Universidad de Holguín "Oscar Lucero Moya"

Facultad de Informática- Matemática

Sistemas Basados en Reglas Difusos Genéticos para la Minería de Datos

Trabajo de Diploma para optar por el título de Ingeniero en Informática

Autora: Dailin Yera Vázquez

Tutores: MSc Yoel Caises Almaguer

Ing Ricardo Navarro Rodriguez

Holguín, Cuba

Junio 2009

Sí ya sabes lo que tienes que hacer y no lo haces entonces estás peor que antes.

Confucio

Agradecimientos

A mis tutores por su guía y ayuda en todo momento

A mi familia por apoyo incondicional

A mi esposo Adrian y mis queridos suegros por su comprensión y ayuda

A Arian por su importante colaboración en los experimentos

A todos los que contribuyeron de una forma u otra con esta investigación

Dedicatoria

A mi abuela Idolidia, que siempre ha vivido con este sueño.

Resumen

En la actualidad el uso de Algoritmos Genéticos para el diseño de Sistemas Basados en Reglas Difusas con capacidades de aprendizaje y adaptación ha dado lugar a los Sistemas Basados en Reglas Difusos Genéticos, los cuales han cobrado bastante auge y se han aplicado a la Minería de Datos, para descubrir conocimiento útil.

En los últimos años se puede encontrar una gran cantidad de publicaciones con propuestas novedosas de este tipo de sistemas. Sin embargo los estudios experimentales realizados por los autores presentan puntos débiles en cuanto a las comparaciones con sistemas similares y la reproducción de los experimentos, debido a la falta de los datos con que se realizaron y de los valores de los parámetros de los algoritmos. Además el uso de herramientas de análisis estadístico constituyen una imperiosa necesidad en el estudio de los Sistemas Basados en Reglas Difusos Genéticos.

Esta investigación está enfocada a la realización de un estudio experimental con varios Sistemas Basados en Reglas Difusos Genéticos, que se encuentran incluidos en la herramienta para la Minería de Datos, KEEL. Comprende la descripción de varios de estos sistemas, los cuales son comparados de acuerdo a diversos criterios usando dos tests estadísticos no paramétricos. Es importante destacar que las bases de datos usadas para probar los algoritmos también se encuentran explícitas y son fácilmente accesibles. Además recoge los fundamentos teóricos de Descubrimiento de Conocimiento en Bases de Datos, Minería de Datos, Aprendizaje Automático, Sistemas Basados en Reglas Difusas, Algoritmos Genéticos y Sistemas Basados en Reglas Difusos Genéticos.

Abstract

Nowadays Genetic Algorithms have been widely used to create Fuzzy Rules Based Systems with added learning capabilities. These systems are known as Genetic Fuzzy Rules Based Systems and they have been successfully applied in Data Mining to discover useful information.

In the last years a lot of publications that present a novel proposal of this kind of systems can be found. However the experimental studies made by the authors have some critical points as regards: the comparison with similar system and the reproduction of the experiments, due to the lack of the data used and the parameters values of the algorithms. Besides the use of statistical analysis tools is a peremptory necessity in the analysis of the Genetic Fuzzy Rules Based Systems.

This investigation is focused to carry out an experimental study with some Genetic Fuzzy Rules Based Systems included in KEEL, which is a Data Mining tool. The description of every one these systems is contained and they are compared according to several criteria using two no-parametric statistical test. It is important to mention that the data bases used to test the algorithms are explicit as well and they are easily accessible. Furthermore this work gathers the theoretical basis of Knowledge Discovery in Databases, Data Mining, Machine Learning, Fuzzy Rules Based Systems, Genetic Algorithms and Genetic Fuzzy Rules Based Systems.

Índice de contenidos

Índice de tablas

Índice de figuras

Introducción

El desarrollo de los ordenadores y de las tecnologías de la información ha facilitado la captura y almacenamiento de datos a un costo muy bajo. En ocasiones se dispone de tanta información que resulta difícil sacarle provecho, debido a que los datos tal cual se almacenan es improbable que proporcionen beneficio de forma directa. Sin embargo existe una gran cantidad de información "oculta" de gran importancia, cuyo descubrimiento resulta un poco complejo con las herramientas tradicionales de gestión de datos. Como consecuencia es necesario aplicar metodologías para el análisis inteligente de los datos y así descubrir dicha información. El proceso que brinda estos beneficios es el Descubrimiento de Conocimiento en Bases de Datos (KDD por sus siglas en Inglés), el cual mediante el procesamiento de grades volúmenes de información pretende encontrar patrones útiles en los datos, contando para ello con varias etapas entre las que se encuentra la Minería de Datos, que es además una de las más importantes, pues se encarga precisamente de encontrar el conocimiento valioso auxiliándose diferentes metodologías y herramientas que permiten automatizar el proceso, las cuales han sido definidas en áreas de investigación como Estadísticas, Inteligencia Artificial y el Aprendizaje Automático. Esta última es la que provee las bases técnicas de la Minería de Datos [45].

El Aprendizaje Automático tiene como objetivo desarrollar métodos computacionales capaces de inducir conocimiento a partir de datos, haciendo uso de metodologías como los Algoritmos Genéticos o las Redes Neuronales. Aunque los Algoritmos Genéticos no fueron expresamente diseñados para el aprendizaje, debido a su capacidad de búsqueda en espacios complejos y mal definidos han sido aplicados satisfactoriamente en una gran variedad de tareas de aprendizaje y descubrimiento de conocimiento, por ejemplo, para aprender un conjunto de reglas o un Sistema Basado en Reglas. En gran parte de los casos las reglas son difusas, por lo que los sistemas con dichas características reciben el nombre de Sistemas Basados en Reglas Difusas (SBRD) y tienen gran aceptación debido que permiten manipular incertidumbres, ambigüedades y contradicciones.

Los sistemas difusos que aprenden usando Algoritmos Genéticos o en general cualquier Algoritmo Evolutivo se conocen como Sistemas Difusos Genéticos. Existen diferentes enfoques, como los Sistemas Neuro-difusos Genéticos o los algoritmos de agrupamiento difuso genético; pero los más populares son los Sistemas Basados en Reglas Difusos Genéticos, (GFRBS, por sus siglas en Inglés) en el que los componentes de un SBRD se definen mediante un Algoritmo Genético; básicamente este tipo de sistemas emplean un proceso de aprendizaje evolutivo para el diseño de la Base de Conocimientos de un SBRD [11]. Han sido aplicados en importantes áreas como el control, la transportación, la modelación, la toma de decisiones y otras.

Según un profundo estudio realizado por Francisco Herrera (2008) [22] en los últimos años se puede encontrar una gran cantidad de publicaciones sobre los Sistemas Difusos Genéticos. Sin embargo un gran número de estas presenta problemas. Entre los puntos críticos más sobresalientes se encuentran que algunos autores no comparan su propuesta con los Sistemas Difusos Genéticos clásicos que se pueden encontrar en la literatura. Aunque los investigadores plantean que el Algoritmo Evolutivo tiene buenos resultados, no ofrecen ninguna justificación para su uso. Los estudios experimentales también reflejan puntos débiles, en ocasiones es imposible reproducir el mismo estudio experimental debido a que no se puede obtener el conjunto de datos con que se realizó o por la falta de los valores de los parámetros usados por el autor en su experimento. Plantea además que cuando se hacen una nueva propuesta se deben justificar su uso, indicando cual es el objetivo para poder encontrar una medida de evaluación (precisión, complejidad…) y realizar un estudio experimental comparando cuál es el mejor enfoque de acuerdo al mismo objetivo y a los mismos componentes del sistema difuso en consideración. En este estudio también se planta que es una imperiosa necesidad el uso de herramientas de análisis estadístico en el estudio de los modelos de Sistemas Difusos Genéticos.

Tomando en cuenta las consideraciones del estudio referenciado anteriormente se delimitó el siguiente **problema científico**: En las propuestas recientes de

Sistemas Basados en Reglas Difusos Genéticos no son suficientes los estudios experimentales realizados ni las comparaciones con sistemas similares.

Tal problema se enmarca en el **objeto de investigación**: Sistemas Difusos Genéticos. Para darle solución al mismo se propuso como **objetivo de investigación**: Realizar un estudio experimental comparativo entre varios Sistemas Basados en Reglas Difusos Genéticos para la clasificación en Minería de Datos.

Este objetivo delimita el **campo de acción**: Sistemas Basados en Reglas Difusos Genéticos para la clasificación en Minería de Datos.

Para guiar el presente trabajo se formularon las siguientes **preguntas científicas**:

1. ¿Cuáles son los presupuestos teóricos de los Sistemas Basados en Reglas Difusos Genéticos para la clasificación en la Minería de Datos?

2. ¿Qué medidas de comparación son necesarias para evaluar el comportamiento de varios Sistemas Basados en Reglas Difusos Genéticos para la clasificación en Minería de Datos?

3. ¿Qué resultados se obtienen del estudio experimental comparativo realizado a los Sistemas Basados en Reglas Difusos Genéticos para la clasificación en Minería de Datos?

4. ¿Qué tan sostenible es este estudio?

Se trazaron las siguientes **tareas de investigación** para dar respuesta a las preguntas anteriores:

1. Elaborar los presupuestos teóricos de los Sistemas Basados en Reglas Difusos Genéticos para la clasificación en Minería de Datos.

2. Describir los Sistemas Basados en Reglas Difusos Genéticos usados para realizar el estudio experimental comparativo.

3. Realizar el estudio experimental comparativo de los Sistemas Basados en Reglas Difusos Genéticos escogidos.

4. Analizar los resultados de los experimentos.

5. Valorar la sostenibilidad del estudio.

Para dar cumplimiento a las **tareas de investigación** se usó la siguiente **metodología investigativa** basada en Métodos Teóricos, Empíricos y Estadísticos-Matemáticos:

Métodos Teóricos

- Análisis y Síntesis: apoyó en la descripción del campo de investigación y los fundamentos que lo conforman a partir del estudio de bibliografía referente al mismo.

- Inducción y deducción: se utilizó en varias etapas de la investigación para arribar a conclusiones generales.

- Histórico-Lógico: fue empleado para el mejor entendimiento de los Sistemas Basados en Reglas Difusos Genéticos a partir de la evolución de los mismos.

Métodos Empíricos

- Análisis bibliográfico: fue de gran importancia en la comprensión de los Sistemas Basados en Reglas Difusos Genéticos para la clasificación en Minería de Datos, su importancia y estado actual.

- Experimentación: contribuyó a evaluar el desempeño de varios Sistemas Basados en Reglas Difusos Genéticos y a elaborar los resultados generales del estudio experimental realizado.

Métodos Estadísticos – Matemáticos

- Tests Estadísticos no-paramétricos: Friedman y Bonferroni-dunn para apoyar la comparación los Sistemas Basados en Reglas Difusos Genéticos.

El documento está estructurado en Introducción, tres Capítulos, Conclusiones, Recomendaciones, Bibliografía y Anexos. El Capítulo I aborda los elementos teóricos relacionados con los Sistemas Basados en Reglas Difusos Genéticos para la tarea de clasificación en la Minería de Datos. En el Capítulo II se introduce y describe la herramienta KEEL usada para llevar a cabo experimentos con distintos Sistemas Basados en Reglas Difusos Genéticos, incluidos en la misma. Además cada uno de ellos se describe de forma clara y concisa. Por último el Capítulo III muestra las principales características de las bases de datos usadas en la experimentación. Además se hace un análisis comparativo de los resultados obtenidos por los sistemas en cuestión, usando para ello específicamente dos test estadísticos no-paramétricos, que permiten determinar si existen diferencias significativas entre los Sistemas Basados en Reglas Difusos Genéticos probados.

Capítulo I. Fundamentos teóricos

Este Capítulo aborda elementos teóricos sobre el Descubrimiento de Conocimiento en Bases de Datos, la Minería de Datos, el Aprendizaje Automático, así como las Sistemas Basados en Reglas Difusas. Se abordan ampliamente los Algoritmos Evolutivos haciendo especial énfasis en los Algoritmos Genéticos. Finalmente se describen detalladamente los Sistemas Difusos Genéticos, en particular los Sistemas Basados en Reglas Difusos Genéticos.

1.1 Descubrimiento de Conocimiento en Bases de Datos

El KDD, según fue definido por Fayyad (1996) [15] “es el proceso no trivial de identificación de patrones válidos, novedosos, potencialmente útiles y en última instancia comprensibles de los datos”

El objetivo fundamental del KDD es encontrar conocimiento útil, válido, relevante y nuevo sobre un fenómeno o actividad mediante algoritmos eficientes, dadas las crecientes órdenes de magnitud en los datos. Los campos de investigación envueltos en este proceso son muy variados: bases de datos y reconocimiento de patrones, Estadística e Inteligencia Artificial, visualización de datos y supercomputación. Los investigadores de KDD incorporan técnicas, algoritmos y métodos de estos campos.

El proceso de Descubrimiento de Conocimiento en Bases de Datos involucra varios pasos [46]:

1. Desarrollo y comprensión del dominio de aplicación e identificación del objetivo del proceso de KDD desde el punto de vista del usuario. [15]

2. Diseño del esquema de un almacén de datos (Data Warehouse) que unifique toda la información recopilada sobre la que se realizará el descubrimiento.

3. Implantación de un almacén de datos que permita la navegación y visualización previa de sus datos, para poder especificar los aspectos a

ser estudiados. Esta es la etapa que puede llegar a consumir el mayor tiempo.

4. Preprocesamiento, el cual incluye la selección, limpieza y transformación de los datos que se van a analizar. La selección se puede hacer tanto de ejemplos (horizontal) como de atributos (vertical). La limpieza y transformación de los datos se logra diseñando una estrategia adecuada para manejar ruido, valores incompletos, secuencias de tiempo, casos extremos, etc.

5. Selección y aplicación de un método de Minería de Datos apropiado. Este paso consiste en escoger la tarea de descubrimiento a realizar, por ejemplo, clasificación, agrupamiento o clustering, regresión, etc. También la selección de él o de los algoritmos a utilizar y la transformación de los datos al formato requerido por el algoritmo escogido. Luego se lleva a cabo el proceso de Minería de Datos donde se buscan patrones que puedan expresarse como un modelo o que evidencien las dependencias de los datos. El modelo encontrado depende de su función (clasificación) y de su forma de representarlo (árboles de decisión, reglas, etc). Se debe especificar un criterio de preferencia para seleccionar un modelo dentro de un conjunto posible de ellos y la estrategia de búsqueda a utilizar, la que por lo general está predeterminada en el algoritmo de minería.

6. Evaluación, interpretación, transformación y representación de los patrones extraídos. En ocasiones se debe regresar a los pasos anteriores y por tanto repetir el proceso, quizás con otros datos, algoritmos, metas y estrategias. Este es un paso crucial en donde se requiere tener conocimiento del dominio. La interpretación puede beneficiarse de procesos de visualización, y sirve también para borrar patrones redundantes e irrelevantes.

7. Difusión y uso del nuevo conocimiento o lo que es igual, la incorporación del conocimiento descubierto al sistema, normalmente para mejorarlo, lo

cual puede incluir resolver conflictos potenciales con el conocimiento existente.

Es importante destacar que la Minería de Datos es un paso dentro del proceso de KDD, como fue descrito anteriormente. No obstante a menudo ambos términos son usados indistintamente, sobre todo en el ámbito comercial lo cual no es correcto.

En lo sucesivo esta investigación se centrará en la Minería de Datos, debido a que constituye una de las áreas en que más se han desarrollado los algoritmos que construyen el objeto de este estudio.

1.1.1 Minería de Datos

Históricamente al proceso de encontrar patrones útiles a partir de datos ha recibido gran variedad de nombres; Minería de Datos, extracción de conocimiento, descubrimiento de información, arqueología de datos y otros. La Minería de Datos ha sido usada mayormente por estadistas, analistas de datos y en las comunidades de sistemas de administración de información. Una de sus definiciones más aceptadas es la siguiente: “Paso consistente en el uso de algoritmos concretos que generan una enumeración de patrones a partir de los datos preprocesados“ [15]. Otra definición de Minería de Datos dada en [30] es la siguiente: “Minería de Datos es el proceso de descubrimiento de varios modelos, resúmenes y valores derivados de una colección de datos”.

Existen dos importantes razones por las que se debe usar Minería de Datos [47]:

1. La tarea de encontrar patrones realmente útiles puede ser desalentadora en ocasiones, debido al hecho de que estos no son aparentes.

2. En la mayoría de las aplicaciones la cantidad de datos es muy grande para ser analizada solamente de forma manual.

La Minería de Datos se auxilia de diferentes tareas, que se pueden clasificar en dos tipos: predictivas y descriptivas. Las primeras emplean el proceso de

inferencia en los datos con el objetivo de hacer predicciones. Las descriptivas, por otra parte, caracterizan las propiedades generales de los datos [19]. Ambas son llevadas a cabo mediante las tareas elementales de la Minería de Datos, entre las que se encuentran la regresión, agrupamiento (*clustering*), resúmenes (*sumarization*), modelos de dependencia, detección de desviaciones y cambios y la clasificación, la que resulta de especial interés para la presente investigación.

La clasificación es la acción de asignar una categoría a un objeto de acuerdo con sus características. En la Minería de Datos, se refiere a la tarea de analizar un conjunto de objetos preclasificados, llamado conjunto de entrenamiento, para aprender un modelo que luego es usado para clasificar objetos desconocidos en una de varias clases predefinidas. Un objeto, también denominado ejemplo, es descrito por un conjunto de atributos o variables, uno de los cuales describe la clase a la que el ejemplo pertenece y es por tanto llamado atributo clase o variable clase. Los demás son llamados atributos independientes o predictores [48]. Por lo general la forma en que se representan los datos en el conjunto de entrenamiento es la siguiente:

Día	Temperatura	Humedad	Viento	Jugar
Soleado	Caliente	Alta	No	No
Nublado	Caliente	Alta	No	Si
Lluvioso	Fría	Normal	Si	Si
Soleado	Media	Normal	Si	Si

Tabla1.1. Estructura del conjunto de entrenamiento

Cada una de las filas de la tabla representa un ejemplo, los que están caracterizados por un conjunto predeterminado de atributos, que en este caso son Día, Temperatura, Humedad y Viento. La última columna representa la clase: en dependencia de las características del tiempo se puede jugar o no.

En el área del Aprendizaje Automático la atención se ha enfocado más en generar expresiones de clasificación que sean fácilmente entendibles por los humanos [48]. Entre las más populares se encuentran, aprendizaje de árboles

de decisión, aprendizaje de reglas de clasificación, los métodos Bayesianos y otras.

1.2 Aprendizaje Automático

Un concepto primordial y diferenciador de las técnicas estadísticas más clásicas es el de Aprendizaje Automático (*Machine Learning*), que fue concebido hace aproximadamente cuatro décadas con el objetivo de desarrollar métodos computacionales que implementarían varias formas de aprendizaje, en particular, mecanismos capaces de inducir conocimiento a partir de datos.

Uno de los conceptos que mejor sintetiza los aspectos que deben ser considerados en el proceso de aprendizaje, es el dado por Herbert Simon (1983): “cualquier cambio en un sistema que le permite desempeñarse mejor la próxima vez, sobre la misma tarea u otra tomada de la misma población”. Esta definición cubre varias actividades que van desde mejorar el rendimiento de un sistema existente hasta la adquisición de nuevos conceptos [1]. Otra definición de aprendizaje lo suficientemente amplia como para cubrir cualquier programa que mejora su desempeño al ejecutar alguna tarea, a través de su experiencia, es la dada por Mitchell (1997) [34]: “Se dice que un programa de cómputo aprende a partir de su experiencia E, con respecto a alguna clase de tareas T y una medida de desempeño P, si el desempeño del programa al realizar las tareas T, mejora con la experiencia E, de acuerdo a la medida P.”

Existen varias técnicas de Aprendizaje Automático, entre las que se encuentra el aprendizaje inductivo, que es la más extensa y estudiada, cuyo objetivo es descubrir descripciones generales que permitan capturar las características comunes de un grupo de ejemplos. Permite obtener conclusiones generales a partir de información específica, las cuales evidentemente son conocimiento nuevo. Algunas técnicas de Aprendizaje Automático usan Algoritmos Genéticos, las cuales son también inductivas [5]. Existen dos tipos de aprendizaje inductivo: aprendizaje supervisado y aprendizaje no supervisado.

En el aprendizaje supervisado la experiencia está constituida por un conjunto de ejemplos cuya clase es conocida, es decir que la entrada es un conjunto de ejemplos clasificados. El aprendizaje se realiza al distinguir los ejemplos que pertenecen a una clase de los del resto, siendo el resultado del proceso una representación de las clases que describen los ejemplos. Como se puede apreciar la tarea de clasificación de la Minería de Datos es una forma aprendizaje supervisado. Por otra parte el aprendizaje no supervisado es más complejo, no existe una clasificación de los ejemplos y se debe encontrar la mejor manera de estructurarlos, obteniendo por lo general una partición en grupos, en los cuales los ejemplos similares están juntos y separados de otros menos parecidos. El resultado de este proceso es una partición de los ejemplos y una descripción de los grupos de la misma. [5]

El Aprendizaje Automático es un área bastante amplia, que de forma general se dedica al estudio de algoritmos computacionales que mejoran automáticamente a través de la experiencia [48]. En ella se han desarrollado satisfactoriamente muchas aplicaciones en los últimos años, al mismo tiempo que han habido importantes avances en la teoría y los algoritmos que forman sus fundamentos. Entre las disciplinas que han contribuido a su desarrollo se encuentran, la Estadística, Redes Neuronales, Algoritmos Evolutivos y otras [34]. Dentro de la Inteligencia Artificial, es la que se encarga de desarrollar métodos capaces de aumentar las capacidades de diversas aplicaciones, como es el caso de los Sistemas Basados en el Conocimiento, de manera que puedan ser más flexibles y eficaces [5].

1.3 Sistemas Basados en el Conocimiento

Los Sistemas Basados en el Conocimiento (SBC) están formados por tres componentes: la Base de Conocimiento (BC), el Mecanismo de Inferencia (MI) y una interfaz. La MI es el módulo encargado de encontrar la solución a un problema dado a partir de los datos iniciales del problema y del conocimiento almacenado en la BC para resolver los problemas del dominio de aplicación para el cual se desarrolla el sistema. [38]

Dentro de las definiciones dadas sobre los SBC se encuentra la de Feigenbaum (1977): “Sistemas cuya capacidad para resolver problemas no reside en la expresión formal ni en los esquemas lógicos de inferencia que emplean sino en el conocimiento que poseen”. Según Jackson (1986) son: “Sistemas que resuelven problemas utilizando una representación simbólica del conocimiento humano”.

Las técnicas de Aprendizaje Automático pueden ser de gran valor para automatizar algunos aspectos de la adquisición de conocimientos. Hay al menos tres formas en que facilitan la adquisición de conocimiento: ayudando a la construcción inicial de la BC, refinándola y adaptándola a ciertos requerimientos. [38]

El conocimiento que se almacena en la BC puede ser representado de diversas formas: reglas, probabilidades o frecuencias, casos, etc. Estos diferentes tipos de conocimiento dan lugar a varios SBC, entre ellos se encuentran, los Sistemas Basados en Probabilidades, Sistemas Basados en Casos y Sistemas Basados en Reglas, los cuales son unos de los más ampliamente usados y en los que se centrará este estudio.

1.3.1 Sistemas Basados en Reglas

Los Sistemas Basados en Reglas usan reglas IF – THEN para representar el conocimiento. La parte IF de la reglas se denomina antecedente, y la parte THEN, consecuente, IF los datos satisfacen la condición del antecedente THEN se ejecutan las instrucciones del consecuente.

El antecedente es una proposición lógica cuya veracidad debe ser determinada; puede ser complejo, formado por varias proposiciones simples cuyos valores de verdad son combinados por conectores AND, OR y NOT. Existe una gran variedad de consecuentes disponibles en los Sistemas Basados en Reglas tales como instrucciones de entrada y salida de datos y conclusiones. [44]

La BC contiene las variables y el conjunto de reglas que definen el problema, y la MI obtiene las conclusiones aplicando la lógica clásica a estas reglas [18]. El

Método de Solución de los Problemas (MSP) en los Sistemas Basados en Reglas consiste en la construcción de una cadena de inferencias que a su vez crea un camino entre la definición del problema y su solución. Dicha cadena de inferencia puede ser construida por dos vías: comenzando con todos los datos conocidos y determinar una solución (*forward chaining*), que es la más usada, o seleccionar una conclusión posible y tratar de probar su validez buscando evidencias que la soporten (*backward chaining*). Este último sufre de inferencias redundantes y ciclos infinitos [40]. En la siguiente imagen se muestra la estructura de este tipo de SBC:

Figura1.1. Estructura de los Sistemas Basados en Reglas

1.4 Sistemas Basados en Reglas Difusas

Desafortunadamente constantemente en la vida se enfrentan incertidumbres, ambigüedades y contradicciones. Por esta razón algunos sistemas usan la misma estructura de los Sistemas Basados en Reglas, pero introducen la teoría de conjuntos difusos, desarrollada por Zadeh (1965), para tratar la incertidumbre de las reglas y la de sus premisas. Los Sistemas Basados en Reglas que utilizan la teoría de conjuntos difusos como herramienta para definir las reglas son los llamados Sistemas Basados en Reglas Difusas (SBRD).

