Universidad de las Ciencias Informáticas
Facultad 6

Título: Implementación y evaluación de un algoritmo basado en Lógica Difusa para la predicción de actividad biológica

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor(es): Juan Manuel Ruiz Godoy
Yanelis Ramírez Hernández

Tutor(es): Dr. Ramón Carrasco Velar
Ing.Ileane Martí Pérez
Ing. Julio Omar Prieto Entenza

Ciudad de la Habana, Cuba.
Junio ,2008
"Año 50 de la Revolución."

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Yanelis Ramírez Hernández

_____

Firma del autor

Juan Manuel Ruiz Godoy

_____

Firma del autor

Ing. Ileane Martí Pérez

_____

Firma del tutor

Julio Omar Prieto Entenza

_____

Firma del tutor

Dr. Ramón Carrasco Velar

_____

Firma del Autor

“Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber”.

Albert Einstein

DATOS DE CONTACTO

Tutores:

Dr. Ramón Carrasco Velar
Centro de Química Farmacéutica, Ciudad de La Habana, Cuba.
ramon.carrasco@cqf.sld.cu
rcarrasco@uci.cu

Ing. Julio Omar Prieto Entenza
Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.
jprieto@uci.cu

Ing. Ileane Martí Pérez
Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.
imarti@uci.cu

AGRADECIMIENTOS

Agradecemos a nuestros padres por su apoyo, confianza y ejemplo en todo momento para que hoy seamos esas personas de las cuales esperan lo mejor. A nuestros tutores Julio, Ileane y Carrasco, por su paciencia, preocupación y por ser quienes dedicaron parte de su tiempo en enseñarnos que la sabiduría no tiene fronteras. A Francisco Herrera por haber respondido con rapidez a nuestra solicitud y habernos puesto en contacto con Alberto Fernández, a quien agradecemos su ayuda de forma incondicional para el desarrollo de la investigación. Gracias a los dos por brindarnos su valioso conocimiento, del cual aprendimos y continuamos aprendiendo. A Aurelio por su orientación y preocupación en el desarrollo de la investigación. A Fidel Castro Ruz por ser el guía impulsor de este gran proyecto, que es la Universidad de las Ciencias Informáticas y por ser un ejemplo a seguir en esta Revolución. A nuestros familiares que confiaron plenamente en nosotros, siempre con la esperanza que venceríamos los obstáculos para lograr nuestros sueños. A Yanelis Benítez porque de una forma u otra nos brindó su apoyo y por sus sabios consejos. A nuestras amistades por preocuparse por nosotros, pues día a día nos preguntaban: ¿y la tesis? A los muchachos del proyecto GRAph-TOol por habernos brindado su ayuda a pesar de estar enfrascados en otras tareas. A todos desde lo más profundo de nuestros corazones,

Muchas Gracias.

DEDICATORIA

- *A mis padres Gisela y Javier por ser lo más preciado que tengo en este mundo, por ser mis fieles amigos y ser las personas que más amo en la vida. Siempre estaré muy orgullosa de ustedes.*
- *A mi hermana Yaniuska por quererme tanto y ver en mí alguien especial.*
- *A mis abuelos Herminia, Elia y Jorge por siempre demostrarme su amor incondicional. Los quiero con la vida.*
- *A Juan Manuel por ser el amor de mi vida, por brindarme su apoyo, comprensión y amor incondicional en los buenos y malos momentos.*
- *A mi tía Lucy y mi tío Héctor por quererme como una hija.*
- *A mis tíos y tías por siempre confiar en mí, por su apoyo y preocupación incondicional.*
- *A mis primos y primas por quererme tanto y preocuparse por mí.*
- *A mis amistades que de una forma u otra hicieron posible este momento.*

Yanelis Ramírez Hernández

- *A mis padres Petrona y Juan Manuel por guiarme en la vida y demostrarme día a día su infinito amor.*
- *A mi hermana Ana María por hacerme sentir su mejor amigo, alguien en quien confía y quiere.*
- *A mis abuelos, en quienes vi un ejemplo para mi persona.*
- *A mi familia por darme en todo momento su confianza y su apoyo.*
- *A mi amada Yanelis, por su confianza y dedicación y por encontrar en ella lo que muchos buscamos y pocos alcanzamos: el amor.*
- *A mis amistades que con su confianza contribuyeron a que este momento fuera posible.*

Juan Manuel Ruiz Godoy

RESUMEN

El presente trabajo surge por la necesidad de disponer en la *Plataforma Inteligente para la Predicción de Actividad Biológica de Compuestos Orgánicos* de un procedimiento para la predicción de la misma a partir de la descripción de la estructura molecular. Se implementó y evaluó el *Fuzzy Rule Learning Algorithm (Ishibuchi-99)*, a diferentes muestras, el cual genera modelos predictivos con un buen porciento de veracidad en los resultados. Este algoritmo combina los principios de la lógica difusa con técnicas de algoritmo genético para su funcionamiento. Se evaluó con las bases de datos del IRIS y WINE, con las que se generaron modelos difusos con porcientos de acierto por encima del 93%. A partir de esos resultados se desarrolló un estudio de clasificación de una muestra de cefalosporinas de cuatro generaciones reportadas como antibacteriales frente a cepas de *E. coli*. La muestra de 104 compuestos se dividió en muestras de entrenamiento (90%) y prueba (10%). Se analizaron las funciones de pertenencia triangular, trapezoidal y de Gauss. El mejor resultado se alcanzó para la función de pertenencia triangular. Con ella se alcanzó un 82.48% de buena clasificación como promedio en las diez muestras de entrenamiento. En las muestras de prueba se logró un 71% de buena predicción como promedio, lo cual se considera un resultado aceptable debido a la complejidad estructural de la muestra. Se propone la inclusión del algoritmo implementado a la plataforma GRAph-TOol con el uso de la función de pertenencia triangular.

PALABRAS CLAVES

Predicción, lógica difusa, algoritmo genético, actividad biológica, modelos difusos.

TABLA DE CONTENIDOS

INTRODUCCIÓN

La bioinformática es una nueva disciplina que surge como respuesta al creciente aumento de los volúmenes de datos biológicos y la facilidad de compartir esta información a través de Internet. Esta ciencia une la biología, la informática y las tecnologías de la información con vista al análisis, organización y distribución de la información biológica. En tal sentido, promete grandes avances en las investigaciones, como por ejemplo, en el diagnóstico, tratamiento y prevención de diversas enfermedades que puedan mejorar la calidad de vida. Es así que el desarrollo de la industria farmacéutica guarda una estrecha relación con la bioinformática, la cual le permite la búsqueda, cada vez más eficiente, de nuevos fármacos para el tratamiento de diversas enfermedades.

Este vínculo entre la bionformática y la industria farmacéutica ha repercutido en la búsqueda de nuevos principios activos a través de ensayos clínicos sobre un gran número de sustancias químicas, lo que requiere de mucho esfuerzo debido a la necesidad de realizar estudios sobre 10 000 compuestos para encontrar el deseado. Esto supone que el costo del desarrollo de un nuevo medicamento desde su obtención en el laboratorio hasta su uso en terapéutica es de 360-500 millones de USD y en un período promedio de doce a quince años [1].

El proceso de obtención de nuevos fármacos es un proceso altamente costoso que requiere investigaciones de gran complejidad. Esta dificultad viene dada, entre otras cosas, por el gran número de compuestos conocidos que ya sobrepasan los veintiséis millones, de los cuales un gran número no han encontrado aplicaciones farmacológicas. Para darle solución a esta problemática comenzaron a desarrollarse diferentes técnicas de procesamiento de datos, que han sido muy útiles en la creación de modelos de predicción, como por ejemplo, la predicción de actividad biológica para el diseño racional de fármacos asistidos por ordenadores. Esto último constituye una valiosa herramienta en el avance para la obtención de nuevos medicamentos, lo que disminuye tanto los costos como los tiempos de ejecución, con la expectativa de alcanzar una mayor probabilidad de éxito.

Entre las ramas que más han aportado al diseño racional de fármacos asistidos por ordenadores se encuentran la Estadística y la Inteligencia Artificial (IA), que brindan un conjunto de técnicas para procesar los datos biológicos. Actualmente existen diversas aplicaciones que hacen uso de estas técnicas como por ejemplo: Codessa (Comprehensive Descriptors for Structural and Statistical Análisis) [2], el Dragon, Molconn-Z y ADAPT (Automated Data Analysis using Pattern Recognition Toolkit) [3] las

cuales constituyen un aporte importante a los estudios de estructura-actividad. Gran parte de estas aplicaciones son internacionales y/o comerciales lo que constituye una dificultad para los especialistas cubanos pues necesitan pagar en divisas o tener conexión a Internet para poder utilizarlos. Se suma el hecho de los problemas con la brecha digital y de la portabilidad de las aplicaciones, ya que por lo general no son multiplataforma.

En busca de una solución a las dificultades planteadas anteriormente el Centro de Química Farmacéutica (CQF) en conjunto con la Facultad 6 de la Universidad de las Ciencias Informáticas (UCI) desarrollan el proyecto denominado: **"Una Plataforma Inteligente para la Predicción de Actividad Biológica de Compuestos Orgánicos"**, la cual posee entre sus necesidades, conocer cuáles son los resultados de predicción de la relación estructura-actividad de compuestos orgánicos con la utilización de diferentes técnicas de IA a partir de la descripción de la molécula.

Como respuesta al desarrollo de la plataforma se realizaron varias investigaciones sobre cómo predecir actividad biológica de compuestos orgánicos con el uso de la lógica difusa, cuyos resultados no fueron los esperados, ya que se obtuvo un porciento de predicción bastante bajo.

Entonces, se plantea la siguiente interrogante: **¿Cómo mejorar la predicción de actividad biológica de compuestos orgánicos para la plataforma GRAph-TOol?**

De lo anterior se define como **objeto de estudio** la lógica difusa aplicada a la predicción de actividad biológica. El **campo de acción** se enmarca en la lógica difusa aplicada al estudio de la relación estructura-actividad de compuestos orgánicos. Para dar solución al problema planteado se trazó como **objetivo general**: *Implementar un algoritmo capaz de generar modelos predictivos de actividad biológica de compuestos orgánicos empleando técnicas de lógica difusa.*

Para cumplir con el objetivo propuesto se plantean los siguientes **objetivos específicos:**

- Implementar un algoritmo basado en lógica difusa.
- Implementar una aplicación de prueba que tenga incorporado el algoritmo seleccionado.
- Validar la implementación con datos reportados.
- Realizar un estudio de clasificación con otro juego de datos con el uso del algoritmo implementado.

Para el cumplimiento de los objetivos específicos se planean las siguientes **tareas**:

- Revisión del estado del arte acerca de sistemas de predicción existentes en el mundo.
- Determinación del algoritmo para la generación de las reglas difusas.
- Determinación de los conjuntos difusos a utilizar.
- Implementación del algoritmo para la generación de reglas.
- Implementación del algoritmo para realizar la predicción basada en los modelos difusos creados.
- Validación del modelo desarrollado, en muestras conocidas.
- Desarrollo de un modelo de clasificación de cefalosporinas reportadas como activas.

El resultado práctico que se espera del presente trabajo de diploma es ganar conocimientos que permitan proponer un método de lógica difusa a implementar en la plataforma GRAph-TOol.

La tesis está estructurada de la siguiente manera: resumen, introducción, tres capítulos, conclusiones, recomendaciones, bibliografía y referencias bibliográficas. Los tres capítulos abarcan los aspectos esenciales relacionados con el contenido de la investigación y la implementación del algoritmo, así como el estudio de clasificación. Su distribución es la siguiente:

Capítulo #1: Fundamentación teórica: En este capítulo se realiza una breve explicación de lo que consistió el estudio realizado sobre los métodos empleados en la creación de nuevos fármacos asistidos por computadora y las técnicas de predicción existentes. Además se muestran una serie de soluciones informáticas vinculadas a la predicción de la relación estructura-actividad.

Capítulo #2: Materiales y Métodos: En este capítulo se describen las tecnologías y herramientas que se emplearon para la realización de la investigación. También se mencionan los aspectos esenciales que presenta el algoritmo seleccionado en su fundamentación matemática, además de presentar el pseudocódigo correspondiente. Además se describe el patrón estrategia como patrón de diseño debido a las ventajas que proporciona.

Capítulo #3: Resultados y Discusión: En este capítulo se reflejan los principales resultados de este trabajo. Se realiza una descripción de las funcionalidades implementadas en el algoritmo seleccionado. Se muestran algunos diagramas relacionados con el diseño, que se proponen para la

creación futura del producto relacionado, así como el uso del patrón estrategia que se utiliza en el desarrollo perspectivo de la implementación definitiva. Por último se presentan las pruebas realizadas en la creación de los modelos difusos y en el proceso de predicción, así como el estudio de clasificación realizado a un conjunto de muestras de cefalosporinas antibacteriales.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se realiza una breve explicación de lo que consistió el estudio realizado sobre los métodos empleados en la creación de nuevos fármacos asistidos por computadora y las técnicas de predicción existentes. Además se muestran una serie de soluciones informáticas vinculadas a la predicción de la relación estructura-actividad.

1.1 Introducción a la modelación molecular.

El modelado molecular es aplicado en una gran variedad de métodos computacionales, con el propósito de identificar las complejas relaciones existentes entre estructuras moleculares y actividades biológicas. Estas relaciones se utilizan de forma predictiva para diseñar compuestos con un determinado perfil de actividad.

Para poder establecer estos modelos es necesario describir la estructura química de manera cuantitativa. Precisamente estos valores cuantitativos son los llamados descriptores, los cuales no son más que números que describen a la molécula. Estos descriptores pueden ser químico-cuánticos, 3D QSAR (Quantitative Structure-Activity Relationships), descriptores químico-físicos, topológicos, topográficos e híbridos.

Para poder determinar un modelo de predicción de estructura-actividad después de haberse seleccionado los descriptores que se emplearán, se pasa a establecer las diferentes relaciones existentes entre la estructura química y la actividad biológica a través de algún método estadístico u otras técnicas como las de IA. Una vez que la relación es generada sólo quedaría determinar la calidad del modelo que puede realizarse con el uso de métodos como el coeficiente de correlación, la validación cruzada o algún otro procedimiento estadístico [3].

La creación de modelos de predicción se ha realizado con sistemas de clasificación basados en árboles de regresión, máquinas de aprendizaje, análisis de discriminantes lineales, redes neuronales, algoritmo genético o lógica difusa. Este último tiene la ventaja de ser más condescendiente con datos imprecisos. Esta capacidad refleja la introducción tanto de nuevas metodologías como de ordenadores con la potencia suficiente para aplicar estos tratamientos al análisis de modelos moleculares detallados.

1.2 Diseño de fármacos asistido por ordenador.

En los años sesenta Hansch y Fujita plantearon relaciones cuantitativas entre parámetros fisicoquímicos moleculares y la actividad biológica mediante técnicas estadísticas. El ordenador empezó a ser necesario para resolver estas relaciones, denominadas QSAR [4].

Los métodos QSAR han demostrado que las relaciones entre la estructura molecular y la actividad biológica de los compuestos se pueden cuantificar matemáticamente a partir de parámetros estructurales simples [5], donde esta relación constituye una alternativa viable para calcular las propiedades físicas, químicas y biológicas de los compuestos. Cabe mencionar que algunas técnicas QSAR que emplean métodos de regresión múltiple fueron aplicadas exitosamente en diversos problemas de diseño de nuevos fármacos.

Actualmente la descripción de las moléculas, para la creación de nuevos medicamentos, no se realiza solamente con la utilización de propiedades químico-físicas determinadas experimentalmente por lo que se han introducido nuevas metodologías basadas en ordenadores para aplicar tratamientos al análisis de modelos moleculares detallados.

Muchos centros relacionados con la creación de nuevos fármacos engloban un número considerable de procedimientos, basados en el uso de ordenadores, encaminados a relacionar actividad con estructura molecular. En este sentido la bioinformática ha tenido un auge gracias al empleo de la computación en el desarrollo de nuevas herramientas para estimar la diversidad molecular y la búsqueda en las bases de datos de compuestos químicos.

El uso de ordenadores, al igual que sucede en todas las áreas del conocimiento, ha facilitado y abaratado los costes de las diversas etapas de I+D (Investigación-Desarrollo) de un nuevo medicamento. Este es obtenido si la acción del fármaco puede alcanzar el proceso bioquímico que debe modular, donde este fármaco es el principio activo de la reacción. El proceso de búsqueda de un medicamento es complejo debido al tiempo de demora (años de estudios) y a las pocas moléculas que llegan al final del ciclo con vistas a ser comercializadas.

