UNIVERSIDAD
DE GRANADA

E.T.S. Ingeniería Informática y de Telecomunicacíon
Departamento de Ciencias de la Computación e Inteligencia Artificial - Instituto Andaluz Interuniversitario en Ciencia de Datos e Inteligencia Computacional (Instituto DaSCI)
Programa de Doctorado en Tecnologías de la Información y la Comunicación

Tesis Doctoral

Una actualización del algoritmo de aprendizaje de reglas difusas de Wang y Mendel para problemas de clasificación con datos masivos

Leonardo Alejandro Jara Barrales

Directores Dr. Antonio González Muñoz
Dr. Francisco G. Raúl Pérez Rodríguez

5 de octubre de 2023

Editor: Universidad de Granada. Tesis Doctorales
Autor: Leonardo Alejandro Jara Barrales
ISBN: 978-84-1195-134-0
URI: https://hdl.handle.net/10481/89155

Leonardo Alejandro Jara Barrales

Una actualización del algoritmo de aprendizaje de reglas difusas de Wang y Mendel para problemas de clasificación con datos masivos

Tesis Doctoral, 5 de octubre de 2023

Directores:
Dr. Antonio González Muñoz
Dr. Francisco G. Raúl Pérez Rodríguez

Universidad de Granada
Programa de Doctorado en Tecnologías de la Información y la Comunicación
Departamento de Ciencias de la Computación e Inteligencia Artificial - Instituto Andaluz Interuniversitario en Ciencia de Datos e Inteligencia Computacional (Instituto DaSCI)
E.T.S. Ingeniería Informática y de Telecomunicacíon
Calle Periodista Manuel Saucedo Aranda
18071, Granada

Abstract

In recent decades, society has witnessed an unprecedented technological transformation. This period has been characterized by the widespread adoption of the Internet, the massive proliferation of mobile devices, and an astonishing increase in data generation. In this context, data analysis has emerged as one of the fastest-growing fields. Specifically, the analysis of massive data, part of what is known as Big Data, has become a fundamental approach to extract knowledge from human behavior and their environment. As a result, many organizations, both in the business and government sectors, have opted to employ these technologies to harness the immense potential of their data and extract valuable insights.

However, this abundance of information presents a significant challenge. While this vast amount of data has the potential to significantly improve the accuracy of data mining algorithms, traditional approaches in this field are ill-equipped to handle the speed and volume requirements that Big Data demands. Therefore, it is necessary to develop new techniques to address these issues and enable effective analysis of this massive data.

In this context, the present thesis focuses on the challenge of extracting meaningful knowledge from these vast volumes of data, with a primary emphasis on machine learning. This discipline, which is part of artificial intelligence, empowers machines to acquire knowledge directly from data and make autonomous decisions without human intervention. Machine learning relies on statistical techniques and algorithms designed to analyze data and reveal underlying patterns. Its applications are diverse, ranging from fraud detection to medical diagnosis. In this research, particular emphasis is placed on supervised learning for classification, a scenario in which labels are assigned to data to categorize them accurately and effectively.

A rule-based learning model grounded in fuzzy logic takes center stage in this research. These models specialize in handling the uncertainty and ambiguity inherent in data. As a starting point, the Wang and Mendel algorithm (WM) is selected, recognized for its simplicity and efficiency when dealing with massive data, albeit accompanied by certain limitations in terms of accuracy and interpretability.

The primary objective of this thesis is to significantly enhance the performance of the WM algorithm, especially when dealing with massive datasets. Firstly, the Quine

McCluskey method is implemented to minimize and simplify the fuzzy rule base, contributing to greater interpretability without sacrificing model accuracy.

Furthermore, the issue of efficiency in the inference process is addressed, particularly in situations involving extensive rule sets, through the incorporation of efficient search techniques and approximate approaches.

Another important enhancement is the introduction of an efficient approach for calculating rule weights. This approach calculates the weight based on the context of the analyzed rule, rather than considering the entire set of generated rules, resulting in a significant reduction in the time required to build the rule base without affecting accuracy.

Finally, an innovative approach focusing on the handling of redundant rules is presented to improve generalization capability and, consequently, in reducing the percentage of uncovered examples and increasing the accuracy of the WM algorithm.

All these contributions are documented in various articles and presentations at scientific conferences, providing detailed descriptions of these improvements and their impact on the overall performance of the WM algorithm. Experimental results strongly support that the proposed methods have significantly improved the performance of this algorithm, not only in terms of classification accuracy but also in interpretability and efficiency of the model.

Resumen

En las últimas décadas, la sociedad ha sido testigo de una transformación tecnológica sin precedentes. Este período se ha caracterizado por la generalización de Internet, la difusión masiva de dispositivos móviles y un asombroso incremento en la generación de datos. En este marco, el análisis de datos ha surgido como uno de los campos de mayor crecimiento. Específicamente, el análisis de datos masivos, una parte de lo que se conoce como Big Data, se ha convertido en un enfoque fundamental para obtener conocimiento a partir del comportamiento humano y su entorno. Como resultado, muchas organizaciones, tanto empresariales como gubernamentales, han optado por emplear estas tecnologías con el fin de aprovechar al máximo su inmenso potencial de datos y extraer valiosa información de los mismos.

Sin embargo, esta abundancia de información plantea un desafío considerable. Aunque esta gran cantidad de datos tiene el potencial de mejorar significativamente la precisión de los algoritmos de minería de datos, los enfoques tradicionales en este campo no están preparados para lidiar con los requisitos de velocidad y volumen que Big Data impone. Por lo tanto, se hace necesario desarrollar nuevas técnicas que aborden estos problemas y permitan un análisis efectivo de estos datos masivos.

En este contexto, la presente tesis se enfoca en el desafío de extraer conocimiento significativo de estos vastos volúmenes de datos, con un énfasis principal en el aprendizaje automático. Esta disciplina, que forma parte de la inteligencia artificial, capacita a las máquinas para adquirir conocimientos directamente a partir de los datos y tomar decisiones autónomas sin intervención humana. El aprendizaje automático se basa en técnicas estadísticas y algoritmos diseñados para analizar datos y revelar patrones subyacentes. Sus aplicaciones son diversas y van desde la detección de fraudes hasta el diagnóstico médico. En esta investigación, se pone un énfasis particular en el aprendizaje supervisado para la clasificación, un escenario en el cual se asignan etiquetas a los datos con el fin de categorizarlos de manera precisa y efectiva.

Un modelo de aprendizaje basado en reglas y fundamentado en la lógica difusa se erige como el protagonista de esta investigación. Estos modelos se especializan en tratar con la incertidumbre y la ambigüedad inherentes a los datos. Como punto de partida, se selecciona el algoritmo de Wang y Mendel (*WM*), reconocido por

su simplicidad y eficiencia al trabajar con datos masivos, aunque acompañado de ciertas limitaciones en cuanto a precisión e interpretabilidad.

El objetivo principal de esta tesis es mejorar sustancialmente el rendimiento del algoritmo *WM*, especialmente cuando se enfrenta a conjuntos de datos masivos. En primer lugar, se implementa el método de Quine McCluskey para minimizar y simplificar la base de reglas difusas, lo que contribuye a una mayor interpretabilidad sin sacrificar la precisión del modelo.

Además, se aborda la cuestión de la eficiencia en el proceso de inferencia, particularmente en situaciones que involucran extensos conjuntos de reglas, mediante la incorporación de técnicas de búsqueda eficientes y enfoques aproximados.

Otra mejora importante es la introducción de una versión eficiente en el cálculo del peso de las reglas. Este enfoque calcula el peso basándose en el entorno de la regla analizada, en lugar de considerar todo el conjunto de reglas generadas, lo que resulta en una reducción significativa en el tiempo requerido para construir la base de reglas, sin afectar la tasa de aciertos.

Finalmente, se presenta un enfoque innovador que se centra en el manejo de reglas redundantes para mejorar la capacidad de generalización y, por ende, en reducir el porcentaje de ejemplos no cubiertos y aumentar la precisión del algoritmo *WM*.

Todas estas contribuciones se encuentran documentadas en diversos artículos y presentaciones en congresos científicos, donde se describen en detalle estas mejoras y su impacto en el rendimiento general del algoritmo *WM*. Los resultados experimentales respaldan de manera sólida que los métodos propuestos han mejorado significativamente el rendimiento de este algoritmo, no solo en términos de la tasa de clasificación, sino también en lo que respecta a la interpretabilidad y eficiencia del modelo.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas e instituciones que han hecho posible la culminación de este trabajo de tesis doctoral. Representa el resultado de varios años de arduo trabajo y dedicación, y no habría sido posible sin su invaluable apoyo y sabios consejos.

En primer lugar, quiero extender mi profundo agradecimiento a mis padres, Leonardo y Ana María. Por guiarme por el camino correcto desde el inicio y por inculcarme los valores que me han orientado a lo largo de este viaje. A pesar de la distancia, su apoyo y su confianza en el desafío que implica estudiar lejos de casa han sido fundamentales en mi camino hacia la obtención de este título.

A mis queridas hermanas, quienes me han respaldado en todos los momentos de mi vida y han contribuido a moldear la persona que soy hoy. Son seres increíbles y especiales con quienes he crecido y evolucionado.

A Milou, mi apoyo emocional incondicional en los momentos felices y complicados que atravesé durante este prolongado período de estudios. Su comprensión y apoyo continuo fueron fundamentales para seguir avanzando.

A Antonio González y Raúl Pérez, mis directores de tesis, les agradezco por brindarme la oportunidad de unirme a su grupo de investigación. Ellos me han guiado a lo largo de estos cinco años con su orientación académica y sus consejos, demostrando en todo momento una gran disposición hacia mi desarrollo académico. Les tengo un profundo respeto y admiración.

A Diego, un amigo que me abrió las puertas de su hogar y su familia, cuando no me conocía y se convirtió en un guía fundamental en mi trayecto hacia el doctorado, que finalmente logramos completar con éxito.

Agradezco al Programa de Becas de Postgrado Çarlos Antonio López"del Gobierno de Paraguay por la oportunidad brindada, que me permitió crecer tanto personal como académicamente.

Finalmente, mi gratitud se extiende a todas las personas que conocí a lo largo de estos años. Son grandes amigos que me brindó la hermosa ciudad de Granada y que

se convirtieron en mi sostén diario, haciéndome sentir como en casa a pesar de estar lejos de mi país.

A todos ustedes, les agradezco de corazón.

Índice general

Índice de figuras

Índice de tablas

1 Introducción

En las últimas décadas, se ha presenciado un avance tecnológico acelerado que ha generado una transformación radical en nuestra forma de vida. Desde la aparición de Internet hasta el auge de los dispositivos móviles y la conectividad ubicua, la tecnología ha permeado todos los aspectos de la sociedad. A principios de 2023, se pronosticaba que casi dos tercios de la población mundial tendrían acceso a Internet, lo que representa aproximadamente 5300 millones de usuarios, equivalente al 66 % de la población global [16]. Uno de los resultados más destacados de este avance tecnológico ha sido el crecimiento exponencial en la generación y almacenamiento de datos.

Cada día, se generan enormes cantidades de datos a través de diversas fuentes, como redes sociales, transacciones comerciales, dispositivos inteligentes, sensores y mucho más. Este incremento masivo en los datos, conocido como explosión de datos o Big Data, ha creado nuevos desafíos y oportunidades para las organizaciones y la sociedad en general. La Figura 1.1 muestra la evolución prevista del tamaño de los datos en los próximos años [101].

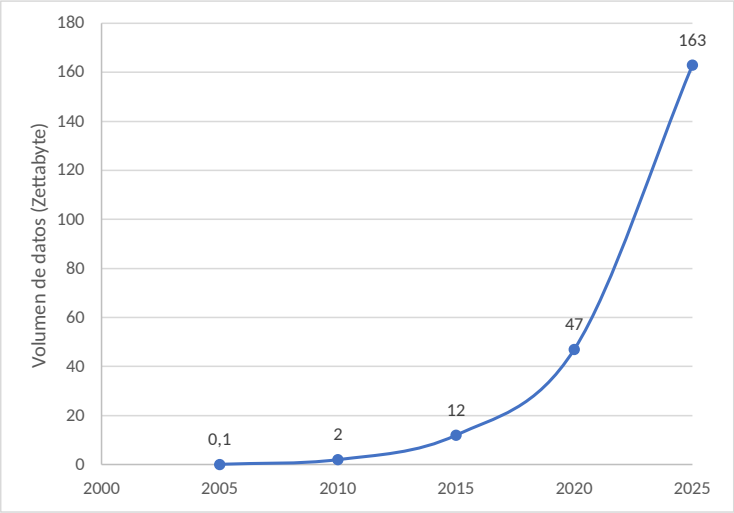

Fig. 1.1.: Aumento previsto del tamaño de los datos en los próximos años.

A medida que la cantidad de datos ha aumentado, también lo ha hecho la necesidad de extraer información útil y conocimiento significativo de ellos. Aquí es donde entra

en juego el aprendizaje automático, una rama de la inteligencia artificial que se enfoca en desarrollar algoritmos y modelos que permiten a las máquinas aprender de los datos y tomar decisiones o realizar tareas sin intervención humana directa.

El aprendizaje automático utiliza técnicas estadísticas y algoritmos de inteligencia artificial para analizar conjuntos de datos y descubrir patrones, tendencias y correlaciones ocultas. Estos modelos pueden aplicarse en una amplia gama de aplicaciones, desde la detección de fraudes y la recomendación de productos hasta el diagnóstico médico y la conducción autónoma como se observa en la Figura 1.2 [16].

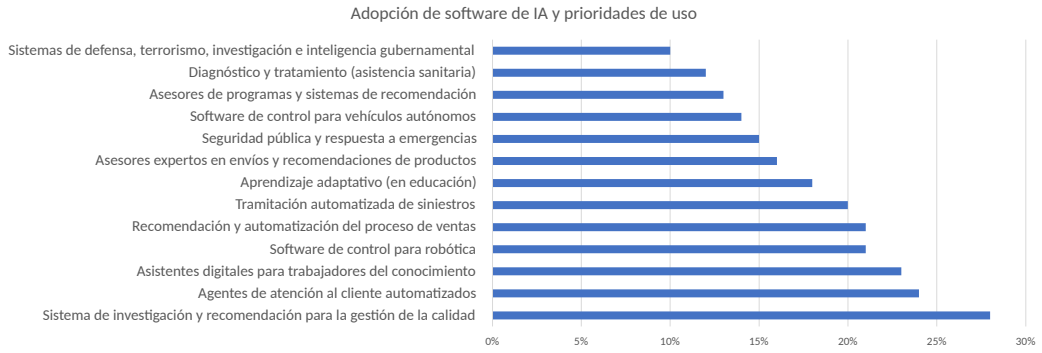

Fig. 1.2.: Adopción de la inteligencia artificial y prioridades de uso.

El avance de la tecnología y el incremento de datos han impulsado la adopción generalizada del aprendizaje automático en diversas industrias y campos. Las organizaciones están utilizando estas técnicas para obtener información valiosa sobre sus clientes, mejorar la eficiencia operativa, optimizar procesos y tomar decisiones más informadas y basadas en datos.

El aprendizaje automático puede dividirse en aprendizaje supervisado, aprendizaje no supervisado y aprendizaje semi-supervisado, dependiendo de si el conjunto de datos tiene etiquetas. El problema de clasificación pertenece a la categoría del aprendizaje supervisado y los valores de las etiquetas son valores discretos.

En la actualidad, se encuentra disponible una amplia gama de modelos de aprendizaje supervisado utilizados para la clasificación, destacando diversos algoritmos clásicos que se pueden consultar en la literatura científica. Entre estos se encuentran los árboles de decisión [77, 23], como el C4.5 [82], los cuales usan estructuras de árboles que realizan decisiones basadas en la información de entrada. Asimismo, se encuentran modelos de aprendizaje basados en redes neuronales [100, 102], que son modelos de aprendizaje profundo compuestos por múltiples capas de neuronas interconectadas. Otro enfoque es el uso de redes bayesianas [45], que constituyen un modelo gráfico para representar las relaciones de probabilidad entre un conjunto

de variables. Además, el aprendizaje basado en instancias mediante el algoritmo KNN [22] clasifica nuevos puntos de datos basándose en las clases de los vecinos más cercanos en el espacio de características. Las máquinas de vectores de soporte (SVM) [8] buscan encontrar un hiperplano óptimo que maximice el margen de separación entre las clases en un espacio de alta dimensión. Por último, el aprendizaje de reglas, ejemplificado por RIPPER [18], se asemeja a los árboles de decisión, pero también permite la inducción directa de reglas a partir de los datos de entrenamiento mediante técnicas específicas.

Un modelo específico de aprendizaje basado en reglas utiliza la lógica difusa como base fundamental, lo que permite adquirir conocimiento en situaciones donde los datos de entrada presentan algún tipo de imprecisión. Estas reglas, conocidas como reglas difusas, representan un enfoque interesante para abordar problemas que involucran incertidumbre y ambigüedad en los datos. En los últimos años, este campo ha experimentado un notable crecimiento con la introducción de varios modelos destacados.

Entre estos modelos se encuentra SLAVE [40] y su evolución NSLV [38], los cuales emplean un algoritmo genético y el modelo iterativo para obtener un conjunto de reglas difusas que representen un buen compromiso entre acierto e interpretabilidad. Otro modelo relevante es FURIA [46], el cual representa una extensión del algoritmo RIPPER que permite obtener reglas difusas. Además, se destaca FARC-HD [2], que se enfoca en abordar problemas de alta dimensionalidad. Por otro lado, ANFIS [51] utiliza un enfoque de aprendizaje híbrido que combina reglas difusas y redes neuronales para mejorar la capacidad de modelado y clasificación.

En el contexto de esta investigación, el algoritmo de Wang y Mendel para clasificación [14], adquiere especial relevancia. No puede calificarse como el mejor desde muchos puntos de vista en comparación con los algoritmos expuestos anteriormente, sin embargo, se ha utilizado frecuentemente en varias aplicaciones como [15, 97, 75, 31, 11] o se han propuesto o utilizado múltiples versiones del mismo para comparar con nuevas propuestas [60, 94, 41, 57, 99, 95]. Entre las razones de este uso frecuente, podemos destacar su simplicidad (se trata básicamente de una asignación de ejemplos a regiones difusas) y su eficiencia (ya que, por su descripción, utilizando una implementación eficiente, el tiempo de aprendizaje es lineal en el número de ejemplos). Por estos motivos, el algoritmo de Wang y Mendel será objeto de un análisis detallado en el marco de esta tesis doctoral, ya que servirá como base fundamental para el desarrollo de la investigación. El estudio exhaustivo de este algoritmo permitirá comprender su funcionamiento, identificar sus fortalezas y

debilidades, y explorar posibles mejoras y aplicaciones en el campo de las reglas difusas y el aprendizaje de clasificación.

1.1. Hipótesis y justificación del estudio

El algoritmo de Wang y Mendel (*WM*) [92] fue originalmente utilizado tanto para problemas de regresión como de clasificación, posteriormente en una revisión del algoritmo propuesta en [14] (y que llamaremos *WM-C*) se modificó el algoritmo para afrontar problemas específicos de clasificación, siendo éste uno de los primeros en ser utilizados en el campo de los conjuntos difusos para clasificación. Aunque se caracteriza por su simplicidad y la obtención rápida de resultados aceptables, su prominencia ha disminuido con la aparición de nuevos algoritmos, como los ya mencionados SLAVE, NSLV, FURIA y FARC-HD, aunque sigue siendo un algoritmo muy utilizado para comparar resultados y para aplicarlo en ciertos problemas.

El objetivo de este estudio de tesis consiste en realizar una investigación exhaustiva sobre las limitaciones del algoritmo *WM-C* y proponer una revisión que permita mejorar su rendimiento en clasificación. Uno de los aspectos destacados de este algoritmo es su eficiencia [3], especialmente cuando se aplica a conjuntos de datos masivos [27], lo que lo distingue de otros algoritmos previamente mencionados. Por lo tanto, nos centraremos especialmente en evaluar la utilidad del algoritmo en la resolución de problemas que involucren grandes volúmenes de datos.

Es relevante destacar que el algoritmo *WM-C* exhibe ciertas limitaciones. En primer lugar, produce un elevado número de reglas como resultado de su proceso de aprendizaje, lo cual puede dificultar su interpretabilidad y su aplicabilidad en contextos prácticos. Además, se ha constatado que su precisión no alcanza niveles suficientemente altos en comparación con otros algoritmos de clasificación. Por último, el cálculo del peso de las reglas resulta ser altamente costoso, especialmente cuando se trata de un gran número de reglas. Por lo tanto, esta investigación se enfocará en mejorar el algoritmo *WM-C* en los siguientes aspectos, que se detallan a continuación:

1. Incluir métodos externos al algoritmo que permitan reducir la cantidad de reglas y/o mejorar la interpretabilidad de las reglas obtenidas por el algoritmo.

2. Proponer un método de inferencia eficiente sobre clasificadores con un número elevado de reglas difusas, debido a que el algoritmo de *WM-C* genera una gran cantidad de reglas cuando existe un elevado número de ejemplos.

3. Modificar la propia estructura del algoritmo de aprendizaje, con la idea de mejorar la precisión y el tiempo necesario para aprender el modelo.

La hipótesis planteada en este estudio sostiene que el algoritmo *WM-C* puede mejorar su rendimiento al ser aplicado en problemas que involucren conjuntos de datos masivos. Se postula que esta mejora se logrará mediante la modificación de varios aspectos clave del algoritmo, que incluyen su estructura de aprendizaje, su modelo de inferencia y la revisión de los resultados generados por el algoritmo. El objetivo de estas modificaciones es lograr un aumento en la precisión y la interpretabilidad, disminuyendo el tiempo de aprendizaje e inferencia del algoritmo en el contexto de problemas con grandes volúmenes de datos. La investigación se enfocará en explorar cómo estos aspectos pueden ser mejorados para alcanzar dicho objetivo.

Por consiguiente, el primer objetivo que nos planteamos es abordar la mejora de la interpretabilidad de las reglas en el algoritmo de estudio. Una estrategia que se considerará es emplear técnicas de minimización de bases de reglas difusas. Mediante la aplicación de la minimización lógica, se busca reducir un conjunto de reglas extenso a otro que utilice menos variables y/o reglas, manteniendo en cierta medida su equivalencia. Para lograr este propósito, se trabajará con el algoritmo de Quine McCluskey [79]. Se espera que, al aplicar este algoritmo, se pueda simplificar y compactar el conjunto de reglas, lo cual facilitará su interpretación y comprensión por parte de los usuarios.

Además, se abordará el desafío de la inferencia en sistemas que contienen un número elevado de reglas. Este estudio no está ligado directamente al algoritmo, pero es perfectamente adecuado para las bases de reglas obtenidas por el mismo. Para ello, se propondrán procesos de cálculo más eficientes con el fin de resolver este problema y mejorar el tiempo de respuesta. La idea principal es reemplazar el modelo de inferencia actual, que implica una verificación regla por regla para determinar la regla ganadora, por un modelo que identifique inicialmente la regla más relevante para un ejemplo dado y a partir de dicha regla, realice un proceso de búsqueda local sobre el entorno de la regla más relevante. Asimismo, se explorarán técnicas aproximadas que, si bien no siempre logran identificar la regla ganadora, pueden ofrecer resultados satisfactorios en un tiempo razonable. Estas técnicas aproximadas podrían proporcionar una alternativa viable cuando la exactitud absoluta no sea primordial y se priorice una respuesta más rápida.

Adicionalmente, se afrontará el desafío del cálculo de pesos. Se propondrá un enfoque de cálculo más eficiente con el objetivo de evitar el recorrido completo todos los ejemplos durante este proceso. Se introducirá una estrategia de reducción

heurística de los ejemplos necesarios para dicho cálculo, con el fin de reducir significativamente el costo asociado al cálculo de pesos.

Por último, se propondrá una técnica basada en seleccionar más de una regla en la vecindad de la regla ganadora, con el objetivo de mejorar la capacidad de generalización del algoritmo. Mediante esta técnica, se espera obtener mejores niveles de precisión al considerar las reglas cercanas a la regla ganadora, lo cual permitirá capturar patrones más amplios y tener en cuenta mas información en el proceso de clasificación.

1.2. Objetivos

Este tesis doctoral tiene como objetivo desarrollar y evaluar diferentes procedimientos que permitan mejorar en algoritmo de *WM-C*, en aspectos claves como la interpretabilidad de la base de reglas, la eficiencia del método de inferencia y mejorar la capacidad de generalización del mismo. A continuación, se detallan los principales objetivos considerados en esta tesis:

Objetivo General:

- Analizar el algoritmo de clasificación de *WM-C* y detectar los problemas que ocasiona su aplicación sobre datos masivos, proponiendo nuevas técnicas en las distintas etapas del modelo de aprendizaje e inferencia.

Objetivos Específicos:

1. Mejorar la interpretabilidad de la Base de Reglas obtenidas por el algoritmo de *WM-C*, con técnicas de simplificación y reducción del costo computacional del algoritmo, utilizando técnicas de minimización, por ejemplo, el algoritmo de Quine McCluskey utilizando diferentes codificaciones.

2. Mejorar el tiempo de respuesta y rendimiento del proceso de inferencia cuando el número de reglas es elevado, proponiendo nuevas técnicas.

3. Proponer mejoras en el rendimiento del algoritmo de *WM-C* mediante una mejora en el proceso de generación de reglas y cálculo de pesos.

1.3. Metodología

Basándonos en el método científico habitual, se propone la siguiente metodología que se adapta a nuestras necesidades específicas:

1. Observación: Identificación y comprensión de los problemas a abordar, interpretabilidad, eficiencia, rendimiento.
2. Formulación de la hipótesis: Mejorar el comportamiento del algoritmo de *WM-C*.
3. Recogida de observaciones: Obtención de resultados como consecuencia del desarrollo de nuevos métodos.
4. Contraste de hipótesis: Comparación de los resultados obtenidos frente a aquellos publicados hasta el momento.
5. Demostración o refutación de hipótesis: Aceptación, rechazo o modificación de métodos diseñados.
6. Tesis o teoría científica: Extracción, aceptación y redacción de las conclusiones durante todo el proceso realizado.

1.4. Contribuciones

- *A preliminary study to apply the Quine McCluskey algorithm for fuzzy rule base minimization.* (FUZZ-IEEE 2020) [54]

  Esta ponencia está dedicada a estudiar un método que permita minimizar la base de reglas que es generada por el Algoritmo Chi (*WM-C*), utilizando el método de Quine McCluskey de forma que se pueda reducir el número de reglas generadas, sin alterar en gran medida la precisión, mejorando así la simplicidad del modelo.

- *Una versión ternaria del algoritmo de Quine McCluskey para la minimización de base de reglas difusas.* (CAEPIA 20/21) [56]

  El objetivo de este ponencia consiste en mejorar la interpretabilidad *FRBS-C*, utilizando procesos de minimización de conjuntos de reglas difusas basados en una modificación del método de Quine McCluskey, que permite trabajar con valores ternarios. De este modo, se pretende mejorar la capacidad de

eliminación de variables irrelevantes en los antecedentes de las reglas y por lo tanto, mejorar la interpretabilidad del modelo sin alterar en gran medida la precisión.

- *Efficient inference models for classification problems with a high number of fuzzy rules* Applied Soft Computing (Q1) 2022. [52]

  En este artículo se exploran nuevas versiones para implementar el método de inferencia, evitando analizar todas las reglas y centrando el análisis en la vecindad de reglas alrededor del ejemplo. Se estudian experimentalmente las condiciones en las que debe aplicarse cada una de ellas. Finalmente, se propone una implementación que combina todas las versiones estudiadas ofreciendo buenos resultados de precisión y una reducción significativa del tiempo de respuesta.

- *A new multi-rules approach to improve the performance of the Chi fuzzy rule classification algorithm.* (FUZZ-IEEE 2022) [53]

  En esta ponencia se presenta una nueva versión del algoritmo *WM-C*. La principal diferencia radica en el proceso de asignación de una regla candidata a cada ejemplo, el algoritmo original asigna sólo una, y en la nueva propuesta se asignan varias reglas. La inclusión de más reglas mejora la capacidad de generalización del conocimiento adquirido, lo que a su vez redunda en una mejora de la precisión. El artículo incluye un amplio estudio experimental para mostrar el rendimiento de la propuesta.

- *Una versión alternativa del algoritmo de Chi, de aprendizaje de reglas difusas para clasificación, basada en la obtención de múltiples reglas sobre cada ejemplo.* (ESTYLF 2022) [55]

  En esta ponencia se plantea una modificación del algoritmo *WM-C* basada en la idea de añadir más de una regla por cada ejemplo. Se proponen tres métodos alternativos y se experimenta sobre distintos conjuntos de ejemplos para estudiar en qué casos es mejor la propuesta presentada.

- *QChi: a faster classification algorithm based on Wang-Mendel algorithm. (Sometido y en tercera revisión)*

  En el presente artículo se introduce *QChi*, una innovadora propuesta que modifica el algoritmo de Wang-Mendel (*WM-C*) para calcular los pesos de las reglas de manera computacionalmente más eficiente, basándose en enfoques de aproximación. Esta optimización produce una reducción significativa en el tiempo de aprendizaje. Además de presentar en detalle la metodología

de *QChi*, este trabajo incluye una rigurosa evaluación experimental en una amplia gama de problemas, algunos de los cuales son considerados desafiantes problemas de Big Data por otros investigadores. Los resultados experimentales confirman que *QChi* logra una drástica reducción en el tiempo de aprendizaje sin utilizar estructuras de procesamiento en paralelo y sin comprometer el rendimiento de la clasificación, consolidando su relevancia en aplicaciones de procesamiento eficiente de datos masivos.

- *Study of the effect of using redundant fuzzy rules in classification problems. (En elaboración)*

  El artículo en cuestión se centra en un estudio exhaustivo de cómo la introducción de redundancia, mediante la inclusión de múltiples reglas para respaldar cada ejemplo, influye de manera positiva en la capacidad de clasificación. Se exploran tres enfoques distintos para generar esta redundancia en un algoritmo de clasificación basado en la propuesta original de Wang y Mendel, pero con una mayor eficiencia computacional, gracias a la introducción de una aproximación rápida para calcular el peso de las reglas (*QChi*). El estudio se respalda en una extensa investigación experimental que evidencia que la implementación de modelos de redundancia de reglas propuestos en este trabajo resulta en una mejora significativa en la capacidad de clasificación.

1.5. Estructura de la Tesis

En el presente trabajo de investigación, se estructura la tesis doctoral en diversos capítulos con el propósito de abordar los objetivos planteados.

En el presente capítulo se proporciona una introducción detallada sobre la motivación principal de la investigación y destaca las contribuciones realizadas en el marco de esta tesis doctoral.

En el Capítulo 2, se establecen las bases teóricas y conceptuales necesarias para comprender los fundamentos de los Sistemas de Clasificación Basados en Reglas Difusas, los cuales serán empleados en los capítulos siguientes. Se presta especial atención a los conceptos clave que se utilizarán a lo largo del estudio.

Los capítulos siguientes, se centran en la resolución de los objetivos planteados. Cada capítulo inicia con una introducción que destaca las ventajas y desventajas del algoritmo *WM-C*, con el propósito de proponer mejoras en las distintas etapas del

modelo de aprendizaje y/o inferencia, tal como se especifica en el Objetivo General de esta tesis doctoral.

En el Capítulo 3, se exponen los métodos desarrollados para la minimización de las Bases de Reglas en los Sistemas de Clasificación Basados en Reglas Difusas. En este capítulo, se presentan detalladamente las técnicas utilizadas y se analizan los resultados obtenidos, lo que conduce al cumplimiento del Objetivo Especifico 1 propuesto en esta tesis doctoral.

Asimismo, en el Capítulo 4 se introduce un nuevo algoritmo que mejora la eficiencia del proceso de inferencia. Este nuevo enfoque logra tiempos significativamente mejores en comparación con el modelo estándar, lo que contribuye a una mejor respuesta, como parte del Objetivo Especifico 2.

En el Capítulo 5 se presenta un nuevo método, que llamamos *QChi*, basado en una reestructuración del algoritmo original, así como la inclusión de un nuevo método de cálculo de peso que reduce considerablemente el tiempo requerido para llevar a cabo dicha operación, y por lo tanto una reducción del tiempo necesario para obtener una base de reglas. La nueva propuesta se compara con otros enfoques aplicados en conjuntos de datos masivos, destacando su eficiencia y rendimiento, en consonancia con lo propuesto en el Objetivo Especifico 3.

Por último, en el Capítulo 6, y utilizando el modelo presentado en el capítulo anterior, se presentan nuevas técnicas para asignar más de una regla en cada paso del algoritmo original, con el propósito de reducir el porcentaje de ejemplos no cubiertos y por lo tanto mejorar la precisión de dicho algoritmo. Se describen los métodos propuestos y se propone un nuevo algoritmo llamado *QChi-MR*, se evalúa su comportamiento en relación al algoritmo original y al algoritmo propuesto en el tema anterior en diferentes situaciones, como parte del Objetivo Especifico 3.

Finalmente, en el Capítulo 7, se presentan las conclusiones derivadas de los capítulos anteriores y señalando posibles líneas de investigación futuras en el ámbito de los Sistemas de Clasificación Basados en Reglas Difusas. Las conclusiones brindan una síntesis integral de los resultados alcanzados y subrayan la relevancia y originalidad de la contribución realizada en esta tesis.

Mediante la estructura de los capítulos mencionados, se abordan de manera progresiva los objetivos establecidos, brindando una perspectiva completa de la investigación realizada en esta tesis doctoral.

Estado del arte

2

2.1. Lógica difusa y reglas difusas

Los sistemas basados en reglas han sido ampliamente reconocidos como una herramienta eficaz en la modelación de la actividad humana relacionada con la resolución de problemas y el comportamiento adaptativo. Estos sistemas presentan una metodología adecuada para representar el conocimiento humano mediante el uso de reglas *IF-THEN* (Si-Entonces), donde la cumplimentación del antecedente de una regla resulta en la ejecución del consecuente, lo que implica la realización de una única acción [61].

Los enfoques tradicionales utilizados para representar el conocimiento, que se basan en la lógica clásica de dos valores, presentan una limitación importante debido a su falta de capacidad para afrontar de manera efectiva los problemas relacionados con la incertidumbre y la imprecisión. En consecuencia, estos enfoques no ofrecen un modelo adecuado para el tipo de pensamiento comúnmente empleado en el razonamiento diario de los individuos. Esta situación ha generado la necesidad imperante de desarrollar enfoques alternativos que sean capaces de abordar la incertidumbre y la precisión en la representación del conocimiento de una forma más realista y coherente.

La aplicación de la Lógica Difusa (*LD*) en los sistemas basados en reglas ha dado lugar a los sistemas basados en reglas difusas (*FRBS*, por sus siglas en inglés)[66]. Estos sistemas utilizan reglas *IF-THEN* cuyos antecedentes y consecuentes están compuestos por declaraciones difusas, las cuales están compuestas por variables lingüísticas (*VL*) y sus correspondientes etiquetas lingüísticas, que se definen mediante conjuntos difusos dependientes del contexto, cuyo significado se especifica mediante funciones de pertenencia [98]. Esta característica presenta dos ventajas fundamentales en comparación con los sistemas basados en reglas clásicos:

- Manejo de la incertidumbre: Los *FRBS* permiten manejar la incertidumbre inherente en muchos problemas del mundo real. Al utilizar valores difusos en los antecedentes y consecuentes de las reglas, se puede representar y procesar

información ambigua o imprecisa, lo cual es común en la toma de decisiones y en la representación del conocimiento humano.

- Flexibilidad en la toma de decisiones: Los *FRBS* permiten una mayor flexibilidad en la toma de decisiones, ya que no se limitan a respuestas binarias de verdadero o falso. En cambio, se puede asignar un grado de pertenencia difusa a las conclusiones, lo que permite una mayor capacidad para representar y manejar la imprecisión y la vaguedad en la toma de decisiones.

Por otro lado, los métodos de inferencia de la *LD*, como el modus ponens, forman las bases del razonamiento aproximado en sistemas basados en reglas [19]. La *LD* proporciona una base computacional única y poderosa para llevar a cabo la inferencia en estos sistemas y a la vez una fundamentación teórica sólida para explicar con mayor detalle la teoría de los conjuntos difusos y su aplicación en diversos contextos.

2.1.1. Introducción a la teoría de conjuntos difusos

Los *FRBS* se fundamentan en la Teoría de Conjuntos Difusos (*TCD*), en la cual el concepto de conjunto difuso se representa mediante una función de pertenencia, que es un mapeo del Universo de Discurso (*UD*) U a una retícula completamente distributiva, generalmente en el intervalo [0, 1] [76]. Formalmente

$$\mu_A : U \longmapsto [0, 1] \tag{2.1}$$

es la función de pertenencia de un conjunto difuso etiquetado con el símbolo A.

Se hace una distinción entre la representación simbólica de un conjunto difuso, que se denota por su etiqueta asociada, y su representación semántica, que se denota por su función de pertenencia. Sin embargo, en general se hace referencia a un conjunto difuso solo mediante su etiqueta. Así, se adopta la notación abreviada de "conjunto difuso A". Para un elemento $x \in U$, su grado de pertenencia al conjunto difuso A se expresa como $\mu_A(x)$. Además, el conjunto de todos los posibles conjuntos difusos definidos en U se denota por $F(U)$.

2.1.2. Operadores sobre conjuntos difusos

Los operadores de conjuntos difusos se basan en funciones que generalizan las definiciones clásicas de intersección, unión y complemento; estos se extienden a través de t-normas, t-conormas y negaciones [76].

Una norma t es una función $\otimes : [0,1] \times [0,1] \longmapsto [0,1]$ que satisface las siguientes propiedades.

- **Monotonicidad:** $a \otimes b \leq c \otimes d$ si $a \leq c$ y $b \leq d$;
- **Conmutatividad:** $a \otimes b = b \otimes a$;
- **Asociatividad:** $a \otimes (b \otimes c) = (a \otimes b) \otimes c$;
- **Elemento neutro:** $a \otimes 1 = a$.

La t-conorma $\oplus$ tiene las mismas propiedades que las normas t, con la diferencia que el elemento de identidad es 0. Por lo general, el complemento 1 se usa como negación, es decir, $n(x) = 1 - x$ (aunque se pueden definir funciones mas generales). La Tabla 2.1 presenta una breve lista de opciones muy comunes de t-normas y t-conormas.

Nombre	$\otimes$	$\oplus$
Gödel	min (x, y)	max (x, y)
Producto	x · y	x + y - xy
Łukasiewicz	max (x + y - 1 , 0)	min (x+y , 1)

Tab. 2.1.: Elecciones comunes de t-normas y t-conormas.

Una opción común es la de utilizar la función min como la t-norma, la función max como la t-conorma y la función con complemento a 1 como negación. Esto lleva a la siguiente definición de operadores de conjuntos difusos.

$$\mu_{A_1 \cap A_2}(x) = min\{\mu_{A_1}(x), \mu_{A_2}(x)\} \tag{2.2}$$

$$\mu_{A_1 \cup A_2}(x) = max\{\mu_{A_1}(x), \mu_{A_2}(x)\} \tag{2.3}$$

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{2.4}$$

La relación de subconjunto proviene directamente del ordenamiento natural de los grados de pertenencia, es decir

$$A_1 \subseteq A_2 \iff \forall_x \in U : \mu_{A_1}(x) \leq \mu_{A_2}(x) \tag{2.5}$$

aunque varias generalizaciones son posibles para incluir conjuntos difusos.

2.1.3. Conjuntos difusos como gránulos de información

La granulación de la información es un concepto que se refiere a la descomposición de un todo en partes significativas [98] (por ejemplo, dividir una imagen satelital en ríos, bosques, carreteras, etc.). La *TCD* proporciona un sólido marco matemático para la computación granular, ya que los conjuntos difusos permiten una descomposición granular del *UD* en partes semánticamente significativas, que suelen etiquetarse con etiquetas lingüísticas. Esta representación basada en conjuntos difusos y etiquetas lingüísticas ofrece una herramienta poderosa para manejar la incertidumbre en la computación granular y en la representación del conocimiento en sistemas informáticos complejos.

Con respecto a otras teorías matemáticas para la computación granular (como la teoría de conjuntos aproximados o el análisis de intervalos), la *TCD* permite definir gránulos de información que son inherentemente imprecisos (debido a la falta de límites) y proporciona una maquinaria matemática para hacer inferencias con gránulos de información. En general, las relaciones difusas multidimensionales permiten descomponer un *UD* complejo y multidimensional en "parches" difusos (que son gránulos de información difusos) en los que se pueden identificar modelos de inferencia simples. Como consecuencia todo el *UD* se descompone en gránulos de información difusa posiblemente superpuestos, cuyos modelos locales pueden tomarse juntos para definir un modelo complejo que, sin embargo, pueden describirse mediante componentes simples. Básicamente esto define a los *FRBS* que se explicarán en detalle a continuación.

2.2. Sistemas de clasificación basados en reglas difusas

El primer tipo de *FRBS* que trata con entradas y salidas reales fue el propuesto por Mamdani [68], quien pudo aumentar la formulación inicial de Zadeh permitiendo la aplicación de la *TCD* a un problema de control.

Los Sistemas de clasificación basados en reglas difusas (*FRBS-C*) son una variante específica de los sistemas basados en reglas difusas (*FRBS*), en la cual se enfocan en clasificar y etiquetar los datos de entrada en diferentes categorías en lugar de modelar y representar el conocimiento humano en su totalidad.

Estos sistemas representan dos conceptos fundamentales: el conocimiento y el razonamiento. La distinción clara entre el conocimiento y las estructuras de procesamiento es un aspecto clave de los sistemas basados en el conocimiento. Por lo tanto, un *FRBS-C* puede considerarse como un tipo de sistema basado en el conocimiento, ya que emplea reglas difusas para realizar la clasificación de los datos.

En la literatura científica, existen numerosas propuestas de algoritmos de aprendizaje de reglas que utilizan diferentes métodos y distintos tipos de reglas. Algunos ejemplos relevantes son los siguientes [40, 36, 38, 58, 62, 70, 5].

Los problemas de clasificación de patrones implican asignar una clase B_j, de un conjunto de clases predefinido $B = \{B_1, \ldots, B_q\}$ a un objeto, el cual está descrito como un punto en un espacio de características específico $x \in S^N$ [19].

El problema de diseñar un clasificador es encontrar una correspondencia

$$D : S^N \rightarrow B \tag{2.6}$$

óptima en el sentido de un cierto criterio $\delta(D)$ que determina el rendimiento del clasificador. El objetivo principal consiste en desarrollar un clasificador con la capacidad de asignar etiquetas de clase en todo el espacio de características, minimizando al máximo posible el error cometido en dicha asignación. El clasificador puede adoptar diversas formas, tales como un conjunto de reglas difusas, un árbol de decisión o una red neuronal, entre otras opciones. Si se emplea un conjunto de reglas difusas como clasificador, se le denomina Sistema de Clasificación Basado en Reglas Difusas. Un *FRBS-C* está compuesto por una Base de Conocimientos (*KB* por sus siglas en ingles) y un Método de Razonamiento Difuso (*FRM* por sus siglas en ingles). La *KB* está constituida por la Base de Reglas (*BR*) y la Base de Datos (*BD*), las cuales describen

la semántica de los subconjuntos difusos asociados a las etiquetas lingüísticas en la parte *IF* de las reglas. Por su parte, el *FRM* utiliza la información proporcionada por la *KB* para determinar una clase de etiqueta adecuada para todos los ejemplos admitidos. Esta estructura se representa gráficamente en la Figura 2.1.

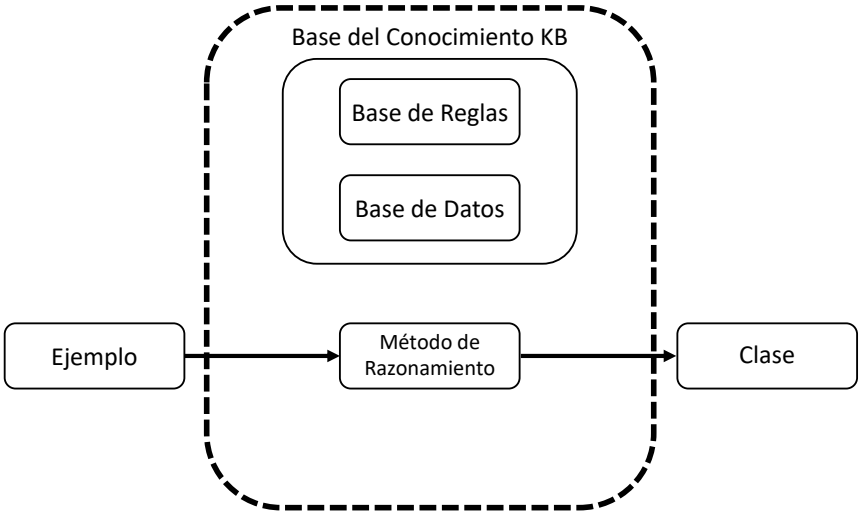

Fig. 2.1.: Estructura general de un sistema de clasificación basado en reglas difusas.

Para construir un *FRBS-C*, se inicia con un conjunto de ejemplos preclasificados, a partir del cual se debe determinar

- El método para encontrar o aprender un conjunto de reglas difusas para el problema de clasificación específico, y
- el método de razonamiento difuso utilizado para clasificar un nuevo ejemplo.

A continuación se proporcionará una explicación más detallada de cada componente de un *FRBS-C*.

2.2.1. Base del conocimiento *KB*

La *KB* desempeña un papel fundamental como un repositorio que almacena el conocimiento específico del problema, estableciendo la relación entre la entrada y la salida del sistema subyacente, lo que desencadena el proceso de inferencia. La *KB* se sustenta en dos niveles de información distintos: el nivel semántico, el cual se define mediante una base de datos (*BD*) que contiene valores lingüísticos (*VL*) y las particiones difusas asociadas, y el nivel simbólico, que se compone de

declaraciones condicionales difusas expresadas en forma de reglas lingüísticas *IF-THEN*, que forman la Base de Reglas (*BR*). Esta representación dual en la *KB* permite una representación eficiente y efectiva del conocimiento específico del problema, al combinar la riqueza semántica de los *VL* y la flexibilidad de las reglas *IF-THEN* difusas para llevar a cabo la inferencia en los *FRBS-C* [76].

Base de datos: Variables lingüísticas y particiones difusas

Los elementos básicos en una *BD* en un *FRBS-C*, son las *VL*. Una *VL* requiere la especificación de un símbolo que indique el nombre de la *VL*. Por lo general, el nombre de una *VL* coincide con el nombre del atributo de un objeto complejo (por ejemplo el atributo “Peso" de un objeto “Animal"). La *VL* puede tomar valores de un conjunto en etiquetas lingüísticas, que son símbolos que suelen coincidir con los términos de lenguaje natural. En algunos casos, el conjunto de etiquetas lingüísticas disponible es complejo; por lo tanto se especifica una gramática generativa, para proporcionar una maquinaria computacional que sea capaz de enumerar todas las etiquetas lingüísticas, así como de verificar si un término pertenece o no al conjunto admisible. Una *VL* mapea sus etiquetas lingüísticas en conjuntos difusos correspondientes, dotándolos así de una semántica.

Este proceso de mapeo es crucial para traducir estructuras lingüísticas, como las reglas, en gránulos de información difusa, que son utilizados en la inferencia aproximada. De esta manera, se logra manipular y procesar información incierta de manera efectiva. La formalización de una *VL* sigue los principios establecidos por Zadeh [98] como se exponen a continuación.

$$\langle X, T, U, G, \mu \rangle \tag{2.7}$$

donde

- X es el nombre de la variable (un símbolo) modelada en la *VL*.
- T es un conjunto (generalmente finito) de etiquetas lingüísticas (símbolos).
- U es el dominio de aplicabilidad de la *VL*.
- G es una gramática que genera los símbolos en T.
- $\mu : T \longmapsto f(U)$ es la interpretación semántica de términos lingüísticos. Dado un término $A \in T$. Esta interpretación como función de pertenencia se denotará como μ_A.

