UNIVERSIDAD DE JAÉN

ESCUELA POLITÉCNICA SUPERIOR DE JAÉN
DEPARTAMENTO DE INFORMÁTICA

TESIS DOCTORAL

MODELOS HÍBRIDOS DE APRENDIZAJE BASADOS EN INSTANCIAS Y REGLAS PARA CLASIFICACIÓN MONOTÓNICA

PRESENTADA POR:
JAVIER GARCÍA FERNÁNDEZ

DIRIGIDA POR:
DR. D. JOSÉ RAMÓN CANO DE AMO
DR. D. SALVADOR GARCÍA LÓPEZ

JAÉN, 8 DE FEBRERO DE 2017

ISBN 978-84-9159-070-5

Universidad de Jaén

Modelos híbridos de aprendizaje basados en instancias y reglas para Clasificación Monotónica

MEMORIA QUE PRESENTA

Javier García Fernández

PARA OPTAR AL GRADO DE DOCTOR EN INFORMÁTICA

Diciembre de 2016

DIRECTORES

José Ramón Cano de Amo y Salvador García López

Departamento de Informática

La memoria titulada "*Modelos híbridos de aprendizaje basados en instancias y reglas de Clasificación Monotónica*", que presenta D. Javier García Fernández para optar al grado de doctor, ha sido realizada dentro del programa de doctorado "*Tecnologías de la Información y la Comunicación* " del Departamento de Informática de la Universidad de Jaén bajo la dirección de los doctores D. José Ramón Cano de Amo y D. Salvador García López.

Jaén, Diciembre de 2016

El Doctorando Los Directores

Fdo: Javier García Fernández Fdo: J.R. Cano de Amo y S. García López

Agradecimientos

GRACIAS A TODOS

Índice

Índice de figuras

Introducción

El presente capítulo constituye una introducción a la memoria de tesis doctoral titulada: Modelos híbridos de aprendizaje basados en instancias y reglas para Clasificación Monotónica. El capítulo comienza con una breve introducción al área de investigación en la que se centra esta memoria, y la motivación para la investigación realizada. A continuación, se exponen los objetivos fijados para llevar a cabo dicha investigación, seguidos de la estructura que se seguirá en el resto de la memoria.

A Motivación

Es conocido que los datos por sí solos no producen beneficio directo. Su verdadero valor radica en la posibilidad de extraer información útil para la toma de decisiones o la exploración y comprensión del fenómeno que produjo los datos. Tradicionalmente en la mayoría de los dominios este análisis de datos se hacía mediante un proceso semiautomático: uno o más analistas con conocimiento de los datos y con la ayuda de técnicas estadísticas proporcionaban resúmenes, generaban informes, validaban modelos, etc. Por ello, surge la necesidad de plantear metodologías para el análisis inteligente de datos que permitan descubrir un conocimiento útil a partir de estos. Este es el concepto de proceso de KDD (descubrimiento de conocimiento en bases de datos; en inglés, Knowledge Discovery in Databases), que puede ser definido como el proceso no trivial de identificar patrones en los datos con las características siguientes: válidos, novedosos, útiles y comprensibles. El proceso de KDD, reflejado en la Figura 1, es un conjunto de pasos interactivos e iterativos, entre los que se incluye el preprocesado de los

datos para corregir los posibles datos erróneos, incompletos o inconsistentes, la reducción del número de registros o características para encontrar los más representativos, la búsqueda de patrones de interés con una representación particular y la interpretación de estos patrones incluso de una forma visual. El descubrimiento de conocimiento en bases de datos combina las técnicas tradicionales de extracción de conocimiento con numerosos recursos desarrollados en el área de la inteligencia artificial. En estas aplicaciones, el término Minería de Datos (MD) es el que ha tenido más aceptación, siendo utilizado con frecuencia para reflejar directamente todo el proceso de KDD [HK11, Agg15]. Relacionado con MD, el término aprendizaje automático (Machine Learning, ML, en inglés) sigue siendo muy extendido entre la comunidad científica en lo que se refiere al diseño de nuevos algoritmos [SSBD14].

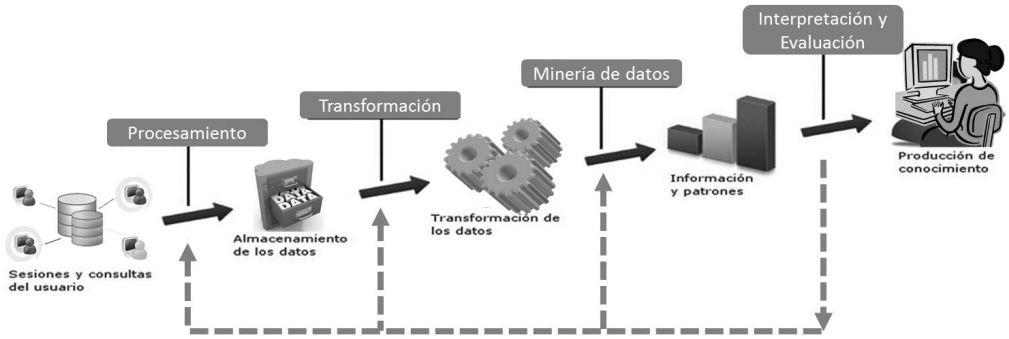


Figura 1: Etapas en el proceso de KDD

El uso de algoritmos de aprendizaje automático presenta dos vertientes: pueden ser empleados simplemente como cajas negras, obteniéndose como resultado tan solo las salidas de los modelos. Sin embargo, algunos algoritmos pueden ser empleados como herramientas de representación de conocimiento, construyendo una estructura simbólica de conocimiento dispuesta a ser útil desde el punto de vista de la funcionalidad, pero también desde la perspectiva de la interpretabilidad. Dependiendo de sus objetivos, los algoritmos de aprendizaje automático pueden ser clasificados en dos áreas diferentes:

- **Aprendizaje Supervisado:** son algoritmos centrados en emplear un conjunto de ejemplos etiquetados, describiendo información de varias variables de entrada, para predecir el valor de varias variables de salida. La categorías más comunes de algoritmos de aprendizaje supervisado son clasificación (donde la variable a predecir es discreta; por ejemplo rojo, verde, azul) y regresión (donde la variable a predecir es continua; por ejemplo: temperatura, peso, ..., etc).

- **Aprendizaje No Supervisado:** son algoritmos centrados en buscar nuevos patrones y relaciones en datos no etiquetados. La categorías más comunes de algoritmos de aprendizaje no supervisado son agrupamiento (el proceso de separar datos en varios grupos, manteniendo ejemplos de los mismos grupos tan similares entre sí como sea posible) y asociación (la identificación de relaciones en datos transaccionales).

Los métodos de clasificación se pueden definir como técnicas que permiten aprender como categorizar elementos dentro de varias clases predefinidas. Un clasificador recibe un conjunto de datos como entrada, definido como *conjunto de entrenamiento*, y aprende un modelo de clasificación con él. En el proceso de validación del clasificador, se emplea un conjunto adicional de ejemplos, no contemplados durante el proceso de aprendizaje (el *conjunto de test*) para comprobar la precisión del clasificador.

En la literatura, existen muchas propuestas diferentes para la realización de tareas de clasificación, incluyendo técnicas estadísticas, funciones discriminantes, redes neuronales, árboles de decisión, máquinas de vectores soporte y muchas otras. Entre ellas, destacan dos subclases de algoritmos:

- Los algoritmos de aprendizaje basados en instancias [Aha97]. El clasificador del vecino más cercano es el algoritmo basado en instancias más conocido. Es también un importante ejemplo dentro de la familia de *algoritmos de aprendizaje perezoso*. Éstos son técnicas que, a diferencia del resto de técnicas de aprendizaje automático, no construyen un clasificador durante su fase de entrenamiento. Este tipo de clasificadores han heredado muchos rasgos beneficiosos, incluyendo su relativa simplicidad, su adaptabilidad a diferentes problemas generales, su flexibilidad a la hora de ser optimizado, la no necesidad de un proceso de entrenamiento y la capacidad de poder incorporar nuevo conocimiento al clasificador de forma sencilla.

- Los algoritmos de aprendizaje basados en inducción de reglas [F̈99]. Son sistemas de aprendizaje que representan un paradigma transparente, fácilmente comprensible y aplicable y más genérico que los árboles de decisión, puesto que muchas de las técnicas utilizadas en los sistemas de aprendizaje de reglas fueron adaptadas directamente de los árboles de decisión. A diferencia del grupo anterior, los algoritmos construyen modelos sencillos de interpretar y rápidos de usar en la fase de predicción. La construcción del modelo es el paso más tedioso de este tipo de algoritmos y su optimización para adaptarlo a problemas complejos supone un proceso muy costoso.

En la literatura especializada, se ha desarrollado la teoría de aprendizaje llamada aprendizaje de ejemplos anidados generalizados, donde el aprendizaje se lleva a cabo por objetos en un espacio Euclídeo de n dimensiones E^n, como hiperrectángulos. Los hiperrectángulos pueden anidarse uno dentro de otro hasta una profundidad arbitraria. Al contrario que la mayoría de procesos de generalización, que sustituyen fórmulas simbólicas por fórmulas más generales, el proceso de generalización de estos algoritmos modifican los hiperrectángulos aumentándolos y cambiando su volumen de acuerdo al problema concreto. Los ejes de estos hiperrectángulos se definen por las variables de cada ejemplo, siendo independiente del tipo de variable, numérica o categórica.

La base de este tipo de modelos es una hibridación de los algoritmos de aprendizaje basados en instancias con los algoritmos de inducción de reglas. De esta manera, los ejemplos se almacenan y se usan estrictamente como puntos en E^n, y las reglas se corresponden con hiperrectángulos, o ejemplos generalizados, que disponen de un determinado volumen y se pueden interpretar como un sub-espacio euclídeo que tiene asociado una categoría de predicción. Entre las ventajas de estos modelos, destacan la simplicidad de los algoritmos de generación de modelos y la representación de modelos, aunando las ventajas relativas a la buena capacidad y adaptación predictiva de clasificadores basados en instancias con la posibilidad de interpretación de hiperrectángulos en forma de reglas. También cabe mencionar la posibilidad del aprendizaje anidado de generalización con excepciones, si tenemos en cuenta que los hiperrectángulos se solapan o incluyen unos dentro de otros y los más pequeños son vistos como excepciones dentro de una regla más genérica. Finalmente, la flexibilidad es otra propiedad que se puede mencionar, puesto que este tipo de modelos permite la inclusión de pesos, cambios de funciones de distancias en las variables, etc.

El razonamiento ordinal se refiere a una categoría importante en los problemas

de predicción. En éstos, tanto los atributos de entrada como las clases toman valores ordinales dentro de dominios determinados [KS13]. La clasificación de ejemplos en categorías ordinales es un problema popular que ha ido atrayendo la atención de los profesionales en minería de datos durante estos últimos años. En la literatura, han aparecido multitud de términos en relación a este problema, tales como, clasificación ordinal, regresión ordinal o *ranking labelling*, pero todos comparten una propiedad en común: el atributo de salida o clase es ordinal.

La restricción de monotonicidad entre atributos de entrada y la clase viene dada por una relación de ordenación parcial que establece el orden entre ejemplos. La restricción de monotonicidad obliga a que el orden también se mantenga en la clase y se trata de un aspecto común en muchas áreas de aplicación. Sin embargo, los modelos obtenidos por las técnicas de aprendizaje automático no garantizan el cumplimiento de esta restricción. Esto ha motivado el desarrollo de algoritmos que sean capaces de manejar dichas restricciones, como árboles de decisión. La clasificación con restricciones monotónicas, también conocida como clasificación monotónica, [BDSP89], es un problema con una restricción clara sobre la monotonía de los datos: un valor mayor en un atributo, mientras los demás se mantienen, no debe asignarse a una clase menor [KS13]. Los árboles de decisión [HCZ$^+$12a, MP15a], la inducción de reglas y el aprendizaje basado en instancias constituyen tres de las técnicas más prometedoras para resolver la clasificación monotónica.

A pesar de los numerosos modelos y enfoques propuestos por diferentes autores para dar soporte a problemas de clasificación monotónica, éstos se han centrado normalmente en técnicas basados en modelos aislados y no híbridos. Además, la aplicación de técnicas en problemas con índole real donde los ejemplos pueden presentar violaciones monotónicas apenas se ha explorado. Los resultados de investigación obtenidos en este ámbito hasta la fecha no son suficientes, ya que surgen nuevas dificultades y retos que requieren un estudio más profundo de los datos y la aplicación de algoritmos de optimización complejos y adaptativos a cada problema. Algunos de estos retos y dificultades se describen a continuación:

- *Implantación de otros modelos complejos en clasificación monotónica*, que aún no se han usado en este campo y que reúnan las características ventajosas que no se están explotando en modelos monotónicos. Este es el caso de los algoritmos basados en ejemplos anidados generalizados, que requieren de una nueva formalización de la teoría para ser aplicados a este campo y un algoritmo sencillo que construya modelos monotónicos con este tipo de

representaciones.

- *Necesidad de obtener modelos predictivos monotónicos sobre bases de datos reales*, donde cabe la posibilidad que los ejemplos presenten violaciones monotónicas. Este campo apenas ha sido explorado y está estrechamente ligado a las aplicaciones que disponemos en la realidad. Los algoritmos deben ser capaces de trabajar con datos imperfectos y obtener modelos cuyas predicciones finales sean lo más monotónicas posibles.
- *Alta complejidad en la obtención de modelos fiables en clasificación monotónica en entornos reales*. Cuando los datos son imperfectos, el espacio de soluciones posibles se hace más complejo y los modelos finales pueden dejar de ser fiables sin una buena heurística de búsqueda. Los algoritmos evolutivos son técnicas inspiradas por la computación natural y diseñadas para abordar problemas de búsqueda y optimización. Estos algoritmos representan un conjunto de soluciones y las evolucionan utilizando diferentes operadores que combinan y distribuyen las soluciones. La buena adaptabilidad de los algoritmos evolutivos en tareas de aprendizaje nos permitirá abordar problemas complejos en clasificación monotónica en entornos reales.

La constante evolución y retos actuales encontrados en los problemas de clasificación monotónica, algunos de los cuales acabamos de describir, condujeron durante el comienzo de esta investigación a formular la siguiente hipótesis de partida:

Los algoritmos de clasificación monotónica existentes no están preparados para abordar problemas más complejos y que presentan imperfecciones en su estado original, representadas como violaciones en la monotonicidad en los datos. Por esta razón, es necesario tratar la clasificación monotónica con técnicas híbridas y algoritmos de optimización avanzados para obtener modelos fiables y sencillos.

B Objetivos

Teniendo en cuenta los retos actuales en el campo de la clasificación monotónica mediante técnicas de inducción de reglas y aprendizaje basado en instancias, el propósito inicial de esta investigación es el desarrollo de modelos híbri-

dos instancias-reglas para clasificación monotónica basados en el paradigma de ejemplos anidados generalizados o aprendizaje basado en hiperrectángulos, caracterizados por la alta flexibilidad en la construcción de fronteras separatorias de clases y por permitir cierta interpretabilidad en la lectura de las reglas que inducen [WD95, Dom96]. Estos nuevos algoritmos incorporarán técnicas de *soft computing*, como los algoritmos evolutivos, para la optimización de la precisión, simplificación y reducción de predicciones no monotónicas de los modelos resultantes.

En base al propósito genérico del desarrollo de algoritmos basados en ejemplos anidados generalizados para clasificación monotónica, nos planteamos los siguientes objetivos:

1. Realización de un *estudio de las propuestas en la literatura en el campo de las técnicas de clasificación basados en generalización de instancias, inducción de reglas clásicas y aprendizaje basado en instancias para clasificación monotónica*. Además, se plantea el estudio del uso de técnicas de *soft computing*, como los algoritmos evolutivos, para optimizar diferentes aspectos durante la obtención de modelos predictivos por técnicas de clasificación monotónicas.

2. Formalización de la *teoría de aprendizaje basada en ejemplos anidados generalizados al problema de la clasificación monotónica y desarrollo de un primer algoritmo voraz* que la utilice para obtener modelos híbridos basados en reglas-instancias, o hiperrectángulos, que produzcan predicciones monotónicas precisas. El algoritmo deberá utilizarse para aprender sobre bases de datos completamente monotónicas.

3. Desarrollo de una técnica basada en algoritmos evolutivos que persigue la selección de los mejores hiperrectángulos para optimizar los modelos de acuerdo a su precisión, simplicidad y ausencia de violaciones de monotonicidad en las predicciones. El uso de algoritmos evolutivos nos permite abordar problemas reales donde no todos los ejemplos cumplen las expectativas de monotonicidad debido al ruido inherente de los datos.

C Estructura

Para alcanzar los objetivos que acabamos de plantear, y según lo establecido en el artículo 23, punto 3, de la normativa vigente para los Estudios de Doctorado en la Universidad de Jaén (Programa RD. 1393/2007), esta memoria de investigación será presentada como un conjunto de artículos publicados por el doctorando. Dichas publicaciones constituyen el núcleo de la tesis, y corresponden a dos artículos científicos publicados y/o aceptados en Revistas Internacionales indexadas por la base de datos JCR (Journal Citation Reports), producida por ISI (Institute for Scientic Information).

Por tanto, la memoria se compone de un total de dos publicaciones, y se estructura en los siguientes capítulos:

- Capítulo 1: en él se presenta una revisión del proceso de descubrimiento de conocimiento en bases de datos y del problema abordado en la presente memoria sobre la clasificación monotónica, haciendo un repaso de los conceptos básicos y antecedentes sobre problemas de minería de datos, aprendizaje supervisado y el problema de la clasificación. Para finalizar este capítulo, introduciremos una visión amplia sobre el problema de la clasificación monotónica y una taxonomía que revisa y caracteriza un gran número de técnicas existentes en la literatura, haciendo hincapié en cuatro clasificadores monotónicos relevantes.

- Capítulo 2: este capítulo presenta un resumen de la investigación realizada para alcanzar los objetivos planteados en esta memoria, y muestra una breve discusión de los resultados obtenidos en cada propuesta. Dichas propuestas son desarrolladas en los dos artículos que se encuentran en las Secciones 3.1 y 3.2.

- Capítulo 3: Este capítulo constituye el núcleo de la tesis doctoral, y contiene las dos publicaciones obtenidas como resultado de esta investigación.

- Capítulo 4: En este capítulo, se señalan las conclusiones y resultados más relevantes de la investigación realizada, así como las futuras líneas de investigación a seguir.

La memoria concluye con una recopilación bibliográfica de las contribuciones más destacadas en la materia estudiada.

Capítulo 1

Conceptos Teóricos y Antecedentes

En este capítulo se revisan los conceptos teóricos y antecedentes fuente de inspiración para comprender la investigación que se presenta en esta memoria. De esta manera, en la Sección 1.1 definimos el proceso de descubrimiento de información en bases de datos, y describimos las etapas que lo componen. En la Sección 1.2 introducimos la fase de minería de datos y en la Sección 1.3 describimos brevemente el aprendizaje supervisado, dentro del cual dedicamos especial atención a la clasificación. En la última sección explicamos con más detalle la clasificación monotónica.

1.1. Descubrimiento de Conocimiento en Bases de Datos

El término descubrimiento de conocimiento en bases de datos (*Knowledge Discovery in Databases*, también conocido como KDD) empezó a utilizarse en 1989 para referirse al proceso complejo de búsqueda de conocimiento en bases de datos, y para enfatizar la aplicación a alto nivel de métodos específicos de extracción de modelos e información a partir de los datos.

Fayyad et al. en [FPSS96] define el KDD como *el proceso no trivial de iden-*

tificar patrones válidos, novedosos, potencialmente útiles y, en última instancia, comprensibles a partir de los datos. En esta definición se resumen cuáles deben ser las propiedades deseables del conocimiento extraído:

- Válido: hace referencia a que los patrones deben seguir siendo precisos para datos nuevos (con un cierto grado de certidumbre), y no sólo para aquellos que han sido usados en su obtención.
- Novedoso: que aporta algo desconocido para el sistema y preferiblemente para el usuario.
- Potencialmente útil: la información debe conducir a acciones que reporten algún tipo de beneficio para el usuario.
- Comprensible: la extracción de patrones no comprensibles dificulta o imposibilita su interpretación, revisión, validación y uso en la toma de decisiones. De hecho, una información incomprensible no proporciona conocimiento (al menos, desde el punto de vista de su utilidad).

Como se deduce de la anterior definición, el KDD es un proceso complejo que incluye no sólo la obtención de los modelos o patrones, sino también la evaluación y posible interpretación de los mismos.

Los principales pasos dentro del proceso de KDD pueden son los siguientes [FPSS96]:

1. Especificación y comprensión del problema: en esta fase se diseña y organiza el dominio de aplicación, el conocimiento previo relevante obtenido por los expertos y los objetivos finales perseguidos por el usuario final. Para ello debemos comprender los datos seleccionados y el conocimiento experto asociado. Este paso requiere cierta dependencia usuario/analista, pues intervienen factores como: los cuellos de botella del dominio, saber qué partes son susceptibles de un procesado automático y cuáles no, cuáles son los objetivos, los criterios de rendimiento exigibles, para qué se usarán los resultados que se obtengan, compromisos entre simplicidad y precisión del conocimiento extraído, etc.
2. Preprocesamiento de los datos [GLH15]: en esta etapa se incluyen tareas como la limpieza de los datos, integración de los datos, eliminación de ruido, estrategias para manejar valores perdidos, normalización de los datos, etc.

Así mismo, se pueden incluir actividades como la búsqueda de características útiles de los datos según sea el objetivo final, la reducción del número de variables y la proyección de los datos sobre espacios de búsqueda en los que sea más fácil encontrar una solución. Este es un paso crítico dentro del proceso global, que requiere un buen conocimiento del problema y una buena intuición, y que, con frecuencia, marca la diferencia entre el éxito o fracaso de la minería de datos.

3. Minería de datos: en esta etapa se pueden seguir diferentes estrategias para obtener modelos, según el interés se centre en clasificación, regresión, agrupamiento de conceptos (*clustering*), detección de desviaciones, etc. En el libro de Han y Kamber ([HK11]), puede verse con detalle los diferentes métodos de minería de datos. En este paso se realiza la búsqueda de conocimiento con una determinada representación del mismo. El éxito de la minería de datos depende en gran parte de la correcta realización de los pasos previos.

4. Interpretación y evaluación: en este paso del proceso se realiza la interpretación del conocimiento extraído, con posibilidad de iterar de nuevo desde el primer paso. La obtención de resultados aceptables dependerá de factores como: definición de medidas de interés del conocimiento (de tipo estadístico, en función de su sencillez, etc.) que permitan filtrarlo de forma automática, existencia de técnicas de visualización para facilitar la valoración de los resultados o búsqueda manual de conocimiento útil entre los resultados obtenidos, consolidación del conocimiento descubierto, incorporándolo al sistema, o simplemente documentándolo y enviándolo a la parte interesada. Este paso incluye la revisión y resolución de posibles inconsistencias con otro conocimiento extraído previamente.

1.2. Minería de datos

La minería de datos se aplica en multitud de áreas, entre ellas se encuentran: aplicaciones empresariales, industriales, toma de decisiones en banca, seguros, finanzas, marketing, control de calidad, retención de clientes, predicción, políticas de acción, sanidad, aplicaciones en investigación científica, análisis y gestión de mercados, análisis de riesgo en banca y seguros, minería de datos en industria, diagnóstico en medicina, etc.

La fase de minería de datos es la más característica del KDD y, por esta razón, muchas veces se utiliza esta fase para nombrar todo el proceso. Consiste en la extracción automatizada o conveniente de los patrones que presentan conocimiento implícitamente almacenado o capturados en una gran base de datos, almacenes de datos, la web, otros repositorios de información masivos, o flujos de datos.

Como un campo multidisciplinario, la minería de datos se basa en el trabajo de multitud de áreas incluyendo la estadística, aprendizaje automático, reconocimiento de patrones, tecnología de bases de datos, recuperación de información, redes de ordenadores, sistemas basados en el conocimiento, inteligencia artificial, computación de alto rendimiento, y visualización de datos. La minería de datos surgió durante la década de 1980, e hizo grandes avances durante la década de 1990, y continúa floreciendo en el nuevo milenio.

La minería de datos es un conjunto de técnicas de análisis de datos que permiten [WF05]:

- Extraer patrones, tendencias y regularidades para describir y comprender mejor los datos.
- Extraer patrones y tendencias para predecir comportamientos futuros.

Debido al gran volumen de datos este análisis no puede ser manual (ni incluso facilitado por herramientas de almacenes de datos y OLAP) sino que debe ser (semi-)automático [OQR07]. Dicho análisis se realiza construyendo un modelo basado en los datos recopilados para este efecto. El modelo es una descripción de los patrones y relaciones entre los datos que pueden usarse para hacer predicciones, para entender mejor los datos o para explicar situaciones pasadas. Para ello es necesario tomar una serie de decisiones antes de empezar el proceso:

- Determinar qué tipo de tarea de minería es la más apropiada. Por ejemplo, podríamos usar la clasificación para predecir en una entidad bancaria los clientes que dejarán de serlo.
- Elegir el tipo de modelo. Por ejemplo, para una tarea de clasificación podríamos usar un árbol de decisión, porque queremos obtener un modelo interpretable de manera sencilla por cualquier usuario.
- Elegir el algoritmo de minería que resuelva la tarea y obtenga el tipo de modelo que estamos buscando. Esta elección es pertinente porque existen muchos métodos para construir los modelos. Por ejemplo, para crear árboles

de decisión para clasificación podríamos usar CART o C4.5, entre otros. En lo que resta de esta sección, describimos las tareas y modelos más utilizados, así como algunos conceptos relacionados con la construcción del modelo.

Dentro de la minería de datos hemos de distinguir tipos de tareas [HK11, WF05], cada una de las cuales puede considerarse como un tipo de problema a ser resuelto por un algoritmo de minería de datos. Esto significa que cada tarea tiene sus propios requisitos, y que el tipo de información obtenida con una tarea puede diferir mucho de la ofrece otra.

Las distintas tareas pueden ser predictivas, descriptivas o de verificación. En la Figura 1.1 se muestra una taxonomía de los diferentes algoritmos de minería de datos, según sea su finalidad:

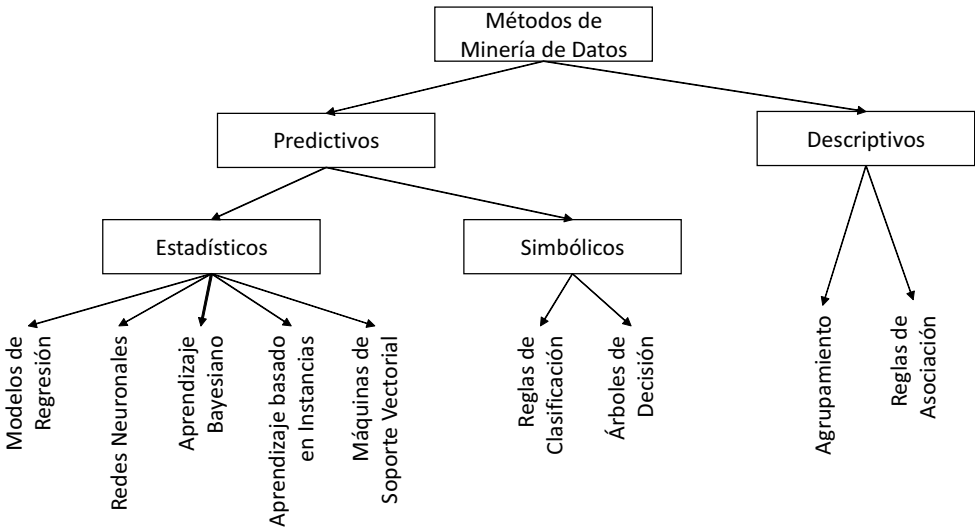


Figura 1.1: Taxonomía de técnicas de minería de datos.

Entre las tareas predictivas encontramos la clasificación y la regresión, mientras que el agrupamiento (*clustering*), las reglas de asociación, las reglas de asociación secuenciales y las correlaciones son tareas descriptivas. Veamos en mayor detalle todas ellas:

- La clasificación es quizá la tarea más utilizada. En ella, cada instancia (o registro de la base de datos) pertenece a una clase, la cual se indica mediante el valor de un atributo que llamamos *clase de la instancia*. Este atributo puede tomar diferentes valores discretos, cada uno de los cuales corresponde a una clase. El resto de los atributos de la instancia se utilizan para predecir la clase. El objetivo es predecir la clase de nuevas instancias. Más concretamente, el objetivo del algoritmo es maximizar la razón de precisión de la clasificación de nuevas instancias, la cual se calcula como el cociente entre las predicciones correctas y el número total de predicciones (correctas e incorrectas).

- La regresión es también una tarea predictiva que consiste en aprender una función real que asigna a cada instancia un valor numérico. Ésta es la principal diferencia respecto a la clasificación; el valor a predecir es numérico. El objetivo en este caso es minimizar el error (generalmente el error cuadrático medio) entre el valor predicho y el valor real.

- El agrupamiento (*clustering*) es la tarea descriptiva por excelencia y consiste en obtener grupos a partir de los datos. Hablamos de grupos y no de clases, porque, a diferencia de la clasificación, en lugar de analizar datos etiquetados con una clase, los analiza para generar esta etiqueta. Los datos son agrupados basándose en el principio de maximizar la similitud entre los elementos de un grupo minimizando la similitud entre los distintos grupos. Es decir, se forman grupos tales que los objetos de un mismo grupo son muy similares entre sí y, al mismo tiempo, son muy diferentes a los objetos de otro grupo. Al agrupamiento también se le denomina segmentación, ya que parte o segmenta los datos en grupos que pueden ser o no disjuntos. El agrupamiento está muy relacionado con la sumarización, que algunos autores consideran una tarea en sí misma, en la que cada grupo formado se considera como un resumen de los elementos que lo forman para así describir de una manera concisa los datos.

- Las reglas de asociación son también una tarea descriptivaque tiene como objetivo identificar relaciones no explícitas entre atributos categóricos. Pueden ser de muchas formas, aunque la formulación más común es del estilo *si el atributo X toma el valor d entonces el atributo Y toma el valor b*. Las reglas de asociación no implican una relación causa-efecto, es decir, puede no existir una causa para que los datos estén asociados. Este tipo de tarea se utiliza frecuentemente en el análisis de la cesta de la compra. La idea

es identificar productos que son frecuentemente comprados juntos, información esta que puede usarse para ajustar los inventarios, para la organización física del almacén o en campañas publicitarias.

Dado que la minería de datos es un campo muy interdisciplinar, existen diferentes paradigmas detrás de las técnicas utilizadas para esta fase: técnicas de inferencia estadística, árboles de decisión, redes neuronales, inducción de reglas, aprendizaje basado en instancias, algoritmos genéticos, aprendizaje bayesiano, programación lógica inductiva y varios tipos de métodos basados en núcleos, entre otros. Cada uno de estos paradigmas incluye diferentes algoritmos y variaciones de los mismos, así como otro tipo de restricciones que hacen que la efectividad del algoritmo dependa del dominio de aplicación, no existiendo lo que podríamos llamar el método universal aplicable a todo tipo de aplicación.

Existen muchos conceptos estadísticos que son la base de muchas técnicas de minería de datos. Ejemplo de ello es la regresión lineal, un método simple pero frecuentemente utilizado para la tarea de regresión. Las técnicas estadísticas no son sólo útiles para regresión, sino que se utilizan también para discriminación (clasificación o agrupamiento). La inferencia de funciones discriminantes que separan clases o grupos, también se puede realizar de manera paramétrica o no paramétrica. El método más conocido es el análisis de discriminantes lineales de Fisher.

