no se pueden aplicar en cualquier problema de detección de anomalías debido a que el conjunto de datos normales y anómalos deben ser separables en el espacio reducido dónde se proyectan.

2.3.4 Métodos basados en el dominio

Los métodos basados en el dominio generan un límite que separa las clases en el espacio basándose en la estructura de los datos de entrenamiento. Estos clasificadores se vuelven insensibles al tamaño y a la densidad de cada clase debido a que la clasificación de nuevos datos se determina únicamente por su ubicación respecto al límite. Para llevar a cabo la separación se aproxima el hiperplano (u otra estructura paramétrica) que separe de forma óptima a los puntos de una clase de la de otra, aunque es habitual trabajar con problemas de más de dos clases combinando varios hiperplanos. Por ello, estas técnicas necesitan datos de ambas clases para entrenarse (normales y anomalías), lo cual es difícil de conseguir en problemas donde casi no hay muestras de anomalías. Una de las técnicas más comunes es la Máquina de Soporte Vectorial o *Support Vector Machine* (SVM) [34] que busca que este hiperplano maximice su distancia (margen) respecto a los puntos del espacio que estén más cerca de él mismo. Los ejemplos más cercanos al hiperplano son los llamados vectores de soporte, y permiten prescindir del resto de ejemplos del conjunto de entrenamiento. En la Figura 2.7 se puede observar un ejemplo de división en dos clases en un espacio 2-D mediante SVM. Es importante destacar que existen variantes útiles en detección de anomalías como es el caso de *One-class SVM* (OSVM).

Debido a la alta sensibilidad a la hora de dividir el espacio que este planteamiento conlleva, es necesario compensar la existencia de datos que pueden estar mal clasificados en las zonas de separación de las clases. Si un punto ruidoso o anómalo se utiliza en el entrenamiento el sistema se puede sobreajustar, lo que empeorará su futuro funcionamiento. Por ello, estos límites entre clases se suelen suavizar. Una forma habitual de hacerlo es establecer un porcentaje de datos normales que pueden quedarse fuera del límite normal a cambio de una separación de las clases menos forzosa. Existen también variantes que utilizan hiperesferas para intentar contener al conjunto de datos normales dejando en el exterior a los considerados anómalos, como es el caso del método *Support Vector Data Description* (SVDD) [35].

Principales ventajas de las técnicas basadas en el dominio

- El entrenamiento es relativamente fácil y la clasificación de nuevos datos es rápida ya que solo se comparan con el modelo ya creado.
- No existen óptimos locales como en las redes neuronales.
- El compromiso entre la complejidad del clasificador y el error puede ser controlado explícitamente.
- Para modelar la separación espacial entre las clases y poder clasificar nuevos datos es

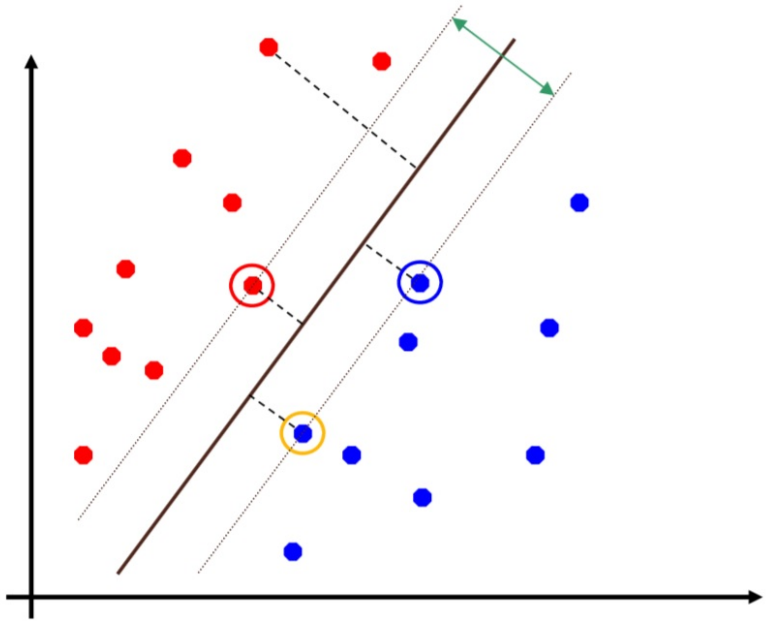

Figura 2.7: Ejemplo de hiperplano que separa dos clases mediante tres puntos soporte en un espacio bidimensional mediante la técnica SVM.

suficiente con almacenar los puntos soporte y no todo el conjunto de datos de entrenamiento.

- Son capaces de distinguir entre más de dos clases, lo que en detección de anomalías puede ser útil en caso de poseer diferentes tipos de anomalías.

Principales desventajas de las técnicas basadas en el dominio

- Ajustar los parámetros de los SVM y decidir la función *kernel* que ajustará regiones complejas no es una tarea trivial, es un proceso que se realiza generalmente mediante prueba y error.
- Son sensibles al ruido, lo que puede provocar sobreajustes.

2.3.5 Métodos basados en la teoría de la información

La entropía en teoría de la información es una medida de incertidumbre asociada a una variable. Para medir el grado de desorden de un conjunto de datos mediante este concepto se utiliza lo que se conoce como función de entropía. Los métodos basados en la teoría de la información procesan el contenido de los conjuntos de datos mediante medidas como la entropía o la entropía relativa. Asumen que los datos anómalos alteran significativamente la información contenida en el conjunto normal. Las métricas se calculan usando el conjunto de

datos al completo, incluyendo al punto a clasificar, y sin él. Los resultados obtenidos se comparan con los producidos por únicamente el conjunto normal, de forma que si se encuentran grandes diferencias estaremos ante una anomalía. Un método de este tipo es HOT SAX [36], útil en problemas de series temporales continuas.

Principales ventajas de las técnicas basadas en la teoría de la información

- Pueden entrenarse de forma no supervisada.
- No asumen información *a priori* sobre la distribución de los datos.

Principales desventajas de las técnicas basadas en la teoría de la información

- La confianza en las predicciones de estas técnicas es altamente dependiente de la métrica utilizada.
- Si el número de anomalías es muy bajo, ciertas métricas son incapaces de detectarlas debido a su bajo impacto.
- Debido a que la clasificación de un dato se realiza midiendo su influencia respecto al conjunto total de datos, no es trivial asociar puntuaciones y establecer umbrales que dividan a la clase normal de la anómala.

Capítulo 3

Fundamentos teóricos

Como hemos visto, existen una gran variedad de enfoques a la hora de resolver problemas *one-class* para la detección de anomalías. Debido a que desarrollar un algoritmo basándose en la perspectiva geométrica del problema es una forma elegante e intuitiva de entender sus soluciones, y a que estas técnicas pueden resolver eficientemente una buena parte de los problemas de clasificación, hemos decidido apostar por este tipo de soluciones, concretamente, la proporcionada por el cierre convexo (*convex hull*). En este capítulo se explican los fundamentos teóricos de las técnicas basadas en cierre convexo, así como el método de la literatura que se ha tomado como base para desarrollar el algoritmo propuesto en este proyecto.

3.1 Cierre convexo

El cierre convexo o *convex hull* (CH) es el poliedro convexo más pequeño que contiene a un conjunto de puntos dados. En la Figura 3.1 se puede observar un ejemplo de cálculo del cierre convexo en dos dimensiones, en este caso el cierre convexo es un polígono. La utilidad de esta estructura en problemas *one-class* es calcular el poliedro que engloba al conjunto de datos normales, de forma que clasificar un nuevo dato consista en comprobar si se sitúa dentro del cierre convexo (dato normal) o en su exterior (anomalía).

3.2 Algoritmo base

A continuación, se van a explicar los fundamentos del algoritmo sobre el que se va a basar en parte este proyecto, desarrollado por Diego Fernández Francos et al. [37] basándose a su vez en el trabajo llevado a cabo por Casale et al. [38]. El algoritmo en cuestión recibe el nombre de *Approximate convex polytope decision ensemble* (APE).

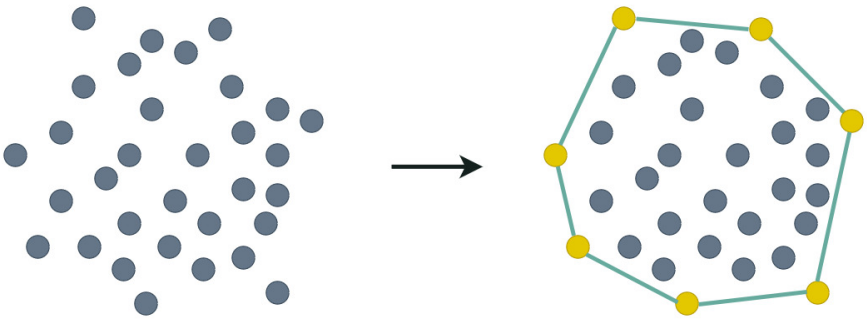

Figura 3.1: A la izquierda se muestran los datos sobre los que se desea calcular el cierre convexo. A la derecha se muestra el polígono obtenido como cierre convexo.

El cierre convexo puede llegar a representar una estructura incluso aún más general que las hiperesferas utilizadas por métodos como *Support Vector Data Description* (SVDD) [35]. Sin embargo, su uso en aplicaciones reales está limitado por el hecho de que su computación en espacios de alta dimensionalidad es extremadamente costosa. Aunque se han propuesto soluciones para contrarrestar esta limitación, existe un gran número de problemas de reconocimiento de patrones en los que su uso no es adecuado, especialmente cuando los recursos computacionales no son especialmente altos. Además, comprobar si un punto se encuentra dentro o fuera del cierre convexo en espacios de alta dimensión sigue siendo un problema de difícil solución. Por ello, varios algoritmos optan por realizar proyecciones aleatorias de los datos a clasificar para reducir la dimensionalidad del espacio de entrada y operar así en entornos más sencillos.

Esta técnica de proyecciones aleatorias se basa en la idea de que los espacios de datos de alta dimensión se pueden proyectar en un espacio dimensional inferior sin perder significativamente la estructura de los datos. Johnson y Linderstrauss [39] han demostrado esta preservación de la estructura de datos, y además, que si los datos se proyectan en un espacio con dimensionalidad proporcional al logaritmo de la cardinalidad del conjunto de datos, dicha preservación se garantiza con una alta probabilidad. La capacidad de reducir la dimensionalidad del problema sin un esfuerzo computacional grande y la conservación de la estructura de los datos, permite crear técnicas de aprendizaje muy simples y poderosas. Utilizando esta técnica el proceso de aprendizaje basado en cierre convexo consiste en dos etapas principales: proyectar los datos en un subespacio aleatorio y calcular el cierre convexo en ese subespacio.

3.2.1 Fase de aprendizaje

Dado un conjunto de datos de entrenamiento de dimensión D, se estima un número de proyecciones τ para aproximarlo en un subespacio bidimensional. Se generan por tanto τ matri-

ces aleatorias correspondientes a las proyecciones. El conjunto de entrenamiento se proyecta en el espacio generado por cada una de estas matrices de proyección. Finalmente, se calculan los vértices del cierre convexo para cada proyección, siendo este el objetivo del entrenamiento. Construir un cierre convexo en espacios bidimensionales y comprobar si un punto cae en su interior son tareas comunes en computación geométrica, por lo que ya se dispone de soluciones muy eficientes (p. ej. algoritmo *QuickHull* [40]).

3.2.2 Fase de test

A la hora de predecir la clase a la que pertenecen nuevos datos, el procedimiento será el siguiente. Se calculan las τ proyecciones del nuevo dato a clasificar. Para cada proyección, y dado el conjunto de vértices del cierre convexo de ese espacio dimensionalmente menor que el original, es posible comprobar fácilmente si el punto se encuentra en el interior del polígono. Un punto se clasificará como normal si se encuentra en el interior de todos los cierres convexos proyectados. En caso de caer fuera del cierre en una o más proyecciones, el punto se considerará anómalo. Existe una gran cantidad de algoritmos para realizar esta comprobación en espacios 2-D como es el caso del algoritmo de *ray casting* [41]. En la Figura 3.2, supongamos que el cuerpo gris es el cierre convexo calculado para el conjunto de entrenamiento, mientras que el punto naranja es el nuevo dato a clasificar. En dos de las tres proyecciones el punto se encuentra fuera del cierre y en una dentro, por lo tanto el punto se clasificará como anomalía.

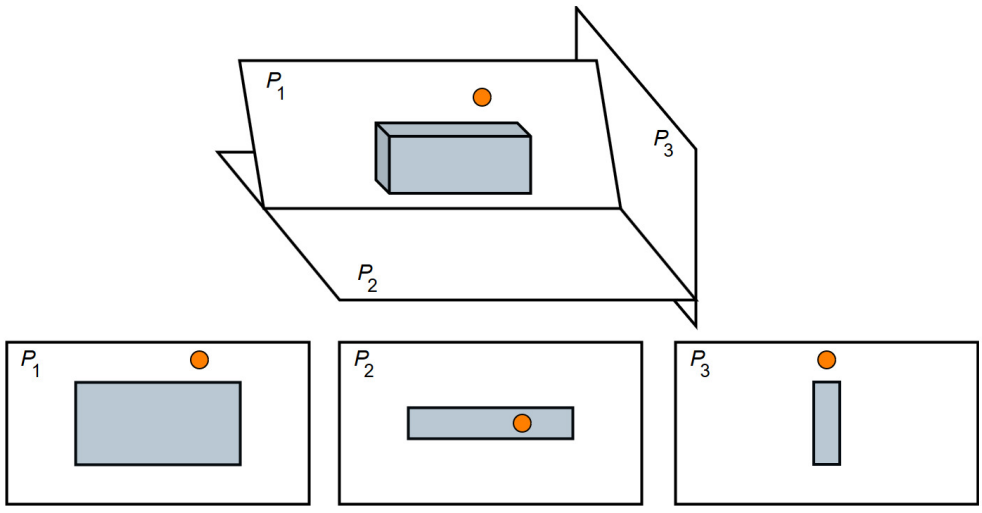

Figura 3.2: Tres proyecciones de un cuerpo tridimensional y un punto sobre subespacios bidimensionales.

3.2.3 Aspectos configurables del algoritmo base

Existen varios aspectos del algoritmo que pueden modificarse para adaptarlo mejor al problema sobre el que se aplica de cara a obtener mejores resultados. A continuación se van a describir los más importantes.

Factor de expansión/contracción

Para combatir el efecto de sobreajuste del cierre convexo con los datos de entrenamiento, Casale et al. proponen el uso de versiones reducidas y expandidas del cierre inicial, al que llaman cierre convexo extendido o *extended convex hull* (ECH). El tamaño de estas versiones expandidas vendrá dado por un parámetro λ, de forma que los nuevos datos se clasificarán en base a este nuevo cierre y no al original. La creación del cierre extendido es impracticable computacionalmente en espacios de alta dimensión, por tanto se lleva a cabo en los τ subespacios bidimensionales generados por las proyecciones. De esta forma, cada vértice de cada t-convex hull computado en la fase de entrenamiento, es expandido o contraído en un espacio bidimensional en función de λ y respecto al punto central c del cierre. Este valor λ expandirá los vértices cuando tome valores mayores a cero, y los contraerá cuando sea un valor negativo. En caso de ser igual a cero el cierre original y el extendido serán el mismo.

Centro del polígono

La versión inicial del algoritmo plantea la existencia de un único centro c en el espacio original del problema, de forma que el centro de un subespacio viene dado por la proyección de este punto c en dicho subespacio, representado como $\bar{c}$. En este planteamiento, el CH expandido tiene en cuenta la distancia de cada vértice al centro del CH en el espacio original para expandir el CH en los subespacios. Esto puede provocar la generación de cierres no convexos, lo que choca con el comportamiento deseado del algoritmo. Para contrarrestar este problema, la solución hallada fue calcular un centro en cada proyección mediante la media de todos los puntos de ese subespacio. Por ello y por la flexibilidad que otorga cada tipo de centro, se han llevado a cabo nuevas aproximaciones para la versión expandida del algoritmo:

- Utilizar la media de todos los puntos en el espacio proyectado

$$\bar{c} = \frac{1}{|\bar{S}|} \sum_i \bar{x_i}, \forall \bar{x_i} \in \bar{S}$$

  donde $\bar{S} \in R^2$ representa la proyección en dos dimensiones del conjunto inicial de puntos $S \in R^d$.

