DISEÑO E IMPLEMENTACIÓN DE SOFTWARE PARA LA DETECCIÓN Y ELIMINACIÓN DE INCONSISTENCIAS EN BASES DE REGLAS DE SISTEMAS DIFUSOS E INSERCIÓN DE REGLAS EN CASO DE INCOMPLETITUD DE LA BASE DE CONOCIMIENTO.

LISBETH ISABEL URUETA VIVANCO

UNIVERSIDAD AUTONOMA DE BUCARAMANGA
INSTITUTO TECNOLOGICO DE MONTERREY
MAESTRIA EN CIENCIAS COMPUTACIONALES
CARTAGENA DE INDIAS
2006.

DISEÑO E IMPLEMENTACIÓN DE SOFTWARE PARA LA DETECCIÓN Y ELIMINACIÓN DE INCONSISTENCIAS EN BASES DE REGLAS DE SISTEMAS DIFUSOS E INSERCIÓN DE REGLAS EN CASO DE INCOMPLETITUD DE LA BASE DE CONOCIMIENTO.

LISBETH ISABEL URUETA VIVANCO

JUAN ANTONIO CONTRERAS MONTES
DOCTOR EN CONTROL INTELIGENTE
INGENIERO ELECTRICISTA

UNIVERSIDAD AUTONOMA DE BUCARAMANGA
INSTITUTO TECNOLOGICO DE MONTERREY
MAESTRIA EN CIENCIAS COMPUTACIONALES
CARTAGENA DE INDIAS
2006.

Nota de Aceptación

_____

_____

_____

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del jurado

_____

Firma del jurado

Cartagena, 2 de Mayo del 2006

Dedicatoria

Al Creador de toda la Inteligencia humana, Jehová Dios.

AGRADECIMIENTOS

A Mayela Vivanco, mi madre, por ser mi compañera, amiga y fuente de inspiración y superación personal.

A Felipe Urueta, mi padre, por su apoyo y estimulo condicional.

A Juan Contreras Montes, mi tutor, por su calidad humana y su insuperable espíritu lucha, así como su presteza a enseñar.

A Martín R. Rodríguez, mi Esposo para ser la nueva razón de llegar a cumplir mis metas, por su amor, paciencia y comprensión.

CONTENIDO

Pág.

LISTA DE TABLAS

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

INTRODUCCION

El trabajo de investigación propuesto en esta tesis corresponde al módulo de validación de la base de reglas de sistemas difusos e inserción de reglas en caso de incompletitud. Este módulo permite mejorar la calidad del conocimiento del sistema experto difuso, disminuir la complejidad computacional del mismo y facilitar la supervisión, adaptación y/o modificación estructural, en el caso de modelos borrosos aplicados a procesos con dinámicas variables o no lineales.

Mediante la creación de Sistema difusos de forma heurística, aplicando los conceptos de la lógica difusa y los algoritmos ajustados para detectar anomalías en la base de conocimientos de los sistemas creados por un experto se asegura de la creación de sistemas que no tengan un grado de error muy bajo a la hora de hacer inferencia en los valores de salida.

Así mismo este trabajo de investigación creo la opción de poder generar un modelo difuso a partir de los datos cargados en un archivo de datos de entrada, que ordena los datos de entrada identificando las variables de entrada y la variable de salida que por defecto se trabajó con una (1). La aplicación permite detectar clases haciendo particiones del antecedente y consecuente lo que ayudará a generar una base de reglas consistente y sin las anomalías más comunes en una base de reglas como son: Contradicciones, Redundancias, Subsumicion, etc.

La aplicación permite además incrementar el número de conjuntos buscando minimizar el error inicial que se establece para que el modelo sea el inicial.

1. ANTECEDENTES EN LA DETECCIÓN Y ELIMINACIÓN DE ANOMALÍAS EN BASES DE REGLAS

1.1 ESTADO DEL ARTE

Uno de los primeros trabajos sobre métodos de detección de consistencia y completitud en bases de reglas fue el desarrollado en 1982 por Suwa, Scott y Shortliffe[1], denominado ONCOCIN RULE CHECKER. En éste, las reglas con la misma conclusión son comparadas para, previa revisión de los antecedentes, detectar conflictos y redundancias.

En los años de 1985 y 1987, Nguyen [2] presenta sus trabajos sobre validación de bases de reglas, incluyendo métodos que detectan, además de las situaciones anteriores, antecedentes innecesarios, reglas circulares y conclusiones no alcanzables, entre otras anomalías.

En 1992, Meseguer y Plaza [3] presentan un metalenguaje (VETA) para validación de bases de conocimiento, el cual se basa en el algoritmo para detección de inconsistencias en bases de reglas mediante redes de Petri planteado anteriormente por Messenger [4] en 19901.

Los métodos o algoritmos para validación de bases de reglas mencionados se basan en el cálculo binario y no hacen referencia a las inconsistencias en bases de reglas difusas.

Tal vez el primer trabajo sobre validación en lógica borrosa es el planteado en 1991 por Yager y Larsen [5], en el cual realizan la detección de inconsistencias mediante el método de reflexión sobre el universo de entrada, tratando de descubrir qué combinaciones de reglas concluyen el conjunto vacío.

En 1994 y 1997 aparecen los trabajos de Dubois y Prado [6], y de Dubois, Prade y Ughetto [7] respectivamente. Plantean la obtención de un coeficiente

[1] Meseguer, P. y Plaza, E. Validation of KBS: The VALID project. In L. Steels y B. Le Pape (Eds.), Enhacing the knowledge engineering process (p. www.iiia.csic.es/People/enric/valid-kbs.html). North-Holland Elsevier. 1992

de inconsistencia a partir de la proyección de la relación borrosa obtenida por la agregación de las reglas.

En 1998 Sala [8] introduce, en su tesis doctoral,"una medida de error de inferencia, obtenida a partir de la interpretación algebraica de las reglas difusas, equivalente a una distancia conceptual adecuada para diferenciar gradualmente desde conceptos "conceptualmente idénticos" a conceptos "contrarios" mediante una medida en el intervalo [0, 1], con la intención de detectar contradicciones y redundancias en las bases de reglas difusas."2

También en 1998, e igualmente en su tesis doctoral, Kim [9] utiliza medidas de inconsistencia por comparación cuantitativa como método para aislar inconsistencias en bases de reglas, analizando la complejidad computacional de cada procedimiento propuesto.

En 1996, se desarrolla una herramienta para clasificación de reglas difusas, a partir de los datos de entrada y salida y de una partición inicial del espacio de entrada, denominada NEFCLASS-X. En 1999 sale una nueva versión desarrollada en lenguaje JAVA denominada NEFCLASS-J.

En el 2002 Lee, Chen y Liu [10] desarrollan un nuevo "método para detectar y eliminar inconsistencias basado en los soportes de necesidad y de posibilidad como una herramienta para estimar la similitud entre dos conjuntos difusos. Además, para prevenir la pérdida de información debido a la eliminación de reglas, proponen un método de inserción de reglas mediante redes neuronales. Demuestran que la existencia de reglas inconsistentes ocasionan dificultad en el aprendizaje, inestabilidad y a menudo interrupción por mínimos locales."3

En el 2002, Senhadji et al. , presentan un nuevo algoritmo, denominado NORFREA, para seleccionar reglas difusas desde una partición reticular inicial del dominio de los datos de entrada, al igual que la herramienta NEFCLASS-J, pero introduciendo unas mejoras que eliminan el efecto de la partición inicial mediante el desplazamiento de los soportes de los conjuntos difusos iniciales, manteniendo la normalidad de los mismos.

---

[2] Sala, A.. Validación y Aproximación Funcional en Sistemas de Control Basados en Lógica Borrosa. Universidad Politécnica de Valencia. Tesis Doctoral. 1998

[3] Lee, H-M, Chen, J-M, and Liu C-L(2002). Inconsistency Resolution and Rule Insertion for Fuzzy Rule-Based Systems. Journal Of Information Science and Engineering. Pág. 187-210

En el 2003, Dubois, Prade y Ughetto [7] plantean una nueva perspectiva en el razonamiento con reglas difusas, basada en la consideración de que la información codificada en un computador podría tener un énfasis negativo o positivo. Proponen una nueva regla composicional de inferencia, adaptada para reglas conjuntivas, específica para información positiva. Esta propuesta constituye una valiosa herramienta para evaluar y/o mejorar la calidad del conocimiento almacenado en una base de reglas difusas.

La investigación de la lógica difusa está dirigida, principalmente, por dos grupos con diferentes tendencias. Por un lado los lógicos matemáticos, quienes ven a la lógica difusa como una rama de la lógica donde pueden encontrar problemas interesantes por resolver; por otro lado están los investigadores dedicados al desarrollo de aplicaciones de la lógica difusa y la computación suave. Sin duda, se avanzaría mucho más rápido en el desarrollo de eficientes algoritmos de control difuso si los grupos mencionados no trabajaran tan aisladamente sino que aunaran sus esfuerzos y trabajaran como un solo grupo con una misma finalidad. Por esta razón el autor ha considerado en su revisión bibliográfica un gran cantidad de material desarrollado por los lógicos matemáticos.

En resumen, a pesar del gran auge de los sistemas difusos, y sus aplicaciones, no existen métodos generales para la validación de bases de reglas que permitan mejorar la calidad del conocimiento y, por ende, ampliar las posibilidades de adaptación y modificación estructural. Igualmente se carece de métodos generales para inserción de reglas en sistemas difusos.

1.2 PLANTEAMIENTO DEL PROBLEMA

1.2.1 DELIMITACION DEL PROBLEMA

La calidad de un sistema experto basado en reglas está determinada por la calidad de la información contenida en su base de conocimientos. Una base de reglas difusas puede presentar, entre otros, los siguientes problemas:

- Contradicciones e inconsistencias de las reglas
- Redundancias en las reglas
- Información incompleta

Sin embargo, la detección de contradicciones en bases de reglas difusas no resulta tan fácil como en las bases de reglas de la lógica clásica o proposicional, en donde la ley del tercio excluido facilita la labor. En muchos sistemas expertos difusos, inclusive en controladores difusos, la existencia de dos reglas vecinas (en una tabla o matriz) con consecuentes disjuntos no indica con certeza la existencia de contradicción, por lo que se requiere acudir a otros alternativas que permitan una mejor valoración. Recientemente, las investigaciones en lógica difusa han intensificado la búsqueda de un método general para la detección de contradicciones en bases de reglas difusas, ya que los métodos actuales se limitan a casos muy particulares.

Por otro lado, en muchos sistemas expertos basados en reglas (en especial en sistemas de diagnóstico médico), dos reglas aparentemente contradictorias nunca se activarían simultáneamente debido a la existencia de otra(s) regla(s) que impide(n) la activación de una de las reglas en contradicción. En estos casos, la incompletitud podría hacer que en una revisión manual se intente eliminar una de las reglas en contradicción cuando en realidad la acción correcta sería acudir al experto para obtener la información adicional necesaria e ingresar la(s) regla(s) que falta(n), imposibilitando la activación de las reglas contradictorias.

La información incompleta de una base de reglas conllevará a resultados incorrectos o a la no activación de ninguna de las reglas aún cuando el valor de la entrada esté en el rango establecido para el universo de discurso de dicha entrada. La eliminación de reglas contradictorias puede ocasionar incompletitud en la base de conocimiento por lo que se requiere de

mecanismos que inserten automáticamente la regla correcta y así evitar áreas vacías en el espacio de trabajo.

La detección y eliminación de redundancias conlleva a una disminución de la complejidad computacional, ya que se posibilita lograr la misma acción con menos reglas. Esto es de vital importancia en especial en los sistemas de tiempo real, más aún en los controladores embebidos, los cuales por lo regular son implementados en microcontroladores con las limitaciones de espacio de memoria inherentes. A pesar de que la detección de redundancias en bases de reglas difusas no reviste la dificultad que se presenta en la detección de contradicciones, frecuentemente se pueden apreciar modelos y controladores borrosos con reglas redundantes en muchas aplicaciones industriales. Esto tal vez es debido a que los software especializados, como MATLAB, FIDE, entre otros, no presentan módulos para detección de redundancias ni contradicciones.

La productividad de un gran número de empresas industriales se basa en gran medida en el desempeño de los sistemas de control de sus procesos. Mejorar el desempeño de los controladores exige el desarrollo de nuevas metodologías para regulación de procesos complejos en presencia de incertidumbre.

La aplicación de técnicas de control inteligente basadas en lógica borrosa ha tenido gran aceptación por parte de los usuarios por el paralelismo con el razonamiento que ellos mismos aplicarían si realizaran el control de un proceso de forma manual. Sin embargo, mucha de las aplicaciones de los sistemas borrosos existentes se ha limitado a procesos de baja dimensionalidad, carecen de métodos de validación de la base de reglas que facilite la labor de diseño y no incluyen herramientas para aprendizaje o adaptación. Todo esto conduce a la necesidad de desarrollar algoritmos de validación y aprendizaje que facilite: la identificación de procesos asi como la reconfiguración en caso de modificación de la dinámica del proceso a controlar.

En resumen, a pesar del gran auge de los sistemas difusos, y sus aplicaciones, no existen métodos generales para la validación de bases de reglas que permitan mejorar la calidad del conocimiento y, por ende, ampliar las posibilidades de adaptación y modificación estructural.

Igualmente se carece de métodos generales para inserción de reglas en sistemas difusos.

1.3 JUSTIFICACION DEL PROBLEMA

Como se ha planteado, la calidad de un sistema difuso depende de la calidad de la base de reglas. Esto implica la necesidad de una herramienta que detecte y elimine de manera automática las reglas contradictorias y redundantes, e inserte las reglas faltantes en caso de incompletitud.

Se describió la dificultad para determinar si una contradicción es intencional, lo cual requiere de introducir reglas de resolución de conflictos, o es producida por error del usuario al introducir las reglas, lo cual obligaría a la eliminación de la regla contradictoria y a la inserción de una nueva regla corregida debido a la incompletitud ocasionada en la eliminación.

Lo anterior ha motivado al autor a la búsqueda de algoritmos eficientes y de aplicación general para la corrección de las anomalías mencionadas en las bases de reglas, lo cual sería un paso significativo en el diseño e implementación de sistemas difusos libres de anomalías, enriqueciendo así la teoría de la lógica difusa.

1.4 HIPOTESIS

En el desarrollo de este proyecto se busca generar nuevos algoritmos que permitan construir un sistema inteligente que detecte y elimine contradicciones, inconsistencias y redundancias en el proceso de validación de bases de reglas difusas, así como para la inserción de reglas en caso de incompletitud de la base de conocimiento. Este sistema implementará, además, un tipo de agente inteligente que será el encargado de realizar estas acciones de manera automática, bajo la autorización de un usuario, y será diseñado teniendo en cuenta los fundamentos teóricos de los diferentes tipos de lógica existentes.

1.5 OBJETIVOS

1.5.1 General

Diseñar e implementar una herramienta software para resolución de inconsistencias e inserción de reglas difusas en bases de conocimiento de sistemas expertos.

1.5.2 Especificos

- Analizar el estado del arte de la detección de contradicciones, inconsistencias y redundancias en Bases de Reglas.

- Hacer un análisis sobre los diferentes tipos de lógicas binarias existentes, sus implicaciones y su posible extensión a la Lógica Difusa.

- Diseñar un algoritmo para detección y eliminación de contradicciones e inconsistencias en Bases de Reglas con antecedentes y consecuentes difusos y/o concretos.

- Diseñar un algoritmo para detección y eliminación de redundancias en Bases de Reglas con antecedentes y consecuentes difusos y/o concretos.

- Diseñar un algoritmo para la inserción de reglas en caso de incompletitud de la base de conocimiento.

- Implementar un sistema inteligente mediante la integración de los algoritmos para detección y eliminación de contradicción e inconsistencias y redundancias, y de inserción de reglas.

1.6 RESULTADOS ESPERADOS

Mediante la realización de este proyecto se pretende conseguir un sistema inteligente que implemente algoritmos eficientes para detección y eliminación de contradicciones, inconsistencias y redundancias en bases de reglas de modelos difusos, así como inserción de reglas en caso de incompletitud de la base de conocimiento. Este software será realizado bajo la arquitectura de un agente inteligente que permita hacer las correcciones de las anomalías mediante una revisión automática de las bases de reglas.

2. PRINCIPIOS DE LÓGICA Y RAZONAMIENTO

Las técnicas para construcción de modelos difusos se pueden resumir básicamente en 4 esquemas [11], los cuales permiten alcanzar el objetivo de todos los métodos propuestos en la mayoría de la literatura que trata sobre el tema, como son:

- Esquema “Table lookup”
- Uso de gradiante descendiente.
- Uso de Clustering y gradiente descendiente.
- Uso de estrategias evolucionarias.

Cada una de ellas presenta sus ventajas y desventajas, el *esquema Table Lookup*, fija de antemano el tipo, número y posición de las funciones de pertenencia y calcula solo los consecuentes de las reglas. La estrategia basada en el *uso del gradiente* descendiente fija de antemano el tipo y número de las funciones de pertenencia y posiciones y el valor de los consecuentes. La estrategia basada en *uso de clustering y gradiente* fija solo el tipo de función de pertenencia y mediante algoritmos de agrupamiento selecciona el número y las posiciones iniciales de las funciones de pertenencia. Y por último, las *estrategias evolucionarías,* son usadas para optimizar todos los posibles aspectos integrados en un modelo difuso, incluyendo la configuración de las entradas usadas para construir el modelo.

Por otro lado, en los sistemas basados en reglas, como los sistemas difusos, la base de conocimiento o de reglas se puede construir de dos formas:

- De manera heurística. Planteando las reglas a partir de la información suministrada por un experto.
- Mediante clasificaciones. A partir de datos de entrada y salida previamente suministrados.