Se pueden encontrar diversos programas para facilitar la gestión y el análisis de ensayos clínicos, los cuales hacen uso de procedimientos computacionales que permiten diseñar nuevos compuestos con

alta probabilidad de presentar la actividad biológica deseada.

Hoy en día existen aplicaciones que son de gran ayuda para los especialistas del sector farmacéutico, entre las que se encuentran: el Codessa [2], programa integral para el desarrollo cuantitativo de estructura/actividad (QSAR), mediante la integración de diversas herramientas computacionales y matemáticas que permite calcular una gran variedad de descriptores moleculares, lo que posibilita realizar análisis de agrupamiento de los datos experimentales y descriptores moleculares. Otra aplicación conocida es el Dragon, la misma calcula un gran número de descriptores, entre lo que se encuentran descriptores topológicos, índices de conectividad, descriptores geométricos, entre otros. Está Molconn-Z, que calcula más de 300 descriptores estructurales para su uso en métodos estadísticos con el fin de crear modelos QSAR para la predicción de la actividad biológica o propiedades físicas de nuevas moléculas. Todas estas aplicaciones generan una gran cantidad de información la cual no siempre se puede procesar con técnicas estadísticas [3]. Es por eso que surge la necesidad de utilizar otras vías como las técnicas de IA que han brindado un gran aporte en los problemas de clasificación, predicción y toma de decisiones.

Dentro de la IA existen diversas técnicas con el objetivo de encontrar mejores resultados, entre las que se encuentran las redes neuronales, las máquinas de soporte vectorial, los algoritmos genéticos (AGs) y los sistemas difusos. Existen herramientas computacionales que hacen uso de estas técnicas como son:

NeuralWorks Preditc [6]: permite crear y desplegar aplicaciones de predicción y clasificación. Combina la tecnología de redes neuronales con algoritmos genéticos, las estadísticas, la lógica difusa y encuentra automáticamente soluciones óptimas o casi óptimas para una amplia gama de problemas.

Discovery Studio y Materials Studio [7]: son colecciones de módulos integrados en un entorno unificado donde se crean, se visualizan y analizan modelos de sistemas químicos y biológicos. Estas colecciones de módulos ofrecen prestaciones en modelado molecular y simulación a nivel de experto desde entornos fáciles de usar y aprender, pero que no renuncian a la potencia y el rigor en los cálculos. Entre los métodos disponibles en Discovery Studio se incorpora el modelado molecular para crear modelos moleculares en tres dimensiones y emplearlos, por ejemplo, para ensayar sitios de unión de un receptor o para identificar nuevos candidatos a fármacos. Esta tecnología se aplica en la investigación farmacéutica para acelerar el proceso de descubrimiento y desarrollo de nuevos fármacos.

ADAPT (Automated Data Analysis using Pattern Recognition Toolkit): es un sistema de programas que le permite al usuario el desarrollo de relaciones estructura-actividad y estructura-propiedad. Brinda al usuario la facilidad de entrada gráfica y almacenamiento de estructuras moleculares y sus datos asociados, generación de estructuras 3D, cálculo de descriptores moleculares y análisis de estos con el empleo de estadística, reconocimiento de patrones o redes de neuronas para construir modelos predictivos. Posee una gran selección de rutinas generadoras de descriptores moleculares (topológicos, geométricos, electrónicos y físico-químicos). Los enfoques estadísticos incluyen regresión lineal múltiple, análisis clúster, discriminante y redes neuronales [3].

1.3 Lógica difusa.

Una de las disciplinas matemáticas con mayor número de seguidores actualmente es la llamada lógica difusa o borrosa, que es la lógica que permite tratar información imprecisa, como estatura media, temperatura baja o mucha fuerza, en términos de conjuntos borrosos o difusos (imprecisos), es decir, es aplicada a conceptos que pueden tomar un valor cualquiera de veracidad dentro de un conjunto de valores que oscilan entre dos extremos, la verdad absoluta y la falsedad total.

Estos conjuntos borrosos se combinan en reglas para definir acciones, como por ejemplo, si la temperatura es alta entonces enfría mucho. De esta manera, los sistemas de control basados en lógica borrosa combinan unas variables de entrada (definidas en términos de conjuntos borrosos), por medio de grupos de reglas que producen uno o varios valores de salida [8].

Las bases teóricas de la lógica borrosa fueron enunciadas en 1965 por el ingeniero Lotfi A. Zadeh, profesor de Ingeniería Eléctrica en la Universidad de California en Berkeley [8]. Zadeh introdujo el concepto de conjunto difuso (Fuzzy Set) bajo el que reside la idea de que los elementos sobre los que se construye el pensamiento humano no son números sino etiquetas lingüísticas. La lógica difusa permite representar el conocimiento común, que es mayoritariamente del tipo lingüístico cualitativo y no necesariamente cuantitativo, en un lenguaje matemático a través de la teoría de conjuntos difusos y funciones características asociadas a ellos. Permite trabajar a la vez con datos numéricos y términos lingüísticos; los términos lingüísticos son inherentemente menos precisos que los datos numéricos pero en muchas ocasiones aportan una información más útil para el razonamiento humano [9].
Con la ayuda de la lógica difusa se desarrolla esta investigación que cuenta con información referente a un grupo de moléculas, que viene dada por el conjunto de descriptores y la concentración de estados

asociada a cada molécula. De esta información se necesita conocer qué descriptores describen a la molécula como activa o inactiva. Para ello resulta útil las reglas del tipo SI-ENTONCES, generadas a través de los conjuntos difusos que se definan, las cuales por su fácil comprensión ayudan a verificar la veracidad de las predicciones realizadas.

1.3.1 Definiciones de los conceptos fundamentales.

En este epígrafe se explica brevemente el significado de algunos conceptos que serán utilizados en el desarrollo del trabajo con el propósito de lograr un buen entendimiento de las palabras técnicas a las que se hacen referencia.

Universo de discurso

Conjunto de valores numéricos que puede tomar para una variable discreta, o el rango de valores posibles para una variable continua. En el caso de la variable lingüística "altura", por ejemplo, podrían ser el conjunto de valores comprendido entre 1.5 y 2.3 m.

Conjunto o subconjunto difuso

Se encuentra asociado a un valor lingüístico, definido por una palabra, adjetivo o etiqueta lingüística A. Al mismo se le añade una función de pertenencia o inclusión.

Función de pertenencia o inclusión

La función de pertenencia o inclusión $\mu_A(x)$, definida como un número entre 0 y 1 (incluyéndolos a los dos), indica el grado en que la variable x está incluida en el concepto representado por la etiqueta A.

Variable lingüística

Se denomina variable lingüística a aquella que puede tomar por valor términos del lenguaje natural, como mucho, poco, positivo, negativo, etc., que son las palabras que desempeñan el papel de etiquetas en un conjunto borroso.

Reglas Difusas

Las reglas borrosas combinan uno o más conjuntos borrosos de entrada, llamados antecedentes o premisas, y les asocian un conjunto borroso de salida, llamado consecuente o consecuencia.

Los conjuntos borrosos de la premisa se asocian mediante conjuntivas lógicas como *y, o* fundamentalmente. Una regla típica, de tipo SI-ENTONCES, para un sistema de control seria "Si error es positivo-pequeño y derivada-de-error es negativo-pequeño entonces acción es positiva-pequeña", que se suele expresar abreviadamente mediante expresiones del tipo Si E es PP y DE es NP Entonces U es PP.

Base de Reglas

Conjunto de reglas que expresan las relaciones conocidas entre antecedentes y consecuentes.
Permiten expresar el conocimiento que se dispone sobre la relación entre antecedentes y consecuentes. Precisándose de varias reglas para expresar este conocimiento de forma completa [8].

1.3.2 Sistema Lógico Difuso (SLD).

Un sistema lógico difuso es un sistema experto en tiempo real, implementado a partir del conocimiento derivado de la experiencia de un operador o de un ingeniero de automatización [10]. Describen un conjunto de reglas con las cuales se forman las acciones que ejecutan el control. Estos sistemas son usados para aproximar descripciones de procesos que son muy complejos para modelarlos de manera precisa [10].

Etapas de un SLD

De manera general un sistema difuso se basa en tres etapas:
1-Fusificación. Término anglófono que se utiliza para convertir los valores nítidos en valores borrosos o difusos.
2-Reglas de Evaluación o Inferencia Difusa como también se conocen.
3-Defusificación. Término anglófono que se utiliza para convertir los valores difusos obtenidos en el proceso de inferencia, en valores nítidos.

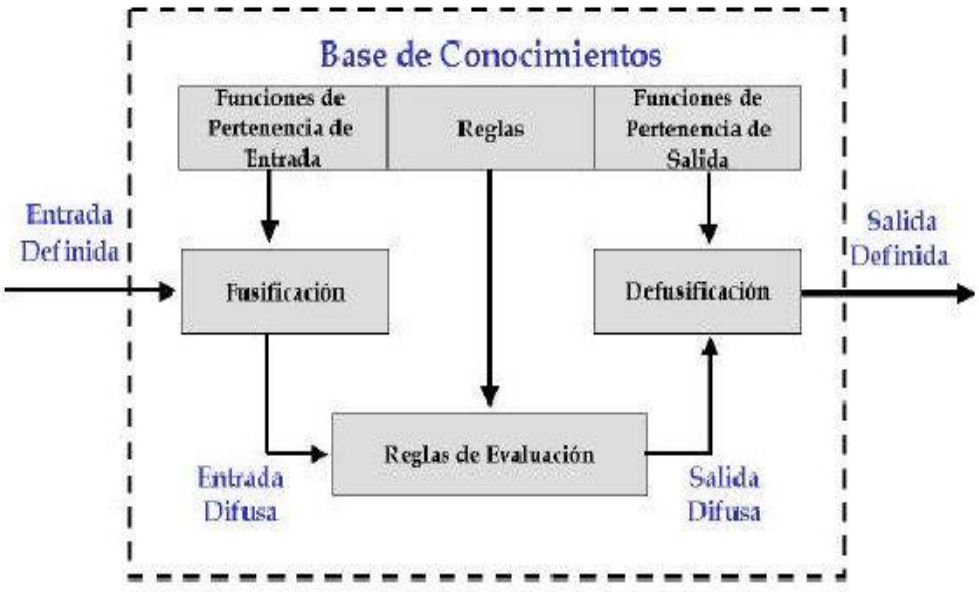


Figura #1 Representación gráfica de las etapas de un SLD.

A continuación se hace una descripción de cada una de estas fases.

Fusificación

Bloque en el que a cada variable de entrada se le asigna un grado de pertenencia a cada uno de los conjuntos difusos que se ha considerado, mediante las funciones características asociadas a estos conjuntos difusos. Las entradas a este bloque son valores concretos de las variables de entrada y las salidas son grados de pertenencia a los conjuntos difusos considerados [9].

Los elementos fundamentales en esta etapa son las Funciones de Pertenencia de Entrada. Existen gran cantidad de funciones de este tipo entre las que podrían citarse:

- Función de pertenencia puntual.
- Función Gamma: Escalones crecientes.
- Función L: Escalones decrecientes.
- Función LAMBDA: Funciones triangulares.
- Función Pi: Funciones trapezoidales.

Reglas de Evaluación

Bloque que mediante los mecanismos de inferencia, relaciona conjuntos difusos de entrada y de salida y que representa a las reglas que definen el sistema. Las entradas a este bloque son conjuntos difusos (grados de pertenencia) y las salidas son también conjuntos difusos, asociados a la variable de salida [9].

Las reglas son sentencias SI-ENTONCES que describen las condiciones (antecedentes) y las acciones (consecuentes) que deben existir para tomar una decisión. Estas reglas pueden ser obtenidas manualmente a partir de la experiencia del experto y los criterios que pueda emitir de la relación entre las variables de entrada y de salida o pueden generarse automáticamente a partir de la utilización de técnicas que logren obtener relaciones entre estas variables. La sintaxis de las reglas es la siguiente:
SI Antecedente 1 Y Antecedente 2. . . ENTONCES Consecuente 1 Y. . .

Mediante la inferencia los sistemas difusos interpretan las reglas de tipo SI-ENTONCES contenidas en su base de conocimientos, con el fin de obtener los valores de salida a partir de los valores que tienen las variables lingüísticas de entrada al sistema. Una vez que las entradas no difusas han sido convertidas a variables de valores lingüísticos (Fusificación), se utiliza la inferencia difusa para identificar las reglas de tipo SI-ENTONCES que se aplican a la situación actual y se calculan los valores lingüísticos de salida [3].

Defusificación

Bloque en el cual a partir del conjunto difuso obtenido en el mecanismo de inferencia y mediante los métodos matemáticos de desdifusión, se obtiene un valor concreto de la variable de salida, es decir, el resultado [9].

Algunas de las funciones utilizadas para realizar este proceso son:

- Centro de área.
- Media difusa ponderada.
- Máximo extremo izquierdo.
- Máximo extremo derecho.
- Centro de gravedad.

De las etapas de un SLD, descritas anteriormente, la obtención de reglas difusas puede hacerse con la utilización de algoritmos o técnicas que logren generar dichas reglas a partir de datos de entrada. En la actualidad existen sistemas que han logrado extraer conocimiento para una base de reglas difusas a partir de un conjunto de datos, por ejemplo en [3] se hace mención de varios sistemas y algoritmos que generan reglas borrosas a partir de conjuntos de datos.

Existen otras aplicaciones como Weka [11] y el RapidMiner (Yale) [12] que también son de gran ayuda en la extracción de conocimiento. Weka contiene herramientas necesarias para realizar transformaciones sobre los datos, tareas de clasificación, regresión, clustering, asociación, visualización. También permite generar reglas, pero estas reglas no son difusas, sino lineales, por lo que no se ajusta al problema que se pretende resolver. La herramienta RapidMiner, es el líder mundial de código abierto para la minería de datos debido a su combinación de su tecnología de primera calidad y su rango de funcionalidad. Además es una herramienta flexible para el aprendizaje y la exploración de la minería de datos, sin embargo el enfoque que utiliza para la generación de reglas en problemas de clasificación no es difuso, por lo que no se ajusta al problema que se plantea en este trabajo.

Otro ejemplo lo constituye KEEL (Knowledge Extraction based on Evolutionary Learning), herramienta de software para evaluar algoritmos evolutivos para problemas de minería de datos que incluye regresión, clasificación, agrupamiento y otros. Contiene una gran colección de algoritmos clásicos de extracción de conocimiento. Incluye además inteligencia computacional basada en algoritmos de aprendizaje evolutivos como son las reglas de aprendizaje en diferentes enfoques (el enfoque Pittsburgh, que consiste en que un conjunto de reglas es tratada como un individuo y el enfoque Michigan donde cada regla es tratada como un individuo, entre otros enfoques), y modelos híbridos como los sistemas difusos genéticos, las redes neuronales evolutivas, entre otros [13].

Dentro de los algoritmos que se encuentran en la herramienta KEEL [13], se hallan los basados en reglas difusas para clasificación. Un ejemplo lo constituyen: Grid Rule Base Generation and Genetic Rule Selection [14], Hybrid Fuzzy GBML [15] y Fuzzy Rule Learning Algorithm [16].

Todos ellos utilizan lógica difusa y algoritmo genético para la generación de reglas difusas, lo cual aporta buenos resultados en trabajos investigativos. El uso de algoritmo genético ayuda mucho en los problemas prácticos que presentan un espacio de soluciones enorme, imposible de explorar

exhaustivamente, ya que pueden explorar el espacio de soluciones en múltiples direcciones a la vez.

Debido a las características que presentan los algoritmos para problemas de clasificación que se encuentran en la herramienta KEEL [13], por ajustarse estos al problema que se desea resolver y por permitir la creación de reglas difusas a partir de valores continuos, se decide llevar a cabo su uso a través de uno de sus algoritmos.

1.4 Algoritmo Genético

Los AGs son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos.

Los AGs usan una analogía directa con el comportamiento natural. Trabajan con una población de individuos, cada uno de los cuales representa una solución factible a un problema dado. A cada individuo se le asigna un valor o puntuación, relacionado con la bondad de dicha solución. En la naturaleza esto equivaldría al grado de efectividad de un organismo para competir por unos determinados recursos. Cuanto mayor sea la adaptación de un individuo al problema, mayor será la probabilidad de que el mismo sea seleccionado para reproducirse, cruzando su material genético con otro individuo seleccionado de igual forma. Este cruce producirá nuevos individuos descendientes de los anteriores, los cuales comparten algunas de las características de sus padres.