- Utilizar la media de los CH vértices $\bar{v_i}$ en el espacio proyectado $\bar{S}$

$$\bar{c} = \frac{1}{\left|CH(\bar{S})\right|} \sum_i \bar{v_i}, \forall \bar{v_i} \in CH(\bar{S})$$

- Utilizar el centroide o baricentro de los puntos proyectados.

Uso distribuido

En la actualidad, el uso de grandes bases de datos se realiza de forma distribuida en varias máquinas en ubicaciones diferentes. Modelar un sistema de predicción como un clasificador *one-class* en este escenario es una tarea compleja, hasta el punto de que la mayoría de los algoritmos *one-class* clásicos son incapaces de resolver esta situación. Almacenar el total del conjunto de datos en una única ubicación provoca que el envío de datos a otros puntos sea costoso e incluso prohibido si se trata con información privada, lo que produce que las tareas de clasificación se acaben realizando únicamente de manera local. Esto también incrementa el tiempo de entrenamiento necesario para obtener el modelo debido a su ejecución en un único nodo. Por ello, es importante asegurar que algoritmos que trabajaban bien en escenarios clásicos puedan ser modificados para su uso en entornos distribuidos.

Para afrontar este problema, Diego Fernández Francos et al. proponen una versión distribuida del algoritmo APE de Casale et al. La arquitectura de esta versión distribuida [37] consistiría en lo siguiente:

- Una red totalmente conectada de *N* nodos independientes: como la idea es tratar con bases de datos distribuidas, se supone que estos nodos son computadoras (con sus propias bases de datos) ubicadas en diferentes partes del mundo. Son independientes porque no necesitan saber nada sobre los datos gestionados en los demás nodos.

- Un nodo principal: es el responsable de comunicarse con todos los demás nodos para iniciar el proceso de entrenamiento, compartiendo la información necesaria. Puede ser cualquiera de los *N* nodos.

- Un nodo de decisión: es el responsable de comunicarse con todos los demás nodos para compartir la información necesaria para clasificar un nuevo dato, y de tomar la decisión de clasificación final. Cualquiera de los *N* nodos puede llegar a ser el nodo de decisión en algún momento, no es un rol asignado de forma estática.

El funcionamiento del algoritmo mediante esta aproximación es el siguiente:

- El conjunto de matrices de proyecciones aleatorias $\{P_t\}, t = 1...\tau$ debe enviarse a cada nodo de entrenamiento. Se elige un nodo N como nodo principal. El conjunto $\{P_t\}$ se crea en este nodo y se envía a los demás.

- Cada nodo de entrenamiento construye su propio modelo M, compuesto por las τ matrices de proyección y sus respectivos vértices expandidos. Tras este proceso, cada nodo de entrenamiento se convierte en un nodo de prueba con su propio modelo. Por tanto, se dispone de N modelos locales diferentes para clasificar un nuevo punto.

- Tras la fase de entrenamiento, si aparece un nuevo dato x en un nodo $i(i = 1...N)$, debe clasificarse. El nodo i será el nodo de decisión durante el proceso de clasificación del dato x. En lugar de enviarse el dato x directamente a los nodos de prueba, el nodo de decisión envía sus proyecciones $\bar{x_t} = P_t x, t = 1...\tau$, evitando así el intercambio de datos privados a través de la red.

- La decisión local sobre el punto x obtenido en cada nodo se entrega al nodo de decisión para que este decida la clasificación final del dato.

Capítulo 4

Propuesta de un algoritmo adaptativo en tiempo real

En este capítulo se detalla el desarrollo del algoritmo propuesto en este Trabajo de Fin de Grado, que basándose en las técnicas de Casale et al. y Diego Fernández Francos et al., aporta diversas novedades respecto a estos métodos. La característica principal de este método será su *capacidad de adaptación en tiempo real* ante la llegada de nuevos datos que modifiquen la forma del clasificador. Además, se llevarán a cabo modificaciones en su funcionamiento que *mejorarán su precisión.*

Antes de comenzar, se van a esclarecer las diferencias entre dos conceptos importantes para el desarrollo. Estos conceptos son los conjuntos de datos estacionarios y no estacionarios. Cuando nos enfrentamos a un problema de clasificación *one-class*, es importante analizar la naturaleza del problema para conocer con qué tipo de datos vamos a trabajar. Cuando nos referimos a un conjunto de datos como estacionarios, hablamos de un grupo de puntos que mantienen características fijas a lo largo del tiempo. Esto se da en problemas de naturaleza invariante en los que los datos a modelar presentan distribuciones o formas similares en todo momento. Un ejemplo de problemas de este tipo son los de naturaleza médica, como la detección de anomalías en mamografías [42]. En este tipo de problemas, los datos normales presentan patrones similares en todo momento ya que el estado normal del paciente cuando no presenta anomalías apenas va a variar dentro de unos márgenes conocidos.

Por otra parte, los conjuntos de datos no estacionarios se dan en problemas en los que las propiedades que caracterizan a los datos normales pueden variar con el paso del tiempo. Esto puede producir que un dato se considere normal en un instante t_1, pero que tras la modificación de las particularidades del conjunto de datos normales, se pase a considerar como una anomalía en un instante posterior t_2. Un ejemplo de este tipo de problemas, son los sistemas

que modelan los valores normales de la bolsa en busca de fluctuaciones bruscas. Las propiedades de la clase normal en este tipo de problemas puede variar ligeramente con el tiempo, por lo que un sistema tradicional que construya un clasificador en base a un conjunto de datos de un intervalo de tiempo concreto fallará si la bolsa se asienta en valores más altos o más bajos (p. ej. técnica no supervisada para la detección de anomalías en el mercado de valores de Catar [43]).

4.1 Objetivos para la mejora del algoritmo base

El objetivo principal es desarrollar un modelo de detección de anomalías que pueda aprender partiendo de un conjunto de datos inicial, y que con el paso del tiempo, pueda seguir aprendiendo de forma dinámica. El aprendizaje inicial se llevará a cabo mediante un conjunto de datos normales, a través de los cuales se modelará a la clase normal. De esta forma, el proceso de clasificar un nuevo dato como normal o anómalo será tan sencillo como comprobar si se encuentra dentro de los límites de la clase normal, o en su exterior.

Debido a que este método está pensado para ser utilizado en tareas de monitorización, los nuevos datos a clasificar que llegarán al modelo lo harán de uno en uno, por lo que se procesarán de forma individual como si de una cola se tratase. La idea es utilizar estos nuevos datos no solo como valores a clasificar, sino como una herramienta para mejorar constantemente la precisión de los límites del modelo. Para ello, el modelo ante la llegada de un nuevo dato, además de proceder a su clasificación, estudiará sus características. De esta manera, si varios datos se clasifican como anomalías pero se encuentran de forma muy cercana a los límites de los datos normales, y su frecuencia de aparición es alta, será conveniente reajustar la forma de la clase normal para abarcar a estos datos. Así, el modelo seguirá actualizándose tras el entrenamiento, y por tanto, aprendiendo a lo largo del tiempo.

Es preciso comentar que esta característica, que permite al modelo seguir aprendiendo tras el entrenamiento, no se encuentra en los modelos de referencia (Casale y Fernández et. al.). Dichos métodos no permiten este planteamiento al trabajar en modo *batch* (los reajustes del modelo no se realizan tras cada dato procesado). Mediante esta nueva característica, el modelo puede aprender de forma dinámica con la llegada de nuevos datos evolucionando constantemente. El modelo es capaz así de seguir aprendiendo con nuevos datos sin tener que empezar el entrenamiento desde cero ni almacenar los datos previos para entrenar de nuevo con ellos.

Otro objetivo planteado en este proyecto es mejorar los modelos de referencia en el sen-

tido de que puedan crear regiones de decisión más ajustadas a los datos mediante la unión de diversos cierres convexos. De esta manera se puede evitar la limitación de la convexidad.

Con estos objetivos, se ha desarrollado un nuevo algoritmo de la forma que se describe a continuación.

4.2 Primera versión del algoritmo: adición de nuevos vértices

Tomando como base los algoritmos de Casale et al. y Diego Fernández Francos et al., la idea inicial es construir un método que trabajando sobre conjuntos de datos estacionarios pueda obtener una mayor precisión que los métodos base. Para ello, los límites del CH inicial se irán reajustando durante la fase de entrenamiento, durante un periodo de tiempo posterior a su formación a partir del conjunto de partida. En este periodo de tiempo, los vértices podrán desplazarse haciendo crecer el cierre, e incluso incorporar nuevos puntos al mismo.

4.2.1 Nuevo valor de expansión/contracción del cierre convexo (lambda)

En algunos casos pueden existir datos normales que se clasifican erróneamente durante el entrenamiento como falsos positivos debido a que se encuentran situados en los límites del cierre. Esto es debido a que no han aparecido durante la fase de formación del CH inicial. En la Figura 4.1 se puede observar un ejemplo gráfico. Para que nuestro algoritmo sea capaz de combatir este comportamiento y se ajuste con mayor precisión a la forma de los datos, hemos decidido que el factor de expansión/contracción λ sea siempre positivo. Este valor es el que permite generar los CHs expandidos o escalados (SCH), de forma que cuenten en todo momento con un margen externo que suavice la clasificación. En la Figura 4.2 se puede observar el cierre expandido con el que se realizarán las clasificaciones.

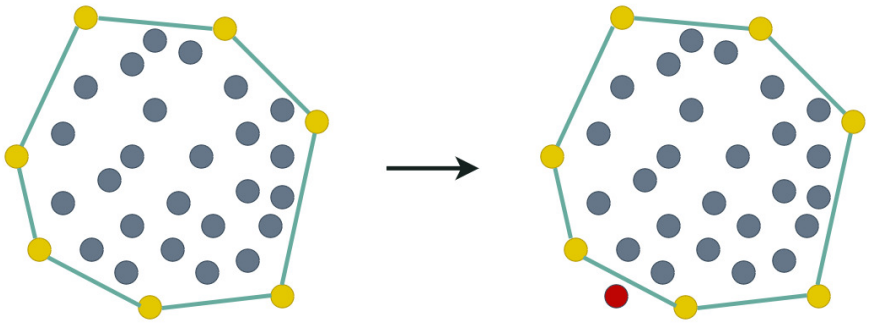

Figura 4.1: El dato en color rojo es un valor normal pero no se usó para la fase de formación del CH inicial por lo que será clasificado como una anomalía.

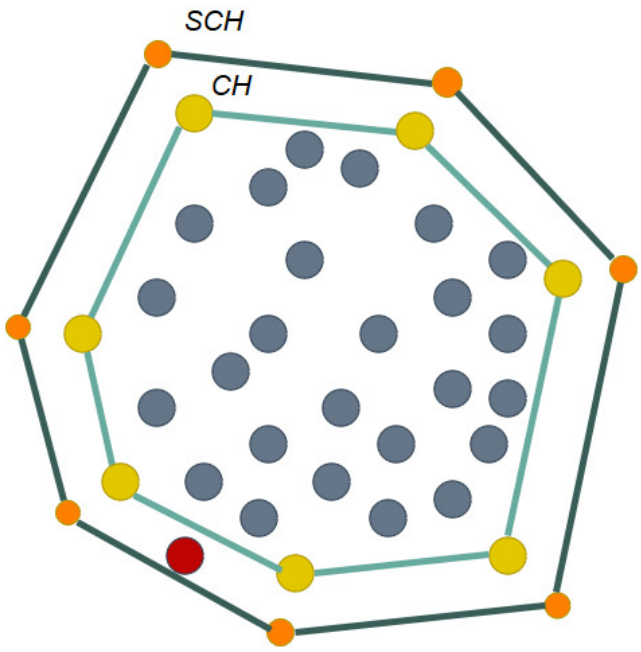

Figura 4.2: Cierre escalado (SCH) a partir de un cierre convexo (CH). El punto rojo representa un dato que gracias al uso del margen será clasificado correctamente.

4.2.2 Reajuste del cierre convexo

Para reajustar el cierre convexo inicial, se tendrán en cuenta las caídas de nuevos datos en los márgenes del SCH. Como hemos visto, si no se dispone de un margen que suavice los límites entre las clases, algunos datos serían clasificados como anomalías por situarse estrictamente fuera del CH inicial. Esta es una de las desventajas de trabajar con técnicas basadas en la geometría de los datos, la posible existencia de ruido en la información y el sobreajuste de los límites que separan las clases.

Por tanto, para mejorar las separaciones entre clases, y así modelar de forma más precisa el conjunto normal, si un número considerable de datos cae de forma sistemática y concentrada entre los límites del CH y el SCH, es decir, en el margen, el comportamiento del algoritmo será ampliar los límites del CH hacia esa zona en la que están apareciendo datos normales no recogidos por el cierre inicial. Para implementar esta funcionalidad, el proceso de entrenamiento se divide en dos fases: una primera fase que con una parte de los datos normales elabora los cierres convexos de cada proyección. Y una segunda, destinada al reajuste del cierre, explicada a continuación:

- Cuando un nuevo dato cae en el margen (espacio existente entre los límites del CH y del SCH), se determina cuál de los vértices del CH es el más cercano mediante su distancia

euclídea.

- La posición de este dato en el espacio se almacena en una lista asociada a este vértice más cercano. Cada vértice por tanto dispondrá de una lista de datos que han caído cerca suya, de modo que cuando este número de datos supere un mínimo establecido, el cierre convexo se expandirá a través de este vértice. Mediante estas expansiones, las regiones serán capaces de crecer para ajustarse con mayor precisión a los datos normales.

- La expansión del vértice consistirá en crear un nuevo vértice en la posición resultante de la media de los datos (centroide) almacenados en su lista.

- Agregando a la lista de vértices que forman el CH este nuevo vértice, se recalcula con ellos el cierre convexo, ya que puede que por la incorporación del nuevo dato se pierda la convexidad. De esta forma, el cierre ha crecido hacia una zona que no abarcaba mediante el CH inicial pero que debería considerarse como normal ya que están apareciendo nuevos datos por esa zona del espacio.

- Una vez creado el nuevo cierre convexo, si el vértice que provocó su expansión sigue formando parte de la lista de vértices del nuevo CH, la lista de datos más cercanos asociada a dicho vértice se reinicia para dar cabida a posibles futuras expansiones. También se creará una lista asociada al nuevo vértice en la que se almacenarán de la misma manera los datos que caigan en el margen cerca de él.

- Finalmente el SCH se modifica en función del nuevo CH.

En la Figura 4.3 se puede observar un ejemplo de reajuste del CH debido a la caída de tres puntos en el margen (puntos rojos), el vértice v_1 es el más cercano a los tres puntos por lo que será el encargado de almacenarlos en su lista. El número mínimo de puntos necesarios para reajustar el CH en este ejemplo es de tres. Por ello, tras la caída del tercer punto, se genera un nuevo vértice v_3 situado en el centroide de los tres puntos.

Si deseásemos añadir un factor temporal a la expansión del CH, podríamos hacerlo a través del tamaño de la lista asociada a cada vértice, tratándola como una ventana temporal. De esta forma, cada vez que le llega un nuevo dato al sistema, saldría de la lista el dato menos reciente (trabajaría como una cola). Así, podemos cercionarnos de que los puntos que han provocado la expansión el CH han aparecido próximos en tiempo, ya que en caso de encontrarse muy separados, los datos más antiguos desaparecerán de la lista y el cierre no llegará a expandirse. El usuario prodrá por tanto adecuar el tamaño de esta lista en función de las características del problema.

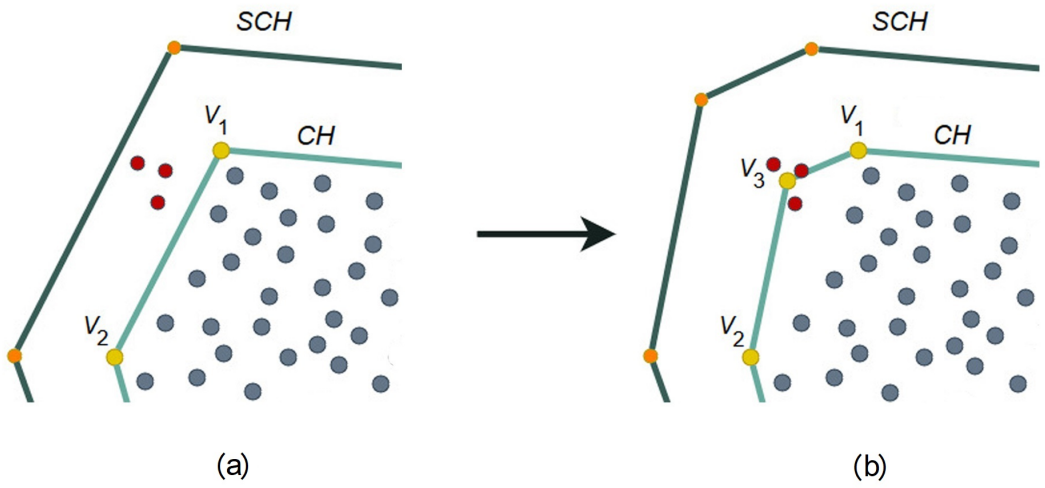

Figura 4.3: Reajuste de un CH por la caída de puntos en su margen. La subfigura (a) contiene el cierre convexo en el estado actual, mientras que la (b) sería el nuevo cierre convexo (con un nuevo vértice V_3) modificando el cierre anterior sobre los datos de referencia del vértice V_1.