Un SBRD está constituido por cuatro componentes, la Interfaz de Fuzzificación, la Base de Conocimientos, el Mecanismo de Inferencia y la Interfaz de Defuzzificación. Los roles de las interfaces de Fuzzificación y Defuzzificación son recíprocos, mientras la primera convierte un valor real en difuso la segunda hace el proceso inverso. La Base de Conocimientos contiene todo el conocimiento que el sistema necesita e incluye las reglas difusas. Finalmente, el Mecanismo de Inferencia se encarga de determinar las salidas difusas

correspondientes a un conjunto de entradas dadas [39]. En la figura siguiente se muestra la estructura de los SBRD:

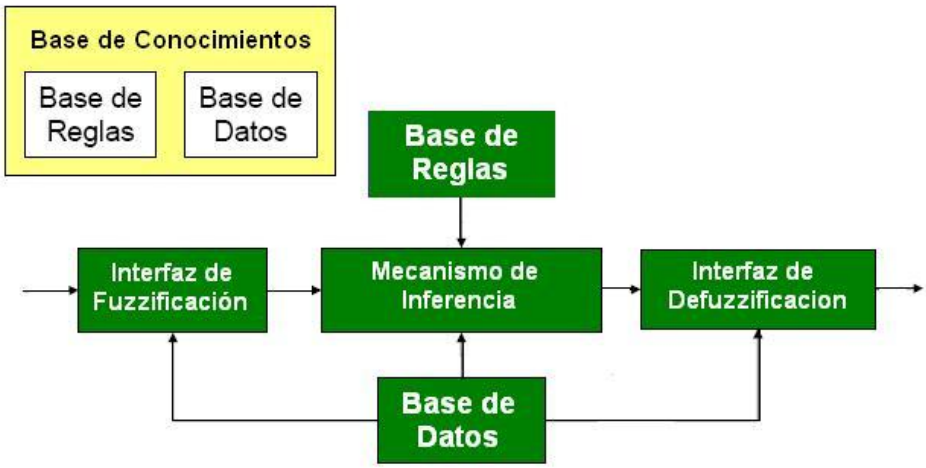

Figura1.2. Esquema de los Sistemas Basados en Reglas Difusas

La Base de Conocimientos está formada por la Base de Datos, la cual contiene todos los datos relativos al sistema, como variables o valores posibles y una Base de Reglas, representando el conocimiento sobre el problema a ser resuelto, constituida por una colección de reglas difusas. La Interfaz de Fuzzificación, transforma datos reales en conjuntos difusos, los cuales son usados por un Mecanismo de Inferencia junto con la Base de Conocimientos para llevar a cabo el proceso de inferencia que se necesita para obtener una salida difusa del SBRD, la cual será convertida por la Interfaz de Defuzzificación en el valor real que mejor la represente.

El Mecanismo de Inferencia actúa de diferente forma en dependencia del tipo de problema a resolver (clasificación o regresión) y el tipo de reglas difusas [22].

De acuerdo al tipo de regla difusa que usan los SBRD se pueden clasificar en; SBRD tipo Mamdani (1975) y tipo TSK (1985, 1988). Mientras las reglas difusas de tipo Mamdani consideran una variable lingüística en el consecuente las reglas difusas TSK se basan en representar el consecuente como una función polinomial del las entradas. La expresión genérica de las reglas TSK es la siguiente:

R_i : **IF** X_1 *es* A_1 **AND … AND** X_n *es* A_n **THEN** $Y = P_1 * X_1 + ... + P_n * X_n + P_0$

Estas reglas son bastantes útiles, sobre todo para la aproximación de funciones. [31]

En el otro tipo de sistemas, la Base de Reglas está compuesta por la colección de reglas difusas con la estructura siguiente:

R_i **: IF** X_1 *es* A_{i1} **AND… AND** X_n *es* A_{in} **THEN** Y *is* B_i

Donde $X_1,...,X_n$ y Y son las variables de entrada y la de salida, respectivamente. Dependiendo de las características de estas reglas se pueden distinguir dos tipos de SBRD tipo Mamdani [14]:

- **SBRD tipo Mamdani descriptivos.** Cuando $X_1,...,X_n$ y Y son variables lingüísticas que tienen asociado un conjunto de términos de posibles valores. De esta forma cada A_{ij} y B_i corresponde a un término lingüístico que tiene asociado un conjunto difuso definiendo su significado, este proceso es uniforme para todas las reglas de la Base de Reglas Difusas. Este tipo de sistemas han sido ampliamente usados con buenos resultados en muchas aplicaciones. No obstante, tiene algunas limitaciones debido a la inflexibilidad del concepto de variables lingüísticas. Su desempeño decrece cuando se enfrenta a problemas complejos en los que pequeños cambios en las entradas producen grandes cambios en la salida.

- **SBRD tipo Mamdani aproximados.** Este enfoque ha sido propuesto en los últimos años con el objetivo de evitar las debilidades de los sistemas antes mencionados. Están basados en el trabajo directo con variables difusas en las reglas, las cuales representan su propia semántica. Las variables X_j y Y respectivamente toman un conjunto difuso diferente A_{ij} y B_i como valor en lugar de términos lingüísticos de un conjunto global de términos. De esta forma se puede decir que las reglas

presentan una “semántica libre”. La ventaja de esta representación es el poder expresivo para aprender reglas difusas que presentan sus propias especificidades en términos de los conjuntos difusos involucrados en ellas. Su desventaja con respecto a los sistemas descriptivos es la pérdida de legibilidad de la Base de Reglas Difusas.

Aparte de la forma simple de las reglas difusas tipo Mamdani descritas anteriormente, existen reglas que poseen además de la clase un grado de certeza en el consecuente, que se conocen como reglas difusas DNF (D*isjunctive Normal Form*) y han sido usadas ampliamente en la práctica [31]. Su expresión genérica se muestra a continuación:

R_i **: IF** X_1 *es* A_{i1} **AND … AND** X_n *es* A_{in} **THEN** Y *is* B_i , **grado de certeza** C

También hay otra forma considerada una extensión de los dos tipos de reglas anteriores cuyo consecuente está formado por grados de certezas para todas las clases, y se denominará en lo adelante DNF extendida [13]. Su estructura es la siguiente:

R_j : **IF** x *es* A^j **THEN grados de certeza** $(s_1{}^j \ldots s_p{}^j)$

Se ha demostrado que el uso de reglas con grado de certeza en el consecuente puede mejorar la precisión en la clasificación. [23, 26]

Una de las ventajas de los SBRD es que cada regla difusa es interpretada a través de términos lingüísticos como “pequeño” y “largo. Además son sistemas que tienen una alta precisión en la clasificación [24]. El secreto de su éxito es que son fáciles de implementar, mantener, entender, son robustos y baratos [31].

Los SBRD se han aplicado en la obtención de modelos que representan realidades complejas y en sistemas de control. También en la clasificación para la detección de patrones, el diagnóstico médico y otras. Se han usado además en la Minería de Datos y descubrimiento de información, para la extracción automática de conocimiento intrínseco contenido en grandes bases de datos,

mediante diferentes métodos de aprendizaje como las Redes Neuronales y los Algoritmos Genéticos, los cuales han sido ampliamente aplicados a este tipo de tareas. Es importante destacar que desde principios de la última década del siglo pasado los Algoritmos Genéticos han sido objeto de gran atención, por parte de investigadores, para su uso en el diseño automático de diferentes componentes de un SBRD [8].

1.5 Algoritmos Evolutivos

En las décadas de 1950 y 1960 se comenzó a estudiar los sistemas evolutivos, con la idea de que la evolución pudiera ser usada como herramienta de optimización para problemas de ingeniería. El objetivo de los mismos fue evolucionar una población de soluciones candidatas de un problema dado, usando operadores inspirados por la variación genética y la selección naturales. Este campo ha sido un área de investigación muy activa, que se ha desarrollado en varias direcciones.

¿Por qué usar la evolución para resolver problemas computacionales? Entre las principales razones se puede mencionar que, en muchas ocasiones se requieren buscar a través de un gran número de posibles soluciones, además el programa debe ser adaptativo para continuar funcionando correctamente en ambientes inestables. Otros problemas necesitan que los programas computacionales sean innovadores. Finalmente algunos demandan soluciones complejas que son difíciles de programar y la evolución biológica es un recurso de inspiración para resolverlos. [33]

Evolución es, en efecto, un método de búsqueda entre una gran cantidad de posibles soluciones, las cuales en biología son el conjunto de secuencias genéticas y las soluciones deseadas son organismos altamente adaptables (organismos que son capaces de sobrevivir y reproducirse en su medio ambiente). Claro que la adaptación de los organismos biológicos depende de varios factores, por ejemplo que tan bien puede resistir las características de su medio ambiente y si pueden competir o cooperar con los otros organismo a su alrededor. El criterio de adaptación cambia continuamente al tiempo que las criaturas evolucionan, por lo que evolución es una búsqueda constante de

conjuntos de posibilidades. Buscar soluciones frente a condiciones cambiantes es precisamente lo que se requiere para programas computacionales adaptativos. [33]

El término Algoritmo Evolutivo se utiliza para describir a los algoritmos que resuelven problemas utilizando algunos de los mecanismos de evolución conocidos [37]. La mayoría de las implementaciones actuales de Algoritmos Evolutivos provienen de cuatro paradigmas básicos, estrechamente relacionados, pero desarrollados de forma independiente: Algoritmos Genéticos, Programación Genética, Estrategias Evolutivas y Programación Evolutiva.

Los Algoritmos Genéticos fueron creados por John Holland en la Universidad de Michigan en la década de 1960. En 1970 había un gran interés por entender mejor el comportamiento de estos algoritmos y se realizaron importantes trabajos por parte de Frantz en 1972 y De Jong en 1975. En los años 80 gran cantidad de investigadores de diversas Universidades continuaron trabajando en esta área, en 1985 se desarrolló la primera Conferencia Internacional sobre Algoritmos Genéticos en Pittsburgh, Pennsylvania, también en esta década se publicaron varios libros. En 1989 se formó la Sociedad Internacional para Algoritmos Genéticos, con el objetivo de coordinar y facilitar las actividades relacionadas con los Algoritmos Genéticos. Para el año 1991 se habían definido Algoritmos Genéticos no estándar para evolucionar conjuntos de reglas, código LISP y Redes Neuronales. El periodo de 1990 hasta la actualidad se ha caracterizado por un gran crecimiento y diversidad de la comunidad de Algoritmos Genéticos. [3]

Un Algoritmo Genético puede ser visto como un método general de optimización, que busca un espacio grande de individuos candidatos para seleccionar el mejor de acuerdo a la función de adaptación [34]. No obstante no hay garantía de encontrar el mejor individuo, pero a menudo tiene éxito en la búsqueda de uno bastante bueno, esta característica constituye una de sus limitaciones.

Independientemente que existen múltiples implementaciones de Algoritmos Genéticos, todos comparten la estructura genérica siguiente: el algoritmo opera actualizando iterativamente un conjunto de hipótesis, llamada “población” (la primera población se genera aleatoriamente). En cada iteración, todos los miembros de la población son evaluados de acuerdo a la función de adaptación; algunos de ellos son seleccionados para reproducirse. Luego se aplican los operadores genéticos, usualmente cruzamiento y mutación, a los individuos seleccionados para producir los descendientes. Después ambas poblaciones, la de padres e hijos, deben ser unidas para crear una nueva generación. Gran parte de los Algoritmos Genéticos mantienen el tamaño de la población en un número fijo *N*, esto significa que solo *N* individuos de la población formada por padres e hijos pueden ser seleccionados para crear la nueva. Una posibilidad puede ser usar todos los hijos generados; suponiendo que sean menor que N, y seleccionar aleatoriamente individuos de la población vieja para completar la nueva. Por otro lado si la cantidad de nuevos individuos es igual a N entonces la población de padres es completamente reemplazada por la de sus descendientes. [3]

Según la estructura genérica de los Algoritmos Genéticos, estos cuentan con los siguientes componentes básicos [34]:

Forma de representar los individuos: típicamente cada organismo en la población consiste de un cromosoma, el cual es asimismo un vector de la forma: $<x_1, x_2, \ldots, x_n>$, donde a cada x_i se le llama gen y al valor que toman se le denomina alelo [9]. Los individuos son usualmente representados por cadenas de bits, por lo que pueden ser fácilmente manipulados por los operadores genéticos; por ejemplo un conjunto de reglas IF-THEN puede ser fácilmente representado de esta forma. Existen otras formas de representar a los individuos, como es la representación real, la entera y otras.

La **función de adaptación** (*fitness function*) define un criterio de evaluación de los individuos para la selección probabilística de los que serán incluidos en la nueva generación. Si la tarea es aprender reglas de clasificación, entonces la función de adaptación típicamente tiene un componente que evalúa la

veracidad de la clasificación de la regla sobre un conjunto de ejemplos de entrenamiento dado.

La forma más simple de algoritmos genéticos incluye solo tres operadores: selección, cruzamiento y mutación.

La **selección** aporta mayor potencia y robustez a la técnica de búsqueda, este operador cumple la función de seleccionar los mejores individuos de manera que estos sean considerados en el proceso de generación de la nueva población [37]. Existen varios mecanismos de selección y diversas implementaciones de estos, entre los más conocidos se encuentran el método de la ruleta (*roulette wheel selection*), selección de torneo (*tournament selection*), selección de rango (*rank selection*) y otras.

El **cruzamiento** produce dos nuevos descendientes a partir de dos individuos, copiando bits seleccionados de cada padre. El bit en la posición *i* en cada descendiente es copiado del bit *i* en uno de los dos padres. La elección de cual padre aporta el bit *i* es determinado por una cadena de bits adicional llamada mascara de cruzamiento. Existen varios métodos de cruzamiento: cruzamiento uniforme (*uniform crossover*), cruzamiento sobre un punto (*single-point crossover*), cruzamiento sobre dos puntos (*two-point crossover*) y otras.

El operador de **mutación** produce un pequeño cambio aleatorio a la cadena de bits mediante la selección aleatoria de un bit y luego reemplazando su valor. La mutación se lleva a cabo generalmente después del cruzamiento.

Tanto para el cruzamiento como para la mutación se fijan probabilidades de ocurrencia. En el caso del cruzamiento la probabilidad debe ser alta para lograr una elevada producción de nuevos individuos. Por el contrario, para la mutación la probabilidad debe ser baja, para evitar de esta forma que se destruyan las ventajas del cruzamiento.

En la Programación Genética creada por Koza en 1992 los individuos en la población son programas computacionales. Los programas manipulados por la programación genética son típicamente representados como árboles. Cada

llamada a una función es representada por un nodo en el árbol y los argumentos de la misma son representados por sus nodos descendientes. [34]

De forma corta, la Programación Genética engendra programas computacionales. Para crear la población inicial un gran número de programas computacionales son generados de forma aleatoria. Cada uno de ellos es ejecutado y el resultado es usado para asignar el valor de adaptación a cada programa, dichos valores son usados como medida de selección de algunos programas que son copiados directamente para formar la nueva generación, la cual es completada con nuevos programas hijos concebidos mediante la aplicación de operadores genéticos a programas padres, que son seleccionados basándose también en sus valores de adaptación. Luego esta nueva población es evaluada y el resultado es usado para asignar el valor de adaptación a cada individuo. Eventualmente este proceso es terminado por la creación y evaluación de un programa “correcto” o el reconocimiento de algún otro criterio de parada. [3]

El cruzamiento se efectúa seleccionando un subárbol de cada uno de los padres e insertando uno en el otro para crear los descendientes. La mutación es aplicada a un solo árbol padre para crear un hijo. El típico operador de mutación usado selecciona un punto dentro del padre y genera un nuevo subárbol aleatorio, usualmente de la misma forma que fue formada la población inicial, remplazando el seleccionado. Los programas computacionales escritos por la Programación Genética no son en lenguajes comunes como MATLAB o Fortran sino por lo general en el lenguaje LISP, el cual usa expresiones en lugar de instrucciones. [20]

Las Estrategias Evolutivas, fueron definidas por Rechenberg (1965, 1973) y Schwefel (1965, 1977) normalmente usan mutaciones distribuidas para modificar vectores de valores reales y resaltan la mutación y recombinación como operadores esenciales. El operador de selección es determinístico y el tamaño de las poblaciones de padres y descendentes usualmente difieren entre ellas [3]. Por ejemplo una estrategia evolucionista podría ser que en cada generación los padres mueran y solo los descendientes interviniesen en el proceso de selección, otra estrategia podría considerar que tanto padres como

hijos participen en el proceso de selección y que en cada generación un pequeño número de descendientes sobreviva [37].

La Programación Evolutiva fue desarrollada por Lawrence J Fogel en 1962 en la Universidad de California en San Diego, la misma hace énfasis en la mutación y no incorpora la recombinación de individuos. El operador de selección es probabilístico. En la actualidad se han usado en espacios de búsqueda de vectores de valores reales, pero fueron inicialmente pensados para evolucionar máquinas de estado finitas. [3]

Independientemente que a menudo los Algoritmos Genéticos son vistos como optimizadotes de funciones, han sido aplicados a una gran variedad de problemas. Su uso en sistemas difusos ha atraído considerablemente la atención debido a que les incorpora capacidades de aprendizaje y adaptación, a este enfoque se le conoce como Sistemas Difusos Genéticos. El paradigma usual es un sistema de clasificación donde se trata de evolucionar un conjunto de reglas difusas. [4]

1.6 Sistemas Difusos Genéticos

En la década de 1990 la falta de capacidades de aprendizaje que caracterizó la mayoría de los sistemas difusos generó un gran interés en su estudio. Dos de los métodos más exitosos han sido los intentos de hibridación hechos en el campo del Soft Computing, donde diferentes técnicas, tales como neuronales y evolutivas, proveen sistemas difusos con capacidades de aprendizaje. [17]

“Soft Computing es un consorcio de metodologías que trabajan sinérgicamente y proveen, de una forma u otra, información flexible para manipular situaciones ambiguas de la vida real”. [35]

Esta área de la computación permite procesar imprecisiones, incertidumbre, verdades parciales por lo que logra una gran robustez y soluciones de bajo costo. La principal característica es que se enfoca en el modelo de la mente humana. Está compuesto de varios campos como: Redes Neuronales, lógica difusa, razonamiento probabilístico, Algoritmos Genéticos y teoría de caos, las que se pueden ver como complementarias. Los métodos de Soft Computing

incorporan todas las características de dichos campos y además tiene la habilidad de vencer las dificultades y limitaciones que caracterizan a cada uno. Los enfoques híbridos posibles son neuronales-difusos, difusos-genéticos, neuronales-genéticos y sistemas neuro-difusos genéticos [49]. Los sistemas neuro-difusos son uno de los más exitosos, sin embargo otro enfoque de hibridación importante son los Sistemas Difusos Genéticos, en los cuales, básicamente, un sistema difuso incorpora habilidades de aprendizaje mediante Algoritmos Evolutivos.

Después de más de quince años de desarrollo, los Sistemas Difusos Genéticos han dejado ya su posición “emergente”, pues cientos de artículos han sido publicados, se han organizado sesiones en conferencias, editado temas especiales sobre ellos en revistas y se han escrito y publicado varios libros. El tema está bastante establecido y se pueden definir tres estados diferentes en su evolución [10]:

- En el primero, entre 1991 y 1997 se definieron las primeras bases del área de investigación. Se hicieron muchas contribuciones al aprendizaje de la Base de Conocimientos de un sistema difuso, usando los tres enfoques clásicos de aprendizaje genético en la derivación de reglas difusas para diferentes tipos de SBRD y diferentes aplicaciones (control difuso, modelación difusa y clasificación difusa). Además surgió la primera propuesta para ajustar funciones de pertenencia.

- Una segunda fase comenzó cerca de 1995 y todavía continúa activa, está especialmente enfocada en el ajuste de los sistemas difusos mediante Algoritmos Evolutivos. La importancia de estos procedimientos para mejorar el desempeño de los sistemas difusos fue reconocida y nuevos componentes de la Bases de Datos fueron adaptados aparte de las funciones de pertenencia difusas consideradas originalmente.

- Por último la tercera fase está basada en la propuesta de nuevos enfoques de aprendizaje de los Sistemas Difusos Genéticos, cuyo comienzo se puede establecer alrededor de 1998. Comprende varios

ramas que van desde los usos menos innovadores de nuevos Algoritmos Genéticos no clásicos, hasta el diseño de Sistemas Difusos Genéticos específicos para tratar con el importante problema de interpretabilidad y la exactitud en la modelación difusa. Además se ha puesto mucho esfuerzo en el diseño de métodos de aprendizaje híbridos, en el uso de Algoritmos Evolutivos multiobjetivo y en el diseño de Sistemas Difusos Genéticos capaces de tratar con problemas complejos de alta dimensión y largas bases de datos.

Existen varios tipos de Sistemas Difusos Genéticos que han sido desarrollados con resultados exitosos, como las Redes Neuronales Difusas Genéticas, que no son más el resultado de adicionar capacidades de aprendizaje genéticas o evolutivas a sistemas que integran conceptos neuronales y difusos. Sin embargo los Sistemas Difusos Genéticos más prominentes son los Sistemas Basados en Reglas Difusos Genéticos cuyo proceso genético aprende o ajusta diferentes componentes de un SBRD. [11]

1.6.1 Sistemas Basados en Reglas Difusos Genéticos

Los Algoritmos Evolutivos, especialmente los Algoritmos Genéticos, han probado ser una poderosa herramienta para automatizar la definición de una Base de Reglas Difusas. En los últimos años sus ventajas han extendido el uso de los Algoritmos Evolutivos en el desarrollo de una gran variedad de enfoques para el diseño de SBRD, los cuales reciben el nombre de Sistemas Basados en Reglas Difusos Genéticos. [14]

El aspecto central en el uso de un Algoritmo Genético para el aprendizaje automático de un SBRD es que el proceso de diseño de la Base de Conocimientos puede ser analizado como un problema de búsqueda o de optimización y los Algoritmos Genéticos son una poderosa herramienta para ello.[22]

El primer paso para el diseño de un Sistema Basado en Reglas Difuso Genético es decidir cuáles partes de la Base de Conocimientos son objeto de optimización por el Algoritmo Genético [21]. Existen varios métodos para el

diseño de los Sistemas Basados en Reglas Difusos Genéticos, que se basan en el aprendizaje o ajuste de los componentes de la Base de Conocimientos.

El ajuste es concerniente con la optimización de un SBRD dado, mientras que el aprendizaje constituye un método de diseño automatizado para un conjunto de reglas que comienza desde cero. El proceso de ajuste asume una Base de Reglas predefinida y tiene el objetivo de encontrar un conjunto de parámetros óptimos para las funciones de pertenencia. El proceso de aprendizaje lleva a cabo una búsqueda más elaborada en el espacio de posibles Bases de Reglas o una Base de Conocimientos completa y no depende de un conjunto de reglas predefinido. [11]

La estructura de los Sistemas Basados en Reglas Difusos Genéticos se muestra en la siguiente figura [22]:

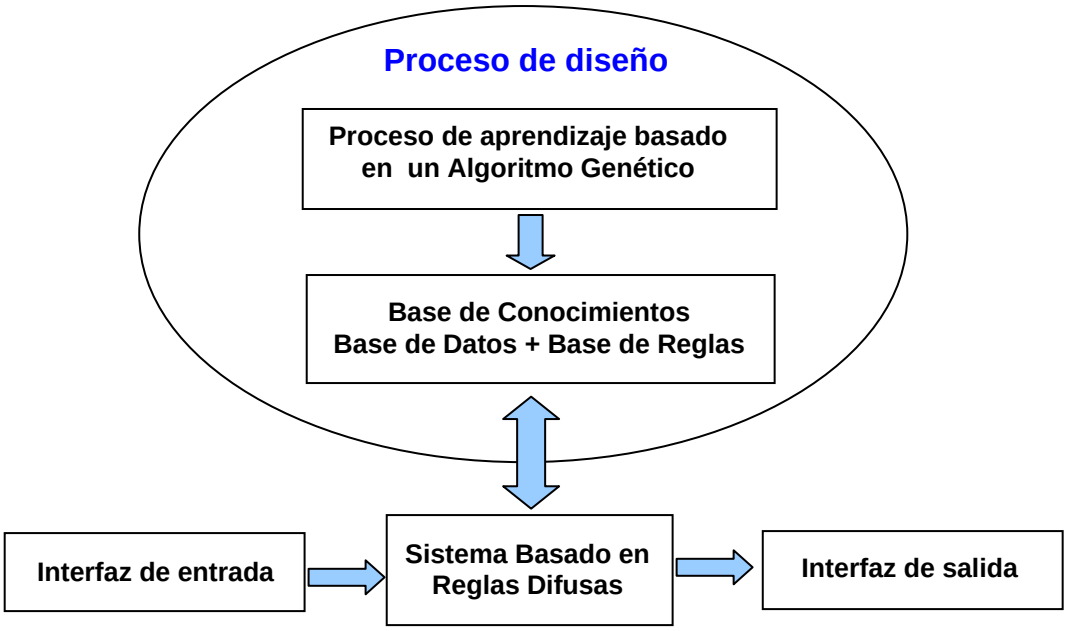

Figura1.3. Esquema de los Sistemas Basados en Reglas Difusos Genéticos

A continuación se describen los métodos seguidos en el diseño de un Sistema Basado en Reglas Difuso Genético para el aprendizaje o ajuste de la Base de Conocimientos [21]:

1. El aprendizaje genético de la Base de Reglas asume un conjunto funciones de pertenencia difusas en la Base de Datos las cuales son

referidas por las reglas mediante etiquetas lingüísticas. La mayoría de los enfoques se han centrado en aprender la Base de Reglas usando una Base de Datos predefinida. La forma usual de definir esta Base de Datos involucra la selección de un número de términos lingüísticos para cada variable lingüística (un número impar entre 3 y 9, el que a menudo es el mismo para todas las variables) y la implantación de los valores de los parámetros del sistema de acuerdo a una distribución uniforme de los términos lingüísticos en el universo de variables en cuestión.