En ambos casos se pueden presentar anomalías en la base de reglas, tales como: reglas contradictorias, reglas subsumidas, reglas redundantes o incompletitud, lo cual afecta la precisión del modelo.

Sin embargo, la detección de contradicciones en bases de reglas difusas no resulta tan fácil como en las bases de reglas de la lógica clásica o proposicional, en donde la ley del tercio excluido facilita la labor. En muchos sistemas expertos

difusos, inclusive en modelos de identificación difusos, la existencia de dos reglas vecinas (en una tabla o matriz) con consecuentes disjuntos no indica con certeza la existencia de contradicción, por lo que se requiere acudir a otros alternativas que permitan una mejor valoración. Recientemente, las investigaciones en lógica difusa han intensificado la búsqueda de un método general para la detección de contradicciones en bases de reglas difusas, ya que los métodos actuales se limitan a casos muy particulares.

Resulta aún más complejo la detección de reglas subsumidas ya que es posible que en una base de conocimiento existan reglas con menos antecedentes que las demás, no por error al introducirlas sino porque son el resultado de eliminar un antecedente que resulta irrelevante ante un determinado rango de entrada.

La detección y eliminación de redundancias conlleva a una disminución de la complejidad computacional, ya que se posibilita lograr la misma acción con menos reglas. Esto es de vital importancia en especial en los sistemas de tiempo real, más aún en los controladores embebidos, los cuales por lo regular son implementados en microcontroladores con las limitaciones de espacio de memoria inherentes. A pesar de que la detección de redundancias en bases de reglas difusas no reviste la dificultad que se presenta en la detección de contradicciones, frecuentemente se pueden apreciar sistemas borrosos con reglas redundantes en muchas aplicaciones industriales. Esto tal vez es debido a que los software especializados, como MATLAB, FIDE, entre otros, no presentan módulos para detección de redundancias ni contradicciones.

La información incompleta de una base de reglas conllevará a resultados incorrectos o a la no activación de ninguna de las reglas aún cuando el valor de la entrada esté en el rango establecido para el universo de discurso de dicha entrada. La eliminación de reglas contradictorias puede ocasionar incompletitud en la base de conocimiento por lo que se requiere de mecanismos que inserten automáticamente la regla correcta y así evitar áreas vacías en el espacio de trabajo.

Para plantear alternativas para disminuir o eliminar el efecto de estas anomalías es necesario un conocimiento amplio de la lógica y de los fundamentos sobre razonamiento bajo incertidumbre, entre otros aspectos.

2.1 Lógica e implicaciones

La lógica ha sido definida como el estudio de las reglas de razonamiento exacto, así como una noción de deducciones que permite realizar inferencias. La lógica se usa para solucionar problemas y usar el conocimiento para aportar soluciones a las necesidades existentes en la vida diaria del mundo real.

2.1.1 Lógica de Proposiciones:

La lógica de proposiciones tiene que ver con el subconjunto de oraciones declarativas que pueden clasificarse como verdaderas o falsas.[12][4]

Una Proposición es una afirmación en que se hace posible determinar su valor de verdad. Las proposiciones Elementales son evaluadas con valores de verdad estipulada por una función de verdad, estas proposiciones pueden ser precisas cuando solo se tiene la opción de dos valores para la función [0, 1] e imprecisas que tiene el rango de valores entre (0, 1). Se les denomina *Imprecisas* dado que no son totalmente ciertas ni totalmente verdaderas.

Una proposición Compuesta tiene un valor que se obtiene a partir de los valores de verdad de las proposiciones que la componen y de las funciones que definen los conectores lógicos que las unen. A las tautologías y contradicciones se les denomina analíticamente verdaderas o falsas respectivamente por que sus valores de verdad pueden determinarse a partir de sus formas simples.

Tabla 1. Operadores Lógicos

Operador	**Significado**
~	NO, Negación.
∧	Y, Conjunción
∨	O, Disyunción
→	Si..Entonces, Condicional
↔	Si y solo si, Doble condicional

[4] GUIARRATANO, Relay. Sistemas Expertos. Principios y Programación. Pág. 83

Partiendo de las anteriores definiciones se puede hablar de equivalencia de proposºiciones cuando una de las dos proposiciones p y q toman el mismo valor de verdad de las proposiciones elementales. Esto se puede cumplir si una de las dos proposiciones es compuesta" [5].

Aquellas proposiciones que siempre se evalúen en 1 se denominan *Tautología*, mientras que las que su valor de verdad siempre sean 0, se les denomina *Contradicciones* (o expresiones inconsistentes) y las que se les puede determinar valor de verdad y falsedad son conocidas como *indeterminaciones* o contingencias.

El proceso de inferencia varia dependiendo del tipo de lógica. Se puede partir desde la Lógica de Proposiciones, que es el simple uso de frases del lenguaje natural que forma proposiciones elementales como compuestas, las cuales se forman mediante los conectores lógicos u operadores lógicos.

Tipos de Implicación en la Lógica de Proposiciones: En la lógica de Proposiciones existen cuatro tipos de implicación como se muestran en la tabla 2

Tabla 2. Tipos de implicaciones Lógicas

IMPLICACIONES	
TIPOS	**DESCRIPCIÓN**
Implicación Rigurosa	$p \to q \Leftrightarrow \neg p \vee q$ $f_{\to}(a,b) = 1 \Leftrightarrow a = 0 \vee b = 1$
Implicación amplia	$p \to q \Leftrightarrow v(p) \le v(q)$ $f_{\to}(a,b) = 1 \Leftrightarrow a \le b$
Doble-implicación rigurosa	$p \to q \Leftrightarrow [p \wedge q] \vee [\neg p \wedge \neg q]$ $f_{\to}(a,b) = 1 \Leftrightarrow [a = b = 1 \vee a = b = 0]$
Doble Implicación amplia	$p \to q \Leftrightarrow [v(p) = v(q)]$ $f_{\to}(a,b) = 1 \Leftrightarrow a = b$

Una implicación es rigurosa si y solo si es implicación amplia; en cambio, cuando se admiten valores de verdad distintos de 0 y 1, una implicación amplia no puede ser rigurosa y viceversa

[5] DIEZ, F. J. Introducción al Razonamiento Aproximado. Departamento de Inteligencia Artificial. UNED. Primera Edición 1998.Revisión: Octubre 2003.

2.1.2 La Lógica Clásica y sus tipos de implicación

La lógica clásica está definida por las funciones de las conectivas, conocidas como Negación, Conjunción, Disyunción, y las propiedades combinadas que incluye la ley distributiva de la conjunción , ley distributiva de la disyunción, la primera ley de Morgan, la segunda Ley de Morgan, absorción de la conjunción y absorción de la disyunción. Las proposiciones que representan estas funciones representan proposiciones genéricas, es decir estas proposiciones pueden ser simples o compuestas y las propiedades se cumplen para todas las proposiciones[6].

La siguiente tabla, la tabla 2.1, muestra los valores de verdad para las funciones que definen los conectores de la lógica clásica.

Tabla 2.1. Valores de verdad para las funciones de los conectores lógicos[7]

P	Q	$f_{\neg}^{C}(p)$	$f_{\wedge}^{C}(p,q)$	$f_{\vee}^{C}(p,q)$	$f_{\rightarrow}^{C}(p,q)$	$f_{\leftrightarrow}^{C}(p,q)$
1	**1**	**0**	**1**	**1**	**1**	**1**
1	**0**	**0**	**0**	**1**	**0**	**0**
0	**1**	**1**	**0**	**1**	**1**	**0**
0	**0**	**1**	**0**	**0**	**1**	**1**

Mediante la anterior tabla, se puede observar que si p→q, es decir que si v(p→q)=1, hay tres posibilidades:

1. v(p)= 0; v(q)=0
2. v(p)=0; v(q)=1 (2.1)
3. v(p)=1; v(q)=1

y se excluye la posibilidad de que v(p)=1, v(q)=0. Esta exclusión comprueba que la función de la implicación lógica→, es tanto una implicación rigurosa como una implicación amplia.

---

[6] DIEZ, F. J. Introducción al Razonamiento Aproximado. Departamento de Inteligencia Artificial. UNED. Primera Edición 1998.Revisión: Octubre 2003. P. 100

[7] Ibíd., P. 101

De igual forma se puede observar que si p↔q entonces v(p)=v(q), lo que demuestra que la lógica clásica se basa en una doble implicación amplia; de hecho; cuando p↔q solo hay dos posibilidades:

1. v(p)=0, v(q)=0
2. v(p)=1, v(q)=1; (2.2)

Lo que la hace una doble implicación rigurosa[8].

2.1.3 La lógica de predicados:

Aunque la lógica de proposiciones es útil, tiene limitaciones. El principal problema de la lógica de proposiciones es que solo puede tratar con afirmaciones completas. Esto quiere decir que no puede examinar la estructura interna de una afirmación.

La lógica de predicado se relaciona con la estructura interna de las afirmaciones, sobre todo, se relaciona con el uso de palabras especiales llamadas cuantificadores, como “todo”, “algo” y “no”.

***Cuantificador Universal*:** Una afirmación cuantificada universalmente tiene el mismo valor de verdad para todos los reemplazos en el mismos dominio. $\forall$ se interpreta como “para todos” o “para cada uno “

Cuantificador Existencial: Este describe una afirmación como verdadera por lo menos para un miembro del dominio**.** $\exists$ se interpreta como “existe al menos un”.

Limitaciones de la lógica de predicado:
“Existen expresiones que no se pueden expresar mediante la lógica de predicado con los cuantificadores universal y existencial.

[8] DIEZ, F. J. Introducción al Razonamiento Aproximado. Departamento de Inteligencia Artificial. UNED. Primera Edición 1998.Revisión: Octubre 2003. P. 100

Otra limitante de esta lógica es en la expresión de cosas que a veces son verdaderas pero no siempre. ” [9]

2.1.4 Lógica Multivaluadas y sus implicaciones.

De la lógica de Proposiciones y de predicados que se basan en valores binarios donde las proposiciones precisas siempre tendrán un valor de verdad asociados en 0 o 1, se pasa a la lógica trivaluada o polivalente como consecuencia del manejo de la proposiciones imprecisas.

La Lógica Multivaluada fue establecida en 1920 por ***Lukasiewicz*** y cuestionaba la lógica clásica o de primer orden. La lógica multivaluada (trivalente) de Lukasiewicz es una lógica extendida de la lógica clásica (basada en dos valores). Esta lógica cumple todas las propiedades de la lógica clásica, con algunas excepciones, y además esta lógica utiliza una implicación amplia y una doble implicación amplia.

La lógica multivaluada de ***Kleene*** es muy similar a la lógica de Lukasiewicz y de igual forma cumple con todas las propiedades de la lógica clásica con excepción de la ley de la contradicción y el tercio excluso.

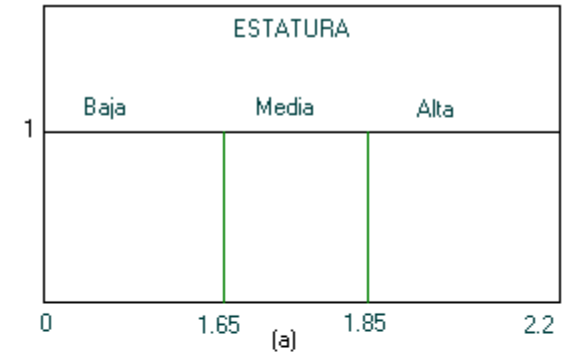
Este tipo de lógica utiliza una implicación rigurosa y una doble implicación rigurosa, esto es lo que la diferencia a la lógica multivaluada de Lukasiewicz.

2.1.5 Lógica Difusa

La lógica clásica, o lógica bivaluada, no resulta adecuada cuando de describir el razonamiento humano se trata, ya que solo “conoce“ dos valores, verdad (1) y falsedad (0), mientras que en la vida real existen hechos que no se pueden definir como totalmente verdaderos o totalmente falsos sino que tienen un grado de verdad, o falsedad, que puede variar de 0 a 1. Un ejemplo sencillo se puede apreciar cuando queremos saber si un vaso está lleno de agua, pero al observarlo notamos que éste no está totalmente lleno. Nuestro sentido común no asigna inmediatamente el valor

[9] GUIARRATANO, Relay. Sistemas Expertos. Principios y Programación. Pág. 91

de 0 (falsedad) a nuestra inquietud sino que razona aceptando que está “algo lleno”, es decir tiene una alta posibilidad de ser catalogado como con un valor más cercano a 1 que a 0.



representación de la variable lingüística
ca bivaluada y la figura 1b muestra una
ma variable lingüística mediante la lógica
ística se han establecido tres valores
ra media y **estatura alta**.

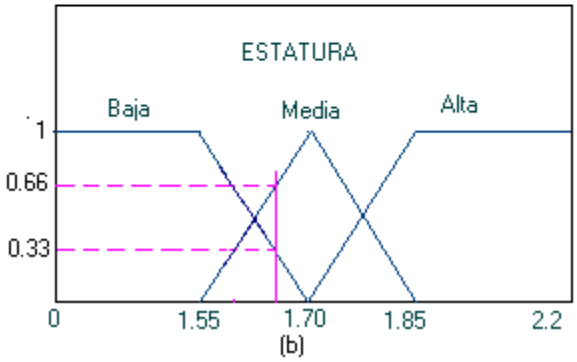


Figura 1. Variable lingüística estatura representada mediante: (a) lógica clásica y (b) lógica difusa.

Se puede apreciar que en la lógica clásica, una persona de 1.65 es considerada de estatura baja, mientras que otra con un (1) milímetro más de estatura, es decir, 1.651 es considerada de estatura media. En la lógica difusa, esa misma persona tiene una posibilidad de ser de estatura baja de 0.33 y de ser de estatura media de 0.66, aproximadamente, es decir tiene más posibilidad de ser de estatura media que de estatura baja, pero tiene pertenencia a dos conjuntos diferentes. De esta manera, la lógica difusa

permite resolver problemas de la vida real con alto nivel de incertidumbre, que difícilmente pueden ser resueltos por la lógica clásica.

La Lógica difusa se desarrolló a partir de 1965 con la aparición de un artículo sobre la teoría de los conjuntos difusos, establecida por el catedrático de la Universidad de Berkeley, California, Lofti A. Zadeh, quien hoy es considerado el padre de la lógica difusa.

Sin embargo, la lógica difusa tiene sus raíces en la lógica multivaluada, cuyos principios fueron desarrollados en los años 20's del siglo pasado por el polaco Jan Lukasiewicks. [10]

2.1.5.1 Funciones de Implicación Difusas.

En la lógica difusa se define la función de implicación mediante la función de los conectores lógicos de la conjunción y la disyunción.

En la lógica difusa al igual que en la lógica clásica existen diferentes tipos de implicaciones como las muestra la tabla 2.2:

Tabla 2.2 Algunas funciones de Implicaciones[11].

Autor	F(p, q)	Año
Lukasiewicz	Min(1, 1-p+q)	1920
Kleene	Max(1-p, q)	1938
Reichenbach	1-p+pq	1935
Zadeh	Max(1-p, min(p, q)	1973
Gödel	$\begin{cases} 1, & sip \le q \\ q & sip \rangle q \end{cases}$	1976
Gaines-Rescher	$\begin{cases} 1 & sip \le q \\ 0 & sip \rangle q \end{cases}$	1969

[10] URUETA, V. Lisbeth. VALDEZ, B. Heydy. CONTRERAS, M. Juan A. LA LOGICA DIFUSA COMO APOYO A LA ENSEÑANZA. Articulo

[11] DIEZ, F. J. Introducción al Razonamiento Aproximado. Departamento de Inteligencia Artificial. UNED. Primera Edición 1998.Revisión: Octubre 2003. P. 119

2.2 TECNICAS DE RAZONAMIENTO

2.2.1 Razonamiento en condiciones de incertidumbre

La Incertidumbre es la falta de información adecuada para tomar decisiones. Existen diferentes técnicas para el manejo de la incertidumbres, entre ellas se puede mencionar "la probabilidad clásica y bayesiana, la teoría de Hartley, la teoría de Shanon, la Teoría de Dempster-Shafer y la teoría de la confusión de Zadeh"[12].

Existen muchos factores que pueden contribuir que un conjunto de datos resulten inciertos a la hora de tomar decisiones que pueden afectar el proceso de inferencia. Entre los tipos de error más comunes se destacan:

- La ambigüedad en el que algo puede ser interpretado o tener más de una explicación.
- La falta de información completa
- La incorrección, donde la información no es correcta.
- Un error sistemático, que es aquel que no es aleatorio sino que se introduce debido a cierta desviación.
- La falta de confiabilidad, debido a datos errados.

"Los sistemas experto o inteligentes pueden estar formados por reglas deductivas o inductivas, donde las reglas inductivas son de naturaleza heurística"[13]. La inducción también puede aplicarse a la generación de regla, ya que esta trata de generalizar de lo particular a lo general.

2.2.1.1 Razonamiento probabilístico

Esta técnica trata con juegos o sistemas de azar. La probabilidad clásica se define mediante la siguiente formula:

$$P=\frac{w}{n} \qquad (2.3)$$

Donde w es el número de éxitos y N es el número de eventos igualmente posibles que son los resultados posibles de un experimento. La probabilidad clásica puede ser empleada para razonamiento en sistemas

[12] GUIARRATANO, Relay. Sistemas Expertos. Principios y Programación. Pág. 165

[13] Ibid;. Pág. 168

determinísticos. Los sistemas **determinísticos** son aquellos en los que al realizar un experimento si se realiza n veces, estas n veces se obtendrá el mismo resultado.

Los sistemas no determinísticos por el contrario son aquellos en los que al realizar un experimento n veces, estas n veces se obtendrán resultados distintos.

La teoría de la probabilidad se basa en tres axiomas principales:

- Axioma 1: 0<= P(E)<=1; rango de probabilidades
  No existe P(E) -1

- Axioma 2: $\sum_i P(E) = 1$, la suma de todos los eventos que no afectan entre si, denominados eventos mutuamente excluyentes, es 1.
  Colorario: P(E)+P(E')=1. esto significa que la probabilidad de que un evento ocurra mas la probabilidad de que no ocurra es igual 1.