Algunas de las técnicas de discriminantes no paramétricos tienen una relación muy estrecha con los métodos basados en núcleo, de los cuales las máquinas de soporte vectorial son su ejemplo más representativo, en el que se busca un discriminante lineal que maximice la distancia a los ejemplos fronterizos de los distintos grupos o clases.

En otras ocasiones, deseamos calcular, para una instancia dada sin clasificar, cuál es la probabilidad de que se le asigne cada una de las clases, y seleccionar la de mayor probabilidad. Ésta es la idea que subyace en los métodos bayesianos. Uno de los métodos más utilizados es el Naive Bayes, que se basa en la regla de Bayes y que asume la independencia de los atributos dada la clase. Este método funciona bien con bases de datos reales, sobre todo cuando se combina con otros procedimientos de selección de atributos que sirven para eliminar la redundancia.

Los árboles de decisión son una serie de condiciones organizadas en forma jerárquica, a modo de árbol. Son muy útiles para encontrar estructuras en espacios de alta dimensionalidad y en problemas que mezclen datos categóricos y numéricos. Esta técnica se usa en tareas de clasificación, agrupamiento y regre-

sión. Los árboles de decisión usados para predecir variables categóricas reciben el nombre de árboles de clasificación, ya que distribuyen las instancias en clases. Cuando los árboles de decisión se usan para predecir variables continuas se llaman árboles de regresión.

Los árboles de decisión pueden considerase una forma de aprendizaje de reglas, ya que cada rama del árbol puede interpretarse como una regla, donde los nodos internos en el camino desde la raíz a las hojas definen los términos de la conjunción que constituye el antecedente de la regla, y la clase asignada en la hoja es el consecuente.

En general, la inducción de reglas es un conjunto de métodos para derivar un conjunto de reglas comprensibles de la forma:

$$SI\ cond_1\ Y\ cond_2\ Y.....Y\ cond_n\ ENTONCES\ pred.$$

El antecedente de la regla (la parte SI) contiene una conjunción de n condiciones sobre los valores de los atributos independientes, mientras que el consecuente de la regla (la parte ENTONCES) contiene una predicción sobre el valor de un atributo objetivo.

Aunque los árboles de decisión pueden también producir un conjunto de reglas (tal y como hemos visto anteriormente), los métodos de inducción de reglas son diferentes ya que:

- las reglas son independientes y no tienen por qué formar un árbol.
- las reglas generadas pueden no cubrir todas las situaciones posibles.
- las reglas pueden entrar en conflicto en sus predicciones; en este caso, es necesario elegir qué regla se debe seguir. Un método para resolver los conflictos consiste en asignar un valor de confianza a las reglas y usar la que tenga mayor confianza.

Algunos métodos de obtención de reglas, en especial las reglas de asociación, se basan en el concepto de conjuntos de items frecuentes (*frequent itemsets*) y utilizan técnicas de conteo y soporte mínimo para obtener las reglas.

Las redes neuronales artificiales (*Artificial Neural Networks* o *AANNs*) son todo un paradigma de computación muy potente que permite modelizar problemas complejos en los que puede haber interacciones no lineales entre variables.

Como los árboles de decisión, las redes neuronales pueden usarse en problemas de clasificación, de regresión y de agrupamiento. Las redes neuronales trabajan directamente con datos numéricos, para usarlas con datos nominales éstos deben transformarse en números primero.

Una red neuronal puede verse como un grafo dirigido con muchos nodos (elementos del proceso) y arcos entre ellos (sus interconexiones). Cada uno de estos elementos funciona independientemente de los demás, usando datos locales (la entrada y la salida del nodo) para dirigir su procesamiento.

En el aprendizaje basado en instancias o casos (*Instance Based Learning* o *IBL*), las instancias se almacenan en memoria, de tal forma que cuando llega una nueva instancia cuyo valor es desconocido se intenta relacionar ésta con las instancias almacenadas (cuya clase o valor es conocida) buscando las que más se parecen, con el objetivo de usar los valores de estas instancias similares para estimar los valores a obtener de la nueva instancia en cuestión. Por lo tanto, más que intentar crear reglas, se trabaja directamente con los ejemplos.

Todo el trabajo en el aprendizaje basado en instancias se hace cuando llega una instancia a clasificar y no cuando se procesa el conjunto de entrenamiento. En este sentido se trata de un método retardado o perezoso, ya que retrasa el trabajo real tanto como sea posible, a diferencia de los otros métodos vistos hasta el momento que tienen un comportamiento anticipativo o voraz, produciendo generalizaciones en cuanto reciben los datos de entrenamiento.

En el aprendizaje basado en instancias, cada nueva instancia se compara con las existentes usando una métrica de distancia, y la instancia más próxima se usa para asignar su clase a la instancia nueva. La variante más sencilla de este método de clasificación es conocido como el vecino más próximo (*nearest neighbor*). Otra variante, conocida como el método de los k vecinos más próximos (k *nearest neighbors*), usa los k vecinos más próximos, en cuyo caso la clase mayoritaria de estos k vecinos se asigna a la nueva instancia.

El aprendizaje basado en instancias es muy útil para trabajar sobre tipos de datos no estándar, como los textos o multimedia. El único requerimiento para incluir un tipo de datos es la existencia de una métrica apropiada de distancia para formalizar el concepto de similitud.

1.3. Aprendizaje Supervisado

En el campo de la minería de datos, los métodos de predicción son comúnmente referidos como aprendizaje supervisado. Los métodos de aprendizaje supervisados se emplean para conseguir el descubrimiento de las relaciones entre las variables de entrada y del atributo objetivo (en la mayoría de ocasiones hablaremos de clase). Las relaciones buscadas son representadas en una estructura llamada modelo. Generalmente, un modelo describe y explica experiencias, que están ocultas en los datos, y que pueden ser usadas en la predicción del valor del atributo objetivo, cuando los valores de entrada son conocidos [GLH15].

El aprendizaje supervisado está presente en muchos dominios de aplicación, como las finanzas, medicina e ingeniería. En un escenario típico de aprendizaje supervisado, partimos de un conjunto de entrenamiento con el objetivo de obtener un modelo que pueda ser usado para predecir ejemplos desconocidos. Este conjunto de entrenamiento puede ser descrito de diferentes formas. La más común es un conjunto de instancias, que es básicamente una colección de tuplas que puede contener duplicados. Cada tupla es descrita por un vector con valores de atributos. Cada atributo tiene un dominio asociado de valores que son conocidos con anterioridad a la tarea de aprendizaje. Los atributos pueden ser: nominales o numéricos (enteros o reales). Los atributos nominales tienen una cardinalidad finita, mientras los dominios de atributos numéricos están delimitados por unos limites inferior y superior. El espacio de instancias (el conjunto de posibles ejemplos) es definido como un producto cartesiano de todos los atributos de entrada. El espacio de instancias universal es definido como un producto cartesiano de todos los dominios de los atributos de entrada y el dominio del atributo objetivo.

Los dos problemas clásicos que pertenecen a la categoría del aprendizaje supervisado son la clasificación y la regresión. En la clasificación, el dominio del atributo objetivo es finito y categórico. Esto es, hay un número finito de clases o categorías para predecir una muestra y estos son conocidos por el algoritmo de aprendizaje. Un clasificador puede asignar una clase a un ejemplo no conocido cuanto este ha sido entrenado a través de un conjunto de entrenamiento. La naturaleza de la clasificación es discriminar unos ejemplos de otros, consiguiendo como uso principal una predicción fiable. De esta manera tenemos modelos ajustados a datos pasados. Si suponemos que el futuro es similar al pasado, entonces asumimos que podemos hacer predicciones correctas para nuevas instancias. Sin embargo, cuando el atributo objetivo esta formado por infinitos valores, como es el caso de la predicción de un número real dentro de un cierto intervalo, habla-

mos de problemas de regresión. Aquí, la propuesta del aprendizaje supervisado es ajustar un modelo para aprender el atributo objetivo de salida como función de los atributos de entrada. Obviamente, el problema de regresión presenta más dificultades que el problema de clasificación y tanto los requerimientos de computación como la complejidad del modelo son mayores. Dentro de la regresión, otro tipo de aprendizaje es el análisis de las series temporales, que tiene que ver con hacer predicciones a lo largo del tiempo. Aplicaciones típicas incluyen el análisis de las cotizaciones, el mercado de valores y la previsión de ventas.

El aprendizaje supervisado ([HK11]) a partir de un conjunto de ejemplos en la forma de pares (entradas, salidas), encuentra las reglas o funciones que mejor modelan la relación entre las entradas y las salidas (a modo de ejemplos, ver Figura 1.2).

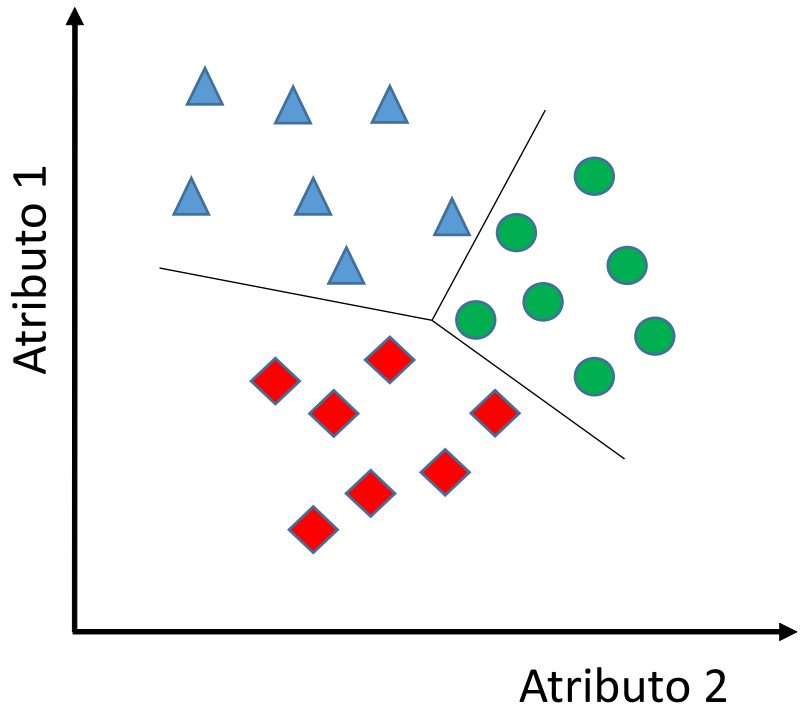


Figura 1.2: Aprendizaje supervisado.

En un problema de aprendizaje supervisado tenemos los siguientes elementos:

- Variable nominal o categórica: valores discretos.
- Variable numérica: valores no discretos.
- Variable de salida: También suele llamarse clase.
- Clase true/false: ejemplo positivo/negativo.
- Clase discreta/numérica: Problema de clasificación/regresión

La Figura 1.3 muestra un ejemplo de la terminología que estamos utilizando.

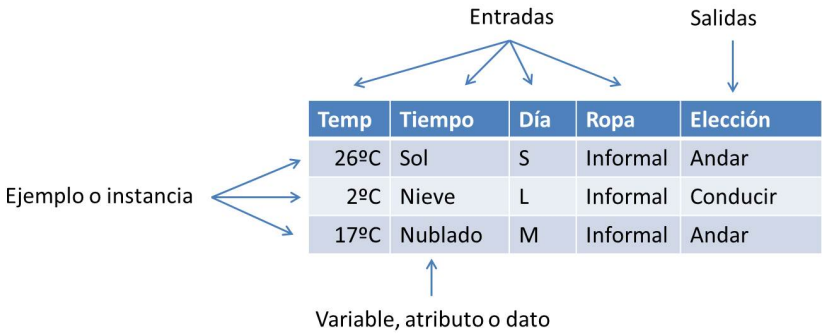


Figura 1.3: Terminología.

1.4. El problema de la clasificación.

La clasificación es una de las tareas de minería de datos más conocidas. Consiste en predecir una determinada clase (categoría) para un objeto. En la tarea de clasificación, dados un conjunto de ejemplos ya clasificados, se pretende construir un modelo que permita clasificar nuevos casos. Es un tipo de aprendizaje supervisado donde se conoce la clase verdadera de cada uno de los ejemplos que

se utilizan para construir el clasificador. Algunas aplicaciones típicas son: aprobación de créditos ([CS13, SAS15]), marketing directo ([HAE13]), detección de fraudes ([BH02]), diagnóstico médico ([AA16, SASS11]), etc.

1.4.1. Definición

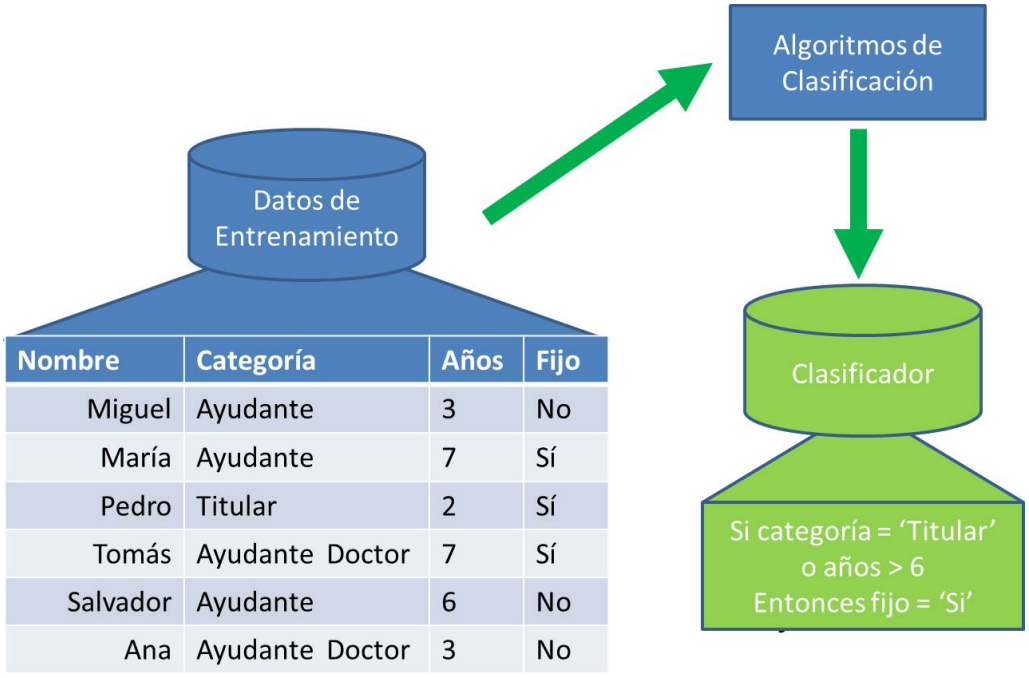
En un problema de clasificación ([HK11]) tenemos un objeto X_i, que se describe a través de un conjunto de características (m variables o atributos): $X_{i1}, X_{i2}, ..., X_{im}$, al cual le va a corresponder una etiqueta o clase entre los posibles valores del conjunto $C = \{c_1, c_2, ..., c_t\}$, nominales o numéricos, y notaremos como $Clase(X_i)$ el valor que le corresponde a X_i. A partir de ahora llamaremos instancia o ejemplo al par $E_i = (X_i, Clase(X_i))$. El objetivo de la tarea de la clasificación es encontrar una función $f: A_1 \times A_2 \times ... \times A_m \rightarrow C$ donde A_i es el dominio donde toma valores el atributo i.

Las características o variables elegidas dependen del problema de clasificación.

Hay que diferenciar dos etapas en la tarea de la clasificación:

1. Construcción del modelo: Este asocia a cada nueva instancia o ejemplo con una determinada categoría. Cada ejemplo (tupla) se sabe que pertenece a una clase (etiqueta del atributo clase). Se utiliza un conjunto de ejemplos, conjunto de entrenamiento (*training set*) para la construcción del modelo. El modelo obtenido se representa como un conjunto de reglas de clasificación, árboles de decisión, fórmula matemática, etc. Esta etapa se puede observar en la Figura 1.4.

2. Utilización del modelo: Tras la construcción del modelo se pueden realizar clasificaciones futuras y también hacer una estimación de la precisión del modelo. Para ello se utilizan un conjunto de ejemplos distintos de los utilizados para su construcción llamados conjunto de prueba (*test set).* Si el conjunto de test no fuese independiente del de entrenamiento ocurre un proceso de sobreajuste (*overfitting*). Para cada ejemplo de test se compara la clase determinada por el modelo con la clase real (conocida). El ratio de precisión es el porcentaje de ejemplos de test que el modelo clasifica correctamente.



Nombre	Categoría	Años	Fijo
Miguel	Ayudante	3	No
María	Ayudante	7	Sí
Pedro	Titular	2	Sí
Tomás	Ayudante Doctor	7	Sí
Salvador	Ayudante	6	No
Ana	Ayudante Doctor	3	No

Figura 1.4: Construcción del modelo.

1.4.2. Evaluación de clasificadores

En la evaluación de los clasificadores hay tres aspectos que se deben estudiar [HK11]: en primer lugar los criterios para evaluar el clasificador, en segundo lugar las técnicas de validación de estos criterios y en tercer lugar las técnicas para poder comparar varios modelos construidos sobre el mismo problema para poder seleccionar el más apropiado y/o preciso.

A continuación vamos a definir algunos conceptos necesarios para la evaluación de un clasificador:

- Matriz de Confusión : Dado un problema de clasificación con t clases, una matriz de confusión es una matriz $t \times t$ en la que una entrada $c_{i,j}$ indica el número de ejemplos que se han asignado a la clase c_j, cuando la clase correcta es c_i.
- Tasa de acierto: $s = \frac{\sum_i c_{i,i}}{n}$(La suma del numero de elemento clasificado

correctamente entre el total de elementos).

- Tasa de error : $\epsilon = 1 - s$.

Tras ver los criterios para valorar la bondad de un clasificador cabe hacerse la siguiente pregunta: ¿en realidad es honesta la estimación de la bondad del clasificador construido?. Utilizar como bondad la tasa de acierto sobre el conjunto de entrenamiento no es realista. El porcentaje obtenido suele ser demasiado optimista debido a que el modelo estará sobre ajustado a los datos utilizados durante el proceso de aprendizaje. Una de las técnicas de validación de clasificadores más utilizadas y reconocidas en la literatura es la validación cruzada, que consiste en:

1. Dividir el conjunto de datos en k-subconjuntos (*folds*), $S_1, \ldots, S_k$ de igual tamaño.

2. Aprender k clasificadores utilizando en cada uno de ellos un conjunto de entrenamiento distinto. Se valida con su conjunto de test correspondiente:

   $Entrenamiento = S_1 \cup S_2 \cup ... \cup S_{i-1} \cup S_{i+1} \cup ... \cup S_k$
   $Test = S_i$

3. Devolver como tasa de acierto (error) el promedio obtenido en las k-iteraciones

1.5. Clasificación Monotónica

La clasificación monotónica tiene sus orígenes en los problemas de clasificación ordinales [GPOSM$^+$16]. Por ejemplo, un empleado puede ser descrito como excelente, bueno o malo y un bono se puede calificar como AAA, AA, A, A-, etc. Al igual que una escala numérica, una escala ordinal tiene un orden, pero a diferencia de ella no posee una noción precisa de distancia. No podemos decir que AA está más cerca de AAA de lo que está de A, ni a la inversa. A este respecto una escala ordinal es similar a una nominal. Los problemas de clasificación ordinales son importantes, ya que son muy comunes en diferente ámbitos:

- Vida diaria [BDST09]. Por ejemplo, la selección de la mejor ruta para trabajar, dónde comprar, qué producto comprar, y dónde vivir, etc.

- En los negocios ([Gam98]), en tareas como la selección y promoción de los empleados, la determinación de la calificación crediticia, calificación de los bonos, el rendimiento económico de los países, sectores y empresas, y la contratación de seguros, etc.
- Entorno académico ([BD92]), tales como Rankings de Manuscritos, evaluación de profesores, admisión de alumnos, y las decisiones sobre las becas para los estudiantes.

Problemas ordinales se han investigado en la literatura en disciplinas científicas tales como la toma de decisiones en psicología y medicina ([DAB97, Roy00]), el derecho ([Kar91]) y la estadística ([BDST09]). En la última década, un número creciente de publicaciones han mostrado un progreso en el aprendizaje artificial de conceptos ordinales. Modelos de aprendizaje automático, tales como árboles de decisión, redes neuronales y máquinas de soporte vectorial han sido adaptados para la clasificación ordinal. Cada modelo tiene diferentes supuestos. Por ejemplo, una de las principales diferencias entre los diversos enfoques hacia el aprendizaje de conceptos ordinales es la manera en la que se trata la monotonía. Algunos clasificadores necesitan ejemplos monótonos para aprender, mientras que otros son capaces de hacerlo a partir de ejemplos no monótonos también.

En las siguientes secciones vamos a explicar brevemente el concepto de restricción monotónica, para pasar a ver algunas de las propuestas clásicas para abordar la clasificación bajo este paradigma.

1.5.1. Concepto de restricción monotónica

En primer lugar definiremos el concepto de relación de orden para pasar a continuación a estudiar el concepto de monotonía [BD95].

Relación de orden [Men64, Bir67, Rom08]

Sea A un conjunto dado no vacío y $\Re$ una relación binaria definida en A, se dice que es una relación de orden, si cumple las siguientes propiedades:

- Reflexividad: todo elemento de A está relacionado consigo mismo. Es decir, $\forall x \in A\colon x \Re x$.
- Antisimetría: si dos elementos de A se relacionan entre sí, entonces ellos son iguales. Es decir, $\forall x, y \in A\colon x \Re y, y \Re x \Rightarrow x = y$.

- Transitividad: si un elemento de A está relacionado con otro, y ese otro a su vez se relaciona con un tercero, entonces el primero estará relacionado también con este último. Es decir, $\forall x, y, z \in A\colon x \Re y, y \Re z \Rightarrow x \Re z$.

Una relación de orden $\Re$ sobre un conjunto puede denotarse con el par ordenado $(A, \preceq)$.

Una relación de orden es total si y solo si todos los elementos de A se relacionan entre sí, es decir:

$$\forall x, y \in A\colon tq(x \preceq y) \ \vee (y \preceq x) \tag{1.1}$$

Una relación de orden es parcial si y solo si al menos un par de elementos de A se relacionan entre sí, es decir:

$$\exists x, y \in A\colon tq(x \preceq y) \ \vee (y \preceq x) \tag{1.2}$$

A continuación definimos una relación de orden parcial en el conjunto de las instancias [BDSP89]. Sea $X_i = (x_{i,1}, x_{i,2}, ..., x_{i,m})$ y $X_h = (x_{h,1}, x_{h,2}, ..., x_{h,m})$ dos instancias en el mismo dominio del problema, descrito por m atributos. Todos los valores de los atributos, $x_{i,j}$ e $x_{h,j}$, se supone que son ordinales (pueden ordenarse) o numéricos. Un orden entre X_i y X_h se define como sigue:

$$\begin{array}{l} X_i = X_h \; si \; x_{i,j} = x_{h,j} \; \forall j = 1, ..., m \\ X_i \succ X_h \; si \; x_{i,j} > x_{h,j} \; \forall j = 1, ..., m \\ X_i \succeq X_h \; si \; x_{i,j} > x_{h,j} \;\; \text{ó} \;\; x_{i,j} = x_{h,i} \forall j = 1, ..., m \\ X_i \prec X_h \; si \; x_{i,j} < x_{h,j} \; \forall j = 1, ..., m \\ X_i \preceq X_h \; si \; x_{i,j} < x_{h,j} \;\; \text{ó} \;\; x_{i,j} = x_{h,j} \forall j = 1, ..., m \end{array} \tag{1.3}$$

Relación monotónica

Sean $(X_i, Clase(X_i))$ y $(X_h, Clase(X_h))$ dos pares (instancia,clase) se dice que son antimonotónicos si:

$$\begin{array}{l} X_i \succeq X_h \; y \; Clase(X_i) < Clase(X_h) \;\; \text{ó} \\ X_i \preceq X_h \; y \; Clase(X_i) > Clase(X_h) \;\; \text{ó} \\ X_i = X_h \; y \; Clase(X_i) \neq Clase(X_h) \end{array} \tag{1.4}$$

Se dice que dos pares $(X_i, Clase(X_i))$ y $(X_h, Clase(X_h))$ son monótonicos entre sí, si no cumplen con alguna de las condiciones establecidas en 1.4.

1.5.2. Criterios de Evaluación

Los clasificadores ordinales no monotónicos pueden ser evaluados usando diferentes métricas [BES09, CS11, JS11, CHSG14]. Las dos métricas más comunes son la precisión y el error medio absoluto, las cuales son utilizadas de forma general en clasificación. El primero de ellos se define como:

$$precisión = \frac{1}{n}\sum_{i=1}^{n}[Clase(X_i) \neq Predicción(X_i)] \tag{1.5}$$

donde la $Clase(X_i)$ y la $Predicción(X_i)$ son respectivamente el valor real de la clase y el valor predicho de la clase de la instancia, y n el número total de instancias. La precisión toma valores dentro del intervalo [0,1].

Esta medida no tiene en cuenta la magnitud del error, en cambio, el error medio absoluto (*Mean Absolute Error*, *MAE* [BES09]) incluye alguna información del orden. Esta métrica es el valor medio de las desviaciones absolutas entre el valor real de la clase y el predicho asociados a la clase de la instancia X_i:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}[\mathcal{O}(Clase(X_i)) - \mathcal{O}(Predicción(X_i))] \tag{1.6}$$

donde MAE toma valores en el intervalo [0,Q-1] y Q es el cardinal del conjunto de los valores de la clase.

Para evaluar específicamente la monotonicidad de los clasificadores monotónicos disponemos de varias métricas. La primera de ellas es el índice de No Monotonicidad (NMI), definida por Ben-David en [BD95] como el cociente del número de violaciones de las restricciones de monotonicidad dividido por el número total de pares de instancias. Si el conjunto de datos D tiene n instancias, el número total de pares será $n^2 - n$. La fórmula para calcular este índice será:

$$NMI(D) = \frac{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} M_{ij}}{n^2 - n} \tag{1.7}$$

donde M es una matrix binaria y M_{ij} es igual a 1 si el par formado por X_i y X_j es no monotónico y 0 en caso contrario.

Para añadir información en los árboles de decisión sobre la monotonicidad se añade a la entropía (*E-score*) una puntuación de la ambigüedad del orden

(*order-ambiguity-score* [BD95]) que se calcula la siguiente manera:

$$A = \begin{cases} 0 & \text{si NMI=0} \\ -(log_2 NMI)^{-1} & \text{en otro caso} \end{cases} \tag{1.8}$$

Índice de No Monotonicidad 1 (NMI_1 [MDP13]): Se define como:

$$NMI_1 = \frac{1}{n(n-1)} \sum_{X \in D} NClash(X) \tag{1.9}$$

donde $NClash(X)$ es el número de instancias del conjunto de datos D que no cumplen las restricciones de monotonicidad con respecto a X.

Índice de No Monotonicidad 2 (NMI_2 [MDP13]): Se define como:

$$NMI_2 = \frac{1}{n} \sum_{X \in D} Clash(X) \tag{1.10}$$

donde $Clash(X) = 1$ si el ejemplo X es no monótono con algún otro ejemplo o instancia de D.

1.5.3. Clasificadores monotónicos: Taxonomía

En la actualidad son cinco las familias más importantes de clasificadores monotónicos. Dicha clasificación ha sido realizada basándonos en sus diferentes enfoques y modelos resultantes (ver Figura 1.5).

- Basados en instancias: Son los más conocidos y los pioneros. En esta sección encontramos aquellos métodos basados en criterios de vecindad para efectuar sus clasificaciones. En la subsección 1.5.4 se explicaran con mayor profundidad.
  - Ordered Learning Model (OLM [BD92]).
  - Ordinal Stochastic Dominance Learner ($OSDL$ [LBCV08]).
  - Monotonic k-Nearest Neighbor ($MkNN$ [DF08]).
- Árboles de decisión y Clasificadores basados en reglas: en este caso, se obtienen modelos de segmentación recursivos o bien conjuntos de reglas independientes. Los métodos que podemos agrupar en este ámbito son:

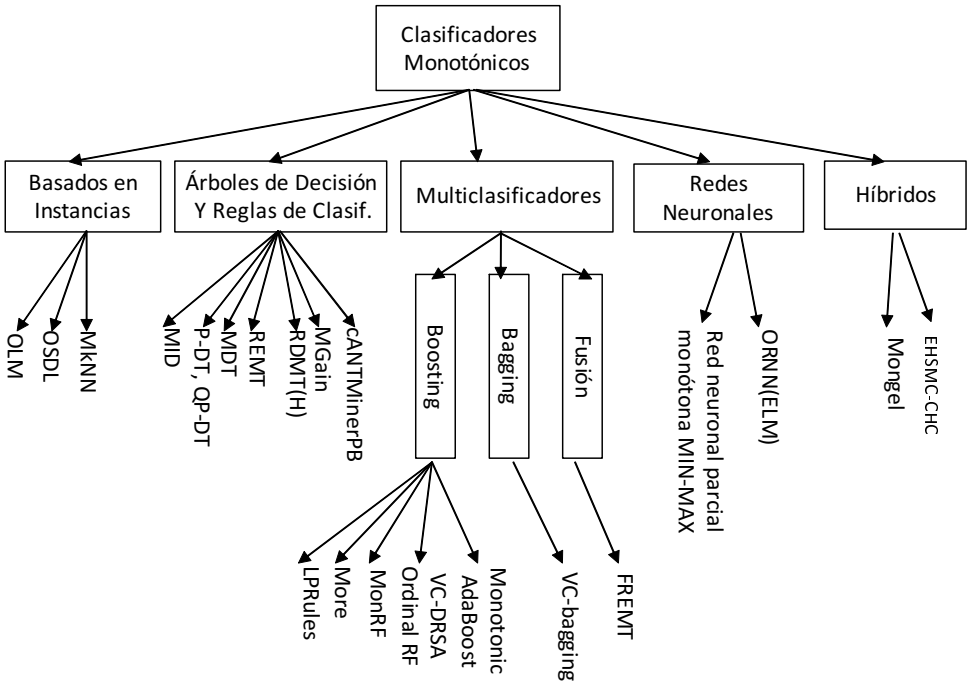


Figura 1.5: Taxonomía de los Clasificadores Monotónicos.

- *MID* [BD95]. Es una extensión del algoritmo ID3 ([Qui93]) propuesta por Ben David (será explicado con más detalle en la subsección 1.5.4).
- *P-DT*, *QP-DT* [KM99]. Makino y otros propusieron un árbol de decisión monótono (P-DT), y un quasi monótono (QP-DT) como extensión del algoritmo ID3 en un problema de clasificación para dos clases. En ambos casos, empiezan con un conjunto de datos monótono, siendo éste un requerimiento inicial necesario, sin embargo en el caso de QP-DT la monotonicidad solo se garantiza en el conjunto de entrenamiento mientras que en P-DT el árbol debe de ser monotónico también.
- *MDT* [LYW03]. El objetivo de este algoritmo es predecir el orden implícito en relación a la comparación de parejas en la clasificación original.
- *REMT* [HCZ$^+$12b]. En esta propuesta se emplea una medida llamada ranking de información mutua, que es una formalización ordinal de la información mutua de Shannon. Dicha medida es sensible a la monoto-

nicidad y robusta frente a datos ruidosos. Con ella, se pueden construir árboles clasificadores binarios garantizando una forma débil de monotonicidad, en caso de que el conjunto de partida fuera monotónicamente consistente. El algoritmo REMT propuesto demostró en los estudios realizados un comportamiento aceptable comparado con clasificadores monótonos y no monótonos.