2. El ajuste genético tiene el objetivo de hacer que los SBRD mejoren su desempeño. A partir de una Base de Reglas predefinida se trata de mejorar la definición preliminar de la Base de Datos buscando un conjunto de parámetros óptimos para las funciones de pertenencia. No obstante, este proceso solo ajusta la forma de las funciones de pertenencia y no el número de términos lingüísticos en cada partición difusa, los cuales permanecen ajustados desde el comienzo del proceso de diseño.

3. El aprendizaje genético de la Base de Conocimientos trata de aprender los dos componentes de la Base de Conocimientos simultáneamente. Trabajando de esta forma tiene la posibilidad de generar mejores definiciones, pero se enfrentan a un largo espacio de búsqueda que hace el proceso de aprendizaje más difícil y lento.

4. En el aprendizaje genético de la Base de Datos *a priori* se consideran dos procesos diferentes para derivar ambos componentes de la Base de Conocimientos (Base de Datos + Base de Reglas). El proceso de generación de la Base de Datos permite aprender la forma de las funciones de pertenencia y otros componentes de la Base de Datos. Además este proceso puede usar una medida para evaluar la calidad de la Base de Datos.

   Existe otra variante para el aprendizaje genético de la Base de Datos que envuelve el aprendizaje de la Base de Reglas, trabajando de la forma siguiente: cada vez que se obtiene una Base de Datos mediante

el proceso de definición, el método de generación de la Base de Reglas es usado para derivar las reglas y se usa alguna medida de error para validar toda la Base de Conocimientos obtenida. Se debe resaltar que este modo de operación incluye un particionamiento del problema de aprendizaje de la Base de Conocimientos.

Cuando se considera la tarea de aprender un conjunto de reglas de un SBRD existen dos métodos que pueden ser usados por el proceso de aprendizaje genético para codificar las reglas en una población de individuos, los cuales de describen a continuación:

En el primero conocido como método Pittsburg, cada miembro de la población representa un conjunto de reglas para el problema en cuestión. Aquí el cruzamiento, y otros operadores son a menudo empleados para cambiar el número de reglas de un miembro de una población dada. Tiene la ventaja de evaluar una solución completa dentro de cada individuo del Algoritmo Genético y por tanto este puede converger a una población homogénea como en un problema de optimización, tomando el mejor individuo, localizado por la búsqueda del Algoritmo Genético, como solución. La desventaja es que cada miembro de la población debe ser completamente evaluado como un conjunto de reglas. [3]

En el método Michigan, cada miembro de la población es una sola regla. Solo se necesita evaluar un individuo, no obstante no se pueden usar los procedimientos usuales de un Algoritmo Genético que convergerá a una población homogénea, pues una regla es improbable que resuelva el problema completo, se debe coevolucionar un conjunto de ellas para que juntas resuelvan el problema. Esto requiere un Algoritmo Genético que proporcione una población diversa. [3]

El método IRL (Iterative Rule Learning) es una variante del Michigan donde cada cromosoma representa una regla también, pero por el contrario solo el mejor individuo obtenido se toma en consideración para formar parte de la Base de Conocimientos final. Por lo que en este enfoque el Algoritmo Genético da una solución parcial al problema de aprendizaje y debe ser ejecutado varias

veces para obtener la Base de Conocimientos completa, la cual está formada por los mejores individuos obtenidos en cada iteración [12]. Una de sus características más importantes es que reduce sustancialmente el espacio de búsqueda, pues en cada iteración solamente se busca una regla, la mejor [4]. MOGUL y SLAVE son dos propuestas que siguen esta filosofía.

Los Sistemas Basados en Reglas Difusos Genéticos han sido ampliamente aplicados en la Minería de Datos, debido a que combinan las ventajas tanto de los SBRD como de los Algoritmos Genéticos. Una de las premisas fundamentales en cualquier proceso de extracción de conocimiento es la generación de modelos legibles, siendo sin duda alguna las reglas difusas una poderosa herramienta para ello [8]. Son consideradas una forma muy útil para representar el conocimiento en el descubrimiento de relaciones intrínsecas en bases de datos, pues permiten enfrentarse con incertidumbres, son fácilmente entendibles, se puede incorporar información adicional de forma muy fácil, los grados de precisión son fácilmente adaptables a las condiciones del problema y el proceso puede ser casi por completo automático. No obstante es necesaria además la extracción automática de reglas difusas con buenas propiedades, para lo cual existen diferentes métodos de aprendizaje entre los que se encuentran los Algoritmos Genéticos. Las principales razones para usarlos, en lugar de otras técnicas de Aprendizaje Automático bien conocidas, son las siguientes. En primer lugar, permiten reducir la complejidad de las reglas [29]. En segundo lugar, usualmente la interpretabilidad y la precisión son objetivos cruciales en el proceso de KDD, donde el conocimiento extraído de las bases de datos debe ser representado de forma comprensible, por lo que es muy importante optimizar ambos objetivos, lo cual es generalmente imposible pues son contradictorios. Sin embargo los Algoritmos Genéticos tienen una poderosa capacidad de búsqueda que permite trabajar con optimización multiobjetivo. En tercer lugar pueden manejar estructuras de representación flexibles mezclando esquemas de codificación o incluyendo restricciones [32]. En la literatura se pueden encontrar disímiles propuestas de Sistemas Basados en Reglas Difusos Genéticos, algunas de las cuales se considera necesario describir.

Capítulo II. Descripción de varios Sistemas Basados en Reglas Difusos Genéticos.

En este Capítulo se resumirán las principales características de la herramienta de Minería de Datos escogida para realizar el estudio experimental comparativo. Además se describirán detalladamente los Sistemas Basados en Reglas Difusos Genéticos, incluidos en la misma que fueron evaluados.

2.1 KEEL: herramienta de Minería de Datos

En la actualidad se pueden encontrar varias librerías y herramientas para la Minería de Datos. Muchas de ellas son distribuidas comercialmente, entre las más conocidas están: SPSS Clementine, Oracle Data Mining y KnowledgeSTUDIO. Sin embargo también existen otras no-comerciales, libres y con código abierto como: MiningMart, Orange, Tanagra y Weka. Esta última es una colección de implementaciones de algoritmos de Aprendizaje Automático en Java y probablemente una de las más conocidas. Dentro de las herramientas no-comerciales también se incluye KEEL (*Knowledge Extraction based on Evolutionary Learning*) que es desarrollada en Java y permite evaluar Algoritmos Evolutivos para problemas de Minería de Datos de varios tipos, como regresión, clasificación, aprendizaje no supervisado, etc. Incluye algoritmos de aprendizaje evolutivos basados en diferentes enfoques, Michigan, Pittsburg e IRL, así como la integración de técnicas de aprendizaje evolutivas con diferentes técnicas de preprocesamiento. Se considera una herramienta muy prometedora para la Minería de Datos debido a que en años recientes los Algoritmos Evolutivos, particularmente los Algoritmos Genéticos, han probado ser una importante técnica para el aprendizaje y la extracción de conocimiento. Es importante destacar que es el primer software de este tipo que contiene una librería de algoritmos de aprendizaje evolutivos con código abierto en Java. [2]

La versión reciente de KEEL consiste en los siguientes componentes:

- Administración de los datos: está compuesta por un conjunto de herramientas que pueden ser usadas para exportar datos al formato de

KEEL o importarlos. También permite construir nuevos datos, editarlos y visualizarlos para aplicar transformaciones y particionarlos, etc.

- Diseño de experimentos: el objetivo de este componente es el diseño de los experimentos deseados con los conjuntos de datos seleccionados, para lo cual se proveen diferentes tipos de validación y problemas de aprendizaje (clasificación, regresión, aprendizaje no supervisado). Este proceso normalmente consiste en la mezcla de Algoritmos Evolutivos y técnicas estadísticas.

- Experimentos educacionales: tiene una estructura similar a la parte anterior, permite diseñar experimentos que serán ejecutados paso a paso con el objetivo de mostrar el proceso de aprendizaje de un modelo determinado con propósitos educativos.

Entre sus principales características se encuentran:

- Contiene una librería estadística para analizar los resultados de los algoritmos, así como un conjunto de test estadísticos para llevar a cabo comparaciones paramétricas y no paramétricas de los algoritmos.

- Posee una interfaz amigable al usuario, orientada al análisis de los algoritmos.

- El software está diseñado para experimentos que contengan múltiples conjuntos de datos y algoritmos conectados entre ellos.

- Los experimentos son generados independientes de la interfaz de usuario, por lo que no es necesario tener la herramienta para poder ejecutarlos, el único requerimiento es que la máquina virtual de Java esté instalada.

- La herramienta no está diseñada para la vista en tiempo real del progreso de los algoritmos, debido a que requieren un gran esfuerzo computacional, pero se genera un script que puede ser ejecutado en un cluster de computadoras.

- Contiene una Librería de Algoritmos de Extracción de Conocimiento con la incorporación de múltiples algoritmos de aprendizaje evolutivo junto con los enfoques clásicos de aprendizaje. Las principales técnicas incluidas son:

    - Modelos de aprendizaje de reglas evolutivos.
    - Sistemas difusos.
    - Redes Neuronales evolutivas.
    - Programación Genética.
    - Descubrimiento de subgrupos.
    - Selección de atributos y discretización.

Entre las ventajas de la herramienta se pueden mencionar:

- Reduce el trabajo de programación. Incluye una librería con algoritmos de aprendizaje evolutivos basados en diferentes paradigmas y simplifica la integración de estos con técnicas de pre-procesamiento

- Amplía el rango de posibles usuarios, pues la extensa librería de Algoritmos Evolutivos junto con un software fácil de usar reduce considerablemente el nivel de conocimiento y experiencia requerido por los investigadores en el campo de la computación evolutiva. Como resultado los que tienen menos conocimiento al usar este framework, serán capaces de aplicar satisfactoriamente los algoritmos a sus problemas.

- Debido al estricto enfoque orientado a objetos usado para la librería y el software, este puede ser usado en cualquier máquina con Java. Cualquier persona pude usar KEEL independientemente de su sistema operativo.

Los criterios abordados con anterioridad permitieron a la autora elegir a KEEL como herramienta de Minería de Datos para la evaluación del desempeño de varios Sistemas Basados Reglas Difusos Genéticos incluidos en la librería de algoritmos de aprendizaje evolutivos que posee la misma.

2.2 Sistemas Basados en Reglas Difusos Genéticos incluidos en KEEL

De los Sistemas Basados Reglas Difusos Genéticos para la tarea de clasificación que posee KEEL algunos fueron seleccionados para realizar el estudio experimental que constituye el objetivo de investigación, ellos son: Fuzzy-GP, Fuzzy- Ishibuchi99, Fuzzy- MOGUL, Fuzzy- AdaBoost, Fuzzy- LogitBoost, Fuzzy- MaxLogitBoost, Fuzzy- SLAVE, Fuzzy- GAP, Fuzzy- SAP y Fuzzy- Ishib-Hybrid, también se incluye MR-GIM (learning Multiple Rules using the Genetic Iterative Model), el cual no forma parte de la librería que contiene la herramienta. Cada uno de estos sistemas será descrito en el transcurso del epígrafe.

2.2.1 Fuzzy-GP

En esta propuesta descrita en [6] el objetivo es obtener reglas difusas mediante un proceso de aprendizaje a través de un Algoritmo Genético. Se obtiene un reducido conjunto de reglas difusas, con pocas condiciones por regla en el antecedente y gran capacidad de generalización debido a la definición de una gramática de libre contexto junto con el uso de un mecanismo de competencia entre reglas. Cada individuo de la población está representado por una regla del tipo DNF:

IF X_1 *es* A_1 **AND... AND** X_n *es* A_n **THEN** Y **is** B**, grado de certeza** C**.**

Donde cada variable X_i toma como valor un conjunto de términos lingüísticos $A_i = (A_{il}\ or ... or\ A_{iLi})$ cuyos miembros están unidos por un operador disyuntivo, mientras que la variable de salida Y tiene uno de los valores de la clase. La regla incluye además un grado de certeza $C \in [0,1]$ que representa la confianza de que la clase representada por el consecuente sea cierta bajo las condiciones del antecedente de la regla.

Los pasos principales de este método son los siguientes:

Paso1: definición de la gramática de acuerdo al problema que se va a resolver. Esta gramática debe especificar la estructura de una regla de la forma IF *antecedente* THEN *consecuente*.

Paso2: la Base de Datos es definida arreglando los parámetros de los conjuntos difusos asociados con las etiquetas presentes en la gramática.

Paso3: aplicación de los operadores genéticos. Una vez que la gramática y la base de datos han sido definidas correctamente se genera aleatoriamente una población inicial de reglas, de acuerdo a la gramática de producción de reglas. En cada iteración, algunos individuos son seleccionados para producir descendientes mediante los operadores genéricos siguientes:

- Cruzamiento: produce un hijo a partir de dos padres seleccionados aleatoriamente, pero bajo la condición de que los descendientes deben ser válidos de acuerdo a la gramática. Una parte del primer padre es seleccionada y reemplazada en el segundo.

- Mutación: una parte de la regla es seleccionada y reemplazada por otra nueva generada aleatoriamente siguiendo el mismo mecanismo que para la creación de la población.

- Condición de caída (*dropping condition*): debido a la naturaleza probabilística de los Algoritmos Genéticos, se pueden generar restricciones redundantes en las reglas. Por tanto es necesario generalizarlas para representar el conocimiento de una forma más concisa. Este operador selecciona aleatoriamente una condición en el antecedente, cuyo atributo no se toma en consideración, de esta forma la regla puede ser generalizada. En resumen se fuerza la selección de atributos en las reglas, permitiendo obtener un pequeño número de variables por antecedente en cada una.

Paso4: evolución de la población. En este momento se aplica un proceso denominado Competición de Tokens (*Token Competition*) [50] que se lleva a

cabo con el objetivo de mantener la diversidad de la población. El mismo asume que cada ejemplo en el conjunto de entrenamiento puede proveer un recurso llamado token, que para poderlo capturar todos los cromosomas en la población deben competir. Si un individuo (regla) logra coincidir con el ejemplo levanta una bandera indicando que el token es tomado y por tanto otros más débiles no pueden entonces obtenerlo. La prioridad para recibir tokens está determinada por la fortaleza de los individuos, los que tienen los mayores valores de adaptación pueden explorar el nicho tomando tantos ejemplos como puedan. Sus valores de adaptación son modificados en dependencia de la cantidad de tokens que pueden tomar, y se define de la siguiente forma:

Modified_fitness = raw_ fitness × count / ideal

Donde ***raw_ fitness*** es el valor de adaptación obtenido de la evaluación de la función de adaptación, ***count*** es el número de tokens que el individo toma en realidad e ***ideal*** es el número total de tokens que puede tomar, que es igual al número de ejemplos con los que coincide.

Como resultado de la Competencia de Tokens, algunos individuos tienen sus valores de adaptación modificados en cero, por lo que deben ser reemplazados por otros que concuerden con ejemplos no cubiertos. En caso de que todos hayan sido cubiertos, estos individuos son eliminados. Finalmente la población es reducida a la mitad de su tamaño inicial.

Paso5: se lleva a cabo la simplificación de la Base de Reglas, una vez terminado el proceso evolutivo se ejecuta una fase de post-procesamiento para eliminar las reglas redundantes.

Paso6: finalmente el método termina con el cálculo del grado de certeza de cada individuo de la población. El mismo se obtiene a partir del cociente siguiente: $\frac{S_j}{S}$ donde S_j es la suma de la cantidad de ejemplos de entrenamiento pertenecientes a la clase del consecuente de la regla y que están cubiertos por su antecedente también y S es la suma de la cantidad de

ejemplos entrenamiento que son cubiertos por el antecedente de la regla independientemente de la clase a que pertenezcan.

2.2.2 Fuzzy- Ishibuchi99

Este es un método propuesto en [24] que consiste en la obtención de un modelo de clasificación mediante la generación de un SBRD. Cada regla difusa es manejada como un individuo asignándosele un valor de adaptación a cada una. Está basado en un proceso de generación heurístico de reglas difusas y el uso de operadores genéticos básicos tales como selección, cruzamiento y mutación. La estructura usada para representar las reglas es la DNF:

R_j : **IF** X_1 *es* A_{j1} **AND... AND** X_n *es* A_{jn} **THEN** c_j , **grado de certeza** C

Donde $A_{j1},...,A_{jn}$ son conjuntos difusos, siendo *n* el número de atributos, c_j es la clase y C es el grado de certeza de la regla difusa. La clase c_j y C de cada regla difusa se determinan mediante un procedimiento heurístico simple, el cual se encuentra detallado en [24].

Los valores que toman las variables en las reglas se agrupan en cinco etiquetas lingüísticas como se muestra en la siguiente tabla:

P: *pequeño*	1
MP: *medio pequeño*	2
M: *medio*	3
ML: *medio largo*	4
L: *largo*	5
I: *irrelevante*	#

Tabla2.1. Etiquetas lingüísticas

Por ejemplo una cadena para ejemplos de cuatro dimensiones sería: “1#3#” denotando la regla:

IF X_1 **es *pequeño* AND** X_3 **es *medio* THEN clase es** c_j , **grado de certeza** C

De forma general este sistema se puede ser descrito según el algoritmo siguiente:

Paso1: generación de la población inicial de N_{pob} reglas difusas, seleccionando aleatoriamente sus conjuntos difusos del antecedente a partir de los seis símbolos correspondientes a los cinco valores lingüísticos e "irrelevante". La probabilidad que tiene cada símbolo de ser escogido es de 1/6. Los componentes que forman el consecuente, c_j y C, son determinados siguiendo el procedimiento heurístico simple.

Paso2: evaluación de cada regla de la población. Se clasifican todos los ejemplos de entrenamiento por el SBRD con el conjunto de reglas difusas S usando un método de razonamiento difuso, el cual dado un ejemplo (X_p) lo clasifica mediante la búsqueda de una regla en el conjunto S, que se denomina "regla ganadora" y se determina según la expresión:

$$\bar{R}_j = Max\left\{\mu_j(X_p) * gc_j\right\}$$

Si más de una regla R_j tiene el máximo valor pero clases diferentes para X_p, este no se clasifica. Otro caso en que el ejemplo no se puede clasificar es cuando no hay ninguna regla compatible con él ($\mu_j(X_p) = 0 \ \forall R_j \in S$). Cuando más de una regla difusa con la misma clase consecuente tiene el mismo valor *max* en la expresión anterior se asume que la regla con el menor subíndice *j* es la ganadora, lo cual previene a la población de ser dominada por reglas difusas homogéneas. A cada regla se le asigna un punto cada vez que un ejemplo es clasificado por ella. Después que todos los ejemplos de entrenamiento son examinados, el valor de adaptación (*fitness (*R_j*)*) correspondiente a cada regla es igual a la cantidad de puntos obtenidos por la misma, es decir a la cantidad de ejemplos que clasifica correctamente. Este valor es actualizado en cada iteración.

Paso3: generación de nuevas reglas mediante operaciones genéticas. Primero se seleccionan un par de reglas de la población de acuerdo a determinada probabilidad basada en el la selección de la ruleta que se determina según la expresión siguiente:

$$P(R_j) = \frac{fitness(R_j) - fitness_{\min}(S)}{\sum_{R_i \in S} \{fitness(R_i) - fitness_{\min}(S)\}}$$

Donde $fitness_{\min}(S)$ es el menor valor de adaptación de las reglas difusas en la población actual. A partir de las reglas escogidas se generan dos más mediante el cruzamiento uniforme para los conjuntos difusos del antecedente. Luego se aplica el operador de mutación a cada conjunto difuso del antecedente de las nuevas reglas generadas por el operador de cruzamiento, los cuales son reemplazados aleatoriamente usando una probabilidad de mutación predefinida. La clase del consecuente y el grado de certeza de cada nueva regla son determinados después del proceso de mutación, siguiendo el procedimiento heurístico. Estas operaciones genéticas son iteradas hasta que un número predefinido de reglas son generadas.

Paso4: reemplazar parte de la población actual por las nuevas reglas**.** Un número predefinido N_{remp} de reglas difusas de la población actual son reemplazadas por las que fueron generadas en el paso anterior. En este método las N_{remp} reglas con los menores valores de adaptación son eliminadas de la población actual para adicionar las nuevas.

Paso5: terminar el algoritmo si se cumple la condición de parada predefinida, si no retornar al **Paso2**. Se puede usar cualquier condición, como puede ser el número total de generaciones. La solución final del este método es el conjunto de reglas difusas que mejor clasifica a todos los ejemplos de entrenamiento. Es decir que la solución final no es la última población sino la mejor de todas.

2.2.3 Fuzzy- MOGUL

Por otra parte MOGUL descrito en [14] es una metodología que consiste en determinadas directivas de diseño que permiten obtener diferentes Sistemas Basados en Reglas Difusos Genéticos capaces de resolver disímiles problemas (modelación difusa, control, clasificación y otras). Cualquier usuario puede añadir sus requerimientos a las especificaciones de MOGUL para construir diferentes tipos de SBRD, tipo Mamdani descriptivo, tipo Mamdani-aproximado

y TSK, que satisfaga sus necesidades. El principal problema que tiene que ser resulto en el diseño de un Sistema Basado en Reglas Difuso Genético es encontrar una representación apropiada capaz de agrupar las características del problema y de representar las posibles soluciones. Clásicamente se han adoptado dos enfoques de aprendizaje genético, el Pittsburg y el Michigan, sin embargo el usado por la propuesta en consideración es el IRL. También es muy importante escoger un esquema apropiado para codificar los individuos, se recomienda usar una codificación entera cuando se representan etiquetas lingüísticas, una real para representar funciones de pertenencia y un esquema denominado codificación angular para los parámetros del consecuente de las reglas TSK. Este método tiene una estructura genérica consistente en tres pasos principales: generación del conjunto inicial de reglas, simplificación y ajuste.

Generación de reglas difusas: se obtiene un conjunto de reglas de clasificación lingüísticas representando el conocimiento existente en los ejemplos de entrenamiento. En todos los casos está compuesto por dos componentes: un Método de Generación de Reglas Difusas, y el Método de Cubrimiento Iterativo. De forma general este proceso puede ser resumido por los pasos siguientes:

Paso1: se define una partición difusa para cada variable con funciones de pertenencia triangular uniformemente distribuidas. El número de términos lingüísticos que forman cada partición debe ser especificado.

Paso2: para cada ejemplo de entrenamiento generar la regla difusa que mejor lo clasifique, teniendo para cada atributo la etiqueta lingüística que mejor lo represente. Incluir esta reglas en el conjunto reglas candidatas B^c si no ha sido incluida antes. Luego evaluar todas las reglas difusas contenidas en B^c y seleccionar la más prometedora.

Paso3: finalmente, la regla obtenida es adicionada al conjunto final de reglas difusas. Los datos cubiertos por este conjunto con determinado grado de certeza son eliminados y no se consideran para futuras iteraciones. El proceso iterativo termina cuando el conjunto de entrenamiento queda vacío, es decir no queda ningún ejemplo sin cubrir.

Proceso de simplificación genética: está basado en una codificación binaria del Algoritmo Genético con la longitud de los cromosomas ajustada. Consiste en la simplificación de la Base de Reglas obtenida hasta el momento eliminando las reglas difusas redundantes o innecesarias que pudieran causar una operación incorrecta, mediante un proceso de selección genética el cual es desarrollado usando un procedimiento de muestreo universal estocástico junto con un esquema de selección elitista. La generación de la población de los descendientes se lleva cabo usando el clásico cruzamiento binario sobre varios puntos y el operador de mutación uniforme. Este paso permitirá obtener diferentes definiciones de la Base de Reglas Difusas con la mejor cooperación entre las reglas que las componen.

Proceso de ajuste genético: los parámetros que definen las funciones de pertenencia de los conjuntos difusos son optimizadas mediante un proceso ajuste genético. Cada cromosoma que forma la población codificará definiciones diferentes de la Base de Datos que será combinada con la Base de Reglas existente. El Algoritmo Genético diseñado emplea un esquema de codificación real, usa un muestreo universal estocástico como procedimiento de selección y el operador de mutación no-uniforme de Michaelewicz. Además emplea el operador de cruzamiento aritmético min-max, el cual hace uso de herramientas difusas para mejorar su desempeño. En el caso de un SBRD tipo Mamdani descriptivo se lleva cabo un ajuste global de las particiones difusas asociadas a cada variable lingüística, pero en la variante aproximada las funciones de pertenencia involucradas en las reglas difusas serán ajustadas individualmente. Por otra parte en el caso de los TSK el antecedente será ajustado de la misma forma que en el enfoque descriptivo y la definición preliminar de los parámetros del consecuente obtenidos en el primer estado serán refinados también. De esta forma el proceso de ajuste genético de las diferentes definiciones de las Bases de Reglas Difusas generadas en el paso anterior proporcionará como salida del Sistema Basado en Reglas Difuso Genético la mejor Base de Reglas Difusas.

