Universidad de Granada

Ciencias de la Computación e Inteligencia Artificial

USO DE TÉCNICAS DE APRENDIZAJE PARA CLASIFICACIÓN ORDINAL Y REGRESIÓN

TESIS DOCTORAL

JUAN CARLOS GÁMEZ GRANADOS

septiembre 2017

Editor: Universidad de Granada. Tesis Doctorales
Autor: Juan Carlos Gámez Granados
ISBN: 978-84-9163-720-2
URI: http://hdl.handle.net/10481/48951

Uso de técnicas de aprendizaje para clasificación ordinal y regresión

Programa de Doctorado
Tecnologías de la Información y la Comunicación

Memoria que presenta
Juan Carlos Gámez Granados

Para optar al título de
Doctor en Informática

Dirigida por
Dr. Antonio González Muñoz
y
Dr. F.G.Raúl Pérez Rodríguez

Ciencias de la Computación e Inteligencia Artificial
septiembre 2017

VISTO BUENO

El Prof. Dr. **Antonio González Muñoz** Catedrático de Universidad y el Prof. Dr. **F.G.Raúl Pérez Rodríguez** Titular de Universidad del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada,

INFORMAN:

Que la memoria titulada:

*"**Uso de técnicas de aprendizaje para clasificación ordinal y regresión**"*

ha sido realizada por **D. Juan Carlos Gámez Granados** bajo nuestra dirección en el Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada para optar al título de **Doctor en Informática**.

En Granada, a 18 de septiembre 2017.

Los directores:

Fdo. Antonio González Muñoz Fdo. F.G.Raúl Pérez Rodríguez

COMPROMISO DERECHOS DE AUTOR

El doctorando y los directores de la Tesis "*Uso de técnicas de aprendizaje para clasificación ordinal y regresión*", garantizamos, al firmar esta Tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la Tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, septiembre 2017

Los directores:

Fdo. Antonio González Muñoz Fdo. F.G.Raúl Pérez Rodríguez

El doctorando:

Fdo. Juan Carlos Gámez Granados

Muchos de los fracasos de los hombres
es porque no se dieron cuenta
lo cerca que estaban del éxito cuando desistieron.
Thomas Alva Edison [1]

1 Thomas Alva Edison (1847 - 1931) empresario e inventor estadounidense. Patentó más de mil inventos siendo los más conocidos la lámpara incandescente y el fonógrafo.

AGRADECIMIENTOS

El agradecimiento
es la parte principal
de un hombre de bien.
Francisco de Quevedo y Villegas [2]

Con el ánimo de no dejarme a nadie atrás, que seguro que lo hago, comienzo estas palabras de agradecimiento.

En primer lugar quiero agradecer su paciencia infinita y su "estar ahí" de mis directores. Antonio por su sabiduría y su buena guía desde que comencé con él mi primer trabajo académico dando lugar al proyecto fin de carrera. Raúl al que conocí algo más tarde del que también he aprendido mucho en menos tiempo. A los dos con los que he compartido, y seguiremos compartiendo, muchos momentos tanto profesionales como personales.

En segundo lugar me gustaría agradecer a los compañeros del Dpto. de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada por su recibimiento, cafés, almuerzos, sobremesas y demás momentos compartidos en Granada.

He de agradecer todo lo aprendido en el terreno profesional y personal a mis compañeros del Dpto. de Arquitectura de Computadores, Electrónica y Tecnología Electrónica de la Universidad de Córdoba. También de la Universidad de Córdoba, agradecer a los compañeros del Dpto de Informática y Análisis Numérico con los que compartimos mucho más que titulación y alumnos. No puedo mencionar a nadie por la extensión que ello conllevaría y la certeza de olvidar a alguien. Cada uno conoce lo aportado en lo que soy y tiene mi agradecimiento día a día que pienso que es lo más importante.

Por supuesto, aunque no está en el orden que debería, tengo que agradecer a mis padres, hermano y familia su apoyo incondicional, que, aún sin conocer en detalle todo lo que conlleva mi trabajo, siempre se interesan por él y están ahí conmigo.

Por último, todo esto no podría ser posible sin mi esposa Noelia, mi media naranja, compañera inseparable de viaje, con la que he compartido más de media vida y espero seguir compartiendo

2 Francisco de Quevedo y Villegas (1580 - 1645) escritor español del Siglo de Oro.

más de la otra media que nos queda; que ha sido capaz de darme cosas imposibles e impensables como nuestras dos hijas Noelia y Carla que forman el equipo gracias al cual he sido capaz de estar aquí y escribir esta tesis. Juntos hemos conseguido muchos objetivos y muchos más que tenemos que conseguir.

A todos los mencionados y los no mencionados que me han ayudado en cada momento.

A todos los que yo llamo MI GENTE!!!!

RESUMEN

Desocupado lector: sin juramento me podrás creer que quisiera que este libro, como hijo del entendimiento, fuera el más hermoso, el más gallardo y más discreto que pudiera imaginarse.
Miguel de Cervantes, Don Quijote de la Mancha [3]

Con la cita con la que comienzo este resumen no pretendo más que expresar el entusiasmo, dedicación y esfuerzo que he puesto para realizar esta tesis y mi deseo de que al lector le parezca interesante y amena.

El **Aprendizaje** es un concepto muy amplio que puede ser aplicado en muy diversos ámbitos. Según la RAE, en su acepción más simple, es la "acción y efecto de aprender algún arte, oficio u otra cosa". Para *aprender* se necesita cuatro factores fundamentales: inteligencia, conocimientos previos, experiencia y motivación. Aquí aparece otro concepto muy importante como es la **Inteligencia**. Entre las acepciones de inteligencia que comprende la RAE encontramos "la capacidad de resolver problemas". Hay problemas de muy diversos tipos. Una de las posibles taxonomías de problemas es la que los divide en *Problemas de Clasificación Nominal* y *Problemas de Regresión*, apareciendo otra categoría que puede considerarse intermedia y es conocida como *Problemas de Clasificación Ordinal* o *Problemas de Regresión Ordinal*.

Debido al vertiginoso avance de la tecnología y los computadores, cada vez es más común la resolución de problemas utilizando estos medios, aplicando estos conceptos a las máquinas y siendo el germen de lo que hoy se conoce como ***Aprendizaje Automático***. El *Aprendizaje Automático* es una rama de la ***Inteligencia Artificial*** cuyo objetivo, de forma genérica, es el desarrollo de técnicas que permitan a las computadoras aprender. De esta forma, esta tesis presenta el "uso de técnicas de aprendizaje para la resolución de problemas de clasificación ordinal y regresión".

3 Miguel de Cervantes Saavedra (1547 - 1616) soldado, novelista, poeta y dramaturgo español. Máxima figura de la literatura española y universalmente conocido por su obra Don Quijote de la Mancha.

Por ello, se comienza con la introducción de los conceptos y técnicas que se usarán a lo largo de la misma, como son: *Aprendizaje Automático*, *Lógica Difusa*, *Algoritmos Genéticos* y *Aprendizaje de Reglas Difusas*, así como los diferentes tipos de problemas expuestos anteriormente y las métricas más usadas en los mismos.

A continuación, se presenta el algoritmo de clasificación nominal que será usado como base, este es, NSLV. Se exponen los principales elementos del mismo que son susceptibles de ser modificados en el desarrollo del algoritmo para problemas de clasificación ordinal.

Posteriormente, se expone la extensión del algoritmo base para clasificación ordinal junto con las modificaciones realizadas sobre el mismo, así como su desarrollo en un conjunto de datos representativos de este tipo de problemas.

Finalmente, se presenta la modificación del algoritmo de clasificación ordinal expuesto anteriormente para abarcar problemas de regresión. Se exponen los cambios realizados así como el comportamiento del mismo con una muestra representativa de datos de este tipo de problemas. Debido al uso como base de un algoritmo de clasificación ordinal, adicional al valor real de salida, se presenta la posibilidad de incorporar un posible rango de error en el resultado.

Como apéndices se muestra de forma breve algunas de las ideas y pruebas realizadas adicionales a las utilizadas en el desarrollo de los algoritmos, junto con los resultados obtenidos. Esto se presenta con la idea de posibilitar un seguimiento del razonamiento llevado a cabo en el desarrollo de estos algoritmos y al mismo tiempo proporcionar al lector una pequeña muestra de los resultados del proceso ensayo/error permitiendo agilizar su labor de investigación en caso de estar interesado en este ámbito.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1

INTRODUCCIÓN

Lo importante es no dejar de hacerse preguntas.
Albert Einstein [1]

ÍNDICE

1 Albert Einstein (1879 - 1955) físico alemán de origen judío nacionalizado suizo y estadounidense. Es considerado como el científico más conocido y popular del siglo XX.

1.1 MOTIVACIÓN

El Aprendizaje Automático (AA) es una de las ramas de investigación más populares de la Inteligencia Artificial (IA). El aprendizaje inductivo es un tipo de aprendizaje automático que tiene como objetivo el desarrollo, de manera automática, de modelos que aprendan de una serie de datos y proporcionen una respuesta sin intervención humana. Estos modelos pueden estar basados en muy diversas técnicas o métodos que permitan este aprendizaje. Entre estos métodos se encuentran los Algoritmos Genéticos (AGs) que utilizan las bases de la evolución biológica para llevar a cabo el aprendizaje. Este aprendizaje puede realizarse de diversa formas, entre las se encuentra el uso de la estrategia de recubrimiento secuencial en la cual el aprendizaje se realiza mediante el cubrimiento de los datos de forma iterativa. Las aplicaciones del AA abarcan áreas como la robótica [2, 32, 42, 50, 51], la medicina [29, 105] o la economía [75, 106, 120], entre otras muchas.

El principal objetivo del aprendizaje inductivo es obtener modelos que, de manera lo más precisa posible, sean capaces de predecir el valor esperado de salida para una entrada determinada. Junto con la precisión, otro factor importante es la interpretabilidad [40, 83] ya que, además de buscar una precisión del sistema, en muchos problemas es deseable que el resultado sea un modelo comprensible, es decir, de fácil interpretación por un ser humano. Para obtener tales modelos interpretables se hace uso de un conjunto de reglas de la forma "SI (antecedente) ENTONCES (consecuente)" basadas en la lógica de proposiciones. El consecuente es el valor de salida, y el antecedente es una expresión proposicional, que puede estar formada por un conjunto de conjunciones de términos o ser simplemente un término. Con el objetivo de obtener un modelo más comprensible se puede utilizar la Lógica Difusa (LD) [136, 137, 138] creando así un conjunto de reglas en las que el antecedente y el consecuente pueden estar formados por variables o términos basados en esta lógica.

Es muy importante la tarea de predecir el valor de una variable de respuesta. Esta variable respuesta puede ser un valor entre dos o múltiples categorías, lo que da lugar a los *problemas de clasificación nominal*, o también puede ser un valor continuo en la recta real, definiendo así los *problemas de regresión*. Entre estos dos tipos de problemas se encuentra los *problemas de clasificación*

ordinal, también llamados *problemas de regresión ordinal*, que son aquellos que, teniendo como salida un valor perteneciente a una categoría, los valores posibles de dichas categorías tienen un orden establecido entre ellos.

Los *problemas de regresión* son muy conocidos y estudiados en la literatura desde hace años [8, 34, 72, 73], sin embargo, los *problemas de clasificación ordinal* son relativamente nuevos, aunque el número de trabajos relacionados con este tipo de problemas va en aumento en los últimos años a nivel internacional por sus múltiples aplicaciones [69, 89, 105, 106]. Los *problemas de clasificación ordinal* son un tipo de problema de aprendizaje que se encuentra situado entre la clasificación nominal y la regresión. De la primera se diferencia en que existe un orden preestablecido entre las clases mientras que de la regresión se distingue en que el conjunto de etiquetas es finito y las diferencias entre los valores de las etiquetas no están definidas. Para ello, el clasificador ordinal debe explotar el conocimiento a priori sobre la distribución de patrones en el espacio de entrada [80]. Además, la evaluación del clasificador debe ser diferente de las consideradas en los clasificadores clásicos debiendo establecer medidas específicas [30, 31, 33].

Hay diferentes algoritmos que utilizan las técnicas, métodos y estrategias mencionados en los párrafos anteriores para abordar problemas de aprendizaje. NSLV [58, 65] es, en concreto, un ejemplo de estos algoritmos de aprendizaje, es un algoritmo de aprendizaje basado en un modelo genético iterativo [17, 62, 65, 103] que hace uso de AGs y LD para problemas de clasificación nominal. Este algoritmo se tomará como base para los desarrollos que se exponen en esta memoria.

La idea original de esta tesis es la creación de un algoritmo al que llamaremos NSLV Ordinal (NSLVOrd). Este algoritmo será una extensión de NSLV para abordar problemas de clasificación ordinal. Adicional a esta idea principal se propone una primera aproximación para abordar problemas de regresión a la que llamaremos NSLV Regresión (NSLVReg). Esta aproximación se basa en la consideración de que la clasificación ordinal se puede entender como un paso previo a la regresión. Para ello se realizarán las modificaciones necesarias sobre NSLVOrd con el objetivo del desarrollo de NSLVReg para abordar problemas de regresión. Por ello, el objetivo de esta tesis es doble, por una parte se pretende la extensión de un modelo genético iterativo (NSLV) para posibilitar el aprendizaje de problemas de clasifica-

ción ordinal. Por otra parte se propone una aproximación para problemas de regresión utilizando el algoritmo de clasificación ordinal, resultado de la consecución del primer objetivo, de forma que se exploten las similitudes entre los problemas ordinales y los problemas de regresión, gracias a lo cual, adicionalmente al valor de salida real se puede incorporar un posible rango de error en el resultado.

1.2 OBJETIVOS

El objetivo principal de este trabajo es *la extensión de un modelo genético iterativo a problemas de clasificación ordinal y regresión*. Este modelo genético iterativo debe realizarse de forma que se obtenga una solución comprensible para lo que se hará uso de la lógica difusa.

Se puede entender que hay dos objetivos principales. Por una parte tenemos la extensión a problemas de clasificación ordinal. Por otra parte nos encontramos con la extensión de la solución sobre problemas de clasificación ordinal para su aplicación a problemas de regresión.

Para la consecución del objetivo relacionado con la clasificación ordinal, tendremos que ir alcanzando los siguientes subobjetivos.

- Estudio de similitudes y diferencias entre clasificación nominal y ordinal para la posible solución de problemas de clasificación ordinal mediante algoritmos de clasificación nominal.
- Estudio del algoritmo de clasificación nominal a usar como base (NSLV) y la posibilidad de extensión a problemas de clasificación ordinal.
- Estudio de las métricas para problemas de clasificación ordinal.
- Desarrollo de un algoritmo para resolver problemas de clasificación ordinal (NSLVOrd).

En primer lugar se realizará un estudio sobre la posible aplicación de algoritmos de clasificación nominal para abordar problemas de clasificación ordinal donde se estudiarán sus peculiaridades. En segundo lugar se analizará y estudiará en profundidad

el algoritmo que usamos de base (NSLV) y su posible aplicación a problemas de clasificación ordinal. En tercer lugar se realizará un estudio sobre las métricas usadas en este tipo de problemas y su posible uso tanto para su evaluación como para el guiado del proceso de aprendizaje. Por último se implementará la extensión del algoritmo para problemas de clasificación ordinal como resultado del planteamiento y pruebas de distintas alternativas. Como resultado se publicará el algoritmo correspondiente de la propuesta final a este tipo de problemas.

En lo que respecta al objetivo de la regresión, para su consecución, se hace necesario alcanzar los siguientes subobjetivos.

- Estudio de similitudes y diferencias entre clasificación ordinal y regresión para la posible solución de problemas de regresión mediante algoritmos de clasificación ordinal.
- Estudio de la extensión y posibilidades del algoritmo de clasificación ordinal (NSLVOrd) para problemas de regresión.
- Estudio de las métricas para problemas de regresión.
- Desarrollo del algoritmo final para resolver problemas de regresión (NSLVReg).

En primer lugar se realizará un estudio de la posibilidad, ventajas e inconvenientes de uso de un algoritmo de clasificación ordinal para abordar problemas de regresión, junto con un breve estudio y clasificación de los algoritmos de regresión para la categorización de nuestra propuesta y la comparación de resultados. En segundo lugar, y relacionado con el punto anterior, se realizará un estudio de las diferentes alternativas de diseño, implementación y uso del algoritmo de clasificación ordinal usado de base, de forma que se pueda solucionar un problema de regresión haciendo uso de las similitudes con los problemas de clasificación ordinal. Finalmente se deberá realizar un desarrollo de la propuesta final después de la evaluación de las distintas opciones y alternativas para abordar problemas de regresión usando como base un algoritmo de clasificación ordinal. Al igual que en el caso de clasificación ordinal, el algoritmo correspondiente a la propuesta final para abordar este tipo de problemas estará disponible para su uso.

1.3 ESTRUCTURA DE CAPÍTULOS

A continuación se describirá brevemente la organización por capítulos de esta memoria de Tesis.

En primer lugar, en el capítulo 2 se presenta una serie de conceptos necesarios para una mejor comprensión del trabajo completo. En este capítulo se tratan los conceptos de *Aprendizaje Inductivo Supervisado* junto con una breve introducción al Aprendizaje Automático (AA), Algoritmos Genéticos (AGs), Lógica Difusa (LD) y Sistemas Basados en Reglas Difusas (SBRDs). Igualmente se definen los problemas de clasificación nominal, clasificación ordinal y regresión junto con la presentación de los algoritmos usados para problemas de clasificación ordinal y regresión que serán usados en las comparativas con nuestra propuesta. Finalmente se presentan las métricas más usadas en clasificación, regresión y clasificación ordinal junto con un ejemplo de uso de las mismas con el objetivo de un mejor entendimiento de la combinación propuesta para el guiado del aprendizaje de los capítulos posteriores.

El capítulo 3 presenta el algoritmo base NSLV usado como punto de partida para el desarrollo de los algoritmos que permiten abordar problemas de clasificación ordinal y regresión. En este capítulo se realiza una breve explicación del algoritmo centrando la atención en los elementos que se modificarán en los capítulos posteriores.

El capítulo 4 presenta la extensión del modelo de aprendizaje utilizado por NSLV a problemas de clasificación ordinal. En él se exponen las modificaciones necesarias sobre NSLV para abordar con éxito los problemas de clasificación ordinal. Finalmente se realiza una serie de pruebas y resultados que muestran el buen comportamiento de la propuesta junto con las conclusiones del capítulo.

El capítulo 5 presenta una aproximación a problemas de regresión basada en las similitudes entre problemas de clasificación ordinal y regresión. Para ello, en este capítulo se presentan las modificaciones sobre NSLVOrd, así como el preprocesamiento necesario, esquema de uso de éste y combinación de resultados necesarios para abordar problemas de regresión. Finalmente se realiza una serie de pruebas y resultados con los algoritmos de la categoría a la que pertenece la propuesta, así como con una muestra representativa del resto de algoritmos para problemas

de regresión donde se muestra el buen comportamiento de la propuesta, finalizando el capítulo con las conclusiones.

Finalmente el capítulo 6 presenta las conclusiones derivadas de esta tesis así como las futuras vías de investigación abiertas a raíz de este trabajo.

1.4 PUBLICACIONES

En esta sección se presentan las principales aportaciones relacionadas con la temática de este trabajo.

- Gámez, Juan Carlos; García, David; González, Antonio y Pérez, Raúl: «Ordinal Classification based on the Sequential Covering Strategy». International Journal of Approximate Reasoning, 2016, 76, pp. 96 - 110. ISSN 0888-613X. doi: http://dx.doi.org/10.1016/j.ijar.2016.05.002. http://www.sciencedirect.com/science/article/pii/S0888613X16300706

- Gámez, Juan Carlos; García, David; González, Antonio y Pérez, Raúl.: «Un algoritmo de clasificación ordinal basado en el modelo de recubrimiento secuencial». En: Actas de la XVI Conferencia de la Asociación Española para la Inteligencia Artificial, pp. 529 - 538, 2015. http://simd.albacete.org/actascaepia15/

A continuación se presentan aportaciones adicionales cuya temática sobre imágenes y aprendizaje será usada en trabajos futuros expuestos en el capítulo 6.

- Gámez, Juan Carlos; González, Antonio; Pérez, Raúl y García-Silvente, Miguel: «Uso de técnicas de preprocesamiento de imágenes y aprendizaje para la detección de cambios de atención de una persona en procesos de interacción persona-robot». En: Actas ROBOT 2011, pp. 389 - 396, 2011.

- Gámez, Juan Carlos; González, Antonio y Pérez, Raúl: «Un modelo difuso para la estimación de la atención en procesos de interacción robot-persona». XI Workshop of Physical Agents 2010, 2010, pp. 169 - 176.

Finalmente se presentan aportaciones adicionales cuya temática relacionada con interpretabilidad y big data también pertenecen a trabajos futuros expuestos en el capítulo 6.

- Gámez, Juan Carlos; García, David; González, Antonio y Pérez, Raúl.: «Un estudio preliminar sobre el uso de técnicas de aprendizaje incremental para problemas de big data». En: Libro de Resúmenes del XVIII Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF 2016), pp. 176 - 177, 2016. ISBN: 978-84-608-7381-5.

- Gámez, Juan Carlos; García, David; González, Antonio y Pérez, Raúl: «On the Use of an Incremental Approach to Learn Fuzzy Classification Rules for Big Data Problems». En: Fuzzy Systems (FUZZ-IEEE), 2016 IEEE International Conference on, , 2016.

- García, David; Gámez, Juan Carlos; González, Antonio y Pérez, Raúl: «Using a sequential covering strategy for discovering fuzzy rules incrementally». En: Fuzzy Systems (FUZZ-IEEE), 2015 IEEE International Conference on, pp. 1 - 8, 2015.
  doi: 10.1109/FUZZ-IEEE.2015.7337924.

- García, David; Gámez, Juan Carlos; González, Antonio y Pérez, Raúl: «An interpretability improvement for fuzzy rule bases obtained by the iterative rule learning approach». International Journal of Approximate Reasoning, 2015, 67, pp. 37 - 58. ISSN 0888-613X.
  doi: 10.1016/j.ijar.2015.09.001.
  http://www.sciencedirect.com/science/article/pii/S0888613X15001425

2

ANTECEDENTES Y PRELIMINARES

A los hombres les encanta maravillarse.
Esto es la semilla de la ciencia.
Ralph Waldo Emerson [1]

ÍNDICE

1 Ralph Waldo Emerson (1803 - 1882) escritor, filósofo y poeta estadounidense. Sus enseñanzas contribuyeron al desarrollo del movimiento del "Nuevo Pensamiento" a mediados del siglo XIX.

2.1 APRENDIZAJE INDUCTIVO SUPERVISADO

2.1.1 *Introducción al Aprendizaje Automático (AA)*

El Aprendizaje Automático (AA) [7] es una de las áreas de investigación más activas de la IA [108, 115]. Se puede definir como el estudio de los algoritmos de computadores que mejoran automáticamente a partir de la experiencia [100]. Se puede decir que el AA es una rama de la IA cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. Este aprendizaje se puede llevar a cabo mediante un proceso inductivo del conocimiento, dando lugar a lo que se conoce como Aprendizaje Inductivo. El Aprendizaje Inductivo consiste en establecer enunciados universales ciertos a partir de la experiencia, esto es, ascender de forma lógica a través del conocimiento científico, desde la observación de los fenómenos o hechos de la realidad a la ley universal que los contiene. Cuando el conocimiento se suministra como un conjunto de hechos o fenómenos en los que se conoce tanto las características que lo componen como la clasificación o resultado de este hecho o fenómeno se denomina Aprendizaje Supervisado. Cuando están presentes las características de ambos conceptos en el mismo proceso de aprendizaje estamos ante un *Aprendizaje Inductivo Supervisado*, como es el caso que nos ocupa.

Otro aspecto a tener muy presente en el aprendizaje es el equilibrio entre *precisión en la predicción* y la *interpretabilidad del modelo*. En función del problema a aprender podremos necesitar un método más restrictivo pero que nos dé como resultado una mayor precisión en la predicción aunque esto suponga que el modelo resultante no sea interpretable. Por contra, en otras ocasiones, podremos necesitar un modelo que sea fácilmente interpretable con unos resultados aceptables aunque no sean tan precisos como en el modelo anterior. En las propuestas presentadas en esta tesis se valora la interpretabilidad sin dejar de lado la precisión. Para ello se va a realizar un aprendizaje de reglas mediante las cuales se posibilite la explicación del comportamiento del sistema a aprender.

El AA tiene una amplia gama de aplicaciones, incluyendo buscadores web [84, 98], medicina [29, 105] y diagnósticos médicos

[18, 85], detección de fraude en el uso de tarjetas de crédito [94], análisis del mercado de valores [75], entre otros.

La figura 2.1 presenta un esquema frecuentemente utilizado en el aprendizaje inductivo supervisado. Uno de los elementos principales de este esquema es el algoritmo de aprendizaje, el cual tiene como entrada un conjunto de ejemplos y genera como salida el modelo predictivo. Otro elemento principal es la representación del conocimiento que condicionará el modelo predictivo, el cual, ante una entrada responderá con un valor de salida esperada para esa entrada. En las secciones siguientes nos centraremos en el algoritmo de aprendizaje y la representación del conocimiento.

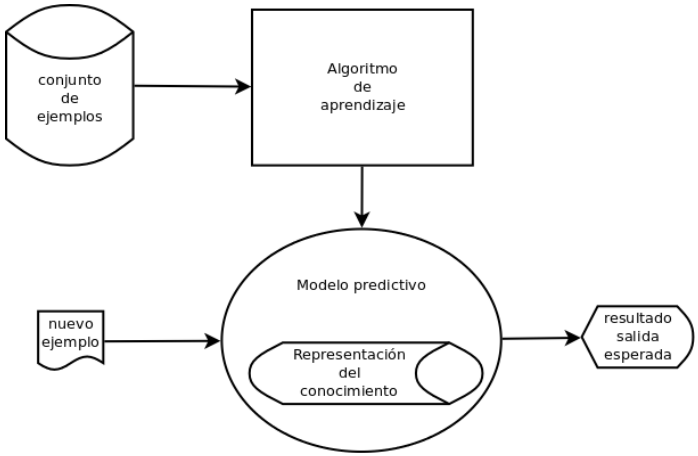


Figura 2.1: Esquema de aprendizaje inductivo supervisado

2.1.2 *Descripción de algunos algoritmos de aprendizaje*

Hay diferentes enfoques o técnicas a la hora de llevar a cabo el aprendizaje inductivo supervisado entre las que destacamos los árboles de decisión, las redes neuronales artificiales, las máquinas de vectores soporte o los algoritmos genéticos entre otros, cuyos ejemplos representativos son usados en las comparativas de los capítulos de las propuestas presentadas en esta tesis.

La técnica de **Árboles de Decisión** es una de las técnicas más simples pero con más éxito del AA. Los árboles de decisión clasifican los ejemplos ordenándolos desde la raíz a los nodos hojas

en donde se produce la clasificación del ejemplo. Cada nodo del árbol especifica una prueba sobre un atributo del ejemplo. Cada ramificación descendiente de ese nodo corresponde a uno de los posibles valores de este atributo. Un ejemplo se clasifica comenzando en el nodo raíz del árbol, probando cada atributo especificado en el nodo y bajando en la ramificación del árbol correspondiente al valor del atributo del ejemplo. A partir de este árbol se puede obtener el conjunto de reglas con lo que se puede entender que se realiza un aprendizaje de reglas mediante esta traducción del árbol de decisión obtenido. La figura 2.2 muestra el ejemplo típico de decisión de jugar o no en función de las condiciones meteorológicas. El algoritmo CART (Classification And Regression Tree), disponible en Keel[2] [14], es una implementación de esta técnica usada tanto para clasificación, como para regresión como su propio nombre indica. Este algoritmo será utilizado en la comparación de resultados de un problema de regresión en el capítulo 5.

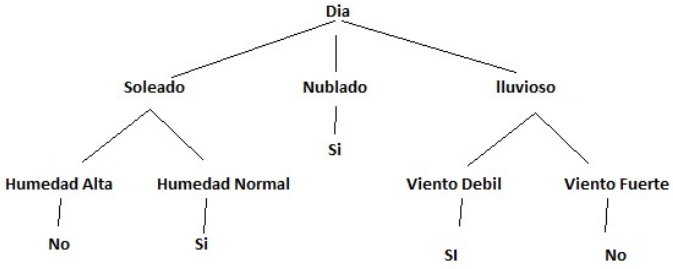


Figura 2.2: Ejemplo de árbol de decisión de juego

La técnica de **Redes Neuronales Artificiales** está basada en la función biológica del cerebro humano [128]. Se trata de un sistema de enlaces de neuronas que colaboran entre sí para producir un estímulo de salida. Las redes neuronales artificiales están formadas por un conjunto de unidades simples pero densamente conectadas donde cada unidad toma un valor de entrada y produce un valor de salida. Estas unidades, llamadas *neuronas*, suelen estar agrupadas por capas donde la salida de cada unidad está conectada a la entrada de la unidad de la capa siguiente. Las conexiones tienen pesos numéricos que se adaptan según la experiencia realizando

2 http://www.keel.es

de esta forma el aprendizaje. La figura 2.3 presenta la estructura general de una red neuronal. El algoritmo Ensemble-R, disponible en Keel[3] [59], que será usado en la comparación de resultados en el capítulo 5, es un ejemplo de implementación de esta técnica para problemas de regresión.

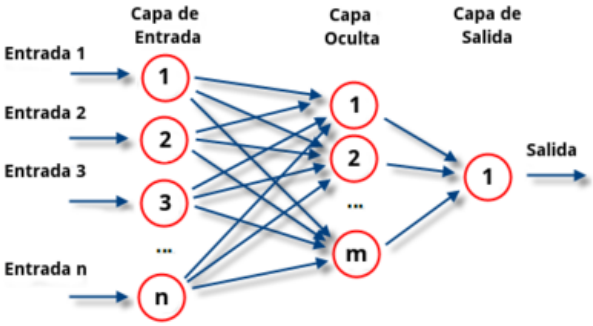


Figura 2.3: Estructura genérica de una red neuronal

Otra técnica muy usada es la técnica de **Máquinas de Vectores Soporte** (SVM en Inglés). Como características podemos destacar que construye un separador de máximo margen, es decir, unos límites de decisión con la mayor distancia posible. Para ello utiliza un conjunto de funciones matemáticas llamadas *kernels* que permiten una transformación de los valores de los ejemplos en otras dimensiones. La figura 2.4 muestra el efecto de la aplicación de los kernels para conseguir la máxima separación entre los ejemplos. Podemos encontrar distintas implementaciones de esta técnica tanto para problemas de clasificación nominal (SVC) como adaptaciones para clasificación ordinal (C-RNK, SVOR-IM), disponibles en LIBSVMLibrary[4] [20] que utilizaremos en la comparación de resultados en el capítulo 4. También encontramos otras implementaciones para problemas de regresión como el algoritmo NU_SVR-R, disponible en Keel[3] [41] que también será usado en la comparación de resultados en el capítulo 5.

Los **Algoritmos Genéticos (AGs)** son procesos de búsqueda heurística que simulan la selección natural. Los AGs son un tipo de algoritmos de optimización muy usados para aprendizaje en

3 http://www.keel.es
4 http://www.csie.ntu.edu.tw/~cjlin/libsvm

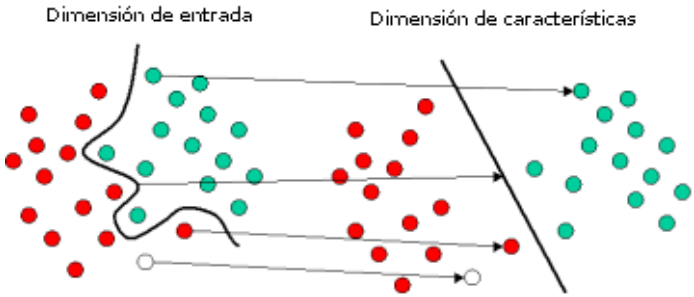


Figura 2.4: Efecto de aplicación de los kernels en Máquinas de vectores soporte

la actualidad. Los AGs proporcionan un método de aprendizaje basados en una analogía a la evolución biológica [100]. Siguiendo la terminología de la teoría de la evolución, las entidades que representan las soluciones al problema se denominan *individuos* o *cromosomas*, y el conjunto de éstos, *población* del AG. Los individuos son modificados por *operadores genéticos* tales como: *cruce*, que consiste en la mezcla de la información de dos o más individuos; *mutación*, que es un cambio aleatorio en la información de los individuos; y *selección*, consistente en la elección de los individuos que sobrevivirán y conformarán la población de la siguiente generación. El AG comienza con una población aleatoria y va obteniendo los mejores individuos aplicando los operadores genéticos de forma similar a como ocurre en la naturaleza. El aprendizaje consiste en buscar, dentro de un conjunto de hipótesis candidatas, la mejor de ellas. Para identificar la mejor hipótesis es necesario definir una medida para el problema en cuestión. A esta medida se le conoce como *función fitness*. La función fitness puede ser definida como la precisión de la hipótesis sobre el conjunto de entrenamiento.

Los AGs se han utilizado para el aprendizaje de reglas difusas con diferentes enfoques. En primer lugar, encontramos el *Enfoque Pittsburgh* [123] donde un cromosoma representa una base de reglas al completo. En cada iteración se modifica el conjunto completo de reglas. El cruce permite la combinación de reglas y la mutación genera nuevas reglas. Por otra parte, se tiene el *Enfoque Michigan* [76] donde un cromosoma representa una regla y la población completa representa la base de reglas. El conjunto

de reglas se modifica en cada iteración. Las operaciones de cruce y mutación se realizan a nivel de individuo, es decir, a nivel de regla. Por último se encuentra el *enfoque de aprendizaje de reglas iterativo* el cual está basado en la *estrategia de recubrimiento secuencial* [100].

El enfoque de aprendizaje de reglas iterativo trata de combinar las mejores características de los enfoques de Michigan y Pittsburgh ya que por una parte, cada individuo representa una regla siendo la población el conjunto de reglas (enfoque Michigan), mientras que por otra parte en cada iteración del AG se van añadiendo reglas modificándose el conjunto completo de reglas (enfoque Pittsburgh). Utilizando este enfoque como base, la estrategia de recubrimiento secuencial lleva a cabo el aprendizaje de forma iterativa de manera que en cada iteración se aprende una regla y, de alguna manera, se "penalizan" los ejemplos que están cubiertos por la regla aprendida a la hora del aprendizaje de la siguiente regla.

El esquema básico de la estrategia de recubrimiento secuencial es el mostrado en el algoritmo 1.

Algoritmo 1 Estrategia de Recubrimiento Secuencial

1. Ejecutar el algoritmo genético para obtener una regla
2. Añadir la regla al final del conjunto de reglas
3. Penalizar los ejemplos cubiertos por esta regla
4. Si el conjunto de reglas obtenido es adecuado para representar el conjunto de ejemplos de entrenamiento el algoritmo termina dando como resultado el conjunto de reglas. En otro caso, volver al paso 1

Para implementar un algoritmo de aprendizaje de reglas basado en un algoritmo genético que utiliza el enfoque de aprendizaje de reglas iterativo es necesario definir:

- Un *criterio para seleccionar la mejor regla* en cada iteración del algoritmo genético. Este criterio suele estar asociado con una o varias características que miden la bondad de una regla. Un ejemplo de estas características puede ser la consistencia, la completitud, la simplicidad, etc.

- Un *criterio de penalización*. Este criterio permite la no consideración de los ejemplos cubiertos por las reglas aprendidas para que pueda seguir avanzando el aprendizaje y aprender el resto de los ejemplos. La forma más fácil de penalizar los ejemplos cubiertos por una regla es eliminarlos del conjunto de ejemplos de entrenamiento.

- Un *criterio de parada* que determine cuándo se han obtenido las reglas que definen de forma adecuada el sistema a modelar.

2.1.3 *Introducción a la Lógica Difusa*

La Lógica Difusa (LD) es una rama de la IA que le permite a una computadora analizar información del mundo real en una escala entre lo verdadero y lo falso [108]. La LD, introducida por L.A. Zadeh en 1965 [136, 137, 138], es una extensión de la lógica tradicional (Booleana) que utiliza conceptos de pertenencia de conjuntos más parecidos a la manera de pensar humana. La principal motivación para utilizar LD es que el lenguaje natural maneja conceptos no precisos como "hace frío" o "la aceleración es alta" que al traducirse del lenguaje humano al contexto de la lógica clásica se produce una pérdida en la riqueza del significado que puede ser importante. Para ello se crea el concepto de conjunto difuso como una generalización del conjunto exacto (crisp en Inglés) tradicional para representar matemáticamente esta imprecisión.

Si tomamos como ejemplo la variable aceleración mostrada en la figura 2.5 se observa que para un valor de 4000 Revoluciones Por Minuto (RPM) se podría tomar dentro del conjunto de aceleración 'baja', aunque también se podría decir que forma parte del conjunto de aceleración 'media' en cierto grado. De esta forma se define un *conjunto difuso* como un conjunto que puede contener elementos de forma parcial. Es decir que la propiedad $x \in A$ puede ser cierta con un grado de verdad. Se mide esta posibilidad de pertenecer (o pertenencia) con un número $\mu_A(x)$, entre 0 y 1, llamado *grado de pertenencia* de x a A. Si es 0, $x \notin A$, si es 1, entonces $x \in A$ totalmente, y si $0 < \mu_A(x) < 1$, $x \in A$ de una manera parcial (figura 2.6).

La figura 2.7 muestra algunos de los principales conceptos de los conjuntos difusos. El *núcleo* de un conjunto difuso A es el

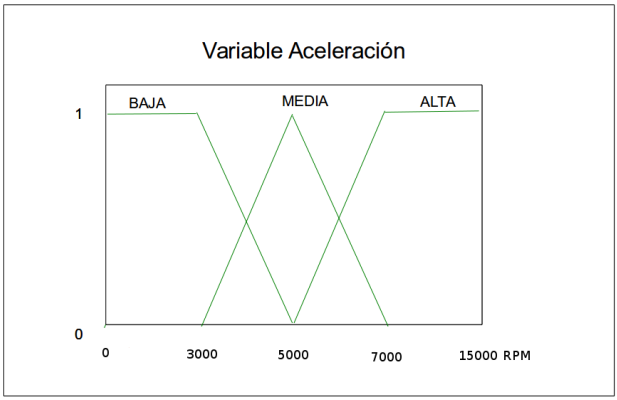


Figura 2.5: Ejemplo de variable lingüística "Aceleración"

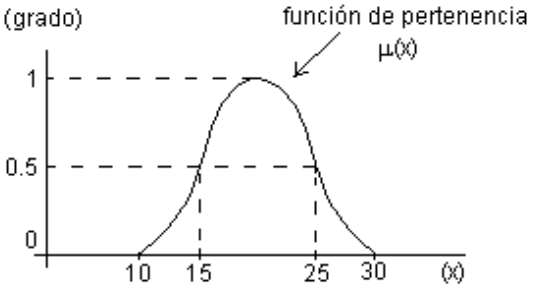


Figura 2.6: Grado de Pertenencia $\mu_A(x)$

conjunto de los elementos x que pertenecen totalmente a A, es decir que verifican $\mu_A(x) = 1$. El *soporte* de conjunto difuso A es el conjunto de los x que pertenecen, en cierta medida, a A. Es decir que verifican $\mu_A(x) > 0$. El *alfa-corte* de un conjunto difuso es el conjunto de los x que pertenecen a A en una medida mayor que α. Aunque depende de la distribución de las etiquetas sobre el universo de discurso, lo habitual, es que los valores del núcleo de la etiqueta representen al conjunto de valores que sin ambigüedad pertenecen a esta etiqueta y no a otras, mientras que el resto de valores del soporte que no están en el núcleo de ninguna etiqueta tengan posibilidad > 0 de pertenecer a más de una etiqueta.

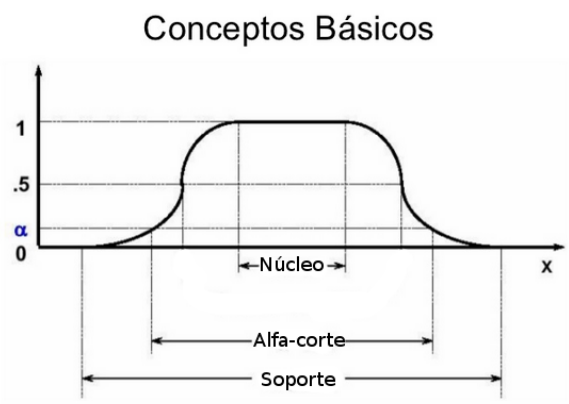


Figura 2.7: Algunos conceptos de los subconjuntos difusos

Estos conjuntos difusos también son llamados *términos lingüísticos* o *etiquetas lingüísticas*. El conjunto de estas etiquetas lingüísticas forman el dominio de lo que se conoce como *variable lingüística*. En el ejemplo de la variable lingüística 'aceleración' se observa que dicha variable está compuesta por tres etiquetas lingüísticas: baja, media y alta. El dominio de esta variable lingüística va de 0 a 15000 RPM. Las etiquetas lingüísticas baja y alta están en los extremos y tienen una función de pertenencia trapezoidal mientras que la función de pertenencia de la etiqueta media es triangular. El núcleo de la etiqueta baja, por ejemplo, se puede representar por el intervalo [0,3000], pudiéndose expresar su soporte también mediante el intervalo [0,5000]. De forma similar, el núcleo de la etiqueta alta está en el intervalo [7000,15000] mientras que el soporte está en [5000,15000]. En el caso de la etiqueta media, el intervalo del núcleo es un punto correspondiente a 5000 RPM ya que la función de pertenencia es triangular, siendo el soporte el intervalo [3000,7000].

En general, no se puede asegurar que el núcleo, ni el soporte de una etiqueta se pueda representar mediante un único intervalo. Sin embargo, en esta memoria definimos los dominios difusos de tal forma que se mantiene la propiedad de que tanto el núcleo como el soporte se pueden representar mediante un único intervalo en las variables cuyo universo de discurso se mueva sobre la recta real. Con esta idea y con el solapamiento que existe entre

los soportes de las etiquetas adyacentes, se pueden definir unos intervalos en la variable lingüística de salida que se utilizarán para abordar problemas de regresión utilizando conjuntos difusos como se expondrá en el capítulo 5.

La LD se utiliza para posibilitar la representación de la naturaleza imprecisa que subyace en el conocimiento. Este conocimiento puede ser representado mediante un conjunto de reglas heurísticas de la forma: IF (antecedente) THEN (consecuente) donde el antecedente, el consecuente o ambos pueden ser definidos mediante conjuntos difusos formando así lo que se conoce como *regla difusa*. En el caso que nos ocupa, la estructura del antecedente es de la forma (X_1 is A_1 and ... and X_n is A_n), siendo X_i las variables lingüísticas y A_i es un valor del subconjunto de valores del dominio de la variable lingüística X_i.

En función de la estructura del consecuente podemos diferenciar entre *reglas difusas tipo Mamdani* y *reglas difusas tipo TSK*. Las reglas difusas tipo *Mamdani* [95] son aquellas en las que el consecuente de la regla es una expresión lingüística de la forma Y is B, siendo Y la variable difusa, y B un conjunto difuso del dominio de la variable Y. De esta forma la estructura de la regla difusa utilizada en esta tesis es de la forma:

$$\text{IF } X_i \text{ is } A_1 \text{ AND } \ldots \text{ AND } X_n \text{ is } A_n \text{ THEN } Y \text{ is } B \ .$$

Siendo un ejemplo de este tipo de reglas el siguiente:

IF ACELERACION is ALTA AND VELOCIDAD is BAJA
THEN CONSUMO is ALTO .

Las reglas difusas son una forma de representación del conocimiento humano. Los sistemas que utilizan este tipo de reglas para la representación simbólica del conocimiento humano se llaman Sistemas Basados en Reglas Difusas (SBRDs). De esta forma se puede definir un Sistemas Basados en Reglas Difusas (SBRDs) como un sistema basado en reglas que utiliza la LD para representación del conocimiento. Concretamente, si el tipo de regla difusa es Mamdani, el sistema se conoce como Sistema Basado en Reglas Difusas (SBRD) tipo Mamdani.