- *RDMT(H)* [MP15b]. El algoritmo se basa en un un árbol clasificador binario parametrizado por una medida de discriminación H empleada para segmentación, así como tres parámetros más para hacer un podado previo. De esta manera, el algoritmo garantiza una forma débil de monotonicidad del árbol resultante.
- *MGain* [ZZZW15]. Se trata de una propuesta basada también en árboles clasificadores binarios. Para ello los autores proponen un índice de consistencia monotónica de un punto de corte con respecto al conjunto de datos, pero manteniendo las prestaciones predictivas del árbol de clasificación clásico.
- *cANTMinerPB* [BO16]. Frente a los casos anteriores donde los modelos resultantes eran árboles de decisión, en este caso se obtiene un conjunto de reglas. Los autores proponen una extensión de un sistema de aprendizaje de clasificadores basados en reglas, donde la extracción de los modelos se lleva a cabo mediante optimización con colonias de hormigas.

- Multiclasificadores: aquí encontramos algoritmos basados en técnicas de *boosting* combinando reglas de decisión, técnicas de *bagging* sobre reglas y árboles de decisión o bien métodos que fusionan modelos. A continuación presentamos los que se encuentran presentes en la literatura:

  - Técnicas de *Boosting*.
    - *LPRules* [KS09]. El algoritmo consiste en tres etapas, de forma que en la primera de ellas se descompone un problema multiclase en una secuencia de subproblemas binarios. A continuación, los datos de cada subproblema binario son monotonizados empleando para ello una aproximación no paramétrica que explota la clase de todas las funciones monótonas. Finalmente, se genera un multiclasificador de reglas usando el método *LPBoost*.

- *MORE* [DKS09]. Este método emplea un esquema de modelado aditivo por etapas hacia adelante para generar un multiclasificador de reglas de decisión para problemas binarios, basado en un mecanismo de *boosting*.
- *MonRF* [GHG15]. El objetivo de los autores en este caso fué el proponer un mecanismo multiclasificador de poda basado en el grado de monotonicidad de los árboles resultantes obtenidos mediante un proceso de *Random Forest*.
- *VC-DRSA Ordinal Random Forest* [WZZZ15]. Este algoritmo tiene como pilar constitutivo la aproximación de consistencia en la dominancia de variables basada en conjuntos aproximados. Consta de tres fases, donde en la primera se realiza un muestreo ordinal aleatorio basado en la aproximación anteriormente citada. En la segunda etapa se construyen árboles de decisión en paralelo empleando las instancias anteriormente elegidas y en la última etapa, se genera un conjunto de reglas mediante un algoritmo ordinal de *Random Forest*. Para ello los autores emplean *MapReduce* y *Hadoop*.
- *Monotonic AdaBoost* [GHG16]. Este algoritmo considera árboles de decisión monotónicos combinados en un esquema *AdaBoost*, empleando un multiclasificador de poda basado en el grado de monotonicidad.

- Técnicas de *Bagging*.
  - *VC-bagging* [BSS10]. Los clasificadores que componen el multiclasificador de *bagging* están formados por reglas de decisión inducidas mediante objetos estructurados que emplean la aproximación de consistencia en la dominancia de variables basada en conjuntos aproximados.
- Técnicas de Fusión.
  - *FREMT* [QXL$^+$15]. Los autores proponen un método para fusionar árboles de decisión basado en dos principios: por un lado un mecanismo de reducción de atributos para el aprendizaje de clasificadores basados en preservación de ranking. Como segundo factor clave, se emplea un principio de fusión basado en probabilidad maximal a través de la combinación de clasificadores base.

- Redes Neuronales: Estas han sido utilizadas en fechas recientes como otro

paradigma alternativo para afrontar problemas de clasificación monotónicos. De entre algunos de los trabajos realizados en este sentido podemos citar:

  - *Red neuronal parcial monótona MIN-MAX* [DV10]. Los autores al comparar esta con otras redes, indican que la propuesta mejora sus prestaciones. Lo justifican dado que presenta como ventaja el considerar todas las variables disponibles con las restricciones apropiadas durante la fase de entrenamiento y test, sin la necesidad de eliminar o forzar la monotonicidad en variables no monotónicas.
  - *ORNN(ELM)* [FRC14]. Este método extiende modelos existentes de aprendizaje automático extremo al ámbito de la regresión ordinal.

- Métodos Híbridos: En este grupo situamos a métodos que combinan diferentes paradigmas. Entre ellos podemos citar:

  - *Mongel* [GFA$^+$15]. Se trata de una de las propuestas que conforman esta tesis doctoral en la cual se combina aprendizaje basado en instancias con inducción de reglas. Dicho método será extensamente descrito en el Capítulo 3, Sección 3.1.
  - *EHSMC-CHC* [GAA$^+$16]. Segunda propuesta original de esta tesis doctoral donde se emplean algoritmos evolutivos para la selección de hiperrectángulos, teniendo como objetivo abordar problemas de clasificación monotónica en entornos reales donde existen imperfecciones en los datos. El algoritmo será descrito en el Capítulo 3, Sección 3.2.

En la siguiente sección se describirán algunos de los métodos de clasificación monotónica más ampliamente utilizados y reconocidos en la literatura.

1.5.4. Clasificadores monotónicos relevantes

A continuación vamos a describir con más detalle tres métodos de aprendizaje basados en instancias: OLM, OSDL y MkNN, y un método basado en árboles de decisión: MID. Estos métodos son a nuestro juicio los más importantes en el ámbito de la clasificación monotónica y por ello los hemos usado para compararlos con los métodos propuestos en nuestro estudio.

1.5.4.1. Ordered Learning Model (OLM [BD92])

Es un algoritmo simple que aprende las relaciones ordinales monótonas de los datos mediante la eliminación de inconsistencias de los pares que violan las restricciones de monotonicidad. El algoritmo está basado en instancias, lo cual significa que almacena los ejemplares de aprendizaje en memoria, y es capaz de deducir las etiquetas de clase de ejemplares no visibles por alguna técnica de extrapolación generalmente local. Los ejemplares de aprendizaje se almacenan como un conjunto de reglas.

En un principio, el conjunto de reglas está vacío. Después, durante el aprendizaje, cada ejemplar se comprueba con cada una de las reglas del conjunto de reglas. Si el ejemplar es antimonótono con una regla, el ejemplar o la regla es seleccionado al azar, mientras que la otra se desecha. Si el ejemplar está seleccionado se debe comprobar de nuevo con cada una de las reglas del conjunto. Si pasa esta prueba de consistencia se añade al conjunto de reglas como una regla más. De este modo, el conjunto es siempre coherente. A este conjunto de ahora en adelante lo notaremos como CISE (*Consistent and Irredudant Examples*).

La clasificación se hace de una manera similar al aprendizaje, comparando el ejemplar a clasificar con las reglas del conjunto de reglas. Las reglas se comprueban en el orden descendiente de los valores de atributo de decisión (es decir, clases). Al ejemplar se le asigna la clase indicada por la primera regla que lo cubre. Esto es equivalente a la asignación de la clase máxima i sugerido por las reglas que cubren al ejemplar. Si no hay una regla en el conjunto de reglas que cubra el ejemplar, dos enfoques son posibles. El primer enfoque y más simple consiste en asignarle al ejemplar la peor clase. En el segundo enfoque, la clase es asignada buscando la regla más cercana al ejemplar de acuerdo con la distancia euclidiana.

OLM produce un conjunto más pequeño de reglas durante el aprendizaje [BD95]. La principal debilidad de este modelo se encuentra en el hecho de que no hace ninguna comprobación de la precisión durante el aprendizaje. Otro inconveniente es que el conjunto de reglas depende del orden en que los ejemplares se procesan. Por otra lado, la clasificación según el vecino más cercano de los ejemplares que no estén cubiertos puede conducir a clasificaciones no monótonas.

A continuación presentamos el pseudocódigo del algoritmo en su fase de aprendizaje (ver Algoritmo 1) y en su fase de clasificación (ver Algoritmo 2). Al comienzo, CISE está vacío. Cada ejemplo se compara con todas las reglas que hay

actualmente en CISE para determinar si se añade o no.

Algoritmo 1 Fase de Aprendizaje del OLM

```
Fase 1:
Ordenamos los ejemplares en orden descendente de la clase de salida;
while haya ejemplos sin comprobar do
    Seleccionamos un nuevo ejemplo, E_i = (X_i, Clase(X_i))
    Marcamos cada ejemplo E_h tq X_h = X_i
    Remplazamos todos los ejemplos marcados con E = (X_i, Clase(X_h)^-)
end while
Fase 2:
CISE = ∅
while haya ejemplos sin comprobar do
    Dado un nuevo ejemplar , E_i = (X_i, Clase(X_i))
    y_max = min{class(x')|x' ∈ T ∧ x ≤ x'}
    if CISE = ∅ then
        CISE = CISE ∪ E_i
    else
        for para todas las reglas de CISE do
            if existe E_h ∈ CISE tal que es redundante o inconsistente con E_i then
                if X_i ≺ X_h y E_i es consistente y no redundante con CISE then
                    CISE = CISE − {E_h}
                    CISE = CISE ∪ {E_i}
                else
                    CISE = CISE − {E_i}
                end if
            end if
        end for
    end if
    if E_h no ha sido rechazado hasta ahora then
        CISE = CISE ∪ {E_h}
    end if
end while
```

Tras crear CISE para clasificar se ordenan las reglas obtenidas de forma decreciente según el valor de su clase.

1.5.4.2. Ordinal Stochastic Dominance Learner (OSDL [LBCV08])

Ofrece una alternativa a OLM, ya que también es un método basado en instancias. Utiliza el concepto de dominancia estocástica ordinal (OSD [CV03]) para resolver la clasificación ordinal con restricciones monotónicas.

Vamos a introducir una notación para definir el objetivo de este algoritmo: $Y = \{Clase(X_h)/X_h \in X\}$, $F(Y)$ el conjunto de todas las posibles distribuciones de probabilidad sobre Y y $\trianglelefteq_1$ la relación de dominancia estocástica de primer orden o débil, que explicaremos posteriormente. El objetivo de este algoritmo es encontrar una función $\bar{F} : X \times Y \to [0, 1]$ que cumpla las siguientes condiciones:

- $\bar{F}$ es no creciente en el primer argumento.

Algoritmo 2 Fase de Clasificación del OLM

```
1: while haya ejemplos E_h = (X_h, Clase(X_h)) do
2:     Comparamos el ejemplar E_h con la regla actual E_i = (X_i, Clase(X_i)) ∈ CISE
3:     if X_h ≻ X_i then
4:         Clasificamos E_h como Clase(X_i)
5:     else
6:         Clasificamos E_h con la media de las clases de los ejemplares más cercanos
7:     end if
8: end while
```

- $\bar{F}$ es no decreciente en el segundo argumento.
- $\bar{F}(\cdot, max(Y)) = 1$.
- $\bar{F}$ minimiza alguna función de riesgo.

Definimos $\bar{\lambda}_{prob} : (X, \preceq_X) \rightarrow (F(Y), \trianglelefteq_1)$ como $\bar{\lambda}_{prob}(X_h) := \bar{f}_{x_h}$ distribución de probabilidad asociada a la función de distribución $\bar{F}_{x_h}(X_h, \cdot)$.

Obtenida esta función de distribución, para cada X_h tenemos una función de distribución $\bar{F}(X_h, \cdot)$ y su correspondiente distribución de probabilidad $\bar{f}_{x_h}$. Posteriormente es fácil calcular el valor asociado a la clase de X_h calculando el valor $E[\bar{f}_{x_h}]$ (esperanza de la distribución de probabilidad asociada a X_h).

Dominancia Estocástica

Este concepto establece una relación de orden en el conjunto de las distribuciones de probabilidad $F(Y)$ [CVB03]. Dadas dos instancias X_i y X_h, las dos distribuciones de probabilidad asociadas a estas instancias f_{x_i} y f_{x_h} decimos que hay una relación estocástica de primer orden (FOSD) entre ellas si (ver Figura 1.6):

$$f_{x_i} \trianglelefteq_1 f_{x_h} \Leftrightarrow \forall\, l \in Y\ F(X_i, l) \geq F(X_h, l) \tag{1.11}$$

En el algoritmo OSDL se construye dos funciones de comparación F_m y F_M: una que se basa en los ejemplares de la mejor clase, entre las que están dominados estocásticamente por una instancia X_i, y la segunda, que se basa en los ejemplares de la peor clase entre los que dominan estocásticamente a X_i. Más precisamente, para una clase dada i:

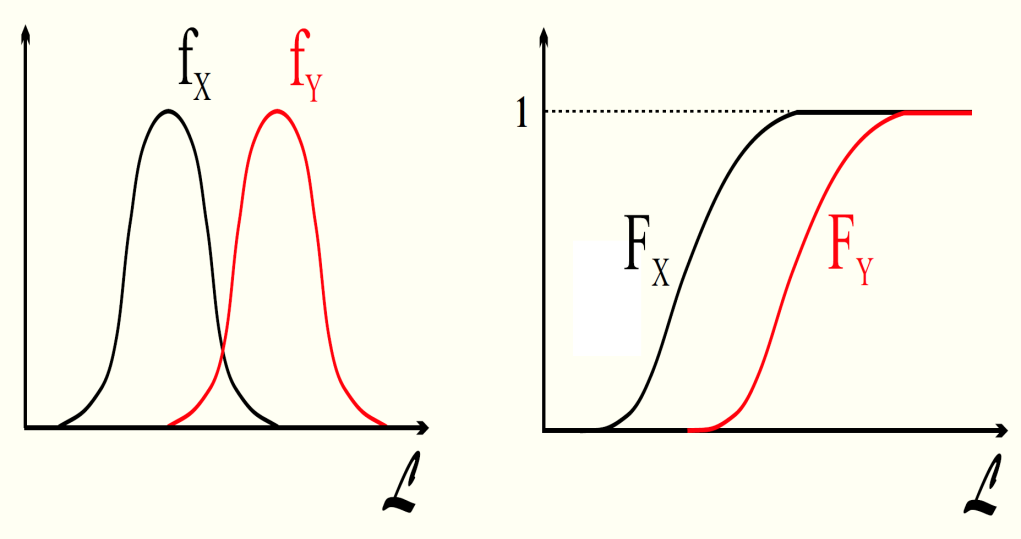


Figura 1.6: Dominancia estocástica.

$$F_m(X_i;l) = \min_{X_h \in (X_i]} \hat{F}_{X_h}(l) \quad , F_M(X_i;l) = \max_{X_h \in [x)} \hat{F}_{X_h}(l) \quad , \tag{1.12}$$

donde $(X_i] = \{X_h \in X / X_h \preceq X_i\}$ es el conjunto de ejemplares dominados por X_i y $[X_i) = \{X_h \in X / X_i \preceq X_h\}$ es el conjunto de ejemplares que dominan a X_i y $\hat{F}_{X_h}(l)$ es el valor de la función de distribución de X_h en la clase l, donde la distribución de probabilidad f_{X_h} toma los siguientes valores $\forall\, l \in Y = Clases(X)$ $f_{X_h}(l) \equiv P(clase(X_h) = l)$. Por otra parte, si$(X_i] = \emptyset$, entonces $F_m(X_i;l) = 1$ y si $[X_i) = \emptyset$, entonces $F_m(X_i;l) = 0$.

Durante la clasificación, se realiza una interpolación de las funciones de correspondencia F_m y F_M. Esta interpolación implica un parámetro de escala $s \in [0;1]$:

$$\bar{F}(X_i;l) = (1-s)F_m(X_i;l) + s * F_M(X_i;l) \tag{1.13}$$

Tal interpolación tiene un inconveniente: para mantener la monotonía de la clasificación se requiere utilizar el mismo valor fijo de s para todos los ejemplares clasificados. Con el fin de tratar los datos inconsistentes y para superar el problema con el parámetro de escala s una versión equilibrada de OSDL se propone

en [CV03]. Esta versión implica la siguiente interpolación entre las funciones de correspondencia:

$$\bar{F}(X_i;l) = \begin{cases} (1-s)F_m(X_i;l) + s * F_M(X_i;l) & \text{si } F_m(X_i;l) \geq F_M(X_i;l) \\ \frac{(1-s')N_m(X_i;l)F_m(X_i;l)+s'*N_M(X_i;l)F_M(X_i;l)}{N_m(X_i;l)+N_M(X_i;l)} & \text{en otro caso} \end{cases} \tag{1.14}$$

donde $s, s' \in [0,1]$, $N_m(X_i;l)$ es el número de instancias de $(X_i]$ cuya clase es $\succ l$ y $N_M(X_i;l)$ es el número de instancias de$[X_i)$ cuya clase es $\preceq l$. Por lo tanto, la versión equilibrada de OSDL introduce ponderación por el número de instancias $N_m(X_i;l)$ y $N_M(X_i;l)$ que se hace para instancias inconsistentes. Tiene el propósito de reducir la influencia de las instancias inconsistentes en la clasificación.

Vistos algunos detalles de cálculo del algoritmo podemos pasar a describir las fases de aprendizaje y clasificación del algoritmo.

En la fase de aprendizaje del algoritmo OSDL, se construye el clasificador en dos etapas como puede verse en el Algoritmo 3:

1. Construcción del conjunto de datos y cálculo de la función de distribución discreta estimada.

2. Calculamos el parámetro s determinado por una validación cruzada leave-one-out.

En el algoritmo cada vez que hay un ejemplar nuevo $(X_h, Clase(X_h))$ para clasificar se aplica la función $AñadirInstancia\,(X_h, Clase(X_h))$, que añade X_h al conjunto de datos y actualiza la función de distribución discreta estimada $\hat{F}_{X_h}$. También se utiliza la función $EliminarInstancia\,(X_h, Clase(X_h))$ en este algoritmo que hace lo contrario que $AñadirInstancia$. Para terminar con la construcción del clasificador se procede a calcular el valor del parámetro s en el conjunto de valores $\{0.1;\ 0.2;\ ...;\ 1\}$ que hace mínimo el valor de los errores (la diferencia entre la clase real y la clase predicha), utilizando un proceso de validación cruzada.

En la fase de clasificación añadimos la instancia, modificamos $\bar{F}(X_i;l)\, y\, F_m(X_i;l)$, para obtener una nueva función de distribución y devolvemos la media de la distribución de probabilidad.

1.5.4.3. MID ([BD95])

Utilizar árboles de decisión basados en teoría de la información que emplean la entropía como criterio de selección de atributos produce árboles de decisión no monotónicos (como se puede observar en la Figura 1.7), que no cumplen con las restricciones de monotonía. A estos algoritmos se los denomina TDIDT (*Top-Down-Induction Decision Trees*, TDIDT [Qui93]).

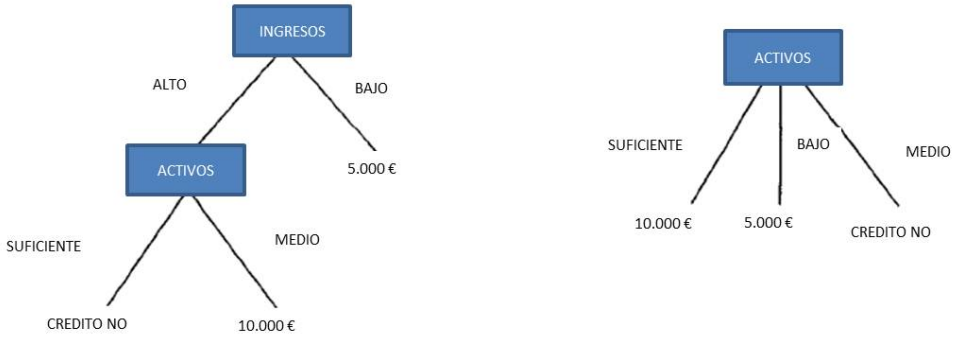


Figura 1.7: Ejemplos de árboles de decisión no monotónicos.

Los algoritmos TDIDT que usan *E-score* como métrica de selección de atributos no consideran el orden en los atributos de entrada y de clase. Estos algoritmos no se adaptan del todo a los problemas monotónicos. Sin embargo, los árboles de decisión proporcionan una precisión aceptable. Desafortunadamente estos dos objetivos entran en conflicto. Como solución a este problema Ben-David propuso la adaptacion del *E-score* de forma que se tuviera en cuenta el orden de los atributos.

A continuación vamos a definir una serie de conceptos que utilizó Ben-David para la construcción de sus árboles de decisión monotónicos, hasta llegar a una métrica que el denomina puntuación de ambigüedad total.

Índice no monotónico

Es el cociente del número de pares de ramas no monotónicas entre sí del árbol de decisión, dividido entre el número total de pares de la rama del árbol de decisión.

Para calcular este índice, dado un árbol con K ramas, construimos una matriz M de orden K, cuyos elementos m_{ij} se definen de la siguiente forma:

$$m_{ij} = \begin{cases} 0 & \text{si las ramas } i,j \text{ no son no monotónicas} \\ 1 & \text{si las ramas } i,j \text{ son no monotonicas} \end{cases} \tag{1.15}$$

El índice no monotónico del árbol se calcula como:

$$I_{a_1,\ldots,a_g} = \frac{\sum_{i=1}^{k}\sum_{i=1}^{k} m_{ij}}{k^2 - k} \tag{1.16}$$

Puntuación de ambigüedad del orden

$$A_{a_1,\ldots,a_g} = \begin{cases} 0 & \text{si } I_{a_1,\ldots,a_g} = 0 \\ -(log\, I_{a_1,\ldots,a_g})^{-1} & \text{en otro caso} \end{cases} \tag{1.17}$$

Puntuación de ambigüedad total

Es la suma de los $E - score$ definida en los árboles obtenidos con el algoritmo ID3 y la puntuación de ambigüedad del orden (1.17):

$$T_{a_1,\ldots,a_g} = E_{a_1,\ldots,a_g} + A_{a_1,\ldots,a_g} \tag{1.18}$$

Esta métrica selecciona aquellos atributos con puntuación total de ambigüedad más baja. Esta puntuación tiene algunas propiedades deseables para los problemas de monotonicidad ya que considera el error en la predicción y el índice de no monotonicidad. Su definición no implica la monotonicidad en el proceso de construcción del árbol. En la mayor parte de los casos el índice no monotonicidad es sustancialmente inferior a 0.50, y en la mayoría de los casos es muy inferior a los valores de *E-score*.

Una forma efectiva de expresar las dependencias entre la entropía y la monotonicidad puede ser conseguida introduciendo un parámetro adicional para el cálculo de la ambigüedad total:

$$T_{a_1,\ldots,a_g} = E_{a_1,\ldots,a_g} + R * A_{a_1,\ldots,a_g} \tag{1.19}$$

El parámetro R expresa la importancia relativa de la monotonía con respecto a la precisión en el problema dado. Cuando $R = 0$, el *score* del ambigüedad total usa sólo su *E-score* componente. Si R es un valor muy alto la consideraciones de monotonía dominan la construcción del árbol de decisión.

1.5.4.4. Monotonic k- Nearest Neighbor Classifier [DF08]

Este algoritmo de Duivesteijn et al modifica el método clásico de los k-vecinos para su aplicación a la clasificación monotónica. Este método no paramétrico está compuesto de dos fases. En la primera fase, los datos de entrenamiento se hacen monótonos reetiquetando el menor número de casos posibles. Este conjunto de datos reetiquetados puede ser visto como un clasificador monótono con una tasa de error más pequeña en los datos de entrenamiento. En la segunda fase, se utiliza el algoritmo del vecino más cercano modificándolo para predecir las etiquetas de clase de nuevos datos, de tal manera que las restricciones de monotonicidad sean satisfechas.

Reetiquetado de los datos

Para reetiquetar los datos es necesario construir el grafo de violación de restricciones de monotonicidad (MGV). Es un grafo dirigido $G = (V, E)$, donde $V = \{1, 2, ..., n\}$ y $(i, j) \in E$ si $X_i \leq X_j$ y $Clase(X_i) > Clase(X_j)$. El MGV es un grafo de orden estrictamente parcial.

Según se describió en [RDBDM06], el máximo subconjunto independiente (no hay vértices adyacentes) del MGV corresponde al subconjunto de tamaño máximo monótono de los datos. La Figura 1.8 muestra un ejemplo de de MGV. Si reetiquetamos el complementario de este subconjunto máximo independiente obtenido del MGV, obtendremos con muy pocos cambios un conjunto de datos monótonos. A pesar de que encontrar un conjunto máximo independiente en un grafo arbitrario es un problema NP-completo, esto no es así en grafos comparables (gráfos con orden parcial). Para este tipo de gráfos, un conjunto independiente máximo corresponde a una anticadena máxima en el orden parcial correspondiente. Esta se puede calcular en $O(n^3)$ resolviendo un problema de flujo mínimo en una red de transporte (problema del viajante) que se construye fácilmente a partir de la gráfica de la comparabilidad. Un ejemplo de red de transporte se puede ver en la Figura 1.9.

Como resumen, para reetiquetar los datos se sigue el siguiente proceso:

1. Construir el grafo de violación de restricciones de monotonicidad, $G = (V, E)$.

2. Transformar el grafo en una red de transporte $G' = (V', E')$.

3. Calculamos el camino de flujo mínimo en la red de transporte.

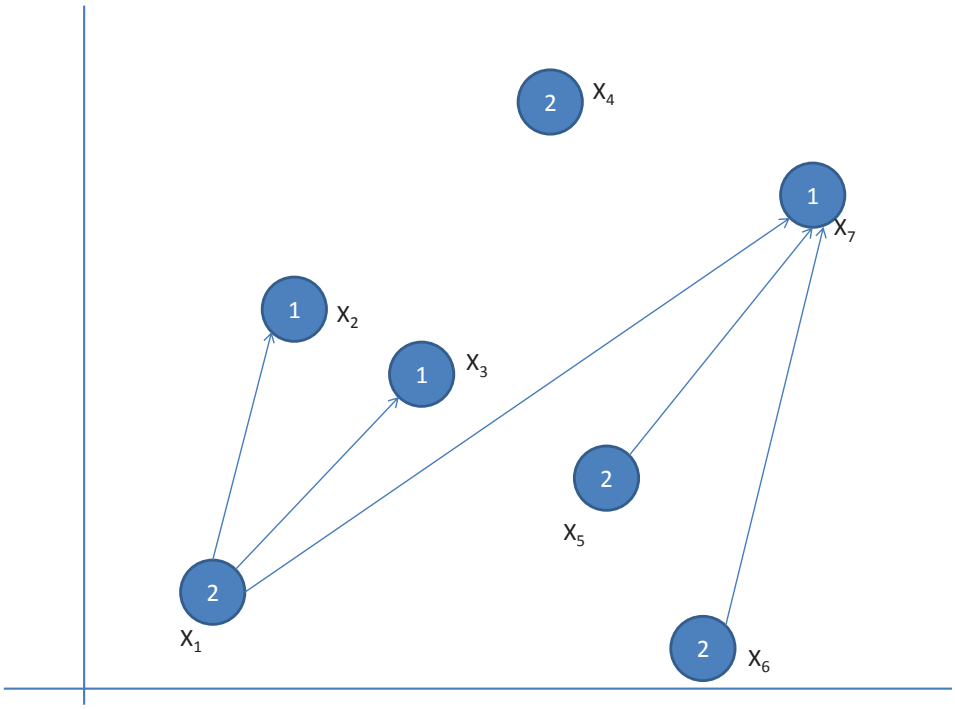


Figura 1.8: Ejemplo de MGV.

4. Los vértices de este conjunto son los del subconjunto independiente máximo M.

5. Reetiquetamos el resto de los que están en el grafo, $R = V/M$.

Algoritmo del vecino más cercano monotónico

Con el fin de satisfacer las restricciones de monotonía, es evidente que la etiqueta de clase asignada a un nuevo punto de datos X_0 se ve limitada a estar en el intervalo $[y_{min}, y_{max}]$, donde

$$y_{min} = max\{Clase(X)|(x, Clase(X)) \in D \wedge X \leq X_0\}$$
$$y$$
$$y_{max} = min\{Clase(X)|(x, Clase(X)) \in D \wedge X_0 \leq X\}$$

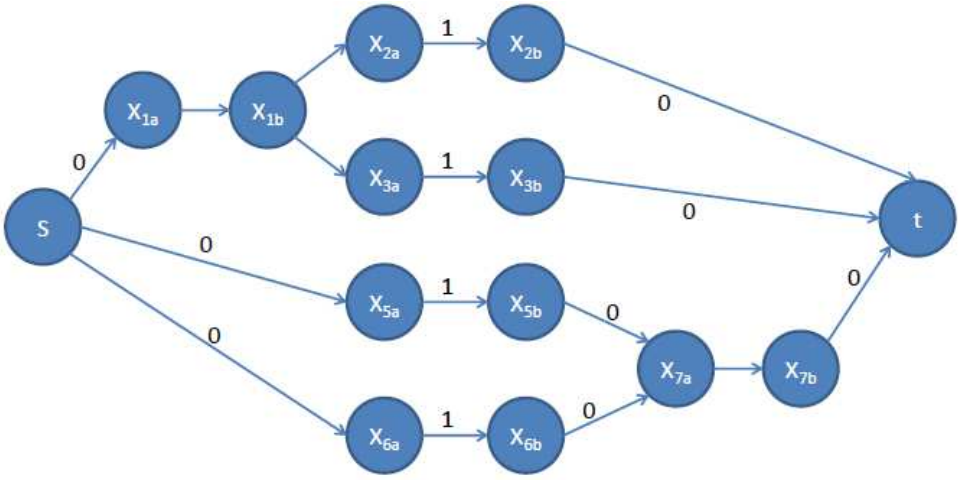


Figura 1.9: Red de transporte sobre un MGV.

donde D es el conjunto de datos reetiquetado. La elección de un valor de este intervalo es libre, sin embargo, y por lo tanto, tiene sentido hacer un mayor uso de los datos observados para guiar esta elección.

Existen dos variantes en la regla del vecino más próximo estándar:

1. Calculamos los k vecinos más cercanos de X_0 en D y predecimos la etiqueta de $[y_{min}, y_{max}]$ que se presenta con mayor frecuencia entre esos k-vecinos. Si ninguna de las k etiquetas es permitida, elegimos al azar entre $[y_{min}, y_{max}]$.

2. Calculamos los k vecinos más cercanos de X_0 en D con etiqueta en $[y_{min}, y_{max}]$ y predecimos la etiqueta por mayoría.

Para ilustrar visualmente la diferencia entre el vecino más cercano estándar y monótono, consideramos un pequeño ejemplo. Supongamos que los datos de entrenamiento se componen de los tres puntos trazados en la Figura 1.10 (se muestran Diagramas de Voronoi). Al lado de cada punto de datos, sus coordenadas (x_1, x_2) y clase. Se puede observar la partición del espacio de entrada de acuerdo con el método del vecino más cercano. Es claro que la regla de asignación resultante no es monótona. En la imagen de la derecha se ha dado la regla de asignación para la predicción del vecino más cercano monótona. Todos los puntos menores que (4,8) no pueden obtener una etiqueta de clase mayor que 1, de forma que la regla de asignación se ha ajustado en consecuencia. Hay que tener en

cuenta que esta regla de asignación no es monótona en general, pero es monótona con respecto a los tres puntos en la muestra de entrenamiento.