4.2.3 Fase de clasificación

La fase de clasificación es similar a la propuesta por las técnicas base. Ante la llegada de un nuevo dato a clasificar, el proceso seguido es el siguiente:

- Se proyecta el dato en cada subespacio bidimensional mediante las τ matrices de proyección.
- Se comprueba en cada subespacio si el dato proyectado está dentro de la región expandida SCH. De no ser así en alguna de las τ proyecciones, la clasificación final del dato será de anomalía. Si en todas las proyecciones se ha proyectado en el interior de los cierres expandidos, será clasificado como un dato normal.

4.2.4 Análisis teórico del modelo

Como se verá más adelante, el algoritmo obtiene buenos resultados con conjuntos de datos convexos o sencillos, representables mediante un solo CH. Por contra, su rendimiento se ve reducido al aplicarse sobre conjuntos de datos con formas no convexas, como es el caso de las medias lunas (como en la Figura 4.4), regiones separadas, con forma de reloj de arena o conjuntos de datos más complejos. Esta desventaja se encuentra de forma intrínseca en la

morfología del cierre convexo, ya que un único cierre no es capaz de modelar conjuntos de datos no convexos. Otras técnicas de detección de anomalías como *k*-NN o *One-class* SVM, proporcionan mayor precisión en estos casos.

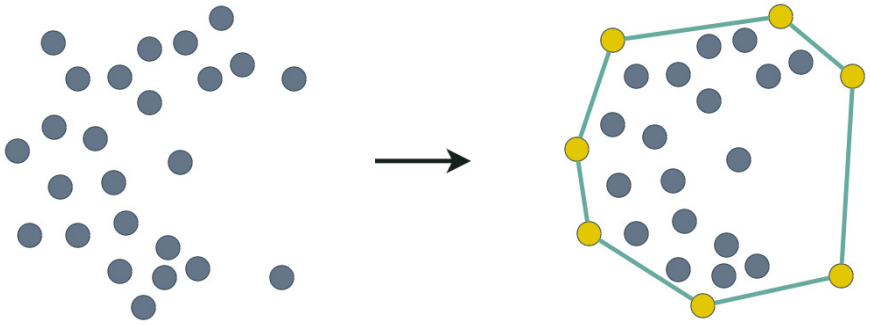

Figura 4.4: Cierre convexo de un conjunto de puntos con morfología de media luna, cuya representación mediante un solo cierre convexo provoca la aparición de falsos negativos, debido a que abarca zonas del espacio más amplias de las deseables.

Debido a este problema, se decidió que una alternativa que mejoraría la captación de anomalías en este tipo de conjuntos, sin perder los beneficios de la técnica original, sería subdividir de forma iterativa los cierres convexos. De esta manera, partiendo de un CH inicial, este se puede subdividir recursivamente en tantos cierres convexos como sean necesarios para aproximar correctamente la forma de los datos normales, y así se pueden representar regiones no convexas como la unión de diversas regiones de tipo convexo. Por tanto, en cada proyección pueden llegar a coexistir varios cierres convexos de forma simultánea, que seguirán reajustándose de forma individual. En la Figura 4.5 se puede observar la subdivisión de un CH en dos para modelar de una forma más precisa al conjunto de datos normales no convexo.

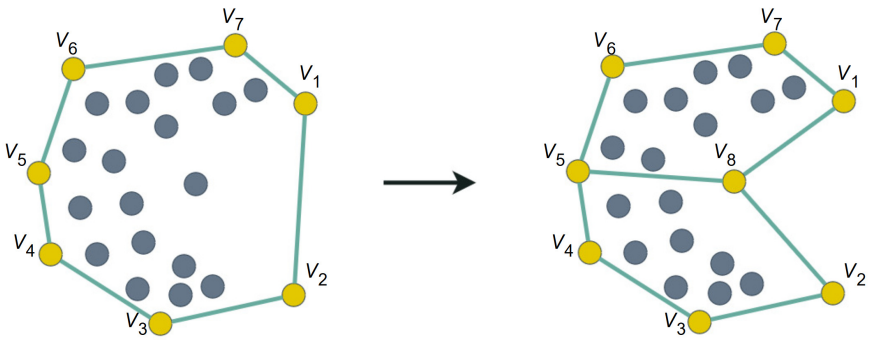

Figura 4.5: Cierre convexo que se ha subdividido en dos cierres convexos para representar con mayor precisión a un conjunto de puntos con morfología de media luna.

Por ello, se decidió mejorar el algoritmo mediante el diseño e implementación de una segunda versión. Esta, mediante múltiples cierres convexos en cada proyección (fruto de un proceso de subdivisión), arrojará mejores resultados cuando se aplique sobre conjuntos de datos no convexos. Esta segunda versión del algoritmo incorpora a la primera versión y añade la capacidad de subdivisión. Debido a esto necesitará de un poder de cómputo más elevado, ya que tratará con un número mayor de cierres, y por tanto, de datos, de forma simultánea. A continuación se detalla el desarrollo de esta segunda versión.

4.3 Segunda versión del algoritmo: subdivisión de regiones

Dado que la subdivisión de los cierres convexos implica más carga computacional al requerir mayor número de comprobaciones a la hora de clasificar un dato, y mayor número de regiones a almacenar en memoria, se debe evitar que realice subdivisiones de forma innecesaria. La decisión de que una región se vaya a subdividir en dos debe tomarse bajo algún criterio establecido. Además, estas subregiones deben construirse de una forma óptima. De la misma manera, cuando una región sea lo suficientemente representativa para el conjunto de datos normales, no deberá seguir subdividiéndose (se congelará), y si una región llega a ser prescindible se eliminará (será podada). A continuación, se va a explicar cómo se ha implementado la capacidad de subdivisión del algoritmo, los elementos que intervienen en ella y los criterios utilizados.

4.3.1 Identificación de zonas no convexas: puntos soporte

La metodología propuesta para subdividir un cierre convexo comienza por establecer una métrica que mida cómo de bien un CH representa al conjunto de datos normales. Un CH está compuesto por vértices (datos) que forman aristas entre ellos. Como se puede observar en la Figura 4.5, cuando un CH envuelve a un conjunto de datos no convexo, regiones vacías en las que no existen datos normales se incluyen erróneamente dentro del CH. Estas regiones son las que la subdivisión debe tratar de expulsar de la clase normal.

Ya que las aristas del CH más próximas a estas regiones vacías no disponen de puntos cercanos (véase arista $V_1 - V_2$ en la Figura 4.5), la solución que se ha propuesto para localizar estas regiones es controlar que cada arista disponga de un punto soporte asociado. El punto soporte de una arista será el dato normal del CH no vértice más próximo al punto medio de la arista. En la Figura 4.6 los vértices V_1 y V_2 forman una arista con punto medio en *c*. Por tanto, el punto soporte de esta arista será el dato que no sea vértice más próximo a *c* situado en el interior del CH, en la imagen el punto $P_{soporte}$. Calculando las distancias a los puntos soporte de todas las aristas (líneas en rojo), se puede observar que las aristas que mejor representan

al conjunto de datos normal poseen puntos soporte más próximos que las que no lo hacen. Es decir, la distancia a sus puntos soporte es mucho menor en comparación con las aristas situadas en regiones vacías. En nuestro ejemplo, la distancia d de la arista que conecta V_1 y V_2 es mucho mayor que la medida en las otras aristas del CH, lo que nos indica que la zona circundante a c es un buen lugar desde el que empezar a subdividir el CH.

Durante el entrenamiento, cada vez que un dato caiga en un CH, las aristas de dicho CH calcularán su distancia hasta el nuevo dato, para que en caso de ser menor que la distancia con su punto soporte actual, este se actualice al nuevo dato. Un punto soporte puede estar compartido por más de una arista al mismo tiempo, esto es común al comienzo del entrenamiento o tras una subdivisión.

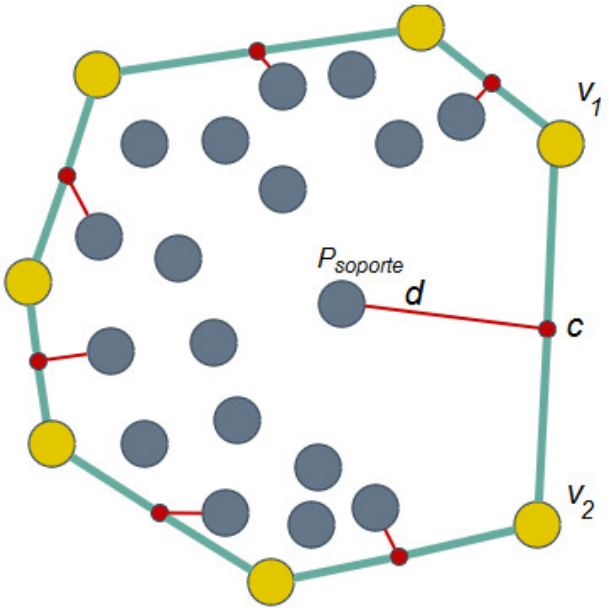

Figura 4.6: Cierre convexo con las distancias a los puntos soporte (líneas en rojo). La arista V_1–V_2 fue la seleccionada para realizar la subdivisión ya que la distancia (d) a su punto soporte ($P_{soporte}$) es la mayor del cierre y supera un umbral establecido para esa proyección.

4.3.2 Proceso de subdivisión

Ahora que mediante las distancias a los puntos soporte disponemos de una métrica que nos permite caracterizar a las aristas, es necesario establecer un criterio que decida cuándo una distancia es suficientemente grande como para subdividir el CH. Ya que el conjunto de datos no va a ser siempre homogéneo, cada proyección puede generar cierres de distintos tamaño. Además, cada problema sobre el que se aplique el algoritmo tendrá un conjunto de datos

totalmente diferente. Debido a esto, la distancia mínima de una arista a su punto soporte para subdividir el CH no puede ser un valor constante definido por el usuario para cada problema y proyección. Para automatizar este proceso se ha decidido utilizar el rango intercuartílico. Esto nos permite, a partir de una serie de distancias, calcular de forma automática el umbral que separa a las distancias normales de las atípicas y extremadamente atípicas.

Rango intercuartílico

En estadística descriptiva, el rango intercuartílico o rango intercuartil (RIC) es una medida de dispersión estadística fruto de la diferencia entre el tercer y el primer cuartil de una distribución. Siendo la mediana la representación del valor de la variable de posición central en un conjunto de datos ordenados, y dado una serie de valores $x_1, x_2, x_3...x_n$ ordenados de forma creciente, el cálculo de los cuartiles se lleva a cabo como sigue:

- Primer cuartil (Q_1): mediana de la primera mitad de valores.
- Segundo cuartil (Q_2): mediana de la serie al completo.
- Tercer cuartil (Q_3): mediana de la segunda mitad de valores.

Conociendo esto, el rango intercuartílico se define como la diferencia entre el tercer cuartil (Q_3) y el primer cuartil (Q_1), es decir:

$$RIC = Q_3 - Q_1$$

En estadística, una observación atípica es la que es numéricamente distante del resto de los datos. A partir de los cuartiles y el RIC podemos clasificar a las distancias como normales o atípicas:

- Valor atípico: un valor q será atípico si:

$$q < Q_1 - 1.5 \cdot RIC,$$

  o

$$q > Q_3 + 1.5 \cdot RIC.$$

- Valor extremadamente atípico: un valor q será extremadamente atípico si:

$$q < Q_1 - 3 \cdot RIC,$$

  o

$$q > Q_3 + 3 \cdot RIC.$$

En nuestro caso, al pretender segmentar las distancias atípicamente grandes de las distancias normales, únicamente utilizaremos los límites superiores. Los inferiores no los necesitamos ya que las distancias atípicamente pequeñas las consideramos también como distancias normales. En la Figura 4.7 se representa un diagrama de cajas que muestra los cuartiles, mediana y valores atípicos de una distribución de datos mediante las fórmulas comentadas. En nuestro problema, nos interesa localizar las distancias que se encuentren por encima de L_s, ya que trabajaremos con el umbral extremadamente atípico.

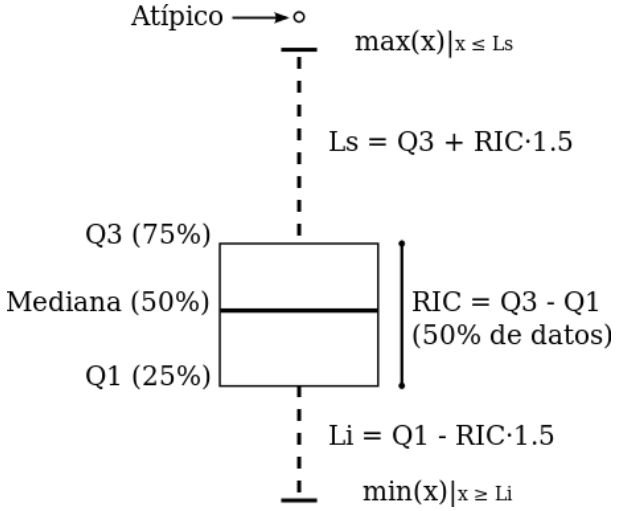

Figura 4.7: Representación mediante un diagrama de cajas de los cuartiles, mediana y valores atípicos.

Selección del vértice pivote

Ahora que disponemos de un criterio apropiado para decidir cuándo subdividir un CH, debemos determinar qué pasos se van a seguir para construir los subcierres convexos a partir del inicial. Nuestro planteamiento para reducir el número de subdivisiones necesarias para modelar la clase de datos normales, consiste en dividir las regiones en dos subregiones de tamaño (área) lo más similar posible.

La idea del procedimiento es encontrar el vértice v_n del cierre que junto al punto central de la arista seleccionada para la división, divide de una forma eficiente el cierre en dos subcierres. La Figura 4.8 ilustra este proceso en el que la arista seleccionada para la división es la formada por v_1 y v_2. Para encontrar v_n, utilizamos el perímetro p (aristas en rojo) que forman las aristas del CH exceptuando la formada por v_1 y v_2. El vértice pivote será el más próximo al punto del perímetro que lo divide a la mitad ($p/2$), nombrado como c. De esta forma, en la Figura 4.8 se puede observar como para el perímetro p formado por los vértices de v_2 a v_1 (sin contar

la arista de v_1 a v_2), el vértice que se encuentra en la posición más próxima al punto medio c es v_6. En la Figura 4.9 se puede observar el resultado final, habiendo tomado el vértice v_6 como pivote para realizar la subdivisión, siendo la línea roja el perímetro calculado y c su punto medio. Hemos utilizado el perímetro del CH ya que conocemos en todo momento las posiciones de los vértices y aunque no proporcione la máxima precisión, otras posibles opciones como calcular el área, en polígonos irregulares como estos y para todas las posibles combinaciones de vértices, suponen procesos computacionalmente más costosos, lo cual se pretende evitar.

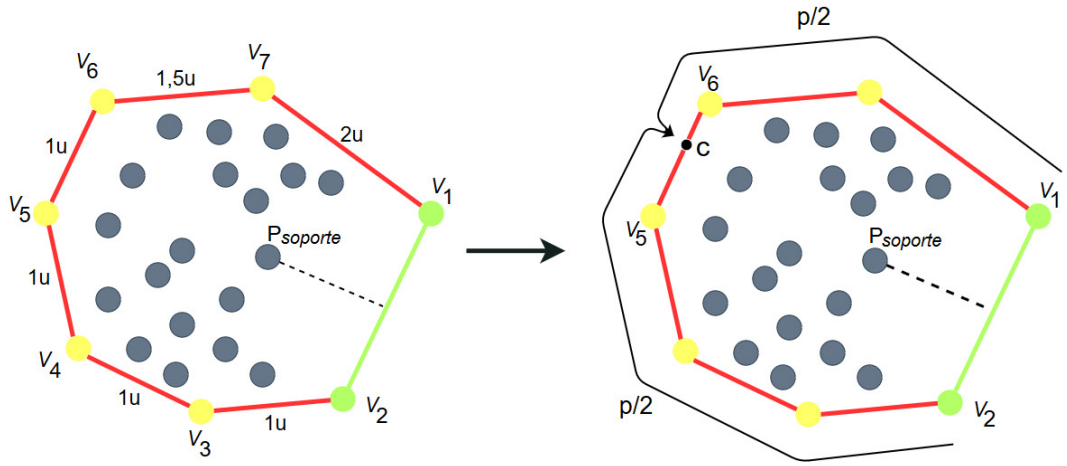

Figura 4.8: En la figura de la izquierda se observa un cierre a punto de ser subdividido. La longitud de cada arista se representa en unidades u, de forma que el perímetro p que forman suma un total de $7.5u$. En la figura de la derecha se muestra el punto c, situado a distancia $p/2$ desde v_1 y v_2. El vértice más cercano a c será el escogido como pivote de la subdivisión.