Como ejemplo se muestran las particiones difusas en la Figura 2.2, en este caso particular consta de 5 funciones de pertenencia difusa triangulares, donde los conjuntos difusos se definen con etiquetas lingüísticas Muy Bajo (*MB*), Bajo (*B*), Medio (*M*), Alto (*A*) y Muy Alto (*MA*), que se encuentran uniformemente distribuidas.

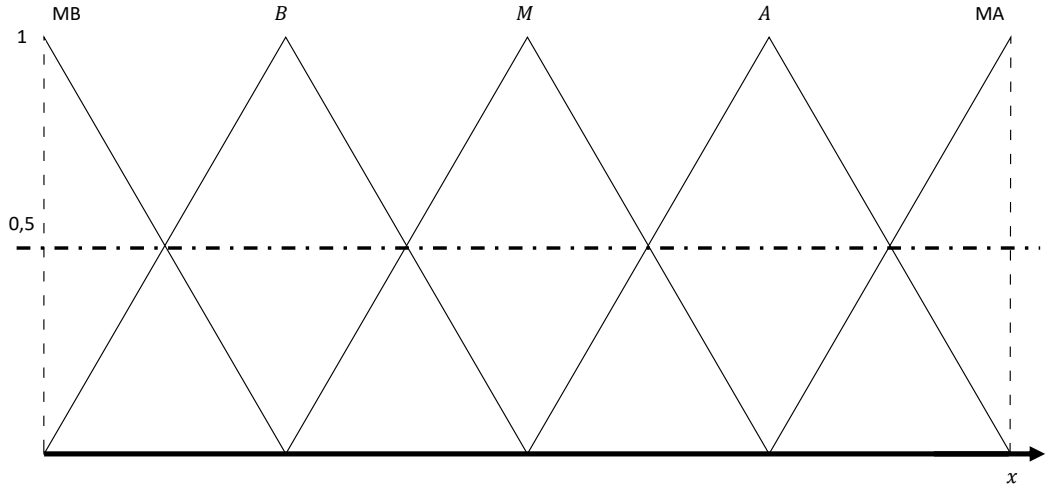

Fig. 2.2.: Ejemplo de partición difusa con 5 etiquetas lingüísticas.

Base de Reglas

La forma habitual de organizar el conocimiento en lenguaje natural es expresar reglas *IF-THEN* del tipo

$$\text{IF Antecendente THEN Consecuente} \tag{2.8}$$

que permiten inferir hechos que coinciden con el consecuente cuando se proporcionan algunos hechos que coinciden con el antecedente. El mayor poder de los *FRBS-C* es proporcionar inferencia incluso en el caso de coincidencia parcial entre hechos y reglas.

Tipos de Reglas Difusas

Podemos generar una *BR* con uno de los tres tipos de reglas siguientes:

1. Reglas difusas con una clase en el consecuente [37]. Este tipo de reglas tiene la siguiente estructura:

$$\text{R: IF } X_1 \text{ is } A^1_{j_1} \text{ and} \ldots \text{and } X_n \text{ is } A^n_{j_n} \text{ THEN } Y \text{ is } B \tag{2.9}$$

donde X_i, $i = 1 \ldots n$ son las *VL* antecedentes y cada una de estas variables tiene un dominio difuso D_i compuesto por $n_i = |D_i|$ etiquetas lingüísticas

$$D_i = \{A^i_1, A^i_2, \ldots, A^i_{n_i}\} \tag{2.10}$$

siendo $A^i_{j_i}$ un valor del dominio D_i. Además, la variable consecuente Y es una variable discreta, con un dominio asociado D_Y que representa la clase a aprender, B es un valor particular de este dominio.

2. Reglas difusas con una clase y un cierto grado en el consecuente [49].

$$\text{R: IF } X_1 \text{ is } A^1_{j_1} \text{ and} \ldots \text{and } X_n \text{ is } A^n_{j_n} \text{ THEN } Y \text{ is } B \; with \; \textbf{weight} \; w(R) \tag{2.11}$$

donde $w(R)$ se denomina peso de la regla de la clasificación en la clase B, para un ejemplo perteneciente al subespacio difuso delimitado por el antecedente.

3. Modelo extendido de reglas difusas [38, 10].

$$\text{R: IF } X_1 \text{ is } \tilde{A^1_{j_1}} \text{ and} \ldots \text{and } X_n \text{ is } \tilde{A^n_{j_n}} \text{ THEN } Y \text{ is } B \; with \; \textbf{weight} \; w(R) \tag{2.12}$$

donde cada variable de entrada X_i adquiere un conjunto de términos lingüísticos representados por $\tilde{A_{j_i}} = \{A_{j_{i_1}} \vee \ldots \vee A_{j_{i_{l_i}}}\}$. Estos términos lingüísticos se combinan mediante un operador disyuntivo (por ejemplo, mediante una T-conorma). A diferencia de las variables de entrada, la variable de salida mantiene su formato compuesto por clases, con una única etiqueta asociada. Esta estructura permite de forma natural la posibilidad de que algunas variables de entrada estén ausentes en cada regla del sistema (simplemente considerando que $\tilde{A_{j_i}}$ abarca todo el conjunto de términos lingüísticos disponibles).

2.2.2. Método de razonamiento difuso

Un *FRM* es un procedimiento de inferencia que extrae conclusiones a partir de un conjunto de reglas difusas *IF-THEN* y un ejemplo dado [19]. La principal ventaja de este tipo de razonamiento radica en su habilidad para obtener un resultado, incluso cuando no existe una coincidencia exacta entre una observación del sistema y los antecedentes de las reglas.

La mejora en la capacidad de generalización se logra al combinar la información proveniente de las reglas activadas con el ejemplo a clasificar. Al considerar múltiples reglas y su contribución al resultado final, se alcanza una mejor adaptabilidad y flexibilidad en el proceso de clasificación [19].

Método clásico de razonamiento difuso: regla ganadora

Supongamos que la *BR* es $R = \{R_1, \ldots, R_L\}$, para un conjunto de ejemplos de entrada $e = (e_1, \ldots, e_n)$, el método clásico de razonamiento difuso considera la regla con la combinación más alta, entre el grado de coincidencia del ejemplo con la parte *IF* y el peso en el consecuente. Posteriormente clasifica e con la clase que obtiene el valor más alto. Este procedimiento se describe en los pasos siguientes.

- Sea $A(R, e)$ la fuerza de activación de la parte *IF* de la regla R (grado de coincidencia). Normalmente, $A(R, e)$ se obtiene aplicando una t-norma al grado de pertenencia de cada variable.

$$A(R, e) = T(\mu_{A^1_{j_1}}(e_1), \ldots, \mu_{A^n_{j_n}}(e_n)) \tag{2.13}$$

  donde $\mu_{A^k_{j_k}}$ es la función de pertenencia de los conjuntos difusos $A^k_{j_k}$.

  En la literatura, existen algunas propuestas para este operador de conjunción; algunas de ellas son las siguientes: Ishibuchi et al., utiliza la t-norma de producto y mínimo [49], González et al., utiliza el mínimo [37], y Mandal et al. [69] y Uebele et al. [91] calculan el grado de coincidencia mediante la media aritmética (que no es una t-norma).

- Sea $h(A(R, e), w(R))$ el grado de asociación del ejemplo e con cada regla R de la base de reglas. Este grado se obtiene aplicando un operador de combinación entre $A(R, e)$ y $w(R)$. Para este operador, en [49] se utilizó el producto y en [69] la media aritmética.

$$h(A(R,e),w(R)) = Op(A(R,e),w(R)) \tag{2.14}$$

- Finalmente, se predice la clase de la regla con el mayor grado de asociación.

$$max_{R \in BR} h(A(R,e),w(R)) \tag{2.15}$$

Gráficamente, este método puede verse como se muestra en la Figura 2.3.

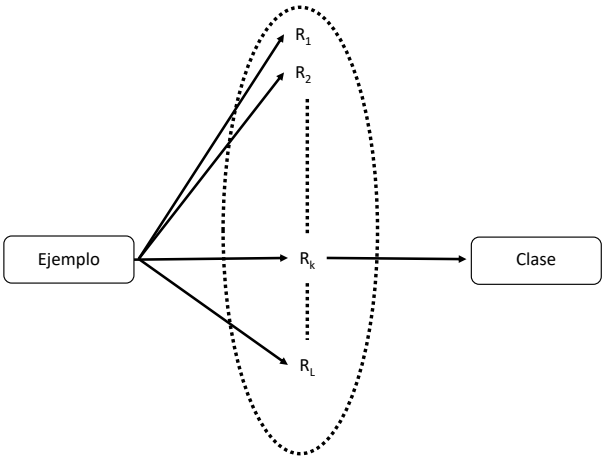

Fig. 2.3.: Método de razonamiento difuso que sólo utiliza la regla ganadora.

Este método de razonamiento utiliza sólo una regla, conocida como "la regla ganadora", en el proceso de inferencia y desperdicia la información asociada a todas aquellas reglas cuyo grado de asociación con el ejemplo de entrada es menor que el grado de asociación de la regla seleccionada.

En la literatura científica, se han llevado a cabo numerosas investigaciones que abordan el análisis, estudio y aplicación del razonamiento difuso en el motor de inferencia de los sistemas basados en el conocimiento. Entre estos trabajos, destacan los siguientes artículos [20, 73, 74, 88, 89], los cuales han contribuido al avance y comprensión de las técnicas de inferencia difusa.

2.3. Algoritmo de Wang y Mendel

El algoritmo de Wang y Mendel (*WM*)[92] es una técnica utilizada en *FRBS* para la identificación y optimización de las reglas difusas que representan el conocimiento

experto en un dominio específico. Fue uno de los primeros métodos en diseñar sistemas difusos a partir de ejemplos.

Dicho método se propuso originalmente para problemas de regresión. Posteriormente, una adaptación del algoritmo se propuso para clasificación en [14], al que denominaremos en este estudio como *WM-C* (conocido también como algoritmo de CHI), donde la salida presenta un valor discreto que representan las clases, convirtiéndolo así en un sistema de clasificación. Esta idea se llevará adelante a lo largo de toda la tesis doctoral.

El algoritmo de *WM-C* tiene las siguientes ventajas [3, 63]:

- Es un algoritmo que destaca por su simplicidad y facilidad de implementación.
- El tiempo usado para aprender es habitualmente muy reducido.
- El algoritmo propone una única regla para cada ejemplo de entrenamiento, lo que hace que en problemas con un número muy reducido de ejemplos el número de reglas en el conjunto final sea también pequeño.

Y las siguientes desventajas

- La falta general de exhaustividad de la *KB* generada resultante tiene consecuencias sobre la capacidad de generalización del sistema, produciendo problema para asignar una salida cuando la entrada no está soportada por ninguna de las reglas generadas por los datos, en definitiva el algoritmo tiene un importante problemas ya que puede dejar muchos ejemplos no cubiertos por ninguna regla.
- Cuando se aborda la problemática del Big Data, es común enfrentarse a conjuntos de datos con un gran número de instancias y/o características. En este contexto, el algoritmo experimenta dificultades debido al crecimiento exponencial del espacio de búsqueda. Este crecimiento desafía el proceso de aprendizaje inductivo y puede ocasionar problemas de escalabilidad y complejidad, lo que a su vez puede generar conjuntos de reglas difíciles de interpretar.

El tipo de regla utilizado en este estudio se ha adaptado específicamente para abordar tareas de clasificación. Se trata de las reglas de tipo 2, las cuales se corresponden con la Ecuación 2.11, estas reglas desempeñan un papel fundamental en el desarrollo de la propuesta de tesis doctoral. A continuación, se muestra la forma de la regla.

$$R : \textbf{IF } X_1 \text{ is } A_{j_1}^1 \text{ and } X_2 \text{ is } A_{j_2}^2 \text{ and } \ldots \text{and } X_n \text{ is } A_{j_n}^n$$
$$\textbf{THEN } Y \text{ is } B \text{ with } \textbf{weight } w(R)$$

Seguidamente se expondrán los distintos tipos de peso que se pueden utilizar en el algoritmo de *WM-C*.

El peso de la Regla

El peso de la regla (weight) ha desempeñado un papel fundamental en el algoritmo *WM-C*. Inicialmente, el algoritmo *WM-C* utilizaba un modelo de peso muy simple [14], en el que para cada ejemplo se calculaba lo siguiente,

$$w(R) = \max_{e \in E} \{A(R, e) \mid clase(e) \; es \; B\} \tag{2.16}$$

donde E es el conjunto de entrenamiento y $clase(e)$ es la clase del ejemplo e.

El consecuente que proporcionaba el valor de peso más alto era el seleccionado para ese antecedente. Este modelo de ponderación es extremadamente simple y no tiene en cuenta cuántos ejemplos apoyan la ponderación de la regla, por lo que posteriormente se propusieron varias alternativas para calcular el peso de la regla. El primero de ellos [40], establece la siguiente relación,

$$w(R) = \frac{n^+(R, E)}{n(R, E)} \tag{2.17}$$

donde $n^+(R, E)$ es el cardinal del conjunto difuso de todos los ejemplos de la clase B en el conjunto de entrenamiento E que tienen una adecuación positiva al antecedente de la regla R, es decir:

$$n^+(R, E) = \sum_{e \in E} A(R, e) \mid clase(e) \; es \; B \tag{2.18}$$

y $n(R, E)$ como el cardinal del conjunto difuso de todos los ejemplos del conjunto de entrenamiento E que tienen adaptación con el antecedente

$$n(R, E) = \sum_{e \in E} A(R, e).$$

Otra propuesta comúnmente utilizada es el método heurístico conocido como Factor de Certeza Penalizado (*PCF*) propuesto en [50] y con un enfoque en problemas de reglas difusas presentado en CHI-BD [27].

$$w(R) = PCF = \frac{n^+(R,E) - n^-(R,E)}{n(R,E)} \tag{2.19}$$

donde definimos $n^-(R,E)$ como el cardinal del conjunto difuso de todos los ejemplos del conjunto de entrenamiento E que tienen una adaptación positiva al antecedente de la regla y no son de la clase B.

$$n^-(R,E) = \sum_{e \in E} A(R,e) \mid clase(e)\ no\ es\ B. \tag{2.20}$$

Otra propuesta se basa en una actualización del *PCF* que se denomina *PCF-CS* [34], adecuada para conjuntos de datos desequilibrados, la cual incluye en la fórmula un coste de clasificación errónea asociado a una clase concreta.

$$w(R) = PCF - CS = \frac{n^+(R,E).C_s(clase(e)) - n^-(R,E).C_s(clase(e))}{n(R,E).C_s(clase(e))} \tag{2.21}$$

donde $C_s(clase(e))$ es el coste de clasificación errónea asociado a la clase $clase(e)$. Los costes se denotan como $C_s(min) = IR$ y $C_s(max) = 1$, donde min y max son las clases minoritaria y mayoritaria, respectivamente, e *IR* es el ratio de desequilibrio denotado como e_{alto}/e_{min}, donde e_{alto} y e_{min} son el número de instancias pertenecientes a las clases mayoritaria y minoritaria, respectivamente.

Otro cálculo del peso se propone en CHI-BD-DRF [1] que con la idea de reducir el cálculo, utiliza un peso que viene definido por la confianza de cada regla en particular:

$$w(R) = \frac{\sigma(ANT(R), B)}{\sigma(ANT(R))} \tag{2.22}$$

donde $\sigma(ANT(R), B)$ cuenta el número de ejemplos que activan la regla R (tanto el antecedente como el consecuente) con cualquier grado de coincidencia y $\sigma((ANT(R))$ cuenta el número de ejemplos que activan el antecedente de la regla R con cualquier grado de coincidencia. Esta ponderación de las reglas no sólo reduce el tiempo de cálculo, sino que también es una aproximación a la fiabilidad de cada regla. También existen otras propuestas enfocadas a problemas de regresión

y basadas en estimaciones, más que en cálculos de los pesos de las reglas, como el presentado en [12].

Construcción de la Base de Reglas

Teniendo en cuenta la estructura de regla descrita en la Ecuación 2.11, para construir la *BR*, se aplica el siguiente algoritmo de aprendizaje.

1. Definición de las etiquetas lingüísticas. Los conjuntos difusos (etiquetas lingüísticas) se construyen, por ejemplo, usando funciones de pertenencia triangulares e igualmente distribuidos en el rango de valores.

2. Generación de una regla difusa para cada ejemplo. Se genera una regla difusa para cada ejemplo E de la siguiente manera.

   - Se calculan los grados de pertenencia de cada valor e a todos los conjuntos difusos diferentes de la variable i-ésima calculados.

   - Para cada variable, se selecciona la etiqueta lingüística con el mayor grado de pertenencia.

   - El antecedente viene determinado por la intersección de las etiquetas lingüísticas seleccionadas y el consecuente es la etiqueta de clase del ejemplo (y_i). Todas las reglas tendrán exactamente el mismo número de antecedentes como variables del problema (n).

   - Se calcula el peso de la regla.

   - Se pueden obtener reglas con el mismo antecedente, pero diferentes consecuente tras el proceso de generación de reglas. En ese caso, solo se conserva la regla con el peso más alto, mientras que las reglas con peso negativo son eliminadas de la base de reglas.

Una vez obtenida la base de reglas, para la clasificación de un nuevo ejemplo E se usará el método de la Regla Ganadora 2.2.2. Con lo cual se obtienen todos los pasos necesarios para utilizar un *FRBS-C* basado en el Algoritmo de *WM-C*.

Evolución del algoritmo de Wang y Mendel

El algoritmo de *WM-C* ha ganado popularidad en los últimos años y ha sido objeto de varias revisiones, incluyendo el trabajo de [3], donde se analizan sus aspectos positivos y negativos, proponiendo posibles áreas de trabajos futuros para abordar sus limitaciones.

Se han desarrollado numerosos trabajos derivados del método *WM-C* con el objetivo de mejorarlo. Algunos trabajos se han centrado en la base de conocimiento *KB*, especialmente en la base de datos *BD*, donde se han modificado las formas regulares de las funciones de pertenencia triangulares y la cantidad de las mismas [21, 60, 94, 26]. Además, algunos estudios han utilizado la selección de características para eliminar variables poco relevantes y reducir el costo computacional [15].

Con respecto a la base de reglas *BR*, se han utilizado métodos no deterministas para minimizarla, como el propuesto por [17]. También se han abordado problemas en el cálculo del peso en *WM*, el cual puede variar en conjuntos de datos desbalanceados [31] o para problemas de clustering [65].

En relación con los desafíos que el algoritmo enfrenta al trabajar con conjuntos de datos masivos, se han propuesto alternativas y enfoques adicionales. Por ejemplo, [27, 28, 63] y CHI-BD-DRF [1] son algoritmos basados en *WM-C* pero adaptados a trabajar con datos masivos . Estos enfoques han incorporado una estructura de computación paralela MapReduce [25, 24] para dividir la carga computacional generada por grandes bases de datos. La utilización de esta estructura de computación paralela ha demostrado ser prometedora en términos de eficiencia y escalabilidad, permitiendo un procesamiento más ágil y efectivo en el contexto de grandes volúmenes de datos.

A lo largo del tiempo, el algoritmo de *WM* ha sido objeto de numerosas aplicaciones en diversas áreas, entre las que podemos citar [90], que lo utilizaron para medir las proyecciones de una red eléctrica, también se desarrolló un modelo para el ajuste del PID en los motores [13], en el diseño de embarcaciones, se utilizó el algoritmo de *WM* para medir la resistencia residual en el diseño de un velero [30], en el ámbito de la distribución de energía eléctrica, se ha aplicado el algoritmo de *WM* en la previsión de la carga eléctrica [97], también ha sido utilizado para predecir el índice de la bolsa de valores [93], otro campo de aplicación relevante es la predicción de la energía fotovoltaica, como se presenta en [75] y el algoritmo de *WM* ha sido empleado en el diseño y optimización de redes postales, como se describe en [6], entre otras aplicaciones.

Actualmente el algoritmo de *WM* continua siendo un algoritmo popular en los *FRBS*, donde periódicamente se realizan nuevas actualizaciones del algoritmo con diferentes enfoques.

2.4. Bases de datos masivas

La sociedad actual se encuentra inmersa en la Era de la Información, caracterizada por la disponibilidad de vastas cantidades de datos. A diario, se generan y almacenan petabytes de información, lo que resulta en un volumen masivo de datos. La velocidad de generación de esta información es vertiginosa, requiriendo un procesamiento en tiempo real. Además, los datos pueden presentarse en diversos formatos, como estructurados, semi estructurados o no estructurados, lo que implica una variedad de fuentes y estructuras.

La calidad de los datos también es fundamental, siendo necesario garantizar su veracidad mediante procesos de limpieza y validación. Por último, es crucial que esta información aporte valor a las organizaciones, ya sea en términos de toma de decisiones, generación de conocimientos o mejora de procesos. Estos cinco conceptos, volumen, velocidad, variedad, veracidad y valor conforman una de las definiciones más ampliamente aceptadas de Big Data [64], como se ilustra en la Figura 2.4.

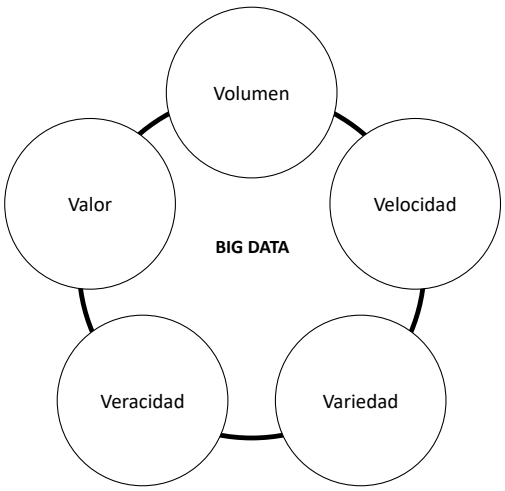

Fig. 2.4.: Representación gráfica del concepto de Big Data.

Si bien los aspectos de volumen, velocidad y variedad se centran en la captura y almacenamiento de datos, los aspectos de veracidad y valor se refieren a la calidad y utilidad de estos. Estos dos últimos aspectos desempeñan un papel fundamental en los procesos de Big Data, ya que la obtención de conocimientos útiles y valiosos

depende en gran medida de la calidad de los datos utilizados. Por lo tanto, la gestión adecuada de estos aspectos se vuelve esencial para maximizar el potencial del análisis de datos en el contexto de Big Data.

El verdadero valor del Big Data no radica únicamente en la abundancia de datos, sino en la capacidad de descubrir patrones inesperados y extraer conocimiento a través de las técnicas adecuadas de ciencia de datos. Aquellas organizaciones que logren extraer información valiosa de grandes volúmenes de datos en un tiempo razonable podrán obtener ventajas significativas sobre sus competidores. Los investigadores académicos también reconocen la importancia de desarrollar modelos robustos y precisos para la minería de datos en aplicaciones de Big Data [64]. Se observa un claro incremento en la cantidad de investigaciones y estudios sobre este tema, y se espera que esta tendencia continúe en el futuro cercano [48].

2.4.1. Problemas y desafíos

Los desafíos presentados por los macrodatos se pueden clasificar en tres categorías principales, según se señala en el estudio de referencia [80]. En primer lugar, se encuentran los desafíos relacionados con los propios datos, que abarcan sus características distintivas y particularidades. En segundo lugar, se plantean los desafíos asociados al proceso de manipulación y análisis de los datos. Por último, se encuentran los desafíos vinculados a la gestión integral de los datos, incluyendo aspectos de seguridad y privacidad. Las características de los macrodatos generan múltiples desafíos, como su volumen masivo y su diversidad en cuanto a formatos y fuentes. Los desafíos en el proceso abarcan la adquisición de los datos, su preprocesamiento, el análisis y la visualización. Por otro lado, los desafíos en la gestión de los datos involucran aspectos fundamentales como la protección de la privacidad y la seguridad de la información.

El problema de los datos

Los investigadores han realizado numerosas definiciones de Big Data y, a partir de su conocimiento, han propuesto diversas características adicionales. Inicialmente, los investigadores [84] introdujeron las tres V de los datos (volumen, variedad y velocidad), posteriormente se agregó la cuarta V para abordar la veracidad, y más recientemente se ha hablado de la quinta V que hace referencia al valor de los datos. Estas cinco características conforman el modelo ampliamente aceptado, representado en la figura 2.4. Además, si consideramos las diez V de los macrodatos [59], se

destacan numerosos desafíos asociados a las características de los datos que merecen ser mencionados. A continuación, se presentan algunos de estos inconvenientes.

1. Volumen: El crecimiento exponencial de los datos provenientes de diversas fuentes internas y externas ha generado un enorme volumen de información, planteando desafíos significativos. Este incremento masivo de datos presenta dificultades inherentes a su gestión, especialmente en lo que respecta a su almacenamiento y procesamiento. Las herramientas tradicionales resultan insuficientes para hacer frente a esta cantidad abrumadora de datos, lo que impulsa la necesidad de desarrollar métodos más innovadores y eficientes. En este contexto, se han propuesto soluciones como el uso de tecnologías como Apache Spark [78], que permiten el almacenamiento, procesamiento y análisis escalable de grandes volúmenes de datos en entornos distribuidos.

2. Variedad: El desafío asociado a la variedad de los datos masivos se relaciona con sus múltiples formas de presentación. Estos datos pueden adoptar formatos estructurados, semiestructurados y no estructurados. Según investigaciones, aproximadamente el 95 % de los datos se encuentra en forma no estructurada, lo que representa un desafío considerable para su conversión a un formato que permita su análisis. La tarea de transformar estos datos en una estructura adecuada para su posterior procesamiento y análisis implica superar obstáculos significativos. Abordar este desafío es esencial para desbloquear el potencial de información valiosa que yace en los datos masivos y permitir su aprovechamiento en diversas áreas de investigación y aplicación.

3. Velocidad: La velocidad se refiere a la velocidad a la que se generan los datos a través de dispositivos. Estos datos pueden ser procesados de dos maneras: mediante el procesamiento por lotes y el procesamiento en tiempo real. En el procesamiento por lotes, los datos se almacenan y luego se procesan en lotes periódicos, mientras que, en el procesamiento en tiempo real, el análisis y procesamiento de datos se lleva a cabo de forma continua a medida que se generan. La capacidad de procesar y analizar datos en tiempo real es crucial en diversos campos, como las compras en línea, donde se busca generar valor para los clientes al ofrecer recomendaciones personalizadas y respuestas rápidas a sus acciones. Esto permite aprovechar la velocidad de generación de datos para tomar decisiones informadas y brindar experiencias más ágiles y personalizadas a los usuarios.

4. Veracidad: La veracidad de los datos se refiere a la calidad y exactitud de la información contenida en ellos. Este aspecto aborda cuestiones como las fabricaciones, imprecisiones, desorden y pruebas inadecuadas presentes en

los datos. La veracidad es especialmente relevante cuando se deben tomar decisiones críticas basadas en los datos disponibles. Por ejemplo, en el contexto de las redes sociales, la veracidad de los datos se relaciona con la confiabilidad de las opiniones de los usuarios. Estas opiniones pueden ser clasificadas como positivas, negativas o neutrales, y es fundamental asegurarse de que sean veraces y representativas de la realidad para poder tomar decisiones informadas. Para garantizar la veracidad de los datos, es necesario implementar técnicas de validación y verificación, así como contar con mecanismos de control de calidad y filtrado de información errónea o engañosa. De esta manera, se puede asegurar que los datos utilizados sean confiables y se eviten sesgos o interpretaciones incorrectas.

5. Valor: El valor es una característica destacada en el contexto de los macrodatos. Estos grandes conjuntos de datos contienen información de gran relevancia que debe ser extraída de manera efectiva para su aplicación en áreas como la inteligencia empresarial y los sectores sanitarios, entre otros. La extracción de información valiosa de los datos plantea un desafío significativo, dado que debe realizarse de manera rentable y con el objetivo de aprovecharla de manera efectiva. De esta manera, se podrá obtener el máximo valor de los macrodatos y utilizarlo para mejorar la toma de decisiones, optimizar procesos y obtener ventajas competitivas en diversas áreas de aplicación.

Problemas en el procesamiento

Los desafíos del proceso están estrechamente vinculados al procesamiento y análisis de grandes conjuntos de datos. Estos desafíos representan una tarea significativa para el proceso, ya que los datos se presentan en diversas formas y convertirlos en un formato único para su análisis resulta en una tarea ardua. El proceso puede dividirse en cuatro etapas fundamentales: la adquisición y almacenamiento de datos, el preprocesamiento de datos, el análisis y modelado de datos, y la visualización de datos [80].

1. Adquisición y almacenamiento de datos: La adquisición de datos se refiere al proceso mediante el cual se obtienen y almacenan datos para su posterior uso como información valiosa. Los datos se adquieren de diversas fuentes, tales como sensores, redes sociales, blogs, entre otros, lo que conlleva la presencia de diferentes formas de datos, ya sean estructurados, semiestructurados o no estructurados. Esta diversidad representa un desafío significativo para los datos. El segundo desafío está relacionado con el almacenamiento de los datos

adquiridos. Dado que los datos son generados a través de diversos dispositivos, no todos ellos tienen necesariamente relevancia o significado para su posterior análisis. Por lo tanto, es necesario aplicar un filtro inteligente que permita identificar y generar conjuntos de datos relevantes. El almacenamiento de estos conjuntos masivos de datos puede dar lugar a sistemas escalables de alto costo diseñados para gestionar eficientemente la enorme cantidad de datos generados.

2. Preprocesamiento de datos: El preprocesamiento de datos desempeña un papel crucial en el descubrimiento de conocimientos a partir de grandes conjuntos de datos. Su objetivo principal es recopilar datos de calidad, ya que los datos de baja calidad pueden conducir a la obtención de conocimientos de baja calidad. Durante esta etapa, se llevan a cabo diversas operaciones para mejorar la calidad de los datos, como la eliminación del ruido, la gestión de valores perdidos, la corrección de datos incoherentes o redundantes, entre otros. Estas acciones preparan los datos para su posterior análisis y aplicación de técnicas de minería de Big Data [35]. En el contexto del preprocesamiento de Big Data, se dedican gran parte de los esfuerzos al método de selección de características, mientras que se suelen pasar por alto algunas de sus familias, como la reducción de dimensiones, la imputación de valores perdidos o el tratamiento del ruido. Estos aspectos son igualmente relevantes y deben ser considerados para obtener resultados confiables y precisos en el análisis de los datos masivos.

3. Análisis y modelización de datos: El análisis de datos constituye un proceso esencial en la búsqueda de información oculta y valiosa en conjuntos de datos, y su aplicación proporciona una base sólida para la toma de decisiones informadas por parte de las organizaciones. Para extraer de manera eficaz el conocimiento de grandes volúmenes de datos, se requiere el empleo de técnicas innovadoras y avanzadas. Estas técnicas buscan revelar patrones y relaciones ocultas en los datos a través del uso de métodos estadísticos y de aprendizaje automático.

4. Visualización de datos: Una técnica de visualización de macrodatos se utiliza para representar de forma visual los datos analíticos. Esta técnica emplea diversos tipos de gráficos y visualizaciones para presentar la información valiosa necesaria para la toma de decisiones informadas [43]. Investigaciones científicas han demostrado que los informes visuales tienen un mayor impacto en la comprensión y búsqueda de información en comparación con los informes basados en texto. Para llevar a cabo la visualización de datos, se

utilizan herramientas especializadas, sin embargo, se ha observado que estas herramientas pueden no ser suficientemente eficientes en entornos en los que los datos crecen constantemente a un ritmo vertiginoso. La necesidad de lidiar con grandes volúmenes de datos en tiempo real plantea desafíos adicionales para la visualización efectiva de los macrodatos.

La presente tesis doctoral se enfoca en el estudio del algoritmo de *WM-C*, con el propósito de realizar mejoras en su estructura para potenciar métricas como la interpretabilidad, la eficiencia y la precisión. Como parte de este estudio, se considera abordar la solución a problemas que involucran conjuntos de datos masivos, los cuales contienen un gran volumen de información, así como el análisis y modelización de estos. En los siguientes capítulos, se llevará a cabo la exploración de los diferentes procesos del algoritmo de *WM-C*, en consonancia con los objetivos establecidos en el marco de esta tesis doctoral.

3 Minimización de la Base de Reglas

En este capítulo se presentan las propuestas desarrolladas en relación con el objetivo específico 1 de esta tesis doctoral. Se exponen tanto las ventajas del algoritmo de Wang y Mendel para clasificación (*WM-C*) como sus inconvenientes, enfocándose específicamente en el proceso de creación de reglas de la base del conocimiento (*KB*), con el propósito de mejorar la interpretabilidad del algoritmo mediante la aplicación de técnicas de minimización en la base de reglas difusas (*BR*).

Tradicionalmente, se ha considerado que una de las grandes ventajas de los Sistemas de Clasificación Basados en Reglas Difusas (*FRBS-C*) es su capacidad para obtener conocimiento interpretable mientras se logran niveles aceptables de precisión. La interpretabilidad es un tema de discusión frecuente en la investigación [76, 83, 4], y existen numerosos estudios y métricas que permiten comparar modelos en términos de interpretabilidad. Uno de los parámetros comúnmente utilizado para estudiar la interpretabilidad de un sistema es el número de reglas, de manera que a mayor cantidad de reglas, menor es la interpretabilidad del sistema [33]. Sin embargo, en algunos casos no es posible obtener un número reducido de reglas como solución a un problema. Esto suele ocurrir, por ejemplo, con el algoritmo (*WM-C*) [14], el cual suele generar un elevado número de reglas, lo que afecta a la interpretabilidad de la solución proporcionada. Este aspecto es de gran importancia, puesto que un algoritmo que genere conocimiento interpretable permite al usuario comprender el proceso seguido en la toma de decisiones.

En la literatura científica, se han propuesto diversos enfoques para mejorar la interpretabilidad de las *BR* en los *FRBS-C*. Uno de los ejemplos es el enfoque del CFM-BD [29], que utiliza técnicas de probabilidad e inducción de reglas para crear una base de reglas más compacta. Además, otro enfoque propuesto es el uso de mapas de Karnaugh para reducir el número de reglas difusas [47]. Aunque esta técnica puede simplificar funciones algebraicas de forma canónica, una limitación importante es que solo puede manejar un máximo de 6 variables binarias. Esto implica que solo se pueden representar dos variables difusas con 3 etiquetas lingüísticas cada una, lo que limita su aplicabilidad en casos más complejos. En un estudio más reciente, se ha investigado el diseño de modelos difusos interpretables utilizando la técnica de

cointensión semántica [72], en combinación con el algoritmo Espresso como método de minimización [7].

Como parte de esta propuesta, se plantea explorar una nueva técnica para mejorar la interpretabilidad de la *BR*, en la cual se propone un cambio en el modelo de reglas básicas (Ecuación 2.11), por un modelo de reglas extendidas (Ecuación 2.12). Tal como se ha mencionado anteriormente, una estrategia para lograr dicha mejora radica en la agrupación de valores de las variables y en la reducción del número de reglas, para lo cual se hará uso del método de minimización de Quine-McCluskey (*QM*) [79]. Este algoritmo se caracteriza por su destreza en la simplificación de funciones booleanas, lo que lo convierte en una herramienta potencialmente valiosa para la simplificación de un conjunto de reglas difusas. Sin embargo, con miras a alcanzar este objetivo, se hace necesario codificar previamente la *BR* en un formato compatible con el método mencionado. A continuación, se presentarán los fundamentos teóricos que sustentan el método de minimización propuesto en el presente estudio.

3.1. Preliminares

En esta sección se presentan las herramientas básicas necesarias para llevar a cabo la propuesta descrita en este trabajo.

3.1.1. Modelo de Regla

Para la generación de la base de reglas, se ha empleado el algoritmo de *WM-C*. En este proceso, se utilizó el peso conocido como *PCF*, cuya definición se encuentra en la Ecuación 2.19. No obstante, es relevante señalar que la propuesta que se presenta no está ligada exclusivamente al algoritmo de *WM-C* y podría aplicarse a cualquier base de reglas con el modelo básico (Ecuación 2.11). La salida del proceso de minimización será un conjunto de reglas del llamado “Modelo extendido de reglas difusas” definido en la Ecuación 2.12. Este modelo permite asignar un subconjunto de las etiquetas difusas de su dominio a una variable, lo que permite simplificar la descripción de la regla difusa.

3.1.2. El método de Quine McCluskey

El método de *QM* [79], es una técnica utilizada para minimizar expresiones lógicas que representan funciones booleanas. Una función booleana está compuesta por un conjunto de variables booleanas que se conectan mediante las operaciones lógicas AND y OR [44]. La propiedad de simplificar funciones booleanas es altamente eficaz para su implementación en algoritmos informáticos. La aplicación del método *QM* permite reducir el número de términos en la expresión lógica, lo que conlleva a una representación más compacta y eficiente de la función booleana.

En esta apartado se presentan las definiciones fundamentales y los pasos necesarios para aplicar el método *QM*.

Definiciones

- Literal: Es una variable lógica o su negación (q o $\bar{q}$).
- Minterm: es una expresión algebraica booleana de n variables booleanas, que solamente se evalúa como verdadera para una única combinación de esas variables.
- Implicante Primo: es el producto que no se puede combinar con otro término para eliminar una variable y una mayor simplificación.
- Implicante Primo esencial: es un implicante primo que es capaz de cubrir una salida de la función que no está cubierta por ninguna combinación de implicantes primos.

Utiliza las siguientes tres leyes básicas de simplificación.

- $q + \bar{q} = 1$ (Complemento)
- $q + q = q$ (Idempotente)
- $q(w + z) = qw + qz$ (Distributiva)

donde *q, w y z* son literales.

En el método de *QM*, la entrada consiste en los minterms de la función booleana que se desea simplificar, y la salida es una expresión lógica simplificada que representa la función booleana original. A continuación, se presentan los pasos para llevar a cabo el proceso de simplificación:

1. Se realiza la búsqueda de los implicantes primos. Para ello, se realiza una sustitución del literal en forma de 0 y 1, lo que genera una tabla. El número de filas de la tabla es inicialmente igual al número total de minterms de la función original no simplificada. Si dos términos sólo se diferencian en un bit y una variable aparece en ambas formas (variable y negación), entonces se puede utilizar la ley del complemento. De manera iterativa, se comparan todos los términos y se genera el implicante primo.

2. En la siguiente etapa, se busca identificar los implicantes primos esenciales a partir de los implicantes primos obtenidos en el paso anterior. Para ello, se genera una tabla en la que se incluyen todos los implicantes primos y se marcan aquellos que cubren un único minterm de la función original. Estos implicantes primos marcados como esenciales no pueden ser omitidos, ya que cubren un minterm que no está cubierto por ningún otro implicante primo. Por otro lado, aquellos implicantes primos que no son marcados como esenciales pueden ser omitidos, ya que están redundando información.

3. Encontrar otro implicante primo: En algunos casos, el implicante primo esencial no cubre todos los términos mínimos de la función, por lo que se requiere encontrar otros implicantes primos para cubrirlos. Para esto, se puede utilizar nuevamente la tabla de implicantes primos obtenidos en el paso anterior y encontrar implicantes primos adicionales que cubran los términos que no fueron cubiertos por los implicantes primos esenciales. De esta manera, se obtiene una simplificación completa de la función lógica original.

4. En la situación en la que no es posible obtener ningún implicante primo esencial, se enfrenta a un problema de cobertura cíclica que se puede resolver mediante el método de Petrick [86]. Este método busca encontrar la combinación más reducida de implicantes primos que cubran todos los términos mínimos.

En general, se considera que el método *QM* proporciona una mejor manera de simplificar una función booleana en comparación con los mapas de Karnaugh. Sin embargo, sigue siendo un problema NP-duro, lo que significa que la complejidad del proceso aumenta de forma exponencial [85].

3.2. Métodos propuestos

A continuación, se exponen dos enfoques para lograr la minimización de base de reglas usando el método de *QM*, donde la principal diferencia entre ambos se basa en la codificación de las variables lingüísticas. El primero utiliza una codificación binaria, mientras que el segundo utiliza una codificación ternaria. Ambos enfoques tienen como objetivo simplificar y reducir el número de reglas en la *BR*, pero se diferencian en el enfoque de codificación utilizado, lo que puede tener implicaciones en los resultados obtenidos y en la interpretabilidad del modelo resultante. A continuación, se describen en detalle ambos enfoques para analizar su efectividad en el proceso de minimización de la *BR* difusa.

Minimización de la Base de Reglas

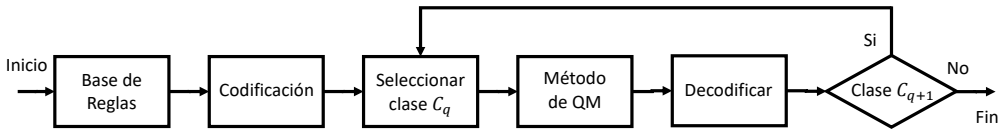

Fig. 3.1.: Minimización de la Base de Reglas.

Se hará referencia al diagrama de bloques de la Figura 3.1 para explicar el proceso de minimización en los métodos propuestos. Dicho proceso consta de cinco bloques claramente definidos, en los cuales se llevan a cabo diversas etapas para obtener la *BR* simplificada. El primer bloque, denominado "**Base de Reglas**", corresponde a la *BR* generada previamente por el algoritmo de *WM-C* (o de cualquier otra forma pero con el mismo tipo de regla) y supondremos que el dominio asociado a cada variable difusa está compuesto por 3 etiquetas uniformemente distribuidas sobre el universo de discurso de la variable, tal como se muestra en la Figura 3.2.

Una vez obtenida la base de reglas, el siguiente proceso se desarrolla en el bloque de "**Codificación**" en el cual se selecciona el antecedente de cada regla contenido en la *BR* y se somete a una transformación que tiene como objetivo modificar la estructura de cada regla para representarla como una función booleana. Este proceso genera vectores codificados que contienen los antecedentes de las reglas, y dichos vectores pueden estar representados en formato binario o ternario.

En el bloque de "**Seleccionar clase**", a cada uno de los vectores codificados se le asocia su respectiva clase, lo cual es esencial para el siguiente paso. Una vez obtenido

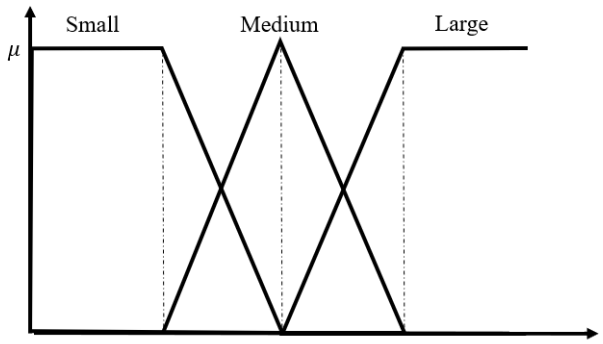

Fig. 3.2.: Dominio difuso de una variable lingüística $X_1, ..., X_n$.

el vector codificado que representa la *BR*, se procede a seleccionar una clase. En el bloque "**Método de QM**" se aplica el algoritmo de minimización de *QM* a todos los vectores codificados pertenecientes a la clase seleccionada, lo que resulta en la minimización de la *BR* codificada.

Posteriormente, en el bloque "**Decodificar**", se realiza el proceso inverso al de codificación, de modo que se transforman los vectores codificados nuevamente en los antecedentes de las reglas correspondientes a la clase en cuestión. Finalmente, se procede a seleccionar la siguiente clase y se regresa al bloque "**Seleccionar clase**" repitiendo el proceso de minimización para dicha clase. Este ciclo se repite hasta que se hayan recorrido todas las clases, y al finalizar este proceso, se obtiene la *BR* minimizada.

3.2.1. Minimización por Codificación Binaria

En el proceso de minimización de conjuntos de reglas difusas obtenidas por el algoritmo *WM-C*, se utiliza el método *QM*, el cual solo puede minimizar funciones booleanas. Por lo tanto, es necesario adaptar los valores lingüísticos de la base de reglas a un conjunto de variables binarias que representen esas variables lingüísticas.

Codificación binaria de la base reglas difusas

Como se mencionó anteriormente, es necesario realizar una transformación en la *BR* generada por el algoritmo de *WM-C* para que sea compatible con el método *QM*. Esta transformación se realiza mediante un proceso de particionamiento del universo de discurso de cada variable difusa, en el cual se crean nuevas variables

binarias que representan las etiquetas de cada partición. De esta forma, se obtiene una representación booleana de la base de reglas difusas, que puede ser utilizada como entrada del bloque "**Método de *QM***" correspondiente a la Figura 3.1.

En el método *QM*, se establece que si dos términos en una función booleana se diferencian en un solo bit, se pueden aplicar la ley del complemento para generar los implicantes primos. Este punto es relevante cuando se trabaja en la codificación de etiquetas lingüísticas para su aplicación en el método de *QM*, por lo que es necesario transformar las etiquetas lingüísticas en etiquetas binarias.

La cantidad de bits necesarios para representar cada etiqueta lingüística viene dada por la siguiente ecuación.

$$Bits = Round(\frac{log_2(L)}{log_2(2)} + 1) \tag{3.1}$$

donde L es el número de etiquetas lingüísticas del dominio y el redondeo es a la baja. Por ejemplo, 3 etiquetas lingüísticas utilizan 2 bits, 5 etiquetas lingüísticas utilizan 3 bits y así sucesivamente. Esta ecuación es una generalización utilizada para tener una referencia de cuántos bits se necesitan para un número impar de etiquetas lingüísticas.

Tomando como referencia las tres etiquetas lingüísticas (Small, Medium y Large), y considerando que la generación de los implicantes primos se produce cuando los Minterm difieren en un bit (la distancia de Hamming [42] entre etiquetas lingüísticas es igual a 1), se necesitan solamente 2 bits para representar cada etiqueta lingüística. Por tanto, se utiliza la siguiente codificación para las 3 etiquetas lingüísticas que se muestran en la Tabla 3.1.

Etiquetas	**Código**
Small	00
Medium	01
Large	11
No aplica	10

Tab. 3.1.: Codificación binaria de las variables lingüísticas.

Una vez obtenida la codificación de las *VL* procedemos a aplicar el método de *QM*. Vamos a ilustrar el proceso a través de un ejemplo.

Método de minimización de Quine McCluskey Binario

Suponiendo que se tienen el siguiente un conjunto de reglas en la *BR*:

R_1 : **IF** X_1 is Medium and X_2 is Small and X_3 is Medium and X_4 is Small **THEN** C_1 with $w(R_1)$

R_2 : **IF** X_1 is Medium and X_2 is Small and X_3 is Medium and X_4 is Medium **THEN** C_1 with $w(R_2)$

Los pasos para realizar la minimización de la *BR* tomando como referencia la Sección 3.2 son los siguientes:

1. En primer lugar, se convierte el antecedente de cada regla de la *BR* en una vector y se le asigna a una tabla de verdad, con valores binarios, siguiendo el sistema de codificación de la Tabla 3.1. Luego, se construye una tabla de verdad para cada clase de la *BR*, como se aprecia en la Tabla 3.2.

	Variable X_1		Variable X_2		Variable X_3		Variable X_4		
Regla	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8	Clase
R_1	0	1	0	0	0	1	0	0	1
R_2	0	1	0	0	0	1	0	1	1

Tab. 3.2.: Tabla de verdad perteneciente a la Clase 1.

2. Al dividir cada variable en pares de dos bits consecutivos, se logra crear una tabla de verdad para cada clase, en la que cada regla se representa como una fila binaria. Esto permite aplicar el método *QM* de forma independiente para cada clase y minimizar la función booleana.

3. Se Repite el mismo proceso con todas las reglas agrupándolas con sus respectivas clases.

4. Tras aplicar el método *QM*, se pueden generar dos tipos de casos. O bien se mantienen los bits originales y no hay simplificación, o bien se sustituye uno de los bits por un $*$, donde $*$ simboliza la eliminación de una variable booleana en el proceso de minimización. En este caso, las opciones son:

   - Caso 1: $X[0*]$, lo que significa que la asignación a la variable es Small o Medium.

- Caso 2: $X[* \ 1]$, que significa que la asignación a la variable es Medium o Large,
- Caso 3: $X[* \ 0]$ y $X[1 \ *]$ no implican simplificación, el primero equivale a utilizar Small y el segundo a utilizar Large.