Durante la fase reproductiva se seleccionan los individuos de la población para cruzarse y producir descendientes, que constituirán, una vez mutados, la siguiente generación. La selección de padres se efectúa al azar con el uso de un procedimiento que favorezca a los individuos mejor adaptados, ya que a cada organismo se le asigna una probabilidad de ser seleccionado que es proporcional a su función de adaptación. Este procedimiento se dice que está basado en la ruleta sesgada. Los individuos bien adaptados se escogerán probablemente varias veces por una generación, mientras que los pobremente adaptados al problema, no se escogerán más que de vez en cuando.

Una vez seleccionados dos padres, sus cromosomas se combinan, donde se utilizan habitualmente los operadores de cruce y mutación. Las formas básicas de dichos operadores se describen a continuación.

El operador de cruce, coge dos padres seleccionados y corta sus cadenas de cromosomas en una posición escogida al azar, para producir dos subcadenas iniciales y dos subcadenas finales. Después se intercambian las subcadenas finales, produciéndose dos nuevos cromosomas completos (véase la Figura #2). Ambos descendientes heredan genes de cada uno de los padres. Este operador se conoce como operador de cruce basado en un punto. Habitualmente el operador de cruce no se aplica a todos los pares de individuos que han sido seleccionados para emparejarse, sino que se aplica de manera aleatoria, normalmente con una probabilidad comprendida entre 0.5 y 1.0. En el caso en que el operador de cruce no se aplique, la descendencia se obtiene simplemente con la duplicación de los padres [17].

Figura #2 Operación de cruzamiento basado en un punto.

El operador de mutación se aplica a cada hijo de manera individual, y consiste en la alteración aleatoria (normalmente con probabilidad pequeña) de cada gen componente del cromosoma. La Figura #3 muestra la mutación del quinto gen del cromosoma. Si bien puede en principio pensarse que el operador de cruce es más importante que el operador de mutación, ya que proporciona una exploración rápida del espacio de búsqueda, éste último asegura que ningún punto del espacio de búsqueda tenga probabilidad cero de ser examinado, y es de capital importancia para asegurar la convergencia de los AGs.

Figura #3 Operación de mutación.

De esta manera se produce una nueva población de posibles soluciones, la cual reemplaza a la anterior y verifica la interesante propiedad de que contiene una mayor proporción de buenas

características en comparación con la población anterior. Favoreciendo el cruce de los individuos mejor adaptados, son exploradas las áreas más prometedoras del espacio de búsqueda. Si el algoritmo genético ha sido bien diseñado, la población convergerá hacia una solución óptima del problema.

El poder de los AGs proviene del hecho de que se trata de una técnica robusta, y pueden tratar con éxito una gran variedad de problemas provenientes de diferentes áreas, que incluye además aquellos en los que otros métodos encuentran dificultades. Si bien no se garantiza que el algoritmo genético encuentre la solución optima del problema, existe evidencia empírica de que se encuentran soluciones de un nivel aceptable, en un tiempo competitivo con el resto de algoritmos de optimización combinatoria. En el caso de que existan técnicas especializadas para resolver un determinado problema, lo más probable es que superen al Algoritmo Genético, tanto en rapidez como en eficacia. El gran campo de aplicación de los AGs se relaciona con aquellos problemas para los cuales no existen técnicas especializadas. Incluso en el caso en que dichas técnicas existan, y funcionen bien, pueden efectuarse mejoras de las mismas hibridándolas con los AGs [17].

1.5 Conclusiones

- Se implementará un algoritmo que permita predecir la relación estructura-actividad de compuestos orgánicos.
- Se utilizarán como técnica de IA la Lógica Difusa, ya que se está en presencia de información de entrada vaga, ambigua, imprecisa y Algoritmo Genético, para la generación de reglas.

CAPÍTULO 2: MATERIALES Y MÉTODOS

En este capítulo se describen las tecnologías y herramientas que se emplearon para la realización de la investigación. También se mencionan los aspectos esenciales que presenta el algoritmo seleccionado en su fundamentación matemática, además de presentar el pseudocódigo correspondiente. Además se describe el patrón estrategia como patrón de diseño debido a las ventajas que proporciona.

2.1 Tecnologías y herramientas

Para el desarrollo de la investigación se utilizan varios programas y herramientas que permiten el buen funcionamiento del trabajo a realizar. La elección de estas tecnologías se hace de acuerdo a las necesidades y del problema a resolver de la plataforma. A continuación se muestran algunas características de estas tecnologías y herramientas:

2.1.1 Lenguaje representativo (UML)

UML (Unified Modeling Language) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir.

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar.

Es un lenguaje de propósito general para el modelado orientado a objetos. UML es también un lenguaje de modelamiento visual que permite una abstracción del sistema y sus componentes [18].

2.1.2 Herramienta CASE (Visual Paradigm)

Visual Paradigm [19] es una herramienta CASE que utiliza “UML” como lenguaje de modelado y soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Se integra con las siguientes herramientas Java:

- Eclipse/IBM WebSphere
- JBuilder
- NetBeans IDE
- Oracle JDeveloper
- BEA Weblogic

Debido a que el sistema operativo que se utiliza es Ubuntu (distribución de Linux) se decidió utilizar el Visual Paradigm para visualizar y diseñar los elementos de software, debido a que es compatible en este sistema operativo.

2.1.3 Lenguaje de programación (Java)

Java es un lenguaje de computadora, diseñado por James Gosling en Sun Microsystems, se basa en C y C++. Es un lenguaje de programación que supone un significativo avance en el mundo de los entornos software, y esto viene avalado por tres elementos claves que diferencian a este lenguaje desde un punto de vista tecnológico:

- Es un lenguaje de programación que ofrece la potencia del diseño orientado a objetos con una sintaxis fácilmente accesible y un entorno robusto y agradable.
- Proporciona un conjunto de clases potente y flexible.
- Pone al alcance de cualquiera la utilización de aplicaciones que se pueden incluir directamente en páginas Web.

Java está diseñado para escribir software robusto, confiable, con una arquitectura neutra que funciona con uniformidad en una gran cantidad de tipos de computadoras. El polimorfismo y la encapsulación de datos son características intrínsecas de Java. Java soporta sincronización de múltiples hilos de ejecución a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas [20].

Debido a las características anteriores, por ser un lenguaje multiplataforma y poseer una gran cantidad de librerías útiles para la plataforma se escogió este lenguaje de programación.

2.1.4 Herramienta de desarrollo (Eclipse)

Eclipse es una plataforma desarrollada en Java y no está específicamente orientada solo al mundo de Linux, sino a cualquier tipo de sistema. Actualmente es la apuesta más fuerte, dado que se basa en Java y su organización es robusta y su desarrollo es continuo, además de fácilmente extensible, proporciona a la comunidad de desarrolladores un sistema de enganche de plug-ins muy adecuado para poder adaptar el entorno a las necesidades de desarrollo [21]. Su misión consiste en evitar tareas repetitivas, facilitar la escritura de código correcto, disminuir el tiempo de depuración e incrementar la productividad del desarrollador.

Debido a las características y ventajas mencionadas anteriormente y por el lenguaje de programación seleccionado para desarrollar la aplicación de prueba, se utiliza como herramienta de desarrollo el Eclipse.

2.2 Métodos

La lógica difusa ha sido principalmente aplicada a problemas de control para obtener automáticamente reglas difusas de información numérica. Esta técnica de IA está estrechamente vinculada con investigaciones científicas que manejan grandes volúmenes de datos con vista a la toma de decisiones. En la actualidad muchos sistemas usan dentro de su implementación la lógica difusa, con la perspectiva de lograr un mejoramiento en la obtención de los resultados. Con este propósito se escoge el algoritmo Fuzzy Rule Learning Algorithm [16] con el objetivo de lograr mejores resultados en la predicción de actividad biológica para la creación de medicamentos en la plataforma.

El algoritmo escogido puede ser visto como un sistema clasificador que genera reglas automáticamente para problemas de clasificación. La simplicidad de la implementación y de la interpretación lingüística de las reglas generadas son dos de las características principales de la elección de este algoritmo. El comportamiento del mismo ha sido evaluado para algunos problemas de prueba conocidos, como por ejemplo con las bases de datos del IRIS y WINE, las cuales se encuentran publicadas y han sido probadas mundialmente, se pueden encontrar en [22]. Además trabaja muy bien en comparación con otros métodos de clasificación tales como técnicas de aprendizaje no difusas y redes neuronales.

El funcionamiento del algoritmo está basado en un SLD (sección 1.3.2) donde cada paso del proceso

presenta características propias del mismo, las cuales serán mencionadas a continuación:

En la fusificación las variables de entrada se transforman en valores difusos, los cuales son calculados a través de la función de pertenencia triangular.

Para la Evaluación o Inferencia Difusa se emplean principios de lógica difusa en combinación con algoritmo genético para la generación de reglas difusas, a través de operaciones genéticas (sección 2.2.1) y un procedimiento heurístico (sección 2.2.2). Ello permite determinar qué relación existe entre las variables independientes (vector de entrada) y las dependientes (es la clasificación que presenta el vector de entrada).

En el proceso de defusificación se convierte la salida difusa del mecanismo de inferencia en una salida que pueda ser interpretada, para ello se utiliza la función matemática de los máximos [23].

2.2.1 Procedimiento del algoritmo genético

El algoritmo Ishibuchi-99 utiliza operaciones de AGs como selección, cruzamiento y mutación para generar una combinación de conjuntos antecedentes de las reglas difusas SI-ENTONCES. La salida a estas operaciones viene dada por los siguientes pasos (16):

Paso1: Generar una población inicial de reglas difusas SI-ENTONCES.
Paso2: Evaluar cada regla difusa en la población actual.
Paso3: Generar nuevas reglas difusas por las operaciones genéticas.
Paso4: Reemplazar una parte de la población actual con las nuevas reglas generadas.
Paso5: Terminar el algoritmo si una condición de parada es satisfecha, sino regresar al paso 2.

Inicialmente se tiene una población inicial de reglas difusas, la cual es creada aleatoriamente con la formación de los conjuntos antecedentes difusos por los símbolos correspondientes a los valores lingüísticos y la etiqueta “don´t care”, esta última brinda la posibilidad de reducir las reglas generadas. Cada símbolo es seleccionado con la probabilidad de 1/n (n es la cantidad de valores lingüísticos). La clase consecuente y el grado de certeza de cada regla difusa serán determinados por el procedimiento heurístico (sección 2.2.2). Luego en el proceso de evaluación se asignará una unidad en recompensa a la regla difusa ganadora cuando ésta clasifica correctamente con un vector, es decir, si la regla y el vector que se analizan presentan la misma clase entonces se le asignará un punto al valor del fitness

de la regla difusa, este valor se actualizará a medida que la regla clasifique correctamente un vector. Esta regla es escogida entre todas las reglas por medio de la fórmula #7, (sección 2.2.2).

De esta población inicial a través de la operación de selección se seleccionan primeramente un par de reglas. Cada regla es seleccionada por una probabilidad de selección basada en la rueda de la ruleta a través de la siguiente fórmula:

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

Fórmula #1

donde $fitness_{min}(S)$ es el menor valor de fitness del conjunto de reglas difusas de la población actual S y $fitness(R_j)$ es el valor del fitness de la regla difusa. El valor del fitness es usado para una evaluación en el proceso de selección.

Del par de reglas seleccionadas, dos reglas son generadas por la operación de cruzamiento para los conjuntos antecedentes difusos. En la siguiente figura se ilustra la operación de cruzamiento:

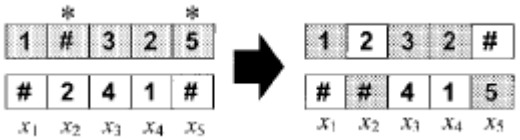


Figura #4 Cruzamiento uniforme para conjuntos difusos antecedentes (* denota una posición de cruzamiento).

Cada conjunto antecedente difuso de las reglas generadas por la operación de cruzamiento es aleatoriamente reemplazado por un conjunto antecedente difuso con el uso de una probabilidad pre-especificada de mutación. La operación de mutación es ilustrada en la siguiente figura:

Figura #5 Mutación para conjuntos difusos antecedentes (* denota una posición de mutación).

La operación de cruzamiento y mutación solo son aplicables a los conjuntos antecedentes difusos. La clase consecuente y el grado de certeza de las reglas nuevamente generadas son determinadas

después de la mutación por el procedimiento heurístico (sección 2.2.2). Tras obtener las nuevas reglas difusas estas reemplazarán aquellas de la población que posean menor valor fitness. La terminación de los pasos ocurrirá cuando se cumpla la condición de parada, que en este caso será la cantidad de generaciones.

2.2.2 Procedimiento heurístico

Cuando los conjuntos difusos antecedentes de cada regla difusa son determinados por la operación de algoritmo genético, se puede determinar la clase consecuente y el grado de certeza por el procedimiento heurístico. A continuación se describen los pasos según [16]:

Se cuenta con un conjunto de vectores reales *m*, de la forma $x_p = (x_{p1}, x_{p2}, \ldots, x_{pn})$,
p = 1,2,..., *m*, clasificados en *c* clases (*c* << *m*).
Las reglas generadas tendrán la forma:
Regla R_j: SI x_1 es A_{j1} y... y x_n es A_{jn} entonces clase C_j con CF= CF_j donde

R_j es la etiqueta de la *jth* regla difusa

$A_{j1}, \ldots\ldots, A_{jn}$ son los conjuntos difusos antecedentes de la regla

C_j es la clase consecuente de la regla

CF_j es el grado de certeza de la regla difusa R_j

Cada regla presenta también un valor de fitness que es usado en la operación de selección, este valor es diferente al grado de certeza, el cual es usado en el razonamiento difuso para la clasificación de nuevos patrones.
Para la determinación de C_j y CF_j se realizan las siguientes operaciones:

Paso1: Se calcula el grado de compatibilidad de cada vector con la regla difusa R_j por la operación del producto:

$$\mu_j(x_p) = \mu_{j1}(x_{p1}) * \ldots * \mu_{jn}(x_{pn})$$

Fórmula #2

donde $\mu_{ji}(x_{pi})$ es la función de pertenencia de A_{ji}.

Paso2: Para cada clase se calcula la suma de los grados de compatibilidad de los vectores con la regla difusa R_j:

$$\beta_{Class\ h}(R_j) = \sum_{x_p \in Class\ h} \mu_j(x_p), \quad h = 1, 2, \ldots, c$$

Fórmula #3

donde $\beta_{Class\ h}(R_j)$ es la suma de los grados de compatibilidad de los vectores de de la clase *h* con la regla difusa R_j.

Paso 3: Encontrar la clase $\tilde{h}_j$ que mayor valor de $\beta_{Class\ h}(R_j)$ tenga.

$$\beta_{Class\ \tilde{h}}(R_j) = \text{Max } \{ \beta_{Class\ 1}(R_j), \ldots, \beta_{Class\ c}(R_j)\}$$

Fórmula #4

Después de haber encontrado la clase de mayor valor de $\beta_{Class\ \tilde{h}}(R_j)$, esa sería la clase consecuente de la regla difusa.

Paso 4: El grado de certeza es determinado de la siguiente forma:

$$CF_j = \left\{ \beta_{Class\ \tilde{h}}(R_j) - \bar{\beta} \right\} \Big/ \sum_{h=1}^{c} \beta_{Class\ h}(R_j)$$

Fórmula #5

donde $\bar{\beta} = \sum_{\substack{h=1 \\ h \neq \tilde{h}_j}}^{c} \beta_{Class\ h}(R_j)/(c-1)$

Fórmula #6

Razonamiento difuso

Una vez que se cuenta con un conjunto de reglas que presentan una clase consecuente y un grado de certeza se lleva a cabo el proceso de clasificación de un vector de entrada por una regla perteneciente al conjunto de reglas, a través de un procedimiento de razonamiento difuso.

Este procedimiento es utilizado en el algoritmo para el proceso de evaluación de cada regla difusa de

la población actual.

El razonamiento difuso funciona por medio de la regla ganadora. La regla ganadora R_j para el vector x_p es determinado como:

$$\mu_j(x_p) * CF_j = Max\{\mu_j(x_p) * CF_j \mid R_j \in S\}$$

Fórmula #7

Por tanto, la regla ganadora tiene el máximo producto de la compatibilidad $\mu_j(x_p)$ y el grado de certeza CF_j. Con ello puede verse que solamente una regla es responsable del resultado de clasificación de cada vector de entrada.