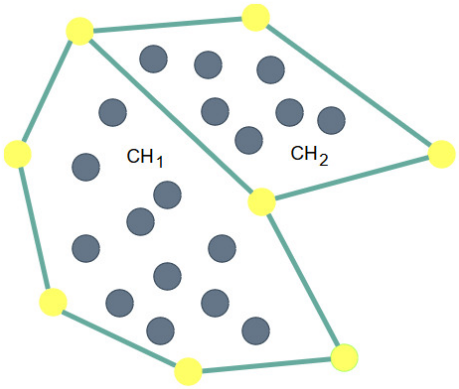

Figura 4.9: Dos cierres convexos creados a partir de una subdivisión.

De esta forma, el proceso de subdivisión de un CH consistiría en:

1. Calcular el umbral extremadamente atípico mediante las distancias a los puntos soporte de todos los cierres convexos actuales en la proyección. Incluir a todos los cierres en el cálculo nos permite relacionarlos. De esta forma podemos detectar cuándo una distancia es mucho más alta o baja que la mayoría, y actuar de la manera que corresponda (subdividiendo el cierre si la distancia es muy alta). Si por contra, calculásemos un umbral para cada CH de la proyección, no estaríamos relacionando los diferentes cierres entre sí, lo que permitiría que se diesen casos de divisiones innecesarias, o que otras que se deberían producir no se lleguen a ejecutar al perder información del entorno. Por ejemplo, un CH muy pequeño en comparación a los demás, que no vale la pena seguir subdividiendo, podría llegar a hacerlo sin esta información del contexto.

2. Localizar la arista con mayor distancia a su punto soporte, siendo esta superior al umbral declarado como atípico en esa proyección.

3. Una vez seleccionada la arista *a*, su punto soporte formará parte de los dos nuevos cierres convexos (véase la Figura 4.8).

4. Se escoge el vértice pivote que divida de la mejor forma posible el CH en dos regiones de tamaños similares mediante el cálculo del perímetro. Este vértice, al igual que el punto soporte, también formará parte de las dos nuevas regiones.

5. Se generan dos nuevos cierres convexos mediante el vértice soporte, el vértice pivote, y los vértices que dejan a cada lado.

6. Las aristas de estos dos nuevos subcierres deberán seleccionar sus puntos soporte reutilizando los puntos soporte del CH anterior a la subdivisión.

Tras la subdivisión de un cierre convexo en dos, las aristas de los nuevos CHs no dispondrán de puntos soporte, por lo que deberán seleccionarlos de entre los datos que caigan en su interior en las próximas iteraciones. Por defecto, los puntos soporte del CH anterior a la subdivisión se reutilizarán como posibles candidatos para las aristas de estos nuevos subcierres. Estos puntos soporte del CH anterior, tras la subdivisión estarán situados en uno o en otro de los subcierres. Por ello, serán candidatos únicamente para las aristas del subcierre dónde se encuentren situados tras la división. Es posible que todos los antiguos puntos soporte se sitúen en uno de los subcierres y que el otro subcierre se inicialice sin puntos soporte. En este caso, las aristas de ese subcierre no dispondrán de puntos soporte hasta que un nuevo dato caiga en su interior. Como norma, un CH no podrá llegar a subdividirse hasta que no disponga de puntos soporte, ya que sin puntos soporte no disponemos de información sobre si es un buen cierre o no.

4.3.3 Proceso de congelación

De igual modo que un CH se debe subdividir si presenta una o varias aristas con distancias que superan el umbral atípico, un CH con todas sus aristas a distancias bajas, y por tanto, bien ajustadas, debe mantenerse hasta el final del proceso de entrenamiento para no dividirlo innecesariamente. A este estado de un CH le hemos llamado congelación. Un CH se congelará si todas sus aristas se encuentran a distancias normales y no ha sufrido cambios en su estructura durante un número de iteraciones establecido. Cuanto mayor sea este margen de iteraciones, los cierres dispondrán de más oportunidades para que los datos caigan en su interior y poder actualizar sus puntos soporte a distancias más pequeñas.

Debido a esto, es conveniente definir un margen suficientemente grande para que transcurra un número de iteraciones apropiado desde que un CH se crea hasta que se pueda volver a subdividir. De no ser así, un CH que representa de una forma muy precisa una subregión del conjunto de datos normales, pero que se acaba de crear a partir de otro CH, es muy probable que posea algunos puntos soporte que estén a una distancia grande y no representen la verdadera calidad que proporciona. Si esta situación se mantiene tras el paso de las iteraciones, el CH se volverá a subdividir de forma precipitada por presentar algún punto soporte que supera la distancia umbral atípica. En cambio, si se le da un margen suficientemente amplio para que actualice todos sus puntos soporte a unos verdaderamente representativos, lo más probable es que no se vuelva a subdividir y acabe por congelarse.

Un CH congelado no se podrá volver a subdividir, pero sí a reajustarse por la caída de datos en su margen. El umbral utilizado como distancia máxima para congelar un CH debe de ser un valor situado en el rango de distancias normales. En nuestro caso, utilizamos el tercer cuartil como umbral. De esta forma, si la distancia de todas las aristas de un CH a sus puntos soporte son inferiores a Q_3 y ha superado el número de iteraciones mínimo establecido, la región se congelará.

4.3.4 Proceso de poda

Debido a que algunos cierres tras varias subdivisiones pueden abarcar regiones totalmente vacías de datos normales, y a que cierres pequeños se pueden llegar a encontrar solapados por los márgenes de otros cierres colindantes, hemos decidido realizar un proceso de poda para deshacernos de ellos y mejorar así la precisión y velocidad del algoritmo. Dicho proceso elimina periódicamente los cierres que no han recibido ningún dato en su interior durante un intervalo de tiempo establecido. De esta forma, nos aseguramos de que un cierre que no recibe datos en su interior y que por tanto no representa a los datos normales, es eliminado.

Cuando un CH es creado, se le asocia una variable contador inicializada a cero que lleva la cuenta de los puntos que caen dentro del cierre. Cuando un dato durante la fase de reajuste cae única y exclusivamente dentro de un cierre convexo, su contador se ve incrementado, indicando que es un cierre en el que caen datos normales. Sin embargo, cuando un dato cae en dos o más cierres al mismo tiempo, ninguno de los cierres ve incrementado su contador. Este comportamiento nos permite que si un cierre está incluido en gran medida en otro cierre mayor, el pequeño acabe por desaparecer.

4.3.5 Análisis teórico del modelo

Tras las modificaciones llevadas a cabo en esta segunda versión del algoritmo, nuestro método es capaz de ajustarse con una precisión mucho mayor a las formas geométricas que presentan los datos normales. Este ajuste más exacto, ya no condicionado por la convexidad de un único cierre, permite la detección de anomalías antes consideradas como datos normales. A estos datos anómalos pasados por alto y clasificados como valores normales, se les conoce como falsos negativos (*FN*). Esta reducción del número de falsos negativos es muy beneficiosa en la naturaleza de nuestro problema, ya que en gran parte de las aplicaciones de la detección de anomalías, la no detección de valores anómalos puede acarrear graves consecuencias. Por ejemplo, un sistema de apoyo para el diagnóstico de cáncer [4], es preferible que se equivoque diagnosticando dicha enfermedad en situaciones en las que existen pocos indicios (falsos positivos), a que por un comportamiento demasiado restrictivo los considere como casos normales, es decir, sanos (falsos negativos).

Además de esto, a diferencia de los métodos propuestos por Casale y Fernández et. al., le hemos dado a nuestra técnica la capacidad de adaptarse en tiempo real. En nuestro método, el reajuste de los cierres convexos mediante su expansión se produce durante la fase de entrenamiento. Tras esta fase, los cierres resultantes se utilizarán para llevar a cabo la clasificación de nuevos datos, pero si se desea, el clasificador se podría seguir reajustando del mismo modo durante esta segunda fase.

Estas mejoras, también implican un mayor coste computacional a la hora de ejecutar el algoritmo, sobre todo durante la fase de entrenamiento. Este incremento del coste se debe al crecimiento del número de cierres convexos en cada proyección. Es necesario mantener además de los puntos que conforman los vértices de los cierres, toda la información auxiliar necesaria para decidir cuándo un cierre debe subdividirse, congelarse, o incluso ser eliminado. Esta información, como medidas de distancia a los puntos soporte desde las aristas, el almacenamiento de los propios puntos soporte, datos que han caído en el margen de los cierres y han de registrarse, o el tiempo que lleva un vértice creado, ralentiza considerablemente el

proceso de entrenamiento del método. El proceso de clasificación, en cambio, no se ve apenas afectado. Esto se debe a que la clasificación de un dato se lleva a cabo mediante su proyección en los diferentes subespacios y comprobando su posición respecto a los cierres, y como se ha visto, este tipo de cálculos geométricos son operaciones muy optimizadas y rápidas.

Debido a que el método está pensado para su uso de forma distribuida, consideramos que el empeoramiento en tiempo de la fase de entrenamiento no es un problema importante, ya que ejecutando esta fase en varios nodos simultáneamente el tiempo se reduce considerablemente.

4.3.6 Pseudocódigo del algoritmo

A continuación, se presentan los pseudocódigos de las fases de entrenamiento y clasificación del algoritmo. En la fase de entrenamiento (véase Algoritmo 1) se detalla cómo se construye el cierre convexo original y cómo se lleva a cabo todo el proceso de reajuste (subdivisión, congelaciones y poda). Para facilitar su entendimiento, se ha decidido encapsular este proceso de reajuste como un procedimiento aislado cuyo pseudocódigo se muestra en el Algoritmo 2. En el pseudocódigo de la fase de clasificación, se formaliza el proceso de etiquetado de un nuevo dato por parte del modelo (véase Algoritmo 3).

Cabe destacar que en estos algoritmos se hacen llamadas a procedimientos cuyo funcionamiento no se explica en detalle, como es el caso de las llamadas a las funciones *congelar* o *subdividir* en el Algoritmo 2. Estas funciones no poseen sus propios pseudocódigos ya que su funcionamiento ya ha sido comentado de forma teórica en puntos anteriores del trabajo, o porque su implementación es sencilla.

Algoritmo 1 Algoritmo de entrenamiento

Entradas:
Conjunto de entrenamiento para la primera fase de creación del cierre convexo inicial $S \in R^d$;
Conjunto de entrenamiento para la segunda fase que permite la modificación del cierre inicial $D \in R^d$;
Número de proyecciones τ;
Parámetro de escalado λ;
Número mínimo de iteraciones para la subdivisión y congelación $minIteraciones$;
Número de datos que deben de caer cerca de un vértice para que este se expanda $minDatosCerca$;

Salidas: Modelo M compuesto por τ matrices de proyección, y sus respectivos CH cierres convexos.

$M = \varnothing$
for $t = 1..\tau$ **do** ▷ Fase 1: generar las proyecciones y los cierres convexos iniciales con el conjunto S
  $P_t \sim N(0, 1)$ ▷ Crear una matriz de proyección aleatoria $[2 \times d]$;
  $\bar{S}_t : \{P_t x \,|\, x \in S\}$ ▷ Proyectar los datos del conjunto S en ese subespacio;
  $\{v_i\}_t = CH(\bar{S}_t)$ ▷ Calcular los vértices del CH;
  $\bar{c} = getCenter(\bar{S}_t)$ ▷ Calcular el centro del CH;
  $v_t^\lambda : \{\lambda v_i + (1 - \lambda)\bar{c} | v_i \in \{v\}_t\}$ ▷ Calcular el SCH en el espacio 2-D;
  $H_t = crearHistorialCH\left(v_t^\lambda\right)$ ▷ Estructura que alberga información del CH;
  $M = M \cup \left(P_t, v_t^\lambda, H_t\right)$ ▷ Almacena P_t, v_t y H_t;
end for
for $i = 1..len(D)$ **do** ▷ Fase 2: reajustar el modelo con cada dato i del conjunto D;
  for $t = 1..\tau$ **do** ▷ Para cada proyección t;
    $\bar{x}_i : \{P_t x_i \,|\, x_i \in D\}$ ▷ Proyectar el punto en ese subespacio;
    $M = reajustarCierres(M, t, \bar{x}_i, minIteraciones, minDatosCerca)$ ▷ Realizar los procesos de expansión, subdivisión, congelación y podado correspondientes en los cierres convexos de la proyección t;
  end for
end for

Algoritmo 2 Procedimiento utilizado durante el entrenamiento para reajustar los cierres de una proyección

Entradas:
Modelo M;
Valor t que indica qué proyección ha de reajustarse;
Punto $\bar{x}$ proyectado en el subespacio t;
Número mínimo de iteraciones $minIteraciones$ escogido para los procesos de subdivisión y congelación;
Parámetro $minDatosCerca$ que indica cuántos datos deben caer cerca de un vértice para que este se expanda;
Salidas: Modelo M actualizado;

1: **procedure** REAJUSTARCIERRES
2: $CHlist : \{CH | CH \in M_t\}$ ▷ Lista de cierres convexos de M asociados a la proyección t;
3: $InfoList : \{H \,|\, H \in M_t\}$ ▷ Información de cada cierre de la proyección t;
4: $listaCandidatosExpansion = []$ ▷ Se crea una lista para decidir si expandir cierres;
5: **for** $i = 1..len(CHlist)$ **do** ▷ Para cada cierre convexo de t;
6: $Pos = determinarLugarCaida(\bar{x}, CHlist_i)$ ▷ Se determina si el nuevo dato "cae" dentro del cierre convexo, en el margen o fuera de él;
7: **if** $Pos = DENTRO$ **then** $actualizarPtsSoporte(InfoList_i)$
8: **else if** $Pos = MARGEN$ **then**
9: $InfoList_i = actualizarPtsCercanos(InfoList_i)$ ▷ Para cada cierre en el que el punto ha caído en el margen, se calcula qué vértice es el más próximo y se añade el punto a su lista de puntos cercanos;
10: $listaCandidatosExpansion = marcarCierre(listaCandidatosExpansion, i)$ ▷ Se registra la caída en el margen del cierre i;
11: **end if**
12: **end for**
13: **if** $((len(listaCandidatosExpansion) == 1)\ and\ (InfoList_i[numeroDatosCerca] \geq minDatosCerca))$ **then** ▷ Si solamente ha caído en un cierre, y el vértice en cuestión supera el número mínimo de datos cerca, se expande;
14: $i = listaCandidatosExpansion[0]$ ▷ El índice del cierre es i;
15: $CHlist_i = expandirCierre(CHlist_i, InfoList_i)$
16: **end if**
17: $\mu = calcularRIC(CHlist, InfoList)$ ▷ Proceso de congelación;
18: $CHlist = congelar(CHlist, InfoList, minIteraciones, \mu)$ ▷ Proceso congelación;
19: $CHlist = subdividir(CHlist, InfoList, minIteraciones, \mu)$ ▷ Proceso subdivisión;
20: $CHlist = eliminar(CHlist, InfoList, minIteraciones, \mu)$ ▷ Proceso podado;
21: $M = actualizarModelo(M, CHlist, InfoList)$ ▷ Actualizar el modelo;
22: **end procedure**

Algoritmo 3 Algoritmo de clasificación

Entradas: Punto $x \in R^d$ a clasificar; Modelo M
Salidas: $Resultado \in \{NORMAL, ANOMALIA\}$

for $t = 1..\tau$ **do**
  $CHlist : \{CH \,|\, CH \in M_t\}$ ▷ Lista de cierres convexos de M asociados a la proyección t;
  $\bar{x_t} = P_t x$ ▷ Proyectar el punto;
  $Resultado = ANOMALIA$;
  for $i = 1..len(CHlist)$ **do** ▷ Para cada cierre convexo de t;
    if $\bar{x_t} \in CHlist_i$ **then**
      $Resultado = NORMAL$
      $Break$ ▷ Si se encuentra en algún cierre se deja de buscar;
    end if
  end for
  if $Resultado == ANOMALIA$ **then**
    $Break$ ▷ Si está fuera de todos los subcierres se clasificará como una anomalía y no será necesario seguir comprobando las proyecciones restantes;
  end if
end for

Capítulo 5

Estudio de rendimiento del algoritmo desarrollado

En este capítulo se va a evaluar el algoritmo desarrollado, estudiando los resultados que proporciona ante diferentes conjuntos de datos artificiales. Además, se probará su eficacia sobre dos conjuntos de datos pertenecientes a problemas reales de detección de anomalías. Para disponer de un punto de referencia a la hora de evaluar los resultados obtenidos por nuestro método, se compararán sus resultados con los obtenidos por distintos algoritmos clásicos en detección de anomalías, permitiéndonos así entender a qué nivel se encuentra nuestro método. Además de probar dichos algoritmos clásicos, hemos decidido incluir en la comparación a la primera versión de nuestro algoritmo, la cual no poseía la capacidad de subdividir el cierre convexo. De esta forma, podremos medir también la mejora obtenida de una versión a otra.