Utilizando el ejemplo que corresponde a la Tabla 3.2, se puede observar que el bit 8 cumple la condición de minimización entre la R_1 y R_2, por lo que se reemplazan sus valores con $*$ y se obtiene la tabla de verdad simplificada que se muestra en la Tabla 3.3.

	Variable X_1		Variable X_2		Variable X_3		Variable X_4		
Regla	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8	Clase
R_3	0	1	0	0	0	1	0	*	1

Tab. 3.3.: Tabla de verdad simplificada perteneciente a la clase 1.

Así, las dos reglas anteriores se han transformado en una única regla que cumple con el modelo extendido de reglas difusas expuesto en la Ecuación 2.12. Luego de pasar por el bloque de "**Decodificación**" de la Figura 3.1, utilizando los casos expuestos en el paso 4 y recalculando el peso, la regla queda de la siguiente forma:

R_3 : **IF** X_1 is Medium and X_2 is Small and X_3 is Medium and X_4 is {Small or Medium} **THEN** B = C_1 with $w(R_3)$

donde $w(R_3)$ es el peso de la nueva regla. Este proceso se aplica a todas las clases, lo que permitiría obtener el conjunto final de reglas simplificado.

3.2.2. Minimización por Codificación Ternaria

En esta sección se describe la extensión a base ternaria del método de *QM*.

Codificación Ternaria de la Base de Reglas Difusas

Como se mencionó anteriormente, para utilizar el método de *QM* es necesario transformar la base de reglas en un formato compatible, donde se utilizan tablas de verdad binarias. En esta segunda parte de la propuesta, se ha adaptado el método de *QM* utilizando una codificación ternaria y se ha trabajado con conjuntos de reglas que utilizan variables difusas las cuales tienen asociados dominios difusos, en este

caso en particular con 3 etiquetas lingüísticas, *Small, Medium, Large*, utilizando el sistema de codificación expuesto en la Tabla 3.4.

Etiquetas	Código
Small	0
Medium	1
Large	2

Tab. 3.4.: Codificación ternaria de las variables lingüísticas.

Método de Minimización de Quine McCluskey Ternario

En este apartado se describen los pasos para la extensión a base ternaria del método de *QM*, los cuales se exponen a continuación:

1. Se convierte el antecedente de cada regla de la base de reglas en un vector y se le asigna a una tabla de verdad, con valores ternarios tomando como referencia el sistema de codificación de la Tabla 3.4. Se construye una tabla distinta para cada clase.

2. Se suman los valores de todas las variables correspondientes a cada fila de la tabla de valores ternarios obtenida en el paso anterior, donde cada fila representa el antecedente de cada una de las reglas, y se reordenan las filas en orden ascendente.

3. Se crea una columna con los *minterms* asociados al número de fila a la que pertenece cada regla.

4. Se comparan las reglas cuya suma solo difiera en una unidad para buscar elementos redundantes. Si tres reglas difieren, en su suma, en una unidad entre sí y tienen la misma variable con distintas etiquetas difusas, se marca esa variable con el símbolo "$*$" para indicar que es redundante. Se asocia la nueva regla con los *minterms* pertenecientes a las reglas combinadas.

5. Se describe un procedimiento para analizar los términos que no se pueden combinar con tres reglas, se comparan reglas cuyas sumas difieren en una unidad para buscar elementos redundantes. Si dos reglas tienen una misma variable con distintas etiquetas lingüísticas y su suma difiere en una unidad, se marca la variable como la unión de las etiquetas difusas correspondientes. Después se asocia la nueva regla con los *minterms* pertenecientes a las reglas

combinadas. Los nuevos valores codificados resultantes de esta operación se muestran en la Tabla 3.5.

Etiquetas	**Código**
Small	0
Medium	1
Large	2
Small + Medium + Large	*
Small + Medium	0-1
Medium + Large	1-2
Small + Large	0-2

Tab. 3.5.: Codificación ternaria ampliada.

6. Los términos que no se pueden combinar, se separan en otra tabla como implicantes primos con su respectivo *minterm* asociado.

7. Se eliminan los duplicados en la columna del *minterm*.

8. Se repite el proceso 4,5,6 y 7, agregando las nuevas variables correspondientes a la redundancia "$*$" del paso 4 y la unión de las etiquetas difusas correspondiente al paso 5, hasta que ninguna fila se puede combinar. Creando así una tabla con los implicantes primos.

9. Para encontrar los implicantes esenciales se crea una tabla de cobertura. Donde las filas corresponden a los *minterm* de las reglas combinadas denominadas implicantes primos del paso 8 y las columnas corresponden a los valores individuales de cada *minterm*. Los minterms correspondientes a las reglas redundantes son omitidos en este paso, no se colocan en las columnas. En la tabla de cobertura los minterms cubren los implicantes primos. Un implicante primo esencial es aquel implicante primo que cubre un minterm y ningún otro implicante primo cubre el mismo minterm.

10. Si los implicantes primos no esenciales no tienen un minterm en común con los implicantes primos esenciales, se vuelve a crear la tabla de cobertura del paso 9 de modo a extraer los nuevos implicantes primos esenciales. Esto se repite hasta no obtener más implicantes primos esenciales.

11. En el caso de que no exista ningún implicante primo esencial se aplica el método de Petrick's [86].

12. Se recalcula el peso de las nuevas reglas.

En el siguiente ejemplo se muestra cómo funciona el método. Se considera un conjunto de reglas difusas, todas pertenecientes a la clase C_1.

R_1 : **IF** X_1 is Small and X_2 is Small and X_3 is Medium and X_4 is Large **THEN** C_1 with $w(R_1)$
R_2 : **IF** X_1 is Small and X_2 is Medium and X_3 is Medium and X_4 is Large **THEN** C_1 with $w(R_2)$
R_3 : **IF** X_1 is Small and X_2 is Large and X_3 is Medium and X_4 is Large **THEN** C_1 with $w(R_3)$
R_4 : **IF** X_1 is Small and X_2 is Small and X_3 is Small and X_4 is Large **THEN** C_1 with $w(R_4)$
R_5 : **IF** X_1 is Medium and X_2 is Small and X_3 is Small and X_4 is Large **THEN** C_1 with $w(R_5)$
R_6 : **IF** X_1 is Large and X_2 is Small and X_3 is Small and X_4 is Medium **THEN** C_1 with $w(R_6)$

Se convierte la base de reglas en una tabla de codificación ternaria (correspondientes a los pasos 1,2 y 3), como puede apreciarse en la Tabla 3.6.

Regla	Minterm	Variable X_1	Variable X_2	Variable X_3	Variable X_4	Suma	Clase
R_4	1	0	0	0	2	2	1
R_1	2	0	0	1	2	3	1
R_5	3	1	0	0	2	3	1
R_6	4	2	0	0	1	3	1
R_2	5	0	1	1	2	4	1
R_3	6	0	2	1	2	5	1

Tab. 3.6.: Base de reglas con codificación ternaria.

Se puede observar que las reglas asociadas a los minterms 2, 5 y 6 tienen todas las etiquetas difusas posibles en la variable X_2 y en el resto de las variables son iguales, por lo que es posible simplificar dichas reglas (paso 4). De manera similar, las reglas asociadas al minterm 1 y 3 comparten dos etiquetas difusas en la variable X_1 y el resto de las variables son iguales, por lo que también pueden simplificarse según el modelo de reglas extendidas (paso 5). Se crea una tabla de implicantes primos utilizando el sistema de codificación ternaria ampliada de la Tabla 3.5.

Minterm	Variable X_1	Variable X_2	Variable X_3	Variable X_4	Clase
2,5,6	0	*	1	2	1
1,3	0–1	0	0	2	1
4	2	0	0	1	1

Tab. 3.7.: Implicantes primos.

En la Tabla 3.7 ya no se pueden realizar mas agrupaciones, a esta tabla se la denominará implicantes primos. A continuación, se completa la Tabla 3.8, que se denominara tabla de cobertura, con el fin de a encontrar los implicantes primos esenciales.

Minterm	1	2	3	4	5	6
2,5,6		x			x	x
1,3	x		x			
4				x		

Tab. 3.8.: Tabla de cobertura.

Se puede observar que cada implicante primo tiene un único valor en cada columna correspondiente a los minterms. Por esto, todos los implicantes primos son esenciales y la base de reglas esta simplificada. Luego de pasar por el bloque de "**Decodificación**" correspondiente a la Figura 3.1 y recalculando el peso, la base de reglas queda de la siguiente forma.

R_7 : **IF** X_1 is Small and X_3 is Medium and X_4 is Large **THEN** C_1 with $w(R_7)$

R_8 : **IF** X_1 is Small or Medium and X_2 is Small and X_3 is Small and X_4 is Large **THEN** C_1 with $w(R_8)$

R_9 : **IF** X_1 is Large and X_2 is Small and X_3 is Small and X_4 is Medium **THEN** C_1 with $w(R_9)$

En la primera regla, se eliminó la variable X_2 y en la segunda regla se usó el modelo de reglas difusas extendidas en la variable X_1. De esta forma, se obtiene la base de reglas simplificada a partir del conjunto original de reglas difusas.

A continuación se realizara la experimentación de ambos métodos propuestos, con el fin de determinar si la precisión se mantiene constante mejorando la interpretabilidad de la *BR* generada por el algoritmo de *WM-C*.

3.3. Análisis e interpretación de los resultados obtenidos

En esta sección, se llevarán a cabo dos experimentos con el objetivo de verificar la eficacia de los métodos propuestos mediante la codificación binaria y ternaria. El propósito principal de estos experimentos es determinar si dichos métodos son capaces de reducir el número de reglas difusas sin comprometer la precisión, en comparación con los resultados obtenidos por el algoritmo de *WM-C*. Además, se busca evaluar cuál de los dos métodos propuestos es más efectivo en la generación de reglas más simples y fácilmente interpretables.

Para llevar a cabo estos experimentos, se han seleccionado conjuntos de datos específicos, los cuales se describen en detalle en la Tabla de Anexo A.1, columna ($MinQM$). Estos conjuntos de datos fueron obtenidos del repositorio de la UCI [71]. Para la generación de la base de reglas (*BR*), se empleó el algoritmo de *WM-C* [14], que ha sido previamente descrito en este estudio.

La implementación del código se ha hecho en Python y se han utilizado dominios difusos compuestos por 3 etiquetas difusas con función de pertenencia triangular como se expone en la Figura 3.2.

Este estudio científico, como se muestra en la Tabla 3.9, se enfoca en comparar las bases de reglas generadas por el algoritmo *WM-C* con dos métodos de simplificación diferentes: la minimización de la base de reglas de *WM-C* utilizando el método de *QM* binario (*BIN*) y la minimización utilizando el método de *QM* ternario (*TER*). Se han evaluado dos parámetros de referencia: la capacidad de predicción (*Test*) y el número de reglas generadas (*#Reglas*). Para realizar este análisis, se ha empleado una validación cruzada con 10 particiones, lo que proporciona resultados más robustos y representativos.

En el análisis efectuado, se constata que tanto el enfoque binario como el ternario logran generar una reducción sustancial en la cantidad de reglas resultantes, logrando comprimirlas hasta alcanzar aproximadamente el 58 % del tamaño observado en el conjunto original de reglas. Este fenómeno se manifiesta de manera destacada en casos particulares, como en el conjunto de datos *magic*, en el cual se logra una reducción cercana al 66 % mediante la codificación binaria y alrededor del 69 % mediante la codificación ternaria, en relación con la base de reglas generadas por el algoritmo de *WM-C*. Con respecto a la capacidad predictiva, se observó una diferencia mínima en los promedios de ambos métodos comparados con el algoritmo de *WM-C*, alrededor del 0,01 %. La mayor diferencia de precisión a favor del algoritmo

Conjunto de Datos	WM-C		BIN		TER	
	Test	*#Reglas*	*Test*	*#Reglas*	*Test*	*#Reglas*
abalone	0,231	69,4	0,23	42,5	0,228	41,5
australian	0,798	313,9	0,783	179	0,783	139,6
automobile	0,611	109,7	0,602	86,4	0,577	81,4
balance	0,893	82	0,84	23,6	0,851	24,1
banana	0,602	7,9	0,58	4	0,558	4
bands	0,686	258,2	0,683	194	0,68	197,9
bupa	0,578	43,3	0,573	21,8	0,57	20,9
cleveland	0,38	239,2	0,354	194,2	0,374	186,7
crx	0,813	258,2	0,808	154,6	0,805	113,8
flare	0,567	127,6	0,58	81	0,492	62,2
german	0,196	885,2	0,196	836,1	0,196	824,8
heart	0,515	217,2	0,493	165,3	0,504	156,9
ionosphere	0,652	228,8	0,655	215,6	0,655	213,3
iris	0,926	14,7	0,94	7,9	0,94	7,9
led7digit	0,67	93,9	0,618	92,9	0,676	50,2
magic	0,764	313,1	0,744	103,7	0,744	95,4
mammographic	0,808	45,5	0,812	19	0,816	17
movement_libras	0,733	260,6	0,731	257,3	0,731	257,3
newthyroid	0,846	20,4	0,81	7,9	0,833	9,5
page-blocks	0,918	49,7	0,918	26,7	0,916	24,5
penbased	0,975	3281	0,973	1352,7	0,972	1427,5
phoneme	0,718	50,1	0,712	17,5	0,709	17,5
pima	0,725	101	0,695	43,9	0,697	41,8
ring	0,552	573	0,545	428,1	0,536	423,9
saheart	0,727	168,7	0,721	78,5	0,721	74,6
segment	0,843	318,5	0,849	139,3	0,852	136,8
shuttle	0,801	28,7	0,802	17,5	0,802	16,7
sonar	0,595	187,2	0,596	185,6	0,596	185,6
spambase	0,728	357	0,693	210,2	0,688	211,1
spectfheart	0,663	235,2	0,663	235,2	0,663	235,2
thyroid	0,92	463,6	0,91	268,8	0,919	190,8
vehicle	0,63	378,6	0,617	211,8	0,618	218,4
vowel	0,639	285,9	0,628	193,9	0,622	189,3
wine	0,926	124,3	0,921	86	0,926	81,8
winequality-white	0,495	294	0,494	133,4	0,481	132
Promedio	0,689	299,58	0,679	180,454	0,678	174,626

Tab. 3.9.: Resultados obtenidos por el algoritmo de *WM-C*, la minimización por codificación binaria (*BIN*) y la minimización por codificación ternaria (*TER*), utilizando una validación cruzada de 10 particiones y un dominio uniformemente distribuido con 3 etiquetas difusas. Donde (Test) es la precisión del conjunto de pruebas y (#Reglas) muestra el número medio de reglas obtenidas en cada método.

de *WM-C* con el caso binario, se encuentra en el conjunto de datos *led7digit* con una diferencia del 7,7 % y con el caso ternario en el conjunto de datos *flare* con un 13 %, este caso puede verse influenciado ya que este conjunto de datos no posee ni una variable continua, lo que puede afectar negativamente a la predicción del algoritmo de *WM-C*.

Para determinar si existen diferencias significativas se aplica una prueba de rangos con signo de Wilcoxon [96] con un intervalo de confianza del 95 %. Los resultados de esta prueba se muestran en la Tabla 3.10, que representa los resultados obtenidos en estas comparaciones, indicando las victorias (W), empates (T) y derrotas (L) junto con el p-valor calculado. Se determina que existen diferencias estadísticamente significativas en lo que respecta a la cantidad de reglas generadas por el algoritmo *WM-C* en comparación con *BIN*, presentando un p-valor de 0,0000003821017. Asimismo, se observan diferencias estadísticamente significativas en relación con *TER*, con un p-valor de 0,0000003819152. Estos resultados sugieren que el algoritmo *WM-C* genera, en promedio, un mayor número de reglas en comparación con los métodos de minimización. Además, al comparar la cantidad de reglas entre *BIN* y *TER*, se detecta una diferencia estadísticamente significativa, con *BIN* produciendo más reglas, obteniendo un p-valor de 0.001780204.

En cuanto a la precisión, el algoritmo *WM-C* exhibe una precisión significativamente mayor en comparación con *BIN*, con un p-valor de 0,0005202754, y en comparación con *TER*, con un p-valor de 0,0003527292. Sin embargo, al comparar la precisión entre los métodos de minimización *BIN* y *TER*, no se encuentran diferencias significativas con un p-valor de 0,8188696.

Reglas		
h	W/T/L	p-valor
WM-C vs BIN	35/1/0	0,0000003821017
WM-C vs TER	35/1/0	0,0000003819152
BIN vs TER	22/6/6	0,001780204
Test		
WM-C vs BIN	26/3/7	0,0005202754
WM-C vs TER	26/3/7	0,0003527292
BIN vs TER	15/10/11	0,8188696

Tab. 3.10.: Pruebas estadísticas que comparan los resultados obtenidos entre el algoritmo de WM-C y los métodos que utilizan la codificación binaria (*BIN*) y ternaria (*TER*). Donde (#Reglas) muestra el número medio de reglas obtenidas en cada método y (Test) es la precisión del conjunto de pruebas.

La Tabla 3.11 presenta un análisis de interpretabilidad entre los enfoques de minimización binaria y ternaria. La columna (*eliminadas*) muestra el promedio de variables irrelevantes detectadas por el algoritmo de minimización ternaria. Cabe

Conjunto de datos	*Eliminadas (TER)*	*Simp BIN*	*Simp TER*
abalone	0,9	0,612	0,596
australian	8,4	0,570	0,443
automobile	0,1	0,788	0,742
balance	25,0	0,288	0,218
banana	1,0	0,506	0,443
bands	2,8	0,751	0,766
bupa	2,3	0,503	0,474
cleveland	0,0	0,812	0,781
crx	10,3	0,599	0,438
flare	8,0	0,635	0,482
german	0,7	0,945	0,932
heart	0,7	0,761	0,722
ionosphere	1,6	0,942	0,932
iris	0,0	0,537	0,537
led7digit	0,0	0,989	0,535
magic	16,1	0,331	0,300
mammographic	6,5	0,418	0,345
movement_libras	0,0	0,987	0,987
newthyroid	1,8	0,387	0,448
page-blocks	3,1	0,537	0,487
penbased	29,2	0,412	0,435
phoneme	7,6	0,349	0,319
pima	8,7	0,435	0,403
ring	23,4	0,747	0,738
saheart	7,1	0,465	0,438
segment	10,6	0,437	0,428
shuttle	1,0	0,610	0,578
sonar	0,0	0,991	0,991
spambase	19,6	0,589	0,590
spectfheart	0,0	1,000	1,000
thyroid	13,1	0,580	0,410
vehicle	6,4	0,559	0,576
vowel	0,5	0,678	0,662
wine	1,6	0,692	0,657
winequality-white	14,2	0,454	0,445
Promedio	6,6	0,626	0,579

Tab. 3.11.: Estudio de interpretabilidad entre la minimización por codificación binaria (*BIN*) y la minimización por codificación ternaria (*TER*), donde (Eliminadas) es la cantidad media de reglas eliminadas en una validación cruzada de 10 particiones y (*Simp BIN*), (*Simp TER*) es la simplicidad media de cada conjunto de datos.

señalar que, como se mencionó anteriormente, la minimización binaria reduce el número de reglas pero no tiene la capacidad de eliminar variables. Las columnas *Simp BIN* y *Simp TER* muestran una métrica de simplicidad para cada conjunto de reglas generadas por los métodos binario y ternario, respectivamente. En particular, la simplicidad de la base de reglas se define en función de la reducción de condiciones en los antecedentes de las reglas. Esta métrica busca evaluar la complejidad y claridad de las reglas difusas resultantes y se define según la Ecuación 3.2.

$$Simp = \frac{(variables.reglas_{(BIN,TER)} - eliminadas)}{variables.reglas_{(CHI)}} \tag{3.2}$$

Donde, $variables.reglas_{(BIN,TER)}$ representa el producto de las variables en el conjunto de datos por la cantidad de reglas generadas mediante el método de codificación binaria o ternaria. El término $eliminadas$ denota la cantidad de variables eliminadas mediante el método de codificación ternaria, y $variables.reglas_{(CHI)}$ es el producto de las variables en el conjunto de datos por la cantidad de reglas generadas por el algoritmo de *WM-C*. El parámetro *Simp* se utiliza para indicar el grado de simplicidad en la base de reglas, donde un valor igual a 1 representa la ausencia de minimización en los métodos, y valores más bajos indican una base de reglas más simple.

Se observa que el método ternario efectivamente es capaz de detectar variables irrelevantes en los antecedentes de las reglas, aunque esa detección no es homogénea. En bases de datos como *iris*, *cleveland* o *sonar* no es capaz de encontrar ninguna variable irrelevante. La razón de esto puede deberse a que el método prioriza la reducción en el número de reglas, y no así a la eliminación de variables a la hora de realizar las agrupaciones. Por otro lado, podemos observar que la simplicidad medida como número medio de condiciones, es menor en el método ternario.

Para realizar una comparación exhaustiva entre los dos modelos, se llevó a cabo un análisis estadístico utilizando el test de Wilcoxon, tal como se describe en detalle en la Tabla 3.12. El objetivo primordial de este análisis fue determinar si el modelo *TER* superaba al modelo *BIN* en términos de simplicidad. Los resultados obtenidos revelaron un p-valor de 0,0001272731, lo que condujo a la confirmación de la

h	W/T/L	p-valor
BIN vs TER	27/4/5	0,0001272731

Tab. 3.12.: Prueba estadística que compara los resultados obtenidos en la simplicidad, entre el algoritmo de WM-C y sus versiones simplificadas utilizando la codificación binaria (*BIN*) y ternaria (*TER*).

hipótesis propuesta. Esta conclusión indica que el enfoque ternario presenta una base de reglas más interpretable, dado que su simplicidad es menor a la obtenida mediante el método binario.

3.3.1. Reflexiones

En ambos métodos de codificación, se ha logrado una reducción promedio del 58 % en la cantidad de reglas difusas en la base de reglas (*BR*), mientras se mantiene una precisión aceptable en comparación con el algoritmo *WM-C*. La codificación ternaria, aunque resulta en una cantidad similar de reglas en promedio en comparación con la codificación binaria, exhibe una ventaja en términos de simplicidad y facilidad de interpretación en la *BR* resultante.

No obstante, es imperativo destacar que estos métodos de codificación enfrentan desafíos significativos, especialmente cuando se aplican a conjuntos de datos que presentan un elevado número de variables. En tales situaciones, la complejidad del método aumenta de forma exponencial en proporción al número de variables, lo que puede restringir su aplicabilidad en el contexto de conjuntos de datos masivos. En el caso de la codificación binaria, el incremento en la cantidad de etiquetas lingüísticas conlleva un aumento en la cantidad de bits, intensificando así la complejidad a medida que se incrementa el número de variables. Asimismo, en el caso de la codificación ternaria, la tarea de eliminar por completo variables poco relevantes se torna más complicada a medida que se incrementa el número de etiquetas lingüísticas.

Por lo tanto, para futuras investigaciones y mejoras, se plantea explorar enfoques más eficientes y escalables que no dependan de manera exponencial del número de variables, lo que permitiría abordar conjuntos de datos más complejos y de mayor escala.

3.4. Contribuciones

- *A preliminary study to apply the Quine McCluskey algorithm for fuzzy rule base minimization.* (FUZZ-IEEE 2020) [54]
- *Una versión ternaria del algoritmo de Quine McCluskey para la minimización de base de reglas difusas.* (CAEPIA 20/21) [56]

4

Inferencia para clasificación en problemas con un número elevado de reglas difusas

En este capítulo abordamos el Objetivo General y el Objetivo Especifico 2 de la presente tesis doctoral. En concreto nos centramos en la problemática que podría aparecer al inferir con el conjunto de reglas que aprende el algoritmo de Wang y Mendel para clasificación (*WM-C*) [14], aunque el problema sería el mismo para inferir sobre un conjunto de reglas básicas obtenidas por otro procedimiento. El modelo de inferencia usado es el estándar de la regla ganadora. Este modelo consume un tiempo proporcional al número de reglas de la base de conocimiento, ya que, dado un ejemplo, comprueba el grado de adaptación con el antecedente de cada una ellas. Este proceso de inferencia se comporta de forma ágil cuando el número de reglas no es excesivo. Sin embargo, ese tiempo polinomial, dependiente del número de reglas, puede ser un problema en cuando hay muchas reglas. En concreto, el algoritmo de *WM-C* tiende a generar muchas reglas cuando trabaja sobre problemas que contienen muchos ejemplos. Por esta razón, en este capítulo exploramos estrategias y enfoques alternativos al procedimiento estándar de inferencia para obtener procesos con un comportamiento más ágil en estos escenarios.

La complejidad del método de inferencia de la regla ganadora (ver 2.2.2) depende de tres factores, el número de reglas, el número de ejemplos y el número de variables consideradas en el antecedente de la regla. Así, cuando el número de reglas o el número de ejemplos que se desean inferir no es grande, el tiempo necesario para realizar la inferencia es despreciable. Sin embargo, cuando alguno de estos factores es grande, el proceso puede llevar mucho tiempo y podrían explorarse métodos alternativos de inferencia que pudieran resultar más eficientes en estos casos.

Aunque pueda existir cierta relación entre los tres factores, por ejemplo, un gran número de ejemplos y variables pueden implicar un gran número de reglas, centraremos nuestro estudio en reducir la influencia del número de reglas en el método de inferencia, ya que tanto el número de variables como el número de ejemplos dependen del problema.

Hasta el momento en la literatura científica, el problema de realizar inferencias sobre un gran conjunto de reglas no ha sido abordado en profundidad. Así, por ejemplo en [9], se propone un modelo de inferencia que combina el algoritmo utilizado en los sistemas Mamdani [67] y TSK [87]. Sin embargo, este experimento se limita a trabajar únicamente con un conjunto de 256 reglas, sin abordar de manera explícita los desafíos que surgen al tratar con un gran número de reglas en problemas de clasificación. No fue hasta la aparición de conjuntos de datos masivos, considerados dentro del contexto de Big Data [81, 28], cuando se evidenció realmente la problemática relacionada con la inferencia en sistemas que generan un alto volumen de reglas. Estos métodos que en algunos casos utilizan el modelo MapReduce [25, 24] y el algoritmo de *WM-C*, aprenden un conjunto de reglas difusas de forma relativamente rápida, pero obteniendo un número de reglas muy elevado. Cuando es necesario realizar la inferencia con conjuntos de datos masivos y sobre un problema con un número de reglas significativo, el proceso de inferencia puede volverse extremadamente lento o incluso imposible de realizar en un tiempo aceptable [39].

Una forma de reducir la influencia del número de reglas en el proceso consiste en evitar la revisión de todas las reglas sobre cada ejemplo, centrando la búsqueda de la regla ganadora en el vecindario de la mejor regla asociada al ejemplo. La idea del vecindario, que se definirá más adelante, corresponde a aquellas reglas que tienen algún emparejamiento con el ejemplo, y que pueden construirse de forma sencilla a partir del mismo.

El principal problema del vecindario de un ejemplo, es que también podría contener un conjunto muy grande de reglas, por lo que es necesario definir ciertas heurísticas que permitan realizar la búsqueda de la mejor regla de la forma más eficientemente posible y reducir así el tiempo de respuesta del método de inferencia.

La idea básica es de este modo utilizar un método de exploración del vecindario para establecer un algoritmo de inferencia de reglas difusas [39]. Para ello se parte de un conjunto de reglas básicas obtenidas mediante algún procedimiento, en este caso en particular se utiliza el conjunto de reglas obtenidas por el algoritmo *WM-C*, y se establecen distintas propuestas de inferencia (basadas en distintas formas de explorar el vecindario de un ejemplo), así como un método híbrido de inferencia de reglas difusas que combina las diferentes propuestas realizadas en este trabajo y que se obtiene a partir de los resultados del estudio experimental.

En las siguientes secciones, se presentarán los fundamentos teóricos y conceptuales que respaldan la investigación realizada en este trabajo científico.

4.1. Preliminares

A continuación, se procede a analizar las posibles problemáticas que plantea la inferencia con reglas difusas en el contexto descrito. En primer lugar, se realiza una descripción detallada del proceso de inferencia, explicando el modelo de reglas difusas con peso utilizado y detallando las acciones necesarias a seguir en el proceso.

Como se expuso anteriormente en la ecuación 2.10, dado un conjunto de variables con sus correspondientes dominios y una variable consecuente Y, que representa la clase a aprender, se define el conjunto de todas las reglas difusas con peso como R^G. Cada regla tiene la forma *IF-THEN*, como se muestra en la ecuación 2.11., y por lo tanto, R^G está compuesto por todas las posibles reglas que se pueden generar con las n variables antecedente, la variable consecuente Y y los diferentes pesos asociados a cada regla.

Ahora, se considera un conjunto particular de reglas difusas para el cual queremos definir el proceso de inferencia, este conjunto podría ser obtenido por cualquier algoritmo de aprendizaje de reglas difusas, en este caso particular se utilizará el algoritmo de Wang y Mendel para clasificación (*WM-C*), ya que forma parte del objetivo de esta tesis. En cualquier caso los métodos que vamos a proponer son aplicables a cualquier conjunto de reglas obtenidas de diferentes formas, pero la elección del algoritmo *WM-C* es apropiada por las siguientes razones:

- Es un algoritmo de aprendizaje sencillo que puede ejecutarse muy rápido y, de hecho, se ha utilizado como algoritmo para abordar problemas de Big Data en los que el número de ejemplos es muy elevado.
- Genera reglas difusas que tienen en cuenta todos los atributos del problema y este hecho es importante para ilustrar la influencia del número de atributos en la inferencia, la cual consume mucho tiempo.
- Puede generar un número muy elevado de reglas para algunos problemas. Esto es muy importante para estudiar la eficacia de los distintos enfoques para realizar la inferencia.

El conjunto de reglas difusas generado por el algoritmo de *WM-C* se definirá como.

$$R^I = \{R_1, R_2, \ldots, R_m\} \tag{4.1}$$

y obviamente.

$$R^I \subseteq R^G \tag{4.2}$$

A partir de este conjunto de reglas R^I y un ejemplo dado $e = (e_1, e_2, \ldots, e_n)$, el razonamiento difuso permite obtener la clase asociada a este ejemplo, a partir del conjunto de reglas. El procedimiento para determinar la clase asociada al ejemplo es sencillo y se realiza utilizando el método de la regla ganadora, como se ha descrito en la sección 2.2.2. En este contexto, las T-normas, como el mínimo o el producto, se emplean con frecuencia, mientras que el producto se utiliza como el operador *Op* en el proceso de inferencia.

De esta manera, el modelo de inferencia de la regla ganadora utilizado en la clasificación puede resolverse mediante el proceso de optimización descrito en la ecuación 2.15. Así, al tomar un ejemplo $e = (e_1, e_2, \ldots, e_n)$, el cálculo de la clase asociada al ejemplo, dado el conjunto de reglas R^I, se puede resolver mediante un algoritmo que verifica secuencialmente la adaptación del ejemplo a cada regla, calcula el peso y selecciona el valor máximo. Este proceso puede entenderse, para cada ejemplo, como una búsqueda de la mejor regla utilizando la ecuación 2.15 como criterio de optimización.

Algoritmo 1 Inferencia estándar - Búsqueda Lineal

```
1: function STANDARD_INFERENCE(e, R^I)
2:     BestCurrentMatching = 0
3:     for i = 1 to m do
4:         CurrentMatching = 1
5:         for j = 1 to n do
6:             CurrentMatching = T(CurrentMatching, μ_{A^j_{i_j}}(e_j))
7:         end for
8:         CurrentMatching = Op(CurrentMatching, w(R_i))
9:         if CurrentMatching > BestCurrentMatching then
10:            BestCurrentMatching = CurrentMatching
11:            Class = ClassOfRule(i)
12:        end if
13:    end for
14:    return Class
15: end function
```

El Algoritmo 1 tiene una complejidad en tiempo de O(mxn), siendo m el número de reglas donde se realiza la inferencia y n el número de variables de cada regla, y todo el proceso se repite para cada ejemplo. Obviamente, la inferencia depende mucho del número de reglas que tenga el sistema. En problemas sencillos realmente

la inferencia se puede aplicar sin ningún inconveniente, sin embargo, hay problemas en los que el número de reglas es muy elevado.

Una mejora simple del algoritmo anterior es cambiar la búsqueda lineal de la mejor regla, por una búsqueda lineal con poda (ver Algoritmo 2). La idea es muy sencilla, durante la búsqueda se calcula el valor de la mejor adaptación entre el ejemplo y la regla calculada hasta el momento, y si el cálculo parcial actual de la adaptación de un componente del ejemplo con parte de la regla ya es peor que la mejor adaptación que se ha calculado, obviamente como se utiliza una t-norma, esa regla ya no puede competir con la mejor regla y se puede abandonar el cálculo del resto de las adaptaciones.

Algoritmo 2 Inferencia estándar - Búsqueda Lineal con Poda

```
1: function STANDARD_INFERENCE_PRUNNED(e, R^I)
2:     BestCurrentMatching = 0
3:     for i = 1 to m do
4:         CurrentMatching = w(R_i)
5:         while j ≤ n and CurrentMatching > BestCurrentMatching do
6:             CurrentMatching = T(CurrentMatching, μ_{A^j_{i_j}}(e_j))
7:             j = j + 1
8:         end while
9:         if CurrentMatching > BestCurrentMatching then
10:            BestCurrentMatching = CurrentMatching
11:            Class = ClassOfRule(i)
12:        end if
13:    end for
14:    return Class
15: end function
```

El Algoritmo 2 es una versión mejorada del algoritmo propuesto en [39], en el que se utiliza el peso propio para realizar la poda. Obviamente el resultado del Algoritmo 2 frente al Algoritmo 1 depende del orden en el que se exploran las reglas, sin embargo, en cualquier caso supone una mejora respecto a la versión básica ya que reduce el número de cálculos necesarios para obtener la clase de la regla ganadora, aunque en el peor de los casos sigue siendo O(mxn). A continuación, se exponen los distintos métodos de inferencia propuestos en esta tesis doctoral.

4.2. Métodos propuestos

Se plantea explorar formas alternativas de realizar la inferencia que puedan reducir el tiempo de cálculo utilizado por los Algoritmos 1 y 2. Fundamentalmente estas

mejoras pueden interpretarse como formas alternativas de resolver el problema de optimización descrito por la Ecuación 2.15. La idea de poda planteada en el Algoritmo 2 es una idea interesante que se mantendrá en las sucesivas mejoras que se proponen.

4.2.1. Búsqueda retroactiva con poda en el vecindario del ejemplo

Para lograr un avance significativo en la mejora del proceso de inferencia cuando se enfrenta a un número considerable de reglas, resulta esencial realizar un cambio radical en el método de búsqueda. En este sentido, se propone una idea central que implica la modificación del espacio de búsqueda. En lugar de considerar el espacio tradicional de búsqueda, representado por R^I, se propone realizar la búsqueda sobre el espacio completo R^G.

Aunque parezca sorprendente, buscar en un espacio mayor puede reducir el tiempo de respuesta, la razón es que la búsqueda se centrará en un conjunto de reglas en el vecindario del ejemplo. Además, la búsqueda comienza en un punto concreto de este espacio que es la regla que mejor se ajusta al ejemplo.

Para ello, dado un ejemplo e, el conjunto de reglas que tiene una adecuación estrictamente positiva con el ejemplo se define por

$$N(e) = \{R \in R^G \mid A(R, e) > 0\} \tag{4.3}$$

$N(e)$ contiene todas las reglas que cubren el ejemplo e en un grado positivo, por lo que contiene las reglas que podrían ser activadas por ese ejemplo, o en otras palabras, el conjunto R^G - $N(e)$ corresponde a las reglas que no pueden ser activadas de ninguna manera por el ejemplo. Llamamos $N(e)$ al vecindario del ejemplo. Los elementos de $N(e)$ no tienen por qué ser reglas del R^I y, por lo tanto, no tienen que ser reglas que participen en la inferencia del ejemplo, es decir,

$$N(e) \not\subseteq R^I \tag{4.4}$$

Sin embargo, la regla ganadora debe pertenecer necesariamente al conjunto $N(e)$, ya que necesita tener una adaptación positiva con el ejemplo, para ser la solución al proceso de optimización definido por la Ecuación 2.15. Por lo tanto, la regla ganadora se encuentra en el conjunto.

$$N(e) \cap R^I \tag{4.5}$$

Una regla relevante del conjunto $N(e)$, es la regla que contiene la mejor adaptación de cada componente del ejemplo con cada variable antecedente de la regla. Esta regla que llamamos *regla central* de $N(e)$ y que denotamos por $C(e)$, no tiene por qué pertenecer al conjunto R^I.

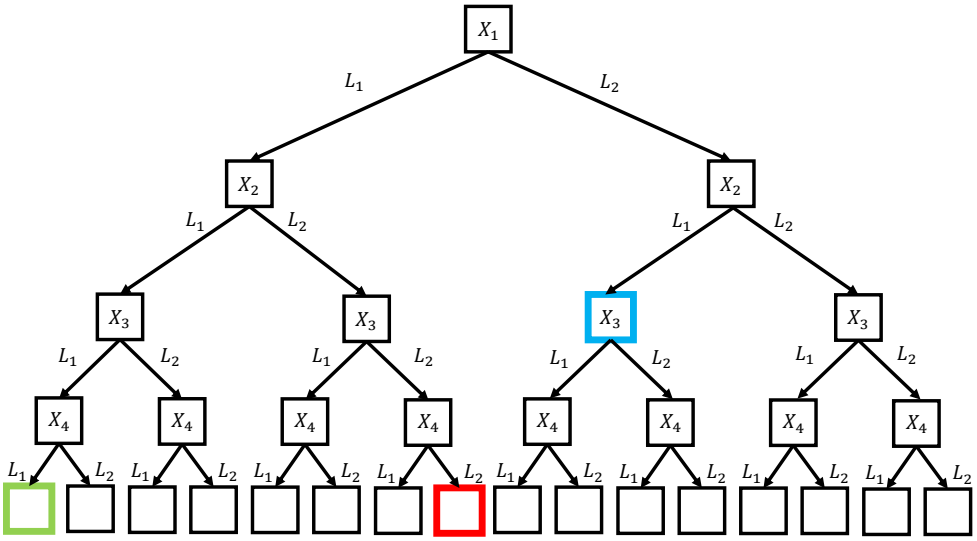

Fig. 4.1.: Árbol que representa el conjunto $N(e)$ donde X_i son las variables antecedentes, L_1 es la etiqueta que mejor se adapta al ejemplo y L_2 es la etiqueta con la peor adaptación al ejemplo, para problemas en los que para cada valor real del dominio sólo hay dos etiquetas con adaptación positiva. La primera de la izquierda representa el antecedente de la *regla central* ($C(e)$), la cual se simboliza con un recuadro en color verde.

La idea es definir un proceso de búsqueda sobre el conjunto $N(e)$, teniendo siempre en cuenta que si la regla explorada no pertenece al conjunto R^I, no se considerará, y tomando como punto de partida la regla central del conjunto $N(e)$.

Así con la idea de explorar el conjunto $N(e)$ para seleccionar la regla ganadora, se representa este conjunto mediante un árbol. Este árbol tiene en cada nodo una variable, y en cada arco un posible valor de esta variable, entre aquellos valores (etiquetas difusas) que tiene adecuación positiva con el ejemplo, representando la asignación de este valor a la variable del nodo. Los nodos hoja de este árbol representan los antecedentes de todas las reglas posibles de $N(e)$.

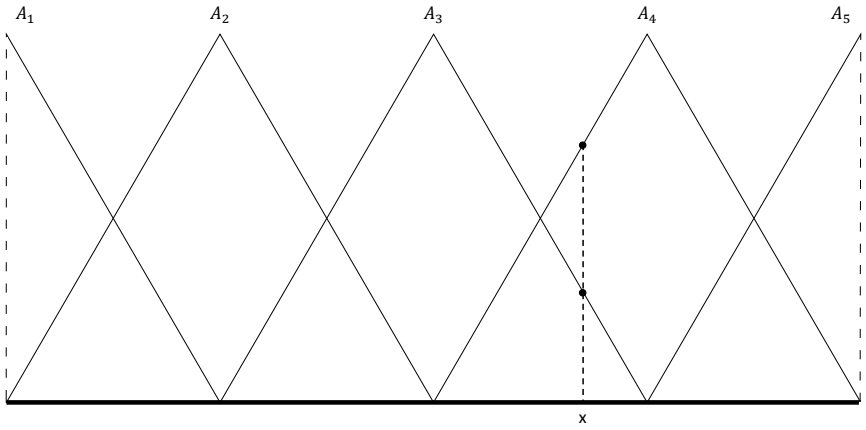

Fig. 4.2.: Un dominio para el que cada valor real tiene adaptación positiva con exactamente dos etiquetas difusas excepto en los puntos finales.

La Figura 4.1 representa un ejemplo en el que tenemos 4 variables antecedentes X_1, X_2, X_3 y X_4, donde por simplicidad en la representación se supone que para cada valor real del dominio, solo hay dos etiquetas con adaptación positiva (ver Figura 4.2) excepto en los puntos finales, como ocurre por ejemplo cuando tomamos dominios donde las etiquetas están uniformemente distribuidas. Denotaremos para L_1 la etiqueta (entre las etiquetas difusas del dominio de la variable correspondiente) con la mejor adaptación al ejemplo y para L_2, la etiqueta con la peor adaptación al ejemplo. Cuando un valor tiene adaptación con más de dos etiquetas, la idea es la misma y se utiliza los literales L_i según sea necesario, y L_1 representaría la etiqueta con la mejor adaptación, L_2 el segundo mejor, y así sucesivamente.

En la Figura 4.1 se muestra el árbol y cada nodo interno representa un antecedente de una regla parcialmente asignada. Por ejemplo, el nodo interno del árbol con un cuadrado en color verde, representa la *regla central* y el cuadrado de color azul, representa la asignación a las variables antecedentes de los siguientes valores.

$$(X_1 = L_2,\ X_2 = L_1,\ X_3 = Sin\ asignar, X_4 = Sin\ asignar) \tag{4.6}$$

por lo tanto, una regla con el siguiente antecedente.

$$\textbf{IF}\ X_1\ is\ L_2\ and\ X_2\ is\ L_1\ \textbf{THEN}\ldots \tag{4.7}$$

Sin embargo, cada nodo de hoja representa una asignación completa de valores a las cuatro variables, como por ejemplo, el nodo de hoja del árbol con un cuadrado en color rojo en la Figura 4.1, representa la asignación a las variables antecedentes de lo siguiente valores

$$(X_1 = L_1,\ X_2 = L_2,\ X_3 = L_2,\ X_4 = L_2) \tag{4.8}$$

y por lo tanto una regla con el siguiente antecedente.

$$\textbf{IF}\ X_1\ is\ L_1\ and\ X_2\ is\ L_2\ and\ X_3\ is\ L_2\ and\ X_4\ is\ L_2\ \textbf{THEN}\ldots \tag{4.9}$$

Este árbol es una representación completa de todos los antecedentes de las reglas de $N(e)$ a través del nodo hoja del árbol. Se utiliza esta representación para describir los distintos algoritmos de búsqueda que representan el proceso de inferencia. La descripción detallada del árbol, donde se asocia cada nivel con una nueva asignación de valor a una variable, permite una fácil implementación de la poda definida en el Algoritmo 2 durante el proceso de búsqueda.

La primera propuesta de cálculo de la regla ganadora basada en la exploración del conjunto $N(e)$ que se propone, se describe mediante el Algoritmo 3 y utiliza un algoritmo de búsqueda retroactiva sobre el árbol asociado a $N(e)$, con las siguientes particularidades:

- En cada nodo de la búsqueda se tiene una asignación parcial de valores a variables (la asignación es completa en los nodos hoja) y se utiliza esta asignación parcial para calcular la adaptación parcial entre el ejemplo y esa asignación, cuando ese valor es inferior al valor de la mejor asignación completa obtenida hasta el momento, se poda el nodo, es decir, el proceso de búsqueda se detiene en ese nodo y provoca un retroceso. Como se mencionó anteriormente, todas las reglas que podrían obtenerse a partir de ese nodo son peores que otra que ya se ha encontrado.
- Cuando durante la búsqueda se llega a un nodo hoja, entonces se tiene una asignación completa, en ese momento se comprueba si existe una regla en R^I con ese antecedente, si no se encuentra en ese conjunto, el nodo hoja se ignora, en caso contrario se considera en el cálculo de la regla ganadora. Se ha utilizado una tabla hash en la implementación de las reglas del conjunto R^I, lo que permite comprobar de manera muy eficiente si el antecedente de una regla en un nodo hoja pertenece o no a R^I.

Algoritmo 3 Búsqueda retroactiva con poda en el vecindario del ejemplo

```
1: function NEIGHBORHOOD_INFERENCE(e, R^I, MaxRules)
2:     vector σ[n]
3:     for i = 1 to n do
4:         σ[i] = i
5:     end for
6:     return BackTrack (1,∅,1.0,0.0,∅, MaxRules)
7: end function
8: function BACKTRACK(i, CurrentRule, CurrentAdapt, &BestAdapt, &CurrentClass,
   &MaxRules)
9:     if (MaxRules ≤ 0) then
10:    else if (CurrentAdapt ≤ BestAdapt) then
11:        return
12:    else if (i == n) then
13:        MaxRules = MaxRules - 1
14:        if (CurrentRule ∈ R^I) then
15:            w = WeightOf(CurrentRule)
16:            CurrentAdapt = Op(CurrentAdapt,w)
17:            if (CurrentAdapt > BestAdapt) then
18:                BestAdapt = CurrentAdapt
19:                CurrentClass = ClassOf(CurrentRule)
20:            end if
21:        end if
22:        return CurrentClass
23:    end if
24:    for j = 1 to |D_i| do
25:        if (μ_{A_j^i}(e_i) >0) then Push_back(v,A_j^i)
26:        end if
27:    end for
28:    Sort(v)
29:    for k = 1 to |v| do
30:        CurrentClass = BackTrack (σ[i+1], Append(CurrentRule, X_i is A_k^i), T(CurrentAdapt,
   A_k^i(e_i)), BestAdapt, CurrentClass,MaxRules)
31:    end for
32:    return CurrentClass
33: end function
```

Debido a la estructura del árbol previamente construido, el proceso de búsqueda Retroactiva (Algoritmo 3) comienza explorando la regla central del ejemplo, y luego continúa con las reglas vecinas, descartando aquellas que no pertenecen al conjunto de reglas de la inferencia R^I, o aquellas de las que se sabe con certeza que tendrán una adaptación inferior que la mejor encontrada hasta el momento.

El parámetro $MaxRule$ usado en el algoritmo, establece el máximo número de reglas que este proceso de inferencia comprobará en el vecindario de la regla

central. Cuando $MaxRule = \infty$, no se considera ninguna limitación en el número de reglas.

En la implementación del Algoritmo 3 se utilizaron dos funciones: una función principal no recursiva (llamada Neighborhood_Inference) que recibe los parámetros anteriores y utiliza a su vez una función auxiliar recursiva (llamada BackTrack), que es la que realmente implementa la inferencia. Con la idea de que la función recursiva pueda ser utilizada en la descripción de los siguientes algoritmos, se utilizará una función σ que asocia cada índice i consigo mismo. Ésta es realmente importante en el siguiente algoritmo.

Los parámetros utilizados en la función BackTrack son:

- i es el índice de la variable que se está evaluando en cada momento, para $1 \leq i \leq n$. Como σ mantiene el índice sin cambios, las variables se procesan en el orden por defecto, es decir, primero la primera variable, luego la segunda y así sucesivamente. En el siguiente algoritmo se cambiará este orden utilizando la función σ.

- La variable $CurrentRule$ almacena el antecedente de la regla a evaluar. Inicialmente es una cadena vacía, y el antecedente se obtiene añadiendo asignaciones de valores a variables, siguiendo el movimiento descendente a través del árbol asociado a $N(e)$. Cuando $i{=}n$, se ha terminado la asignación y se obtiene un antecedente completo de una regla.

- La variable $CurrentAdapt$ es el grado de adaptación del antecedente descrito en $CurrentRule$ de la primera a la i-esima variable.