2.2.3 Pseudocódigo del algoritmo Ishibuchi-99

Al realizar el estudio del basamento teórico del algoritmo Ishibuchi-99 [16], implementado por el grupo de desarrolladores de la herramienta informática KEEL [13], se comienza entonces con el análisis del código fuente de su implementación, para un mayor entendimiento de sus funcionalidades. Dentro de las clases implementadas, la fundamental es **Algorithm**, pues en ella se implementan los métodos que realizan las operaciones de algoritmo genético con la lógica difusa. A continuación se presenta el pseudocódigo de la misma.

```
public class Algorithm {
        .
        .
        .
    public void execute() {

     Calcular_particiones_difusas_iniciales ();

     //Genera la población inicial de reglas difusas
     Inicializa ();

     //Evalúa cada regla de la población actual con el aumento de su valor fitness
     Evalua();

    //fitness_mejor_pob: porciento de entrenamiento de la primera población
     fitness_mejor_pob = Porcentaje_Entrenamiento();
```

```
while ( numEvaluaciones < numGeneraciones) {
        while (hijo < numeroReemplazos) {
                Seleccion ();
                Cruce (Individuo Madre, Individuo Padre);
                Mutacion ();
                Obtener_clase_y_grado_de_certeza (Individuo indiv);
                hijo ++;
                Mutacion ();
                Obtener_clase_y_grado_de_certeza (Individuo indiv);
                 hijo ++;
    }
    Reemplazar_reglas ();
    Evalua ();
     fitness = Porcentaje_Entrenamiento();

     if (fitness > fitness_mejor_pob) {

fitness_mejor_pob = fitness;

MejorPoblacion = Poblacion;
          }
}
```

//Una vez terminada las generaciones, se continúa con lo siguiente:

```
Poblacion = MejorPoblacion;

Eliminar_reglas_fitness_cero();
```

```
        //Luego se produce la salida de dos ficheros con las particiones difusas  y el conjunto de reglas
        //con condicionales respectivamente.
        EscribeFichero(ruta del fichero de particiones difusas, texto);
        EscribeFichero(ruta del fichero del conjunto de reglas con condicionales,
                                texto);

        //Posteriormente se produce la salida de los ficheros de entrenamiento y
        //prueba con su respectiva predicción.
    }           .
                .
                .
}
```

El algoritmo para el cálculo del grado de compatibilidad utiliza la función de pertenencia triangular, la cual es representada de forma homogénea para cada partición difusa. Esta función se utiliza habitualmente en sistemas borrosos sencillos, pues permite definir un conjunto borroso con pocos datos, y calcular su valor de pertenencia con pocos cálculos. A continuación se muestran algunas de sus características:

Método Triangular

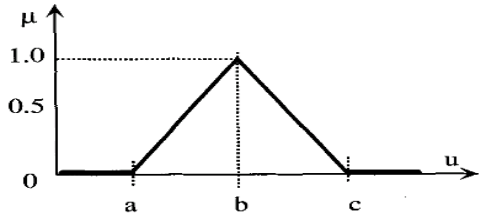


Figura #6 Función de pertenencia de tipo triangular.

La función de tipo triangular puede definirse como:

$$T(u;a,b,c) = \begin{cases} 0 & u < a \\ \dfrac{u-a}{b-a} & a \le u \le b \\ \dfrac{c-a}{c-b} & b \le u \le c \\ 0 & u > c \end{cases}$$

Fórmula #8

Esta función es adecuada para modelar propiedades con un valor de inclusión distinto de cero para un rango de valores estrecho en torno a un punto b [8].

2.3 Patrón de diseño

Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva y reusable. Se puede aplicar a diferentes problemas de diseño en distintas circunstancias. Facilitan el aprendizaje al programador inexperto, con el establecimiento de parejas problema-solución. Además ayudan a reutilizar código [24]. En este trabajo se utiliza como patrón de diseño el patrón estrategia [25] con vistas a mejorar la legibilidad del código del algoritmo seleccionado.

La estructura presentada por un Patrón Estrategia es la siguiente:

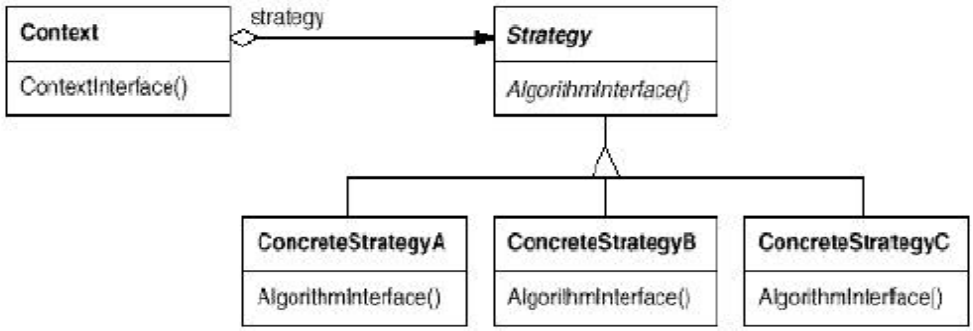


Figura #7 Estructura del Patrón Estrategia

Las clases que aparecen en la figura se explican aquí:

Contexto. Un objeto *Contexto* es un cliente que usa una operación de la interfaz ofrecida por *EstrategiaAbstracta*. Por tanto, mantiene una referencia a un objeto que es instancia de alguna clase de las derivadas por herencia de la clase *EstrategiaAbstracta*. *Contexto* desconoce qué clases implementarán la operación o cómo lo harán.

EstrategiaAbstracta. Esta clase define el comportamiento fundamental de las clases *Estrategia*. Al ser una clase abstracta o una interfaz, no pueden crearse instancias de ella. Para realizar operaciones, los clientes deben usar instancias de sus subclases concretas.

EstrategiaConcreta1, EstragegiaConcreta2, etc. Las instancias de estas clases especializan el comportamiento básico especificado por *EstrategiaAbstracta*. La especialización se consigue mediante

la implementación de las operaciones de *EstrategiaAbstracta* o la definición de otras nuevas. Los objetos cliente tienen referencias a instancias de estas clases.

Este patrón proporciona grandes ventajas:

- Permite definir familias de algoritmos relacionados, lo que posibilita agrupar funciones comunes y facilitar la reutilización del código.
- Es una cómoda alternativa a la subclasificación. Para compartir comportamientos suelen generarse subclases y redefinir los métodos comunes. Este patrón permite que el comportamiento cambie dinámicamente, en tiempo de ejecución.
- Elimina el uso de sentencias condicionales (*if*, *switch/case*), cuyo abuso hace difícil de leer el código. En lugar de comprobar dinámicamente que comportamiento hay que elegir, la elección se hace cuando se crea un objeto Estrategia.
- Permite al cliente elegir entre diversas implementaciones de una misma operación.

2.4 Conclusiones:

- Se presentan una serie de herramientas y tecnologías a utilizar durante el desarrollo de la investigación.
- Se describen las características del algoritmo seleccionado así como una breve explicación de la relación de algoritmo genético y lógica difusa que emplea para la generación de modelos difusos.
- Se presenta el pseudocódigo del algoritmo Ishibuchi-99, así como la función de pertenencia que emplea.
- Se describe el patrón estrategia con el objetivo de aprovechar sus ventajas para la implementación del algoritmo seleccionado.

CAPÍTULO 3: RESULTADOS Y DISCUSIÓN

En este capítulo se reflejan los principales resultados de este trabajo. Se realiza una descripción de las funcionalidades implementadas en el algoritmo seleccionado. Se muestran algunos diagramas relacionados con el diseño, que se proponen para la creación futura del producto relacionado, así como el uso del patrón estrategia que se utiliza en el desarrollo perspectivo de la implementación definitiva. Por último se presentan las pruebas realizadas en la creación de los modelos difusos y en el proceso de predicción, así como el estudio de clasificación realizado a un conjunto de muestras de cefalosporinas antibacteriales.

3.1 Modelo Conceptual

Un modelo conceptual es la representación gráfica de los principales conceptos en la representación del conocimiento. Permiten transmitir con claridad y organización los mensajes conceptuales, así como las relaciones entre los conceptos. En la figura #8 se muestra un mapa conceptual que representa una secuencia lógica de la propuesta de solución completa para el futuro desarrollo de la aplicación. Se evidencia la existencia de un administrador y un especialista, cada uno con una tarea bien definida, donde el administrador es el encargado de crear los modelos difusos con el uso de moléculas, las cuales están descritas a través de descriptores y poseen una concentración efectiva. Luego estos modelos generados serán utilizados como base de conocimiento para realizar la predicción por el especialista, el cual debe cargar además las moléculas a predecir. Es válido destacar que crear los modelos difusos es una actividad que se realiza independientemente de realizar predicción, pues primeramente se requiere crear los modelos difusos para poder realizar la predicción sobre las moléculas.

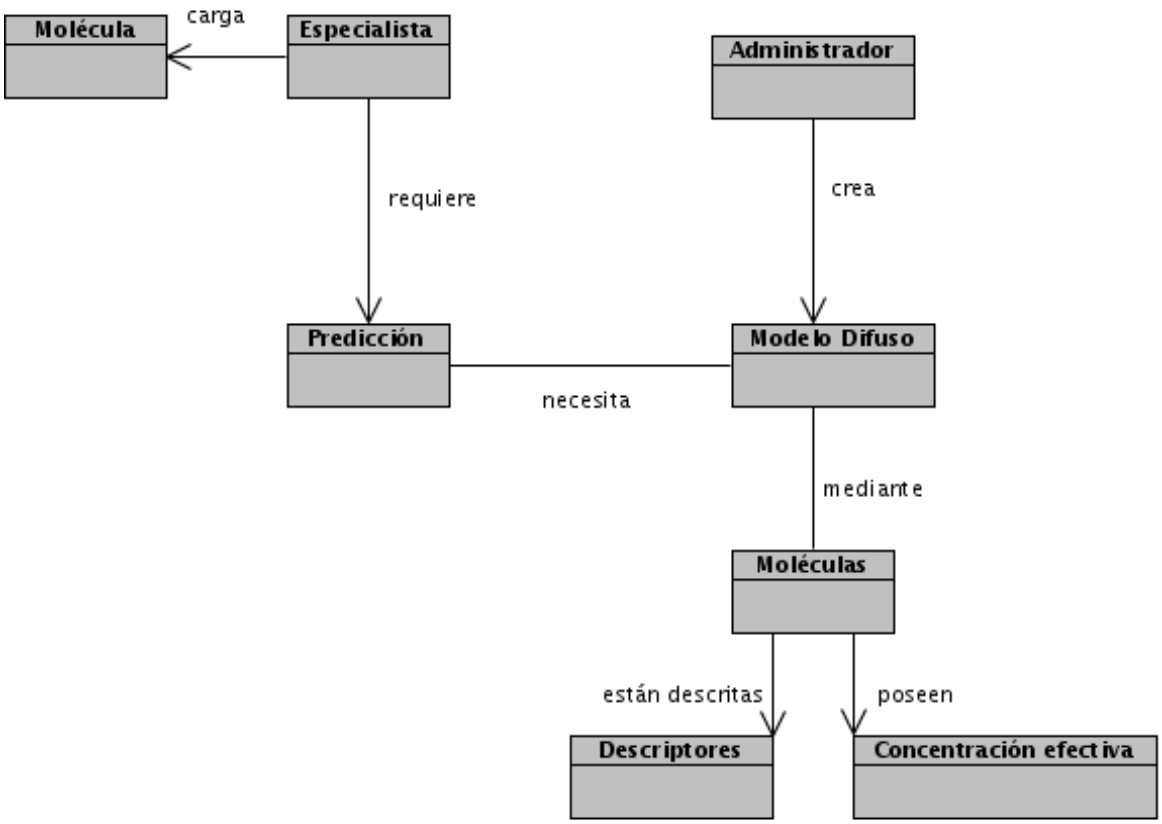


Figura #8 Modelo Conceptual

3.2 Patrón y diagrama de clases de la aplicación de prueba.

3.2.1 Patrón estrategia

Para la implementación del algoritmo Ishibuchi-99 [16] se usan tres funciones de pertenencia: Triangular (sección 2.2.3), Trapezoidal y Gauss (sección 3.3.3). Ello trae como consecuencia que la comprensión del código se dificultara un poco debido al empleo de varias condicionales. Por ello fue necesario buscar una solución a través de un patrón de diseño, donde fue escogido el Patrón Estrategia [25] por las ventajas (sección 2.3) que proporciona al código.

Estructura del Patrón estrategia aplicado al algoritmo

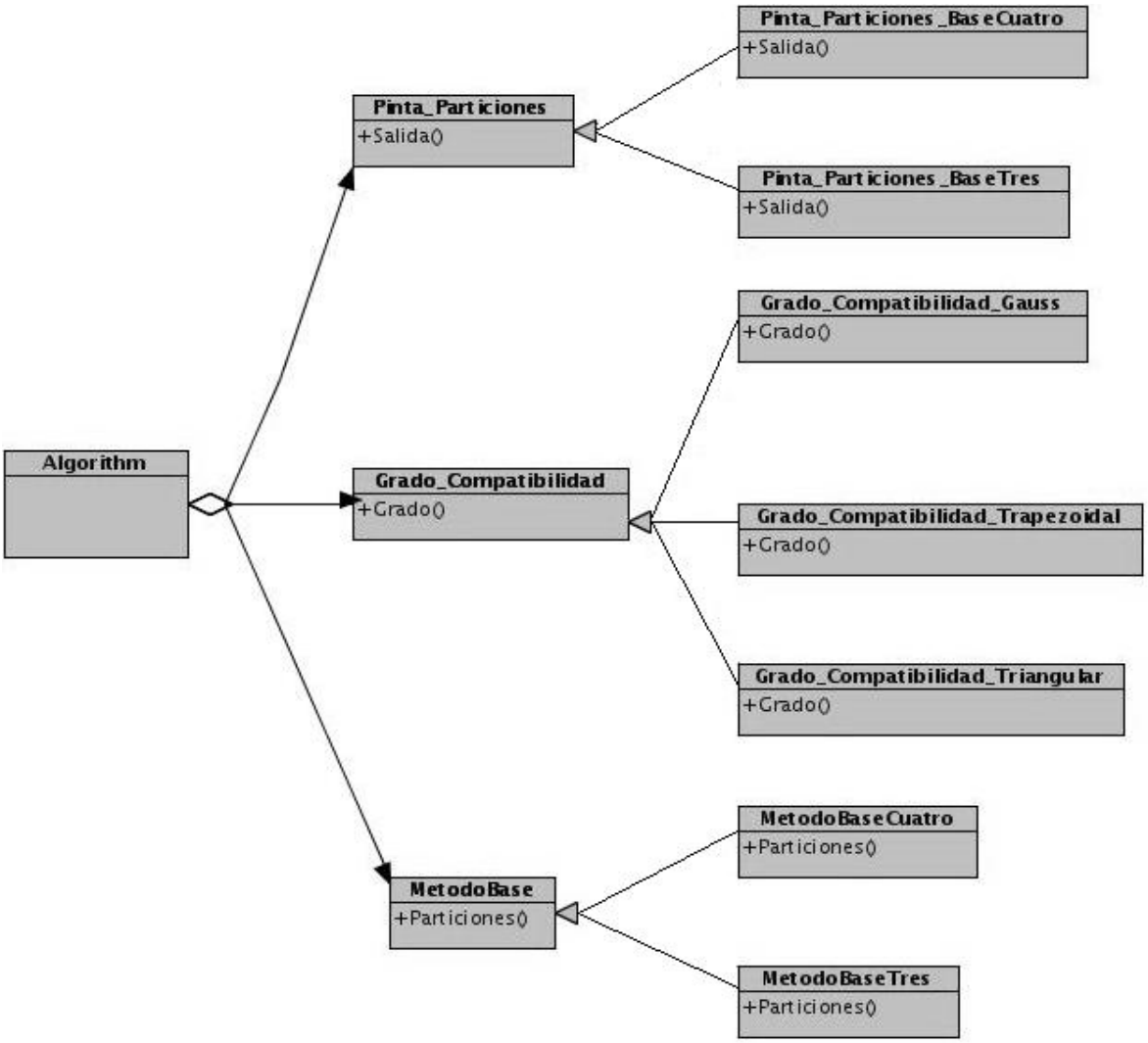


Figura #9 Patrón Estrategia para la creación de modelos difusos.