2.1.4 *Aprendizaje de Reglas Difusas*

Hay distintas técnicas automáticas para el aprendizaje del conjunto de reglas difusas que forman la base de conocimiento del SBRD.

Por una parte se pueden diferenciar técnicas que se denominan *ad-hoc* que van generando tanto los antecedentes como los consecuentes de las reglas en función de los ejemplos a cubrir. Algunas de las ventajas de esta técnica son la simplicidad y fácil entendimiento; el bajo costo computacional cuando la dimensión del problema no es muy elevada; y su posible uso para generar modelos difusos preliminares. El trabajo de de Wang-Mendel [131] es uno de los más representativo de esta técnica.

Por otra parte se pueden utilizar redes neuronales para la generación de las reglas difusas [108]. Algunas de las ventajas de esta técnica son la capacidad de aprendizaje; la capacidad de generalización; y la robustez frente al ruido. Por contra, el uso de redes neuronales hace difícil la interpretación de la funcionalidad, al mismo tiempo que es difícil determinar el número de neuronas y capas de la red. Uno de los ejemplos más representativos de esta técnica es el caso de Sistema de Inferencia Difuso Basado en una Red Adaptativa (ANFIS en Inglés) [82].

Por último, hay otras alternativas que hacen uso de los AGs para generar los antecedentes y/o consecuentes de las reglas. Algunos de los ejemplos que podemos encontrar en la literatura son [24, 25, 26, 78, 119, 127] que serán usados para la comparativa de resultados en los capítulos siguientes. Este último es el modelo de aprendizaje para SBRDs que se usa en las propuestas que se presentan en esta tesis.

2.2 CLASIFICACIÓN, REGRESIÓN Y CLASIFICACIÓN ORDINAL

Cada vez es más común el análisis profundo de cantidades de datos para determinados aspectos de la vida. El objetivo de este análisis va desde sistemas de ayuda a la decisión para trasplantes de órganos [105] hasta análisis de riesgos crediticios [75, 120] usando datos de diverso tipo. En función del tipo de datos que se utilice en el problema a analizar podemos encontrar distintos tipos de problemas. En una primera clasificación podemos diferenciar

entre: *datos numéricos*, cuyo dominio son los números enteros o reales; y *datos categóricos* [1], cuyo dominio es un conjunto de símbolos definido. Por una parte, los datos numéricos dan lugar a lo que se conoce como *problemas de regresión*. Por otra parte, encuadrado en los datos categóricos, podemos distinguir entre: *datos nominales*, cuyo dominio es un conjunto de símbolos sin orden alguno entre ellos; y *datos ordinales*, cuyo dominio es un conjunto de símbolos en el que existe una relación de orden. Los problemas que hacen uso de datos nominales son conocidos como *problemas de clasificación* o *problemas de clasificación nominal*, mientras que los datos ordinales definen los *problemas de clasificación ordinal* [121] que también podemos encontrar en la literatura como *problemas de regresión ordinal* [67, 68, 70, 74].

2.2.1 *Clasificación (Nominal)*

Una posible definición de un Problema de Clasificación Nominal puede ser la siguiente. Dado un problema con espacio de entrada X y un conjunto de clases finito $C = C_1, C_2, \ldots, C_Q$, el objetivo es buscar una aplicación entre el espacio de entrada y el conjunto de salida ($\phi : X \to C$). Los patrones son representados por un conjunto donde cada elemento contiene un vector de características K-dimensional ($x \in X \subseteq R^K$) y una etiqueta de la clase ($y \in C$). El conjunto de entrenamiento está formado por N patrones de la forma $T = (x_i, y_i) | x_i \in X, y_i \in C, i = 1, \ldots, N$ donde $x_i = (x_{i1}, x_{i2}, \ldots, x_{iK})$.

Uno de los ejemplos de problemas de clasificación más conocido es el llamado IRIS. A muy groso modo el problema consiste en la clasificación de la flor "Lirio" (Iris en Inglés) dentro de las clases Iris-Setosa, Iris-Versicolor e Iris-Virginica en función de una serie de características, en concreto, en función del largo y ancho del pétalo y el sépalo. De esta manera el problema consiste en obtener como salida la clase a la que pertenece una flor dando como entrada los valores de estas características de pétalo y sépalo.

Hay numerosos problemas de clasificación en los que se han hecho uso tanto de algoritmos de aprendizaje de reglas con distintas variantes [36, 51, 102] como de otros tipos de algoritmos de aprendizaje [50, 60, 61].

2.2.2 *Regresión*

Una posible definición de un Problema de Regresión puede ser la siguiente. Dado un problema con espacio de entrada X, el objetivo es buscar una aplicación entre el espacio de entrada y un valor real de salida Y ($\phi : X \rightarrow Y$, $Y = f(X)$). Los patrones son representados por un conjunto donde cada elemento contiene un vector de características K-dimensional ($x \in X \subseteq \mathbb{R}^K$) y un valor de salida ($y \in \mathbb{R}$). El conjunto de entrenamiento contiene N patrones de la forma $T = (x_i, y_i) | x_i \in X, y_i \in \mathbb{R}, i = 1, \ldots, N$ donde $x_i = (x_{i1}, x_{i2}, \ldots, x_{iK})$.

En la literatura podemos encontrar muchos ejemplos de problemas de regresión, uno de ellos puede ser el llamado Housing. A muy groso modo el problema consiste en obtener el valor de una casa en la periferia de Boston en función de características de entrada como porcentaje de crimen, concentración de óxido nitroso, ratio de alumno-profesor, entre otros. Igual que en caso de clasificación, el problema consiste en obtener un valor de salida en función de unos valores de entrada para las características que definen el problema. La diferencia con la clasificación nominal se encuentra en que en este caso no se obtiene una clase, sino un valor de salida numérico.

Numerosos son los problemas de regresión con buenos resultados en los que se han usado tanto algoritmos de aprendizaje de reglas [39, 126] como otros tipos de algoritmos de aprendizaje [8, 43, 73, 125].

Algoritmos para regresión

En esta sección se presenta una breve descripción de los algoritmos utilizados para la comparativa con el algoritmo que se presenta en el capítulo 5.

Los problemas de regresión se pueden abordar utilizando diferentes técnicas. En la lista siguiente se expone la clasificación que realiza Keel[5] [3, 6] para los algoritmos usados en la comparación con el propuesto en esta tesis. Posteriormente se explicará, de forma breve, dichos algoritmos.

5 http://www.keel.es/

- Métodos Estadísticos
    - LinearLMS-R
    - PolQuadraticLMS-R
- SVM para regresión
    - NU_SVR-R
- Redes Neuronales para regresión
    - Redes Neuronales para regresión
        - RBFN-R
        - Inc-RBFN-R
        - MLP-BP-R
        - IRProp+-R
        - Ensemble-R
    - Redes Neuronales Evolutivas para regresión
        - NNEP-R
- Regresión simbólica difusa evolutiva
    - GFS-GAP-Sym-R
    - GFS-GSP-R
    - GFS-GP-R
    - GFS-SAP-Sym-R
- Aprendizaje de reglas para regresión
    - Árboles de decisión para regresión
        - M5-R
        - M5Rules-R
        - CART-R
    - Aprendizaje de reglas difusas para regresión
        - FRSBM-R
        - WM-R
    - Aprendizaje de reglas difusas evolutivas para regresión
        - TSK-IRL-R
        - MOGUL-IRLSC-R
        - GFS-GPG-R
        - MOGUL-IRLHC-R
        - MOGUL-TSK-R
        - GFS-SP-R
        - Thrift-R
        - GFS-RB-MF-R

Respecto a los **métodos estadísticos** se puede observar que hay dos aproximaciones según se utiliza regresión lineal (LinearLMS-R) o cuadrática (PolQuadraticLMS-R) [116].

Como algoritmo representativo de la categoría **Máquinas de Vectores Soporte (SVM)** se muestra (NU_SVR-R) [41] donde los autores proponen una técnica para trabajar con la selección de conjuntos en los métodos de descomposición tipo optimización mínima secuencial para el entrenamiento de SVM.

En la categoría de **redes neuronales** aparecen dos subcategorías. La primera de ellas corresponde con las *redes neuronales aplicadas a regresión* donde se encuentran los algoritmos (RBFN-R) [16], (Incr-RBFN-R) [107] que utilizan redes neuronales con funciones de base radial. En esta misma subcategoría se encuentran (MLP-BR-R) [114], (iRProp+-R) [81, 130] que utilizan redes neuronales tipo "Backpropagation", cerrando esta subcategoría tenemos el algoritmo (Ensemble-R) [59] donde se aborda la regresión desde el punto de vista de los multiclasificadores de redes neuronales. La segunda subcategoría utiliza *redes neuronales con programación evolutiva* siendo su representante el algoritmo (NNEP-R) [97].

A continuación aparece la categoría **regresión simbólica difusa evolutiva** donde encontramos diferentes algoritmos de aprendizaje difuso simbólico basado en programación genética (GFS-GP-R) y operadores gramaticales de programación genética (GFS-GAP-Sym-R) [118], combinándolo también con enfriamiento simulado (GFS-GSP-R) y (GFS-SAP-Sym-R) [118, 119]

La última categoría es la correspondiente al **aprendizaje de reglas**. Esta categoría está compuesta por tres subcategorias. La primera subcategoría es la referente a *árboles de decisión* que se encuentra representada por dos algoritmos correspondientes a la implementación del algoritmo M5 para regresión (M5-R) y (M5Rules-R) [77, 109, 129], junto con el algoritmo CART para regresión (CART-R) [14]. La segunda subcategoría hace referencia al *aprendizaje de reglas difusas* donde se encuentran los algoritmos sobre modelado basado en conjuntos aleatorios (FRSBM-R) [117] y el algoritmo de Wang-Mendel para regresión (WM-R) [131]. Por último se muestra la subcategoría correspondiente al *aprendizaje de reglas difusas evolutivas* donde encontramos una amplia representación de las distintas alternativas existentes para abarcar la regresión desde este punto de vista. Haciendo una pequeña subdivisión se pueden observar algoritmos que utilizan reglas tipo TSK como (TSK-IRL-R) [25] o (MOGUL-TSK-R) [4] frente a algoritmos

que utilizan reglas tipo Mamdani como (MOGUL-IRLSC-R) [24] o (MOGUL-IRLHC-R) [26]. Por otra parte se presentan algoritmos que usan operadores gramaticales de programación genética junto con (GFS-SP-R) [119] y sin (GFS-GPG-R) enfriamiento simulado. Para finalizar encontramos la implementación del algoritmo Thrift (Thrift-R) [127], junto a un algoritmo que ajusta la función de pertenencia para construir bases de reglas difusas basadas en AG (GFS-RB-MF-R) [24, 78].

Uno de los objetivos de esta tesis es el desarrollo de un algoritmo de aprendizaje de reglas difusas basado en la estrategia de recubrimiento secuencial que utiliza un AG como mecanismo de búsqueda de la mejor regla para abordar problemas de regresión, como veremos en el capítulo 5.

2.2.3 *Clasificación Ordinal*

De forma similar a los apartados anteriores, una posible definición de un Problema de Clasificación Ordinal puede ser la siguiente [120]. Dado un problema con espacio de entrada X y un conjunto finito $C = C_1, C_2, \ldots, C_Q$, el objetivo es buscar una función que mapee el espacio de entrada en el conjunto de salida ($\phi : X \rightarrow C$). Lo más importante es que este conjunto de etiquetas tiene una relación de orden de la forma $C_1 \prec C_2 \prec \cdots \prec C_Q$ donde $\prec$ indica el orden entre las diferentes etiquetas. También se define $O(C_q) = q$ como la posición que ocupa la etiqueta q en la escala ordinal. Los patrones son representados por un conjunto donde cada elemento contiene un vector de características K-dimensional ($x \in X \subseteq R^K$) y una etiqueta de la clase ($y \in C$). El conjunto de entrenamiento está formado por N patrones de la forma $T = (x_i, y_i) | x_i \in X, y_i \in C, i = 1, \ldots, N$ donde $x_i = (x_{i1}, x_{i2}, \ldots, x_{iK})$. Como se puede apreciar la diferencia respecto a lo definido en el apartado 2.2.1 es la relación de orden subyacente.

En la literatura podemos encontrar diversas adaptaciones de problemas de regresión a clasificación ordinal, sin embargo no hay muchos problemas que en su origen se consideran ordinales. Los principales ejemplos de problemas originalmente ordinales, disponibles en Weka[6], son: ESL (employee selection) que contiene un conjunto de perfiles de personas demandantes de un empleo; SWD (social workers decisions) que contiene evaluaciones de tra-

6 http://www.cs.waikato.ac.nz/ml/weka/

bajadores sociales que ayudan a los jueces en la toma de decisión sobre la opción de dejar a los niños presuntamente maltratados o descuidados con su familia o no; LEV (lecturers evaluation) que contiene información sobre la evaluación de clases o conferencias; y ERA (employee rejection/acceptance) que contiene información sobre las cualidades más importantes que deben tener los candidatos a cierto tipo de trabajos. En el caso de la clasificación ordinal, al igual que en los casos de clasificación y regresión, el objetivo es obtener un valor de salida en función de unos valores de entrada. En este caso el valor de salida es una clase al igual que ocurre en clasificación, con la diferencia de que en este caso hay establecido un orden entre las clases. Esta diferencia con la clasificacion nominal lo asemeja a la regresión ya que los valores numéricos resultados de la regresión tienen un orden establecido al igual que ocurre en la clasificación ordinal.

Cada vez son más las problemas en los que la aplicación de la clasificación ordinal obtiene muy buenos resultados debido a la naturaleza del problema [29, 38, 69, 69, 101, 105, 106]. Esto es gracias a que el clasificador ordinal explota las características peculiares de este tipo de problemas derivadas de la ordenación de las clases de la variable de salida.

Algoritmos para clasificación ordinal

La clasificación ordinal está alcanzando cada vez mayor relevancia en los problemas de aprendizaje. A pesar de ello, aún no hay muchos algoritmos para este tipo de problemas y aún menos de ellos disponibles para su uso. Además no hay categoría específica para este tipo de problemas en los principales software de aprendizaje automático como Keel [3, 6], Weka [71] o KNIME [11, 12], teniendo que realizar tareas de preprocesamiento y posprocesamiento para poder hacer uso de otros tipos de algoritmos en este tipo de problemas. Es por ello que se plantea como uno de los objetivos de esta tesis el desarrollo de un algoritmo de este tipo y, como no puede ser de otra forma, su puesta a disposición de la comunidad investigadora. Esta sección muestra una breve descripción de los principales algoritmos, hasta donde nuestro conocimiento alcanza, para abordar problemas de clasificación ordinal, utilizando los que están disponibles para la comparativa con el algoritmo que se presenta en el capítulo 4.

En la literatura se presentan distintas aproximaciones para abordar problemas de clasificación ordinal. Por una parte se encuentran algunas que usan SVM con distintas modificaciones como RED-SVM [90] que aplica SVM a una reducción a problemas binarios ponderados, junto a SVOR-EX y SVOR-IM [23] que aplica SVM imponiendo restricciones explícitas e implícitas. Por otra parte, en la literatura aparece POM [99] y GPOR [21] que utiliza un modelo de posibilidades proporcionales y procesos gausianos respectivamente, junto con una aproximación mediante análisis discriminante de núcleos llamado KDLOR [124]. Por último, se pueden encontrar aproximaciones basadas en clasificación binaria como ASAOR [44] que es una implementación del algoritmo C4.5 para ordinal y EBC [88] que utiliza una clasificación binaria extendida para abordar la clasificación ordinal.

Otro de los objetivos de esta tesis es el desarrollo de un algoritmo de aprendizaje de reglas difusas para abordar problemas de clasificación ordinal como veremos en el capítulo 4.

2.3 MÉTRICAS

En las secciones anteriores se han presentado unas breves nociones sobre los tipos de problemas de aprendizaje junto a algunos de los algoritmos más usados para abordar cada tipo de problema. El siguiente paso es establecer un mecanismo que pueda indicar cómo de bueno es un algoritmo en comparación con otro para cada problema en cuestión.

En este apartado se exponen algunas de las métricas más comunes en cada uno de los tipos de problemas de aprendizaje. En primer lugar comenzamos con las métricas para problemas de clasificación, continuando con las métricas para problemas de regresión para finalizar con las métricas más usadas en problemas de clasificación ordinal.

2.3.1 *Métricas para Clasificación*

De una forma general podemos decir que la *Clasificación Nominal* o simplemente *Clasificación* es un problema de aprendizaje en el que el objetivo es predecir la categoría o clase a la que pertenece un patrón o ejemplo.

En primer lugar vamos a realizar un breve repaso de las métricas más utilizadas en clasificación [28]. Antes de enumerar las métricas para clasificación, se hace necesario definir el concepto de matriz de confusión. La matriz de confusión $M(g)$ de un clasificador binario g se define como:

$$M(g) = \begin{pmatrix} tp & fn \\ fp & tn \end{pmatrix} .$$

donde:

- tp (*true positive en Inglés*): es el número de elementos pertenecientes a la clase positiva que el clasificador ha predicho como positivos
- fn (*false negative en Inglés*): es el número de elementos de la clase positiva que han sido clasificados como negativos.
- fp (*false positive en Inglés*): es el número de elementos pertenecientes a la clase negativa que han sido clasificados como positivos.
- tn (*true negative en Inglés*): es el número de elementos de la clase negativa que han sido predichos como negativos.

La generalización a Q clases y N ejemplos de entrenamiento sería:

$$M(g) = \left\{ n_{ij}; \sum_{i,j=1}^{Q} n_{ij} = N \right\} .$$

donde n_{ij} representa el número de patrones predichos por el clasificador g en la clase j cuando en realidad pertenecen a la clase i. También se define $n_{i\bullet}$ como el número de patrones que pertenecen a la clase i y $n_{\bullet j}$ como el número de patrones clasificados en la clase j. La diagonal de la matriz de confusión se corresponde

con los patrones correctamente clasificados y los elementos de fuera de la diagonal principal con los errores de clasificación.

		clase predicha					
		1	$\cdots$	j	$\cdots$	Q	
clase real	1	n_{11}	$\cdots$	n_{1j}	$\cdots$	n_{1Q}	$n_{1\bullet}$
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
	i	n_{i1}	$\cdots$	n_{ij}	$\cdots$	n_{iQ}	$n_{i\bullet}$
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
	Q	n_{Q1}	$\cdots$	n_{Qj}	$\cdots$	n_{QQ}	$n_{Q\bullet}$
		$n_{\bullet 1}$	$\cdots$	$n_{\bullet j}$	$\cdots$	$n_{\bullet Q}$	N

Dentro de las métricas para clasificación podemos destacar las siguientes:

1. *Precisión (Correct Classification Rate en Inglés) (CCR)*: Rendimiento global de un clasificador, es decir, porcentaje de patrones correctamente clasificados.

$$\mathrm{CCR} = \frac{tp + tn}{tp + tn + fp + tn} = \frac{1}{N}\sum_{i=1}^{Q} n_{ii} \,.$$

2. *Error Medio Cero-Uno (Mean Zero-One Error en Inglés) (MZE)*: Error global de un clasificador, es decir, el porcentaje de patrones incorrectamente clasificados. Es contrario a CCR.

$$\mathrm{MZE} = 1 - \mathrm{CCR} \,.$$

3. *Sensibilidad (Sensibility en Inglés) (S)* o *Radio de verdaderos positivos (True Positive Rate o Recall en Inglés) (TPR)*: Porcentaje de patrones de la clase positiva correctamente clasificados con respecto al total de patrones de la clase positiva.

$$S = \mathrm{TPR} = \frac{tp}{tp + fn} \,.$$

En el caso de múltiples clases se define S_i como el porcentaje de elementos de la clase i correctamente clasificados.

$$S_i = \mathrm{TPR}_i = \frac{n_{ii}}{n_{i\bullet}} \,.$$

Finalmente se define la mínima sensibilidad (MS) como el mínimo de todos estos porcentajes e indica el rendimiento de la clase peor clasificada.

$$MS = \text{mín}(S_i; i = 1, \dots, Q) .$$

4. *Media de las Sensibilidades de todas las clases (Macro-Media en Inglés) (MM)*: en esta métrica se consideran todas las clases con la misma importancia.

$$MM = \frac{1}{Q} \sum_{i=1}^{Q} S_i .$$

5. *Especificidad (Specificity en Inglés) (SP)* o *Radio de verdaderos negativos (True Negative Rate en Inglés) (TNR)*: porcentaje de ejemplos de la clase negativa correctamente clasificados respecto al total de elementos de la clase negativa.

$$SP = TNR = \frac{tn}{fp + tn} .$$

En el caso de múltiples clases se fija la clase objeto de estudio y el resto de clases se consideran negativas.

$$SP_i = TNR_i = \frac{\sum_{j=1,k=1,j\neq i,k\neq i}^{Q} n_{jk}}{\sum_{j=1,j\neq i}^{Q} n_{j\bullet}} = \frac{\sum_{j=1,k=1,j\neq i,k\neq i}^{Q} n_{jk}}{N - n_{i\bullet}} .$$

6. *Radio de falsos positivos (False Positive Rate en Inglés) (FPR)*: porcentaje de ejemplos de la clase negativa incorrectamente clasificados respecto al total del elementos de la clase negativa.

$$FPR = \frac{fp}{fp + tn} = 1 - SP .$$

Al igual que TNR (SP), en el caso de múltiples clases se fija la clase objeto de estudio y el resto de clases se consideran negativas.

$$FPR_i = \frac{n_{\bullet i} - n_{ii}}{\sum_{j=1,j\neq i}^{Q}} = \frac{n_{\bullet i} - n_{ii}}{N - n_{i\bullet}} = 1 - SP_i .$$

7. *Kappa de Cohen*: mide el grado de acuerdo entre los valores reales del problema y los predichos por el clasificador. El rango de la misma es de $[-1, 1]$ donde 1 indica máximo acuerdo, -1 máximo desacuerdo y toma el valor 0 cuando no existen relación entre los valores.

$$K = \frac{P_0 - P_e}{1 - P_e} .$$

P_0 es la proporción de concordancia observada (CCR) y P_e es la proporción de concordancia esperada por puro azar.

$$P_0(CCR) = \frac{1}{N} \sum_{i=1}^{Q} n_{ii} .$$

$$P_e = \frac{1}{N^2} \sum_{i=1}^{Q} n_{i\bullet} n_{\bullet i} .$$

8. *Área bajo la curva (Area under curve en Inglés) (AUC)*: la curva Característica Operativa del Receptor (Receiver Operating Characteristic en Inglés) (ROC) (figura 2.8) es una representación gráfica del "radio de verdaderos positivos" (TPR o S) frente al "radio de falsos positivos" (FPR) para un sistema clasificador según se varía un umbral de discriminación.

$$AUC = \frac{1 + TPR - FPR}{2} .$$

Para el caso de múltiples clases, hay variedad de aproximaciones sin tener ningún consenso entre todos los investigadores, aunque la más usada calcula los valores de AUC para cada clase (AUC_i) y se realiza la media ponderada de dichos valores.

$$AUC_i = \frac{1 + TPR_i - FPR_i}{2} .$$

$$AUCMedia = \frac{1}{Q} \sum_{i=1}^{Q} AUC_i .$$

A modo de guía para interpretar las curvas ROC se han establecido los siguientes intervalos para los valores de AUC y la valoración del clasificador:

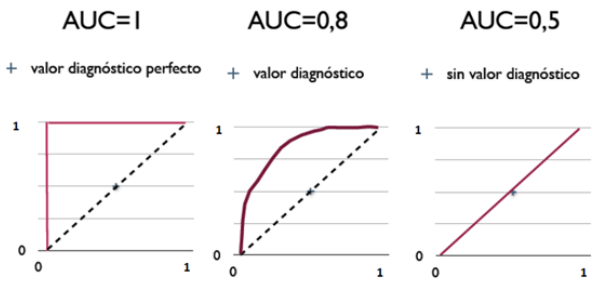


Figura 2.8: Ejemplo de Curvas ROC con diferentes valores para AUC

- [0.5, 0.6): malo.
- [0.6, 0.75): regular.
- [0.75, 0.9): bueno.
- [0.9, 0.97): muy bueno.
- [0.97, 1): excelente.

Un estudio detallado sobre AUC se puede encontrar en [27] donde se realiza un análisis sobre la influencia del guiado de un algoritmo de aprendizaje con esta métrica respecto de la precisión.

2.3.2 *Métricas para Regresión*

Los *Problemas de Regresión* intentan predecir un valor real asociado a una determinada entrada. Un ejemplo de un problema de regresión es la predicción del valor del consumo energético de una ciudad a partir de unos valores de climatología, época del año, día de la semana, etc y basándose en lo observado en el pasado.

Respecto a las métricas para regresión, hay una gran variedad, siendo las más usadas las que se muestran a continuación. Hay que indicar que en los problemas de regresión se suele utilizar la variable Y para hacer referencia a los valores reales de la salida y la variable $\hat{Y}$ para los valores resultantes de la predicción.

1. *Error cuadrático medio (Mean Square Error en Inglés) (*MSE*)*: media de los errores cuadráticos cometidos en la predicción.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2 .$$

2. *Raíz del Error cuadrático medio (Root Mean Square Error en Inglés) (*RMSE*)*: raíz cuadrada de la media de los errores cuadráticos cometidos en la predicción.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2} .$$

3. *Error Medio Absoluto (Mean Absolute Error en Inglés) (*MAE*)*: media de los errores en valor absoluto cometidos en la predicción.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{Y}_i - Y_i| .$$

Las métricas de clasificación están acotadas entre los valores 0 y 1 o -1 y 1 independientemente del problema en cuestión. En el caso de las métricas de regresión conocemos que el valor mínimo es el 0, mientras que el valor máximo es dependiente del problema en cuestión. Con el propósito de evitar esta dependencia del problema y poder comparar los valores sobre diferentes problemas, proponemos las siguientes métricas que podemos denominar como "normalizadas" y cuyo resultado es un porcentaje de error sobre el dominio de la variable de salida. Las métricas que se proponen son las siguientes:

4. MSE^N: porcentaje de la media de los errores cuadráticos cometidos en la predicción respecto al dominio de la variable de salida. Se puede entender como el MSE normalizado.

$$MSE^N = \frac{1}{N * A^2}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2 .$$

donde A es la amplitud del rango de la variable de salida.

5. $RMSE^N$: porcentaje de la raíz cuadrada de la media de los errores cuadráticos cometidos en la predicción respecto al dominio de la variable de salida. Se puede entender como el RMSE normalizado.

$$RMSE^N = \sqrt{MSE^N} = \sqrt{\frac{1}{N * A^2} \sum_{i=1}^{N} (\hat{Y}_i - Y_i)^2} .$$

6. MAE^N: porcentaje de la media de los errores en valor absoluto cometidos en la predicción respecto al dominio de la variable de salida. Se puede entender como el MAE normalizado

$$MAE^N = \frac{1}{N * A} \sum_{i=1}^{N} |\hat{Y}_i - Y_i| .$$

2.3.3 *Métricas para Clasificación Ordinal*

De forma genérica se puede decir que un *Problema de Clasificación Ordinal* [121] es un problema de aprendizaje en el que el objetivo es predecir la categoría a la que pertenece un patrón con la peculiaridad de la existencia de una relación de orden entre las categorías.

Como se ha expuesto en secciones anteriores, los problemas de clasificación ordinal tienen aspectos comunes entre la clasificación y la regresión. La similitud entre la clasificación ordinal y nominal se encuentra en que el objetivo de ambas es clasificar correctamente un conjunto de clases finitas. La diferencia estriba en la existencia o no de un orden en las clases. Respecto a la clasificación ordinal y regresión, ambas presentan un cierto orden en los valores de salida, pero mientras que en regresión la variable es continua e infinita, en clasificación ordinal ésta es discreta y finita. Para la evaluación de los algoritmos que permitan modelar problemas de clasificación ordinal habrá que considerar métricas que tengan en cuenta las características especiales de estos tipos de problemas.

Recientemente han aparecido algunos trabajos [10, 19, 30, 31, 67, 120] sobre las métricas que mejor permiten evaluar los algoritmos para problemas de clasificación ordinal. Además, por las características que comparte la clasificación ordinal tanto con

la clasificación como con la regresión, para la evaluación de los algoritmos de aprendizaje se pueden utilizar tanto métricas de clasificación como métricas de regresión con pequeñas variaciones. De esta forma, la formulación para las métricas más usadas en problemas de clasificación ordinal son las siguientes.

En general vamos a considerar un problema ordinal con Q clases y N patrones. Sobre éste vamos a definir $O(y_i)$ como el orden de la clase en la que está el valor real y_i, siendo $O(\hat{y}_i)$ el orden de la clase en la que está el valor predicho $\hat{y}_i$. Además se considera que las clases pueden enumerarse comenzando con un valor 1 hasta un valor Q con lo que la distancia entre las clases es 1 haciendo coincidir el valor de la clase con la posición que ocupa.

1. *Error Medio Absoluto Ordinal (Ordinal Mean Absolute Error en Inglés) (*OMAE*)*: desviación absoluta media entre la clase predicha y la real. Se puede entender como una combinación entre el MAE de regresión y el CCR de clasificación ya que mide la variación entre el valor real y el predicho, considerando igualmente el número de ejemplos de la clase.

$$\mathrm{OMAE} = \frac{1}{N}\sum_{i=1}^{N} e(x_i) = \frac{1}{N}\sum_{q,k=1}^{Q} |q-k| n_{qk} \, .$$

donde $e(x_i) = |O(y_i) - O(\hat{y}_i)|$ es la distancia entre las clases del valor real y el predicho y n_{qk} es el número de ejemplos de la clase q predichos como de la clase k.

2. *Media del Error Medio Absoluto (Average Mean Absolute Error en Inglés) (*AMAE*)*: media de las OMAEs de cada una de las clases.

   Para definir AMAE, en primer lugar hay que definir OMAE de cada clase:

$$\mathrm{OMAE}_q = \frac{1}{n_{q\bullet}}\sum_{k=1}^{Q} |q-k| n_{qk} \, .$$

   Así, otra posible definición de OMAE del problema considerando el OMAE_q podría ser:

$$\mathrm{OMAE} = \frac{1}{N}\sum_{q=1}^{Q} n_{q\bullet}\mathrm{OMAE}_q \, .$$

Para finalmente definir la AMAE como:

$$\text{AMAE} = \frac{1}{Q}\sum_{q=1}^{Q} \text{OMAE}_q \ .$$

3. *Máximo del Error Medio Absoluto (Maximal Mean Absolute Error en Inglés) (MMAE)*: máximo de las OMAEs de cada una de las clases. Indica la clase de mayor distancia entre los valores reales y predichos.

$$\text{MMAE} = \text{máx}\, \text{OMAE}_q ; q = 1, \ldots, Q \ .$$

4. r_s *(Coeficiente de correlación de Spearman)*: mide la dependencia entre dos variables, que en el caso de problemas ordinales se obtiene con el orden de las etiquetas de la clase real y predicha de la forma

$$r_s = \frac{\sum_{i=1}^{N}(\ (O(y_i) - \overline{O(y)}) \ (O(\hat{y_i}) - \overline{O(\hat{y})}) \ )}{\sqrt{\sum_{i=1}^{N}(O(y_i) - \overline{O(y)})^2}\ \sqrt{\sum_{i=1}^{N}(O(\hat{y_i}) - \overline{O(\hat{y})})^2}} \ .$$

donde $\overline{O(y)}$ y $\overline{O(\hat{y})}$ representan la media de $O(y_i)$ y $O(\hat{y_i})$ respectivamente para $i = 1, \ldots, N$.

5. τ_b *de Kendall*: mide el grado de asociación entre dos variables. Igual que en el caso anterior, en problemas ordinales se obtiene con el orden de las etiquetas de la clase real y predicha.

$$\tau_b = \frac{\sum_{i,j=1}^{N}(\hat{c_{ij}} c_{ij})}{\sqrt{(\sum_{i,j=1}^{N} \hat{c_{ij}}^2)(\sum_{i,j=1}^{N} c_{ij}^2)}} \ .$$

donde $\hat{c_{ij}}$ es "+1" si $O(\hat{y_i}) > O(\hat{y_j})$, "0" si $O(\hat{y_i}) = O(\hat{y_j})$, y "−1" si $O(\hat{y_i}) < O(\hat{y_j})$ para $i = 1, \ldots, N$. De forma similar sería para c_{ij}.

6. OC_β^γ *(Ordinal Classification Index, Índice de Clasificación Ordinal)*: métrica propuesta en [19] que puede ser obtenida directamente a partir de la matriz de confusión y cuyo

procesamiento no es muy complejo computacionalmente hablando. Establece como mejor camino el formado por la diagonal de la matriz de confusión teniendo en cuenta las posibles desviaciones respecto de este camino para el cálculo del error en la clasificación. El parámetro β se utiliza para la penalización por tomar un camino no óptimo, lo que se traduce en una mala clasificación. El rango es $[0..1]$ donde 0 significa que ha clasificado correctamente y 1 indicaría el máximo error en la clasificación para los parámetros considerados. El parámetro γ se utiliza para el tipo de medición de la distancia. Los valores de los parámetros suelen ser $\beta = \frac{0,25}{N(Q-1)}$ o $\beta = \frac{0,75}{N(Q-1)}$ y $\gamma = 1$ si se considera la distancia de Manhattan, donde N es el número de ejemplos y Q es el número de clases. La formulación general es:

$$OC_\beta^\gamma = \min\{1 - \frac{\sum_{(r,c)\in path} n_{r,c}}{N + (\sum_{\forall(r,c)} n_{r,c}|r-c|^\gamma)^{1/\gamma}} + \beta \sum_{(r,c)\in path} n_{r,c}|r-c|^\gamma\} .$$

donde r y c son los índices de fila y columna de la matriz de confusión respectivamente, $n_{r,c}$ es el valor correspondiente a la fila r y columna c de la matriz de confusión.

En la práctica se calcula utilizando los valores de la matriz de confusión. Para ello se calculan unos valores intermedios de ponderaciones de la desviación de la diagonal, creando dos matrices (w y W) según las ecuaciones siguientes. Finalmente el valor de OC_β^γ es el valor $W_{Q,Q}$.

$$w_{r,c} = -\frac{n_{r,c}}{N + (\sum_{\forall(r,c)} n_{r,c}|r-c|^\gamma)^{1/\gamma}} + \beta n_{r,c}|r-c|^\gamma .$$

$$W_{r,c} = w_{r,c} + \min\{W_{r-1,c-1}, W_{r-1,c}, W_{r,c-1}\} .$$

donde $W_{1,1} = 1 + w_{1,1}$ y $W_{Q,Q} = OC_\beta^\gamma$

2.3.4 *Ejemplo de uso de las métricas*

Para un estudio más profundo y una mejor comprensión de estas métricas vamos a considerar las siguientes casuísticas expuestas

en el siguiente ejemplo. El ejemplo trata de predecir la evaluación de 8 individuos para una determinada asignatura. Para predecir el valor de salida se utilizan 17 clasificadores. De estos clasificadores, 16 van a tomar los mismos datos de entrada generando distintos valores de salida correspondientes a 8 casuísticas distintas. Además se van a considerar otros datos de entrada con el último clasificador que se corresponde con el caso de la peor de las clasificaciones posibles. Cada una de estas casuísticas se componen de dos "subcasos" con valores diferentes pero que cuya clase es la misma. Estos "subcasos" se utilizan para el análisis de las métricas de regresión. Los valores reales y los predichos para cada una de las casuísticas se muestran en la tabla 2.1 .

ej	y	ŷ							
		C_1		C_2		C_3		C_4	
		C_{1a}	C_{1b}	C_{2a}	C_{2b}	C_{3a}	C_{3b}	C_{4a}	C_{4b}
1	1 (S)	1 (S)	4 (S)	5 (A)	6 (A)	7 (N)	8 (N)	9 (SB)	10 (SB)
2	4 (S)	4 (S)	1 (S)	5 (A)	6 (A)	7 (N)	8 (N)	9 (SB)	10 (SB)
3	5 (A)	4 (S)	1 (S)	5 (A)	6 (A)	7 (N)	8 (N)	9 (SB)	10 (SB)
4	6 (A)	4 (S)	1 (S)	6 (A)	5 (A)	7 (N)	8 (N)	9 (SB)	10 (SB)
5	7 (N)	4 (S)	1 (S)	6 (A)	5 (A)	7 (N)	8 (N)	9 (SB)	10 (SB)
6	8 (N)	4 (S)	1 (S)	6 (A)	5 (A)	8 (N)	7 (N)	9 (SB)	10 (SB)
7	9 (SB)	4 (S)	1 (S)	6 (A)	5 (A)	8 (N)	7 (N)	9 (SB)	10 (SB)
8	10 (SB)	4 (S)	1 (S)	6 (A)	5 (A)	8 (N)	7 (N)	10 (SB)	9 (SB)

ej	y	ŷ							
		C_{ok}		C_i		C_{e_1}		C_{e_2}	
		C_{oka}	C_{okb}	C_{ia}	C_{ib}	C_{e_1a}	C_{e_1b}	C_{e_2a}	C_{e_2b}
1	1 (S)	1 (S)	4 (S)	9 (SB)	10 (SB)	5 (A)	6 (A)	9 (SB)	10 (SB)
2	4 (S)	4 (S)	1 (S)	9 (SB)	10 (SB)	5 (A)	6 (A)	9 (SB)	10 (SB)
3	5 (A)	5 (A)	6 (A)	7 (N)	8 (N)	4 (S)	1 (S)	9 (SB)	10 (SB)
4	6 (A)	6 (A)	5 (A)	7 (N)	8 (N)	4 (S)	1 (S)	9 (SB)	10 (SB)
5	7 (N)	7 (N)	8 (N)	6 (A)	5 (A)	9 (SB)	10 (SB)	4 (S)	1 (S)
6	8 (N)	8 (N)	7 (N)	6 (A)	5 (A)	9 (SB)	10 (SB)	4 (S)	1 (S)
7	9 (SB)	9 (SB)	10 (SB)	4 (S)	1 (S)	8 (N)	7 (N)	4 (S)	1 (S)
8	10 (SB)	10 (SB)	9 (SB)	4 (S)	1 (S)	8 (N)	7 (N)	4 (S)	1 (S)

ej	y	ŷ
		C_E
1	1 (S)	10 (SB)
2	1 (S)	10 (SB)
3	1 (S)	10 (SB)
4	1 (S)	10 (SB)
5	1 (S)	10 (SB)
6	1 (S)	10 (SB)
7	1 (S)	10 (SB)
8	1 (S)	10 (SB)

Tabla 2.1: Ejemplo con valores reales y predichos para distintas casuísticas

Este ejemplo se puede tratar como un problema de regresión donde el valor de resultado es el valor numérico. También se puede tratar como un problema ordinal en el que en función de una calificación numérica se le asocia una calificación categórica. Las clases correspondientes a los valores numéricos que consideramos son los que estamos acostumbrados a utilizar normalmente, a saber, de $[0-5)$ se calificará como "suspenso", de $[5-7)$ como "aprobado", de $[7-9)$ será "notable" siendo el último de $[9-10]$ para "sobresaliente". Como nota indicar que en este caso no se considera la "matrícula de honor" porque no aporta una información relevante para el objetivo del problema. Considerando que estamos ante un problema ordinal se pueden construir una matriz de confusión para cada una de las 9 casuística independientemente del subcaso que consideremos. La tabla 2.2 muestra las matrices de confusión correspondientes a las casuísticas que vamos a estudiar.

C_1			
2	0	0	0
2	0	0	0
2	0	0	0
2	0	0	0

C_2			
0	**2**	0	0
0	**2**	0	0
0	**2**	0	0
0	**2**	0	0

C_3			
0	0	**2**	0
0	0	**2**	0
0	0	**2**	0
0	0	**2**	0

C_4			
0	0	0	**2**
0	0	0	**2**
0	0	0	**2**
0	0	0	**2**

C_{ok}			
2	0	0	0
0	**2**	0	0
0	0	**2**	0
0	0	0	**2**

C_i			
0	0	0	**2**
0	0	**2**	0
0	**2**	0	0
2	0	0	0

C_{e_1}			
0	**2**	0	0
2	0	0	0
0	0	0	**2**
0	0	**2**	0

C_{e_2}			
0	0	0	**2**
0	0	0	**2**
2	0	0	0
2	0	0	0

C_E			
0	0	0	**8**
0	0	0	0
0	0	0	0
0	0	0	0

Tabla 2.2: Matrices de confusión de las distintas casuísticas

Los casos considerados cubren en gran medida la casuística que puede encontrarse en cualquier problema real. De los casos C_1 a C_4 hacen referencia a los casos en los que el clasificador ha predicho los ejemplos como de la clase 1 a las 4 respectivamente. El caso C_{ok} es el correspondiente a una clasificación correcta. El caso C_i es el correspondiente a una clasificación inversa respecto a la correcta. Los casos C_{e_1} y C_{e_2} son los correspondientes a una

clasificación errónea en los que en el primer caso el error que se comete es menor que en el segundo. Por último presentamos el caso C_E correspondiente a una clasificación completamente errónea en la que todos los ejemplos son de la clase 1 y se ha predicho como de la clase 4 con el máximo error si tomamos salidas reales en vez de clases.

Como se puede observar en la tabla 2.1, para cada caso hay dos subcasos con diferentes valores predicho y real dentro de la misma clase de forma que hay una menor diferencia entre valor predicho y real en los primeros subcasos ($C_x a$), siendo la diferencia mayor en los segundos subcasos ($C_x b$).

	C_1		C_2		C_3		C_4		
	$C_1 a$	$C_1 b$	$C_2 a$	$C_2 b$	$C_3 a$	$C_3 b$	$C_4 a$	$C_4 b$	
CCR	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	...
MM	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	...
MS	0	0	0	0	0	0	0	0	...
$\overline{TPR/S}$	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	...
$\overline{TNR/SP}$	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	...
$\overline{FPR}$	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	...
Kappa	0	0	0	0	0	0	0	0	...
$\overline{AUC}$	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	...

	C_{ok}		C_i		C_{e_1}		C_{e_2}		C_E
	$C_{ok} a$	$C_{ok} b$	$C_i a$	$C_i b$	$C_{e_1} a$	$C_{e_1} b$	$C_{e_2} a$	$C_{e_2} b$	C_E
CCR	1	1	0	0	0	0	0	0	0
MM	1	1	0	0	0	0	0	0	0
MS	1	1	0	0	0	0	0	0	0
$\overline{TPR/S}$	1	1	0	0	0	0	0	0	0
$\overline{TNR/SP}$	1	1	0,67	0,67	0,67	0,67	0,67	0,67	-
$\overline{FRP}$	0	0	0,33	0,33	0,33	0,33	0,33	0,33	-
Kappa	1	1	-0,33	-0,33	-0,33	-0,33	-0,33	-0,33	0
$\overline{AUC}$	1	1	0,33	0,33	0,33	0,33	0,33	0,33	-

Tabla 2.3: Resultados de la métricas de clasificación sobre el ejemplo

Las tablas 2.3, 2.4 y 2.5 muestran los valores de las métricas de clasificación, regresión y clasificación ordinal respectivamente en el ejemplo que estamos considerando. En los resultados de las métricas de clasificación y clasificación ordinal no hay diferencias entre los dos subcasos ya que para el cálculo de estas métricas se utiliza la matriz de confusión que hemos visto que es la misma en ambos casos presentando valores diferentes en el caso de regresión.

En la tabla 2.3 se presentan los resultados para las métricas de clasificación. En esta tabla se observa que para el caso de C_E no es posible el cálculo de TNR_i y FPR_i obteniéndose una división entre

	C_1		C_2		C_3		C_4		
	C_1a	C_1b	C_2a	C_2b	C_3a	C_3b	C_4a	C_4b	
MSE	11.38	36.13	5.88	10,63	6.88	11.63	14.88	21.63	...
RMSE	3.37	6.01	2.42	3.26	2.62	3.41	3.86	4.65	...
MAE	2.63	5.63	1.88	2.88	1.88	2.88	2.88	3.88	...
MSE^N	0,14	0,45	0,07	0,13	0,08	0,14	0,18	0,27	...
$RMSE^N$	0,37	0,67	0,27	0,36	0,29	0,38	0,43	0,52	...
MAE^N	0,29	0,63	0,21	0,32	0,21	0,32	0,32	0,43	...