- La variable $BestAdapt$ almacena el valor de la mejor adaptación de un antecedente completo entre todos lo que se han evaluado previamente. Inicialmente su valor es cero.

- La variable $CurrentClass$ se utiliza para almacenar el consecuente o la clase de la regla que proporcionó el valor de $BestAdapt$.

- La variable $MaxRule$ es el número máximo de reglas en el vecindario del ejemplo a explorar.

Algunos de los argumentos de la función BackTrack tienen un símbolo & que indica que esos argumentos se pasan por referencia, mientras que los que no lo tienen se pasan por valor. Esta distinción entre ambos tipos de paso de argumentos es relevante para el comportamiento del proceso recursivo.

Por último, también se utiliza en el algoritmo el ejemplo para el que se está realizando la inferencia e, siendo e_i el i-ésimo componente de este ejemplo, los dominios de las variables antecedentes, siendo D_i el dominio de la i-ésima variable y R^I el conjunto de reglas utilizadas en el proceso de inferencia.

A continuación, describimos el proceso en detalle. Hay tres condiciones de parada.

- La primera utiliza el parámetro de $MaxRules$ y se explicará más adelante.
- La segunda se aplica cuando la adaptación obtenida hasta la variable i, es decir $CurrentAdapt$, no supera la adaptación de la mejor regla existente en la base de reglas ya encontrada, es decir, el valor de $BestAdapt$. En este caso, se hace una poda y se va buscar otra opción.
- La última condición de parada se produce cuando todas las variables del antecedente (i=n) están instanciadas. En este caso, se comprueba si la regla con el antecedente actual está en el conjunto de reglas R^I. Si lo es, se toma el peso de la regla y se calcula la función h a partir de la Ecuación 2.15. Si el valor de la función h es mayor que el de $BestAdapt$, entonces $BestAdapt$ se modifica con esta nueva adaptación y la clase es la clase de esta regla. En caso contrario, esta regla no se considera como alternativa para dar salida a la inferencia y se ignora.

Cuando ninguna de las tres condiciones de parada es verdadera, entonces todas las etiquetas de la variable X_i que tienen adaptación con el componente i del ejemplo, se toman y se ordenan de mayor a menor en un vector v. Para cada una de las etiquetas con adaptación distinta de cero de la variable i-esima, se realiza una llamada recursiva (paso 30) utilizando ahora la siguiente variable (X_{i+1}), y los siguientes parámetros.

- Una nueva $CurrentRule$ en la que, al antecedente definido por la $CurrentRule$ anterior, se le añade una nueva componente de antecedente, tal que "X_i es L_k", siendo L_k la etiqueta k-ésima del dominio D_i reordenado,
- Una nueva variable $CurrentAdapt$ definida como la t-norma entre la adaptación previamente calculada y la adaptación entre la componente del ejemplo con la etiqueta k-ésima,
- y finalmente las variables $BestAdapt$ y $CurrentClass$.

La ordenación descrita en las líneas 24 a 28 del Algoritmo 3 consiste en detectar aquellas etiquetas difusas con adecuación positiva con la componente del ejemplo,

y una vez detectadas, ordenarlas de forma decreciente siguiendo la estructura del árbol descripto anteriormente.

Ahora se presentará la justificación del primer criterio de parada utilizado en el algoritmo. Este algoritmo tiene la capacidad de explorar las reglas en el vecindario de un ejemplo, como se ha mencionado anteriormente, y es importante destacar que estas reglas no necesariamente pertenecen al conjunto R^I. En muchos casos, esta búsqueda resulta más eficiente que explorar directamente el conjunto R^I. Sin embargo, cuando el ejemplo no está cubierto por ninguna regla de dicho conjunto, el proceso se vuelve ineficaz, ya que requeriría explorar todas las reglas en el vecindario del ejemplo antes de determinar que ninguna regla es aplicable. Para evitar este problema, se introduce el parámetro $MaxRules$, el cual limita el número de reglas que el algoritmo explora. Cuando $MaxRules$ se toma como infinito, la salida de este algoritmo debe ser exactamente la misma que la de los dos algoritmos anteriores, salvo posibles empates en el valor de la función h. Esto puede deberse a que en los dos modelos anteriores existe un orden entre las reglas, y ese orden permite resolver empates. En el nuevo algoritmo no se tiene un orden de las reglas en el sentido anterior, por lo que se puede resolver de diferentes maneras. Para valores pequeños del parámetro, el algoritmo se convierte en un método aproximado para hacer la inferencia.

4.2.2. Búsqueda heurística retroactiva con poda en el vecindario del ejemplo

El Algoritmo 3 es el modelo de referencia para todos los procesos de búsqueda en el vecindario que se han definido en este trabajo. Una primera mejora de este algoritmo está relacionada con el uso de una función heurística que permite explorar las variables X_i en un orden distinto al predeterminado. El orden en el que se exploran las variables influye obviamente en el proceso de poda, así una vez que se ha calculado la adaptación parcial entre el componente del ejemplo y las diferentes etiquetas difusas de las variables, si la adaptación entre la etiqueta que mejor se adapta y la adaptación entre la etiqueta que peor se adapta es muy grande, eso significa que el uso de esta variable es más interesante que si la diferencia entre las adaptaciones es menor. La razón es que cuanto mayor es la diferencia, mayor es el valor de la mejor adaptación y, por tanto, se comienza explorando las variables con las etiquetas que tienen la mejor adaptación, según la heurística descrita a continuación:

$$\Delta(i) = max_j(\mu_{A_j^i}(e_i)) - min_j(\mu_{A_j^i}(e_i)) \tag{4.10}$$

El Algoritmo 4 es una versión adaptada del Algoritmo 3 en las que se explora las variables no en un orden predeterminado sino en un orden decreciente basado en la heurística Δ. La reordenación de las variables hace uso de la función σ que registra el orden más conveniente para su exploración siguiendo la idea de la heurística.

Algoritmo 4 Búsqueda heurística retroactiva con poda en el vecindario del ejemplo

1: **function** HEURISTIC_NEIGHBORHOOD_INFERENCE(e, R^I, $MaxRules$)
2: vector $\sigma[n]$, $\Delta[n]$
3: **for** $i = 1$ to n **do**
4: $\sigma[i] = i$
5: $\Delta(i)$=$max_j(\mu_{A_j^i}(e_i))$ - $min_j(\mu_{A_j^i}(e_i))$
6: **end for**
7: Sort(σ) using Δ heuristic
8: **return** BackTrack (σ[1],$\emptyset$,1.0,0.0,$\emptyset$, $MaxRules$)
9: **end function**

4.2.3. Búsqueda heurística retroactiva con poda en el vecindario cercano del ejemplo usando una medida de distancia

El último de los algoritmos que proponemos basado en la exploración del vecindario del ejemplo se inspira en el Algoritmo 4, pero añade un nuevo criterio heurístico. En este caso se mantiene la exploración del vecindario del ejemplo, el uso de un proceso de poda y de la heurística Δ descripta anteriormente. La nueva propuesta comienza explorando la regla central del ejemplo y a partir de ella explora las reglas que se han ordenado heurísticamente de mejor a peor, pero ahora en lugar de explorar todas las reglas del vecindario, la búsqueda se limita a un vecindario cercano a la regla central, y la idea de cercanía vendrá determinada por un parámetro de distancia.

Para ello definimos una medida de distancia entre reglas. Dadas dos reglas del conjunto R^G

$$R_1 : \textbf{IF } X_1 \text{ is } A_{r_1}^1 \text{ and } X_2 \text{ is } A_{r_2}^2 \text{ and } \ldots \text{and } X_n \text{ is } A_{r_n}^n$$

$$\textbf{THEN } Y \text{ is } B_1 \text{ with } \textbf{weight } w(R_1)$$

$$R_2 : \textbf{IF } X_1 \text{ is } A^1_{s_1} \text{ and } X_2 \text{ is } A^2_{s_2} \text{ and } \ldots \text{and } X_n \text{ is } A^n_{s_n}$$

$$\textbf{THEN } Y \text{ is } B_2 \text{ with } \textbf{weight } w(R_2)$$

Se define la distancia entre las dos reglas del siguiente modo

$$d(R_1, R_2) = \sum_{i=1}^{n} \delta_i(R_1, R_2) \tag{4.11}$$

donde

$$\delta_i(R_1, R_2) = \begin{cases} 0 \quad \text{si} \qquad A^i_{r_i} = A^i_{s_i} \\ \\ 1 \qquad \text{en caso contrario.} \end{cases} \tag{4.12}$$

De este modo, la distancia entre dos reglas se define como el número de variables antecedentes que tienen diferente asignación a cada regla, este concepto es el análogo al utilizado en la distancia de Hamming [42]. La definición del vecindario cercano hace uso de la distancia a la regla central de la siguiente manera:

$$N^d(e) = \{R \in N(e) | d(R, C(e)) \leq d\} \tag{4.13}$$

casos especiales de esta definición serian:

$$N^0(e) = \{C(e)\} \tag{4.14}$$

y

$$N^\infty(e) = N(e) \tag{4.15}$$

Es decir, cuando la distancia es cero solo se obtiene en el conjunto la regla central y cuando la distancia es infinita el conjunto es todo $N(e)$.

Pues bien, la idea básica del Algoritmo 5 es mantener los elementos que se han añadido en los Algoritmos 3 y 4, pero haciendo uso de un parámetro d que limita la búsqueda al espacio $N^d(e)$, con la idea de hacer más eficiente el proceso de búsqueda y haciendo uso del criterio heurístico, de que la proximidad de la regla central tiene influencia directa en la detección de la regla ganadora.

En la descripción de este algoritmo se cambiara la estructura de la función recurrente BACKTRACK por la nueva función BACKTRACK2, para incluir el parámetro distancia y utilizarlo convenientemente.

Algoritmo 5 Búsqueda heurística retroactiva con poda en el vecindario cercano del ejemplo definido por una medida de distancia

function HEURISTIC_NEARBY_NEIGHBORHOOD_INFERENCE(e, R^I, d)
  vector $\sigma[n]$, $\Delta[n]$
  for $i = 1$ to n **do**
    $\sigma[i] = i$
    $\Delta(i)$=$max_j(\mu_{A^i_j}(e_i))$ - $min_j(\mu_{A^i_j}(e_i))$
  end for
  Sort(σ) using Δ heuristic
  return BackTrack2 (σ[1],$\emptyset$,1.0,0.0,$\emptyset$, d)
end function
function BACKTRACK2(i, $CurrentRule$, $CurrentAdapt$, $\&BestAdapt$, $\&CurrentClass$, $Distance$)
  if ($Distance$ <0) **then**
    return
  else if ($CurrentAdapt \leq BestAdapt$) **then**
    return
  else if ($i == n$) **then**
    if ($CurrentRule \in R^I$) **then**
      w = WeightOf($CurrentRule$)
      $CurrentAdapt$ = Op($CurrentAdapt$,w)
      if ($CurrentAdapt > BestAdapt$) **then**
        $BestAdapt = CurrentAdapt$
        $CurrentClass$ = ClassOf($CurrentRule$)
      end if
    end if
    return $CurrentClass$
  end if
  for $j = 1$ to $|D_i|$ **do**
    if ($\mu_{A^i_j}(e_i) > 0$) **then** Push_back(v,A^i_j)
    end if
  end for
  Sort(v)
  for $k = 1$ to $|v|$ **do**
    $CurrentClass$ = BackTrack2($\sigma[i+1]$, Append($CurrentRule$, X_i is A^i_k), $T(CurrentAdapt, A^i_k(e_i), BestAdapt, CurrentClass, Distance - (k-1))$
  end for
  return $CurrentClass$
end function

4.3. Análisis e interpretación de los resultados obtenidos

Una vez que se han descrito las diferentes propuestas para realizar inferencia sobre reglas difusas en problemas de clasificación, en esta sección se analizará el comportamiento de estos métodos. Este estudio se realiza en varios pasos, primero se analizará el comportamiento de los algoritmos que se pueden considerar exactos, es decir que devuelven exactamente la clase ganadora. Este primer estudio debería involucrar a los Algoritmos 1, 2, 3 y 4, aunque finalmente no se considerará el Algoritmo 1, ya que sus resultados son iguales que los producidos por el Algoritmo 2 en cuanto a la precisión, y los tiempos de ejecución son significativamente peores que los obtenidos por el segundo algoritmo. El segundo paso involucra al Algoritmo 5, ya que no es un algoritmo exacto, es un algoritmo aproximado en el que la salida del algoritmo puede no ser la regla ganadora. Una vez completados estos dos pasos se propondrá un enfoque híbrido que combine los algoritmos anteriores y obtenga mejores resultados.

Además, en el estudio se distinguirá entre el algoritmo basado en una búsqueda secuencial sobre el conjunto de reglas, es decir, el Algoritmo 2, al que se referirá como enfoque secuencial (el Algoritmo 1 también se incluye en este enfoque), y los algoritmos basados en una búsqueda en el vecindario de la regla central del ejemplo (Algoritmos 3 y 4) a los que se hará referencia como enfoque recursivo.

Ambos algoritmos del enfoque recursivo hacen uso del parámetro $MaxRules$ de la función BACTRACK, que determina el número máximo de reglas que se exploran en el vecindario de la regla central del ejemplo antes de abandonar la búsqueda. Este parámetro tomará el valor de 1024 a lo largo de la experimentación y se justificará este valor más adelante.

En la experimentación realizada se utilizaron los conjuntos de datos correspondientes a la Tabla del Anexo A.1, columna (Inf), la cual contiene los 50 problemas de clasificación extraídos del repositorio de aprendizaje automático de la UCI [71]. Todos los conjuntos de datos seleccionados son ampliamente conocidos y utilizados en la comunidad científica. Además, estos conjuntos de datos abarcan un amplio espectro de situaciones diferentes en el aprendizaje automático, incluyendo problemas binarios, multiclase con un gran número de clases, problemas con pocas o muchas variables, problemas con pocos o muchos ejemplos, entre otros. Esto permite un análisis exhaustivo y representativo del comportamiento de los métodos propuestos.

A lo largo del presente estudio experimental, se emplearán para definir el dominio de las variables continuas, cinco conjuntos difusos triangulares uniformemente distribuidos, con un punto de corte en 0,5 para realizar la discretización en los diversos conjuntos de datos considerados. Estos conjuntos de datos poseen dominios similares a los que se encuentran descritos en la Figura 4.2.

El orden de complejidad de una única inferencia de los algoritmos del enfoque secuencial depende del número de reglas y del número de variables implicadas en dichas reglas. El orden de complejidad de los algoritmos del enfoque recursivo depende exponencialmente del número de variables continuas. En particular, el orden es $O(2^C)$, donde C es el número de variables continuas del problema en particular, ya que dado el modelo de discretización que se ha utilizado, un valor real tiene una adaptación con a lo sumo dos etiquetas difusas en su dominio. Para obtener el conjunto de reglas con el que realizar la inferencia en los diferentes conjuntos de datos se utilizará el algoritmo de *WM-C*.

Todos los experimentos se han realizado en un ordenador con 8 procesadores IntelCore i7-6700 y 15.6 Gb. de memoria, funcionando con el sistema operativo Ubuntu 20.04.2 de 64 bits.

En particular, la Tabla 4.1 recoge los resultados obtenidos por el algoritmo de *WM-C* utilizando una validación cruzada con 10 particiones, donde la columna (*Reglas*) representa el número medio de reglas obtenidas y *Train* es la precisión en el conjunto de entrenamiento.

Los datos mostrados en la columna *Train* se obtienen directamente del algoritmo de aprendizaje y no se utiliza un algoritmo de inferencia. El algoritmo de *WM-C* coloca una cuadrícula en el espacio de búsqueda y cuenta los ejemplos que caen en cada cuadrícula. Cuando todos los ejemplos han sido colocados en la cuadrícula correspondiente, mira sobre cada cuadrícula y asocia como una clase, la clase mayoritaria entre los ejemplos que están en esa cuadrícula. De esta forma se sabe que ejemplos están bien clasificados (los que son de la clase mayoritaria), y los que estarán mal clasificados.

Algunas conclusiones que se pueden extraer de la experimentación recogida en la Tabla 4.1 son las siguientes:

- El algoritmo de *WM-C* puede generar un número elevado de reglas. Los ejemplos más destacados en los que podemos observar este elevado número de reglas son *hepmass* que cuenta con más de 9 millones de reglas e *higgs* con más de 8 millones y medio de reglas.

Conjunto de Datos	Reglas	Train
abalone	207,3	31,15
adult	19796,4	89,99
australian	500,9	96,81
automobile	124,3	100
balance	562,5	100
banana	19,6	77,08
bands	326,1	100
bupa	120,8	76,52
census	80968,1	99,26
cleveland	258,4	98,99
coil2000	7532,2	98,74
connect-4	60801,3	100
covtype*	23087,9	84,91
crx	522,7	98,47
fars*	39933,7	100
flare	269,8	85,27
german	898,4	100
heart	235,6	100
hepmass	9404420	99,9
higgs	8706010	95,54
ionosphere	273,4	100
iris	41,8	95,33
kddcup*	2868	99,95
led7digit	82,1	79,6
letter	7585,2	88,15

Base Datos	Reglas	Train
magic	1912,4	84,01
mammog	160,3	87,35
movement	294,6	100
newthyroid	47,2	95,35
page-blocks	163,8	93,82
penbased	7739	99,87
phoneme	228,2	82,38
pima	426,6	90,49
ring	5060,3	99,34
saheart	354	96,75
satimage	2149,3	78,21
segment	888,4	95,76
shuttle	73,8	91,59
sonar	187,2	100
spambase	1379,8	86,67
spectfheart	240,3	100
susy	274477	70,39
texture	3697,8	97,69
thyroid	943,9	94,25
twonorm	6654,2	99,99
vehicle	712,7	98,58
vowel	649,7	98,69
wine	159,4	100
wineq-red	771	83,99
wineq-white	1086,8	66,35

Tab. 4.1.: Resultados obtenidos por el algoritmo *WM-C* utilizando una validación cruzada de 10 particiones. La columna (Reglas) muestra el número medio de reglas obtenidas y (Train) la precisión en el conjunto de entrenamiento medido en %.

- Si se calcula la proporción de ejemplos por reglas, se observa que la mayoría de los conjuntos de datos tiene un valor inferior a 2 (concretamente 27 de los 50 conjuntos de datos). Este hecho indica que para estos problemas, las reglas se apoyan en muy pocos ejemplos. Solo 8 de los 50 tienen un valor superior a 15 para este indicador.

Como ya se ha comentado en las secciones anteriores donde se han presentado los algoritmos de inferencia, las dos aproximaciones diferentes ponen el esfuerzo computacional en diferentes áreas del espacio de búsqueda, mientras que los algoritmos de aproximación secuencial hacen una búsqueda sobre el conjunto de reglas y por tanto su orden de complejidad, está asociado con el número de variables del problema y el número de reglas, las versiones de la aproximación recursiva buscan en el vecindario de la regla central del ejemplo, y por lo tanto su orden de complejidad es exponencial en el número de variables continuas del problema. Por tanto, para estudiar la eficiencia de estas aproximaciones parece interesante hacer una revisión por casos, teniendo en cuenta el número de reglas y el número de variables continúas involucradas en el problema.

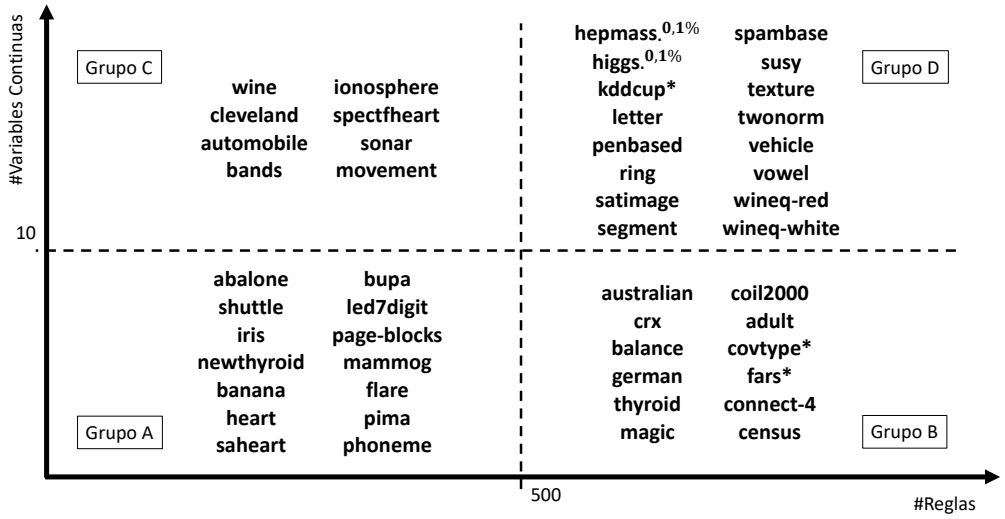

Fig. 4.3.: Los cuatro grupos (A,B,C y D) considerados en esta experimentación, teniendo en cuenta el número de reglas obtenidas por el algoritmo *WM-C* y el número de variables continuas de cada conjunto de datos.

La figura 4.3 muestra la división realizada para esta experimentación. Los 50 conjuntos de datos se han dividido en 4 grupos:

- Grupo A: Tiene 14 conjuntos de datos y esta compuesto por aquellos con un número menor o igual a 10 variables continuas y un número de reglas menor o igual a 500.

- Grupo B: Tiene 12 conjuntos de datos y esta compuesto por los que tienen un número menor o igual a 10 variables continuas y un número de reglas mayor a 500.

- Grupo C: Tiene 8 conjuntos de datos y está compuesto por los que tienen más de 10 variables continuas, pero menos o igual a 500 reglas.

- Grupo D: Tiene 16 conjuntos de datos y está compuesto por aquellos con más de 10 variables continuas y más de 500 reglas.

A continuación, se presenta un estudio específico para cada uno de estos grupos.

4.3.1. Estudio sobre los conjuntos de datos del Grupo A

En esta subsección se estudiarán los conjuntos de datos con un número reducido de reglas y un número reducido de variables continuas.

En la Tabla 4.2 se presentan los resultados obtenidos por los Algoritmos 2, 3 y 4 en los conjuntos de datos del grupo A, utilizando una validación cruzada con 10 particiones. Como se ha indicado anteriormente, el Algoritmo 1 se ha eliminado del estudio para facilitar la legibilidad de las tablas, ya que sus resultados en tiempo son siempre superiores a los del Algoritmo 2 y en precisión son exactamente los mismos.

Dado que el valor del parámetro $MaxRules$ se estableció en $2^{10} = 1024$ y el número de variables continuas es menor o igual a 10, se explora todo el vecindario en estos casos, lo que significa que los resultados de la inferencia secuencial (Algoritmo 2) coinciden con los de las versiones recursivas (Algoritmos 3 y 4) en términos de exactitud. Por esta razón, en la Tabla 4.2 solo se presenta una columna *Test* para la precisión, el *%NC* que representa al porcentaje de ejemplo no cubiertos por ninguna regla en el conjunto de test y *Alg2(μs)*, *Alg3(μs)* y *Alg4(μs)* es el tiempo medio consumido (en microsegundos) por cada algoritmo para realizar la inferencia de un ejemplo.

Conjunto de Datos	Test	%NC	Alg2(μs)	Alg3(μs)	Alg4(μs)
abalone	16,70	0,12	29,00	19,23	16,46
shuttle	92,42	0,01	23,74	13,86	10,74
iris	95,33	0,00	12,29	3,17	4,72
newthyroid	95,35	0,93	13,42	3,79	3,79
banana	79,38	0,02	6,97	1,34	1,70
heart	22,22	74,07	57,04	33,23	13,71
saheart	60,82	6,49	77,26	27,40	17,59
bupa	55,65	2,90	38,58	10,21	8,11
led7digit	63,60	15,20	22,86	2,60	4,42
page-blocks	93,70	0,20	42,58	6,83	8,09
mammog	76,87	7,47	26,91	2,44	3,85
flare	61,54	25,14	40,02	3,26	4,94
pima	66,02	3,26	82,90	13,33	10,10
phoneme	80,59	0,00	44,44	4,39	4,64
Promedio	68,58	9,7	37,00	10,36	8,06

Tab. 4.2.: Resultados obtenidos en el Grupo A (conjuntos de datos con bajo número de reglas y variables continuas). Las columnas muestran la siguiente información: (Test), es la precisión en el conjunto de test, (%NC), porcentaje de ejemplos del conjunto de test no disparados por ninguna regla y (Alg2(μs), Alg3(μs) y Alg4(μs)) tiempo medio consumido (en microsegundos) por cada algoritmo para realizar la inferencia de un ejemplo. La última fila muestra la media de cada una de las columnas.

La diferencia obvia entre estos algoritmos es el tiempo empleado en realizar la inferencia. Las tres últimas columnas muestran el tiempo medio que se tarda en realizar la inferencia de un ejemplo para cada uno de los algoritmos. Puede observarse que, en todos los casos, el Algoritmo 2 presenta un consumo de tiempo superior que, de media, puede estimarse es 4 veces más que en los otros algoritmos. Comparando las dos versiones recursivas, no hay una conclusión clara, aunque el Algoritmo 4 fue ligeramente mejor. En cualquier caso las mediciones son en microsegundos (utilizando las funciones de la librería "time.h"de C/C^{++}) y por tanto la diferencia entre ambos no es realmente significativa.

En un análisis algo más profundo se observa que la reducción de tiempo es mayor para conjuntos de datos con pocas variables continuas (como *flare* o *mammog* con 0 y 1 respectivamente), siendo del orden de 10 veces más rápido. Esto confirma el análisis anterior sobre el orden de complejidad de los algoritmos. Los Algoritmos 3 y 4 serán más eficientes y la ventaja de su uso se reflejará más con problemas con bajo número de variables continuas. A pesar de todo lo anterior y de la ventaja en eficiencia demostrada en la experimentación para los problemas del grupo A, no existe una diferencia real en su uso. Un tiempo medio de 37 microsegundos por inferencia es un tiempo muy razonable y para la mayoría de los problemas (excepto los problemas con un número elevado de ejemplos) el enfoque funcionará bien.

Para determinar si existen diferencias significativas entre ambos algoritmos en el tiempo de inferencia se aplica una prueba de rangos con signo de Wilcoxon [96] con un intervalo de confianza del 95 %. Los resultados de esta prueba, para los diferentes algoritmos, se muestran en la Tabla 4.3, que representa los resultados obtenidos en estas comparaciones, indicando las victorias (W), empates (T) y derrotas (L) entre los algoritmos junto con el p-valor calculado.

Se puede constatar que existen diferencias significativas entre el Algoritmo 2 y los Algoritmos 3 y 4, lo que sugiere que el Algoritmo 2 requiere más tiempo de ejecución en comparación. Por otro lado, al comparar el Algoritmo 3 y 4, se puede concluir que no existen diferencias significativas entre ambos en términos de tiempo requerido para la inferencia.

h	W/T/L	p-valor
Alg2 vs Alg3	14/0/0	0,0001220703
Alg2 vs Alg4	14/0/0	0,0001220703
Alg3 vs Alg4	6/1/7	0,2348113

Tab. 4.3.: Pruebas estadísticas que comparan los resultados obtenidos en el tiempo de inferencia del Grupo A entre los distintos algoritmos analizados.

4.3.2. Estudio sobre los conjuntos de datos del Grupo B

En esta sección, se llevará a cabo un estudio de conjuntos de datos con un alto número de reglas y un bajo número de variables continuas.

La Tabla 4.4 muestra los resultados obtenidos por los Algoritmos 2, 3 y 4 en el grupo B de conjunto de datos. Al igual que en el grupo anterior, la columna *Test* muestra la capacidad de predicción en el conjunto de test y la columna *%NC* muestra el porcentaje de ejemplos del conjunto de test que no se disparan por ninguna de las reglas aprendidas. La razón es exactamente la misma que la indicada anteriormente, ya que el número de variables continuas de los conjuntos de datos incluidos en este grupo vuelve a ser inferior o igual a 10.

Conjunto de Datos	Test	%NC	Alg2(μs)	Alg3(μs)	Alg4(μs)
australian	54,64	34,06	97,87	27,35	12,88
crx	40,74	52,37	103,55	25,26	14,81
balance	0,00	100,00	77,14	1,52	2,33
german	0,60	99,40	142,84	25,19	13,70
thyroid	91,99	1,36	209,86	13,20	14,62
magic	81,63	0,09	399,34	24,21	14,28
coil2000	24,27	73,66	2108,63	75,94	81,93
adult	63,32	21,37	4733,32	20,89	13,16
covtype*	85,69	0,01	8644,59	64,92	54,62
fars*	44,88	55,12	14934,89	33,82	35,16
connect-4	0,00	100,00	21070,95	14,50	27,31
census	48,09	50,08	30766,76	108,33	48,39
Promedio	44,65	48,96	6940,81	36,26	27,77

Tab. 4.4.: Resultados obtenidos en el Grupo B (conjuntos de datos con alto número de reglas y bajo número de variables continuas). Las columnas muestran la siguiente información: (Test), precisión en el conjunto de prueba, (%NC), porcentaje de ejemplos del conjunto de test no disparados por ninguna regla y (Alg2(μs), Alg3(μs) y Alg4(μs)) tiempo medio consumido (en microsegundos) por cada algoritmo para realizar la inferencia de un ejemplo. La última fila muestra la media de cada una de las columnas.

A priori este grupo representa la situación más favorable para el enfoque recursivo y la más desfavorable para el enfoque secuencial, ya que la eficacia del primero depende del número de variables continuas, y en este grupo es menor o igual a 10, y la complejidad del algoritmo del segundo enfoque depende del número de reglas, y en todos estos casos el número de reglas es superior a 500.

Los resultados confirman empíricamente lo anterior y el tiempo de ejecución obtenido por los Algoritmos 3 y 4 es significativamente mejor que el obtenido por el Algoritmo 2. Por término medio, los resultados del enfoque recursivo son del orden de unas 200 veces más rápidos que los de la versión secuencial. Esta diferencia

también está relacionada con el número de variables continuas. Por ejemplo, en el conjunto de datos *connect-4*, que no tiene variables continuas, la versión recursiva es del orden de más de 700 veces más rápida que el tiempo del Algoritmo 2. Para el conjunto de datos *australian*, se obtienen tiempos de inferencia similares, y a pesar de ello, pasa de 97 microsegundos a 27 microsegundos en el peor de los casos, es decir, las versiones recursivas son más de tres veces más rápidas que las del Algoritmo 2.

El porcentaje de ejemplos no disparados por ninguna regla, columna *%NC*, teóricamente perjudica al enfoque recursivo, ya que le obliga a explorar todo el vecindario para no encontrar ninguna regla que disparar. Pero cuando el conjunto de variables continuas es bajo, ese tiempo es asumible, como ocurre en este grupo.

Por otra parte, puede observarse como el número de reglas afecta negativamente a la eficacia del enfoque secuencial. El número de reglas oscila entre 500 en el conjunto de datos *australian* y 80968 en el conjunto de datos *census*. Los resultados demuestran que para este grupo, las versiones recursivas son significativamente mejores, produciendo reducciones de tiempo muy importantes en la inferencia. Por ejemplo, el tiempo consumido por la inferencia en el Algoritmo 2 para realizar la validación cruzada de 10 particiones en el conjunto de datos *census* (que tiene un total de 142521 ejemplos) fue de 73 minutos, mientras que el Algoritmo 4 necesitó menos de 7 segundos para realizar la misma tarea.

En relación al comportamiento de los algoritmos de la versión recursiva para los conjuntos de datos de este grupo podemos decir que el Algoritmo 4 consume menos tiempo en la mayoría de los casos (en 7 de los 12 conjuntos de datos) que el Algoritmo 3, pero no se encuentra diferencias estadísticamente significativas entre ellos.

h	W/T/L	p-valor
Alg2 vs Alg3	12/0/0	0,0004882813
Alg2 vs Alg4	12/0/0	0,0004882813
Alg3 vs Alg4	7/0/5	0,1513672

Tab. 4.5.: Pruebas estadísticas que comparan los resultados obtenidos en el tiempo de inferencia del grupo B entre los distintos algoritmos analizados.

Se puede constatar en la Tabla 4.5 aplicando el test de Wilcoxon, que existen diferencias significativas entre el Algoritmo 2 y los Algoritmos 3 y 4, lo que sugiere que el Algoritmo 2 requiere más tiempo de ejecución en comparación. Por otro lado, al comparar el Algoritmo 3 y 4, se puede concluir que no existen diferencias significativas entre ambos en términos de tiempo requerido para la inferencia.

4.3.3. Estudio sobre los conjuntos de datos del Grupo C.

En esta subsección, se estudian conjuntos de datos con un número reducido de reglas (menos de 500) y un número elevado de variables continuas (más de 10). Desde el punto de vista de los órdenes de complejidad, este grupo representa el caso más favorable para el enfoque secuencial. En este estudio, se mostrará si los resultados empíricos respaldan este comportamiento teórico.

	Alg2		Alg3		Alg4	
Conjunto de Datos	Test	Tiempo(μs)	Test	Tiempo(μs)	Test	Tiempo(μs)
wine	75,28	48,38	46,07	471,56	58,43	232,68
cleveland	18,86	61,86	18,52	46,92	18,86	24,13
automobile	42,14	30,42	27,67	515,86	40,25	294,08
bands	33,42	71,20	8,49	756,22	16,44	350,86
ionosphere	54,99	91,12	20,23	594,07	34,19	318,15
spectfheart	23,97	120,77	0,00	690,15	0,00	538,99
sonar	16,83	109,09	0,00	781,63	0,00	639,81
movement	63,33	171,56	18,33	892,87	22,78	769,66
Promedio	41,10	88,05	17,41	593,66	23,87	396,05

Tab. 4.6.: Resultados obtenidos en el Grupo C - primera parte (conjuntos de datos con bajo número de reglas y alto número de variables continuas). Las dos primeras columnas están asociadas al Algoritmo 2, las dos siguientes al Algoritmo 3 y las dos últimas al Algoritmo 4. La última fila muestra la media de cada una de las columnas. Para cada algoritmo, se muestran la precisión en el conjunto de prueba (Test) y el tiempo consumido por la inferencia de un ejemplo en microsegundos (Tiempo(μs)).

	Alg2		Alg3		Alg4	
Conjunto de Datos	Test*	%NC	Test*	%NC	Test*	%NC
wine	96,40	21,91	97,62	52,81	98,11	40,45
cleveland	65,88	71,38	66,27	72,05	65,88	71,38
automobile	88,16	52,20	100,00	72,33	95,52	57,86
bands	66,67	49,86	72,09	88,22	68,97	76,16
ionosphere	97,97	43,87	98,61	79,49	98,36	65,24
spectfheart	48,12	50,19	0,00	100,00	0,00	100,00
sonar	97,22	82,69	0,00	100,00	0,00	100,00
movement	93,06	31,94	100,00	81,67	100,00	77,22
Promedio	81,69	50,51	66,82	80,82	65,86	73,54

Tab. 4.7.: Resultados obtenidos en el Grupo C - segunda parte (conjuntos de datos con bajo número de reglas y alto número de variables continuas). Las dos primeras columnas están asociadas al Algoritmo 2, las dos siguientes al Algoritmo 3 y las dos últimas al Algoritmo 4. Para cada algoritmo, se muestra la precisión en el conjunto de prueba considerando sólo los ejemplos que activan alguna regla (Test*) y el porcentaje de ejemplos que no activan ninguna regla (%NC).

En este grupo de datos con un número elevado de variables continuas, el parámetro $MaxRules$ de la función BackTrack se toma como $1024 = 2^{10}$, igual que en los grupos anteriores. Sin embargo, a diferencia de los grupos anteriores, este grupo

presenta un número de variables continuas mayor a 10, lo que implica que, al tomar este valor como límite, es posible que no se explore completamente el vecindario de la regla central del ejemplo y, por lo tanto, los resultados obtenidos por el enfoque secuencial no necesariamente coinciden con los del enfoque recursivo, ya que no se exploran todas las reglas posibles.

Además, dado que los Algoritmos 3 y 4 utilizan diferentes heurísticas para explorar el vecindario, los resultados obtenidos entre ellos también son diferentes. Para obtener una mayor claridad en los resultados y presentar tablas más legibles, se ha dividido la información relevante sobre el comportamiento de estos algoritmos en dos tablas. La Tabla 4.6 (C-1) muestra los resultados de precisión en el conjunto de prueba (*Test*) y el tiempo consumido por la inferencia en cada uno de los algoritmos, mientras que la Tabla 4.7 (C-2) informa sobre la precisión en el conjunto de prueba solo para aquellos ejemplos que activan alguna regla (*Test**) y el porcentaje de ejemplos que no son activados por ninguna regla (*%NC*).

Los resultados la de Tabla 4.6 muestran que los algoritmos del enfoque recursivo no son muy apropiados para este tipo de conjunto de datos, ya que presenta un tiempo de inferencia elevado y una precisión peor que el enfoque secuencial. Bajo estas características, el modelo secuencial es preferible. Por término medio, el Algoritmo 2 ofrece más del doble de precisión y es 5 veces más rápido que los otros dos algoritmos.

Un análisis más detallado revela que la diferencia de precisión entre los dos enfoques se debe al elevado número de ejemplos que no activan ninguna regla cuando se utiliza la versión recursiva. La columna *%NC* asociada al Algoritmo 2 de la Tabla 4.7 muestra el porcentaje de ejemplos que no son activados por ninguna regla. Comparando esta columna con la misma información, pero de los Algoritmos 3 y 4, se observa que salvo el conjunto de datos de *cleveland* donde los porcentajes son similares, el incremento de este parámetro es siempre superior a 17 puntos. Hay dos situaciones extremas a destacar, los conjuntos de datos *specfheart* y sonar, donde el enfoque recursivo no consigue emparejar ningún ejemplo con ninguna regla.

De la Tabla 4.7 podemos extraer las siguientes conclusiones:

- La diferencia en el parámetro *%NC* entre los Algoritmos 3 y 4 y el Algoritmo 2 muestra la proporción de ejemplos que son activados por reglas que no se encuentran entre las 1024 mejor ajustadas. Además, los Algoritmos 3 y 4 utilizan diferentes heurísticas para estimar el orden en la evaluación y que es diferente en ambos.

- Del hecho anterior se deduce que los grados de adaptación con los que los ejemplos activan las reglas toman valores bajos.
- Cuando los ejemplos del conjunto de prueba se distribuyen correctamente entre los clústeres que fueron reconocidos por el algoritmo de aprendizaje sobre los ejemplos de entrenamiento, el enfoque recursivo funciona bien, incluso mejor que el enfoque secuencial, dando valores similares en el conjunto de datos *cleveland*.
- Al comparar los resultados obtenidos por los Algoritmos 3 y 4, se observa que, en todos los conjuntos de datos, este último obtiene mejores resultados en los dos parámetros estudiados. Esto indica que la heurística utilizada en este algoritmo mejora experimentalmente a la propuesta del Algoritmo 3. De media la mejora un 6 % en precisión y unos 200 microsegundos menos por inferencia.

La conclusión para los conjuntos de datos del grupo C es que el enfoque secuencial del Algoritmo 2 es la mejor opción, ya que no se puede asegurar nada sobre el comportamiento de los ejemplos de prueba.

	Test		Tiempo	
h	W/T/L	p-valor	W/T/L	p-valor
Alg2 vs Alg3	8/0/0	0,0078125	0/1/7	0,015625
Alg2 vs Alg4	7/1/0	0,02249427	1/0/7	0,015625
Alg3 vs Alg4	0/2/6	0,03603169	8/0/0	0,0078125

Tab. 4.8.: Pruebas estadísticas que comparan los resultados obtenidos en la precisión (Test) y el tiempo de inferencia (Tiempo) del grupo C-1 entre los distintos algoritmos analizados.

Aplicando el test de Wilcoxon, se observa en la Tabla 4.8 tomando como referencia la primera parte, que existen diferencias significativas entre el Algoritmo 2 y los Algoritmos 3 y 4 en la precisión, lo que sugiere que el Algoritmo 2 tiene un mayor porcentaje de precisión. En tiempo de inferencia, el Algoritmo 2 requiere menos tiempo de ejecución. Por otro lado, al comparar el Algoritmo 3 y 4 en precisión, se puede concluir que existen diferencias significativas entre ambos en términos a favor del Algoritmo 4 y en tiempo de inferencia, el Algoritmo 3 requiere un mayor tiempo de ejecución.

En la Tabla 4.9 y tomando como referencia la segunda parte, se observa que no existen diferencias significativas entre el Algoritmo 2 y los Algoritmos 3 y 4 en la precisión. Por otro lado, al comparar el Algoritmo 3 y 4 en Test*, se puede concluir que no existen diferencias significativas entre ambos en términos de precisión y en *%NC*, el Algoritmo 3 es inferior.

	Test*		%NC	
h	W/T/L	p-valor	W/T/L	p-valor
Alg2 vs Alg3	2/0/6	0,7421875	0/0/8	0,0078125
Alg2 vs Alg4	2/1/5	0,9326466	0/1/7	0,02249427
Alg3 vs Alg4	4/3/1	0,2807127	0/2/6	0,03603169

Tab. 4.9.: Pruebas estadísticas que comparan los resultados obtenidos en la precisión (Test) y el porcentaje de ejemplos no cubiertos (%NC) del grupo C-2 entre los distintos algoritmos analizados.

4.3.4. Estudio sobre los conjuntos de datos del Grupo D

Finalmente, en esta subsección se estudian los conjuntos de datos con un elevado número de reglas (más de 500) y un elevado número de variables continuas (más de 10). En los tres grupos anteriores se ha concluido que los resultados empíricos han demostrado lo que se podía intuir a partir del orden de complejidad de cada algoritmo. Por un lado, el grupo B con muchas reglas favorecería el enfoque recursivo, el grupo C con muchas variables continuas hacía que el enfoque secuencial fuera el recomendado. En el caso del grupo A, donde las variables que incrementan el tiempo estaban en valores bajos para ambos enfoques, la experimentación demostró que el enfoque recursivo ofrecía mejores resultados. En este último grupo encontramos conjuntos de datos con un alto número de reglas y un alto número de variables continuas, es decir, este grupo contiene conjuntos de datos donde las variables que afectan al alto consumo de tiempo se encuentran con valores altos para ambos enfoques y, por lo tanto, es más difícil estimar cuál puede ofrecer mejores resultados.

En este grupo se han evaluado 16 conjuntos de datos, y al igual que en el grupo C se han dividido los resultados en dos tablas. La Tabla 4.10 (D-1) recoge los resultados sobre la precisión y el tiempo consumido en la inferencia de un ejemplo para cada uno de los algoritmos.

Al igual que en la experimentación llevada a cabo en todos los grupos anteriores, para este grupo se utilizará 1024 como valor del parámetro $MaxRules$ del procedimiento BackTrack y por la misma razón ya comentada en el grupo C, la salida de los algoritmos en el enfoque recursivo no tiene por qué coincidir con la salida devuelta por el enfoque secuencial, ni tampoco tiene porque coincidir la salida de los algoritmos del enfoque recursivo. Además, dentro de este grupo hay dos conjuntos de datos que son particularmente complejos y requieren una inferencia muy eficiente, ya que tienen más de 10 millones de ejemplos, generan más de 8 millones de reglas y tienen 28 variables, todas continuas. Estos conjuntos de datos son *higgs* y *hepmass*. Debido a los altos valores de estos parámetros, la inferencia

	Alg2		Alg3		Alg4	
Conjunto de Datos	Test	Tiempo(μs)	Test	Tiempo(μs)	Test	Tiempo(μs)
hepmass$^{0,1\,\%}$	75,77	3907504,76	13,83	1539,08	29,71	1072,97
higgs$^{0,1\,\%}$	58,54	3860736,36	27,58	996,74	49,40	503,74
kddcup*	99,95	650,58	99,95	22,91	99,95	31,10
letter	77,74	2195,16	76,13	150,19	77,43	72,81
penbased	95,58	1316,72	91,11	186,34	94,01	67,62
ring	68,92	1804,92	49,34	396,78	54,64	240,84
satimage	65,25	2025,13	56,25	439,11	64,83	313,37
segment	91,73	190,59	87,36	137,19	90,52	70,71
spambase	81,31	872,66	77,25	215,23	80,64	98,74
susy	73,81	104224,60	73,68	139,22	73,80	79,05
texture	92,85	2166,64	51,84	658,25	87,44	382,74
twonorm	89,15	2088,24	19,84	634,60	34,14	385,67
vehicle	60,28	223,55	42,44	448,42	52,01	208,21
vowel	93,64	127,75	93,64	53,22	93,43	22,06
wineq-red	56,85	205,61	56,79	76,43	56,79	30,72
wineq-white	49,92	352,38	49,84	58,75	49,92	24,60
Promedio	76,96	492917,85	60,43	384,53	68,04	225,31

Tab. 4.10.: Resultados obtenidos en el Grupo D - primera parte (conjuntos de datos con un elevado número de reglas y un elevado número de variables continuas). Las dos primeras columnas están asociadas al Algoritmo 2, las dos siguientes al Algoritmo 3 y las dos últimas al Algoritmo 4. La última fila muestra la media de cada una de las columnas. Para cada algoritmo, se muestran la precisión en el conjunto de prueba (Test) y el tiempo consumido por la inferencia en microsegundos de un ejemplo (Tiempo(μs)). El valor $^{0,1\,\%}$ en el nombre de los conjuntos de datos *higgs* y *hepmass* indica que los resultados se han obtenido utilizando validaciones cruzada de 10 particiones, pero utilizando sólo el 0,1 % de los ejemplos del conjunto de prueba en cada una de las diez validaciones.

lleva mucho tiempo. Por esta razón, tanto en este estudio como en los siguientes se hace un tratamiento especial de estos dos conjuntos de datos. Se puede observar que en la Tabla 4.10, ambos conjuntos de datos tienen sobre su nombre 0,1 %. Este valor se refiere a que se ha realizado la validación cruzada de 10 particiones como en todos los conjuntos de datos de este estudio, pero en este caso, en la inferencia, en lugar de tomar el 100 por cien de los ejemplos, solo se ha tomado el 1 por mil (0,1 por ciento) de los ejemplos en cada una de las validaciones.

Bajo estas condiciones, los tres algoritmos analizados realizan la inferencia utilizando los mismos modelos y se llevan a cabo experimentos con exactamente los mismos ejemplos, a pesar de que esta experimentación solo se realice sobre el 0,1 % del total de ejemplos que deberían ser probados. Desde el punto de vista del tiempo consumido por la inferencia, solo afecta el hecho de que esta media se haya realizado con un menor número de ejemplos, pero la media que se obtendría y aplicaría sobre todos los ejemplos debería ser muy similar. Si se observa el valor obtenido por el Algoritmo 2 sobre el conjunto de datos *hepmass*, se puede notar que cada

	Alg2		Alg3		Alg4	
Conjunto de Datos	Test*	%NC	Test*	%NC	Test*	%NC
hepmass$^{0,1\,\%}$	77,65	2,42	79,56	82,62	78,35	62,08
higgs$^{0,1\,\%}$	58,59	0,08	60,82	54,66	58,47	15,50
kddcup*	99,95	0,00	99,95	0,01	99,95	0,00
letter	77,75	0,02	77,18	1,36	77,64	0,27
penbased	98,85	3,31	98,93	7,91	98,92	4,96
ring	72,18	4,51	85,26	42,14	79,07	30,91
satimage	65,25	0,00	64,16	12,32	65,23	0,61
segment	93,22	1,60	93,21	6,28	93,52	3,20
spambase	84,70	4,00	84,51	8,59	84,63	4,72
susy	73,81	0,00	73,77	0,11	73,81	0,01
texture	92,92	0,07	91,61	43,42	93,11	6,09
twonorm	91,88	2,97	91,29	78,27	90,99	62,49
vehicle	61,74	2,36	66,48	36,17	63,86	18,56
vowel	93,73	0,10	93,73	0,10	93,72	0,30
wineq-red	57,53	1,19	57,58	1,38	57,61	1,44
wineq-white	50,01	0,18	49,96	0,24	50,02	0,20
Promedio	78,11	1,43	79,25	23,47	78,68	13,21

Tab. 4.11.: Resultados obtenidos en el Grupo D - segunda parte (conjuntos de datos con un elevado número de reglas y un elevado número de variables continuas). Las dos primeras columnas están asociadas al Algoritmo 2, las dos siguientes al Algoritmo 3 y las dos últimas al Algoritmo 4. La última fila muestra la media de cada una de las columnas. Para cada algoritmo, la precisión en el conjunto de test considerando sólo los ejemplos que activan alguna regla (Test*) y el porcentaje de ejemplos que no activan ninguna regla (%NC). El valor $^{0,1\,\%}$ sobre el nombre de *higgs* y *hepmass* indica que los resultados se han obtenido utilizando diez validaciones cruzadas pero utilizando sólo el 0,1 % del ejemplo de la parte de prueba en cada una de las diez validaciones.

inferencia consume casi 4 segundos en el ordenador en el que se ha realizado la experimentación, (3907505 μ segundos exactamente), este conjunto de datos tiene 10,5 millones de ejemplos. En una validación cruzada completa, es necesario realizar inferencia sobre todos los ejemplos, es decir, es necesario realizar 10,5 millones de inferencias consumiendo una media de 4 segundos solo en la inferencia. Esto significa que se necesitan 42 millones de segundos (1,3 años) para realizar el experimento. Aplicando la reducción del 0,1 % se ha logrado reducir de 1,3 años a unos 4,86 días. Del mismo modo, el tiempo necesario para *higgs* también fue de unos 5 días frente a los 1,4 años que se habrían necesitado al tomar todos los ejemplos para la prueba.