Dado que los autores del algoritmo base de partida lo bautizaron como *DSCH* (Cierre Convexo Escalado Distribuido, del inglés Distributed Scaled Convex Hull), a lo largo de este capítulo utilizaremos las siglas *OnlineS-DSCH* (del inglés Online and Subdivisible Distributed Scaled Convex Hull) para referirnos a la versión avanzada del algoritmo desarrollado en la Sección 4.3, mientras que a su versión previa sin capacidad de subdivisión, comentada en la Sección 4.2, nos referiremos como *Online-DSCH* (del inglés Online Distributed Scaled Convex Hull).

En primer lugar, se describirán los materiales y métodos utilizados en este estudio de rendimiento para, finalmente, presentar y discutir los resultados.

5.1 Conjuntos de datos artificiales

Para probar la eficacia de nuestro método, en esta fase de experimentación hemos decidido emplear nuestros propios conjuntos de datos artificiales ya que podemos construirlos con formas geométricas personalizadas. Así, ya que le hemos dado la capacidad de reajustarse a nuestro algoritmo, podemos probar su eficacia ajustándose a formas geométricas más o menos complejas. Para generar estos conjuntos de datos, se han utilizado funciones implementadas en la librería *Scikit-learn*, que permiten generar distribuciones espaciales con formas preestablecidas.

Todos los conjuntos de datos artificiales están compuestos por un total de 17.000 ejemplos cada uno con tres variables de entrada. De estos 17.000 datos, 15.000 se han destinado a la fase de entrenamiento (88,23%), y los 2.000 datos restantes a la fase de *test* (11,77%). Los datos utilizados durante la fase de entrenamiento son todos ellos datos normales, en cambio, en los datos utilizados para la fase de *test*, un 1% son datos anómalos. La fase de entrenamiento se divide a su vez en dos procesos: la generación de los cierres convexos en las distintas proyecciones a partir de un conjunto de datos inicial, y el reajuste de los mismos a partir de un segundo conjunto de datos, lo que equivale a simular un aprendizaje *online*. Para formar el cierre inicial en cada proyección (primer proceso), se han destinado 5.000 de los 15.000 datos de entrenamiento. Para llevar a cabo el reajuste de los cierres (segundo proceso), los 10.000 restantes. Ya que los otros algoritmos utilizados en la comparación no dividen su fase de entrenamiento en dos procesos como nuestra técnica (5.000 y 10.000 datos), estos algoritmos utilizarán los 15.000 datos de entrenamiento de forma conjunta. A continuación, se van a describir las características principales de los distintos conjuntos de datos artificiales utilizados.

5.1.1 Datos cuasiesféricos

La forma cuasiesférica en los datos es frecuente en detección de anomalías. Es el problema más simple e ideal para su resolución mediante técnicas basadas de cierre convexo, ya que el conjunto de datos normales puede ser fácilmente representado mediante unos pocos cierres. En algunos casos, solamente será necesario un cierre convexo por proyección, como se puede observar en la Figura 5.1. Por ello, nuestra versión del algoritmo con subdivisión realmente no va a emplear todo su potencial ante este conjunto de datos. De todas formas, todos los algoritmos deberían de ser capaces de resolverlo sin problemas. Aunque en un espacio tridimensional este conjunto de datos presente forma de cuasiesfera y nos refiramos a él como tal, al ser proyectado su forma es la de una circunferencia, de ahí su simpleza.

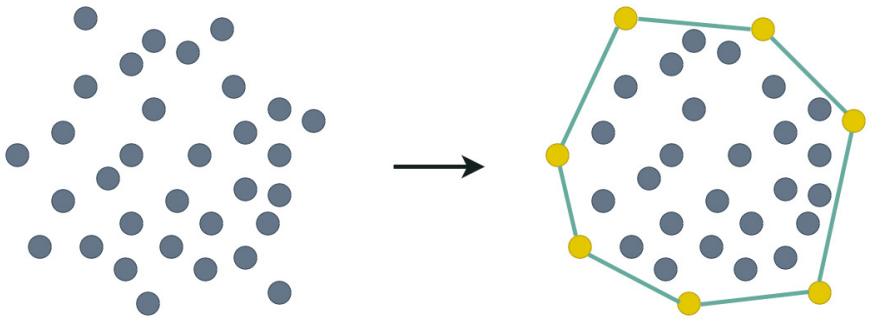

Figura 5.1: A la izquierda se muestra un conjunto de datos artificiales con forma cuasiesférica. A la derecha se muestra su delimitación mediante un único cierre convexo.

5.1.2 Datos en dos cuasiesferas no solapadas

El conjunto de datos compuesto por dos cuasiesferas separadas representa dos zonas de datos normales aisladas en el espacio. Esta separación en el espacio tridimensional provoca que en ciertas proyecciones a espacios bidimensionales las dos circunferencias se proyecten de igual modo separadas. Si nuestra técnica no fuese capaz de subdividirse, englobaría a ambas circunferencias en un solo cierre convexo, lo que incluiría también al espacio existente entre las dos regiones. Este comportamiento puede provocar grandes problemas, ya que clasificaría datos anómalos como normales (falsos negativos) por el hecho de encontrarse entre ambas clases normales, como se puede observar en la Figura 5.2. En este conjunto de datos, nuestra versión del algoritmo con subdivisión debe obtener mejores resultados que la primera versión que operaba con un solo cierre.

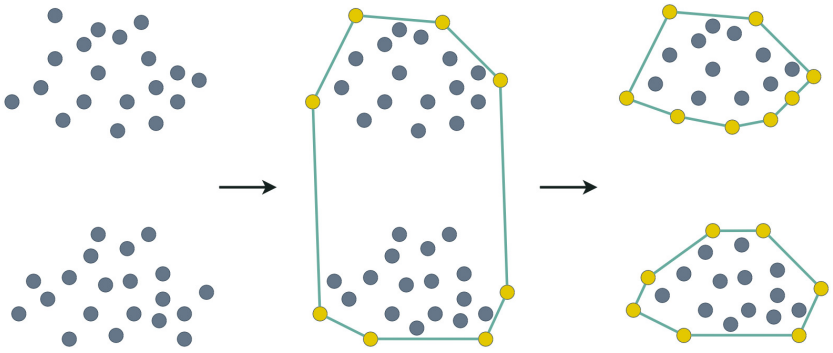

Figura 5.2: A la izquierda, se muestran dos conjuntos de datos artificiales con forma cuasiesférica separados en el espacio. En el centro, la segmentación de los datos normales que llevaría a cabo el algoritmo sin subdivisión de cierres convexos. A la derecha, la separación ideal de los datos normales a modelar por el algoritmo con subdivisión.

5.1.3 Datos en forma de reloj de arena

Este conjunto de datos es conocido como reloj de arena debido a la estrechez que presenta en mitad de la nube de datos, convirtiéndolo en un conjunto complejo de modelar mediante un solo cierre convexo sin abarcar zonas innecesarias del espacio. Como se puede ver en la Figura 5.3 existen dos zonas principales donde se encuentran los datos normales, pero debido a su cercanía, no llegan a encontrarse totalmente separados en el espacio.

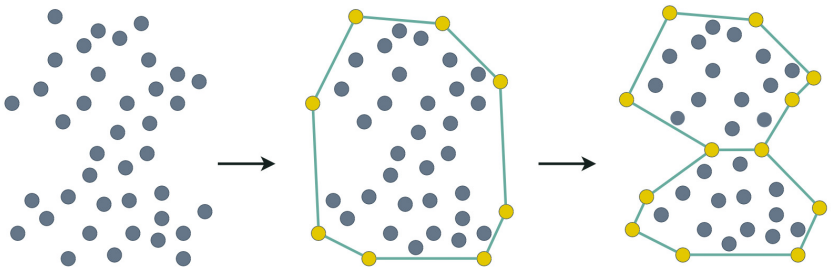

Figura 5.3: A la izquierda, se muestran dos conjuntos de datos artificiales con forma cuasiesférica solapados en el espacio. En el centro, la segmentación de los datos normales que llevaría a cabo el algoritmo sin subdivisión de cierres convexos. A la derecha, la separación ideal de los datos normales a modelar por el algoritmo con subdivisión.

5.1.4 Datos en forma de media luna

Los conjuntos de datos con forma geométrica de media luna son muy comunes en detección de anomalías. Debido a su forma no convexa, precisamos de la versión con subdivisión de nuestro algoritmo para obtener buenos resultados en este tipo de problemas. En la Figura 5.4 se puede observar cómo la región vacía en la zona cóncava de la media luna puede empeorar gravemente la clasificación de anomalías situadas en esa zona, las cuales serán consideradas como datos normales si se utiliza un único cierre convexo.

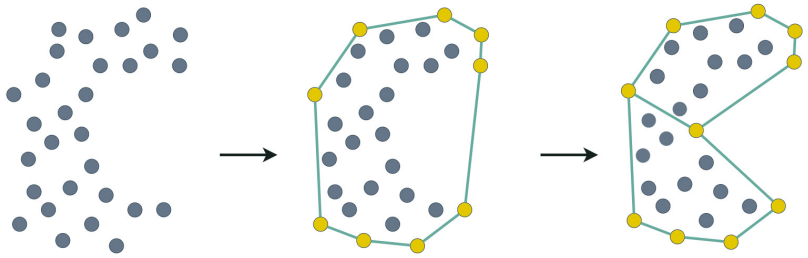

Figura 5.4: A la izquierda, se muestra un conjunto de datos artificiales con forma de media luna. En el centro, la segmentación de los datos normales que llevaría a cabo el algoritmo sin subdivisión de cierres convexos. A la derecha, la separación ideal de los datos normales a modelar por el algoritmo con subdivisión.

5.1.5 Datos en forma de "S"

Este tipo de conjuntos con forma serpenteada o de "S" no son tan comunes en problemas reales. Sin embargo, hemos decidido incluirlos en las pruebas ya que pueden ser útiles para medir la flexibilidad de nuestro algoritmo a la hora de reajustarse. Se pueden considerar como una versión de los conjuntos de datos con forma de media luna más compleja. Su morfología se puede observar en la Figura 5.5.

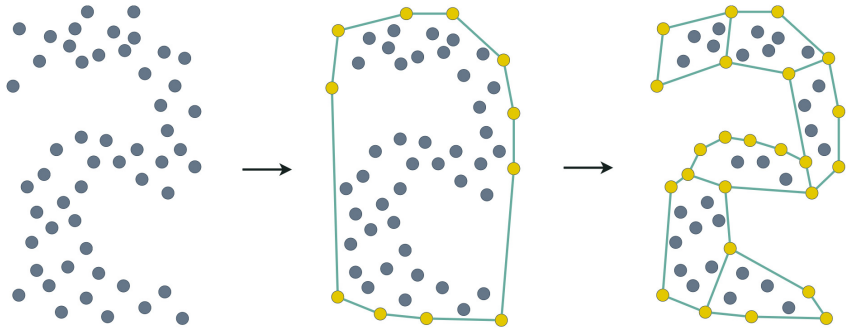

Figura 5.5: A la izquierda, se muestra un conjunto de datos artificiales con forma "S". En el centro, la segmentación de los datos normales que llevaría a cabo el algoritmo sin subdivisión de cierres convexos. A la derecha, la separación ideal de los datos normales a modelar por el algoritmo con subdivisión.

5.2 Conjuntos de datos reales

Además de los conjuntos artificiales, se han empleado también conjuntos reales de mayor dimensión para conocer el rendimiento del modelo en escenarios más realistas.

5.2.1 Calcificaciones en mamografías

Este conjunto ha sido proporcionado por Aleksandar Lazarevic, tras su uso para detección de microcalcificaciones en mamografías [44]. Está formado por un total de 11183 datos, cada uno con 6 atributos numéricos. Los datos se encuentran etiquetados como datos normales o anomalías (microcalcificaciones), existiendo un total de 260 datos etiquetados como anómalos (2,3%).

De los 11.183 datos, 9.000 se han destinado a la fase de entrenamiento (80,48%), y los 2.183 datos restantes a la fase de *test* (19,52%). Los datos utilizados durante la fase de entrenamiento son todos ellos datos normales. Los datos anormales se utilizan solo para la fase de *test*, lo que supone un 11,91% del total. Para formar el cierre inicial en cada proyección, se han destinado 4.500 de los 9.000 datos de entrenamiento. Para llevar a cabo el reajuste posterior de los cierres se han destinado los 4.500 restantes. Ya que los otros algoritmos utilizados en la comparación no dividen su fase de entrenamiento en dos como nuestra técnica, estos algoritmos utilizarán los 9.000 datos de entrenamiento de forma conjunta.

5.2.2 Fraude en tarjetas de crédito

El algoritmo también se probó sobre un segundo conjunto de datos real de mayor dimensionalidad. Este conjunto representa fraudes en tarjetas de crédito, y se encuentra disponible de manera pública en Kaggle [45]. Está formado por un total de 284.808 datos, cada uno con 30 atributos numéricos. Los datos se encuentran etiquetados como datos normales o anomalías (fraudes), existiendo un total de 492 datos etiquetados como anómalos (0,17%).

De la totalidad de los datos se han destinado a la fase de entrenamiento el 80% de los datos, y el 20% restante a la fase de *test*. Los datos utilizados durante la fase de entrenamiento son todos ellos datos normales. Los datos anormales se utilizan solo para la fase de *test*, lo que supone un 0,86% del total. Del mismo modo que en los anteriores conjuntos de datos, ya que los otros algoritmos utilizados en la comparación no dividen su fase de entrenamiento en dos como nuestra técnica, estos algoritmos utilizarán la totalidad de los datos destinados al entrenamiento de forma conjunta.

La Tabla 5.1 resume las características de los conjuntos de datos empleados.

	Nombre	Nº características	Entrenamiento	Test	Anomalías
Artificial	Una cuasiesfera Dos cuasiesferas Reloj de arena Media luna Forma de S	3	15.000 (inicial = 5.000)	2.000	20 (1,00%)
Real	Mamografías	6	9.000 (inicial = 4.500)	2.183	260 (11,91%)
	Tarjetas de crédito	30	227.846 (inicial = 150.000)	56.962	492 (0,86%)

Tabla 5.1: Características de los conjuntos de datos empleados.

Cabe destacar también que todas las variables de entrada de los diferentes algoritmos han sido previamente normalizadas. El método de normalización empleado ha sido *MinMax Scaler*, o escalado de variables, mediante el cual cada variable de entrada X se normaliza dado uno límite superior X_{max} y un límite inferior X_{min} de la siguiente manera:

$$X_{normalizado} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

5.3 Algoritmos empleados en las pruebas y sus configuraciones

Como se ha explicado, a la hora de resolver en detección de anomalías existe una gran variedad de aproximaciones válidas. Para obtener los hiperparámetros óptimos de cada algoritmo utilizado en las pruebas se ha empleado un *grid search* entre un conjunto de valores preestablecidos. A continuación, se va a describir el funcionamiento de los algoritmos clásicos en detección de anomalías seleccionados para llevar a cabo la comparación de los resultados y sus hiperparámetros:

- **One-Class SVM**: *O-SVM* es la técnica por antonomasia dentro de las del tipo que se basan en el dominio [26]. Genera un límite que separa las clases en el espacio, para ello aproxima el hiperplano que separa de forma óptima a los puntos de una clase de la de otra. Esta técnica busca que este hiperplano maximice su distancia (margen) respecto a los puntos del espacio que estén más cerca de él mismo. Los ejemplos más cercanos al hiperplano son los llamados vectores de soporte, y permiten prescindir del resto de datos del sistema. Debido a que la clasificación de nuevos datos se determina únicamente por su ubicación respecto al límite, estos clasificadores son insensibles al tamaño y a la densidad de cada clase. Este algoritmo utiliza tres hiperparámetros:

- ν: Límite superior en la fracción de errores de entrenamiento y límite inferior de la fracción de vectores de soporte. Debe encontrarse en el intervalo (0, 1].