	C_{ok}		C_i		C_{e_1}		C_{e_2}		C_E
	$C_{ok}a$	$C_{ok}b$	C_ia	C_ib	$C_{e_1}a$	$C_{e_1}b$	$C_{e_2}a$	$C_{e_2}b$	C_E
MSE	0	3	20	36	4	12	25	48.5	81
RMSE	0	1.73	4.47	6	2	3.46	5	6.96	9
MAE	0	1.5	3.75	5.25	1.75	3.25	4.75	6.75	9
MSE^N	0	0,04	0,25	0,44	0,05	0,15	0,31	0,6	1
$RMSE^N$	0	0,19	0,5	0,67	0,22	0,38	0,56	0,77	1
MAE^N	0	0,17	0,42	0,58	0,19	0,36	0,53	0,75	1

Tabla 2.4: Resultados de la métricas de regresión sobre el ejemplo

	C_1		C_2		C_3		C_4		
	C_1a	C_1b	C_2a	C_2b	C_3a	C_3b	C_4a	C_4b	
OMAE	1.5	1.5	1	1	1	1	1.5	1.5	...
MMAE	3	3	2	2	2	2	3	3	...
mMAE	0	0	0	0	0	0	0	0	...
AMAE	1.5	1.5	1	1	1	1	1.5	1.5	...
r_s	0	0	0	0	0	0	0	0	...
τ_b	0	0	0	0	0	0	0	0	...
OC_β^γ	0,86	0,86	0,75	0,75	0,75	0,75	0,86	0,86	...
	C_{ok}		C_i		C_{e_1}		C_{e_2}		C_E
	$C_{ok}a$	$C_{ok}b$	C_ia	C_ib	$C_{e_1}a$	$C_{e_1}b$	$C_{e_2}a$	$C_{e_2}b$	C_E
OMAE	0	0	2	2	1	1	2.5	2.5	3
MMAE	0	0	3	3	1	1	3	3	3
mMAE	0	0	1	1	1	1	2	2	0
AMAE	0	0	2	2	1	1	2.5	2.5	0,75
r_s	1	1	-1	-1	0,6	0,6	-0,89	-0,89	0
τ_b	1	1	-1	-1	0,33	0,33	-0,82	-0,82	0
OC_β^γ	0	0	0,98	0,98	0,88	0,88	1	1	1

Tabla 2.5: Resultados de la métricas de clasificación ordinal sobre el ejemplo

0 en el caso de $i = 1$. Por ello tampoco es posible el cálculo de AUC. Además, respecto a la métrica Kappa Cohen hay que notar que presenta un valor igual en el peor de los casos (C_E) que los casos en los que sí existe clasificación correcta de alguna clase (C_1 a C_4). Del resto de métricas hay poco que destacar, exceptuando que consideran con un error o acierto igual a diferentes grupos de casos como de C_1 a C_4 o C_{e_1} y C_{e_2} lo que es esperado ya que

estas métricas no consideran las peculiaridades de los problemas ordinales.

La tabla 2.4 presenta los resultados para las métricas de regresión. En ella se puede observar el funcionamiento esperado con un valor mayor en los subcasos $C_x b$ respecto de los casos $C_x a$. Por otra parte se observan valores distintos en los casos C_1 a C_4 ya que el intervalo de posibles valores de la clase de los "suspensos" es de 1 a 4 siendo mayor que en el resto de clases cuyo intervalo es 1. Por último se observa la diferencia de valores entre las métricas clásicas como MSE, RMSE y MAE, por ello la propuesta de las métricas normalizadas MSE^N, $RMSE^N$ y MAE^N donde se observa una graduación desde un valor 0 para el mejor de los resultados ($C_{ok} a$) hasta el valor 1 en el peor caso (C_E). Con el objetivo de poder comparar en media además de en ranking los resultados de los distintos problemas con distintos rangos de variable de salida es con lo que se definen estas métricas en este capítulo y se utiliza la métrica $RMSE^N$ en el capítulo 5.

En la tabla 2.5 se presentan los resultados para las métricas de clasificación ordinal. En esta tabla se puede observar como casos similares como C_1 a C_4 o C_{e_x} presentan resultados distintos para las métricas. De esta forma se puede decidir si un clasificador ha tenido más error que otro desde el punto de vista ordinal. Por ejemplo se observa que el caso C_{e_1} es ligeramente menos erróneo que C_{e_2} o que los casos C_1 y C_4 presentan la misma clasificación siendo peor que la que presentan C_2 y C_3 que también son iguales entre sí. Respecto a las métricas r_s o τ_b se observa que para los casos C_1 a C_4 presentan el mismo valor aunque los errores en la clasificación son distintos. La métrica OC^{γ}_{β} presenta valores cercanos en los casos C_{e_1} y C_1 o C_4 lo que puede dar lugar a considerar que el error en estos casos es similar mientras que ha habido patrones correctamente clasificados en C_1 o C_4 siendo toda la clasificación errónea en C_{e_1}. Además presenta el mismo valor para los casos C_E y C_{e_2} indicando que el resultado entre ambos es indiferente cuando en realidad son casos distintos.

El objetivo del análisis llevado a cabo en este apartado es la elección de la mejor métrica que pueda guiar el proceso de aprendizaje del algoritmo que se expone en el capítulo 4. Según lo expuesto en los párrafos anteriores, para una clasificación nominal se tendría que descartar las métricas de clasificación TNR, FPR y AUC ya que pueden dar lugar a valores "no determinados" como en el caso C_E. De forma similar se debería descartar la métrica

Kappa Cohen por lo expuesto en el párrafo anterior. Respecto al resto de métricas de clasificación nominal se podría utilizar cualquiera de ellas para un problema de clasificación nominal aunque se observa que ninguna de ellas puede diferenciar los casos C_1 a C_4 por lo que no pueden ser usadas de forma independiente para clasificación ordinal.

Uno de nuestros objetivos es la creación de un algoritmo para clasificación ordinal. Debido a su similitud con regresión, se podría pensar en utilizar una de las métricas de regresión. A pesar de ser indicativas de buenos resultados en clasificación, incluso en clasificación ordinal, hay casos en los que pueden dar lugar a confusión como en el caso de $C_{e_1}a$ en el que no ha habido ninguna clasificación correcta, pero por los valores que se obtienen de las métricas, se puede considerar mejor que cualquiera de los casos de C_1 a C_4 en los que sí ha habido clasificaciones correctas. Otra situación similar ocurre entre los casos $C_{e_1}b$ y C_1b o C_4b.

Otra opción sería utilizar las métricas de clasificación ordinal. De ellas se debería descartar r_s, τ_b y OC_β^γ por lo expuesto anteriormente. Sobre el resto de métricas también pueden dar lugar a un mal guiado en el aprendizaje ya que si tomamos AMAE y mMAE indica que el caso C_E es mejor o igual que los casos de C_1 a C_4. Algo similar ocurre si consideramos MMAE y OMAE con los casos C_e1 y de C_1 a C_4.

Por todo lo anterior, para el guiado del algoritmo de aprendizaje para clasificación ordinal se utilizará una combinación de las métricas más usadas en clasificación nominal y ordinal, esto es, CCR y OMAE como se expondrá en el capítulo 4 de forma que se salven las problemáticas expuestas en los párrafos anteriores.

3

ALGORITMO DE PARTIDA - NSLV

Nunca te das cuenta de lo que ya has hecho.
Sólo puedes ver lo que te queda por hacer.
Marie Curie [1]

ÍNDICE

1 Marie Salomea Sklodowska-Curie (1867 - 1934) científica polaca nacionalizada francesa. Pionera en el campo de la radiactividad.

3.1 MODELO DE REGLA DE NSLV

El objetivo de un algoritmo de aprendizaje inductivo, como el que nos ocupa, consiste en establecer una hipótesis para explicar un conjunto de ejemplos que describen el comportamiento de un determinado sistema. El conocimiento que dicha hipótesis aporta para comprender el sistema viene determinado por el tipo de representación que el algoritmo de aprendizaje utiliza para almacenar la información.

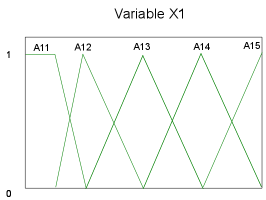
Para el desarrollo de las propuestas que presentamos en esta tesis se utiliza NSLV [58] como base, el cual es una versión mejorada del algoritmo SLAVE [104]. Un estudio más exhaustivo y con mayor detalle en la evolución y distintas versiones de NSLV se presenta en [55, 56, 58, 65].

En el caso de NSLV, la información se almacena usando un conjunto de reglas difusas. La parte antecedente se compone de la conjunción de una o más variables, mientras que la parte consecuente indica el concepto que debe ser asignado a un ejemplo cuando éste se adapte a la parte antecedente. Adicional a la regla difusa se le añade un valor de forma que se indique una ponderación o peso de la regla dentro del conjunto de reglas. Dicho peso se calcula en función de la clasificación de los ejemplos. En resumen, para la representación del conocimiento se utiliza una modificación de las reglas difusas tipo Mamdani, similar a las definidas en 2.1.3, en las que los valores que puede tomar un atributo pueden ser un subconjunto del dominio difuso de la variable asociada a ese atributo y a las que se añade un valor de peso quedando la regla como se indica a continuación:

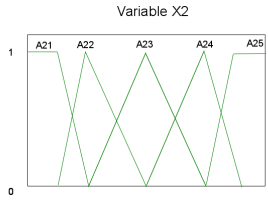
$$\text{IF } X_i \text{ is } \widehat{A_1} \text{ AND } \ldots \text{ AND } X_n \text{ is } \widehat{A_n} \text{ THEN } Y \text{ is } B \text{ WITH WEIGHT } w\,.$$

donde $X_1, \ldots, X_n$ son atributos, $A = (\widehat{A_1}, \ldots, \widehat{A_n})$ son los valores que toma cada atributo. Cada $\widehat{A_i}$ es un subconjunto de D_i, el dominio difuso de X_i. Y es la variable consecuente y B es el valor que toma esta variable consecuente. Esta regla es denotada como $R_B(A)$. Finalmente, $w \in [0..1]$ es la ponderación o peso asociado a la regla.

A modo de ejemplo, si consideramos una regla con dos variables lingüísticas antecedente en la que cada variable lingüística antecedente tiene 5 etiquetas lingüísticas como las presentadas en la figura 3.1, un ejemplo de regla podría ser el siguiente:



(a) Dominio de la variable X1



(b) Dominio de la variable X2

Figura 3.1: Dominio de las variables X1 y X2.

IF X_1 is $\{A_{12}, A_{13}\}$ AND X_2 is $\{A_{21}, A_{22}, A_{23}\}$
THEN Y is $clase1$ WITH WEIGHT 0.5 .

donde $\{A_{12}, A_{13}\}$ y $\{A_{21}, A_{22}, A_{23}\}$ se interpretan como una disyunción de la forma $(A_{12}$ or $A_{13})$ y $(A_{21}$ or A_{22} or $A_{23})$ respectivamente.

3.2 ESTRATEGIA DE RECUBRIMIENTO SECUENCIAL DE NSLV

NSLV [65] es un algoritmo de clasificación basado en el enfoque de *aprendizaje de reglas iterativo* [63] que usa una *estrategia de recubrimiento secuencial* con un *algoritmo genético* para la búsqueda de la mejor regla. La estrategia de recubrimiento secuencial es una estrategia de descomposición en la que en cada paso se va obteniendo una regla de forma iterativa. Esta estrategia reduce el problema del aprendizaje de un conjunto de reglas disyuntivas a un conjunto de problemas de aprendizaje de una única regla.

El algoritmo que implementa la estrategia de recubrimiento secuencial de NSLV se muestra en el algoritmo 2.

La entrada del algoritmo es el conjunto de ejemplos E siendo la salida el conjunto de reglas que tratan de describir el comportamiento de E. Inicialmente el conjunto de reglas está vacío (paso 1). En cada iteración se obtiene una nueva regla (paso 2 y 3.c) que se añade al conjunto de reglas aprendidas (paso 3.a) y se penalizan los ejemplos cubiertos para posibilitar el aprendizaje

Algoritmo 2 Estrategia de Recubrimiento Secuencial de NSLV.

RECUBRIMIENTO_SECUENCIAL (E)

1. Learned_rules ← {}
2. Rule ← LEARN_ONE_RULE (E)
3. **While** (PERFORMANCE (Rule, E) > 0) **Do**
   a) Learned_rules ← Learned_rules + Rule
   b) E ← PENALIZE (Learned_rules, E)
   c) Rule ← LEARN_ONE_RULE (E)
4. **Return** Learned_rules

de nuevas reglas con el mismo o diferente consecuente (paso 3.b). El proceso se repite mientras que la función Performance de la regla extraída sea mayor que cero (paso 3). Finalmente se devuelve el conjunto de reglas aprendidas (paso 4).

Como se expuso en el capítulo 2, para implementar un algoritmo de aprendizaje de reglas basado en un AG que utiliza el enfoque de aprendizaje de reglas iterativo es necesario definir:

- Un *criterio para seleccionar la mejor regla* en cada iteración del algoritmo de recubrimiento secuencial. Este criterio está implementado dentro de la función LEARN_ONE_RULE. Esta función se implementa mediante un AG el cual devuelve la regla que mejor representa el conjunto de ejemplos. El criterio para la elección de la mejor regla está relacionado con una extensión de las condiciones clásicas de consistencia y completitud junto con las condiciones de simplicidad y comprensibilidad como se expone en la sección siguiente.
- Un *criterio de penalización* de ejemplos para la regla aprendida, el cual está implementado en la función PENALIZE.
- Un *criterio de parada* que determine cuándo se han obtenido las reglas que definen de forma adecuada el sistema a modelar. Este criterio se implementa en la función PERFORMANCE.

Estos criterios se exponen con mayor detalle en las secciones siguientes.

3.3 CRITERIO DE SELECCIÓN DE LA MEJOR REGLA

En esta sección se presenta el primero de los criterios que se necesitan definir para implementar un algoritmo de aprendizaje como el que nos ocupa. Este criterio es necesario para la elección de la mejor regla en cada iteración del algoritmo de recubrimiento secuencial y se implementa en la función LEARN_ONE_RULE. El objetivo de esta función es encontrar la regla que más se adapte a los criterios que definen la mejor regla. Para la elección de la mejor regla se utiliza una función fitness definida mediante una extensión de las condiciones clásicas de consistencia y completitud junto con las condiciones de simplicidad y comprensibilidad que son expuestos más adelante. Para proponer una versión difusa de estos conceptos se hace necesario decidir si un ejemplo es positivo o negativo a una regla y, de alguna manera, contar el número de ejemplos en cada caso.

La primera aproximación para el cálculo de ejemplos positivos y negativos está basada en una medida de posibilidad obtenida de la siguiente forma.

Dados a y b dos conjuntos difusos sobre un mismo conjunto de referencia U, y $*$ una t-norma. Definimos la *compatibilidad entre* a *y* b como la siguiente función:

$$\sigma(a, b) = \sup_{x \in U} \{\mu_a(x) * \mu_b(x)\} .$$

donde $\mu_r(s)$ es el grado de pertenencia del valor s al conjunto difuso r definido en 2.1.3.

Para el cálculo de la compatibilidad entre dos conjuntos difusos se puede realizar la consideración siguiente. Sean Dom_1 y Dom_2 dos dominios constituidos por conjuntos difusos sobre un mismo referencial U, y $C_1 \subseteq Dom_1$ y $C_2 \subseteq Dom_2$ dos conjuntos de conjuntos difusos, definimos la *compatibilidad entre estos conjuntos* como:

$$\sigma(C_1, C_2) = \sup_{a \in C_1} \sup_{b \in C_2} \sigma(a, b) .$$

A partir de lo anterior definimos un valor de posibilidad:

$$\mathrm{Poss}(A_i|e_i) = \frac{\sigma(e_i, A_i)}{\sigma(e_i, D_i)} .$$

donde $e = (e_1, e_2, \ldots, e_n)$ es un ejemplo con 'n' variables antecedentes, $A_i \subseteq D_i$ es valor de la variable X_i y D_i es el dominio de esta variable. Este valor representa la adaptación entre el componente $i-esimo$ del ejemplo e y el conjunto difuso A_i.

A continuación podemos definir dos conceptos de adaptación, uno para la parte antecedente y otro para la parte consecuente de la siguiente forma. La adaptación de un ejemplo al antecedente es:

$$U(e, A) = *_{i=1\ldots n}\mathrm{Poss}(A_i|e_i) .$$

donde $*$ es una t-norma y $\mathrm{Poss}(A_i|e_i)$ ha sido definida anteriormente.

La adaptación de un ejemplo al consecuente es definida como:

$$U(e, B) = \frac{\sigma(\mathrm{class}(e), B)}{\sigma(\mathrm{class}(e), F)} .$$

donde $\mathrm{class}(e)$ representa el consecuente del ejemplo e, B es el valor de la variable consecuente y F es el dominio finito asociado a la variable consecuente, siendo $\sigma(a, b)$ la *compatibilidad entre los conjuntos difusos a y b* definida anteriormente.

Haciendo uso de los conceptos anteriores podemos definir los conceptos de mejor adaptación entre el ejemplo e y el conjunto de reglas aprendidas (Rules) cuyo consecuente es la misma clase que el ejemplo ($\lambda^{+}_{\mathrm{Rules}}(e)$) como:

$$\begin{aligned}\lambda^{+}_{\mathrm{Rules}}(e) = \quad & \max\{U(e, A) * U(e, B) \mid \\ & e \in E \text{ and } \forall R_B(A) \in \mathrm{Rules} \text{ and } \mathrm{Class}(e) = B\} .\end{aligned}$$

donde $\mathrm{Rules} = \{R_{B_1}(A_1), \ldots, R_{B_q}(A_q)\}$ es el conjunto de reglas aprendido.

De forma similar se define la mejor adaptación entre el ejemplo e y el conjunto de reglas aprendidas ($Rules$) cuyo consecuente es una clase distinta a la del ejemplo ($\lambda^{-}_{Rules}(e)$) como:

$$\begin{aligned}\lambda^{-}_{Rules}(e) = \quad & \text{máx}\{U(e,A) * U(e,\overline{B}) \mid \\ & e \in E \text{ and } \forall R_B(A) \in Rules \text{ and } Class(e) \neq B\}\,.\end{aligned}$$

A continuación se pueden definir los ejemplos positivos ($n^{+}_{E}(R_B(A))$) y negativos ($n^{-}_{E}(R_B(A))$) a una regla de la siguiente forma.

$$\begin{aligned}n^{+}_{E}(R_B(A)) = \quad & |(U(e,A) * U(e,B)| \text{ tal que} \\ & \lambda^{-}_{Rules}(e) > \lambda^{+}_{Rules}(e) \text{ and} \\ & U(e,A) * U(e,B) > \lambda^{+}_{Rules}(e)\,.\end{aligned}$$

y

$$\begin{aligned}n^{-}_{E}(R_B(A)) = \quad & |(U(e,A) * U(e,\overline{B})| \text{ tal que} \\ & \lambda^{+}_{Rules}(e) > \lambda^{-}_{Rules}(e) \text{ and} \\ & U(e,A) * U(e,\overline{B}) > \lambda^{+}_{Rules}(e)\,.\end{aligned}$$

Con la función fitness se lleva a cabo la evaluación de las reglas para la selección de la mejor regla. En esta función se debe tener en cuenta los conceptos anteriores de forma que se maximice el número de ejemplos correctamente clasificados al mismo tiempo que se minimice el número de ejemplos incorrectamente clasificados. Además se pretende obtener el menor número posible de reglas siendo éstas lo más simples y comprensibles posible. Tendiendo todo esto en cuenta, para el cálculo del fitness se van a utilizar una serie de conceptos que se explican a continuación.

- **Consistencia**: En la lógica clásica, se dice que una regla satisface la condición de *consistencia* cuando la regla satisface una descripción de alguna clase y no de cualquier otra clase. En [64] se definió una extensión suave de esa consistencia con el objetivo de poder adaptar este concepto a la lógica difusa y poder admitir algún ruido en las reglas. Para ello, en primer lugar definimos Δ como el conjunto de todas las posibles reglas y definimos Δ^{k}_{E} como el conjunto de aquellas reglas cuyo número de ejemplos cubiertos positivamente

por la regla ($n_E^+(R_B(A))$) ponderado por un valor k es mayor que el número de ejemplos cubiertos negativamente por esa regla ($n_E^-(R_B(A))$):

$$\Delta_E^k = \left\{ R_B(A) \mid n_E^-(R_B(A)) < k n_E^+(R_B(A)) \right\} .$$

con

$$\Delta_E^0 = \left\{ R_B(A) \mid n_E^-(R_B(A)) = 0 \right\} .$$

De esta forma, el grado en que una regla satisface la consistencia suave dentro de un umbral de ruido $k_1, k_2 \in [0, 1]$ y $k_1 < k_2$ se define a partir de pertenencia a los conjuntos definidos anteriormente de la forma:

$$\Gamma_{k_1,k_2}(R_B(A), E) = \begin{cases} 1 & \text{si } R_B(A) \in \Delta_E^{k_1} \\ \frac{k_2(n_E^+(R_B(A)) - n_E^-(R_B(A)))}{n_E^+(R_B(A))(k_2 - k_1)} & \text{si } R_B(A) \in (\Delta_E^{k_2} - \Delta_E^{k_1}) \\ 0 & \text{en otro caso} \end{cases} .$$

Si fijamos los parámetros $k_1 = 0$ y $k_2 = 1$ obtenemos la siguiente expresión para el grado en que una regla satisface la condición de consistencia suave ($\Gamma_{0,1}(R_B(A), E)$).

$$\Gamma_{0,1}(R_B(A), E) = \frac{n_E^+(R_B(A)) - n_E^-(R_B(A))}{n_E^+(R_B(A))} .$$

- **Completitud**: La condición de *completitud* hace referencia al hecho de que cada ejemplo de una clase debe ser satisfecho por alguna regla de esa clase. Se define una extensión de esta completitud con el objetivo de poder admitir algún ruido en las reglas. Así se define el *grado de completitud de una regla* ($\Lambda(R_B(A), E)$) en función del número de ejemplos cubiertos positivamente por una regla ($n_E^+(R_B(A))$) y el número total de ejemplos de una clase (n_{E_B}) de la forma:

$$\Lambda(R_B(A), E) = \frac{n_E^+(R_B(A))}{n_{E_B}} .$$

donde $n_{E_B} = \sum_{e \in E} U(e, B)$ y ($n_E^+(R_B(A))$) es el número de ejemplos positivos a una regla definido anteriormente.

Con los dos conceptos anteriores se calcula la **Consistencia por completitud** en la que se tiene en cuenta el grado en que una regla satisface tanto la condición de completitud

como la condición de consistencia. Esta medida es la que se utiliza en la función fitness siendo su formulación la siguiente:

$$\Psi(R_B(A), E) = \Gamma_{0,1}(R_B(A), E) \times \Lambda(R_B(A), E) \, .$$

$$\Psi(R_B(A), E) = \frac{n_E^+(R_B(A)) - n_E^-(R_B(A))}{n_{E_B}} \, .$$

- **Simplicidad**: La *simplicidad en variables de una regla* o simplemente *simplicidad* ($svar(R_B(A))$) mide el número de variables irrelevantes de la regla respecto al número de variables totales como se muestra a continuación:

  $$svar(R_B(A)) = \frac{i(R_B(A))}{n} \, .$$

  donde $i(R_B(A))$ es el número de *variables irrelevantes* obtenido como el resultado de restar al número de variables totales, el número de *variables relevantes*. **Variables relevantes** son aquellas cuyo valor de medida de información es mayor o igual que el umbral de esa regla.

- **Comprensibilidad**: La *simplicidad en valores de una regla* o *comprensibilidad* ($sval(R_B(A))$) se mide como el número de variables estables respecto al número total de variables como se muestra a continuación:

  $$sval(R_B(A)) = \frac{1 + e(R_B(A))}{1 + p} \, .$$

  donde $e(R_B(A))$ es el número de *variables estables* y p ($p \leqslant n$) es el número de variables con un dominio ordenado. Por una parte, **variables estables** son aquellas variables que cumplen las dos condiciones siguientes: en primer lugar, el valor de medida de información es mayor o igual que el umbral de esa regla, es decir, son variables relevantes; en segundo lugar, las etiquetas activas de dicha variable son consecutivas, es decir, tienen un sólo intervalo de $1's$ dentro del rango de la variable. Para intentar clarificar este concepto, los siguientes ejemplos corresponderían a la codificación

de una variable estable con 5 etiquetas (11000, 01000, 00111, 01110), no siendo variables estables los siguientes (01010, 10011, 00000). Por otra parte, se considera que **una variable tiene un dominio ordenado** cuando las etiquetas de esa variable están ordenadas. Como ejemplo de variable con dominio ordenado puede ser $altura = \{bajo, medio, alto\}$ siendo $altura = \{medio, bajo, alto\}$ una variable con un dominio no ordenado.

Con todos los conceptos anteriores la función fitness que guiará el proceso de aprendizaje es la siguiente:

$$fitness(R_B(A)) = \quad [\Psi(R_B(A)), svar(R_B(A)), sval(R_B(A))] \, .$$

Una vez obtenidos los valores correspondientes a los conceptos que intervienen en la función fitness se lleva a cabo la ordenación de las reglas en el orden en que son presentados los conceptos, de forma que en primer lugar se tiene en cuenta Ψ, cuando se produce un empate entre varias reglas en este valor, el desempate se produce con el valor del concepto $svar$ y, si es necesario, se considera el valor $sval$ para el desempate. Si después de realizar este proceso sigue produciéndose un empate se procede a la elección de la regla que ocupa la posición más baja en la lista de reglas a seleccionar, es decir, la que se obtuvo antes.

3.3.1 *El Algoritmo Genético de NSLV*

La selección de la mejor regla es el componente principal del algoritmo de recubrimiento secuencial. NSLV usa un AG, implementado en la función LEARN_ONE_RULE, para obtener la mejor regla en cada iteración del algoritmo de recubrimiento secuencial. En este algoritmo, una regla es codificada por un individuo de la población cuya estructura y niveles asociados se exponen a continuación.

Codificación de un individuo de la población

La salida del AG es la mejor regla según los criterios de selección del algoritmo. Esta regla debe ser codificada de forma que sea manejable por el AG. En toda regla hay un antecedente y un consecuente, por lo que el individuo de la población debe tener, al menos, la codificación correspondiente a estos elementos.

Respecto al consecuente estamos ante un problema de clasificación, con lo que se debe guardar una codificación de la clase de la regla. Para codificar la clase de la regla se utiliza un número entero.

En lo que respecta al antecedente hay que tener en cuenta varias consideraciones. En primer lugar, cada variable antecedente se representa mediante una variable lingüística formada por una serie de conjuntos difusos. Además, la regla debe indicar si la etiqueta lingüística asociada al conjunto difuso de la variable correspondiente debe ser considerada o no en la regla. Para codificar la consideración o no de esta etiqueta lingüística se utiliza un valor binario donde cada variable se codifica mediante una cadena de elementos binarios de tamaño igual al número de conjuntos difusos que compone la variable. Finalmente, en el individuo se almacena la concatenación de estas cadenas binarias.

En segundo lugar, con la idea de realizar un "filtrado automático" de las variables antecedentes que intervienen en la regla, se codifica un valor que indique la relevancia de la variable antecedente en la regla. Para ello se almacena un valor real entre el 0 y el 1 para cada variable antecedente, junto a un valor umbral por debajo del cual la variable antecedente no es relevante para la regla.

De esta forma la codificación de un individuo de la población es la que se presenta en la figura 3.2, donde se pueden diferenciar los siguientes niveles:

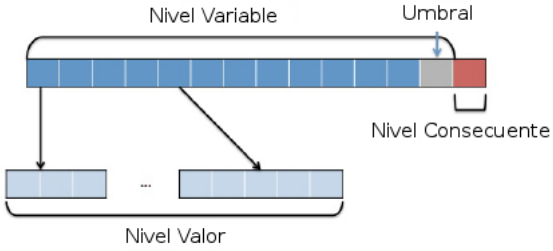


Figura 3.2: Codificación de un individuo de la población del algoritmo genético en NSLV

- **Nivel de Variable**: este nivel contiene un gen por cada variable de entrada del problema. Cada gen representa un valor real que es interpretado como el grado de relevancia de una variable predictiva de la regla. Además, se añade un gen especial en este nivel que es interpretado como un umbral de activación, esto es, las variables cuyo gen asociado tienen valores menores que este umbral no son considerados como relevantes para la regla. Esta consideración permite desarrollar un proceso de selección de características embebido.

- **Nivel de Valor**: está compuesto por una secuencia de asignaciones a las variables predictivas, donde cada asignación variable/valor está representada por una cadena binaria. El tamaño de esta cadena está determinado por el número de etiquetas lingüísticas de cada variable. Un valor igual a 1 indica que la etiqueta lingüística es considerada en la regla, mientras que un valor igual a 0 indica que no es considerada en la regla. Este nivel está compuesto por la concatenación de estas cadenas binarias para todas las variables de entrada.

- **Nivel Consecuente**: codifica el valor de la variable consecuente de la regla. Está compuesto por un gen que es representado por un valor entero.

Para una mejor comprensión vamos a suponer que queremos codificar una regla similar a la mostrada en 3.1 utilizando las variables mostradas en la figura 3.1 en la que encontramos dos variables antecedentes con cinco etiquetas lingüísticas cada una. Vamos a suponer que el valor del nivel de variable de los antecedentes son 0.26 y 0.18 respectivamente, con un valor de umbral de 0.21. Las etiquetas activas del primer antecedente son la segunda y tercera, mientras que el segundo antecedente tiene las tres primeras etiquetas activas. Finalmente el valor que codifica la clase consecuente de la regla es un valor igual a 1. De esta forma, la representación del individuo se muestra en la figura 3.3, siendo su codificación la siguiente:

- Nivel variable: 0,26 0,18 0,21

- Nivel valor: 01100 11100

- Nivel consecuente: 1

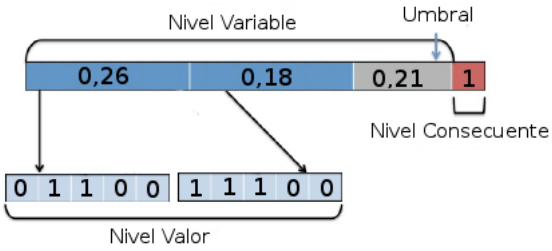


Figura 3.3: Ejemplo de codificación de un individuo de la población del algoritmo genético en NSLV

Con estos valores de nivel de variable, la regla correspondiente a este ejemplo sería la siguiente:

$$\text{IF } X_1 \text{ is } \{A_{12}, A_{13}\} \text{ THEN } Y \text{ is } 1$$

donde no aparece la segunda variable antecedente ya que el valor en el nivel de variable es menor que el umbral.

Inicialización de la población

La inicialización de la población es muy importante en los AG ya que pueden condicionar el guiado del proceso de búsqueda hacia una buena solución, así como el tiempo de procesamiento en este proceso de búsqueda. En NSLV la inicialización de la población tiene en cuenta los ejemplos de entrenamiento como se muestra a continuación.

En primer lugar se seleccionan, de forma aleatoria, tantos ejemplos como individuos va a componer la población, distribuidos uniformemente en función del número de clases del problema, de forma que el individuo i pertenezca a la clase "i módulo $numClases$". De esta forma se pretende cumplir dos objetivos. Por una parte, se pretende que la convergencia hacia la mejor regla sea lo más rápida posible. Por otra parte se pretende que en la población estén representados todas las clases del problema con lo que se podrán obtener reglas para todas las clases del problema.

La inicialización de cada nivel del individuo expuesto en la sección anterior es como sigue:

- En lo que respecta a la codificación binaria se inicializará cada bloque correspondiente a una variable con todo ceros y un uno en el lugar que corresponda con la etiqueta de mayor adaptación del ejemplo con la variable. Por ejemplo si tenemos un problema con dos variables antecedentes y la mayor adaptación del ejemplo con la primera variable se corresponde con la etiqueta segunda y la mayor adaptación del ejemplo con la variable segunda se corresponde con la etiqueta quinta, la codificación binaria del individuo sería 01000 00001. En el caso de que no haya ejemplos de una clase determinada se usará la probabilidad inicial de la codificación binaria para asignar de forma aleatoria valores 0 y 1 a esta codificación del individuo.
- La inicialización de la codificación entera del individuo se realiza estableciendo un valor aleatorio dentro del rango de enteros.
- Por último, la codificación real hace referencia al grado de relevancia de la variable en cuestión. Para esta estimación se usa una medida de información basada en la medida de entropía de Kullback-Leiber [86]. Una descripción más detallada de su uso se presenta en [56]. Para la inicialización de la codificación real se toma el máximo y el mínimo de esta medida de información de cada clase considerando todas las variables y seleccionando de forma aleatoria un valor real en ese intervalo.

Evolución del AG

Una vez se tiene la población inicializada y definidas las funciones utilizadas en el algoritmo de recubrimiento secuencial, pasamos a la descripción de la evolución del AG cuyo resultado es la regla a incluir en el conjunto de reglas final. Como se ha expuesto en las secciones anteriores, NSLV utiliza una modificación del algoritmo de recubrimiento secuencial en el que van compitiendo todas las clases al mismo tiempo realizándose el aprendizaje de una clase determinada en una ejecución y de otra clase distinta, o la misma incluso, en la ejecución siguiente. Además, en este aprendizaje no se eliminan los ejemplos cubiertos, sino que se marcan para

modificar la valoración de su consideración en las ejecuciones siguientes.

El AG que se usa en NSLV es un AG *de estado estacionario* en el que la población está compuesta de subpoblaciones o nichos de clases a aprender, de forma que se asegura la no extinción de nichos de la población, evitando así la desaparición de clases. Además, de esta forma se tiene en cuenta el mantenimiento de la diversidad de la población, asegurando que hay individuos de todas las clases para poder realizar el aprendizaje de todas ellas. Al ser un algoritmo de estado estacionario, se permite que las poblaciones se mantengan de una iteración del algoritmo a otra, con lo que se pretende reducir el tiempo de aprendizaje al no tener que crear poblaciones iniciales en cada iteración y debido a que la población con la que se trabaja en una iteración contiene los mejores individuos de la iteración anterior.

El reemplazo de la población se produce mediante una sustitución de los peores individuos. El mecanismo de reemplazo consiste en seleccionar dos individuos de la población y aplicar los cruces y mutaciones correspondientes para después introducir los nuevos individuos, eliminando los dos peores de aquella subpoblación en la que no hay riesgo extinción. Se considera que una subpoblación está en riesgo de extinción cuando tiene menos de m individuos siendo $m = N_{poblaciones}/(N_{clases} + 1)$ donde $N_{poblaciones}$ es el número de individuos de la población y N_{clases} es el número de clases del problema.

Operadores genéticos

Los operadores genéticos usados en NSLV para generar los nuevos individuos son los siguientes:

- **Operador de selección**: como se ha expuesto anteriormente, los individuos de la población son ordenados según la evaluación de la función fitness. La selección de los mismos se lleva a cabo de forma aleatoria evitando seleccionar dos veces el mismo individuo.

- **Operador de cruce de dos puntos**: para la actuación de este operador de forma genérica, se establecen dos puntos de corte y se intercambian las partes centrales de los individuos como indica la figura 3.4. Para la realización del operador de cruce se establecen unas probabilidades para cada nivel

del individuo realizándose el cruce en cada nivel como se expone en los párrafos siguientes.

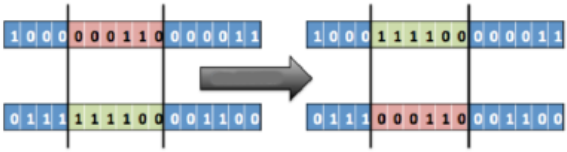


Figura 3.4: Operador de cruce de dos puntos

- **Operador de mutación**: en este caso se lleva a cabo una mutación uniforme y, al igual que el cruce, para la actuación de la mutación se tiene en cuenta las probabilidades de mutación de cada nivel. Además en cada nivel se realiza la mutación de forma diferente como se expone a continuación.

En el caso de NSLV, los operadores genéticos actúan en cada nivel de la siguiente forma:

- **Nivel de Variable**: contiene un gen por cada variable de entrada del problema junto a un valor de umbral de activación. Los valores de este nivel son valores reales. Los dos operadores genéticos usados en este nivel son el operador de cruce de dos puntos y el operador de mutación uniforme. El cruce se realiza incluyendo el umbral ya que éste se considera dentro del nivel de variable. Para la mutación, en función de la probabilidad de mutación evaluada en cada gen, se aumenta o disminuye un 10 % el valor de este nivel dentro de los rangos permitidos sin realizar mutación alguna sobre el último valor que es el valor umbral.

- **Nivel de Valor**: Los valores de este nivel son valores binarios formados por la concatenación de cadenas binarias que representan las adaptaciones de las etiquetas de las variables de entrada. Los dos operadores genéticos usados en este nivel son el cruce de dos puntos y la mutación binaria uniforme. El cruce se realiza sobre todo el nivel, independientemente de los elementos del nivel (bits) que correspondan a una variable u otra. La mutación en este caso, al tratarse de elementos binarios, se realiza modificando

el valor de 1 a 0 y viceversa. Estas operaciones se realizan en función de las probabilidades de mutación y cruce de este nivel.

- **Nivel Consecuente**: codifica el valor de la variable consecuente de la regla y está formado por un valor entero. Como sólo es un valor no tiene sentido el operador de cruce de dos puntos. La mutación entera uniforme es el único operador genético considerado en este nivel. Esta mutación se podría realizar, al igual que la del nivel de variable, aumentando o disminuyendo valores del entero dentro del rango, pero en esta ocasión se ha optado por tomar un valor entero aleatorio dentro del rango y distinto del actual. Igual que en los casos anteriores, para la mutación se tiene cuenta la probabilidad de mutación de este nivel.

Condición de finalización del AG

Al finalizar la ejecución del AG se obtiene la mejor regla. Este final de la ejecución se produce cuando se verifican algunas de las siguientes condiciones:

- El número de iteraciones es mayor que un valor determinado
- Hay seleccionada una regla con un valor fitness que no mejora en un número fijo de iteraciones.

3.4 CRITERIO DE PENALIZACIÓN

El criterio de penalización de ejemplos se implementa en la función PENALIZE. NSLV no lleva a cabo una eliminación de ejemplos en el aprendizaje, sino que los marca de forma que modifica la forma en que son considerados en función de la clase de la regla y el ejemplo. De esta forma, en NSLV no se consideran los ejemplos cubiertos o no por una regla como tal, sino que se asocia un cierto grado de cobertura positiva o negativa que indica el grado en el que el ejemplo es correcta o incorrectamente clasificado por el conjunto de reglas aprendidas utilizando los conceptos presentados en 3.3. La evaluación de una regla es guiada por los ejemplos no cubiertos por las reglas anteriores. Los ejemplos

marcados no afectan a la regla de forma positiva pero sí influyen en la evaluación de las reglas que pertenecen a otras clases.

3.5 CRITERIO DE PARADA

El criterio de parada para el algoritmo de recubrimiento secuencial de NSLV se implementa mediante la función PERFORMANCE. Esta función tiene como entradas una regla (Rule) y un conjunto de ejemplos (E) devolviendo como salida un valor que indica cómo de representativa es esta regla sobre el conjunto de ejemplos. Este valor está relacionado con la función de evaluación fitness y los conceptos que en ella se utilizan y que son definidos en la sección 3.3. Cuando el valor devuelto por esta función es mayor que cero significa que esta regla mejora el desarrollo del algoritmo por lo que se añade la regla y se continúa con el aprendizaje. En caso contrario, se da por terminado el aprendizaje y el algoritmo devuelve el conjunto de reglas aprendidas.

3.6 MODELO DE INFERENCIA

NSLV utiliza el mecanismo de inferencia de la regla ganadora. En el cálculo de esta regla ganadora se considera la adaptación del ejemplo al antecedente definido en la sección 3.3, así como el peso asociado a una regla. Haciendo uso de los conceptos presentados en la sección 3.3 se puede definir el *peso asociado a una regla* como:

$$\omega_E(R_B(A)) = \frac{n_E^+(R_B(A)) + 1}{n_E^+(R_B(A)) + n_E^-(R_B(A)) + 1} .$$

De esta forma, dados un conjunto de reglas (Rules) formado por q reglas de la forma ($Rules = \{R_{B_1}(A_1), \ldots, R_{B_q}(A_q)\}$ con pesos ($\Omega = \{\omega_1, \ldots, \omega_q\}$) y un ejemplo e, el motor de inferencia asigna al ejemplo e la clase B_j asociado a la regla $R_{B_j}(A_j)$ si se verifica que

$$j = \underset{0 \leqslant i \leqslant q}{\operatorname{argmax}} \{U(e, A_i) * \omega_i\} .$$

En caso de empate se tomaría en primer lugar la regla con mayor peso. Si además de esta consideración sigue habiendo un empate, se tomaría la regla que ha sido aprendida primero.

4

EXTENSIÓN DEL MODELO DE CLASIFICACIÓN NOMINAL A CLASIFICACIÓN ORDINAL (NSLVORD)

La imaginación es más importante que el conocimiento.
El conocimiento es limitado, mientras que la imaginación no.
Albert Einstein [1]

ÍNDICE

1 Albert Einstein (1879 - 1955) físico alemán de origen judío nacionalizado suizo y estadounidense. Es considerado como el científico más conocido y popular del siglo XX.

4.1 INTRODUCCIÓN

Los *Problemas de Clasificación Ordinal* [92, 99] son problemas de aprendizaje supervisado que comparten características con la Clasificación Nominal [7] y la Regresión [35]. Respecto a la Clasificación Nominal, la Clasificación Ordinal también aprende clases, pero en este caso hay un orden establecido entre las clases a aprender. Respecto a la Regresión, la Clasificación Ordinal aprende un conjunto ordenado de clases con la diferencia de que el número de clases está formado por un conjunto finito y la distancia entre los valores de las clases no está definido.

La principal diferencia entre los problemas de clasificación nominal y los problemas de clasificación ordinal es la existencia de un orden establecido entre las clases en el caso de la clasificación ordinal. Por ello, los errores en la clasificación son significativamente diferentes en función de la distancia a la clase correcta. Como ejemplo de problema de clasificación ordinal y para enfatizar la importancia de estos errores, podemos considerar el riesgo crediticio asociado con las obligaciones financieras de una empresa [75] en el que, por ejemplo, si consideramos el ratio de crédito a largo plazo de S&P encontramos las clases ($AAA, AA+, AA, AA-, A+, ..., C, RD, SD, D$). En este ejemplo, si consideramos que la clasificación actual de una empresa es AA y un algoritmo lo clasifica como $AA-$ y otro algoritmo lo clasifica como D, ambos algoritmos han realizado una clasificación errónea, pero el error en el segundo algoritmo es mayor que en el primero, lo que podría acarrear decisiones problemáticas en la empresa.

En la literatura podemos encontrar diferentes formas de abordar la clasificación ordinal. Algunos trabajos [22, 23, 87] tratan la clasificación ordinal convirtiendo la clase a una cantidad numérica en una fase de preprocesamiento y aplicando un algoritmo de regresión a los datos transformados. Una vez obtenidos los resultados, se vuelve a realizar otra transformación de vuelta a un valor discreto en una fase de posprocesamiento. La desventaja de este método es que sólo puede ser aplicado en conjunción con un esquema de regresión. Otra aproximación es convertir el aprendizaje de un clasificador ordinal en un conjunto de clasificadores binarios [44, 79, 88]. Los clasificadores binarios correspondientes son entrenados utilizando un algoritmo de clasificación nominal bien conocido realizando finalmente otra conversión para con-

seguir el resultado. Finalmente otras propuestas [37, 38, 46, 91] hacen uso de técnicas de boosting y gradiente descendente para abordar este tipo de problemas.

En un primer intento para abordar los problemas de clasificación ordinal, se realizó una aproximación basada en la idea presentada en [9]. En ella se pretendía solucionar este tipo de problemas creando un multiclasificador jerárquico. Debido a los malos resultados obtenidos, se plantea el desarrollo de un algoritmo específico para este tipo de problemas, para lo cual se lleva a cabo una serie de modificaciones de fondo sobre el algoritmo de clasificación nominal NSLV que se toma como base.