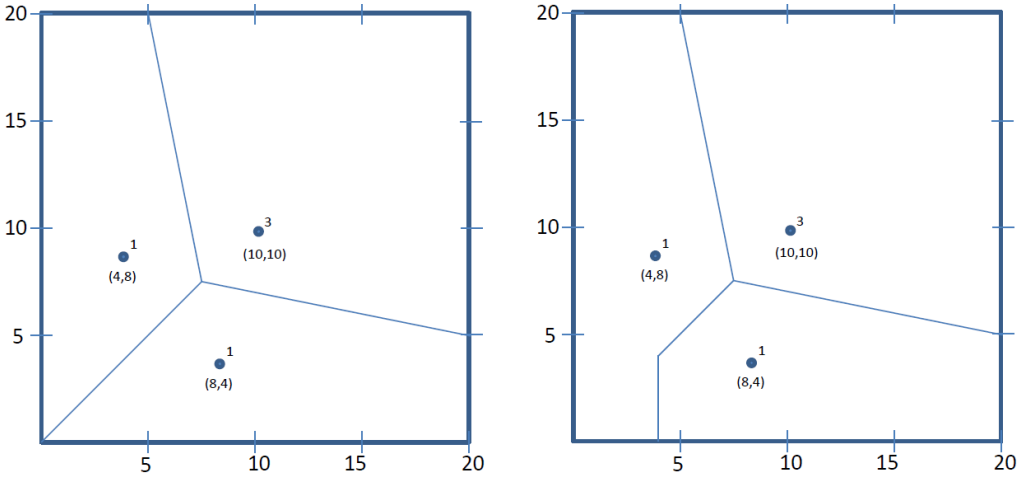


Figura 1.10: Diagramas de Voronoi del vecino más cercano.

1.6. Aprendizaje mediante ejemplos anidados generalizados

El aprendizaje mediante ejemplos anidados generalizados (Nested Generalized Examples, NGE) es un paradigma de aprendizaje basado en ejemplos con clase, donde una hipótesis inducida tiene la forma gráfica de un conjunto de hiperrectángulos en un espacio n dimensional Euclídeo. Los ejemplares para cada clase pueden ser tanto hiperrectángulos como instancias simples [Sal91]. La entrada de un sistema NGE es un conjunto de ejemplos de entrenamiento, cada uno descrito como un vector de parejas *atributo_numérico/valor* y una clase asociada. Los atributos pueden ser tanto numéricos como categóricos. Los atributos numéricos se presentan normalmente normalizados en el intervalo $[0, 1]$.

En NGE, un conjunto inicial de puntos dados en el espacio n dimensional Euclídeo se generaliza en un conjunto más pequeño de hiperrectángulos en forma de elementos contenidos. La elección de qué hiperrectángulo generalizar desde el subconjunto de puntos o desde otros hiperrectángulos y de cómo generalizarlo,

depende del algoritmo concreto de NGE empleado.

En las siguiente secciones, describimos los conceptos esenciales para entender el modelo de aprendizaje NGE, junto a los algoritmos más comunes que pertenecen a este paradigma. Primero, explicamos los conceptos necesarios para entender el funcionamiento de la clasificación con este tipo de técnicas (Sección 1.6.1). Después, se describen los dos algoritmos clásicos basados en hiperrectángulos, BNGE en la Sección 1.6.2.1, y RISE in la Sección 1.6.2.2.

1.6.1. Coincidencia de puntos y clasificación

La coincidencia de puntos es una de las características centrales del aprendizaje por NGE y permite incluso cierta configuración y ajuste, si se desea. De forma general, este proceso calcula la distancia entre un nuevo ejemplo y un objecto almacenado en memoria del modelo (un hiperrectángulo). Para el resto de la sección, nos referiremos al ejemplo a clasificar como E y al hiperrectángulo como H, independientemente de si H está formado por un punto simple o si tiene algún volumen.

El modelo calcula un valor de coincidencia entre E y H midiendo la distancia euclídea entre dos objetos. La distancia euclídea es bien conocida cuando H es un punto simple. En caso contrario, la distancia se calcula como sigue (considerando atributos numéricos):

$$D_{EH} = \sqrt{\sum_{i=1}^{m} \left(\frac{dif_i}{max_i - min_i} \right)^2}$$

donde

$$dif_i = \begin{cases} E_{f_i} - H_{upper} \text{ cuando } E_{f_i} > H_{upper} \\ H_{lower} - E_{f_i} \text{ cuando } E_{f_i} < H_{lower} \\ 0 \text{ en otro caso} \end{cases}$$

m es el número de dimensiones o atributos de los datos, E_{f_i} es el valor del atributo i-ésimo del ejemplo, H_{upper} y H_{lower} son los valores más altos y más bajos de H para un atributo específico y max_i y min_i son los valores máximo y mínimo para el atributo i-ésimo en los datos de entrenamiento, respectivamente.

La distancia medida por esta fórmula es equivalente a la longitud de una línea dibujada perpendicularmente desde el punto E_{f_i} a la superficie, arista o esquina más cercana de H. Nótese que los puntos internos a un hiperrectángulo tienen una distancia de 0 a dicho hiperrectángulo. En el caso de solapamiento de hiperrectángulos, se pueden utilizar varias estrategias para resolver la coincidencia. La más usual es aquella que asocia el punto que cae dentro del hiperrectángulo más pequeño de entre los que solapan a la vez. El tamaño de un hiperrectángulo se define en términos de su volumen. En atributos nominales, la distancia es 0 cuando dos atributos tienen la misma categoría, y 1 en caso contrario. También hay que considerar que un hiperrectángulo que tiene un hipervolumen de menos dimensiones que otro, siempre será menor.

La teoría NGE también se refiere a la existencia de pesos asociados con los atributos en los ejemplos, pero no se suelen considerar porque se pueden usar independientemente a la inducción basada en hiperrectángulos y porque dificulta la interpretabilidad de los modelos obtenidos. Además, en [WD95], los autores notificaron que el uso de pesos no siempre mejora el rendimiento de un algoritmo NGE. De hecho, mostraron que los pasos basados en información mutua podrían ser apropiados en la mayoría de los casos.

1.6.2. Propuesta Clásicas

EACH, BNGE y RISE son las propuestas pioneras para el aprendizaje NGE. EACH no se considera en esta sección porque los autores de BNGE demostraron que su propuesta claramente generaliza y mejora al algoritmo EACH.

1.6.2.1. BNGE: Batch Nested Generalized Exemplar

BNGE es una versión por lotes del primer modelo NGE (también conocido como EACH [Sal91]) y se propuso para arreglar algunas deficiencias presentadas por el primero algoritmo NGE [WD95]. Se cambió la manera incremental de actuación por un modo por lotes y también se añadieron algunas modificaciones en la regla de coincidencia, como la inclusión de un mecanismo para tratar con valores perdidos y el manejo de todos los posibles valores nominales en la definición de un hiperrectángulo. La generalización de un hiperrectángulo se llevo a cabo expandiendo sus fronteras justo hasta cubrir el ejemplo deseado.

BNGE sólo mezcla hiperrectángulos si el nuevo hiperrectángulo generalizado

no cubre (o no solapa con) cualquier otro hiperrectángulo de otra clase. No permite el solapamiento o anidamiento, que son dos debilidades identificadas en el proceso incremental de NGE seguido por EACH.

1.6.2.2. RISE: Unificando la Inducción basada en Instancias y Reglas

RISE [Dom96] es un algoritmo propuesto para vencer algunas de las limitaciones del aprendizaje basado en instancias y la inducción de reglas mediante la unificación de ambos. El algoritmo sigue las mismas indicaciones explicadas anteriormente, pero también introduce algunas mejoras con respecto a los cálculos de las distancias, puesto que se usa la distancia SVDM [WM97] en los atributos nominales. RISE selecciona la regla con la mayor precisión (usando la correción de Laplace existente en numerosas técnicas de inducción de reglas [F̈99]) en vez de elegir la regla más pequeña que cubre el ejemplo.

BNGE y RISE siguen un mecanismo similar para producir hiperrectángulos. Comienzan desde un conjunto de entrenamiento completo e intentan mezclar los ejemplos/hiperrectángulos más cercano mientras la precisión global no se perjudique. RISE usa una metodología *leave-one-out* para calcular el rendimiento y, de forma contraria a BNGE, permite el anidamiento y el solapamiento entre hiperrectángulos.

Algoritmo 3 Constructor del clasificador OSDL

for todas las instancias $(X_h, Clase(X_h))$ **do**
  $AñadirInstancia(X_h, Clase(X_h))$
end for
if $minX \notin DataSet$ **then**
  $AñadirInstancia(minX, minClases(X))$
end if
if $maxX \notin DataSet$ **then**
  $AñadirInstancia(maxX, maxClases(X))$
end if
for todas las instancias $(X_h, Clase(X_h))$ **do**
  $EliminarInstancia(X_h, Clase(X_h))$
  Calculamos los límites de $F(X_h;\cdot)$ y $F_M(X_h;\cdot)$
  Inicializamos el vector error.
  for s' = 0 hasta s' = 10 **do**
    $s \leftarrow s'/10$
    $F_s(X_h,\cdot) = (1-s) * F_m(X_h,\cdot) + s * F_M(X_h,\cdot)$
    $error[s'] = error[s'] + Abs(i - AsignarClase(f_s(X_h,\cdot)))$
  end for
  $AñadirInstancia(X_h, Clase(X_h))$
end for
for s' = 0 hasta s' = 10 **do**
  $R[s'] \leftarrow error[s']/Cardinal(X)$
end for
$s \leftarrow (argmin_{s'=0,\ldots,10} R[s'])/10;$

Capítulo 2

Discusión de los Resultados

Las siguientes secciones resumen y discuten los resultados obtenidos en cada etapa específica de la tesis.

2.1. MoNGEL: Aprendizaje Monotónico basado en Ejemplos Anidados Generalizados

Nos proponemos el uso y la formalización de la aproximación del aprendizaje basado en ejemplos anidados generalizados con restricciones monotónicas, proponiendo el algoritmo MoNGEL. El aprendizaje se efectúa mediante el almacenamiento de objetos en el espacio Euclídeo de n dimensiones que pueden ser o puntos o hiperrectángulos. Este tipo de técnicas hibridan el aprendizaje basado en instancias con el aprendizaje basado en reglas en un modelo combinado.

Se lleva a cabo un análisis experimental sobre una gran colección de conjuntos de datos monotónicos. MoNGEL se compara con otras técnicas basadas en aprendizaje basado en instancias y/o reglas, como el clasificador de vecinos cercanos monotónico (Monotonic k-NN), el *modelo de aprendizaje ordinal* (OLM) y el algoritmo de *aprendizaje ordinal de dominancia estocástica* (OSDL). Los resultados obtenidos se verifican mediante el uso de tests estadísticos no paramétricos y muestran que MoNGEL mejora al resto de técnicas de clasificación monotóni-

ca mencionadas en precisión, error absoluto medio y simplicidad de los modelos construidos. Además, la cuestión clave de nuestra propuesta es que el modelo resultante se compone por ejemplos generalizados que cumplen totalmente entre sí con todas las restricciones monotónicas.

2.2. Selección de Hiperrectángulos para Clasificación Monotónica mediante el uso de Algoritmos Evolutivos

Nos planteamos la selección de los hiperrectángulos más efectivos por medio de la aplicación de algoritmos evolutivos en problemas de clasificación monotónica. La generación de un número óptimo de hiperrectángulos para clasificar un conjunto de puntos es un problema NP-duro. Los algoritmos heurísticos producen normalmente un subconjunto de tamaño excesivo con hiperrectángulos innecesarios. Por tanto, hay una necesidad de seleccionar solo aquellos más influyentes y esto se puede hacer fácilmente con algoritmos evolutivos en etapas de reducción de datos.

La técnica propuesta, denominada EHSMC-CHC, se compara con un exhaustivo análisis experimental que involucra un gran número de conjuntos de datos relativos a problemas reales de clasificación y regresión. Al tratarse de datos reales, se entiende que son datos imperfectos y presentan violaciones de la restricción de monotonicidad en algunos ejemplos. Comparamos nuestra propuesta con otras técnicas de aprendizaje monotónico que pertenecen a ambos paradigmas, como OLM, OSDL, k-NN monotónico y MID (inducción de árboles de decisión monotónicos). El diseño experimental incorpora el uso de tests estadísticos no paramétricos. Los resultados muestran una mejora significativa en la precisión de los modelos que están formados por un conjunto muy reducido de hiperrectángulos. Además, al igual que el algoritmo anterior, el modelo resultante se compone por ejemplos generalizados que cumplen totalmente entre sí con todas las restricciones monotónicas, a pesar de que ahora los datos de entrada presentan imperfecciones.

Capítulo 3

Publicaciones

3.1. MoNGEL: Monotonic Nested Generalized Exemplar Learning

- Estado: En impresión.
- Título: MoNGEL: Monotonic Nested Generalized Exemplar Learning.
- Autores: Javier García, Habib M. Fardoun, Daniyal M. Alghazzawi, José-Ramón Cano y Salvador García.
- Revista: Pattern Analysis and Applications
- ISSN: 1433-7541
- Factor de Impacto (JCR 2015): 1.104
- Cuartiles por Área de Conocimiento:
    - Cuartil 3 en *Computer Science, Artificial Intelligence*, Ranking 80/130

Pattern Anal Applic
DOI 10.1007/s10044-015-0506-y

THEORETICAL ADVANCES

MoNGEL: monotonic nested generalized exemplar learning

Javier García[1] · Habib M. Fardoun[2] · Daniyal M. Alghazzawi[2] · José-Ramón Cano[3] · Salvador García[4]

Received: 27 February 2015 / Accepted: 20 July 2015

Abstract In supervised prediction problems, the response attribute depends on certain explanatory attributes. Some real problems require the response attribute to represent ordinal values that should increase with some of the explaining attributes. They are called classification problems with monotonicity constraints. In this paper, we aim at formalizing the approach to nested generalized exemplar learning with monotonicity constraints, proposing the monotonic nested generalized exemplar learning (MoNGEL) method. It accomplishes learning by storing objects in $\mathbb{R}^n$, hybridizing instance-based learning and rule learning into a combined model. An experimental analysis is carried out over a wide range of monotonic data sets. The results obtained have been verified by non-parametric statistical tests and show that MoNGEL outperforms well-known techniques for monotonic classification, such as ordinal learning model, ordinal stochastic dominance learner and *k*-nearest neighbor, considering accuracy, mean absolute error and simplicity of constructed models.

Keywords Monotonic classification · Instance-based learning · Rule induction · Nested generalized examples

✉ Salvador García
salvagl@decsai.ugr.es

Javier García
jgf00002@red.ujaen.es

Habib M. Fardoun
hfardoun@kau.edu.sa

Daniyal M. Alghazzawi
dghazzawi@kau.edu.sa

José-Ramón Cano
jrcano@ujaen.es

[1] Department of Computer Science, University of Jaén, 23071 Jaén, Spain

[2] Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

[3] Department of Computer Science, EPS of Linares, University of Jaén, Calle Alfonso X El Sabio S/N, 23700 Linares, Jaén, Spain

[4] Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain

1 Introduction

Knowledge extraction from ordinal or ordered concepts has attracted the interest of data mining communities in recent years [36]. An ordinal data set is that with an ordinal output attribute. In the problem of ordinal classification with monotonicity constraints the goal is to predict for a given example one of the ordered class labels [7]. Examples are described by attributes with ordered values and monotonicity constraints are present: a higher value of an attribute of an example, fixing other attributes' values, should not decrease its class assignment.

A common form of prior knowledge in data analysis concerns the monotonicity of relations between the dependent and explanatory variables [32]. As an example, consider a university acceptance procedure and assume that one candidate obtains scores at least as good on all admission criteria as a second candidate. However, the second is admitted while the first is not. It seems obvious that this monotonic relationship is not only evident, but also required in order not to commit to unacceptable admission rules. When traditional classification algorithms are used to build an admission rule from a data set

Published online: 29 July 2015

containing past decisions of admissions/rejections, it may happen that such an anti-monotone choice is obtained.

Two important reasons are at least identified for explaining why knowledge about monotonicity should be exploited in a learning task [8]. Firstly, monotonicity imposes constraints on the prediction function. This decreases the size of the hypothesis space and also the complexity of the model. Secondly, the domain experts decide the acceptance or rejection of the models yielded if they are consistent with the domain knowledge, regardless of their accuracy.

The monotonicity constraint is very common in practice and many data mining algorithms have been adapted to be able to handle such constraints in one form or another. There are two steps for dealing with monotonic classification problems. The first one is the "monotonization" of the data set [38], which excludes the examples that violate the monotonic restrictions; and the second one is imposing constraints for learning only monotone classification functions. Proposals of this type are: classification trees and rule induction [6, 10, 13, 19, 29, 31, 37], neural networks [12, 20], and instance-based learning [5]. Monotonic classification frequently arises in social sciences, risk analysis, medicine, finance, information retrieval, education, etc. [11]

Exemplar-based learning [2] considers a set of methods widely used in machine learning and data mining [28]. The most famous algorithm belonging to this family is the k-nearest neighbor (k-NN) and derivatives [15]. A similar scheme is based on the nested generalized exemplar (NGE) theory. It was introduced by [39] and proposes several significant updates to the exemplar-based learning model. A major one is retaining the concept of storing verbatim examples in memory but also allowing examples to be generalized.

In NGE theory, generalizations take the form of hyperrectangles in $\mathbb{R}^n$ [16, 41]. With respect to instance-based classification [1], these generalizations increase the understanding of the data stored and the achievement of a substantial compression of the data, reducing the storage requirements. Considering rule induction [21], the ability of modeling decision surfaces by hybridizations between distance-based methods (Voronoi diagrams) and parallel axis separators could improve the performance of the models in domains with clusters of exemplars or exemplars strung out along a curve. It has also shown promising results in other learning domains, such as imbalanced learning [25, 35], and data reduction processes [18, 24, 26].

In this paper, we propose the usage, for the first time in the literature, of NGE learning in monotonic classification tasks, aiming at increasing accuracy for this type of problem by means of generating monotone generalized examples to perform the classification rule. We compare our approach with other monotonic instance-based learning models, such as monotonic k-NN [17], ordinal learning model (OLM) [5] and ordinal stochastic dominance learner (OSDL) [33]. The empirical study has been checked using non-parametrical statistical testing [14, 22, 23]. The results show an improvement in accuracy whereas the number of examples stored in the final subset is further reduced. Apart from this, the main key point of our proposal is that the model built is composed of generalized examples which all fulfill the monotonicity constraints.

The paper is organized as follows. In Sect. 2, we present some background concepts: the ordinal classification with monotonic constraints, previous approaches and NGE learning. Section 3 is devoted to describing our proposal of NGE and its adaptation to satisfy the monotonicity constraints. Section 4 describes the experimental framework, and Sect. 5 examines the results obtained in the empirical study and presents a discussion and analysis. Finally, Sect. 6 concludes the paper.

2 Background

We will do a brief review of the monotonic classification including the description of instance-based techniques in Sects. 2.1 and 2.2. The NGE classification will be described in Sect. 2.3.

2.1 Monotonic classification

Ordinal classification problems are those in which the class is neither numeric nor nominal. Instead, the class values are ordered. For instance, a worker can be described as ''excellent'', ''good'' or ''bad'', and a bond can be evaluated as ''AAA'', ''AA'', ''A'', ''A-'', etc. Similar to a numeric scale, an ordinal scale has an order, but it does not possess a precise concept of distance. Ordinal classification problems are important, since they are fairly common in our daily life. Employee selection and promotion, determining credit rating, bond rating, economic performance of countries, industries and firms, and insurance underwriting, are examples of ordinal problem solving in business. Rating manuscripts, evaluating lecturers, student admissions, and scholarships decisions for students, are examples of ordinal decision making in academic life. Ordinal problems have been investigated in scientific disciplines such as information retrieval, psychology, and statistics for many decades [8].

A monotonic classifier is one that will not violate monotonicity constraints. Informally, the monotonic classification implies that the assigned class values are monotonically non-decreasing (in ordinal order) with the attribute values. More formally, let $\{\mathbf{x}_i, \text{class}(\mathbf{x}_i)\}$ denote a

set of examples with attribute vector $\mathbf{x}_i = (\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,n})$ and a class, $\text{class}(\mathbf{x}_i)$, being m the number of instances and n the number of attributes. Let $\mathbf{x}_i \succeq \mathbf{x}_h$ if $\forall_{j=1,\ldots,n}, \mathbf{x}_{i,j} \geq \mathbf{x}_{h,j}$.

A data set $\{\mathbf{x}_i, \text{class}(\mathbf{x}_i)\}$ is monotonic if and only if all the pairs of examples i, h are monotonic with respect to each other [6] (see Eq. 1).

$$\mathbf{x}_i \succeq \mathbf{x}_h \Rightarrow \text{class}(\mathbf{x}_i) \geq \text{class}(\mathbf{x}_h), \quad \forall_{i,h}. \tag{1}$$

Some monotonic ordinal classifiers require monotonic data sets to successfully learn, although there are others that are capable of learning from anti-monotonic data sets as well.

2.2 Monotonic instance-based learning proposals

Three instance-based learning methods are pioneers and well known in the field of monotonic classification. In the following, we will discuss each one in detail.

- The OLM [5] is a very simple algorithm that learns ordinal concepts by eliminating anti-monotonic pairwise inconsistencies. The generated concepts can be viewed as rules. During the learning phase, each example is checked against every rule in a rule base, which is initially empty. If an example is inconsistent with a rule in the rule base, one of them is selected at random while the other is discarded, but if the example is selected, it must be checked for consistency against all the other monotonicity rules. If it passes this consistency test, it is added as a rule. Consequently, the rule base is kept monotonic at all times. Classification is done conservatively. All the rules are checked in decreasing order of class values against an attribute vector, and the vector is classified as the class of the first rule that covers it. If such a rule does not exist, the attribute vector is assigned the lowest possible class.
- As its name implies, OSDL [33] is based on the concept of ordinal stochastic dominance (OSD). The rationale behind OSD can be given through an example: in life insurance, one may expect a stochastically greater risk to the insurer from older and sicker applicants than from younger and healthier ones. Higher premiums should reflect greater risks and vice versa. There are several definitions of stochastic ordering. The stochastic order computes when a random variable is bigger than another. Considering this order, stochastic dominance can be established as a form of stochastic order. In this case, a probability distribution over possible predictions can be ranked. The ranking depends of the nature of the data set. Stochastic dominance refers to a set of relations that may hold between a pair of distributions. The most commonly used, as stochastic ordering, is first OSD, which was used by [9] in the OSDL. For each vector $\mathbf{x}_i$, the OSDL computes two mapping functions: one that is based on the examples that are stochastically dominated by $\mathbf{x}_i$ with the maximum label (of that subset), and the second is based on the examples that cover (i.e., dominate) $\mathbf{x}_i$, with the smallest label. Later, an interpolation between the two class values (based on their position) is returned as a class.
- The monotonic k-NN was proposed in Duivesteijn and Feelders [17]. This method consists of two steps. In the first step, the training data are made monotone by relabeling as few cases as possible. This relabeled data set may be considered as the monotone classifier with the smallest error index in the training data. In the second step, we use a modified nearest neighbor rule to predict the class labels of new data so that violations of the restrictions of monotonicity will not occur. Considering the monotonic nearest neighbor rule, the class label assigned to a new data point $\mathbf{x}_0$ must lie in the interval $[\text{class}_{\min}, \text{class}_{\max}]$, where

$$\text{class}_{\min} = \max\{\text{class}(\mathbf{x}_i) | \{\mathbf{x}_i, \text{class}(\mathbf{x}_i)\} \wedge \mathbf{x}_i \leq \mathbf{x}_0\} \tag{2}$$

and

$$\text{class}_{\max} = \min\{\text{class}(\mathbf{x}_i) | \{\mathbf{x}_i, \text{class}(\mathbf{x}_i)\} \wedge \mathbf{x}_0 \leq \mathbf{x}_i\} \tag{3}$$

where $\{\mathbf{x}_i, \text{class}(\mathbf{x}_i)\}$ is the monotone data set. To conserve the monotonicity the choice of the class value for $\mathbf{x}_0$ must be in this interval.

2.3 NGE learning

NGE is a learning paradigm based on class exemplars, where an induced hypothesis has the graphical shape of a set of hyperrectangles in $\mathbb{R}^n$. Exemplars of classes are either hyperrectangles or single instances [39]. The input of an NGE system is a set of training examples, each described as a vector of pairs *numeric_attribute/value* and an associated class. Attributes can either be numerical or categorical. Numerical attributes are usually normalized in the [0, 1] interval.

In NGE, an initial set of hyperrectangles in $\mathbb{R}^n$ formed by single points directly taken from the data is generalized into a smaller set of hyperrectangles regarding the elements that it contains. Choosing which hyperrectangle is generalized from a subset of points or other hyperrectangles and how it is generalized depend on the concrete NGE algorithm employed.

The matching process is one of the central features in NGE learning and it allows some customization, if desired. Generally speaking, this process computes the distance between a new example and an exemplar memory object (a generalized example). Let the example to be classified be termed as $\mathbf{x}_0$ and the generalized example as G, regardless

of whether G is formed by a single point or a hyperrectangle if it has some volume.

The model computes a match score between $\mathbf{x}_0$ and G by measuring the Euclidean distance between two objects. The Euclidean distance is well known when G is a single point. Otherwise, the distance is computed as follows (considering numerical attributes):

$$D_{\mathbf{x}_0 G} = \sqrt{\sum_{j=1}^{n} \left(\frac{\mathrm{dif}_j}{\max_j - \min_j} \right)^2} \tag{4}$$

$$\mathrm{dif}_j = \begin{cases} \mathbf{x}_{0,j} - \max(G_j) & \text{when } \mathbf{x}_{0,j} > \max(G_j), \\ \min(G_j) - \mathbf{x}_{0,j} & \text{when } \mathbf{x}_{0,j} < \min(G_j), \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where m is the number of attributes of the data, $\mathbf{x}_{0,j}$ is the value of the jth attribute of the example $\mathbf{x}_0$, $\max(G_j)$ and $\min(G_j)$ are the maximum and minimum values of G for the jth attribute and $\max_j$ and $\min_j$ are the maximum and minimum values for jth attribute in training data, respectively.

The distance measure represents the length of a line dropped perpendicularly from the point $\mathbf{x}_0$ to the nearest surface, edge or corner of G. Note that internal points in a hyperrectangle have a distance of 0 to that rectangle. In the case of overlapping rectangles, several strategies could be implemented, but it is usually accepted that a point falling in the overlapping area belongs to the smaller rectangle (the size of a hyperrectangle is defined in terms of volume). The volume is computed following the indications given in Wettschereck and Dietterich [41]. In nominal attributes, the distance is 0 when two attributes have the same categorical label, and 1 on the contrary.

3 Monotonic nested generalized exemplar learning: MoNGEL

The method proposed in this paper, named monotonic nested generalized exemplar learning (MoNGEL), is thoroughly explained in this section. This method builds a model obtaining a set of rules which do not violate the monotonicity restrictions from the set of training instances trying to preserve as much as possible the behavior of the algorithms based on instances. First, in Sect. 3.1 we explain some definitions required to understand the proposed model, and after, in Sect. 3.2 we describe the phases of the building of the model.

3.1 Definitions

We represent each instance with $\mathbf{x}_i = (\mathbf{x}_{i,1}; \mathbf{x}_{i,2}; \ldots; \mathbf{x}_{i,n}; \mathrm{class}(\mathbf{x}_i))$, where $\mathbf{x}_{i,j}$ is the input value of the instance i in the jth attribute, n is the total number of input attributes and $\mathrm{class}(\mathbf{x}_i)$ is the class or output value of the instance i. Let the rules be denoted by $R : (\prod_j A_j; C)$, where A_j is a condition belonging to the antecedent and C is the response. A formal definition is given next:

$$\begin{aligned} & R_{:} : A_1 \times A_2 \times \cdots \times A_n \Rightarrow C, \\ \text{with} \quad & A_j = \begin{cases} [j_{\min}, j_{\max}] & \text{if numerical,} \\ \{\alpha, \beta, \ldots\} & \text{if nominal,} \end{cases} \\ & C = \text{a value of the class,} \end{aligned} \tag{6}$$

where $j_{\min}$ and $j_{\max}$ are the minimum and maximum boundaries for the condition j, respectively, and $\alpha, \beta, \ldots$ are the possible categorical values belonging to the domain of the jth attribute, if it is nominal.

The rules should not break monotonicity in the sense we saw in Sect. 2.1. Hence, we define a partial order relation in the set of disjoint rules (where R and R' are disjoint if $\exists j, A_j \bigcap A'_j = \emptyset$), deciding when a condition is higher than, lower than or equal to another:

$$\begin{aligned} A_j \preceq A'_j & \text{ if } j\text{th attribute is numeric and } j_{\max} \leq j'_{\min} \\ & \text{ or if } j\text{th attribute is nominal and } A_j \subseteq A'_j, \\ A_j \succeq A'_j & \text{ if } j\text{th attribute is numeric and } j_{\max} \geq j'_{\min} \\ & \text{ or if } j\text{th attribute is nominal and } A'_j \subseteq A_j, \\ A_j = A'_j & \text{ if } j\text{th attribute is numeric and } j_{\min} = j'_{\min} \text{ and } j_{\max} = j'_{\max} \\ & \text{ or if } j\text{th attribute is nominal and } A_j \equiv A'_j. \end{aligned} \tag{7}$$

Then let two rules R and R' be defined as:

$$\begin{aligned} R \preceq R' \quad & \text{if} \quad A_j \preceq A'_j \quad \forall j \quad \text{and} \quad \exists l, 1 \leq l \leq n, A_l < A'_l, \\ R = R' \quad & \text{if} \quad A_j = A'_j \quad \forall j, \\ R \succeq R' \quad & \text{if} \quad A_j \succeq A'_j \quad \forall j \quad \text{and} \quad \exists l, 1 \leq l \leq n, A_l > A'_l. \end{aligned} \tag{8}$$

Two rules are comparable if $R \preceq R'$ or $R \succeq R'$, and non-comparable in contrary case, since we cannot establish an order between them. Hence, a rule $R : (\prod_j A_j, C)$ will be anti-monotonic with respect to another $R' : (\prod_j A'_j, C')$ if:

$$\begin{aligned} R \preceq R' \quad & \text{and} \quad C > C' \quad \text{or} \\ R \succeq R' \quad & \text{and} \quad C < C' \quad \text{or} \\ R = R' \quad & \text{and} \quad C \neq C'. \end{aligned} \tag{9}$$

The distance between an instance $\mathbf{x}_i$ and a rule R is defined as follows:

$$D_{\mathbf{x}_i R} = \sqrt{\sum_{j=1}^{n} \left(\frac{\mathrm{dis}_j}{\mathrm{Range}} \right)^2}, \tag{10}$$

where dis_j and Range are defined differently depending on the type of the jth attribute. If the jth attribute is numeric:

$$\text{Range} = \max_j - \min_j,$$
$$\text{dis}_j = \begin{cases} \mathbf{x}_{i,j} - j_{\max} & \text{if } \mathbf{x}_{i,j} > j_{\max}, \\ j_{\min} - \mathbf{x}_{i,j} & \text{if } \mathbf{x}_{i,j} < j_{\min}, \\ 0 & \text{otherwise.} \end{cases} \quad (11)$$

If the jth attribute is nominal:

$$\text{dis}_j = \begin{cases} 0 & \mathbf{x}_{i,j} \in A_j, \\ 1 & \text{otherwise,} \end{cases}$$
$$\text{Range} = \text{Number of possible values of the } j\text{th attribute.} \quad (12)$$

where $\mathbf{x}_{i,j}$ is the value of the jth attribute of the instance i, $j_{\max}$ and $j_{\min}$ are the maximum and minimum values of the rule R in the jth attribute and $\max_j$ and $\min_j$ are the maximum and minimum values of the jth attribute in the data set.