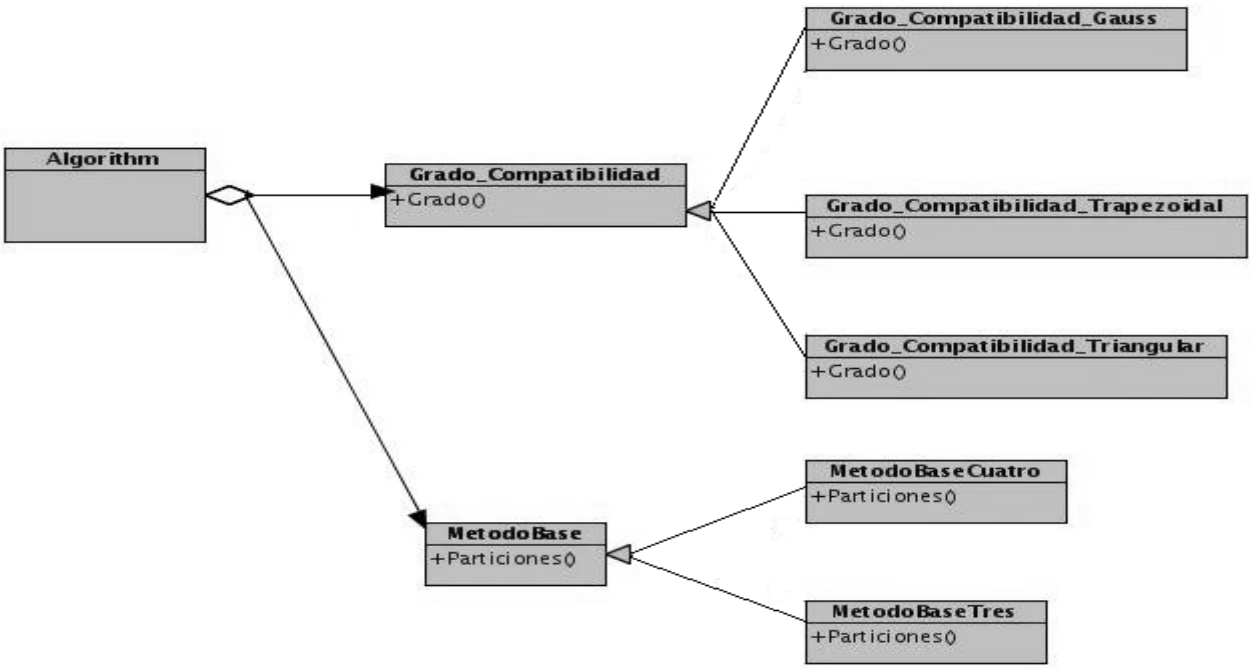


Figura #10 Patrón Estrategia para realizar la predicción

En las figuras 9 y 10 respectivamente se muestran las relaciones que existen entre las clases involucradas con el diseño del Patrón Estrategia. Ellas son Algorithm (clase contexto), Pinta_Particiones, Grado_Compatibilidad y MetodoBase (todas ellas constituyen las interfaces abstractas) y el resto constituyen las estrategias concretas que heredan de cada una de las interfaces abstractas.

3.2.2 Diagrama de clases de la aplicación de prueba

Un diagrama de clases es un tipo de diagrama que describe la estructura necesaria para la implementación del sistema. Esto se muestra gráficamente a través de las relaciones entre sus clases y atributos, todo esto teniendo en cuenta las librerías, lenguaje de programación, entre otros. Para el diagrama de clases de la funcionalidad Crear Modelo Difuso ver Anexo #1 y para Realizar Predicción ver Anexo #2.

3.3 Modificación en la implementación del algoritmo Ishibushi-99

Una vez presentado el modelo conceptual (sección 3.1) del problema a resolver se observa que las funcionalidades crear modelos difusos y realizar predicción son situaciones tratadas de manera diferente. Sin embargo en el pseudocódigo de Ishibuchi-99 que se muestra en la sección 2.2.3 se aprecia que tanto la funcionalidad crear los modelos difusos como realizar predicción se encuentran en una misma clase. Esto resultaba un inconveniente para la resolución del problema en cuestión, ya que el mismo debía dividirse en dos momentos. Un primer momento sería la creación de los modelos difusos y un segundo momento sería realizar la predicción con la utilización de los modelos difusos generados. Una vez hecho el análisis de cómo dividir el código y los nuevos métodos que se adicionarían se comenzó la implementación de ambos códigos. Es válido aclarar que las modificaciones que aquí se explican son realizadas sobre la implementación del algoritmo Ishibuchi-99 [16] necesarias para la investigación, pero no afectan los pasos que se siguen en su fundamentación teórica.

3.3.1 Pseudocódigo para crear modelos difusos

Los modelos difusos o también llamados base de conocimientos son el conjunto de reglas generadas a partir de datos de entrenamientos que constituyen las variables de entrada. En la aplicación de prueba estos modelos difusos son generados en ficheros.

A continuación se muestra el pseudocódigo para la creación de los modelos difusos:

```
public class Algorithm {
        .
        .
        .
    public void execute() {

     Calcular_particiones_difusas_iniciales ();

     //Genera la población inicial de reglas difusas
     Inicializa ();

     //Evalúa cada regla de la población actual con el aumento de su valor fitness
     Evalua();

    //fitness_mejor_pob: porciento de entrenamiento de la primera población
```

```
fitness_mejor_pob = Porcentaje_Entrenamiento();

   while ( numEvaluaciones < numGeneraciones) {

        while (hijo < numeroReemplazos) {

                Seleccion ();

                Cruce (Individuo Madre, Individuo Padre);

                Mutacion ();

                Obtener_clase_y_grado_de_certeza (Individuo indiv);

                hijo ++;
                Mutacion ();

                Obtener_clase_y_grado_de_certeza (Individuo indiv);

                 hijo ++;
   }

   Reemplazar_reglas ();

   Evalua ();

    fitness = Porcentaje_Entrenamiento();


    if (fitness > fitness_mejor_pob) {

fitness_mejor_pob = fitness;

MejorPoblacion = Poblacion;

        }

}
```

//Una vez terminada las generaciones, se continúa con lo siguiente:

```
Poblacion = MejorPoblacion;
Eliminar_reglas_fitness_cero();
```

//Este método es una nueva funcionalidad en este código.

```
EliminarReglasRepetidas();
```

```
        //Luego se produce la salida de dos ficheros con las particiones difusas  y el
        //conjunto de reglas con condicionales respectivamente.
        EscribeFichero(ruta del fichero de particiones difusas, texto);
        EscribeFichero(ruta del fichero del conjunto de reglas con condicionales,
                        texto);

        //Este método es una nueva funcionalidad en este código.
        EscribeFichero(ruta del fichero del conjunto de reglas sin condicionales,
                        texto);
    }
                .
                .
                .
}
```

A pesar de presentar muchos de los métodos de la implementación original puede verse que se crean otros nuevos:

EliminarReglasRepetidas(): Este método permitirá eliminar las reglas que se repiten en la población final, es decir, aquellas reglas que presenten el mismo grado de certeza y la misma clase.

EscribeFichero(ruta del fichero del conjunto de reglas sin condicionales, texto): Este método guardará el nombre de la función de pertenencia que se utilizó al crear el modelo difuso, la cantidad de variables lingüísticas, el nombre de las clases que se utilizaron y el conjunto de reglas sin condicionales. Esta información se utilizará en la predicción de las moléculas.

Luego para la ejecución de la funcionalidad Crear Modelo Difuso es necesario cargar un fichero con los datos de entrenamiento, además se procede a la entrada de varios parámetros, datos que son imprescindibles para las operaciones del algoritmo. Estos parámetros son:

Semilla: Valor para la generación de los números aleatorios.
Número de etiquetas: Es la cantidad de valores lingüísticos a utilizar.
Población inicial: Es el tamaño de la población inicial.
Número de evaluaciones: Representa la cantidad de generaciones. Es la condición de parada del algoritmo.

Número de reemplazos: Cantidad de reglas a ser reemplazadas en la población.
Probabilidad de cruzamiento: Valor utilizado para la operación de cruzamiento.
Probabilidad de mutación: Valor utilizado para la operación de mutación.
Probabilidad de Don't care: Representa la probabilidad de ser escogida esta etiqueta para formar parte del antecedente de la regla.

Los resultados serán guardados en los tres ficheros de salida que son creados por medio de los métodos **EscribeFichero (…)**.

3.3.2 Pseudocódigo para realizar predicción

Una vez que se cuenten con los modelos difusos se está en condiciones de realizar la predicción de los nuevos datos que están sin clasificar. Seguidamente se muestra el pseudocódigo que permite realizar la predicción de dichos valores:

```
public class Algorithm {
                .
                .
                .
        public void execute() {

        Lectura_Fichero(ruta, entradas);

        Inicializar_Campos();

        Calcular_particiones_difusas_iniciales();

        // Por último se realiza la salida del fichero con los datos y su predicción.
        EscribeFichero(ruta del fichero con los datos predichos, texto);
       }
                .
                .
                .
  }
```

Excepto **Calcular_particiones_difusas_iniciales() y EscribeFichero(…)** los otros dos métodos que se encuentran contenido dentro de **execute()** son nuevos para esta parte del algoritmo:

Lectura_Fichero(ruta, entradas): El mismo permitirá leer todo lo que se encuentra en el fichero de las reglas sin condicionales que estará ubicado en la **ruta** que se especifique, además de entrarle la cantidad de variables lingüísticas(**entradas**) presentes en la muestra.

Inicializar_Campos(): Una vez realizada la lectura del fichero anterior este método inicializará cada uno de los parámetros que son necesarios para la ejecución del algoritmo.

Para su ejecución se leerá un fichero que contiene el conjunto de reglas difusas generadas por la funcionalidad Crear Modelo Difuso y un fichero con las moléculas traídas por el especialista, con los cuales se realiza la predicción. A continuación por medio de la función de los máximos [23] comienza el proceso de defusificación, cuyos resultados son guardados en un fichero y así concluye el proceso de predicción de las moléculas.

3.3.3 Características de las funciones de pertenencia implementadas

Para el desarrollo del algoritmo se implementaron dos funciones de pertenencia: función Trapezoidal y Gauss, cuyas características se muestran a continuación:

Método de Gauss

Figura #11: Función de pertenencia de tipo Gauss

$$A(x) = e^{-k(x-m)^2}$$

Fórmula #9

Esta función tiene forma de campana, y resulta adecuada para los conjuntos definidos en torno a un valor *m*, como medio, normal, cero.

La implementación de esta función en el código queda de la siguiente manera:

```
//Son los puntos bases de la partición difusa gaussiana
x0, x1, x2;
double k = (x2-x0)/2;
//valor_ejemplo guarda el valor al que se le calculará el grado de compatibilidad. La variable grado guarda el
//grado de compatibilidad del valor_ejemplo con respecto a la función de pertenencia.
if ((valor_ejemplo > x0) && (valor_ejemplo < x2)) {
            grado = Math.pow((valor_ejemplo - x1), 2);
            grado = Math.pow(Math.E, -k*grado);
          } else {
   grado = 0.0;
}
```

Método Trapezoidal

La función de tipo trapezoidal se define por cuatro puntos *a*, *b*, *c*, *d*. Esta función es cero para valores menores de *a* y mayores que *d*, vale uno entre *b* y *c*, y toma valores en [0,1] entre *a* y *b*, y entre *c* y *d* [8].

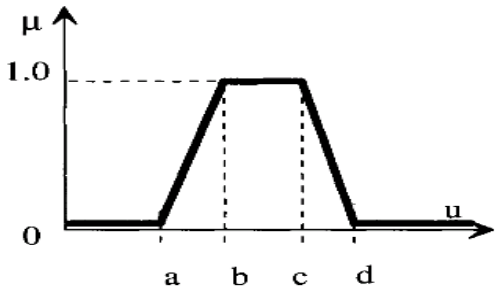


Figura #12 Función de pertenencia de tipo trapezoidal.

Se define con:

$$S(u;a,b,c,d)=\begin{cases}0 & u<a\\ \left(\frac{u-a}{b-a}\right) & a\le u\le b\\ 1 & b\le u\le c\\ \left(\frac{d-u}{d-c}\right) & c\le u\le d\\ 0 & u>d\end{cases}$$

Fórmula #10

La implementación de esta función en el código queda de la siguiente manera:

```
//Son los puntos bases de la partición difusa trapezoidal
x0, x1, x2 ,x3;
//valor_ejemplo guarda el valor al que se le calculará el grado de compatibilidad. La variable grado guarda el
//grado de compatibilidad del valor_ejemplo con respecto a la función de pertenencia.
if ((valor_ejemplo > x0) && (valor_ejemplo < x3))
        if (valor_ejemplo < x1) {
                grado = ((valor_ejemplo - x0) / (x1 - x0));
        } else {
                if (valor_ejemplo >= x1 && (valor_ejemplo <= x2)) {
                   grado = 1.0;
                } else {
                   grado = ((x3 - valor_ejemplo) / (x3 - x2));
                   }
        }
} else {
        grado = 0.0
}
```

Para el funcionamiento del algoritmo se escogieron las funciones de pertenencia trapezoidal y gauss pues son las funciones básicas que más se usan en los sitemas difusos de predicción.

3.3.4 Implementación con el empleo del Patrón Estrategia

Una vez analizados los pseudocódigos de Crear Modelo Difuso y Realizar Predicción y la codificación de las funciones de pertenencia Gauss y Trapezoidal, se procede a la implementación del patrón estrategia siguiendo lo planteado en la sección 3.2.1. A continuación se describen brevemente las tres interfaces que aparecen en las figuras 9 y 10 (sección 3.2.1) como parte del patrón estrategia.

Interfaz abstracta **MetodoBase**

```
public interface MetodoBase {
        public double[] Particiones(int numEtiquetas, int entradas,
                                        double[] maximo,double[] minimo);
}
```

Esta interfaz presenta el método Particiones (...), el cual será redefinido en las clases que constituyen las estrategias concretas: MetodoBaseTres, MetodoBaseCuatro. El método devolverá un arreglo con los valores de los intervalos de los conjuntos difusos, en dependencia de los puntos bases de la función de pertenencia.

Interfaz abstracta **Grado_Compatibilidad**

```
public interface Grado_Compatibilidad {
        public double Grado(double[] arreglo, Individuo individuo,
                        double[] ejemplo, int numEtiquetas, int entradas);
}
```

Esta interfaz presenta el método Grado (...), el cual será redefinido en las clases que constituyen las estrategias concretas: Grado_Compatibilidad_Triangular, Grado_Compatibilidad_Gauss y Grado_Compatibilidad_Trapezoidal. El método devolverá un valor que será el grado de compatibilidad de los valores que conforman el vector con la regla difusa, en dependencia de la función de pertenencia. Este cálculo se realiza a través de la fórmula #2 (sección 2.2.2).

Interfaz abstracta **Pinta_Particiones**

```
public interface Pinta_Particiones {
        public String Salida(int entradas, int numEtiquetas, double[] arreglo);
}
```

Esta interfaz presenta el método Salida (...), el cual será redefinido en las clases que constituyen las estrategias concretas: Pinta_Particiones_BaseCuatro y Pinta_Particiones_BaseTres. El método devolverá una cadena con la representación de los conjuntos difusos por cada variable lingüística, en dependencia de la función de pertenencia que se utilice.

3.4 Pruebas

En la presente investigación se realizaron diferentes pruebas con el objetivo de validar la calidad de los modelos creados por la aplicación de prueba, además de ver la capacidad de la combinación de los principios de la lógica difusa con técnicas de algoritmo genético para la clasificación de los compuestos orgánicos. Para la realización de las pruebas se utilizaron diferentes ensayos, el primero con 150 vectores de entrada perteneciente a la base de datos IRIS, base de datos que presenta una compilación de informes electrónicos sobre sustancias que se encuentran en el medio ambiente y su potencial de causar efectos sobre la salud humana y se encuentra publicada en [22], el segundo ensayo de tamaño 178 perteneciente a la base de datos WINE, contiene los resultados de un análisis químico de los vinos cultivados y una tercera muestra de cefalosporinas [26], que no son más que compuestos antibacteriales que poseen un amplio rango de actividad con una excelente tolerancia en niños.

3.4.1 Pruebas de entrenamiento para la muestra 1

Para las pruebas realizadas con la base de datos IRIS se utilizaron 4 variables de entrada para una muestra de 150 elementos. Se le aplicó la técnica de validación cruzada con la cual se generaron 10 ficheros de entrenamientos, con 135 vectores de entrada, y sus respectivos ficheros de prueba, con 15 vectores de entrada. Para la creación de los modelos difusos los valores pasados a los parámetros que se exponen en la sección 3.3.1 son los siguientes:

Semilla: 12345678

Número de etiquetas: 5

Población inicial: 100

Número de evaluaciones: 10 000

Número de reemplazos: 20

Probabilidad de cruzamiento: 1.0

Probabilidad de mutación: 0.1

Probabilidad de Don't care: 0.5

Al generar los modelos difusos, con la utilización de las tres funciones de pertenencia se obtuvieron los siguientes resultados para los ficheros de entrenamiento:

Ficheros de entrenamiento	Clases	% acierto (Triangular)	% acierto (Gauss)	% acierto (Trapezoidal)
iris-10-1tra	3	93.33	89.23	97.78
iris-10-2tra	3	94.07	89.63	97.04
iris-10-3tra	3	94.07	90.37	97.04
iris-10-4tra	3	95.56	88.89	97.78
iris-10-5tra	3	95.56	89.63	97.78
iris-10-6tra	3	94.81	88.15	96.30
iris-10-7tra	3	93.33	89.63	97.04
iris-10-8tra	3	94.07	89.63	97.04
iris-10-9tra	3	94.07	88.89	97.04
iris-10-10tra	3	94.07	89.63	97.78

Tabla #1 Resultados en la creación de los modelos difusos con la muestra de la base de datos IRIS.