En este capítulo se presenta la modificación del algoritmo de clasificación nominal NSLV [65] para abordar problemas de clasificación ordinal. Las modificaciones que se presentan son las necesarias para convertirlo en un algoritmo propio y ajustado para clasificación ordinal. Para su adaptación a este tipo de problemas se plantean varias modificaciones en distintos componentes del algoritmo y desde diferentes consideraciones. En primer lugar se debe considerar la graduación de una clasificación errónea de un ejemplo a una clase a la que no pertenece, es decir, en el algoritmo original los valores correspondientes a la asociación de un ejemplo a una clase a la que no pertenece es la misma, cuestión que debe cambiar para considerar las peculiaridades de la clasificación ordinal. La modificación de estos valores produce una modificación de todos los conceptos derivados incluido el peso de una regla, por lo que, en segundo lugar se debe dar mayor relevancia a los pesos de las reglas. Por último, se debe modificar la función objetivo del AG incluyendo la consideración de las diferentes distancias entre los errores de la clasificación según el análisis de métricas y las conclusiones expuestas en 2.3.4. Igualmente, con la idea de mejorar el algoritmo aunque sin relación directa con la clasificación ordinal se proponen otras mejoras que son presentadas en las secciones siguientes.

4.2 PROPUESTA

En esta sección presentamos la propuesta de algoritmo de clasificación ordinal llamado "NSLV Ordinal (NSLVOrd)". NSLVOrd está basado en el algoritmo NSLV [58] el cual ha sido expuesto en el capítulo 3 y es usado en resolución de problemas de clasifi-

cación nominal [57]. A este algoritmo se le realizan una serie de modificaciones y mejoras con el que se consigue otro algoritmo específico para problemas de clasificación ordinal.

El algoritmo base (NSLV) utiliza un AG para la búsqueda de la mejor regla junto la estrategia de recubrimiento secuencial para el aprendizaje de las reglas difusas que definen el comportamiento del sistema. Las mejoras que aquí se presentan abarcan tanto la estrategia de recubrimiento secuencial como las que tienen relación con el AG.

Respecto al AG se proponen varias modificaciones de los conceptos utilizados por NSLV y presentados en 3.3 con el objetivo principal de adaptación del algoritmo a la clasificación ordinal proponiendo al mismo tiempo mejoras aplicables al AG de base. También se propone la inclusión de un nuevo criterio que considera los éxitos y los errores de una regla ponderados con la distancia entre la clase real y la predicha.

Respecto a la estrategia de recubrimiento secuencial se presentan ligeras modificaciones con el objetivo de obtener un aprendizaje más óptimo y con una base de reglas más simple. Estas modificaciones se presentan en las secciones siguientes.

4.2.1 *Alternativas genéricas de mejora aplicables al algoritmo base*

Adicionalmente, se proponen y valoran otras alternativas de mejora las cuales se presentan a continuación de forma breve.

En un primer lugar se plantea una idea similar a la ascensión de colinas [115] para mejorar la regla obtenida por el AG. La propuesta funciona de manera que una vez obtenida la mejor regla mediante el AG, se comprueban los vecinos de una distancia de Hamming menor o igual a 1 con lo que se podría conseguir mayor generalidad y llegar a un resultado más simple. Para la consideración de vecinos de distancia 1 puede tomarse la codificación de los distintos antecedentes del individuo como un todo (opción a). De esta forma si el individuo está formado por "n" antecedentes y cada antecedente "i" tiene "b_i" bits para su codificación se tendrían un total de $(\sum_{i=1}^{n} b_i) + 1$ combinaciones. Por otra parte se puede considerar los vecinos de distancia menor o igual a 1 para cada uno de los antecedentes de forma independiente y después considerar sus combinaciones, con lo que hay un mayor número de individuos a evaluar (opción b), obteniéndose un total

de $\prod_{i=1}^{n}(b_i + 1)$ combinaciones. Un ejemplo de estas alternativas se muestran en la tabla 4.1 donde para un individuo codificado como 10101 101 11000 con tres antecedentes codificados con 5, 3 y 5 bits respectivamente se obtienen un total de 14 combinaciones para la opción a) y 144 combinaciones para la opción b). Debido a la cantidad de combinaciones no se han presentado todas, pero se indica con **negrita** los bits que se modifican respecto a la combinación anterior y con *itálica* el grupo de bits correspondientes a los antecedentes que no se modifican respecto a la combinación anterior. Después de diversas pruebas se llega a la conclusión de que en función del tamaño del individuo puede ser muy costoso en tiempo de procesamiento y en muchos casos no se llega a una mejora significativa por lo que se descarta dando pie a la siguiente idea.

En segundo lugar, con esta misma idea se plantea otro mecanismo de generalización de la regla aprendida más simple pero que puede dar mejores resultados. En esta ocasión se consideran las etiquetas que están activas en los antecedentes (con valor del bit igual a 1) y se lleva a cabo una búsqueda de posibles uniones entre las etiquetas activas en estos antecedentes de forma que la regla cubra el máximo espacio posible del universo de la variable antecedente con la que se esté trabajando, siempre, claro está, teniendo en cuenta que el comportamiento al menos se mantenga. Para una mejor comprensión de esta alternativa se utiliza el mismo ejemplo anterior. Si representamos el primer antecedente como muestra la figura 4.1 en la que se sombrean las etiquetas activas, se observa que entre cada par de etiquetas activas puede haber una etiqueta no activa (no sombreada) a considerar. Con esta consideración se consigue una generalización de la regla que podría coincidir con la lógica que gobierna el sistema. De esta forma pasamos de considerar "combinaciones entre bits en general" a considerar "combinaciones entre bits con valor 1" en cada antecedente, o lo que es lo mismo a "combinaciones de conjuntos de bits a valor cero existentes entre dos bits a valor uno ($conjCeros_i$)". Con esta alternativa se consigue una forma de evaluación de vecinos más lógica que la opción a) y similar a la utilizada en la opción b) pero con una reducción notable del número de combinaciones, pasando a un total de $\sum_{i=1}^{n} 2^{conjCeros_i}$ considerando las combinaciones entre todos los antecedentes. En el caso del ejemplo anterior se tendrían tres conjuntos de ceros con lo que hay 8 combinaciones posibles que son mostradas en la tabla 4.2

opcion a)	opción b)	opcion a)	opción b)
10101 101 11000	10101 101 11000		
10101 101 11001	*10101 101 11001*		
10101 101 11010	*10101 101 11010*		
10101 101 11100	*10101 101 11100*	10001 101 11000	*10001 101 11000*
10101 101 10000	*10101 101 10000*		*10001 101 11001* ... (4 +)
10101 101 01000	*10101 101 01000*		*10001 101 01000*
10101 100 01000	*10101 100 11000*		*10001 100 11000*
	10101 100 11001 ... (4 +)		*10001 100 11001* ... (4 +)
	10101 100 01000		*10001 100 01000*
10101 111 01000	*10101 111 11000*		*10001 111 11000*
	10101 111 11001 ... (4 +)		*10001 111 11001* ... (4 +)
	10101 111 01000		*10001 111 01000*
10101 001 01000	*10101 011 11000*		*10001 011 11000*
	10101 011 11001 ... (4 +)		*10001 011 11001* ... (4 +)
	10101 011 01000		*10001 011 01000*
10100 101 11000	*10100 101 11000*	11101 101 11000	*11101 101 11000*
	10100 101 11001 ... (4 +)		*11101 101 11001* ... (4 +)
	10100 101 01000		*11101 101 01000*
	10100 100 11000		*11101 100 11000*
	10100 100 11001 ... (4 +)		*11101 100 11001* ... (4 +)
	10100 100 01000		*11101 100 01000*
	10100 111 11000		*11101 111 11000*
	10100 111 11001 ... (4 +)		*11101 111 11001* ... (4 +)
	10100 111 01000		*11101 111 01000*
	10100 001 11000		*11101 001 11000*
	10100 001 11001 ... (4 +)		*11101 1001 11001* ... (4 +)
	10100 001 01000		*11101 1001 01000*
10111 101 11000	*10111 101 11000*	00101 101 11000	*00101 101 11000*
	10111 101 11001 ... (4 +)		*00101 101 11001* ... (4 +)
	10111 101 01000		*00101 101 01000*
	10111 100 11000		*00101 100 11000*
	10111 100 11001 ... (4 +)		*00101 100 11001* ... (4 +)
	10111 100 01000		*00101 100 01000*
	10111 111 11000		*00101 111 11000*
	10111 111 11001 ... (4 +)		*00101 111 11001* ... (4 +)
	10111 111 01000		*00101 111 01000*
	10111 001 11000		*00101 001 11000*
	10111 001 11001 ... (4 +)		*00101 001 11001* ... (4 +)
	10111 001 01000		*00101 001 01000*

Tabla 4.1: Codificación de ejemplo para distancia de Hamming igual a 1

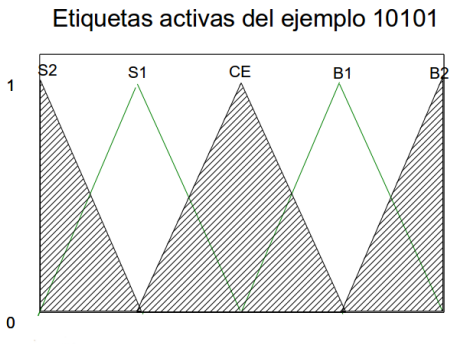


Figura 4.1: Etiquetas activas del antecedente (10101) del ejemplo

10101 101 11000
10101 111 11000
11101 101 11000
11101 111 11000
10111 101 11000
10111 111 11000
11111 101 11000
11111 111 11000

Tabla 4.2: Codificación de ejemplo para la unión de etiquetas activas de los antecedentes

Por último, otra propuesta de mejora es no fijar la clase en el aprendizaje, sino que en la evolución de la población se consideren todas las clases. Para ello, un individuo no está asociado a una clase concreta, si no que a la hora de obtener la regla se selecciona aquella clase que mejor comportamiento presenta como se expone más adelante. Esta alternativa, al igual que la expuesta en el párrafo anterior, obtiene buenos resultados por lo que se incluye en la propuesta que aquí se presenta.

4.2.2 *Modificaciones en el Algoritmo Genético (AG)*

El AG del algoritmo base NSLV está ideado para obtener una buena clasificación nominal. El AG de esta propuesta debe estar orientado a clasificación ordinal. La principal diferencia entre estos dos tipos de problemas es la consideración de un orden entre las clases. Para considerar estas diferencias se proponen las siguientes modificaciones en el AG.

Adaptación de conceptos a la clasificación ordinal

Con el objetivo de tener en cuenta esta diferencia en el caso de una clasificación errónea se redefine la adaptación entre un ejemplo y una clase diferente a la perteneciente al ejemplo como sigue:

$$\widetilde{U(e,\overline{B})} = |Rank(B) - Class(e))| \,.$$

siendo $Rank(B)$ el ranking en la lista de clases ordenada.

Partiendo de esta redefinición y considerando que se debe dar mayor relevancia a los pesos de las reglas por lo expuesto en la sección anterior, se hace necesario redefinir los conceptos en los que interviene y que se presentan a continuación.

Para ello, en primer lugar se redefine la mejor adaptación entre el ejemplo e y el conjunto de reglas aprendidas (Rules) cuyo consecuente es la misma clase que el ejemplo ($\lambda^{+}_{Rules}(e)$) de la siguiente forma:

$$\begin{aligned}\lambda^{+}_{Rules}(e) = \quad & \text{máx}\{U(e,A) * U(e,B) * \omega_E(R_B(A)) \mid \\ & e \in E \text{ and } \forall R_B(A) \in Rules \text{ and } Class(e) = B\} \,.\end{aligned}$$

Similar a la anterior, se redefine la mejor adaptación entre el ejemplo e y el conjunto de reglas aprendidas (Rules) cuyo consecuente es una clase distinta a la del ejemplo ($\lambda^{-}_{Rules}(e)$) como:

$$\begin{aligned}\lambda^{-}_{Rules}(e) = \quad & \text{máx}\{U(e,A) * \widetilde{U(e,\overline{B})} * \omega_E(R_B(A)) \mid \\ & e \in E \text{ and } \forall R_B(A) \in Rules \text{ and } Class(e) \neq B\} \,.\end{aligned}$$

Se hace necesario la definición de dos nuevos conceptos, estos son los pesos de las reglas que consiguen esta máxima adaptación.

Con ello tendríamos $\omega_{\lambda^+_{Rules}(e)}$ asociado a $\lambda^+_{Rules}(e)$ y $\omega_{\lambda^-_{Rules}(e)}$ asociado a $\lambda^-_{Rules}(e)$ respectivamente definiéndose como sigue:

$$\begin{aligned}\omega_{\lambda^+_{Rules}(e)} = \quad & \{\omega_E(R_B(A)) \mid \\ & \lambda^+_{Rules}(e) = (U(e,A) * U(e,B) * \omega_E(R_B(A)))\}\,.\end{aligned}$$

$$\begin{aligned}\omega_{\lambda^-_{Rules}(e)} = \quad & \{\omega_E(R_B(A)) \mid \\ & \lambda^-_{Rules}(e) = (U(e,A) * \widetilde{U(e,B)} * \omega_E(R_B(A)))\}\,.\end{aligned}$$

Además se introducen otros dos nuevos conceptos que son los conceptos de cobertura positiva ($covering^+_E(R_B(A), Rules)$) y cobertura negativa ($covering^-_E(R_B(A), Rules)$) de una regla ($R_B(A)$) sobre un conjunto de ejemplos (E) considerando un conjunto de reglas aprendidas ($Rules$). Estos conceptos que servirán para posteriormente redefinir el número de ejemplos positivos ($n^+_E(R_B(A))$) y negativos ($n^-_E(R_B(A))$) a una regla.

El concepto de cobertura positiva ($covering^+_E(R_B(A), Rules)$) hace referencia al conjunto de ejemplos que pertenecen a la clase de la regla y que no estando cubiertos por el conjunto de reglas aprendidas, se cubren al incorporar la nueva regla. Este concepto se define de la siguiente forma:

$$\begin{aligned}covering^+_E(R_B(A), Rules) = \{e \in E \mid\ & Class(e) = B \\ \text{and } \lambda^-_{Rules}(e) > \lambda^+_{Rules}(e) \text{ and } & \begin{cases} U(e,A) * \omega(R_B(A)) > \lambda^-_{Rules}(e) \\ \text{OR} \\ U(e,A) * \omega(R_B(A)) = \lambda^-_{Rules}(e)) \\ \text{and } \omega(R_B(A) > \omega_{\lambda^-_{Rules}(e)} \end{cases}\end{aligned}\,.$$

De forma similar, se define el concepto de cobertura negativa ($covering^-_E(R_B(A), Rules)$ que indica el conjunto de ejemplos que no siendo de la clase de la regla, al incluir la nueva regla los clasifica de forma errónea ya sea porque antes no estaban clasificados y ahora los clasifica erróneamente, o porque antes sí estaban bien clasificados y ahora los desclasifica. La definición de este concepto es la siguiente:

$$
\begin{aligned}
&\text{covering}_E^-(R_B(A), \text{Rules}) = \{e \in E \mid \text{Class}(e) \neq B \\
&\text{and} \begin{cases}
\lambda^-_{\text{Rules}}(e) > \lambda^+_{\text{Rules}}(e) \text{ and} & \begin{cases} U(e,A) * \omega(R_B(A)) > \lambda^-_{\text{Rules}}(e) \\ \text{OR} \\ U(e,A) * \omega(R_B(A)) = \lambda^-_{\text{Rules}}(e)) \\ \text{and } \omega(R_B(A) > \omega_{\lambda^-_{\text{Rules}}(e)} \end{cases} \\
\text{OR} & \\
\lambda^-_{\text{Rules}}(e) < \lambda^+_{\text{Rules}} \text{ and} & \begin{cases} U(e,A) * \omega(R_B(A)) > \lambda^+_{\text{Rules}}(e) \\ \text{OR} \\ U(e,A) * \omega(R_B(A)) = \lambda^+_{\text{Rules}}(e)) \\ \text{and } \omega(R_B(A) > \omega_{\lambda^+_{\text{Rules}}(e)} \end{cases}
\end{cases}
\end{aligned}
$$

Finalmente, haciendo uso de los conceptos anteriores se redefine el número de ejemplos positivos ($n^+_E(R_B(A))$) y negativos ($n^-_E(R_B(A))$) a una regla, conceptos que se usan en la definición de la función fitness.

De esta forma, se define el número de ejemplos positivos a una regla ($n^+_E(R_B(A))$) como:

$$n^+_E(R_B(A)) = \sum_{e \in \text{covering}^+_E(R_B(A), \text{Rules})} (U(e,A) \times U(e,B) \times \omega(R_B(A))) .$$

Igualmente, la redefinición del número de ejemplos negativos ($n^-_E(R_B(A))$) es:

$$n^-_E(R_B(A)) = \sum_{e \in \text{covering}^-_E(R_B(A), \text{Rules})} (U(e,A) \times \widetilde{U(e,\overline{B})} \times \omega(R_B(A))) .$$

Nuevo criterio en la función fitness

El aprendizaje va guiado por una función fitness que indica de alguna forma la cercanía a la mejor solución. En NSLV la función fitness está formada por tres componentes que son evaluados según su orden lexicográfico. En primer lugar se encuentra la "consistencia por completitud" que mide el grado en que una regla satisface tanto la condición de completitud como la condición de consistencia, es decir, mide el grado en que una regla satisface una descripción de una clase y no de cualquier otra, al mismo tiempo que mide que tenga el menor número de ejemplos sin satisfacer. En segundo lugar se encuentra la "simplicidad" que, en

cierto modo, mide el grado en que una regla está formada por menos antecedentes. Por último se encuentra la "comprensibilidad" que mide, de alguna forma, el grado simplicidad en los antecedentes de la regla. En estos componentes no hay nada que obtenga ventaja de las peculiaridades de la clasificación ordinal. Con el objetivo de introducir estas características en la función fitness se puede plantear un criterio resultado del análisis de las distintas métricas para los diferentes tipos de problemas presentado en 2.3. Después de diversas pruebas con estas métricas y combinaciones de las mismas se observa que la que mejor resultado presenta es la modificación de Error Medio Absoluto Ordinal (Ordinal Mean Absolute Error en Inglés) (OMAE) junto con Precisión (Correct Classification Rate en Inglés) (CCR) pero modificada de forma que se puedan comparar y combinar. Este nuevo criterio considera la ponderación de los éxitos y los errores de la regla haciendo uso de las definiciones presentadas en la sección anterior.

Para tener en cuenta el éxito de una regla, proponemos una medida basada en la métrica CCR en la que se considera el número de ejemplos bien clasificados respecto al número total de ejemplos. Además, para considerar la ponderación de los errores de una regla, se usa una idea subyacente en la métrica OMAE donde se propone una medida de los ejemplos negativos respecto al total de ejemplos. Finalmente, estas medidas de éxito y error se normalizan en el mismo intervalo para que sean mutuamente comparables. De esta forma, la fórmula para el nuevo criterio queda de la siguiente manera:

$$\Phi(R_B(A)) = \frac{\frac{n_E^+(R_B(A))}{n_E} + (1 - \frac{n_E^-(R_B(A))}{n_E * (Q-1)})}{2} .$$

donde n_E representa el número total de ejemplos y Q es el número de clases del problema.

Este nuevo criterio se añade al principio de la función de evaluación quedando ésta como sigue:

$$fitness(R_B(A)) = [\Phi(R_B(A)), \Psi(R_B(A)), svar(R_B(A)), sval(R_B(A))] .$$

Independencia del consecuente en la búsqueda de la regla

La modificación que aquí se propone no tiene relación directa con las peculiaridades de los problemas de clasificación ordinal, por lo que puede ser aplicable al algoritmo que se utiliza como base. Se presenta una modificación en el manejo del nivel de consecuente en la búsqueda de la mejor regla. Esta modificación pretende no fijar el consecuente, sino que se realiza un aprendizaje conjunto de todos los consecuentes. De esta forma, se lleva a cabo un barrido más amplio del espacio de búsqueda, al mismo tiempo que se realiza una competición entre la adecuación de los ejemplos a las reglas, fijando el consecuente al final del proceso de búsqueda. Para ello, se almacenan los distintos valores de los fitness de todas las clases del problema con lo que se pretende conseguir un aprendizaje de la regla sin estar condicionado al establecimiento del consecuente a priori.

Con esta modificación se consiguen varias mejoras en diferentes aspectos. Respecto a la población, se posibilita la disminución del número de individuos al tener en consideración a todas las clases en todos los individuos. Igualmente, el mantenimiento de la diversidad se considera desde el momento en que se consideran todas las clases en cada individuo.

Respecto a los operadores genéticos, en el caso de los niveles de variable y valor no hay variaciones. En lo que respecta al nivel de consecuente no tiene sentido operador alguno, ya que estos valores son calculados.

Por último, respecto a la evolución del algoritmo, a pesar de que se invierte más tiempo en la evaluación del fitness de todas las clases del problema, se comprueba que la convergencia es más rápida y más completa, en lo que a espacio de búsqueda se refiere, al tener en consideración todas las clases.

4.2.3 *Estrategia de recubrimiento secuencial para Clasificación Ordinal*

En esta sección se presenta el algoritmo correspondiente a la estrategia de recubrimiento secuencial propuesta para problemas de clasificación ordinal (algoritmo 3). Este algoritmo presenta un cambio en la estructura del mismo con el objetivo de su mejora y optimización, obteniendo un conjunto de reglas más simple en el menor tiempo posible. En dicho algoritmo encontramos

una serie de funciones las cuales son implementadas para la adaptación del mismo a la clasificación ordinal teniendo presente las características y peculiaridades de estos problemas.

Algoritmo 3 Estrategia de Recubrimiento Secuencial de NSLVOrd.

RECUBRIMIENTO_SECUENCIAL_ORDINAL (E)

1. RemovedRules ← true;
2. LearnedRules ← SET_DEFAULT_RULE(E);
3. **While** (RemovedRules) **Do**
   a) Rule:= LEARN_ONE_ORD_RULE(E);
   b) **While** (PERFORMANCE_ORD(Rule,E)) **Do**
      i. LearnedRules ← LearnedRules + Rule;
      ii. E ← PENALIZE(LearnedRules,E);
      iii. Rule ← LEARN_ONE_ORD_RULE(E);
   c) RemovedRules ← FILTER_RULES(LearnedRules,E);
4. **Return** LearnedRules

La entrada del algoritmo es el conjunto de ejemplos E siendo la salida el conjunto de reglas LearnedRules que tratan de describir el comportamiento de E. Inicialmente la variable RemovedRules es puesta a true (paso 1) para permitir la entrada en el primer bucle *while* (paso 3) que controlará la salida del algoritmo cuando no sea posible obtener un conjunto de reglas más simple. Después de inicializar esta variable de control, se inicializa el conjunto de reglas con una regla por defecto (paso 2). A continuación nos adentramos en el primer bucle *while* (paso 3) en el que lleva a cabo el aprendizaje de forma similar al que se realiza en NSLV mediante el algoritmo 2 mostrado en 3.2. Al igual que en NSLV, en cada iteración se obtiene una nueva regla aprendida (paso 3.a y 3.b.iii) que se añade al conjunto de reglas aprendidas (paso 3.b.i) penalizando los ejemplos cubiertos para permitir el aprendizaje de nuevas reglas (paso 3.b.ii). Este proceso iterativo se repite mientras se mejore el comportamiento del sistema (paso 3.b). Aunque la secuencia de pasos de esta parte del algoritmo es similar al algoritmo de NSLV, la diferencia estriba en la implementación de los mismos ya que se hace uso de la información ordinal de

este tipo de problemas. Finalmente, una vez que se ha obtenido el conjunto de reglas, se lleva a cabo un filtrado con el objetivo de una simplificación de este conjunto de reglas aprendido (paso 3.c). Finalmente, una vez que se ha obtenido el conjunto de reglas más ajustado y simple posible que describa el comportamiento del sistema, este conjunto de reglas (LearnedRules) es devuelto en el paso 4.

Hay cuatro diferencias fundamentales entre esta propuesta de recubrimiento secuencial y la original de NSLV presentada en 3.2.

1. Se añade una regla por defecto al comienzo del algoritmo (paso 2) cuyo valor consecuente es la clase mayoritaria. Esta regla por defecto cubre los ejemplos de la clase mayoritaria. Al introducir esta regla por defecto se produce una optimización en tiempo y número de reglas ya que de otra forma se debe realizar el aprendizaje añadiendo todas la reglas necesarias hasta que se cubran estos ejemplos.

2. Se modifica la función Learn_One_Ord_Rule que es la encargada de obtener la mejor regla para hacer uso de las propiedades de los problemas ordinales (paso 3.a y 3.b.iii). Además lleva a cabo las mejoras introducidas respecto las definiciones de nuevos conceptos, la representación del individuo que independiza el aprendizaje de la clase a aprender y la generalización de la regla. En esta función se implementa los conceptos y modificaciones presentadas en las secciones 4.2.2 y 4.2.2.

3. Para el cálculo de la mejora obtenida con la nueva regla se utiliza la función Performance_Ord, la cual, además de los conceptos de consistencia y completitud presentados en 3.3, tiene en cuenta el error en la clase erróneamente clasificada al mismo tiempo que evita el sobreaprendizaje según se ha expuesto en la sección 4.2.2.

4. Se lleva a cabo un filtrado de reglas antes de terminar el algoritmo (paso 3.c). De esta forma se eliminan reglas obteniendo unos resultados más simples y sin pérdida en la descripción del comportamiento del sistema.

4.3 PRUEBAS Y RESULTADOS

En esta sección mostramos las pruebas realizadas para comprobar el buen funcionamiento del algoritmo propuesto, así como los resultados obtenidos. Para el análisis de los resultados se consideran algunas de las métricas presentadas en 2.2. En concreto, desde el punto de vista de clasificación se usa la métrica MZE, la cual puede ser definida mediante el CCR como $\mathrm{MZE} = 1 - \mathrm{CCR}$. Desde el punto de vista de clasificación ordinal, se usa la métrica MAE la cual calcula la dispersión del error global. Además, usamos la métrica ordinal MMAE ya que calcula la cantidad de desviación a partir de la etiqueta verdadera, o lo que es lo mismo, el error de la clase peor clasificada, lo cual es muy importante en clasificación ordinal.

Para realizar estas pruebas se han usado los conjuntos de datos[2] mostrados en la tabla 4.3. Cada conjunto de datos está formado por una partición de 30 conjuntos de ejemplos de entrenamiento y 30 conjuntos de ejemplos de test asociados, los cuales se obtienen con una técnica de "holdout stratified" usando el 75 % de los patrones para entrenamiento y el 25 restante para test. Esta estructura facilita la repetición de las pruebas.

4.3.1 *Análisis de NSLV y NSLVOrd*

En esta primera prueba se presenta el comportamiento del algoritmo ordinal propuesto respecto a la versión usada de base. Para NSLV se ha usado la versión disponible en el software Keel [3, 6]. La implementación de NSLVOrd es fácilmente integrable en Keel, admite el formato Weka [71] y está disponible para descarga[3]. El propósito de esta prueba es chequear la mejora de esta modificación de NSLV para abordar problemas de clasificación ordinal.

Como se ha expuesto, la métrica MMAE indica el error producido en la clase peor clasificada, la cual es muy importante en problemas de clasificación ordinal. La tabla 4.4 presenta los resultados de NSLV y NSLVOrd con esta métrica utilizando los 16 conjuntos de datos. Cada valor representa la media de los 30 conjuntos de test de cada conjunto de datos. En esta tabla se pue-

2 http://www.uco.es/grupos/ayrna/ucobigfiles/datasets-orreview.zip
3 http://isg.ugr.es/descargas/

Nombre	Atrib (Num/Nom)	Ejemplos (part/entren/test)	C
automobile	25 (15/1)	4590 (205/153/52)	6
balance-scale	4 (4/0)	14040 (625/468/157)	3
bondrate	10 (4/6)	1260 (57/42/15)	5
contact-lenses	5 (0/4)	540 (24/18/6)	3
ERA	4 (4/0)	22500 (1000/750/250)	9
ESL	4 (4/0)	10980 (488/366/122)	9
eucalyptus	19 (14/5)	16560 (736/552/184)	5
LEV	4 (4/0)	22500 (1000/750/250)	5
newthyroid	5 (5/0)	4830 (215/161/54)	3
pasture	22 (21/1)	810 (36/27/9)	3
squash-stored	24 (21/3)	1170 (52/39/13)	3
squash-unstored	24 (21/3)	1170 (52/39/13)	3
SWD	10 (10/0)	22500 (1000/750/250)	4
tae	5 (1/4)	3390 (151/113/38)	3
toy	2 (2/0)	6750 (300/225/75)	5
winequality-red	11 (11/0)	35970 (1599/1199/400)	6

Tabla 4.3: Conjunto de datos usados para las pruebas donde *Atrib (Num/Nom)* es el número de atributos totales, numéricos y nominales, *Ejemplos (part,entren,test)* es el número total de ejemplos, número de ejemplos en cada partición, número de ejemplos de entrenamiento de cada partición y número de ejemplos de test de cada partición, y *C* es el número de clases

de observar un mejor comportamiento de NSLVOrd respecto a NSLV. Para realizar un análisis más profundo se hace uso del test de Wilcoxon [132] ya que es un test no paramétrico muy usado para comparar dos algoritmos. En este test se usa un parámetro α para determinar si hay diferencias significativas entre los algoritmos siendo los valores más comunes $\alpha = 0.05$ y $\alpha = 0.10$. Este test establece una hipótesis nula de igualdad de medias, la cual es rechazada cuando el $p - value$ calculado es menor o igual al $\alpha - value$ considerado. Si la hipótesis es rechazada significa que los algoritmos comparados son significativamente diferentes.

Los resultados de aplicar el test de Wilcoxon con un $\alpha - value = 0.05$ se muestran en la tabla 4.5 donde se observa que teniendo en cuenta la métrica MMAE el algoritmo NSLVOrd es mejor que NSLV aunque no de manera significativa.

Con el objetivo de un análisis más detallado, se utilizan las métricas MZE y MAE cuyos resultados se muestran en las tablas

MMAE Test	NSLV	NSLVOrd
automobile	1,007 (2)	**0,783 (1)**
balance-scale	**0,974 (1)**	0,987 (2)
bondrate	**1,800 (1)**	2,150 (2)
contact-lenses	**0,872 (1)**	0,908 (2)
ERA	2,914 (2)	**2,134 (1)**
ESL	2,852 (2)	**1,618 (1)**
eucalyptus	0,826 (2)	**0,818 (1)**
LEV	**1,075 (1)**	1,192 (2)
newthyroid	0,326 (2)	**0,157 (1)**
pasture	0,611 (2)	**0,556 (1)**
squash-stored	0,843 (2)	**0,813 (1)**
squash-unstored	**0,750 (1)**	0,906 (2)
SWD	**0,922 (1)**	1,008 (2)
tae	0,781 (2)	**0,740 (1)**
toy	0,614 (2)	**0,438 (1)**
winequality-red	**2,243 (1)**	2,418 (2)
MEDIA	1,213 (2)	**1,102 (1)**
RNK	1,562 (2)	**1,438 (1)**

Tabla 4.4: Valores MMAE para test - NSLV vs. NSLVOrd

	VS	$p_{Wilcoxon}$
MMAE Test	NSLV	$\geqslant 0.2$

Tabla 4.5: Resultados obtenidos por el test de Wilcoxon usando la métrica MMAE para la comparativa del algoritmo NSLVOrd - NSLV vs. NSLVOrd

4.6 y 4.7 donde se observa que NSLVOrd continúa siendo mejor que NSLV.

Al igual que antes, para realizar un análisis más profundo, se realiza el test de Wilcoxon sobre estos resultados cuya salida se muestra en la tabla 4.8 donde se observa que con estas métricas NSLVOrd es significativamente mejor que NSLV

MZE Test	NSLV	NSLVOrd
automobile	0,304 (2)	**0,253 (1)**
balance-scale	**0,234 (1)**	0,240 (2)
bondrate	0,520 (2)	**0,516 (1)**
contact-lenses	0,250 (2)	**0,244 (1)**
ERA	0,981 (2)	**0,740 (1)**
ESL	0,476 (2)	**0,375 (1)**
eucalyptus	**0,446 (1)**	0,455 (2)
LEV	0,548 (2)	**0,377 (1)**
newthyroid	0,074 (2)	**0,047 (1)**
pasture	0,322 (2)	**0,255 (1)**
squash-stored	0,469 (2)	**0,398 (1)**
squash-unstored	**0,270 (1)**	0,339 (2)
SWD	0,579 (2)	**0,432 (1)**
tae	0,472 (2)	**0,434 (1)**
toy	0,271 (2)	**0,147 (1)**
winequality-red	0,458 (2)	**0,388 (1)**
MEDIA	0,417 (2)	**0,353 (1)**
RNK	1,812 (2)	**1,188 (1)**

Tabla 4.6: Valores MZE para test - NSLV vs. NSLVOrd

4.3.2 *Análisis con la versión ordinal de algoritmos conocidos*

En este apartado se compara nuestra propuesta con la versión ordinal de algoritmos bien conocidos cuya implementación está disponible. En concreto se comparan 5 algoritmos utilizando los conjuntos de datos mostrados en la tabla 4.3. Los tres primeros algoritmos son versiones adaptadas del algoritmo SVM para problemas de clasificación ordinal que están disponibles en la LIBSVMLibrary [20]: SVC, C-RNK y SVOR-IM. El cuarto algoritmo es una implementación ordinal del algoritmo C4.5 disponible en Weka [71] llamado ASAOR. El último es NSLVOrd, la propuesta que aquí se presenta. Una breve descripción de estos algoritmos es:

- SVC: *Cost Support Vector Classification* [79]. Es muy usada en clasificación nominal. Trata los problemas multi-clase como problemas uno contra uno. Si consideramos k clases,

MAE Test	NSLV	NSLVOrd
automobile	0.433 (2)	**0.362 (1)**
balance-scale	**0.345 (1)**	0.359 (2)
bondrate	**0.629 (1)**	0.680 (2)
contact-lenses	**0.383 (1)**	0.400 (2)
ERA	1.807 (2)	**1.316 (1)**
ESL	0.658 (2)	**0.423 (1)**
eucalyptus	**0.545 (1)**	0.566 (2)
LEV	0.569 (2)	**0.412 (1)**
newthyroid	0.074 (2)	**0.048 (1)**
pasture	0.344 (2)	**0.263 (1)**
squash-stored	0.492 (2)	**0.421 (1)**
squash-unstored	**0.285 (1)**	0.354 (2)
SWD	0.595 (2)	**0.462 (1)**
tae	0.591 (2)	**0.555 (1)**
toy	0.279 (2)	**0.169 (1)**
winequality-red	0.516 (2)	**0.456 (1)**
MEAN	0.534 (2)	**0.453 (1)**
RNK	1.688 (2)	**1.312 (1)**

Tabla 4.7: Valores MAE para test - NSLV vs. NSLVOrd

	VS	$p_{Wilcoxon}$
MZE Test	NSLV	0.002686
MAE Test	NSLV	0.010986

Tabla 4.8: Resultados obtenidos por el test de Wilcoxon usando las métricas MZE y MAE para la comparativa del algoritmo NSLVOrd - NSLV vs. NSLVOrd

se crean $k(k-1)/2$ clasificadores para todas las posibles combinaciones entre clases.

- C-RNK: El núcleo usado en SVM-Rank [87]. SVM-Rank es una versión SVM usada para aprendizaje de ranking de forma eficiente.
- SVOR-IM: *Support Vector Ordinal Regression with Implicit Constraints* [23]. Es una adaptación de la descomposición tipo optimización mínima secuencial para SVM en el que intenta determinar los límites considerando todos los ejemplos de entrenamiento.

- ASAOR: *A Simple Approach to Ordinal Classification* [44]. Convierte un problema ordinal con k clases en $k-1$ problemas de 2 clases. Para la inferencia estima la probabilidad de k clases ordinales combinando los $k-1$ modelos de aprendizaje.

Las pruebas consisten en la ejecución de todos los algoritmos usando los parámetros por defecto definidos por los autores sobre los mismos conjuntos de datos presentados al comienzo de esta sección. El objetivo es conseguir un buen comportamiento en todas las clases, esta es la idea que subyace tras la métrica MMAE la cual calcula el error de la peor clase.

MMAE Test	SVC	C-RNK	SVORIM	ASAOR	NSLVOrd
automobile	3.000 (5)	2.967 (3.5)	2.967 (3.5)	1.088 (2)	**0.783 (1)**
balance-scale	1.000 (5)	0.290 (2)	**0.287 (1)**	0.859 (3)	0.987 (4)
bondrate	2.900 (4)	2.900 (4)	2.900 (4)	2.367 (2)	**2.150 (1)**
contact-lenses	2.000 (5)	1.333 (4)	1.167 (3)	**0.878 (1)**	0.908 (2)
ERA	2.158 (2)	2.282 (4)	2.290 (5)	2.160 (3)	**2.134 (1)**
ESL	1.473 (2)	2.333 (5)	2.122 (4)	**1.361 (1)**	1.618 (3)
eucalyptus	2.640 (5)	1.808 (3)	1.810 (4)	**0.674 (1)**	0.818 (2)
LEV	1.246 (4)	1.240 (2.5)	1.240 (2.5)	1.252 (5)	**1.192 (1)**
newthyroid	1.000 (4)	1.000 (4)	1.000 (4)	0.229 (2)	**0.157 (1)**
pasture	1.533 (4)	2.000 (5)	1.000 (3)	**0.500 (1)**	0.556 (2)
squash-stored	2.000 (4)	2.000 (4)	2.000 (4)	0.888 (2)	**0.813 (1)**
squash-unstored	1.333 (4)	2.000 (5)	1.000 (3)	**0.561 (1)**	0.906 (2)
SWD	1.083 (2)	1.088 (4)	1.088 (4)	1.088 (4)	**1.008 (1)**
tae	1.045 (4)	0.824 (2)	0.830 (3)	1.246 (5)	**0.740 (1)**
toy	1.744 (5)	0.747 (2)	0.753 (3)	0.878 (4)	**0.438 (1)**
winequality-red	2.463 (3)	2.554 (5)	2.546 (4)	**2.107 (1)**	2.418 (2)
MEAN	1.789 (5)	1.710 (4)	1.562 (3)	1.133 (2)	**1.102 (1)**
RNK	3.875 (5)	3.688 (4)	3.438 (3)	2.375 (2)	**1.625 (1)**

Tabla 4.9: Valores MMAE para test - Versión ordinal de los algoritmos bien conocidos.

Los resultados para los 16 conjuntos de datos y los 5 algoritmos se muestran en la tabla 4.9 para la métrica MMAE. Igual en el caso anterior, cada valor representa el valor medio calculado sobre los 30 conjuntos de test de cada conjunto de datos. En esta tabla se puede observar que NSLVOrd presenta el mejor comportamiento.

Para analizar los resultados en profundidad y así determinar si hay diferencias significativas entre los algoritmos es aconsejable utilizar el test de Shaffer [122] ya que requiere un análisis de más de 2 algoritmos. En este test también se utiliza un $\alpha - value$ siendo los más comunes $\alpha = 0,05$ y $\alpha = 0,10$. Al igual que en el test de Wilcoxon, se establece una hipótesis de igualdad de medias que debe ser verificada para poder afirmar que un algoritmo es significativamente diferente, es decir, la hipótesis es rechazada si el $p - value$ calculado es menor o igual que el $\alpha - value$ considerado. De esta forma, los algoritmos son significativamente diferentes si la hipótesis es rechazada.

i	hypothesis	$p_{Shaffer}$
1	SVC vs .NSLVOrd	0.00057
2	C-RNK vs .NSLVOrd	0.001348
3	SVORIM vs .NSLVOrd	0.007114
4	SVC vs .ASAOR	0.043742
5	C-RNK vs .ASAOR	0.113286
6	SVORIM vs .ASAOR	0.229387
7	ASAOR vs .NSLVOrd	0.71885
8	SVC vs .SVORIM	1.301544
9	C-RNK vs .SVORIM	1.309442
10	SVC vs .C-RNK	1.309442

Tabla 4.10: Resultados obtenidos por el test de Shaffer usando la métrica MMAE - Versión ordinal de los algoritmos conocidos.

VS	$p_{Wilcoxon}$
SVC	4.272E-4
C-RNK	7.63E-4
SVORIM	0.0016784
ASAOR	$\geqslant 0.2$

Tabla 4.11: Resultados obtenidos por el test de Wilcoxon usando la métrica MMAE para la comparativa con el algoritmo NSLVOrd - Versión ordinal de los algoritmos conocidos.

La tabla 4.10 muestra los resultados del test de Shaffer considerando la métrica MMAE. En las tablas 4.9 y 4.10 podemos observar que, considerando esta métrica, nuestra propuesta consigue un error más bajo para la clase peor ajustada, por lo que

podemos decir que nuestro algoritmo consigue el mejor resultado en el peor de los casos, aunque no hay diferencias significativas con el algoritmo ASAOR que consigue el segundo lugar. Otra forma de comparación es utilizar el test de Wilcoxon comparando uno contra uno el algoritmo que se presenta con el resto de algoritmos. Estos resultados se muestran en la tabla 4.11 en la que se corrobora que hay diferencias significativas con SVC, C-RNK y SVORIM pero no hay diferencias significativas con ASAOR.

MZE Test	SVC	C-RNK	SVORIM	ASAOR	NSLVOrd
automobile	0.678 (3)	0.735 (4.5)	0.735 (4.5)	0.304 (2)	**0.253 (1)**
balance-scale	0.105 (3)	**0.061 (1)**	0.062 (2)	0.243 (5)	0.240 (4)
bondrate	0.422 (2)	0.422 (2)	0.422 (2)	0.467 (4)	0.516 (5)
contact-lenses	0.366 (3)	0.383 (5)	0.372 (4)	0.250 (2)	**0.244 (1)**
ERA	0.739 (2)	0.757 (5)	0.757 (4)	**0.738 (1)**	0.740 (3)
ESL	**0.316 (1)**	0.337 (3)	0.335 (2)	0.360 (4)	0.375 (5)
eucalyptus	0.657 (3)	0.762 (5)	0.762 (4)	**0.361 (1)**	0.455 (2)
LEV	0.371 (3)	**0.371 (1.5)**	**0.371 (1.5)**	0.387 (5)	0.377 (4)
newthyroid	0.252 (3)	0.280 (4.5)	0.280 (4.5)	0.084 (2)	**0.047 (1)**
pasture	0.689 (5)	0.667 (3.5)	0.667 (3.5)	**0.248 (1)**	0.255 (2)
squash-stored	0.579 (4)	0.579 (4)	0.579 (4)	**0.398 (1.5)**	**0.398 (1.5)**
squash-unstored	0.485 (3)	0.538 (4.5)	0.538 (4.5)	**0.226 (1)**	0.339 (2)
SWD	0.430 (4)	0.428 (2.5)	0.428 (2.5)	**0.426 (1)**	0.432 (5)
tae	0.492 (3)	0.488 (2)	0.498 (4)	0.605 (5)	**0.434 (1)**
toy	0.666 (5)	0.435 (3)	0.441 (4)	0.306 (2)	**0.147 (1)**
winequality-red	0.425 (3)	0.438 (5)	0.437 (4)	0.397 (2)	**0.388 (1)**
MEAN	0.480 (3)	0.480 (4)	0.480 (5)	0.362 (2)	**0.353 (1)**
RNK	3.125 (3)	3.500 (5)	3.438 (4)	**2.469 (1)**	**2.469 (1)**

Tabla 4.12: Valores MZE para test - Versión ordinal de los algoritmos bien conocidos.

A continuación, las tablas 4.12 y 4.13 muestran los resultados para las métricas MZE y MAE respectivamente. En estas tablas se puede observar el buen comportamiento de NSLVOrd igualando en un primer puesto con la métrica MZE y obteniendo un segundo puesto con la métrica MAE aunque posteriormente se muestra que no hay diferencias significativas mediante los test de Shaffer y Wilcoxon. Además, se puede observar que el algoritmo SVC presenta el peor resultado tanto en MAE como en MMAE lo cual es explicable ya que este algoritmo no considera las características de los problemas de clasificación ordinal.

