Ministerio de Educación Superior
Universidad Central "Marta Abreu" de Las Villas
Facultad de Matemática Física y Computación

TRABAJO DE DIPLOMA

Estimación automática de parámetros de funciones de pertenencia.

Autor

Alain José Varela Martin

Tutores

MSc. Yanet Rodríguez Sarabia

Lic. Rafael Jesús Falcón Martínez

Dra. María Matilde García Lorenzo

2005
"Año de la alternativa bolivariana para las Américas"

Hago constar que este trabajo ha sido realizado en la facultad de Matemática, Física y Computación de la Universidad Central “Marta Abreu“ de las Villas como parte de la culminación de los estudios de licenciatura en Ciencias de la Computación, autorizando a que el mismo sea actualizado por la institución para los fines que estime conveniente, tanto de forma total como parcial y que además no podrá ser presentado en eventos ni publicado sin la previa autorización de la universidad.

_____

Firma del Autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro, y que el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del Tutor

Firma del Jefe del Seminario
de Inteligencia Artificial

Pensamiento

Ciencia y libertad son llaves maestras que han abierto las puertas por donde entran los hombres a torrentes, enamorados del mundo venidero. Diríase que al venir a la tierra tantas coronas de cabezas de reyes, las cogieron los hombres en sus manos y se han ceñido a las sienes sus fragmentos.

José Martí

Dedicatoria

A mi hijo Alain, la parte más tierna de mi disco duro,
la vacuna infalible contra los programas malignos de
la tristeza, mi acceso directo al porvenir.
A la mujer que me trajo a la vida, expresión carnal de
la palabra sacrificio, sostén y brújula de mi existencia.
A mis abuelos y familiares, que me alentaron con cariñosa
persistencia a sembrar y cosechar mis sueños.
A la revolución, que me ha enseñado de qué parte está el
Deber y el valor infinito de la solidaridad humana

Agradecimientos

A mis tutores, la MSc. Yanet Rodríguez, la Dra. María Matilde
Y al Lic. Rafael Falcón por su sabio y valioso
asesoramiento en la realización de este trabajo.
Al Ing. Leonardo del Toro por el apoyo que fraternalmente me ha brindado.
A los compañeros de la empresa Avícola de Ciego de Ávila,
que asumieron con generosa dedicación, la impresión de estas páginas.
Al proyecto social cubano, que me mostró la estrella y me
dio el aliento y la fuerza para alcanzarla.

Resumen

Obtener los términos lingüísticos y la función de pertenencia (FP) correspondiente es una tarea que requiere conocimiento experto en el área de aplicación. Adaptar manualmente una FP a un problema concreto no es fácil, pequeñas modificaciones pueden causar cambios drásticos en la eficiencia del modelo.

Este trabajo tiene como resultado un módulo de preprocesamiento para ser incorporado a la nueva versión de la herramienta computacional, que implementa un modelo híbrido RNA-RBC utilizando conjuntos borrosos para modelar los rasgos lineales. Este módulo brinda facilidades al usuario para la edición manual de FP; permite obtener de forma automática, a partir de ejemplos de entrenamiento, los parámetros de varios tipos de FP; y ajustar las FP iniciales para mejorar el desempeño de la RNA cuando utiliza rasgos borrosos.

Este módulo sería imprescindible para validar el nuevo modelo utilizando archivos de datos internacionales, y de mucha utilidad en la aplicación del mismo a problemas reales. Además las heurísticas propuestas pueden ser incorporadas fácilmente por otros usuarios que requieran modelar sus datos utilizando conjuntos borrosos.

Abstract

To obtain the linguistic terms and their corresponding membership functions (MF) is a task that requires expert knowledge in the application area. Manually adapting a MF to a specific problem is not easy because small modifications can cause drastic changes in the model's performance.

This work provides as outcome a preprocessing module to be incorporated to the new version of the NeuroDeveloper computational tool that implements a hybrid CBR-ANN model using fuzzy logic to model lineal features. This module enables users to manually edit MFs; it allows the retrieval –via automation from training examples-, of the parameters of several types of MFs; and fits the initial MFs to improve the performance of the RNA when it uses fuzzy features.

This module would be essential to validate the new model using well-known international datasets and extremely useful in the model's real applications. In addition, the heuristic proposals can be easily incorporated by other users who might require to model data as fuzzy.