- Axioma 3: $P(E_i \cup E_2)=P(E_1)+ P(E_2)$
  $E_i \cup E_2$ Son mutuamente excluyentes

La probabilidad clásica no puede responder a preguntas como cuál es la probabilidad de que su unidad de disco se descomponga mañana o cuál es la expectativa de vida para Juan.

La Probabilidad experimental define la probabilidad de un evento, P(E); como u limite de una distribución de frecuencia:

$$P(E) = \lim_{N->\infty} \frac{f(E)}{N}; \qquad (2.4)$$

donde f(E) es la frecuencia de los resultados de u evento para N resultados totales observados. La idea de esta es medir la frecuencia con que ocurre un evento, durante un gran numero de pruebas y a partir de esta, inducir la probabilidad experimental.

La probabilidad subjetiva trata con eventos que no se pueden reproducir y no tiene base histórica que sirva de extrapolación, realmente es una creencia u opinión expresada como una probabilidad, mas que una probabilidad basada en axiomas o mediciones empíricas.

La Probabilidad compuesta trata con eventos independientes cuando estos no se afectan entre sí de ninguna manera y por eventos estocásticos, que independientemente si y solo si la formula previa es verdadera.

En la Probabilidad condicional, los eventos que no son mutuamente excluyentes se influyen entre si, saber que un evento ha ocurrido puede llevar a revisar la probabilidad de que otro evento ocurra. La Ley Multiplicativa define a la probabilidad de un evento A, dado que ocurrió B y se le representa como muestra la siguiente ecuación.

$$P(A|B)=\frac{P(A\cap B)}{P(B)}\ ;\ P(B)\neq 0, \qquad (2.5)$$

P(B) es la probabilidad a priori.

N(A) o N(B) N(M) = espacio

$$P(A)=\frac{N(A)}{N(M)}=\frac{4}{8} \qquad P(B)=\frac{N(B)}{N(M)}=\frac{6}{8} \qquad (2.6)$$

Para dos eventos $P(A \cap B)=P(A|B)*P(B)$ (2.7)

La probabilidad condicional P(A | B) puede interpretarse como el grado de factibilidad de que A es verdadero dado B, aunque en el sentido clásico no es necesariamente una probabilidad.

Si P(A|B)= 1 se considera que A = V pero si es igual a 0 se considera A =F, mientras que 0<P(A|B)<1, significan que no estamos seguros de que A sea verdadero o falso. En estadística, la hipótesis se usa para algunas proposiciones cuya verdad o falsedad no se conoce con certeza sobre la base de alguna evidencia. En este caso la probabilidad condicional se le llama posibilidad. P(H|E) se le conoce como la posibilidad o grado de certeza.

Probabilidad ≈Eventos Repetibles
Posibilidad ≈ grado de factibilidad de eventos no repetibles.

Factibilidad previa en H es $N(H)\frac{P(H)}{P(H')}$

Factibilidad posterior es: $M(H\mid E)=\frac{P(H\mid E)}{P(H'\mid E)}$

Razón de la posibilidad PS = $\frac{P(E \mid H)}{P(E \mid H')}$

M(H|E)=PS*M(H), donde PS=Posibilidad de suficiencia

Si PS = ∞, entonces la evidencia E es lógicamente suficiente para concluir que H es verdadero. Se tiene que P(H|E) =1 y P(H|E')=0. Un perito humano debe proporcionar los valores de posibilidad *PS* y *PN* para calcular la factibilidad posterior. *PS* muestra como cambia la factibilidad previa cuando la evidencia esta presente. *PN* muestra como cambian cuando esta ausente.

2.2.2. Razonamiento Inexacto

En un sistema basado en Reglas, parte de la validación consiste en minimizar la incertidumbre de las cadenas de inferencia de las cadenas de inferencia, si la verificación puede verse como la disminución de incertidumbres locales, la validación disminuye la incertidumbre global de todo el sistema experto.

La Incertidumbre asociada con la compatibilidad de reglas se da por:

- Contradicción,
- Subordinación,
- Redundancia,
- Reglas ausentes, y
- Fusión de datos.

La contradicción de reglas, varias reglas pueden dispararse con consecuencias contradictorias lo que puede ocurrir si los antecedentes no se especifican de manera apropiada. Ejemplo:

1) Si hay fuego $\Rightarrow$ arrojarle agua
2) Si hay Fuego $\Rightarrow$ No arrojarle agua

Las reglas resultan contradictorias porque no se incluyeron antecedentes que diferenciaran el tipo de material sobre el que se presentaba un fuego o incendio.

Por otro lado, una regla está subordinada a otra si una parte de su antecedente es un subconjunto de otra regla. Ejemplo:

1) Si A $\wedge$B $\wedge$C$\wedge$C $\Rightarrow$ D
2) Si A$\wedge$B $\Rightarrow$ D

Un conjunto de Reglas presenta subordinación cuando existe una que esta incluida o subsumida en otra como se puede ver en las reglas 1) y 2):

1) Si A $\wedge$B $\wedge$C$\wedge$C $\Rightarrow$ D con PS_1
2) Si A$\wedge$B $\Rightarrow$ D con PS_2

Las reglas redundantes tienen igual consecuencia y evidencia. Una regla redundante puede conducir a una mayor eficiencia del sistema si en la lista aparece primero un patrón poco frecuente.

La incertidumbre asociada a la resolución por conflictos es otra de las causas posibles para hallar anomalías en una base de reglas, estas son:

- Prioridad Explicita de Reglas
- Prioridad Implícita de Reglas
    i. Especificidad de patrones
    ii. Hechos recientes que coinciden con patrones
    iii. Ordenamiento de patrones
    iv. Orden en que se introducen las reglas.

3 ALGORITMOS PARA DETECCIÓN Y ELIMINACIÓN DE CONTRADICCIONES E INCONSISTENCIAS EN BASES DE REGLAS CON ANTECEDENTES Y CONSECUENTES DIFUSOS.

Existen dos formas de construir un sistema difuso

- A partir de la información suministrada por un experto (heurística)
- A partir de datos de entrada y salida (identificación borrosa)

En la primera de ellas, la detección de anomalías resulta relativamente fácil y se basa, principalmente, en el ordenamiento y etiquetado de las particiones de los universos de entrada y salida. Las estrategias hasta ahora propuestas para construir sistemas difusos mediante la segunda forma conllevan, por lo general, a sistemas no interpretables y con un elevado número de parámetros pero con una precisión mayor que la de los sistemas construidos mediante la primera forma.

En la búsqueda de técnicas para detección de anomalías en sistemas difusos construidos mediante técnicas de identificación se logró desarrollar un método para construir sistemas borrosos altamente interpretables, con bajo número de parámetros y con un error de inferencia casi nulo, lo que hace innecesario el proceso de detección de anomalías.

En este capítulo se presentarán los algoritmos para detectar anomalías en sistemas difusos heurísticos y, posteriormente, se presentarán los algoritmos para construir sistemas difusos interpretables a partir de los datos, con bajo o nulo nivel de anomalías. Posteriormente se presentarán los resultados en problemas clásicos y la comparación con los resultados obtenidos por otros métodos.

3.1 ANOMALIAS

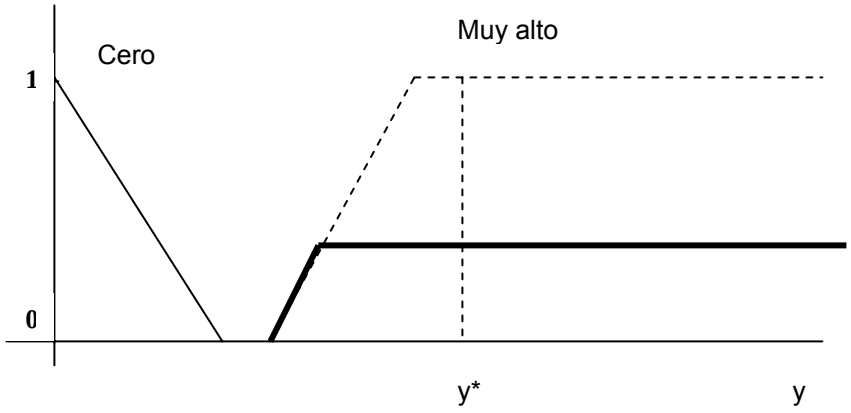
En los sistemas borrosos construidos a partir de la información suministrada por un experto se pueden encontrar, por lo general, las siguientes anomalías:

- Dominancia de los conjuntos grandes
- Reglas contradictorias
- Reglas redundantes o subsumidas
- Espacios no cubiertos del universo de salida (reglas inexistentes)

3.1.1 Dominancia de los conjuntos grandes

La dominancia de los conjuntos grandes produce un cambio inesperado en la salida, como se aprecia en la figura.

Figura 2. Anomalía por Dominancia de los conjuntos grandes.



Como se aprecia en la figura 3.1, la conclusión de una regla Zadhe-Mandani (consecuente "Y es muy Alto", con activación en antecedente 0.1) agregada a la conclusión de otra (consecuente "Y es Cero" con activación de antecedente 1. Al realizar la agregación mediante disyunción y calcular el centro de gravedad el resultado es incorrecto, ya que la conclusión debería ser el conjunto "Y es cero" ya que allí es donde tiene el mayor grado de pertenencia.

Los algoritmos propuestos para la construcción de sistemas borrosos se basan en particiones triangulares suma 1 y en un método de inferencia que impiden la presencia de esta anomalía.

3.1.2 Reglas contradictorias

Una contradicción entre dos reglas, como

$$A_1 \wedge A_2 \Rightarrow B$$
$$A_1 \wedge A_2 \Rightarrow C$$

Puede ser detectada fácilmente mediante una inspección de los consecuentes de reglas que tienen antecedentes iguales. Solo basta con determinar qué reglas con antecedentes idénticos tienen consecuentes disjuntos, los cuales generalmente se encuentran ubicados en posiciones no consecutivas en una distribución matricial.

Sin embargo, es frecuente encontrar sistemas difusos, construidos a partir de datos, que presentan solapamiento en más de dos conjuntos, con lo cual dos conjuntos ubicados en casillas no consecutivas pueden no ser disjuntos, por lo cual es recomendable emplear una métrica (como la medida de entropía de Yager) para determinar previamente el grado de contradicción entre cada uno de los consecuentes. Los algoritmos propuestos para la construcción de sistemas borrosos presenta un entrenamiento basado en el error de inferencia que eliminan la presencia de contradicciones

3.1.3 Reglas redundantes y subsumidas

Si en una base de reglas cuyos antecedentes tienen una partición de tres conjuntos difusos cada uno se encuentran las siguientes reglas

$$\vdots$$
$$A_{1i} \wedge A_{2j} \wedge A_{31} \rightarrow C_i$$
$$A_{1i} \wedge A_{2j} \wedge A_{32} \rightarrow C_i$$
$$A_{1i} \wedge A_{2j} \wedge A_{33} \rightarrow C_i$$
$$\vdots$$

Se debe detectar que las 3 reglas enunciadas pueden ser remplazadas por una sola regla descrita por

$$A_{1i} \wedge A_{2j} \rightarrow C_i$$

Disminuyendo así la carga computacional. Esto es posible hacerlo mediante el la distribución jerárquica de los conjuntos de los antecedentes en un árbol.
Si en la base de datos existen reglas del tipo

$$\vdots$$
$$A_{1i} \wedge A_{2j} \wedge A_{3m} \rightarrow C_i$$
$$A_{1i} \wedge A_{2j} \rightarrow C_i$$
$$\vdots$$

es necesario determinar la existencia de una regla subsumida mediante una estrategia similar a la empleada para el caso de redundancia

Los algoritmos propuestos para la construcción de sistemas borrosos presentan un entrenamiento basado en el error de inferencia que facilita la clasificación o agrupamientos e imposibilita la presencia de redundancias.

3.1.4 Incompletitud

La determinación de la incompletitud de la base de reglas se puede hacer mediante una revisión de la activación de las salidas (etiquetas) ó, si se emplean datos de entrada y salida, mediante una verificación del cubrimiento del espacio de datos de salida por la base de reglas.

En la base de reglas de un modelo o un sistema difuso se pueden encontrar distintos tipos de anomalías como las mencionadas en la sección 2.2.2. Estas anomalías pueden detectarse, básicamente, con el empleo de alguno de los siguientes métodos:

- **Mediante un ordenamiento de la base de reglas.**

- **Mediante una medida de desempeño de cada una de las reglas**. Este caso es aplicable cuando existen datos de entrada y salida del sistema.

El primero de los casos no representa mayor complejidad; sin embargo, no garantiza el reconocimiento de zonas anómalas (contradicciones y redundancias) para todo el rango de valores de los datos de entrada. El segundo método ha sido explorado ampliamente por diversos investigadores sin lograr un método general que pueda ser aplicado con éxito en todo sistema difuso. En este proyecto se ha hecho una lectura y análisis previo de diversos métodos para extracción de reglas, así como para detección de inconsistencias, a partir de datos de entrada y salida, lo cual ha permitido diseñar un novedoso algoritmo que presenta ventajas ante los ya existentes y que será tratado a continuación.

El diseño del algoritmo de detección de inconsistencias en bases de reglas difusas se fundamenta en la medida del error de inferencia de las mismas.

3.2 ALGORITMOS PARA DETECCIÓN DE ANOMALÍAS EN SISTEMAS BORROSOS HEURÍSTICOS

3.2.1 Contradicción.

Como se mencionó en el apartado 2.2.2, la contradicción en bases de reglas puede aparecer si los antecedentes y consecuentes (etiquetas) no se especifican de forma adecuada, en el caso de bases de conocimientos generadas por el experto, o por el empleo de algoritmos para extracción de reglas a partir de datos de entrada y salida que no consideran una medida para la detección automática de anomalías.

A continuación se muestran dos soluciones algorítmicas para solucionar la incoherencia que puede ser la consecuencia de una contradicción en la base de reglas.

Algoritmo 1. Detección y eliminación de contradicción por etiquetas.

Para i = 1
  Para j= i+1
    Repetir (hasta que Antecedentes de la Regla (i) y Regla(j) sean iguales y Consecuentes de Regla (i) y Regla (j) sean diferentes.)

      Si (Antecedentes de la Regla (i)=Antecedentes de la Regla(j)) y (Consecuentes de la Regla(i) ≠ Consecuentes de la Regla (j))entonces “Contradicción”
      Eliminar Regla(j).
      Fin si
    Fin Repetir.
  Fin para.
Fin para.

3.2.2 Redundancia

La redundancia se da en una base de conocimiento, si existen reglas que repiten el mismo antecedente y los mismos consecuentes.

A continuación se muestran dos soluciones algorítmicas para solucionar la redundancia que puede ser la consecuencia de una contradicción en la base de reglas:

Algoritmo 3. Búsqueda de Reglas redundante en una Base de Reglas.

Para i =1
Para j= i+1

Repetir (hasta que Antecedentes de la Regla (i) y Regla(j) sean iguales o diferentes y Consecuentes de Regla (i) y Regla (j) sean iguales.)
Si (Antecedentes de la Regla (i)=Antecedentes de la Regla(j)) o(Antecedentes de la Regla(i)≠ Antecedentes de la Regla(j)) y (Consecuentes de la Regla(i) = Consecuentes de la Regla (j))entonces “Redundancia”

Eliminar de la Base de Reglas las Redundancia.
Fin si
Fin Repetir.
Fin para.
Fin para.

3.2.3 Incompletitud (Reglas ausentes)

Cuando existen datos de entrada para los cuales ninguna regla es activada o que, a pesar de activarse todas las reglas existen valores del espacio de salida que nunca llegan a alcanzarse. Sin embargo, es necesario aclarar que la ultima situación mencionada puede deberse al método de inferencia empleado, lo cual puede a veces corregirse con solo ampliar el soporte de los conjuntos extremos del universo de salida.

Algoritmo 5. Búsqueda de Reglas Ausentes en una Base de Reglas

Dado un vector de entrada[xi] y un vector de salida [yi]
Revisar que todos los datos de yi sean activados.

3.3 ERROR DE INFERENCIA

Los algoritmos desarrollados para la construcción de sistemas borrosos interpretables están basados en el método de error de inferencia planteado por Sala (1998) en su tesis doctoral.

En lógica borrosa es común tratar con reglas del tipo “Si u es A, entonces y es B”, donde u y y representan dos variables numéricas, y $A \subset U$ y $B \subset Y$, son dos conjuntos borrosos de entrada y salida respectivamente, definidos en los universos U y Y. La regla mencionada es equivalente a la inecuación

$$\mu A(u) \leq \mu B(y) \qquad (1)$$

El error de inferencia ε, concebido como la distancia conceptual al conjunto conclusión de la regla, está dado por

$$\in \approx \begin{cases} 0 & \ldots\ldots\ldots\ldots\ldots\ldots\ldots u_A(u) \leq u_A(y) \\ u_A(u) - u_B(y) & \ldots\ldots\ldots u_A(u) > u_B(y) \end{cases} \qquad (2)$$

El conjunto conclusión de la regla está definido por

$$C(u) = \{y^* \in Y \mid u_B(y^*) \geq u_A(u)\} \qquad (3)$$

y la distancia conceptual de un $y \in Y$ al conjunto conclusión está dada por

$$d_{uB}(y, C(u)) = \min_{y^* \in C(u)} \left| u_B(y) - u_B(y^*) \right| \qquad (4)$$

Para una base de reglas N, todas con antecedentes conjuntos borrosos definidos en un universo U y consecuentes conjuntos borrosos definidos en un universo Y, la función de error de inferencia global está dada por

$$\in (u,y) = \left(\sum_{i=1}^{N} \phi_i(u,y) \in_i (u,y)^p \right)^{\frac{1}{p}} \qquad (5)$$

Al conjunto $C^*(u) \subset Y$ que minimiza el error de inferencia global (5) se le denomina conjunto conclusión de la inferencia ideal.