MAE Test	SVC	C-RNK	SVORIM	ASAOR	NSLVOrd
automobile	1.109 (5)	1.011 (3.5)	1.011 (3.5)	0.400 (2)	**0.362 (1)**
balance-scale	0.126 (3)	**0.066 (1)**	0.067 (2)	0.300 (4)	0.359 (5)
bondrate	0.624 (2)	0.624 (2)	0.624 (2)	0.624 (4)	0.680 (5)
contact-lenses	0.533 (5)	0.439 (4)	0.400 (2)	**0.367 (1)**	0.400 (3)
ERA	1.333 (5)	1.324 (4)	1.314 (2)	**1.222 (1)**	1.316 (3)
ESL	**0.337 (1)**	0.372 (3)	0.365 (2)	0.384 (4)	0.423 (5)
eucalyptus	1.200 (5)	1.082 (4)	1.082 (3)	**0.383 (1)**	0.566 (2)
LEV	**0.406 (1)**	0.407 (2.5)	0.407 (2.5)	0.424 (5)	0.412 (4)
newthyroid	0.252 (3)	0.280 (4.5)	0.280 (4.5)	0.084 (2)	**0.048 (1)**
pasture	0.867 (4)	1.000 (5)	0.667 (3)	**0.248 (1)**	0.263 (2)
squash-stored	0.733 (4)	0.733 (4)	0.733 (4)	0.444 (2)	**0.421 (1)**
squash-unstored	0.510 (3)	0.615 (5)	0.538 (4)	**0.239 (1)**	0.354 (2)
SWD	0.453 (4)	**0.442 (1.5)**	**0.442 (1.5)**	0.447 (3)	0.462 (5)
tae	0.706 (5)	**0.523 (1)**	0.526 (2)	0.686 (4)	0.555 (3)
toy	0.884 (5)	0.435 (3)	0.441 (4)	0.356 (2)	**0.169 (1)**
winequality-red	0.486 (3)	0.489 (5)	0.488 (4)	**0.441 (1)**	0.456 (2)
MEAN	0.660 (5)	0.615 (4)	0.587 (3)	**0.441 (1)**	0.453 (2)
RNK	3.625 (5)	3.312 (4)	2.875 (3)	**2.375 (1)**	2.812 (2)

Tabla 4.13: Valores MAE para test - Versión ordinal de los algoritmos bien conocidos.

i	hypothesis	$p_{Shaffer}$
1	C-RNK vs ASAOR	0.65073
2	C-RNK vs NSLVOrd	0.65073
3	SVORIM vs ASAOR	0.65073
4	SVORIM vs NSLVOrd	0.65073
5	SVC vs ASAOR	1.442525
6	SVC vs NSLVOrd	1.442525
7	SVC vs C-RNK	2.00934
8	SVC vs SVORIM	2.00934
9	C-RNK vs SVORIM	2.00934
10	ASAOR vs NSLVOrd	2.00934

Tabla 4.14: Resultados obtenidos por el test de Shaffer usando la métrica MZE - Versión ordinal de los algoritmos bien conocidos.

Al igual que anteriormente, para realizar un análisis sobre las diferencias significativas, se lleva a cabo el test de Shaffer y Wilcoxon con estos resultados. Las tablas 4.14 y 4.15 muestran los resultados con el test de Shaffer sobre las métricas MZE y

i	hypothesis	$p_{Shaffer}$
1	SVC vs ASAOR	0.253473
2	C-RNK vs ASAOR	0.561195
3	SVC vs NSLVOrd	0.876603
4	SVC vs SVORIM	1.078275
5	C-RNK vs NSLVOrd	2.22656
6	SVORIM vs ASAOR	2.22656
7	C-RNK vs SVORIM	2.22656
8	ASAOR vs NSLVOrd	2.22656
9	SVC vs C-RNK	2.22656
10	SVORIM vs NSLVOrd	2.22656

Tabla 4.15: Resultados obtenidos por el test de Shaffer usando la métrica MAE - Versión ordinal de los algoritmos bien conocidos.

VS	$p_{Wilcoxon}$
SVC	0.03864
C-RNK	0.01825
SVORIM	0.01825
ASAOR	$\geqslant 0.2$

Tabla 4.16: Resultados obtenidos por el test de Wilcoxon usando la métrica MZE para la comparativa con el algoritmo NSLVOrd - Versión ordinal de los algoritmos bien conocidos.

VS	$p_{Wilcoxon}$
SVC	0.02496
C-RNK	0.09344
SVORIM	0.19282
ASAOR	$\geqslant 0.2$

Tabla 4.17: Resultados obtenidos por el test de Wilcoxon usando la métrica MAE para la comparativa con el algoritmo NSLVOrd - Versión ordinal de los algoritmos bien conocidos.

MAE respectivamente. Igualmente las tablas 4.16 y 4.17 muestran los resultados del test de Wilcoxon sobre las métricas MZE y MAE respectivamente. Se puede observar que no hay diferencias significativas al realizar estos análisis.

Para concluir, en términos de ajustabilidad global, el algoritmo propuesto proporciona resultados que son comparables a los

obtenidos con otros métodos bien conocidos tanto desde el punto de vista de clasificación (MZE) como desde la perspectiva de la regresión (MAE). Además, la propuesta aquí presentada mejora los resultados de la métrica ordinal usada cuando el error medido como distancia entre la clase real y la clase predicha considera el peor caso (MMAE).

4.4 CONCLUSIONES

Este capítulo presenta una propuesta para abordar la clasificación ordinal. El algoritmo propuesto se llama NSLVOrd, el cual es una extensión del algoritmo NSLV.

En este capítulo se ha presentado las modificaciones necesarias a realizar sobre los distintos componentes de NSLV para abordar la clasificación ordinal. En primer lugar, en lo que respecta a AG se ha propuesto la redefinición de conceptos usados en el algoritmo base NSLV al mismo tiempo que se ha propuesto una modificación de la representación del individuo así como varias mejoras que permiten una optimización en el aprendizaje. Por último, respecto a la estrategia de recubrimiento secuencial, se ha propuesto una modificación de la misma con la que se consigue una optimización así como la consideración de los aspectos clave de la clasificación ordinal.

Para mostrar el buen comportamiento de nuestra propuesta para este tipo de problemas, se ha llevado a cabo un análisis comparativo con el algoritmo de base NSLV, así como con otros algoritmos de aprendizaje bien conocidos. Finalmente podemos observar que nuestra propuesta presenta un buen comportamiento para todas las métricas utilizadas e incluso mejora el ajuste de la clase peor clasificada.

5

EXTENSIÓN DEL MODELO DE CLASIFICACIÓN ORDINAL A REGRESIÓN (NSLVREG)

A veces, cuando innovas, cometes errores.
Es mejor admitirlos rápidamente
y seguir mejorando tus otras innovaciones.
Steven Paul Jobs [1]

ÍNDICE

1 Steven Paul Jobs (1955 - 2011) fue un famoso empresario e informático estadounidense, presidente de Apple Inc. y máximo accionista individual de The Walt Disney Company. Era una de las más importantes figuras de la industria de la computación y del entretenimiento digital.

5.1 INTRODUCCIÓN

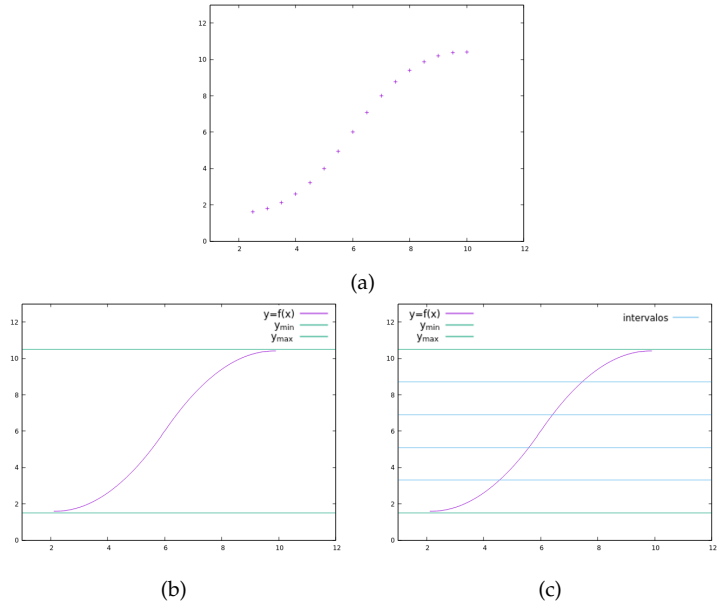
Los *Problemas de Regresión* son aquellos problemas en los que el valor de resultado es un dato numérico. Los problemas de regresión [35] y los problemas de clasificación ordinal [92, 99] comparten algunas características. Ambos aprenden a partir de un conjunto de ejemplos cuya variable de salida tiene un cierto orden. En el caso de clasificación ordinal la salida está compuesta por un conjunto de etiquetas de salida, mientras que para el caso de regresión la salida puede tomar cualquier valor perteneciente al conjunto de los números reales dentro de un intervalo. Además, mientras que en clasificación ordinal la distancia entre los valores de las etiquetas es desconocido, en regresión la distancia entre cada par de valores es conocido. Debido a estas similaridades, en este capítulo se presenta una aproximación para resolver problemas de regresión mediante una extensión del algoritmo NSLVOrd presentado en el capítulo 4, que llamaremos NSLV Regresión (NSLVReg).

Con el objetivo de aclarar la idea subyacente en esta propuesta vamos a utilizar el siguiente ejemplo. Supongamos que se desea obtener un conjunto de reglas para aproximar el conjunto de datos mostrado en la tabla 5.1, cuya representación gráfica se muestra en la figura 5.1(a).

x	y	x	y	x	y
2,5	1,63	5	4,00	7,5	8,78
3	1,80	5,5	4,93	8	9,40
3,5	2,13	6	6,00	8,5	9,88
4	2,60	6,5	7,08	9	10,20
4,5	3,23	7	8,00	9,5	10,38
10	10,40				

Tabla 5.1: Datos de ejemplo para regresión

Una posible aproximación de los datos del ejemplo podría realizarse mediante la función que se muestra en la figura 5.1(b). Sobre esta gráfica, y con la idea de utilizar el algoritmo de clasificación ordinal para resolver este problema de regresión, se podría realizar una primera aproximación a groso modo discretizando la variable "y" usando "k" particiones. De esta forma se divide la variable de salida en intervalos de tamaño $\frac{y_{max}-y_{min}}{k}$



(a)

(b)

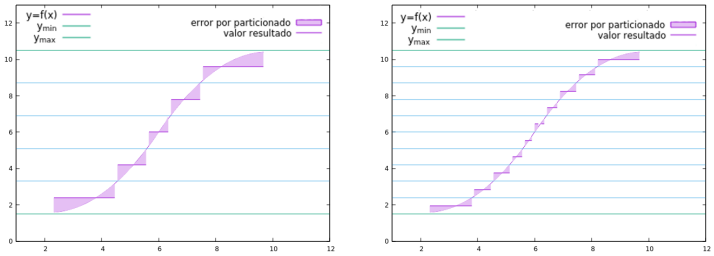
(c)

Figura 5.1: Representación de los datos de ejemplo para regresión

que denotaremos como "interv", quedando la variable particionada de la forma $[y_{min}, y_{min} + interv), [y_{min} + interv, y_{min} + 2 * interv), \ldots, [y_{min} + (k-1) * interv, y_{max}]$ como muestra la figura 5.1(c) para el caso de realizar la partición en 5 intervalos. Considerando cada intervalo como una clase, se puede plantear como un problema de clasificación ordinal.

Supongamos que tras aplicar un algoritmo de clasificación ordinal, como por ejemplo NSLVOrd, obtenemos un valor de error de cero en la métrica CCR. Además, supongamos que para dar el valor de salida correspondiente al problema de regresión se devuelve el valor correspondiente al punto medio del intervalo, de forma que en cada punto se comete un error que está acotado por $\frac{y_{max} - y_{min}}{2 * k}$. Un ejemplo de aproximación con 5 intervalos se muestra en la figura 5.2(a) donde se observa el error cometido por la consideración de los 5 intervalos.

Para conseguir una mejor aproximación se puede pensar en aumentar el número de particiones de la variable de salida ya que aumentando este número, y considerando que el error de la clasificación sigue siendo cero, se comete un menor error al discretizar como se muestra en la figura 5.2(b).



(a) (b)

Figura 5.2: Aproximación a regresión con discretización de salida en 5 y 10 intervalos

El aumento considerable del número de particiones puede acarrear un problema ya que se aumenta la complejidad del aprendizaje tanto en tiempo como en tamaño, incrementándose, al mismo tiempo, el número de reglas necesarias para clasificar cada una de las clases correspondientes a las particiones definidas en la variable de salida, con lo que el tamaño de la base de reglas podría aumentar considerablemente.

Aún considerando un error de cero en la clasificación, el valor real devuelto se corresponde con el valor medio del intervalo considerado, pudiendo éste coincidir o no con el valor real del ejemplo. Para ilustrar esta situación vamos a considerar los valores $e1 = (6, 6)$ y $e2 = (6.5, 7.08)$ de los datos del ejemplo con un particionamiento en 5 intervalos. La figura 5.3 se presenta el particionamiento en 5 intervalos en el que se muestra el valor medio el cual es el valor devuelto como resultado. Para el caso de $e1$ el valor y coincide con el valor medio de una de las clases por lo que el valor devuelto es correcto. En el caso de $e2$ el valor y no coincide con ninguna clase devolviendo el valor $y = 7.8$ ya que corresponde a la clase más cercana cometiendo un error de 0.72 como se muestra en la figura 5.4.

Con el objetivo de mejorar la aproximación sin incluir demasiada complejidad al problema se plantea realizar una discretización similar a la anterior pero con un pequeño desplazamiento en la definición de las particiones de la variable de salida. Como valor de salida se puede devolver, por ejemplo, el valor medio de los valores resultado de estos intervalos. En la figura 5.5 se presenta un ejemplo de particionamiento en 5 intervalos y dos desplazamientos sobre dichos intervalos. En esta figura se muestra los valores

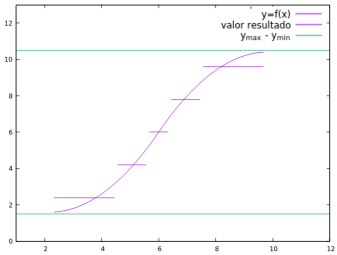


Figura 5.3: Valores medios devueltos como resultado en 5 intervalos sin desplazamiento

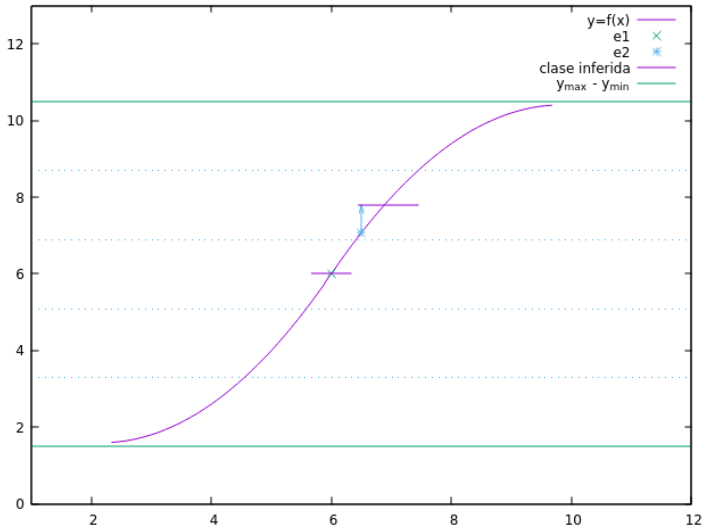


Figura 5.4: Ejemplos de valores devueltos como resultado en 5 intervalos sin desplazamiento

medios devueltos como resultado en los intervalos creados. Considerando los mismos datos de ejemplo que en el caso anterior ($e1 = (6, 6)$ y $e2 = (6.5, 7.08)$), para el caso de $e1$ el resultado es el mismo que en el caso en que no se consideren desplazamientos ya que se toma la media de los valores devueltos, siendo el resultado el mismo ya que los valores de los desplazamientos es el mismo en ambos sentidos. Para el caso $e2$, si consideramos las clases desplazadas los valores de las clases más cercanas son 7.8 para el central, 7,2 para el desplazamiento a la izquierda y 6,6 para

el desplazamiento a la derecha ya que la inferencia devuelve la clase anterior como se muestra en la figura 5.6 . El valor que se devuelve es el valor medio de estos tres, devolviendo un valor de 7.2 obteniendo un error de 0.12 con lo que se obtiene una disminución considerable del error.

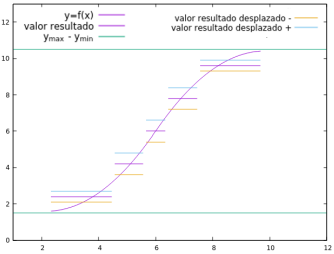


Figura 5.5: Valores medios devueltos como resultado en 5 intervalos con desplazamiento

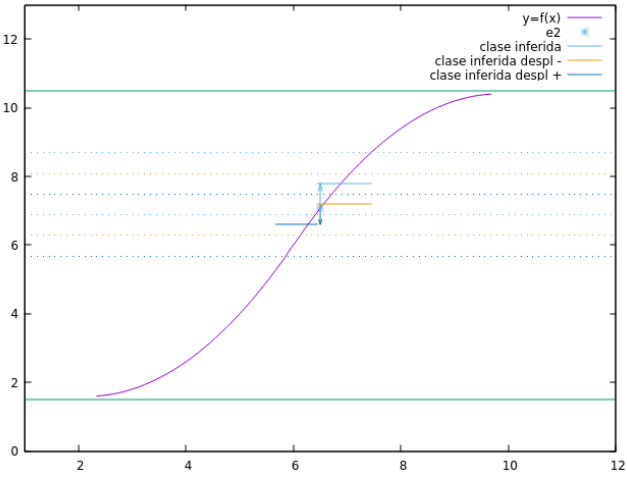


Figura 5.6: Valores medios devueltos como resultado en 5 intervalos con desplazamiento

Estas son las principales ideas que forman la base de la propuesta que aquí se presenta para abordar problemas de regresión a partir del algoritmo de clasificación ordinal NSLVOrd. En las siguientes secciones se expone de forma más concreta la implementación de estas ideas. Finalmente se expone la comparativa

realizada con otros algoritmos de regresión así como los resultados de la misma.

5.2 PROPUESTA

En la sección anterior se han presentado las ideas que subyacen en este capítulo para resolver problemas de regresión utilizando el algoritmo de clasificación ordinal NSLVOrd. Para ello se proponen dos técnicas, por una parte se establece una discretización homogénea de la variable de salida obteniendo intervalos que serán convertidos a clases ordinales y así poder abordar el problema como un problema de clasificación ordinal. Por otra parte se fijan dos desplazamientos en los intervalos resultado de la discretización como se ha expuesto en la sección anterior. Resultado de esta división en intervalos y de los dos desplazamientos se pueden plantear tres problemas de clasificación ordinal muy similares entre sí, característica que puede ser utilizada en el aprendizaje.

Una vez obtenidos los problemas ordinales se puede realizar el aprendizaje de dichos problemas utilizando, por ejemplo, el algoritmo NSLVOrd. En este punto es donde se plantean una serie de alternativas a estudiar.

- En primer lugar hay que evaluar tanto el número de particiones necesarias como el valor del desplazamiento en las mismas para el problema en consideración.

  Respecto a la creación de las particiones y consecuentemente de los distintos problemas ordinales, después de diversas pruebas, cuyos resultados se muestran en el apéndice A, se llega a la conclusión de que el aumento del número de particiones puede mejorar el valor de regresión obtenido en la salida. Como se adelantó en la sección anterior, este aumento no puede ser muy elevado ya que se aumenta la complejidad del aprendizaje, llegando incluso a obtener peores resultados que con un número menor de particiones.

  En lo que respecta al desplazamiento, con él se pretende que haya un solapamiento en los intervalos considerados, por lo que el rango considerado está entre un 0 y un 50 porciento de la anchura del intervalo. Este desplazamiento se introduce para mejorar los resultados como se expuso en la sección anterior. Además, se considera que el despla-

zamiento es el mismo en ambos sentidos para así facilitar el tratamiento del mismo. Resultados de los distintos valores de este desplazamiento se muestran en el apéndice B en el que se corrobora la idea original que al introducir desplazamientos se mejoran los resultados.

- En segundo lugar hay que realizar un estudio de las diferentes formas de combinar los resultados obtenidos de cada clasificador para devolver finalmente el valor real correspondiente al problema de regresión.

  Respecto a la devolución del valor final como resultado del problema de regresión se pueden plantear diferentes alternativas. Si consideramos los clasificadores como cajas negras de los que obtenemos un valor de salida hay diversas formas de combinar estos valores como por ejemplo con la media. Por otra parte, si consideramos los clasificadores como módulos dentro de un sistema más complejo y en cuyo interior se puede adentrar, surgen otras opciones a considerar que van desde el valor de la regla disparada en cada definición del problema, hasta la consideración de todas las bases de reglas junto con sus características para realizar la combinación de todas ellas. Una breve muestra de las alternativas consideradas junto a los resultados obtenidos se muestra en el apéndice C en el que se concluye que la mejor opción es la consideración como cajas negras utilizando la media para el cálculo del valor de regresión.

- En tercer lugar hay que estudiar cómo realizar la ejecución de los distintos clasificadores ordinales que llevarán a cabo el aprendizaje sobre cada definición del problema.

  Considerando la idea planteada hasta el momento hay que ejecutar al menos tres clasificadores sobre problemas diferentes pero muy similares al mismo tiempo. En una primera aproximación, considerando cada clasificador como un problema independiente, se puede plantear su ejecución tanto de forma secuencial como paralela. Por otra parte se puede plantear la consideración de las similitudes existentes entre las definiciones de estos problemas con lo que se pueden plantear la ejecución de algunos clasificadores reutilizando el aprendizaje realizado en otros. El apéndice D muestra alguna de las alternativas consideradas junto a los resultados obtenidos en los que se concluye que la mejor opción es

la reutilización del aprendizaje de unos clasificadores para otros.

Como consecuencia de los distintos experimentos realizados con las diferentes alternativas comentadas anteriormente, se toma como mejor opción la creación de un sistema modular en el que se integra una modificación del algoritmo de aprendizaje NSLVOrd. Este algoritmo de aprendizaje basado en NSLVOrd que llamaremos 'NSLVOrd_mod' se modifica para posibilitar un aprendizaje incremental con distintas particiones homogéneas de la variable de salida. Otra modificación permite el aprendizaje de cada 'NSLVOrd_mod' partiendo de aprendizaje de otros algoritmos 'NSLVOrd_mod'. Finalmente, el sistema modular ejecuta los aprendizajes con los algoritmos 'NSLVOrd_mod' de forma que se obtenga un aprendizaje general al mismo tiempo que realiza la combinación necesaria para devolver el valor real correspondiente al problema de regresión.

Una novedad de la propuesta que aquí se presenta es la posibilidad de dar como resultado tanto el valor real como un intervalo de error asociado a ese valor. Este intervalo hace referencia al posible error cometido debido a la utilización de un algoritmo de clasificación ordinal para la resolución del problema de regresión.

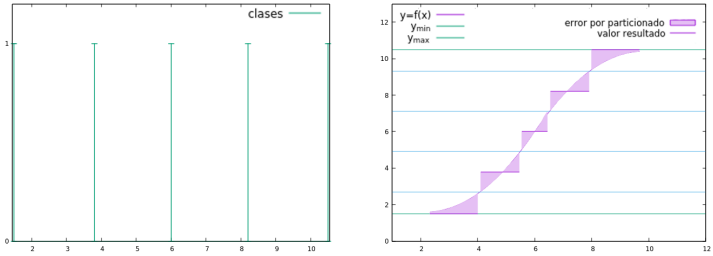
El sistema propuesto junto con las modificaciones necesarias se presentan en las secciones siguientes.

5.2.1 *Discretización de la variable de salida*

Inicialmente se realiza una discretización de la variable de salida en 5 intervalos. Se comienza con este número de intervalos puesto que, a pesar de ser un número de intervalos relativamente pequeño, es lo suficientemente representativo como para una primera aproximación. Además el hecho de comenzar con pocos intervalos facilita el primer aprendizaje, lo cual puede condicionar los procesos de aprendizaje sucesivos.

Como comentamos anteriormente, el algoritmo de clasificación ordinal a utilizar como base es NSLVOrd el cual tiene la variable de salida definida como un conjunto de clases como se muestra en la figura 5.7(a) para el caso de 5 clases. Para poder realizar la equivalencia entre particiones y clases de la variable de salida se modifican las particiones con lo que la aproximación sería como se muestra en la figura 5.7(b) en las que las clases de los extremos

coinciden con los valores mínimo y máximo de la variable de salida, realizando la partición con esta consideración.



(a) (b)

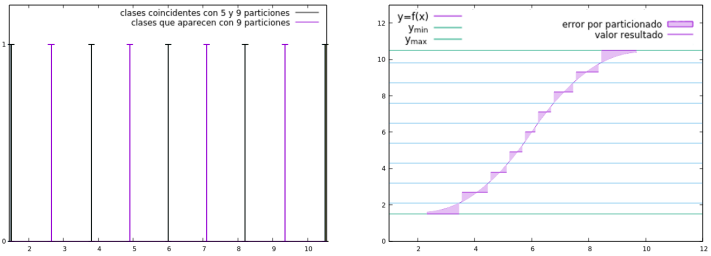
Figura 5.7: Variable de salida de NSLVOrd correspondiente a 5 clases

Para obtener una mejor aproximación se plantea un aprendizaje en diferentes niveles como se expone en las secciones siguientes. En cada nivel se aumenta el número de intervalos en los que se particiona la variable de salida. Esta nueva partición se produce cada vez que se aumenta de nivel con lo que se genera una nueva definición del problema ordinal correspondiente. Con el objetivo de poder reutilizar el aprendizaje del nivel anterior pero con un nivel de granularidad mayor, se crean las particiones de forma que al realizar la conversión a clases del problema ordinal se obtiene una clase entre cada par de clases del nivel inferior. Un ejemplo de las clases correspondientes al primer y segundo nivel se muestra la figura 5.8(a) con lo que se obtiene la partición de segundo nivel con 9 intervalos obteniendo una aproximación como la mostrada en la figura 5.8(b). El número de intervalos de cada nivel es calculado como resultado de la ecuación 5.1.

$$numInterv_i = 2^{i+2} + 1 \,. \tag{5.1}$$

donde i inicialmente es "0" e indica el nivel de aprendizaje. El "1" se le añade para obtener un número impar de intervalos y así simplificar la equivalencia entre intervalos y clases de la variable de salida del problema ordinal correspondiente.

Respecto al valor de salida, hay que tener en cuenta que hay un error cometido que debe ser asumido. Este error es debido al particionado de la variable de salida. Aún considerando una clasificación correcta, hay un intervalo de error acotado en función



(a) (b)

Figura 5.8: Variable de salida de NSLVOrd correspondiente a 9 clases

del número de intervalos. Así con un número de intervalos igual a 5 correspondiente al nivel 0 tendríamos un error acotado del 25 % de la variable de salida. En el siguiente nivel, nivel 1, se particiona la variable de salida en 9 intervalos con un error del 12,5 %. El siguiente, nivel 2, tiene 17 intervalos con 6,25 % de error y así continuamos con un nivel 3 con 33 intervalos y un error de 3,125 %; un nivel 4 con 65 intervalos y 1,562 % de error, y así sucesivamente. Con 6 niveles, se tiene 129 intervalos con un error de 0,781 %, es decir, un error de menos de 1 % pero a costa de tener que resolver un problema ordinal con 129 clases lo que lo convierte en un problema realmente exigente en tiempo y con grandes posibilidades de error en la clasificación debido a la cercanía de las clases. Para disminuir este error que debe ser asumido evitando los inconvenientes del aumento del número de intervalos, se aplica el desplazamiento en los intervalos expuesto en la sección anterior (fig. 5.5).

5.2.2 *Esquema de aprendizaje para regresión*

En esta sección exponemos el algoritmo que lleva a cabo el aprendizaje para problemas de regresion, el cual está formado por un conjunto de clasificadores ordinales y un método para combinarlos. En la literatura podemos encontrar diferentes aproximaciones de multiclasificadores [59]: unos usan diferentes esquemas de combinación de los clasificadores [45, 46]; otros utilizan diferentes esquemas de combinación de los resultados de los clasificadores [15]. Tanto unos como otros son denominados ensemble de forma genérica. La propuesta que aquí se presenta es la que me-

jor resultados ha obtenido al considerarla como un ensemble de clasificadores en el que se establece un esquema de ejecución de clasificadores junto con un esquema de combinación de resultados. Los resultados de las distintas alternativas se muestran en los apéndices correspondientes.

El algoritmo que se propone realiza el aprendizaje en diferentes niveles en los que se va mejorando la aproximación a los resultados deseados mediante la discretización de la variable de salida en diferente número de intervalos. Igualmente para mejorar esta aproximación se realizan tres procesos de aprendizaje en cada nivel, uno por el problema obtenido como resultado del particionamiento de la variable de salida, y otros dos generados por el problema obtenido como resultado de los desplazamientos sobre el particionamiento de la variable de salida de ese nivel, que denotaremos como desplazamiento a izquierda y desplazamiento a derecha. El aprendizaje de cada uno de estos problemas se realiza mediante la ejecución de un conjunto de clasificadores ordinales definidos sobre diferentes definiciones del problema ordinal correspondiente al nivel y desplazamiento con el que estamos trabajando. Debido a la similitud de las definiciones de los diferentes problemas, así como a la posible reutilización de parte del aprendizaje entre estos clasificadores se observa que la mejor alternativa es la compartición del aprendizaje adquirido como se muestra en los apéndices correspondientes y se expone en las secciones siguientes.

Estrategia de recubrimiento secuencial para regresión

El aprendizaje de cada uno de los problemas ordinales se lleva a cabo mediante una estrategia de recubrimiento secuencial similar a la presentada en el algoritmo 3 de la sección 4.2.3 del capítulo 4 con las diferencias principales que se exponen a continuación.

En el algoritmo original (algoritmo 3) no es necesario el establecimiento de la definición del problema puesto que es intrínseco al mismo. En el caso que nos ocupa, esta definición del problema está determinada por el nivel y el desplazamiento de la fase de aprendizaje en cada momento. Igualmente se considera la posible existencia de una base de reglas inicial, por lo que se hace necesario estos parámetros como entrada del algoritmo. La definición del problema concreto se establece en cada momento en función

de los parámetros de entrada del algoritmo como se indica a continuación.

El algoritmo 4 es el correspondiente a la estrategia de recubrimiento secuencial que lleva a cabo el aprendizaje de reglas en el caso que nos ocupa. La entrada del algoritmo es un conjunto de ejemplos (E), el número de intervalos (nInterv) en función del nivel en que nos encontremos y, si es necesario, la base de reglas (ruleBase) previamente aprendida, el sentido del desplazamiento (direction) y el valor del desplazamiento (shift). El algoritmo devuelve como resultado el conjunto de reglas aprendido (LearnedRules) que forma la base de reglas para el problema ordinal que estamos tratando.

Algoritmo 4 Estrategia de recubrimiento sec. de NSLVOrd_mod

LEARN_RULE_BASE (E, nInterv,ruleBase,direction,shift)

1. RemovedRules ← true;
2. problem ← SET_PROBLEM_DEFINITION(E, nInterv, ruleBase, direction, shift);
3. LearnedRules ← SET_DEFAULT_RULE(E, problem);
4. **While** (RemovedRules) **Do**
   a) Rule ← LEARN_ONE_ORD_RULE(E, problem);
   b) **While** (PERFORMANCE_ORD(Rule, E, problem)) **Do**
      i. LearnedRules ← LearnedRules + Rule;
      ii. E ← PENALIZE(LearnedRules, E, problem);
      iii. Rule ← LEARN_ONE_ORD_RULE(E, problem);
   c) RemovedRules ← FILTER_RULES(LearnedRules, E, problem);
5. **Return** LearnedRules

Inicialmente la variable RemovedRules es puesta a true (paso 1) para permitir la entrada en el primer bucle *while* (paso 4) que controla la salida del algoritmo cuando no sea posible obtener un conjunto de reglas más simple. Después de inicializar esta variable de control, se lleva a cabo la definición del problema. Para ello, por una parte se utiliza el número de intervalos y, si lo hay, el sentido y

valor del desplazamiento. Por otra parte, si hay una base de reglas inicial, se utiliza junto con el conjunto de ejemplos para establecer la definición del problema actual en función de estos parámetros (paso 2). A continuación se inicializa el conjunto de reglas con una regla por defecto (paso 3). Lo siguiente es la comprobación del primer bucle *while* (paso 4) en el que lleva a cabo el aprendizaje de forma similar al que se realiza en NSLVOrd mediante el algoritmo 3 mostrado en 4.2.3. Al igual que en NSLVOrd, en cada iteración se obtiene una nueva regla aprendida (paso 4.a y 4.b.iii) que se añade al conjunto de reglas aprendidas (paso 4.b.i) penalizando los ejemplos cubiertos para permitir el aprendizaje de nuevas reglas (paso 4.b.ii). Este proceso iterativo se repite mientras se mejore el comportamiento del sistema (paso 4.b). Esta secuencia de pasos es similar a la que realiza NSLVOrd con la diferencia de que es necesario la definición del problema para realizar los mismos. Finalmente, una vez que se ha obtenido el conjunto de reglas, se lleva a cabo un filtrado con el objetivo de una simplificación de este conjunto de reglas aprendido (paso 4.c). Una vez que se ha obtenido el conjunto de reglas más ajustado y simple posible que describa el comportamiento del sistema, este conjunto de reglas (*LearnedRules*) es devuelto en el paso 5.

Una diferencia fundamental respecto a NSLVOrd original es la introducción de los elementos necesarios para la consideración del nivel y desplazamiento en el que estamos, junto con la consideración de una posible base de reglas inicial. Todos estos elementos establecen la nueva definición del problema ordinal concreto. Por una parte, la información referente al número de intervalos junto con el sentido del desplazamiento y el valor del mismo nos define las clases de salida del problema ordinal. Por otra parte debemos considerar la información referente a la base de reglas y los ejemplos de entrenamiento. En el caso de que no haya base de reglas inicial no hay que realizar más consideraciones. En caso contrario hay que realizar una comprobación y adaptación de las reglas iniciales a la nueva definición del problema considerando los ejemplos cubiertos por estas reglas como se indica a continuación.

Al igual que NSLVOrd, para la búsqueda de la mejor regla se utiliza un AG como el expuesto en 3.3.1 con las modificaciones presentadas en 4.2.2. En el AG que se utiliza, cada individuo de la población representa una regla. El aprendizaje se produce de forma iterativa mediante la evolución de la población. La regla

que se obtiene es la correspondiente al mejor individuo después de la evolución de la población. Para considerar la base de reglas inicial, después de probar varias alternativas, algunas de las cuales se muestran en el apéndice D, se realiza una transformación de esta base de reglas a individuos de la población del AG. Una vez realizada esta inicialización de la población se puede entender que se parte de la población que mejor resultado presenta para el problema anterior ya resuelto, el cual, a su vez, es muy similar al que estamos tratando. De esta manera se crea una situación ventajosa para el aprendizaje del nuevo problema en el que muchos de estos individuos se convertirán en reglas con pequeñas modificaciones que permitan su adaptación al problema en cuestión. Este proceso se realiza en la función SET_PROBLEM_DEFINITION del paso 2. A continuación se realiza el aprendizaje se forma similar a como se realiza en NSLVOrd con la diferencia de que partimos de una población muy adaptada al problema en cuestión, lo que optimiza el proceso de aprendizaje.

Otra diferencia es el uso de la definición del problema en cada una de las funciones que intervienen en el algoritmo. Este uso es necesario ya que el problema concreto depende del nivel y desplazamiento concreto.

Algoritmo de aprendizaje para regresión

En la figura 5.9 se puede observar un diagrama que muestra cómo se lleva a cabo el aprendizaje para el problema de regresión completo. Hay que notar que se utiliza el algoritmo 4 expuesto anteriormente y que se denota como 'NSLVOrd_mod' en dicho diagrama.

En el diagrama se pone de manifiesto la necesidad de diferentes definiciones del problema a lo largo del aprendizaje. Estas definiciones son las derivadas de los distintos intervalos en que es particionada la variable de salida en cada nivel, así como a los desplazamientos de los mismos.

El proceso de aprendizaje es un proceso incremental en el que el sistema aprende un conjunto de tres bases de reglas en cada nivel en el que se encuentra. Al comienzo se realiza una discretización en 5 intervalos como los mostrados en la figura 5.7(b). Esta discretización nos condiciona las clases correspondientes a la definición del problema ordinal correspondiente (figura 5.7 (a)), que junto con el conjunto de ejemplos y un conjunto de re-

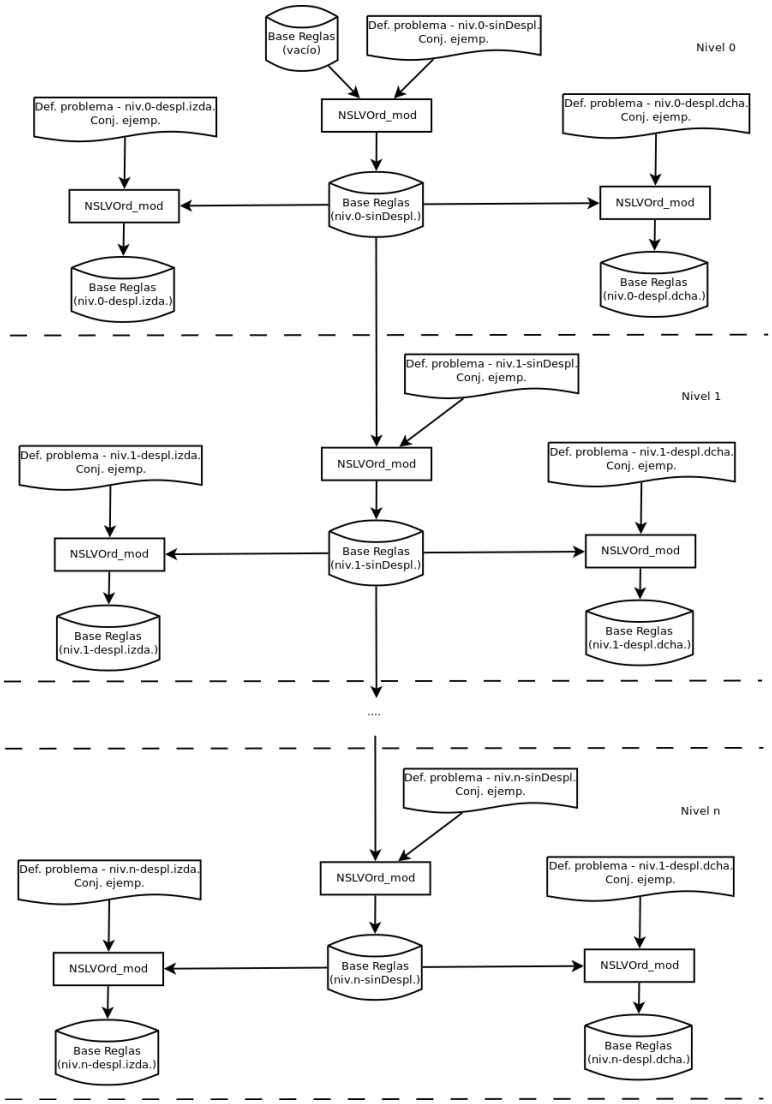


Figura 5.9: Diagrama del algoritmo de aprendizaje para regresión

glas inicialmente vacío son las entradas del clasificador ordinal NSLVOrd_mod que dará como resultado el conjunto de reglas correspondiente a la parte central del primer nivel con la discretización sin desplazamiento en los intervalos. En este mismo nivel se lleva a cabo los desplazamientos 'a izquierda' y 'a derecha'

de los intervalos posicionados en los extremos del esquema de funcionamiento, dando como resultado otras dos definiciones del problema ordinal. Cada una de estas definiciones junto con el conjunto de reglas previamente aprendido componen las entradas de los clasificadores ordinales NSLVOrd_mod correspondientes a los diferentes procesos de aprendizaje de las bases de reglas con desplazamiento a izquierda y derecha respectivamente del nivel en el que nos encontramos, es decir, el primer nivel. En el aprendizaje de cada clasificador ordinal correspondiente a los extremos, se lleva a cabo una adaptación de las reglas de entrada, resultado de la clasificación central, para adaptarlas a la nueva definición de los problemas de los extremos en los cuales se considera el desplazamiento. Una vez realizada esta adaptación se lleva a cabo el aprendizaje de nuevas reglas que serán añadidas a las adaptadas de forma que se mejore la aproximación.

En este punto se ha realizado el aprendizaje de las bases de reglas de primer nivel correspondiente a la discretización en 5 intervalos. Como comentamos anteriormente el aprendizaje es un proceso incremental en el que en cada nivel se realiza un incremento del número de intervalos con el objetivo de obtener una mejor aproximación. Para los niveles posteriores se procede de forma similar a la forma de proceder del primer nivel realizándose tres procesos de aprendizaje y obteniendo tres bases de reglas. En estos niveles y derivado de la discretización aplicada se produce igualmente tres nuevas definiciones del problema ordinal en cada nivel. La principal diferencia se produce en el aprendizaje del conjunto de reglas de la discretización sin desplazamiento (parte central) para el cual no se parte de un conjunto de reglas vacío, sino que se parte del conjunto de reglas de la discretización sin desplazamiento (parte central) del nivel anterior. Para ello, igual que se realiza en los casos en los que se produce el desplazamiento, se lleva a cabo una adaptación del conjunto de reglas de entrada a la nueva definición del problema ordinal y, posteriormente, se lleva a cabo el aprendizaje de las nuevas reglas para la definición del problema en cuestión. El aprendizaje correspondiente a las discretizaciones con desplazamiento de cada nivel (parte de los extremos del esquema) se realiza de forma similar a como se realiza en el primer nivel. Este mismo procedimiento se realiza en cada nuevo nivel a considerar.

Un aumento excesivo del número de particiones puede ser contraproducente en lo que a mejora de precisión y tiempo empleado

en el aprendizaje se refiere. Por ello se hace necesario establecer un mecanismo para detener la consideración de nuevos niveles. En este caso se establecen dos modos de parada, mediante una definición del número de niveles a la hora de comenzar el aprendizaje, o mediante un criterio que define si hay una mejora en la respuesta del sistema deteniendo la consideración de nuevos niveles cuando no se produzca esta mejora. Si está definido en número de niveles como parámetros del aprendizaje se utiliza este criterio de parada. Si por el contrario este valor no está definido se utiliza el criterio de mejora en la consideración de nuevos niveles.

Algoritmo 5 Algoritmo de aprendizaje para regresión

REGRESSION_LEARNING_ALGORITHM (E, numLevels, shift)

1. RuleSet $\leftarrow \emptyset$;
2. $i \leftarrow 0$;
3. $nInterv \leftarrow 2^{(i+2)} + 1$;
4. $Rules_0^c \leftarrow$ LEARN_RULE_BASE (E,nInterv, NULL, NULL, 0);
5. $Rules_0^l \leftarrow$ LEARN_RULE_BASE (E,nInterv, $Rules_0^c$, left, shift);
6. $Rules_0^r \leftarrow$ LEARN_RULE_BASE (E,nInterv, $Rules_0^c$, right, shift);
7. RuleSet $\leftarrow \begin{pmatrix} Rules_0^c & Rules_0^l & Rules_0^r \end{pmatrix}$;
8. i ++;
9. **While** (stopCondition(numLevels, E, RuleSet)) **Do**
   a) $nInterv \leftarrow 2^{(i+2)} + 1$;
   b) $Rules_i^c \leftarrow$ LEARN_RULE_BASE (E,nInterv, $Rules_{i-1}^c$, NULL, 0);
   c) $Rules_i^l \leftarrow$ LEARN_RULE_BASE (E,nInterv, $Rules_i^c$, left, shift);
   d) $Rules_i^r \leftarrow$ LEARN_RULE_BASE (E,nInterv, $Rules_i^c$, right, shift);
   e) RuleSet $\leftarrow \begin{pmatrix} Rules_0^c & Rules_0^l & Rules_0^r \\ \vdots & \vdots & \vdots \\ Rules_i^c & Rules_i^l & Rules_i^r \end{pmatrix}$;
   f) i ++;
10. **Return** {RuleSet};

Una vez presentado a groso modo el funcionamiento del algoritmo de aprendizaje para regresión (figura 5.9). El algoritmo 5

muestra con más detalle los pasos a realizar para el aprendizaje. La entrada del algoritmo es un conjunto de ejemplos (E), el número de niveles a considerar (numLevels) y el desplazamiento de los intervalos (shift). El algoritmo devuelve una matriz de bases de reglas formado por tantas filas como niveles considerados y 3 columnas correspondientes a los tres aprendizajes que se realizan en cada nivel.