A continuación se muestra un gráfico que representa los resultados expuestos anteriormente:

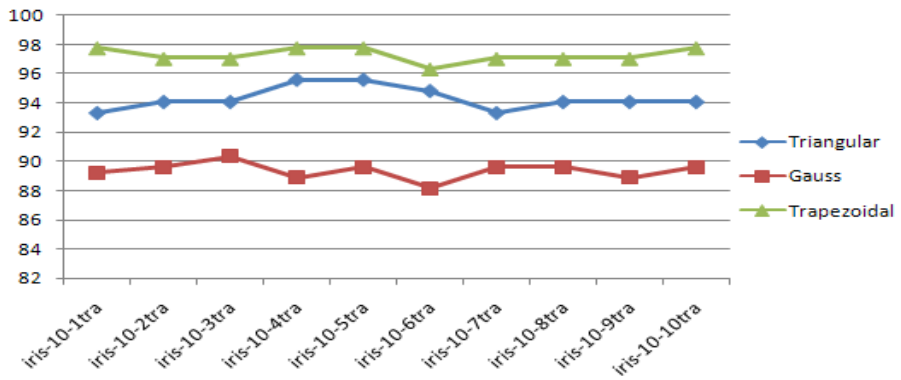


Figura #13 Gráfica con los resultados de la creación de modelos difusos para el ensayo IRIS.

Se puede observar que al generar los modelos difusos se obtuvieron resultados con un buen porcentaje, obteniéndose 95.56 como mayor porciento para la función de pertenencia triangular, 90.37% para la función de gauss y 97.78% para la función trapezoidal. Se observa además que los mejores porcientos se obtienen con la función trapezoidal.

3.4.2 Pruebas de entrenamiento para la muestra 2

Las pruebas realizadas con la base de datos WINE se realizaron con 13 variables de entrada con una muestra de tamaño 178. Se le aplicó la técnica de validación cruzada con la cual se generaron 10 ficheros de entrenamiento, con 160 vectores de entrada, y sus respectivos ficheros de prueba, con 18 vectores de entrada. Para la creación de los modelos difusos los valores pasados a los parámetros que se exponen en la sección 3.3.1 son los siguientes:

Semilla: 12345678

Número de etiquetas: 3

Población inicial: 100

Número de evaluaciones: 10 000

Número de reemplazos: 20

Probabilidad de cruzamiento: 1.0

Probabilidad de mutación: 0.1

Probabilidad de Don't care: 0.5

Al generar los modelos difusos, con la utilización de las tres funciones de pertenencia se obtuvieron los siguientes resultados para los ficheros de entrenamiento:

Ficheros de entrenamiento	Clases	% acierto (Triangular)	% acierto (Gauss)	% acierto (Trapezoidal)
wine-10-1tra	3	96.88	92.5	98.13
wine-10-2tra	3	96.25	89.38	96.25
wine-10-3tra	3	96.25	90.0	96.25
wine-10-4tra	3	96.88	93.13	96.88
wine-10-5tra	3	96.25	90.0	98.13
wine-10-6tra	3	96.25	89.38	98.75
wine-10-7tra	3	94.38	93.13	96.88
wine-10-8tra	3	96.88	92.5	98.13
wine-10-9tra	3	96.89	93.14	95.65
wine-10-10tra	3	96.27	91.30	96.27

Tabla #2 Resultados en la creación de los modelos difusos con la muestra de la base de datos WINE.

A continuación se muestra un gráfico que representa los resultados expuestos anteriormente:

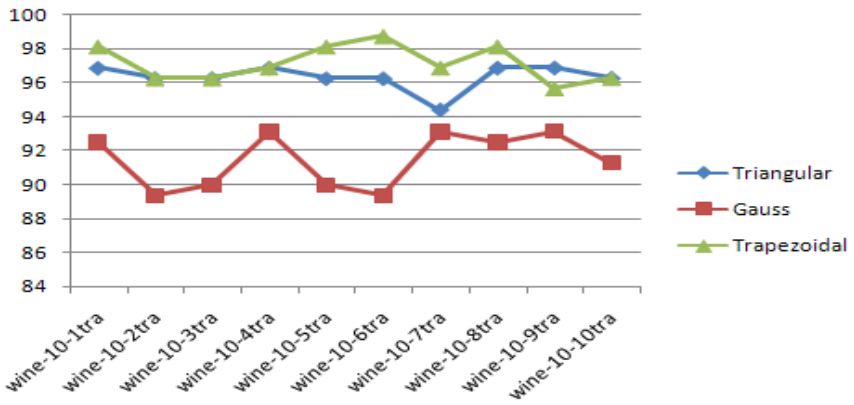


Figura #14 Gráfica con los resultados de la creación de modelos difusos para el ensayo WINE.

Se puede observar que al generar los modelos difusos se obtuvieron resultados con un buen porcentaje, obteniéndose 96.89 como mayor porciento para la función de pertenencia triangular, 93.14% para la función de gauss y 98.75% para la función trapezoidal. Se observa además que los mejores porcientos se obtienen, al igual que con la muestra de IRIS, con la función trapezoidal.

3.4.3 Estudio de clasificación de cefalosporinas

Producto a los resultados obtenidos con las bases de datos IRIS y WINE se decidió realizar un estudio de clasificación de una muestra de 104 cefalosporinas reportadas como activas [26].

Esta muestra fue utilizada en trabajos anteriores para desarrollar un modelo de relación estructura química-actividad a través de un análisis de regresión múltiple. En esta investigación se decidió incrementar el número de variables de entrada y modificar su naturaleza teniendo en cuenta que el problema es de clasificación y no cuantitativo. La muestra cuenta con un total de 11 variables de entrada, a la cual se le aplicó la técnica de validación cruzada donde se generaron 10 ficheros de entrenamiento, con 93 vectores de entrada, y sus respectivos ficheros de prueba, con 11 vectores de entrada. Para la creación de los modelos difusos los valores pasados a los parámetros que se exponen en la sección 3.3.1 son los siguientes:

Semilla: 12345678
Número de etiquetas: 2
Población inicial: 100

Número de evaluaciones: 10 000

Número de reemplazos: 20

Probabilidad de cruzamiento: 1.0

Probabilidad de mutación: 0.1

Probabilidad de Don't care: 0.5

Al generar los modelos difusos, con la utilización de las tres funciones de pertenencia se obtuvieron los siguientes resultados para los ficheros de entrenamiento:

Ficheros de entrenamiento	Clases	% acierto (Triangular)	% acierto (Gauss)	% acierto (Trapezoidal)
potec-10-1tra	2	83,87	88,17	82,8
potec-10-2tra	2	82,8	82,8	81,72
potec-10-3tra	2	83,87	83,87	81,72
potec-10-4tra	2	80,65	79,57	80,65
potec-10-5tra	2	84,04	82,99	84,04
potec-10-6tra	2	78,72	81,91	82,99
potec-10-7tra	2	82,99	81,91	80,85
potec-10-8tra	2	84,04	82,98	82,98
potec-10-9tra	2	80,85	84,04	81,91
potec-10-10tra	2	82,98	81,91	80,85

Tabla #3 Resultados en la creación de los modelos difusos con la muestra de cefalosporinas.

A continuación se muestra un gráfico que representa los resultados expuestos anteriormente:

Figura #15 Gráfica con los resultados de la creación de modelos difusos de la muestra de cefalosporinas.

Puede observarse que los valores obtenidos oscilan entre los 79 y 88 porcientos. Entre los factores que afectan estos resultados pueden citarse el hecho de la clasificación inicial de los datos, la cual fue a criterio del especialista por medio del criterio de la mediana, por otro lado está el hecho de que las cefalosporinas son compuestos difíciles de trabajar debido a su compleja estructura química, lo cual está condicionado por los diferentes parámetros que le dan la propiedad de su actividad [26].

3.4.4 Pruebas de predicción

En esta sección se mostrarán los resultados obtenidos en las pruebas de predicción realizadas con los ficheros de pruebas mencionados anteriormente. Estos resultados van a estar en correspondencia con el modelo difuso utilizado, es decir, dependerá del modelo difuso obtenido con el fichero de entrenamiento.

Para los ficheros de prueba de la base de datos IRIS se obtuvieron los siguientes resultados:

Ficheros de prueba	Clases	Cant. Atributos	Cant. de datos	Datos mal clasificados (Triangular)	Datos mal clasificados (Gauss)	Datos mal clasificados (Trapezoidal)
iris-10-1test	3	4	15	0	2	0
iris-10-2test	3	4	15	1	2	1
iris-10-3test	3	4	15	1	3	0
iris-10-4test	3	4	15	3	1	1
iris-10-5test	3	4	15	1	2	1
iris-10-6test	3	4	15	0	0	0
iris-10-7test	3	4	15	0	2	1
iris-10-8test	3	4	15	1	2	1
iris-10-9test	3	4	15	1	1	0
iris-10-10test	3	4	15	1	1	0

Tabla #4 Resultados de la predicción con la base de datos IRIS.

A continuación se muestra un gráfico que representa los resultados expuestos anteriormente:

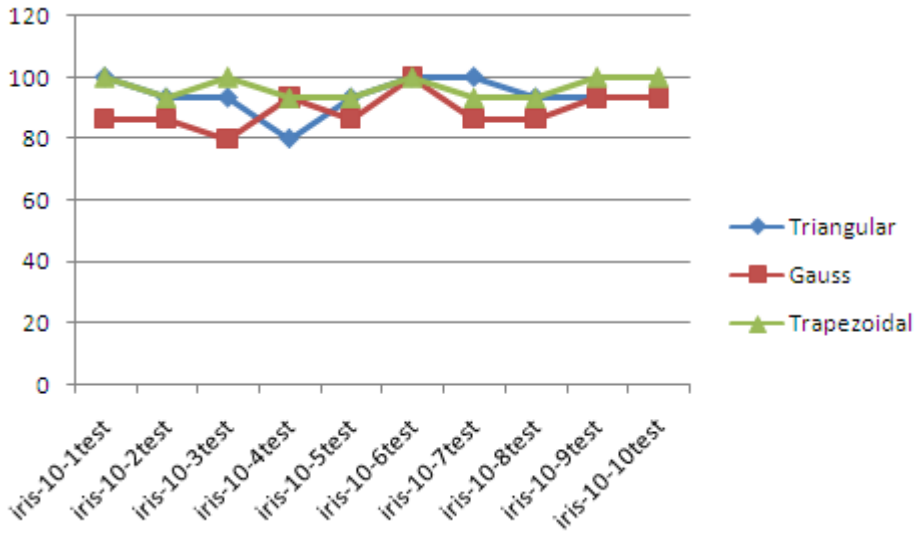


Figura #16 Gráfica de los resultados en la predicción del ensayo IRIS.

Puede observarse que se obtuvo un buen resultado en la predicción con las funciones de pertenencia Triangular y Trapezoidal, mientras que decae el resultado con la función de pertenencia gaussiana. Resalta en los datos de la tabla que la función trapezoidal fue la que mejor clasificación realizó pues de los 10 ficheros de pruebas en 5 se obtuvo 100% de buena predicción.

Para los ficheros de prueba de la base de datos WINE se obtuvieron los siguientes resultados:

Ficheros de prueba	Clases	Cant. Atributos	Cant. de datos	Datos mal clasificados (Triangular)	Datos mal Clasificados (Gauss)	Datos mal clasificados (Trapezoidal)
wine-10-1test	3	13	18	0	3	0
wine-10-2test	3	13	18	1	3	0
wine-10-3test	3	13	18	1	3	0
wine-10-4test	3	13	18	1	2	1
wine-10-5test	3	13	18	2	6	1
wine-10-6test	3	13	18	2	4	1
wine-10-7test	3	13	18	0	2	0
wine-10-8test	3	13	18	0	3	1
wine-10-9test	3	13	18	1	9	0
wine-10-10test	3	13	18	1	2	0

Tabla #5 Resultados de la predicción con la base de datos WINE.

A continuación se muestra un gráfico que representa los resultados expuestos anteriormente:

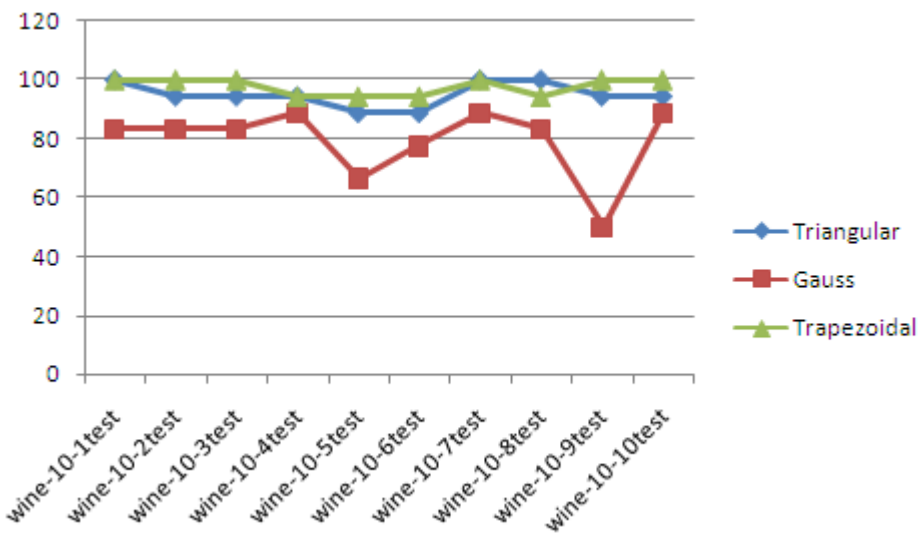


Figura #17 Gráfica de los resultados en la predicción del ensayo WINE.

Puede observarse que se obtuvo, al igual que con las muestras del IRIS, buenos resultados en la predicción con las funciones de pertenencia Triangular y Trapezoidal, mientras que decae el resultado con la función de pertenencia gaussiana. Aquí se comprueba que la función trapezoidal fue la que mejor clasificación realizó pues de los 10 ficheros de pruebas en 6 se obtuvo un 100% de buena predicción.

Para los ficheros de prueba de las cefalosporinas se obtuvieron los siguientes resultados:

Ficheros de prueba	Clases	Cant. Atributos	Cant. de datos	Datos mal clasificados (Triangular)	Datos mal Clasificados (Gauss)	Datos mal clasificados (Trapezoidal)
potec-10-1test	2	11	11	4	5	5
potec-10-2test	2	11	11	2	3	5
potec-10-3test	2	11	11	6	5	4
potec-10-4test	2	11	11	3	5	4
potec-10-5test	2	11	11	3	4	3
potec-10-6test	2	11	11	3	4	4
potec-10-7test	2	11	11	2	4	4
potec-10-8test	2	11	11	3	5	4
potec-10-9test	2	11	11	0	5	4
potec-10-10test	2	11	11	3	3	3

Tabla #6 Resultados de la predicción con la muestra de cefalosporinas.

A continuación se muestra un gráfico que representa los resultados expuestos anteriormente:

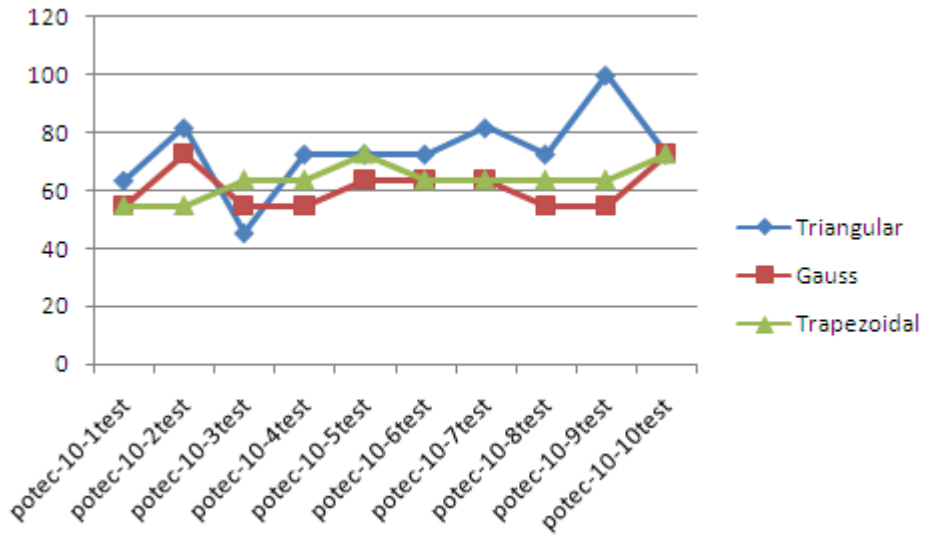


Figura #18 Gráfica de los resultados en la predicción de las cefalosporinas.