3.4 OBTENCION DE UN MODELO DIFUSO A PARTIR DE DATOS DE ENTRADA Y SALIDA.

El principal aporte de esta investigación es el desarrollo una metodología para la obtención de un sistema difuso a partir de datos experimentales de entrada y salida.

La metodología propuesta tiene tres etapas: en la primera, se emplea el error de inferencia para detectar clases o agrupamientos posibles en los datos; en la segunda, se generan las reglas; en la tercera, se generan la forma, número y distribución de los conjuntos difusos.

Como se ha mencionado existen dos formas básicas para construir un sistema difuso: la primera de ellas se basa en implementar la experiencia de un experto mediante reglas difusas (que implementa el método anterior ítem 3.2); la segunda, consiste en extraer las reglas a partir de datos de entrada y salida. De ésta última se destacan los métodos basados en redes neuronales, algoritmos genéticos y las técnicas de agrupamiento o clasificación. Los algoritmos de agrupamiento borroso representan la técnica más adecuada para la obtención de modelos difusos, siendo los métodos de Fuzzy C-Means (Bezdek, 1987) [12] y de Gustafson-Kessel (Gustafson y Kessel, 1979) [13] los más empleados. Se han realizado diversas variaciones a estos algoritmos de agrupamiento. Nauck y Kruse (1995, 1999) [14][15] proponen técnicas de agrupamiento neuro-difusas; Espinosa y Vandewalle (2000) [11]presentan una metodología para extraer reglas a partir de los datos en un marco de integridad lingüística incluyendo algoritmos de fusión para agrupar conjuntos cuyos valores modales estén a una distancia muy cercana. Sala (1998, 2001) [8] introduce una novedosa técnica basada en el error de inferencia para aproximar funciones empleando partición suma 1 con conjuntos triangulares; Diez et al (2004)[16] proponen variaciones a los algoritmos de agrupamiento para mejorar la interpretabilidad y descubrir estructuras afines locales en los modelos borrosos obtenidos. Paiva y Dourado (2004)[17] presentan un modelo generado por medio del entrenamiento de una red neuro-difusa implementado en dos fases: en la primera fase, se obtiene la estructura del modelo empleando un algoritmo clustering substractivo, lo cual permite extraer las reglas a partir de datos de entrada y salida; en la segunda fase, se realiza la sintonización de los parámetros del modelo mediante una red neuronal que emplea backpropagation. Guillaume y Charnomordic (2004) [18] proponen una estrategia para generar particiones difusas interpretables a partir de los datos en la cual incorporan un algoritmo, que denominan partición jerárquica difusa (HFP), que en vez de incorporar datos en cada iteración agrega conjuntos difusos. También presentan un algoritmo de

fusión de los conjuntos difusos basado en métricas adecuadas que garanticen la interpretabilidad semántica.

3.4.1 Detección de las clases

El número de clases o agrupamientos posibles de los datos experimentales de entrada salida determina el número de reglas de sistema borroso. Para obtener el número de clases posibles en los datos se propone un nuevo algoritmo basado en la extensión cilíndrica de los valores modales de los conjuntos del antecedente (o conjunción de los valores modales de los conjuntos de los antecedentes) en el espacio de datos de salida. Los datos resultantes de la proyección estarán todos sobre la recta z=1, lo que facilita la detección de las clases mediante un algoritmo de agrupamiento jerárquico. El algoritmo requiere previamente: un conjunto de datos experimentales de entrada salida {uk, yk}, una partición inicial suma 1 de la(s) variable(s) de entrada mediante n conjuntos triangulares normales distribuidos uniformemente sobre el espacio de entrada (definido por el vector de datos experimentales de entrada).

Algoritmo 1. Detección de agrupamientos

Dado una colección de datos experimentales {uk, yk} $U \times V$; k =1, ..., Np

- Crear una partición triangular suma 1 del antecedente $A \subset U$ en el rango establecido por los valores mínimo y máximo de los datos de entrada, con un número n de conjuntos borrosos
- Determinar las posiciones en el arreglo de datos de entrada para el cual los valores de pertenencia son iguales a 1 (valores modales)
- Crear una tabla de datos (yj, 1), donde j corresponde a una posición del vector de entrada tal que $u_A(x_j) = 1$
- Determinar las clases o grupos existentes en los datos (cluster jerárquico)
- Incrementar el número de conjuntos n de la(s) variable(s) de entrada y volver al punto (a)
- Seguir incrementando hasta que las clases determinadas no varíen con el incremento de n

El algoritmo entrega el número m de clases que será igual el número de reglas necesarias para aproximar la función

3.4.2 Generación de las reglas

El número clases detectadas representan el número de reglas que debe tener el modelo difuso. Se pueden presentar casos, para controladores de más de una entrada, en que el número de clases es inferior al número de posibles combinaciones de los conjuntos de las entradas, lo cual indica que existirán combinaciones que producen el mismo conjunto conclusión. Por lo tanto, las reglas con igual consecuente se deben resumir en una sola regla mediante la disyunción de las conjunciones de sus antecedentes.

Algoritmo 2. Generación de reglas

Dada m clases y el rango de los universos de entrada y salida

- Hallar el número n de conjuntos para cada una de las q variables de entrada aproximándolo al valor superior más cercano en caso de que el valor resultante sea un fraccionario

$$n = \sqrt[q]{m} \tag{8}$$

- Si el número posible de reglas n^q es igual a m entonces conformar las reglas según las clases a las que quede asignada cada combinación de entrada (ir a algoritmo 1). En caso de que el número posible de reglas sea superior a m indicaría que al menos dos reglas tienen la misma conclusión por lo que se deben resumir en una regla cuyo antecedente sea la disyunción de los antecedentes de las reglas en cuestión. Este caso no se presenta en el caso de sistemas de una sola entrada una sola salida

3.4.3 Generación de la partición de antecedente

El algoritmo anterior nos permite hallar el número de conjuntos de cada variable de entrada pero no la distribución ni las funciones de pertenencia. El algoritmo propuesto realiza una extensión de los conjuntos de la variable de entrada en el universo de salida, asignando el grado de pertenencia de cada elemento xk a su respectiva imagen yk. Las funciones de pertenencia discreta definitivas de los conjuntos difusos de entrada y la distribución en el universo de entrada se generan haciendo una redistribución de los intervalos de evaluación considerados para las funciones discretas triangulares inicialmente supuestas de acuerdo a los intervalos de los datos de salida. Esto, en principio, limita el algoritmo a funciones biyectivas; sin embargo, se puede emplear en caso de funciones no biyectivas dividiendo el problema en modelos locales.

Algoritmo 3. Generación y distribución de funciones de pertenencia

Dado el número n de conjuntos por variables y los rangos de los universos de entrada y salida de los datos experimentales:

- Crear una partición triangular suma 1 del antecedente A $\subset$ U en el rango establecido por los valores mínimo y máximo de los datos de entrada, con un número n de conjuntos borrosos (valor n obtenido en el algoritmo 2). Cada conjunto se define mediante un arreglo del tamaño del vector de datos experimentales de entrada $u_{A_i}(x_k)$
- Proyectar los conjuntos de entrada en el universo de salida mediante la ecuación (6)

$$u_B(y_k) = u_{A_i}(x_k)$$

  creando una tabla de datos (yk, uB(yk))

- Determinar el conjunto de singletons de salida de los valores yk con grado de pertenencia asociado de 1. Estas serán las ubicaciones de los singletons de salida.

  ym = {yk |uB(yk) =1} (9)

- Redistribuir los intervalos de evaluación de acuerdo a los intervalos en que están distribuidos los datos de salida, estableciendo previamente los valores mínimo (ymin) y máximo (ymax) de la variable de salida

$$x_k = \frac{(y_k - y_{min})}{(y_{max} - y_{min})} \quad (10)$$

- Asignar a cada valor de entrada xk el correspondiente grado de pertenencia de yk; o sea

$$u_{A_i}(x_k) = u_B(y_k)$$

  creando una tabla de datos (xk, uAi(yk)) para cada uno de los conjuntos de cada variable de entrada

  La partición generada para el antecedente es suma 1 aunque muy posiblemente no sea triangular.

3.4.4 Método de inferencia

Métodos de inferencia como el del centro de gravedad no son apropiados para aproximación funcional, como se puede apreciar si se trata de aproximar una función tan sencilla como la recta y = x. Por lo tanto se tomará la inferencia propuesta por Sala (1998)

$$y^*(u) = \frac{\sum_{k=1}^{n} u_k^i(u) y_m(i)}{\sum_{k=1}^{n} u_k^i(u)} \qquad (10)$$

donde $u_k^i(u)$ es el grado de pertenencia del dato de entrada en el antecedente i(en el caso de sistemas con más de una entrada corresponderá a la conjunción de las entradas), $y_m(i)$ es el valor del singleton correspondiente a la salida i. El denominador siempre arroja un valor igual a 1 cuando se trata particiones suma 1, el cual es el caso del método de aproximación propuesto en esta investigación..

3.5 RESULTADOS

A continuación se presentan los resultados obtenidos al aplicar el método propuesto a problemas clásicos como el horno de gas de Box_Jenkins [19] y el sistema no lineal multivariable dado por la ecuación $y = (1 + x_1^{-2} + x_2^{-1.5})^2,\ 1 \le x_1, x_2 \le 5$

Ejemplo 3.1: Horno de gas de Box-Jenkins
A partir de un conjunto de datos de un sistema de una entrada, u, y una salida, y:

$$Z^N = \{[u(1), y(1)], [u(2), y(2)], \ldots, [u(N), y(N)]\} \qquad (11)$$

se puede obtener un modelo borroso representado por un conjunto de reglas del tipo

If y(t-1) is A_1i and y(t-2) is A_2i and …
  and u(t-(d)) is B_1i and u(t-(d+1)) is B_2i and …
  then y(t) is C_1i.

donde d representa el tiempo de retardo del sistema

Uno de los problemas clásicos en la modelación e identificación de sistemas es el *horno de gas* planteado por Box y Jenkins. El conjunto de datos está compuesto de 296 pares de datos de entrada-salida. Los datos de entrada corresponden a la rata de flujo de gas que va a ser quemado y los datos de salida a la concentración de dióxido de carbono. El objetivo es predecir la salida usando valores pasados de la entrada y la salida.

Varios autores han trabajado este problema con diferente número de valores pasados de entrada y salida. Gaweda y Zurada [20] emplean, en la ecuación de regresión, las variables u(k-1), u(k-2), u(k-3), y(k-1), y(k-2) y y(k-3), mientras que Pavia y Dourado [17] emplean en la ecuación de regresión solo las variables y(t-1) y u(t-4), logrando un error RMS de 0.390 en la identificación.

Con las variables propuestas por Pavia y Dourado se aplicó el método propuesto y se obtuvo la partición mostrada en la figura 3 y el resultado mostrado en la figura 4. Los singletons del consecuente se ajustaron por mínimos cuadrados en el proceso de entrenamiento y luego se ajustaron los valores modales de los conjuntos difusos empleando gradiente descendiente. La tabla I presenta la base de reglas del modelo difuso y la tabla II muestra una comparación con los resultados obtenidos con otros métodos.

Figura 3. Funciones de pertenencia para el ejemplo de modelación del horno Box-Jenkins

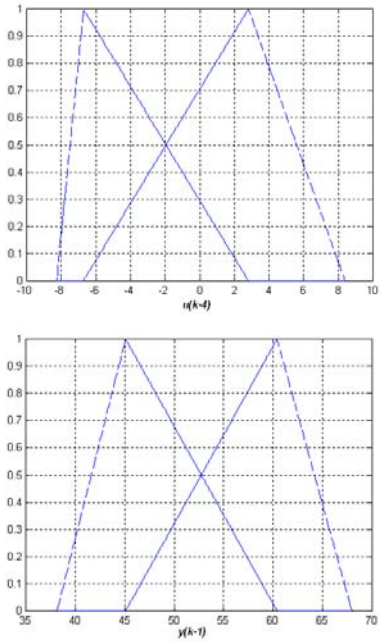
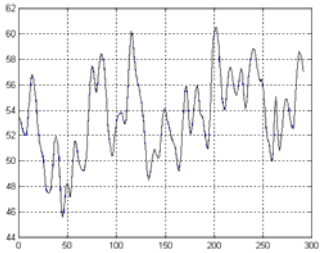


Figura 4. Desempeño del modelo difuso del horno de gas de Box-Jenkins



Los conjuntos de salida son del tipo singleton con ubicaciones en 44.78 y 65.3.

Tabla 3. Descripción lingüística del sistema borroso

Regla	u(k-4)	$\wedge$	y(k-1) $\Rightarrow$ y(k)
1	A	B	44.78
2	B	A	65.30

El modelo difuso obtenido tiene un buen desempeño con un bajo número de parámetros y una adecuada interpretabilidad en el sentido de que un operador humano tendrá un claro significado de las reglas.

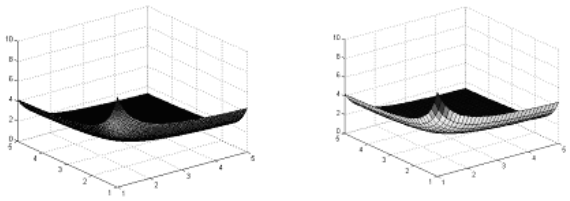
Tabla 3.1. Comparación de resultados de varios métodos al problema del horno de gas de Box-Jenkins

Modelo	MSE	Reglas	Parámetros
Modelo SI de Kim et al (1998) [20]	0.048	2	110
Modelo de Gaweda y Zurada (2003)[18]	0.055	2	38
Nuestro modelo	0.066	2	6
Modelo Sugeno. Wang y Langari (1995) [21]	0.066	2	N/A
Modelo ARMA. Box-Jenkins (1976) [17]	0.202	N/A	N/A

Ejemplo 3.2: Sistema no lineal multivariable

El sistema no lineal descrito por (11) ha sido utilizado por autores como Sugeno y Yukasawa [21], Emami et al [22] y Díez et al [16] para verificar el desempeño de técnicas de identificación borrosa. Para iniciar el proceso de identificación se generaron los datos de entrada salida a partir (11). La figura 5 muestra a la izquierda a la superficie no lineal a identificar y a la derecha el resultado obtenido del modelo borroso.

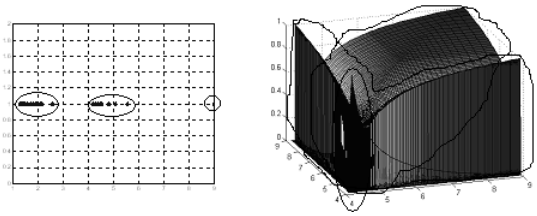
Figura 5. Aproximación a una superficie no lineal



El algoritmo se inició con 2 conjuntos triangulares suma 1 por cada entrada, distribuidos uniformemente sobre el espacio de entrada, definiendo 3 clases o agrupamientos en los datos. A pesar de que en la iteración siguiente se obtuvieron nuevamente 3 clases se procedió a seguir incrementando el número de conjuntos hasta 5 por cada entrada manteniéndose las mismas 3 clases. Este ejercicio se hizo solo para verificar los resultados del método.

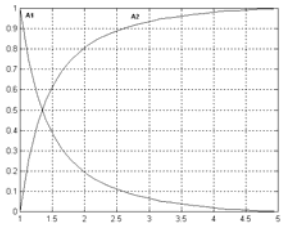
En la figura 6 se muestra a la derecha la ubicación de la extensión de los valores modales, resultados de las conjunciones de las entradas, en el espacio de salida. A la izquierda se realizó la extensión de todas las combinaciones de los valores modales de las entradas en el espacio de salida mostrando también las 3 clases. Esto indica que la aproximación se puede hacer con solo 2 conjuntos por entrada

Figura 6. Clases detectadas en los datos de la superficie no lineal



La figura 7 muestra las funciones de pertenencia generadas y su distribución en el espacio de entrada. La partición de las variables de entrada se generó mediante la proyección del espacio de salida en los universos de las entradas.

Figura 7. Partición generada para el antecedente



La ubicación de los singleton de salida quedó establecida en: $y_1 = 9$; $y_2 = 4.1$; $y_3 = 1.31$. La base de reglas quedó de la forma:

$$A_1 \wedge A_1 \rightarrow y_1$$
$$(A_1 \wedge A_2) \vee (A_2 \wedge A_1) \rightarrow y_2$$
$$A_2 \wedge A_2 \rightarrow y_3$$

4. DISEÑO DE LA APLICACIÓN SOFTWARE PARA DETECCION, ELIMINACION DE ANOMALIAS EN BASES DE REGLAS DE SISTEMAS DIFUSOS

La aplicación resultado de esta investigación se describe de esta manera:

1. El primer panel de la aplicación permite al usuario crear un modelo difuso “desde 0”, introduciendo sus variables de entrada y salida, la configuración de los conjuntos de cada variable y la base de reglas partiendo del conocimiento previo del comportamiento del sistema.

   La siguiente acción de este panel es Detectar las anomalías (Redundancias, Contradicciones), que se hallen en la base de reglas, utilizando los métodos antes mencionadas. Por ultimo, esta aplicación mide la eficiencia de la base de reglas creada.