- Kernel: Especifica el tipo de *kernel* que utilizará el algoritmo. Este *kernel* es la función matemática que permite proyectar los datos de entrada sobre un espacio de mayor dimensionalidad, de forma que se pueda encontrar un hiperplano que los separe.

- γ: Define cuánta influencia tiene un solo dato del entrenamiento.

Se presenta en la Tabla 5.2 los valores óptimos que tomaron estos hiperparámetros durante las diferentes pruebas.

Conjunto de datos	ν	**Kernel**	γ
Una cuasiesfera	0,021	rbf	0,010
Dos cuasiesferas	0,022	rbf	0,010
Reloj de arena	0,027	rbf	0,015
Media luna	0,022	rbf	0,015
Forma de *S*	0,027	rbf	0,010
Mamografías	0,100	rbf	0,250
Tarjetas de crédito	0,021	rbf	0,300

Tabla 5.2: Configuración de hiperparámetros utilizada para el algoritmo *One-Class SVM*.

- **Robust Covariance**: O *RC*, en español *Covarianza Robusta* [46], es un método de tipo probabilístico. Su funcionamiento se basa en estimar la matriz de covarianza de una población. Esta versión garantiza mediante un estimador *"robusto"* que el algoritmo no se vea influenciado por los datos anómalos a la hora de aproximar el modelo.

  - Contaminación: Es la proporción de puntos más aislados que serán clasificados como anomalías.

Se presenta en la Tabla 5.3 los valores que tomaron sus hiperparámetros.

Conjunto de datos	Contaminación
Una cuasiesfera	0,001
Dos cuasiesferas	0,001
Reloj de arena	0,005
Media luna	0,001
Forma de *S*	0,003
Mamografías	0,100
Tarjetas de crédito	0,100

Tabla 5.3: Configuración de hiperparámetros utilizada para el algoritmo *Robust Covariance* que obtuvo mejores resultados durante las pruebas.

- **Isolation Forest**: o *IF* es una técnica basada en el dominio que utiliza árboles de decisión [47]. Se basa en la idea de que los datos anómalos se encuentran aislados de los datos normales, y que por tanto son más fáciles de identificar, para realizar particiones en el espacio. Cada nodo de un árbol corresponde a una partición aleatoria del espacio, de forma que nodos hijos serán a su vez particiones del espacio de su nodo padre. De esta forma, las anomalías se situarán de forma aislada en particiones del espacio (nodos) cercanos a la raíz, y los datos normales en la zonas más alejadas, ya que para aislarlos individualmente es necesario particionar el espacio mucho más.

  - Contaminación: Es la proporción de puntos más aislados que serán clasificados como anomalías.

  - Random state: Semilla utilizada por el generador de números aleatorios. El algoritmo aísla a los datos seleccionando de forma aleatoria (en base a la semilla) una característica y un valor de separación (entre los valores máximo y mínimo de dicha característica).

Se presenta en la Tabla 5.4 los valores que tomaron sus hiperparámetros.

Conjunto de datos	Contaminación	Random state
Una cuasiesfera	0,01	20
Dos cuasiesferas	0,02	30
Reloj de arena	0,02	15
Media luna	0,02	25
Forma de *S*	0,01	30
Mamografías	0,25	50
Tarjetas de crédito	0.10	40

Tabla 5.4: Configuración de hiperparámetros utilizada para el algoritmo *Isolation Forest* que obtuvo mejores resultados durante las pruebas.

- **Local Outlier Factor**: *LOF* [48] es una técnica basada en la densidad relativa. Esta técnica estima la densidad de los vecindarios de forma que un dato que se encuentre en una vecindad con baja densidad será anómalo, mientras que un dato que se encuentre en una vecindad con alta densidad se considerará normal.

  - Contaminación: Es la proporción de puntos más aislados que serán clasificados como anomalías.
  - Número de vecinos: Número de vecinos de un punto utilizados para las operaciones del algoritmo.

  Se presenta en la Tabla 5.5 los valores que tomaron sus hiperparámetros.

Conjunto de datos	Contaminación	Número de vecinos
Una cuasiesfera	0,005	55
Dos cuasiesferas	0,005	35
Reloj de arena	0,005	35
Media luna	0,010	55
Forma de *S*	0,005	45
Mamografías	0,250	50
Tarjetas de crédito	0,100	55

Tabla 5.5: Configuración de hiperparámetros utilizada para el algoritmo *Local Outlier Factor* que obtuvo mejores resultados durante las pruebas.

En cuanto a nuestros algoritmos, a continuación se presenta en la Tabla 5.6 los valores que tomaron sus hiperparámetros durante las diferentes pruebas.

Conjunto de datos	Proyecciones	λ	Datos para expandirse	Iteraciones para dividir	Iteraciones para congelar	Iteraciones para podar
Una cuasiesfera	50	1,3	2	1.500	1.500	1.500
Dos cuasiesferas	50	1,3	2	1.500	1.500	1.500
Reloj de arena	50	1,2	2	1.500	1.500	1.500
Media luna	50	1,3	2	1.500	1.500	1.500
Forma de S	75	1,2	2	1.500	1.500	1.500
Mamografías	125	1,15	2	1.000	1.000	1.000
Tarjetas de crédito	200	1,20	2	10.000	10.000	10.000

Tabla 5.6: Configuración de hiperparámetros utilizada por nuestro algoritmo con subdivisión durante las pruebas.

Ya que es necesario definir un margen suficientemente grande para no eliminar CHs de forma precipitada, en nuestras pruebas hemos decidido que el intervalo de tiempo para la poda coincida con el de congelación, ya que ambos deben de ser grandes y de esta forma se facilita la configuración del modelo.

5.4 Medidas de rendimiento y metodología de evaluación

Ya que nos encontramos ante un problema de detección de anomalías, para evaluar los resultados obtenidos por los distintos métodos durante las pruebas, utilizaremos las métricas clásicas para problemas de clasificación. Estas métricas estadísticas tienen su fundamento en el número de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos:

- **Verdadero Positivo (VP)**: Dato anómalo clasificado por el sistema también como anómalo.
- **Verdadero Negativo (VN)**: Dato normal clasificado por el sistema también como normal.
- **Falso Positivo (FP)**: Dato normal clasificado por el sistema como anómalo.
- **Falso Negativo (FN)**: Dato anómalo clasificado por el sistema como normal.

Estas métricas permiten analizar la validez de un modelo en base a las clasificaciones que ha realizado sobre un conjunto de datos. Las métricas más comunes, y las que vamos a emplear en este capítulo, son la sensibilidad, la especificidad, la precisión y la similitud:

- **Sensibilidad**: Mide la probabilidad de clasificar a un dato anómalo como tal. Obtener buenos resultados en este aspecto es fundamental para los modelos de detección de anomalías. La sensibilidad se define como:

$$Sensibilidad = \frac{VP}{VP + FN}$$

- **Especificidad**: Mide la probabilidad de clasificar a un dato normal como tal. Como hemos visto, en detección de anomalías, clasificar a un dato normal erróneamente no es tan crítico como hacerlo con las anomalías, pero sigue siendo un aspecto relevante. La especificidad se define como:

$$Especificidad = \frac{VN}{VN + FP}$$

- **Precisión**: Mide la probabilidad de clasificar a un dato correctamente, relacionando el número de datos verdaderos con el número total de datos. La precisión se define como:

$$Precision = \frac{VP + VN}{VP + VN + FP + FN}$$

- **Similitud**: Relaciona la precisión y la sensibilidad, su fin es poder comparar los resultados obtenidos por el modelo respecto a los ideales de una manera más directa. Mide el equilibrio existente entre precisión y sensibilidad, permitiéndonos decidir entre distintos pares de valores de precisión y sensibilidad cuál es la mejor combinación. La similitud se define como:

$$Similitud = 1 - \frac{\sqrt{(1-p)^2 + (1-s)^2}}{\sqrt{2}}$$

  donde p es la precisión de la clasificación y s la sensibilidad.

En la siguiente sección se muestran los resultados obtenidos por los diferentes algoritmos sobre cada conjunto de datos. Para obtener resultados más robustos cada prueba se ha ejecutado un total de cinco veces. Por tanto, los valores finales son el promedio de los cinco resultados obtenidos en cada métrica. En el caso de los conjuntos artificiales, se ha generado un nuevo conjunto de datos para cada una de las 5 pruebas, conservando la forma pero variando aspectos como el tamaño y la distribución de los datos en el espacio. En el caso de los conjuntos de datos reales, dado que no es posible generar datos nuevos, estos se han particionado (usando una *five-fold cross validation*).

5.5 Resultados

5.5.1 Cuasiesfera

En la Tabla 5.7 se muestran los resultados obtenidos por los diferentes métodos tras su ejecución sobre conjuntos de datos con forma de cuasiesfera. El mejor algoritmo ha sido *RC* debido a sus resultados en la métrica de similitud. En el resto de las métricas *RC* también ha sido el primero, excepto en sensibilidad, dónde nuestro algoritmo *OnlineS-DSCH* lo supera. Esto se debe a que nuestro algoritmo ha detectado alguna anomalía que *RC* ha pasado por alto, aunque la diferencia es mínima (en torno al 0,5%). En las métricas de especificidad, precisión y similitud *OnlineS-DSCH* obtiene resultados cercanos a las otras técnicas, llegando a superarlas en varios casos. Respecto a la versión sin subdivisión *Online-DSCH, OnlineS-DSCH* mejora notablemente en sensibilidad y similitud, a costa de disminuir su precisión. Esto se debe a que *OnlineS-DSCH* se ajusta de una forma más agresiva a la forma de los datos normales para no incluir anomalías en su interior. Esto provoca que en algunos casos se dejen datos normales fuera de los límites normales, aumentando el número de falsos positivos durante las pruebas.

Conjunto de datos	Sensibilidad	Especificidad	Precisión	Similitud
OnlineS-DSCH	**98,75±1,1**	95,74±1,7	95,77±2,1	96,85±1,7
Online-DSCH	89,50±3,2	99,02±0,6	98,93±0,3	92,53±2,3
O-SVM	98,50±0,7	97,78±0,9	97,79±1,2	98,11±1,0
RC	98,25±1,2	**99,77±0,1**	**99,76±0,4**	**98,75±0,9**
IF	70,47±4,3	98,92±0,8	98,62±0,9	79,10±3,1
LOF	98,05±1,2	99,52±0,3	99,51±0,3	98,58±0,9

Tabla 5.7: Resultados medios obtenidos y desviaciones típicas para el conjunto de datos artificial con forma cuasiesférica.

5.5.2 Dos cuasiesferas

En la Tabla 5.8 se muestran los resultados obtenidos por los diferentes métodos tras su ejecución sobre conjuntos de datos con forma de dos cuasiesferas separadas.

De acuerdo a la similitud el algoritmo que mejores resultados ha obtenido es *LOF*, siendo superado por *RC* en especificidad y precisión (aunque la diferencia es mínima). *OnlineS-DSCH* mejora respecto a su versión sin subdivisión en cuanto a la sensibilidad y similitud, pero no en especificidad y precisión, debido de nuevo al aumento del número de falsos positivos. A pesar de ello obtiene un buen rendimiento, superando incluso a la mayoría de técnicas en el número de verdaderos positivos detectados.

Conjunto de datos	Sensibilidad	Especificidad	Precisión	Similitud
OnlineS-DSCH	98,50±0,7	94,30±2,5	94,34±2,1	95,86±1,6
Online-DSCH	90,00±2,1	99,05±0,3	98,96±1,4	92,88±1,5
O-SVM	98,25±0,6	98,40±0,7	98,39±0,5	98,32±0,5
RC	93,75±2,1	**99,84±0,1**	**99,78±0,3**	95,58±1,4
IF	93,90±2,2	98,45±1,0	98,41±2,1	95,54±1,7
LOF	**99,05±0,3**	99,55±0,4	99,54±0,3	**99,26±0,2**

Tabla 5.8: Resultados medios obtenidos y desviaciones típicas para el conjunto de datos artificial con forma de dos cuasiesferas.

En la Figura 5.6 se puede observar imágenes reales del reajuste progresivo de los cierres ante un conjunto de este tipo. Las seis imágenes que conforman la figura pertenecen a la misma proyección en instantes de tiempo distintos. Los puntos azules representan a los datos utilizados para generar el cierre inicial proyectados (la subfigura *a* plasma los primeros instantes tras su generación), los de color verde los utilizados para reajustar los cierres progresivamente, y los amarillos a los centros de cada cierre. Las líneas rojas representan a los límites de los cierres convexos, las líneas negras a los límites de dichos cierres convexos escalados respecto a su centro. Las líneas violetas unen a los centros de las aristas con sus puntos soporte. Como se puede ver en la subfigura *e*, tras múltiples subdivisiones se consigue representar a cada cuasiesfera de datos normales (circunferencias tras ser proyectadas) de forma aislada, mejorando la precisión del algoritmo. Como se puede ver en la subfigura *f*, los cierres se siguen reajustando ante la caída de datos normales aún encontrándose congelados.

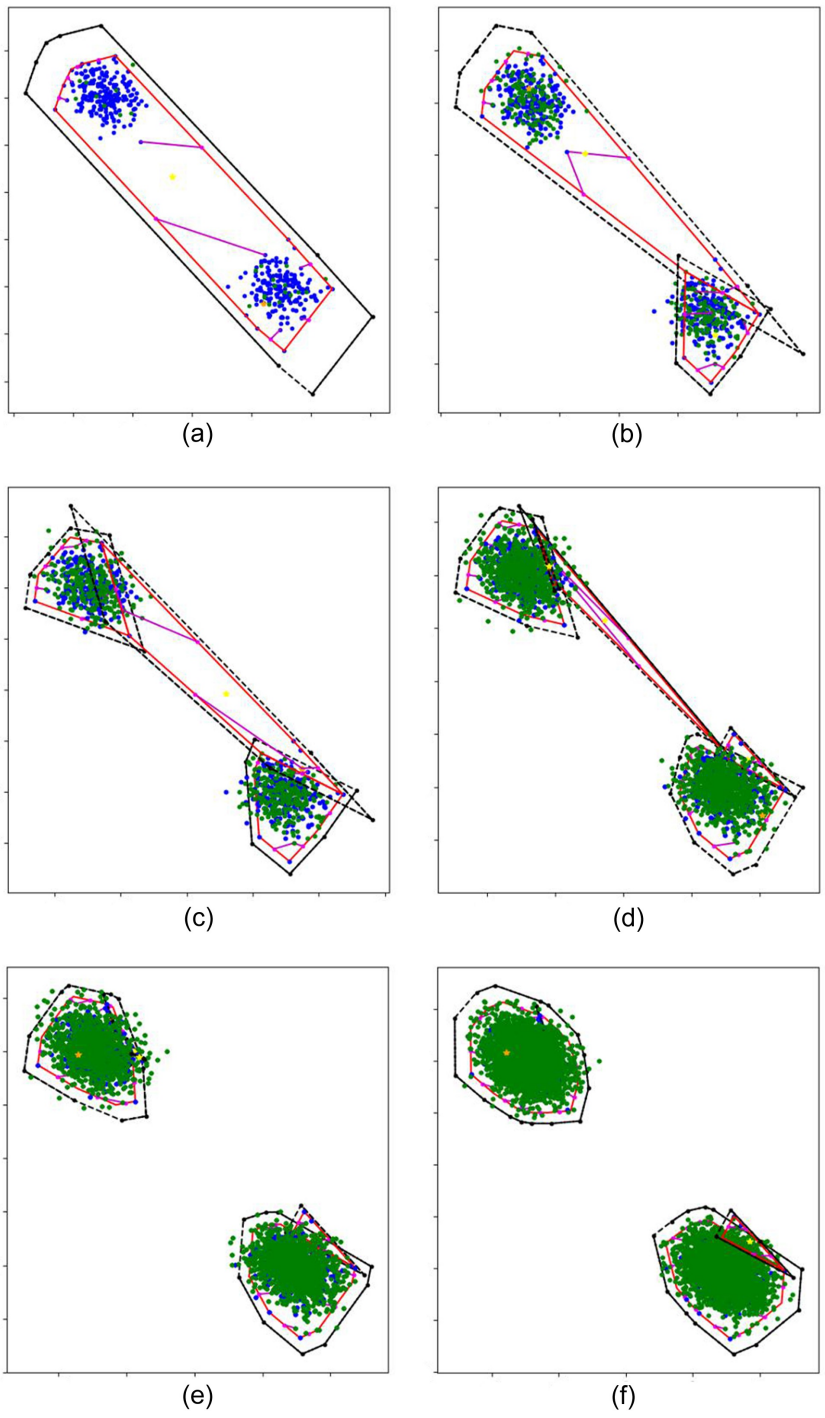

Figura 5.6: Reajuste progresivo del algoritmo sobre dos cuasiesferas separadas en el espacio.