En cuanto al análisis de tiempo, se observa que en el enfoque recursivo se obtienen los tiempos de ejecución más bajos para todos los conjuntos de datos, con la única excepción de *vehicle* cuando se utiliza el Algoritmo 3. Por otro lado, el Algoritmo 4 obtiene los tiempos más bajos excepto para el conjunto de datos *kddcup**.

En este estudio comparativo, el Algoritmo 4 es aproximadamente 1/3 más rápido que el Algoritmo 3. Comparando el Algoritmo 2 y el Algoritmo 4, se observa que, con conjunto de datos con un número elevado de reglas, las diferencias de tiempo de ambos enfoques son más pronunciadas. Si se toman todos los conjuntos de datos, excepto los que tienen más reglas, *susy*, *higgs* y *hepmass*, el Algoritmo 4 es de media 7 veces más rápido que el algoritmo 2. Si solo se consideran los 3 últimos conjuntos de datos, el Algoritmo 4 es, por término medio, 5000 veces más rápido que el Algoritmo 2.

Aunque los algoritmos del enfoque recursivo son más rápidos en tiempo, en promedio se produce una reducción significativa de la precisión global, del 79,96 % del Algoritmo 2 al 68,94 % de Algoritmo 4. Esta perdida media del 8 % no es uniforme en todos los conjuntos de datos. Se puede indicar que hay 10 conjuntos de datos en los que la diferencia de precisión es inferior al 2 % (*vowel, segment, wineq-red, whineq-white, kddcup*, spambase, penbased, satimage, letter, susy*), y sin embargo en *twonorm* y *hepmass* esa diferencia es superior al 45 %.

En la Tabla 4.11 (D-2) se observa un estudio algo más detallado de donde se proceden están discrepancias en la precisión y se pueden formular las siguientes conclusiones:

- La diferencia de precisión de los 10 conjuntos de datos mencionados anteriormente se debe a la existencia de ejemplos no cubiertos por las reglas.
- La discrepancia entre *hepmass* y *twonorm* también se debe a los ejemplos sin cubrir. En este caso, hay una diferencia muy grande de ejemplos sin cubrir.
- El porcentaje de precisión en los ejemplos cubiertos es muy similar. Solo es destacable la diferencia a favor del Algoritmo 4 en el conjunto de datos ring. la razón debe ser que hay más reglas alejadas de la máxima adaptación en el ejemplo con un peso mayor que las reglas más cercanas. El enfoque secuencial se aleja más, mientras que en la versión recursiva, la regla activada por el otro enfoque queda fuera de su espacio de búsqueda. Una diferencia grande pero menor se da en *vehicle* (2,12 %), para el resto las diferencia son menores al 1 %.
- Parece que el uso de reglas más cercanas al óptimo pueden dar una ventaja en la inferencia, ya que aunque las diferencias suelen ser pequeñas (menos del 1 %), tienden a favorecer el enfoque recursivo.
- El Algoritmo 4 tiene una mayor capacidad de predicción en los ejemplos que activan alguna regla comparándolo con el Algoritmo 3. En los casos en lo que

esto no ocurre (*vehicle, ring, higgs, hepmass*), es porque hay una diferencia significativa en el porcentaje de ejemplos que no activan ninguna regla en los dos algoritmos. Por ejemplo, en *higgs*, el Algoritmo 3 tiene un porcentaje de aciertos 2 puntos mayor que el Algoritmo 4, pero hay una diferencia de 40 puntos de menos ejemplos cubiertos por el algoritmo 3. Estas elevadas diferencias en el porcentaje de ejemplos no cubiertos entre los dos algoritmos hacen que la heurística implementada en el Algoritmo 4 produzca mejores resultados en tiempo, cobertura de reglas y precisión en comparación con el Algoritmo 3.

	Test		Tiempo	
h	W/T/L	p-valor	W/T/L	p-valor
Alg2 vs Alg3	14/2/0	0,001097051	15/0/1	0,000213623
Alg2 vs Alg4	14/2/0	0,001097051	16/0/0	0,00003051758
Alg3 vs Alg4	1/2/13	0,002097643	15/0/1	0,00006103516

Tab. 4.12.: Pruebas estadísticas que comparan los resultados obtenidos en la precisión (Test) y en el tiempo de inferencia (Tiempo) del grupo D-1 entre los distintos algoritmos analizados.

Se observa que en la Tabla 4.12 aplicando el test de Wilcoxon, y tomando como referencia la primera parte, que existen diferencias significativas entre el Algoritmo 2 y los Algoritmos 3 y 4 en la precisión, lo que sugiere que el Algoritmo 2 tiene un mayor porcentaje de precisión, y en tiempo de inferencia, el Algoritmo 2 requiere un mayor tiempo de ejecución. Por otro lado, al comparar el Algoritmo 3 y 4 en precisión, se puede concluir que existen diferencias significativas entre ambos en términos de precisión a favor del algoritmo 4 y en tiempo de inferencia el Algoritmo 3 requiere un mayor tiempo de ejecución.

	Test*		%NC	
h	W/T/L	p-valor	W/T/L	p-valor
Alg2 vs Alg3	8/2/6	0,729827	0/1/15	0,0007265139
Alg2 vs Alg4	6/2/8	0,3149359	0/1/15	0,0007247147
Alg3 vs Alg4	7/1/8	0,8870649	14/0/2	0,0007629395

Tab. 4.13.: Pruebas estadísticas que comparan los resultados obtenidos en la precisión (Test) y en el porcentaje de ejemplos no cubiertos (%NC) del grupo D-2 entre los distintos algoritmos analizados.

En la Tabla 4.13 y tomando como referencia la segunda parte, se observa que no existen diferencias significativas entre el Algoritmo 2 y los Algoritmos 3 y 4 en la precisión y el *%NC* del Algoritmo 2 es inferior. Por otro lado, al comparar el Algoritmo 3 y 4 en Test*, se puede concluir que no existen diferencias significativas entre ambos en términos de precisión, pero en %NC si las hay y el Algoritmo 4 es superior.

Del análisis anterior se pueden concluir dos cosas:

- El Algoritmo 4 presenta una capacidad de predicción similar a la ofrecida por el enfoque secuencial con una reducción muy significativa del tiempo cuando el primero dispara una regla, pero.
- el alto porcentaje de ejemplos que no se activan con el enfoque recursivo frente al secuencial, hace que este enfoque no sea utilizable para todos los problemas.

Con estas dos conclusiones en mente, se analiza ahora el comportamiento del Algoritmo 5 que es una versión modificada del Algoritmo 4 en la que se añade una nueva heurística consistente en utilizar una distancia entre reglas para realizar la exploración de la vecindad de la regla central.

4.3.5. Estudio del Algoritmo 5

En el análisis previo sobre el conjunto de datos que contienen un alto número de reglas y variables continuas, se ha observado que el enfoque recursivo tiene un comportamiento que produce una reducción de tiempo muy significativa, pero para algunos conjuntos de datos, el alto número de ejemplos que no son disparados por ninguna regla produce un resultado indeseable. A menudo, el problema se debe a que las reglas disparadas tienen valores muy bajos en el grado de adaptación entre el ejemplo y el antecedente.

Para ver en qué medida estos grados de adaptación teórica afectan a los resultados obtenidos en el enfoque recursivo, se analizará ahora el Algoritmo 5. Este algoritmo es una versión del Algoritmo 4 con una restricción adicional de limitar el vecindario donde buscar la regla. Esta restricción viene definida por el parámetro d que toma valores enteros y que se definió en la descripción del Algoritmo 5.

La Tabla 4.14 muestra los resultados de la precisión, tanto en el número total de ejemplos (*Test*), como sobre los ejemplos activados por la regla (*Test**), para 5 valores del parámetro de distancia d.

En la columna 0 de *Test*, el Algoritmo 5 utiliza una distancia $d = 0$ por lo que comprueba si la regla central al ejemplo pertenece al conjunto de reglas de inferencia y en caso afirmativo la utiliza, en caso contrario no devuelve ninguna salida. Así, los porcentajes cercanos al 100 % indicarían que los datos utilizados para las pruebas se mueven en las mismas cuadriculas que los datos en los que se aprendieron las reglas, y además, esas cuadriculas contienen ejemplos que son en su mayoría de la

	Test					Test*				
Conjunto de Datos	0	1	2	3	5	0	1	2	3	5
hepmass$^{0,1\,\%}$	0,76	6,26	20,67	38,78	63,73	81,09	80,06	79,94	79,62	78,33
higgs$^{0,1\,\%}$	13,25	32,45	46,46	53,52	57,74	64,63	59,27	58,4	58,4	58,63
kddcup*	99,92	99,95	99,95	99,95	99,95	99,95	99,95	99,95	99,95	99,95
letter	57,06	74,44	77,21	77,54	77,62	83,01	77,96	77,71	77,64	77,64
penbased	33,77	67,41	84,53	91,31	95,16	99,41	99,22	99	98,94	98,88
ring	29,03	44,88	50,73	55,00	62,97	96,67	90,96	84,32	78,1	72,54
satimage	46,95	59,47	63,68	64,88	65,27	66,37	65,27	65,28	65,27	65,29
segment	69,57	85,71	90,09	90,91	91,52	94,36	93,97	93,87	93,54	93,42
spambase	62,69	75,79	79,68	80,81	81,23	81,23	84,28	84,69	84,74	84,71
susy	66,71	73,43	73,80	73,80	73,81	68,71	73,65	73,83	73,81	73,81
texture	33,64	66,18	82,56	88,80	91,85	92,55	92,86	92,71	92,92	92,85
twonorm	0,14	3,34	15,09	36,97	75,15	100	89,17	91,63	91,63	91,66
vehicle	9,46	28,13	45,86	55,44	59,57	78,43	69,79	66,9	65,14	61,99
vowel	44,14	81,01	91,62	93,43	93,64	95,62	95,02	94,48	93,81	93,73
wineq-red	43,65	52,78	55,66	56,54	56,85	68,1	58,86	57,53	57,62	57,57
wineq-white	40,83	48,55	50,04	49,92	49,92	57,14	49,55	50,26	50,03	50,01
Promedio	40,72	56,24	64,23	69,23	74,75	82,95	79,99	79,41	78,82	78,19

Tab. 4.14.: Resultados obtenidos en precisión sobre el total de ejemplos de Prueba (Test) y precisión considerando sólo los ejemplos del total que son disparados por alguna regla (Test*) para el conjunto de valores de distancia 0,1,2,3 y 5.

clase de la que fueron aprendidas, este es el caso del conjunto de datos *kddcup**. Por otro lado, los conjuntos de datos con valores cercanos al 0 % indicarían lo contrario, es decir, o bien los datos de prueba no están en las cuadriculas que se crearon en el entrenamiento o si están en esas cuadriculas, no hubo una buena distribución de clases o la clase de los ejemplos de prueba no coincide con los del entrenamiento.

Para tratar de entender lo que ocurre en realidad, se puede observar en la columna 0 de *Test**, que describe la precisión de la base de reglas cuando la inferencia activó al menos una regla. Se puede observar que en este caso los valores de precisión son altos. Por lo tanto, la razón fundamental de los valores bajos en la columna 0 de *Test* es porque no se activa ninguna regla, ya que los resultados cuando se activan las reglas son bastante aceptables, por ejemplo, si se observa en *hepmass*, se sitúa en el 0.76 % en todos los ejemplos, pero tiene una probabilidad de acierto del 81,09 % cuando se activa alguna regla. Análogamente, *twonorm* pasa del 0,14 % al 100 % cuando se activa una regla.

Además, si se observa la tendencia de los valores de *Test** a medida que aumenta el valor de d, se puede notar que los valores tienden a mantenerse, aunque disminuyen ligeramente. El comportamiento promedio muestra una caída suave en la precisión. Sin embargo, cuando el valor de d aumenta, los valores de *Test* también aumentan considerablemente. En conclusión, al aumentar el valor de d, se permite la búsqueda de reglas que están más alejadas de la regla central del ejemplo, lo que resulta en

un aumento en el porcentaje de precisión global y una ligera disminución en el porcentaje de precisión cuando el sistema empareja una regla con el ejemplo.

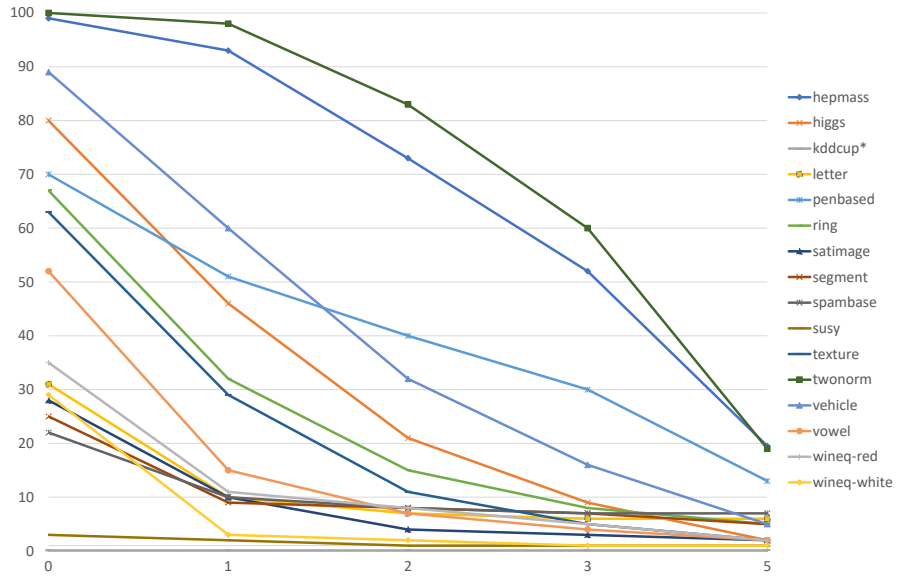

Fig. 4.4.: Porcentaje de ejemplos no activados por ninguna regla para el conjunto de distancias 0, 1, 2, 3 y 5.

En la Figura 4.4, se puede observar el comportamiento de la inferencia en relación con el porcentaje de ejemplos que no son activados por ninguna regla, a medida que aumenta el valor de d. Se puede ver que hay una disminución significativa en todos los conjuntos de datos. Este hecho, junto con los dos anteriores, indica que al aumentar el valor de d, se aumenta la precisión en todos los ejemplos, disminuyendo el porcentaje de ejemplos que no son activados por ninguna regla y manteniendo (aunque ligeramente disminuyendo) la precisión de las reglas cuando el sistema utiliza una de ellas.

Como conclusión a lo anterior, se podría considerar aumentar el valor de d hasta encontrar un punto de inflexión donde la disminución de *Test** sea significativa o el porcentaje de ejemplos no disparados por ninguna regla se reduzca al máximo. Sin embargo, no es posible hacerlo, ya que aumentar d implica un aumento exponencial en el espacio de búsqueda de las reglas, lo que implica un consumo de tiempo excesivo. Por lo tanto, se debe encontrar un equilibrio entre el valor de d y el tiempo de procesamiento disponible para obtener un rendimiento óptimo en el sistema.

La Tabla 4.15 informa del tiempo medio (expresado en microsegundos) consumido al realizar la inferencia de un ejemplo para cada uno de los conjuntos de datos estudiados y para los distintos valores de d. Se observa que para valores de d <3

	Tiempo				
Conjunto de Datos	0	1	2	3	5
hepmass$^{0,1\,\%}$	0,22	1,05	8,42	3355,04	26151,90
higgs$^{0,1\,\%}$	0,23	0,97	5,38	1414,23	4947,68
kddcup*	0,34	0,27	0,32	0,29	29,42
letter	0,16	0,34	0,63	0,78	93,12
penbased	0,16	0,25	0,42	0,65	95,10
ring	0,19	0,57	2,01	6,56	3316,88
satimage	0,35	1,91	14,05	79,32	96419,89
segment	0,22	0,38	0,65	0,95	225,42
spambase	0,53	0,69	0,96	1,49	651,02
susy	0,15	0,33	0,65	0,81	87,59
texture	0,40	2,44	24,34	168,44	238850,91
twonorm	0,19	0,71	3,03	13,34	9585,85
vehicle	0,22	0,46	1,46	3,81	1175,22
vowel	0,15	0,17	0,25	0,24	38,61
wineq-red	0,12	0,20	0,30	0,35	43,28
wineq-white	0,14	0,21	0,28	0,32	41,17
Promedio	0,23	0,68	3,95	315,41	23859,57

Tab. 4.15.: Tiempo consumido en microsegundos (μs) por la ejecución de la inferencia de un ejemplo para el conjunto de valores de distancia 0,1,2,3 y 5.

el consumo es muy pequeño y por tanto se producen inferencias muy rápidas, con el valor de 3 las inferencias empiezan a consumir mucho tiempo, y a partir de ese valor, el tiempo se dispara exponencialmente.

A partir de estos resultados, el uso de este algoritmo con el valor bajo de d produce muy buenos resultados de predicción, en muchos casos requiriendo una cantidad de tiempo muy pequeña. Sin embargo, el hecho de que el porcentaje de ejemplos no activados por ninguna regla sea elevado, lo hace inviable como método único para realizar inferencias. Pero si parece que podría utilizarse como método ágil con un alto grado de precisión en una primera etapa de un proceso de inferencia combinado con otras técnicas.

4.3.6. Enfoque híbrido de inferencia para problemas con muchas variables continuas y muchas reglas

En la sección previa se presentaron los resultados alcanzados por el Algoritmo 5 con el propósito de analizar el comportamiento del Algoritmo 4 al reducir el vecindario de búsqueda de las reglas. Los resultados demostraron que el algoritmo propuesto no es una solución universal para todos los problemas definidos en el grupo D en términos generales, pero podría ser un componente de un algoritmo de

inferencia más complejo, ya que permite descubrir reglas con una buena precisión de clasificación de forma eficiente.

El principal inconveniente del algoritmo anterior es que el porcentaje de ejemplos que no son activados por ninguna regla tiende a ser muy elevado debido a la restricción del espacio de búsqueda de la regla. En este caso, se ha demostrado que el Algoritmo 4 ofrece un alto nivel de precisión en los ejemplos que activan alguna regla, y a la vez ofrece un tiempo reducido para encontrar dicha regla.

De la misma forma que el Algoritmo 5, cuando se selecciona un valor de d bajo, el Algoritmo 4, tiende a tener un alto porcentaje de ejemplos no disparados por ninguna regla. Esto ocurre en muchas ocasiones cuando el número de variables continuas es elevado en relación con el valor dado al parámetro $MaxRules$. Este porcentaje es alto, pero inferior al que ofrece el Algoritmo 4 y por lo tanto se lo utilizará cuando el Algoritmo 5 no dispara ninguna regla. Por otro lado, para aquellos ejemplos que no sean activados ni por el Algoritmo 4 ni 5, se utilizará la inferencia de aproximación secuencial (Algoritmo 2)

Una descripción algorítmica de este proceso se muestra en el Algoritmo 6

Algoritmo 6 Algoritmo de inferencia híbrido que combina los enfoques de inferencia secuencia y retroactiva

```
1: function HYBRID_INFERENCE(e, R^I, d, MaxRules)
2:     class = Heuristic_Nearby_Neighborhood_Inference(e, R^I, d)
3:     if (class != ∅) then
4:         return class
5:     end if
6:     class = Heuristic_Neighborhood_Inference(e, R^I, MaxRules)
7:     if (class != ∅) then
8:         return class
9:     end if
10:    class = Standard_Inference_Prunned(e, R^I)
11:    return class
12: end function
```

En el estudio experimental realizado sobre los conjuntos de datos del grupo D, se mantuvieron las mismas condiciones que en estudios anteriores, con el valor de d fijado en 2 para el Algoritmo 5 y un límite máximo de reglas establecido en 1024 para el Algoritmo 4.

Los resultados obtenidos se exponen en la Tabla 4.16. Comparándolos con los obtenidos con la inferencia estándar (ver resultados obtenidos por le Algoritmo 2 en las Tablas 4.10 y 4.11) se puede destacar que:

Conjunto de Datos	Test	Test*	%NC	Tiempo(μs)
hepmass$^{0,1\,\%}$	75,89	77,77	2,42	2151904,76
higgs$^{0,1\,\%}$	58,60	58,65	0,08	404635,45
kddcup*	99,95	99,95	0,00	29,57
letter	77,65	77,66	0,02	59,91
penbased	95,52	98,80	3,31	136,37
ring	68,76	72,01	4,51	787,38
satimage	65,25	65,25	0,00	1455,80
segment	91,69	93,18	1,60	67,18
spambase	81,31	84,70	4,00	138,78
susy	73,84	73,84	0,00	71,31
texture	92,44	92,50	0,07	2695,45
twonorm	89,15	91,88	2,97	1775,08
vehicle	61,94	63,44	2,36	264,18
vowel	93,54	93,63	0,10	20,61
wineq-red	56,66	57,34	1,19	29,09
wineq-white	50,08	50,17	0,18	24,32
Promedio	77,02	78,17	1,43	160255,95

Tab. 4.16.: Resultados obtenidos en el Grupo D de conjuntos de datos (elevado número de reglas y variables continuas) en el algoritmo de inferencia híbrido. Hay cuatro columnas asociadas al nombre del conjunto de datos: (Test) y (Test*) son el porcentaje de ejemplos correctamente clasificados sobre el total de ejemplos y sólo sobre los ejemplos cubiertos respectivamente, (%NC) porcentaje de ejemplos no disparados por ninguna regla y (Tiempo(μs)) el tiempo medio consumido en la inferencia de un ejemplo. La última fila muestra el resultado medio de cada columna.

- En ambas versiones coincide el porcentaje de ejemplos que no disparan ninguna regla (*%NC*), y no solo un valor promedio de 1,43, sino que coincide en todos los conjuntos de datos. Esto indica que en ambas versiones se activan los mismos ejemplos y en consecuencia, el algoritmo de inferencia híbrido no activa ningún ejemplo distinto al activado por el enfoque secuencial.

- La precisión total de los ejemplos (la columna *Test*), ofrece de media un resultado muy similar, incluso algo superior para la nueva propuesta híbrida. Si se analiza este valor para cada conjunto de datos, se observa que las diferencias son muy inferiores a 0,5 puntos, excepto para el caso de *vehicle* donde la diferencia es de 1,66 puntos a favor de la nueva propuesta. A diferencia de todas les versiones anteriores de inferencia, esta es la única cuyos resultados son comparables (incluso mejores, aunque no significativamente) con el enfoque secuencial en este parámetro.

- La precisión de los ejemplos que se activan es muy similar entre las 2 versiones e incluso similar con los valores de la columna *Test*, ya que el porcentaje de ejemplos no activados por ninguna regla es bajo.

- En cuanto al tiempo empleado por la inferencia, la versión híbrida obtiene el mejor resultado en 14 de los 16 conjuntos de datos. La máxima reducción de tiempo se produce en el conjunto de datos *susy*, donde la versión híbrida es 1400 veces más rápida que el algoritmo de enfoque secuencial. En cualquier caso, este resultado no es frecuente, ya que no es normal tener un incremento tan elevado. Si se elimina este conjunto de datos en el cálculo medio, el aumento de la velocidad es de 8 veces más rápido.

Después de analizar los resultados obtenidos en la Tabla 4.16 y compararlos con los resultados obtenidos por el Algoritmo 2 en las Tablas 4.10 y 4.11, se puede concluir que el algoritmo de inferencia híbrido es una excelente alternativa a la inferencia estándar en situaciones con muchas reglas y variables continuas. Este algoritmo proporciona resultados similares en términos de precisión con una reducción significativa en el tiempo de ejecución.

A continuación, en el apartado de reflexiones se realizará un análisis global de todas las experimentaciones realizadas en este capítulo.

4.3.7. Reflexiones

El estudio experimental ha revelado que la versión conocida como “estándar”, que resuelve el problema utilizando el enfoque secuencial, obtiene resultados significativamente mejores cuando el número de reglas es bajo y el número de variables continuas es alto. En los casos en los que el número de variables continuas es bajo, el enfoque recursivo arroja los mejores resultados. Sin embargo, para el grupo de problemas que involucra un gran número tanto de reglas como de variables continuas, ninguno de los dos enfoques resultó adecuado por sí solo, debido al incremento de ejemplos no cubiertos. Para este último caso, mediante la integración de los métodos estándar y dos algoritmos de exploración de vecindarios en un algoritmo de inferencia híbrido, se ha logrado definir un método de inferencia adecuado para problemas con un elevado número de reglas y un elevado número de variables continuas.

4.4. Contribuciones

- *Efficient inference models for classification problems with a high number of fuzzy rules* Applied Soft Computing (Q1) 2022. [52]

5

Una reinterpretación de la forma en la que *WM-C* calcula el peso de las reglas.

En el siguiente capítulo, se abordan el Objetivo Específico 3 de esta tesis doctoral. Se enfoca en uno de los procesos que requiere más tiempo de aprendizaje al generar la base de reglas en el algoritmo de Wang y Mendel: la asignación de pesos. En este sentido, se propone un método computacionalmente menos costoso que consiste básicamente en reducir el conjunto de ejemplos sobre el cual se realiza el cálculo de los pesos.

El algoritmo de clasificación propuesto por Wang y Mendel (*WM-C*) [14] (ver 2.3) presenta diversas ventajas, como su simplicidad y eficiencia. No obstante, en situaciones particulares, como problemas especialmente complejos que surgen en el contexto del Big Data (ver 2.4.1), ha sido necesario recurrir a arquitecturas de procesamiento paralelo, como MapReduce [81, 32, 27, 1]. Estas arquitecturas se vuelven indispensables debido a que el cálculo de los pesos de las reglas se convierte en un cuello de botella en el algoritmo. Dicho cálculo requiere considerar la adaptación de todos los ejemplos para cada regla seleccionada. Aunque este cálculo tiene un orden polinómico, en problemas con numerosos ejemplos y un gran número de reglas, el proceso puede resultar muy lento en términos computacionales.

Los modelos de peso utilizados en este algoritmo han variado a lo largo del tiempo, desde el más simple utilizado en la versión original definido en la Ecuación 2.16, que no es operativo porque no tiene en cuenta el número de ejemplos soportados por cada regla, hasta modelos basados en el uso de frecuencias definido en la Ecuación 2.17. Sobre estos últimos, existen diferentes variantes como la propuesta denominada PCF expuesta en la Ecuación 2.19. Finalmente, también hay algunas propuestas que usan algoritmos de optimización para estimar a posteriori los pesos de las reglas[12].

En todos los casos, los modelos de obtención de pesos utilizados hasta la fecha comparten una característica común: representan una limitación significativa para el algoritmo *WM-C* (o sus versiones) a la hora de resolver problemas más complejos,

ya que en todos los casos utilizan el conjunto completo de ejemplos para realizar el cálculo del peso, lo cual es un serio problema cuando es necesario procesar un conjunto de ejemplos muy grande.

Con el propósito de abordar este problema, hemos realizado un análisis exhaustivo del algoritmo, evaluando cada una de sus componentes. A partir de este estudio, se confirmó que el cálculo del peso de cada regla es efectivamente el aspecto más costoso del algoritmo. En esta línea, proponemos en este capítulo un nuevo algoritmo que emplea una definición novedosa de los elementos involucrados en el cálculo del peso, principalmente basada en una reducción heurística de los ejemplos necesarios para dicho cálculo. Además, la nueva propuesta está estructurada de manera que optimiza todos los cálculos. Esta propuesta se valida considerando los diferentes modelos de peso previamente propuestos [14, 27].

El nuevo algoritmo desarrollado, denominado *QChi* (*Quick Chi*), ha demostrado ser más eficiente y rápido en su ejecución. Específicamente, ha logrado reducir significativamente el tiempo de cálculo en comparación con la versión original, sin comprometer el nivel de rendimiento de clasificación. Esto implica que, para ciertos problemas, no es necesario recurrir a estructuras de cálculo en paralelo, ya que el algoritmo *QChi* puede obtener resultados comparables con mayor eficiencia.

5.1. Preliminares

En esta sección se describen aquellos conceptos necesarios para comprender la propuesta, como el tipo de regla difusa, los tipos de pesos utilizados en la regla, la descripción de los diferentes elementos involucrados en el algoritmo de *WM-C* y un análisis de la complejidad de las diferentes etapas del algoritmo.

El tipo de regla con el desarrollamos la propuesta es la utilizada originalmente en [14], que se describe en la Ecuación 2.11. Se presenta nuevamente a continuación.

$$R: \textbf{IF } X_1 \text{ is } A^1_{j_1} \text{ and } X_2 \text{ is } A^2_{j_2} \text{ and } \ldots \text{and } X_n \text{ is } A^n_{j_n}$$
$$\textbf{THEN } Y \text{ is } B \text{ with } \textbf{weight } w(R)$$

Notaremos al antecedente de la regla como

$$ANT(R) = (A^1_{j_1}, A^2_{j_2}, \ldots A^n_{j_n}). \tag{5.1}$$

Las reglas extraídas por un algoritmo de aprendizaje inductivo se obtienen a partir de un conjunto de entrenamiento E. Cada ejemplo e en E tiene la forma $e = (e_1, e_2, \ldots, e_n, clase(e))$, donde $clase(e)$ es la clase asociada al ejemplo. En el marco de este análisis, se empleará el modelo de pesos conocido como PCF, el cual se encuentra definido en la Ecuación 2.19 y se presenta a continuación.

$$w(R) = PCF = \frac{n^+(R,E) - n^-(R,E)}{n(R,E)}$$

El Algoritmo 7 muestra una descripción del algoritmo *WM* aplicado a problemas de clasificación. El algoritmo tiene como entrada un conjunto de ejemplos E y una definición de los dominios asociados a las variables D (ver 2.10) implicadas en el problema de clasificación. La definición de los dominios también incluye el número y la semántica de las etiquetas lingüísticas. El algoritmo genera como salida un conjunto de reglas difusas con peso, que se almacenan en la estructura de datos $Rules$.

Algoritmo 7 Algoritmo *WM-C*

```
1:  function WM_ALGORITHM(E, D, Rules)
2:      Rules = {∅}
3:      for e ∈ E do
4:          r = BestAdaptationRule(e, D))
5:          AddToRuleSet(r, Rules)
6:      end for
7:      for r ∈ Rules do
8:          Calculate(n+(r,E), n−(r,E), n(r,E))
9:          w = GetWeight(r, n+(r,E), n−(r,E), n(r,E))
10:         PutWeight(r, w, Rules)
11:         ConflictResolution(r, Rules)
12:     end for
13:     return Rules
14: end function
```

En este algoritmo se pueden distinguir dos partes o etapas diferentes, una primera etapa de generación de reglas, ubicada entre los pasos 3 y 6 (el primer ciclo), y una segunda etapa encargada del cálculo de pesos y eliminación de reglas no representativas, que estaría entre los pasos 7 y 14 (correspondientes al segundo ciclo).

En relación a la primera etapa del algoritmo, se aplican dos procesos sobre cada ejemplo. En el primero de ellos, llamado *BestAdaptationRule*, se selecciona la regla r que mejor se adapta al ejemplo e y a los dominios de las variables D. En el segundo proceso, *AddToRuleSet*, se agrega la regla r previamente seleccionada al conjunto de reglas $Rules$.

En cuanto a la complejidad de estos procesos, se puede decir que, si el conjunto de reglas se modela utilizando una tabla hash, el tiempo para agregar una regla a ese conjunto, incluso verificando que no existía antes, es de orden constante. El orden de complejidad asociado con la obtención del antecedente con la mejor adaptación de un ejemplo es de orden n x L, donde n es el número de variables antecedentes y L es el número máximo de etiquetas difusas que componen los diferentes dominios de las variables, es decir, $L = max_i\{L_i\}$. Normalmente, L suele ser un valor menor que 9 y, en consecuencia, se puede considerar constante, en cuyo caso podríamos considerar que el orden de complejidad de este proceso depende exclusivamente de n. Dado que este proceso se repite para cada ejemplo en el conjunto E, el orden de complejidad de la parte de generación de reglas es O(n x $|E|$), donde $|E|$ es el número de ejemplos en el conjunto E.

La segunda etapa del algoritmo se ocupa del cálculo del peso de cada una de las reglas; consta de 4 pasos que deben aplicarse a cada una de las reglas obtenidas en la parte anterior. Estos pasos son, la función *Calculate*, que calcula los valores de $n^+(r)$, $n^-(r)$, $n(r)$ que permiten establecer el valor exacto del peso, dado un modelo de peso (cualquiera de los comentados anteriormente) y los valores calculados en el paso anterior, la función *GetWeight* obtiene el valor concreto del peso, la función *PutWeight* establece ese valor en la regla concreta (y elimina la regla r del conjunto de reglas si el peso es negativo), y la función *ConflictResolution* resuelve los posibles conflictos entre reglas, es decir, si hay varias reglas con el mismo antecedente pero diferente consecuente, sólo se incluye en $Rules$ la regla con mayor peso.

En resumen, el orden de complejidad de los tres últimos pasos de la segunda parte del algoritmo son constantes si se utilizan estructuras de datos adecuadas, pero el paso 8, es decir, el cálculo de los valores para establecer el peso, es el proceso más costoso computacionalmente. Si asumimos que L, el número de etiquetas difusas definidas en el dominio de la variable es constante, el orden de complejidad es n x $|E|$ por regla, siendo n el número de variables involucradas en el antecedente de la regla. Dado que el proceso se aplica para cada regla, el tiempo consumido en este proceso es O($|Rules|$ x n x $|E|$). En el peor de los casos, $|Rules| = |E|$ y, por lo tanto, el orden de complejidad es O(n x $|E|^2$).

Aunque el orden de complejidad de todo el proceso es polinómico, para problemas de Big Data en los que se trabaja con una cantidad masiva de datos y se generan un número muy alto de reglas, el cálculo exacto del peso ralentiza significativamente el proceso de extracción de reglas. De hecho, en trabajos como [81, 32, 27, 1] se hace uso del modelo MapReduce [25, 24], una técnica de procesamiento paralelo cuyo objetivo es reducir este tiempo de extracción de reglas.

Dado que el cálculo de pesos es el principal cuello de botella del algoritmo, se desea conocer la importancia del cálculo preciso de los pesos y si una estimación más rápida y simple permite una capacidad de predicción similar, pero con un tiempo de cálculo mucho más corto. La respuesta se encuentra en un nuevo algoritmo que se describe en la siguiente sección.

5.2. Método propuesto: El algoritmo *QChi* (Quick Chi).

Como ya hemos mencionado la idea central de la propuesta es acelerar el cálculo del peso de la regla reduciendo el conjunto de ejemplos sobre los que se calcula. Los tipos de pesos a utilizar son los mismos que se describen anteriormente, pero el cálculo de $n^+(R, E)$, $n^-(R, E)$ y $n(R, E)$ ahora se realizará en un subconjunto de E.

Se reutiliza el concepto de la "regla central" previamente definido en el Capítulo 4, donde se establece como la regla que posee el mayor grado de adaptación entre cada componente del ejemplo y cada componente de la regla, y se denotará como $C(e)$. La regla central no es más que la regla que asocia el algoritmo *WM-C* a cada ejemplo en el paso 4 del Algoritmo 7

Ahora, dada una regla R, primero se selecciona el subconjunto de ejemplos de E que tienen en su regla central el mismo antecedente que la regla R, es decir,

$$E^R = \{e \in E \mid ANT(C(e)) = ANT(R)\} \tag{5.2}$$

y en segundo lugar, en el cálculo del peso utilizamos $n^+(R, E^R)$, $n^-(R, E^R)$ y $n(R, E^R)$ en lugar de $n^+(R, E)$, $n^-(R, E)$ y $n(R, E)$ respectivamente.

Obviamente, en la mayoría de los casos

$$|E^R| \ll |E| \tag{5.3}$$

y el tiempo necesario para calcular $n^+(R, E^R)$, $n^-(R, E^R)$ y $n(R, E^R)$, y por lo tanto, el peso de la regla, se reduce de manera muy significativa.

Con este cambio, se asocia a cada regla una estimación de su peso exacto basado en tener en cuenta solo aquellos ejemplos que presentan la máxima compatibilidad con esa regla, dejando fuera de ese cálculo al resto de los ejemplos del conjunto de entrenamiento. Así, en el Algoritmo 8 se presenta un algoritmo de aprendizaje de reglas difusas para la clasificación basado en el algoritmo *WM-C* pero utilizando los subconjuntos E^R para obtener las estimaciones del peso. Además, el nuevo algoritmo está estructurado de manera diferente al original para realizar eficiente e incrementalmente los cálculos necesarios para obtener las estimaciones de peso.

Algoritmo 8 Algoritmo *QChi*

```
1:  function QChi(E, D, Rules)
2:      AntSet = {∅}, Rules = { ∅}
3:      for e ∈ E do
4:          a = BestAdaptAntedecent (e, D))
5:          UpdateAntSet (a, AntSet)
6:      end for
7:      for a ∈ AntSet do
8:          r = BuildRule(a, &w)
9:          if (w > 0) then
10:             AddToRuleSet(r, w, Rules)
11:         end if
12:     end for
13:     return Rules
14: end function
```

Las entradas del algoritmo son un conjunto de ejemplos E y una descripción del dominio de cada variable involucrada en el problema D. La salida es un conjunto de reglas difusas basado en el mismo modelo de reglas que se utiliza en el algoritmo de *WM-C*.

Al igual que con el algoritmo original, también se pueden distinguir dos etapas en el algoritmo propuesto: la primera con la generación de las reglas potenciales que podrían ser seleccionadas (desde el paso 2 hasta el 6) y la segunda con la elección final de las reglas. La principal diferencia es ahora el lugar donde se realiza

el proceso “más costoso” de cálculo del peso. En este nuevo algoritmo ese proceso se realiza en la primera parte, concretamente en el procedimiento *UpdateAntSet*. Así, en el paso 4, la función *BestAdaptAntecedent*, toma un ejemplo e y devuelve el antecedente a que tiene la mejor adaptación con ese ejemplo. La secuencia de antecedentes generada en este paso se almacena en el conjunto $AntSet$. Durante la ejecución del algoritmo, es *UpdateAntSet* el encargado de mantener actualizada la base de antecedentes generados. Esto significa que para cada antecedente se almacenan los valores acumulados de las adaptaciones de los ejemplos que ya han sido procesados para cada uno de los valores potenciales de la clase. Dado que el cálculo del peso se realizará teniendo en cuenta únicamente los ejemplos que consideraron ese antecedente como el mejor, este proceso de actualización tiene un orden de complejidad constante cuando se utiliza una estructura de tipo tabla hash para su implementación.

Es fácil ver cómo el orden de complejidad de esta primera etapa del algoritmo coincide con el orden de complejidad de la primera etapa del algoritmo 7. En ambos casos es O(n x $|E|$), donde $|E|$ es el número de ejemplos en el conjunto E y n es el número de variables involucradas en el problema de clasificación.

Siguiendo los pasos del algoritmo, al final del primer ciclo justo antes del paso 7, en el conjunto $AntSet$ tenemos todos los antecedentes que se generan a partir del conjunto completo de ejemplos de entrenamiento E (tal y como haría la versión original). En el paso 7, la función *BuildRule* se encarga de dar la regla final para cada antecedente considerado en el proceso anterior. Es decir, dado un antecedente a, completa la regla r añadiendo cuál será su consecuente y qué peso se le asignará. Para determinar estos valores, se calcula el valor del peso para cada posible valor del consecuente, y se devuelve el que obtenga el mayor valor de peso w (Se marca con un ’&’ en el algoritmo para indicar que es un parámetro de salida) y su consecuente asociado. El orden de complejidad asociado a este procedimiento es constante.

En los pasos 9 y 10, solo se incluyen en el conjunto final de $Rules$ aquellas reglas cuyo valor de peso sea estrictamente mayor que 0, mediante la función *AddToRuleSet*. Si se utiliza una estructura de datos eficiente para la inserción, el orden de complejidad es constante. Finalmente, el algoritmo devuelve $Rules$.

El orden de complejidad del algoritmo en su conjunto es el de la parte computacionalmente más costosa del algoritmo, que corresponde a la primera parte, es decir, O(n x $|E|$). Este es un orden de complejidad mejor que el del algoritmo original, que sería O(n x $|E|^2$) y esta diferencia se debe, como se mencionó anteriormente, a la forma diferente de calcular el peso.

5.3. Análisis e interpretación de los resultados obtenidos

Una vez que se ha descrito el algoritmo, en esta sección, vamos a mostrar como experimentalmente el uso de la estimación de peso propuesta no solo produce una reducción muy importante en el tiempo necesario para el aprendizaje, sino que también es capaz de mantener la capacidad de predicción obtenida al usar el peso utilizado en la versión original y basado en un cálculo exhaustivo en todos los ejemplos.

Esta sección experimental la dividiremos en dos partes. En la primera parte (subsección 5.3.1) se trata de demostrar experimentalmente el comportamiento del algoritmo *QChi* frente a la versión original del algoritmo *WM-C*, y para ello utilizamos 30 conjuntos de datos del repositorio UCI [71]. Los 30 conjuntos de datos seleccionados son considerados "simples" en el sentido de que ni por el número de ejemplos que contienen, ni por el número de variables involucradas, ni por el número de reglas que se generan, el cálculo exhaustivo del peso se considera realmente un inconveniente, ya que en comparación con otros algoritmos de clasificación, sus tiempos de aprendizaje son muy rápidos.

En la segunda parte (subsección 5.3.2) de esta sección experimental, utilizaremos conjuntos de datos masivos que suelen considerarse como muy costosos en términos de tiempo para la versión original de *WM-C* y para los cuales se han propuesto versiones de computación en paralelo utilizando el algoritmo *WM-C* como base. En particular, se usaron en propuestas que utilizan el paradigma MapReduce, con el fin de acelerar el tiempo necesario para el cálculo del peso.

5.3.1. Comparación de los algoritmos QChi y WM-C

En la primera parte de este estudio se utilizaron 30 conjuntos de datos para la clasificación que se describen en la Tabla del Anexo A.1, columna ($QChi$), todos ellos extraídos del repositorio de aprendizaje automático UCI [71].

Todos los experimentos descritos se realizaron en una computadora con CPU Intel Core i7-9750H @ 2.60GHz y 16 Gb de memoria, ejecutando el sistema operativo Ubuntu 20.04.3 LTS x86 64. Los algoritmos descritos aquí se implementaron en C++.

En el marco de la experimentación llevada a cabo en este estudio, se ha optado por emplear el método de inferencia descrito en el Capítulo 4, en particular se ha utilizado el Algoritmo 4. Como se ha señalado previamente, este algoritmo se caracteriza por su complejidad algorítmica, la cual es independiente del número de reglas involucradas. Este hecho hace que el tiempo de respuesta sea muy rápido en comparación con el ofrecido por la implementación usual de la inferencia.

En cuanto al peso, en esta experimentación se utilizó el modelo PCF descrito anteriormente en la Ecuación 2.19, y en relación con los dominios, todas las variables continuas se han discretizado utilizando un conjunto de 3,5 y 7 etiquetas difusas distribuidas uniformemente.

En esta primera parte de la experimentación, todos los resultados se han obtenido utilizando validación cruzada de 10 particiones, y se estudian tres parámetros: precisión (*Test*), número de reglas obtenidas por los algoritmos (*#Reglas*) y el tiempo (en segundos) necesario para extraer el conjunto de reglas difusas (*Tiempo*). En la Tabla 5.1 se muestran los resultados obtenidos utilizando 3 etiquetas difusas y se puede observar que las columnas se han agrupado para cada uno de los parámetros anteriores y para cada grupo hay una columna con los resultados de *QChi* y otra con los resultados de *WM-C*. En el caso del parámetro de *Tiempo*, se incluye una columna adicional denominada *Tasa*, que indica cuántas veces más rápido es *QChi* con respecto a *WM-C*. Los mejores resultados para cada conjunto de datos y para cada uno de los parámetros estudiados se resaltan en negrita.

Al analizar los datos de la Tabla 5.1, podemos ver que en todos los casos *QChi* obtiene un número de reglas similar o ligeramente superior a *WM-C*. En concreto, en media, el número de reglas obtenidas por *QChi* es un 6 % superior al obtenido por *WM-C*. La causa está en la resolución de conflictos y se debe al modelo de peso utilizado en *QChi*, que no permite eliminar todas las reglas que elimina *WM-C*. Sin embargo, dado que el número de reglas no es muy elevado, esto no es un problema relevante.

En relación al tiempo de aprendizaje, es muy evidente que *QChi* es significativamente más rápido que *WM-C* para todos los problemas estudiados, llegando a ser 185 veces más rápido para conjuntos de datos con muchos ejemplos y que generan muchas reglas, como en el caso de *adult*. En promedio, para todos los conjuntos de datos estudiados, es más de 40 veces más rápido.

Al analizar la capacidad predictiva, la conclusión no es tan simple. Podemos observar en la Tabla 5.1 que en algunos casos *QChi* es mejor y en otros *WM-C* es mejor. Si profundizamos más, podemos observar que en el caso del conjunto de datos *iris*, por

	Test		#Reglas		Tiempo(S)		
Conjunto de datos	QChi	WM-C	QChi	WM-C	QChi	WM-C	Tasa
abalone	**11,48**	0,02	**26,6**	1,2	**0,038691**	0,092450	2,39
adult	67,54	**67,97**	14577,8	14577,8	**0,621978**	115,650221	185,94
australian	69,86	**71,45**	350,0	350,0	**0,008228**	0,053709	6,53
banana	60,11	**60,21**	7,9	7,9	**0,020787**	0,034530	1,66
bands	**65,75**	64,93	292,4	292,4	**0,009184**	0,058442	6,36
bupa	**58,55**	57,97	45,7	45,7	**0,003249**	0,011463	3,53
coil2000	24,32	24,32	7392	7392	**0,959500**	12,142169	12,65
crx	53,29	**53,75**	435,2	435,2	**0,007456**	0,059354	7,96
iris	**94,67**	92,67	14,7	14,7	**0,000737**	0,001083	1,47
letter	**50,22**	34,07	**1320,9**	381,8	**0,304632**	11,143852	36,58
magic	**80,64**	76,75	354,1	354,1	**0,182082**	2,224813	12,22
newthyroid	**89,77**	84,65	20,5	20,5	**0,001017**	0,001867	1,84
mammogr	**79,28**	78,92	120,4	120,4	**0,005345**	0,018071	3,38
page-blocks	**92,54**	91,89	**53,3**	51,6	**0,055508**	0,154406	2,78
penbased	97,35	**97,67**	**3333**,8	3303,5	**0,153424**	7,068353	46,07
phoneme	**72,35**	71,82	58,4	58,4	**0,034246**	0,088160	2,57
pima	71,88	**72,79**	110,7	110,7	**0,010259**	0,033721	3,29
ring	**88,24**	55,19	1080,0	1080,0	**0,121407**	3,822583	31,49
saheart	69,48	**72,29**	190,9	190,9	**0,003201**	0,034588	10,81
satimage	**74,37**	47,47	**1458,6**	1228,7	**0,234403**	12,964517	55,31
segment	**86,62**	86,06	**318,3**	282,8	**0,037196**	0,281023	7,56
shuttle	**82,47**	80,16	**28,8**	26,7	**0,597692**	0,984147	1,65
spambase	**72,92**	71,50	372,1	372,1	**0,256961**	1,638114	6,37
texture	**77,38**	70,80	**1187,7**	907,2	**0,223541**	6,355889	28,43
thyroid	**92,35**	92,03	**463,0**	461,9	**0,130387**	0,785351	6,02
twonorm	88,84	**90,54**	1539,6	1539,6	**0,160099**	5,414788	33,82
vehicle	58,63	**59,69**	**405,6**	262,6	**0,014785**	0,177454	12,00
vowel	**64,24**	49,60	**279,6**	153,8	**0,015012**	0,066303	4,42
wineq-red	**52,03**	51,84	**219,7**	129,2	**0,019048**	0,156741	8,23
wineq-white	41,26	**48,43**	**260,9**	76,6	**0,064612**	0,678111	10,50
Promedio	**69,61**	65,91	**1210,6**	1141,0	**0,14**	6,07	42,42

Tab. 5.1.: Resultados obtenidos por el algoritmo *QChi* frente a *WM-C* utilizando una validación cruzada de 10 particiones y un dominio uniformemente distribuido con 3 etiquetas difusas en todas las variables continuas, donde la columna (Test) es la precisión en el conjunto de prueba, (#Reglas) muestra el número medio de reglas obtenidas, (Tiempo) es el número de segundos consumidos por el algoritmo de aprendizaje y (Tasa) es una medida que indica cuantas veces es más rápido *QChi* con respecto a *WM-C*.

ejemplo, ambos algoritmos tienen el mismo número de reglas; de hecho, obtienen exactamente las mismas reglas, pero con diferentes pesos. Sin embargo, el resultado de precisión es mejor en *QChi* que en *WM-C*. Algo similar sucede en otros conjuntos de datos, como magic o newthyroid, entre otros. Sin embargo, podemos observar el efecto opuesto en conjuntos de datos como *australian*, *saheart* o *twonorm*.