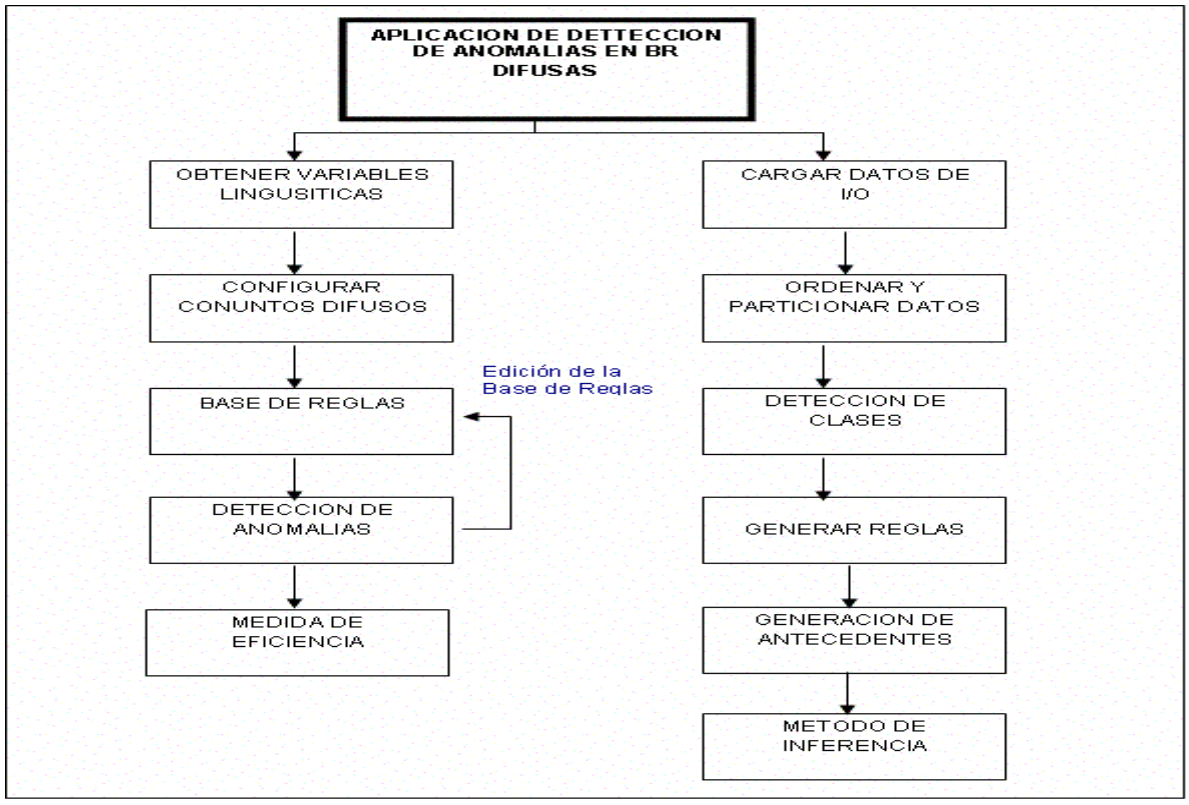
2. El segundo panel permite obtener un modelo de un sistema difuso a partir de los datos de entrada y salida, donde primero se detectan las clases, generan las reglas y las funciones de pertenencias y el método de inferencia.

Teniendo los dos modelos difusos utilizando las estrategias para eliminar redundancias de las bases de reglas de los mismos y con las medidas de eficiencia utilizadas se llegará a la conclusión de cuál es mejor y cuál

utilizar debido a la optimidad en las salidas del sistema debido a la base de reglas que es eficiente debido a contener pocas anomalías.

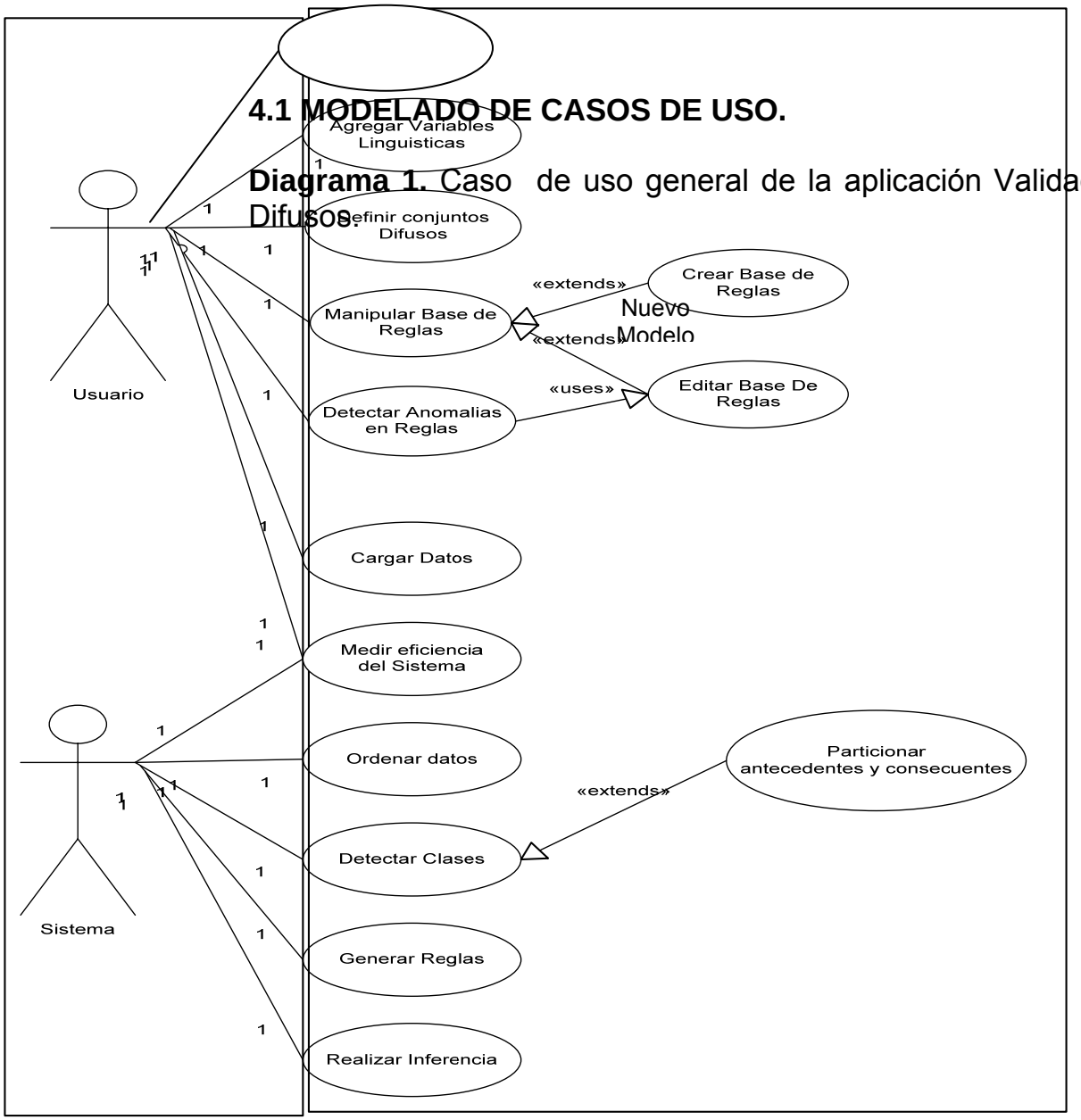
La estructura de la aplicación es la que muestra la figura 4.1

Figura 8. Estructura de la aplicación Validador Modelos Difusos.



4.1 MODELADO DE CASOS DE USO.

Diagrama 1. Caso de uso general de la aplicación Validador de Modelos Difusos.



4.1.1 Descripción de los actores.

Actor	**Usuario**
Caso de Uso	Agregar Variables, Definir Conjuntos Difusos, Manipular Base de Reglas, Detectar Anomalías en Reglas, Cargar Datos, Medir eficiencia**.**
Tipo	Primario
Descripción	Es una persona que con conocimientos en Lógica Difusa que es capaz de crear un Modelo y analizar la Base de reglas.

Actor	**Sistema**
Caso de Uso	Medir Eficiencia, Ordenar Datos, Detectar Clases, Generar Reglas, Realizar Inferencias
Tipo	Primario
Descripción	Es el mismo sistema que al recibir la carga de los datos de un archivo se encarga de realizar el modelo a partir de los datos de entrada.

4.1.2 Descripción de los casos de uso.

Caso de Uso	Nuevo Modelo
Actores	Usuario
Tipo	Básico
Propósito	Crear modelo con su número de entradas y salidas.
Resumen	El usuario inicia este caso de uso, escogiendo la opción nuevo del menú archivo de la barra de menús de la aplicación.
Precondiciones	Se requiere haber ejecutado el programa desde el icono de Validador o desde programas.
Flujo Principal	Se presenta al usuario la pantalla principal (P-1) el usuario seleccionar de la barra de menú Archivo, la opción nuevo . Al seleccionar esta, aparecerá la pantalla “Nuevo” (P-2) donde se colocara el nombre del controlador y el numero de variables de entrada.
Subflujos	Ninguno
Excepciones	Ninguno

Caso de Uso	Crear variables Lingüísticas
Actores	Usuario
Tipo	Básico
Propósito	Configurar y crear variables de entrada y salida
Resumen	El usuario inicia este caso de uso. Tiene la capacidad de darle nombre a cada variable y guardar los datos de las variables incluyendo el punto de evaluación y el Nº de conjuntos
Precondiciones	Se requiere haber creado un modelo anteriormente.
Flujo Principal	Se presenta este caso de uso una vez se ha creado el nombre de modelo y se han determinado las variables de entrada y salida. Se presenta al usuario la pantalla (P-3)Una vez que se han guardado las Variables de entrada, se selecciona la pestaña “variables de salida” que permitirá escribir la información de las Variables de salida. Después de guardar las variables (S-1) aparecer un cuadro de dialogo informando que a sido guardado, si se desea definir los conjuntos de cada una de las variables, se da clic sobre el hipervínculo “definir Conjunto”(S-2) y se activa el caso de uso “definir conjunto”. Si el usuario desea ver las reglas se podrá activar desde el caso de uso Ver reglas(S-3) que permite editar las reglas.
Subflujos	S-1, Guardar variables S-2, Definir conjuntos. S-3, Ver Reglas
Excepciones	Ninguno

Caso de Uso	Ver variables Lingüísticas
Actores	Usuario
Tipo	Incluido
Propósito	Ver las variables creadas y editarlas.
Resumen	El usuario inicia este caso de uso. Tiene la capacidad de editar toda la información de las variables de entrada y salida.
Precondiciones	Se requiere haber creado las variables de entrada y salida.
Flujo Principal	Se presenta este caso de uso una vez que el usuario elija la opción ver reglas, que le presentara la pantalla (P-4), al realizar las ediciones si es el caso, se tendrán que cambiar los conjuntos, ya que la información del modelo tendrá que

	cambiar.
Subflujos	S-1, Definir conjuntos.
Excepciones	Ninguno

Caso de Uso	Definir Conjuntos difusos
Actores	Usuario
Tipo	Básico
Propósito	Configurar los rangos o universos de los conjuntos de las variables lingüísticas.
Resumen	El usuario inicia este caso de uso. Ofrece la funcionalidad para configurar, modificar y borrar conjunto para las variables lingüísticas de entrada y Salida.
Precondiciones	Se requiere haber creado las variables lingüísticas de entrada y Salida.
Flujo Principal	Se presenta al usuario la pantalla (P-4) en la que el usuario escogerá la variable a la que le configurar los conjuntos, de esta forma se guarda el nombre del conjunto y los parámetros del mismo. Al hacer clic sobre el hipervínculo “Guardar Conjunto” se irán adicionando a la lista que esta en la pantalla
Subflujos	Ninguno
Excepciones	Ninguno

Caso de Uso	Construir reglas
Actores	Usuario
Tipo	Básico
Propósito	Crear, editar las reglas de la base de conocimiento.
Resumen	El usuario inicia este caso de Uso. Ofrece la funcionalidad para Editar crear y editar las reglas.
Precondiciones	Se requiere haber creado con anterioridad los conjuntos difusos para cada variable.
Flujo Principal	Se presenta al usuario la pantalla (P-5).en la que selecciona de un árbol los conjuntos que harán parte del antecedente y del consecuente de las reglas de la base de conocimiento. Después de creadas las reglas se puede proceder a ver las reglas (S-3)

Subflujos	Ninguno
Excepciones	(E-1).*Variables o Conjuntos No creados:* Falta crear las variables y conjuntos de las variables para proceder a construir las reglas.

Caso de Uso	Ver Reglas
Actores	Usuario
Tipo	Incluido
Propósito	Mostar las reglas creadas al usuario.
Resumen	El usuario inicia este caso de uso. Su funcionalidad es para ver el resumen de las reglas que componen la base de conocimiento del Modelo.
Precondiciones	Se requiere con anterioridad haber creado la base de reglas del modelo.
Flujo Principal	El usuario escoge desde el Menú Reglas de la Barra de Menú la opción Ver reglas al hacer clic sobre esta se presenta la pantalla (P-9)
Subflujos	Ninguno
Excepciones	Ninguno

Caso de Uso	Detectar Anomalías
Actores	Usuario
Tipo	Incluido
Propósito	Encontrar las reglas que tienen consecuentes y antecedentes que no hacen de la labor optima en la base de conocimiento.
Resumen	El usuario inicia el caso de uso. Tiene la capacidad de seleccionar como se harán las detecciones si por Contradicciones o por Redundancias.
Precondiciones	Es requerida la creación de una base de reglas o tener abierto un modelo que posea una base de conocimiento.
Flujo Principal	El usuario selecciona del Menú Reglas el submenú Detección de Anomalías que le permite al usuario seleccionar como comenzara a realizar la detección: Por Contradicción o por Redundancia
Subflujos	Ninguno
Excepciones	Ninguno

Caso de Uso	Contradicciones
Actores	Usuario, Sistema
Tipo	Extendido
Propósito	Encontrar en la base de conocimiento las reglas que presenten consecuentes iguales y antecedentes iguales.
Resumen	Este caso de uso lo inicia el usuario indicando que tipo de anomalías deberá el sistema buscar, si contradicciones por consecuentes iguales o antecedentes iguales.
Precondiciones	Es requerido que el usuario haya creado una base de reglas y se haya seleccionado la opción Contradicciones del Menú Detección de Anomalías.
Flujo Principal	Al usuario se le presenta la pantalla Contradicciones (P-7), y selecciona por checkbox la opción si será por Consecuentes iguales o si por antecedentes iguales. Luego se le permitirá seleccionar las reglas con estos rasgos y eliminarlas de la base
Subflujos	Ninguno
Excepciones	(E-1).*Acción No permitida*: El usuario debe dejar por lo menos una regla en la lista para poder realizar la inferencia del sistema.

Caso de Uso	Redundancias
Actores	Usuario, Sistema
Tipo	Extendido
Propósito	Eliminar las reglas que presenten redundancias.
Resumen	Este caso de uso lo inicia el usuario indicando que tipo de anomalía deberá el sistema buscar, en este caso Redundancias, por antecedentes y consecuentes iguales o reglas subsumidas.
Precondiciones	Es requerido que el usuario haya creado una base de reglas y se seleccione la opción Redundancia del Menú Detección de Anomalías.
Flujo Principal	Al usuario se le presenta la pantalla Redundancias (P-8), y selecciona por checkbox la opción si será por Consecuentes y antecedentes iguales. Luego se le permitirá seleccionar las reglas con estos rasgos y eliminarlas de la base de reglas.
Subflujos	Ninguno
Excepciones	(E-1) *Acción No permitida*: El usuario no puede borrar todas

	las reglas por que esto no generara salidas en la inferencia del sistema.