2.2.4 Fuzzy- SLAVE

SLAVE (*Structural Learning Algorithm in Vague Enviroment*) [16, 17] es un algoritmo de aprendizaje inductivo de reglas difusas que sigue el esquema básico del modelo genético iterativo o lo que es lo mismo el enfoque IRL. La tarea principal de este algoritmo consiste en dado un conjunto de ejemplos de entrenamiento E y una clase B, encontrar la mejor combinación de los valores $A(A_1, \ldots, A_p)$ de las variables del antecedente para describir los ejemplos en el conjunto E. Esta tarea se lleva a cabo de la siguiente forma: la mejor regla que representa los ejemplos de la clase B es seleccionada e incluida en el conjunto de reglas. Si se necesitan más reglas para cubrir los ejemplos de la clase B, los cubiertos por la regla anterior son eliminados del conjunto de entrenamiento y el proceso se repite hasta que no se necesiten más reglas. Cuando todas las reglas para una clase han sido aprendidas, el mismo proceso se repite para el resto de las clases. La mejor regla es aquella que cubre la mayor cantidad de ejemplos con la menor cantidad posible de ejemplos negativos. Esta tarea se formaliza como un problema de búsqueda, y de entre los distintos algoritmos que son aplicables, se ha utilizado un Algoritmo Genético, el cual aprende una sola regla en cada iteración y el conjunto de reglas es obtenido a través de una secuencia de iteraciones. Además trabaja a dos niveles, el primero busca las variables relevantes para la descripción de la clase, mientras el segundo busca la mejor asignación de valores a estas variables.

El primero de los niveles, denominado nivel de variable, codifica la relevancia o irrelevancia de una variable para una regla particular. Se usa una codificación real con n+1 componentes (siendo n el número de variables implicadas en el problema). El i-ésimo elemento contiene un valor entre 0 y 1 que representa el grado de relevancia de la i-ésima variable con respecto a la clase. El valor n+1 es un valor entre 0 y 1 que representa un umbral de activación, de manera que la i-ésima variable se considerará relevante para el antecedente de la regla si su valoración es mayor o igual que este umbral de activación. En otro caso, la variable se considerará como irrelevante y será eliminada de la descripción de la regla.

En la generación de la población inicial, se calcula una estimación de la relevancia a priori de cada variable con respecto a la clase. El proceso evolutivo cambiará estos valores iniciales para obtener una mejor estimación de la relevancia de cada una de las variables. Los operadores genéticos definidos para evolucionar estos valores son los operadores de cruce uniforme y el de mutación no uniforme.

El segundo de los niveles, denominado nivel de valor, representa las asignaciones de valores a las variables. Se utiliza codificación binaria que expresa la presencia o ausencia de la asignación a una variable de cada uno de los valores de su dominio. SLAVE usa el modelo de regla difusa DNF que permite asignar un subconjunto de valores de su dominio a cada una de las variables. Los operadores genéticos definidos sobre este nivel son el cruce sobre dos puntos y la mutación uniforme.

El objetivo del Algoritmo Genético es, dada una clase, seleccionar el mejor antecedente. La medida fundamental utilizada para valorar una regla se basa en las condiciones de consistencia y completitud. Además, en la evaluación de las reglas, también se tienen en cuenta dos funciones que miden la simplicidad y la comprensibilidad de las mismas. La función de evaluación del Algoritmo Genético combina las medidas anteriores utilizando una evaluación funcional lexicográfica, es decir, hay establecido un orden de prioridad en la evaluación de los criterios.

El Algoritmo Genético sigue un modelo generacional elitista y con operador de selección basado en la ruleta, donde la probabilidad de los individuos se calcula siguiendo un ranking lineal. El proceso termina cuando la mejor solución encontrada no se mejora en n generaciones, devolviendo la mejor regla.

2.2.5 Fuzzy- Ishib-Hybrid

Este algoritmo, descrito en [27], está basado en una combinación entre los métodos Pittsburg y Michigan con el objetivo de construir una Base de Reglas Difusas para un problema de clasificación. Primeramente se genera una

población donde cada individuo es un conjunto de reglas difusas (Pittsburg). Luego se utilizan operaciones genéticas para crear nuevas reglas difusas bajo el enfoque Michigan como un tipo de mutación heurística para modificar parcialmente cada conjunto de reglas.

Se tiene m ejemplos etiquetados $x_p = (x_{p1},...,x_{pn})$, $p = 1,2,...,m$ de M clases como datos de entrenamiento. Un conjunto de valores lingüísticos y sus funciones de pertenencia son dados para describir cada atributo. Las reglas difusas usadas son de la forma DNF:

R_q : **IF** X_1 *es* A_{q1} **AND... AND** X_{qn} *es* A_{qn} **THEN clase es** C_q **, con** CF_q

Donde $q = 1,2,...,N_{regla}$, R_q es la q-ésima regla difusa, $x_p = (x_{p1},...,x_{pn})$ conjunto de ejemplos, A_{qn} es un conjunto difuso antecedente con un valor lingüístico (*pequeño, largo*), C_q es la clase, CF_q es el peso de la regla y N_{regla} es el número de reglas difusas.

Para determinar la clase C_q se define el grado de compatibilidad de cada ejemplo de entrenamiento x_p con el antecedente $A_q = (A_{q1},...,A_{qn})$ usando el producto siguiente:

$$\mu_{A_q}(x_p) = \mu_{A_{q1}}(x_{p1}) \cdot \mu_{A_{q2}}(x_{p2}) \cdot \ldots \cdot \mu_{A_{qn}}(x_{pn})$$

Donde $\mu_{A_{qi}}$ es la función de pertenencia del conjunto difuso del antecedente A_{qi}. La probabilidad condicional difusa $\Pr(Clase\, h \mid A_q)$ de la clase h ($h = 1, 2, \ldots, M$) para el antecedente A_q es determinada como sigue:

$$\Pr(Clase\, h \mid A_q) = \sum_{x_p \in clase\, h} \mu_{A_q}(x_p) \Big/ \sum_{p=1}^{m} \mu_{A_q}(x_p)$$

Luego la clase C_q de la regla difusa R_q es la que tiene mayor valor de probabilidad condicional.

En el razonamiento difuso un nuevo ejemplo x_p es clasificado por la "regla ganadora" R_w perteneciente al conjunto de reglas difusas S, la cual se determina de la forma:

$$\mu_{A_w}(x_p)\cdot CF_w = \{\max \mu_{A_q}(x_p)\cdot CF_q \,|\, R_q \in S\}$$

La regla R_w es la que tiene el mayor producto entre $\mu_{A_q}(x_p)$ y el peso de la regla CF_q. Si varias reglas tienen el mismo valor max pero diferentes clases para el nuevo ejemplo, ó ninguna regla es compatible con él, la clasificación es rechazada. El peso CF_q es calculado siguiendo la siguiente ecuación:

$$CF_q = \Pr(Clase\, C_q \,|\, A_q) - \sum_{\substack{h=1 \\ h\neq C_q}}^{M} \Pr(Clase\, h \,|\, A_q)$$

Cuando CF_q es negativo no se genera ninguna regla difusa con el antecedente A_q. Es decir la definición del conjunto S no incluye reglas con pesos negativos.

Cada regla difusa se codifica como una cadena usando los conjuntos difusos del antecedente con quince símbolos (0, 1,…, 9, a, b, c, d, e) que representan un término "irrelevante" y catorce conjuntos difusos con su respectiva etiqueta lingüística Por ejemplo "0102d0" denota la regla difusa **IF** x_2 *es* S^2 **AND** x_4 *es* L^2 **AND** x_5 *es* ML^2 **THEN** *clase* C_q **con** CF_q, las condiciones x_1, x_3 *y* x_6 son irrelevantes y por tanto representadas con un "0".

El método Michigan usado por Fuzzy-Ishib-Hybrid consiste en los siguientes pasos:

Paso1: generar aleatoriamente la población inicial con N_{regla} reglas difusas compuesta por una cadena de longitud n seleccionando aleatoriamente cada uno de los quince símbolos con una probabilidad de 1/15.

Paso2: calcular el valor de adaptación de cada regla en la población actual. La evaluación de cada una de ellas se lleva a cabo mediante la clasificación de todos los ejemplos de entrenamiento por el conjunto de reglas S usando el método de la "regla ganadora". Una vez terminado el proceso anterior el valor de adaptación de cada regla $fitness(R_q)$ es calculado como el número de ejemplos de entrenamiento correctamente clasificados por la R_q.

Paso3: crear N_{nuevas}reglas difusas usando operaciones genéticas. Se seleccionan dos reglas difusas de la población actual usando la selección de torneo, a partir de estos individuos se generan dos reglas nuevas mediante el cruzamiento uniforme con una probabilidad predefinida. Cada símbolo de las nuevas cadenas generadas es reemplazado aleatoriamente por otros usando una probabilidad de mutación definida de antemano. Las tres operaciones genéticas mencionadas son iteradas hasta que un número N_{nuevas} de nuevas cadenas halla sido generada.

Paso4: reemplazar los N_{nuevas} peores individuos con los menores valores de adaptación en la población actual por las N_{nuevas}nuevas reglas para formar la nueva generación.

Paso5: los procedimientos anteriores son aplicados nuevamente desde el **Paso2** a la nueva población hasta que se halla generado un número predefinido de generaciones. Aunque se pueden usar otras condiciones de parada.

En el método Pittsburg usado en esta propuesta, un conjunto de reglas con N_{regla} reglas difusas es codificado como una cadena concatenada de longitud $n \times N_{regla}$, donde cada subcadena de longitud n representa una regla. Este algoritmo se puede definir según los pasos:

Paso1: generar aleatoriamente N_{pob} conjunto de reglas con N_{regla} reglas difusas de la misma forma que para el método Michigan. Los cuales componen la primera población.

Paso2: evaluar cada conjunto de reglas S_i mediante la clasificación de los ejemplos de entrenamiento, siendo sus valores de adaptación calculados como la cantidad de ejemplos clasificados correctamente por S_i **.**

Paso3: generar de forma aleatoria (N_{pob}-1) conjunto de reglas usando operaciones genéticas. Dos individuos son seleccionados de la población actual mediante el esquema binario de selección de torneo para crear dos nuevos usando el cruzamiento uniforme con su respectiva probabilidad predefinida. Luego cada símbolo de las cadenas creadas es reemplazado aleatoriamente por otros empleando una probabilidad de mutación definida con anterioridad. Estos operadores son iterados hasta que (N_{pob}-1) conjuntos de reglas hallan sido generados.

Paso4: el mejor conjunto de reglas en la población actual es añadido a los nuevos para formar la nueva población de tamaño N_{pob}.

Paso5: repetir nuevamente el proceso desde el **Paso2** hasta que se cumpla la condición de parada, que en este caso es usado el número total de generaciones.

El algoritmo híbrido compuesto por las dos variantes especificadas con anterioridad se puede describir como sigue:

Paso1: generar N_{pob} conjuntos de reglas con N_{regla} reglas difusas.

Paso2: calcular el valor de adaptación de cada conjunto de reglas de la población actual.

Paso3: generar (N_{pob}-1) conjuntos de reglas mediante la selección, cruzamiento y mutación de la misma forma que en el método Pittsburg. Aplicar

una sola iteración del método Michigan a cada uno de los conjuntos de reglas generadas con una probabilidad predefinida.

Paso4: agregar el mejor conjunto de reglas de la población actual a los nuevos (N_{pob}-1) conjuntos para formar la próxima generación.

Paso5: retornar la **Paso2** mientras no se satisfaga la condición de parada**.**

2.2.6 Fuzzy- SAP

La codificación de la Base de Reglas Difusas mediante un Algoritmo Genético, junto con sus correspondientes operadores de cruzamiento y mutación, pueden ser usados por otros esquemas de búsqueda, mejorando el comportamiento de estos últimos. Como consecuencia práctica de esto se ha desarrollado un método basado en *Simulated Annealing* (SA) para inducir la estructura de un clasificador difuso, que en este caso es una cadena válida en una gramática de libre contexto cuya definición se encuentra explícita en [41].

Se pueden usar dos formas diferentes de codificación de la base de reglas: la lineal y de árbol, siendo esta última la usada en este caso. Con la codificación en forma de árbol, bases de reglas completas son consideradas como una cadena simple en una gramática de libre contexto y son representadas mediante un árbol o por un par, compuesto por un árbol sintáctico y una cadena de parámetros. Ambos enfoques de esta última forma de codificación son denominados gramática basada en Programación Genética, la cual permite representar la base de reglas de forma más compacta. La sintaxis de las reglas difusas sigue el modelo DNF:

IF x_i *es* l_{ij} **THEN** c_k **, grado de certeza** α

Donde l_{ij} es la etiqueta lingüística número j del atributo i y define la relación siguiente:

$$R(x_1, \ldots, x_m, c) = \begin{cases} \alpha & si\ x_i = l_{ij}\ y\ c = c_k \\ 0 & en\ otro\ caso \end{cases}$$

En la gramática basada en Programación Genética, solo los subárboles generados por la misma regla de producción pueden ser intercambiados, para garantizar que la sintaxis de los descendientes sea correcta. No hay números en los nodos terminales de los árboles, pero si punteros a un arreglo. La mutación puede ser definida de diferentes formas, macromutación de subárbol y de cadena, las cuales serán usadas dentro de un método de búsqueda basado en SA para aprender simultáneamente las particiones difusas y la estructura superficial de la base de reglas. El operador de macromutación de subárbol adaptado a SA intercambia un subárbol del individuo que va a mutar por un subárbol del mismo tipo de un individuo aleatorio. El arreglo de parámetros no es alterado. Mientras que en la macromutación de cadena el arreglo de parámetros en el individuo que va a mutar es cruzado con el arreglo de parámetros de un individuo aleatorio. La expresión lingüística de la base de reglas no se modifica.

El algoritmo de macromutación se define como sigue:

Se necesita: C, p, T, K_1, K_2

Devuelve: C_1

Si $U(0,1) < p$ entonces

$A =$ vector real aleatorio

$$param(C) = param(C) * \left(T / K_1 \right) + \left(1 - \left(T / K_1 \right) \right) * A$$

Si no

Repetir

$E = \exp r(C)$

Seleccionar un subárbol E

Reemplazarlo por un subárbol generado aleatoriamente hasta que la distancia $(E, \exp r(C)) < T / K_2$

$\exp r(C) = E$

Sea C el genotipo de un individuo, $param(C)$ es el arreglo de parámetros y $\exp r(C)$ el árbol sintáctico de la cadena, T representa la temperatura actual en

SA, p es el balance entre la mutación de subárbol y la de cadena, si $p = 0$ se lleva a cabo la mutación de subárbol y si $p = 1$ entonces la mutación de cadena, esta consiste en aplicar la recombinación entre el individuo y uno generado aleatoriamente con un parámetro de amplitud que depende de la temperatura actual por una constante K_1 . La mutación de subárbol es dentro de un ciclo que termina cuando la distancia entre el resultado y el individuo es menor que un límite que también depende de la temperatura por el factor K_2.

El operador de macromutación y la representación de la Base de Reglas Difusas son incorporados a un algoritmo de SA, el cual depende de una temperatura inicial y una temperatura final, que sirve para detener el proceso de aprendizaje. El algoritmo de SA se describe a continuación:

Se necesita: $factor\_enfriamiento$, $T_{inicial}$, T_{final}, p , K_1, K_2

Devuelve: C_{mejor}

$T = T_{inicial}$

$C = C_{mejor} = individuo aleatorio$

Mientras $T > T_{final}$

$C_1 = macromutacion(C, p, T, K_1, K_2)$

$delta = error(C_1) - error(C)$

$V =$ un valor aleatorio con distribución uniforme $U(0,1)$

Si $(delta < 0)$ OR $\left(V < \exp^{-\left(delta / T\right)}\right)$ entonces

$C = C_1$

Si $(C < C_{mejor})$ entonces $C_{mejor} = C$

$T = T * factor\_enfriamiento$

La operación de macromutación adaptada a SA en el algoritmo descrito anteriormente es derivada del cruzamiento usado por el algoritmo Fuzzy-GAP, el cual une dos tipos de cruzamiento aleatoriamente, el usual de intercambio de subárboles y el cruzamiento entre cadenas, ambos mencionados anteriormente. El algoritmo Fuzzy-GAP fue también evaluado y su objetivo es

clasificar un ejemplo mediante la combinación cuadrática de sus atributos. Los pesos de dicha combinación son ajustados como un discriminante cuadrático. Un ejemplo es clasificado con la clase que tenga el mejor valor para la combinación cuadrática de sus atributos. Es importante señalar que las reglas que extrae son tipo Mamdani.

2.2.7 Algoritmos boosting

Los algoritmos *boosting* son técnicas de modelación estadística que combinan clasificadores de baja calidad, también denominados “hipótesis débiles”, para obtener un clasificador general que se desempeña mejor que cualquiera de sus componentes. Una regla difusa puede ser considerada un caso particular de una hipótesis débil y una base de reglas difusas puede ser comparada con una combinación de estas [28]. Existen tres variantes bastante conocidas de estos algoritmos Fuzzy-AdaBoost, Fuzzy-LogitBoost y Fuzzy-MaxLogitBoost que han sido usados para aprender conjuntos de reglas difusas en problemas de clasificación.

La forma de las reglas usadas por los tres es DNF extendida:

R_j : **IF** $x\ es\ A^j$ **THEN grados de certeza** $(s_1{}^j \ldots s_p{}^j)$

Donde x es el vector de atributos $x = (x_1,...,x_n) \in X$, siendo X el espacio de atributos, A^j es el producto cartesiano de los conjuntos difusos definidos para cada atributo $A^j = A_1{}^j \times A_2{}^j \times ... \times A_n{}^j$, p es el número de clases, j representa la cantidad de reglas y se encuentra definida en el rango $1 \leq j \leq N$.

Fuzzy-AdaBoost [28] consiste en invocar repetidamente un algoritmo de aprendizaje para generar sucesivamente un conjunto simple de reglas difusas de baja calidad. Cada una recibe una cierta cantidad de votos en dependencia de su precisión en la clasificación sobre el conjunto de entrenamiento. Cada vez que un nueva regla es generada y adicionada al conjunto se determinan nuevamente los pesos de los ejemplos de entrenamiento (de esta forma las futuras reglas se enfocarán en los ejemplos más difíciles) y se asignan los

votos correspondientes a la regla. El algoritmo (para dos clases) se puede definir como sigue:

Se proporcionan el conjunto de entrenamiento $(x_i, y_i),...,(x_m, y_m)$, con m ejemplos clasificados, donde $x_i \in X$, $1 \le y_i \le p$, $1 \le i \le m$, $y_i \in (-1,1)$

Número de reglas difusas $H \in (1,...,N)$

Se definen las variables locales:

$w \in R^m$ Pesos de los ejemplos de entrenamiento.

$\alpha \in R^N$ Votos de las hipótesis débiles.

$c \in (-1,1)^m$ Consecuentes de las hipótesis débiles.

$s : A \times (-1,1) \to (0,1)$ Relaciones difusas = consecuentes de las reglas.

Paso1: inicializar $w_i \leftarrow 1/m$, $\alpha^j = 0, s_k{}^j = 0$ con $1 \le k \le p$

Paso2: repetir el siguiente procedimiento H veces.

Paso1: proceso de generación de una nueva regla difusa mediante un Algoritmo Genético, se crea la regla difusa de la forma **IF** $x \; es \; A^h$ **THEN** c^h con $A^h \in A$ que minimiza el valor de adaptación, el cual se puede determinar siguiendo varios criterios como por ejemplo:

Fitness (IF $x \; es \; A^h$ **THEN** c^h **)** $= \sum_i w_i \exp(-y_i c^j A^h(x_i))$

Paso2: calcular el número de votos de la regla generada en el paso anterior *votos* (**IF** $x \; es \; A^h$ **THEN** c^h) = α^h. Este valor se puede determinar de varias formas, una de las cuales es la especificada en [25]:

$$\alpha_j = \log\left(\frac{1-E_j}{Ej}\right)$$

Donde E_j es el error de clasificación y se determina:

$$E_j = \frac{\sum_{i:y_i \neq c_j} w_i \max(\varepsilon, A^j(x_i))}{\sum_i w_i \max(\varepsilon, A^j(x_i))}$$

ε : parámetro pequeño que se halla por adelantado.

Paso3: actualizar los pesos de los ejemplos. La implementación original de AdaBoost actualiza solo los pesos de los ejemplos que son clasificados correctamente. El peso de una ejemplo (x_i, y_i) es multiplicado por el factor β_i si la *j*-regla clasifica correctamente el ejemplo de lo contrario no se modifica.

$$w_i \leftarrow \begin{cases} w_i \rightarrow y_i \neq c^j \\ \beta_i w_i \rightarrow y_i = c^j \end{cases} \qquad \beta_i = \left(\frac{E_j}{1-E_j}\right)^{A^j(x_i)}$$

$A^j(x_i)$ es el grado de clasificación entre la regla difusa y el ejemplo de entrenamiento.

No obstante la mayoría de las implementaciones de este algoritmo actualizan simultáneamente los pesos de todos los ejemplos. Los que son correctamente clasificados obtienen menores pesos que aquellos que son mal clasificados.

Paso3: convertir los votos en valores de certeza.

Para toda regla j,

Si ($\alpha^j > 0$ AND $c^j = 0$) OR ($\alpha^j < 0$ AND $c^j = -1$), $s^j{}_1 = \left|\alpha^j\right| \Big/ \max\left|\alpha^j\right|$

Sino $s^j{}_{-1} = \left|\alpha^j\right| \Big/ \max\left|\alpha^j\right|$

Paso4: salida del algoritmo.

Para toda regla j,

Si ($s^j{}_1 \neq 0$ ó $s^j{}_{-1} \neq 0$) se emite la regla:

IF x_1 **es** $A_1{}^j$ **AND…** x_n **es** $A_n{}^j$ **THEN (**$s^j{}_1, s^j{}_{-1}$**)**

En el paso de generación de las reglas difusas, se usan varios esquemas de codificación en dependencia del tipo de reglas, generalmente para las descriptivas se usa una cadena de n enteros, representando las etiquetas de los términos lingüísticos.

Los algoritmos Fuzzy-LogitBoost y Fuzzy-MaxlogitBoost son bastante similares. Usan un modelo aditivo extendido que invoca repetidamente un algoritmo de aprendizaje para generar sucesivamente un conjunto de hipótesis débiles, las cuales son reglas difusas extraídas de los datos mediante un Algoritmo Genético. En cada iteración una nueva regla es adicionada a la base, mientras que a los ejemplos en el conjunto de entrenamiento se les asigna nuevos pesos. La diferencia entre ambos radica en la inferencia difusa. Los métodos de razonamiento difuso definen como las reglas son combinadas e infieren dada una entrada la correspondiente salida. Al ejemplo x se le asigna la clase:

$$\arg\max_{k=1,\cdots,p} \bigvee_j^N A^j(x) \wedge s_k{}^j$$

En ella "$\wedge$" y "$\vee$" pueden ser implementados por varios operadores, por ejemplo "$\vee$" puede ser el operador Max o la suma aritmética y "$\wedge$" es usualmente el operador Min o el producto. En el LogitBoost se combina el producto con la suma aritmética para llevar a cabo la inferencia difusa. Por lo que la expresión anterior se puede escribir de la siguiente forma:

$$\arg\max_{k=1,\cdots,p} \sum_j A^j(x).s_k{}^j$$

Sin embargo para el caso de MaxLogitBoost la inferencia es hecha combinando los votos con el operador Max en lugar de la suma quedando la expresión de la forma:

$$\arg\max_{k=1,\cdots,p}\left(\max_{j} A^{j}(x).s_{k}{}^{j}\right)$$

La cual no es una suma de términos y por tanto no es un modelo aditivo, pero se define la función:

$$I(x,j)=\begin{cases}1\rightarrow si\,la\,regla\;j\,clasifica\,correctamente\,al\,ejemplo\,x\\0\rightarrow en\,caso\,contrario\end{cases}$$

Y la expresión anterior se puede reescribir:

$$\arg\max_{k=1,\cdots,p}\sum_{j} I(x,j).A^{j}(x).s_{k}{}^{j}$$

Entonces la expresión $r(x,\gamma_j)=I(x,j).A^{j}(x)$ es considerada como una hipótesis débil con grado de certeza $s_k{}^{j}$, con $1\le k\le p$, se considera x como un conjunto de ejemplos y γ_j identifica los términos lingüísticos de la regla.