5.5.3 Reloj de arena

En la Tabla 5.9 se muestran los resultados obtenidos por los diferentes métodos tras su ejecución sobre conjuntos de datos con forma de reloj de arena. De acuerdo a la similitud, el algoritmo que mejores resultados ha obtenido es *RC*. En especificidad y precisión *LOF*, y *OnlineS-DSCH* en sensibilidad. *OnlineS-DSCH* vuelve a mejorar respecto a su versión sin subdivisión en sensibilidad y similitud, esta vez casi sin perder rendimiento en especificidad y precisión. En general obtiene buenos resultados, superando además en similitud a la mayoría de técnicas.

Conjunto de datos	Sensibilidad	Especificidad	Precisión	Similitud
OnlineS-DSCH	**97,50±1,7**	96,78±1,2	96,79±1,5	97,12±1,5
Online-DSCH	85,60±3,4	99,05±1,2	98,92±0,9	89,79±1,7
O-SVM	91,50±2,0	97,80±2,1	97,74±1,1	93,79±1,8
RC	96,50±1,1	99,64±0,2	99,61±0,2	**97,51±0,8**
IF	95,95±1,3	98,64±0,7	98,62±0,9	96,96±1,1
LOF	95,75±1,3	**99,66±0,2**	**99,62±0,2**	97,12±0,9

Tabla 5.9: Resultados medios obtenidos y desviaciones típicas para el conjunto de datos artificial con forma de reloj de arena.

En la Figura 5.7 se pueden observar imágenes reales del reajuste progresivo de los cierres ante un conjunto de datos con forma de reloj de arena. Las seis imágenes que conforman la figura pertenecen a la misma proyección en instantes de tiempo distintos. En este caso, el algoritmo ha generado un gran número de cierres convexos para representar a la región normal. Este comportamiento puede provocar que determinadas zonas puedan llegar a sobreajustarse, como podría ser el caso de la zona superior de la región en la subfigura *e*. La agresividad a la hora de subdividir y generar cierres se puede modificar configurando los hiperparámetros del algoritmo. Si se dispone de conocimiento a *a priori* de la forma de los datos normales, y nos encontramos ante un conjunto que no va a necesitar de un gran número de cierres para ser representado, podemos configurar el algoritmo para que no realice demasiadas subdivisiones. Esto se podría conseguir balanceando el número de datos destinados a generar el cierre inicial y los destinados a realizar el reajuste, incrementando el umbral que permite a un cierre subdividirse (aumentando el valor procedente del *RIC*) o aumentando el número de iteraciones mínimas para subdividir los cierres.

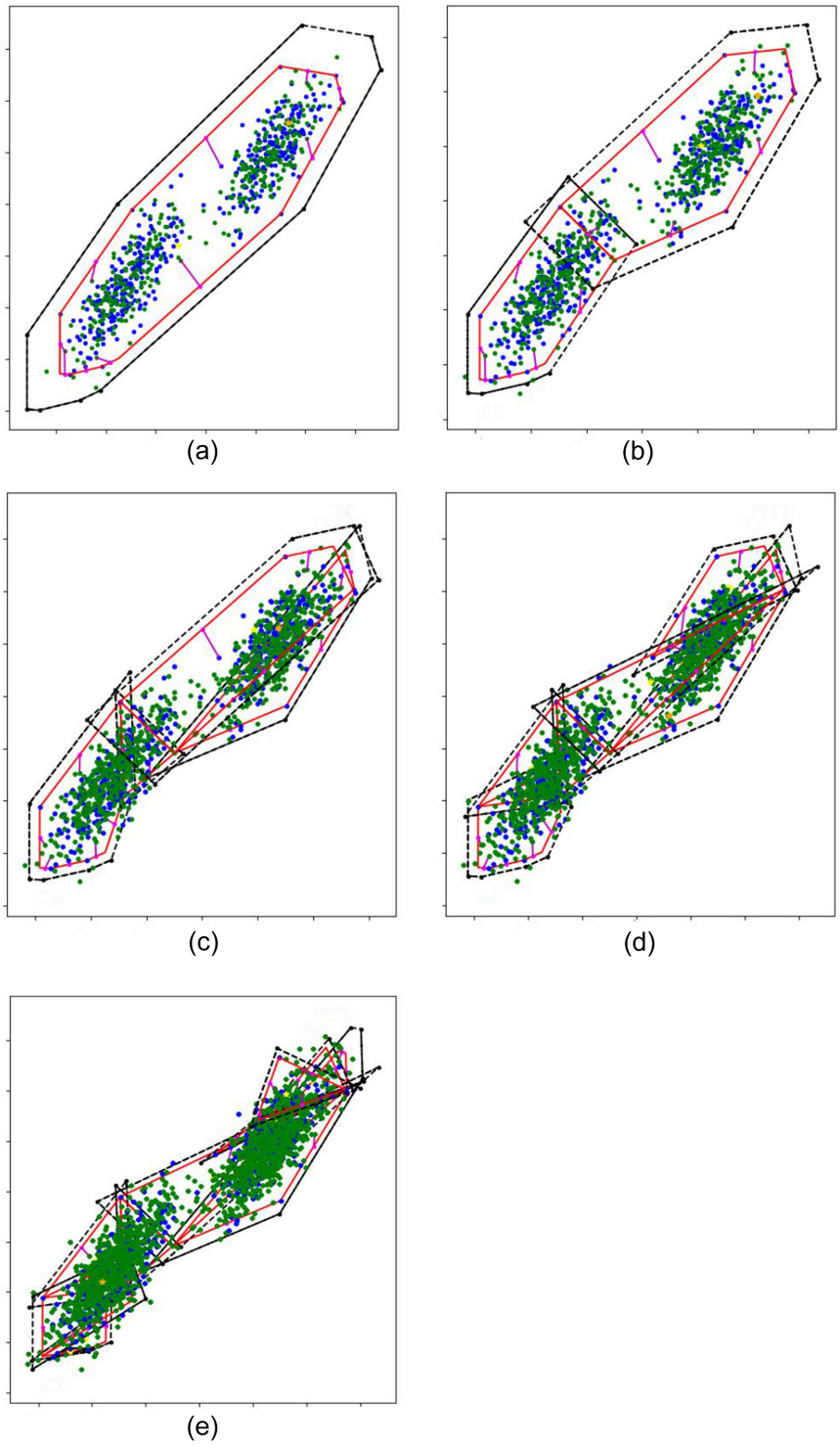

Figura 5.7: Reajuste progresivo del algoritmo sobre un conjunto con forma de reloj de arena.

5.5.4 Media luna

En la Tabla 5.10 se muestran los resultados obtenidos por los diferentes métodos tras su ejecución sobre conjuntos de datos con forma de media luna. El algoritmo que mejores resultados ha obtenido en similitud es *RC*. Además de esto, ha sido el primero tanto en especificidad como en precisión, siendo únicamente superado en la métrica de sensibilidad por *OnlineS-DSCH*. Nuestro algoritmo *OnlineS-DSCH*, mejora notablemente en sensibilidad respecto a su versión sin subdivisión. Esto se debe a la forma cóncava del conjunto de datos. Gracias a la subdivisión de los cierres, *OnlineS-DSCH* es capaz de detectar anomalías antes pasadas por alto en su versión *Online-DSCH*. En general, obtiene buenos resultados, siendo el segundo algoritmo en cuanto a similitud.

Conjunto de datos	Sensibilidad	Especificidad	Precisión	Similitud
OnlineS-DSCH	**97,50±0,9**	95,70±1,2	95,72±0,8	96,48±0,8
Online-DSCH	63,75±14,6	98,60±0,7	98,25±1,1	74,33±5,3
O-SVM	93,50±2,2	96,45±2,3	96,42±1,3	94,75±1,8
RC	96,50±1,3	**99,65±0,2**	**99,62±0,3**	**97,51±0,9**
IF	88,50±2,2	98,40±0,9	98,30±1,4	91,74±1,7
LOF	86,75±3,3	99,45±0,2	99,32±0,3	90,61±2,3

Tabla 5.10: Resultados medios obtenidos y desviaciones típicas para el conjunto de datos artificial con forma de media luna.

5.5.5 Forma de "S"

En la Tabla 5.11 se muestran los resultados obtenidos por los diferentes métodos tras su ejecución sobre conjuntos de datos con forma de "S".

El algoritmo que mejores resultados ha obtenido en similitud es *O-SVM*. Además de esto, *O-SVM* ha sido el mejor en sensibilidad, y *LOF* en especificidad y precisión. Nuestro algoritmo *OnlineS-DSCH* mejora notablemente en sensibilidad respecto a su versión sin subdivisión, de nuevo, debido a la forma cóncava del conjunto de datos. Sin embargo, para obtener esta mejora, su especificidad y precisión se han visto reducidas en torno a un 8% respecto a su versión anterior *Online-DSCH*. A pesar de este empeoramiento, nuestro algoritmo obtiene en general buenos resultados, superando a varias de las técnicas en sensibilidad, y alcanzando resultados considerables en las demás métricas.

Conjunto de datos	Sensibilidad	Especificidad	Precisión	Similitud
OnlineS-DSCH	84,95±2,1	89,25±2,7	89,21±2,0	86,90±2,0
Online-DSCH	39,20±12,2	97,99±1,8	97,41±0,6	56,96±9,4
O-SVM	**91,75±1,8**	96,45±0,7	96,40±1,5	**93,61±1,6**
RC	82,45±3,5	98,75±1,6	98,59±1,0	87,53±3,5
IF	86,65±3,1	96,65±1,4	96,55±1,7	90,23±2,4
LOF	89,5±2,6	**99,29±1,3**	**99,19±0,2**	92,55±1,5

Tabla 5.11: Resultados medios obtenidos y desviaciones típicas para el conjunto de datos artificial con forma de "S".

En la Figura 5.8 se pueden observar imágenes reales del reajuste progresivo de los cierres ante un conjunto de datos con forma de *S*. Las cuatro imágenes que conforman la figura pertenecen a la misma proyección en instantes de tiempo distintos. En este caso, el algoritmo también ha generado un gran número de cierres convexos para representar a la región normal. A diferencia de la figura 5.7, en esta prueba, una de las zonas cóncavas no ha sido expulsada de la región normal mediante las múltiples subdivisiones (la zona inferior). Esto es debido a la utilización de las distancias a los puntos soporte como criterio de subdivisión, el cual puede no ser el mejor en algunos casos. Sin embargo, ya que el algoritmo utiliza múltiples proyecciones para clasificar un dato como normal u *outlier*, el cómputo global del resultado compensa este tipo de errores, ya que por ejemplo, aunque en la proyección de la figura 5.8 un dato anómalo se clasifique como normal, con que en alguna de las otras proyecciones "mejor ajustadas" se detecte como anómalo será suficiente.

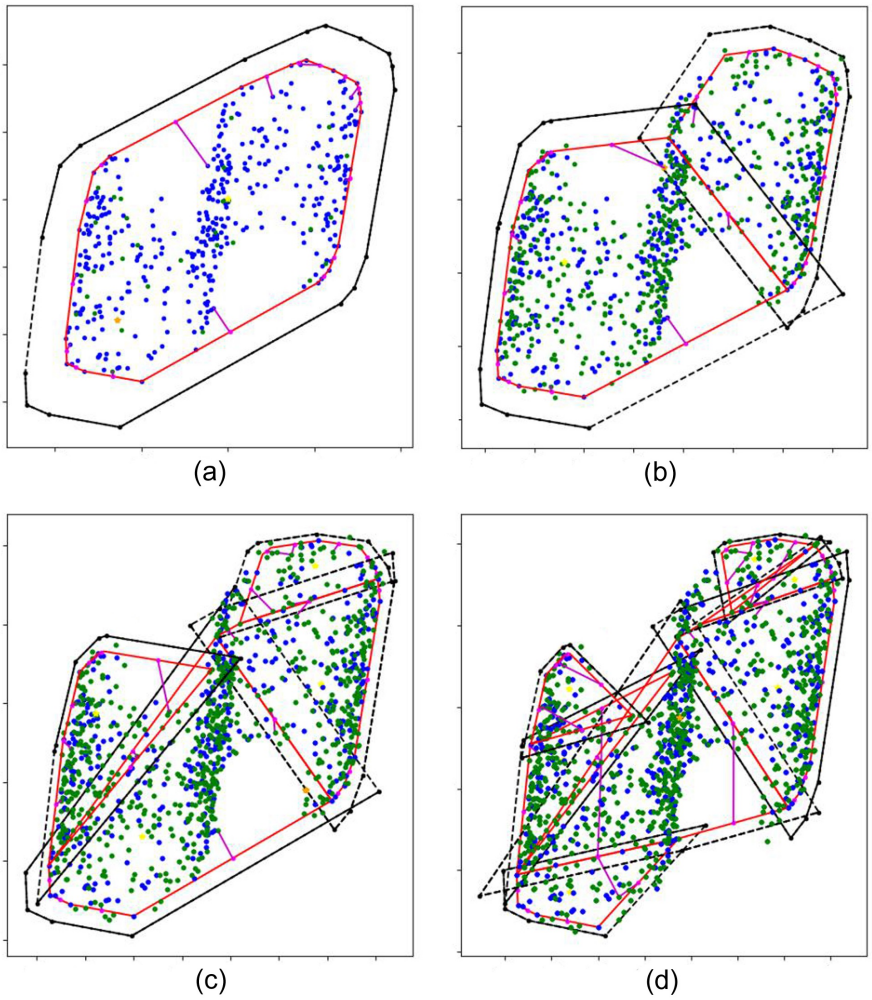

Figura 5.8: Reajuste progresivo del algoritmo sobre un conjunto con forma de *S*.

5.5.6 Microcalcificaciones en mamografías

En la Tabla 5.12 se muestran los resultados obtenidos por los diferentes métodos tras su ejecución sobre el conjunto de microcalcificaciones en mamografías. Los algoritmos que mejores resultados han obtenido en la métrica de similitud fueron *IF* y *LOF*, con valores muy similares. Este último, *LOF*, fue el algoritmo que obtuvo mejores resultados en sensibilidad. El método *Online-DSCH* fue el primero en especificidad y precisión, debido a que al no poder subdividirse trata prácticamente a todos los datos como normales. Este comportamiento indeseado se ve reflejado en la métrica de similitud, en la que obtiene un resultado deficiente. Nuestro algoritmo *OnlineS-DSCH* mejora respecto a su versión sin subdivisión, convirtiéndose a pesar de su baja potencia, en un modelo realista. Obtiene un rendimiento ligeramente

superior a *O-SVM*.

Como vemos, a diferencia de los generados artificialmente, nos encontramos ante un conjunto de datos cuya distribución de las anomalías en el espacio dificulta gravemente su modelado por la mayoría de las técnicas. Esto puede deberse a ruido existente en los datos y a la complejidad del problema, la cual dificulta una clara separación de la clase normal y anómala. A pesar de ello, los resultados obtenidos por *OnlineS-DSCH* no distan demasiado de los obtenidos por los otros métodos, obteniendo en la métrica de similitud un valor de 74,06%, y siendo 81,07% la máxima puntuación obtenida por el mejor de los algoritmos probados en dicha métrica.

Conjunto de datos	Sensibilidad	Especificidad	Precisión	Similitud
OnlineS-DSCH	70,77±4,2	78,39±2,3	77,85±3,3	74,06±3,2
Online-DSCH	21,92±24,6	**99,79±0,2**	**94,29±2,0**	44,64±12,3
O-SVM	60,19±8,4	90,53±2,1	88,39±2,7	70,66±3,5
RC	72,69±4,0	89,80±2,5	88,59±1,9	79,06±2,9
IF	87,31±2,8	75,66±3,2	76,48±2,3	**81,07±2,4**
LOF	**88,46±1,9**	74,73±3,2	75,69±2,6	80,96 ±3,2

Tabla 5.12: Resultados medios obtenidos y desviaciones típicas para el conjunto de datos real de microcalcificaciones en mamografías.