Subsistema de Generación de sistemas difusos a partir de los datos

Caso de Uso	Cargar Datos
Actores	Usuario
Tipo	Básico
Propósito	Seleccionar un archivo de datos de entrada y salida
Resumen	Este caso de uso lo inicia el usuario haciendo clic sobre la opción Importar datos, al ser seleccionado permite escoger un archivo plano del tipo de datos de entrada y salida y descargarlo en una Matriz y luego usar la información.
Precondiciones	Para que este caso de uso funcione es necesario que se seleccione un archivo de texto del tipo de datos de entrada y salida. Y con más anterioridad el usuario ha tenido que iniciar la aplicación.
Flujo Principal	Al usuario se le presenta la pantalla principal y este debe seleccionar la opción importar datos para que se carguen los datos del archivo con que se va a trabajar.
Subflujos	(S-1) Se activa el ordenamiento de los datos.
Excepciones	(E-1) El archivo no tiene el formato para cargar los datos

Caso de Uso	Ordenar Datos
Actores	Sistema
Tipo	Incluido
Propósito	Ordenar los datos del archivo en forma descendente
Resumen	Este caso de uso lo invoca el caso de uso anterior (Cargar Datos), para que se guarden en la matriz los datos ordenados con respecto a los datos de entrada.
Precondiciones	Es requisito que se haya realizado la carga de los datos del archivo de datos de entrada y salida.
Flujo Principal	Después de cargado la aplicación detecta la cantidad de entradas en el archivo y ordena los datos de la primera

	entrada y los demás con respecto a la primera.
Subflujos	
Excepciones	Ninguno

Caso de Uso	Detectar Clases
Actores	Sistema
Tipo	Incluidos
Propósito	Divide los datos para crear conjuntos en el rango de los datos de entrada y así crear agrupamientos
Resumen	Este caso de uso es iniciado por el sistema, una vez que se ha hecho el ordenamiento de los datos del archivo de entrada y salida.
Precondiciones	Es requisito que se haya ordenado el archivo con anterioridad.
Flujo Principal	Dividir los datos de entrada en 2 conjuntos difusos de partición suma 1.
Subflujos	(S-1) Particionar Antecedentes y consecuentes.
Excepciones	Ninguno

Caso de Uso	Particionar Antecedentes y consecuentes
Actores	Sistema
Tipo	Incluido
Propósito	Aumentar el numero de conjuntos de las variables de entrada hasta conseguir un valor de error menor que el esperado.
Resumen	Incrementar el número de conjuntos n de la(s) variable(s) de entrada y Seguir incrementando hasta que las clases determinadas no varíen con el incremento de n
Precondiciones	Es requisito que se haya hecho con anterioridad la Detección de Clases.
Flujo Principal	Incrementa el número de conjuntos difusos de partición suma 1, para lograr llegar al punto de error menor que el establecido para encontrar un modelo ideal.
Subflujos	Ninguno
Excepciones	Ninguno

Caso de Uso	Generar Reglas
Actores	Sistema
Tipo	Incluido
Propósito	Crear las reglas del sistema teniendo en cuenta el valor modal y su singleton correspondiente.
Resumen	Este caso de uso lo lleva a cabo el mismo sistema, cuando después de crear los conjuntos y ubicar los singleton correspondientes
Precondiciones	Es requisito haber creado los conjuntos de las entradas y ubicar los singleton.
Flujo Principal	El caso de uso genera las reglas a partir de las pertenencias de los valores modales en los conjuntos particionados de las variables de entrada y salida.
Subflujos	(S-1) Realizar la siguiente partición de conjuntos si el error no llega a ser menor o igual a esperado.
Excepciones	Ninguno

4.2 MODELADO DE INTERFACES

Teniendo en cuanta las funcionalidad del sistema descrita en los casos de uso, a continuación se muestran las pantallas diseñadas y desarrolladas en la aplicación

Figura 9. Pantalla Principal (P-1)

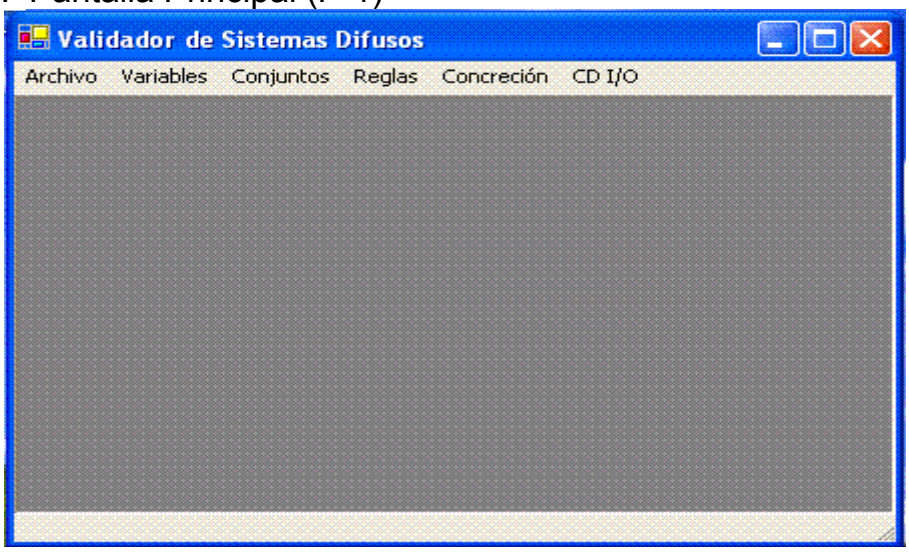


Figura 10. Pantalla Nuevo Modelo (P-2)

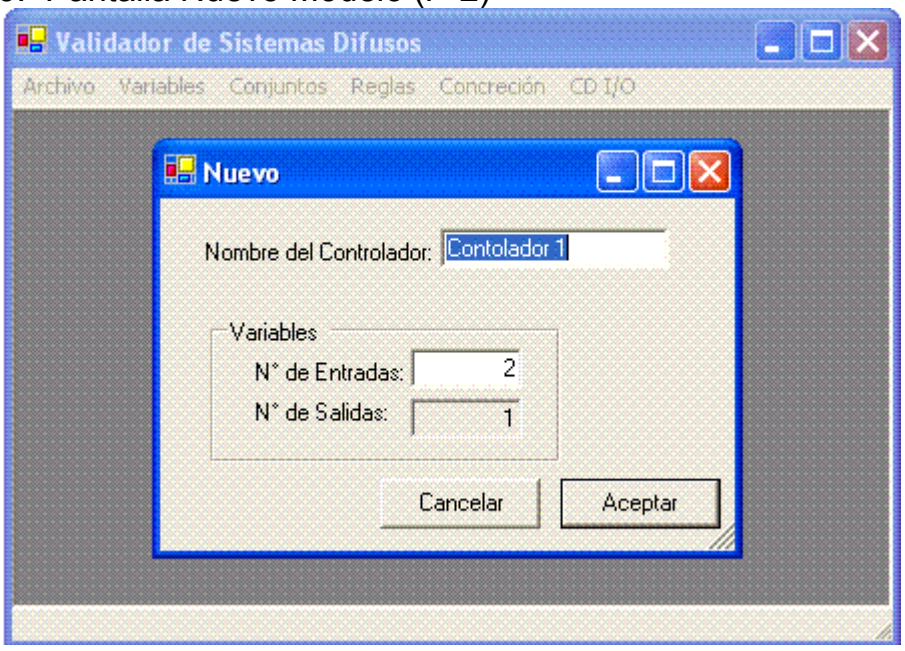


Figura 11. Pantalla Variables (P-3), desplegando la opción de Variables de entrada (P-3.1)

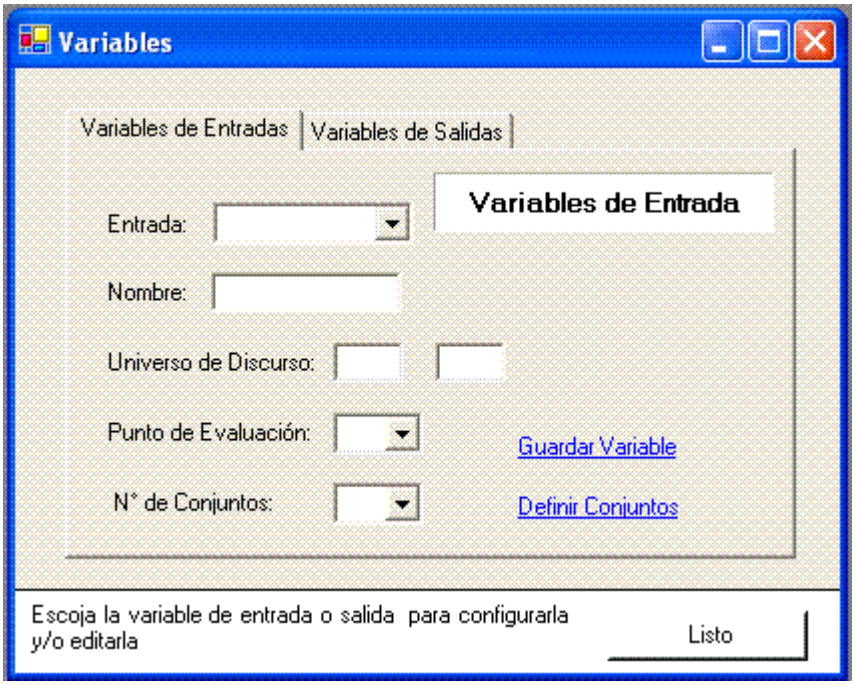


Figura 12. Pantalla Variables (P-3), desplegando la opción de Variables de Salida(P-3.2)

.

Figura 13. Pantalla Conjuntos Difusos (P-4)

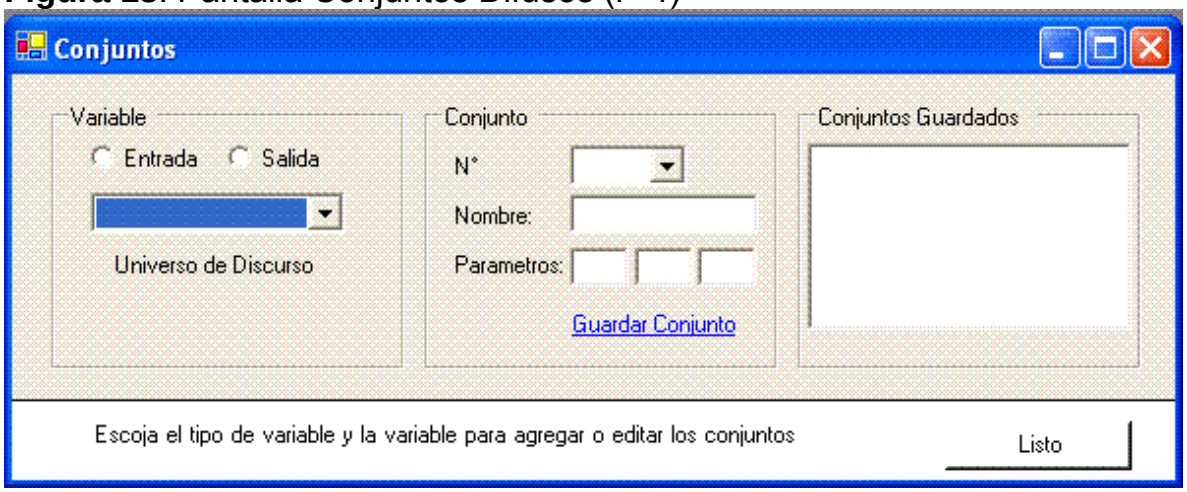


Figura 14. Pantalla Ver Variables

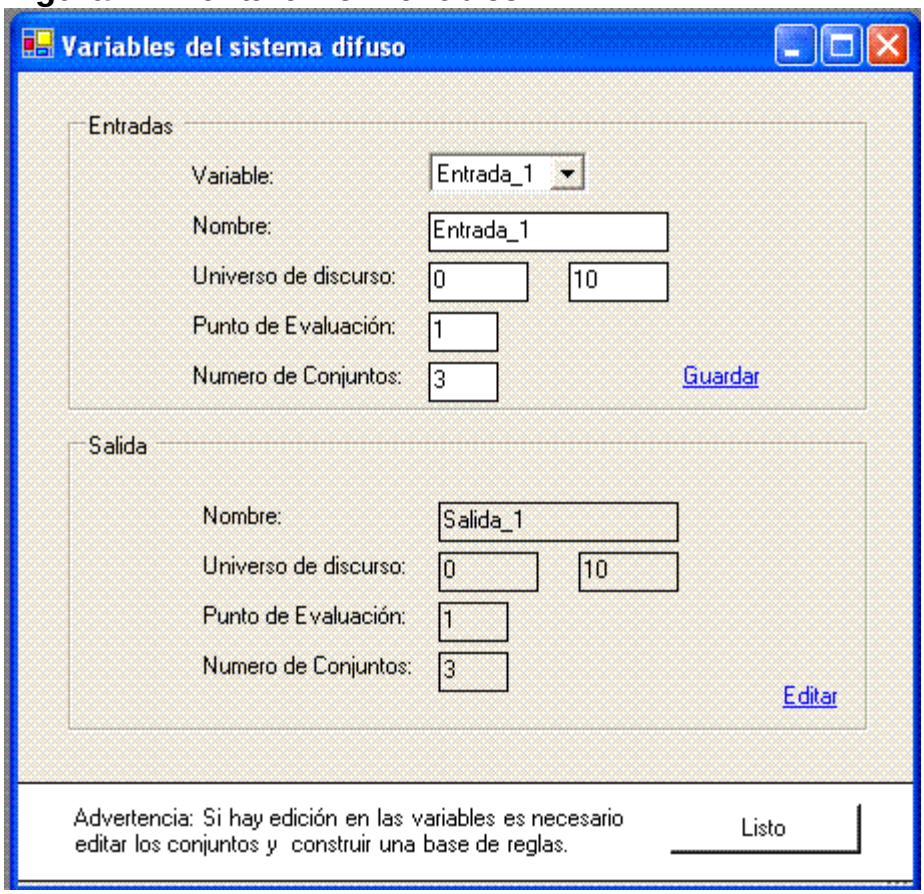


Figura 15. Pantalla Ver Conjuntos

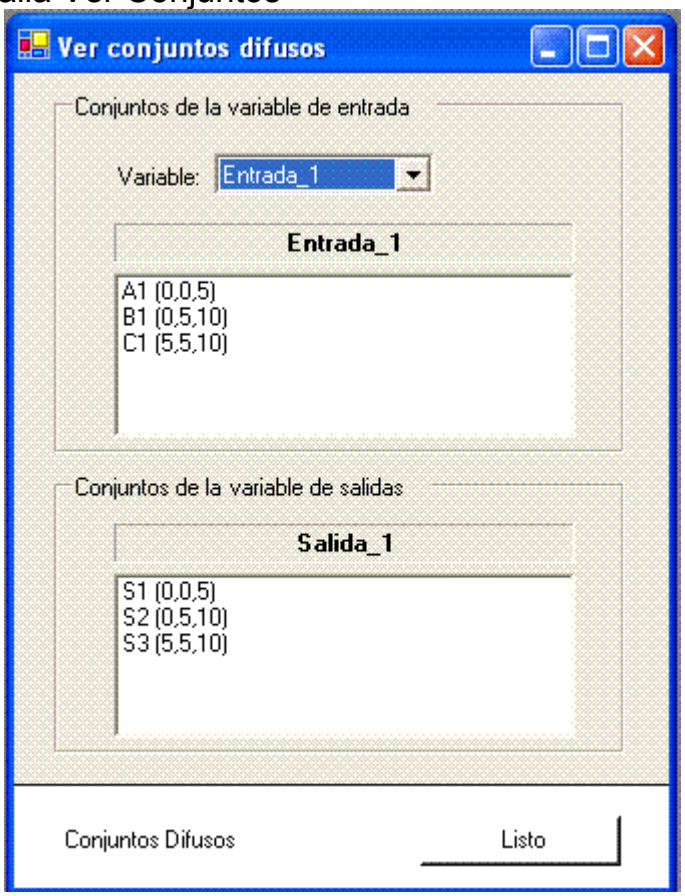


Figura 16. Pantalla Construir Reglas (P-5)

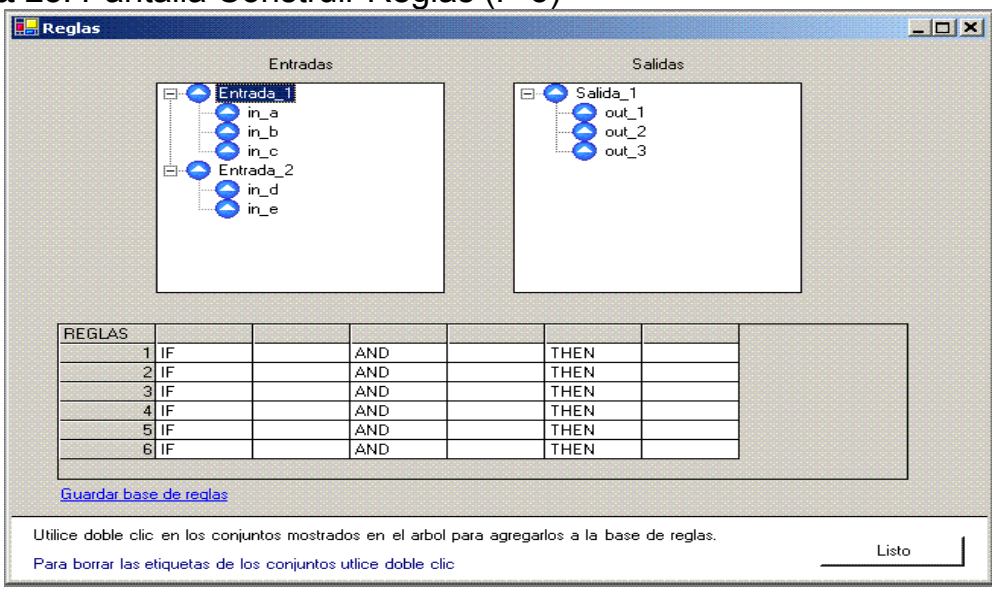


Figura 17. Pantalla Detección de Anomalías -Contradicciones (P-6)

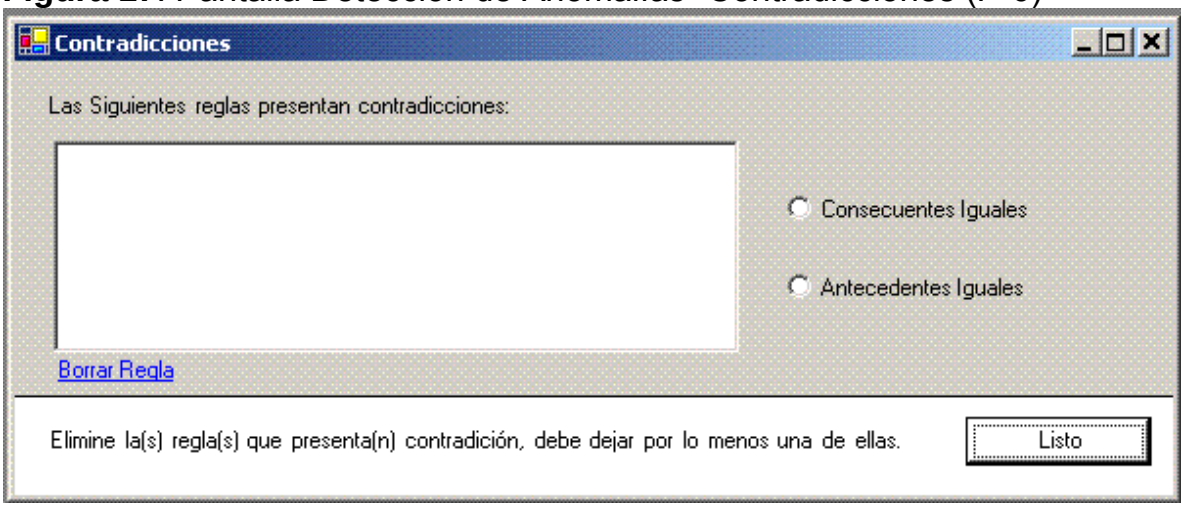


Figura 18. Pantalla Detección de Anomalías – Redundancias(P-7)