Introducción

En el grupo de Inteligencia Artificial de la facultad de MFC se desarrolló un modelo hibrido [GAR96] que combina las Redes Neuronales Artificiales (RNA) y el Razonamiento Basado en Casos (RBC), el cual ha sido utilizado exitosamente para resolver problemas de diagnóstico [BEL02]. El modelo de RNA implementado requiere en el diseño de su topología asociar una neurona a cada valor posible de un rasgo. El problema surge cuando se utilizan rasgos lineales: continuos o discretos, pues siguiendo esta idea la cantidad de neuronas crecería ilimitadamente afectando el desempeño de la RNA.

Para resolver lo anterior, se trabaja actualmente en la extensión de este modelo utilizando los conjuntos borrosos para representar este tipo de rasgo [GAR00]. Esto conlleva a seleccionar los términos lingüísticos y la FP correspondiente a cada uno de ellos, como parte del procedimiento de seleccionar los valores representativos y asociar a cada uno de estos una neurona. La versión actual de la herramienta que implementa este modelo, y su primera aplicación real [ROD03], deja en manos del usuario la definición de la FP.

Obtener los términos lingüísticos y la FP asociada es una tarea que requiere conocimiento experto en el área de aplicación. Si para esta tarea dependemos de expertos humanos y no están disponibles, entonces las FP no pueden ser definidas; aunque no ocurra lo anterior, el tiempo de desarrollo puede incrementarse; o los sistemas difusos desarrollados pueden no tener un buen desempeño y tal vez en esto influya que las FP definidas no sean apropiadas. Adaptar manualmente una FP a un problema concreto no es fácil, pequeñas modificaciones pueden causar cambios drásticos en la eficiencia del modelo.

Es por ello que se hace necesario incluir en la herramienta referida, un módulo de preprocesamiento para obtener de forma automática, a partir de ejemplos de entrenamiento, los parámetros de las FP para los rasgos que se modelen utilizando conjuntos borrosos. Este módulo sería imprescindible para validar el nuevo modelo utilizando archivos de datos internacionales, y de mucha utilidad en la aplicación del modelo a problemas reales. Además la implementación computacional de los algoritmos seleccionados, en forma de componentes totalmente extensibles, permitiría que estas se utilicen por otros usuarios que requieran modelar sus datos utilizando conjuntos borrosos.

Atendiendo a todo lo anterior, en el presente trabajo fijamos el objetivo general siguiente:

Diseño e implementación de componentes para proponer y ajustar de forma automática funciones de pertenencia.

Y los objetivos específicos que a continuación se relacionan:

1. Facilitar la edición manual de funciones de pertenencia.

2. Proponer e implementar heurísticas para, a partir de ejemplos, obtener de forma automática valores para los parámetros de funciones de pertenencia.
3. Implementar un algoritmo para ajustar los parámetros de una función de pertenencia, para un modelo RNA-Borroso.

Capítulo 1 Las Redes Neuronales y la representación del conocimiento usando conjuntos borrosos

El concepto de *Soft Computing* (SC) comenzó a cristalizarse durante los últimos años. A partir de las dificultades mostradas por la Inteligencia Artificial, basada principalmente en lógica de predicados y técnicas de manipulación de símbolos, para construir máquinas las cuales pudieran ser llamadas inteligentes en el sentido de tener éxito en aplicaciones reales se han desarrollado una colección de métodologías con ese propósito y que hoy se agrupan en el concepto de SC. La esencia de la SC es que ella se acomoda a la imprecisión del mundo real. La guía principal de la SC es explotar la tolerancia a la imprecisión, la incertidumbre y la verdad parcial para alcanzar tratabilidad, robustez y soluciones de bajo costo. Las Redes Neuronales Artificiales (RNA), la teoría de los conjuntos borrosos, y los Algoritmos Genéticos (AG) se consideran las raíces de la SC o inteligencia computacional como también fue denominada a inicio de la década del 90 [SAN00].

1.1. Las Redes Neuronales Artificiales

Las RNA no son más que un modelo artificial y simplificado del cerebro humano, que es el ejemplo más perfecto del que disponemos para un sistema que es capaz de adquirir conocimiento a través de la experiencia. Una RNA es “un nuevo sistema para el tratamiento de la información, cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano: la neurona”.

Debido a su constitución y a sus fundamentos, las RNA presentan un gran número de características semejantes a las del cerebro. Por ejemplo, son capaces de aprender de la experiencia, de generalizar de casos anteriores a nuevos casos, de abstraer características esenciales a partir de entradas que representan información irrelevante, etc. Esto hace que ofrezcan numerosas ventajas y que este tipo de tecnología se esté aplicando en múltiples áreas.