The distance between two rules R and R' is defined as:

$$D_{RR'} = \sqrt{\sum_{j=1}^{n} \left(\frac{\text{dis}_j}{\text{Range}} \right)^2}, \quad (13)$$

where dis_j and Range are defined as:

$$\text{dis}_j = \left| \frac{j_{\max} + j_{\min}}{2} - \frac{j'_{\max} + j'_{\min}}{2} \right|, \quad (14)$$
$$\text{Range} = \max_j - \min_j,$$

if the jth attribute is numeric and

$$\text{dis}_j = 1 - \frac{\#(A_j \bigcap A'_j)}{\#(A_j \bigcup A'_j)}, \quad (15)$$

if the jth attribute is nominal. The geometrical interpretation of the distances is given next. The measured distance between an instance and a rule is equivalent to the length of a line dropped perpendicularly from the instance to the nearest surface, edge or corner of the rule. Furthermore, the measured distance between two rules is the real distance between the average points of each condition by considering numeric attributes (Eq. 14) and the dissimilarity between each condition by considering nominal attributes (Eq. 15).

3.2 Model building

Our algorithm follows the steps of the batch operating form of NGE studied in Wettschereck and Dietterich [41]. In summary, firstly all the instances are transformed to zero-dimensional rules, formed by a single point, and are gathered to obtain an initial set of rules. Secondly, for every rule, the comparable rule of the same class and with minimum distance is searched for within the set of rules. Each rule is then iteratively generalized (or expanded) to cover the nearest rule until a non-comparable rule or a rule with a different class value is found. Finally, an anti-monotonic test is conducted aiming at removing those rules which are anti-monotonic to each other, trying to remove the smallest number of rules.

In the initialization phase of the rules set, all instances are transformed to rules of dimension zero (the limits of the numerical ranges are equal and nominal sets consist of a single value). Then, the rules are sorted by their class value in descending order to discern the subset of rules for each class. Finally, the repeated rules are removed from the rules set with the goal of ignoring the repeated examples from the training set. We can see this process in more detail in Algorithm 1.

Algorithm 1 Initialization phase

S, Set of Rules.
$S = \emptyset$.
For each instance $\mathbf{x}_i$ of the training set
  $\mathbf{x}_i$ is transformed to a rule R_i of dimension 0.
  $S = S \bigcup R_i$.
Sort the rules in S by its class value in descending order, obtaining the subset S_C for each class.
Remove the repeated rules in all S_C.

Algorithm 2 Generalization phase

Repeat
  For each class C
    Randomize the order of presentation of rules for class
    For each rule $R_i \in S_C$
      Find the nearest rule $R'_i \in S_C$ such that it is comparable with R_i.
      If R'_i exists
        $R''_i = Generalize(R_i, R'_i)$.
        If for all $R_j \in S_C$, R_j and R''_i are disjoints
          Delete R'_i.
          Replace R_i by R''_i.
Until no generalization is done.

In the next phase, known as the generalization phase, all the rules are checked to find that which is comparable, has the same class value and has the minimum distance. If it exists, it will be replaced by a generalized rule which covers both, provided that the new generalized rule does not overlap with any existing rule of the same class. This process is then repeated until there is no possibility of generalizing anymore. This phase is sensitive to the order of presentation of instances, hence the algorithm incorporates a randomization process on the order of presentation of instances for each class. The procedure is depicted in Algorithm 2.

The *Generalize* procedure receives the rules $R : (\prod_j A_j, C)$ and $R' : (\prod_j A'_j, C)$ as input, and computes an output rule $R'' : (\prod_j A''_j, C)$ given by:

$$R'' : A''_1 \times A''_2 \times \cdots \times A''_n \Rightarrow C,$$
$$\text{with} \quad A''_j = \begin{cases} [\min(j_{\min}, j'_{\min}), \max(j_{\max}, j'_{\max})] & \text{if numerical,} \\ A_j \bigcup A'_j & \text{if nominal.} \end{cases} \quad (16)$$

Finally, in the last phase called removal of anti-monotonic rules, we construct a monotonicity violation matrix $N = n_{ip}$

where $n_{ip} = 1$ if the rule i is anti-monotonic with the rule p and $n_{ip} = 0$ if the rule i is not anti-monotonic with the rule p. It is worth mentioning that not to be anti-monotonic is not equal to being monotonic, because two rules could also not be comparable. Afterwards, we calculate the number of violations of each of the rules and register this value in a list $V = v_i$. Once we have computed v_i for all rules, we find the maximum value of the list V, $v_{\max}$, and remove the rule R_i that corresponds to $v_{\max}$. In case of a tie, the rule covering fewer instances is removed. In the case of a next tie in terms of number of instances, a random chosen rule is then removed. Afterwards, both the matrix N and the vector V are updated and the process is repeated until no violations of monotonicity are registered; that is, until $v_i = 0, \forall i$. The detailed procedure is illustrated in Algorithm 3.

Algorithm 3 Elimination of anti-monotonic rules

Construct the monotonicity violation matrix N, and the list V.
While i exists such that $v_i > 0$
  Obtain the row having the greatest number of monotonicity violations in V.
  In case of a tie, locate that row representing the rule which covers the smallest number of instances: p. If there are more than one row, choose one randomly.
  Remove the row p and column p in N.
  Update the list V according to N.

Once the set of rules is built, the classification is performed by computing the distance of the instance with all the rules stored. The nearest rule makes the decision of class using its output value C. In case of a tie, or when two or more rules have a distance of zero to the instance to classify, choose the rule with smaller volume (according to the initial motivation of NGE learning which attempts to learn generalizations with exceptions [39]), which is calculated with the following formula:

$$V(R) = \prod_j L_j, \tag{17}$$

where L_j is computed for each condition as

$$L_j = \begin{cases} j_{\max} - j_{\min} & \text{if numerical and } j_{\max} \neq j_{\min}, \\ 1 & \text{if numerical and } j_{\max} = j_{\min}, \\ \dfrac{\#(A_j)}{|A_j|} & \text{if nominal,} \end{cases} \tag{18}$$

where $\#(A_j)$ is the number of categorical elements covered by the rule in the jth attribute and $|A_j|$ is the number of possible values of the jth attribute (equal to Range used previously).

4 Experimental framework

This section exhibits the methodology used in the experimental study that compares our proposal MoNGEL with other monotonic learning approaches: OLM, OSDL and k-NN. We will explain the configuration of the experiment: monotonic data sets and parameters for the algorithms together with the performance measures and statistical tests applied. To allow the readers to directly experiment with the proposal and reproduce the results achieved, we have created a webpage associated to this paper including the data sets with same partitions, description of parameters used for the algorithm and source code of the MoNGEL proposal: http://www4.ujaen.es/~jrcano/Research/MoNGEL/index.html.

4.1 Data sets and parameters

In this study, the classifiers are run over 20 data sets from the KEEL-data set and UCI repositories [3, 4], four classical ordinal classification data sets from [7] and firstly used in pioneer papers on the topic (BD), and another three from regression problems which were transformed into classification problems by discretizing the class value in [17], maintaining the same order. In the former 20 standard classification data sets, the class attribute is transformed into an ordinal attribute, assigning each category a number in such way that the number of pairs of anti-monotonic examples is minimized.

Table 1 summarizes the data involved in this study and presents, for each data set, the number of instances (#Ins), number of attributes (#Att), number of numerics attributes (#Nu), number of nominals attributes (#No), number of class values (#CL) and the source (#Sou). The data sets considered are partitioned using the tenfold cross validation procedure. The parameters of the used algorithms are presented in Table 2.

The choice of parameters has been done according to the standards and recommendations given by the authors in the original proposal papers. Due to the fact that the experimental evaluation comprises a high number of data sets, it is unreasonable to adjust each parameter individually for each data set. In fact, our purpose is just the opposite; we aim at comparing them in the most general scenario possible. We have not performed any tuning to adapt these parameters, because our objective is not to maximize the accuracy or any other performance metric, but to fairly compare the algorithms and their robustness in a common environment and upon different data sets. Furthermore, our proposal does not require to set any parameter, so it can be considered as a real advantage regarding to the contestant algorithm.

4.2 Performance measures

A collection of several evaluation metrics has been considered:

- Accuracy is the number of successful hits relative to the total number of classifications. It has been by far the

Table 1 Datasets

Dataset	#Ins	#Att	#Nu	#No	#CL	#Sou
Australian	690	14	8	6	2	KEEL
Auto-mpg4classes	392	6	6	0	4	REGR
Automobile	159	25	15	10	6	KEEL
Bands	365	19	19	0	2	KEEL
Bostonhousing4classes	506	12	12	0	4	REGR
Cleveland	297	13	13	0	5	KEEL
Dermatology	358	34	34	0	6	KEEL
Era	1000	4	4	0	9	BD
Esl	488	4	4	0	9	BD
Glass	214	9	9	0	7	KEEL
Heart	270	13	13	0	2	KEEL
Hepatitis	80	19	19	0	2	KEEL
Housevotes	232	16	0	16	2	KEEL
Ionosphere	351	34	34	0	2	KEEL
Iris	150	4	4	0	3	KEEL
Lev	1000	4	4	0	5	BD
Machinecpu4classes	209	6	6	0	4	REGR
Mammographic	830	5	0	5	2	KEEL
Newthyroid	215	5	5	0	3	KEEL
Pima	768	8	8	0	2	KEEL
Saheart	462	9	8	1	2	KEEL
Segment	2310	19	19	0	7	KEEL
Sonar	208	60	60	0	2	KEEL
Swd	1000	10	10	0	4	BD
Titanic	2201	3	3	0	2	KEEL
Vowel	990	13	13	0	11	KEEL
Wine	178	13	13	0	3	KEEL
Wisconsin	683	9	9	0	2	KEEL

Table 2 Parameters considered for the algorithms

Algorithm	Parameters
OLM	ModeResolution = conservative
	ModeClassification = conservative
OSDL	CassificationType = media; balanced = no
	Weighted = no; TuneInterpolationParameter = no
	LowerBound = 0; UpperBound = 1
k-NN	NeighborNumber = 3; Neigborhood = INRANGE
	DistanceType = Euclidean
MoNGEL	It has no parameters to be fixed

most commonly used metric for assessing the performance of classifiers for years [30].

- Mean absolute error (MAE) is calculated as the sum of the absolute values of the errors and then dividing it by the number of classifications. The errors between the real label and the predicted label are estimated by the difference between the ordinal class values. It has been selected considering the studies by [27], which conclude that MAE is one of the best performance metrics in ordered classification.
- Anti-monotonic Index (NMI) is computed as the rate of number of violations of monotonicity divided by the total number of pairs of examples in a data set.
- Number of rules is the number of elements (rules, generalized examples or simple instances) forming the model used for classification of unseen cases.

4.3 Statistical tests for evaluation

Several hypothesis testing procedures are considered to determine the most relevant differences found between the methods [14, 22]. To address this, the use of non-parametric tests will be preferred to parametric ones, since the initial conditions that guarantee the reliability of the latter may not be satisfied, causing the statistical analysis to lose credibility. We will use the Wilcoxon and the Friedman ranks tests [23], to contrast the behavior of each algorithm. They are used to highlight the existence of significant differences between methods.

5 Results and analysis

This section shows the results obtained in the experimental study as well as the analysis based on them. Tables 3 and 4 report the results measured by accuracy, MAE and NMI in test data and the number of rules for each approach considered in this paper, except for the OSDL algorithm. The best case in each data set is stressed in bold. The last row in each table shows the average considering all data sets.

At a glance, observing Tables 3 and 4 we can make the following analyses:

- The MoNGEL proposal yields the best average result in accuracy over test data. It is slightly more accurate than *k*-NN and much more accurate than OLM and OSDL.
- The *k*-NN proposal gets the best average result in MAE over test data. MoNGEL obtains a very close result to *k*-NN, but it is clearly better than OLM and OSDL.
- The MoNGEL proposal obtains the best average result in NMI over test data. The value obtained is slightly lower than that obtained by OLM and much higher than that reported by *k*-NN and OSDL.
- The OLM proposal achieves the best average result in the number of rules over test data. However, MoNGEL is very close to OLM in this aspect and much better than *k*-NN and OSDL, which use the whole training set to classify.

Table 3 Accuracy and MAE reported by all algorithms in all data sets

Datasets	Accuracy								MAE							
	OLM		OSDL		KNN		MoNGEL		OLM		OSDL		KNN		MoNGEL	
	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std
Australian	0.7232	0.0620	**0.8319**	0.0646	0.7043	0.0464	0.6797	0.1557	0.2768	0.0620	**0.1681**	0.0646	0.2957	0.0464	0.3203	0.1557
Auto-mpg4classes	0.6812	0.0571	0.3854	0.0553	**0.6859**	0.0571	0.6119	0.0661	**0.3879**	0.0797	0.8291	0.0741	0.3962	0.1030	0.5414	0.0906
Automobile	0.2333	0.1327	0.3758	0.1168	0.6921	0.0698	**0.8308**	0.0881	2.1046	0.4104	0.8958	0.2259	0.4783	0.1356	**0.2508**	0.1242
Bands	0.3807	0.0159	0.6274	0.0459	0.6439	0.0380	**0.6764**	0.0578	0.6193	0.0159	0.3726	0.0459	0.3561	0.0380	**0.3236**	0.0578
Bostonhousing4classes	0.3003	0.0247	0.2569	0.0111	**0.5616**	0.0709	0.5019	0.0608	1.3045	0.0773	1.0099	0.0332	**0.6238**	0.1182	0.8261	0.0943
Cleveland	**0.5793**	0.0490	0.5421	0.0604	0.5595	0.0622	0.5353	0.0830	0.8311	0.1243	0.7848	0.1361	**0.6823**	0.1174	0.8159	0.1873
Dermatology	0.4499	0.0474	0.1593	0.0544	0.8073	0.0487	**0.8128**	0.0559	1.3821	0.1193	1.6421	0.1372	0.4665	0.1297	**0.3858**	0.1313
Era	**1.0000**	0.0000	**1.0000**	0.0000	**1.0000**	0.0000	**1.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000
Esl	0.9179	0.0228	0.9364	0.0324	**0.9384**	0.0402	0.9365	0.0322	0.0923	0.0284	**0.0656**	0.0341	0.0719	0.0492	0.0738	0.0411
Glass	0.3175	0.0243	0.3223	0.0222	**0.6732**	0.0877	0.6686	0.0805	1.7994	0.0461	1.8000	0.1764	**0.6409**	0.2160	0.6435	0.1382
Heart	0.6704	0.0767	0.6259	0.0672	0.7296	0.0952	**0.7667**	0.0938	0.3296	0.0767	0.3741	0.0672	0.2704	0.0952	**0.2333**	0.0938
Hepatitis	0.2375	0.0875	0.8000	0.0612	0.8000	0.1392	**0.8125**	0.1008	0.7625	0.0875	0.2000	0.0612	0.2000	0.1392	**0.1875**	0.1008
Housevotes	0.9047	0.0696	0.9096	0.0450	**0.9486**	0.0420	0.9005	0.0519	0.0953	0.0696	0.0904	0.0450	**0.0514**	0.0420	0.0995	0.0519
Ionosphere	0.6580	0.0346	0.7237	0.0921	0.7603	0.0784	**0.7917**	0.0760	0.3420	0.0346	0.2763	0.0921	0.2397	0.0784	**0.2083**	0.0760
Iris	0.9000	0.0447	0.3733	0.0442	**0.9867**	0.0267	0.9533	0.0521	0.1000	0.0447	0.9067	0.1200	**0.0133**	0.0267	0.0467	0.0521
Lev	**1.0000**	0.0000	**1.0000**	0.0000	**1.0000**	0.0000	**1.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000
Machinecpu4classes	0.6267	0.0877	0.6362	0.1031	**0.6893**	0.1222	0.6562	0.1263	0.5031	0.1279	0.4067	0.1226	**0.4062**	0.1698	0.5107	0.2137
Mammographic	**0.9892**	0.0114	0.9831	0.0154	0.9843	0.0143	0.9855	0.0072	**0.0108**	0.0114	0.0169	0.0154	0.0157	0.0143	0.0145	0.0072
Newthyroid	0.6223	0.0751	0.1818	0.0403	0.8002	0.0586	**0.8271**	0.0534	0.5504	0.1367	0.8413	0.0638	0.2879	0.0669	**0.2476**	0.0829
Pima	0.8151	0.0403	0.6224	0.0147	**0.8515**	0.0253	0.8321	0.0343	0.1849	0.0403	0.3776	0.0147	**0.1485**	0.0253	0.1679	0.0343
Saheart	0.6862	0.0322	0.6839	0.0862	0.6905	0.0637	**0.6950**	0.0548	0.3138	0.0322	0.3161	0.0862	0.3095	0.0637	**0.3050**	0.0548
Segment	0.3061	0.0176	0.1684	0.0060	0.9645	0.0102	**0.9710**	0.0061	2.4022	0.0576	2.8597	0.1965	0.0684	0.0164	**0.0515**	0.0096
Sonar	0.4662	0.0162	0.5724	0.0764	0.8267	0.0687	**0.8645**	0.0842	0.5338	0.0162	0.4276	0.0764	0.1733	0.0687	**0.1355**	0.0842
Swd	**1.0000**	0.0000	**1.0000**	0.0000	**1.0000**	0.0000	**1.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000
Titanic	**1.0000**	0.0000	**1.0000**	0.0000	**1.0000**	0.0000	**1.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000	**0.0000**	0.0000
Vowel	0.0909	0.0072	0.0859	0.0081	0.9788	0.0071	**0.9949**	0.0068	5.0000	0.0000	4.8273	0.2312	0.0444	0.0240	**0.0101**	0.0156
Wine	0.3484	0.0341	0.3314	0.0284	**0.8088**	0.0798	0.7301	0.1024	0.9660	0.0734	0.9667	0.0759	**0.2696**	0.1210	0.3369	0.1401
Wisconsin	0.8815	0.0412	0.9547	0.0188	0.9781	0.0198	**0.9796**	0.0227	0.1185	0.0412	0.0453	0.0188	0.0219	0.0198	**0.0204**	0.0227
Average	0.6352	0.0397	0.6104	0.0418	**0.8237**	0.0490	0.8220	0.0555	0.7504	0.0648	0.7322	0.0791	**0.2333**	0.0687	0.2413	0.0736

Table 4 NMI and number of rules reported by all algorithms in all data sets

Datasets	NMI								Number of rules					
	OLM		OSDL		KNN		MoNGEL		OLM		KNN		MoNGEL	
	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std
Australian	29.41	1.69	28.05	4.44	21.53	5.37	**18.10**	15.24	**2.1**	0.30	621	0.00	47.9	3.33
Auto-mpg4classes	5.03	1.61	**3.74**	1.20	4.77	1.53	4.39	1.59	**160**	5.92	352	0.00	243.9	3.27
Automobile	0.17	0.33	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**113.3**	2.19	143	0.00	120.2	2.27
Bands	**0.00**	0.00	**0.00**	0.00	0.03	0.09	0.03	0.09	**323.6**	1.20	328	0.00	324.4	1.02
Bostonhousing4classes	0.07	0.07	**0.00**	0.00	0.10	0.11	0.04	0.07	**418**	3.71	455	0.00	433.7	1.95
Cleveland	**0.87**	0.62	1.06	0.51	1.08	0.55	1.31	0.70	**199**	2.83	267	0.00	230.8	1.78
Dermatology	0.23	0.20	0.32	0.19	**0.18**	0.20	**0.18**	0.14	**276.1**	3.08	322	0.00	289.5	2.01
Era	**12.98**	2.24	**12.98**	2.24	**12.98**	2.24	**12.98**	2.24	**33**	0.00	900	0.00	34	0.00
Esl	64.49	3.22	64.70	3.14	64.40	3.43	**64.29**	3.48	**49.9**	1.58	439	0.00	83	1.90
Glass	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**190**	0.89	192	0.00	190	0.89
Heart	**0.60**	0.70	0.88	0.85	0.66	0.61	**0.60**	0.62	**159.7**	4.78	243	0.00	199.8	1.66
Hepatitis	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**66.1**	0.83	72	0.00	67.8	0.60
Housevotes	5.85	1.57	4.77	2.27	5.00	2.42	**3.75**	2.17	**38.1**	3.88	208	0.00	146.5	2.58
Ionosphere	0.39	0.38	**0.37**	0.29	0.49	0.45	**0.37**	0.40	**265.7**	4.73	315	0.00	287.5	2.01
Iris	25.52	4.64	**4.48**	5.06	22.76	4.56	23.24	4.20	**18.6**	1.02	135	0.00	48.2	1.78
Lev	**19.30**	2.20	**19.30**	2.20	**19.30**	2.20	**19.30**	2.20	**33.4**	0.00	900	0.00	52.7	0.46
Machinecpu4classes	**5.77**	2.48	6.73	3.05	6.16	2.97	5.87	3.08	**93**	2.19	188	0.00	113.7	1.85
Mammographic	32.53	2.67	32.73	2.42	32.44	2.57	**32.26**	2.61	**21**	1.10	747	0.00	87.6	2.87
Newthyroid	6.20	3.06	**0.09**	0.18	3.05	2.29	3.07	2.71	**90.5**	2.38	193	0.00	122.3	2.00
Pima	4.65	1.50	**0.14**	0.25	5.70	1.43	5.90	1.59	**190.1**	7.78	691	0.00	472.8	4.38
Saheart	**1.75**	0.81	3.45	1.15	3.15	1.14	2.49	0.95	**226**	6.59	415	0.00	313.9	3.53
Segment	0.01	0.01	**0.00**	0.00	0.01	0.01	0.01	0.01	**1861.6**	4.29	2079	0.00	1868.5	3.96
Sonar	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	187.2	0.40	**187**	0.00	187.2	0.40
Swd	**10.02**	2.94	**10.02**	2.94	**10.02**	2.94	**10.02**	2.94	**62**	0.00	900	0.00	82.9	0.70
Titanic	**2.83**	0.52	**2.83**	0.52	**2.83**	0.52	**2.83**	0.52	**2**	0.00	1980	0.00	3	0.00
Vowel	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**891**	0.00	**891**	0.00	**891**	0.00
Wine	**0.00**	0.00	0.07	0.22	0.07	0.22	0.07	0.22	**146.1**	2.17	160	0.00	151	1.10
Wisconsin	**27.79**	4.62	40.00	2.19	38.01	2.54	38.49	2.54	**72.4**	5.00	614	0.00	201.9	2.74
Average	9.16	1.36	**8.45**	1.26	9.10	1.44	8.91	1.80	**221.1**	2.5	533.8	0.00	260.6	1.8

When considering particular cases in the reported results, we can point out that the nature of the data sets varies and the results achieved for each algorithm highly depend on the complexity and particularities of each one. In particular, we can see:

- MoNGEL clearly outperforms OLM and OSDL in accuracy and MAE in almost all data sets except in *Australian*. Surprisingly, OSDL offers the best outcome by far in this data set. The justification for this is not easy to discern, since it confirms the well-known theorem of "No free lunch" for supervised learning [42] which states that there is no algorithm better than another in all possible scenarios or problems.
- A remarkable trend can be observed when considering a partial set of data sets where all the attributes are numerical, especially the three obtained from regression problems (*Auto-mpg*, *Bostonhousing*, *Machinecpu*) and *Wine* cases. In all these data sets, *k*-NN shows significantly better accuracy and MAE results than MoNGEL. The reason behind this could be due to the number of neighbors managed by *k*-NN in our experiments. A 3-NN is considered and it is more robust against noisy instances, which may frequently appear in data sets with numerical attributes. In principle, NGE learning is based on computing only the nearest instance/rule for the sake of the trade-off between simplicity or interpretability of the models and accuracy. Nevertheless, the competitiveness of MoNGEL with 3-NN is demonstrated in the rest of data sets, allowing us to reduce the negative effects derived from noise and errors contained in the data with the generalization of examples into rules.

Table 5 Time for model building, classification of train dataset and classification of test dataset reported by all algorithms in all data sets

Datasets	Time model (ms)						Time clasification train (ms)								Time clasification test (ms)							
	OLM		OSDL		MoNGEL		OLM		OSDL		KNN		MoNGEL		OLM		OSDL		KNN		MoNGEL	
	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std
Australian	**0.2**	0.4	15.6	0.5	176.6	29.0	1.5	4.5	**0.0**	0.0	159.5	6.3	1.6	4.8	**0.0**	0.4	**0.0**	4.5	14.1	0.0	**0.0**	0.3
auto-mpg4classes	**14.1**	4.7	15.4	0.5	62.6	0.5	**0.0**	0.0	31.5	0.7	133.2	7.1	12.4	6.2	**0.0**	4.7	**0.0**	0.0	9.4	0.0	**0.0**	5.9
automobile	**3.5**	6.1	15.6	0.7	29.5	4.5	3.3	6.4	21.2	8.6	78.1	0.3	**1.5**	4.5	**0.0**	6.1	0.1	6.4	13.8	0.0	1.6	2.2
bands	**26.4**	7.0	74.9	6.5	50.0	6.0	**7.9**	7.9	361.1	4.7	256.2	7.6	23.6	7.6	**0.0**	7.0	38.9	7.9	21.4	0.0	4.7	1.2
bostonhousing4classes	**31.1**	0.3	38.6	8.5	71.8	7.6	**8.3**	7.1	95.7	4.7	312.6	0.5	17.1	4.7	**1.8**	0.3	14.1	7.1	32.0	4.7	3.2	3.7
cleveland	**14.3**	4.8	15.4	0.5	47.6	2.2	**9.2**	7.5	23.6	8.0	132.8	7.8	9.3	7.6	**0.0**	4.8	1.6	7.5	14.0	0.0	**0.0**	2.8
Dermatology	**15.6**	0.5	78.1	0.3	53.1	7.7	**0.0**	0.0	426.0	10.5	374.4	7.4	21.8	7.7	**0.0**	0.5	47.6	0.0	36.1	0.0	**0.0**	3.1
era	14.5	2.3	20.9	3.4	**14.1**	4.7	**0.4**	0.8	20.1	4.2	455.1	5.9	3.1	6.2	**0.0**	2.3	1.1	0.8	43.8	0.0	**0.0**	0.0
esl	**0.0**	0.0	15.5	0.5	17.4	4.9	3.2	6.4	15.8	0.4	129.6	7.0	**0.0**	0.0	**0.0**	0.0	**0.0**	6.4	7.6	0.0	**0.0**	1.6
glass	**0.0**	0.0	**0.0**	0.0	14.0	4.7	8.0	7.6	**0.0**	0.0	48.2	3.9	1.5	4.5	**0.0**	0.0	**0.0**	7.6	0.6	0.0	**0.0**	0.9
heart	**3.2**	6.4	32.7	4.5	37.4	7.8	**0.0**	0.0	136.5	6.2	109.3	0.5	11.1	7.3	**0.0**	6.4	15.9	0.0	10.9	0.0	**0.0**	4.8
hepatitis	**11.0**	7.2	14.2	4.7	15.7	0.5	**0.0**	0.0	1.5	4.5	31.3	0.5	**0.0**	0.0	**0.0**	7.2	**0.0**	0.0	**0.0**	0.0	**0.0**	0.8
housevotes	**0.0**	0.0	**0.0**	0.0	15.5	0.5	**0.0**	0.0	15.6	0.5	93.9	0.3	1.6	4.8	**0.0**	0.0	0.1	0.0	14.1	0.0	**0.0**	3.9
ionosphere	**16.0**	0.0	29.9	4.7	59.6	6.9	**1.5**	4.5	64.0	4.7	361.3	6.2	23.4	7.6	**1.6**	0.0	4.6	4.5	34.3	4.8	**1.6**	4.7
iris	**0.0**	0.0	**0.0**	0.0	15.2	1.2	**0.0**	0.0	15.6	0.7	23.2	7.8	0.2	0.6	**0.0**	0.0	0.1	0.0	1.6	0.0	**0.0**	1.0
lev	**12.3**	6.2	15.5	0.5	15.7	0.5	**1.6**	4.8	25.0	7.8	468.1	2.1	12.6	6.3	**0.0**	6.2	**0.0**	4.8	46.8	0.0	**0.0**	0.0
machinecpu4classes	**12.5**	6.3	30.3	2.5	18.5	6.5	**1.5**	4.5	47.1	0.7	46.6	0.5	4.8	7.3	**0.0**	6.3	2.1	4.5	4.8	0.0	**0.0**	2.2
mammographic	**0.0**	0.0	31.2	7.2	170.3	10.9	**0.0**	0.0	50.1	6.5	412.2	6.9	7.8	7.8	**0.0**	0.0	11.1	0.0	42.1	0.0	1.6	1.1
newthyroid	**12.5**	6.3	15.7	0.5	31.5	0.5	1.5	4.5	**0.0**	0.0	46.7	0.5	**0.0**	0.0	**0.0**	6.3	**0.0**	4.5	**0.0**	0.0	**0.0**	2.4
pima	31.3	0.5	**14.8**	2.6	381.3	18.7	**6.4**	7.8	31.3	0.5	500.7	3.2	32.9	4.7	**0.0**	0.5	3.1	7.8	46.8	0.0	1.6	7.8
saheart	**15.8**	0.4	125.0	0.0	137.4	6.2	**0.5**	1.5	698.1	16.1	222.0	6.3	17.3	4.6	**0.0**	0.4	77.9	1.5	15.6	0.0	3.2	6.6
segment	307.4	23.3	**34.4**	6.3	620.7	12.2	95.9	7.6	**60.9**	8.3	8180.6	26.7	376.0	3.0	22.0	23.3	**6.3**	7.6	945.2	7.8	45.6	4.3
sonar	**15.6**	0.5	15.7	0.5	31.2	0.4	**1.5**	4.5	11.0	7.2	235.9	4.7	15.6	0.5	**1.5**	0.5	1.6	4.5	21.8	4.5	1.6	0.4
swd	**14.0**	4.7	17.2	4.6	25.0	7.8	**0.0**	0.0	21.9	7.9	909.6	6.9	11.5	9.7	1.6	4.7	**0.0**	0.0	96.7	4.8	1.6	0.0
titanic	**0.0**	0.0	15.7	0.5	17.0	4.7	**0.0**	0.0	62.4	0.5	1857.0	13.2	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0	196.7	0.0	**0.0**	0.0
vowel	86.1	14.4	**17.5**	4.5	126.6	10.9	29.8	4.6	**24.9**	7.5	1127.5	10.1	76.6	4.9	5.2	14.4	**1.6**	4.6	132.9	6.9	7.8	0.0
wine	3.6	6.1	**1.6**	4.8	15.5	0.5	**0.0**	0.0	3.1	6.2	48.4	4.5	**0.0**	0.0	**0.0**	6.1	**0.0**	0.0	1.6	0.0	**0.0**	2.2
wisconsin	**15.9**	0.3	31.1	0.3	176.6	9.9	**0.4**	1.2	31.5	0.5	439.1	5.0	15.5	0.5	**0.0**	0.3	6.3	1.2	46.9	0.0	3.2	5.0
Average	**24.2**	3.9	26.2	2.5	87.4	6.4	**6.5**	3.4	82.0	4.6	614.0	5.7	25.0	4.4	**1.2**	3.9	8.4	3.4	66.1	1.2	2.8	2.5

- Another interesting fact is that MoNGEL reports very competitive results when tackling data sets with many attributes, such the case of the data sets *Automobile*, *Bands*, *Dermatology*, *Hepatitis*, *Ionosphere*, *Segment* and *Sonar*. The more attributes a data set has, the harder is to handle them with approaches that do not learn a model, such as OSDL and *k*-NN. Irrelevant and harmful attributes hinder the predictive capabilities of these algorithm and diminish their final accuracy. MoNGEL is able to mitigate these effects due to the fact that these kind of attributes may be generalized to form rules.