Al comienzo del algoritmo se lleva a cabo la inicialización del conjunto de bases de reglas RuleSet a vacío (paso 1) así como de las variables i y nInterv (paso 2 y 3) a los valores correspondientes (0 y 5 respectivamente) para poder comenzar con el aprendizaje. En primer lugar se lleva a cabo un proceso de aprendizaje con 5 clases o intervalos y sin desplazamiento en las mismas consiguiendo la primera base de reglas ($Rules_0^c$) (paso 4). Una vez obtenido esta base de reglas se lleva a cabo el desplazamiento de las intervalos a izquierda y derecha, y comienza el aprendizaje con la nueva representación del problema haciendo uso de la base de reglas aprendidas para el caso central, obteniendo las bases de reglas correspondientes ($Rules_0^l$ y $Rules_0^r$) (pasos 5 y 6). Estas bases de reglas se añaden al conjunto de bases de reglas RuleSet (paso 7).

Una vez finalizado el aprendizaje de primer nivel se actualiza la variable i (paso 8) y se lleva a cabo el aprendizaje del resto de niveles dentro del bucle *while* hasta que se cumpla la condición de parada (stopCondition) (paso 9). Dentro del bucle *while*, en primer lugar, se calcula el nuevo número de clases de la variable de salida correspondiente a los intervalos obtenidos de la discretización del nivel que estemos considerando (paso 9.a). A continuación se lleva a cabo el aprendizaje de las tres bases de reglas de la siguiente manera. En primer lugar (paso 9.b) se realiza la adaptación de las reglas correspondientes al aprendizaje de los intervalos no desplazados de la iteración anterior, es decir, se utiliza la base de reglas sin desplazamiento del nivel anterior como base para obtener la nueva base de reglas. Aquí se lleva a cabo tanto la adaptación de las reglas aprendidas del nivel anterior como el aprendizaje de nuevas reglas. En segundo lugar (paso 9.c y 9.d) se lleva a cabo el aprendizaje de las nuevas reglas utilizando como base las aprendidas en el caso sin desplazamiento similar a como se ha realizado en el primer nivel. Posteriormente se actualiza el conjunto de base de reglas con las nuevas bases de

reglas aprendidas (paso 9.e). Finalmente se actualiza la variable i (paso 9.f).

Al final del algoritmo (paso 10) se devuelve la matriz de bases de reglas.

La condición de parada del algoritmo está implementada en la función stopCondition. Si está definido el número de niveles, la condición de parada se produce cuando se llega a ese número de niveles. Si por el contrario no está definido este número de niveles, la salida del bucle *while* se produce cuando no se consigue mejora en el aprendizaje sobre el conjunto de ejemplos de entrenamiento. Esta mejora se mide mediante la comparación de los valores de RMSE del nivel anterior y el actual.

Combinación de resultados de los clasificadores ordinales

En este capítulo se plantea abordar un problema de regresión mediante un conjunto de clasificadores ordinales. La solución del problema de regresión debe ser un valor real, mientras que la solución del problema ordinal es una clase. A lo largo del capítulo se ha planteado la transformación del problema de regresión a un conjunto de problemas de clasificación ordinal. Cada uno de estos problemas ordinales es resuelto por un clasificador distinto. En esta sección se presenta la combinación de los resultados de los distintos clasificadores ordinales para la consecución de un valor real del problema de regresión.

Para la elección de la mejor combinación de resultados de los clasificadores originales se han realizado diversas pruebas con gran variedad de alternativas presentadas en el apéndice C. La alternativa con mejores resultados es aquella que toma los clasificadores como cajas negras devolviendo como valor real el punto medio del intervalo correspondiente a la clase de cada definición del problema. El valor real que se obtiene como resultado del problema de regresión es la media aritmética de los valores resultado de todos los clasificadores ordinales.

Como se ha expuesto anteriormente, una novedad de la propuesta que aquí se presenta es la posibilidad de obtener un intervalo de error asociado al valor real del problema de regresión. Este rango se obtiene de las diferentes definiciones del problema ordinal. Al realizar una discretización para obtener el problema ordinal correspondiente se genera un rango en el que debe estar el valor real de salida. Este rango es el que se utiliza para el cálculo

del intervalo de error. Para este cálculo también se pueden plantear distintas alternativas. En la solución presentada, el intervalo de error que se obtiene es el resultado de las medias de todos los intervalos obtenidos de cada clasificador ordinal, de manera similar a como se obtiene el valor real de resultado.

5.3 RESULTADOS

Tanto NSLV como NSLVOrd son algoritmos de aprendizaje de reglas difusas con resultados muy competitivos tanto en problemas de clasificación nominal [57] como en problemas de clasificación ordinal [53]. En esta sección presentamos los resultados de la propuesta presentada que puede ser entendida como una extensión de estos algoritmos a problemas de regresión. Para el análisis de los resultados se hace uso de la métrica Raíz del Error cuadrático medio (Root Mean Square Error en Inglés) (RMSE) ya que es la más usual para problemas de regresión. Con el objetivo de poder comparar de forma más cómoda los resultados de distintos problemas con diferentes rangos en la variable de salida, se hace uso de la métrica RMSE^N propuesta en 2.3.2.

Para realizar estas pruebas se ha usado el conjunto de datos mostrado en la tabla 5.2 que está disponible en Keel[2] [3, 6]. Este conjunto de datos se encuentra particionado mediante un procedimiento de validación cruzada de 5. Para estas pruebas se ha utilizado estas particiones ya que son muy usadas en la literatura y pueden ser comparadas fácilmente. Además al disponer de las particiones, la ejecución de las pruebas e incluso otras comparaciones posteriores son posibles de una manera sencilla.

La implementación de NSLVReg está disponible en la web del grupo de investigación[3]. Esta implementación está preparada para ser integrada en el software Keel además de permitir su ejecución de forma independiente. También permite usar el formato de datos de entrada de Weka [71].

Las secciones siguientes muestran los resultados de las pruebas desarrolladas. La sección 5.3.1 muestra los resultados de la comparación de nuestra propuesta con los algoritmos de la categoría a la cual pertenece. En la sección 5.3.2 se muestra un análisis comparativo utilizando un extenso y variado conjunto de algorit-

2 http://www.keel.es/
3 http://isg.ugr.es/descargas

Name	Att (Real/Int)	Examp (train/test)
autompg6	5 (2/3)	392 (314/78)
autompg8	7 (2/5)	392 (314/78)
dee	6 (6/0)	365 (182/183)
diabetes	2 (2/0)	43 (34/9)
ele-2	4 (4/0)	1056 (845/211)
friedman	5 (5/0)	1200 (600/600)
laser	4 (4/0)	993 (794/199)
machineCPU	6 (0/6)	209 (104/105)

Tabla 5.2: Conjuntos de datos usados en las pruebas donde *Att (Real/Int)* es el número de atributos total, atributos reales y atributos enteros, y *Examp (train,test)* es el número de ejemplos total, el número de ejemplos de entrenamiento y test de cada partición

mos disponibles en Keel para regresión. Por último se muestra el comportamiento del intervalo de error que proporciona nuestro algoritmo junto al valor de regresión.

5.3.1 *Análisis comparativo con los algoritmos de su categoría disponibles en Keel*

La propuesta que se presenta en este capítulo puede ser encuadrada dentro de la categoría de algoritmos de *Aprendizaje de Reglas Difusas Evolutivas* para regresión. En esta sección se compara la propuesta presentada con los 8 algoritmos de esta categoría disponibles en Keel, utilizando los 8 conjuntos de datos mostrados en la tabla 5.2.

El experimento consiste en ejecutar los algoritmos de esta categoría, incluyendo nuestra propuesta, usando los parámetros por defecto definidos por los autores sobre el mismo conjunto de datos mostrados en la tabla 5.2 al principio de esta sección. En el caso de nuestra propuesta se ha definido un número de niveles igual a 4, un 25 % de desplazamiento, la media como método de combinación de resultados intra y entre niveles y una generación horizontal de individuos del AG.

La tabla 5.3 muestra los resultados con la métrica RMSE. En ella se observa que nuestra propuesta obtiene el primer lugar. Cada valor de esta tabla representa el valor medio calculado sobre las 5 particiones de test de cada conjunto de datos. Esta comparativa

RMSE Test	TSK-IRL-R	MOGUL-IRLSC-R	GFS-GPG-R	MOGUL-IRLHC-R	. . .
autoMPG6	4,734 (9)	3,202 (3)	3,584 (7)	3,215 (4)	. . .
autoMPG8	3,022 (3)	3,328 (5)	3,724 (7)	3,020 (2)	. . .
dee	4,353 (9) *	0,495 (6)	0,493 (5)	0,466 (3)	. . .
diabetes	1,516 (9)	0,779 (5)	**0,557 (1)**	0,898 (7)	. . .
ele-2	213,315 (4)	305,539 (5)	366,634 (7)	309,385 (6)	. . .
friedman	1,645 (2)	2,399 (4)	3,714 (8)	2,480 (6)	. . .
laser	**7,594 (1)**	11,377 (4)	22,577 (9)	12,641 (5)	. . .
machineCPU	79,326 (7)	**62,847 (1)**	101,626 (8)	73,509 (5)	. . .
RNK	5,500 (6)	4,125 (3)	6,500 (8)	4,750 (5)	. . .

RMSE Test	MOGUL-TSK-R	GFS-SP-R	Thrift-R	GFS-RB-MF-R	NSLVReg
autoMPG6	3,480 (5)	2,993 (2)	3,486 (6)	3,844 (8)	**2,938 (1)**
autoMPG8	3,638 (6)	3,047 (4)	4,016 (8)	4,171 (9)	**2,752 (1)**
dee	1,080 (8)	0,486 (4)	0,461 (2)	0,536 (7)	**0,431 (1)**
diabetes	1,220 (8)	0,732 (4)	0,794 (6)	0,700 (3)	0,608 (2)
ele-2	**91,049 (1)**	196,331 (3)	386,505 (8)	455,410 (9)	185,091 (2)
friedman	**1,466 (1)**	2,961 (7)	2,471 (5)	3,747 (9)	1,976 (3)
laser	7,951 (2)	9,839 (3)	22,400 (8)	22,083 (7)	15,099 (6)
machineCPU	70,039 (3)	72,556 (4)	77,430 (6)	313,634 (9)	69,102 (2)
RNK	4,250 (4)	3,875 (2)	6,125 (7)	7,625 (9)	**2,250 (1)**

Tabla 5.3: RMSE Media Test - Aprendizaje de Reglas Difusas Evolutivas - "*" indica que ha dado como resultado algunos valores fuera del rango de la variable de salida - comparativa de NSLVReg con su categoría.

está formada por un conjunto de más de dos algoritmos, por lo que para realizar un análisis en profundidad de los resultados se utiliza el test de Shaffer [122] con un α valor de $\alpha = 0.05$ o $\alpha = 0.10$. En este test se debe verificar la hipótesis nula de igualdad de medias para verificar si un algoritmo es significativamente mejor que otro. En la tabla 5.4 se presentan los resultados del test de Shaffer donde se puede observar que sólo hay diferencias significativas entre NSLVReg y el algoritmo que ocupa la última posición en ranking.

Para llevar a cabo un análisis más profundo se usa el test de Wilcoxon [132] con el objetivo de comparar nuestra propuesta con las otras de forma individual. La tabla 5.5 muestra los resultados del test de Wilcoxon. En esta tabla podemos observar que NSLVReg es el primero en ranking pero en este caso hay diferencias significativas con los algoritmos que ocupan las tres últimas posiciones en ranking.

El rango de los resultados de la métrica RMSE depende del intervalo de la variable de salida del problema, por lo que depende

i	hypothesis	$p_{Shaffer}$
1	GFS-RB-MF-R vs .NSLV-Reg	0.003118
2	GFS-GPG-R vs .NSLV-Reg	0.053502
3	Thrift-R vs .NSLV-Reg	0.130375
4	GFS-SP-R vs .GFS-RB-MF-R	0.172757
5	MOGUL-IRLSC-R vs .GFS-RB-MF-R	0.29644
6	MOGUL-TSK-R vs .GFS-RB-MF-R	0.383903
7	TSK-IRL-R vs .NSLV-Reg	0.493419
8	MOGUL-IRLHC-R vs .GFS-RB-MF-R	1.001385
9	GFS-GPG-R vs .GFS-SP-R	1.546559
10	MOGUL-IRLHC-R vs .NSLV-Reg	1.546559
11	MOGUL-IRLSC-R vs .GFS-GPG-R	1.822423
12	GFS-GPG-R vs .MOGUL-TSK-R	2.207661
13	GFS-SP-R vs .Thrift-R	2.207661
14	TSK-IRL-R vs .GFS-RB-MF-R	2.655198
15	MOGUL-IRLSC-R vs .Thrift-R	3.170795
16	MOGUL-TSK-R vs .NSLV-Reg	3.170795
17	MOGUL-IRLSC-R vs .NSLV-Reg	3.170795
18	MOGUL-TSK-R vs .Thrift-R	3.170795
19	GFS-GPG-R vs .MOGUL-IRLHC-R	3.622367
20	TSK-IRL-R vs .GFS-SP-R	3.765332
21	GFS-SP-R vs .NSLV-Reg	3.765332
22	Thrift-R vs .GFS-RB-MF-R	4.099825
23	TSK-IRL-R vs .MOGUL-IRLSC-R	4.099825
24	MOGUL-IRLHC-R vs .Thrift-R	4.099825
25	TSK-IRL-R vs .MOGUL-TSK-R	4.335725
26	GFS-GPG-R vs .GFS-RB-MF-R	4.524452
27	TSK-IRL-R vs .GFS-GPG-R	4.652088
28	MOGUL-IRLHC-R vs .GFS-SP-R	4.70535
29	TSK-IRL-R vs .MOGUL-IRLHC-R	4.70535
30	TSK-IRL-R vs .Thrift-R	4.70535
31	MOGUL-IRLSC-R vs .MOGUL-IRLHC-R	4.70535
32	MOGUL-IRLHC-R vs .MOGUL-TSK-R	4.70535
33	GFS-GPG-R vs .Thrift-R	4.70535
34	MOGUL-TSK-R vs .GFS-SP-R	4.70535
35	MOGUL-IRLSC-R vs .GFS-SP-R	4.70535
36	MOGUL-IRLSC-R vs .MOGUL-TSK-R	4.70535

Tabla 5.4: Resultados obtenidos por el test de Shaffer para RMSE - comparativa de NSLVReg con su categoría

el problema. Con el propósito de evitar esta dependencia del problema y poder realizar una comparación más cómoda de los resultados, se propuso la métrica $RMSE^N$ en 2.3.2. La tabla 5.6

VS	$p_{Wilcoxon}$
TSK-IRL-R	0.19532
MOGUL-IRLSC-R	$\geqslant 0.2$
GFS-GPG-R	0.015626
MOGUL-IRLHC-R	0.10938
MOGUL-TSK-R	$\geqslant 0.2$
GFS-SP-R	0.14844
Thrift-R	0.007812
GFS-RB-MF-R	0.007812

Tabla 5.5: Resultados obtenidos por el test de Wilcoxon para RMSE - comparativa de NSLVReg con su categoría.

muestra los resultados utilizando esta métrica en la que se puede comparar la media además del ranking. En esta tabla se observa que nuestra propuesta obtiene el primer lugar tanto en ranking como en media.

Al igual que con la métrica RMSE, para realizar un análisis más profundo se utilizan los test de Shaffer y Wilcoxon cuyos resultados se muestran en las tablas 5.7 y 5.8 respectivamente. En estas tablas se observa una pequeña variación en los valores aunque sigue mostrando diferencias significativas con el último algoritmo en caso del test de Shaffer (tabla 5.7) y con los tres últimos en el caso del test de Wilcoxon (tabla 5.8).

Por último, como resultado de estas comparativas, se puede concluir que NSLVReg presenta el mejor comportamiento de los algoritmos de su categoría, aunque no se puede afirmar que sea significativamente mejor que otros que le siguen.

5.3.2 *Análisis comparativo con los algoritmos de regresión disponibles en Keel*

Una vez comprobado el buen funcionamiento de nuestra propuesta en relación a los algoritmos de su categoría, vamos a estudiar el comportamiento del mismo respecto a una muestra representativa del resto de algoritmos de regresión disponibles en Keel.

Igual que en la sección anterior se ejecutan todos los algoritmos, incluyendo nuestra propuesta, usando los parámetros por defecto definidos por los autores sobre el mismo conjunto de datos (tabla 5.2). En el caso de nuestra propuesta se ha vuelto a utilizar un

$RMSE^N$ Test	TSK-IRL-R	MOGUL-IRLSC-R	GFS-GPG-R	MOGUL-IRLHC-R	. . .
autoMPG6	0,126 (9)	0,085 (3)	0,095 (7)	0,086 (4)	. . .
autoMPG8	0,080 (2,5)	0,089 (5)	0,099 (7)	0,080 (2,5)	. . .
dee	1,000 (9) *	0,114 (6)	0,113 (5)	0,107 (3)	. . .
diabetes	0,421 (9)	0,216 (5)	**0,155 (1)**	0,249 (7)	. . .
ele-2	0,025 (4)	0,036 (5,5)	0,043 (7)	0,036 (5,5)	. . .
friedman	0,059 (2)	0,086 (4)	0,133 (8)	0,089 (6)	. . .
laser	**0,030 (1)**	0,045 (4)	0,089 (9)	0,050 (5)	. . .
machineCPU	0,069 (7)	**0,055 (1)**	0,089 (8)	0,064 (5)	. . .
MEDIA	0,226 (9)	0,091 (3)	0,102 (5)	0,095 (4)	. . .
RNK	5,438 (6)	4,188 (3)	6,500 (8)	4,750 (5)	. . .

$RMSE^N$ Test	MOGUL-TSK-R	GFS-SP-R	Thrift-R	GFS-RB-MF-R	NSLVReg
autoMPG6	0,093 (5,5)	0,080 (2)	0,093 (5,5)	0,102 (8)	**0,078 (1)**
automPG8	0,097 (6)	0,081 (4)	0,107 (8)	0,111 (9)	**0,073 (1)**
dee	0,248 (8)	0,112 (4)	0,106 (2)	0,123 (7)	**0,099 (1)**
diabetes	0,339 (8)	0,203 (4)	0,221 (6)	0,194 (3)	0,169 (2)
ele-2	**0,011 (1)**	0,023 (3)	0,046 (8)	0,054 (9)	0,022 (2)
friedman	**0,052 (1)**	0,106 (7)	0,088 (5)	0,134 (9)	0,071 (3)
laser	0,031 (2)	0,039 (3)	0,088 (8)	0,087 (7)	0,059 (6)
machineCPU	0,061 (3)	0,063 (4)	0,068 (6)	0,274 (9)	0,060 (2)
MEDIA	0,117 (7)	0,088 (2)	0,102 (6)	0,135 (8)	**0,079 (1)**
RNK	4,312 (4)	3,875 (2)	6,062 (7)	7,625 (9)	**2,250 (1)**

Tabla 5.6: $RMSE^N$ Media Test - Aprendizaje de Reglas Difusas Evolutivas - "*" indica que ha dado como resultado algunos valores fuera del rango de la variable de salida - comparativa de NSLVReg con su categoría.

número de niveles igual a 4, un 25 % de desplazamiento, la media como método de combinación de resultados intra y entre niveles y una generación horizontal de individuos del AG.

i	hypothesis	$p_{Shaffer}$
1	GFS-RB-MF-R vs .NSLV-Reg	0.003118
2	GFS-GPG-R vs .NSLV-Reg	0.053502
3	Thrift-R vs .NSLV-Reg	0.150222
4	GFS-SP-R vs .GFS-RB-MF-R	0.172757
5	MOGUL-IRLSC-R vs .GFS-RB-MF-R	0.337669
6	MOGUL-TSK-R vs .GFS-RB-MF-R	0.435641
7	TSK-IRL-R vs .NSLV-Reg	0.557805
8	MOGUL-IRLHC-R vs .GFS-RB-MF-R	1.001385
9	GFS-GPG-R vs .GFS-SP-R	1.546559
10	MOGUL-IRLHC-R vs .NSLV-Reg	1.546559
11	MOGUL-IRLSC-R vs .GFS-GPG-R	2.007623
12	TSK-IRL-R vs .GFS-RB-MF-R	2.423276
13	GFS-GPG-R vs .MOGUL-TSK-R	2.423276
14	GFS-SP-R vs .Thrift-R	2.423276
15	MOGUL-TSK-R vs .NSLV-Reg	2.904139
16	MOGUL-IRLSC-R vs .NSLV-Reg	3.298747
17	MOGUL-IRLSC-R vs .Thrift-R	3.298747
18	GFS-GPG-R vs .MOGUL-IRLHC-R	3.622367
19	MOGUL-TSK-R vs .Thrift-R	3.622367
20	GFS-SP-R vs .NSLV-Reg	3.765332
21	TSK-IRL-R vs .GFS-SP-R	4.061328
22	Thrift-R vs .GFS-RB-MF-R	4.061328
23	MOGUL-IRLHC-R vs .Thrift-R	4.391444
24	TSK-IRL-R vs .MOGUL-IRLSC-R	4.697036
25	TSK-IRL-R vs .MOGUL-TSK-R	4.935766
26	GFS-GPG-R vs .GFS-RB-MF-R	4.935766
27	TSK-IRL-R vs .GFS-GPG-R	4.935766
28	MOGUL-IRLHC-R vs .GFS-SP-R	4.935766
29	TSK-IRL-R vs .MOGUL-IRLHC-R	4.935766
30	TSK-IRL-R vs .Thrift-R	4.935766
31	MOGUL-IRLSC-R vs .MOGUL-IRLHC-R	4.935766
32	GFS-GPG-R vs .Thrift-R	4.935766
33	MOGUL-IRLHC-R vs .MOGUL-TSK-R	4.935766
34	MOGUL-TSK-R vs .GFS-SP-R	4.935766
35	MOGUL-IRLSC-R vs .GFS-SP-R	4.935766
36	MOGUL-IRLSC-R vs .MOGUL-TSK-R	4.935766

Tabla 5.7: Resultados obtenidos por el test de Shaffer para $RMSE^N$ - comparativa de NSLVReg con su categoría

VS	$p_{Wilcoxon}$
TSK-IRL-R	$\geqslant 0.2$
MOGUL-IRLSC-R	0.0664
GFS-GPG-R	0.02344
MOGUL-IRLHC-R	0.07812
MOGUL-TSK-R	$\geqslant 0.2$
GFS-SP-R	0.10938
Thrift-R	0.007812
GFS-RB-MF-R	0.007812

Tabla 5.8: Resultados obtenidos por el test de Wilcoxon para $RMSE^N$ - comparativa de NSLVReg con su categoría

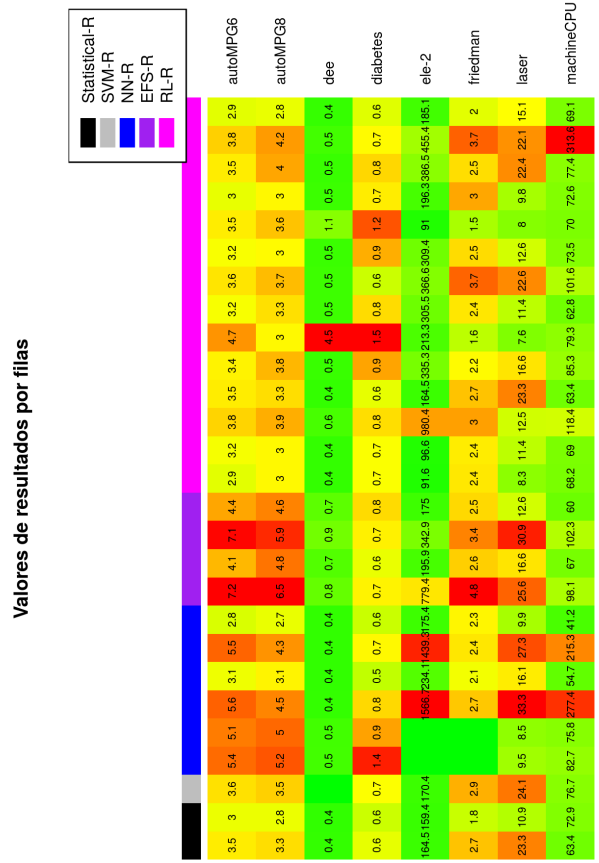


Figura 5.10: Mapa de colores con los resultados de los algoritmos de regresión de Keel

En este análisis comparativo intervienen una gran cantidad de algoritmos. Con el objetivo de simplificar el estudio de los mismos, la figura 5.10 muestra un mapa de colores donde de un vistazo se puede hacer una idea de los resultados. En este mapa los valores más claros indican buenos resultados, mientras que los más oscuros indican los peores. Como muestra el mapa de colores, la columna correspondiente a nuestra propuesta no tiene colores oscuros, por lo que de un vistazo se puede observar el buen comportamiento de la propuesta.

Debido a la cantidad de algoritmos y datos a mostrar se han particionado los resultados en tres tablas aunque los valores de ranking corresponden al conjunto de todos los algoritmos que intervienen en la comparativa. De esta forma, las tablas 5.9, 5.10 y 5.11 muestran los resultados experimentales con la métrica RMSE obtenida con 27 algoritmos y 8 conjuntos de datos. Igual que en la sección anterior, cada valor de la tabla anterior representa el valor medio calculado sobre las 5 particiones de test de cada conjunto de datos. En estas tablas se observa que, a pesar de que nuestra propuesta no es la primera en ranking, tiene una buena posición en el ranking situándose en el primer cuartil. Para llevar a cabo un análisis más profundo se pueden usar varios test no-paramétricos. El experimento completo requiere analizar más de 2 algoritmos, así es aconsejable usar el test de Shaffer [122] con un α valor de $\alpha = 0.05$ o $\alpha = 0.10$. En este test se debe verificar la hipótesis nula de igualdad de medias para verificar si un algoritmo es significativamente mejor que otro. Debido a la longitud de los resultados de este test por el gran número de algoritmos a comparar, la tabla 5.12 muestra un resumen de la misma donde los resultados de nuestra propuesta está presente. En esta tabla se puede observar que no hay diferencias significativas entre nuestra propuesta y los algoritmos mejor situados que ella.

Con el objetivo de comparar nuestra propuesta con las otras de forma individual y añadido a la longitud de los resultados del test de Shaffer se hace uso del test de Wilcoxon [132], el cual es muy popular para comparaciones en las que sólo intervienen dos algoritmos. En el test de Wilcoxon, al igual que en el de Shaffer, también se usa un parámetro α para decidir si hay diferencias significativas entre los algoritmos. Los valores más frecuentes son, de forma similar a Shaffer, $\alpha = 0.05$ o $\alpha = 0.10$. También establece una hipótesis nula de igualdad de medias, la cual es rechazada cuando el $p - value$ calculado en este test es mejor o igual al

RMSE Test	Regresión Estadística Regresión Estadística		SVM4R SVM4R	NN4R NN4R NN4R	. . . . . .
	LinearLMS-R	PolQuadraticLMS-R	NU_SVR-R	RBFN-R	
autoMPG6	3,483 (12,5)	3,047 (5)	3,636 (16)	5,402 (23)	. . .
autoMPG8	3,345 (11)	2,768 (3)	3,489 (13)	5,241 (25)	. . .
dee	0,410 (6)	0,406 (3)	4,353 (26,5) ***	0,545 (19)	. . .
diabetes	0,627 (6,5)	0,556 (3)	0,687 (11)	1,403 (26)	. . .
ele-2	164,521 (5,5)	159,356 (4)	170,431 (7)	8481,560 (26,5) ***	. . .
friedman	2,700 (17,5)	1,753 (3)	2,858 (19)	27,926 (26,5) ***	. . .
laser	23,267 (21,5)	10,868 (8)	24,070 (23)	9,475 (5)	. . .
machineCPU	63,380 (5,5)	72,931 (13)	76,678 (16) **	82,694 (19)	. . .
RNK	10,750 (7)	5,250 (2)	16,438 (17)	21,250 (25)	. . .

RMSE Test	NN4R NN4R				. . .
	Incr-RBFN-R	MLP-BP-R	iRProp+-R	Ensemble-R	
autoMPG6	5,104 (22)	5,580 (25)	3,090 (6)	5,516 (24)	. . .
autoMPG8	4,959 (24)	4,509 (21)	3,096 (9)	4,296 (20)	. . .
dee	0,520 (17)	0,410 (6)	0,409 (4)	**0,398 (1)**	. . .
diabetes	0,945 (24)	0,786 (20)	**0,532 (1)**	0,702 (13)	. . .
ele-2	* (26,5)	1566,674 (25)	234,050 (14)	1439,308 (24)	. . .
friedman	* (26,5)	2,651 (16)	2,106 (5)	2,396 (10)	. . .
laser	8,514 (4)	33,257 (27)	16,094 (15)	27,316 (25)	. . .
machineCPU	75,840 (15)	277,390 (26)	54,745 (2)	215,296 (25)	. . .
RNK	19,875 (23)	20,750 (24)	7,000 (4)	17,750 (20)	. . .

RMSE Test	NN4R NN4R	EFS4R			. . .
	NNEP-R	GFS-GAP-Sym-R	GFS-GSP-R	GFS-GP-R	
autoMPG6	**2,813 (1)**	7,203 (27)	4,134 (19)	7,093 (26)	. . .
autoMPG8	**2,746 (1)**	6,536 (27)	4,780 (23)	5,915 (26)	. . .
dee	0,403 (2)	0,809 (23)	0,663 (21)	0,923 (24)	. . .
diabetes	0,553 (2)	0,720 (14)	0,638 (8)	0,724 (15)	. . .
ele-2	175,428 (9)	779,370 (22) **	195,883 (11)	342,861 (18)	. . .
friedman	2,275 (7)	4,841 (25)	2,599 (15)	3,359 (22)	. . .
laser	9,921 (7)	25,561 (24)	16,626 (17)	30,859 (26)	. . .
machineCPU	**41,175 (1)**	98,118 (21)	66,976 (7)	102,260 (23)	. . .
RNK	**3,750 (1)**	22,875 (27)	15,125 (16)	22,500 (26)	. . .

Tabla 5.9: RMSE Media Test (1/3)- algoritmos de regresión disponibles en Keel - "*" indica que ninguna ejecución terminó antes de 12h - "**" indica que algunas ejecuciones no terminaron antes de 12h - "***" indica que ha dado como resultado algunos valores fuera del rango de la variable de salida - comparativa NSLVReg con algoritmos de Keel

$\alpha - value$ considerado. Si la hipótesis es rechazada significa que los algoritmos son significativamente diferentes.

RMSE Test	EFS4R	RL4R			...
		DT4R			
	GFS-SAP-Sym-R	M5-R	M5Rules-R	CART-R	
autoMPG6	4,437 (20)	2,943 (3)	3,179 (7)	3,805 (17)	...
autoMPG8	4,551 (22)	3,035 (6)	3,038 (7)	3,878 (17)	...
dee	0,740 (22)	0,417 (8,5)	0,417 (8,5)	0,585 (20)	...
diabetes	0,781 (19)	0,683 (9,5)	0,683 (9,5)	0,776 (17)	...
ele-2	175,036 (8)	91,644 (2)	96,563 (3)	980,384 (23)	...
friedman	2,510 (14)	2,388 (9)	2,351 (8)	3,046 (21)	...
laser	12,612 (12)	8,275 (3)	11,380 (10)	12,469 (11)	...
machineCPU	60,004 (3)	68,215 (8)	68,969 (9)	118,360 (24)	...
RNK	15,000 (15)	6,125 (3)	7,750 (6)	18,750 (21)	...

RMSE Test	RL4R				...
	FRL4R		EFRL4R		
	FRSBM-R	WM-R	TSK-IRL-R	MOGUL-IRLSC-R	
autoMPG6	3,483 (12,5)	3,422 (10)	4,734 (21)	3,202 (8)	...
autoMPG8	3,346 (12)	3,774 (16)	3,022 (5)	3,328 (10)	...
dee	0,410 (6)	0,456 (11)	4,353 (26,5) ***	0,495 (16)	...
diabetes	0,627 (6,5)	0,939 (23)	1,516 (27)	0,779 (18)	...
ele-2	164,521 (5,5)	335,311 (17)	213,315 (13)	305,539 (15)	...
friedman	2,700 (17,5)	2,231 (6)	1,645 (2)	2,399 (11)	...
laser	23,267 (21,5)	16,595 (16)	**7,594 (1)**	11,377 (9)	...
machineCPU	63,380 (5,5)	85,293 (20)	79,326 (18)	62,847 (4)	...
RNK	10,875 (8)	14,875 (14)	14,188 (13)	11,375 (10)	...

Tabla 5.10: RMSE Media Test (2/3)- algoritmos de regresión disponibles en Keel - "*" indica que ninguna ejecución terminó antes de 12h - "**" indica que algunas ejecuciones no terminaron antes de 12h - "***" indica que ha dado como resultado algunos valores fuera del rango de la variable de salida - comparativa NSLVReg con algoritmos de Keel

Los resultados de aplicar el test de Wilcoxon con un $\alpha - value$ de 0.05 se muestran en la tabla 5.13. En esta tabla se puede observar que, considerando la métrica RMSE, el primer algoritmo en ranking (NNEP-R) no es significativamente mejor que nuestra propuesta. Además, se puede observar que nuestra propuesta es significativamente mejor que los algoritmos que pertenecen a la segunda mitad del ranking, es decir, a partir de WM-R que ocupa la posición 14 en adelante.

A continuación se muestran los resultados utilizando la métrica $RMSE^N$ con el propósito de realizar una comparación dentro del mismo rango de valores. Los resultados para esta métrica se muestran en las tablas 5.14, 5.15 y 5.16. En ellas se observa que hay algunas leves modificaciones de puestos debido a la resolución

RMSE Test	RL4R				...
	EFRL4R				
	GFS-GPG-R	MOGUL-IRLHC-R	MOGUL-TSK-R	GFS-SP-R	
autoMPG6	3,584 (15)	3,215 (9)	3,480 (11)	2,993 (4)	...
autoMPG8	3,724 (15)	3,020 (4)	3,638 (14)	3,047 (8)	...
dee	0,493 (15)	0,466 (13)	1,080 (25)	0,486 (14)	...
diabetes	0,557 (4)	0,898 (22)	1,220 (25)	0,732 (16)	...
ele-2	366,634 (19)	309,385 (16)	**91,049 (1)**	196,331 (12)	...
friedman	3,714 (23)	2,480 (13)	**1,466 (1)**	2,961 (20)	...
laser	22,577 (20)	12,641 (13)	7,951 (2)	9,839 (6)	...
machineCPU	101,626 (22)	73,509 (14)	70,039 (11)	72,556 (12)	...
RNK	16,625 (18)	13,000 (12)	11,250 (9)	11,500 (11)	...

RMSE Test	RL4R			
	EFRL4R			
	Thrift-R	GFS-RB-MF-R		NSLVReg
autoMPG6	3,486 (14)	3,844 (18)		2,938 (2)
autoMPG8	4,016 (18)	4,171 (19)		2,752 (2)
dee	0,461 (12)	0,536 (18)		0,431 (10)
diabetes	0,794 (21)	0,700 (12)		0,608 (5)
ele-2	386,505 (20)	455,410 (21)		185,091 (10)
friedman	2,471 (12)	3,747 (24)		1,976 (4)
laser	22,400 (19)	22,083 (18)		15,099 (14)
machineCPU	77,430 (17)	313,634 (27)		69,102 (10)
RNK	16,625 (18)	19,625 (22)		7,125 (5)

Tabla 5.11: RMSE Media Test (3/3)- algoritmos de regresión disponibles en Keel - "*" indica que ninguna ejecución terminó antes de 12h - "**" indica que algunas ejecuciones no terminaron antes de 12h - "***" indica que ha dado como resultado algunos valores fuera del rango de la variable de salida - comparativa NSLVReg con algoritmos de Keel

de los valores de resultado. Por ello, en este caso NSLVReg obtiene un cuarto puesto en ranking, siendo lo más interesante su coincidencia con el puesto obtenido en media, lo que indica que el funcionamiento en media es muy aceptable, cuestión que no ocurre por ejemplo con M5-R que obtiene un tercer puesto en ranking pero un quinto puesto en media.

Igual que en los casos anteriores, vamos a realizar un análisis más profundo mediante los test de Shaffer y Wilcoxon. Los resultados del test de Shaffer se muestran en la tabla 5.17 donde, igual que se hizo con anterioridad, se muestran sólo los valores en los que interviene nuestra propuesta. Igualmente los resultados del test de Wilcoxon se muestran en la tabla 5.18. En estas tablas se observan unos resultados similares a los obtenidos utilizando

i	hypothesis	$p_{Shaffer}$
13	GFS-GAP-Sym-R vs .NSLV-Reg	0.023494
16	GFS-GP-R vs .NSLV-Reg	0.034776
25	RBFN-R vs .NSLV-Reg	0.120911
30	MLP-BP-R vs .NSLV-Reg	0.179562
37	Incr-RBFN-R vs .NSLV-Reg	0.395798
42	GFS-RB-MF-R vs .NSLV-Reg	0.491942
52	CART-R vs .NSLV-Reg	1.022824
67	Ensemble-R vs .NSLV-Reg	2.071006
87	GFS-GPG-R vs .NSLV-Reg	4.319078
88	Thrift-R vs .NSLV-Reg	4.319078
91	NU-SVR-R vs .NSLV-Reg	4.907835
119	GFS-GSP-R vs .NSLV-Reg	10.297651
122	GFS-SAP-Sym-R vs .NSLV-Reg	10.95525
128	WM-R vs .NSLV-Reg	11.439304
143	TSK-IRL-R vs .NSLV-Reg	15.780298
171	MOGUL-IRLHC-R vs .NSLV-Reg	25.812517
214	GFS-SP-R vs .NSLV-Reg	37.840514
219	MOGUL-IRLSC-R vs .NSLV-Reg	38.653198
225	MOGUL-TSK-R vs .NSLV-Reg	39.417533
237	FRSBM-R vs .NSLV-Reg	41.709211
238	LinearLMS-R vs .NSLV-Reg	41.709211
254	NNEP-R vs .NSLV-Reg	41.709211
292	PolQuadLMS-R vs .NSLV-Reg	41.709211
324	M5-R vs .NSLV-Reg	41.709211
333	M5Rules-R vs .NSLV-Reg	41.709211
346	iRProp+-R vs .NSLV-Reg	41.709211

Tabla 5.12: Resultados obtenidos por el test de Shaffer para RMSE - comparativa NSLVReg con algoritmos de Keel

RMSE lo que nos refuerza el buen comportamiento de la propuesta que aquí presentamos sin diferencias significativas con los que obtienen un mejor resultado.

Como hemos comentado a lo largo del capítulo, otra de las aportaciones de esta propuesta es la posibilidad de incluir un intervalo de error junto al valor de regresión. La tabla 5.19 muestra el porcentaje de error de ese intervalo de error en training y test para las bases de datos consideradas. En esta tabla se observa que en media se consigue que más del 80 % de los ejemplos de test se encuentren dentro del intervalo de error devuelto por el algoritmo. De forma gráfica, y a título representativo, se muestra la figura 5.11 donde se observa lo expuesto en la tabla para el caso de

VS	$p_{Wilcoxon}$
LinearLMS-R	$\geqslant 0.2$
PolQuadLMS-R	$\geqslant 0.2$
NU-SVR-R	0.19532
RBFN-R	0.07812
Incr-RBFN-R	0.07812
MLP-BP-R	0.015626
iRProp+-R	$\geqslant 0.2$
Ensemble-R	0.015626
NNEP-R	$\geqslant 0.2$
GFS-GAP-Sym-R	0.007812
GFS-GSP-R	0.14844
GFS-GP-R	0.007812
GFS-SAP-Sym-R	$\geqslant 0.2$
M5-R	$\geqslant 0.2$
M5Rules-R	$\geqslant 0.2$
CART-R	0.10938
FRSBM-R	$\geqslant 0.2$
WM-R	0.007812
TSK-IRL-R	0.19532
MOGUL-IRLSC-R	$\geqslant 0.2$
GFS-GPG-R	0.015626
MOGUL-IRLHC-R	0.10938
MOGUL-TSK-R	$\geqslant 0.2$
GFS-SP-R	0.14844
Thrift-R	0.007812
GFS-RB-MF-R	0.007812

Tabla 5.13: Resultados obtenidos por el test de Wilcoxon para RMSE - comparativa NSLVReg con algoritmos de Keel

resultados de test la base de datos ”Ele-2” cuyo comportamiento no es el mejor pero está en la media de los errores en test.

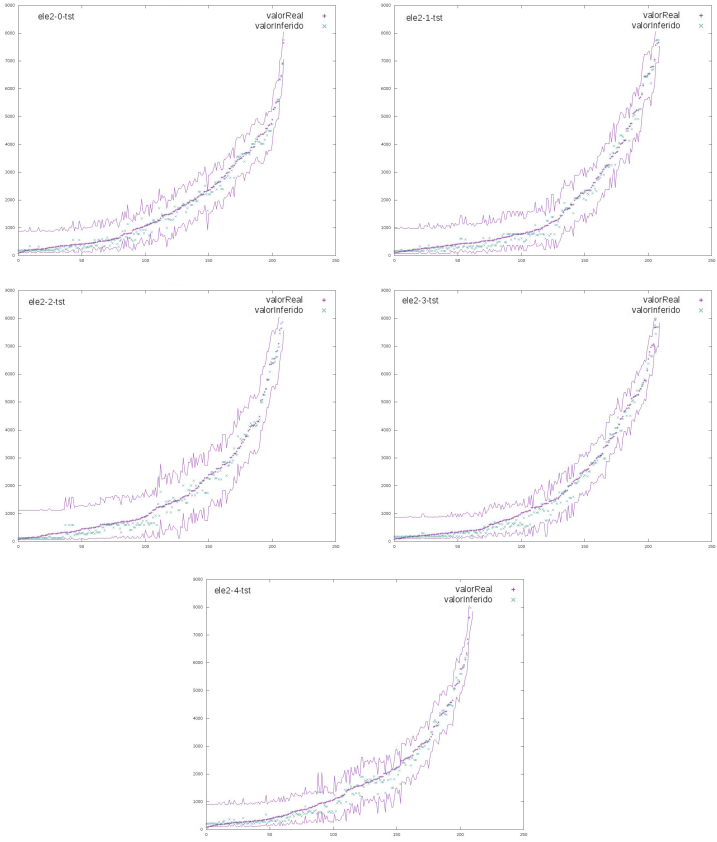


Figura 5.11: Resultados de regresión y el intervalo de error para el caso de dos particiones de test con la base de datos de Ele-2

$RMSE^N$ Test	Regresión Estadística Regresión Estadística		SVM4R SVM4R	NN4R NN4R NN4R	. . . . . .
	LinearLMS-R	PolQuadraticLMS-R	NU_SVR-R	RBFN-R	
autoMPG6	0,093 (12,5)	0,081 (5)	0,097 (16)	0,144 (23)	. . .
autoMPG8	0,089 (11)	0,074 (3)	0,093 (13)	0,139 (25)	. . .
dee	0,094 (5,5)	0,093 (2,5)	1,000 (26,5) ***	0,125 (19)	. . .
diabetes	0,174 (6,5)	0,154 (2,5)	0,191 (11)	0,390 (26)	. . .
ele-2	0,019 (5)	0,019 (5)	0,020 (7)	1,000 (26,5) ***	. . .
friedman	0,097 (17,5)	0,063 (3)	0,102 (19)	1,000 (26,5) ***	. . .
laser	0,091 (21,5)	0,043 (8)	0,094 (23)	0,037 (5)	. . .
machineCPU	0,055 (5)	0,064 (13,5)	0,067 (16) **	0,072 (19)	. . .
MEDIA	0,089 (8,5)	0,074 (2)	0,208 (24)	0,363 (27)	. . .
RNK	10,562 (7)	5,312 (2)	16,438 (17)	21,250 (25)	. . .