5.5.7 Fraudes en tarjetas de crédito

En la Tabla 5.13 se muestran los resultados obtenidos por los diferentes métodos tras su ejecución sobre el conjunto de fraudes en tarjetas de crédito. Los algoritmos que mejores resultados han obtenido en la métrica de similitud fueron *O-SVM*, *RC* y *LOF* con valores muy cercanos. *IF* fue el algoritmo que obtuvo mejores resultados en sensibilidad. El método *Online-DSCH* fue el primero en especificidad y precisión, debido a que al no poder subdividirse trata prácticamente a todos los datos como normales. Este comportamiento indeseado se ve reflejado en la métrica de similitud, en la que obtiene un resultado deficiente. Nuestro algoritmo *OnlineS-DSCH* mejora respecto a su versión sin subdivisión, obteniendo un rendimiento similar a los demás métodos en la métrica de similitud.

Conjunto de datos	Sensibilidad	Especificidad	Precisión	Similitud
OnlineS-DSCH	80,38±4,4	96,19±1,4	95,92±2,7	85,80±1,9
Online-DSCH	32,37±17,0	**99,89±0,3**	**98,80±1,5**	52,17±11,4
O-SVM	87,37±2,1	95,85±1,3	95,72±1,8	**90,54 ±2,8**
RC	89,87±1,5	89,93±1,0	89,92±1,1	89,89±2,3
IF	**91,25±2,7**	84,24±2,6	84,35±3,1	87,32±3,9
LOF	89,62±1,9	89,04±2,2	89,05±3,0	89,33±3,7

Tabla 5.13: Resultados medios obtenidos y desviaciones típicas para el conjunto de datos real de fraudes en tarjetas de crédito

A modo de conclusión, hemos decidido llevar a cabo una ordenación de los distintos algoritmos en función de sus similitudes medias, permitiéndonos así comparar su rendimiento de forma global. Esta similitudes medias se han calculado promediando las similitudes obtenidas por cada algoritmo sobre los diferentes conjuntos de datos, tanto reales como artificiales (un total de siete conjuntos). Estos resultados se presentan en la Tabla 5.14

Algoritmo	Similitud media
LOF	**92,43±5,9**
RC	92,01±6,6
O-SVM	90,91±8,8
OnlineS-DSCH	89,78±7,8
IF	88,62±6,3
Online-DSCH	69,21±19,1

Tabla 5.14: Similitud media y desviación típica.

Podemos concluir que gracias a la capacidad de subdivisión de los cierres (*OnlineS-DSCH*) hemos conseguido mejorar notablemente la primera versión del algoritmo (*Online-DSCH*), como se refleja en la similitud media. Además de esto, en comparación con los algoritmos clásicos, si bien *OnlineS-DSCH* ocupa la penúltima (cuarta) posición, si observamos su similitud media (89,78), esta solo dista en aproximadamente dos puntos y medio de la mejor (92,43). Por tanto, podemos concluir que hemos sido capaces de obtener un algoritmo con capacidad de aprendizaje en tiempo real sin que ello suponga una merma importante en su rendimiento en la detección de anomalías.

Capítulo 6

Conclusiones y trabajo futuro

PARA finalizar, en este capítulo se establecen las principales conclusiones obtenidas durante la realización de este proyecto de investigación, y la propuesta de diferentes líneas de trabajo futuro que podrían ser de interés.

6.1 Conclusiones

Tras la realización de este trabajo de fin de grado se han obtenido las siguientes conclusiones:

- El nuevo algoritmo desarrollado *OnlineS-DSCH* ha cumplido los objetivos establecidos en este proyecto de obtener un método de detección de anomalías con capacidad de aprendizaje online con una eficacia en las pruebas similar a varios de los métodos clásicos en detección de anomalías.
- Debido a su capacidad de subdivisión, se posiciona como una alternativa ante problemas con conjuntos de datos no convexos, en los que el algoritmo base de Diego Fernández Francos et al. no era apropiado.
- El comportamiento del algoritmo desarrollado es fácilmente configurable mediante sus parámetros. Una ventaja adicional frente a otras técnicas empleadas durante las pruebas es que no es necesario conocer *a priori* el porcentaje de datos anómalos existentes en el conjunto de datos. Este número de anomalías no siempre es una característica conocida del problema.
- Debido a que el algoritmo se basa en la geometría de los datos, a diferencia de otros métodos, las predicciones realizadas por *OnlineS-DSCH* son justificables.
- Igual que el algoritmo base de Diego Fernández Francos et al., *OnlineS-DSCH* es un método cuyos modelos de detección de anomalías son relativamente ligeros. No es necesa-

rio almacenar todos los puntos utilizados para la creación o adaptación de los cierres. El modelo únicamente necesita los vértices de cada cierre convexo para decidir si un nuevo se encuentra dentro de los límites de la clase normal. Esta capacidad para desechar u olvidar la información poco útil, convierte a *OnlineS-DSCH* en una técnica apropiada para problemas de monitorización. En estos problemas, los modelos no pueden ser demasiado pesados ni pueden crecer de forma lineal con el tiempo, a medida que procesan la información.

- Se ha comprobado que incrementando el número de proyecciones con las que trabaja el algoritmo, este es capaz de seguir obteniendo buenos resultados ante conjuntos de datos de mayor dimensionalidad, es decir, no se ve limitado por la dimensionalidad del problema. Por tanto, aunque el proceso de subdivisión, congelación y poda implementado en *OnlineS-DSCH* conlleve una carga computacional extra, los tiempos más largos de entrenamiento podrán compensarse mediante el uso distribuido del algoritmo.

- El modelo es paralelizable mediante la ejecución de cada una de las divisiones de las proyecciones en núcleos independientes. Además sigue siendo distribuible como la versión de Diego Fernández Francos et al.

6.2 Trabajo futuro

A continuación, se plantean diferentes líneas de trabajo futuro:

- Aunque la implementación del algoritmo *OnlineS-DSCH* llevada a cabo en *Python* ha sido optimizada mediante el uso de estructuras de datos especiales (por ejemplo, utilizando diccionarios en casos donde las operaciones de búsqueda son prioritarias), una forma de reducir aún más el tiempo invertido en el entrenamiento sería seguir optimizando dicho código. Una posible mejora podría ser el uso de la librería *Pandas* [49] para la manipulación de los datos.

- Para la realización de las pruebas de este trabajo no se ha hecho uso de la capacidad de cálculo distribuido del algoritmo *OnlineS-DSCH*, ya que nuestro objetivo fue mejorar su adaptabilidad ante conjuntos no convexos. Una de las ventajas principales del algoritmo es su capacidad de ejecución en diferentes nodos (uso distribuido), lo que mejora tanto el tiempo como la seguridad del sistema ya que se evita el envío directo de los datos. Durante este proyecto, el algoritmo se ha probado únicamente de forma local, por lo tanto, una posible línea de trabajo futuro es probar su funcionamiento de forma distribuida. Debido a la repartición de la carga de trabajo entre los nodos, se deberá estudiar si existe una mejora considerable en tiempo durante el entrenamiento.

- Una posible característica que no se ha llegado a implementar pero que podría mejorar la adaptabilidad del algoritmo, es la fusión de cierres convexos en las proyecciones. Este comportamiento consistiría en fusionar cierres que por solapamiento o por su forma, puedan ser representados por un único cierre convexo. Esta unión de cierres simplificaría la representación de las proyecciones, lo que conllevaría una disminución del número de comprobaciones a la hora de clasificar un nuevo dato (menor número de cierres), y por tanto, del tiempo de ejecución.

- Si se desea aplicar el algoritmo sobre conjuntos de datos no estacionarios, se podría implementar un factor de olvido en los vértices de los cierres. Este factor de olvido permite reforzar aquellos vértices más importantes para el cierre y castigar a los menos representativos. En lugar de trabajar únicamente con un único valor λ constante, cada vértice dispondrá del suyo. Cada vez que se procese un nuevo dato, los vértices más cercanos incrementarán su valor de λ, lo que expandirá el cierre en esa dirección, mientras que los valores λ de los vértices más alejados se verán decrementados, contrayendo así el cierre. Mediante este mecanismo, los cierres podrían desplazarse por el espacio de los datos adaptándose a variaciones en los mismos.

Apéndices

Apéndice A

Glosario de acrónimos

CH *Convex Hull.*

SCH *Scaled Convex Hull.*

OnlineS-DSCH *Online and Subdivisible Distributed Scaled Convex Hull.*

Online-DSCH *Online Distributed Scaled Convex Hull.*

O-SVM *One-class Support Vector Machines.*

LOF *Local Outlier Factor.*

RIC *Rango Intercuartílico.*

LIDIA *Laboratorio de Investigación y Desarrollo en Inteligencia Artificial.*

Bibliografía

[1] Lidia. [Online]. Available: https://www.lidiagroup.org/

[2] Amazon, "Amazon robotics," Accedido en 25-06-2019 a url https://www.amazonrobotics.com/, 2019.

[3] N. I. Nwulu and O. P. Agboola, "Modelling and predicting electricity consumption using artificial neural networks," in *2012 11th International Conference on Environment and Electrical Engineering*, May 2012, pp. 1059–1063.

[4] Huiying Li, Dechang Li, Changhai Zhang, and Shubin Nie, "An application of machine learning in the criterion updating of diagnosis cancer," in *2005 International Conference on Neural Networks and Brain*, vol. 1, Oct 2005, pp. 187–190.

[5] V. Këpuska and G. Bohouta, "Next-generation of virtual personal assistants (microsoft cortana, apple siri, amazon alexa and google home)," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan 2018, pp. 99–103.

[6] S. Thakre, A. K. Gupta, and S. Sharma, "Secure reliable multimodel biometric fingerprint and face recognition," in *2017 International Conference on Computer Communication and Informatics (ICCCI)*, Jan 2017, pp. 1–4.

[7] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, April 1997.

[8] G. van Rossum. (1991) Python web. [Online]. Available: https://www.python.org/

[9] T. Oliphant. (2005) Numpy web. [Online]. Available: https://numpy.org/

[10] D. Cournapeau. (2010) Scikit-learn web. [Online]. Available: https://scikit-learn.org/stable/#

[11] JetBrains. (2010) Pycharm web. [Online]. Available: https://www.jetbrains.com/pycharm/

[12] B. H. Anthony Goldbloom. (2010) Kaggle web. [Online]. Available: https://www.kaggle.com/

[13] I. of Electrical, E. Engineers, I. of Engineering, and Technology. Ieee xplore web. [Online]. Available: https://ieeexplore.ieee.org/Xplore/home.jsp

[14] J. Spolsky. (2011) Trello web. [Online]. Available: https://trello.com/

[15] P. H. Tom Preston-Werner, Chris Wanstrath. (2008) Github web. [Online]. Available: https://github.com

[16] Google. (2012) Google drive web. [Online]. Available: https://drive.google.com/drive/

[17] P. Brachet. (2003) Texmaker web. [Online]. Available: https://www.xm1math.net/texmaker/

[18] Salario medio desarrollador software en españa. [Online]. Available: https://www.payscale.com/research/ES/Job=Software_Developer/Salary

[19] D. Khandelwal, D. Shanbhag, A. Shriyan, R. Thorve, and Y. Borse, "Lemeno: Personalised news using machine learning," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, Aug 2018, pp. 1–4.

[20] M. T. Islam, B. M. N. Karim Siddique, S. Rahman, and T. Jabid, "Food image classification with convolutional neural network," in *2018 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, vol. 3, Oct 2018, pp. 257–262.

[21] T. Tongloy, S. Chuwongin, K. Jaksukam, C. Chousangsuntorn, and S. Boonsang, "Asynchronous deep reinforcement learning for the mobile robot navigation with supervised auxiliary tasks," in *2017 2nd International Conference on Robotics and Automation Engineering (ICRAE)*, Dec 2017, pp. 68–72.

[22] L. Wu and M. Li, "Applying the cg-logistic regression method to predict the customer churn problem," in *2018 5th International Conference on Industrial Economics System and Industrial Security Engineering (IEIS)*, Aug 2018, pp. 1–5.

[23] Y. Tan, L. Tan, G. Yun, and W. Zheng, "Bilevel genetic algorithm with clustering for large scale traveling salesman problems," in *2016 International Conference on Information System and Artificial Intelligence (ISAI)*, June 2016, pp. 365–369.

[24] R. Chutia and M. Bhuyan, "Online concentration independent feature dimension reduction of metal oxide gas sensor based e-nose," in *IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012)*, March 2012, pp. 247–250.

[25] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, pp. 3448 – 3470, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S138912860700062X

[26] G. Ratsch, S. Mika, B. Scholkopf, and K. . Muller, "Constructing boosting algorithms from svms: an application to one-class classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1184–1199, Sep. 2002.

[27] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1541880.1541882

[28] M. A. F. Pimentel, D. A. Clifton, L. A. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

[29] D.-Y. Yeung and C. Chow, "Parzen-window network intrusion detectors," in *In Proceedings of the Sixteenth International Conference on Pattern Recognition*, 2002, pp. 385–388.

[30] P. Soucy and G. W. Mineau, "A simple knn algorithm for text categorization," in *Proceedings 2001 IEEE International Conference on Data Mining*, Nov 2001, pp. 647–648.

[31] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," *SIGMOD Rec.*, vol. 29, no. 2, pp. 93–104, May 2000. [Online]. Available: http://doi.acm.org/10.1145/335191.335388

[32] G. Williams, R. Baxter, Hongxing He, S. Hawkins, and Lifang Gu, "A comparative study of rnn for outlier detection in data mining," in *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, Dec 2002, pp. 709–712.

[33] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of Educational Psychology*, vol. 24, no. 6, pp. 417–441, 1933.

[34] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep 1995. [Online]. Available: https://doi.org/10.1023/A:1022627411411

[35] D. M. Tax and R. P. Duin, "Support vector data description," *Machine Learning*, vol. 54, no. 1, pp. 45–66, Jan 2004. [Online]. Available: https://doi.org/10.1023/B:MACH.0000008084.60811.49

[36] E. J. Keogh, J. D. Lin, and A. W.-C. Fu, "Hot sax: efficiently finding the most unusual time series subsequence," *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pp. 8 pp.–, 2005.

[37] D. Fernández-Francos, . Fontenla-Romero, and A. Alonso-Betanzos, "One-class convex hull-based algorithm for classification in distributed environments," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–11, 2018.

[38] P. Casale, O. Pujol, and P. Radeva, "Approximate polytope ensemble for one-class classification," *Pattern Recognition*, vol. 47, no. 2, pp. 854 – 864, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320313003282

[39] W. B. Johnson and J. Lindenstrauss, "Extensions of lipschitz mappings into a hilbert space," *Contemporary mathematics*, vol. 26, no. 189-206, pp. 1 – 1, 1984.

[40] C. B. Barber, D. P. Dobkin, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Trans. Math. Softw.*, vol. 22, no. 4, pp. 469–483, Dec. 1996. [Online]. Available: http://doi.acm.org/10.1145/235815.235821

[41] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. Berlin, Heidelberg: Springer-Verlag, 1985.

[42] G. Quellec, M. Lamard, M. Cozic, G. Coatrieux, and G. Cazuguel, "Multiple-instance learning for anomaly detection in digital mammography," *IEEE Transactions on Medical Imaging*, vol. 35, no. 7, pp. 1604–1614, July 2016.

[43] H. Al-Thani, H. Hassen, S. Al-Maadced, N. Fetais, and A. Jaoua, "Unsupervised technique for anomaly detection in qatar stock market," in *2018 International Conference on Computer and Applications (ICCA)*, Aug 2018, pp. 116–9.

[44] K. S. Woods, C. C. Doss, K. W. Bowyer, J. L. Solka, C. E. Priebe, and W. P. K. JR, "Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 6, pp. 1417–1436, 1993.

[45] Credit card fraud detection. [Online]. Available: https://www.kaggle.com/dileep070/anomaly-detection/

[46] Robust covariance api web. [Online]. Available: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.covariance

[47] F. T. Liu, K. M. Ting, and Z. hua Zhou, "Isolation forest," in *In ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining. IEEE Computer Society*, pp. 413–422.

[48] M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," in *PROCEEDINGS OF THE 2000 ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA*. ACM, 2000, pp. 93–104.

[49] W. McKinney. (2008) Pandas. [Online]. Available: https://pandas.pydata.org/