Regarding the efficiency of the algorithms compared, we can highlight the following observations by considering the results reported in Table 5. All the algorithms were run on the same computer with these features: I5-4460 3.2 GHz, 8 GB RAM and SD 128 GB disk.

- MoNGEL is the algorithm that requires most time to learn the model. It takes around three times longer than OLM and OSDL. It could be considered as a weakness, but it is worth mentioning that the time in modeling is spent only once. The most important point is to make quicker predictions and it will be exhibited by the classification times.
- When considering classification times, both for classifying the whole training and test sets, MoNGEL offers the second best outcome. OLM is quicker than MoNGEL but, as we saw before, the accuracy offered by OLM is clearly outperformed by MoNGEL. *k*-NN requires much time to make predictions with similar accuracy to MoNGEL. OSDL is also slower than MoNGEL in classification times.

In summary, our proposal MoNGEL achieves excellent results in terms of accuracy and MAE, it requires few rules and yields more monotonic models. Clearly, MoNGEL is as good as *k*-NN in precision performance, but it requires a much lower number of generalized instances for building the model. With respect to the simplicity of the model obtained, MoNGEL requires more rules than OLM but instead it obtains a better performance in accuracy and MAE. Regarding computational times, MoNGEL provides a good trade-off between efficacy and efficiency when we measure the time required to classify new examples. It is worth spending a little more time to learn a NGE model if it will be then used many times with very low response times.

Table 6 collects the results of applying the Wilcoxon test to MoNGEL and the rest of the methods studied in this paper. In each one of the cells, three symbols can appear: $+$, $=$ or $-$. They represent either that MoNGEL outperforms ($+$), is similar to ($=$) or is worse ($-$) in performance than the rest of the methods. The value in brackets is the *p*-value obtained in the comparison. Other statistical studies can be performed using non-parametric multiple comparison tests. These types of procedures study the set of results obtained by all the algorithms to compute a ranking that takes into account the multiple comparison. The smaller the ranking, the better is the algorithm. Rankings of the Friedman test are depicted in Table 7.

Table 6 Wilcoxon test report

Algorithm	Accuracy	MAE	NMI	Number of rules
OLM	+(0.000)	+(0.000)	=(0.589)	=(1)
OSDL	+(0.000)	+(0.001)	=(1)	–
k-NN	=(1)	=(1)	=(0.577)	+(0.000)

Table 7 Rankings obtained by the Friedman test

Algorithm	Accuracy	MAE	NMI	Number of rules
OLM	3.0357	3.1071	2.5714	**1.0893**
OSDL	3.2321	3.0536	2.4643	–
k-NN	**1.8393**	**1.875**	2.6429	2.9286
MonGEL	1.8929	1.9643	**2.3214**	1.9821

The results of the statistical tests allow us to highlight the following observations:

- MoNGEL is the best approach compared with the other techniques, except in the number of rules where OLM is the best.
- Our algorithm significantly outperforms OLM and OSDL in accuracy and MAE. No differences are reported when comparing it with *k*-NN.
- No statistical differences have been detected in NMI.

6 Concluding remarks

The goal of this paper was to present a proposal of an algorithm for NGE learning for classification with monotonicity constraints named MoNGEL. It creates an initial set of rules from the training data and then it performs a generalization process focused on maximizing the accuracy while minimizing the number of instances retained and conserving the anti-monotonic index. The results showed that MoNGEL allows us to obtain very precise models with a low anti-monotonic index and few rules. We have compared it with well-known monotonic learning approaches and the effectiveness of our approach is very competitive.

As future work, we are interested in the application of NGE learning to large monotonic data sets by considering data with a large number of instances and attributes. Reduction techniques for data partitioning as Big Data solutions [40], together with space partitioning data

structures for organizing points in multi dimensional spaces, such as KD-Trees, Ball Trees or locally sensitive hashing [34] will be studied in further works.

Acknowledgments The authors are very grateful to the anonymous reviewers for their valuable suggestions and comments to improve the quality of this paper.

Compliance with ethical standards

Conflict of interest We declare that we have no conflict of interest.

References

1. Aha DW (ed) (1997) Lazy learning. Springer, New York
2. Aha DW, Kibler D, Albert MK (1991) Instance-based learning algorithms. Mach Learn 6(1):37–66
3. Alcala-Fdez J, Fernández A, Luengo J, Derrac J, García S, Sánchez L, Herrera F (2011) KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J Mult-Valued Log Soft Comput 17(2–3):255–287
4. Bache K, Lichman M (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml. Accessed 21 May 2015
5. Ben-David A (1992) Automatic generation of symbolic multiattribute ordinal knowledge-based dsss: methodology and applications. Decis Sci 23:1357–1372
6. Ben-David A (1995) Monotonicity maintenance in information-theoretic machine learning algorithms. Mach Learn 19(1):29–43
7. Ben-David A, Sterling L, Pao YH (1989) Learning, classification of monotonic ordinal concepts. Comput Intel 5:45–49
8. Ben-David A, Sterling L, Tran T (2009) Adding monotonicity to learning algorithms may impair their accuracy. Expert Syst Appl 36(3):6627–6634
9. Cao-Van K (2003) Supervised ranking, from semantics to algorithms. Ph.D. dissertation, Ghent University, Ghent
10. Cao-Van K, Baets BD (2003) Growing decision trees in an ordinal setting. Int J Intel Syst 18(7):733–750
11. Chen CC, Li ST (2014) Credit rating with a monotonicity-constrained support vector machine model. Expert Syst Appl 41(16):7235–7247
12. Daniels H, Velikova M (2010) Monotone and partially monotone neural networks. IEEE Trans Neural Netw 21(6):906–917
13. Dembczyński K, Kotłowski W, Słowiński R (2009) Learning rule ensembles for ordinal classification with monotonicity constraints. Fundam Inform 94(2):163–178
14. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
15. Derrac J, García S, Herrera F (2014) Fuzzy nearest neighbor algorithms: taxonomy, experimental analysis and prospects. Inf Sci 260:98–119
16. Domingos P (1996) Unifying instance-based and rule-based induction. Mach Learn 24(2):141–168
17. Duivesteijn W, Feelders A (2008) Nearest neighbour classification with monotonicity constraints. In: ECML/PKDD (1), pp 301–316
18. Escalante HJ, Marin-Castro M, Morales-Reyes A, Graff M, Rosales-Pérez A, y Gómez MM, Reyes CA, González JA (2015) MOPG: a multi-objective evolutionary algorithm for prototype generation. Pattern Anal Appl. doi:10.1007/s10044-015-0454-6 (in press)
19. Feelders AJ, Pardoel M (2003) Pruning for monotone classification trees. In: IDA. Lecture notes in computer science, vol, 2810. Springer, New York, pp 1–12
20. Fernández-Navarro F, Riccardi A, Carloni S (2014) Ordinal neural networks without iterative tuning. IEEE Trans Neural Netw Learn Syst 25(11):2075–2085
21. Fürnkranz J (1999) Separate-and-conquer rule learning. Artif Intel Rev 13:3–54
22. García S, Herrera F (2008) An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. J Mach Learn Res 9:2677–2694
23. García S, Fernández A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. Inf Sci 180(10):2044–2064
24. García S, Derrac J, Luengo J, Carmona CJ, Herrera F (2011) Evolutionary selection of hyperrectangles in nested generalized exemplar learning. Appl Soft Comput 11(3):3032–3045
25. García S, Derrac J, Triguero I, Carmona CJ, Herrera F (2012) Evolutionary-based selection of generalized instances for imbalanced classification. Knowl-Based Syst 25(1):3–12
26. García S, Luengo J, Herrera F (2015) Data preprocessing in data mining. Springer, New York
27. Gaudette L, Japkowicz N (2009) Evaluation methods for ordinal classification. In: Canadian conference on AI. Lecture notes in computer science, vol 5549, pp 207–210
28. Han J, Kamber M (2011) Data mining: concepts and techniques. Morgan Kaufmann Publishers Inc., San Francisco
29. Hu Q, Che X, Zhang L, Zhang D, Guo M, Yu D (2012) Rank entropy-based decision trees for monotonic classification. IEEE Trans Knowl Data Eng 24(11):2052–2064
30. Japkowicz N, Shah M (eds) (2011) Evaluating learning algorithms: a classification perspective. Cambridge University Press, Cambridge
31. Kotłowski W, Słowiński R (2009) Rule learning with monotonicity constraints. In: ICML, vol 382
32. Kotłowski W, Słowiński R (2013) On nonparametric ordinal classification with monotonicity constraints. IEEE Trans Knowl Data Eng 25(11):2576–2589
33. Lievens S, Baets BD, Cao-Van K (2008) A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting. Ann Oper Res 163(1):115–142
34. Liu T, Moore AW, Gray A (2006) New algorithms for efficient high-dimensional nonparametric classification. J Mach Learn Res 7:1135–1158
35. López V, Fernández A, García S, Palade V, Herrera F (2013) An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics. Inf Sci 250:113–141
36. Mariolis IG, Dermatas E (2013) Automatic classification of seam pucker images based on ordinal quality grades. Pattern Anal Appl 16(3):447–457
37. Potharst R, Feelders AJ (2002) Classification trees for problems with monotonicity constraints. SIGKDD Explor 4(1):1–10
38. Potharst R, Ben-David A, van Wezel MC (2009) Two algorithms for generating structured and unstructured monotone ordinal datasets. Eng Appl Artif Intel 22(4–5):491–496
39. Salzberg S (1991) A nearest hyperrectangle learning method. Mach Learn 6(3):251–276
40. Triguero I, Peralta D, Bacardit J, García S, Herrera F (2015) MRPR: a mapreduce solution for prototype reduction in big data classification. Neurocomputing 150:331–345
41. Wettschereck D, Dietterich TG (1995) An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. Mach Learn 19(1):5–27
42. Wolpert DH (1996) The lack of a priori distinctions between learning algorithms. Neural Comput 8(7):1341–1390

3.2. Hyperrectangles Selection for Monotonic Classification by Using Evolutionary Algorithms

- Estado: Publicado.
- Título: Hyperrectangles Selection for Monotonic Classification by Using Evolutionary Algorithms.
- Autores: Javier García, Adnan M. Albar, Naif R. Aljohani, José-Ramón Cano y Salvador García.
- Revista: International Journal of Computational Intelligence Systems, Vol. 9, No. 1 (2016) 184-201
- ISSN: 1875-6891
- Factor de Impacto (JCR 2015): 0.391
- Cuartiles por Área de Conocimiento:
  - Cuartil 4 en *Computer Science, Artificial Intelligence*, Ranking 125/130
  - Cuartil 4 en *Computer Science, Interdisciplinary Applications*, Ranking 101/104

International Journal of Computational Intelligence Systems, Vol. 9, No. 1 (2016) 184-201

Hyperrectangles Selection for Monotonic Classification by Using Evolutionary Algorithms

Javier García [1] , Adnan M. AlBar,[2] Naif R. Aljohani,[2] José-Ramón Cano [1] , Salvador García [3]

[1] Department of Computer Science, University of Jaén,
EPS of Linares, Calle Alfonso X el Sabio S/N,
Linares, 23700, Spain

E-mails: jgf00002@red.ujaen.es, jrcano@ujaen.es

[2] Information Systems Department, Faculty of Computing and Information Technology,
King Abdulaziz University, Kingdom of Saudi Arabia

E-mails: ambar@kau.edu.sa, nraljohani@kau.edu.sa

[3] Department of Computer Science and Artificial Intelligence, University of Granada,
Calle Periodista Daniel Saucedo Aranda S/N,
Granada, 18071, Spain

E-mail: salvagl@decsai.ugr.es

Received 4 July 2015

Accepted 3 January 2016

Abstract

In supervised learning, some real problems require the response attribute to represent ordinal values that should increase with some of the explaining attributes. They are called classification problems with monotonicity constraints. Hyperrectangles can be viewed as storing objects in $\mathbb{R}^n$ which can be used to learn concepts combining instance-based classification with the axis-parallel rectangle mainly used in rule induction systems. This hybrid paradigm is known as nested generalized exemplar learning. In this paper, we propose the selection of the most effective hyperrectangles by means of evolutionary algorithms to tackle monotonic classification. The model proposed is compared through an exhaustive experimental analysis involving a large number of data sets coming from real classification and regression problems. The results reported show that our evolutionary proposal outperforms other instance-based and rule learning models, such as OLM, OSDL, *k*-NN and MID; in accuracy and mean absolute error, requiring a fewer number of hyperrectangles.

Keywords: Monotonic Classification, Nested Generalized Examples, Evolutionary Algorithms, Rule Induction, Instance-based Learning.

1. Introduction

The classification with monotonicity constraints, also known as monotonic classification [1], is an ordinal classification problem where a monotonic restriction is present: a higher value of an attribute in an example, fixing other values, should not decrease its class assignment. The monotonicity of relations between the dependent and explanatory variables is very usual as a prior knowledge form in data classification [2]. To illustrate, while considering a credit card application [3], a $1000 to $2000 income may be

considered a medium value of income in a data set. If a customer A has a medium income, a customer B has a low income (i.e. less than $1000) and the rest of input attributes remain the same, there is a relationship of partial order between A and B: $B < A$. Considering that the application estimates lending quantities as output class, it is quite obvious that the loan that the system should give to customer B cannot be greater than the given to customer A. If so, a monotonicity constraint is violated in the decision.

The estimation of the knowledge about monotonicity in learning models is of a great interest for two main arguments [4]. Firstly, monotonicity imposes constraints on the prediction function. This decreases the size of the hypothesis space and also the complexity of the model. Secondly, the domain experts decide the acceptance or rejection of the models yielded if they are consistent with the domain knowledge, regardless of their accuracy [5].

Many data learning algorithms have been adapted to be able to handle monotonicity constraints in several styles. There are two steps to treat with monotonic classification problems. The first one is to preprocess the data [6] in order to "monotonize" the data set [7], rejecting the examples that violate the monotonic restrictions or selecting features to improve classification performance and avoid overfitting [8,9]; and the second one is to force learning only monotone classification functions. Proposals of this type are: classification trees and rule induction [10,11,12,13], neural networks [14] and instance-based learning [15,16,17].

Instance-based learning was baptized as learning family in [18] and considers a set of methods widely used in machine learning [19]. An analogous scheme for instance-based learning is the Nested Generalized Exemplar (NGE) theory. It was announced in [20] and perform various major adjustments to the instance-based learning model. The most relevant is that it allows two type of examples, standard examples represented as single points and generalized examples which fill a hyper-volume in $\mathbb{R}^n$. They are closely connected to the nearest neighbor classifier (NN) [21], in such a way that they are intended to enhance it. NGE learning algorithms are being more popular in recent years due to the simplicity and effectiveness of the outcome they provide.

In this manner, the hyperrectangles are generalization of examples in $\mathbb{R}^n$, according to NGE theory. Single and generalized examples coexist and hyperrectangles may be nested and inner hyperrectangles serve as exceptions to surrounding hyperrectangles. They constitute axis-parallel rectangle representations as in many of the rule learning systems [22]. Using this model, a new example can be classified by estimating the Euclidean distance between it and every of the hyperrectangles stored. The class is predicted according to the label associated with the nearest hyperrectangle. In the case that two or more hyperrectangles cover the example, it is necessary to resolve a possible conflict derived from different labels among the hyperrectangles [20].

The profits of combining hyperrectangles with instances to build classification models are pointed out in the literature [23,24,25]. Looking at rule induction [22], the modeling of decision surfaces derived from combinations between parallel axis separators and Voronoi diagrams (typical decision surfaces of NN classifiers) may adapt the prediction to examples drawn along curves, improving the performance in complex domains. Regarding instance-based classification [18], the hyperrectangles adds interpretability to the model and reduces storage requirements.

The generation of an optimal minimal number of hyperrectangles for classifying a set of points is NP-hard. Heuristic algorithms produce a large but finite subset of hyperrectangles from the training data. Nevertheless, it may be easy that almost all hyperrectangles modeled could be unnecessary. Thus, there is a need for selecting the most influential ones, and it can be done by using data reduction schemes [6]. Evolutionary Algorithms (EAs) [26] have been used for data reduction with promising results [27]. They have been favorably used in NGE learning in the past [28,29].

In this paper, we propose the utilization of EAs for hyperrectangles' selection in monotonic classification tasks. Our goal is to increase the performance in this type of problem by means of selecting the best suitable set of hyperrectangles which optimizes the nearest hyperrectangle classification with monotonicty constraints. We compare our algorithm

with other monotonic learning models belonging to both instance-based and rule learning paradigms. They are OLM [15], MID [10], OSDL [16] and monotonic k-NN [17]. The experimental design incorporates non-parametrical statistical testing [30,31]. The results show a significant improvement in accuracy whereas the number of examples stored in the final subset is further reduced. Besided, the main key point of our proposal is that the model built is composed of generalized examples which all fulfill the monotonicity constraints.

The paper is organized as follows. Section 2 provides an background in monotonic classification and the NGE learning model. In Section 3, all topics concerning the approach proposed are described. In Section 4 the experimentation framework is given and in Section 5 the results and analysis are presented. In Section 6, the conclusions are highlighted.

2. Background

We will do a brief review of the monotonic classification including the description of the most used techniques for monotonic classification in Subsections 2.1 and 2.2. The NGE classification will be described in Subsection 2.3.

2.1. Monotonic Classification

Ordinal classification problems are those in which the class is neither numeric nor nominal. Instead, the class values are ordered. For instance, a worker can be described as "excellent", "good" or "bad", and a bond can be evaluated as "AAA", "AA", "A", "A-", etc. Similar to a numeric scale, an ordinal scale has an order, but it does not possess a precise concept of distance. Ordinal classification problems are important, since they are fairly common in our daily life. Employee selection and promotion, determining credit rating, bond rating, economic performance of countries, industries and firms, and insurance underwriting, are examples of ordinal problem-solving in business. Rating manuscripts, evaluating lecturers, student admissions, and scholarships decisions for students, are examples of ordinal decision-making in academic life. Ordinal problems have been investigated in scientific disciplines such as information retrieval, psychology, and statistics for many decades [4] .

A monotonic classifier is one that will not violate monotonicity constraints. Informally, the monotonic classification implies that the assigned class values are monotonically non-decreasing (in ordinal order) with the attribute values. More formally, let $\{\mathbf{x}_i, \text{class}(\mathbf{x}_i)\}$ denote a set of examples with attribute vector $\mathbf{x}_i = (\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,m})$ and a class, $\text{class}(\mathbf{x}_i)$, being n the number of instances and m the number of attributes. Let $\mathbf{x}_i \succeq \mathbf{x}_h$ iff $\forall_{j=1,\ldots,m}, \mathbf{x}_{i,j} \geqslant \mathbf{x}_{h,j}$.

A data set $\{\mathbf{x}_i, \text{class}(\mathbf{x}_i)\}$ is monotonic if and only if all the pairs of examples i, h are monotonic with respect to each other [10] (see equation 1).

$$\mathbf{x}_i \succeq \mathbf{x}_h \implies \text{class}(\mathbf{x}_i) \geqslant \text{class}(\mathbf{x}_h), \forall_{i,h} \quad (1)$$

Some monotonic ordinal classifiers require monotonic data sets to successfully learn, although there are others that are capable of learning from non-monotonic data sets as well.

2.2. Monotonic Classification Methods

Four monotonic learning methods are pioneers and well-known in the field of monotonic classification. In the following, we will discuss each one in detail.

- The Ordinal Learning Model (OLM) [15] is a very simple algorithm that learns ordinal concepts by eliminating non-monotonic pairwise inconsistencies. The generated concepts can be viewed as rules. During the learning phase, each example is checked against every rule in a rule-base, which is initially empty. If an example is inconsistent with a rule in the rule-base, one of them is selected at random while the other is discarded, but if the example is selected, it must be checked for consistency against all the other monotonicity rules. If it passes this consistency test, it is added as a rule. Consequently, the rule-base is kept monotonic at all times. Classification is done conservatively. All the rules are checked in decreasing order of class values against an attribute vector, and the vector is classified as the class of the first

rule that covers it. If such a rule does not exist, the attribute vector is assigned the lowest possible class.

- As its name implies, Ordinal Stochastic Dominance Learner (OSDL) [16], is based on the concept of Ordinal Stochastic Dominance (OSD). The rationale behind OSD can be given through an example: In life insurance, one may expect a stochastically greater risk to the insurer from older and sicker applicants than from younger and healthier ones. Higher premiums should reflect greater risks and vice versa. There are several definitions of stochastic ordering. The stochastic order computes when a random variable is bigger than another. Considering this order, stochastic dominance can be established as a form of stochastic order. In this case, a probability distribution over possible predictions can be ranked. The ranking depends of the nature of the data set. Stochastic dominance refers to a set of relations that may hold between a pair of distributions.
  The most commonly used, as stochastic ordering, is first OSD, which was used by [36] in the OSDL. For each vector $\mathbf{x}_i$, the OSDL computes two mapping functions: one that is based on the examples that are stochastically dominated by $\mathbf{x}_i$ with the maximum label (of that subset), and the second is based on the examples that cover (i.e., dominate) $\mathbf{x}_i$, with the smallest label. Later, an interpolation between the two class values (based on their position) is returned as a class.
- The monotonic *k*NN (M*k*-NN) was proposed in [17]. This method consists of two steps. In the first step, the training data is made monotone by relabeling as few cases as possible. This relabeled data set may be considered as the monotone classifier with the smallest error index in the training data. In the second step, we use a modified nearest neighbour rule to predict the class labels of new data so that violations of the restrictions of monotonicity will not occur. Considering the monotonic nearest neighbor rule, the class label assigned to a new data point $\mathbf{x}_0$ must lie in the interval $[\text{class}_{min}, \text{class}_{max}]$, where

$$\text{class}_{min} = \max\{\text{class}(\mathbf{x}_i) | \{\mathbf{x}_i, \text{class}(\mathbf{x}_i)\} \wedge \mathbf{x}_i \leqslant \mathbf{x}_0\} \quad (2)$$

and

$$\text{class}_{max} = \min\{\text{class}(\mathbf{x}_i) | \{\mathbf{x}_i, \text{class}(\mathbf{x}_i)\} \wedge \mathbf{x}_0 \leqslant \mathbf{x}_i\} \quad (3)$$

where $\{\mathbf{x}_i, \text{class}(\mathbf{x}_i)\}$ is the monotone data set. To conserve the monotonicity the choice of the class value for $\mathbf{x}_0$ must be in this interval.

- A monotone extension of ID3 (MID) was proposed by Ben-David [10] using an additional impurity measure for splitting, the total ambiguity score. However, the resulting tree may not be monotone anymore even when starting from a monotone data set. MID defines the *total-ambiguity-score* as the sum of the entropy score of ID3 and the order-ambiguity-score. This last score is defined in terms of the non-monotonicity index of the tree, which computes the number of pair branches that are non-monotonic regarding the total possible non-monotonic pairs there may be.

2.3. NGE Learning

NGE is a learning paradigm based on class exemplars, where an induced hypothesis has the graphical shape of a set of hyperrectangles in $\mathbb{R}^n$. Exemplars of classes are either hyperrectangles or single instances [20]. The input of an NGE system is a set of training examples, each described as a vector of pairs $numeric\_attribute/value$ and an associated class. Attributes can either be numerical or categorical. Numerical attributes are usually normalized in the $[0,1]$ interval.

In NGE, an initial set of hyperrectangles in $\mathbb{R}^n$ formed by single points directly taken from the data is generalized into a smaller set of hyperrectangles regarding the elements that it contains. Choosing which hyperrectangle is generalized from a subset of points or other hyperrectangles and how it is generalized depends on the concrete NGE algorithm employed.

The matching process is one of the central features in NGE learning and it allows some customiza-

tion, if desired. Generally speaking, this process computes the distance between a new example and an exemplar memory object (a generalized example). Let the example to be classified be termed as $\mathbf{x}_0$ and the generalized example as G, regardless of whether G is formed by a single point or if it has some volume.

The model computes a match score between $\mathbf{x}_0$ and G by measuring the Euclidean distance between two objects. The Euclidean distance is well-known when G is a single point. Otherwise, the distance is computed as follows (considering numerical attributes):

$$D_{\mathbf{x}_0 G} = \sqrt{\sum_{j=1}^{m} \left(\frac{\text{rdif}_j}{\max_j - \min_j} \right)^2} \tag{4}$$

$$\text{rdif}_j = \begin{cases} \mathbf{x}_{0,j} - \max(G_j) & \text{when } \mathbf{x}_{0,j} > \max(G_j), \\ \min(G_j) - \mathbf{x}_{0,j} & \text{when } \mathbf{x}_{0,j} < \min(G_j), \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where m is the number of attributes of the data, $\mathbf{x}_{0,j}$ is the value of the jth attribute of the example $\mathbf{x}_0$, $\max(G_j)$ and $\min(G_j)$ are the maximum and minimum values of G for the jth attribute and $\max_j$ and $\min_j$ are the maximum and minimum values for jth attribute in training data, respectively.

The distance measure represents the length of a line dropped perpendicularly from the point $\mathbf{x}_0$ to the nearest surface, edge or corner of G. Note that internal points in a hyperrectangle have a distance of 0 to that rectangle. In the case of overlapping rectangles, several strategies could be implemented, but it is usually accepted that a point falling in the overlapping area belongs to the smaller rectangle (the size of a hyperrectangle is defined in terms of volume). The volume is computed following the indications given in [23]. In nominal attributes, the distance is 0 when two attributes have the same categorical label, and 1 on the contrary.

3. Evolutionary Selection of Hyperrectangles for Monotonic Clasification

The approach proposed in this paper, named Evolutionary Hyperrectangle Selection for Monotonic clasification by CHC (*EHSMC-CHC*), is fully explained in this section. First, we approximate the NGE theory to monotonic classification in Subsection 3.1 by providing some definitions. Then, we describe the process for generating the initial set of hyperrectangles in Subsection 3.2. After this, we introduce the CHC model used as an EA to perform hyperrectangle selection in Subsection 3.3. Finally, the specific issues regarding representation and fitness function are specified in Subsection 3.4.

3.1. Definitions

We represent each instance with $\mathbf{x}_i = (\mathbf{x}_{i,1}; \mathbf{x}_{i,2}; .....; \mathbf{x}_{i,m}; \text{class}(\mathbf{x}_i))$, where $\mathbf{x}_{i,j}$ is the input value of the instance i in the attribute j, m is the total number of input attributes and $\text{class}(\mathbf{x}_i)$ is the class or output value of the instance i. Let the hyperrectangles be denoted by $H : (\prod_j A_j; C)$, where A_j is a condition belonging to the antecedent and C is the response. A formal definition is given next:

$$\begin{aligned} &H : A_1 \times A_2 \times ..... \times A_m \Rightarrow C, \\ &\text{with} A_j = \begin{cases} [j_{\min}, j_{\max}] & \text{if numerical,} \\ \{\alpha, \beta, ...\} & \text{if nominal,} \end{cases} \\ &C = \text{a value of the class,} \end{aligned} \tag{6}$$

where $j_{\min}$ and $j_{\max}$ are the minimum and maximum boundaries for the condition j respectively, and $\alpha, \beta, \ldots$ are the possible categorical values belonging to the domain of the attribute j, if it is nominal.

The hyperrectangles should not break monotonicity in the sense we saw in Subsection 2.1. Hence, we define a partial order relation in the set of disjoint hyperrectangles, deciding when a precedent is higher than, lower than or equal to another:

$$\begin{array}{ll} A_j \preceq A'_j & \text{if attribute } j \text{ is numeric and } j_{\max} \leqslant j'_{\min} \\ & \text{or if attribute } j \text{ is nominal and } A_j \subseteq A'_j, \\ A_j \succeq A'_j & \text{if attribute } j \text{ is numeric and } j_{\max} \geqslant j'_{\min} \\ & \text{or if attribute } j \text{ is nominal and } A'_j \subseteq A_j, \\ A_j = A'_j & \text{if attribute } j \text{ is numeric and} \\ & j_{\min} = j'_{\min} \text{ and } j_{\max} = j'_{\max} \\ & \text{or if attribute } j \text{ is nominal and } A_j \equiv A'_j. \end{array} \tag{7}$$

Then let two hyperrectangles H and H' be defined as:

$$\begin{array}{lll} H \preceq H' & \text{if } A_j \preceq A'_j \forall j & \text{and } \exists l, 1 \leqslant l \leqslant m, A_l < A'_l, \\ H = H' & \text{if } A_j = A'_j \forall j, & \\ H \succeq H' & \text{if } A_j \succeq A'_j \forall j & \text{and } \exists l, 1 \leqslant l \leqslant m, A_l > A'_l. \end{array} \tag{8}$$

Two hyperrectangles are comparable if $H \preceq H'$ or $H \succeq H'$, and non-comparable in contrary case, since we cannot establish an order between them. Hence, an hyperrectangle $H : (\prod_j A_j, C)$ will be non-monotonic with respect to another $H' : (\prod_j A'_j, C')$ if:

$$\begin{array}{l} H \preceq H' \text{ and } C > C' \text{ or} \\ H \succeq H' \text{ and } C < C' \text{ or} \\ H = H' \text{ and } C \neq C'. \end{array} \tag{9}$$

The distance between an instance $\mathbf{x}_i$ and an hyperrectangle H is defined as follows:

$$D_{\mathbf{x}_i H} = \sqrt{\sum_{j=1}^{m} \left(\frac{\text{dis}_j}{\text{Range}} \right)^2}, \tag{10}$$

where dis_j and Range are defined differently depending on the type of the attribute. If the attribute is numeric:

$$\begin{array}{l} \text{Range} = \max_j - \min_j, \\ \text{dis}_j = \begin{cases} \mathbf{x}_{i,j} - j_{\max} \text{ if } \mathbf{x}_{i,j} > j_{\max}, \\ j_{\min} - \mathbf{x}_{i,j} \text{ if } \mathbf{x}_{i,j} < j_{\min}, \\ 0 \text{ otherwise.} \end{cases} \end{array} \tag{11}$$

If the attribute is nominal:

$$\begin{array}{l} \text{dis}_j = \begin{cases} 0 & \mathbf{x}_{i,j} \in A_j, \\ 1 & \text{otherwise,} \end{cases} \\ \text{Range} = \text{Num. of possible values of the attribute.} \end{array} \tag{12}$$

where $\mathbf{x}_{i,j}$ is the value of the j-th attribute of the instance i, $j_{\max}$ and $j_{\min}$ are the maximum and minimum values of the hyperrectangle H in the attribute j and $\max_i$ and $\min_i$ are the maximum and minimum values of the attribute j in the data set.

The distance between two hyperrectangles H and H' is defined as:

$$D_{HH'} = \sqrt{\sum_{j=1}^{m} \left(\frac{\text{dis}_j}{\text{Range}} \right)^2}, \tag{13}$$

where dis_j and Range are defined as:

$$\begin{array}{l} \text{dis}_j = |\frac{j_{\max} + j_{\min}}{2} - \frac{j'_{\max} + j'_{\min}}{2}|, \\ \text{Range} = \max_j - \min_j, \end{array} \tag{14}$$

if the attribute j is numeric and

$$\text{dis}_j = 1 - \frac{\#(A_j \bigcap A'_j)}{\#(A_j \bigcup A'_j)}, \tag{15}$$

if the attribute j is nominal.

3.2. Getting the Initial Set of Hyperrectangles

We start from a training set TR with n instances which consists of pairs $(\mathbf{x}_i, \text{class}(\mathbf{x}_i)), i = 1, ..., n$. Each one of the n instances has m input attributes.