En el gráfico se puede observar que los porcientos en los ficheros de prueba son resultados desfavorables debido a que los modelos utilizados no permitieron una buena clasificación de los datos. Las razones fueron explicadas en la sección 3.4.3.

3.4.5 Penalización de datos mal clasificados

En esta sección se analiza a modo de prueba una de las extensiones del algoritmo Ishibuchi-99 [16] con el objetivo de obtener un mejor porciento de acierto en la clasificación de los datos. Para ello se utiliza la función de Gauss, debido a que fue la función de pertenencia con la que se obtuvieron resultados por debajo del 90% en los ficheros de entrenamientos y pruebas de la base de datos IRIS. Para el análisis se empleó la fórmula #11, a través de la cual se irán penalizando los valores de fitness de aquellas reglas que clasifiquen mal a un vector de entrada.

$$fitness(R_j) = NCP(R_j) - w_{error} * NMP(R_j)$$

Fórmula #11

Donde $fitness(R_j)$ es el valor fitness de la regla R_j, $NCP(R_j)$ es el número de vectores de entrada bien clasificados por la regla R_j, w_{error} es el valor de penalización al valor fitness de la regla R_j y $NMP(R_j)$ es el número de vectores de entrada mal clasificados por la regla R_j.

Una vez implementada la fórmula #11 en el código se comienza el proceso de creación de los modelos difusos. Para el experimento se mantienen los mismos parámetros de la sección 3.4.1 con los cuales se obtuvieron los siguientes resultados reflejados en la siguiente tabla:

Ficheros de entrenamiento	Clases	Cant. Atributos	Cant. de datos	Porciento de entrenamiento
iris-10-1tra	3	4	135	94,81
iris-10-2tra	3	4	135	95,56
iris-10-3tra	3	4	135	96,3
iris-10-4tra	3	4	135	96,3
iris-10-5tra	3	4	135	94,81
iris-10-6tra	3	4	135	92,59
iris-10-7tra	3	4	135	94,81
iris-10-8tra	3	4	135	94,07
iris-10-9tra	3	4	135	94,07
iris-10-10tra	3	4	135	95,55

Tabla #7 Resultados en la creación de modelos difusos con w_{error}=5 con el ensayo IRIS.

A continuación se muestra un gráfico que representa los resultados expuestos anteriormente:

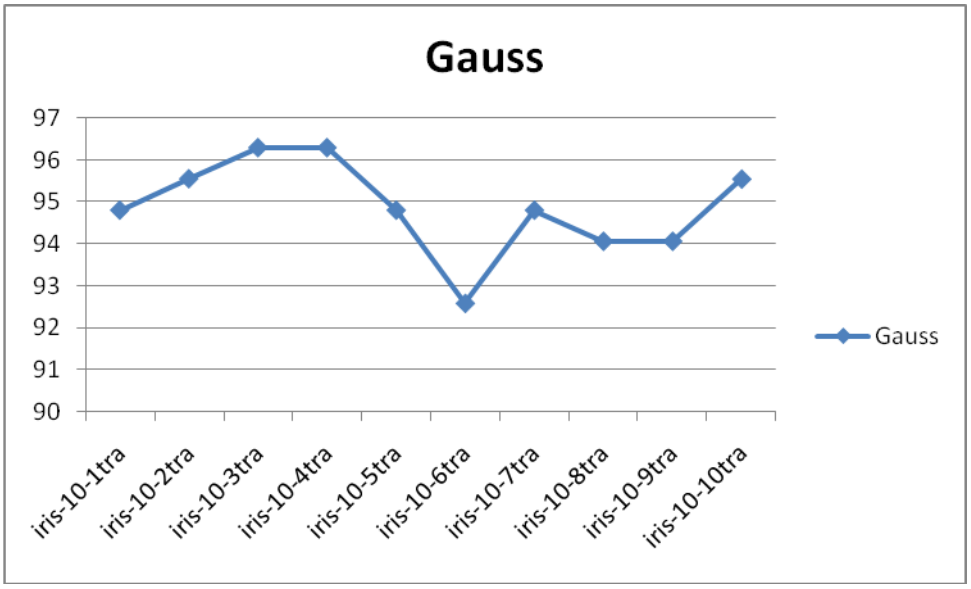


Figura #19 Resultados en la creación de modelos difusos con el uso de la extensión del algoritmo aplicando la función Gauss.

Se puede observar que los porcientos de acierto obtenidos con los ficheros de entrenamiento subieron de un 90% de los modelos difusos anteriores, a un 96% en estos modelos difusos. Ello demuestra que la extensión implementada en el algoritmo provoca un aumento en los porcientos de acierto en la creación de los modelos difusos. Además las predicciones realizadas sobre los ficheros de prueba muestran la veracidad de dichos resultados:

Ficheros de pruebas	Clases	Cant. Atributos	Cant. de datos	Datos mal clasificados (Gauss)	Porciento de acierto
iris-10-1test	3	4	15	0	100
iris-10-2test	3	4	15	0	100
iris-10-3test	3	4	15	0	100
iris-10-4test	3	4	15	1	93,33
iris-10-5test	3	4	15	2	86,67
iris-10-6test	3	4	15	0	100
iris-10-7test	3	4	15	0	100
iris-10-8test	3	4	15	1	93,33
iris-10-9test	3	4	15	0	100
iris-10-10test	3	4	15	1	93,33

Tabla #8 Resultados en la predicción con w_{error}=5 con el ensayo IRIS.

A continuación se muestra un gráfico que representa los resultados expuestos anteriormente:

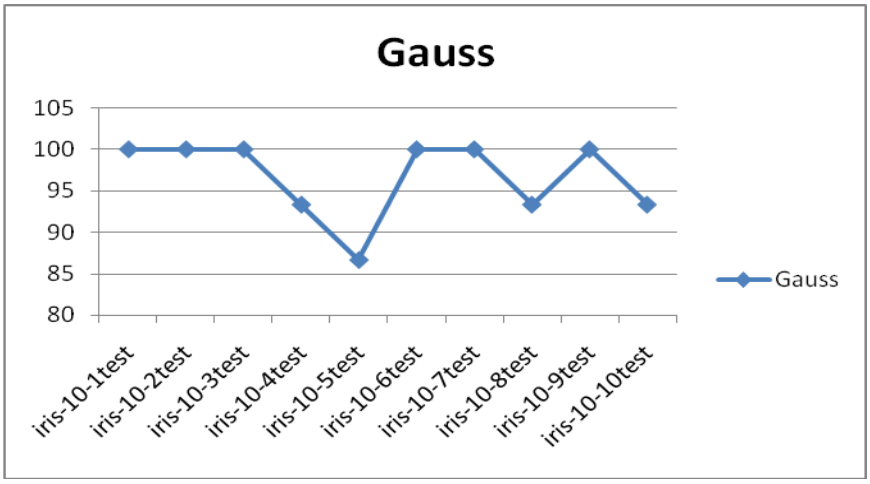


Figura #20 Resultados en la predicción con el uso de la extensión del algoritmo aplicando la función Gauss.

Se puede observar en la gráfica anterior cómo en varias ocasiones se obtuvieron predicciones de un 100%, en comparación con las realizadas en la sección 3.4.4, donde solo se obtuvo 100% en una ocasión.

3.5 Conclusiones

- Se implementó una aplicación de prueba para evaluar los valores de clasificación del algoritmo Ishibuchi-99.
- Se implementaron dos nuevas funciones de pertenencia Gauss y Trapezoidal.
- Se implementó el patrón estrategia por la posibilidad de agrupar funciones comunes y facilitar la reutilización del código.
- Se realizaron pruebas sobre bases de datos conocidas como IRIS y WINE obteniéndose resultados comparables a otros reportados en la literatura.
- Se realizaron pruebas con una muestra de datos reales (cefalosporinas), con la obtención de resultados de clasificación prometedores.
- Se implementó una de las extensiones del código Ishibuchi-99 obteniéndose mejores resultados en la predicción.

CONCLUSIONES GENERALES

Con la realización de este trabajo se arribaron a las siguientes conclusiones, las que cumplen con los objetivos trazados.

- Se implementó el algoritmo Ishibuchi-99 para la generación de modelos difusos de predicción de actividad biológica basado en principios de lógica difusa combinada con algoritmo genético.
- Se implementó una aplicación de prueba que permitió incorporar y validar el algoritmo seleccionado.
- Se validó el algoritmo implementado a través de los modelos difusos obtenidos con los ensayos del IRIS y WINE con porcientos de acierto aceptables. Estas pruebas realizadas con el ensayo del IRIS arrojaron resultados de 95.56% para la función de pertenencia triangular, 90.37% para gauss y 97.78% para la trapezoidal. Para el caso del ensayo WINE se obtuvieron resultados de 96.89% para la función de pertenencia triangular, 93.14% para gauss y 98.75% para la trapezoidal.
- Se realizó un estudio de clasificación de cefalosporinas antibacterianas con las cuales se generaron modelos difusos con un porciento de veracidad aceptable por encima del 82% en las muestras de entrenamiento y del 70% en las muestras de prueba.

RECOMENDACIONES

- Extender el estudio a otros ensayos.
- Utilizar la implementación del algoritmo en medios de cómputo más potentes con el uso de mayores volúmenes de datos, para verificar que se creen mejores modelos difusos.
- Profundizar en el estudio de las extensiones del código Ishibuchi-99.
- Incorporar el algoritmo a la plataforma GRAph-TOol.

REFERENCIAS

1. **Marrero Ponce, Yovani, et al.** TOMOCOMD-CARDD: Un Novedoso Enfoque para el Diseño Racional Insílico de fármacos antimaláricos. *Fórum de Ciencia y Técnica.* [Online] [Cited: 10 26, 2007.] http://www.forum.villaclara.cu/ponencias/trabajo/2.
2. Semichem. [En línea] [Citado el: 05 de 12 de 2007.] http://www.semichem.com/codessa/default.php.
3. **Martí Pérez, Ileane y Socorro Paz, Onailet.** *Módulo de predicción de la actividad biológica anticancerígena partiendo de descriptores topológicos e híbridos.* La Habana: s.n., Julio del 2007.
4. **Gutiérrez de Terán Castañón, Hugo.** Modelización molecular de los receptores de adenosina y sus ligandos en el marco de diseño de fármacos asistido por ordenador. [En línea] Febrero de 2004. [Citado el: 08 de 11 de 2007.] http://www.tesisenxarxa.net/TESIS_UPF/AVAILABLE/TDX-0930104-091416//thgc1de1.pdf.
5. . **Porragas, Gabriela Espinosa.** Modelos QSPR/QSAR/QSTR basados en sistemas neuronales cognitivos. [En línea] [Citado el: 08 de 11 de 2007.] http://www.tdr.cesca.es/TDX-1016102-101509/index_cs.html.
6. neuralware. [En línea] [Citado el: 05 de 12 de 2007.] http://www.neuralware.com/products.jsp.
7. El software más avanzado para investigar en ciencias de la vida y de los materiales. *Fórum Tecnológico.* [En línea] 2006. [Citado el: 05 de 12 de 2007.] http://www.forumtecnologico.com/articulos_html/ft_10_html/10-09/index.asp.
8. **Martín del Brío, Bonifacio y Sanz Molina, Alfredo.** *Redes Neuronales y Sistemas Difusos.* España: s.n., Marzo, 2001.
9. Capítulo 2: CONCEPTOS FUNDAMENTALES DE LÓGICA DIFUSA. [En línea] [Citado el: 19 de 12 de 2007.] http://www.tesisenxarxa.net/TESIS_UPC/AVAILABLE/TDX-0207105-105056//04Rpp04de11.pdf.
10. **Villanueva Garza, César Alberto y Soto Rodríguez, Rogelio**. [Online] [Cited: 11 20, 2007.] http://64.233.169.104/search?q=cache:xGpC0BUdG4MJ:www.aistmexico.org.mx/descargar.asp%3Fid%3D36+concepto+de+sistema+de+control+difuso&hl=es&ct=clnk&cd=4

&gl=cu.

11. **Hernández Orallo, José y Ferri Ramírez, Cèsar.** *Curso de Doctorado Extracción Automática de Conocimiento.* Marzo 2006.
12. rapid-i.com. [En línea] http://rapid-i.com/content/view/114/134/lang,en/.
13. KEEL. [En línea] [Citado el: 12 de 11 de 2007.] http://www.keel.es/
14. **H. Ishibuchi, K. Nozaki, N. Yamamoto, H. Tanaka**. KEEL. [Online] 08 1995. [Cited: 12 03, 2007.] http://sci2s.ugr.es/keel/pdf/algorithm/articulo/1995-IEEE_TFS-Ishibuchi-GA-Selection.pdf.
15. **H. Ishibuchi, T. Yamamoto, T. Nakashima**. KEEL. [Online] 04 2005. [Cited: 12 03, 2007.] http://sci2s.ugr.es/keel/pdf/algorithm/articulo/2005%20-%20Ishibuchi%20-IEEE_TSMC%20-%20Hybridization%20of%20Fuzzy%20GBML%20Approaches.pdf.
16. **H. Ishibuchi, T. Nakashima, T. Murata.** KEEL. [Online] 10 1999. [Cited: 12 03, 2007.] http://sci2s.ugr.es/keel/pdf/algorithm/articulo/1999-IEEE_TSMC-Ishibuchi-FRBCS.pdf.
17. [Online] [Cited: 01 12, 2008.] http://eddyalfaro.galeon.com/geneticos.html.
18. Creangel.com. [En línea] [Citado el: 25 de 11 de 2007.] http://www.creangel.com/uml/intro.php
19. Sitio de descargas de software. [En línea] [Citado el: 25 de 11 de 2007.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/
20. **Flanagan, David.** *Java en pocas palabras.* México: s.n., 1998.
21. **Montero, Ildefonso.** portalmundos.com. [En línea] [Citado el: 10 de 12 de 2007.] http://mundoinformatica.portalmundos.com/herramientas-de-desarrollo-de-aplicaciones-para-sistemas-gnulinux-alternativas-de-mercado/.
22. [Online] [Cited: 12 01, 2007.] ftp://ftp.ics.uci.edu/pub/machine-learning-databases
23. **Cordón, Oscar, José del Jesus, María y Herrera, Francisco.** *A proposal on reasoning methods in fuzzy rule-based classification systems.* [En línea] http://sci2s.ugr.es.
24. **Lagos Torres, Manuel.** *Introducción al diseño con patrones.* [En línea] http://www.elrincondelprogramador.com/default.asp?id=29&pag=articulos/leer.asp.
25. **Abián, Miguel Ángel.** javaHispano. [En línea] http://www.javahispano.org/contenidos/es/el_patrn_estrategia.
26. **Carrasco Velar, Ramón.** *Nuevos descriptores atómicos y moléculares para estudios de*

estructura-actividad: Aplicaciones. Ciudad de La Habana : s.n., 2003.

BIBLIOGRAFÍA

1. [En línea] [Citado el: 01 de 12 de 2007.] ftp://ftp.ics.uci.edu/pub/machine-learning-databases.

2. [En línea] [Citado el: 01 de 12 de 2007.] http://eddyalfaro.galeon.com/geneticos.html..

3. *Capítulo 2: CONCEPTOS FUNDAMENTALES DE LÓGICA DIFUSA.* [En línea] [Citado el: 19 de 12 de 2007.] http://www.tesisenxarxa.net/TESIS_UPC/AVAILABLE/TDX-0207105-105056//04Rpp04de11.pdf..

4. Creangel.com. [En línea] [Citado el: 25 de 11 de 2007.] http://www.creangel.com/uml/intro.php..

5. *El software más avanzado para investigar en ciencias de la vida y de los materiales. Fórum Tecnológico.* [En línea] 2006. [Citado el: 05 de 12 de 2007.] http://www.forumtecnologico.com/articulos_html/ft_10_html/10-09/index.asp.

6. KEEL. [En línea] [Citado el: 12 de 11 de 2007.] http://www.keel.es/.

7. neuralware. [En línea] [Citado el: 05 de 12 de 2007.] http://www.neuralware.com/products.jsp.