La codificación usada por ambos algoritmos es la binaria, por ejemplo: si se tiene el conjunto de términos lingüísticos (*bajo, medio, alto*) representando tres variables del problema, el antedecente **IF** x_1 **es *alto*** **AND** x_2 **es *medio*** **AND** x_3 **es *bajo*** es codificado con la cadena 001 010 100. [42]

El algoritmo Fuzzy-LogitBoost [36] se puede definir como sigue:

Repetir $j=1,...,N$

Para las clases desde $k=1,...,p$

Para $i=1,...,n$ determinar $p_{ijk}=e^{f_{ij-1k}}\Big/\left(1+e^{f_{ij-1k}}\right)$

Para $i=1,...,n$ determinar $w_{ijk}=p_{ijk}\left(1-p_{ijk}\right)$

f representa a los atributos y se define en el rango $1\le f\le n$

Determinar mediante un Algoritmo Genético el antecedente A^j que minimiza:

$$fitness(A^j) = \sum_i^n w_{ijk}\left(s^j . A^j(x_i) - \frac{y_{ik} - p_{ijk}}{w_{ijk}} \right)^2$$

Donde

$$s^j = \frac{\sum_i (y_{ik} - p_{ijk}) A^j(x_i)}{\sum_i w_{ijk}[A^j(x_i)]^2}$$

Para $i = 1,...,n$ determinar $f_{ijk} = f_{ij-1k} + s^j . A^j(x_i)$

Si $s^j > 0$ emitir la regla **IF** x es A^j **THEN clase =** c_k **con grado de certeza** s^j

Sino emitir la regla **IF** x es A^j **THEN (** $s_1{}^j \ldots s_k{}^j \ldots s_p{}^j$ **)**

Mientras que el Fuzzy-MaxLogitBoost [42] consiste en los siguientes pasos:

Repetir $j = 1,...,N$

Paso1: Se lleva a cabo un procedimiento para agregar una nueva regla difusa a la base tomando como entrada es una base de reglas de tamaño N y el antecedente de la regla difusa A^{N+1} y proporciona como salida es una base de reglas de tamaño $N+1$ y los valores de adaptación $S_{kj} = s_k{}^j$ con $k = 1,...,p$, $j = 1,...,N$ usando un Algoritmo Genético.

Paso2: Seleccionar la base de reglas de tamaño j con el menor valor de adaptación.

Repetir $j = 1,...,N$

Paso1: Hacer cero todos los valores de adaptación $s_k{}^j = 0$ excepto el máximo, $s_{p(j)}{}^j$

Paso2: Emitir la regla **IF** x es A^j **THEN (** $s_1{}^j \ldots s_{p(j)}{}^j \ldots s_p{}^j$ **)**

2.2.8 MR-GIM

A partir de SLAVE se desarrolló una variante conocida como NSLV la que sirvió de base para la definición de MR-GIM (Multiple Rules using the Genetic Iterative Model). Este algoritmo hereda la mayor parte de la funcionalidad del algoritmo SLAVE aunque extendiendo su modelo genético iterativo para aprender una regla completa en cada iteración. SLAVE fija la clase a aprender y utiliza el Algoritmo Genético para encontrar el mejor antecedente para dicha clase. Este hecho crea una dependencia entre el conocimiento aprendido y el orden en el que se aprenden las clases. La nueva extensión da lugar a un nuevo método que reformula el Algoritmo Genético para eliminar la dependencia y extraer la mejor regla posible en cada iteración sin tener que fijar la clase.

Una de las modificaciones fundamentales realizadas al Algoritmo Genético de SLAVE es que las poblaciones cambian de generacionales a estacionarias, por lo que casi la totalidad de los individuos se trasladan intactos de una generación a la siguiente. Esto ocasiona un problema adicional, y está relacionado con la diversidad necesaria para que los individuos no converjan a un óptimo local. Para evitar esta situación, se incluyó el concepto de subpoblación, en el que se mantuvieran los mejores individuos de cada clase, evitando así la pérdida de los buenos individuos encontrados hasta el momento.

MR-GIM surge a partir de esta modificación realizada a SLAVE, el hecho de mantener los mejores individuos por clase en cada población, hace posible que se proceda a valorar la calidad de cada individuo, es decir, de los mejores individuos por clase, en cada subpoblación, con vistas a no solo seleccionar uno, como hace SLAVE, sino a extraer todos aquellos que han alcanzado la madurez suficiente y puedan ser seleccionados. Esto provoca una mejora en tiempo, debido a que se aprenden más individuos a partir de una única población, de lo contrario, era necesario un mayor número de iteraciones para lograr aprender un modelo completo.

El proceso para seleccionar los mejores individuos de la población se detalla en los siguientes pasos:

Mientras el individuo seleccionado mejore la precisión del algoritmo

Paso1: eliminar los ejemplos cubiertos positivamente por el individuo seleccionado.

Paso2: recalcular la función de adaptación para cada individuo.

Paso3: ordenar la población de acuerdo a la función de adaptación.

Paso4: seleccionar el mejor individuo.

Como puede apreciarse, este proceso selecciona el primer individuo de la población, que es el mejor global. Luego elimina los ejemplos que ya cubre, y actualiza la función de adaptación del resto, con vistas a estimar cuál individuo cubre mejor el espacio de ejemplos no cubiertos. Este individuo es seleccionado y así se repite el proceso hasta que no queden individuos por procesar o el mejor detectado no mejore la precisión.

Cada uno de los Sistemas Basados en Reglas Difusos Genéticos descritos arriba fue probado con diferentes bases de datos usando la novedosa herramienta para la Minería de Datos, KEEL. Los resultados obtenidos se analizarán en el siguiente capítulo haciendo uso de varios tests estadísticos.

Capítulo III. Estudio Experimental

En este Capítulo se muestra el análisis comparativo de los resultados experimentales obtenidos por los Sistemas Basados en Reglas Difusos Genéticos descritos anteriormente. Se presentan las bases de datos usadas en los experimentos, los tests estadísticos utilizados para comparar los sistemas y finalmente los resultados del estudio realizado.

3.1 Bases de Datos

Para realizar los experimentos con los Sistemas Basados en Reglas Difusos Genéticos se usaron diecisiete bases de datos del UCI Repository of Machine Learning Databases and Domain Theories. En la siguiente tabla se resumen sus características principales. Se muestran la cantidad de ejemplos (Ex), de atributos (Atrib), de clases (Clas) y el número de atributos cuyo dominio es nominal.

Bases de datos	Ex	Atrib	Clas	Nom
Automobile (aut)	205	25	6	10
Bupa (bpa)	345	6	2	0
Contraceptive Method Choice (cmc)	1473	9	3	7
Horse Colic (col)	368	22	2	15
Glass (gls)	214	9	6	0
Heart-C (h-c)	303	13	2	7
Heart-S (h-s)	270	13	2	7
Iris (irs)	150	4	3	0
Pima (pim)	768	8	2	1
Sonar (son)	208	60	2	0
Tao (tao)	888	2	2	0
Thyroid (thy)	215	5	5	0
Vehicle (veh)	846	18	3	0
Wisconsin Breast Cancer (wbcd)	699	9	4	9
Wisconsin Diagnostic Breast Cancer (wdbc)	569	30	2	0
Wine (wne)	178	13	3	0
Wisconsin Prognostic Breast Cancer (wpbc)	198	33	2	0

Tabla 3.1. Descripción de las bases de datos

3.2 Tests estadísticos

Existen, principalmente, dos grupos de tests estadísticos: los paramétricos y los no paramétricos. Los primeros se utilizan para identificar si la diferencia entre

dos algoritmos es estadísticamente significativa, por lo general son correctos y seguros, y más sensibles (perceptivos) que los no paramétricos; pero cuando los datos no siguen una distribución conocida o no se puede suponer que siguen una distribución determinada estos tests no son confiables y es necesario usar tests no paramétricos, los cuales utilizan una representación ordinal de los valores basada en el orden de los algoritmos para cada uno de los problemas. [7]

Los tests que se utilizaron para la comparación de los Sistemas Basados en Reglas Difusos Genéticos o algoritmos incluidos en la presente investigación son no paramétricos, pues los resultados obtenidos de la experimentación no cumplen con las premisas para poder llevar a cabo correctamente las comparaciones paramétricas. Friedman [43] fue usado para determinar si existe diferencia significativa entre al menos dos algoritmos y en los casos positivos, se empleó Bonferroni-dunn [43] para precisar cuáles son los algoritmos cuya diferencia es significativa. Ambos son de los más usados para comparar algoritmos de aprendizaje.

- Test de Firedman: ordena el valor medio obtenido de los algoritmos con cada base de datos (de mayor a menor), asignándosele un valor (ranking) R_j, para el algoritmo j, $R_j \in (1,\ldots,k)$, donde k es el número total de algoritmos. En el Anexo IV se muestran los rankings de los algoritmos con cada base de datos para cada uno de los criterios de comparación. Luego se suman los rankings obtenidos por cada algoritmo $\sum R_j$. La distribución usada es la chi-cuadrado χ^2 y su valor aproximado para el test de Friedman se determina según la ecuación:

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum R_j^2 - \frac{k(k+1)^2}{4}\right]$$

Donde N representa la cantidad de bases de datos.

Seguidamente se determina el valor crítico de la distribución chi-cuadrado χ^2 para un error $\alpha = 0.05$ con $k-1$ grados de libertad.

Habrá diferencia significativa entre al menos dos algoritmos si el valor aproximado para Friedman χ_F^2 es mayor o igual al valor crítico de la distribución chi-cuadrado χ^2.

- Test de Bonferroni-Dunn: consiste en determinar el valor denominado diferencia crítica (DC) según la expresión:

$$DC = q_\alpha \sqrt{\frac{k \cdot (k+1)}{6N}}$$

Donde k y N representan la cantidad de algoritmos y de bases de datos respectivamente.

Dos algoritmos son significativamente diferentes si el valor de la diferencia entre sus valores medios es mayor o igual a la DC.

3.3 Resultados obtenidos de la experimentación.

El estudio en cuestión se realizó con diecisiete bases de datos y once algoritmos o Sistemas Basados en Reglas Difusos Genéticos. Usualmente el principal objetivo de estos sistemas es maximizar su desempeño o precisión, no obstante también es muy importante la comprensibilidad o interpretabilidad del modelo difuso aprendido. Esta última es una propiedad sujetiva que depende de varios factores como: el número de reglas, la cantidad de variables por regla, la cantidad de condiciones y otras [25]. Los criterios mencionados anteriormente fueron considerados para realizar la comparación, tomándose para medir el desempeño el porcentaje de clasificación de cada algoritmo tanto con los datos de entrenamiento como de prueba. Los resultados obtenidos por los algoritmos con las bases de datos para cada criterio de comparación se muestran en el Anexo I.

Se empleó validación cruzada con 10 particiones, una de las más usadas en la realización de experimentos de este tipo. Se dividió cada base de datos en 10 partes y luego se tomaron 9 de estas para entrenamiento y la restante para prueba. El procedimiento se repitió diez veces, quedando siempre una partición

diferente como conjunto de prueba. Es decir, se aprende el modelo con el 90% de los datos y se prueba su calidad con el 10% restante. También fue necesario trasformar las bases de datos al formato que utiliza la herramienta KEEL, para lo cual se realizó un pequeño programa con el objetivo de agilizar el proceso.

Para la ejecución de Fuzzy-Ishib-Hybrid fue necesario procesar algunas bases datos para eliminar los valores ausentes debido a que este algoritmo no los acepta. Se usó para ello el algoritmo MV-Most Common incluido también en KEEL, que para cada instancia en la base de datos evalúa si es un valor ausente y en caso que exista alguno, se sustituye por el valor de la media/moda del atributo. Además es importante mencionar que KEEL permite establecer los valores deseados para cada uno de los parámetros de los algoritmos, pero en este caso se decidió mantener los valores por defecto que trae la herramienta.

Entre las precondiciones de Fuzzy-MOGUL no hay ningún requisito referido a las bases de datos con valores ausentes, sin embargo en sus resultados se puede apreciar que el por ciento de clasificación con las bases de datos que tienen valores ausentes es bastante bajo, llegando a cero en el caso de la base de datos col.

Los valores obtenidos para los tests estadísticos fueron: χ^2 = 18.30, en el test de Bonferroni-Dunn DC = 3.19, para el desempeño en el conjunto de entrenamiento y en el de prueba χ_F^2 = 41.43 y χ_F^2 = 21.98 respectivamente. En el caso de la cantidad de reglas χ_F^2 = 146.8, con respecto a la cantidad de condiciones χ_F^2 = 139.85 y para la cantidad de variables por regla χ_F^2 = 136.03. Según los resultados del test de Friedman para todas las medidas de comparación existen diferencias significativas entre al menos dos algoritmos.

A continuación se muestran las tablas, con los resultados de la diferencia entre los valores medios de los algoritmos, para cada criterio de comparación. Las celdas amarillas indican que existen diferencias significativas, según el test de Bonferroni-dunn. Es decir que el valor de la diferencia de los valores medios es

mayor que la DC = 3.19. Los valores negativos indican que el valor medio del algoritmo de la columna es menor que el de la fila. La tabla de Training indica la precisión de los algoritmos con los datos de entrenamiento. La de Test, que es más importante, permite medir la calidad en cuanto a la precisión del modelo aprendido. Las restantes tablas ayudan a evaluar la interpretabilidad de los modelos aprendidos. En los epígrafes que siguen se hace un análisis detallado de los resultados que muestra cada tabla.

Training	AdaBoost	GAP	Ishibuchi99	LogitBoost	SAP	MaxLogitBoost	GP	SLAVE	MOGUL	Ishib-Hybrid	MR-GIM
AdaBoost											
GAP	-5,85										
Ishibuchi99	-6,82	-0,96									
LogitBoost	-12,32	-6,47	-5,50								
SAP	-5,19	0,66	1,63	7,13							
MaxLogitBoost	-4,28	1,57	2,53	8,04	0,91						
GP	-5,16	0,69	1,65	7,16	0,03	-0,88					
SLAVE	-8,87	-3,02	-2,05	3,45	-3,68	-4,59	-3,71				
MOGUL	2,66	8,51	9,48	14,98	7,85	6,94	7,82	11,53			
Ishib-Hybrid	-4,72	1,13	2,10	7,60	0,47	-0,44	0,44	4,15	-7,38		
MR-GIM	-14,46	-8,61	-7,64	-2,14	-9,27	-10,18	-9,30	-5,59	-17,12	-9,74	

Tabla 3.2. Diferencia entre los valores medios obtenidos por los algoritmos con los datos de entrenamiento

Test	AdaBoost	GAP	Ishibuchi99	LogitBoost	SAP	MaxLogitBoost	GP	SLAVE	MOGUL	Ishib-Hybrid	MR-GIM
AdaBoost											
GAP	-5,39										
Ishibuchi99	-6,50	-1,11									
LogitBoost	-8,15	-2,76	-1,65								
SAP	-5,70	-0,32	0,79	2,44							
MaxLogitBoost	-3,57	1,82	2,93	4,58	2,14						
GP	-4,90	0,49	1,60	3,25	0,80	-1,33					
SLAVE	-5,37	0,01	1,12	2,77	0,33	-1,81	-0,47				
MOGUL	16,91	22,30	23,40	25,06	22,61	20,48	21,81	22,28			
Ishib-Hybrid	-3,06	2,33	3,44	5,09	2,65	0,51	1,84	2,32	-19,97		
MR-GIM	-9,90	-4,51	-3,40	-1,75	-4,19	-6,33	-5,00	-4,52	-26,80	-6,84	

Tabla 3.3. Diferencia entre los valores medios obtenidos por los algoritmos con los datos de prueba

Reglas	AdaBoost	GAP	Ishibuchi99	LogitBoost	SAP	MaxLogitBoost	GP	SLAVE	MOGUL	Ishib-Hybrid	MR-GIM
AdaBoost											
GAP	5,18										
Ishibuchi99	-44,95	-50,13									
LogitBoost	-17,00	-22,18	27,95								
SAP	5,18	0,00	50,13	22,18							
MaxLogitBoost	0,58	-4,60	45,53	17,58	-4,60						
GP	5,18	0,00	50,13	22,18	0,00	4,60					
SLAVE	-0,16	-5,34	44,79	16,84	-5,34	-0,74	-5,34				
MOGUL	-204,52	-209,70	-159,57	-187,52	-209,70	-205,10	-209,70	-204,36			
Ishib-Hybrid	-3,18	-8,35	41,78	13,82	-8,35	-3,75	-8,35	-3,02	201,35		
MR-GIM	-20,04	-25,22	24,91	-3,04	-25,22	-20,62	-25,22	-19,88	184,48	-16,87	

Tabla 3.4. Diferencia entre los valores medios de los algoritmos en cuanto a la cantidad de reglas extraídas

VariablesR	AdaBoost	GAP	Ishibuchi99	LogitBoost	SAP	MaxLogitBoost	GP	SLAVE	MOGUL	Ishib-Hybrid	MR-GIM
AdaBoost											
GAP	-77,56										
Ishibuchi99	8,69	86,25									
LogitBoost	0,23	77,78	-8,46								
SAP	7,27	84,83	-1,42	7,04							
MaxLogitBoost	-0,84	76,71	-9,53	-1,07	-8,11						
GP	5,71	83,27	-2,98	5,48	-1,56	6,55					
SLAVE	7,21	84,77	-1,48	6,99	-0,06	8,06	1,50				
MOGUL	-5,50	72,06	-14,19	-5,72	-12,77	-4,65	-11,21	-12,71			
Ishib-Hybrid	1,88	79,44	-6,81	1,66	-5,39	2,73	-3,83	-5,33	7,38		
MR-GIM	-37,51	40,05	-46,20	-37,74	-44,78	-36,67	-43,22	-44,72	-32,01	-39,39	

Tabla 3.5. Diferencia entre los valores medios obtenidos por los algoritmos en cuanto a las variables por regla

Condiciones	AdaBoost	GAP	Ishibuchi99	LogitBoost	SAP	MaxLogitBoost	GP	SLAVE	MOGUL	Ishib-Hybrid	MR-GIM
AdaBoost											
GAP	-118,92										
Ishibuchi99	-42,81	76,11									
LogitBoost	-179,89	-60,97	-137,08								
SAP	77,29	196,21	120,09	257,18							
MaxLogitBoost	14,55	133,46	57,35	194,44	-62,74						
GP	-442,04	-323,12	-399,24	-262,15	-519,33	-456,59					
SLAVE	51,84	170,76	94,65	231,73	-25,45	37,29	493,88				
MOGUL	-2899,41	-2780,49	-2856,60	-2719,52	-2976,69	-2913,95	-2457,36	-2951,25			
Ishib-Hybrid	4,62	123,54	47,43	184,51	-72,66	-9,92	446,66	-47,22	2904,03		
MR-GIM	58,90	177,82	101,70	238,79	-18,39	44,35	500,94	7,06	2958,30	54,28	

Tabla 3.6. Diferencia entre los valores medios obtenidos por los algoritmos en cuanto a cantidad de condiciones

3.3.1 Precisión de los algoritmos en el conjunto de datos de entrenamiento

Si un algoritmo tiene mejor desempeño o precisión significa que tiene un mayor por ciento de clasificación, lo cual es bueno. De acuerdo a los resultados que se muestran en la Tabla 3.2, Pág. 62 entre Fuzzy-GAP y Fuzzy-Ishibuchi99 no existen diferencias significativas, solamente son superados por MR-GIM y Fuzzy-LogitBoost y por otro lado tienen una mayor precisión que Fuzzy-AdaBoost y Fuzzy-MOGUL. Para con los restantes algoritmos las diferencias no son relevantes.

Los desempeños de Fuzzy-Ishibuchi-Hybrid, Fuzzy-SAP, Fuzzy-MaxLogitBoost y Fuzzy-GP son bastante similares, no existen diferencias significativas, pero con respecto a Fuzzy-MOGUL y Fuzzy-AdaBoost sus desempeños son mejores, mientras que ocurre lo opuesto con los algoritmos MR-GIM, Fuzzy-LogitBoost y Fuzzy-SLAVE. En cuanto a este último existen diferencias con Fuzzy-MOGUL y Fuzzy-AdaBoost también a su favor, sin embargo es superado por MR-GIM y Fuzzy-LogitBoost.

De forma general los mejores algoritmos son MR-GIM, Fuzzy-LogitBoost, entre ellos no existe diferencia significativa y superan a todos los demás. Por el contrario los peores son Fuzzy-MOGUL y Fuzzy-AdaBoost, cuyos desempeños también son similares, pero las diferencias con los restantes es en su detrimento. De acuerdo a los resultados obtenidos, en la siguiente tabla se muestran los algoritmos ordenados en cuanto a la calidad de este parámetro.

No	Algoritmos
1	MR-GIM y Fuzzy-LogitBoost
2	Fuzzy-SLAVE
3	Fuzzy-GAP y Fuzzy-Ishibuchi99
4	Fuzzy-Ishibuchi- Hybrid, Fuzzy-SAP, Fuzzy-MaxLogitBoost y Fuzzy-GP
5	Fuzzy-MOGUL y Fuzzy-AdaBoost

Tabla 3.7. Lista ordenada de los algoritmos de cuerdo a su desempeño en el conjunto de datos de entrenamiento

3.3.2 Precisión de los algoritmos en el conjunto de datos de prueba

En este caso, al igual que para el anterior, los mejores algoritmos son los que mayor precisión obtienen. En la Tabla 3.3, Pág. 62 se puede apreciar que el comportamiento de Fuzzy-AdaBoost en el conjunto de datos de prueba refleja que la diferencia con respecto a Fuzzy-Ishibuchi-Hybrid no es significativa y que logró superar a Fuzzy-MOGUL, lo cual no constituye una mejoría, al contrario su desempeño es inferior en comparación al resultado obtenido en el conjunto de entrenamiento. Además todos los restantes algoritmos lo superan. La precisión de Fuzzy-MOGUL es inferior a la del resto de los algoritmos, convirtiéndolo en el peor.

Fuzzy-LogitBoost disminuyó su desempeño igualmente, sigue siendo mejor que el de Fuzzy-AdaBoost, Fuzzy-MaxLogitBoost, Fuzzy-GP, Fuzzy-MOGUL, Fuzzy-Ishibuchi-Hybrid, pero con respecto a los demás no existe diferencia significativa. En el caso de Fuzzy-GAP su comportamiento es superior a Fuzzy-AdaBoost y Fuzzy-MOGUL, pero continúa por debajo del de MR-GIM. Con el resto de los algoritmos la diferencia carece de relevancia. Por otro lado entre Fuzzy-Ishibuchi99 y MR-GIM existen diferencias significativas a favor de este último, sin embargo con respecto a Fuzzy-AdaBoost, Fuzzy-MOGUL y Fuzzy-Ishibuchi-Hybrid la balanza se inclina hacia Fuzzy-Ishibuchi99. Con los restantes algoritmos la diferencia es despreciable.

En lo que respecta a Fuzzy-SAP existe diferencia significativa a su favor con Fuzzy-AdaBoost y Fuzzy-MOGUL, pero con MR-GIM es en su detrimento. Con respecto a Fuzzy-MaxLogitBoost y Fuzzy-GP su desempeño es bastante similar. Fuzzy-SLAVE, de manera general, disminuyó su precisión con respecto a la obtenida con los datos de entrenamiento, solamente logró mejorar a Fuzzy-MOGUL y Fuzzy-AdaBoost, aunque la diferencia con Fuzzy-LogitBoost, ahora, no resulta significativa, en lo que respecta a MR-GIM no mejoró su comportamiento.

Los resultados de Fuzzy-Ishibuchi-Hybrid no fueron halagadores, debido a que solo logró mejorar sus resultados con respecto a Fuzzy-MOGUL y por el

contrario, fueron superados por Fuzzy-LogitBoost, Fuzzy-Ishibuchi99 y MR-GIM. En cuanto a los restantes algoritmos la diferencia no es significativa.

Según lo antes expuesto se puede concluir que MR-GIM sigue siendo el mejor, supera a todos los demás. Por otro lado Fuzzy-MOGUL se quedó solo en el último lugar. De manera general la precisión de todos los algoritmos en el conjunto de prueba disminuyó con respecto a la alcanzada con los datos de entrenamiento, lo cual no es buen resultado y significa que los algoritmos han "sobreaprendido", es decir las reglas extraídas son muy específicas. La siguiente tabla muestra los algoritmos ordenados:

No	Algoritmos
1	MR-GIM
2	Fuzzy-LogitBoost
3	Fuzzy-Ishibuchi99
4	Fuzzy-SAP, Fuzzy-SLAVE y Fuzzy-GAP
5	Fuzzy-MaxLogitBoost y Fuzzy-GP
6	Fuzzy-Ishib- Hybrid
7	Fuzzy-AdaBoost
8	Fuzzy-MOGUL

Tabla 3.8. Lista ordenada de los algoritmos de cuerdo a su desempeño en el conjunto de datos de prueba

En el Anexo II se puede apreciar más claramente la diferencia entre los resultados obtenidos en training y test.

3.3.3 Cantidad de reglas extraídas por los algoritmos

La Tabla 3.4, Pág. 63 refleja que entre MR-GIM y Fuzzy-LogitBoost no existe diferencia significativa en cuanto a este parámetro, sin embargo con los restantes no ocurre lo mismo. En el caso de MR-GIM la cantidad de reglas extraídas es menor que las de Fuzzy-Ishibuchi99 y Fuzzy-MOGUL y mayor que la de los restantes algoritmos. Sin embargo las de Fuzzy-LogitBoost solo son superadas por las de Fuzzy-MOGUL. Es importante señalar que para este parámetro menor significa que se usan menos reglas para describir el modelo de aprendizaje y eso es bueno, siempre que no afecte la precisión del algoritmo.

Existe diferencia significativa entre Fuzzy-Ishibuchi99 y todos los algoritmos, siendo solamente superado por Fuzzy-MOGUL. En lo que respecta a Fuzzy-SAP, Fuzzy-GP y Fuzzy-GAP, los tres extraen la misma cantidad de reglas, que es menor que las del resto de los algoritmos.