In this first phase will get a hyperrectangle set HS with N hyperrectangles whose rule representation consists of pairs $(H_i, \text{class}(H_i)), i = 1, ..., N$, where H_i defines a set of conditions $(A_1, A_2, ..., A_m)$ and $\text{class}(H_i)$ defines the associated class label of the hyperrectangle. Each one of the N hyperrectangles has m conditions which can be numerical conditions, expressed in terms of minimum and maximum values in intervals $[0, 1]$; or they can be categorical conditions, by using a set of possible values $A_i = \{v_{1i}, v_{2i}, ..., v_{vi}\}$, assuming that it has vi different values. Note that we make no distinction between a hyperrectangle with volume and minimal hyperrectangles formed by isolated points.

In this first stage of our method, we have used a simple heuristic which is fast and yields acceptable results. The heuristic yields a hyperrectangle from each example in the training set. For each one, it finds the $k-1$ nearest neighbors being the $k-$th neighbor an example of a different class. Then, each hyperrectangle is expanded considering these $k-1$ neighbors by using, in the case of numerical attributes, the minimal and maximal values as the limits of the interval defined, or getting all the different categorical values, in the case of nominal attributes, to form a subset of possible values from them.

Once all the hyperrectangles are obtained, the duplicated ones are removed, keeping one representative in each case. Hence $|HS| \leqslant |TR|$. Note that point hyperrectangles are possible to be obtained using this heuristic when the nearest neighbor of an instance belongs to a different class.

3.3. CHC Model

As an evolutionary computation method, we have used the CHC model [32]. CHC is a classical evolutionary model that introduces important aspects to obtain a trade-off between exploration and exploitation; such as incest prevention, restoration of the search process when it becomes locked and the fight among parents and offspring into the replacement process.

During each generation the CHC realizes the following stages:

- *NC* children born after mating of *NC* individuals of the father population.
- Then, among the 2*NC* individuals formed by parents and children takes place a struggle for survival in which only remain *NC* individuals for the new generation.

CHC also implements a form of heterogeneous recombination using HUX, a special recombination operator [32]. HUX exchanges half of the bits that differ between parents, where the bit position to be exchanged is randomly determined. CHC also employs a method of incest prevention. Before applying HUX to the two parents, the Hamming distance between them is measured. Only those parents who differ from each other by some number of bits (mating threshold) are mated. The initial threshold is set at $L/4$, where L is the length of the chromosomes. If no offspring are inserted into the new population then the threshold is reduced by one.

CHC also performs a form of heterogeneous HUX recombination using a special operator recombination [32]. HUX exchanged half the bits that differ between parents, where the bit position to exchange is determined randomly. CHC also employs a method of preventing incest. Before applying HUX to the both parents, the Hamming distance between them is measured. Only parents who are distinguished by some number of bits (mating threshold) are coupled. The initial threshold is stable at L / 4, where L is the length of the chromosomes. If no offspring are inserted into the new population then the threshold it decrements by one.

No mutation is applied during the recombination phase. Instead, when the population converges or the search stops making progress (i.e., the difference threshold has dropped to zero and no new offspring are being generated which are better than any member of the parent population) the population is reinitialized to introduce new diversity to the search. The chromosome representing the best solution found over the course of the search is used as a template to reseed the population. Reseeding of the population is accomplished by randomly changing 35% of the bits in the template chromosome to form each of the other $NC-1$ new chromosomes in the population. The search is then resumed.

3.4. Representation and Fitness Function

Let $S \subseteq HS$ be the subset of selected hyperrectangles that result from the run of a hyperrectangle selection algorithm. Hyperrectangle selection can be considered as a search problem to which EAs can be applied. We take into account two important issues: the specification of the representation of the solutions and the definition of the fitness function.

- *Representation:* We use a binary representation, a chromosome consists of N genes (one from each hyperrectangle in HS) with two possible values: 0 and 1. If the gene is 1, the associated hyperrectan-

gle is included in the subset represented by S. If this is 0, this is not included.

- *Fitness Function:* Let S be a subset of hyperrectangles of HS and be coded by a chromosome. We define a fitness function based on the accuracy (classification rate) and the Non-Monotonic Index (NMI) evaluated over TR through the formula.

$$Fitness(S) = (1-\lambda)\cdot(\beta\cdot(\alpha\cdot clas\_rat + (1-\alpha)\cdot perc\_red) + (1-\beta)\cdot cover) + \lambda\cdot NMI\_red. \quad (16)$$

clas_rat means the percentage of correctly classified objects from TR using S. *perc_red* is defined as

$$perc\_red = 100\cdot\frac{|HS|-|S|}{|HS|}. \quad (17)$$

cover refers the total coverage of examples in TR in the subset of selected hyperrectangles or, in other words, the number of examples of TR whose distance value has been equal to 0 (examples covered by hyperrectangles in S).
NMI_red is defined as

$$NMI\_red = 50 * \frac{NMI\_initial - NMI\_current}{NMI\_initial} \quad (18)$$

where *NMI_initial* is the number of monotonic violations computed by the predictions made for all the rules derived from the training set and divided by the total number of pairs of examples. *NMI_current* is similarly computed to *NMI_initial*, but instead using the rules selected by the chromosome. *NMI_red* represents the relative reduction of *NMI* achieved by each chromosome and it is weighted by a constant factor of 50 due to the fact that this measure represents low values with respect to classification and reduction rates.

The objective of the EAs is to maximize the fitness function defined, i.e., maximize the classification rate and coverage while minimizing the number of hyperrectangles selected and the anti-monotonic index. Although the fitness function defined is focused on discarding single trivial hyperrectangles (points), exceptions could be present in special cases where points are necessary to achieve high rates of accuracy. Thus, the necessity of using single hyperrectangles or not will be determined by the tradeoff accuracy-coverage and will be conditioned by the problem tackled.

Regarding the parameters, we preserved the value of $\alpha = 0.5$ as the best choice, due to the fact that it was analyzed in previous works related to instance selection [33,34,35]. We have conducted several experimental evaluations to estimate the best values of these parameters. For the sake of shortness, we determined that a suitable value for β should fall near 0.66 and the value for λ should be close to 0.25.

It is worth mentioning that our objective is to identify the best values of the parameters that configure the evolutionary approach in a general set up. It is true that for a specific classification problem, these values could be tuned in order to optimize the results achieved, but this may affect to other aspects, like efficiency or simplicity. General rules can be given about this topic:

- The number of evaluations and population size are the main factors for yielding good results in accuracy and simplicity. The raise of these values has a negative effect on efficiency. In larger problems, it may be necessary to increase both values, but we will show in the experimental study that values of 10,000 and 50 work appropriately, respectively.
- Parameters α and β allow us to obtain a desired trade-off between the accuracy and the number of hyperrectangles created. In the case of obtaining poor accuracy rates in a specific problem we have to increase α or decrease β. In contrary case, when the rules yielded are numerous and we are interested in producing simpler models, we have to increase β or decrease α. Besides, a large value of λ penalizes the maximization of the accuracy and minimization of the number of hyperrectangles in favor of the loss of anti-monotonic index.

Regarding to the specification of the classification conflicts, we employ the same mechanisms as shown in [20]. In short, they are:

- If no hyperrectangle covers the example, the class

of the nearest hyperrectangle defines the prediction.

- If various hyperrectangles cover the example, the one with lowest volume is the chosen to predict the class, allowing exceptions within generalizations.
  Our approach computes the volume of a hyperrectangle in the following way:

$$V_H = \prod_{j}^{m} L_j, \tag{19}$$

where L_j is computed for each condition as

$$L_j = \begin{cases} H_{\text{upper}} - H_{\text{lower}} \text{ if numeric and } H_{\text{upper}} \neq H_{\text{lower}}, \\ 1 \text{ if numeric and } H_{\text{upper}} = H_{\text{lower}}, \\ \frac{\text{num. values selected}}{vi} \text{ if nominal.} \end{cases} \tag{20}$$

4. Experimental Framework

In this section, we present the experimental framework developed to analyze and compare our proposal EHSMC-CHC with other well-known algorithms presented in this domain. The study will incorporate two types of data set. On the one hand, we will involve real life data sets coming from standard classification problems whose classes are not initially ordered. For our purposes, we instigate the sorting of the class values, assigning each category a number. The order among classes is made according to the arrangement that achieves the fewest number of monotonicity violations reported in each data set. On the other hand, we will use regression data sets whose output attribute has a numeric domain which will be ordered into categorical values. In this second subset, the order of classes is natural and we tackle real ordinal prediction problems.

In the next subsection, the experimental methodology will be specified, including data sets, performance measures, parameters of the algorithms and statistical validation.

4.1. Experimental methodology

The elements included in the framework are the following:

- *Data sets*: The study includes a total of 30 data sets, as a result of the joint selection of standard classification data sets (CLAS) whose class attribute is transformed into an ordinal attribute, assigning each category a number in such a way that the number of pairs of non-monotonic examples is minimized, and regression data sets (REGR) whose class attribute is discretized into 4 or 10 categorical values, keeping the class distribution balanced. Also, four classical ordinal classification data sets (ORD) are included in this study (*era*, *esl*, *lev* and *swd*) [1]. Their main characteristics are described in Table 1. They are classical data sets used in the classification scope and extracted from the UCI and KEEL repositories [37,38]. The run of the algorithms has been done following a 10-fold cross validation procedure (10-fcv).
  Table 1 shows the name of the data sets and their number of instances, variables and classes. In case of containing missing values, the total number of instances of the original data set appears in parenthesis. In this paper, these instances have been ignored.
- *Evaluation metrics*:
  - Accuracy (Acc), defined as the number of successful hits relative to the total number of classifications. It has been by far the most commonly used metric for assessing the performance of classifiers for years [39].
  - Mean Absolute Error (MAE), calculated as the sum of the absolute values of the errors and then dividing it by the number of classifications. The errors between the real label and the predicted label are estimated by the difference between the ordinal class values. Various studies conclude that MAE is one of the best performance metrics in ordered classification [40].
  - Monotonic Accuracy (MAcc), computed as standard Acc, but only considering those examples that completely fulfill the monotonicity constraints. In other words, non-monotonic

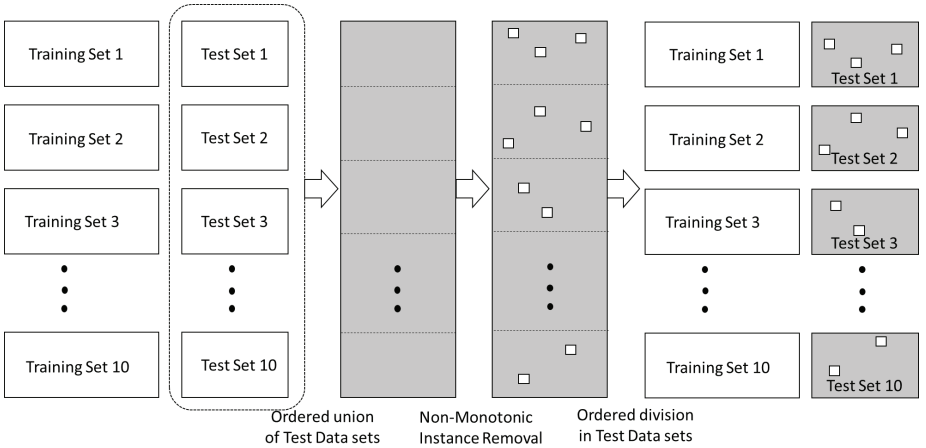


Figure 1: Process of transformation the 10-fcv to the new one by removing the non-monotonic instances from the test data sets.

comparable examples do not take part in the calculation of MAcc. The sense behind this is to simulate that future examples to be classified will check the monotonicity assumption. This metric serves as a manner of monotonicity level measurement of the predictions carried out. To address this, the process followed is presented in Figure 1 where the partitions used in 10-fcv are modified conserving the training data sets but removing the non-monotonic instances in the test data sets. We must point out that in the new validation data sets the test instances belong to the original data set, they are real instances, not artificial. In the same way for this new data sets obtained, we must highlight that the test data sets have been generated using different strategies than the noise removal filter or relabeling, searching to fair comparisons. The process presented in Figure 1 does not present any kind of randomness due to it is based in a deterministic greedy algorithm (see Algorithm 1).

- Monotonic Mean Absolute Error (MMAE), used to pursue the same goal of MAcc, but using MAE instead of Acc. Obviously, the data sets used in this case are the same than in the MAcc analysis.
- Non-Monotonicity Index (NMI) [41], defined as the number of clash-pairs divided by the total number of pairs of examples in the predictions made by an algorithm:

$$NMI = \frac{1}{n(n-1)} \sum_{x \in D} NClash(x), \qquad (21)$$

where x is an example from the data set D. $NClash(x)$ is the number of examples from D that do not meet the monotonicity restrictions (or clash) with x and n is the number of instances in D.

- Number of Rules (NRules), number of rules/hyperrectangles that compose the models produced by the algorithms.

- *Parameters configuration*: See Table 2.
- *Statistical procedures*: Several hypothesis testing procedures are considered to determine the most relevant differences found among the methods [30]. The use of non-parametric tests will be preferred to parametric ones, since the initial conditions that guarantee the reliability of the latter may not be satisfied. This could cause the statistical analysis to lose credibility. Friedman ranks test [31] is used to contrast the behavior of each algorithm. It highlights the significant differences between methods if they appear.

Table 1: Description of the 30 data sets used in the study.

Data set	Ins.	At.	Cl.	Type	Data set	Ins.	At.	Cl.	Type
appendicitis	106	7	2	CLAS	**hepatitis**	80 (155)	19	2	CLAS
bands	365 (539)	19	2	CLAS	**ionosphere**	351	33	2	CLAS
baseball10cl	337	16	10	REG	**iris**	150	4	3	CLAS
baseball4cl	337	16	4	REG	**machinecpu10cl**	209	6	10	REG
breast	277 (286)	9	2	CLAS	**machinecpu4cl**	209	6	4	REG
cleveland	297 (303)	13	5	CLAS	**movement_libras**	360	90	15	CLAS
dee10cl	365	6	10	REG	**newthyroid**	215	5	3	CLAS
dee4cl	365	6	4	REG	**pima**	768	5	2	CLAS
dermatology	358 (366)	34	6	CLAS	**sonar**	208	60	2	CLAS
ecoli	336	7	8	CLAS	**spectfheart**	267	4	2	CLAS
era	1000	4	9	ORD	**swd**	1000	10	4	ORD
esl	488	4	9	ORD	**vowel**	990	13	11	CLAS
german	1000	20	2	CLAS	**wdbc**	569	30	2	CLAS
glass	214	9	7	CLAS	**wine**	178	13	2	CLAS
haberman	306	3	2	CLAS	**wisconsin**	683 (699)	9	2	CLAS

Table 2: Parameters considered for the algorithms compared.

Algorithm	Parameters
Mk-NN	$k = \{1, 3\}$, distance = euclidean
OLM	modeResolution = conservative modeClassification = conservative
OSDL	classificationType = media, balanced = No weighted = No, tuneInterpolationParameter = No, lowerBound = 0, upperBound = 1 interpolationParameter = 0.5, interpolationStepSize = 10
MID	confidence = 0.25, 2 items per leaf, R = 1
EHSMC-CHC	Popul. Size = 50, Num. Evaluations = 10000 $\alpha = 0.5$, $\beta = 0.66$, $\lambda = 0.25$

Table 3: Results for Acc and MAE

	Acc						MAE					
	EHSMC-CHC	**MID**	**OLM**	**OSDL**	**M1-NN**	**M3-NN**	**EHSMC-CHC**	**MID**	**OLM**	**OSDL**	**M1-NN**	**M3-NN**
appendicitis	**0.8673**	0.8236	0.8018	0.7736	0.1855	0.1591	**0.1327**	0.1764	0.1982	0.2264	0.8145	0.8409
bands	0.6658	0.6194	0.3702	0.6219	**0.6859**	0.6250	0.3342	0.3806	0.6298	0.3781	**0.3141**	0.3750
baseball10cl	**0.2796**	0.2493	0.1988	0.2643	0.2494	0.2314	**1.3814**	1.4658	2.0048	1.5855	1.7870	1.8635
baseball4cl	**0.6647**	0.5850	0.5289	0.4983	0.4748	0.4961	**0.3678**	0.4802	0.5869	0.6028	0.6822	0.6611
breast	**0.7146**	0.6749	0.6817	0.5447	0.6272	0.6310	**0.2854**	0.3251	0.3183	0.4553	0.3728	0.3690
cleveland	0.5389	0.5517	**0.5657**	0.5655	0.5254	0.5253	0.9424	0.6931	0.8477	**0.6471**	0.7436	0.7774
dee10cl	**0.2875**	0.2633	0.1781	0.0985	0.2384	0.2766	1.3971	**1.2493**	2.6059	4.5396	1.8289	1.7619
dee4cl	**0.6493**	0.6053	0.4682	0.2522	0.5454	0.5809	**0.4085**	0.4221	0.7838	1.4927	0.5887	0.5259
dermatology	**0.9354**	0.9239	0.3800	0.1259	0.7898	0.8094	**0.1435**	0.1691	1.5331	1.7599	0.4625	0.4259
ecoli	**0.8126**	0.8007	0.5715	0.0324	0.5923	0.5893	**0.7016**	0.7738	1.5379	2.3482	1.4875	1.5829
era	0.1620	**0.2740**	0.1690	0.2320	0.1430	0.1410	2.1480	1.3630	2.1500	**1.2850**	2.3360	2.3400
esl	0.5353	0.6784	0.5963	**0.6785**	0.3956	0.3915	0.7750	0.3525	0.4324	**0.3521**	0.7418	0.7459
german	0.6970	0.6920	**0.7110**	0.3740	0.6370	0.6330	0.3030	0.3080	**0.2890**	0.6260	0.3630	0.3670
glass	0.6975	0.6200	0.3244	0.3379	**0.7172**	0.6783	0.5602	0.7595	1.7816	1.7729	**0.5408**	0.6460
haberman	**0.7514**	0.7120	0.3496	0.7158	0.4803	0.4771	**0.2486**	0.2880	0.6504	0.2842	0.5197	0.5229
hepatitis	**0.8343**	0.7735	0.2497	0.8118	0.8222	0.8030	**0.1657**	0.2265	0.7503	0.1882	0.1778	0.1970
ionosphere	**0.9174**	0.9002	0.6269	0.7407	0.7071	0.6702	**0.0826**	0.0998	0.3731	0.2593	0.2929	0.3298
iris	0.9467	**0.9533**	0.9333	0.3667	**0.9533**	**0.9533**	0.0533	**0.0467**	0.0667	0.9067	**0.0467**	**0.0467**
machinecpu10cl	0.2864	0.3298	0.3205	**0.3493**	0.3105	0.2962	1.6021	**1.2550**	1.4960	1.2631	1.4176	1.4938
machinecpu4cl	0.6457	0.6267	0.6219	0.5736	0.6267	**0.6505**	0.4352	**0.3924**	0.4452	0.4843	0.4117	0.3929
movement_libras	**0.7222**	0.6667	0.3000	0.0611	0.6139	0.5778	**1.2444**	1.5667	4.8306	7.0333	1.7528	2.0944
newthyroid	**0.9491**	0.9264	0.5636	0.1439	0.7957	0.7723	**0.0833**	0.1104	0.7662	0.8654	0.3165	0.3578
pima	0.6787	0.7265	0.7110	0.6563	**0.7358**	0.7320	0.3213	0.2735	0.2890	0.3437	**0.2642**	0.2680
sonar	0.7312	0.7460	0.4662	0.5419	**0.8555**	0.8307	0.2688	0.2540	0.5338	0.4581	**0.1445**	0.1693
spectfheart	**0.7942**	0.7608	0.1983	0.7830	0.7236	0.7050	**0.2058**	0.2392	0.8017	0.2170	0.2764	0.2950
swd	0.3930	0.5660	0.4040	**0.5820**	0.3360	0.3360	0.6630	0.4670	0.7510	**0.4350**	0.8810	0.8810
vowel	0.7838	0.7990	0.0909	0.0859	**0.9949**	0.9788	0.4313	0.5586	5.0000	4.8273	**0.0101**	0.0444
wdbc	**0.9364**	0.9350	0.3392	0.3568	0.3465	0.3341	**0.0636**	0.0650	0.6608	0.6432	0.6535	0.6659
wine	**0.9431**	0.9160	0.3042	0.3261	0.7542	0.8147	**0.0569**	0.0840	0.9536	0.9487	0.3180	0.2408
wisconsin	0.8230	0.9460	0.8872	0.9593	**0.9622**	**0.9622**	0.1770	0.0540	0.1128	0.0407	**0.0378**	**0.0378**
Average	0.6881	**0.6882**	0.4637	0.4485	0.5942	0.5887	0.5328	**0.4966**	1.1393	1.2423	0.6861	0.7107

Table 4: Results for MAcc and MMAE

	MAcc						MMAE					
	EHSMC-CHC	**MID**	**OLM**	**OSDL**	**M1-NN**	**M3-NN**	**EHSMC-CHC**	**MID**	**OLM**	**OSDL**	**M1-NN**	**M-3NN**
appendicitis	**0.9431**	0.9111	0.8018	0.7736	0.1500	0.1167	**0.0569**	0.0889	0.1982	0.2264	0.2111	0.8500
bands	0.6663	0.6175	0.3724	0.6253	**0.6896**	0.6281	**0.3337**	0.3825	0.6276	0.3747	0.3635	0.3104
baseball10cl	0.2920	0.2700	**0.5783**	0.5350	0.2841	0.2647	**1.3342**	1.3383	0.5048	0.5411	1.8795	1.5814
baseball4cl	**0.7119**	0.6155	0.2237	0.3072	0.5209	0.5464	**0.3123**	0.4414	1.9201	1.3864	0.5367	0.6134
breast	**0.7902**	0.7534	0.7769	0.5710	0.6958	0.6996	**0.2098**	0.2466	0.2231	0.4290	0.2309	0.3042
cleveland	**0.5682**	0.5517	0.5575	0.5673	0.5536	0.5533	0.8920	0.6931	0.8448	**0.6583**	0.6963	0.6811
dee10cl	0.3204	0.2960	**0.5069**	0.2439	0.2776	0.3237	1.3093	1.1540	**0.7144**	1.5358	1.3429	1.4843
dee4cl	**0.6807**	0.6350	0.2085	0.0991	0.5898	0.6270	**0.3728**	0.3829	2.5892	4.7203	0.3818	0.5036
dermatology	**0.9370**	0.9313	0.4037	0.1334	0.8357	0.8489	**0.1360**	0.1583	1.5298	1.7417	0.3153	0.3405
ecoli	**0.8521**	0.8366	0.5653	0.0235	0.6696	0.6566	**0.4925**	0.6243	1.5889	2.3213	1.0180	1.0977
era	0.1626	**0.7239**	0.2837	0.5134	0.1811	0.1836	1.9900	**0.4726**	1.7797	0.5651	2.5840	2.0704
esl	0.7429	**0.8340**	0.6758	0.8119	0.4476	0.4399	0.2752	**0.1861**	0.3318	0.2035	1.6437	0.6796
german	0.7035	0.6973	**0.7260**	0.3815	0.6496	0.6465	0.2965	0.3027	**0.2740**	0.6185	0.3556	0.3504
glass	0.6983	0.6200	0.3227	0.3411	**0.7221**	0.6832	0.5580	0.7595	1.7892	1.7667	0.6372	**0.5307**
haberman	**0.9460**	0.8009	0.2777	0.7157	0.5857	0.5814	**0.0540**	0.1991	0.7223	0.2843	0.8754	0.4143
hepatitis	**0.8343**	0.7735	0.2497	0.8118	0.8222	0.8030	**0.1657**	0.2265	0.7503	0.1882	0.2000	0.1778
ionosphere	**0.9290**	0.9111	0.6576	0.7703	0.7416	0.7027	**0.0710**	0.0889	0.3424	0.2297	0.2212	0.2584
iris	0.9595	0.9590	0.9267	0.3667	**0.9662**	**0.9662**	0.0405	0.0410	0.0733	0.9133	**0.0271**	0.0338
machinecpu10cl	0.3302	0.3907	**0.6707**	0.6252	0.3831	0.3619	1.4441	1.0995	**0.3969**	0.4158	1.2460	1.2553
machinecpu4cl	0.6660	0.6640	0.3719	0.4290	0.6815	**0.7026**	0.4119	0.3570	1.3555	1.0573	**0.3382**	0.3496
movement_libras	**0.7226**	0.6715	0.3145	0.0643	0.6434	0.6049	**1.2474**	1.5329	4.7228	6.9883	1.5169	1.4765
newthyroid	**0.9561**	0.9366	0.5721	0.1504	0.8310	0.8065	**0.0687**	0.0924	0.7486	0.8591	0.3393	0.2562
pima	0.6787	0.7587	0.7005	0.6563	**0.7894**	0.7863	0.3213	0.2413	0.2995	0.3437	**0.2062**	0.2106
sonar	0.7457	0.7460	0.4662	0.5419	**0.8555**	0.8307	0.2543	0.2540	0.5338	0.4581	**0.1445**	0.1693
spectfheart	**0.7917**	0.7575	0.2008	**0.7917**	0.7315	0.7110	**0.2083**	0.2425	0.7992	**0.2083**	0.2880	0.2685
swd	0.4101	0.8485	0.4682	**0.8878**	0.3876	0.3876	0.6151	0.1561	0.6393	**0.1122**	1.0399	0.7730
vowel	0.7838	0.7990	0.0909	0.0859	**0.9949**	0.9788	0.4313	0.5586	5.0000	4.8273	0.0444	**0.0101**
wdbc	**0.9364**	0.9315	0.3073	0.2970	0.4626	0.4495	**0.0636**	0.0685	0.6927	0.7030	0.6184	0.5374
wine	**0.9462**	0.9229	0.3258	0.3258	0.7947	0.8595	**0.0538**	0.0771	0.9324	0.9490	0.2169	0.2581
wisconsin	0.8230	0.9529	0.8641	0.9571	**0.9708**	**0.9708**	0.1770	0.0471	0.1359	0.0429	**0.0280**	0.0292
Average	0.7176	**0.7373**	0.4823	0.4801	0.6303	0.6240	0.4732	**0.4171**	1.1020	1.1890	0.6516	0.5959

Table 5: Results for NMI and MRules

	NMI(%)						NRules		
	EHSMC-CHC	**MID**	**OLM**	**OSDL**	**M1-NN**	**M3-NN**	**EHSMC-CHC**	**MID**	**OLM**
appendicitis	5.44	**0.00**	0.22	**0.00**	14.34	15.37	9.8	**8.7**	23.3
bands	**0.00**	**0.00**	**0.00**	0.01	0.01	0.01	**49**	60	102.9
baseball10cl	**1.13**	10.68	12.58	12.45	14.09	13.98	**33.7**	93.2	215.7
baseball4cl	**0.00**	11.14	13.48	13.36	12.13	12.26	**40.2**	61.2	123.4
breast	**0.16**	0.64	3.48	2.35	3.78	3.78	25.5	142.9	**18.4**
cleveland	**0.00**	0.40	0.89	0.98	1.01	0.89	**13.3**	60.1	35.7
dee10cl	0.84	2.82	**0.02**	4.40	4.99	4.95	**27.5**	92.5	94.5
dee4cl	0.56	2.41	**0.02**	4.86	4.26	4.13	35.7	55.1	**27.5**
dermatology	0.02	**0.00**	0.05	0.19	0.16	0.16	13.3	**13.2**	36.9
ecoli	0.17	11.31	**0.02**	12.72	13.40	13.17	**33.4**	37.7	75.8
era	**8.79**	17.03	13.49	12.59	12.25	12.27	**9.4**	26.9	38.2
esl	**0.44**	61.70	61.92	61.56	63.28	63.30	**24.4**	55.4	39.6
german	**0.00**	0.02	0.06	0.05	0.05	0.06	**38.6**	288.8	288.8
glass	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**28.6**	33.4	191
haberman	3.46	**0.29**	0.56	3.80	19.72	19.74	25.4	**8.9**	25.2
hepatitis	**0.00**	**0.00**	0.10	0.10	0.40	**0.00**	**5.3**	9.8	17.3
ionosphere	0.09	**0.00**	0.29	0.23	0.45	0.40	**24.1**	24.5	116.8
iris	**0.00**	23.00	3.89	20.97	23.05	23.05	6.1	**4.4**	5.5
machinecpu10cl	**0.87**	5.18	4.88	4.92	6.78	6.93	**17.9**	53.3	58.9
machinecpu4cl	**0.20**	3.95	6.24	6.19	5.05	4.89	21.3	35.6	**20.4**
movement_libras	0.06	0.15	**0.02**	0.57	0.51	0.51	**44.1**	55.2	51.2
newthyroid	0.20	**0.05**	0.09	3.32	1.76	2.16	**8**	11.5	33.3
pima	**0.01**	3.47	0.12	3.67	5.50	5.52	**37.9**	52.7	39.3
sonar	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	28.8	**22.3**	187.2
spectfheart	**0.00**	**0.00**	**0.00**	**0.00**	0.03	0.03	**22.7**	27.6	239.5
swd	**1.50**	10.73	10.11	9.04	9.29	9.29	**4.1**	57.8	41.6
vowel	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	142.6	**104.1**	891
wdbc	0.01	**0.00**	**0.00**	1.37	3.35	3.34	17	**16.7**	64.3
wine	**0.00**	**0.00**	0.01	0.01	0.33	0.26	9.4	**7.9**	17.9
wisconsin	**0.00**	36.83	37.66	32.46	37.55	37.85	**6.2**	22.8	9.7
Average	**0.80**	6.73	5.67	7.07	8.58	8.61	**26.78**	51.47	104.36

Algorithm 1 Greedy Non-Monotonic Instances Removal Algorithm for test partitions.

```
function GREEDYREMOVE(T - dataset)
    while NumberOfTotalCollisions(T)>0 do
        maxColis=0, instancSelec=0;
        for each instance i in T do
            Colis=NumberOfCollisionsProduced(i,T);
            if Colis>maxColis then
                maxColis=Colis, instancSelec=i;
            end if
        end for
        T = T - instancSelec;
    end while
    return T
end function
```

5. Results and Analysis

This section shows the results obtained in the experimental study as well as the analysis based on them. Tables 3 and 4 report the results measured by accuracy, MAE and their monotonic versions explained before (MAcc and MMAE) in test data. Furthermore, Table 5 shows the NMI measured in the prediction for each algorithm and the number of rules reported for those algorithms which produce rules: MID and OLM. The best case in each data set is stressed in bold. The last row in each table shows the average measured by considering all data sets.

At a glance, observing Tables from 3 to 5, we can make the following analyses:

- The EHSMC-CHC proposal yields the second best average result in accuracy over test data, the first best average results is offered by MID. It is slightly more accurate than Mk-NN and much more accurate than OLM and OSDL.
- The MID proposal gets the best average result in MAE over test data. EHSMC-CHC obtains a very close result to MID, but it is clearly better than OLM and OSDL.
- By considering the monotonic measures, MAcc and MMAE, EHSMC-CHC is again slightly outperformed by MID, obtaining the second best results. Also, both are clearly superior to the rest of algorithms.
- The EHSMC-CHC proposal obtains the best average result in NMI over test data. The differences in NMI are very clear, thus our approach obtains very good monotonic predictions.
- Finally, the number of rules required by EHSMC-CHC is the lowest regarding the three algorithms compared. Hence, the models produces are simpler than the ones produced by MID and OLM.