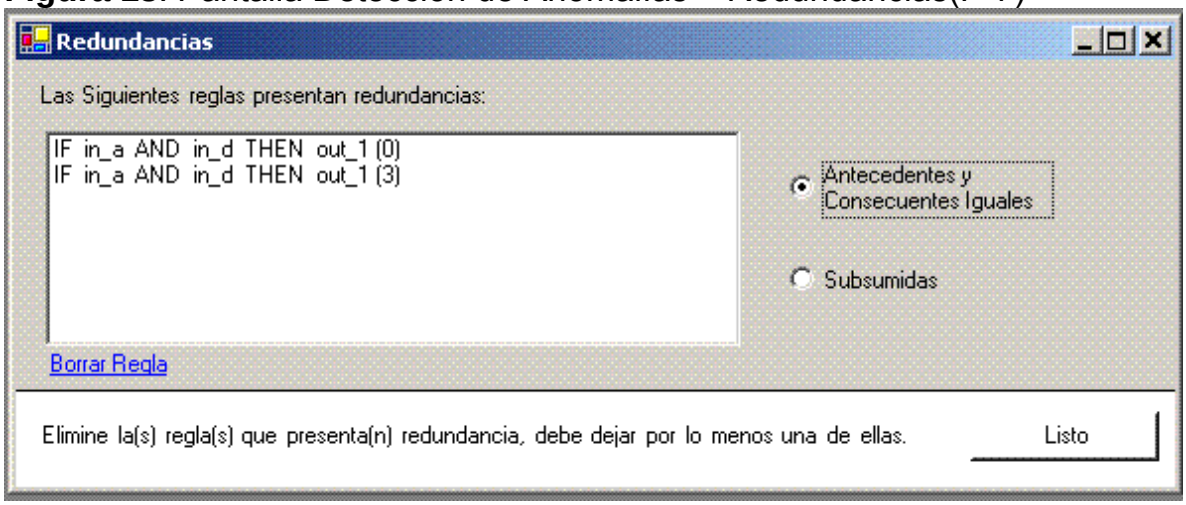


Figura 19. Pantalla Ver Reglas (P-8)

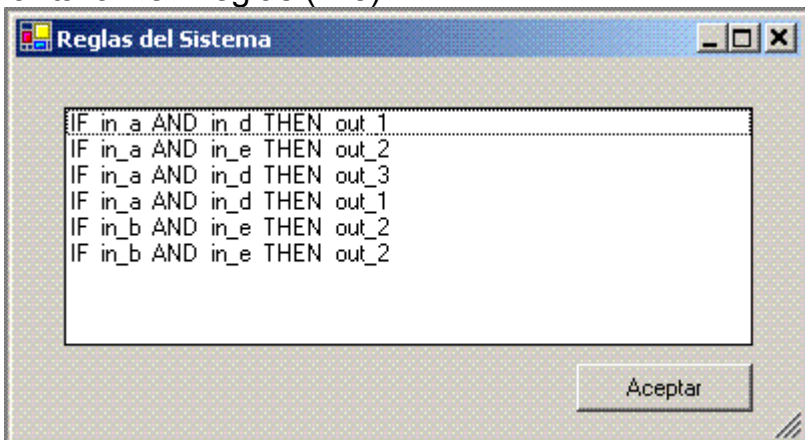


Figura 20. Pantalla de entradas en la importación de los datos

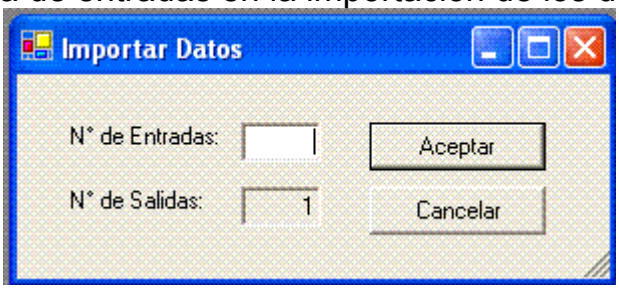
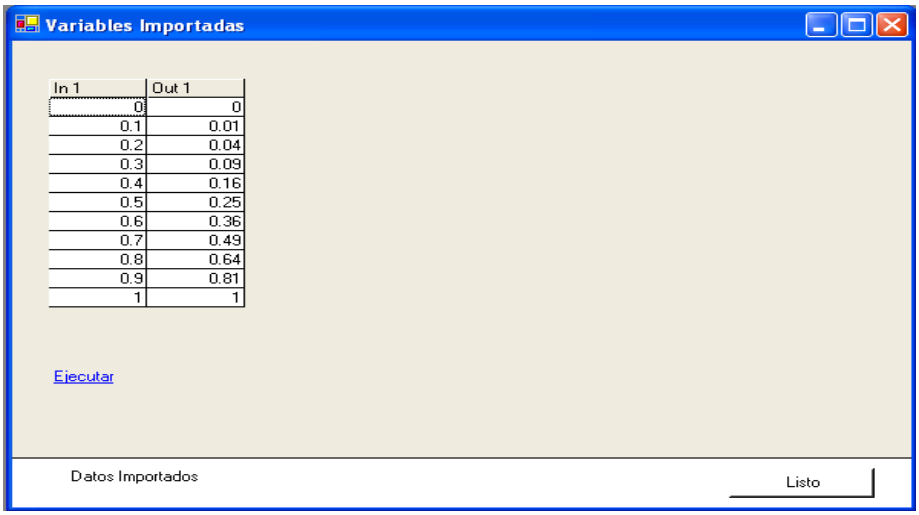


Figura 21. Pantalla de Generación del Modelo Difuso validado.



CONCLUSION

Se presenta un método(segundo metodo, capitulo 3) basado en la minimización del error de inferencia para la identificación de sistemas mediante modelos borrosos interpretables, generando partición suma 1 de los antecedentes a partir de una partición triangular suma 1 inicialmente supuesta. Se comprobó la utilidad del sistema en la aproximación de funciones no lineales unidimensionales y multidimensionales.

Mediante el segundo método expuesto se pueden determinar las clases o agrupamientos en los datos experimentales, lo cual permite el cálculo del número de reglas borrosas y la generación del número de conjuntos posibles en cada antecedente.

El segundo método ha resultado útil para identificar sistemas o funciones monótonas pero puede ser extendido a otro tipo de funciones haciendo previamente una partición de los datos en subconjuntos que representen funciones monótonas. Lo anterior se debe a que el método exige que las funciones que representan los datos sean biyectivas.

REFERENCIAS BIBLIOGRAFICAS

[1] Suwa, M., Scott, A. y Shortliffe, E. (1982). An approach to verifying completeness and consistency in a rule-based expert system. AI Magazine, 3(4), 16-21.

[2] Nguyen, T., Perkins, W., Laffey, T. y Pecora, D. (1985). Checking and expert system knowledge base for consistency and completeness. In Proc. of IJCAI-85 (pp. 375-378)

[2] Nguyen, T., Perkins, W., Laffey, T. y Pecora, D. (1987). Knowledge base verification. AI Magazine, 8(2), 69-75.

[3] Meseguer, P. (1990). A new meted to checking rule bases for inconsistency: A Petry net approach. In Proc. of ECAI'90 (pp. 437-442).

[4] Meseguer, P. y Plaza, E. (1992). Validation of KBS: The VALID project. In L. Steels y B. Le Pape (Eds.), Enhacing the knowledge engineering process (p. www.iiia.csic.es/People/enric/valid-kbs.html). North-Holland Elsevier.

[5] Yager, R. y Larsen, H. (1991). On discovering potential inconsistencies in validating uncertain knowledge bases by reflecting on the input. IEEE Transactions on Systems, Man and Cybernetics, 21(4), 790-801.

[6] Dubois, D. y Prade, H. (1994). On the validation of fuzzy knowledge bases. In S. Tafestas y A. Venetsanopoulos (Eds.), Fuzzy reasoning in information, decision and control systems (pp. 31-49). Dordretch, Germany: Kluwer.

[7] Dubois, D., Prade, H. y Ughetto, L (1997). Checking the coherence and redundancy of fuzzy knowledge bases. IEEE Transaction on Fuzzy Systems, 5(3).

[8] Sala, A. (1998). Validación y Aproximación Funcional en Sistemas de Control Basados en Lógica Borrosa. Universidad Politécnica de Valencia. Tesis Doctoral.

[9] Kim, G. (1998). A Model Validation Methodology for Isolating Knowledge Between Fuzzy Rule-Based and Quantitative Models Usig Fuzzy Simulation. University of Florida. PhD thesis.

[10] Lee, H.-M., Chen, J.-M. y Liu, C.-L. (2002). Inconsistency resolution and rule insertion for fuzzy rule-based systems. Journal of Information Science and Engineering 18, 187-210.

[11]Espinosa, Constructing Fuzzy Models From Input-Output Data. (2000).
[12] Bezdek J. C. (1987). Pattern recognition with Fuzzy Objective Function Algortithms. Ed. Plenum Press.

[13] Guztafson E. E., Kessel W. C. (1979). Fuzzy Clustering with a Fuzzy Covariance Matrix. IEEE CDC, San Diego, California, pp. 503 – 516.

[14] Nauck, D., Kruse, R., "Nefclass - a neuro-fuzzy approach for the classification of data", In Proceedings of the Symposium on Applied Computing, 1995.

[15] Nauck, D., Kruse, R., “Neuro-fuzzy systems for function approximation”. Fuzzy Sets and System. 101(2), pp. 261-271. Jan. 1999.

[16] Díez J. L., Navarro J. L., Sala A. (2004). Algoritmos de Agrupamiento en la Identificación de Modelos Borrosos. RIAI: Revista Iberoamericana de Automática e Informática Industrial

[17] Paiva, R. P., Dourado, A., “Interpretability and learning in neuro-fuzzy systems”, Fuzzy Sets and System. 147(2004), pp. 17-38. 2004.

[18] Guillaume, S., Carnomordic, B., “Generating an interpretable Family of Fuzzy Partitions Form Data”, IEEE Trans. Fuzzy Systems, vol. 12, No. 3, pp. 324 – 335, Jun. 2004.

[19] Box, G., Jenkins, G., Times Series Analysis, Forecasting and control, 2nd ed. Holden Day, San Francisco, 1976

[20] Gaweda, A., Zurda, J., “Data-Driven Linguistic Modeling Using Relational Fuzzy Rules”, IEEE Trans.Fuzzy System, vol 11, pp. 121-134, Feb.2003

[21] Sugeno, M., Yasukawa, T., “A fuzzy logic based approach to qualitative modeling”. Transactions on Fuzzy Systems, vol. 1, No. 1, pp. 7-31. 1993

[22] Emami, M., Turksen, I., Goldenberg, A., “Development of a systematic methodology of fuzzy logic modeling. Transaction of Fuzzy Systems, vol. 6, No. 3, pp.346-360. 1998.

BIBLIOGRAFIA

- Aström, K. (1995). Hybryd systems. (Notes Colloquium of the EHCM Network: Application of Nonlinear and Adaptative Control to Physical Systems. Rome)

- Berenji, H., y Khedkar, P. (1992). Learning and tuning fuzzy logic controllers through reinforcements. ITNN, 3(5), 724-740.

- Aström, K., J., A. y Arzén, K.-E. (1986). Expert control. Automática, 22(3), 227-286.

- Dubois, D, Prado, H(1980) Fuzzy Sets in Approximate Reasoning: A Personal View. Academic Press. New York

- Dubois, D., Prade, H. (1998). Soft computing, fuzzy logic and artificial intelligence. Soft Computing 2. 7-11.

- Hájek, P. (1998). Metamathematics of Fuzzy Logic. Kluwer Academic Publisher. Dordrecht, The Netherlands.

- Hillera, J. y V. Martínez (2000). Redes Neuronales Artificiales, Alfaomega, México.

- Jantzen, J. (1998) The Self Organising Fuzzy Controller. Technical University of Denmark, Department of Automation. Tech. report no 98-H 869 (soc).

- Jiménez, S y Roman, E.(2000).Agentes Inteligentes. Universidad Central de Venezuela, Facultad de Ciencias, Escuela de Computación. ISSN 1316-6239

- Knapp, T. y Iserman, R. (1990). Supervision and coordination of parameter-adaptive controllers. In Proc. automat. Control conf. (pp. 1632-1637)

- Kosko. B. (1992). Neural networks and fuzzy systems. A dynamical systems approach to machine intelligence. Englewoods Cliff, New Jersey: Prentice Hall.

- Lin, C.-T. y Lee, G. (1991). Neural network-based fuzzy logic control and decision system. IEEE transactions on Computers – Special Issue on Artificial Neural Networks, 40(12), 1320-1336.

- Lukasiewickz, J. (1920). Logike trojwartoscieweg. Ruch. Filosofiece, 169.

- Morari, M. y Zafirou, E. (1989). Robust process control. Englewood Cliffs, New Jersey: Prentice Hall.

- Olivas V, J.A. La Lógica Borrosa y sus Aplicaciones

- Passino, K.M. and S. Yurkovich (1998). Fuzzy Control. Addison Wesley Logman, Inc., California

- Pedrycz, W. (1991). Neurocomputations in relational systems. IEEE Transactions on Systems, Man and Cybernetics, 26(4), 627-636

- Procyk, T. y E. H. Mamdani, (1978), a linguistic self-organizing process controller, Automatica 15. !5:30.

- Zadeh, L. (1965). Fuzzy sets. Information and Control, 8, 338-353

- Zadeh, L. (1973). Outline of a new approach to the analysis of complex system and decision processes. IEEE Transactions on Systems, Man and Cybernetics 3. 28-44.

ANEXOS

ANEXO 1:
ARTICULO CIENTIFICO

“Algoritmos de Aproximación Funcional Para Generación de Sistemas Difusos Interpretables”

Algoritmos de Aproximación Funcional Para Generación de Sistemas Difusos Interpretables

Juan Contreras Montes, Roger Misa Llorca, Lisbeth Urueta Vivanco [*]

Escuela Naval Almirante Padilla, Cartagena, Colombia
Instituto Superior Politécnico José Antonio Echaverría, La Habana. Cuba
Corporación Universitaria Rafael Núñez, Cartagena, Colombia

Resumen: Este artículo trata sobre la implementación de algoritmos de aproximación funcional difusa para la identificación de sistemas lineales y no lineales, basados en el método de error de inferencia. La metodología empleada en este trabajo para la obtención del modelo borroso a partir de datos de entrada y salida es presentada en tres fases: en la primera, se emplea el error de inferencia para detectar clases o agrupamientos posibles en los datos; en la segunda, se generan las reglas; en la tercera, se generan la forma, número y distribución de los conjuntos difusos. Se propone también el empleo de un método de inferencia ideal y se presentan los resultados obtenidos en sistemas uniimensionales y multivariables.

Palabras claves: aproximación funcional, error de inferencia, principio de extensión, función biyectiva

1. Introducción

La modelación de sistemas no lineales puede ser abordada mediante la descomposición del sistema global en submodelos o modelos locales, los cuales pueden ser lineales en un rango de validez determinado. Sin embargo, la aplicación de esta técnica puede presentar el problema al momento de seleccionar el número e identificación de los modelos locales (Díez et al, 2004).

La lógica borrosa ha sido una valiosa alternativa para la modelación o identificación de sistemas mediante modelos borrosos, en especial de aquellos que revisten cierta complejidad bien sea por un alto número de variables involucradas o por un comportamiento no lineal. Sin embargo, la construcción de

[*] epcontrerasj@ieee.org, rmisa@electrica.cujae.edu.cu, lisbethu@curn.edu.co

un modelo borroso requiere de la selección de un gran número de parámetros: número, forma y distribución de las funciones de pertenencia, construcción de la base de reglas, selección de operadores lógicos, selección del método de inferencia, entre otros, lo cual requiere de criterios sistemáticos para una acertada selección (Espinosa y Vandewalle, 2000).

Existen dos formas básicas para construir un modelo borroso: la primera de ellas se basa en implementar la experiencia de un experto mediante reglas borrosas; la segunda, consiste en extraer las reglas a partir de datos de entrada y salida. De ésta última se destacan los métodos basados en redes neuronales, algoritmos genéticos y las técnicas de agrupamiento o clasificación. Los algoritmos de agrupamiento borroso representan la técnica más adecuada para la obtención de modelos borrosos, siendo los métodos de Fuzzy C-Means (Bezdek, 1987) y de Gustafson-Kessel (Gustafson y Kessel, 1979) los más empleados. Se han realizado diversas variaciones a estos algoritmos de agrupamiento. Nauck y Kruse (1995, 1999) proponen técnicas de agrupamiento neuro-difusas; Espinosa y Vandewalle (2000) presentan una metodología para extraer reglas a partir de los datos en un marco de integridad lingüística incluyendo algoritmos de fusión para agrupar conjuntos cuyos valores modales estén a una distancia muy cercana. Sala (1998, 2001) introduce una novedosa técnica basada en el error de inferencia para aproximar funciones empleando partición suma 1 con conjuntos triangulares; Diez et al (2004) proponen variaciones a los algoritmos de agrupamiento para mejorar la interpretabilidad y descubrir estructuras afines locales en los modelos borrosos obtenidos. Paiva y Dourado (2004) presentan un modelo generado por medio del entrenamiento de una red neuro-difusa implementado en dos fases: en la primera fase, se obtiene la estructura del modelo empleando un algoritmo clustering substractivo, lo cual permite extraer las reglas a partir de datos de entrada y salida; en la segunda fase, se realiza la sintonización de los parámetros del modelo mediante una red neuronal que emplea backpropagation. Guillaume y Charnomordic (2004) proponen una estrategia para generar particiones difusas interpretables a partir de los datos en la cual incorporan un algoritmo, que denominan partición jerárquica difusa (HFP), que en vez de incorporar datos en cada iteración agrega conjuntos difusos. También presentan un algoritmo de fusión de los conjuntos difusos basado en métricas adecuadas que garanticen la interpretabilidad semántica.