Pasemos ahora a definir los conceptos básicos de los que consta toda RNA, expuestos en [MAT01]:

- Neurona: Cualquier modelo de red neuronal consta de estos dispositivos elementales de proceso. Se pueden encontrar, generalmente, tres tipos:

    a) Aquellas que reciben estímulos externos, relacionadas con el aparato sensorial, que tomarán la información de entrada. Se denominan neuronas sensoras.

    b) Dicha información se trasmite a ciertos elementos internos que se ocupan de su procesamiento. Es en las sinapsis y neuronas correspondientes a este segundo nivel donde se genera cualquier tipo de representación interna de la información. Se les conoce como neuronas de preprocesamiento.

c) Una vez ha finalizado el período de procesamiento, la información llega a las unidades de salida, cuya misión es dar la respuesta del sistema. Son también llamadas unidades externas.

- Peso (*weight*): Las conexiones entre las neuronas son las que determinan el estado de activación de las mismas. Para medir la fortaleza de una conexión o enlace entre dos neuronas i y j se utiliza una medida numérica, que es lo que conocemos por peso (w_{ij}). Así, si el valor de este peso es positivo, se dice que la conexión es excitadora; si es negativo, estamos en presencia de una conexión inhibidora; y la ausencia de enlace se denota haciendo $w_{ij} = 0$.

- Función de entrada (*input function*): Es la forma que tiene la neurona de combinar todos los valores de entrada que le llegan –procedentes del exterior o de otras neuronas- en uno solo, cuantificado por los pesos de dichos enlaces. También se le conoce como entrada neta, y existen múltiples formas de cálculo. La idea siempre es combinar la entrada proveniente de cada neurona con el peso del enlace con ella. Una forma típica es esta:

$$Net_i = \sum_j w_{ij} * x_j$$

- Función de activación (*activation function*): Una neurona biológica puede estar activa (excitada) o inactiva (no excitada); es decir, que tiene un "estado de activación". Las neuronas artificiales también tienen diferentes estados de activación; algunas de ellas solamente dos, al igual que las biológicas, pero otras pueden tomar cualquier valor dentro de un conjunto determinado.

  La función de activación calcula el estado de actividad de una neurona; transformando la entrada global (menos el umbral, θ_i) en un valor (estado) de activación, cuyo rango normalmente va de (0 a 1) o de (–1 a 1). Esto es así, porque una neurona puede estar totalmente inactiva (0 o –1) o activa (1).

- Función de salida (*output function*): El último componente que una neurona necesita es la función de salida. El valor resultante de esta función es la salida de la neurona i (out_i); por ende, la función de salida determina qué valor se transfiere a las neuronas vinculadas. Si la función de activación está por debajo de un umbral determinado, ninguna salida se pasa a la neurona subsiguiente. Normalmente, no cualquier valor es permitido como una entrada para una neurona, por lo tanto, los valores de salida están comprendidos en el rango [0, 1] o [-1, 1]. También pueden ser binarios {0, 1} o {-1, 1}.

Desde el surgimiento de las RNA se han desarrollado diversos modelos orientados a resolver problemas de agrupamiento, clasificación y búsqueda asociativa de información [HIL95] [ZUR92]. En particular nos referiremos a continuación al modelo Activación Interactiva y Competencia (IAC), orientado a la última de las aplicaciones mencionadas [MCC89]

En la topología de este modelo las neuronas se distribuyen en grupos y se establecen enlaces entre las neuronas de grupos diferentes y del mismo grupo. Las conexiones entre las neuronas de un mismo grupo son excitadoras, mientras que las conexiones dentro de los grupos son inhibidoras. La esencia del modelo es que las neuronas de grupos diferentes tratan de excitarse mutuamente de modo que cada unidad trata de incrementar el nivel de activación de sus unidades adyacentes en otros grupos, y dentro de cada grupo se establece una competencia en la cual cada neurona trata de disminuir el nivel de activación de sus compañeras de grupo.

En la etapa de explotación de esta red neuronal las unidades de procesamiento cambian su activación considerando su activación actual, la entrada desde otras neuronas (del mismo grupo y desde otros grupos), y la entrada exterior a la red. La entrada total a una unidad o neurona se calcula por:

$$Net_i = \alpha \sum w_{ij} Output_j + \gamma \sum w_{ik} Output_k + Estr \cdot ExtInput_i$$

donde :

$$Output_j = \begin{cases} a_j & si\ a_j > 0 \\ 0 & \text{en otro caso} \end{cases}$$

para las neuronas j que no pertenecen al grupo de la neurona i

$$Output_k = \begin{cases} a_k & si\ a_k > 0 \\ 0 & \text{en otro caso} \end{cases}$$

para las neuronas k que pertenecen al grupo de la neurona l

w_{ij} : peso del enlace entre la neurona i y la neurona j.