En la búsqueda de consolidar las conclusiones, se ha extendido la experimentación utilizando dominios con distribución uniforme con 5 etiquetas difusas en la Tabla 5.2 y 7 etiquetas difusas en la Tabla 5.3. Los nuevos resultados respecto al número de reglas y tiempo de aprendizaje refuerzan las conclusiones previas. En el caso de 5 etiquetas, *QChi* genera en promedio un 5 % más de reglas y es 25 veces más rápido, y con 7 etiquetas, genera un 1 % más de reglas y es 60 veces más rápido.

Al ampliar el estudio a un número mayor de etiquetas difusas en los dominios, es conveniente estudiar la capacidad de aprendizaje para verificar los resultados. Al usar 5 etiquetas difusas, ambos algoritmos generan exactamente las mismas reglas para el conjunto de datos *iris* (lo mismo que para 3 etiquetas), pero en este caso *WM-C* obtiene una precisión del 95,33 % y *QChi* del 94,67 % (justo lo contrario de lo que sucedió para 3 etiquetas difusas). Sin embargo, en los conjuntos de datos *magic* o *newthyroid*, *QChi* sigue mostrando mejores resultados y sigue siendo cierto que ambos tienen las mismas reglas.

Para determinar si existen diferencias significativas entre ambos algoritmos en la capacidad predictiva y en el tiempo de aprendizaje, se aplica una prueba de rangos con signo de Wilcoxon [96] con un intervalo de confianza del 95 %. Los resultados de esta prueba, para diferentes números de etiquetas difusas en los dominios, se muestran en la Tabla 5.4, que representa los resultados obtenidos en estas comparaciones, indicando las victorias (W), empates (T) y derrotas (L) de *WM-C* frente a *QChi* junto con el p-valor calculado.

En el caso de dominios con tres etiquetas, si formulamos la hipótesis nula de que la capacidad predictiva de ambos algoritmos es igual, la prueba devuelve un p-valor inferior a 0,05, por lo que se rechaza la hipótesis con una confianza del 95 %. Al descartar la igualdad, podemos suponer que los valores obtenidos por *QChi* son superiores a los obtenidos por *WM-C*. Por lo tanto, se puede inferir que *QChi* es significativamente mejor en capacidad predictiva que *WM-C* en estos conjuntos de datos cuando se utilizan 3 etiquetas.

También se puede observar que para 5 etiquetas se acepta que los resultados de la prueba de ambos algoritmos son iguales. Sin embargo, para 7 etiquetas, se rechaza la hipótesis de igualdad y se puede asumir que se acepta que la capacidad

	Test		#Reglas		Tiempo(s)		
Conjunto de Datos	QChi	WM-C	QChi	WM-C	QChi	WM-C	Tasa
abalone	**16,70**	1,05	**122,7**	12,2	**0,03687**	0,16382	4,44
adult	63,32	**64,38**	19796,4	19796,4	**0,69185**	150,03886	216,87
australian	54,64	54,64	498,9	498,9	**0,00985**	0,07474	7,59
banana	79,38	**80,36**	19,6	19,6	**0,02437**	0,04136	1,70
bands	16,16	**16,44**	325,3	325,3	**0,00905**	0,02881	3,18
bupa	55,65	**59,71**	120,8	120,8	**0,00196**	0,01101	5,61
coil2000	24,27	24,27	7451,9	**7455,8**	**0,89667**	11,31296	12,62
crx	40,74	**41,19**	522,7	522,7	**0,01236**	0,06412	5,19
iris	94,67	**95,33**	45,5	45,5	**0,00118**	0,00191	1,63
letter	**77,43**	74,60	**7488,1**	4962,3	**0,38233**	34,05002	89,06
magic	**81,63**	80,80	1912,4	1912,4	**0,21723**	7,05275	32,47
newthyroid	**95,35**	91,16	48,0	48,0	**0,00130**	0,00542	4,17
mammogr	**76,87**	76,75	**160,0**	159,6	**0,00564**	0,03283	5,83
page-blocks	**94,39**	93,90	**163,1**	161,9	**0,06455**	0,22144	3,43
penbased	94,01	**94,30**	**7739,0**	7737,1	**0,18145**	10,59466	58,39
phoneme	**80,59**	78,66	228,2	228,2	**0,03800**	0,21914	5,77
pima	66,02	**70,57**	426,6	426,6	**0,00844**	0,06223	7,38
ring	**54,62**	53,24	5059,5	5059,5	**0,17224**	11,32908	65,78
saheart	60,82	**62,12**	354,0	354,0	**0,00678**	0,05100	7,53
satimage	**64,83**	51,24	**2137,4**	1859,1	**0,25657**	15,07542	58,76
segment	**90,39**	88,48	**884,9**	863,3	**0,04983**	0,40701	8,17
shuttle	**92,42**	84,24	**71,9**	70,1	**0,61837**	1,45169	2,35
spambase	80,66	80,66	1379,5	1379,6	**0,28493**	3,75509	13,18
texture	**87,40**	85,84	**3694,8**	3473,3	**0,25783**	10,71664	41,56
thyroid	91,96	**92,10**	941,2	**943,9**	**0,13164**	1,25975	9,57
twonorm	34,14	**34,93**	6654,2	6654,2	**0,17540**	11,33177	64,60
vehicle	51,89	**54,49**	**707,8**	683,2	**0,01870**	0,18749	10,03
vowel	**93,43**	93,03	**649,7**	638,8	**0,01469**	0,13457	9,16
wineq-red	56,79	**59,60**	**763,4**	614,2	**0,02520**	0,28823	11,44
wineq-white	49,80	**53,08**	**1043,2**	651,5	**0,06547**	1,70265	26,01
Promedio	**67,37**	66,37	**2380,4**	2255,9	**0,15536**	9,05555	26,45

Tab. 5.2.: Resultados obtenidos por el algoritmo *QChi* frente a *WM-C* utilizando una validación cruzada de 10 particiones y un dominio uniformemente distribuido con 5 etiquetas difusas en todas las variables continuas, donde la columna (Test) es la precisión en el conjunto de prueba, (#Reglas) muestra el número medio de reglas obtenidas, (Tiempo) es el número de segundos consumidos por el algoritmo de aprendizaje y (Tasa) es una medida que indica cuantas veces es más rápido *QChi* con respecto a *WM-C*.

	Test		#Reglas		Tiempo(s)		
Conjunto de datos	QChi	WM-C	QChi	WM-C	QChi	WM-C	Tasa
abalone	**18,83**	3,07	**272,1**	39,6	**0,034729**	0,266552	7,68
adult	59,46	**60,29**	23543,9	**23544**	**0,607013**	143,108063	235,76
australian	44,35	**45,51**	571,6	571,6	**0,006745**	0,067386	9,99
banana	86,23	**87,36**	32,7	32,7	**0,021274**	0,045084	2,12
bands	4,11	4,11	328,5	328,5	**0,007951**	0,023434	2,95
bupa	54,20	**58,55**	186,2	186,2	**0,004179**	0,020969	5,02
coil2000	23,71	23,71	7455,3	**7455,9**	**0,910875**	10,609804	11,65
crx	30,78	**31,24**	551,6	551,6	**0,00908**	0,066846	7,36
iris	92,00	**92,67**	66,1	66,1	**0,001002**	0,002167	2,16
letter	82,13	**82,79**	**13086,4**	12643,9	**0,371883**	40,1901	108,07
magic	80,29	**81,38**	4517,4	4517,4	**0,237756**	15,002445	63,10
newthyroid	91,16	**92,09**	67,4	67,4	**0,001202**	0,003458	2,88
mammogr	73,49	**74,82**	183,1	**183,9**	**0,005499**	0,023199	4,22
page-blocks	**94,50**	94,13	**292,4**	289,4	**0,057951**	0,332775	5,74
penbased	77,49	**77,52**	9394,6	9394,6	**0,179212**	11,401409	63,62
phoneme	**82,85**	82,07	471,1	471,1	**0,035796**	0,355269	9,92
pima	60,94	**62,37**	599,5	599,5	**0,008396**	0,099735	11,88
ring	39,78	**39,96**	6612,4	6612,4	**0,165381**	10,547319	63,78
saheart	46,75	**46,97**	399,1	399,1	**0,005524**	0,040247	7,29
satimage	70,30	**72,70**	**3237,3**	2992,7	**0,249837**	15,094193	60,42
segment	**88,57**	88,53	**1226,7**	1217,1	**0,047331**	0,442371	9,35
shuttle	**97,70**	91,59	**85,5**	84,6	**0,583724**	1,567628	2,69
spambase	75,77	**76,70**	2154,8	**2155,6**	**0,270649**	4,471193	16,52
texture	75,31	**75,60**	**4389,1**	4366,3	**0,289956**	8,216433	28,34
thyroid	91,40	**91,60**	**1165,3**	1161,5	**0,125682**	1,376095	10,95
twonorm	0,26	0,26	6660	6660	**0,157665**	7,579011	48,07
vehicle	29,08	**29,31**	**755,6**	753,4	**0,021342**	0,129281	6,06
vowel	92,73	**93,03**	**748,6**	748,4	**0,013008**	0,16008	12,31
wineq-red	**59,29**	59,10	**1028,2**	951	**0,025189**	0,364136	14,46
wineq-white	54,33	**55,29**	**2145,5**	1640,2	**0,081322**	2,643661	32,51
Promedio	**62,59**	62,48	**3074,3**	3022,9	**0,15124**	9,14168	60,45

Tab. 5.3.: Resultados obtenidos por el algoritmo *QChi* frente a *WM-C* utilizando una validación cruzada de 10 particiones y un dominio uniformemente distribuido con 7 etiquetas difusas en todas las variables continuas, donde la columna (Test) es la precisión en el conjunto de prueba, (#Reglas) muestra el número medio de reglas obtenidas, (Tiempo) es el número de segundos consumidos por el algoritmo de aprendizaje y (Tasa) es una medida que indica cuantas veces es más rápido *QChi* con respecto a *WM-C*.

Test		
WM-C vs QChi	W/T/L	p-valor
3 etiquetas	10/1/19	0,03995
5 etiquetas	14/3/13	0,69180
7 etiquetas	21/3/6	0,01125

Tiempo de Aprendizaje		
WM-C vs QChi	W/T/L	p-valor
3 etiquetas	0/0/30	0,000000001
5 etiquetas	0/0/30	0,000000001
7 etiquetas	0/0/30	0,000000001

Tab. 5.4.: Pruebas estadísticas que comparan los resultados obtenidos en precisión (Test) y tiempo de aprendizaje por *WM-C* y *QChi* para distintos números de etiquetas difusas en los dominios asociados a variables continuas.

predictiva de *WM-C* es mejor que la de *QChi*. En el análisis comparativo del tiempo de aprendizaje entre ambos algoritmos, se ha constatado la presencia de diferencias significativas en relación con los tres tipos de etiquetas evaluadas. Los resultados de este análisis revelan de manera concluyente que el algoritmo *QChi* presenta un tiempo de aprendizaje notablemente inferior en comparación con el que se obtiene mediante la ejecución del algoritmo de *WM-C*.

En resumen, no se puede concluir que el cálculo exhaustivo del valor de peso utilizado habitualmente en el algoritmo *WM-C* conduzca a una mejora significativa en los resultados de la capacidad predictiva y, por lo tanto, un modelo de cálculo más simple y selectivo que proporcione una estimación rápida de estos valores, como el propuesto aquí, puede reducir significativamente el tiempo de aprendizaje y producir una buena capacidad predictiva.

5.3.2. QChi frente a las propuestas de MapReduce

Algunos autores han visto la posibilidad de utilizar el algoritmo *WM-C* para abordar problemas de Big Data. En la sección 5.3.1 se muestra el tiempo de aprendizaje del algoritmo *WM-C* para conjuntos de datos con menos de 60 mil ejemplos y se puede observar que sus tiempos, en comparación con los de otros clasificadores conocidos, son realmente bajos, aunque en problemas más complejos donde se generan muchas reglas, el tiempo de aprendizaje aumenta significativamente.

Si observamos la Tabla 5.1, podemos ver que el conjunto de datos *shuttle*, que contiene 57,999 ejemplos, tiene un tiempo de aprendizaje de menos de un segundo (0,9841 en promedio). El tiempo es muy bajo y la razón es que genera en promedio

solo 26,7 reglas. Sin embargo, otros conjuntos de datos, como *adult*, con 45,222 ejemplos, generan 14,578 reglas y esto hace que el tiempo de aprendizaje sea cercano a dos minutos. Obviamente, estos resultados de tiempo aumentan para conjuntos de datos con muchos más ejemplos y generando muchas más reglas.

Con la idea de abordar problemas más complejos utilizando el algoritmo *WM-C*, algunos autores han propuesto insertar este algoritmo en un esquema de procesamiento paralelo conocido como MapReduce [25] y la idea de esta parte experimental es conocer la relación entre *QChi* y estos métodos.

Como se describe en el Algoritmo 7, el cuello de botella del algoritmo *WM-C* se encuentra en el cálculo del peso, cuyo orden de complejidad depende del producto del número de ejemplos por el número de reglas. Las versiones propuestas que utilizan este esquema buscan aliviar este cálculo mediante el uso de procesamiento paralelo.

Conjunto de Datos	Vt	Vc	Ex	(P_{may}:P_{min})
kddcup*	41	26	4.856.151	(3.883.370 : 972.791)
poker*	10	10	946.799	(513.702 : 433.097)
covtype*	54	10	495.141	(283.301 : 211.840)
census	41	13	142.521	(134.359 : 8.162)
fars*	29	5	62.123	(42.116 : 20.007)

Tab. 5.5.: Conjuntos de datos utilizados en el estudio experimental de [32], donde (Vt) es el número total de variables, (Vc) son las variables continuas, (Ex) es el número total de ejemplos y (P_{may}:P_{min}) es el número de ejemplos de la clase mayoritaria y minoritaria.

En [81] se presentan las primeras propuestas que combinan el algoritmo *WM-C* (se le denomina algoritmo Chi) con MapReduce, y resultan en un algoritmo llamado Chi-FRBCS-BigData. Este algoritmo divide el conjunto de ejemplos de entrenamiento en tantos subconjuntos como procesamiento paralelo se vaya a realizar. En cada subconjunto se aplica el algoritmo *WM-C* y se obtiene un conjunto de reglas. En el proceso final, se toma cada conjunto de reglas obtenido de cada subconjunto de ejemplos y se construye el conjunto final de reglas combinándolos todos. En este proceso de fusión, puede haber reglas con el mismo antecedente y consecuente diferente cuyo conflicto se resuelve incluyendo la regla con el consecuente con mayor peso asociado y reglas con el mismo antecedente y el mismo consecuente, pero diferentes pesos. En estos casos, se definen dos versiones diferentes: Chi-FRBCS-BigData-Max que toma el peso de la regla con mayor peso y Chi-FRBCS-BigData-Ave que incluye la regla con el peso promedio entre las reglas involucradas.

Los resultados obtenidos con estas dos versiones no son exactamente los mismos que los obtenidos por *WM-C* en los mismos conjuntos de datos, pero los autores

		Test			Tiempo(s)			
		Chi-FRBCS-BigData			Chi-FRBCS-BigData			
Conjunto Datos	Etiquetas	32 maps	512 maps	QChi	32 maps	512 maps	QChi	Tasa
kddcup*	3	99,92498	99,94120	**99,95502**	7890,87	9602,99	**161,32**	48,9
poker*	3	58,92973	56,79368	**61,42863**	2210,13	4492,45	**10,35**	213,5
covtype*	3	74,61723	73,65237	**77,13470**	391,40	3880,21	**20,02**	19,5
census	3	93,48355	92,66282	**95,96784**	388,64	714,42	**5,20**	74,7
fars*	3	94,25642	93,56599	**100,00000**	141,92	377,25	**1,71**	83,0
Promedio		84,24	83,32	**86,90**	2204,60	3813,5	**39,72**	55.5
kddcup*	5	99,95345	99,95085	**99,95745**	18710,18	15784,56	**167,11**	94,5
poker*	5	56,87383	56,79511	**57,88643**	8675,26	4475,27	**12,06**	371,1
covtype*	5	83,24885	81,34559	**85,24857**	4663,55	4910,73	**20,06**	232,5
census	5	94,68176	94,18538	**96,33379**	650,44	762,38	**5,66**	114,9
fars*	5	99,86419	99,84249	**100,00000**	180,77	376,45	**1,91**	94,6
Promedio		86,92	86,42	**87,88**	6576,04	5261,88	**45,27**	116,2
kddcup*	7	99,95500	99,95425	**99,95809**	25638,19	22540,5	**180,76**	124,7
poker*	7	59,98018	59,92133	**60,35220**	8286,67	4387,05	**12,24**	358,4
covtype*	7	87,43990	86,15001	**88,91371**	6891,16	5571,43	**20,94**	266,1
census	7	95,54433	95,36667	**96,59426**	658,55	760,92	**5,41**	121,7
fars*	7	99,94522	99,94522	**100,00000**	182,27	376,11	**1,96**	92,9
Promedio		88,57	88,27	**89,16**	8331,37	6727,20	**44,26**	152,0

Tab. 5.6.: Resultados obtenidos por el algoritmo *QChi* frente a Chi-FRBCS-BigData, donde la columna (Test) es la precisión en el conjunto de prueba, (Tiempo) es el número de segundos consumidos por el algoritmo de aprendizaje, (maps) conocido como mapper, es un servidor de Hadoop que ejecuta las funciones de MapReduce y (Tasa) es una medida que indica cuantas veces es más rápido *QChi* con respecto al menor valor de Chi-FRBCS-BigData.

concluyen al respecto con *"En términos de precisión alcanzada por los algoritmos considerados en el estudio podemos observar que, en general, las versiones de Chi-FRBCS-BigData son capaces de proporcionar mejores resultados de clasificación que la versión secuencial"*, lo que demuestra que el cálculo exacto del peso no garantiza una mejor capacidad predictiva.

En [32] se extiende el estudio del comportamiento de Chi-FRBCS-BigData-Max cambiando la granularidad de las particiones difusas utilizadas. Para dicho estudio se utilizan los 5 conjuntos de datos descritos en la Tabla 5.5, donde *kddcup** es una versión binaria del conjunto de datos *KDDCup1999* tomando solo ejemplos de la clase de denegación de servicio (DOS) vs. clase normal, *poker* es una versión binaria del conjunto de datos *poker** tomando solo las clases 0 y 1, *covtype** es una versión binaria del conjunto de datos *CovType* tomando solo las clases 2 y 1, y *fars** es una versión binaria del conjunto de datos *Fars* tomando solo las clases Fatal_Injury y No_Injury.

La Tabla 5.6 muestra un resumen de los resultados obtenidos por Chi-FRBCS-BigData (correspondiente a Chi-FRBCS-BigData-Max) y ofrecidos por sus autores en [32], a los que se suman los resultados obtenidos por *QChi*, en los que se han reproducido

las mismas condiciones utilizadas por los primeros y para los mismos conjuntos de datos.

Se estudian dos parámetros: la capacidad predictiva en el conjunto de prueba, utilizando la medida estándar de precisión, y el tiempo de aprendizaje. Los resultados que se muestran en la tabla corresponden a la media de una validación cruzada de 10 particiones. El modelo de peso utilizado es el PCF (ecuación 2.19) y el método de razonamiento es la regla ganadora. Un parámetro importante para la versión Chi-FRBCS-BigData es el número de mappers (mapper es un servidor de Hadoop que ejecuta las funciones de MapReduce) utilizados en el esquema MapReduce. En la tabla, se han tomado los valores extremos que se consideraron en el experimento propuesto por los autores en [32], que corresponden a 32 y 512 mappers (maps). La infraestructura utilizada para obtener los resultados fue un clúster con 16 nodos conectados con un Infiniband de 40Gb/s. Cada nodo está equipado con dos microprocesadores Intel E5-2620 (a 2 GHz, 15 MB de caché) y 64 GB de memoria principal que funcionan bajo Linux CentOS 6.6. El nodo principal del clúster está equipado con dos microprocesadores Intel E5645 (a 2,4 GHz, 12 MB de caché) y 96 GB de memoria principal. Además, el clúster funciona con Hadoop 2.6.0 (Cloudera CDH5.4.0), donde el nodo principal está configurado como nombre-nodo y job-tracker, y el resto son nodos de datos y task-trackers.

Los resultados de *QChi* se obtuvieron en una CPU Intel Core i7-9750H con una velocidad de reloj de 2,60 GHz y 16 GB de memoria, ejecutando el sistema operativo Ubuntu 20.04.3 LTS x86 64.

En la Tabla 5.6 se destacan en negrita los mejores resultados para cada uno de los dos parámetros estudiados. Se puede observar que para todos los conjuntos de datos y todas las granularidades diferentes, el tiempo de aprendizaje ofrecido por *QChi* es muy inferior a los ofrecidos por las versiones Chi-FRBCS-BigData, independientemente del número de mappers utilizados. La última columna (Tasa) indica cuántas veces es más rápido *QChi* que la versión más rápida de Chi-FRBCS-BigData (entre 32 y 512 mappers). Puede verse que *QChi* se vuelve comparativamente más eficiente a medida que aumenta la granularidad.

En relación con la capacidad de predicción, podemos observar que nuestra propuesta obtiene los mejores resultados en todas las granularidades estudiadas. Estos resultados muestran que el enfoque Chi-FRBCS-BigData no es mejor en precisión y se ve ampliamente superado por los tiempos de aprendizaje obtenidos por *QChi*.

En [27], se propone un algoritmo alternativo a Chi-FRBCS-BigData. La nueva propuesta, llamada CHI-BD, utiliza *WM-C* como algoritmo base y también utiliza una

arquitectura MapReduce. Aunque los dos enfoques son muy similares en la forma en que abordan el problema de clasificación utilizando procesamiento paralelo, tienen dos diferencias importantes. La primera es que CHI-BD reproduce el mismo resultado que se obtendría aplicando el algoritmo de *WM-C*, ya que reproduce fielmente el peso que el algoritmo original asociaría a cada una de las reglas proporcionadas como salida. La segunda diferencia es que se incluye un método de preprocesamiento de pesos durante el proceso de aprendizaje, lo que acelera el cálculo de los pesos. El estudio experimental utiliza 20 conjuntos de datos de clasificación binaria que fueron generados a partir de ocho conjuntos de datos del repositorio UCI. La Tabla del Anexo A.2, muestra los conjuntos de datos considerados para la experimentación y recopila varios datos como el número total de ejemplos (Ex), el número de ejemplos de la clase mayoritaria y minoritaria (Pmay:Pmin), el número total de variables involucradas en el problema (Vt) y cuántas de ellas son continuas (Vc).

Los conjuntos de datos *census*, *higgs*, *skin* y *susy* son binarios y, por lo tanto, se usaron en su versión original. El resto son multiclase y se transformaron a binarios considerando los ejemplos de una de las clases como la clase positiva y el resto como la clase negativa. Las abreviaturas en el nombre de los conjuntos de datos son las siguientes: *cov* (Covtype), *far* (Fars), *KDD* (KDDCup1999) y *pok* (Poker). Para conjuntos de datos multiclase, se ha añadido el identificador de clase positiva al nombre del conjunto de datos. Para los conjuntos de datos donde los valores asociados a la clase son numéricos, se identifican con el número asociado a la clase que se considera como positiva. Para los demás conjuntos de datos, se realizó la siguiente asociación: para *Far*, Fat es la clase Fatal_Injury, Inc es Incapaciting_Injury, No es No_Injury y Nin es Nonincapaciting_Evident_Injury. En el caso de *KDD*, se consideraron cuatro conjuntos de datos: Two, Nor, prb y r21.

En este estudio se establecen las siguientes condiciones:

- Se utiliza una validación cruzada de 5 particiones.
- Se utiliza una configuración del algoritmo de *WM-C* como algoritmo base, que usa una discretización de 3 etiquetas de las variables continuas, el método de inferencia de regla ganadora y el modelo de peso PCF-CS (Ecuación 2.21).
- Debido a que la mayoría de los conjuntos de datos utilizados están fuertemente desequilibrados, el Área Bajo la Curva ROC (AUC) y la Media Geométrica (GM) se utilizan como medida de calidad de la clasificación.
- Todos los métodos paralelos se han ejecutado en un clúster de 8 nodos conectados mediante una red LAN Ethernet de 1 Gb/s. Para obtener más detalles, consulte el artículo [27].

	Test (Media Geométrica)			Test (AUC)			Tiempo(s)			#Reglas		
Conjunto de Datos	CHI^{BD}_{Local}	CHI^{BD}_{Global}	QChi	CHI^{BD}_{Local}	CHI^{BD}_{Global}	QChi	CHI^{BD}_{Local}	CHI^{BD}_{Global}	QChi	CHI^{BD}_{Local}	CHI^{BD}_{Global}	QChi
census	0,403	**0,5231**	0,36442	0,5757	0,6220	**0,6541**	26	96	**5,3**	64137	**63598**	64166,8
cov_1	0,7528	0,7531	**0,7714**	0,7530	0,7532	**0,7592**	68	76	**24,8**	8275	**7940**	8859,2
cov_2	0,7296	0,7291	**0,7700**	0,7379	0,7373	**0,7730**	70	75	**25,7**	8372	**8108**	8862,2
cov_3	0,9551	**0,9565**	0,9517	0,9553	**0,9572**	0,7926	69	74	**22,2**	8438	**8249**	8866
cov_7	0,9089	0,9281	**0,9313**	0,9109	**0,9285**	0,8225	70	75	**22,5**	8181	**7917**	8859,7
Media cov	0,8366	0,8417	**0,8561**	0,8393	**0,8441**	0,7868	69,3	75,0	**23,8**	8317	**8053**	8861,7
far_Fat	0,5871	0,5871	**0,9123**	0,6723	0,6722	**0,9908**	24	81	**2,9**	**49707**	**49707**	49727,4
far_Inc	0,5572	0,6919	**0,7419**	0,6370	**0,7123**	0,7037	24	80	**2,9**	49692	**49130**	49725,2
far_No	0,8336	0,8675	**0,9049**	0,8438	0,8728	**0,9139**	23	82	**2,6**	49700	**49584**	49720,6
far_Nin	0,5307	**0,7139**	0,7108	0,6195	**0,7283**	0,6728	24	81	**2,8**	49686	**48984**	49711,6
Media far	0,6272	0,7151	**0,8175**	0,6931	0,7464	**0,8203**	23,8	81,0	**2,8**	49696	**49351**	49721,2
KDD_dos	0,9991	0,9991	**0,9992**	**0,9991**	**0,9991**	0,9989	4362	**78**	155,7	5753	5747	**5387,4**
KDD_nor	0,9992	0,9992	**0,9993**	**0,9992**	**0,9992**	0,9985	4317	**76**	154,1	5755	5734	**5379,8**
KDD_prb	0,9911	0,9924	**0,9925**	0,9911	**0,9925**	0,9587	4402	**77**	153,4	5750	5701	**5379,6**
KDD_r2l	0,9251	**0,9840**	0,9837	0,9279	**0,9841**	0,5529	4412	**75**	155,9	5744	5686	**5381,8**
Media KDD	0,9786	**0,9937**	**0,9937**	0,9793	**0,9937**	0,8773	4373,3	**76,5**	154,8	5751	5717	**5382,1**
pok_0	0,6183	**0,6336**	0,6224	0,6206	**0,6360**	0,6224	58	107	**10,3**	56023	**54523**	56181,5
pok_1	0,5616	**0,5848**	0,5679	0,5658	**0,5859**	0,5698	59	110	**10,7**	55986	**54254**	56181,5
pok_2	0,3713	**0,6703**	0,5411	0,5473	**0,6709**	0,5414	57	109	**10,0**	55408	**46618**	56177,4
pok_3	0,272	**0,7387**	0,5954	0,5302	**0,7388**	0,5349	59	106	**11,0**	55480	**44632**	56190
Media poker	0,4558	**0,6569**	0,5817	0,5660	**0,6579**	0,5671	58,3	108,0	**10,5**	55724	**50007**	56182,6
skin	0,9595	**0,9597**	0,9381	0,9604	**0,9605**	0,9246	22	53	**1,3**	**23**	**23**	25
susy	0,5477	0,5524	**0,6722**	0,6210	0,6242	**0,6860**	2019	103	**90,5**	9675	**9505**	10516,6
Promedio	0,7117	0,7823	**0,7879**	0,7718	**0,8085**	0,7676	1061,3	84,9	**45,5**	29041	**27665**	29338,3
higgs	0,5772	**0,5847**	0,5639	0,5776	**0,5848**	0,5642	15115	9658	**263,2**	762443	**666068**	832577

Tab. 5.7.: Resultados obtenidos por el algoritmo *QChi* frente a CHI^{BD}_{Local} y CHI^{BD}_{Global}, utilizando una validación cruzada de 5 particiones y un dominio uniformemente distribuido con 3 etiquetas en todas las variables continuas, donde las columnas (Test) corresponden a la media geométrica y el área bajo la curva en el conjunto de prueba, (Tiempo) es el número de segundos consumidos por el algoritmo de aprendizaje y (#Reglas) muestra el número medio de reglas obtenidas.

La tabla 5.7 muestra los resultados obtenidos por Chi-FRBCS-BigData (en la tabla CHI^{BD}_{Local}), Chi-BD (en la tabla CHI^{BD}_{Global}) y *QChi* para los parámetros de capacidad predictiva (medidos mediante media geométrica y AUC) y tiempo de aprendizaje. Tanto para CHI^{BD}_{Local} como para CHI^{BD}_{Global} se toman los valores proporcionados en [27] para el caso de 32 mappers ya que como los propios autores indican es el máximo que se puede ejecutar en paralelo en su cluster. Los resultados relativos a *QChi* mostrados en la tabla se han obtenido utilizando la misma configuración que los obtenidos en dicho trabajo pero en este caso se ha ejecutado sobre una CPU Intel Core i7-9750H @ 2,60GHz y 16 Gb de memoria corriendo sobre el sistema operativo Ubuntu 20.04.3 LTS x86 64, el mismo utilizado en los estudios anteriores.

La tabla resalta los mejores resultados obtenidos para cada uno de los parámetros. También se incluye una fila con el promedio de aquellos conjuntos de datos que se han utilizado para generar más de un problema. Cabe señalar que el conjunto de datos *higgs* no se ha considerado en el cálculo de la media debido a su alto

tiempo de ejecución (incluso más en el caso de modelos que utilizan MapReduce) que distorsiona su valor.

Se puede observar que el número de reglas obtenido por *QChi* en la mayoría de los casos es superior al obtenido por los otros dos enfoques, siendo CHI$^{BD}_{Global}$ el que tiende a obtener el valor más bajo. Un comportamiento similar (más acentuado en conjuntos de datos con muchos ejemplos) en relación con el número de reglas se puede observar con los resultados obtenidos en la Tabla 5.1.

En cuanto al tiempo de aprendizaje, se observa que *QChi* obtiene los mejores resultados en 16 de los 20 problemas, siendo superado únicamente por CHI$^{BD}Global$ en los problemas relacionados con el conjunto de datos *KDD*. Al observar los tiempos promedio, se puede ver que es del orden de dos veces más rápido que CHI$^{BD}Global$ y más de 23 veces más rápido que CHI$^{BD}_{Local}$. En el caso especial del conjunto de datos *higgs*, estas diferencias son aún mayores, siendo más de 35 y 50 veces más rápido que ellos, respectivamente.

En cuanto al rendimiento de la prueba utilizando la media geométrica, los mejores resultados los obtienen sobre todo *QChi* (10 de 20) y CHI$^{BD}_{Global}$ (9 de 20), mientras que CHI$^{BD}_{Local}$ sólo es el mejor en *higgs*. Si observamos con más detenimiento estos resultados en grupos de conjuntos de datos generados a partir del mismo conjunto de datos, podemos ver que para el conjunto de datos *Poker*, CHI$^{BD}_{Global}$ obtiene claramente los mejores resultados, para el conjunto de datos *Fars*, *QChi* es globalmente mejor, mientras que los resultados obtenidos por ambos enfoques en los conjuntos de datos *KDDCup1999* y *covtype* son muy similares

En relación con el rendimiento de la prueba mediante AUC, a diferencia del anterior, los mejores resultados los obtiene CHI$^{BD}_{Global}$ (14 de 20), mientras que CHI$^{BD}_{Local}$ iguala como mejor a CHI$^{BD}_{Global}$ en 2 de esos 16. Para esta medida, *QChi* es el mejor sólo en 6.

En [27] se incluye un estudio estadístico que muestra que existen diferencias significativas a favor de CHI$^{BD}_{Global}$ en el rendimiento de clasificación. Para encontrar diferencias significativas entre el rendimiento de clasificación y el tiempo de aprendizaje de estos tres métodos, se han utilizado las pruebas de rango con signo de Wilcoxon para comparar los dos métodos de MapReduce anteriores con *QChi*.

En la Tabla 5.8 se muestran los 6 estudios realizados, donde se representan los resultados obtenidos en estas comparaciones, indicando las victorias (W), empates (T) y derrotas (L) de CHI$^{BD}_{Local}$ frente a *QChi* junto con el p-valor calculado. La primera estudia el rendimiento de clasificación utilizando la Media Geométrica entre CHI$^{BD}_{Local}$ y *QChi*, donde se rechaza la hipótesis de igualdad a un nivel de confianza

del 95 %. Por tanto, podemos suponer que la capacidad de clasificación es mayor en *QChi*. Considerando la medida AUC, en el segundo estudio no se encuentran diferencias significativas entre ambos algoritmos, ya que no se puede rechazar la hipótesis de igualdad.

El tercer estudio también considera los mismos métodos, pero en términos de tiempo de aprendizaje. En este caso, se vuelve a descartar la hipótesis de igualdad, pero se acepta la hipótesis de que CHI^{BD}_{Local} requiere bastante más tiempo de aprendizaje.

En los tres últimos estudios, *QChi* se compara con CHI^{BD}_{Global}. En relación con la capacidad predictiva (tanto con la media geométrica como con el AUC), la prueba acepta que ambos tienen la misma capacidad y, por tanto, no se encuentran diferencias significativas entre ellos. Aunque en el caso de la medida AUC el balance de ganancias/pérdidas es muy favorable a CHI^{BD}_{Global}, el p-valor devuelto por el test indica que las diferencias encontradas entre los algoritmos no son suficientes para descartar la hipótesis de igualdad.

Este hecho demuestra nuevamente que realizar un cálculo exhaustivo y exacto de los pesos no garantiza una reducción del error en comparación con la propuesta realizada aquí de una estimación de peso computacionalmente más económica.

Por último, y en relación con el tiempo de aprendizaje, las pruebas muestran que CHI^{BD}_{Global} requiere significativamente más tiempo de aprendizaje. Como ejemplo, se puede indicar que en el tiempo que CHI^{BD}_{Global} necesitó para abordar el problema *higgs* utilizando un clúster de 32 mappers, *QChi* utilizando un solo ordenador podría haber resuelto el problema *higgs* con 400 millones de ejemplos.

En resumen, tanto en la primera serie de experimentos con conjuntos de datos convencionalmente empleados en el campo del aprendizaje automático, como en la segunda serie de experimentos que involucraron conjuntos de datos masivos, se

Estudio	CHI^{BD}_{Local} vs QChi	W/T/L	p-valor
1	Test GM	4/0/16	0,00679
2	Test AUC	9/0/11	0,64766
3	Tiempo Aprendizaje	20/0/0	0,00010

Estudio	CHI^{BD}_{Global} vs QChi	W/T/L	p-valor
4	Test GM	10/0/10	0,89603
5	Test AUC	14/0/6	0,15365
6	Tiempo Aprendizaje	16/0/4	0,01687

Tab. 5.8.: Pruebas estadísticas que comparan los resultados obtenidos por *QChi*, CHI^{BD}_{Global} y CHI^{BD}_{Local}

evidenció una notable disminución en el tiempo de aprendizaje al hacer uso de la técnica denominada *QChi*.

5.3.3. Reflexiones

En este estudio, se ha presentado una alternativa que mantiene una complejidad computacional de orden lineal. Esta alternativa se basa en la estimación de los valores de peso en lugar de calcularlos de forma exacta. Esta estimación se lleva a cabo considerando solo los ejemplos que tienen la regla en cuestión como la regla con mejor adaptación. Además, se ha propuesto una reestructuración algorítmica del algoritmo de *WM-C* con el objetivo de optimizar todo el proceso de cálculo de esta estimación. En la fase experimental, se han realizado comparaciones exhaustivas entre los resultados obtenidos por *QChi* y el algoritmo original *WM-C* en conjuntos de datos que no son masivos. Los resultados evidencian que la nueva propuesta logra una reducción significativa en el tiempo de aprendizaje y que el cálculo de los pesos utilizando valores exactos no necesariamente se traduce en una mejora en la capacidad de clasificación.

Adicionalmente, se han efectuado comparaciones entre los resultados de *QChi* y los de algunos algoritmos que emplean el esquema MapReduce en conjuntos de datos masivos. En este contexto, también se ha demostrado que el tiempo de aprendizaje es considerablemente menor al mismo tiempo que se mantiene la capacidad predictiva.

5.4. Contribuciones

- *QChi: a faster classification algorithm based on Wang-Mendel algorithm. (Sometido a tercera revisión)*

6 Uso de reglas redundantes en un sistema de clasificación

En el presente capítulo, finalmente abordamos el Objetivo Específico 3 de esta tesis doctoral. Se realiza un estudio detallado sobre cómo el añadir redundancia en el algoritmo de *QChi* (Capitulo 5), mediante la inclusión de múltiples reglas para soportar cada ejemplo, afecta positivamente a la capacidad de clasificación. Para tal efecto se exploran tres posibles alternativas para generar dicha redundancia, y comparándolo con el algoritmo de Wang y Mendel (*WM-C*), se determinara cuál es la mejor propuesta.

Uno de los problemas principales que aqueja al algoritmo *WM-C* radica en la falta general de exhaustividad de la base de reglas resultante. Este inconveniente tiene repercusiones significativas en la capacidad de generalización del sistema resultante, generando dificultades para asignar una salida cuando la entrada no se ajusta a ninguna de las reglas generadas a partir de los datos. En otras palabras, cuando un ejemplo de entrada no activa ninguna de las reglas previamente establecidas, resulta imposible asignar una salida adecuada [3].

En este estudio se propone una modificación del enfoque básico del algoritmo *WM-C*. Mientras que el algoritmo original añade solo una regla por cada ejemplo del conjunto de entrenamiento, esta investigación propone una nueva estrategia que implica la generación de múltiples reglas difusas para cada ejemplo. El objetivo de esta modificación es mejorar la precisión del algoritmo al obtener reglas que recubren de manera más completa el espacio de las soluciones. Sin embargo, es importante destacar que esta modificación provoca un aumento del número total de reglas, lo que a su vez puede impactar tanto en el tiempo requerido para realizar el aprendizaje como la inferencia del modelo.

En el Capítulo 4, se introdujo una propuesta de modelo de inferencia eficiente como una alternativa a la implementación convencional del proceso de inferencia basado en reglas difusas. Esta innovación dio como resultado un tiempo de razonamiento independiente del número de reglas presentes en el sistema. La propuesta que se presenta en este capítulo aprovecha esta ventaja. Esto implica que la complejidad adicional que podría surgir debido a un alto número de reglas no será un obstáculo,

lo que lo convierte en una opción prometedora para mejorar el algoritmo original en los escenarios que serán analizados en la fase experimental.

Además, como ya hemos mencionado en el capítulo 5, otro aspecto que representa un cuello de botella en el algoritmo de *WM-C*, es el cálculo del peso de la regla. Para abordar esta limitación, empleamos el modelo *QChi*, que permite una mejora significativa en la eficiencia del cálculo del peso, permitiendo así una ejecución eficiente incluso frente a un gran número de ejemplos.

Basándose en la idea inicialmente expuesta acerca de la redundancia en el algoritmo de *WM-C*, junto con la implementación del método de inferencia eficiente y el uso de *QChi*, se plantea desarrollar una nueva técnica denominada QCHI-Multi-Rules (*QCHI-MR*), con la que se espera lograr una mejora significativa en la precisión, manteniendo en tiempos muy razonables la eficiencia del algoritmo de aprendizaje.

6.1. Preliminares

La propuesta de integrar reglas redundantes en el modelo de aprendizaje se basa en seleccionar reglas en el vecindario de la regla con la mejor adaptación a cada ejemplo. Por lo tanto, primero definiremos el conjunto de reglas con adaptación positiva al ejemplo y luego el procedimiento para explorar este conjunto de reglas.

6.1.1. El conjunto de reglas con adaptación positiva a un ejemplo

El modelo de reglas que se utilizara es el definido en la Ecuación 2.11 que se expone nuevamente a continuación.

$$R : \textbf{IF } X_1 \text{ is } A^1_{j_1} \text{ and } X_2 \text{ is } A^2_{j_2} \text{ and } \ldots \text{and } X_n \text{ is } A^n_{j_n}$$
$$\textbf{THEN } Y \text{ is } B \text{ with } \textbf{weight } w(R)$$

En el Algoritmo 8 (*QChi*), en el paso 4 (*BestAdaptationRule*), se elige la regla que mejor se adapta al ejemplo. Concretamente, se podría definir un conjunto que engloba todas las reglas con adaptación positiva a ese ejemplo. La regla seleccionada por el algoritmo proviene de este conjunto, aunque es importante destacar que se podría asignar cualquier otra regla del mismo conjunto al ejemplo.

En el Capítulo 4 se definió el concepto de vecindario de un ejemplo. La misma idea se utilizará ahora para definir el conjunto de reglas con adaptación positiva al ejemplo que serían las candidatas para ser incluidas por el nuevo algoritmo. Sin embargo es necesario modificar el concepto de vecindario entonces definido para incluir reglas completas (en el Capítulo 4 en el vecindario tan solo se consideraba el antecedente de las reglas). Para ello ahora definimos un nuevo concepto de vecindario que incluye todas las reglas difusas R con adaptación positiva al ejemplo e del conjunto de entrenamiento E. La representación de este conjunto es la siguiente:

$$\overline{N}(e) = \{R \in R^G \mid \overline{A}(R, e) > 0\} \tag{6.1}$$

donde R^G es el conjunto de todas las posibles reglas que se pueden generar con las n variables antecedentes y

$$\overline{A}(R, e) = \begin{cases} A(R, e) & \text{si} \quad clase(e) = B \\ \\ 0 & \text{en caso contrario.} \end{cases} \tag{6.2}$$

en donde $A(R, e)$ es la adaptación entre el antecedente de la regla R y el ejemplo e, definido en la Ecuación 2.13 del Capítulo 2, y clase(e) es la clase asignada al ejemplo e del conjunto de entrenamiento.

Es posible establecer una relación de orden en $\overline{N}(e)$ utilizando la función $\overline{A}(R, e)$ de la siguiente manera:

$$R_1 \leq R_2 \quad \text{si y solo si} \quad \overline{A}(R_1, e) \leq \overline{A}(R_2, e). \tag{6.3}$$

La regla seleccionada en cada ejemplo e tanto por el algoritmo *WM-C* como por *QChi* corresponde a la regla que maximiza el valor utilizando la relación previamente definida. Esta regla es denominada la "regla central" del ejemplo y se representa como $C(e)$, siguiendo la notación establecida en el Capítulo 5. Sin embargo, como se mencionó previamente, el conjunto $\overline{N}(e)$ contiene múltiples reglas, y el objetivo de esta investigación es determinar si la capacidad de generalización del algoritmo podría mejorar asignando a cada ejemplo e más de una regla del conjunto $\overline{N}(e)$.

Además, dado que parece más conveniente asignar a cada ejemplo e las mejores reglas en algún sentido que debemos definir, la relación de orden descrita anteriormente debe ser un aspecto importante a tener en cuenta al asignar más de una regla

a cada ejemplo. Sin embargo, es importante destacar que ordenar el conjunto $\overline{N}(e)$ no es una tarea sencilla.

6.1.2. Exploración del vecindario de un ejemplo

En el problema abordado, se manejan n variables, en donde supondremos que todas las variables continuas se discretizan mediante un conjunto de etiquetas difusas. Además, se supone que los dominios han sido definidos de manera que la adaptación de cada valor real con estas etiquetas puede tener como máximo adaptación positiva con dos etiquetas. Por lo tanto, el cardinal del conjunto $\overline{N}(e)$ sería como máximo 2^n. Sin embargo, dado que el tamaño de este conjunto podría ser considerablemente grande para algunos valores de n, se vuelve esencial realizar la exploración de $\overline{N}(e)$ de manera eficiente.

La exploración empleada se fundamenta en el proceso de inferencia recursiva de las reglas difusas presentado en el Capítulo 4. En términos generales, se adopta un modelo de búsqueda de reglas que implica asignar una variable antecedente y un valor para dicha variable en cada iteración. Una vez definido un orden en la selección de variables y sus valores, el procedimiento se convierte en una búsqueda recursiva en profundidad.

Este proceso se detalla a continuación. Partiendo de un ejemplo e, es evidente que todas las reglas en el conjunto $\overline{N}(e)$ comparten la misma clase que el ejemplo, por lo que establecemos esta clase como el valor para todas las posibles reglas del conjunto. A continuación, se procede a buscar los antecedentes de estas reglas. Se toma la primera de las variables antecedente, según el criterio de selección establecido para las variables. Si esta variable es discreta, se le asigna directamente el valor que posee en el ejemplo y esta variable queda excluida del proceso de búsqueda. En el caso de que la variable sea continua, se elige inicialmente una de las etiquetas con adaptación positiva, siguiendo los criterios establecidos para los valores. Posteriormente, se selecciona la siguiente variable y se repite el proceso de manera análoga. En situaciones de retroceso, se procede a seleccionar otra etiqueta.

El proceso de búsqueda empleado se caracteriza por ser completo, lo que significa que explora exhaustivamente todas las reglas contenidas en el conjunto $\overline{N}(e)$. Sin embargo, el objetivo principal radica en obtener un ordenamiento efectivo de dicho conjunto, basado en el criterio de adaptación $\overline{A}(R, e)$. Para lograr este propósito, se hace necesario implementar heurísticas que guíen la selección de variables y valores en cada etapa del proceso de búsqueda.