La metodología empleada en este trabajo para la obtención del modelo borroso a partir de datos de entrada y salida está basada en el método del error de inferencia de Sala (98) y es presentada en tres fases: en la primera, se emplea el error de inferencia para detectar clases o agrupamientos posibles en los datos; en la segunda, se generan las reglas; en la tercera, se generan la forma, número y distribución de los conjuntos difusos. Se propone también el empleo de un método de inferencia ideal.

2. Error de inferencia

En lógica borrosa es común tratar con reglas del tipo “Si *u* es A, entonces *y* es B”, donde *u* y *y* representan dos variables numéricas, y $A \subset U$ y $B \subset Y$, son dos conjuntos borrosos de entrada y salida respectivamente, definidos en los universos U y Y. La regla mencionada es equivalente a la inecuación

$$\mu_A(u) \leq \mu_B(y) \tag{1}$$

El error de inferencia ε, concebido como la distancia conceptual al conjunto conclusión de la regla, está dado por

$$\in \approx \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots u_A(u) \leq u_A(y) \\ u_A(u) - u_B(y) \ldots\ldots\ldots u_A(u) > u_B(y) \end{cases} \tag{2}$$

El conjunto conclusión de la regla está definido por

$$C(u) = \{y^* \in Y \mid u_B(y^*) \geq u_A(u)\} \tag{3}$$

y la distancia conceptual de un $y \in Y$ al conjunto conclusión está dada por

$$d_{uB}(y, C(u)) = \min_{y^* \in C(u)} \left| u_B(y) - u_B(y^*) \right| \tag{4}$$

Para una base de reglas N, todas con antecedentes conjuntos borrosos definidos en un universo U y consecuentes conjuntos borrosos definidos en un universo Y, la función de error de inferencia global está dada por

$$\in (u, y) = \left(\sum_{i=1}^{N} \phi_i(u, y) \in_i (u, y)^p \right)^{\frac{1}{p}} \tag{5}$$

Al conjunto $C^*(u) \subset Y$ que minimiza el error de inferencia global (5) se le denomina conjunto conclusión de la inferencia ideal.

El método propuesto para crear un sistema borroso basado en reglas que se aproxime a una función de una entrada una salida, con un error de inferencia nulo, debe cumplir con la condición

$$u_A(x) = u_B(y) \tag{6}$$

en el caso de una regla del tipo "Si *u* es A, entonces *y* es B". Si el sistema tiene *n* entrada se debe representar por una regla del tipo "Si u_1 es A_1 , u_2 es A_2 , …, u_n es A_n, entonces *y* es B", por lo cual el sistema generado deberá cumplir con la condición

$$((u_{A_1}(x_k) \wedge (u_{A2}(x_k) \wedge ... \wedge (u_{An}(x_k)) = u_B(y) \tag{7}$$

donde $\wedge$ representa una t-norma de la lógica borrosa.

3. Detección de las clases

El número de clases o agrupamientos posibles de los datos experimentales de entrada salida determina el número de reglas de sistema borroso. Para obtener el número de clases posibles en los datos se propone un nuevo algoritmo basado en la extensión cilíndrica de los valores modales de los conjuntos del antecedente (o conjunción de los valores modales de los conjuntos de los antecedentes) en el espacio de datos de salida. Los datos resultantes de la proyección estarán todos sobre la recta z=1, lo que facilita la detección de las clases mediante un algoritmo de agrupamiento jerárquico. El algoritmo requiere previamente: un conjunto de datos experimentales de entrada salida $\{u_k, y_k\}$, una partición inicial suma 1 de la(s) variable(s) de entrada mediante *n* conjuntos triangulares normales distribuidos uniformemente sobre el espacio de entrada (definido por el vector de datos experimentales de entrada).

Algoritmo 1. Detección de agrupamientos

Dado una colección de datos experimentales $\{u_k, y_k\}$ $U \times V$; k =1, ..., N_p

(a) Crear una partición triangular suma 1 del antecedente **A** $\subset$ **U** en el rango establecido por los valores mínimo y máximo de los datos de entrada, con un número n de conjuntos borrosos
(b) Determinar las posiciones en el arreglo de datos de entrada para el cual los valores de pertenencia son iguales a 1 (valores modales)

(c) Crear una tabla de datos (y_j, *1*), donde j corresponde a una posición del vector de entrada tal que $u_A(x_j) = 1$

(d) Determinar las clases o grupos existentes en los datos (cluster jerárquico)

(e) Incrementar el número de conjuntos n de la(s) variable(s) de entrada y volver al punto (a)
Seguir incrementando hasta que las clases determinadas no varíen con el incremento de n

El algoritmo entrega el número m de clases que será igual el número de reglas necesarias para aproximar la función

4. Generación de las reglas

El número clases detectadas representan el número de reglas que debe tener el sistema borroso. Se puede presentar casos, para sistemas de más de una entrada, en que el número de clases es inferior al número de posibles combinaciones de los conjuntos de las entradas, lo cual indica que existirán combinaciones que producen el mismo conjunto conclusión. Por lo tanto, las reglas con igual consecuente se deben resumir en una sola regla mediante la disyunción de las conjunciones de sus antecedentes.

Algoritmo 2. Generación de reglas

Dada m clases y el rango de los universos de entrada y salida

(a) Hallar el número n de conjuntos para cada una de las *q* variables de entrada aproximándolo al valor superior más cercano en caso de que el valor resultante sea un fraccionario

$$n = \sqrt[q]{m} \tag{8}$$

(b) Si el número posible de reglas n^q es igual a m entonces conformar las reglas según las clases a las que quede asignada cada combinación de entrada (ir a algoritmo 1). En caso de que el número posible de reglas sea superior a m indicaría que al menos dos reglas tienen la misma conclusión por lo que se deben resumir en una regla cuyo antecedente sea la disyunción de los antecedentes de las reglas en cuestión. Este caso no se presenta en el caso de sistemas de una sola entrada una sola salida

5. Generación de la partición de antecedente

El algoritmo anterior nos permite hallar el número de conjuntos de cada variable de entrada pero no la distribución ni las funciones de pertenencia. El algoritmo propuesto realiza una extensión de los conjuntos de la variable de entrada en el universo de salida, asignando el grado de pertenencia de cada elemento x_k a su respectiva imagen y_k. Las funciones de pertenencia discreta definitivas de los conjuntos difusos de entrada y la distribución en el universo de entrada se generan haciendo una redistribución de los intervalos de evaluación considerados para las funciones discretas triangulares inicialmente supuestas de acuerdo a los intervalos de los datos de salida. Esto, en principio, limita el algoritmo a funciones biyectivas; sin embargo, se puede emplear en caso de funciones no biyectivas dividiendo el problema en modelos locales ( ).

Algoritmo 3. Generación y distribución de funciones de pertenencia

Dado el número n de conjuntos por variables y los rangos de los universos de entrada y salida de los datos experimentales

(a) Crear una partición triangular suma 1 del antecedente **A** $\subset$ **U** en el rango establecido por los valores mínimo y máximo de los datos de entrada, con un número n de conjuntos borrosos (valor n obtenido en el algoritmo 2). Cada conjunto se define mediante un arreglo del tamaño del vector de datos experimentales de entrada $u_{A_i}(x_k)$

(b) Proyectar los conjuntos de entrada en el universo de salida mediante la ecuación (6)

$$u_B(y_k) = u_{A_i}(x_k)$$

creando una tabla de datos (y_k, $u_B(y_k)$)

(c) Determinar el conjunto de singletons de salida de los valores y_k con grado de pertenencia asociado de 1. Estas serán las ubicaciones de los singletons de salida.

$$y_m = \{y_k \mid u_B(y_k) = 1\} \quad (9)$$

(d) Redistribuir los intervalos de evaluación de acuerdo a los intervalos en que están distribuidos los datos de salida, estableciendo previamente los valores mínimo (y_{min}) y máximo (y_{max}) de la variable de salida

$$x_k = \frac{(y_k - y_{min})}{(y_{max} - y_{min})} \quad (10)$$

(e) Asignar a cada valor de entrada x_k el correspondiente grado de pertenencia de y_k; o sea

$$u_{A_i}(x_k) = u_B(y_k)$$

creando una tabla de datos (x_k, $u_{Ai}(y_k)$) para cada uno de los conjuntos de cada variable de entrada

La partición generada para el antecedente es suma 1 aunque muy posiblemente no sea triangular.

6. Método de inferencia

Métodos de inferencia como el del centro de gravedad no son apropiados para aproximación funcional, como se puede apreciar si se trata de aproximar una función tan sencilla como la recta $y = x$. Por lo tanto se tomará la inferencia propuesta por Sala ( )

$$y^*(u) = \frac{\sum_{k=1}^{n} u_k^i(u) y_m(i)}{\sum_{k=1}^{n} u_k^i(u)} \quad (10)$$

donde $u_k^i(u)$ es el grado de pertenencia del dato de entrada en el antecedente *i*(en el caso de sistemas con más de una entrada corresponderá a la conjunción de las entradas), $y_m(i)$ es el valor del singleton correspondiente a la salida *i*. El denominador siempre arroja un valor igual a 1 cuando se trata particiones suma 1, el cual es el caso del método de aproximación propuesto en este artículo.

7. Resultados

7.1 Identificación de funciones unidimensionales

Los datos de entrada y salida de una planta de tratamiento de aguas residuales industriales fueron procesados mediante el método propuesto obteniendo inicialmente la aproximación mostrada en la figura 1. Se obtuvo un modelo borroso basado en 6 reglas arrojando partición suma 1.

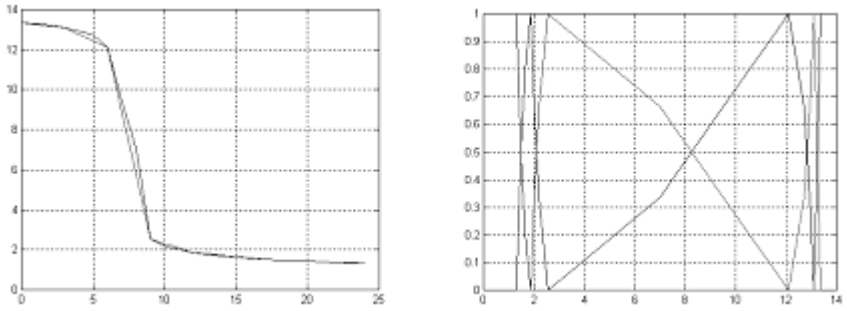


Fig. 1. Aproximación a una curva de pH con un sistema borroso de 6 reglas

7.2 Identificación de funciones multivariables

El sistema no lineal descrito por la ecuación

$$y = (1 + x_1^{-2} + x_2^{-1.5})^2; \ldots 1 \le x_1, x_2 \le 5 \quad (11)$$

ha sido utilizado por autores como Sugeno y Yukasawa (1993), Emami et al (1998) y Díez et al (2004) para verificar el desempeño de técnicas de identificación borrosa. Para iniciar el proceso de identificación se generaron primero los datos de entrada salida a partir de la ecuación (11). La figura 2 muestra a la izquierda a la superficie no lineal a identificar y a la derecha el resultado obtenido del modelo borroso.

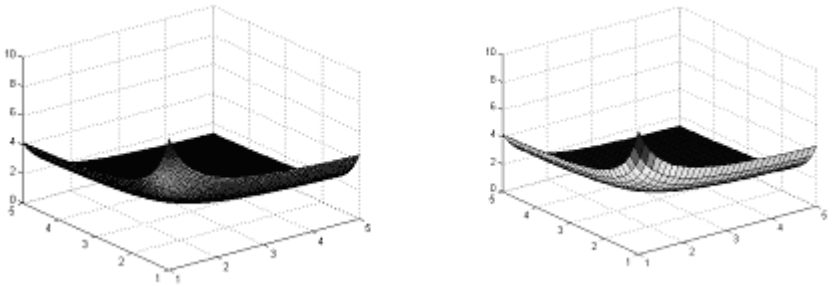


Fig. 2. Aproximación a una superficie no lineal

El algoritmo se inició con 2 conjuntos triangulares suma 1 por cada entrada, distribuidos uniformemente sobre el espacio de entrada, definiendo 3 clases o agrupamientos en los datos. A pesar de que en la iteración siguiente se obtuvieron nuevamente 3 clases se procedió a seguir incrementando el número de conjuntos hasta 5 por cada entrada manteniéndose las mismas 3 clases. Este ejercicio se hizo solo para verificar los resultados del método.

En la figura 3 se muestra a la derecha la ubicación de la extensión de los valores modales, resultados de las conjunciones de las entradas, en el espacio de salida. A la izquierda se realizó la extensión de todas las combinaciones de las entradas en el espacio de salida mostrando también las 3 clases.

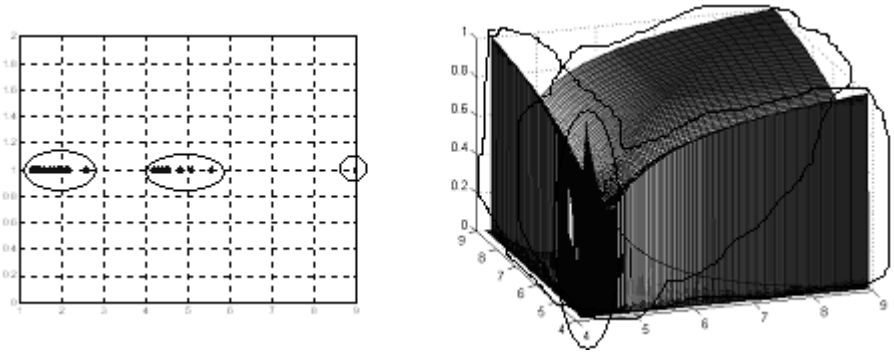


Fig. 3. Clases detectadas en los datos de la superficie no lineal

La figura 4 muestra las funciones de pertenencia generadas y su distribución en el espacio de entrada.

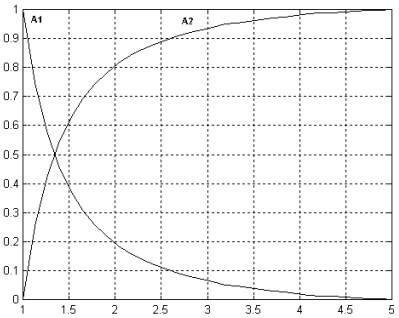


Fig. 4. Partición generada para el antecedente

La ubicación de los singleton de salida quedó establecida en: y_1=9; y_2=4.1; y_3=1.31. La base de reglas quedó de la forma:

$$A_1 \wedge A_1 \rightarrow y_1$$
$$(A_1 \wedge A_2) \vee (A_2 \wedge A_1) \rightarrow y_2$$
$$A_2 \wedge A_2 \rightarrow y_3$$

8. Conclusiones

Se presenta un método basado en la minimización del error de inferencia para la identificación de sistemas mediante modelos borrosos interpretables, generando partición suma 1 de los antecedentes a partir de una partición triangular suma 1 inicialmente supuesta. Se comprobó la utilidad del sistema en la aproximación de funciones no lineales unidimensionales y multidimensionales.

Mediante el método expuesto se pueden determinar las clases o agrupamientos en los datos experimentales, lo cual permite el cálculo del número de reglas borrosas y la generación del número de conjuntos posibles en cada antecedente.

El método ha resultado útil para identificar sistemas o funciones monótonas pero puede ser extendido a otro tipo de funciones haciendo previamente una partición de los datos en subconjuntos que representen funciones monótonas. Lo anterior se debe a que el método exige que las funciones que representan los datos sean biyectivas.

BIBLIOGRAFÍA

1. Bezdek J. C. (1987). Pattern recognition with Fuzzy Objective Function Algortithms. Ed. Plenum Press.
2. Díez J. L., Navarro J. L., Sala A. (2004). Algoritmos de Agrupamiento en la Identificación de Modelos Borrosos. RIAI: Revista Iberoamericana de Automática e Informática Industrial.
3. Emami, M., Turksen, I., Goldenberg, A., “Development of a systematic methodology of fuzzy logic modeling. Transaction of Fuzzy Systems, vol. 6, No. 3, pp.346-360. 1998.
4. Espinosa, J., Vandewalle, J., “Constructing fuzzy models with linguistic integrity form numerical data-afreli algorithm”, IEEE Trans. Fuzzy Systems, vol. 8, pp. 591 – 600, Oct. 2000.
5. Guillaume, S., Carnomordic, B., “Generating an interpretable Family of Fuzzy Partitions Form Data”, IEEE Trans. Fuzzy Systems, vol. 12, No. 3, pp. 324 – 335, Jun. 2004.
6. Guztafson E. E., Kessel W. C. (1979). Fuzzy Clustering with a Fuzzy Covariance Matrix. IEEE CDC, San Diego, California, pp. 503 – 516.
7. Nauck, D., Kruse, R., "Nefclass - a neuro-fuzzy approach for the classification of data", In Proceedings of the Symposium on Applied Computing, 1995.
8. Nauck, D., Kruse, R., “Neuro-fuzzy systems for function approximation”. Fuzzy Sets and System. 101(2), pp. 261-271. Jan. 1999.
9. Paiva, R. P., Dourado, A., “Interpretability and learning in neuro-fuzzy systems”, Fuzzy Sets and System. 147(2004), pp. 17-38. 2004.
10. Sala, A. (1998). Validación y Aproximación Funcional en Sistemas de Control Basados en Lógica Borrosa. Universidad Politécnica de Valencia. Tesis Doctoral.
11. Sala, A., Albertos, P., Inference error minimisation: fuzzy modelling of ambiguous functions. Fuzzy Sets and Systems, 121 (2001) pp. 95 – 111. 2001
12. Sugeno, M., Yasukawa, T., “A fuzzy logic based approach to qualitative modeling”. Transactions on Fuzzy Systems, vol. 1, No. 1, pp. 7-31. 1993

ANEXO 2:

GUIA DE INSTALACION Y DE USUARIO 1.0