La diferencia entre Fuzzy-AdaBoost, Fuzzy-SALVE, Fuzzy-MaxLogitBoost y Fuzzy-Ishib-Hybrid es irrelevante, excepto entre los dos últimos, siendo mayor la cantidad extraída por Fuzzy-Ishib-Hybrid y únicamente superan a Fuzzy-SAP, Fuzzy-GP y Fuzzy-GAP. De todos los algoritmos el que mayor cantidad de reglas extrae es Fuzzy-MOGUL. En la siguiente tabla se muestran los algoritmos ordenamos en cuanto a este parámetro de menor a mayor:

No	Algoritmos
1	Fuzzy-SAP, Fuzzy-GAP y Fuzzy-GP
2	Fuzzy-MaxLogitBoost, Fuzzy-AdaBoost, Fuzzy-SLAVE y Fuzzy-Ishib- Hybrid
3	Fuzzy-LogitBoost y MR-GIM
4	Fuzzy-Ishibuchi99
5	Fuzzy-MOGUL

Tabla 3.9. Lista ordenada de los algoritmos de cuerdo a la cantidad de reglas

3.3.4 Cantidad de variables por regla

Como se puede apreciar en la Tabla 3.5, Pág. 63 entre Fuzzy-AdaBoost, Fuzzy-LogitBoost, Fuzzy-MaxLogitBoost y Fuzzy- Ishib-Hybrid las diferencias, en cuanto a este parámetro no resultan significativas y superan a Fuzzy-Ishibuchi99, Fuzzy-SAP, Fuzzy-GP y Fuzzy-SLAVE, cuya cantidad de variables por regla es bastante similar, siendo los que menor cantidad poseen, todos los algoritmos restantes los superan.

Fuzzy-GAP es el que tiene mayor cantidad de variables por regla. Tanto Fuzzy-MOGUL como MR-GIM son únicamente superados por él. Con respecto a estos dos últimos la diferencia es a favor de MR-GIM. Tener menos variables por reglas, al igual que para el criterio de comparación anterior, significa que se usa menor cantidad de variables por reglas para representar el modelo de aprendizaje, lo cual también es bueno hasta cierto punto. En la tabla que se muestra a continuación se encuentran los algoritmos ordenados de menor a mayor de acuerdo a la cantidad variables por regla:

No	Algoritmos
1	Fuzzy-Ishibuchi99, Fuzzy-SAP, Fuzzy-SLAVE y Fuzzy-GP
2	Fuzzy-MaxLogitBoost, Fuzzy-AdaBoost, Fuzzy-Ishib- Hybrid y Fuzzy-LogitBoost
3	Fuzzy-MOGUL
4	MR-GIM
5	Fuzzy-GAP

Tabla 3.10. Lista ordenada de los algoritmos de cuerdo a la cantidad de variables por regla

3.3.5 Cantidad de condiciones de las reglas extraídas

En una regla se toman como condiciones los términos de la forma $X_i \; es \; A_n$ que se encuentran en el antecedente de la misma, si se usan muchas condiciones esto puede repercutir desfavorablemente en la comprensibilidad del modelo. En cuanto a este criterio de comparación la Tabla 3.6, Pág. 64 muestra que existen diferencias significativas entre todos los algoritmos. En el caso de Fuzzy-AdaBoost, su cantidad de condiciones es mayor que las de Fuzzy-SAP, Fuzzy-MaxLogitBoost, Fuzzy-SLAVE, Fuzzy-Ishib-Hybrid y MR-GIM. Por otro lado Fuzzy-GAP y Fuzzy-Ishibuchi99 son superados solamente por Fuzzy-MOGUL, Fuzzy-LogitBoost y Fuzzy-GP, aunque entre ambos el de mayor cantidad de condiciones es el primero.

Fuzzy-GP excede a todos, excepto a Fuzzy-MOGUL. En lo que respecta a Fuzzy- Ishib-Hybrid, la cantidad de condiciones es mayor que la de MR-GIM, Fuzzy-MaxLogitBoost, Fuzzy-SAP y Fuzzy-SLAVE. Este último es superado además por Fuzzy-MOGUL, pero con respecto a Fuzzy-SAP y MR-GIM no sucede lo mismo.

Fuzzy-LogitBoost y Fuzzy-Ishibuchi99 son superados por Fuzzy-GP y Fuzzy-MOGUL y en el caso de del segundo también se suma Fuzzy-GAP. En lo que respecta a ellos dos Fuzzy-LogitBoost posee mayor cantidad de condiciones. Por último Fuzzy-MaxLogitBoost únicamente supera a MR-GIM, Fuzzy-SAP y Fuzzy-SLAVE.

En general Fuzzy-MOGUL es el que posee mayor cantidad de condiciones, por el contrario de Fuzzy-SAP, que es superado por todos. MR-GIM ocupa el segundo lugar, superando solamente a Fuzzy-SAP. En la tabla que sigue están

los algoritmos ordenados de menor a mayor en cuanto a este criterio de comparación:

No	Algoritmos
1	Fuzzy-SAP
2	MR-GIM
3	Fuzzy-SLAVE
4	Fuzzy-MaxLogitBoost
5	Fuzzy-Ishib- Hybrid
6	Fuzzy-AdaBoost
7	Fuzzy-Ishibuchi99
8	Fuzzy-GAP
9	Fuzzy-LogitBoost
10	Fuzzy-GP
11	Fuzzy-MOGUL

Tabla 3.11. Lista ordenada de los algoritmos de cuerdo a la cantidad de condiciones

Usualmente la interpretabilidad del modelo aprendido por los algoritmos influye en su por ciento de clasificación y en este caso también se puede demostrar a partir de los resultados obtenidos en la experimentación. Sin duda alguna Fuzzy-MOGUL es el algoritmo con peor desempeño, tanto con los datos de entrenamiento como los de prueba y es el que mayor cantidad de reglas y condiciones posee. También con respecto a las variables por regla es uno de de los que mayor número tiene. Por otra parte, Fuzzy-GP, Fuzzy-GAP y Fuzzy-SAP, cuya cantidad de reglas es la menor, tienen una mayor precisión que Fuzzy-MOGUL. Es importante señalar que Fuzzy-SAP también tiene el mínimo número de condiciones y variables por reglas, además en cuanto a este último criterio de comparación Fuzzy-GAP no es superado por ninguno de los restantes algoritmos. Sin embargo MR-GIM, sin lugar a dudas, es el algoritmo con mejor desempeño aunque sea superado solamente por Fuzzy-GAP en cuanto a la cantidad de variables por reglas y su cantidad de condiciones únicamente exceda a la de Fuzzy-SAP, el cual posee la menor cantidad. Según lo antes expuesto se puede concluir que si bien los modelos con muchas reglas, variables por regla y condiciones influyen desfavorablemente en el desempeño de los algoritmos, los que tienen valores pequeños asociados a estos criterios tampoco arrojan resultados halagadores, pues el modelo resulta insuficiente para lograr una buena clasificación. En el Anexo III se puede apreciar gráficamente lo antes expuesto.

3.4 Valoración de Sostenibilidad

La Minería de Datos tiene una gran importancia, debido a que permite descubrir conocimiento valioso procesando grandes cantidades de información que el hombre de forma manual no podría por lo que se han desarrollado gran variedad de sus técnicas para la extracción automática de conocimiento, entre ellas los Sistemas Basados en Reglas Difusos Genéticos han sido objeto de gran atención. Los mismos incorporan tanto los beneficios de los SBRD como de los Algoritmos Genéticos, siendo de gran utilidad para la Minería Datos. Esta tiene un sin número de aplicaciones como por ejemplo: en el diagnóstico medico, las finanzas, sistemas de seguridad, telecomunicaciones, marketing, detección de fraude, experimentos científicos y otras, las cuales repercuten en el ámbito administrativo, socio-humanista, ambiental y tecnológico y la presente investigación no queda exenta de este fenómeno.

En el aspecto **administrativo** esta investigación no generan ingresos directos y su aporte económico se desprende de la utilidad que tienen los Sistemas Basados en Reglas Difusos Genéticos para la Minería de Datos, pues esta es ampliamente aplicada en la esfera comercial. El costo asociado a la experimentación realizada solo incluye gasto en electricidad porque en ocasiones, dependiendo sobre todo de la dimensión de la base de datos, algunos Sistemas Basados en Reglas Difusos Genéticos pueden demorar días para concluir su ejecución. La herramienta de Minería de Datos para el diseño de los experimentos es libre, con código abierto desarrollada en Java y las bases de datos utilizadas en el estudio se pueden encontrar y descargar libremente en Internet.

Desde el punto de vista **socio-humanista** es un estudio que permite ampliar los conocimientos de los interesados en Sistemas Basados en Reglas Difusos Genéticos y sirve de base para posteriores investigaciones, sobre todo para validar nuevas propuestas de los sistemas en cuestión. El impacto en el ámbito **ambiental**, no es significativo solamente se ocasionan gastos de energía eléctrica. El documento está redactado de forma clara, no se usan letras que atentan contra su legibilidad y los resultados experimentales fueron ilustrados mediante gráficas y tablas con el fin de propiciar un entendimiento fácil y

didáctico del contenido, evitando así causar estrés en el lector.

Un análisis desde el punto de vista **tecnológico** evidencia que este estudio se enmarca en un campo de investigación actual. Para su desarrollo desde el punto de vista de hardware no se necesitan grandes recursos, con respecto al software todo lo necesario es perfectamente accesible. Para consultar el documento solo se requiere una PC donde se pueda leer y una impresora en caso que se desee imprimir.

De acuerdo con el análisis realizado desde el punto de vista de las cuatro dimensiones se puede concluir que el estudio es sostenible, actual y de gran utilidad para posteriores propuestas de Sistemas Basados en Reglas Difusos Genéticos.

Conclusiones

Se realizó un amplio y detallado estudio de los Sistemas Basados en Reglas Difusos Genéticos, así como de la teoría que conforman sus fundamentos. Se describieron, de forma clara, algunos de los más conocidos y referenciados en la bibliografía. La mayoría de ellos están incluidos en KEEL, una herramienta para la Minería de Datos, bastante reciente y que posee una amplia librería de algoritmos de aprendizaje basados en Algoritmos Evolutivos.

Con cada sistema descrito se diseñó un experimento con diecisiete bases de datos, libremente accesibles, usando la propia herramienta KEEL, manteniéndose los valores de los parámetros de los algoritmos que trae esta por defecto. Fueron comparados de acuerdo a su desempeño en el conjunto de datos de entrenamiento y en el de prueba, así como las cantidades de reglas extraídas, de variables por reglas y de condiciones, usando dos tests estadísticos no paramétricos, el de Friedman y el de Bonferroni-Dunn, los cuales ayudaron a determinar los algoritmos cuya diferencia es estadísticamente significativa.

De forma general, con respecto a su desempeño, que es uno de los aspectos de comparación más importantes, los algoritmos MR-GIM y Fuzzy-LogitBoost son los mejores. Por el contrario Fuzzy-AdaBoost y Fuzzy-MOGUL tienen los peores porcentajes de clasificación. También, según los resultados obtenidos, se puede constatar que modelos de aprendizaje demasiado complejos o demasiado simples influyen desfavorablemente en la clasificación.

Recomendaciones

- Realizar los experimentos bajo las mismas condiciones para poder compararlos en cuanto al tiempo de aprendizaje.

- Incluir dentro de los criterios de comparación la cantidad de variables como otro parámetro para medir la simplicidad de los modelos de aprendizaje.

- Procesar las bases de datos con valores ausentes, con vistas a eliminarlos, para probar nuevamente el algoritmo Fuzzy-MOGUL y comprobar si mejoran sus resultados con dichas bases de datos.

- Experimentar con otros Sistemas Basados en Reglas Difusos Genéticos incluidos en la herramienta KEEL, para los cuales es necesario que las bases de datos con valores ausentes sean procesadas también.

Referencias bibliográficas

1. *Aprendizaje Automático y Minería de Datos*. 2006: Departamento de Informática Universidad Nacional de San Luis, Argentina.

2. Alcalá-Fdez, J., et al., *KEEL: A Software Tool to Assess Evolutionary Algorithms for Data Mining Problems.* 2005.

3. Back, T., D.B. Fogel, and Z. Michalewicz, *Evolutionary Computation 1 Basic Algorithms and Operators*. 2000: Institute of Physics Publishing.

4. Beasley, D., D.R. Bull, and R.R. Martin, *An Overview of Genetic Algorithms: Part 1, Fundamentals.* Inter-University Committee on Computing, 1993.

5. Béjar, J., *Inteligencia Artificial. Apuntes de Aprendizaje Automático*. 2006, Creative Common: http: //creativecommons.org/licenses/by-nc-sa/2.0.

6. Berlanga, F.J., M.J.d. Jesus, and F. Herrera, *Learning Fuzzy Rules using Genetic Programming: Context-free grammar definition for high-dimensionality problems* IEEE.

7. Cabrera, D.M., *Algoritmos Meméticos con Aplicación Adaptativa de la Búsqueda Local para Optimización Continua*, in *Departamento de Ciencias de la Computación e Inteligencia Artificial*. 2007, Universidad de Granada.

8. Casillas, J., P. Martínez, and A.D. Benítez, *Learning Consistent, Complete and Compact Sets of Fuzzy Rules in Conjunctive Normal Form for Regression Problems.* 2008.

9. Coello, C.A.C., *Curso de Computación Evolutiva*. 1998: Laboratorio Nacional de Informática Avanzada, Veracruz, México.

10. Cordón, O., *Genetic Fuzzy Systems: What's Next? An Introduction to the Special Section.* IEEE TRANSACTIONS ON FUZZY SYSTEMS, 2007.

11. Cordón, O., et al., *Ten years of genetic fuzzy systems: current framework and new trends.* www. ElsevierComputerScience.com, 2003.

12. Cordón, O., F. Herrera, and M.J.d. Jesus, *Genetic Learning of Fuzzy Rule-Based Classification Systems Cooperating with Fuzzy Reasoning Methods.* INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, 1998.

13. Cordón, O., F. Herrera, and M.J.d. Jesus, *A proponsal on reasoning methods in fuzzy rule-based classification systems.* International Journal of Aproximate Reasoning, 1999.

14. Cordón, O., et al., *MOGUL: A Mothodology to Obtain Genetic Fuzzy Rule Based-Systems under the Iterative Rule Learning Approach.* International Journal of Intelligent Systems, 1999(Department of Computer Science and Artificial Intelligence, University of Granada, Granada, España).

15. Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth, *From Data Mining to Knowledge Discovery in Databases.* American Association for Artificial Intelligence., 1996.

16. González, A. and R. Pérez, *SLAVE: A Genetic Learning System Based on an Iterative Approach.* IEEE, 1999.

17. González, A. and R. Pérez, *Selection of Relevant Features in a Fuzzy Genetic Learning Algorithm.* IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, 2001.

18. Gutiérrez, J.M., *Sistemas Expertos Basados en Reglas*: Dpto. de Matemática Aplicada. Universidad de Cantabria, España.

19. Han, J. and M. Kamber, *Data Mining: Concepts and Techniques* 2000: Morgan Koufman Publishers.

20. Haupt, R.L. and S.E. Haupt, *Practical Genetic Algorithms. Second Edition*. 2004: John Wiley & Sons.

21. Herrera, F., *Genetic Fuzzy Systems: Status, Critical Considerations and Future Directions.* International Journal of Computational Intelligence Research 2005.

22. Herrera, F., *Genetic Fuzzy Systems: Taxonomy, Current Research Trends and Prospects.* IEEE 2008.

23. Ishibuchi, H. and T. Nakashima, *Effect of Rule Weights in Fuzzy Rule-Based Classification Systems.* IEEE TRANSACTIONS ON FUZZY SYSTEMS, 2001.

24. Ishibuchi, H., T. Nakashima, and T. Murata, *Performance Evaluation of Fuzzy Classifier Systems for Multidimensional Pattern Classification Problems.* IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, 1999.

25. Ishibuchi, H. and T. Yamamoto, *Fuzzy Rule Selection by multiobjective genetic local search algorithms and rule evaluation measures in data mining.* www. ElsevierComputerScience.com, 2003.

26. Ishibuchi, H. and T. Yamamoto, *Rule Weight Specification in Fuzzy Rule-Based Classification Systems.* IEEE TRANSACTIONS ON FUZZY SYSTEMS, 2005.

27. Ishibuchi, H., T. Yamamoto, and T. Nakashima, *Hybridization of Fuzzy GBML Approaches for Pattern Classification Problems* IEEE TRansACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, 2005.

28. Jesus, M.J.d., et al., *Induction of Fuzzy-Rule-Based Classifiers With Evolutionary Boosting Algorithms.* IEEE TRANSACTIONS ON FUZZY SYSTEMS, 2004.

29. Jin, Y., *Fuzzy Modeling of High-Dimensional Systems: Complexity Reduction and Interpretability Improvement.* IEEE TRANSACTIONS ON FUZZY SYSTEMS, 2000.

30. Kantardzic, M., *Data Mining: Concepts, Models, Methods, and Algorithms*. 2003: John Wiley & Sons

31. Kasabov, N.K., *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering*. 1996: Massachusetts Institute of Technology.

32. Martínez-López, F.J. and J. Casillas, *Marketing Intelligent Systems for consumer behaviour modelling by a descriptive induction approach based on Genetic Fuzzy Systems.* Industrial Marketing Management, 2008.

33. Mitchell, M., *An Introduction to Genetic Algorithms*. 1999: A Bradford book. Massachusetts Institute of Technology Press.

34. Mitchell, T.M., *Machine Learning* 1997: McGraw-Hill Science Publicher.

35. Mitra, S. and T. Acharya, *Data Mining Multimedia, Soft Computing, and Bioinformatics*. 2003: John Wiley & Sons.

36. Otero, J. and L. Sánchez, *Induction of descriptive fuzzy classifiers with the Logitboost algorithm.* Published online: Springer-Verlag 2005.

37. Pérez, P.Y.P., *Algoritmos genéticos breve monografía*. 2004: Universidad de la Ciencias Informáticas, Grupo de Inteligencia Artificial, Sherpa.

38. Pérez, R.B., *Curso Introductorio a la Inteligencia Artificial*: Departamento de Ciencia de la Computación Universidad Central de Las Villas

39. Requena, I., A. Blanco, and M. Delgado, *A Constructive Method for Building Fuzzy Rule-Based Systems*: Dept. Computer Science and Artificial Intelligence University of Granada, España.

40. Russell, S. and P. Norvig, *Artificial Intelligence. A Modern Approach. Second Edition* 2003: Pearson Education.

41. Sánchez, L., I. Couso, and J.A. Corrales, *Combining GP operators with SA search to evolve fuzzy rule based classifiers.* www. ElsevierComputerScience.com, 2001(Dep. Informática, Universidad de Oviedo, España).

42. Sánchez, L. and J. Otero, *Boosting fuzzy rules in classification problems under single-winner inference.* Universidad de Oviedo. Depto. Informática. España, 2003.

43. Sheskin, *Handbook of Parametric and no-Parametric Estatistical Test*. 2004: Chapman & Hal.

44. Siler, W. and J.J. Buckley, *Fuzzy Expert Systems and Fuzzy Reasoning*. 2005: John Wiley & Sons.

45. Witten, I.H. and E. Frank, *Data Mining. Practical Machine Learning Tools and Techniques, Second Edition*. 2005: Morgan Koufmann Publishers.

46. Vallejos, S.J., *Minería de Datos*. 2006, Universidad Nacional del Nordeste, Argentina.

47. Zhang, C. and S. Zhang, *Association Rule Mining. Models and Algorithms*. 2002: Springer Publishing.

48. Wang, J., *Encyclopedia of Data Warehousing and Mining*. 2006: IDEA GROUP REFERENCE.

49. Tzafestas, S.G. and K.D. Blekas, *Hybrid Soft Computing Systems: A Critical Survey with Engineering Applications*: Department of Electrical and Computer Engineering.

50. Wong, M.L. and K.S. Leung, *Data Mining using grammar based genetic programming and applications*. 2000: Kluwer Academics Publishers.

Bibliografía

Aprendizaje Automático y Minería de Datos. 2006: Departamento de Informática, Universidad Nacional de San Luis, Argentina.

Alcalá-Fdez, J., et al., *KEEL: A Software Tool to Assess Evolutionary Algorithms for Data Mining Problems.* 2005.

Back, T., D.B. Fogel, and Z. Michalewicz, *Evolutionary Computation 1 BasicAlgorithms and Operators*. 2000: Institute of Physics Publishing.

Back, T., D.B. Fogel, and Z. Michalewicz, *Evolutionary Computation 2 Advanced Algorithms and Operators* 2000: Institute of Physics Publishing.

Beasley, D., D.R. Bull, and R.R. Martin, *An Overview of Genetic Algorithms: Part 1, Fundamentals.* Inter-University Committee on Computing, 1993.

Beasley, D., D.R. Bull, and R.R. Martin, *An Overview of Genetic Algorithms: Part 2, Research Topics.* Inter-University Committee on Computing, 1993.

Béjar, J., *Inteligencia Artificial. Apuntes de Aprendizaje Automático*. 2006, Creative Common: http: //creativecommons.org/licenses/by-nc-sa/2.0.

Benítez, J.M., et al., *Neural Methods for Obtaining Fuzzy Rules*: Dept. of Computer Science and Artificial Intelligence E.T.S. Computer Engineering, University of Granada; España.

Berlanga, F.J., M.J.d. Jesus, and F. Herrera, *Learning Fuzzy Rules using Genetic Programming: Context-free grammar definition for high-dimensionality problems* IEEE.

Cabrera, D.M., *Algoritmos Meméticos con Aplicación Adaptativa de la Búsqueda Local para Optimización Continua*, in *Departamento de Ciencias de la Computación e Inteligencia Artificial*. 2007, Universidad de Granada.

Casillas, J., B. Carse, and L. Bull, *Fuzzy-XCS: A Michigan Genetic Fuzzy System.* IEEE TRANSACTIONS ON FUZZY SYSTEMS, 2007.

Casillas, J., et al., *Accuracy Improvements to Find the Balance Interpretability-Accuracy in Linguistic Fuzzy Modeling: An Overview.* Springer Science, 2003.

Casillas, J. and F.J.M. López, *Mining uncertain data with multiobjective genetic fuzzy systems to be applied in consumer behaviour modelling.* www. sciencedirect.com, 2007.

Casillas, J., P. Martínez, and A.D. Benítez, *Learning Consistent, Complete and Compact Sets of Fuzzy Rules in Conjunctive Normal Form for Regression Problems.* 2008.

Coello, C.A.C., *Curso de Computación Evolutiva*. 1998: Laboratorio Nacional de Informática Avanzada, Veracruz, México.

Cordón, O., *Genetic Fuzzy Systems: What's Next? An Introduction to the Special Section.* IEEE TRANSACTIONS ON FUZZY SYSTEMS, 2007.

Cordón, O., et al., *Ten years of genetic fuzzy systems: current framework and new trends.* www. ElsevierComputerScience.com, 2003.

Cordón, O., F. Herrera, and M.J.d. Jesus, *Genetic Learning of Fuzzy Rule-Based Classification Systems Cooperating with Fuzzy Reasoning Methods.* INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, 1998.

Cordón, O., F. Herrera, and M.J.d. Jesus, *A proponsal on reasoning methods in fuzzy rule-based classification systems.* International Journal of Aproximate Reasoning, 1999.

Cordón, O., et al., *MOGUL: A Mothodology to Obtain Genetic Fuzzy Rule Based-Systems under the Iterative Rule Learning Approach.* International Journal of Intelligent Systems, 1999(Department of Computer Science and Artificial Intelligence, University of Granada, Granada, España).

Cox, E., *The Fuzzy Systems Handbook* 1994: Chapman and Hall Publishers.

Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth, *From Data Mining to Knowledge Discovery in Databases.* American Association for Artificial Intelligence., 1996.

González, A. and R. Pérez, *Un Algoritmo Iterativo para la Generación de Reglas Difusas.*

González, A. and R. Pérez, *SLAVE: A Genetic Learning System Based on an Iterative Approach.* IEEE, 1999.

González, A. and R. Pérez, *Selection of Relevant Features in a Fuzzy Genetic Learning Algorithm.* IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, 2001.

Gutiérrez, J.M., *Sistemas Expertos Basados en Reglas*: Dpto. de Matemática Aplicada. Universidad de Cantabria, España.

Gutiérrez, J.M., *Modelos de Redes Probabilísticas en Sistemas Expertos.* 1998.

Han, J. and M. Kamber, *Data Mining: Concepts and Techniques* 2000: Morgan Koufman Publishers.

Haupt, R.L. and S.E. Haupt, *Practical Genetic Algorithms. Second Edition*. 2004: John Wiley & Sons.

Hernández, A.G., *Aprendizaje Automático: Conceptos básicos.* 2004.

Herrera, F., *Genetic Fuzzy Systems: Status, Critical Considerations and Future Directions.* International Journal of Computational Intelligence Research 2005.

Herrera, F., *Genetic Fuzzy Systems: Taxonomy, Current Research Trends and Prospects.* IEEE 2008.

Hong, T.-P., C.-S. Kuo, and S.-C. Chi, *Mining association rules from quantitative data.* www. ElsevierComputerScience.com, 1999.

Ishibuchi, H., T. Murata, and I.B. Turksen, *Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems* Elsevier Science, 1997.

Ishibuchi, H. and T. Nakashima, *Effect of Rule Weights in Fuzzy Rule-Based Classification Systems.* IEEE TRANSACTIONS ON FUZZY SYSTEMS, 2001.

Ishibuchi, H., T. Nakashima, and T. Murata, *Performance Evaluation of Fuzzy Classifier Systems for Multidimensional Pattern Classification Problems.* IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, 1999.