En cuanto al criterio empleado para la selección de valores, la heurística es sencilla. Las etiquetas se escogen siguiendo un orden descendente basado en la adaptación de cada etiqueta con respecto al componente del ejemplo. En otras palabras, se comienza seleccionando la etiqueta que presenta la mejor adaptación con el componente del ejemplo, seguida por la etiqueta que ocupa el segundo lugar en adaptación, y así sucesivamente.

Por ende, suponiendo que para un ejemplo dado e y una variable antecedente X_i, $L_1^i(e)$ representa una de las etiquetas con adaptación positiva para el i-ésimo componente de e, y $L_2^i(e)$ denota la otra etiqueta con adaptación positiva, procedemos de la siguiente manera: si $L_2^i(e) \leq L_1^i(e)$, seleccionamos inicialmente la etiqueta $L_1^i(e)$ como el primer valor a explorar, mientras que $L_2^i(e)$ se examinará en un paso posterior.

En cuanto al criterio seleccionado para las variables, la heurística se basa en la siguiente función.

$$h(i) = max_{i=1,\ldots,n}\{\overline{A}(e, L_1^i(e)) - \overline{A}(e, L_2^i(e))\} \tag{6.4}$$

donde i es el índice de la variable X_i, $L_1^i(e)$ y $L_2^i(e)$ son las etiquetas del dominio de la variable i-ésima con adaptación positiva con el i-ésimo componente del ejemplo y $L_2^i(e) \leq L_1^i(e)$.

La función h permite ordenar las variables en el proceso de búsqueda recursiva en profundidad definido anteriormente, de modo que la variable con el valor más alto de h se selecciona en primer lugar.

La idea básica de esta heurística es que cuanto mayor sea la diferencia entre la adaptación con la etiqueta que tiene la mejor adaptación y la etiqueta con la peor adaptación, mayor será el valor que esta variable aporta a la adaptación conjunta con el antecedente de la regla. De hecho, la primera regla explorada que utiliza estas dos heurísticas y la búsqueda recursiva en profundidad es la regla central del ejemplo $C(e)$.

6.2. Métodos Propuestos: El algoritmo *QCHI-MR*

Por lo tanto, la idea básica de nuestra propuesta es definir un nuevo algoritmo que hemos llamado *QCHI-MR* (QCHI-Multi-Rules), inspirado en el algoritmo *WM-C* pero utilizando el modelo para estimar el peso de las reglas *QChi* expuesto en el Capítulo

5 y tomando más de una regla (Multi-Rule o MR) asociada a cada ejemplo y no solo la regla central.

Algoritmo 9 Algoritmo QCHI-MR

```
1: function QCHI-MR(E, D, Rules)
2:     AntSet = {∅}, Rules = { ∅}
3:     for e ∈ E do
4:         Ants = BetterAdaptAntedecents (e, D, Crt))
5:         for a ∈ Ants do
6:             UpdateAntSet (a, AntSet)
7:         end for
8:     end for
9:     for a ∈ AntSet do
10:        r = BuildRule(a, w)
11:        if (w > 0) then
12:            AddToRuleSet(r, w, Rules)
13:        end if
14:    end for
15:    return Rules
16: end function
```

Si comparamos el algoritmo *QCHI-MR* (Algoritmo 9) con *QChi* (Algoritmo 8), podemos observar que ambos contienen dos ciclos que componen la estructura de todo el proceso, siendo el segundo ciclo (líneas 9 a 14) exactamente igual en ambos casos. Por lo tanto, la segunda parte del proceso no cambia entre los dos algoritmos. En el primer ciclo (líneas 3 a 8), la diferencia radica en la función *BestAdaptAntecedent*, que ahora ha sido reemplazada por *BetterAdaptAntecedents*. Esta segunda función, dado un ejemplo e, devuelve un conjunto de antecedentes $Ants$ (no solo el de la regla central) siguiendo un criterio determinado (que denominamos Crt en la descripción del algoritmo) para seleccionar dicho conjunto. Ahora, cada antecedente de $Ants$ se utiliza para actualizar el conjunto *AntSet* mediante la función *UpdateAntSet* (líneas 5 a 7).

La descripción anterior no especifica qué reglas se asociarían con cada ejemplo, de hecho, existen muchas formas de seleccionar esas reglas, por lo tanto, se añade un criterio (denominado Crt en el algoritmo) para realizar esa selección en *BetterAdaptAntecedents*. En primer lugar, asignar todas las reglas al conjunto $\overline{N}(e)$ probablemente no sea conveniente por razones de eficiencia, por lo que es necesario seleccionar algunas de estas reglas. A continuación, analizaremos diferentes

propuestas del algoritmo *QCHI-MR* según el criterio elegido para seleccionar esas reglas.

Los criterios establecidos en este contexto contemplan, como mínimo, la inclusión del antecedente de la regla central de cada ejemplo en el conjunto de entrenamiento, según se detalla en el paso 4 del Algoritmo 9. Los criterios Crt que se han tomado en cuenta son los siguientes:

1. *Seleccionar un número fijo k entre los mejores antecedentes*. La primera propuesta consiste en seleccionar las mejores reglas del conjunto $\overline{N}(e)$. Dado que el número de reglas contenidas en este conjunto puede ser muy grande, establecemos un parámetro k que determina la cantidad de reglas a seleccionar. Utilizamos el proceso de exploración de reglas del conjunto $\overline{N}(e)$ descrito en la sección anterior, que como se mencionó anteriormente, proporciona una buena aproximación del orden descrito por la función de adaptación entre la regla y el ejemplo, y detenemos la selección al elegir la $k - esima$ regla.

2. *Seleccionar todos los antecedentes con una adaptación mayor o igual a un umbral t*. Las reglas del conjunto $\overline{N}(e)$ se exploran siguiendo el mismo procedimiento que se mencionó anteriormente, pero ahora seleccionamos todas las reglas que verifican la siguiente expresión:

$$\overline{A}(R, e) \geq t \quad \text{con } t \in (0, 1] \tag{6.5}$$

   donde t es un parámetro que representa el umbral a partir del cual seleccionaremos las reglas. El parámetro t codifica el porcentaje mínimo de adaptación que un antecedente debe tener en relación con el ejemplo, al compararlo con el valor absoluto de adaptación que alcanza el antecedente de la regla central en el mismo ejemplo. Así, el valor de adaptación de la regla central se toma como el valor de referencia que el resto de los antecedentes deben superar en un t por ciento para considerarse dentro de este criterio.

3. *Seleccionar todos los antecedentes con una distancia de Hamming [42] menor o igual a d con respecto al antecedente de la regla central*. Una vez más, se recurre al proceso de exploración previamente descrito, siguiendo el orden de las reglas propuesto, donde se seleccionan aquellas reglas que guardan similitud con la regla central con respecto al ejemplo en cuestión. Cabe destacar que, en este proceso, la primera regla a explorar siempre es la regla central. El criterio de similitud entre reglas se define mediante una medida de distancia, conforme a lo expuesto en la Ecuación 4.13.

En el caso de reglas con el mismo consecuente, la medida de distancia se define como el número de variables que presentan una asignación de etiquetas diferente en el antecedente de las reglas comparadas. En este contexto, una distancia de 0 implica que únicamente se considera la regla central en relación con el ejemplo, lo que equivale al algoritmo *WM-C*. Si se establece una distancia de 1, se incluyen aquellas reglas que difieren en una variable antecedente como máximo, y si la distancia es de 2, las reglas pueden diferir en dos variables antecedentes como máximo, y así sucesivamente. Por lo tanto, esta propuesta depende de un parámetro d que determina la distancia máxima a considerar entre las reglas

En resumen, se han presentado múltiples propuestas para definir el algoritmo *QCHI-MR*, y en cada una de ellas se dispone de un parámetro que debe ser establecido. En la siguiente sección, se estudian experimentalmente las diferentes propuestas presentadas anteriormente para la inclusión de reglas redundantes, así como la selección de los parámetros más adecuados en cada caso.

6.3. Análisis e interpretación de los resultados obtenidos

El objetivo consiste en mostrar si alguna de las tres propuestas consideradas de *QCHI-MR* consigue mejorar la capacidad de predicción en comparación con los resultados obtenidos por *WM-C* o el propio *QChi*. Además, también estimar si alguna de las tres propuestas se puede considerar superior a las otras dos.

En la primera parte de este estudio, se utilizarán 30 conjuntos de datos para la clasificación, los cuales se describen en detalle en la Tabla del Anexo A.1, columna (*MR*). Estos conjuntos de datos son extraídos del repositorio de la UCI [71]. En la segunda parte de la experimentación, se emplearán conjuntos de datos de mayor dimensión que presentan desafíos significativos para los algoritmos convencionales de aprendizaje automático. Estos conjuntos de datos se detallan en la Tabla del Anexo A.2.

Todos los experimentos se llevaron a cabo en una computadora con un procesador Intel Core i7-9750H CPU @ 2.60GHz y 16 GB de memoria, ejecutando el sistema operativo Ubuntu 20.04.3 LTS x86 64. Los resultados sobre los parámetros estudiados, que se mostrarán en tablas, se obtuvieron utilizando una validación cruzada de 10 particiones.

Además, para este estudio experimental se emplearán dominios formados por 5 conjuntos difusos triangulares uniformemente distribuidos y que se cortan en 0,5 para codificar las variables continuas.

Dado que las diferentes propuestas solucionan la falta de generalización del algoritmo original al agregar reglas, es normal que en muchos casos se obtengan un gran número de reglas, lo cual requiere un mecanismo eficiente para realizar la inferencia. Como anteriormente hemos mencionado, se utiliza la inferencia de reglas difusas propuesta en el Capítulo 4 (ver Algoritmo 4), que tiene una complejidad algorítmica que no depende del número de reglas y para el cálculo del peso se utiliza el modelo propuesto en el Capítulo 5 (ver Algoritmo 8). Este hecho hace que el tiempo de respuesta sea muy rápido en comparación con el ofrecido por la implementación habitual de la inferencia y del cálculo del peso de cada regla.

En esta experimentación tomaremos los resultados obtenidos por el algoritmo de *WM-C* tanto medios como sobre cada una de las bases de datos seleccionadas como referencia para estudiar el cambio que se produce en las distintas propuestas para incluir redundancia.

La Tabla 6.1 muestra los resultados obtenidos en la primera parte de este estudio por el algoritmo de *WM-C*, donde (*Test*) indica la precisión teniendo en cuenta todos los ejemplos en el conjunto de prueba, (*Tiempo*) indica el tiempo de ejecución necesario para el aprendizaje (en segundos), (*Infer*) es el tiempo de inferencia en el conjunto de prueba (en segundos), (*%NC*) indica el porcentaje de ejemplos no cubiertos encontrados durante el proceso de inferencia en el conjunto de prueba y (*Test-NC*) es la precisión al eliminar del conjunto de prueba todos los ejemplos que no están cubiertos por ninguna regla.

A continuación, se estudia cada propuesta para incluir más de una regla para cada ejemplo y se estima el valor más apropiado del parámetro en cada una de ellas.

6.3.1. Criterio de selección: las mejores *k* reglas

Se comienza por estudiar el primer criterio y lo que podría parecer una extensión más natural para agregar más de una regla siguiendo la idea del algoritmo *WM-C*, que consiste en añadir las k mejores reglas con adaptación al ejemplo, donde k es un parámetro a establecer.

En este criterio es interesante conocer cómo se comporta el algoritmo a medida que aumenta el número de reglas, es decir, a medida que el parámetro k aumenta, y en

Conjunto de datos	Test	Tiempo	Infer	%NC	Test-NC
abalone	1,05	0,18	0,02	96,19	33,09
adult	64,38	157,23	0,08	21,37	81,88
australian	54,64	0,07	0,00	34,06	82,73
banana	80,36	0,05	0,00	0,02	80,37
bands	16,62	0,05	0,01	74,88	66,17
bupa	59,71	0,01	0,00	2,89	61,55
cleveland	18,54	0,02	0,00	71,68	65,52
coil2000	24,27	12,65	0,10	73,66	92,14
crx	41,17	0,08	0,00	52,37	86,54
iris	95,33	0,00	0,00	0,00	95,33
letter	74,60	39,00	0,36	0,78	75,19
magic	80,80	8,15	0,05	0,09	80,87
newthyroid	91,17	0,01	0,00	0,93	92,01
mammogr	76,74	0,03	0,00	7,48	82,91
page-blocks	93,90	0,26	0,01	0,20	94,09
penbased	94,31	12,24	0,09	4,96	99,22
phoneme	78,66	0,24	0,00	0,00	78,66
pima	70,59	0,10	0,00	3,26	72,99
ring	53,24	13,24	0,21	30,91	77,07
saheart	62,12	0,06	0,00	6,48	66,39
satimage	51,23	17,88	0,26	0,75	51,62
segment	88,48	0,46	0,02	3,33	91,54
shuttle	84,24	1,64	0,10	0,01	84,25
spambase	80,66	4,27	0,05	4,72	84,66
thyroid	92,10	1,35	0,01	1,35	93,35
twonorm	34,93	12,20	0,28	62,49	93,09
vehicle	54,49	0,22	0,02	18,80	67,10
vowel	93,03	0,15	0,00	0,30	93,31
wineq-red	59,60	0,34	0,02	1,44	60,48
wineq-white	53,08	1,90	0,05	0,25	53,21
Promedio	70,73	0,49	0,05	10,39	78,65

Tab. 6.1.: Resultados obtenidos con el algoritmo *WM-C* utilizando una validación cruzada de 10 particiones y un dominio uniformemente distribuido con 5 etiquetas difusas en todas las variables continuas, donde (*Test*) indica la precisión teniendo en cuenta todos los ejemplos en el conjunto de prueba, (*Tiempo*) indica el tiempo de ejecución necesario para el aprendizaje (en segundos), (*Infer*) es el tiempo de inferencia en el conjunto de prueba (en segundos), (*%NC*) indica el porcentaje de ejemplos no cubiertos encontrados durante el proceso de inferencia en el conjunto de prueba y (*Test-NC*) es la precisión al eliminar del conjunto de prueba todos los ejemplos que no están cubiertos por ninguna regla.

consecuencia si es posible estimar un valor para el parámetro k dado un problema de clasificación.

Si un problema tiene n variables continuas y se utiliza el método de discretización para estas variables, dado en esta parte experimental, entonces el número máximo de antecedentes para una regla que tiene adaptación con un ejemplo es 2^n. En esta experimentación, se estudia el comportamiento considerando las primeras 11 potencias de 2.

		k										
	WM-C	1	2	4	8	16	32	64	128	256	512	1024
Test	64,13	65,06	66,87	68,26	69,12	69,40	70,15	70,73	70,76	70,96	71,13	71,31
Tiempo	9,47	0,14	0,15	0,17	0,19	0,25	0,33	0,49	0,76	1,31	2,40	4,44
Infer	0,06	0,04	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,06	0,05	0,05
%NC	19,19	15,94	14,31	13,15	12,04	11,15	10,44	10,39	11,31	10,90	10,53	10,21
Test-NC	77,91	77,56	78,22	78,78	78,72	78,37	78,41	78,65	78,60	78,47	78,35	78,30

Tab. 6.2.: Promedio de valores obtenidos utilizando el algoritmo de *WM-C* y usando el criterio de selección las mejores k reglas

La Tabla 6.2 muestra los resultados obtenidos al tomar diferentes valores de k en el promedio de todos los conjuntos de datos analizados. Basándonos en los resultados obtenidos, se pueden extraer las siguientes conclusiones:

- La capacidad predictiva aumenta con el valor de k. Existe un crecimiento más pronunciado hasta $k = 64$, de más del 5.4 %, a partir de ahí hasta $k = 1024$ el crecimiento es más suave, aproximadamente del 0.6 %.
- El tiempo necesario para aprender el conjunto de reglas también tiene un crecimiento monótono con el valor de k, y también se pueden observar dos tasas de crecimiento diferentes en él. Una más suave hasta $k = 64$, donde el tiempo es veinte veces menor que el que tomaría el algoritmo *WM-C*, mientras que a partir de ese valor se observa un crecimiento mucho más pronunciado. En cualquier caso, para el valor más alto de $k = 1024$ considerado en este estudio, que representa el tiempo de aprendizaje más largo entre todos los k estudiados, su tiempo es en promedio aproximadamente la mitad del tiempo consumido por el algoritmo *WM-C*.
- Se utilizó la prueba de rangos con signo de Wilcoxon [96] con un intervalo de confianza del 95 %, para verificar si existen diferencias significativas entre la precisión que logramos para $k = 64$ y $k = 1024$. Observamos que no existen diferencias significativas con un valor de $p = 0{,}1918751$, lo que demuestra que $k = 64$ alcanza una alta precisión con menos tiempo de entrenamiento en comparación con $k = 1024$.

- El tiempo de inferencia muestra un comportamiento casi constante, con un ligero aumento de una décima de segundo en relación con el algoritmo *WM-C*. En este experimento, como ya hemos dicho se ha utilizado la versión de inferencia propuesta en el Algoritmo 4. Esta versión hace que el tiempo de inferencia no dependa directamente del tamaño de la base de reglas. Obviamente, utilizando la versión tradicional de inferencia, el tiempo aumentaría proporcionalmente al valor de k, ya que el número de reglas también aumenta en la misma proporción. También se utilizó la obtención del peso basado en *QChi* que se expuso en el Capítulo 5

- El porcentaje de ejemplos que no son cubiertos por ninguna regla disminuye a medida que aumenta el valor de k. Aunque no tiene un comportamiento estrictamente monótono, refleja claramente la tendencia de dicha reducción. Tiene un comportamiento monótono hasta $k = 64$, lo que implica una mejora en la capacidad de generalización del 5.4 %, después de lo cual la mejora es pequeña. Se puede observar que en $k = 64$, la mejora en la predicción coincide con la mejora en la generalización. Por lo tanto, mejorar la precisión está directamente relacionado con mejorar la generalización del conocimiento que implica incluir más reglas.

- Por último, la capacidad predictiva sobre los ejemplos que activan una regla se mantiene constante por debajo del 79 %. Este último parámetro muestra que, a pesar de incluir muchas más reglas en la base de conocimiento, no afecta, en general, la capacidad de predicción, aunque, como indicaba el parámetro anterior, sí mejora la generalización de dicho conocimiento. Además, se puede observar que el porcentaje de precisión calculado de esta manera está aproximadamente un 6 % por encima de la precisión del algoritmo de *WM-C*.

La Tabla 6.3 muestra los resultados obtenidos para cada uno de los conjuntos de datos tomando $k = 64$ como valor. Se puede observar una clara tendencia de mejora en la precisión general, una reducción en el porcentaje de ejemplos no cubiertos en la misma proporción que la mejora en la precisión, y un aumento en el tiempo de aprendizaje que no es excesivo.

	$k = 64$				
Conjunto de Datos	Test	Tiempo	Infer	%NC	Test-NC
abalone	16,75	0,20	0,02	17,80	20,36
adult	67,10	1,55	0,10	18,02	81,85
australian	67,68	0,04	0,00	19,13	83,62
banana	80,36	0,05	0,00	0,00	80,36
bands	36,71	0,03	0,01	45,03	66,43
bupa	61,94	0,01	0,00	0,61	62,34
cleveland	32,64	0,02	0,00	49,54	64,33
coil2000	24,30	0,94	0,09	73,63	92,15
crx	52,17	0,02	0,00	39,24	85,82
iris	95,33	0,00	0,00	0,00	95,33
letter	84,78	2,21	0,18	0,01	84,78
magic	81,69	0,89	0,05	0,02	81,70
newthyroid	92,55	0,00	0,00	0,48	93,01
mammogr	78,06	0,01	0,00	5,45	82,54
page-blocks	94,08	0,21	0,01	0,05	94,13
penbased	97,69	1,18	0,08	1,37	99,05
phoneme	78,72	0,14	0,00	0,00	78,72
pima	72,80	0,03	0,00	1,17	73,68
ring	65,99	0,80	0,22	12,41	75,32
saheart	66,45	0,03	0,00	1,52	67,44
satimage	66,70	0,78	0,24	0,08	66,75
segment	91,99	0,16	0,02	1,56	93,45
shuttle	85,47	2,67	0,11	0,01	85,48
spambase	85,25	0,63	0,06	2,02	87,01
texture	93,55	0,90	0,21	1,76	95,22
thyroid	92,46	0,42	0,02	0,93	93,33
twonorm	76,27	1,01	0,30	16,89	91,77
vehicle	62,06	0,10	0,02	4,03	64,66
vowel	96,26	0,09	0,00	0,00	96,26
wineq-red	62,73	0,09	0,01	0,56	63,09
wineq-white	54,80	0,28	0,04	0,12	54,87
Promedio	70,73	0,49	0,05	10,39	78,65

Tab. 6.3.: Resultados obtenidos utilizando el criterio de las mejores k reglas con $k = 64$, utilizando una validación cruzada de 10 particiones. La columna (Test) es la precisión en el conjunto de prueba, (Tiempo) es el número de segundos consumidos por el algoritmo de aprendizaje, (Infer) corresponde al número de segundos consumidos por el proceso de inferencia, (%NC) es el porcentaje de ejemplos no cubiertos y (Test-NC) es la precisión al eliminar del conjunto de prueba todos los ejemplos que no están cubiertos por ninguna regla.

6.3.2. Criterio de selección: reglas con un grado de adaptación superior a un umbral *t*

El segundo criterio que se estudia consiste en seleccionar para cada ejemplo todas aquellas reglas cuyos antecedentes tengan una adaptación con el ejemplo mayor o igual a t, donde $t \in (0, 1]$. El parámetro t establece el valor mínimo de adaptación que una regla debe tener para considerarse relevante y potencialmente incluida en el conjunto final de reglas. Cabe destacar que t no es un valor absoluto de adaptación, sino relativo a la adaptación en comparación con el antecedente de la regla que tiene la adaptación máxima con ese ejemplo. De esta manera, el proceso de selección tiene un paso inicial en el que se calcula el grado de adaptación de la regla central y se establece como valor de referencia. En el resto del proceso de selección de reglas, se buscan los antecedentes cuyo valor de adaptación supere ese t sobre el valor de referencia. Evidentemente, valores cercanos a 1 establecen valores estrictos y hace que el número de reglas a incluir sea pequeño, mientras valores cercanos al 0 implica la inclusión de la mayoría de las reglas con algo de adaptación con el ejemplo.

En la Tabla 6.4 se muestra un estudio en el que se recogen los valores promedio utilizando todos los conjuntos de datos indicados en la Tabla del Anexo A.1, columna (MR), para diferentes valores de t. Esta tabla refleja los valores obtenidos para los mismos parámetros ya indicados en la Tabla 6.2 y se pueden obtener las siguientes conclusiones:

		t								
	WM-C	1	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2
Test	64,13	65,06	65,85	66,63	67,38	68,17	68,81	69,65	70,42	71,23
Tiempo	9,47	0,15	0,16	0,19	0,25	0,37	0,59	1,13	2,54	7,59
Infer	0,06	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,06	0,06
%NC	19,19	15,94	15,07	14,17	13,33	12,62	11,79	10,92	10,20	9,43
Test-NC	77,91	77,53	77,66	77,81	77,96	78,21	78,23	78,29	78,52	78,78

Tab. 6.4.: Promedio de valores obtenidos utilizando el algoritmo de *WM-C* y usando el criterio de umbral t.

- La tendencia observada al reducir el valor de t es similar a la producida al aumentar el parámetro k en el criterio anterior para todos los casos estudiados, es decir, un aumento en la capacidad de clasificación general, una reducción en el porcentaje de ejemplos no cubiertos y un aumento en el tiempo requerido para el aprendizaje. Los parámetros de tiempo de inferencia y precisión en los ejemplos cubiertos por al menos una regla también se mantienen más o menos constantes.

- En un análisis más detallado, se puede observar que la monotonicidad en el crecimiento de la precisión general y en la reducción del porcentaje de ejemplos no cubiertos es estricta, por lo que se obtienen los mejores valores con el valor $t = 0{,}2$.

- Centrándonos en el valor de $t = 0{,}2$ en comparación con los resultados globales obtenidos con $k = 64$, podemos observar que son bastante similares, destacando que la versión con umbral logra reducir en más de medio punto el porcentaje de ejemplos no cubiertos, aunque el tiempo de aprendizaje es 4 veces mayor. Este último hecho no es demasiado significativo, ya que los tiempos de aprendizaje utilizando este algoritmo son pequeños.

Una de las conclusiones del análisis anterior es que los mejores resultados combinados de este criterio se obtienen para el valor $t = 0{,}2$. La Tabla 6.5 muestra en detalle los resultados de los 5 parámetros que se han estado estudiando en todos los conjuntos de datos analizados. Se puede observar que el conjunto de datos $Satimage$ tiene un tiempo de aprendizaje de 205,83 segundos, un resultado mucho más alto que el resto de los conjuntos de datos, por lo que el promedio general se ve afectado. También podemos observar una tendencia al aumento del tiempo de aprendizaje para aquellos conjuntos de datos con más de 10 variables continuas, esto se debe a la explosión combinatoria que se genera en el número de reglas al aumentar estas variables.

6.3.3. Criterio de selección: reglas con una medida de distancia inferior o igual al valor *d*

El último de los criterios contemplados en este estudio consiste en considerar como reglas adicionales todas aquellas que cumplan con estar cerca de la regla central en relación con su morfología, sin tener en cuenta su grado de adaptación. Para ello, dado un ejemplo, calculamos la regla central. El antecedente de la regla central se tomará como referencia. A partir de ella, podemos considerar que una regla está a una distancia d de la regla central, si exactamente d variables del antecedente toman un valor diferente al tomado por esa variable en la regla central. Naturalmente, los demás valores difusos (etiquetas) que puedan tomar las reglas consideradas deben tener una adaptación mayor que cero para el ejemplo.

Obviamente, el conjunto de reglas cubiertas por este criterio contiene muchas de las reglas ya cubiertas por los criterios anteriores, pero ahora también incluye reglas con bajos grados de adaptación (y por lo tanto, no se encuentran entre las mejores k

	$t = 0{,}2$				
Conjunto de Datos	Test	Tiempo	Infer	%NC	Test-NC
abalone	20,72	0,12	0,01	0,05	20,73
adult	66,78	1,47	0,09	18,45	81,89
australian	62,75	0,01	0,00	24,64	83,26
banana	80,89	0,04	0,00	0,00	80,89
bands	43,14	0,09	0,01	32,34	64,35
bupa	61,46	0,01	0,00	0,89	61,97
cleveland	28,89	0,02	0,00	55,27	64,57
coil2000	24,31	1,25	0,11	73,63	92,19
crx	47,93	0,01	0,00	43,94	85,45
iris	95,33	0,00	0,00	0,00	95,33
letter	86,19	4,10	0,23	0,01	86,19
magic	82,41	0,83	0,05	0,03	82,43
newthyroid	91,19	0,00	0,00	0,93	92,05
mammogr	77,57	0,01	0,00	5,94	82,46
page-blocks	94,72	0,08	0,00	0,07	94,79
penbased	98,01	1,40	0,07	1,13	99,13
phoneme	80,12	0,08	0,00	0,00	80,12
pima	70,97	0,01	0,00	1,44	72,02
ring	73,00	2,69	0,22	8,12	79,46
saheart	61,23	0,01	0,00	2,17	62,58
satimage	72,73	205,83	0,53	0,00	72,73
segment	92,12	0,35	0,02	1,30	93,34
shuttle	84,38	1,80	0,12	0,01	84,39
spambase	86,14	1,30	0,04	2,22	88,10
thyroid	92,49	0,20	0,02	1,08	93,50
twonorm	85,82	5,02	0,24	6,78	92,07
vehicle	65,11	0,40	0,02	2,02	66,42
vowel	96,36	0,08	0,00	0,00	96,36
wineq-red	60,29	0,13	0,01	0,44	60,56
wineq-white	53,90	0,31	0,03	0,14	53,98
Promedio	71,23	7,59	0,06	9,43	78,78

Tab. 6.5.: Resultados obtenidos utilizando el criterio de umbral t, con $t = 0,2$, utilizando una validación cruzada de 10 particiones. La columna (Test) es la precisión en el conjunto de prueba, (Tiempo) es el número de segundos consumidos por el algoritmo de aprendizaje, (Infer) corresponde al número de segundos consumidos por el proceso de inferencia, (%NC) es el porcentaje de ejemplos no cubiertos y (Test-NC) es la precisión al eliminar del conjunto de prueba todos los ejemplos que no están cubiertos por ninguna regla.

ni superan el umbral t), que se considerarán debido a que su morfología las hace estar cerca del antecedente con la mejor adaptación.

La Tabla 6.6 muestra el promedio en todos los conjuntos de datos indicados en la Tabla del Anexo A.1, columna (*MR*), para diferentes valores de d y los diferentes parámetros estudiados, y se puede concluir lo siguiente:

- Al igual que con el aumento de k y la reducción de t, al aumentar d se observa una mejora en la precisión general. Podemos observar en la Tabla 6.6 que el aumento entre $d = 0$ y $d = 1$ es del 4.78 %, entre $d = 1$ y $d = 2$ es del 1.89 % y finalmente entre $d = 2$ y $d = 3$ es del 0.61 %. Podemos ver que en la última comparación no hay diferencias significativas en el aumento de la precisión. Esto podría indicar que con distancias mayores a 2 se incluyen un número excesivo de reglas que provocan una pérdida de capacidad de predicción. No debemos olvidar que aquí se incluyen reglas que son cercanas en su morfología pero que pueden tener valores de adaptación muy bajos.

- Se utilizó la prueba de rangos con signo de Wilcoxon [96] con un intervalo de confianza del 95 %, para verificar si existen diferencias significativas entre la precisión para $d = 2$ y $d = 3$. Observamos que no hay diferencias significativas con un valor de $p = 0{,}06061728$, lo que demuestra que $d = 2$ logra una mejora significativa en la precisión, con un tiempo de entrenamiento menor en comparación con $d = 3$.

- La tendencia en el resto de los parámetros estudiados es similar a la ofrecida por los otros criterios: el porcentaje de ejemplos que no son activados por ninguna regla disminuye a medida que aumenta el valor de d y el tiempo requerido para el aprendizaje aumenta exponencialmente con d, mientras que el tiempo de inferencia y el porcentaje de precisión en los ejemplos cubiertos por alguna regla se mantienen más o menos constantes.

		d			
	WM-C	0	1	2	3
Test	64,13	65,10	69,88	71,77	72,38
Tiempo	9,47	0,15	0,36	1,53	9,01
Infer	0,06	0,05	0,05	0,05	0,05
%NC	19,19	15,98	10,44	8,55	7,89
Test-NC	77,91	77,82	78,21	78,65	78,77

Tab. 6.6.: Promedio de valores obtenidos utilizando el algoritmo de *WM-C* y usando el criterio de distancia d.

- Si se examinan detenidamente los mejores valores obtenidos por este criterio para los parámetros estudiados y se comparan con los resultados obtenidos por los criterios anteriores, se puede ver que se obtienen valores superiores al 70 % en el parámetro de test (un valor que solo fue superado por el primer criterio con $k = 1024$) y también se reduce el porcentaje de ejemplos no cubiertos por ninguna regla a un valor por debajo del 12 % (tampoco alcanzado por ningún valor considerado en los criterios anteriores). Sin embargo, la desventaja de estos resultados es que el tiempo de aprendizaje es significativamente mayor, especialmente para valores de $d > 2$.
- Aunque los valores obtenidos con $d = 2$ no sean los mejores en todos los parámetros estudiados, para este valor del parámetro d se obtienen valores de precisión y reducción de ejemplos no cubiertos por ninguna regla que mejoran los criterios anteriores y requieren un tiempo de aprendizaje que puede considerarse razonable.

En la Tabla 6.7 se muestran los resultados obtenidos para cada conjunto de datos utilizando el criterio de reglas con una distancia $d = 2$. Como se observa en la Tabla 6.5, se puede ver que en el conjunto de datos $Satimage$ hay un tiempo de aprendizaje alto de 17.17 segundos y un aumento general en el tiempo de aprendizaje para todos los conjuntos de datos con más de 10 variables continuas, al igual que con el método $t = 0,2$.

De esta forma se llevó a cabo un estudio que implicó la exposición de las ventajas y desventajas de cada método analizado. En la siguiente subsección analizaremos el impacto de la precisión de los métodos propuestos, sobre los conjuntos de datos con un alto porcentaje de ejemplos no cubiertos.

6.3.4. Estudio sobre problemas con un alto porcentaje de ejemplos no cubiertos

Al analizar la Tabla 6.1, se observa que hay situaciones para las cuales el algoritmo *WM-C* presenta problemas de generalización, ya que existe un número muy alto de ejemplos no cubiertos. En esta sección, se busca evaluar el desempeño de las diferentes propuestas presentadas aquí en comparación con el algoritmo *WM-C* basándose en el porcentaje de ejemplos no cubiertos en los conjuntos de test.

En la Tablas 6.8 y 6.9, se divide el estudio entre aquellos conjuntos de datos que, según el algoritmo *WM-C*, obtienen un porcentaje de ejemplos no cubiertos (%NC) inferior al 4 %, y aquellas que obtienen un resultado superior al 4 %, respectivamente.

	$d = 2$				
Conjunto de Datos	Test	Tiempo	Infer	%NC	Test-NC
abalone	16,94	0,13	0,02	0,05	16,95
adult	67,06	1,73	0,09	18,02	81,81
australian	67,25	0,03	0,00	19,57	83,56
banana	80,36	0,03	0,00	0,00	80,36
bands	49,30	0,12	0,01	25,67	66,36
bupa	63,99	0,01	0,00	0,61	64,39
cleveland	29,94	0,02	0,00	53,23	64,37
coil2000	24,30	1,25	0,10	73,63	92,15
crx	52,33	0,03	0,00	39,24	86,07
iris	96,00	0,00	0,00	0,00	96,00
letter	83,98	5,82	0,20	0,00	83,98
magic	82,34	1,15	0,05	0,01	82,34
newthyroid	92,10	0,00	0,00	0,48	92,55
mammogr	78,06	0,01	0,00	5,45	82,54
page-blocks	94,26	0,38	0,01	0,05	94,31
penbased	98,48	2,10	0,08	0,67	99,15
phoneme	79,09	0,11	0,00	0,00	79,09
pima	72,67	0,02	0,00	1,04	73,44
ring	70,84	3,87	0,18	6,12	75,46
saheart	64,73	0,03	0,00	1,09	65,46
satimage	66,04	17,17	0,27	0,00	66,04
segment	92,03	0,59	0,02	1,13	93,08
shuttle	89,87	2,83	0,13	0,01	89,87
spambase	85,62	3,07	0,06	1,72	87,12
thyroid	92,61	0,32	0,02	0,93	93,48
twonorm	86,81	4,15	0,26	5,99	92,34
vehicle	65,72	0,31	0,02	1,30	66,57
vowel	96,06	0,12	0,00	0,00	96,06
wineq-red	60,48	0,16	0,01	0,31	60,66
wineq-white	53,92	0,41	0,03	0,08	53,97
Promedio	71,77	1,53	0,05	8,55	78,65

Tab. 6.7.: Resultados obtenidos utilizando el criterio de distancia d, con $d = 2$, utilizando una validación cruzada de 10 particiones. La columna (Test) es la precisión en el conjunto de prueba, (Tiempo) es el número de segundos consumidos por el algoritmo de aprendizaje, (Infer) corresponde al número de segundos consumidos por el proceso de inferencia, (%NC) es el porcentaje de ejemplos no cubiertos y (Test-NC) es la precisión al eliminar del conjunto de prueba todos los ejemplos que no están cubiertos por ninguna regla.

Además, utilizaremos los mejores parámetros de los distintos criterios obtenidos en cada método.

Se muestra que, en ambos casos, los resultados obtenidos por las tres propuestas logran aumentar la capacidad de predicción. Sin embargo, se puede observar que este aumento es mucho menor en los conjuntos de datos donde el algoritmo *WM-C* tiene un bajo porcentaje de ejemplos no cubiertos, siendo del orden de 2 a 3 puntos porcentuales más. En contraste, se observa que el aumento es mucho mayor en los conjuntos de datos donde el algoritmo *WM-C* presenta mayores problemas de generalización, obteniendo resultados promedio generales de un aumento de entre 10 o 12 puntos porcentuales más.

También se puede apreciar un comportamiento diferente en relación con el tiempo de aprendizaje. En el primer caso, donde el problema de generalización no es tan intenso, se observa que el método basado en seleccionar las k mejores reglas es el que requiere menos tiempo de aprendizaje en comparación con el algoritmo *WM-C*, a diferencia de los otros métodos que se mantienen en el mismo orden. Mientras tanto, en el segundo caso, todos los métodos estudiados son del orden de 10 veces más rápidos que el correspondiente al algoritmo *WM-C*. A continuación, se procederá a realizar una comparación integral de todos los métodos con el objetivo de determinar cuál de ellos arroja los mejores resultados.

6.3.5. Comparación de los métodos propuestos

Se muestra una comparación entre los mejores resultados (seleccionando los parámetros que han dado los mejores resultados para cada criterio) de los métodos propuestos promediados en todos los conjuntos de datos en la Tabla 6.10. Todos los métodos mejoran significativamente la capacidad de predicción en comparación con el algoritmo *WM-C*, y el tiempo de aprendizaje se mantiene más bajo. En promedio, se puede observar que el método con $t = 0{,}2$ y $d = 2$ tiene un tiempo de aprendizaje más alto que el correspondiente a $k = 64$, esto se debe a lo que explicamos anteriormente, que para conjuntos de datos con más de 10 variables continuas, el tiempo de aprendizaje aumenta en esos métodos.

El tiempo de inferencia se mantiene constante en todos los casos, y los ejemplos no cubiertos disminuyen en una proporción similar al aumento en la precisión.

La Tabla 6.11 muestra un estudio estadístico utilizando Comparaciones por pares mediante la prueba de rangos con signo de Wilcoxon con corrección de continuidad

	Test			
Conjunto de Datos	WM-C	(k=64)	(t=0,2)	(d=2)
banana	80,36	80,36	80,89	80,36
bupa	59,71	61,94	61,46	63,99
iris	95,33	95,33	95,33	96,00
letter	74,60	84,78	86,19	83,98
magic	80,80	81,69	82,41	82,34
newthyroid	91,17	92,55	91,19	92,10
page-blocks	93,90	94,08	94,72	94,26
phoneme	78,66	78,72	80,12	79,09
pima	70,59	72,80	70,97	72,67
satimage	51,23	66,70	72,73	66,04
segment	88,48	91,99	92,12	92,03
shuttle	84,24	85,47	84,38	89,87
thyroid	92,10	92,46	92,49	92,61
vowel	93,03	96,26	96,36	96,06
wineq-red	59,60	62,73	60,29	60,48
wineq-white	53,08	54,80	53,90	53,92
Promedio	77,93	80,79	80,97	80,99

	Tiempo			
	WM-C	(k=64)	(t=0,2)	(d=2)
Promedio	4,47	0,50	13,37	1,82

Tab. 6.8.: Resultados obtenidos para el algoritmo *WM-C* y los mejores resultados de cada uno de los métodos propuestos para conjuntos de datos con menos del 4 % de ejemplos no cubiertos, donde (Test) es la precisión en el conjunto de prueba y (Tiempo) es el tiempo promedio en segundos consumidos por el algoritmo de aprendizaje.

	Test			
Conjunto de Datos	WM-C	(k=64)	(t=0,2)	(d=2)
adult	64,38	67,10	66,78	67,06
australian	54,64	67,68	62,75	67,25
bands	16,62	36,71	43,14	49,30
cleveland	18,54	32,64	28,89	29,94
coil2000	24,27	24,30	24,31	24,30
crx	41,17	52,17	47,93	52,33
mammogr	76,74	78,06	77,57	78,06
penbased	94,31	97,69	98,01	98,48
ring	53,24	65,99	73,00	70,84
saheart	62,12	66,45	61,23	64,73
spambase	80,66	85,25	86,14	85,62
twonorm	34,93	76,27	85,82	86,81
vehicle	54,49	62,06	65,11	65,72
Promedio	52,01	62,49	63,13	64,65

	Tiempo			
	WM-C	(k=64)	(t=0,2)	(d=2)
Promedio	16,34	0,49	1,05	1,29

Tab. 6.9.: Resultados obtenidos para el algoritmo *WM-C* y los mejores resultados de cada uno de los métodos propuestos para conjuntos de datos con más del 4 % de ejemplos no cubiertos, donde (Test) es la precisión en el conjunto de prueba y (Tiempo) es el tiempo promedio en segundos consumidos por el algoritmo de aprendizaje.

con un intervalo de confianza del 95 %, obteniendo los siguientes resultados en relación al p-valor de dos de los principales parámetros analizados.

- Parámetro Test: Se observa que existen diferencias significativas entre el algoritmo *WM-C* y todos los métodos propuestos. También se puede notar que no hay diferencias significativas entre todos los métodos propuestos.
- Parámetro Tiempo: Existen diferencias significativas entre el algoritmo *WM-C* y todos los métodos propuestos. Entre los métodos, podemos destacar que no hay diferencias significativas entre $k = 64$ y $t = 0{,}2$, pero sí hay diferencias significativas entre $d = 2$ y $k = 64$, $t = 0{,}2$.

	WM-C	(k=64)	(t=0,2)	(d=2)
Test	64,13	70,73	71,23	71,77
Tiempo	9,47	0,49	7,59	1,53
Infer	0,06	0,05	0,06	0,05

Tab. 6.10.: Comparación entre el algoritmo de *WM-C* y el promedio de los métodos propuestos para $k = 64$, $t = 0,2$ y $d = 2$. Donde (Test) es la precisión en el conjunto de prueba, (Tiempo) es el número de segundos consumidos por el algoritmo de aprendizaje e (Infer) corresponde al número de segundos consumidos por el proceso de inferencia.

	Test		
	(k=64)	(d=2)	(t=0,2)
(d=2)	0,61	-	-
(t=0,2)	0,61	0,53	-
(WM-C)	2,0e-05	1,6e-05	3,4e-05

	Tiempo		
	(k=64)	(d=2)	(t=0,2)
(d=2)	0,0038	-	-
(t=0,2)	0,4898	0,0038	-
(WM-C)	1,2e-05	0,0038	0,0025

Tab. 6.11.: Comparaciones por pares mediante la prueba de rangos con signo de Wilcoxon con corrección de continuidad, entre el algoritmo de *WM-C* y el promedio de los métodos propuestos para $k = 64$, $t = 0,2$ y $d = 2$. Donde (Test) es la precisión en el conjunto de prueba, (Tiempo) es el número de segundos consumidos por el algoritmo de aprendizaje.

En conclusión, se puede afirmar que todos los métodos presentan una mayor precisión y un menor tiempo de aprendizaje en comparación con el algoritmo *WM-C*. No obstante, es importante destacar que en el método $d = 2$, se observa un tiempo de aprendizaje más prolongado en comparación con los otros métodos propuestos.

Considerando lo anterior y el estudio de cada propuesta por separado realizado anteriormente, se puede afirmar que la propuesta más interesante para incluir reglas redundantes es la de seleccionar las k mejores reglas. Asimismo, se sugiere tomar el valor $k = 64$ como referencia general, aunque este parámetro podría ajustarse de manera específica al conjunto de datos en cuestión.

6.3.6. Estudio de problemas con un gran número de ejemplos

El análisis se enfoca en la evaluación del método que ha demostrado el mejor rendimiento en experimentos previos, concretamente, el enfoque descrito en la Subsección 6.3.1, utilizando el parámetro $k = 64$. Este enfoque se denominará *QCHI-MR*$_{k=64}$ y se examinará en problemas que involucran un gran volumen de ejemplos.

Para contrastar *QCHI-MR*$_{k=64}$, se lo comparará con un modelo de clasificación basado en reglas difusas denominado como CHI$^{BD}_{Global}$ [27], el cual utiliza el paradigma MapReduce [25, 24] aplicado al algoritmo *WM-C* para abordar problemas de Big Data.

Dentro del estudio experimental propuesto [27], se realiza un análisis utilizando 20 conjuntos de datos de clasificación binaria, derivados de 8 conjuntos de datos originales de UCI. En la Tabla del Anexo A.2, se detallan las características clave de estos conjuntos, incluyendo el número total de ejemplos, la relación entre las clases mayoritarias y minoritarias, el total de variables y cuántas de ellas son continuas.

Las condiciones experimentales se establecieron en los siguientes términos:

- Se implementó un esquema de validación cruzada de 5 particiones.
- Para la configuración del algoritmo de *WM-C*, se empleó una discretización de tres etiquetas para las variables continuas, se hizo uso del método de inferencia de la regla ganadora y se aplicó el modelo de peso PCF-CS.
- La precisión de CHI^{BD}_{Global} es la misma que la generada por el algoritmo de *WM-C*.
- Dado el desequilibrio característico en la mayoría de los conjuntos de datos, se seleccionaron el Área Bajo la Curva ROC (AUC) y la Media Geométrica (GM) como medidas de calidad de clasificación.
- La ejecución de todos los métodos paralelos se llevó a cabo en un clúster compuesto por 8 nodos interconectados mediante una red LAN Ethernet de 1Gb/s. Para obtener información adicional y más detallada, se recomienda consultar el artículo científico correspondiente.

En la Tabla 6.12, se presenta una comparativa entre el método propuesto, *QCHI-MR*$_{k=64}$, y los resultados obtenidos utilizando CHI^{BD}_{Global} [27]. Los valores (*Test*) indican la precisión teniendo en cuenta todos los ejemplos del conjunto de prueba, mientras que los valores (*Tiempo*) denotan el tiempo de ejecución necesario para el proceso de aprendizaje, medido en segundos. Es importante destacar que tanto el algoritmo *WM-C* como CHI^{BD}_{Global} tiene el mismo nivel de precisión, lo que se refleja en la columna (*Test*). Sin embargo, la diferencia en el tiempo de aprendizaje radica en que CHI^{BD}_{Global} utiliza el paradigma MapReduce. El tiempo de aprendizaje de CHI^{BD}_{Global} corresponde al resultado del tiempo de la versión que utiliza procesamiento paralelo con 32 mappers (mapper es un servidor de Hadoop que ejecuta las funciones de MapReduce).

Se utiliza el modelo PCF-CS en lugar del modelo PCF para la ponderación, lo que puede resultar en una reducción adicional en el número de reglas generadas. Esto conlleva a ligeras variaciones en los valores de tiempo de ejecución. Por lo tanto, los tiempos registrados en la tabla deben considerarse como estimaciones adecuadas de lo que podría requerir CHI^{BD}_{Global} en estos escenarios particulares.

	QCHI-MR$_{k=64}$		WM-C		CHI$^{BD}_{Global}$
Conjunto de Datos	Test	Tiempo	Test	Tiempo	Tiempo
census	53,92	11,25	52,61	1414,20	96,00
covtype1	78,60	52,30	75,52	1986,00	76,00
covtype2	77,20	51,44	73,65	1994,84	75,00
covtype3	96,55	51,90	96,20	1985,41	74,00
covtype7	98,25	53,95	98,03	1987,32	75,00
fars_Fat	96,56	8,33	92,38	484,09	81,00
fars_In	85,01	8,42	81,11	482,73	80,00
fars_No	93,35	8,54	89,25	483,00	82,00
fars_Nin	83,37	8,54	79,62	483,78	81,00
KDD_dos	99,94	262,03	99,93	4282,89	78,00
KDD_normal	99,92	265,93	99,91	4257,21	76,00
KDD_probe	99,98	259,33	99,97	4208,90	77,00
KDD_r2l	99,98	263,56	99,97	4220,76	75,00
poker0	63,77	39,90	63,53	6853,13	107,00
poker1	59,44	42,03	59,18	6692,17	110,00
poker2	95,24	42,78	95,24	6706,81	109,00
poker3	97,89	42,59	97,89	6707,06	106,00
skin	90,63	1,91	90,63	2,12	53,00
susy	67,16	247,17	62,48	15409,65	103,00
Promedio	86,14	90,63	84,58	3718,00	84,95

Tab. 6.12.: Resultados obtenidos por el algoritmo QCHI-MR frente a *WM-C* y CHI$^{BD}_{Global}$ utilizando una validación cruzada de 5 particiones y un dominio uniformemente distribuido con 3 etiquetas en todas las variables continuas, donde la columna (Test) es la precisión en el conjunto de prueba y (Tiempo) es el número de segundos consumidos por el algoritmo de aprendizaje.