$RMSE^N$ Test	NN4R NN4R				. . .
	Incr-RBFN-R	MLP-BP-R	iRProp+-R	Ensemble-R	
autoMPG6	0,136 (22)	0,148 (25)	0,082 (6)	0,147 (24)	. . .
autoMPG8	0,132 (24)	0,120 (21)	0,082 (9)	0,114 (20)	. . .
dee	0,119 (17)	0,094 (5,5)	0,094 (5,5)	**0,091 (1)**	. . .
diabetes	0,263 (24)	0,218 (20)	**0,148 (1)**	0,195 (13)	. . .
ele-2	* (26,5)	0,185 (25)	0,028 (14)	0,170 (24)	. . .
friedman	* (26,5)	0,095 (16)	0,075 (5)	0,086 (10)	. . .
laser	0,033 (4)	0,130 (27)	0,063 (15)	0,107 (25)	. . .
machineCPU	0,066 (15)	0,242 (26)	0,048 (2)	0,188 (25)	. . .
MEDIA	0,344 (26)	0,154 (23)	0,078 (3)	0,137 (20)	. . .
RNK	19,875 (23)	20,688 (24)	7,188 (5)	17,750 (20)	. . .

$RMSE^N$ Test	NN4R NN4R	EFS4R			. . .
	NNEP-R	GFS-GAP-Sym-R	GFS-GSP-R	GFS-GP-R	
autoMPG6	**0,075 (1)**	0,192 (27)	0,110 (19)	0,189 (26)	. . .
autoMPG8	**0,073 (1,5)**	0,174 (27)	0,127 (23)	0,157 (26)	. . .
dee	0,093 (2,5)	0,186 (23)	0,152 (21)	0,212 (24)	. . .
diabetes	0,154 (2,5)	0,200 (14)	0,177 (8)	0,201 (15)	. . .
ele-2	0,021 (8,5)	0,092 (22) **	0,023 (11,5)	0,040 (17,5)	. . .
friedman	0,081 (7)	0,173 (25)	0,093 (15)	0,120 (22)	. . .
laser	0,039 (6,5)	0,100 (24)	0,065 (16,5)	0,121 (26)	. . .
machineCPU	**0,036 (1)**	0,086 (21)	0,059 (7)	0,089 (22,5)	. . .
MEDIA	**0,072 (1)**	0,150 (22)	0,101 (12)	0,141 (21)	. . .
RNK	**3,812 (1)**	22,875 (27)	15,125 (16)	22,375 (26)	. . .

Tabla 5.14: $RMSE^N$ Media Test (1/3)- algoritmos de regresión disponibles en Keel - "*" indica que ninguna ejecución terminó antes de 12h - "**" indica que algunas ejecuciones no terminaron antes de 12h - "***" indica que ha dado como resultado algunos valores fuera del rango de la variable de salida - comparativa NSLVReg con algoritmos de Keel

$RMSE^N$ Test	EFS4R	RL4R			...
		DT4R			
	GFS-SAP-Sym-R	M5-R	M5Rules-R	CART-R	
autoMPG6	0,118 (20)	0,078 (2,5)	0,085 (7,5)	0,101 (17)	...
autoMPG8	0,121 (22)	0,081 (7)	0,081 (7)	0,103 (17)	...
dee	0,170 (22)	0,096 (8,5)	0,096 (8,5)	0,134 (20)	...
diabetes	0,217 (19)	0,190 (9,5)	0,190 (9,5)	0,216 (17,5)	...
ele-2	0,021 (8,5)	0,011 (2)	0,011 (2)	0,116 (23)	...
friedman	0,090 (14)	0,086 (10)	0,084 (8)	0,109 (21)	...
laser	0,049 (11,5)	0,032 (3)	0,045 (9,5)	0,049 (11,5)	...
machineCPU	0,052 (3)	0,060 (9)	0,060 (9)	0,103 (24)	...
MEDIA	0,105 (16)	0,079 (5)	0,082 (6)	0,116 (17)	...
RNK	15,000 (14)	6,438 (3)	7,625 (6)	18,875 (21)	...

$RMSE^N$ Test	RL4R				...
	FRL4R		EFRL4R		
	FRSBM-R	WM-R	TSK-IRL-R	MOGUL-IRLSC-R	
autoMPG6	0,093 (12,5)	0,091 (10)	0,126 (21)	0,085 (7,5)	...
autoMPG8	0,089 (11)	0,100 (16)	0,080 (4,5)	0,089 (11)	...
dee	0,094 (5,5)	0,105 (11)	1,000 (26,5) ***	0,114 (16)	...
diabetes	0,174 (6,5)	0,261 (23)	0,421 (27)	0,216 (17,5)	...
ele-2	0,019 (5)	0,040 (17,5)	0,025 (13)	0,036 (15,5)	...
friedman	0,097 (17,5)	0,080 (6)	0,059 (2)	0,086 (10)	...
laser	0,091 (21,5)	0,065 (16,5)	**0,030 (1)**	0,045 (9,5)	...
machineCPU	0,055 (5)	0,075 (20)	0,069 (18)	0,055 (5)	...
MEDIA	0,089 (8,5)	0,102 (14,5)	0,226 (25)	0,091 (10)	...
RNK	10,562 (7)	15,000 (14)	14,125 (13)	11,500 (10)	...

Tabla 5.15: $RMSE^N$ Media Test (2/3)- algoritmos de regresión disponibles en Keel - "*" indica que ninguna ejecución terminó antes de 12h - "**" indica que algunas ejecuciones no terminaron antes de 12h - "***" indica que ha dado como resultado algunos valores fuera del rango de la variable de salida - comparativa NSLVReg con algoritmos de Keel

$RMSE^N$ Test	RL4R				...
	EFRL4R				
	GFS-GPG-R	MOGUL-IRLHC-R	MOGUL-TSK-R	GFS-SP-R	
autoMPG6	0,095 (15)	0,086 (9)	0,093 (12,5)	0,080 (4)	...
autoMPG8	0,099 (15)	0,080 (4,5)	0,097 (14)	0,081 (7)	...
dee	0,113 (15)	0,107 (13)	0,248 (25)	0,112 (14)	...
diabetes	0,155 (4)	0,249 (22)	0,339 (25)	0,203 (16)	...
ele-2	0,043 (19)	0,036 (15,5)	0,011 (2)	0,023 (11,5)	...
friedman	0,133 (23)	0,089 (13)	**0,052 (1)**	0,106 (20)	...
laser	0,089 (20)	0,050 (13)	0,031 (2)	0,039 (6,5)	...
machineCPU	0,089 (22,5)	0,064 (13,5)	0,061 (11)	0,063 (12)	...
MEDIA	0,102 (13)	0,095 (11)	0,117 (18)	0,088 (7)	...
RNK	16,688 (19)	12,938 (12)	11,562 (11)	11,375 (9)	...

$RMSE^N$ Test	RL4R			
	EFRL4R			
	Thrift-R	GFS-RB-MF-R		NSLVReg
autoMPG6	0,093 (12,5)	0,102 (18)		0,078 (2,5)
autoMPG8	0,107 (18)	0,111 (19)		**0,073 (1,5)**
dee	0,106 (12)	0,123 (18)		0,099 (10)
diabetes	0,221 (21)	0,194 (12)		0,169 (5)
ele-2	0,046 (20)	0,054 (21)		0,022 (10)
friedman	0,088 (12)	0,134 (24)		0,071 (4)
laser	0,088 (19)	0,087 (18)		0,059 (14)
machineCPU	0,068 (17)	0,274 (27)		0,060 (9)
MEDIA	0,102 (14,5)	0,135 (19)		0,079 (4)
RNK	16,438 (17)	19,625 (22)		7,000 (4)

Tabla 5.16: $RMSE^N$ Media Test (3/3)- algoritmos de regresión disponibles en Keel - "*" indica que ninguna ejecución terminó antes de 12h - "**" indica que algunas ejecuciones no terminaron antes de 12h - "***" indica que ha dado como resultado algunos valores fuera del rango de la variable de salida - comparativa NSLVReg con algoritmos de Keel

i	hypothesis	$p_{Shaffer}$
11	GFS-GAP-Sym-R vs .NSLV-Reg	0.020576
15	GFS-GP-R vs .NSLV-Reg	0.034776
23	RBFN-R vs .NSLV-Reg	0.107198
27	MLP-BP-R vs .NSLV-Reg	0.182917
34	Incr-RBFN-R vs .NSLV-Reg	0.354527
39	GFS-RB-MF-R vs .NSLV-Reg	0.441464
47	CART-R vs .NSLV-Reg	0.833643
65	Ensemble-R vs .NSLV-Reg	1.884314
83	GFS-GPG-R vs .NSLV-Reg	3.793268
85	NU-SVR-R vs .NSLV-Reg	4.508044
87	Thrift-R vs .NSLV-Reg	4.508044
114	GFS-GSP-R vs .NSLV-Reg	9.628811
117	GFS-SAP-Sym-R vs .NSLV-Reg	10.297651
118	WM-R vs .NSLV-Reg	10.297651
143	TSK-IRL-R vs .NSLV-Reg	15.246197
169	MOGUL-IRLHC-R vs .NSLV-Reg	25.040279
210	MOGUL-TSK-R vs .NSLV-Reg	36.29225
212	MOGUL-IRLSC-R vs .NSLV-Reg	36.29225
218	GFS-SP-R vs .NSLV-Reg	36.627211
241	LinearLMS-R vs .NSLV-Reg	41.878873
244	FRSBM-R vs .NSLV-Reg	41.878873
261	NNEP-R vs .NSLV-Reg	41.878873
295	PolQuadLMS-R vs .NSLV-Reg	41.878873
335	M5Rules-R vs .NSLV-Reg	41.878873
337	M5-R vs .NSLV-Reg	41.878873
343	iRProp+-R vs .NSLV-Reg	41.878873

Tabla 5.17: Resultados obtenidos por el test de Shaffer para $RMSE^N$ - comparativa NSLVReg con algoritmos de Keel

VS	$p_{Wilcoxon}$
LinearLMS-R	0.19532
PolQuadLMS-R	$\geqslant 0.2$
NU-SVR-R	0.015626
RBFN-R	0.02344
Incr-RBFN-R	0.03906
MLP-BP-R	0.015626
iRProp+-R	$\geqslant 0.2$
Ensemble-R	0.015626
NNEP-R	$\geqslant 0.2$
GFS-GAP-Sym-R	0.007812
GFS-GSP-R	0.019533
GFS-GP-R	0.007812
GFS-SAP-Sym-R	0.10938
M5-R	$\geqslant 0.2$
M5Rules-R	$\geqslant 0.2$
CART-R	0.015626
FRSBM-R	0.19532
WM-R	0.007812
TSK-IRL-R	$\geqslant 0.2$
MOGUL-IRLSC-R	0.0664
GFS-GPG-R	0.02344
MOGUL-IRLHC-R	0.07812
MOGUL-TSK-R	$\geqslant 0.2$
GFS-SP-R	0.10938
Thrift-R	0.007812
GFS-RB-MF-R	0.007812

Tabla 5.18: Resultados obtenidos por el test de Wilcoxon para $RMSE^N$ - comparativa NSLVReg con algoritmos de Keel

Base de Datos	% Train	% Test
autoMPG6	2.17	17.34
autoMPG8	1.59	21.44
dee	1.71	33.70
diabetes	18.64	58.61
ele-2	0.85	0.66
friedman	0.35	20.17
laser	1.64	2.52
machineCPU	2.51	7.62
MEDIA	3.27	18.01

Tabla 5.19: Porcentaje de ejemplos fuera del intervalo de error

5.4 CONCLUSIONES

Este capítulo presenta una propuesta para abordar los problemas de regresión. Se presenta una extensión de NSLVOrd [52, 53] a problemas de regresión llamada NSLVReg. Como novedad se presenta un intervalo de error que nos estima cuál puede ser el error que puede haberse producido junto al valor de salida real. Esto es gracias a la consideración de un algoritmo de clasificación ordinal como algoritmo de base para el aquí expuesto.

Para el desarrollo de esta propuesta ha sido necesaria una redefinición del algoritmo de aprendizaje NSLVOrd usado como base. Se definen dos conjuntos de bases de reglas diferentes. Por una parte hay diferentes bases de reglas definidas por distintos niveles de precisión. Por otra parte hay diferentes bases de reglas en el mismo nivel de precisión, las cuales son derivadas de un desplazamiento en los intervalos en los que se discretiza la variable de salida del problema. El sistema aprende las bases de reglas de una forma incremental tanto entre niveles como dentro del mismo nivel. También ha sido necesaria la redefinición del algoritmo que implementa la estrategia de recubrimiento secuencial para la consideración en el nuevo aprendizaje de las reglas aprendidas previamente. Finalmente son necesarios la combinación de los resultados de la inferencia para considerar las bases de reglas dentro de un mismo nivel así como entre los distintos niveles.

Para mostrar el buen comportamiento de la propuesta aquí presentada para este tipo de problemas se han realizado varias experimentaciones donde se muestran los buenos resultados obtenidos tanto en comparación con algoritmos de su categoría como comparándolo con un amplio abanico de algoritmos de regresión.

La propuesta presentada en este capítulo puede dar lugar a diversas líneas de trabajos futuros. Por una parte se podría considerar otros mecanismos de inferencia más elaborados tanto dentro de un nivel de bases de reglas, como entre los distintos niveles. Además, se puede considerar la posibilidad de realizar el disparo de varias reglas y las distintas posibilidades de combinación de los resultados de los diferentes disparos de las reglas. También se puede estudiar el uso de reglas aproximativas básicas que no dificulten el entendimiento de las reglas al mismo tiempo que puedan ayudar a la mejora de la precisión. En el ámbito de la optimización de algoritmos se abre la posibilidad de paralelización tanto a bajo nivel mediante el uso de hardware de propósito

específico (GPU) como mediante el uso de algún tipo de clusters de equipos. Por último, como sucede en todos los algoritmos que utilizan genéticos se podría analizar la influencia de incluir otros tipos de cruces y mutaciones.

6

CONCLUSIONES Y TRABAJOS FUTUROS

Muchos de los fracasos de los hombres es porque no se dieron cuenta lo cerca que estaban del éxito cuando desistieron.
No sólo he inventando la lámpara incandescente, sino que he descubierto cientos de formas de cómo no hay que hacerla
Thomas Alva Edison [1]

ÍNDICE

1 Thomas Alva Edison (1847 - 1931) empresario e inventor estadounidense. Patentó más de mil inventos siendo los más conocidos la lámpara incandescente y el fonógrafo.

6.1 CONCLUSIONES

El objetivo de esta tesis es *la extensión de un modelo genético iterativo a problemas de clasificación ordinal y regresión*. Este objetivo puede ser considerado como un doble objetivo donde se abordan por una parte los problemas de clasificación ordinal y por otra parte los problemas de regresión. Para ello se plantearon una serie de objetivos y subobjetivos que han sido cubiertos a lo largo de los diferentes capítulos de esta memoria.

El capítulo 4 muestra la evolución de los planteamientos realizados para la consecución del primer objetivo de este trabajo, es decir, la implementación de un algoritmo para abordar problemas de clasificación ordinal. Como resultado se ha desarrollado el algoritmo NSLV Ordinal (NSLVOrd), el cual muestra un buen comportamiento para el tipo de problemas para el que ha sido desarrollado. En este capítulo se ha propuesto una serie de mejoras sobre el algoritmo base NSLV junto con la modificación de algunos elementos del AG. Finalmente se ha propuesto una nueva función fitness basada en una combinación de las métricas mostradas en el capítulo 2 de forma que se exploten las peculiaridades de este tipo de problemas. La consecución de este objetivo se puede observar mediante las pruebas, resultados y conclusiones mostrados en el mismo capítulo.

Como segundo objetivo, en el capítulo 5 de esta memoria se plantea una forma de abordar problemas de regresión haciendo uso de las características que comparte la clasificación ordinal con la regresión. Para ello, en este capítulo se presentan todas las modificaciones necesarias para el desarrollo de un algoritmo (NSLVReg) que aborde este tipo de problemas. Entre estas modificaciones, es de destacar las diversas discretizaciones de la variable de salida en función del nivel del aprendizaje, lo que define la precisión teórica inicial a la que se puede llegar. También se proponen diversos esquemas de ejecución y combinación del algoritmo ordinal NSLVOrd. Igualmente se exponen diferentes combinaciones de los resultados de los clasificadores ordinales optando finalmente por la mejor solución. Por último, un punto importante de la propuesta planteada en este capítulo, es la posibilidad de dar como resultado un intervalo de error del valor de salida adicional a este valor de salida. Este intervalo de error hace referencia el posible error cometido por el uso de algoritmos de clasificación ordinal para la resolución de problemas de regre-

sión. Al final del capítulo 5 se muestran las pruebas, resultados y conclusiones que indican la consecución de este segundo objetivo.

Por último, como conclusión general, podríamos decir que se han cubierto los dos objetivos principales de esta tesis con el desarrollo de una propuesta para problemas de clasificación ordinal, así como otra propuesta que resuelve problemas de regresión.

6.2 TRABAJOS FUTUROS

El estudio y desarrollo de las tareas que componen los objetivos y subobjetivos de este trabajo pueden dar lugar a varias ideas de mejora del mismo. Estas ideas podrían formar diferentes líneas de trabajo futuro. Entre las ideas planteadas a lo largo del desarrollo de este trabajo podemos destacar las siguientes.

Referente a la propuesta de clasificación ordinal, se podría plantear el estudio de diferentes mecanismos de preprocesamiento de datos para el ajuste de las etiquetas de las variables haciendo uso de AG [5, 47] o EM (Expectation-Maximisation) [13]. Continuando con el preprocesamiento de datos, otra posible vía de estudio es el uso de selección de instancias [96, 112, 113] con el objetivo de minimizar el tiempo de procesamiento derivado del uso de un AG. Con este mismo objetivo se podría estudiar diferentes modificaciones en la población, ya sea con reinicialización de la población [133] o mediante diferentes subpoblaciones independientes [66]. También relativo al AG un posible trabajo futuro sería el estudio del cambio en la función fitness convirtiendo el algoritmo en un multiobjetivo haciendo uso de los frentes de pareto [66, 133].

Referente a la propuesta de regresión, se podría estudiar la aplicación de otros mecanismos de combinación e interpolación de reglas [134, 135]. Respecto a la inferencia, la propuesta realizada utiliza la inferencia tipo Mandani ya que utilizamos este tipo de reglas, aunque se podrían estudiar el comportamiento del sistema con reglas tipo Tsukamoto o AnYa como indica el "Standard IEEE 1855 ™-2016" [2] y aplicar su inferencia, así como el uso de más de un consecuente en cada regla difusa. Respecto a la interfaz de defuzificación, se ha implementado el modo de defuzificar primero y agregar después, aunque podría ser interesante agregar primero y defuzificar después, así como podría ser interesante

2 http://standards.ieee.org/findstds/standard/1855-2016.html

el estudio de otros métodos de defuzificación, junto con técnicas de combinación de términos lingüísticos también presentados en el mismo estándar y cuya implementación está disponible en [3]. Desde otro punto de vista, tomando la propuesta como un tipo de multiclasificador, se podría estudiar la aplicación de técnicas multiclasificadores [48] para el ajuste de la precisión en regresión.

Finalmente, como trabajo futuro más cercano se plantea la extensión del algoritmo de clasificación ordinal a BigData [40, 93, 110, 111]. De este trabajo futuro ya se han realizado algunos avances en [49, 54, 55]. Igualmente se puede plantear la extensión a BigData del algoritmo de regresión propuesto.

Como broche final, todos los estudios de investigación que realizamos no tiene sentido si no es de aplicación a problemas reales y no sólo a bases de datos de benchmark. Para ello se está desarrollando un proyecto con título *Apoyo al desarrollo de la motricidad gruesa en niños con problemas utilizando sensores 3D de última generación* que ha sido presentado en los Proyectos UCO-SOCIAL-INNOVA del II Plan Propio GALILEO de Innovación y Transferencia [4] [5] y que ha obtenido la máxima puntuación. Una prueba piloto se está desarrollando en el Centro de Atención Temprana de la Universidad de Córdoba. Los resultados de este proyecto esperamos que sean directamente aplicables a los demás Centros de Atención Temprana de todas las regiones que quieran incorporarlo con lo que se pretende que se beneficien de estos avances tanto los niños como los especialistas que trabajan con estos niños.

3 http://www.uco.es/JFML/home

4 https://www.uco.es/investigacion/transferencia/plan-propio-innovacion-y-transferencia#modalidad-iv-uco-social-innova

5 https://sede.uco.es/bouco/bandejaAnuncios/BOUCO/2016/00306#

APÉNDICES

El fracaso es una gran oportunidad para empezar otra vez con más inteligencia.
Henry Ford [6]

6 Henry Ford (1863-1947) fundador de la compañía Ford Motor Company y padre de las cadenas de producción modernas utilizadas para la producción en masa.

A

INFLUENCIA DEL NÚMERO DE INTERVALOS EN REGRESIÓN

En este apéndice se muestran los resultados del incremento en el número de intervalos que se consideran en el algoritmo de aprendizaje ordinal. Con este incremento de intervalos se pretende disminuir el error cometido al particionar la variable de salida del problema de regresión original. Como conclusión de los resultados se observa que, en general, no se disminuye el error incrementándose de forma notable el tiempo de computo.

En la tabla A.1 se muestran los resultados respecto de la métrica RMSE para la resolución de los problemas de regresión cuando se utilizan distintas particiones de la variable de salida. Estas particiones condicionan el número de clases que se considera en el algoritmo NSLVOrd utilizado como base. Se observa que el incremento del número de intervalos en los que se particiona la variable de salida no implica una mejora en los resultados.

RMSE Media Test	5 clases	7 clases	11 clases	21 clases	51 clases
autompg6	4,253 (5)	4,221 (4)	3,843 (2)	4,152 (3)	**3,795 (1)**
autompg8	4,274 (5)	**3,825 (1)**	3,850 (2)	3,913 (4)	3,893 (3)
dee	0,618 (5)	**0,577 (1)**	0,607 (2,5)	0,607 (2,5)	0,617 (4)
diabetes	0,751 (4)	0,709 (2)	0,749 (3)	0,759 (5)	**0,702 (1)**
ele-2	614,922 (5)	527,070 (4)	440,548 (3)	**339,063 (1)**	340,294 (2)
friedman	3,553 (3)	**3,367 (1)**	3,376 (2)	3,591 (4)	3,601 (5)
laser	25,220 (5)	22,690 (4)	18,175 (2)	**16,733 (1)**	18,435 (3)
machineCPU	91,446 (5)	69,348 (3)	72,540 (4)	**55,044 (1)**	61,211 (2)
RNK	4,625 (5)	**2,500 (1)**	2,562 (2)	2,688 (4)	2,625 (3)

Tabla A.1: RMSE Media Test - Número de clases en NSLVOrd para problemas de regresión

La tabla A.2 muestra los tiempos empleados en cada uno de los problemas de regresión para cada una de las clases consideradas en el algoritmo NSLVOrd utilizado de base. Junto a estos valores aparece el porcentaje de incremento en tiempo respecto al resultado con 5 clases. Se observa un notable incremento en tiempo

derivado del aumento del número de clases que no implica una mejora en los resultados como se muestra en la tabla A.1.

Tiempo (seg)	5 clases	7 clases	11 clases	21 clases	51 clases
autompg6	12,8	22,4 (1,75 %)	41,4 (3,32 %)	282,6 (22,08 %)	1923,8 (150,3 %)
autompg8	12,2	22 (1,8 %)	47 (3,85 %)	324,6 (26,61 %)	2192,6 (179,72 %)
dee	7,6	13 (1,71 %)	31,8 (4,18 %)	135 (17,76 %)	852,6 (112,18 %)
diabetes	1,4	2 (1,43 %)	3,8 (2,71 %)	12 (8,57 %)	26 (18,57 %)
ele-2	20,4	33,2 (1,63 %)	88,4 (4,33 %)	389,4 (19,09 %)	2346,6 (115,03 %)
friedman	49,8	78,8 (1,58 %)	161,8 (3,25 %)	4463,4 (89,63 %)	25395 (509,94 %)
laser	25,2	33,8 (1,34 %)	67 (2,66 %)	498 (19,76 %)	2426,4 (96,29 %)
machineCPU	2,8	4,8 (1,71 %)	9,2 (3,29 %)	40,2 (14,36 %)	145,8 (52,07 %)

Tabla A.2: Tiempos e incremento respecto al tiempo empleado con 5 clases - Número de clases en NSLVOrd para problemas de regresión

B

INFLUENCIA DE LOS DESPLAZAMIENTOS DE LOS INTERVALOS EN REGRESIÓN

En este apéndice se muestran los resultados de la consideración de desplazamiento en los intervalos de la variable de salida. Con estos desplazamientos se produce un solapamiento en dichos intervalos. El objetivo es mejorar los resultados como se muestra en la tabla B.1. En dicha tabla se observa una mejora en los resultados con distintos valores de desplazamiento a partir de un 25 %, aunque esa mejora es dependiente del problema en cuestión sin poder considerar un desplazamiento mejor que otro.

RMSE Media Test	1 Base de Reglas	Desp 10 %	Desp 20 %	Desp 25 %	Desp 30 %	Desp 40 %
autompg6	4,221 (6)	3,559 (5)	3,470 (4)	**3,338 (1)**	3,447 (3)	3,359 (2)
autompg8	3,825 (6)	3,502 (5)	3,230 (2)	3,406 (4)	**3,130 (1)**	3,304 (3)
dee	0,577 (6)	0,498 (2)	0,512 (5)	**0,498 (1)**	0,505 (3)	0,511 (4)
diabetes	0,709 (6)	0,636 (5)	0,615 (3)	**0,570 (1)**	0,577 (2)	0,619 (4)
ele-2	527,070 (6)	393,060 (5)	371,402 (4)	349,528 (3)	346,834 (2)	**329,443 (1)**
friedman	3,367 (6)	2,659 (3)	2,871 (5)	**2,620 (1)**	2,767 (4)	2,646 (2)
laser	22,690 (6)	17,627 (5)	16,599 (2)	16,653 (4)	**16,087 (1)**	16,607 (3)
machineCPU	69,348 (2)	71,349 (3)	74,605 (6)	73,896 (5)	71,718 (4)	**67,596 (1)**
RNK	5,500 (6)	4,125 (5)	3,875 (4)	**2,500 (1)**	**2,500 (1)**	**2,500 (1)**

Tabla B.1: RMSE Media Test - Desplazamientos en los intervalos para problemas de regresión

C

COMBINACIÓN DE LAS SALIDAS DE LOS CLASIFICADORES ORDINALES EN REGRESIÓN

En este apéndice se muestran los resultados de las diferentes combinaciones de las salidas de los clasificadores NSLVOrd con y sin desplazamiento en los intervalos de la variable de salida.

Para obtener el valor real del problema de regresión hay un gran número de alternativas. En primer lugar, si se considera cada clasificador como una caja negra que devuelve un valor correspondiente a la clase resultado, se puede plantear un considerable número de opciones debido a la diversidad de configuraciones derivadas de combinar desplazamientos y niveles. En segundo lugar, si se considera el algoritmo para resolver problemas de regresión como un todo en el que se puede acceder a cada una de las bases de reglas junto con sus características como, por ejemplo, dominio de las variables, adaptaciones de los ejemplos, peso de la regla, etc, se abre otro conjunto de alternativas más amplio si cabe. A estas opciones hay que añadir las que pueden surgir de las combinaciones de los dos grupos anteriores.

- *Combinación dentro de un nivel*: A continuación, a modo de ejemplo, se presenta algunas de las alternativas consideradas para combinar los tres clasificadores de un mismo nivel:

    1. En una primera alternativa, el valor resultado de la combinación de las tres bases de reglas puede ser obtenido mediante la *media* de los valores:

$$\text{Media} = \frac{y'_i + y'_c + y'_d}{3} .$$

    donde el subíndice de $y'_{subindex}$ es el valor real calculado a partir de las salidas de los clasificadores con un desplazamiento a la izquierda, derecha o sin desplazamiento respectivamente. El valor inferido es el valor de la media del intervalo correspondiente a la

clase de la variable de salida asociada con la regla disparada.

2. En una segunda alternativa, para el cálculo del valor resultado puede ser considerados otros aspectos como la adaptación del ejemplo a la regla así como el peso de la regla junto con el cálculo de los valores medios. De esta forma se obtiene una *media ponderada* como se muestra en la fórmula siguiente.

$$\text{MediaPonderada} = \frac{y'i * \text{Pond}_i + y'c * \text{Pond}_c + y'd * \text{Pond}_d}{\text{Pond}_i + \text{Pond}_c + \text{Pond}_d}.$$

donde $\text{Pond}_{subindex} = U_{subindex}(e, A) * w_{subindex}(R_B(A))$ y el subíndice indica la base de reglas considerada en cada caso, es decir, la que tiene el desplazamiento a la izquierda a la derecha o no tiene desplazamiento.

3. Una tercera opción es la siguiente. Las diferencias entre las tres bases de reglas se fundamenta en un desplazamiento en los intervalos que definen los valores de salida. Estos intervalos están solapados en un grado que es función del desplazamiento definido. Por todo ello, otra aproximación puede estar basada en el grado de solapamiento entre los intervalos asociados a cada clase resultantes de las inferencias de cada clasificador. Dentro de esta aproximación pueden ser consideradas las siguientes alternativas para el cálculo del valor resultado.

   A continuación se muestran los posibles solapamientos entre los intervalos y cómo se actúa en cada caso:

   a) *Ningún intervalo está solapado (figura C.1.a)*: En este caso consideramos que puede haber un error en las salidas de las inferencias. Como no se puede asegurar si el error se produce en todas o sólo en algunas de esas inferencias, se devuelve el punto medio de la unión de todos los intervalos.

   b) *Sólo se produce un solapamiento entre dos intervalos quedando el intervalo correspondiente a la tercera clase al margen de éstas (figura C.1.b)*: En este caso se puede considerar que el intervalo que está fuera

es un error devolviendo el punto medio de la unión de los intervalos que están solapados, lo que se puede entender como una actitud *agresiva*. También se puede considerar que no hay error y devolver al igual que antes el punto medio de la unión todos los intervalos, lo que se puede entender como una actitud *conservadora*.

c) *Un intervalo tiene solapamiento con los otros dos aunque éstos no están solapados entre sí (figura C.1.c)*: En este caso consideramos que los tres resultados de la inferencia son correctos aunque no muy cercanos entre sí, por lo que puede haber una gran variación o diferencia de error con el valor real. Al igual que en el caso a) se devuelve el punto medio de la unión de todos los intervalos.

d) *Todos los intervalos están solapados (figura C.1.d)*: En este caso consideramos que los tres resultados de la inferencia son correctos y muy cercanos al valor real. Igual que en el caso anterior y el primero se devuelve el punto medio de la unión de todos los intervaloss.

En la tabla C.1 se muestran los resultados de las alternativas expuestas anteriormente donde se observa que la opción con mejor resultados es la que utiliza la media aritmética.

RMSE Media Test	Media	Media Ponderada	Agresiva	Conservadora
autompg6	**3,338 (1)**	3,380 (2)	3,478 (3)	3,482 (4)
autompg8	**3,406 (1)**	3,433 (2)	3,464 (4)	3,454 (3)
dee	**0,498 (1)**	0,506 (2)	0,514 (4)	0,511 (3)
diabetes	**0,570 (1)**	0,580 (2)	0,649 (4)	0,646 (3)
ele-2	349,528 (3)	349,859 (4)	**298,448 (1)**	323,098 (2)
friedman	**2,620 (1)**	2,629 (2)	2,979 (4)	2,972 (3)
laser	16,653 (4)	16,436 (3)	**15,863 (1)**	16,054 (2)
machineCPU	73,896 (4)	71,253 (3)	**66,013 (1)**	67,710 (2)
RNK	**2,000 (1)**	2,500 (2)	2,750 (3)	2,750 (3)

Tabla C.1: RMSE Media Test - Combinación de salidas en el mismo nivel para problemas de regresión

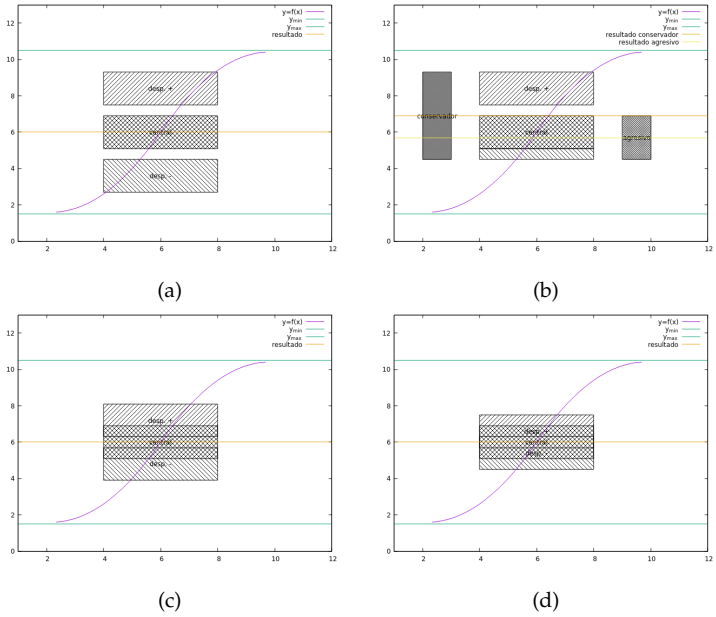


Figura C.1: Tipos de solapamientos entre intervalos

- *Combinación de diferentes niveles*: Hasta este punto se han presentado distintas alternativas de combinación de resultados dentro de un nivel. Lo siguiente sería estudiar las alternativas de combinación de los valores resultado de cada nivel. Si consideramos que de cada nivel se obtiene un único valor de alguna de las formas comentadas anteriormente, obtenemos tantos valores como niveles de aprendizaje se tenga. A partir de estos valores se puede obtener un valor real como resultado del problema de regresión mediante cualquiera de las fórmulas estadísticas como la media, mediana, media ponderada asignando un peso a cada nivel, etc.

  Otra forma de obtener el valor de regresión es considerando el algoritmo de regresión como un todo en el que se consideran al mismo tiempo los diferentes valores dentro de un nivel y entre niveles. Con esta consideración, se pueden plantear diferentes alternativas. Una de ellas consiste en la creación de una matriz de clases de resultado. Esta propuesta trata de tomar ventaja del algoritmo ordinal del que parte. La salida del algoritmo ordinal es una clase que

tiene asociada un intervalo en el cual se encuentra el valor resultado. En las propuestas previas se da el valor medio de este intervalo como valor real de resultado. En este caso se pretende que el resultado sea tanto el valor real como el intervalo en el que se encuentra la solución. Para ello se plantea limitar los intervalos lo máximo posible. Esta limitación se realiza gracias a la subdivisión de las clases conforme vamos avanzando en los niveles de bases de reglas, junto con el desplazamiento que se produce en cada nivel. Para el manejo de estos valores, clases e intervalos se crean diferentes matrices de tantas filas como niveles de bases de reglas tengamos y tres columnas correspondientes a las bases de reglas de cada nivel (izquierda, central, derecha). De esta forma una matriz de clases inferidas correspondiente a un ejemplo podría ser la siguiente:

$$\begin{pmatrix} 2 & 2 & 2 \\ 4 & 4 & 4 \\ 7 & 9 & 7 \\ 15 & 14 & 14 \end{pmatrix}$$

Considerando la subdivisión de las clases mostradas en la figura 5.8(a) y codificando las clases desde 0 a $(numLevels - 1)$, se puede observar que el número de la clase de nivel i se corresponde con el doble del número de clase del nivel anterior, es decir, si en el nivel 0 tenemos una clase 2, en el nivel 1 se correspondería con la clase 4 y en el nivel 2 con la 8 y así sucesivamente. El cálculo de la clase en cada nivel se obtiene según la ecuación $2^{(numNiveles - nivelActual - 1)}$ considerando que la primera fila es la correspondiente al 0 como primer nivel y así sucesivamente. De esta forma, tomando como referencia el último nivel, la matriz anterior se podría convertir en la siguiente:

$$\begin{pmatrix} 16 & 16 & 16 \\ 16 & 16 & 16 \\ 14 & 18 & 14 \\ 15 & 14 & 14 \end{pmatrix}$$

Con esto se tiene la clasificación del ejemplo como si de un único nivel se tratara. En este ejemplo se puede observar que la mayoría de los clasificadores lo han clasificado como de clase 16. Según lo mostrado en la figura 5.8(a) se observa que la clase de nivel 0 (con 5 clases) se podría corresponder con tres clases del nivel 1 (con 9 clases). Esto hace pensar que el valor 15 de la última fila podría ser válido, e incluso mejor que el de la clase 16 ya que la clasificación se ha "ajustado" más al valor de salida real. Al mismo tiempo hay que recordar que se trabaja con bases de reglas con los intervalos desplazados (figura 5.5) por lo que los valores 14 del último nivel también podrían ser válidos e incluso mejores que los de los niveles anteriores.

Para hacer uso de todo lo anterior esta alternativa crea una matriz como la anterior y sobre esa matriz se lleva a cabo un sistema de votación ponderada con el nivel de aprendizaje, en el que se decidirá qué clase del último nivel se debe tomar como la mejor. Una vez seleccionada la clase, el resultado será el punto medio del intervalo asociado a la clase seleccionada. Adicional a este valor, se presenta un rango o intervalo de error que estará definido por los valores mínimos y máximos del intervalo asociado a la clase seleccionada.

Para el sistema de votación, en primer lugar se crea un vector con tantas posiciones como etiquetas tengamos en el último nivel, anotándose sobre el mismo cuando se encuentre la etiqueta en la matriz. Con la idea de dar más peso a las etiquetas obtenidas en el último nivel respecto a las anteriores se pondera cada fila con la ecuación $2^{nivelActual}$. De esta forma, en la posición 16 tendríamos el valor $2^0 * 3 + 2^1 * 3 = 9$. El resto de valores del vector correspondientes al ejemplo que estamos tratando para los índices desde el 13 al 19 serían:

$$..., 0, 16, 8, 9, 0, 4, 0, ...$$

Para incorporar los posibles desplazamientos en las etiquetas se crea otro vector del mismo tamaño que el anterior en el que cada posición i tendrá el valor correspondiente a

$v[i-1] + 2 * [vi] + v[i+1]$. De esta forma, con la consideración de los valores anterior y posterior se tiene en cuenta el desplazamiento y con la ponderación del valor de la posición actual se da más peso a esa etiqueta. De esta forma el vector para los índices desde el 13 al 19 quedaría:

$$..., 16, 40, 41, 26, 13, 8, 4, ...$$

Como se puede observar, aunque inicialmente se puede pensar que la mejor clase es la 16, el vector anterior nos indica que la mejor clase es la 15, aunque está muy cerca la 14.

La tabla C.2 muestra los resultados de la ejecución del aprendizaje. La primera columna hace referencia a la ejecución de un sólo nivel con 7 intervalos en la variable de salida y un 25 % de desplazamiento correspondiente al mejor valor de la tabla C.1. Los resultados de la segunda columna corresponden a la ejecución del aprendizaje con 33 intervalos y un 25 % de desplazamiento en un único nivel de aprendizaje. Para los resultados de la tercera columna se ha realizado un aprendizaje en varios niveles obteniendo varias bases de reglas en los diferentes niveles. Los valores de las tres primeras columnas son calculados con la media aritmética de los valores resultantes de cada nivel, los cuales también son calculados mediante la media aritmética que según se expuso en los párrafos anteriores es la opción que mejor resultados presenta. Por último, la columna cuarta muestra los resultados calculados con la matriz explicada anteriormente para 4 niveles de aprendizaje con un total de 33 intervalos con el objetivo de poder comparar con las columnas anteriores. De esta tabla se desprende que, como se comentó anteriormente, no por aumentar el número de clases se mejoran los resultados. Además, se observa que la opción que mejor resultado obtiene es la que permite el aprendizaje en varios niveles de clasificadores NSLVOrd realizando la combinación de todos ellos mediante las medias aritméticas.

RMSE Media Test	7 clases	33 clases	33 cl - 4 niveles	Matriz
autompg6	3,338 (2)	3,594 (3)	**3,172 (1)**	3,799 (4)
autompg8	3,406 (4)	3,358 (2)	**2,787 (1)**	3,403 (3)
dee	0,498 (2,5)	0,498 (2,5)	**0,437 (1)**	0,525 (4)
diabetes	**0,570 (1)**	0,673 (3)	0,638 (2)	0,718 (4)
ele-2	349,528 (4)	230,586 (2)	**209,852 (1)**	248,570 (3)
friedman	2,620 (2)	3,022 (4)	**2,058 (1)**	2,933 (3)
laser	16,653 (2)	**13,668 (1)**	19,972 (3)	32,948 (4)
machineCPU	73,896 (4)	**58,687 (1)**	65,499 (2)	72,482 (3)
RNK	2,688 (3)	2,312 (2)	**1,500 (1)**	3,500 (4)

Tabla C.2: RMSE Media Test - Combinación de salidas en el todos los nivel para problemas de regresión

D

ALTERNATIVAS DE EJECUCIÓN DE CLASIFICADORES ORDINALES EN EL SISTEMA COMPLETO PARA REGRESIÓN

En este apéndice se muestran los resultados de las diferentes formas de realizar el aprendizaje. Más concretamente se presentan los resultados de las distintas alternativas a la hora de ejecutar la modificación del algoritmo NSLVOrd en los diferentes desplazamientos de los distintos niveles que forman el aprendizaje completo del problema de regresión.

Algunas de las diferentes alternativas son las siguientes:

- *Algoritmos independientes*: Cada una de las ejecuciones de los desplazamientos y niveles se pueden considerar como independientes por lo que se podrían ejecutar todos ellos tanto en forma secuencial como paralela.
- *Copia de las Bases de Reglas*: Debido a las similaridades entre los distintos problemas que se generan por los desplazamientos y las particiones de la variable de salida se puede pensar en reutilizar el aprendizaje generado anteriormente. Esta opción abre varias alternativas al mismo tiempo entre las que destacamos:
  - *Copia horizontal*: Copiar las BR resultado del clasificador central a los clasificadores de los extremos (correspondientes a los desplazamientos) y al central del siguiente nivel.
  - *Copia vertical*: Copiar las BR resultado de cada clasificador a su correspondiente del siguiente nivel, es decir, la BR del clasificador del desplazamiento que llamamos a la izquierda se copiará al clasificador correspondiente al desplazamiento a la izquierda del siguiente nivel, realizando lo mismo con el resto de clasficadores de cada nivel.
- *Generación de individuos del* AG *a partir de las BR*: En este caso no se produce una copia de las bases de reglas, sino que se realiza una traducción de cada regla de la base de reglas

al individuo correspondiente que la produjo, introduciendo este individuo en la población del AG del clasificador correspondiente. Al igual que en el caso de la copia de las bases de reglas se pueden presentar otras alternativas entre las que destacamos las mismas que en caso anterior:

- *Generación horizontal*: A partir de las BR resultado del clasificador central se generan los individuos para los clasificadores de los extremos (correspondientes a los desplazamientos) y al central del siguiente nivel.
- *Generación vertical*: A partir de las BR resultado de cada clasificador se generan los individuos para su correspondiente del siguiente nivel, es decir, a partir de la BR del clasificador del desplazamiento que llamamos a la izquierda se generarán los individuos para el clasificador correspondiente al desplazamiento a la izquierda del siguiente nivel, realizando lo mismo con el resto de clasficadores de cada nivel.

La tabla D.1 muestra los resultados de estas alternativas donde se observa que la que presenta mejor resultado es la opción de generación horizontal de individuos, siendo ésta la idea que se presenta en esta tesis.