8. Semichem. [En línea] [Citado el: 05 de 12 de 2007.] http://www.semichem.com/codessa/default.php.

9. *Sitio de descargas de software.* [En línea] [Citado el: 25 de 11 de 2007.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/.

10. **Badenas, Federico Gago.** *Métodos computacionales de modelado molecular.* [En línea] [Citado el: 06 de 11 de 2007.] http://www2.uah.es/farmamol/Public/PDF_files/RAF_1993.pdf.

11. **Fernández Carrasco, Oscar M., García León, Delba y Beltrán Benavides, Alfa.** [En línea] 1995. http://bvs.sld.cu/revistas/aci/vol3_3_95/aci05395.htm.

12. **Flanagan, David.** *Java en pocas palabras.* México : s.n., 1998.

13. **Gómez, Isabel.** . *Bioinformática o la nueva ciencia de la comunicación.* [En línea] 2005. [Citado el: 05 de 02 de 2008.] http://www.microsoft.com/spain/enterprise/perspectivas/numero_14/research.mspx. 14.

14. **Gutiérrez de Terán Castañón, Hugo.** . *Modelización molecular de los receptores de adenosina y sus ligandos en el marco de diseño de fármacos asistido por ordenador.* [En línea] Febrero de 2004. [Citado el: 08 de 11 de 2007.] http://www.tesisenxarxa.net/TESIS_UPF/AVAILABLE.

15. **H. Ishibuchi, K. Nozaki, N. Yamamoto, H. Tanaka.** KEEL. [En línea] 08 de 1995. [Citado el: 03 de 12 de 2007.] http://sci2s.ugr.es/keel/pdf/algorithm/articulo/1995-IEEE_TFS-Ishibuchi-GA-Selection.pdf.

16. **H. Ishibuchi, T. Nakashima, T. Murata.** KEEL. [En línea] 10 de 1999. [Citado el: 03 de 12 de 2007.] http://sci2s.ugr.es/keel/pdf/algorithm/articulo/1999-IEEE_TSMC-Ishibuchi-FRBCS.pdf.

17. **H. Ishibuchi, T. Yamamoto, T. Nakashima.** KEEL. [En línea] 04 de 2005. [Citado el: 03 de 12 de 2007.] http://sci2s.ugr.es/keel/pdf/algorithm/articulo/2005%20-%20Ishibuchi%20-IEEE_TSMC%20-%20Hybridization%20of%20Fuzzy%20GBML%20Approaches.pdf.

18. **León, Lic. José Angel Morales.** ilustrados.com. *Síntesis de derivados ceténicos de la n-(fur-2-il-metil)-2-ciano-3,3-bis(metiltio)-acrilamida.* [En línea] [Citado el: 06 de 11 de 2007.] http://www.ilustrados.com/publicaciones/EEFZVAAlulTACyxJJa.php.

19. **Marrero Ponce, Yovani.** Fórum de Ciencia y Técnica. *TOMOCOMD-CARDD: Un Novedoso Enfoque para el Diseño Racional Insílico de fármacos antimaláricos.* [En línea] [Citado el: 26 de 10 de 2007.] http://www.forum.villaclara.cu/ponencias/trabajo/2.

20. **Martí Pérez, Ileane y Socorro Paz, Onailet.** *Módulo de predicción de la actividad biológica anticancerígena partiendo de descriptores topológicos e híbridos. .* La Habana : s.n., Julio del 2007.

21. **Martín del Brío, Bonifacio y Sanz Molina, Alfredo.** *Redes Neuronales y Sistemas Difusos.* España : s.n., Marzo, 2001.

22. **Montero, Ildefonso.** portalmundos.com. . [En línea] [Citado el: 10 de 12 de 2007.] http://mundoinformatica.portalmundos.com/herramientas-de-desarrollo-de-aplicaciones-para-sistemas-gnulinux-alternativas-de-mercado/.

23. **Porragas, Gabriela Espinosa.** *Modelos QSPR/QSAR/QSTR basados en sistemas neuronales cognitivos.* [En línea] [Citado el: 08 de 11 de 2007.] http://www.tdr.cesca.es/TDX-1016102-101509/index_cs.html.

24. **Prieto Entenza, Julio Omar y Reyes Crespo, Adisley.** *Predicción de actividad anticancerígena de compuestos orgánicos partiendo de fragmentos, utilizando Lógica Difusa.* La Habana : s.n., Julio del 2007.

25. **Quispe, José Carlos Chahuara.** *Control Neuro-Difuso aplicado a una Grúa Torre.* [En línea] [Citado el: 07 de 02 de 2008.] http://sisbib.unmsm.edu.pe/bibvirtualdata/Tesis/Ingenie/chahuara_qj/Cap2.pdf.

26. **Ramírez, MSc. Juan Carlos.** Biblioteca Virtual en Vacunas. *La Bioinformática en la obtención de vacunas. .* [En línea] [Citado el: 03 de 11 de 2007.] http://www.bvv.sld.cu/download.php?url=117072029420.pdf.

27. **Villanueva Garza, César Alberto y Soto Rodríguez, Rogelio.** [En línea] [Citado el: 20 de 11 de 2007.] http://64.233.169.104/search?q=cache:xGpC0BUdG4MJ:www.aistmexico.org.mx/descargar.asp%3Fid%3D36+concepto+de+sistema+de+control+difuso&hl=es&ct=clnk&cd=4&gl=cu.

28. Ruby on Rails. [En línea] 16 de 06 de 2007. http://rubyonrails-

cusco.blogspot.com/2007/06/subversion-control-de-versiones-con.html.

29. **Lagos Torres, Manuel.** *Introducción al diseño con patrones.* [En línea] http://www.elrincondelprogramador.com/default.asp?id=29&pag=articulos/leer.asp.

30. **Carrasco Velar, Ramón.** *Nuevos descriptores atómicos y moléculares para estudios de estructura-actividad: Aplicaciones.* Ciudad de La Habana : s.n., 2003.

31. rapid-i.com. [En línea] http://rapid-i.com/content/view/114/134/lang,en/.

32. **Hernández Orallo, José y Ferri Ramírez, Cèsar.** *Curso de Doctorado Extracción Automática de Conocimiento.* Marzo 2006.

33. **Cordón, Oscar, José del Jesus, María y Herrera, Francisco.** *A proposal on reasoning methods in fuzzy rule-based classification systems.* [En línea] http://sci2s.ugr.es.

34. **Abián, Miguel Ángel.** javaHispano. [En línea] http://www.javahispano.org/contenidos/es/el_patrn_estrategia.

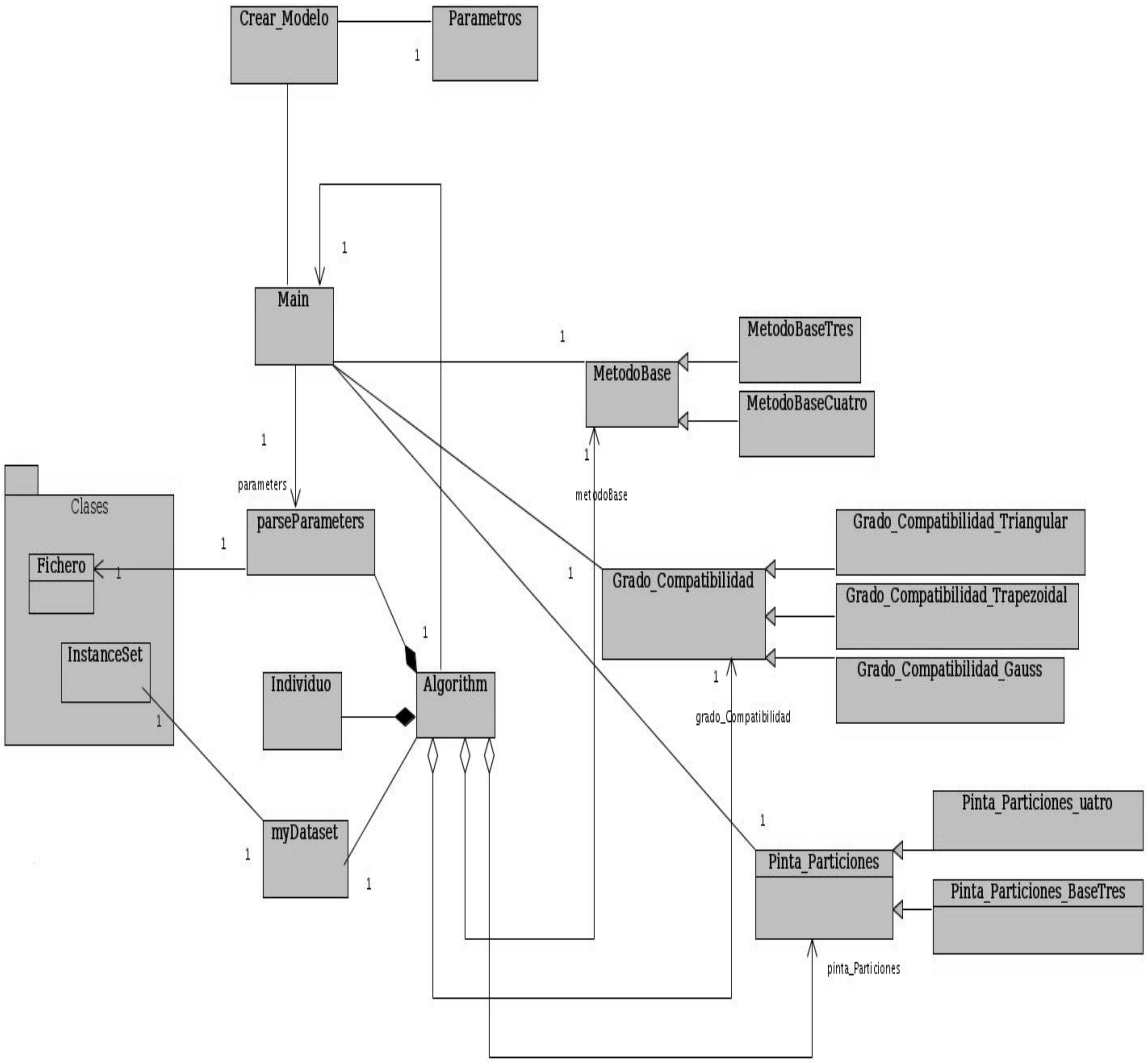
35. **Dürsteler, Juan C.** La revista digital de InfoVis.net. *Mapas Conceptuales.* [En línea] http://www.infovis.net/printMag.php?num=141&lang=1.

36. El Mundo Informático. [En línea] http://jorgesaavedra.wordpress.com/category/patrones-grasp/.

ANEXOS

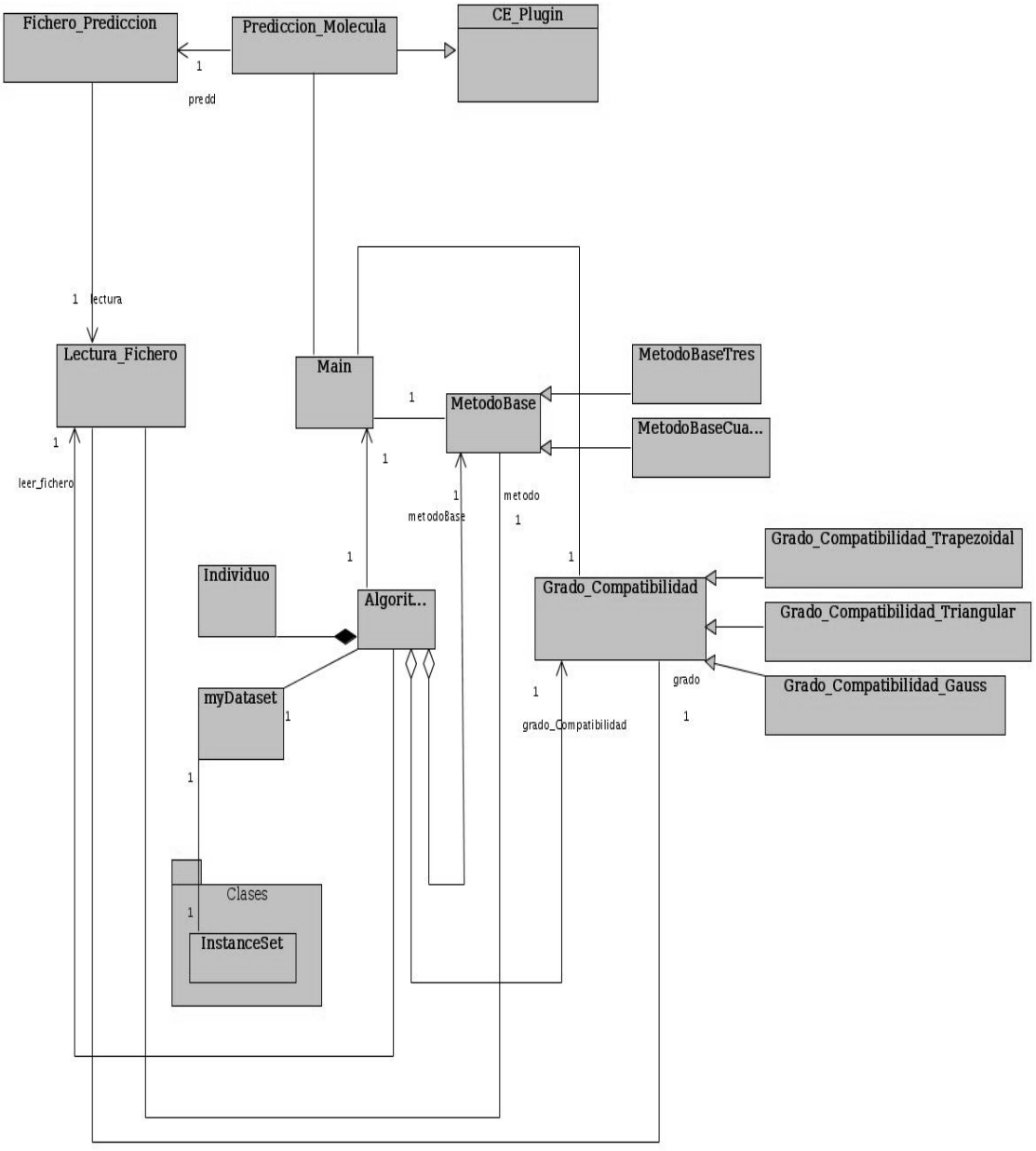
Anexo #1

Diagrama de clases de la funcionalidad Crear Modelo Difuso.



Anexo #2

Diagrama de clases de la funcionalidad Realizar Predicción.



GLOSARIO DE TÉRMINOS

Actividad Biológica: Actividad que caracteriza el comportamiento biológico en compuestos químicos.

Administrador: Persona que se encarga de generar los nuevos modelos y realizar el entrenamiento de los mismos.

Bioinformática: Es la aplicación de métodos informáticos sobre sistemas de cómputo y tratamiento de la información para el análisis de datos experimentales (de nivel molecular, principalmente) de sistemas biológicos, así como la simulación de los mismos.

Compuestos orgánicos: Son sustancias químicas basadas en cadenas de carbono e hidrógeno. En muchos casos contienen oxígeno, y también nitrógeno, azufre, fósforo, boro y halógenos.

CQF: Centro de Química Farmacéutica (CQF) es una institución para el desarrollo de investigaciones científico-tecnológicas dirigidas hacia la obtención de sustancias bioactivas para uso humano.

Descriptor: Número que caracteriza estructuralmente la molécula.

Descriptores topológico: Número que se calcula a partir de algoritmos que se aplican a la matriz de adyacencia o de conectividad del grafo molecular desprovisto de hidrógeno.

Entrenamiento: Acción que se realiza para el aprendizaje de las neuronas.

Especialista: Cliente que utilizará el sistema.

Estadística: Es una rama de la matemática que se refiere a la recolección, estudio e interpretación de los datos obtenidos en un estudio.

Metodología: Define quién hace qué, cómo y cuando.

Modelo: Representación abstracta de la realidad.

Modelo difuso: Modelo que se crea a partir del las reglas difusas generadas.

Molécula: La partícula más pequeña de una sustancia, que mantiene las propiedades químicas específicas de esa sustancia.

Multiplataforma: Término usado para referirse a los programas, sistemas operativos, lenguajes de programación u otra clase de software, que puedan funcionar en diversas plataformas.

Predicción: Acción que el sistema realiza para emitir un resultado.

Plataforma: Es el principio, ya sea de hardware o software, sobre el cual un programa puede ejecutarse.

Plug-in: Es una aplicación informática que interactúa con otra aplicación para

aportarle una función o utilidad específica**.**

Procedimiento Heurístico: Procedimiento para resolver un problema de optimización bien definido mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución.

Software: Término genérico que designa al conjunto de programas que posibilitan realizar una tarea específica en un ordenador.