In summary, our proposal EHSMC-CHC achieves competitive results in terms of accuracy and MAE by considering both classical and monotonic measures. It requires few rules and yields more monotonic models. Clearly, EHSMC-CHC is as good as MID in precision performance, but it requires a much lower number of generalized instances for building the model. With respect to the monotonicity in predictions, the NMI associated with EHSMC-CHC is also better than the reported by MID. Hence, EHSMC-CHC requires less rules and makes predictions with a higher monotonic index with similar behavior to MID.

Table 6 collects the results of applying the Wilcoxon test to EHSMC-CHC and the rest of the methods studied in this paper. In each one of the cells, three symbols can appear: +, = or –. They represent either that EHSMC-CHC outperforms (+), is similar to (=) or is worse (–) in performance than the rest of the methods. The value in brackets is the p-value obtained in the comparison. Other statistical

Table 6: Wilcoxon p-values reported comparing with EHSMC-CHC

Algorithm	Acc	MAE	MAcc	MMAE	NMI	NRules
MID	=(0.2580)	=(0.7577)	=(0.1642)	=(0.6071)	+(0.0034)	+(0.0009)
OLM	+(0.0001)	+(0.0002)	+(0.0001)	+(0.0020)	+(0.0258)	+(0.0000)
OSDL	+(0.0005)	+(0.0040)	+(0.0017)	+(0.0086)	+(0.0000)	—
M1-NN	+(0.0052)	+(0.0080)	+(0.0248)	+(0.0081)	+(0.0000)	—
M3-NN	+(0.0015)	+(0.0035)	+(0.0081)	+(0.0137)	+(0.0000)	—

Table 7: Rankings reported by the test of Friedman

Algorithm	Acc	MAE	MAcc	MMAE	NMI	NRules
EHSMC-CHC	**2.3**	**2.4333**	**2.3833**	**2.45**	**1.8833**	**1.4333**
MID	2.6667	2.5	2.9333	2.8667	2.6	2.05
OLM	4.5	4.7667	4.35	4.4333	3.2333	2.5167
OSDL	4.1333	4.1	4.3333	4.35	3.7833	—
M1-NN	3.4333	3.2833	3.3	3.4667	4.8167	—
M3-NN	3.9667	3.9167	3.7	3.4333	4.6833	—

studies can be performed by using non-parametric multiple comparison tests. These types of procedures study the set of results obtained by all the algorithms to compute a ranking that takes into account the multiple comparison. The smaller the ranking, the better the algorithm is. Rankings of the Friedman test are depicted in Table 7.

The results of the statistical tests allow us to highlight the following observations:

- EHSMC-CHC and MID are the best approaches compared with the other techniques regarding accuracy and MAE. No statistical differences are registered between both, thus they behave equally in performance.
- Our algorithm significantly outperforms OLM, OSDL, M1-NN and M3-NN in Acc, MAE, MAcc and MMAE.
- Considering NMI and NRules, our approach is the best according to the statistical report.

6. Conclusions

The purpose of this paper was to present a proposal of hyperrectangle selection for monotonic classification through evolutionary algorithms. The novel algorithm named *EHSMC-CHC* constitutes a novel approach for nested generalized exemplar learning in classification with monotonicity constrains. It builds a beginning set of hyperrectangles by using a heuristic on training data and then it carries out a selection process focused on maximizing the performance of several objectives: accuracy, coverage of examples and reduction of the monotonicity violations registered in the model learned with the lowest possible number of hyperrectangles.

The results showed that *EHSMC-CHC* will allow us to produce very accurate models with a low number of hyperrectangles with very few monotonicity ruptures. We have compared it with the most important monotonic learning approaches, including both rule and instance based learners, and the performance of the models obtained by *EHSMC-CHC* in several data sets overcomes the offered by them.

Acknowledgments

This work is supported by the National Research Project TIN2014-57251-P.

References

1. A. Ben-David, L. Serling and Y. Pao, "Learning and classification of monotonic ordinal concepts," *Computational Intelligence*, **5**, 45–49 (1989).
2. W. Kotłowski and R. Słowiński, "On nonparametric ordinal classification with monotonicity constraints," *IEEE Transactions on Knowledge and Data Engineering*, **25:11**, 2576–2589 (2013).
3. C. -C. Chen and S. -T. Li, "Credit rating with a monotonicity-constrained support vector machine model," *Expert Systems with Applications*, **41:16**, 7235–7247 (2014).
4. A. Ben-David, L. Sterling and T. Tran, "Adding monotonicity to learning algorithms may impair their accuracy," *Expert Systems with Applications*, **36:3**, 6627-6634 (2009).
5. A. Gegov, N. Gobalakrishnan and D. Sanders, "Filtration of Non-Monotonic Rules for Fuzzy Rule Base Compression," *International Journal of Computational Intelligence Systems*, **7:2**, 382-400 (2014).
6. S. García, J. Luengo and F. Herrera, "Data Preprocessing in Data Mining," Springer, 2015.
7. R. Potharst, A. Ben-David and M. C. van Wezel, "Two algorithms for generating structured and unstructured monotone ordinal datasets," *Engineering Applications of Artificial Intelligence*, **22:4-5**, 491-496 (2009).
8. Q. Hu, W. Pan, L. Zhang, D. Zhang, Y. Song, M. Guo and D. Yu, "Feature Selection for Monotonic Classification," *IEEE Transactions on Fuzzy Systems*, **20:1**, 69-81 (2012).
9. W. Pan, Q. Hu, Y. Song and D. Yu, "Feature selection for monotonic classification via maximizing monotonic dependency," *International Journal of Computational Intelligence Systems*, **7:3**, 543-555 (2014).
10. A. Ben-David, "Monotonicity maintenance in information-theoretic machine learning algorithms," *Machine Learning*, **19:1**, 29-43 (1995).
11. R. Potharst and A. J. Feelders, "Classification trees for problems with monotonicity constraints," *SIGKDD Explorations*, **4:1**, 1-10 (2002).
12. K. Dembczyński, W. Kotłowski, and R. Słowiński, "Learning rule ensembles for ordinal classification with monotonicity constraints," *Fundamenta Informaticae*, **94:2**, 163-178 (2009).
13. Q. Hu, X. Che, Z. Lei, D. Zhang, M. Guo and D. Yu, "Rank entropy-based decision trees for monotonic classification," *IEEE Transactions on Knowledge Data Engineering*, **24:11**, 2052-2064 (2012).
14. H. Daniels and M. Velikova, "Monotone and partially monotone neural networks," *IEEE Transactions on Neural Networks*, **21:6**, 906-917 (2010).
15. A. Ben-David, "Automatic generation of symbolic multiattribute ordinal knowledge-based DSSs: methodology and applications," *Decision Sciences*, **23**, 1357-1372 (1992).
16. S. Lievens, B. De Baets and K. Cao-Van, "A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting," *Annals of Operational Research*, **163:1**, 115-142 (2008).
17. W. Duivesteijn and A. Feelders, "Nearest neighbour classification with monotonicity constraints," In *ECML/PKDD* , **1**, 301-316 (2008).
18. D. W. Aha, D. Kibler and M. K. Albert, "Instance-based learning algorithms," *Machine Learning*, **6:1**, 37-66 (1991).
19. P. Flach, "Machine Learning: The Art and Science of Algorithms that Make Sense of Data," Cambridge University Press, 2012.
20. S. Salzberg, "A nearest hyperrectangle method," *Machine Learning*, **6:1**, 151-276 (1991).
21. T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, **13:1**, 21-27 (1967).
22. J. Fürnkranz, D. Gamberger and N. Lavrač, "Foundations of Rule Learning," Springer, 2012.
23. D. Wettschereck and T. G. Dietterich, "An experimental comparison of the nearest neighbor and nearest-hyperrectangle algorithms," *Machine Learning*, **19**, 5-27 (1995).
24. P. Domingos, "Unifying instance-based and rule-based induction," *Machine Learning*, **24**, 141-168 (1996).
25. O. Luaces and A. Bahamonde, "Inflating examples to obtain rules," *International Journal of Intelligent Systems*, **18:11**, 1113-1142 (2003).
26. D. Simon, "Evolutionary Optimization Algorithms," Wiley, 2013.
27. A. A. Freitas, "Data Mining and Knowledge Discovery with Evolutionary Algorithms," Springer, 2002.
28. S. García, J. Derrac, J. Luengo, C. J. Carmona and F. Herrera, "Evolutionary Selection of Hyperrectangles in Nested Generalized Exemplar Learning," *Applied Soft Computing*, **11:3**, 3032-3045 (2011).
29. S. García, J. Derrac, I. Triguero, C. J. Carmona and F. Herrera, "Evolutionary-based selection of generalized instances for imbalanced classification," *Knowledge-Based Systems*, **25**, 3-12 (2012).
30. J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, **7**, 1-30 (2006).
31. S. García, A. Fernández, J. Luengo and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power," *Information Sciences*, **180**, 2044-2064 (2010).
32. L. J. Eshelman, "The CHC adaptive search algorithm: How to safe search when enganging in nontraditional genetic recombination, " in G. Rawlings (Ed.), "Foundations of Genetic Algorithms". Morgan Kaufmann,

1991.
33. J. -R. Cano, F. Herrera and M. Lozano, "Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study," *IEEE Transactions on Evolutionary Computation*, **7:6**, 561-575 (2003).
34. J. Derrac, S. García and F. Herrera, "A survey on evolutionary instance selection and generation," *International Journal of Applied Metaheuristic Computing*, **1:1**, 60-92 (2010).
35. J. Derrac, C. Cornelis, S. García and F. Herrera, "Enhancing evolutionary instance selection algorithms by means of fuzzy rough set based feature selection," *Information Sciences*, **186:1**, 73-92 (2012).
36. K. Cao-Van, "Supervised Ranking, from semantics to algorithms," *Ph.D. dissertation*, Ghent University (2003).
37. M. Lichman, "UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]". Irvine, CA: University of California, School of Information and Computer Science. (2013).
38. J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, " KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework," *Journal of Multiple-Valued Logic and Soft Computing*, **17:2-3**, 255–287 (2011).
39. R. C. Prati, G. Batista, M. Monard, "A survey on graphical methods for classification predictive performance evaluation", *IEEE Transactions on Knowledge and Data Engineering*, **23:11**, 1601–1608 (2011).
40. N. Japkowicz, M. Shah, "Evaluating Learning Algorithms. A classification Perspective", Cambridge University Press, 2014.
41. H. Daniels, M. Velikova, "Derivation of monotone decision models from noisy data", *IEEE Transactions on Systems, Man and Cybernetics - Part C*, **36**, 705–710 (2006).

Capítulo 4

Conclusiones y Trabajos Futuros

4.1. Conclusiones

En esta tesis se ha abordado el problema de la clasificación monotónica a través de técnicas que hibridan el aprendizaje basado en instancias con la inducción de reglas, con el objetivo de analizar, diseñar e implementar distintas técnicas para obtener modelos predictivos monotónicos precisos y más simples.

El objetivo inicial de la tesis era obtener un profundo conocimiento del área, especialmente en cuanto a clasificadores monotónicos y paradigmas existentes en la literatura especializada. Para ello, se ha llevado a cabo un estudio teórico de los principales métodos propuestos en la literatura, centrado en extraer las características más comunes entre ellos y las que son únicas para cada enfoque. Los resultados de este estudio constituyen la realización de una taxonomía que sirve de apoyo a la investigación realizada durante esta tesis. Además, se concluye que la aplicación de técnicas *soft computing* está aún por explorar en este campo.

El segundo objetivo fue la formalización de la teoría de aprendizaje basada en ejemplos anidados generalizados en la clasificación con restricciones monotónicas

y la propuesta de un algoritmo voraz llamado MoNGEL aplicado a bases de datos completamente monotónicas. Nuestro algoritmo crea un conjunto inicial de reglas o hiperrectángulos conservando las propiedades monotónicas globales desde el conjunto de entrenamiento y lleva a cabo un proceso de generalización enfocado en maximizar la precisión a la vez que minimiza el número de instancias que se retienen. Los resultados indicaron que MoNGEL permite obtener modelos monotónicos muy precisos con pocas reglas y, comparado con el estado-del-arte, nuestra propuesta resulta ser muy competitiva.

Finalmente, el tercer objetivo asociado al tratamiento de problemas de clasificación monotónicos reales con imperfecciones en los datos se logró con la propuesta de un algoritmo de selección de hiperrectángulos basado en algoritmos evolutivos llamado EHSMC-CHC. El aprendizaje de modelos monotónicos a partir de bases de datos no completamente monotónicas está poco explorada y es una situación muy común en aplicaciones reales. El algoritmo construye un conjunto inicial de hiperrectángulos mediante una heurística sencilla en el conjunto de entrenamiento. Después lleva a cabo un proceso de selección evolutiva centrado en maximizar varios objetivos simultáneamente: la precisión del modelo, la cobertura de ejemplos y la reducción del número de violaciones de monotonicidad que se registran en el propio modelo con el menor número de hiperrectángulos. Los resultados obtenidos son muy prometedores y en muchos de los casos de estudio, nuestra propuesta mejora significativamente a las técnicas de aprendizaje basado en instancias y reglas del estado-del-arte.

4.2. Trabajos Futuros

El trabajo llevado a cabo durante esta tesis ha señalado nuevas lineas prometedoras de investigación, tanto para mejorar el rendimiento de los modelos propuestos, como para aplicarlos a nuevos problemas emergentes y con nuevas tecnologías.

Soluciones Big Data

Se espera que el *Big Data* sea uno de los principales desafíos para minería de datos en un futuro cercano [FdRL+14]. Las inmensas cantidades de datos que están disponibles en muchos campos ofrecen excelentes oportunidades de investigación. Por un lado, hay una urgente necesidad de sistemas que son capaces de

tratar con dichas grandes cantidades de datos de una manera eficiente y escalable. Por otro lado, todos los datos pueden proporcionar nuevo conocimiento e información para solucionar nuevos problemas.

En clasificación monotónica, la disposición de grandes cantidades de datos es una realidad en aplicaciones reales relacionadas con la banca o académicas. El uso de aproximaciones *Big Data* para procesar y generar modelos monotónicos en grandes volúmenes de datos es un reto que todavía no se ha afrontado. Las técnicas de reducción de datos para *Big Data* ([TPB+15]), junto a estructuras de particionamiento del espacio para organizar puntos en espacios multidimensionales, tales como *KD-Trees*, *Ball Trees* o *hashing* local sensible [LMG06] son ideales para empezar esta futura línea de investigación.

Clasificación Monotónica mediante Hiperrectángulos Difusos

En la literatura especializada, se ha propuesto alguna idea sobre el aprendizaje basado en ejemplos anidados generalizados difusos [dSLdCNR07]. La extensión del modelo difuso de hiperrectángulos tiene cabida también en la clasificación monotónica puesto que uno de los principales objetivos de las sistemas difusos para aprendizaje es aportar un mayor grado de simpleza en los modelos obtenidos.

Un modelo basado en hiperrectángulos difusos permitiría *fuzzificar* las reglas, como en los sistemas difusos basados en reglas [CdJH99]; *fuzzificar* los vecindarios y distancias, como en el algoritmo de vecinos cercanos difusos [DGH14], o ambos a la vez. Además, es factible incluir optimización genética en ambos procesos [FLdJH15, DCGH16], consiguiendo modelos híbridos muy compactos y adaptables a cualquier problema.

Técnicas y Problemas de Optimización Complejos

En esta tesis, se ha abordado la optimización de la selección de un subconjunto de hiperrectángulos previamente creados. Sin embargo, existen problemas de optimización más complejos que se pueden aplicar también al aprendizaje mediante ejemplos anidados generalizados. Entre éstos, cabe destacar la modificación de la forma, ya sea por reducción o por alargamiento de los hiperrectángulos en una o más dimensiones, o el ajuste del posicionamiento de los mismos en el espacio multidimensional.

Estos problemas de optimización requieren codificaciones más potentes de soluciones basadas en números reales. Además, existen técnicas evolutivas y operadores que están específicamente diseñados para tratar problemas de codificación real complejos, como pueden ser las técnicas de evolución diferencial ([DS11]) o la optimización basadas en arrecifes de coral [SSS14].

Extensiones del Problema de la Clasificación Monotónica

Se puede hablar de posibles extensiones del problema de la clasificación monotónica a partir de diferentes factores. Por ejemplo, se podría considerar el priorizar algunos atributos de entrada sobre otros, dando lugar a diferentes grados de violaciones de monotonicidad. Actualmente, la clasificación monotónica maneja un enfoque nítido de la restricción: un ejemplo es o no es monotónico con respecto a otro. Pero también se podría considerar la posibilidad de darle una graduación a partir de la semántica e importancia de los atributos de entrada [GG16].

Por otro lado, la clasificación monotónica se ve como una extensión natural y lógica de la clasificación ordinal. Sin embargo, otros paradigmas predictivos que requieren interpretación de los resultados por parte de los usuarios se pueden beneficiar de la generación de modelos monotónicos en situaciones reales, como puede ser el problema del descubrimiento de subgrupos [NLW09], clasificación semi-supervisada [TGH15] o clasificación multi-etiqueta o multi-dominio [HCRdJ16].

4.3. Publicaciones Adicionales

Junto a las publicaciones en revista, la tesis también tiene asociada un par de contribuciones a congresos internacionales y nacionales que se detallan a continuación.

4.3.1. A Nearest Hyperrectangle Monotonic Learning Method

- Título: A Nearest Hyperrectangle Monotonic Learning Method.
- Autores: Javier García, José-Ramón Cano y Salvador García.
- Congreso: The 11th International Conference on Hybrid Artificial Intelligent Systems (HAIS).
- Serie: Lecture Notes in Computer Science 9648.
- Fecha y Lugar de Celebración: del 18 al 20 de Abril de 2016, Sevilla (España).
- Páginas: 311 – 322.

4.3.2. Hyperrectangles Selection for Monotonic Classification by Using Evolutionary Algorithms

- Título: Hyperrectangles Selection for Monotonic Classification by Using Evolutionary Algorithms.
- Autores: Javier García, José-Ramón Cano y Salvador García.
- Congreso: XVII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 2016)
- Fecha y Lugar de Celebración: del 13 al 16 de Septiembre de 2016, Salamanca (España).

Bibliografía

[AA16] Almadni D. y Abhari A. (2016) Comparative analysis of classification models in diagnosis of type 2 diabetes. En *Proceedings of the Modeling and Simulation in Medicine Symposium*, MSM '16, páginas 7:1–7:5. Society for Computer Simulation International, San Diego, CA, USA.

[Agg15] Aggarwal C. C. (2015) *Data Mining: The Textbook*. Springer Publishing Company, Incorporated.

[Aha97] Aha D. W. (Ed.) (1997) *Lazy Learning*. Springer.

[BD92] Ben-David A. (1992) Automatic generation of symbolic multiattribute ordinal knowledge-based dsss: methodology and applications. *Decision Sciences* 23: 1357–1372.

[BD95] Ben-David A. (1995) Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning* 19(1): 29–43.

[BDSP89] Ben-David A., Sterling L., y Pao Y. H. (1989) Learning, classification of monotonic ordinal concepts. *Computational Intelligence* 5: 45–49.

[BDST09] Ben-David A., Sterling L., y Tran T. (2009) Adding monotonicity to learning algorithms may impair their accuracy. *Expert Systems with Applications* 36(3): 6627–6634.

[BES09] Baccianella S., Esuli A., y Sebastiani F. (2009) Evaluation measures for ordinal regression. En *Intelligent Systems Design and*

Applications, 2009. ISDA'09. Ninth International Conference on, páginas 283–287. IEEE.

[BH02] Bolton R. J. y Hand D. J. (08 2002) Statistical fraud detection: A review. *Statist. Sci.* 17(3): 235–255.

[Bir67] Birkhoff G. (1967) Lattice theory. En *Colloquium Publications*, volumen 25. Amer. Math. Soc., 3. edition.

[BO16] Brookhouse J. y Otero F. E. B. (2016) *Monotonicity in Ant Colony Classification Algorithms*, páginas 137–148. Springer International Publishing, Cham.

[BSS10] Blaszczynski J., Slowinski R., y Stefanowski J. (2010) *Rough Sets and Current Trends in Computing: 7th International Conference, RSCTC 2010, Warsaw, Poland, June 28-30,2010. Proceedings*, chapter Ordinal Classification with Monotonicity Constraints by Variable Consistency Bagging, páginas 392–401. Springer Berlin Heidelberg, Berlin, Heidelberg.

[CdJH99] Cordón O., del Jesus M. J., y Herrera F. (1999) A proposal on reasoning methods in fuzzy rule-based classification systems. *International Journal of Approximate Reasoning* 20(1): 21–45.

[CHSG14] Cruz M., Hervas C., Sanchez J., y Gutierrez P. (2014) Metrics to guide a multi-objective evolutionary algorithm for ordinal classification. *Neurocomputing* 135: 21 – 31.

[CS11] Cardoso J. S. y Sousa R. (2011) Measuring the performance of ordinal classification. *International Journal of Pattern Recognition and Artificial Intelligence* 25(8): 1173–1195.

[CS13] Chitra K. y Subashini B. (July 2013) Automatic credit approval using classification method. *International Journal of Scientific & Engineering Research* 4: 2026–2029.

[CV03] Cao-Van K. (2003) *Supervised Ranking, from semantics to algorithms*. Ph.D. dissertation, Ghent University.

[CVB03] Cao-Van K. y Baets B. D. (2003) Growing decision trees in an ordinal setting. *International Journal of Intelligent Systems* 18(7): 733–750.

[DAB97] Daniel A. Bloch B. W. S. (1997) Monotone discriminant functions and their applications in rheumatology. *Journal of the American Statistical Association* 92(437): 144–153.

[DCGH16] Derrac J., Chiclana F., García S., y Herrera F. (2016) Evolutionary fuzzy k-nearest neighbors algorithm using interval-valued fuzzy sets. *Information Sciences* 329(C): 144–163.

[DF08] Duivesteijn W. y Feelders A. (2008) Nearest neighbour classification with monotonicity constraints. En *ECML/PKDD (1)*, páginas 301–316.

[DGH14] Derrac J., García S., y Herrera F. (2014) Fuzzy nearest neighbor algorithms: Taxonomy, experimental analysis and prospects. *Information Sciences* 260: 98–119.

[DKS09] Dembczyński K., Kotłowski W., y Słowiński R. (2009) Learning rule ensembles for ordinal classification with monotonicity constraints. *Fundamenta Informaticae* 94(2): 163–178.

[Dom96] Domingos P. (1996) Unifying instance-based and rule-based induction. *Machine Learning* 24(2): 141–168.

[DS11] Das S. y Suganthan P. N. (2011) Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation* 15(1): 4–31.

[dSLdCNR07] de Sá Lisboa F. O. S., do Carmo Nicoletti M., y Ramer A. (2007) A version of the NGE model suitable for fuzzy domains. *Journal of Intelligent and Fuzzy Systems* 18(1): 1–17.

[DV10] Daniels H. y Velikova M. (2010) Monotone and partially monotone neural networks. *IEEE Transactions on Neural Networks* 21(6): 906–917.

[FÖ99] Fürnkranz J. (1999) Separate-and-conquer rule learning. *Artificial Intelligence Review* 13: 3–54.

[FdRL+14] Fernández A., del Río S., López V., Bawakid A., del Jesús M. J., Benítez J. M., y Herrera F. (2014) Big data with cloud computing: an insight on the computing environment, mapreduce, and

programming frameworks. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery* 4(5): 380–409.

[FLdJH15] Fernández A., López V., del Jesús M. J., y Herrera F. (2015) Revisiting evolutionary fuzzy systems: Taxonomy, applications, new trends and challenges. *Knowledge-Based Systems* 80: 109–121.

[FPSS96] Fayyad U. M., Piatetsky-Shapiro G., y Smyth P. (1996) Advances in knowledge discovery and data mining. chapter From Data Mining to Knowledge Discovery: An Overview, páginas 1–34. American Association for Artificial Intelligence, Menlo Park, CA, USA.

[FRC14] Fernández-Navarro F., Riccardi A., y Carloni S. (2014) Ordinal neural networks without iterative tuning. *IEEE Transactions on Neural Networks and Learning Systems* 25(11): 2075–2085.

[GAA+16] García J., AlBar A. M., Aljohani N. R., Cano J.-R., y García S. (2016) Hyperrectangles selection for monotonic classification by using evolutionary algorithms. *International Journal of Computational Intelligence Systems* 9(1): 184–201.

[Gam98] Gamarnik D. (1998) Efficient learning of monotone concepts via quadratic optimization. En *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, páginas 134–143. ACM, New York, NY, USA.

[GFA+15] García J., Fardoun H. M., Alghazzawi D. M., Cano J.-R., y García S. (2015) Mongel: monotonic nested generalized exemplar learning. *Pattern Analysis and Applications* páginas 1–12.

[GG16] Gutiérrez P. A. y García S. (2016) Current prospects on ordinal and monotonic classification. *Progress in Artificial Intelligence* 5(3): 171–179.

[GHG15] González S., Herrera F., y García S. (2015) Monotonic random forest with an ensemble pruning mechanism based on the degree of monotonicity. *New Generation Computing* 33(4): 367–388.

[GHG16] González S., Herrera F., y García S. (2016) Managing monotonicity in classification by a pruned adaboost. En *International Con-*

ference on Hybrid Artificial Intelligence Systems, páginas 512–523. Springer.

[GLH15] García S., Luengo J., y Herrera F. (2015) *Data Preprocessing in Data Mining*. Springer.

[GPOSM+16] Gutiérrez P. A., Pérez-Ortiz M., Sánchez-Monedero J., Fernandez-Navarro F., y Hervás-Martínez C. (January 2016) Ordinal regression methods: survey and experimental study. *IEEE Transactions on Knowledge and Data Engineering* 28(1): 127–146. JCR(2014): 2.240 Position: 12/139 (Q1) Category: COMPUTER SCIENCE, INFORMATION SYSTEMS.

[HAE13] Hany. A. Elsalamony A. M. E. (August 2013) Bank direct marketing based on neural network. *International Journal of Engineering and Advanced Technology* 2(6).

[HCRdJ16] Herrera F., Charte F., Rivera A. J., y del Jesús M. J. (2016) *Multilabel Classification - Problem Analysis, Metrics and Techniques*. Springer.

[HCZ+12a] Hu Q., Che X., Zhang L., Zhang D., Guo M., y Yu D. (2012) Rank entropy-based decision trees for monotonic classification. *IEEE Transactions on Knowledge Data Engineering* 24(11): 2052–2064.

[HCZ+12b] Hu Q., Che X., Zhang L., Zhang D., Guo M., y Yu D. (2012) Rank entropy-based decision trees for monotonic classification. *IEEE Transactions on Knowledge and Data Engineering* 24(11): 2052–2064.

[HK11] Han J. y Kamber M. (2011) *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc.

[JS11] Japkowicz N. y Shah M. (Eds.) (2011) *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press.

[Kar91] Karpf J. (1991) Inductive modelling in law: Example based expert systems in administrative law. En *Proceedings of the 3rd International Conference on Artificial Intelligence and Law*, ICAIL '91, páginas 297–306. ACM, New York, NY, USA.

[KM99] Kazuhisa MAKINO Takashi SUDA H. O. T. I. (January 1999) Data analysis by positive decision trees. *IEICE TRANSACTIONS on Information and Systems* E82-D(1): 76–88.

[KS09] Kotłowski W. y Słowiński R. (2009) Rule learning with monotonicity constraints. En *ICML*, volumen 382.

[KS13] Kotłowski W. y Słowiński R. (2013) On nonparametric ordinal classification with monotonicity constraints. *IEEE Transactions on Knowledge Data Engineering* 25(11): 2576–2589.

[LBCV08] Lievens S., Baets B. D., y Cao-Van K. (2008) A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting. *Annals of Operaional Research* 163(1): 115–142.

[LMG06] Liu T., Moore A. W., y Gray A. (2006) New algorithms for efficient high-dimensional nonparametric classification. *Journal of Machine Learning Research* 7: 1135–1158.

[LYW03] Lee J. W., Yeung D. S., y Wang X. (2003) Monotonic decision tree for ordinal classification. En *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volumen 3, páginas 2623–2628. IEEE.

[MDP13] Milstein I., David A. B., y Potharst R. (2013) Generating noisy monotone ordinal datasets. *Artificial Intelligence Research* 3(1): p30.

[Men64] Mendelson E. (1964) *Introduction to Mathematical Logic*. Van Nostrand, Princeton, New Jersey, first edition.

[MP15a] Marsala C. y Petturiti D. (2015) Rank discrimination measures for enforcing monotonicity in decision tree induction. *Information Sciences* 291(Complete): 143–171.

[MP15b] Marsala C. y Petturiti D. (2015) Rank discrimination measures for enforcing monotonicity in decision tree induction. *Information Sciences* 291: 143–171.

[NLW09] Novak P. K., Lavrač N., y Webb G. I. (2009) Supervised descriptive rule discovery: A unifying survey of contrast set, emerging

pattern and subgroup mining. *Journal of Machine Learning Research* 10: 377–403.

[OQR07] Orallo J. H., Quintana M. J. R., y Ramírez C. F. (2007) *Introducción a la minería de datos*. Pearson Prentice Hall.

[Qui93] Quinlan J. R. (1993) *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc.

[QXL+15] Qian Y., Xu H., Liang J., Liu B., y Wang J. (2015) Fusing monotonic decision trees. *IEEE Transactions on Knowledge and Data Engineering* 27(10): 2717–2728.

[RDBDM06] Rademark M., De Baets B., y De Meeyer H. (2006) On the role of of maximal independent sets in cleaning data for supervised ranking. En *IEEE International Conference on Fuzzy Systems*, páginas 1619–1624.

[Rom08] Roman S. (2008) *Lattices and Ordered Sets*. Springer New York.

[Roy00] Royston P. (2000) A useful monotonic non-linear model with applications in medicine and epidemiology. *Statistics in Medicine* 19(15): 2053–2066.

[Sal91] Salzberg S. (1991) A nearest hyperrectangle learning method. *Machine Learning* 6(3): 251–276.

[SAS15] Sobhy M., Abbas S., y Salem A.-B. M. (2015) Knowledge discovery for banking risk management. En *Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication*, IPAC '15, páginas 86:1–86:6. ACM, New York, NY, USA.

[SASS11] Soni J., Ansari U., Sharma D., y Soni S. (March 2011) Article: Predictive data mining for medical diagnosis: An overview of heart disease prediction. *International Journal of Computer Applications* 17(8): 43–48.

[SSBD14] Shalev-Shwartz S. y Ben-David S. (2014) *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA.

[SSS14] S. Salcedo-Sanz J. Del-Ser I. L.-T. S. G.-L. J. A. P.-F. (2014) The coral reefs optimization algorithm: A novel metaheuristic for efficiently solving optimization problems. *The Scientific World Journal* página 739768.

[TGH15] Triguero I., García S., y Herrera F. (2015) Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge Information Systems* 42(2): 245–284.

[TPB+15] Triguero I., Peralta D., Bacardit J., García S., y Herrera F. (2015) MRPR: A mapreduce solution for prototype reduction in big data classification. *Neurocomputing* 150: 331–345.

[WD95] Wettschereck D. y Dietterich T. G. (1995) An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine Learning* 19(1): 5–27.

[WF05] Witten I. H. y Frank E. (2005) *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[WM97] Wilson D. R. y Martinez T. R. (1997) Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research* 6(1): 1–34.

[WZZZ15] Wang S., Zhai J., Zhang S., y Zhu H. (2015) An ordinal random forest and its parallel implementation with mapreduce. En *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*, páginas 2170–2173. IEEE.

[ZZZW15] Zhang J., Zhai J., Zhu H., y Wang X. (2015) Induction of monotonic decision trees. En *2015 International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR)*, páginas 203–207. IEEE.