RMSE Media Test	Indep.	Cop. Hor.	Cop. Vert.	Gen. Hor.	Gen. Vert.
autoMPG6	3,172 (5)	3,113 (3)	3,141 (4)	**3,060 (1)**	3,098 (2)
autoMPG8	**2,787 (1)**	2,943 (5)	2,811 (2)	2,837 (3)	2,911 (4)
dee	0,437 (3,5)	0,456 (5)	0,437 (3,5)	0,436 (2)	**0,432 (1)**
diabetes	0,638 (4)	0,657 (5)	0,614 (2)	**0,612 (1)**	0,636 (3)
ele-2	209,852 (4)	214,589 (5)	208,979 (3)	**205,479 (1)**	206,496 (2)
friedman	**2,058 (1)**	2,307 (5)	2,133 (3)	2,087 (2)	2,142 (4)
laser	19,972 (5)	18,587 (2)	**17,758 (1)**	19,188 (4)	19,159 (3)
machineCPU	65,499 (3)	**64,600 (1)**	73,154 (4)	65,349 (2)	75,609 (5)
RNK	3,312 (4)	3,875 (5)	2,812 (2)	**2,000 (1)**	3,000 (3)

Tabla D.1: RMSE Media Test - Alternativas de ejecución de clasificadores ordinales para problemas de regresión

La tabla D.2 muestra los tiempos para cada una de las alternativas, donde se observa que la opción que mejor resultado obtiene en tiempo es la *Copia Horizontal* que es la que peor resultados obtiene utilizando la métrica RMSE. La que mejor resultado en

RMSE obtiene es *Generación Horizontal*. Para comprobar si la diferencia de tiempos entre la que obtiene mejor resultados en RMSE (*Generación Horizontal*) y el resto es significativa se realiza el test de Wilcoxon [132] cuyos resultados se muestran en la tabla D.3, en la cual se observa que no hay diferencias significativas con ninguna de las alternativas de ejecución propuestas. Por ello se elige como mejor opción para implementación es la que obtiene mejores resultados en RMSE.

Tiempos (seg)	Indep.	Cop. Hor.	Cop. Vert.	Gen. Hor.	Gen. Vert.
autoMPG6	972,600 (4)	**735,200 (1)**	894,800 (2)	998,200 (5)	916,000 (3)
autoMPG8	1229,200 (5)	**858,600 (1)**	1101,800 (2)	1150,000 (3)	1178,200 (4)
dee	1133,800 (5)	**789,400 (1)**	1041,000 (2)	1099,400 (4)	1096,800 (3)
diabetes	12,200 (5)	**7,200 (1)**	9,400 (2)	11,400 (4)	11,200 (3)
ele-2	2528,000 (4)	**1770,800 (1)**	2355,800 (2)	2525,400 (3)	2581,600 (5)
friedman	12407,000 (4)	**11854,400 (1)**	12536,800 (5)	12207,800 (2)	12292,000 (3)
laser	1403,000 (4)	**1092,000 (1)**	1292,600 (2)	1490,000 (5)	1396,600 (3)
machineCPU	172,800 (3)	**94,200 (1)**	137,600 (2)	179,000 (4)	183,000 (5)
RNK	4,250 (5)	**1,000 (1)**	2,375 (2)	3,750 (4)	3,625 (3)

Tabla D.2: Tiempos - Alternativas de ejecución de clasificadores ordinales para problemas de regresión

VS	$p_{Wilcoxon}$
Independiente	$\geqslant 0.2$
Copia Horizontal	$\geqslant 0.2$
Copia Vertical	$\geqslant 0.2$
Generación Vertical	$\geqslant 0.2$

Tabla D.3: Resultados obtenidos por el test de Wilcoxon para Tiempos - comparativa de Generación Horizontal con el resto.

ACRÓNIMOS

AA Aprendizaje Automático

AG Algoritmo Genético

AGs Algoritmos Genéticos

AMAE Media del Error Medio Absoluto (Average Mean Absolute Error en Inglés)

ANFIS Adaptive-Network-Based Fuzzy Inference System

AUC Área bajo la curva (Area under curve en Inglés)

CCR Precisión (Correct Classification Rate en Inglés)

FPR Radio de falsos positivos (False Positive Rate en Inglés)

IA Inteligencia Artificial

LD Lógica Difusa

MAE Error Medio Absoluto (Mean Absolute Error en Inglés)

MM Media de las Sensibilidades de todas las clases (Macro-Media en Inglés)

MMAE Máximo del Error Medio Absoluto (Maximal Mean Absolute Error en Inglés)

MSE Error cuadrático medio (Mean Square Error en Inglés)

MZE Error Medio Cero-Uno (Mean Zero-One Error en Inglés)

NSLVOrd NSLV Ordinal

NSLVReg NSLV Regresión

OMAE Error Medio Absoluto Ordinal (Ordinal Mean Absolute Error en Inglés)

RMSE Raíz del Error cuadrático medio (Root Mean Square Error en Inglés)

ROC Característica Operativa del Receptor (Receiver Operating Characteristic en Inglés)

RPM Revoluciones Por Minuto

S Sensibilidad (Sensibility en Inglés)

SBRD Sistema Basado en Reglas Difusas

SBRDs Sistemas Basados en Reglas Difusas

SP Especificidad (Specificity en Inglés)

SVM Máquinas de Vectores Soporte

TPR Radio de verdaderos positivos (True Positive Rate o Recall en Inglés)

TNR Radio de verdaderos negativos (True Negative Rate en Inglés)

BIBLIOGRAFÍA

[1] Agresti, Alan: *Categorical data analysis*. John Wiley & Sons, **2013**.

[2] Aguirre, Eugenio; Gámez, Juan Carlos y González, Antonio: «A multi-agent system based on fuzzy logic applied to RoboCupś environment». *Mathware & soft computing*, **2001**, 8(2), pp. 153–178.

[3] Alcalá, J; Fernández, A; Luengo, J; Derrac, J; García, S; Sánchez, L y Herrera, F: «Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework». *Journal of Multiple-Valued Logic and Soft Computing*, **2010**, 17(2-3), pp. 255–287.

[4] Alcalá, R.; Alcala-Fdez, J.; Casillas, J.; Cordón, O. y Herrera, F.: «Local Identification of Prototypes for Genetic Learning of Accurate TSK Fuzzy Rule-Based Systems.» *International Journal of Intelligent Systems*, **2007**, 22(9), pp. 909–941.

[5] Alcalá, R.; Gacto, M. J. y Herrera, F.: «A Fast and Scalable Multiobjective Genetic Fuzzy System for Linguistic Fuzzy Modeling in High-Dimensional Regression Problems». *IEEE Transactions on Fuzzy Systems*, **2011**, 19(4), pp. 666–681. ISSN 1063-6706. doi: 10.1109/TFUZZ.2011.2131657.

[6] Alcalá-Fdez, J.; Sánchez, L.; García, S.; del Jesus, M.J.; Ventura, S.; Garrell, J.M.; Otero, J.; Romero, C.; Bacardit, J.; Rivas, V.M.; Fernández, J.C. y Herrera, F.: «KEEL: a software tool to assess evolutionary algorithms for data mining problems». *Soft Computing*, **2009**, 13(3), pp. 307–318. ISSN 1432-7643. doi: 10.1007/s00500-008-0323-y. http://dx.doi.org/10.1007/s00500-008-0323-y

[7] Alpaydin, Ethem: *Introduction to Machine Learning*. The MIT Press, 2nd[a] edición, **2010**. ISBN 026201243X, 9780262012430.

[8] Asl, Mohammad Rahmani; Xu, Weili; Shang, Jin; Tsai, Barry y Molloy, Ian: «Regression-based building energy

performance assessment using building information model (BIM)». *IBPSA-USA Journal*, **2016**, 6(1).

[9] Bacardit, Jaume y Krasnogor, Natalio: *Empirical Evaluation of Ensemble Techniques for a Pittsburgh Learning Classifier System*. pp. 255–268. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-88138-4, **2008**. doi: 10.1007/978-3-540-88138-4_15.
http://dx.doi.org/10.1007/978-3-540-88138-4_15

[10] Baccianella, S.; Esuli, A y Sebastiani, F.: «Evaluation Measures for Ordinal Regression». En: *Intelligent Systems Design and Applications, 2009. ISDA '09. Ninth International Conference on*, pp. 283–287, **2009**. doi: 10.1109/ISDA.2009.230.

[11] Berthold, Michael R.; Cebron, Nicolas; Dill, Fabian; Gabriel, Thomas R.; Kötter, Tobias; Meinl, Thorsten; Ohl, Peter; Sieb, Christoph; Thiel, Kilian y Wiswedel, Bernd: «KNIME: The Konstanz Information Miner». En: *Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007)*, Springer. ISBN 978-3-540-78239-1. ISSN 1431-8814, **2007**.

[12] Berthold, Michael R.; Cebron, Nicolas; Dill, Fabian; Gabriel, Thomas R.; Kötter, Tobias; Meinl, Thorsten; Ohl, Peter; Thiel, Kilian y Wiswedel, Bernd: «KNIME - the Konstanz Information Miner: Version 2.0 and Beyond». *SIGKDD Explor. Newsl.*, **2009**, 11(1), pp. 26–31. ISSN 1931-0145. doi: 10.1145/1656274.1656280.
http://doi.acm.org/10.1145/1656274.1656280

[13] Bilmes, Jeff: «A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models». *Informe técnico*, **1998**.

[14] Breiman, L.; Friedman, J.H.; Olshen, R.A. y Stone, C.J.: *Classification and Regression Trees*. Chapman and Hall (Wadsworth and Inc.), **1984**.

[15] Breiman, Leo: «Bagging Predictors». *Machine Learning*, **1996**, 24(2), pp. 123–140. ISSN 1573-0565. doi: 10.1023/A:1018054314350.
http://dx.doi.org/10.1023/A:1018054314350

[16] Broomhead, D.S. y Lowe, D.: «Multivariable Functional Interpolation and Adaptive Networks». *Complex Systems*, **1998**, 11, pp. 321–355.

[17] Caises, Y.; González, A.; Leyva, E. y Pérez, R.: «An Efficient Inductive Genetic Learning Algorithm for Fuzzy Relational Rules». *Por aparcer en: International Journal of Computational Intelligence Systems*, **2011**.

[18] Cardoso, Jaime S; da Costa, Joaquim F Pinto y Cardoso, Maria J: «Modelling ordinal relations with SVMs: An application to objective aesthetic evaluation of breast cancer conservative treatment». *Neural Networks*, **2005**, 18, pp. 808–817.

[19] Cardoso, Jaime S. y Sousa, Ricardo: «Measuring the Performance of Ordinal Classification». *IJPRAI*, **2011**, 25(8), pp. 1173–1195. doi: 10.1142/S0218001411009093. http://dx.doi.org/10.1142/S0218001411009093

[20] Chang, Chih-Chung y Lin, Chih-Jen: «LIBSVM: A library for support vector machines». *ACM Transactions on Intelligent Systems and Technology*, **2011**, 2, pp. 27:1–27:27. http://www.csie.ntu.edu.tw/~cjlin/libsvm

[21] Chu, Wei y Ghahramani, Zoubin: «Gaussian processes for ordinal regression». En: *Journal of Machine Learning Research*, pp. 1019–1041, **2005**.

[22] Chu, Wei y Keerthi, S Sathiya: «New approaches to support vector ordinal regression». En: *Proceedings of the 22nd international conference on Machine learning*, pp. 145–152. ACM, **2005**.

[23] Chu wei, Sathiya Keerthi: «Support vector ordinal regression». *Neural computation*, **2007**, 19(3), pp. 792–815.

[24] Cordón, O. y Herrera, F.: «A Three-Stage Evolutionary Process for Learning Descriptive and Approximate Fuzzy Logic Controller Knowledge Bases from Examples». *International Journal of Approximate Reasoning*, **1997**, 17(4), pp. 369–407.

[25] Cordón, O. y Herrera, F.: «A Two-Stage Evolutionary Process for Designing TSK Fuzzy Rule-Based Systems».

IEEE Transactions on Systems and Man and Cybernetics and Part B: Cybernetics, **1999**, 29(6), pp. 703–715.

[26] Cordón, O. y Herrera, F.: «Hybridizing Genetic Algorithms with Sharing Scheme and Evolution Strategies for Designing Approximate Fuzzy Rule-Based Systems». *Fuzzy Sets and Systems*, **2001**, 118(2), pp. 235–255.

[27] Cortes, Corinna y Mohri, Mehryar: «AUC optimization vs. error rate minimization». En: *in Advances in Neural Information Processing Systems*, MIT Press, **2004**.

[28] Cruz-Ramírez, Manuel: *Redes Neuronales Evolutivas Multiobjetivo para Clasificación Nominal y Ordinal. Aplicaciones - Multiobjetive Evolutionary Neural Networks for Nominal And Ordinal Classification. Applications*. Tesis doctoral, Universidad de Granada, **2013**.

[29] Cruz-Ramírez, Manuel; Hervás-Martínez, César; Fernández, Juan Carlos; Briceño, Javier y de la Mata, Manuel: «Multi-objective evolutionary algorithm for donor-recipient decision system in liver transplants». *European Journal of Operational Research*, **2012**, 222(2), pp. 317 – 327. ISSN 0377-2217. doi: 10.1016/j.ejor.2012.05.013. http://www.sciencedirect.com/science/article/pii/S0377221712003621

[30] Cruz-Ramírez, Manuel; Hervás-Martínez, César; Sánchez-Monedero, Javier y Gutiérrez, Pedro Antonio: «A preliminary study of ordinal metrics to guide a multi-objective evolutionary algorithm». En: *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*, pp. 1176–1181. ISSN 2164-7143, **2011**. doi: 10.1109/ISDA.2011.6121818.

[31] Cruz-Ramírez, Manuel; Hervás-Martínez, César; Sánchez-Monedero, Javier y Gutiérrez, Pedro Antonio: «Metrics to guide a multi-objective evolutionary algorithm for ordinal classification». *Neurocomputing*, **2014**, 135, pp. 21–31.

[32] Cuesta, J.M.L.; Prior, R.A.; Martínez, J.M.M.; Martínez, F.F.; Barra-Chicote, R.; D'Haro, L.F.; Enríquez, J.F.L.; de Córdoba Herralde, R.; Macías, J.; Guarasa, R.S.S.H. y

otros: «Desarrollo de un Robot-Guía con Integración de un Sistema de Diálogo y Expresión de Emociones: Proyecto ROBINT Development of a Tour-Providing Robot Integrating Dialogue System and Emotional Speech: ROBINT Project». *Impresos de Inscripción*, **2008**, 40, pp. 51–58.

[33] DA COSTA, JOAQUIM F PINTO; ALONSO, HUGO Y CARDOSO, JAIME S: «The unimodal model for the classification of ordinal data». *Neural Networks*, **2008**, 21, pp. 78–91.

[34] DANTONE, M.; GALL, J.; FANELLI, G. Y VAN GOOL, L.: «Real-time facial feature detection using conditional regression forests». CVPR, **2012**.

[35] DAVID G. KLEINBAUM, MITCHEL KLEIN: *Logistic Regression. A Self-Learning Text*. Springer New York, third[a] edición, **2010**. doi: 10.1007/978-1-4419-1742-3.

[36] DAVID GARCÌA, ANTONIO GONZÀLEZ RAÚL PÉREZ JOHN B. THEOCHARIS, DIMITRIS STAVRAKOUDIS: «A Fuzzy Rule-Based Feature Construction Approach Applied to Remotely Sensed Imagery», **2015**. doi: doi:10.2991/ifsa-eusflat-15.2015.180.

[37] DEMBCZYŃSKI, KRZYSZTOF Y KOTŁOWSKI, WOJCIECH: «Decision rule-based algorithm for ordinal classification based on rank loss minimization». En: *Preference learning, ECML/PKDD workshop*, , **2009**.

[38] DEMBCZYŃSKI, KRZYSZTOF; KOTŁOWSKI, WOJCIECH Y SŁOWIŃSKI, ROMAN: «Ordinal Classification with Decision Rules». En: ZbigniewW. Ras; Shusaku Tsumoto y Djamel Zighed (Eds.), *Mining Complex Data*, volumen 4944 de *Lecture Notes in Computer Science*, pp. 169–181. Springer Berlin Heidelberg, **2008**. ISBN 978-3-540-68415-2. doi: 10.1007/978-3-540-68416-9_14.
http://dx.doi.org/10.1007/978-3-540-68416-9_14

[39] DESTERCKE, SÉBASTIEN; GUILLAUME, SERGE Y CHARNOMORDIC, BRIGITTE: «Building an interpretable fuzzy rule base from data using Orthogonal Least Squares - Application to a depollution problem». *Fuzzy Sets and Systems*, **2007**, 158(18), pp. 2078 – 2094. ISSN 0165-0114. doi: http://dx.doi.org/10.1016/j.fss.2007.04.026.

http://www.sciencedirect.com/science/article/pii/S016501140700200X

[40] Elkano, Mikel; Galar, Mikel; Sanz, José y Bustince, Humberto: «Optimización del Sistema de Clasificación Basado en Reglas Difusas Chi-FRBCS-BigDataCS». En: José Miguel Puerta; José A. Gámez; Bernabé Dorronsoro; Edurne Barrenechea; Alicia Troncoso; Bruno Baruque y Mikel Galar (Eds.), *Actas de la XVI Conferencia de la Asociación Española para la Inteligencia Artificial, CAEPIA 2015, Albacete, Noviembre 9-12, 2015*, ISBN 978-84-608-4099-2, **2015**. http://simd.albacete.org/actascaepia15/

[41] Fan, R.E.; Chen, P.H. y Lin, C.J.: «Working set selection using the second order information for training SVM». *Journal of Machine Learning Research*, **2005**, 6, pp. 1889–1918.

[42] Fanelli, G.; Dantone, M.; Fossati, A.; Gall, J. y Gool, L. Van: «Random Forests for Real Time 3D Face Analysis». *International Journal of Coputer Vision (IJCV)*, **2012**.

[43] Fang, Tingting y Lahdelma, Risto: «Evaluation of a multiple linear regression model and {SARIMA} model in forecasting heat demand for district heating system». *Applied Energy*, **2016**, 179, pp. 544 – 552. ISSN 0306-2619. doi: http://dx.doi.org/10.1016/j.apenergy.2016.06.133. http://www.sciencedirect.com/science/article/pii/S0306261916309217

[44] Frank, Eibe y Hall, Mark: «A Simple Approach to Ordinal Classification». *ECML 2001*, **2001**, pp. 145–157.

[45] Freund, Y. y Schapire, R.: «A desicion-theoretic generalization of on-line learning and an application to boosting». En: *Computational learning theory*, pp. 23–37. Springer, **1995**.

[46] Freund, Yoav; Iyer, Raj; Schapire, Robert E y Singer, Yoram: «An efficient boosting algorithm for combining preferences». *The Journal of machine learning research*, **2003**, 4, pp. 933–969.

[47] Gacto, M.J.; Galende, M.; Alcalá, R. y Herrera, F.: «METSK-HDe: A multiobjective evolutionary algorithm to learn accurate TSK-fuzzy systems in high-dimensional

and large-scale regression problems». *Information Sciences*, **2014**, 276, pp. 63 – 79. ISSN 0020-0255. doi: http://dx.doi.org/10.1016/j.ins.2014.02.047. http://www.sciencedirect.com/science/article/pii/S0020025514001534

[48] Galar, M.; Fernandez, A.; Barrenechea, E.; Bustince, H. y Herrera, F.: «A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches». *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, **2012**, 42(4), pp. 463–484. ISSN 1094-6977. doi: 10.1109/TSMCC.2011.2161285.

[49] Gámez, J. C.; García, D.; González, A. y Pérez, R.: «On the Use of an Incremental Approach to Learn Fuzzy Classification Rules for Big Data Problems». En: *Fuzzy Systems (FUZZ-IEEE), 2016 IEEE International Conference on*, , **2016**.

[50] Gámez, J.C.; González, A. y Pérez, R.: «Un modelo difuso para la estimación de la atención en procesos de interacción robot-persona». *XI Workshop of Physical Agents 2010*, **2010**, pp. 169–176.

[51] Gámez, J.C.; González, A.; Pérez, R. y M., García-Silvente: «Uso de técnicas de preprocesamiento de imágenes y aprendizaje para la detección de cambios de atención de una persona en procesos de interacción persona-robot». En: *Actas ROBOT 2011*, pp. 389–396, **2011**.

[52] Gámez, Juan Carlos; García, David; González, Antonio y Pérez, Raúl.: «Un algoritmo de clasificación ordinal basado en el modelo de recubrimiento secuencial». En: *Actas de la XVI Conferencia de la Asociación Española para la Inteligencia Artificial*, pp. 529–538, **2015**. http://simd.albacete.org/actascaepia15/

[53] Gámez, Juan Carlos; García, David; González, Antonio y Pérez, Raúl: «Ordinal Classification based on the Sequential Covering Strategy». *International Journal of Approximate Reasoning*, **2016**, 76, pp. 96 – 110. ISSN 0888-613X. doi: http://dx.doi.org/10.1016/j.ijar.2016.05.002. http://www.sciencedirect.com/science/article/pii/S0888613X16300706

[54] Gámez, Juan Carlos; García, David; González, Antonio y Pérez, Raúl.: «Un estudio preliminar sobre el uso de técnicas de aprendizaje incremental para problemas de big data». En: *Libro de Resúmenes del XVIII Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF 2016)*, pp. 176–177, **2016**. ISBN: 978-84-608-7381-5.

[55] García, D.; Gámez, J. C.; González, A. y Pérez, R.: «Using a sequential covering strategy for discovering fuzzy rules incrementally». En: *Fuzzy Systems (FUZZ-IEEE), 2015 IEEE International Conference on*, pp. 1–8, **2015**. doi: 10.1109/FUZZ-IEEE.2015.7337924.

[56] García, David: *Extensions of the Genetic Iterative Approach for Learning Fuzzy Rules*. Tesis doctoral, Universidad de Granada, **2015**.

[57] García, David; Gámez, Juan Carlos; González, Antonio y Pérez, Raúl: «An interpretability improvement for fuzzy rule bases obtained by the iterative rule learning approach». *International Journal of Approximate Reasoning*, **2015**, 67, pp. 37 – 58. ISSN 0888-613X. doi: 10.1016/j.ijar.2015.09.001. http://www.sciencedirect.com/science/article/pii/S0888613X15001425

[58] García, David; González, Antonio y Pérez, Raúl: «Overview of the SLAVE learning algorithm: A review of its evolution and prospects». *International Journal of Computational Intelligence Systems*, **2014**, 7(6), pp. 1194–1221. doi: 10.1080/18756891.2014.967008. http://dx.doi.org/10.1080/18756891.2014.967008

[59] García-Pedrajas, N.; García-Osorio, C. y Fyfe, C.: «Nonlinear Boosting Projections for Ensemble Construction». *Journal of Machine Learning Research*, **2007**, 8, pp. 1–33.

[60] Geetha, A.; Ramalingam, V.; Palanivel, S. y Palaniappan, B.: «Facial expression recognition-A real time approach». *Expert Systems with Applications*, **2009**, 36(1), pp. 303–308.

[61] Gizatdinova, Yulia y Surakka, Veikko: «Automatic edge-based localization of facial features from images with complex facial expressions». *Pattern Recognition*

Letters, **2010**, 31(15), pp. 2436 – 2446. ISSN 0167-8655. doi: 10.1016/j.patrec.2010.07.020.
http://www.sciencedirect.com/science/article/B6V15-50PCM4T-2/2/0baa56ef42a4dcf0f6b47827df0afb99

[62] González, A. y Pérez, R.: «SLAVE: A genetic learning system based on an iterative approach». *IEEE Transactions on Fuzzy Systems*, **1999**, 7(2), pp. 176–191.

[63] González, Antonio y Herrera, Francisco: «Multi-stage genetic fuzzy systems based on the iterative rule learning approach». *Mathware & soft computing. 1997 Vol. 4 Núm. 3*, **1997**.

[64] González, Antonio y Pérez, Raul: «Completeness and consistency conditions for learning fuzzy rules». *Fuzzy Sets and Systems*, **1998**, 96(1), pp. 37–51.

[65] González, Antonio y Pérez, Raúl: «Improving the genetic algorithm of SLAVE». *Mathware & Soft Computing*, **2009**, 16(1), pp. 59–70.

[66] Gu, Fangqing; Liu, Hai-Lin y Tan, Kay Chen: «A hybrid evolutionary multiobjective optimization algorithm with adaptive multi-fitness assignment». *Soft Computing*, **2015**, 19(11), pp. 3249–3259. ISSN 1433-7479. doi: 10.1007/s00500-014-1480-9.
http://dx.doi.org/10.1007/s00500-014-1480-9

[67] Gutiérrez, Pedro Antonio; Pérez-Ortiz, M; Fernández-Navarro, Francisco; Sánchez-Monedero, Javier y Hervás-Martínez, César: «An experimental study of different ordinal regression methods and measures». En: *Hybrid Artificial Intelligent Systems*, pp. 296–307. Springer, **2012**.

[68] Gutiérrez, Pedro Antonio; Pérez-Ortiz, María; Sánchez-Monedero, Javier y Hervás-Martínez, César: «Estudio comparativo de distintos métodos de umbral en regresión ordinal», **2013**.

[69] Gutiérrez, Pedro Antonio; Salcedo-Sanz, Sancho; Hervás-Martínez, César; Carro-Calvo, Leopoldo; Sánchez-Monedero, Javier y Prieto, Luis: «Ordinal and

nominal classification of wind speed from synoptic pressure patterns». *Engineering Applications of Artificial Intelligence*, **2013**, 26(3), pp. 1008–1015.

[70] Gutiérrez, Pedro Antonio; Tiňo, Peter y Hervás-Martínez, César: «Ordinal regression neural networks based on concentric hyperspheres». *Neural Networks*, **2014**, 59(0), pp. 51 – 60. ISSN 0893-6080. doi: 10.1016/j.neunet.2014.07.001.
http://www.sciencedirect.com/science/article/pii/S0893608014001580

[71] Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P. y Witten, I.H.: «The WEKA data mining software: an update». *ACM SIGKDD Explorations Newsletter*, **2009**, 11(1), pp. 10–18.

[72] Harrell, FE; Lee, KL; Califf, RM; Pryor, DB y Rosati, RA: «Regression modelling strategies for improved prognostic prediction». *Statistics in medicine*, **1984**, 3(2), pp. 143–152. ISSN 0277-6715. doi: 10.1002/sim.4780030207.
http://dx.doi.org/10.1002/sim.4780030207

[73] Harrell Jr, Frank E; Lee, Kerry L; Matchar, David B y Reichert, Thomas A: «Regression models for prognostic prediction: advantages, problems, and suggested solutions.» *Cancer treatment reports*, **1985**, 69(10), pp. 1071–1077.

[74] Herbrich, Ralf; Graepel, Thore y Obermayer, Klaus: «Large margin rank boundaries for ordinal regression». *Advances in neural information processing systems*, **1999**, pp. 115–132.

[75] Hernández, Moisés; Rodríguez, Tinguaro y Montero, Javier.: «Credit ranting using fuzzy algorithms». En: *Actas de la XVI Conferencia de la Asociación Española para la Inteligencia Artificial*, pp. 539–548, **2015**.
http://simd.albacete.org/actascaepia15/

[76] Holland, John H.: «Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems». En: R. S. Michalski; J. G. Carbonell y T. M. Mitchell (Eds.), *Machine Learning, an artificial intelligence approach: Volume II*, Morgan Kaufmann, Los Alamos, CA, **1986**.

[77] HOLMES, G.; HALL, M. y FRANK, E.: «Generating Rule Sets from Model Trees». En: *Proceedings of the 12th Australian Joint Conference on Artificial Intelligence: Advanced Topics in Artificial Intelligence*, volumen 1747 de *Lecture Notes on Computer Science*, pp. 1–12. Springer-Verlag, **1999**.

[78] HOMAIFAR, A. y MCCORMICK, E.: «Simultaneous Design of Membership Functions and Rule Sets for Fuzzy Controllers Using Genetic Algorithms». *IEEE Transactions on Fuzzy Systems*, **1995**, 3(2), pp. 129–139.

[79] HSU, CHIH-WEI y LIN, CHIH-JEN: «A comparison of methods for multiclass support vector machines». *Neural Networks, IEEE Transactions on*, **2002**, 13(2), pp. 415–425.

[80] HUHN, JENS C y HULLERMEIER, EYKE: «Is an ordinal class structure useful in classifier learning?» *International Journal of Data Mining, Modelling and Management*, **2008**, 1(1), pp. 45–67.

[81] IGEL, CHRISTIAN y HÜSKEN, MICHAEL: «Empirical evaluation of the improved Rprop learning algorithms». *Neurocomputing*, **2003**, 50, pp. 105–123.

[82] JANG, J.-S.R.: «ANFIS: adaptive-network-based fuzzy inference system». *IEEE Transactions on Systems, Man, and Cybernetics*, **1993**, 23(3), pp. 665–685. ISSN 00189472. doi: 10.1109/21.256541.
http://ieeexplore.ieee.org/document/256541/

[83] JIN, YAOCHU: «Fuzzy modeling of high-dimensional systems: complexity reduction and interpretability improvement». *IEEE Transactions on Fuzzy Systems*, **2000**, 8(2), pp. 212–221. ISSN 1063-6706. doi: 10.1109/91.842154.

[84] JOACHIMS, THORSTEN: «Optimizing Search Engines Using Clickthrough Data». En: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pp. 133–142. ACM, New York, NY, USA. ISBN 1-58113-567-X, **2002**. doi: 10.1145/775047.775067.
http://doi.acm.org/10.1145/775047.775067

[85] KONONENKO, IGOR: «Machine learning for medical diagnosis: history, state of the art and perspective». *Artificial Intelligence in Medicine*, **2001**, 23(1), pp. 89 – 109. ISSN 0933-3657.

doi: http://dx.doi.org/10.1016/S0933-3657(01)00077-X. http://www.sciencedirect.com/science/article/pii/S093336570100077X

[86] Kullback, Solomon: *Information theory and statistics*. Courier Dover Publications, **1997**.

[87] Kuo, Tzu-Ming; Lee, Ching-Pei y Lin, Chih-Jen: *Large-scale Kernel RankSVM*. capítulo 91, pp. 812–820, **2014**. doi: 10.1137/1.9781611973440.93. http://epubs.siam.org/doi/abs/10.1137/1.9781611973440.93

[88] Li, Ling y Lin, Hsuan-Tien: «Ordinal regression by extended binary classification». En: *Advances in neural information processing systems*, pp. 865–872, **2006**.

[89] Lim, Tjen-Sien; Loh, Wei-Yin y Shih, Yu-Shan: «A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms». *Machine learning*, **2000**, 40(3), pp. 203–228.

[90] Lin, H. T. y Li, L.: «Reduction from Cost-Sensitive Ordinal Ranking to Weighted Binary Classification». volumen 24, pp. 1329–1367, **2012**. doi: 10.1162/NECO_a_00265.

[91] Lin, Hsuan-Tien y Li, Ling: «Large-Margin Thresholded Ensembles for Ordinal Regression: Theory and Practice». En: JoséL. Balcázar; PhilipM. Long y Frank Stephan (Eds.), *Algorithmic Learning Theory*, volumen 4264 de *Lecture Notes in Computer Science*, pp. 319–333. Springer Berlin Heidelberg, **2006**. ISBN 978-3-540-46649-9. doi: 10.1007/11894841_26. http://dx.doi.org/10.1007/11894841_26

[92] Liu, Ivy y Agresti, Alan: «The analysis of ordered categorical data: An overview and a survey of recent developments». *Test*, **2005**, 14(1), pp. 1–73.

[93] López, Victoria; del Río, Sara; Benítez, José Manuel y Herrera, Francisco: «Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data». *Fuzzy Sets and Systems*, **2015**, 258, pp. 5 – 38. ISSN 0165-0114. doi: http://dx.doi.org/10.1016/j.fss.2014.01.015. Special issue:

Uncertainty in Learning from Big Data.
http://www.sciencedirect.com/science/article/pii/S0165011414000566

[94] MAES, SAM; TUYLS, KARL; VANSCHOENWINKEL, BRAM y MANDERICK, BERNARD: «Credit card fraud detection using Bayesian and neural networks». En: *Proceedings of the 1st international naiso congress on neuro fuzzy technologies*, pp. 261–270, **2002**.

[95] MAMDANI, E. H.: «Application of fuzzy algorithms for control of simple dynamic plant». *Electrical Engineers, Proceedings of the Institution of*, **1974**, 121(12), pp. 1585–1588. ISSN 0020-3270. doi: 10.1049/piee.1974.0328.

[96] MARCHIORI, E.: «Class Conditional Nearest Neighbor for Large Margin Instance Selection». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **2010**, 32(2), pp. 364–370. ISSN 0162-8828. doi: 10.1109/TPAMI.2009.164.

[97] MARTÍNEZ-ESTUDILLO, A.C.; MARTÍNEZ-ESTUDILLO, F.J.; HERVÁS-MARTÍNEZ, C. y GARCÍA, N.: «Evolutionary Product Unit based Neural Networks for Regression». *Neural Networks*, **2006**, 19(4), pp. 477–486.

[98] MCCALLUM, ANDREW; NIGAM, KAMAL; RENNIE, JASON y SEYMORE, KRISTIE: «A machine learning approach to building domain-specific search engines». Citeseer, **1999**.

[99] MCCULLAGH, PETER: «Regression models for ordinal data». *Journal of the royal statistical society. Series B (Methodological)*, **1980**, pp. 109–142.

[100] MITCHELL, T.M.: *Machine Learning*. McGraw-Hill international editions - computer science series. McGraw-Hill Education, **1997**. ISBN 9780070428072.
https://books.google.es/books?id=xOGAngEACAAJ

[101] MONTAÑÉS, ELENA; SUÁREZ-VÁZQUEZ, ANA y QUEVEDO, JOSÉ RAMÓN: «Ordinal classification/regression for analyzing the influence of superstars on spectators in cinema marketing». *Expert Systems with Applications*, **2014**, 41(18), pp. 8101 – 8111. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2014.07.011.

http://www.sciencedirect.com/science/article/pii/S0957417414004096

[102] Muñoz-Salinas, R.; Aguirre, E. y García-Silvente, M.: «Detection of doors using a genetic visual fuzzy system for mobile robots». *Autonomous Robots*, **2006**, 21(2), pp. 123–141.

[103] Pérez, A.G.R.: «Una mejora del modelo genético iterativo de SLAVE». *ESTYLFo8,*, **2008**.

[104] Pérez, Francisco G. Raúl: *Aprendizaje de Reglas Difusas usando algoritmos genéticos*. Tesis doctoral, Universidad de Granada, **1997**.

[105] Pérez-Ortiz, María; Cruz-Ramírez, Manuel; Ayllón-Terán, María Dolores; Heaton, Nigel; Ciria, Rubén y Hervás-Martínez, César: «An organ allocation system for liver transplantation based on ordinal regression». *Applied Soft Computing*, **2014**, 14, Part A(0), pp. 88 – 98. ISSN 1568-4946. doi: 10.1016/j.asoc.2013.07.017. Special issue on hybrid intelligent methods for health technologies. http://www.sciencedirect.com/science/article/pii/S1568494613002639

[106] Pérez-Ortiz, María; de la Paz-Marín, Mónica; Gutiérrez, Pedro Antonio y Hervás-Martínez, César: «Classification of {EU} countriesprogress towards sustainable development based on ordinal regression techniques». *Knowledge-Based Systems*, **2014**, 66(0), pp. 178 – 189. ISSN 0950-7051. doi: 10.1016/j.knosys.2014.04.041. http://www.sciencedirect.com/science/article/pii/S0950705114001713

[107] Plat, J.: «A Resource Allocating Network for Function Interpolation». *Neural Computation*, **1991**, 3(2), pp. 213–225.

[108] Ponce Cruz, Pedro: *Inteligencia Artificial con aplicaciones a la Ingeniería*. 1ª edición, **2011**. https://lelinopontes.files.wordpress.com/2014/09/inteligencia-artificial-con-aplicaciones-a-la-ingenierc3ada.pdf

[109] Quinlan, J.R.: «Learning with continuous classes». En: *5th Australian joint conference on artificial intelligence*, pp. 343–348. Citeseer, **1992**.

[110] Rivera, AJ; Pérez-Godoy, MD; Pulgar, F y del Jesus, MJ: «GenRBFNSpark: A first implementation in Spark of a genetic algorithm to RBFN design». En: José Miguel Puerta; José A. Gámez; Bernabé Dorronsoro; Edurne Barrenechea; Alicia Troncoso; Bruno Baruque y Mikel Galar (Eds.), *Actas de la XVI Conferencia de la Asociación Española para la Inteligencia Artificial, CAEPIA 2015, Albacete, Noviembre 9-12, 2015*, ISBN 978-84-608-4099-2, **2015**.
http://simd.albacete.org/actascaepia15/

[111] Rodríguez, Miguel Ángel y Peregrín, Antonio: «Big Data y Algoritmos Genéticos Distribuidos en el aprendizaje de reglas de clasificación». En: José Miguel Puerta; José A. Gámez; Bernabé Dorronsoro; Edurne Barrenechea; Alicia Troncoso; Bruno Baruque y Mikel Galar (Eds.), *Actas de la XVI Conferencia de la Asociación Española para la Inteligencia Artificial, CAEPIA 2015, Albacete, Noviembre 9-12, 2015*, ISBN 978-84-608-4099-2, **2015**.
http://simd.albacete.org/actascaepia15/

[112] Rodríguez-Fdez, I.; Mucientes, M. y Bugarín, A.: «An instance selection algorithm for regression and its application in variance reduction». En: *Fuzzy Systems (FUZZ), 2013 IEEE International Conference on*, pp. 1–8. ISSN 1098-7584, **2013**. doi: 10.1109/FUZZ-IEEE.2013.6622486.

[113] Rodríguez-Fdez, I.; Mucientes, M. y Bugarín, A.: «Reducing the complexity in genetic learning of accurate regression TSK rule-based systems». En: *Fuzzy Systems (FUZZ-IEEE), 2015 IEEE International Conference on*, pp. 1–8, **2015**. doi: 10.1109/FUZZ-IEEE.2015.7337930.

[114] Rojas, R. y Feldman, J.: *Neural Networks: A Systematic Introduction*. Springer-Verlag and Berlin and New-York, **1996**.

[115] Russell, Stuart y Norvig, Peter: *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd[a] edición, **2009**. ISBN 0136042597, 9780136042594.
http://stpk.cs.rtu.lv/sites/all/files/stpk/materiali/MI/Artificial%20Intelligence%20A%20Modern%20Approach.pdf

[116] Rustagi, J.S.: *Optimization Techniques in Statistics*. Academic Press, **1994**.

[117] Sánchez, L.: «A Random Sets-Based Method for Identifying Fuzzy Models». *Fuzzy Sets and Systems*, **1998**, 98(3), pp. 343–354.

[118] Sánchez, L. y Couso, I.: «Fuzzy Random Variables-Based Modeling with GA-P Algorithms», **2000**, pp. 245–256.

[119] Sánchez, L.; Couso, I. y Corrales, J.A.: «Combining GP Operators with SA Search to Evolve Fuzzy Rule Based Classifiers». *Information Sciences*, **2001**, 136(1-4), pp. 175–191.

[120] Sànchez-Monedero, J.; Campoy-Muñoz, Pilar; Gutiérrez, P.A. y Hervás-Martínez, C.: «A guided data projection technique for classification of sovereign ratings: The case of European Union 27». *Applied Soft Computing*, **2014**, 22(0), pp. 339 – 350. ISSN 1568-4946. doi: 10.1016/j.asoc.2014.05.008.
http://www.sciencedirect.com/science/article/pii/S1568494614002348

[121] Sánchez-Monedero, Javier: *Retos en clasificación ordinal: redes neuronales artificiales y métodos basados en proyecciones. Challenges in ordinal classification: artificial neural net- works and projection-based methods*. Tesis doctoral, Universidad de Granada, **2013**.
http://www.uco.es/ayrna/publications/thesis/ThesisDissertationJSM.pdf

[122] Shaffer, Juliet Popper: «Modified Sequentially Rejective Multiple Test Procedures». *Journal of the American Statistical Association*, **1986**, 81(395), p. 826.

[123] Smith, Stephen Frederick: *A Learning System Based on Genetic Adaptive Algorithms*. Tesis doctoral, Pittsburgh, PA, USA, **1980**. AAI8112638.

[124] Sun, Bing-Yu; Li, Jiuyong; Wu, D.D.; Zhang, Xiao-Ming y Li, Wen-Bo: «Kernel Discriminant Learning for Ordinal Regression». *Knowledge and Data Engineering, IEEE Transactions on*, **2010**, 22(6), pp. 906–910. ISSN 1041-4347. doi: 10.1109/TKDE.2009.170.

[125] TABASI, SANAZ; ASLANI, ALIREZA y FOROTAN, HABIB: «Prediction of Energy Consumption by Using Regression Model». *Computational Research Progress in Applied Science & Engineering Â©PEARL publication CRPASE*, **2016**, 02(03), pp. 110–115. ISSN 2423-4591.

[126] TAKAGI, T. y SUGENO, M.: «Fuzzy identification of systems and its applications to modeling and control». *IEEE Transactions on Systems, Man, and Cybernetics*, **1985**, SMC-15(1), pp. 116–132. ISSN 0018-9472. doi: 10.1109/TSMC.1985.6313399.

[127] THRIFT, P.: «Fuzzy logic synthesis with genetic algorithms». En: *Proceedings of the Fourth International Conference on Genetic Algorithms(ICGA91)*, pp. 509–513, **1991**.

[128] VIEIRA, JOSE; DIAS, F MORGADO y MOTA, ALEXANDRE: «Neuro-fuzzy systems: a survey». En: *5th WSEAS NNA International Conference*, , **2004**.

[129] WANG, I. y WITTEN, I.H.: «Inducing model trees for predicting continuous classes». En: *9th European Conference on Machine Learning*, pp. 128–137, **1997**.

[130] WANG, J.H.; YU, Y.W. y TSAI, J.H.: «On the internal representations of product units». *Neural Processing Letters*, **2000**, 12(3), pp. 247–254.

[131] WANG, L-X y MENDEL, JERRY M: «Generating fuzzy rules by learning from examples». *IEEE Transactions on systems, man, and cybernetics*, **1992**, 22(6), pp. 1414–1427.

[132] WILCOXON, FRANK: «Individual comparisons by ranking methods». *Biometrics bulletin*, **1945**, pp. 80–83.

[133] WU, YAN; JIN, YAOCHU y LIU, XIAOXIONG: «A directed search strategy for evolutionary dynamic multiobjective optimization». *Soft Computing*, **2015**, 19(11), pp. 3221–3235. ISSN 1433-7479. doi: 10.1007/s00500-014-1477-4. http://dx.doi.org/10.1007/s00500-014-1477-4

[134] YANG, L.; CHAO, F. y SHEN, Q.: «Generalised Adaptive Fuzzy Rule Interpolation». *IEEE Transactions on Fuzzy Systems*, **2016**, PP(99), pp. 1–1. ISSN 1063-6706. doi: 10.1109/TFUZZ.2016.2582526.

[135] YANG, LONGZHI y SHEN, QIANG: «Closed form fuzzy interpolation». *Fuzzy Sets and Systems*, **2013**, 225, pp. 1 – 22. ISSN 0165-0114. doi: http://dx.doi.org/10.1016/j.fss.2013.04.001. http://www.sciencedirect.com/science/article/pii/S0165011413001486

[136] ZADEH, L.A.: «The concept of a linguistic variable and its application to approximate reasoning (I)». *Information Sciences*, **1975**, 8(3), pp. 199 – 249. ISSN 0020-0255. doi: 10.1016/0020-0255(75)90036-5. http://www.sciencedirect.com/science/article/pii/0020025575900365

[137] ZADEH, L.A.: «The concept of a linguistic variable and its application to approximate reasoning (II)». *Information Sciences*, **1975**, 8(4), pp. 301 – 357. ISSN 0020-0255. doi: 10.1016/0020-0255(75)90046-8. http://www.sciencedirect.com/science/article/pii/0020025575900468

[138] ZADEH, L.A.: «The concept of a linguistic variable and its application to approximate reasoning (III)». *Information Sciences*, **1975**, 9(1), pp. 43 – 80. ISSN 0020-0255. doi: 10.1016/0020-0255(75)90017-1. http://www.sciencedirect.com/science/article/pii/0020025575900171

Y así, del mucho leer y del poco dormir, se le secó el celebro de manera que vino a perder el juicio.
Miguel de Cervantes [1]

1 Miguel de Cervantes Saavedra (1547 - 1616) soldado, novelista, poeta y dramaturgo español. Máxima figura de la literatura española y universalmente conocido por su obra Don Quijote de la Mancha.