Estudio	WM-C vs QCHI-MR$_{k=64}$	W/T/L	p-valor
1	Test	0/3/16	0,00002
2	Tiempo	19/0/0	0,00001

Estudio	CHI$^{BD}_{Global}$ vs QCHI-MR$_{k=64}$	W/T/L	p-valor
3	Test	0/3/16	0,00002
4	Tiempo	14/0/5	0,70856

Tab. 6.13.: Pruebas estadísticas que comparan los enfoques QCHI-MR con *WM-C* y CHI$^{BD}_{Global}$, donde la (Test) es la precisión en el conjunto de prueba y (Tiempo) es el número de segundos consumidos por el algoritmo de aprendizaje.

Se puede observar que en promedio hay un aumento en la precisión del orden del 1.5 % aproximadamente en *QCHI-MR*$_{k=64}$, y el tiempo de ejecución tiene valores similares a los obtenidos por CHI$^{BD}_{Global}$.

Para contrastar estos resultados en términos de precisión y tiempo de aprendizaje, se aplicó la prueba estadística de rangos con signo de Wilcoxon [96] con un intervalo de confianza del 95 %. La Tabla 6.13 presenta los resultados de las cuatro comparaciones realizadas. Se resumen las victorias (W), empates (T) y derrotas (L) de cada método en estas comparaciones, junto con los p-valor calculados. Esto permite una evaluación de las diferencias entre el método propuesto *QCHI-MR*$_{k=64}$ y los otros enfoques, *WM-C* y CHI$^{BD}_{Global}$. A continuación, se presentan las conclusiones derivadas de estos análisis comparativos.

- Al comparar el algoritmo *WM-C* con *QCHI-MR*$_{k=64}$ se puede observar que el método propuesto es estadísticamente mejor en cuanto a la precisión, y además el tiempo de aprendizaje es significativamente menor al obtenido por el algoritmo *WM-C*.

- Al comparar CHI$^{BD}_{Global}$ y *QCHI-MR*$_{k=64}$ se puede observar que el método propuesto es estadísticamente mejor en cuanto a la precisión (aplicando el mismo análisis realizado con el algoritmo de *WM-C*) y en cuanto al tiempo de aprendizaje, no hay una diferencia significativa entre ambos algoritmos.

A continuación, se analizara exclusivamente el conjunto de datos *higgs*. Este conjunto de datos consta de 11 millones de ejemplos y se utiliza con frecuencia para determinar el rendimiento de los algoritmos de clasificación cuando se enfrentan a problemas con un número muy grande de ejemplos, donde para la mayoría de los algoritmos de clasificación existentes, el tiempo requerido para el entrenamiento es tan alto que ni siquiera son aplicables. La Tabla 6.14 amplía el estudio anterior

a diferentes valores del parámetro k, donde se comparan los resultados de los algoritmos *WM-C* y CHI$^{BD}_{Global}$ en el conjunto de datos *higgs* con diferentes versiones de nuestra propuesta *QCHI-MR*$_k$. Los resultados obtenidos por CHI$^{BD}_{Global}$ se ilustran en las Tablas 5 y 7 del artículo [27], donde la calidad de la clasificación se mide utilizando la media geométrica (GM) y el área bajo la curva (AUC). En relación con *QChi-MR*$_k$, todos los experimentos se han realizado en un Intel® Core™ i5-11400 de 11ª generación a 2,60 GHz × 12 y 64 Gb de memoria funcionando con el sistema operativo Ubuntu 22.04.3 LTS x86 64.

Los datos para el algoritmo *WM-C* se han obtenido de la siguiente manera: el resultado en GM y AUC es el mismo que el obtenido por CHI$^{BD}_{Global}$, ya que este algoritmo, según afirman sus autores, es una versión paralelizada (bajo la arquitectura MapReduce) que reproduce los mismos resultados que los obtenidos por *WM-C*. En cuanto al tiempo de aprendizaje (*Tiempo*) del algoritmo *WM-C*, se presenta el resultado obtenido a través de una implementación propia. En esta implementación, una vez que se conoce el número de reglas en el conjunto final, es posible estimar con precisión el tiempo necesario para calcular el peso. Para el conjunto de datos *higgs*, se estima que el tiempo promedio requerido es de aproximadamente dos millones doscientos mil segundos, equivalente a unos 25 días. En comparación con este tiempo, resulta evidente que CHI$^{BD}_{Global}$ proporciona una alternativa en la que la base de reglas se genera en aproximadamente dos horas y media.

La comparación revela que *QChi* requiere menos de 2 minutos, significativamente menos tiempo en comparación con CHI$^{BD}_{Global}$, aunque su capacidad de clasificación, medida en términos de media geométrica (GM) y área bajo la curva (AUC), son ligeramente inferiores a la de CHI$^{BD}_{Global}$. Sin embargo, los siguientes resultados indican que *QCHI-MR*$_k$ puede considerarse una alternativa viable a CHI$^{BD}_{Global}$ para

Metodo	Test (GM)	Test (AUC)	Tiempo
WM-C	0,5847	0,5848	*2.200.000,00
CHI$^{BD}_{Global}$	0,5847	0,5848	9.658,00
QChi	0,5639	0,5642	117,35
QCHI-MR$_{k=2}$	0,5725	0,5727	122,62
QCHI-MR$_{k=4}$	0,5818	0,5820	135,86
QCHI-MR$_{k=8}$	0,5924	0,5924	179,91
QCHI-MR$_{k=16}$	0,6026	0,6024	233,13
QCHI-MR$_{k=32}$	0,6103	0,6100	374,81
QCHI-MR$_{k=64}$	0,6147	0,6143	672,90

Tab. 6.14.: Comparación entre *WM-C* y CHI$^{BD}_{Global}$ frente a QCHI-MR para el conjunto de datos higgs, donde la (GM) es la media geométrica, (AUC) es el área bajo la curva y (Tiempo) es el número de segundos consumidos por el algoritmo de aprendizaje.

este problema, ya que reduce sustancialmente el tiempo de aprendizaje y mejora la capacidad de clasificación.

Los resultados óptimos en términos de tiempo de aprendizaje y capacidad predictiva se logran con la versión de *QCHI-MR*$_k$ cuando $k \geq 8$ (para $k = 8$ requiere menos de 3 minutos). Esta versión demuestra ser posiblemente la más eficaz, mejorando la capacidad de clasificación mientras reduce el tiempo necesario para obtener esta mejora. Además, se observa que, a medida que k aumenta, hay una mejora en la capacidad predictiva, y el tiempo también aumenta, aunque no en la misma proporción que se esperaría al duplicar el valor de k.

A continuación, en el apartado de Reflexiones, se llevará a cabo un análisis integral de todas las experimentaciones, con el propósito de sintetizar el trabajo realizado.

6.3.7. Reflexiones

Se puede afirmar que todos los métodos analizados en este estudio presentan una mayor precisión y un menor tiempo de aprendizaje en comparación con el algoritmo *WM-C* cuando se trabaja con conjuntos de datos no masivos. Luego de un análisis detallado de cada método, se concluye que la propuesta más interesante para incluir reglas redundantes es la de seleccionar las k mejores reglas, siendo el valor $k = 64$ una referencia general, aunque su ajuste podría ser necesario para adaptarse a conjuntos de datos específicos.

En el caso de conjuntos de datos masivos, al comparar el algoritmo *WM-C* con *QCHI-MR*$_{k=64}$ se evidencia que el método propuesto es estadísticamente mejor en términos de precisión, además de requerir significativamente menos tiempo de aprendizaje.

Otro de los aspectos importantes que se han puesto de manifiesto es la relación entre los resultados de *QCHI-MR*$_{k=64}$ frente a las propuestas que hacen uso de arquitecturas paralelas. Al comparar CHI$^{BD}_{Global}$ y *QCHI-MR*$_{k=64}$ se observa que no hay una diferencia significativa en el tiempo de aprendizaje entre ambos algoritmos pero en capacidad de predicción el método *QCHI-MR*$_{k=64}$ es superior. En particular, en el conjunto de datos *higgs*, se destaca que QCHI-MR$_{k=8}$ requiere aproximadamente 3 minutos, mucho menos tiempo que CHI$^{BD}_{Global}$ el cual ronda las 2 horas y media, y su capacidad de clasificación, medida en términos de la media geométrica y en AUC, es mejor. Los mejores resultados en tiempo y capacidad predictiva se obtienen con la versión de *QCHI-MR*$_k$ con valores de k igual o mayor a 8.

6.4. Contribuciones

- *A new multi-rules approach to improve the performance of the Chi fuzzy rule classification algorithm.* (FUZZ-IEEE 2022) [53]
- *Una versión alternativa del algoritmo de Chi, de aprendizaje de reglas difusas para clasificación, basada en la obtención de múltiples reglas sobre cada ejemplo.* (ESTYLF 2022) [55]
- *Study of the effect of using redundant fuzzy rules in classification problems. (En elaboracion)*

7 Conclusiones

7.1. Conclusiones

En este estudio se han abordado diversas mejoras y propuestas para optimizar el algoritmo *WM-C* en el contexto de sistemas de clasificación basados en reglas difusas. Cada capítulo se centró en una perspectiva diferente sobre la que se presentaron unas conclusiones parciales. En esta sección mostraremos un resumen de las conclusiones de cada capítulo, así como un breve análisis conjunto de las aportaciones.

En el Capítulo 3 dedicado a la minimización de la base de reglas, cuyo propósito es mejorar la interpretabilidad del modelo, se exploraron dos métodos de codificación para reducir la cantidad de reglas difusas en la base resultante. Se observó que ambas estrategias lograron disminuir considerablemente la cantidad de reglas mientras se mantenía una precisión aceptable en comparación con el algoritmo *WM-C*. Aunque la codificación ternaria y binaria tuvieron un impacto similar en la cantidad de reglas, la codificación ternaria se destacó por generar reglas más simples y fácilmente interpretables. Sin embargo, se reconoció que ambos métodos presentan limitaciones en términos de escalabilidad, especialmente para conjuntos de datos con un alto número de variables.

En el Capítulo 4 centrado en la mejora del tiempo de respuesta del método de inferencia, se evidenció que diferentes enfoques de inferencia son más adecuados según las características del problema. El enfoque estándar funcionó bien cuando el número de reglas era bajo y el número de variables continuas era alto. Por otro lado, el enfoque recursivo fue efectivo en situaciones con un bajo número de variables continuas. Sin embargo, para problemas con un alto número de reglas y variables continuas, la integración de métodos en un algoritmo híbrido demostró ser la solución más eficaz.

En el Capítulo 5, se propuso una mejora en el tiempo de respuesta del algoritmo de clasificación mediante la introducción de una alternativa basada en estimaciones de peso en lugar de cálculos exactos. Los resultados experimentales demostraron que la propuesta denominada como *QChi* fue superior en rendimiento comparada con la del algoritmo de *WM-C*, reduciendo significativamente el tiempo de aprendizaje

sin comprometer la capacidad de clasificación. También se comparó favorablemente con enfoques basados en MapReduce.

El capítulo 6 se centró en la incorporación de reglas redundantes en un sistema de clasificación, con el propósito de obtener reglas que cubran de manera más efectiva el conjunto de ejemplos y, por ende, reducir el porcentaje de ejemplos no cubiertos. Se concluyó que los métodos analizados en este estudio presentaron una mayor precisión y un menor tiempo de aprendizaje en comparación con el algoritmo *WM-C*. La estrategia de selección de las k mejores reglas se mostró como la más prometedora, con el valor $k = 64$ como referencia general para conjuntos de datos convencionales y $k = 8$ para conjuntos de datos masivos. Esta propuesta resultó en mejoras significativas en términos de precisión y tiempo de aprendizaje en conjuntos de datos masivos, mostrando su capacidad para competir con otros métodos más complejos.

A lo largo de este tesis doctoral, se han propuesto una serie de mejoras sustanciales en diversas facetas del algoritmo de aprendizaje *WM-C*. Estas modificaciones han sido diseñadas para potenciar el rendimiento del algoritmo frente a una variedad de desafíos, incluyendo la manipulación de conjuntos de datos de gran tamaño. Al abordar estos aspectos desde múltiples perspectivas, se ha logrado reducir la complejidad de las reglas sin menoscabar la precisión, mejorando significativamente la interpretabilidad del modelo resultante. Asimismo, se han implementado mejoras en la eficiencia de la inferencia, adaptándola de manera acorde a las particularidades de cada problema. La sustitución de cálculos exactos por estimaciones de peso ha conducido a una disminución notable en el tiempo de aprendizaje. La incorporación de reglas redundantes ha fortalecido la precisión y al mismo tiempo ha acelerado el proceso de aprendizaje, particularmente en contextos que involucran volúmenes masivos de datos. En su conjunto, estas innovaciones apuntan a ofrecer un enfoque de clasificación más eficaz y eficiente para una amplia gama de aplicaciones.

7.2. Trabajos futuros

Se identifican varios caminos de investigación prometedores a partir de los resultados presentados en este estudio. Aunque los enfoques propuestos han mejorado significativamente la eficiencia y precisión del algoritmo *WM-C*, objetivo de esta tesis, persisten desafíos en relación con la escalabilidad, interpretabilidad y adaptación a conjuntos de datos más complejos, caracterizados por un mayor número de ejemplos y variables. Estas áreas ofrecen oportunidades para investigaciones futuras.

En relación a la mejora de la interpretabilidad (Capítulo 3) la principal limitación de la propuesta tiene que ver con resolver problemas con muchas variables y/o valores en el dominio de cada variable, ya que la complejidad del método de Quine-McCluskey aumenta exponencialmente en función del número de variables. Así, el método binario normalmente requiere un número muy elevado de variables por su propia forma de codificación y el método ternario, si bien es menos dependiente del aumento de valores en el dominio, la complejidad del problema afecta notablemente en la detección y eliminación de variables irrelevantes.

Como perspectiva para futuras investigaciones, se plantea la exploración de métodos más eficientes y escalables que no dependan de manera exponencial del número de variables y/o valores del dominio, lo que permitiría abordar conjuntos de datos más complejos. Se podrían aplicar los conceptos de vecindad de un ejemplo definido en el Capítulo 4 u otro tipo de heurísticas, con el fin de acotar el cálculo necesario para la minimización de reglas utilizando métodos aproximados que no tengan un alto costo computacional.

Los métodos de inferencia presentados en el Capítulo 4 han demostrado mejoras notables en la reducción del tiempo de inferencia en conjuntos de datos sobre ciertos tipos de bases de reglas (normalmente asociados a problemas con un número no muy elevado de variables continuas), sin embargo un problema que no ha quedado completamente cerrado, es encontrar una inferencia eficiente para problemas en los que haya un gran número de variables continuas y a la vez un número elevado de reglas difusas.

En el Capítulo 5, correspondiente a la propuesta de cálculo de peso basada en aproximaciones (*QChi*), se plantea un debate relevante en torno a la utilidad de calcular los pesos utilizando valores exactos en la generación de la base de reglas. Se observa que, al emplear una estimación del peso, se logran mejoras significativas en el tiempo de aprendizaje, especialmente a medida que el tamaño del conjunto de datos se vuelve considerable, lo que resulta beneficioso para trabajar con conjuntos de datos masivos.

Como posibles mejoras adicionales a esta aproximación, se sugiere la realización de un análisis detallado de la distribución de los datos usando técnicas de preprocesamiento. Este análisis podría proporcionar indicios valiosos para desarrollar enfoques de aproximación de pesos aún más eficientes.

Por otra parte, la versión extendida de *QChi*, es decir *QChi-MR*, consigue claras mejoras en predicción, pero plantea un problema de interpretabilidad (que ya tiene el algoritmo original *WM-C*, pero que en este caso se incrementa) al generar

un número elevado de reglas. Este problema no se ve afectado en el proceso de inferencia al utilizar las técnicas propuestas en esta tesis, pero sin duda afecta notablemente a la interpretabilidad.

En relación con propuestas de mejora, una vía de trabajo sería explorar nuevos criterios de selección de las mejores reglas, o estudiar combinaciones de los criterios ya propuestos.

Por último, otras dos líneas de trabajo que quedan abiertas son las siguientes. La primera es el uso de multiclasificadores de forma que se pueda modificar la granularidad de los modelos de acuerdo con las necesidades específicas del conjunto de datos, aprovechando la eficiencia computacional inherente a las propuestas presentadas. La segunda consiste en aplicar los modelos a conjuntos de datos todavía más complejos, superando en tamaño incluso a los casos utilizados en esta tesis doctoral. Para lograrlo, se podría implementar una arquitectura de procesamiento paralelo utilizando técnicas tipo MapReduce. Este enfoque tendría el potencial de reducir aún más los tiempos de aprendizaje e inferencia del clasificador, lo que sería fundamental en el procesamiento eficiente de conjuntos de datos masivos, una tendencia cada vez más relevante en el campo del análisis de datos y la inteligencia artificial

Bibliografía

[1] Fatemeh Aghaeipoor, Mohammad Masoud Javidi, Isaac Triguero y Alberto Fernández. "Chi-BD-DRF: design of scalable fuzzy classifiers for big data via a dynamic rule filtering approach". En: *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE. 2020, págs. 1-7 (vid. págs. 26, 28, 95, 99).

[2] Jesús Alcalá-Fdez, Rafael Alcala y Francisco Herrera. "A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning". En: *IEEE Transactions on Fuzzy systems* 19.5 (2011), págs. 857-872 (vid. pág. 5).

[3] Diego Alvarez-Estevez y Vicente Moret-Bonillo. "Revisiting the Wang–Mendel algorithm for fuzzy classification". En: *Expert Systems* 35.4 (2018), e12268 (vid. págs. 6, 24, 28, 117).

[4] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". En: *Information fusion* 58 (2020), págs. 82-115 (vid. pág. 35).

[5] Carmen Biedma-Rdguez, María José Gacto, Augusto Anguita-Ruiz et al. "Learning positive-negative rule-based fuzzy associative classifiers with a good trade-off between complexity and accuracy". En: *Fuzzy Sets and Systems* 465 (2023), pág. 108511 (vid. pág. 17).

[6] Mladenka Blagojević, Milica Šelmić, Dragana Macura y Dragana Šarac. "Determining the number of postal units in the network–Fuzzy approach, Serbia case study". En: *Expert systems with Applications* 40.10 (2013), págs. 4090-4095 (vid. pág. 28).

[7] Robert K Brayton, Gary D Hachtel, Curt McMullen y Alberto Sangiovanni-Vincentelli. *Logic minimization algorithms for VLSI synthesis*. Vol. 2. Springer Science & Business Media, 1984 (vid. pág. 36).

[8] Christopher JC Burges. "A tutorial on support vector machines for pattern recognition". En: *Data mining and knowledge discovery* 2.2 (1998), págs. 121-167 (vid. pág. 5).

[9] Danyal Bustan, Hoda Moodi, Naser Pariz y Nika Azmoodeh. "A fuzzy inference method for systems with large number of rules". En: *NAFIPS 2006-2006 Annual Meeting of the North American Fuzzy Information Processing Society*. IEEE. 2006, págs. 397-402 (vid. pág. 56).

[10] Jorge Casillas y Pedro Martinez. "Consistent, complete and compact generation of DNF-type fuzzy rules by a Pittsburgh-style genetic algorithm". En: *2007 IEEE International Fuzzy Systems Conference*. IEEE. 2007, págs. 1-6 (vid. pág. 21).

[11]Pei-Chann Chang, Jih-Chang Hieh y T Warren Liao. “Evolving fuzzy rules for due-date assignment problem in semiconductor manufacturing factory”. En: *Journal of Intelligent Manufacturing* 16 (2005), págs. 549-557 (vid. pág. 5).

[12]Dewang Chen, Wenlin Tong, Yunhu Huang, Lingxi Li y Jing Zhang. “FLOWFS: fast learning-algorithm with optimal weights for fuzzy systems”. En: *International Journal of Fuzzy Systems* 24.7 (2022), págs. 3162-3173 (vid. págs. 27, 95).

[13]Haonan Chen, Tao Wu, Wangyong He, Yongbo Li y Xinmei Wang. “An improved Wang-Mendel algorithm in Engine Test Bench control system”. En: *2021 China Automation Congress (CAC)*. IEEE. 2021, págs. 252-257 (vid. pág. 28).

[14]Zheru Chi, Hong Yan y Tuan Pham. *Fuzzy algorithms: with applications to image processing and pattern recognition*. Vol. 10. World Scientific, 1996 (vid. págs. 5, 6, 24, 25, 35, 48, 55, 95, 96).

[15]Marcos Evandro Cintra, Camargo Heloisa de Arruda y Maria Carolina Monard. “Fuzzy feature subset selection using the Wang & Mendel method”. En: *2008 Eighth International Conference on Hybrid Intelligent Systems*. IEEE. 2008, págs. 590-595 (vid. págs. 5, 28).

[16]Cisco. *Cisco Annual Internet Report-Cisco Annual Internet Report (2018–2023) White Paper*. 2020 (vid. págs. 3, 4).

[17]Marco Cococcioni, Luca Foschini, Beatrice Lazzerini y Francesco Marcelloni. “Complexity reduction of Mamdani fuzzy systems through multi-valued logic minimization”. En: *2008 IEEE International Conference on Systems, Man and Cybernetics*. IEEE. 2008, págs. 1782-1787 (vid. pág. 28).

[18]William W Cohen. “Fast effective rule induction”. En: *Machine learning proceedings 1995*. Elsevier, 1995, págs. 115-123 (vid. pág. 5).

[19]Oscar Cordón, Maria José Del Jesus y Francisco Herrera. “A proposal on reasoning methods in fuzzy rule-based classification systems”. En: *International Journal of Approximate Reasoning* 20.1 (1999), págs. 21-45 (vid. págs. 14, 17, 22).

[20]Oscar Cordón, F Herrera, F Hoffmann y L Magdalena. “Evolutionary tuning and learning of fuzzy knowledge bases”. En: *Genetic fuzzy systems* 19 (2001) (vid. pág. 23).

[21]Oscar Cordón y Francisco Herrera. “A three-stage evolutionary process for learning descriptive and approximate fuzzy-logic-controller knowledge bases from examples”. En: *International Journal of Approximate Reasoning* 17.4 (1997), págs. 369-407 (vid. pág. 28).

[22]Thomas Cover y Peter Hart. “Nearest neighbor pattern classification”. En: *IEEE transactions on information theory* 13.1 (1967), págs. 21-27 (vid. pág. 5).

[23]Adele Cutler, D Richard Cutler y John R Stevens. “Random forests”. En: *Ensemble machine learning: Methods and applications* (2012), págs. 157-175 (vid. pág. 4).

[24]Jeffrey Dean y Sanjay Ghemawat. “MapReduce: a flexible data processing tool”. En: *Communications of the ACM* 53.1 (2010), págs. 72-77 (vid. págs. 28, 56, 99, 139).

[25]Jeffrey Dean y Sanjay Ghemawat. “MapReduce: simplified data processing on large clusters”. En: *Communications of the ACM* 51.1 (2008), págs. 107-113 (vid. págs. 28, 56, 99, 109, 139).

[26]Eleonora D’Andrea y Beatrice Lazzerini. “A hierarchical approach to multi-class fuzzy classifiers”. En: *Expert Systems with Applications* 40.9 (2013), págs. 3828-3840 (vid. pág. 28).

[27]Mikel Elkano, Mikel Galar, Jose Sanz y Humberto Bustince. “CHI-BD: A fuzzy rule-based classification system for Big Data classification problems”. En: *Fuzzy Sets and Systems* 348 (2018), págs. 75-101 (vid. págs. 6, 26, 28, 95, 96, 99, 111-114, 139, 140, 143, 161).

[28]Mikel Elkano, Mikel Galar, Jose Sanz y Humberto Bustince. “CHI-PG: A fast prototype generation algorithm for Big Data classification problems”. En: *Neurocomputing* 287 (2018), págs. 22-33 (vid. págs. 28, 56).

[29]Mikel Elkano, Jose Antonio Sanz, Edurne Barrenechea, Humberto Bustince y Mikel Galar. “CFM-BD: A distributed rule induction algorithm for building compact fuzzy models in big data classification problems”. En: *IEEE Transactions on Fuzzy Systems* 28.1 (2019), págs. 163-177 (vid. pág. 35).

[30]Zongwen Fan, Raymond Chiong, Zhongyi Hu, Sandeep Dhakal y Yuqing Lin. “A two-layer Wang-Mendel fuzzy approach for predicting the residuary resistance of sailing yachts”. En: *Journal of Intelligent & Fuzzy Systems* 36.6 (2019), págs. 6219-6229 (vid. pág. 28).

[31]Alberto Fernández, Salvador García, Francisco Herrera y María José del Jesús. “An analysis of the rule weights and fuzzy reasoning methods for linguistic rule based classification systems applied to problems with highly imbalanced data sets”. En: *Applications of Fuzzy Sets Theory: 7th International Workshop on Fuzzy Logic and Applications, WILF 2007, Camogli, Italy, July 7-10, 2007. Proceedings 7*. Springer. 2007, págs. 170-178 (vid. págs. 5, 28).

[32]Alberto Fernández, Sara del Río, Abdullah Bawakid y Francisco Herrera. “Fuzzy rule based classification systems for big data with MapReduce: granularity analysis”. En: *Advances in Data Analysis and Classification* 11 (2017), págs. 711-730 (vid. págs. 95, 99, 109-111).

[33]Maria Jose Gacto, Rafael Alcalá y Francisco Herrera. “Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures”. En: *Information Sciences* 181.20 (2011), págs. 4340-4360 (vid. pág. 35).

[34]Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince y Francisco Herrera. “A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches”. En: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.4 (2011), págs. 463-484 (vid. pág. 26).

[35]Salvador García, Sergio Ramírez-Gallego, Julián Luengo, José Manuel Benítez y Francisco Herrera. “Big data preprocessing: methods and prospects”. En: *Big Data Analytics* 1.1 (2016), págs. 1-22 (vid. pág. 33).

[36]Antonio Gonzalez y Raúl Perez. “A learning system of fuzzy control rules based on genetic algorithms”. En: *Genetic algorithms and soft computing, Physica-Verlag* (1996), págs. 202-225 (vid. pág. 17).

[37]Antonio Gonzalez y Raul Perez. “Completeness and consistency conditions for learning fuzzy rules”. En: *Fuzzy Sets and Systems* 96.1 (1998), págs. 37-51 (vid. págs. 21, 22).

[38]Antonio González y Raúl Pérez. “Improving the genetic algorithm of SLAVE”. En: *Mathware & Soft Computing* 16.1 (2009), págs. 59-70 (vid. págs. 5, 17, 21).

[39]Antonio González, Raúl Pérez y Rocío Romero-Záliz. “Reasoning Methods in Fuzzy Rule-based Classification Systems for Big Data Problems.” En: *IoTBDS*. 2019, págs. 255-261 (vid. págs. 56, 59).

[40]Antonio González, Raúl Pérez y Jose L Verdegay. “Learning the structure of a fuzzy rule: a genetic approach”. En: *Fuzzy Systems and Artificial Intelligence* 3.1 (1994), págs. 57-70 (vid. págs. 5, 17, 25).

[41]Jin Gou, Feng Hou, Wenyu Chen, Cheng Wang y Wei Luo. “Improving Wang–Mendel method performance in fuzzy rules generation using the fuzzy C-means clustering algorithm”. En: *Neurocomputing* 151 (2015), págs. 1293-1304 (vid. pág. 5).

[42]Richard W Hamming. *Coding and information theory*. Prentice-Hall, Inc., 1986 (vid. págs. 41, 69, 123).

[43]Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar et al. “The rise of “big data” on cloud computing: Review and open research issues”. En: *Information systems* 47 (2015), págs. 98-115 (vid. pág. 33).

[44]Brian Holdsworth y Clive Woods. *Digital logic design*. Elsevier, 2002 (vid. pág. 37).

[45]Dawn E Holmes y Lakhmi C Jain. “Introduction to Bayesian networks”. En: *Innovations in Bayesian networks: Theory and applications*. Springer, 2008, págs. 1-5 (vid. pág. 4).

[46]Jens Hühn y Eyke Hüllermeier. “FURIA: an algorithm for unordered fuzzy rule induction”. En: *Data Mining and Knowledge Discovery* 19 (2009), págs. 293-319 (vid. pág. 5).

[47]C-C Hung y B Fernandez. “Minimizing rules of fuzzy logic system by using a systematic approach”. En: *[Proceedings 1993] Second IEEE International Conference on Fuzzy Systems*. IEEE. 1993, págs. 38-44 (vid. pág. 35).

[48]Anayo Chukwu Ikegwu, Henry Friday Nweke, Chioma Virginia Anikwe, Uzoma Rita Alo y Obikwelu Raphael Okonkwo. “Big data analytics for data-driven industry: a review of data sources, tools, challenges, solutions, and research directions”. En: *Cluster Computing* 25.5 (2022), págs. 3343-3387 (vid. pág. 30).

[49]Hisao Ishibuchi, Ken Nozaki e Hideo Tanaka. “Distributed representation of fuzzy rules and its application to pattern classification”. En: *Fuzzy sets and systems* 52.1 (1992), págs. 21-32 (vid. págs. 21, 22).

[50]Hisao Ishibuchi y Takashi Yamamoto. “Rule weight specification in fuzzy rule-based classification systems”. En: *IEEE transactions on fuzzy systems* 13.4 (2005), págs. 428-435 (vid. pág. 26).

[51]J-SR Jang. “ANFIS: adaptive-network-based fuzzy inference system”. En: *IEEE transactions on systems, man, and cybernetics* 23.3 (1993), págs. 665-685 (vid. pág. 5).

[52]Leonardo Jara, Rubén Ariza-Valderrama, Juan Fernández-Olivares, Antonio González y Raúl Pérez. “Efficient inference models for classification problems with a high number of fuzzy rules”. En: *Applied Soft Computing* 115 (2022), pág. 108164 (vid. págs. 10, 93).

[53]Leonardo Jara, Antonio González y Raúl Pérez. “A new multi-rules approach to improve the performance of the Chi fuzzy rule classification algorithm”. En: *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE. 2022, págs. 1-6 (vid. págs. 10, 145).

[54]Leonardo Jara, Antonio González y Rauĺ Pérez. “A preliminary study to apply the Quine McCluskey algorithm for fuzzy rule base minimization”. En: *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE. 2020, págs. 1-6 (vid. págs. 9, 53).

[55]Leonardo Jara, Antonio González y Rauĺ Pérez. “Una version alternativa del algoritmo de Chi, de aprendizaje de reglas difusas para clasificacion, basada en la obtencion de multiples reglas sobre cada ejemplo”. En: *XXI Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF22)*. 2022 (vid. págs. 10, 145).

[56]Leonardo Jara, Antonio González y Rauĺ Pérez. “Una Versión Ternaria del Algoritmo de Quine Mccluskey para la Minimización de Base de Reglas Difusas”. En: *XIX CONFERENCIA DE LA ASOCIACIÓN ESPAÑOLA PARA LA INTELIGENCIA ARTIFICIAL (CAEPIA 20/21)*. 2021, págs. 268-273 (vid. págs. 9, 53).

[57]Yali Jin, Weihua Cao, Min Wu y Yan Yuan. “Accurate fuzzy predictive models through complexity reduction based on decision of needed fuzzy rules”. En: *Neurocomputing* 323 (2019), págs. 344-351 (vid. pág. 5).

[58]Mustafa Karabulut. “Fuzzy unordered rule induction algorithm in text categorization on top of geometric particle swarm optimization term selection”. En: *Knowledge-Based Systems* 54 (2013), págs. 288-297 (vid. pág. 17).

[59]Nawsher Khan, Mohammed Alsaqer, Habib Shah et al. “The 10 Vs, issues and challenges of big data”. En: *Proceedings of the 2018 international conference on big data and education*. 2018, págs. 52-56 (vid. pág. 30).

[60]Daijin Kim. “Improving the fuzzy system performance by fuzzy system ensemble”. En: *Fuzzy Sets and Systems* 98.1 (1998), págs. 43-56 (vid. págs. 5, 28).

[61]Han Liu, Alexander Gegov y Frederic Stahl. “Categorization and construction of rule based systems”. En: *Engineering Applications of Neural Networks: 15th International Conference, EANN 2014, Sofia, Bulgaria, September 5-7, 2014. Proceedings 15*. Springer. 2014, págs. 183-194 (vid. pág. 13).

[62]Yu Liu, Zheng Qin, Zhewen Shi y Junying Chen. “Rule discovery with particle swarm optimization”. En: *Advanced Workshop on Content Computing*. Springer. 2004, págs. 291-296 (vid. pág. 17).

[63]Victoria López, Sara Del Río, José Manuel Benítez y Francisco Herrera. “Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data”. En: *Fuzzy Sets and Systems* 258 (2015), págs. 5-38 (vid. págs. 24, 28).

[64]Julián Luengo, Diego García-Gil, Sergio Ramírez-Gallego, Salvador García y Francisco Herrera. “Big data preprocessing”. En: *Cham: Springer* (2020) (vid. págs. 29, 30).

[65]Edwin David Lughofer. “FLEXFIS: A robust incremental learning approach for evolving Takagi–Sugeno fuzzy models”. En: *IEEE Transactions on fuzzy systems* 16.6 (2008), págs. 1393-1410 (vid. pág. 28).

[66]Luis Magdalena. “Fuzzy rule-based systems”. En: *Springer handbook of computational intelligence* (2015), págs. 203-218 (vid. pág. 13).

[67]Mamdani. “Application of fuzzy logic to approximate reasoning using linguistic synthesis”. En: *IEEE transactions on computers* 100.12 (1977), págs. 1182-1191 (vid. pág. 56).

[68]Ebrahim H Mamdani. “Application of fuzzy algorithms for control of simple dynamic plant”. En: *Proceedings of the institution of electrical engineers*. Vol. 121. 12. IET. 1974, págs. 1585-1588 (vid. pág. 17).

[69]Deba Prasad Mandal, CA Murthy y Sankar K Pal. “Formulation of a multivalued recognition system”. En: *IEEE Transactions on Systems, Man, and Cybernetics* 22.4 (1992), págs. 607-620 (vid. pág. 22).

[70]Eghbal G Mansoori, Mansoor J Zolghadri y Seraj D Katebi. “SGERD: A steady-state genetic algorithm for extracting fuzzy classification rules from data”. En: *IEEE Transactions on Fuzzy Systems* 16.4 (2008), págs. 1061-1071 (vid. pág. 17).

[71]Kelly Markelle, Longjohn Rachel y Nottingham Kolby. *The UCI Machine Learning Repository*. 2023 (vid. págs. 48, 71, 102, 124, 160).

[72]Corrado Mencar, Ciro Castiello, Raffaele Cannone y Anna Maria Fanelli. “Design of fuzzy rule-based classifiers with semantic cointension”. En: *Information Sciences* 181.20 (2011), págs. 4361-4377 (vid. pág. 36).

[73]Kevin Michell y Werner Kristjanpoller. “Strongly-typed genetic programming and fuzzy inference system: An embedded approach to model and generate trading rules”. En: *Applied Soft Computing* 90 (2020), pág. 106169 (vid. pág. 23).

[74]Masaharu Mizumoto y Hans-Jürgen Zimmermann. “Comparison of fuzzy reasoning methods”. En: *Fuzzy sets and systems* 8.3 (1982), págs. 253-283 (vid. pág. 23).

[75]Mohammad Monfared, Meghdad Fazeli, Richard Lewis y Justin Searle. “Fuzzy predictor with additive learning for very short-term PV power generation”. En: *Ieee Access* 7 (2019), págs. 91183-91192 (vid. págs. 5, 28).

[76]Alonso Moral, Ciro Castiello, Luis Magdalena y Corrado Mencar. *Explainable fuzzy systems*. Springer, 2021 (vid. págs. 14, 15, 19, 35).

[77]Sreerama K Murthy. “Automatic construction of decision trees from data: A multi-disciplinary survey”. En: *Data mining and knowledge discovery* 2 (1998), págs. 345-389 (vid. pág. 4).

[78]Lekha R Nair, Sujala D Shetty y Siddhanth D Shetty. “Applying spark based machine learning model on streaming big data for health status prediction”. En: *Computers & Electrical Engineering* 65 (2018), págs. 393-399 (vid. pág. 31).

[79]PW Chandana Prasad, Azam Beg y Ashutosh Kumar Singh. “Effect of Quine-McCluskey simplification on Boolean space complexity”. En: *2009 Innovative Technologies in Intelligent Systems and Industrial Applications*. IEEE. 2009, págs. 165-170 (vid. págs. 7, 36, 37).

[80]R Rawat y R Yadav. “Big data: Big data analysis, issues and challenges and technologies”. En: *IOP Conference Series: Materials Science and Engineering*. Vol. 1022. 1. IOP Publishing. 2021, pág. 012014 (vid. págs. 30, 32).

[81]Sara del Rio, Victoria López, José Manuel Benítez y Francisco Herrera. “A mapreduce approach to address big data classification problems based on the fusion of linguistic fuzzy rules”. En: *International Journal of Computational Intelligence Systems* 8.3 (2015), págs. 422-437 (vid. págs. 56, 95, 99, 109).

[82]Steven L Salzberg. *C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993*. 1994 (vid. pág. 4).

[83]Arash Shaban-Nejad, Martin Michalowski, John S Brownstein y David L Buckeridge. “Guest editorial explainable AI: towards fairness, accountability, transparency and trust in healthcare”. En: *IEEE Journal of Biomedical and Health Informatics* 25.7 (2021), págs. 2374-2375 (vid. pág. 35).

[84]Tejal Shah, Fethi Rabhi y Pradeep Ray. “Investigating an ontology-based approach for Big Data analysis of inter-dependent medical and oral health conditions”. En: *Cluster Computing* 18 (2015), págs. 351-367 (vid. pág. 30).

[85]Vladimír Siládi y Tomáš Filo. “Quine-McCluskey algorithm on GPGPU”. En: *AWER-Procedia Information Technology and Computer Science* 4 (2013), págs. 814-820 (vid. pág. 38).

[86]Antonín Svoboda y Donnamaie E White. “Advanced Logical Circuit Design Techniques”. En: (1979) (vid. págs. 38, 45).

[87]Tomohiro Takagi y Michio Sugeno. “Fuzzy identification of systems and its applications to modeling and control”. En: *IEEE transactions on systems, man, and cybernetics* 1 (1985), págs. 116-132 (vid. pág. 56).

[88]Chin Hooi Tan, Keem Siah Yap, Hisao Ishibuchi, Yusuke Nojima y Hwa Jen Yap. “Application of fuzzy inference rules to early semi-automatic estimation of activity duration in software project management”. En: *IEEE Transactions on Human-Machine Systems* 44.5 (2014), págs. 678-688 (vid. pág. 23).

[89]Kazuo Tanaka y B Werners. *An introduction to fuzzy logic for practical applications*. Springer, 1997 (vid. pág. 23).

[90]Fabiano Castro Torrini, Reinaldo Castro Souza, Fernando Luiz Cyrino Oliveira y Jose Francisco Moreira Pessanha. “Long term electricity consumption forecast in Brazil: a fuzzy logic approach”. En: *Socio-Economic Planning Sciences* 54 (2016), págs. 18-27 (vid. pág. 28).

[91]Volkmar Uebele, Shigeo Abe y Ming-Shong Lan. “A neural-network-based fuzzy classifier”. En: *IEEE Transactions on Systems, Man, and Cybernetics* 25.2 (1995), págs. 353-361 (vid. pág. 22).

[92]L-X Wang y Jerry M Mendel. “Generating fuzzy rules by learning from examples”. En: *IEEE Transactions on systems, man, and cybernetics* 22.6 (1992), págs. 1414-1427 (vid. págs. 6, 23).

[93]Li-Xin Wang. “Fast training algorithms for deep convolutional fuzzy systems with application to stock index prediction”. En: *IEEE Transactions on fuzzy systems* 28.7 (2019), págs. 1301-1314 (vid. pág. 28).

[94]Li-Xin Wang. “The WM method completed: a flexible fuzzy system approach to data mining”. En: *IEEE Transactions on fuzzy systems* 11.6 (2003), págs. 768-782 (vid. págs. 5, 28).

[95]Yuangang Wang, Haoran Liu, Wenjuan Jia et al. “Deep Fuzzy Rule-Based Classification System With Improved Wang–Mendel Method”. En: *IEEE Transactions on Fuzzy Systems* 30.8 (2021), págs. 2957-2970 (vid. pág. 5).

[96]Frank Wilcoxon. “Individual comparisons by ranking methods”. En: *Breakthroughs in Statistics: Methodology and Distribution*. Springer, 1992, págs. 196-202 (vid. págs. 50, 76, 105, 127, 133, 142).

[97]Xueming Yang, Jiangye Yuan, Jinsha Yuan y Huina Mao. “An improved WM method based on PSO for electric load forecasting”. En: *Expert Systems with Applications* 37.12 (2010), págs. 8036-8041 (vid. págs. 5, 28).

[98]Lotfi Asker Zadeh. “The concept of a linguistic variable and its application to approximate reasoning—I”. En: *Information sciences* 8.3 (1975), págs. 199-249 (vid. págs. 13, 16, 19).

[99]Yanwei Zhai, Zheng Lv, Jun Zhao, Wei Wang y Henry Leung. “Data-driven inference modeling based on an on-line Wang-Mendel fuzzy approach”. En: *Information Sciences* 551 (2021), págs. 113-127 (vid. pág. 5).

[100]Guoqiang Peter Zhang. “Neural networks for classification: a survey”. En: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 30.4 (2000), págs. 451-462 (vid. pág. 4).

[101]Yong Zhong, Liang Chen, Changlin Dan y Amin Rezaeipanah. “A systematic survey of data mining and big data analysis in internet of things”. En: *The Journal of Supercomputing* 78.17 (2022), págs. 18405-18453 (vid. pág. 3).

[102]Zhi-Hua Zhou. “Rule extraction: Using neural networks or for neural networks?” En: *Journal of Computer Science and Technology* 19.2 (2004), págs. 249-253 (vid. pág. 4).

Anexos

A

Conjunto de datos	Vt	Vc	Ex	Class	MinQM	Inf	QChi	MR
abalone	8	7	4174	28	x	x	x	x
adult	14	6	45222	2	-	x	x	x
appendicitis	7	7	106	2	-	-	-	-
australian	14	7	690	2	x	x	x	x
automobile	25	15	159	6	x	x	-	-
balance	4	0	625	3	x	x	-	-
banana	2	2	5300	2	x	x	x	x
bands	19	19	365	2	x	x	x	x
bupa	20	6	345	2	x	x	x	x
census	41	9	142521	2	-	x	-	x
cleveland	13	13	297	5	x	x	-	x
coil2000	85	2	9822	2	-	x	x	x
connect-4	42	0	67557	3	-	x	-	-
covtype_2_vs_1	54	10	495141	2	-	x	-	x
crx	15	6	653	2	x	x	x	x
ecoli	7	7	336	8	-	-	-	-
fars_0_vs_4_	29	5	62123	2	-	x	-	x
flare	11	0	1066	6	x	x	-	-
german	20	3	1000	2	x	x	-	-
haberman	3	3	306	2	-	-	-	-
heart	13	5	270	2	x	x	-	-
hepmass	28	28	10500000	2	-	x	-	-
higgs	28	28	11000000	2	-	x	-	x
ionosphere	35	32	351	2	x	x	-	-
iris	4	4	150	3	x	x	x	x
kddcup*	41	27	4856150	2	-	x	-	x
led7digit	7	7	500	10	x	x	-	-
letter	16	16	20000	26	-	x	x	x
magic	10	10	19020	2	x	x	x	x
mammographic	5	1	830	2	x	x	x	x

movement	90	90	360	15	x	x	-	-
newthyroid	5	5	215	3	x	x	x	x
page-blocks	10	10	5472	5	x	x	x	x
penbased	16	16	10992	10	x	x	x	x
phoneme	5	5	5404	2	x	x	x	x
pima	8	8	768	2	x	x	x	x
poker*	10	10	946799	2	-	-	-	x
ring	20	20	7400	2	x	x	x	x
saheart	9	8	462	2	x	x	x	x
satimage	50	36	6435	7	-	x	x	x
segment	19	19	2310	7	x	x	x	x
shuttle	9	9	57999	7	x	x	x	x
skin	3	3	245057	2	-	-	-	x
sonar	60	60	208	2	x	x	-	-
spambase	57	57	4597	2	x	x	x	x
spectfheart	44	44	267	2	x	x	-	-
susy	18	18	5000000	2	-	x	-	x
texture	40	40	5500	11	-	x	x	-
thyroid	21	6	7200	3	x	x	x	x
titanic	3	3	2201	2	-	-	-	-
twonorm	20	20	7400	2	-	x	x	x
vehicle	18	18	846	4	x	x	x	x
vowel	13	11	990	11	x	x	x	x
wifi	7	7	2000	4	-	-	-	-
wine	13	13	178	3	x	x	-	-
winequality-red	11	11	1599	11	-	x	x	x
winequality-white	11	11	4898	11	x	x	x	x

Tab. A.1.: Conjuntos de datos consideradas en esta tesis doctoral que pertenecen al repositorio UCI [71], donde (Vt) es el número total de variables, (Vc) son las variables continuas, (Ex) es el número total de ejemplos, (Class) el número de clases, (MinQM) corresponde a los conjuntos de datos utilizados en el Capítulo 3, (Inf) corresponde a los conjuntos de datos utilizados en el Capítulo 4, (QChi) corresponde a los conjuntos de datos utilizados en el Capítulo 5 y (MR) corresponde a los conjuntos de datos utilizados en el Capítulo 6.

Conjunto de Datos	Vt	Vc	Ex	(P_{may}:P_{min})
census	41	13	142,521	(134,359 : 8,162)
cov_1	54	10	581,012	(369,172 : 211,840)
cov_2	54	10	581,012	(297,711 : 283,301)
cov_3	54	10	581,012	(545,258 : 35,754)
cov_7	54	10	581,012	(560,502 : 20,510)
far_Fat	29	5	100,968	(58,852 : 42,116)
far_Inc	29	5	100,968	(85,896 : 15,072)
far_No	29	5	100,968	(80,961 : 20,007)
far_Nin	29	5	100,968	(87,078 : 13,890)
higgs	28	28	11,000,000	(5,829,123 : 5,170,877)
KDD_dos	41	26	4,898,431	(3,883,370 : 1,015,061)
KDD_nor	41	26	4,898,431	(3,925,650 : 972,781)
KDD_prb	41	26	4,898,431	(4,857,329 : 41,102)
KDD_r21	41	26	4,898,431	(4,897,305 : 1,126)
pok_0	10	10	1,025,009	(513,701 : 511,308)
pok_1	10	10	1,025,009	(591,912 : 433,097)
pok_2	10	10	1,025,009	(976,181 : 48,828)
pok_3	10	10	1,025,009	(1,003,375 : 21,634)
skin	3	3	245,057	(194,198 : 50,859)
susy	18	18	5,000,000	(2,712,173 : 2,287,827)

Tab. A.2.: Conjuntos de datos utilizados en el estudio experimental de [27], donde (Vt) es el número total de variables, (Vc) son las variables continuas, (Ex) es el número total de ejemplos y (P_{may}:P_{min}) es el número de ejemplos de la clase mayoritaria y minoritaria. Los conjuntos de datos census, higgs, skin y susy son binarios y, por lo tanto, se utilizaron en su versión original. El resto son multiclase y se transformaron a binarios considerando los ejemplos de una de las clases como la clase positiva y el resto como la clase negativa. Para los conjuntos de datos multiclase, se agregó el identificador de la clase positiva al nombre del conjunto de datos. Para los conjuntos de datos donde los valores asociados a la clase son numéricos, se identifican con el número asociado a la clase que se considera como positiva. Para los demás conjuntos de datos, se realizó la siguiente asociación: para Far, Fat representa la clase Fatal_Injury, Inc representa Incapaciting_Injury, No representa No_Injury y Nin representa Nonincapaciting_Evident_Injury. En el caso de KDD, los cuatro conjuntos de datos considerados son Two, Nor, prb y r21.