Ishibuchi, H. and Y. Nojima, *Analysis of interpretability-accuracy trade-of fuzzy systems by multiobjective fuzzy genetics-based machine learning.* International Journal of Approximate Reasoning, 2006.

Ishibuchi, H., et al., *Selecting Fuzzy If-Then Rules for Classification Problems Using Genetic Algorithms* IEEE TRANSACTIONS ON FUZZY SYSTEMS, 1995.

Ishibuchi, H. and T. Yamamoto, *Fuzzy Rule Selection by multiobjective genetic local search algorithms and rule evaluation measures in data mining.* www.ElsevierComputerScience.com, 2003.

Ishibuchi, H. and T. Yamamoto, *Rule Weight Specification in Fuzzy Rule-Based Classification Systems.* IEEE TRANSACTIONS ON FUZZY SYSTEMS, 2005.

Ishibuchi, H., T. Yamamoto, and T. Nakashima, *Hybridization of Fuzzy GBML Approaches for Pattern Classification Problems* IEEE TRansACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, 2005.

Jesus, M.J.d., et al., *Induction of Fuzzy-Rule-Based Classifiers With Evolutionary Boosting Algorithms.* IEEE TRANSACTIONS ON FUZZY SYSTEMS, 2004.

Jin, Y., *Fuzzy Modeling of High-Dimensional Systems: Complexity Reduction and Interpretability Improvement.* IEEE TRANSACTIONS ON FUZZY SYSTEMS, 2000.

Kantardzic, M., *Data Mining: Concepts, Models, Methods, and Algorithms*. 2003: John Wiley & Sons

Kasabov, N.K., *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering*. 1996: Massachusetts Institute of Technology.

Kecman, V., *Learning and Soft Computing. Suport Vector Machines, Neural Networks, and Fuzzy Logic Models*. 2001: A Bradford book. Massachusetts Institute of Technology.

Li, T., C.-S. Perng, and S. Ma, *Guest editorial: special issue on temporal data mining: theory, algorithms and applications.* Springer Science, 2008.

Martínez-López, F.J. and J. Casillas, *Marketing Intelligent Systems for consumer behaviour modelling by a descriptive induction approach based on Genetic Fuzzy Systems.* Industrial Marketing Management, 2008.

Mikut, R., J. Jäkel, and L. Gröll, *Interpretability issues in data-based learning of fuzzy systems.* Elsevier Science, 2005.

Mitchell, M., *An Introduction to Genetic Algorithms*. 1999: A Bradford book. Massachusetts Institute of Technology Press.

Mitchell, T.M., *Machine Learning* 1997: McGraw-Hill Science Publicher.

Mitra, S. and T. Acharya, *Data Mining Multimedia, Soft Computing, and Bioinformatics*. 2003: John Wiley & Sons.

O'Reilly, U.-M., et al., *Genetic Programming Theory and Practice II*. 2005: Springer Science.

Otero, J. and L. Sánchez, *Induction of descriptive fuzzy classifiers with the Logitboost algorithm.* Published online: Springer-Verlag 2005.

Pérez, P.Y.P., *Algoritmos genéticos breve monografía*. 2004: Universidad de la Ciencias Informáticas, Grupo de Inteligencia Artificial, Sherpa.

Pérez, R.B., *Curso Introductorio a la Inteligencia Artificial*: Departamento de Ciencia de la Computación Universidad Central de Las Villas

Piatetsky-Shapiro, G., *Data mining and knowledge discovery 1996 to 2005: overcoming the hype and moving from "university" to "business" and "analytics".* Springer Science, 2007.

Puig, A.O., J. Casillas, and E.B. Mansilla, *Fuzzy-UCS: A Michigan-style Learning Fuzzy-Classifier System for Supervised Learning.* EEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.

Requena, I., A. Blanco, and M. Delgado, *A Constructive Method for Building Fuzzy Rule-Based Systems*: Dept. Computer Science and Artificial Intelligence University of Granada, España.

Russell, S. and P. Norvig, *Artificial Intelligence. A Modern Approach. Second Edition* 2003: Pearson Education.

Rutkowski, L., *Flexible Neuro-Fuzzy Systems. Structures, Learning and Performance Evaluation*. 2004: Kluwer Academic Publishers.

Sánchez, L., I. Couso, and J.A. Corrales, *Combining GP operators with SA search to evolve fuzzy rule based classifiers.* www. ElsevierComputerScience.com, 2001(Dep. Informática, Universidad de Oviedo, España).

Sánchez, L. and J. Otero, *Boosting fuzzy rules in classification problems under single-winner inference.* Universidad de Oviedo. Depto. Informática. España, 2003.

Sheskin, *Handbook of Parametric and no-Parametric Estatistical Test*. 2004: Chapman & Hal.

Siler, W. and J.J. Buckley, *Fuzzy Expert Systems and Fuzzy Reasoning*. 2005: John Wiley & Sons.

Thornton, C. and B. Boulay, *Artificial Intelligence Strategies, Applications, and Models Through Search Second Edition*. 1998: AMERICAN MANAGEMENT ASSOCIATION

Tzafestas, S.G. and K.D. Blekas, *Hybrid Soft Computing Systems: A Critical Survey with Engineering Applications*, : Department of Electrical and Computer Engineering.

Vallejos, S.J., *Minería de Datos*. 2006, Universidad Nacional del Nordeste, Argentina.

Wang, J., *Encyclopedia of Data Warehousing and Mining*. 2006: IDEA GROUP REFERENCE.

Wang, L. and X. Fu, *Data Mining with Computational Intelligence* 2005: Springer-Verlag.

Whitley, D., *A genetic algorithm tutorial* 1994: Chapman & Hall

William J. Raynor, J., *The International Dictionary of Artificial Intelligence*. 1999: Glenlake Publishing Company.

Witten, I.H. and E. Frank, *Data Mining. Practical Machine Learning Tools and Techniques, Second Edition*. 2005: Morgan Koufmann Publishers.

Wolkenhauer, O., *Data Engineering Fuzzy Mathematics in Systems Theory and Data Anabsis* 2001: John Wiley & Sons, .

Wong, M.L. and K.S. Leung, *Data Mining using grammar based genetic programming and applications*. 2000: Kluwer Academics Publishers.

Ye, N., *The Handbook of Data Mining*. 2003: Lawrence Erlbaum Associates Publishers.

Zhang, C. and S. Zhang, *Association Rule Mining. Models and Algorithms*. 2002: Springer Publishing.

Glosario de Términos

Conjuntos difusos: son una generalización de la teoría de conjuntos clásica, donde cada elemento x en el universo X tiene grado de pertenencia ente 0 y 1 en un conjunto A. Han sido propuestos para lidiar con la incertidumbre.

Data Warehouse: se refiere a una amplia colección centralizada de datos comparativos u otro tipo de datos organizacional, frecuentemente obtenidos a partir de varias fuentes.

Disjunctive Normal Form (DNF): función booleana definida por un conjunto de variables $X = (x_1, x_2 \ldots)$. Una función en DNF puede ser escrita como una disyunción de términos.

Función de pertenencia: una función $m(A, x)$, la cual retorna un valor de verdad de un conjunto u objeto. Típicamente usada en lógica difusa donde mide la pertenencia de un objeto a un conjunto.

Heurística: técnicas de aproximación para resolver problemas. Son generalmente usadas cuando la solución exacta a un problema específico es desconocida o cuando se conoce un método exacto pero es demasiado difícil o imposible de aplicar. Ofrece una solución bastante buena a un costo razonable.

Interpretabilidad: se refiere a la capacidad del modelo difuso de expresar el comportamiento del sistema de una forma entendible. En este término se pueden englobar diferentes criterios como completitud, consistencia o transparencia. De forma simple significa que los seres humanos sean capaces de entender el comportamiento del sistema difuso inspeccionando la base de reglas.

Métodos Bayesianos: proveen un método formal para razonar sobre eventos inciertos. Están basados en la teoría de la probabilidad y usan técnicas probabilísticas para valorar y propagar la incertidumbre.

Modelos aditivos: técnica de modelación que usa sumas lineales de posibles variables de entrada transformadas para predecir la variable de salida. Estos

modelos son usados en algunos sistemas de aprendizaje automático, como los algoritmos boosting.

Modelo: representación matemática, lógica o física, generalmente abstracta, de un proceso, objeto o concepto.

Nominal: característica cuyos valores mantienen el significado ante un reordenamiento de atributos o un cambio de valores que se le da un significado. Un ejemplo seria la variable sexo, que puede ser codificada como 0, masculino y 1, femenino.

Precisión: la precisión de un sistema de aprendizaje automático es medida como el porcentaje de predicciones correctas o clasificaciones hechas por un modelo sobre un conjunto de datos específico.

Términos lingüísticos: se emplea en equivalencia a **valores lingüísticos** y **etiquetas lingüísticas.**

Valores ausentes: están referidos al hecho de que existen bases de datos cuyos atributos tienen valores que son desconocidos.

Variables lingüísticas: son variables cuyos valores son términos lingüísticos asociados a un conjunto difuso. Un ejemplo de variable lingüística pudiera ser **Velocidad** con términos lingüísticos “Lento”, “Medio”, “Rápido”.

Anexos

Anexo I. Resultados obtenidos por los algoritmos con cada base de datos

Training											
Bases de datos	**AdaBoost**	**GAP**	**Ishibuchi99**	**LogitBoost**	**SAP**	**MaxLogitBoost**	**GP**	**SLAVE**	**MOGUL**	**Ishib-Hybrid**	**MR-GIM**
aut	15,39	51,81	70,13	53,49	50,89	31,11	55,67	82,66	45,04	49,00	94,3
bpa	71,62	71,27	60,09	78,51	68,59	73,62	61,58	65,44	82,577	73,43	66,7
cmc	53,1	49,75	45,01	58,39	50,15	49,07	51,15	25,96	64,49	45,08	54,1
col	66,36	85,96	70	78,71	85,97	71,08	86,14	70,65	0,00	76,58	89,6
gls	46,51	58,26	69,05	76,94	52,8	68,49	52,63	73,94	77,37	66,05	78,5
hc	80,74	78,25	87,45	95,45	79,35	84,38	84,31	80,42	51,09	76,82	88
hs	80,41	81,06	88,76	96,04	79,87	84,24	81,19	78,85	49,88	79,30	89
irs	96	91,4	95,62	98,66	96,96	98,3	92,07	96,37	99,33	99,11	96,4
pim	78,21	76,59	69,9	83,37	76,47	80,31	76,59	77,71	88,54	76,76	79,7
son	47,32	78,63	76,97	56,35	76,22	46,63	77,88	87,44	36,11	54,28	93,4
tao	82,31	78,44	81,49	87,07	86,95	86,75	82,37	82,09	93,76	91,21	82,6
thy	91,37	90,49	87,39	98,19	90,69	95,61	87,39	93,95	98,71	96,28	94,1
veh	53,53	56,06	64,45	70,38	49,88	61,16	53,23	75,64	81,03	51,64	83
wbcd	93,61	93,57	96,77	97,89	94,25	97,58	96,85	97,87	95,71	97,09	97,4
wdbc	69,02	85,51	92,59	96,01	80,02	93,24	94,86	93,83	90,75	91,47	96,5
wne	96,34	96,15	98,13	99,44	96,21	98,69	90,32	97,82	85,68	95,01	97,9
wpbc	93,63	91,76	77,55	100	88,45	68,01	79,01	85,63	30,16	76,60	80,1
Promedio	71,50	77,35	78,31	83,82	76,69	75,78	76,66	80,37	68,84	76,22	85,96
Desv	22,32	15,05	14,68	16,16	16,54	20,01	15,68	17,08	28,82	17,84	11,85

Tabla Anexol.1. Resultados obtenidos por cada algoritmo en el conjunto de entrenamiento de cada base de datos.

Test											
Bases de Datos	**AdaBoost**	**GAP**	**Ishibuchi99**	**LogitBoost**	**SAP**	**MaxLogitBoost**	**GP**	**SLAVE**	**MOGUL**	**Ishib-Hybrid**	**MR-GIM**
aut	12,23	41,47	57,07	47,28	46,8	22,88	51,66	61,48	28,04	38,50	69,6
bpa	64,06	60,82	58,25	70,16	63,49	62,59	55,32	60,53	59,98	63,77	58,4
cmc	51,39	48,33	43,92	53,9	49,01	46,17	47,65	24,31	37,90	43,79	42,2
col	66,64	83,68	71,84	70,58	83,97	71,47	85,86	63,02	0,00	66,21	83,9
gls	42,87	54,35	59,88	67,02	48,54	60,82	44,99	61,33	56,90	59,95	69,5
hc	74,88	73,49	77,87	76,24	76,88	73,27	78,17	72,24	25,38	69,98	77,8
hs	74,07	78,88	78,14	74,81	74,07	78,15	72,96	70	19,63	69,63	76,6
irs	96	90,66	95,33	95,33	95,33	95,33	86,66	94	94,67	96,00	94,6
pim	72,78	73,83	68,75	75,91	72,53	75	73,58	74,1	70,98	73,45	73,5
son	46,61	61,07	61,59	51,95	69,14	46,62	70,69	70,62	5,30	40,81	72,5
tao	82,15	76,15	81,19	87,02	86,91	86,49	82,82	80,93	91,90	90,62	81,5
thy	89,81	88,78	87,04	96,25	88,54	92,64	85,49	90,75	93,03	93,03	92,2
veh	50,36	52	61,47	64,41	50,57	58,55	48,68	66,3	63,03	50,82	68,5
wbcd	91,58	93,49	95,41	93,15	93,33	95,99	95,71	96	75,83	96,00	96,4
wdbc	50,5	81,89	92,56	72,79	71,17	91,42	93,15	91,56	76,01	90,00	93,5
wne	94,29	94,55	93,85	95,14	95,85	93,82	83,14	92,09	58,30	88,73	93,3
wpbc	89,86	88,23	76,35	96,63	80,91	59,5	76,83	72,17	5,78	70,78	74,3
Promedio	67,65	73,04	74,15	75,80	73,36	71,22	72,55	73,03	50,74	70,71	77,55
Desv	22,91	16,95	15,60	16,15	16,84	20,80	16,69	17,70	31,99	19,55	14,36

Tabla Anexol.2. Resultados obtenidos por los algoritmos en el conjunto de prueba de cada base de datos

Reglas											
Base de datos	**AdaBoost**	**GAP**	**Ishibuchi99**	**LogitBoost**	**SAP**	**MaxLogitBoost**	**GP**	**SLAVE**	**MOGUL**	**Ishib-Hybrid**	**MR-GIM**
aut	8,00	6,00	63,20	25,00	6,00	8,00	6,00	15,70	80,00	9,40	44,88
bpa	8,00	2,00	36,50	25,00	2,00	7,50	2,00	5,80	220,00	13,20	40,58
cmc	8,00	3,00	81,80	25,00	3,00	8,00	3,00	4,70	783,40	6,40	54,04
col	8,00	2,00	80,30	25,00	2,00	7,80	2,00	3,30	2,20	3,80	19,57
gls	8,00	6,00	50,50	25,00	6,00	8,00	6,00	13,80	103,80	9,50	40,19
hc	8,00	2,00	72,90	25,00	2,00	8,00	2,00	7,10	158,00	18,40	19,14
hs	8,00	2,00	67,00	25,00	2,00	8,00	2,00	8,40	153,00	14,90	21,11
irs	8,00	3,00	13,20	25,00	3,00	6,80	3,00	3,70	66,90	14,90	52,67
pim	8,00	2,00	70,50	25,00	2,00	8,00	2,00	9,60	448,20	11,90	28,39
son	8,00	2,00	54,00	25,00	2,00	1,00	2,00	8,80	100,60	3,60	39,42
tao	8,00	2,00	15,60	25,00	2,00	7,50	2,00	3,00	68,70	20,00	14,72
thy	8,00	3,00	29,10	25,00	3,00	7,90	3,00	5,70	102,40	12,20	12,56
veh	8,00	4,00	81,50	25,00	4,00	8,00	4,00	26,10	483,40	9,90	33,33
wbcd	8,00	2,00	45,80	25,00	2,00	7,80	2,00	6,30	375,30	11,30	4,72
wdbc	8,00	2,00	45,70	25,00	2,00	7,90	2,00	4,70	306,20	9,50	3,69
wne	8,00	3,00	61,50	25,00	3,00	8,00	3,00	3,90	86,60	9,50	12,36
wpbc	8,00	2,00	31,10	25,00	2,00	8,00	2,00	8,10	74,20	11,60	35,35
Promedio	8,00	2,82	52,95	25,00	2,82	7,42	2,82	8,16	212,52	11,18	28,04
Desv	0,00	1,33	22,19	0,00	1,33	1,68	1,33	5,82	204,91	4,41	15,97

Tabla Anexol.3. Cantidad de reglas extraídas por los algoritmos en cada base de datos

Variables por regla											
Base de datos	**AdaBoost**	**GAP**	**Ishibuchi99**	**LogitBoost**	**SAP**	**MaxLogitBoost**	**GP**	**SLAVE**	**MOGUL**	**Ishib-Hybrid**	**MR-GIM**
aut	16,45	28,75	2,97	13,52	3,00	18,44	6,27	5,69	25,00	12,80	66,34
bpa	3,59	148,30	1,58	3,70	5,05	4,22	3,35	3,80	6,00	5,27	42,03
cmc	3,56	68,23	2,62	4,39	3,03	5,20	5,80	1,95	9,00	5,57	57,30
col	14,29	124,70	2,80	14,57	3,15	14,83	2,65	2,94	22,00	7,10	36,96
gls	3,83	15,97	2,41	4,70	3,32	6,86	3,63	3,47	9,00	5,95	64,49
hc	6,96	20,05	3,17	6,82	5,35	6,48	9,25	3,53	13,00	6,13	43,23
hs	7,08	106,55	3,41	6,91	5,00	6,73	4,95	3,34	13,00	7,02	35,56
irs	1,74	13,63	1,16	1,90	3,90	2,45	2,47	1,21	4,00	3,70	63,33
pim	4,73	110,85	2,13	5,13	2,10	5,14	4,55	3,79	8,00	5,66	33,98
son	47,71	168,40	3,65	46,69	2,85	50,00	9,55	6,31	60,00	34,12	53,37
tao	1,75	94,80	1,47	1,58	2,60	1,61	1,80	1,30	2,00	1,92	44,07
thy	3,21	66,30	1,39	2,78	4,63	3,67	2,30	2,31	5,00	4,26	30,23
veh	11,78	42,03	2,25	10,52	3,35	14,63	5,03	6,41	18,00	9,12	73,40
wbcd	4,26	129,60	1,82	5,48	6,10	3,91	4,75	3,29	9,00	5,40	62,37
wdbc	22,39	142,45	1,65	21,48	1,95	22,46	8,85	3,48	30,00	15,34	32,69
wne	7,00	86,70	1,77	7,15	2,80	8,19	5,10	3,02	13,00	7,73	60,11
wpbc	25,21	136,70	1,55	24,36	3,80	25,06	8,20	7,08	33,00	16,45	23,74
Promedio	10,91	88,47	2,22	10,69	3,65	11,76	5,21	3,70	16,41	9,03	48,42
Desv	11,85	50,77	0,77	11,37	1,19	12,16	2,50	1,73	14,44	7,58	15,08

Tabla Anexol.4. Cantidad de variables por regla de los algoritmos con cada base de datos

Condiciones											
Base de datos	**AdaBoost**	**GAP**	**Ishibuchi99**	**LogitBoost**	**SAP**	**MaxLogitBoost**	**GP**	**SLAVE**	**MOGUL**	**Ishib-Hybrid**	**MR-GIM**
aut	131,60	172,50	187,50	338,00	18,00	147,50	377,70	87,40	2000,00	120,30	40,49
bpa	28,70	296,60	60,90	92,60	10,10	31,70	588,20	21,80	1320,00	69,70	37,10
cmc	28,50	204,70	214,30	109,80	9,10	41,60	616,50	9,10	7050,60	36,40	51,39
col	114,30	249,40	225,10	364,20	6,30	115,70	173,40	9,70	48,40	33,00	16,85
gls	30,60	95,80	121,50	117,60	19,90	54,90	585,50	47,30	934,20	55,90	37,38
hc	55,70	40,10	231,90	170,50	10,70	51,80	564,10	24,80	2054,00	112,60	21,78
hs	56,60	213,10	228,90	172,70	10,00	53,80	563,30	27,60	1989,00	83,20	22,96
irs	13,90	40,90	15,50	47,40	11,70	16,70	526,70	4,50	267,60	55,10	45,33
pim	37,80	221,70	151,00	128,30	4,20	41,10	646,50	35,90	3585,60	66,80	30,08
son	381,70	336,80	197,80	1167,20	5,70	50,00	585,10	53,30	6036,00	119,20	33,17
tao	14,00	189,60	21,70	39,40	5,20	12,20	722,20	3,90	137,40	38,40	15,25
thy	25,70	198,90	40,40	69,40	13,90	29,00	411,70	13,10	512,00	52,00	17,21
veh	94,20	168,10	183,10	263,10	13,40	117,00	569,80	160,60	8701,20	90,10	35,11
wbcd	34,10	259,20	99,40	137,10	12,20	30,50	313,60	21,20	3377,70	60,20	15,59
wdbc	179,10	284,90	75,40	537,10	3,90	177,40	570,80	15,40	9186,00	145,50	8,26
wne	56,00	260,10	109,40	178,80	8,40	65,50	690,20	11,60	1125,80	73,10	19,10
wpbc	201,70	273,40	48,10	609,10	7,60	200,50	493,60	55,70	2448,60	194,10	35,86
Promedio	87,31	206,22	130,11	267,19	10,02	72,76	529,35	35,46	2986,71	82,68	28,41
Desv	94,80	84,35	77,56	284,33	4,56	57,29	139,46	39,19	2971,36	43,35	12,26

Tabla Anexol.5. Cantidad de condiciones de los algoritmos con cada base de datos

Anexo II. Precisión del modelo aprendido por cada algoritmo

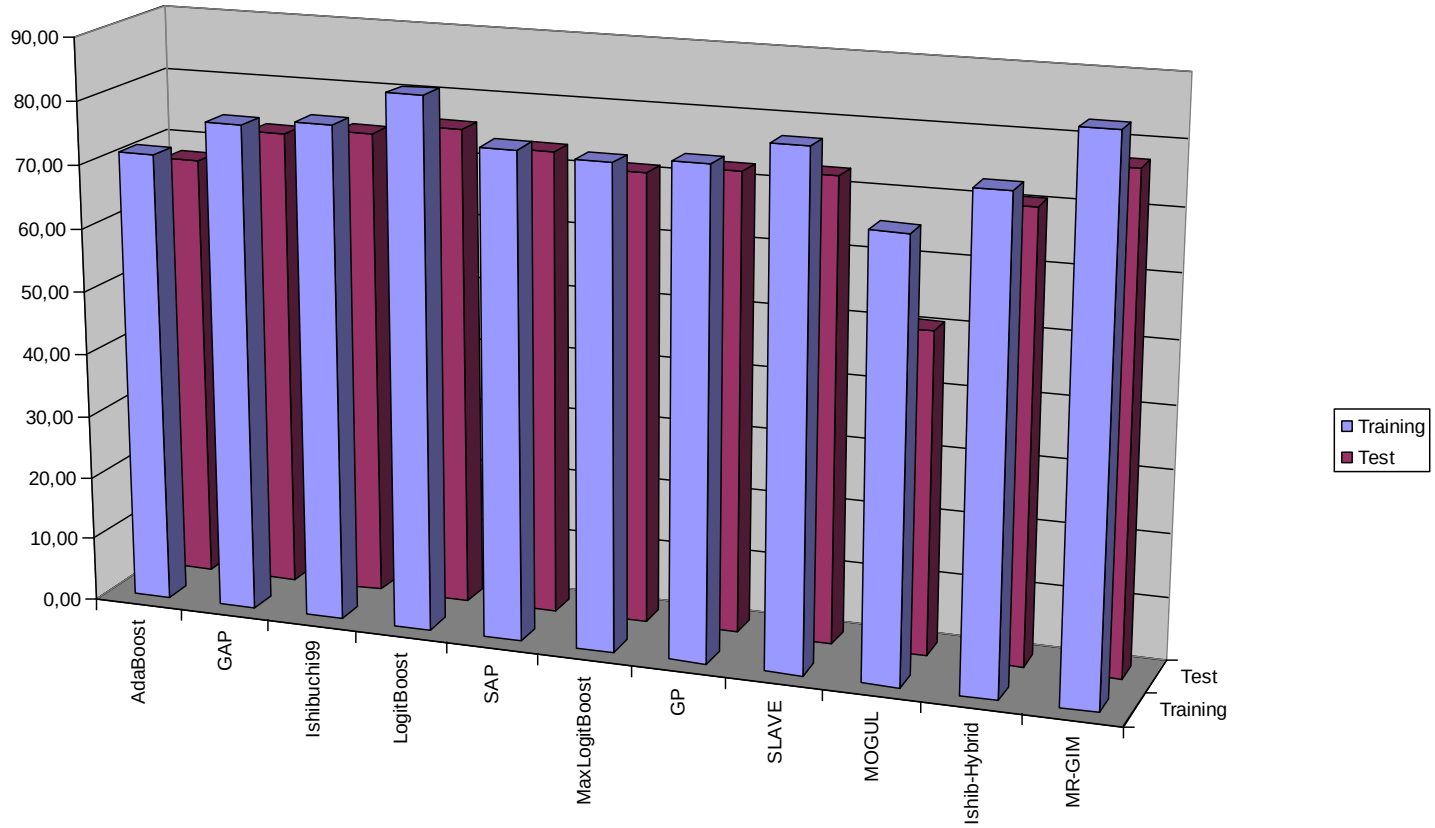

Figura AnexoII. Precisión de cada algoritmo en los conjuntos de entrenamiento y prueba

Anexo III. Simplicidad y comprensibilidad del modelo aprendido por los algoritmos

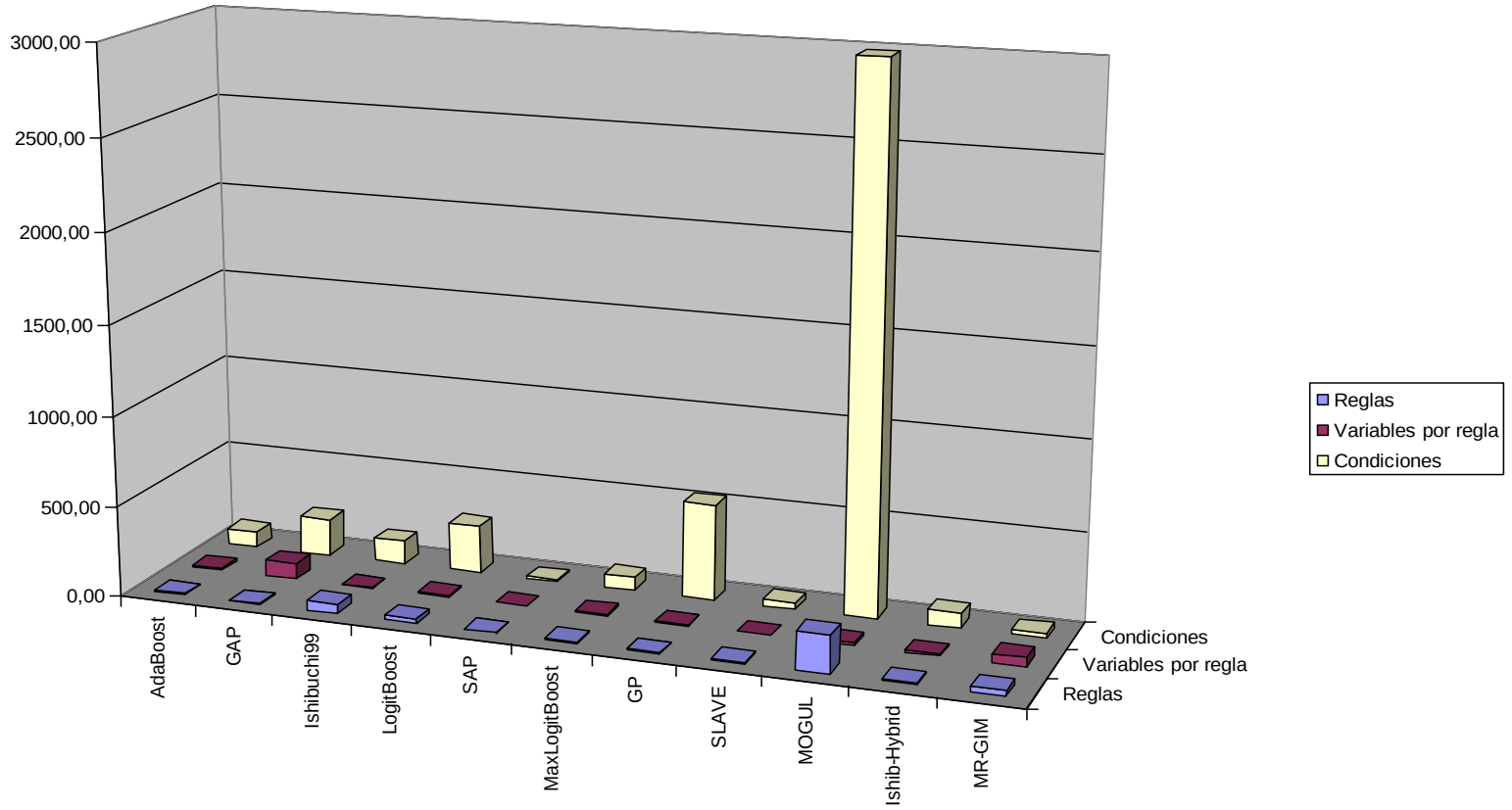

Figura AnexoIII. Resultados obtenidos por los algoritmos en cuanto a cantidad de reglas, variables por regla y condiciones

Anexo IV Valores ranking de los algoritmos para el Test de Friedman

Training											
Bases de datos	**AdaBoost**	**GAP**	**Ishibuchi99**	**LogitBoost**	**SAP**	**MaxLogitBoost**	**GP**	**SLAVE**	**MOGUL**	**Ishib-Hybrid**	**MR-GIM**
aut	15,39 (11)	51,81 (6)	70,13 (3)	53,49 (5)	50,89 (7)	31,11 (10)	55,67 (4)	82,66 (2)	45,04 (9)	49 (8)	94,3 (1)
bpa	71,62 (5)	71,27 (6)	60,09 (11)	78,51 (2)	68,59 (7)	73,62 (3)	61,58 (10)	65,44 (9)	82,58 (1)	73,43 (4)	66,7 (8)
cmc	53,1 (4)	49,75 (7)	45,01 (10)	58,39 (2)	50,15 (6)	49,07 (8)	51,15 (5)	25,96 (11)	64,49 (1)	45,08 (9)	54,1 (3)
col	66,36 (10)	85,96 (4)	70 (9)	78,71 (5)	85,97 (3)	71,08 (7)	86,14 (2)	70,65 (8)	0 (11)	76,58 (6)	89,6 (1)
gls	46,51 (11)	58,26 (8)	69,05 (5)	76,94 (3)	52,8 (9)	68,49 (6)	52,63 (10)	73,94 (4)	77,37 (2)	66,05 (7)	78,5 (1)
h-c	80,74 (6)	78,25 (9)	87,45 (3)	95,45 (1)	79,35 (8)	84,38 (4)	84,31 (5)	80,42 (7)	51,09 (11)	76,82 (10)	88 (2)
h-s	80,41 (7)	81,06 (6)	88,76 (3)	96,04 (1)	79,87 (8)	84,24 (4)	81,19 (5)	78,85 (10)	49,88 (11)	79,3 (9)	89 (2)
irs	96 (8)	91,4 (11)	95,62 (9)	98,66 (3)	96,96 (5)	98,3 (4)	92,07 (10)	96,37 (7)	99,33 (1)	99,11 (2)	96,4 (6)
pim	78,21 (5)	76,59 (8,5)	69,9 (11)	83,37 (2)	76,47 (10)	80,31 (3)	76,59 (8,5)	77,71 (6)	88,54 (1)	76,76 (7)	79,7 (4)
son	47,32 (9)	78,63 (3)	76,97 (5)	56,35 (7)	76,22 (6)	46,63 (10)	77,88 (4)	87,44 (2)	36,11 (11)	54,28 (8)	93,4 (1)
tao	82,31 (8)	78,44 (11)	81,49 (10)	87,07 (3)	86,95 (4)	86,75 (5)	82,37 (7)	82,09 (9)	93,76 (1)	91,21 (2)	82,6 (6)
thy	91,37 (7)	90,49 (9)	87,39 (10,5)	98,19 (2)	90,69 (8)	95,61 (4)	87,39 (10,5)	93,95 (6)	98,71 (1)	96,28 (3)	94,1 (5)
veh	53,53 (8)	56,06 (7)	64,45 (5)	70,38 (4)	49,88 (11)	61,16 (6)	53,23 (9)	75,64 (3)	81,03 (2)	51,64 (10)	83 (1)
wbcd	93,61 (10)	93,57 (11)	96,77 (7)	97,89 (1)	94,25 (9)	97,58 (3)	96,85 (6)	97,87 (2)	95,71 (8)	97,09 (5)	97,4 (4)
wdbc	69,02 (11)	85,51 (9)	92,59 (6)	96,01 (2)	80,02 (10)	93,24 (5)	94,86 (3)	93,83 (4)	90,75 (8)	91,47 (7)	96,5 (1)
wine	96,34 (6)	96,15 (8)	98,13 (3)	99,44 (1)	96,21 (7)	98,69 (2)	90,32 (10)	97,82 (5)	85,68 (11)	95,01 (9)	97,9 (4)
wpbc	93,63 (2)	91,76 (3)	77,55 (8)	100 (1)	88,45 (4)	68,01 (10)	79,01 (7)	85,63 (5)	30,16 (11)	76,6 (9)	80,1 (6)
Sum Rank	7,53	7,44	6,97	2,65	7,18	5,53	6,82	5,88	5,94	6,76	3,29

Tabla AnexoIV.1. Valores ranking del por ciento de clasificación obtenido por los algoritmos con los conjuntos de datos de entrenamiento

Test											
Bases de datos	**AdaBoost**	**GAP**	**Ishibuchi99**	**LogitBoost**	**SAP**	**MaxLogitBoost**	**GP**	**SLAVE**	**MOGUL**	**Ishib-Hybrid**	**MR-GIM**
aut	12,23 (11)	41,47 (7)	57,07 (3)	47,28 (5)	46,8 (6)	22,88 (10)	51,66 (4)	61,48 (2)	28,04 (9)	38,5 (8)	69,6 (1)
bpa	64,06 (2)	60,82 (6)	58,25 (10)	70,16 (1)	63,49 (4)	62,59 (5)	55,32 (11)	60,53 (7)	59,98 (8)	63,77 (3)	58,4 (9)
cmc	51,39 (2)	48,33 (4)	43,92 (7)	53,9 (1)	49,01 (3)	46,17 (6)	47,65 (5)	24,31 (11)	37,9 (10)	43,79 (8)	42,2 (9)
col	66,64 (8)	83,68 (4)	71,84 (5)	70,58 (7)	83,97 (2)	71,47 (6)	85,86 (1)	63,02 (10)	0 (11)	66,21 (9)	83,9 (3)
gls	42,87 (11)	54,35 (8)	59,88 (6)	67,02 (2)	48,54 (9)	60,82 (4)	44,99 (10)	61,33 (3)	56,9 (7)	59,95 (5)	69,5 (1)
h-c	74,88 (6)	73,49 (7)	77,87 (2)	76,24 (5)	76,88 (4)	73,27 (8)	78,17 (1)	72,24 (9)	25,38 (11)	69,98 (10)	77,8 (3)
h-s	74,07 (6,5)	78,88 (1)	78,14 (3)	74,81 (5)	74,07 (6,5)	78,15 (2)	72,96 (8)	70 (9)	19,63 (11)	69,63 (10)	76,6 (4)
irs	96 (1,5)	90,66 (10)	95,33 (4,5)	95,33 (4,5)	95,33 (4,5)	95,33 (4,5)	86,66 (11)	94 (9)	94,67 (7)	96 (1,5)	94,6 (8)
pim	72,78 (8)	73,83 (4)	68,75 (11)	75,91 (1)	72,53 (9)	75 (2)	73,58 (5)	74,1 (3)	70,98 (10)	73,45 (7)	73,5 (6)
son	46,61 (9)	61,07 (6)	61,59 (5)	51,95 (7)	69,14 (4)	46,62 (8)	70,69 (2)	70,62 (3)	5,3 (11)	40,81 (10)	72,5 (1)
tao	82,15 (7)	76,15 (11)	81,19 (9)	87,02 (3)	86,91 (4)	86,49 (5)	82,82 (6)	80,93 (10)	91,9 (1)	90,62 (2)	81,5 (8)
thy	89,81 (7)	88,78 (8)	87,04 (10)	96,25 (1)	88,54 (9)	92,64 (4)	85,49 (11)	90,75 (6)	93,03 (2,5)	93,03 (2,5)	92,2 (5)
veh	50,36 (10)	52 (7)	61,47 (5)	64,41 (3)	50,57 (9)	58,55 (6)	48,68 (11)	66,3 (2)	63,03 (4)	50,82 (8)	68,5 (1)
wbcd	91,58 (10)	93,49 (7)	95,41 (6)	93,15 (9)	93,33 (8)	95,99 (4)	95,71 (5)	96 (2)	75,83 (11)	96 (3)	96,4 (1)
wdbc	50,5 (11)	81,89 (7)	92,56 (3)	72,79 (9)	71,17 (10)	91,42 (5)	93,15 (2)	91,56 (4)	76,01 (8)	90 (6)	93,5 (1)
wne	94,29 (4)	94,55 (3)	93,85 (5)	95,14 (2)	95,85 (1)	93,82 (6)	83,14 (10)	92,09 (8)	58,3 (11)	88,73 (9)	93,3 (7)
wpbc	89,86 (2)	88,23 (3)	76,35 (6)	96,63 (1)	80,91 (4)	59,5 (10)	76,83 (5)	72,17 (8)	5,78 (11)	70,78 (9)	74,3 (7)
Sum Rank	6,82	6,06	5,91	3,91	5,71	5,62	6,35	6,24	8,44	6,53	4,41

Tabla AnexoIV.2. Valores ranking del por ciento de clasificación obtenido por los algoritmos con los conjuntos de datos de prueba

Reglas											
Bases de datos	**AdaBoost**	**GAP**	**Ishibuchi99**	**LogitBoost**	**SAP**	**MaxLogitBoost**	**GP**	**SLAVE**	**MOGUL**	**Ishib-Hybrid**	**MR-GIM**
aut	8 (7,5)	6 (10)	63,2 (2)	25 (4)	6 (10)	8 (7,5)	6 (10)	15,7 (5)	80 (1)	9,4 (6)	44,88 (3)
bpa	8 (6)	2 (10)	36,5 (3)	25 (4)	2 (10)	7,5 (7)	2 (10)	5,8 (8)	220 (1)	13,2 (5)	40,58 (2)
cmc	8 (5,5)	3 (10)	81,8 (2)	25 (4)	3 (10)	8 (5,5)	3 (10)	4,7 (8)	783,4 (1)	6,4 (7)	54,04 (3)
col	8 (4)	2 (10)	80,3 (1)	25 (2)	2 (10)	7,8 (5)	2 (10)	3,3 (7)	2,2 (8)	3,8 (6)	19,57 (3)
gls	8 (7,5)	6 (10)	50,5 (2)	25 (4)	6 (10)	8 (7,5)	6 (10)	13,8 (5)	103,8 (1)	9,5 (6)	40,19 (3)
h-c	8 (6,5)	2 (10)	72,9 (2)	25 (3)	2 (10)	8 (6,5)	2 (10)	7,1 (8)	158 (1)	18,4 (5)	19,14 (4)
h-s	8 (7,5)	2 (10)	67 (2)	25 (3)	2 (10)	8 (7,5)	2 (10)	8,4 (6)	153 (1)	14,9 (5)	21,11 (4)
irs	8 (6)	3 (10)	13,2 (5)	25 (3)	3 (10)	6,8 (7)	3 (10)	3,7 (8)	66,9 (1)	14,9 (4)	52,67 (2)
pim	8 (7,5)	2 (10)	70,5 (2)	25 (4)	2 (10)	8 (7,5)	2 (10)	9,6 (6)	448,2 (1)	11,9 (5)	28,39 (3)
son	8 (6)	2 (9)	54 (2)	25 (4)	2 (9)	1 (11)	2 (9)	8,8 (5)	100,6 (1)	3,6 (7)	39,42 (3)
tao	8 (6)	2 (10)	15,6 (4)	25 (2)	2 (10)	7,5 (7)	2 (10)	3 (8)	68,7 (1)	20 (3)	14,72 (5)
thy	8 (6)	3 (10)	29,1 (2)	25 (3)	3 (10)	7,9 (7)	3 (10)	5,7 (8)	102,4 (1)	12,2 (5)	12,56 (4)
veh	8 (7,5)	4 (10)	81,5 (2)	25 (5)	4 (10)	8 (7,5)	4 (10)	26,1 (4)	483,4 (1)	9,9 (6)	33,33 (3)
wbcd	8 (5)	2 (10)	45,8 (2)	25 (3)	2 (10)	7,8 (6)	2 (10)	6,3 (7)	375,3 (1)	11,3 (4)	4,72 (8)
wdbc	8 (5)	2 (10)	45,7 (2)	25 (3)	2 (10)	7,9 (6)	2 (10)	4,7 (7)	306,2 (1)	9,5 (4)	3,69 (8)
wne	8 (6,5)	3 (10)	61,5 (2)	25 (3)	3 (10)	8 (6,5)	3 (10)	3,9 (8)	86,6 (1)	9,5 (5)	12,36 (4)
wpbc	8 (7,5)	2 (10)	31,1 (3)	25 (4)	2 (10)	8 (7,5)	2 (10)	8,1 (6)	74,2 (1)	11,6 (5)	35,35 (2)
Sum Rank	6,32	9,94	2,35	3,41	9,94	7,03	9,94	6,71	1,41	5,18	3,76

Tabla AnexoIV.3. Valores ranking de la cantidad de reglas del modelo de aprendizaje de cada algoritmo

Variables por regla											
Bases de datos	**AdaBoost**	**GAP**	**Ishibuchi99**	**LogitBoost**	**SAP**	**MaxLogitBoost**	**GP**	**SLAVE**	**MOGUL**	**Ishib-Hybrid**	**MR-GIM**
aut	16,45 (5)	28,75 (2)	2,97 (11)	13,52 (6)	3 (10)	18,44 (4)	6,27 (8)	5,69 (9)	25 (3)	12,8 (7)	66,34 (1)
bpa	3,59 (9)	148,3 (1)	1,58 (11)	3,7 (8)	5,05 (5)	4,22 (6)	3,35 (10)	3,8 (7)	6 (3)	5,27 (4)	42,03 (2)
cmc	3,56 (8)	68,23 (1)	2,62 (10)	4,39 (7)	3,03 (9)	5,2 (6)	5,8 (4)	1,95 (11)	9 (3)	5,57 (5)	57,3 (2)
col	14,29 (6)	124,7 (1)	2,8 (10)	14,57 (5)	3,15 (8)	14,83 (4)	2,65 (11)	2,94 (9)	22 (3)	7,1 (7)	36,96 (2)
gls	3,83 (7)	15,97 (2)	2,41 (11)	4,7 (6)	3,32 (10)	6,86 (4)	3,63 (8)	3,47 (9)	9 (3)	5,95 (5)	64,49 (1)
h-c	6,96 (5)	20,05 (2)	3,17 (11)	6,82 (6)	5,35 (9)	6,48 (7)	9,25 (4)	3,53 (10)	13 (3)	6,13 (8)	43,23 (1)
h-s	7,08 (4)	106,55 (1)	3,41 (10)	6,91 (6)	5 (8)	6,73 (7)	4,95 (9)	3,34 (11)	13 (3)	7,02 (5)	35,56 (2)
irs	1,74 (9)	13,63 (2)	1,16 (11)	1,9 (8)	3,9 (4)	2,45 (7)	2,47 (6)	1,21 (10)	4 (3)	3,7 (5)	63,33 (1)
pim	4,73 (7)	110,85 (1)	2,13 (10)	5,13 (6)	2,1 (11)	5,14 (5)	4,55 (8)	3,79 (9)	8 (3)	5,66 (4)	33,98 (2)
son	47,71 (5)	168,4 (1)	3,65 (10)	46,69 (6)	2,85 (11)	50 (4)	9,55 (8)	6,31 (9)	60 (2)	34,12 (7)	53,37 (3)
tao	1,75 (7)	94,8 (1)	1,47 (10)	1,58 (9)	2,6 (3)	1,61 (8)	1,8 (6)	1,3 (11)	2 (4)	1,92 (5)	44,07 (2)
thy	3,21 (7)	66,3 (1)	1,39 (11)	2,78 (8)	4,63 (4)	3,67 (6)	2,3 (10)	2,31 (9)	5 (3)	4,26 (5)	30,23 (2)
veh	11,78 (5)	42,03 (2)	2,25 (11)	10,52 (6)	3,35 (10)	14,63 (4)	5,03 (9)	6,41 (8)	18 (3)	9,12 (7)	73,4 (1)
wbcd	4,26 (8)	129,6 (1)	1,82 (11)	5,48 (5)	6,1 (4)	3,91 (9)	4,75 (7)	3,29 (10)	9 (3)	5,4 (6)	62,37 (2)
wdbc	22,39 (5)	142,45 (1)	1,65 (11)	21,48 (6)	1,95 (10)	22,46 (4)	8,85 (8)	3,48 (9)	30 (3)	15,34 (7)	32,69 (2)
wne	7 (7)	86,7 (1)	1,77 (11)	7,15 (6)	2,8 (10)	8,19 (4)	5,1 (8)	3,02 (9)	13 (3)	7,73 (5)	60,11 (2)
wpbc	25,21 (3)	136,7 (1)	1,55 (11)	24,36 (5)	3,8 (10)	25,06 (4)	8,2 (8)	7,08 (9)	33 (2)	16,45 (7)	23,74 (6)
Sum Rank	6,29	1,29	10,65	6,41	8,00	5,47	7,76	9,35	2,94	5,82	2,00

Tabla AnexoIV.4. Valores ranking de la cantidad de variables por regla del modelo de aprendizaje de cada algoritmo

Condiciones											
Bases de datos	**AdaBoost**	**GAP**	**Ishibuchi99**	**LogitBoost**	**SAP**	**MaxLogitBoost**	**GP**	**SLAVE**	**MOGUL**	**Ishib-Hybrid**	**MR-GIM**
aut	131,6 (7)	172,5 (5)	187,5 (4)	338 (3)	18 (11)	147,5 (6)	377,7 (2)	87,4 (9)	2000 (1)	120,3 (8)	40,49 (10)
bpa	28,7 (9)	296,6 (3)	60,9 (6)	92,6 (4)	10,1 (11)	31,7 (8)	588,2 (2)	21,8 (10)	1320 (1)	69,7 (5)	37,1 (7)
cmc	28,5 (9)	204,7 (4)	214,3 (3)	109,8 (5)	9,1 (10,5)	41,6 (7)	616,5 (2)	9,1 (10,5)	7050,6 (1)	36,4 (8)	51,39 (6)
col	114,3 (6)	249,4 (2)	225,1 (3)	364,2 (1)	6,3 (11)	115,7 (5)	173,4 (4)	9,7 (10)	48,4 (7)	33 (8)	16,85 (9)
gls	30,6 (10)	95,8 (5)	121,5 (3)	117,6 (4)	19,9 (11)	54,9 (7)	585,5 (2)	47,3 (8)	934,2 (1)	55,9 (6)	37,38 (9)
h-c	55,7 (6)	40,1 (8)	231,9 (3)	170,5 (4)	10,7 (11)	51,8 (7)	564,1 (2)	24,8 (9)	2054 (1)	112,6 (5)	21,78 (10)
h-s	56,6 (7)	213,1 (4)	228,9 (3)	172,7 (5)	10 (11)	53,8 (8)	563,3 (2)	27,6 (9)	1989 (1)	83,2 (6)	22,96 (10)
irs	13,9 (9)	40,9 (6)	15,5 (8)	47,4 (4)	11,7 (10)	16,7 (7)	526,7 (1)	4,5 (11)	267,6 (2)	55,1 (3)	45,33 (5)
pim	37,8 (8)	221,7 (3)	151 (4)	128,3 (5)	4,2 (11)	41,1 (7)	646,5 (2)	35,9 (9)	3585,6 (1)	66,8 (6)	30,08 (10)
son	381,7 (4)	336,8 (5)	197,8 (6)	1167,2 (2)	5,7 (11)	50 (9)	585,1 (3)	53,3 (8)	6036 (1)	119,2 (7)	33,17 (10)
tao	14 (8)	189,6 (2)	21,7 (6)	39,4 (4)	5,2 (10)	12,2 (9)	722,2 (1)	3,9 (11)	137,4 (3)	38,4 (5)	15,25 (7)
thy	25,7 (8)	198,9 (3)	40,4 (6)	69,4 (4)	13,9 (10)	29 (7)	411,7 (2)	13,1 (11)	512 (1)	52 (5)	17,21 (9)
veh	94,2 (8)	168,1 (5)	183,1 (4)	263,1 (3)	13,4 (11)	117 (7)	569,8 (2)	160,6 (6)	8701,2 (1)	90,1 (9)	35,11 (10)
wbcd	34,1 (7)	259,2 (3)	99,4 (5)	137,1 (4)	12,2 (11)	30,5 (8)	313,6 (2)	21,2 (9)	3377,7 (1)	60,2 (6)	15,59 (10)
wdbc	179,1 (5)	284,9 (4)	75,4 (8)	537,1 (3)	3,9 (11)	177,4 (6)	570,8 (2)	15,4 (9)	9186 (1)	145,5 (7)	8,26 (10)
wne	56 (8)	260,1 (3)	109,4 (5)	178,8 (4)	8,4 (11)	65,5 (7)	690,2 (2)	11,6 (10)	1125,8 (1)	73,1 (6)	19,1 (9)
wpbc	201,7 (5)	273,4 (4)	48,1 (9)	609,1 (2)	7,6 (11)	200,5 (6)	493,6 (3)	55,7 (8)	2448,6 (1)	194,1 (7)	35,86 (10)
Sum Rank	7,29	4,06	5,06	3,59	10,79	7,12	2,12	9,26	1,53	6,29	8,88

Tabla AnexoIV.5. Valores ranking de la cantidad de condiciones del modelo de aprendizaje de cada algoritmo.