$ExtInput_i$: entrada externa a la neurona i

Una vez que se ha calculado la entrada a la neurona el cambio del valor de la activación se calcula según la expresión:

$$\Delta a_i = \begin{cases} (max - a_i) net_i - decay(a_i - rest) & \text{si } net_i > 0 \\ (a_i - min) net_i - decay(a_i - rest) & \text{en otro caso} \end{cases}$$

La nueva activación de la neurona se calcula por:

$$a_{i+1} = \begin{cases} max & \text{si } a_i + \Delta a_i > max \\ min & \text{si } a_i + \Delta a_i < min \\ a_i + \Delta a_i & \text{en otro caso} \end{cases}$$

En las fórmulas expuestas intervienen varios parámetros que en general son iguales para todas las neuronas de la red y que se explican a continuación.

Max: valor máximo de la activación.

Min: valor mínimo de la activación.

Estr: influencia de las entradas externas a la red.

α: escala de la fortaleza de las entradas excitadoras a las neuronas desde otras neuronas.

γ: escala de la fortaleza de las entradas inhibidoras a las neuronas desde otras neuronas.

Rest: nivel de activación de reposo al cual tienden las activaciones en ausencia de entradas externas.

Decay: fortaleza de la tendencia al nivel de reposo.

Aunque no constituye un parámetro de la red, debe aproximarse la cantidad de iteraciones que se necesitan realizar, lo cual se utilizaría como criterio de parada en caso de que la red no lograra estabilizarse. Ideas de cómo influye la variación de estos parámetros en la respuesta de este modelo de red se presenta en [ROD98].

1.2. Los conjuntos borrosos

La idea de los conjuntos borrosos [ZAD76] viene de la siguiente observación: las clases de objetos en la vida diaria no tienen límites bien definidos. La fuente de imprecisión es la ausencia de criterios definidos rigurosamente de membresía a clases en lugar de la presencia de variables aleatorias, y la noción de conjuntos borrosos es completamente de naturaleza no estadística. Los conjuntos borrosos y los sistemas construidos a partir de ellos, han servido como una de las herramientas para resolver problemas en presencia de vaguedad; por ejemplo, cuando es necesario usar conceptos vagos como Alto, Viejo, etc.

Por ejemplo, el área de la medicina es un excelente exponente donde la incertidumbre juega un papel principal, y por tanto son aplicables los conjuntos borrosos. Características tales como la vaguedad, la incertidumbre lingüística, la imprecisión de las medidas y subjetividad están presentes en cualquier problema de diagnóstico médico. En un paciente el concepto de presión alta difiere, dependiendo de si este es hipotenso

o hipertenso; por otro lado, varios expertos pueden tener diversas opiniones sobre que valores de presión pueden ser considerados altos [KUN98]

Los conjuntos borrosos son creados a partir de la variable lingüística en más detalle. Para modelar una variable utilizando conjuntos borrosos se hace corresponder una variable numerable continua con una variable lingüística distribuida. Una variable lingüística se caracteriza por un quíntuplo (x, T(x), X, G, M) en el cual x es el nombre de la variable, T(x) es el conjunto de términos, o sea, el conjunto de sus valores o términos lingüísticos, X es el universo de discurso, G es la regla sintáctica la cual genera los términos en T(x), y M es una regla semántica la cual asocia a cada valor lingüístico A su significado M(A), donde M(A) denota un conjunto borroso en X.

Formalmente un conjunto borroso A definido sobre un universo X es un par de la forma (x, φA(x)). Donde φA(x) es llamada la función de pertenencia o membresía (FP) para el conjunto A. La FP asigna a cada elemento de X un grado de pertenencia en el intervalo [0,1]. A X se llama universo de discurso y puede ser un espacio discreto o continuo.

Si X es una colección de objetos denotados genéricamente por x, entonces un conjunto borroso A en X se define como un conjunto de pares ordenados:

$A= \{(x, \varphi A(x)) \mid x \in X\}$

Donde φA(x) es llamada FP para el conjunto A. La FP asigna a cada elemento de X un grado de membresía en el intervalo [0,1]. A X se llama universo de discurso y puede ser un espacio discreto o continuo.

Ejemplo 1:

x=Edad

T(x)= {bebe, niño, joven, adulto, viejo, anciano}

X= [0,...,150]

Por ejemplo, Juan es joven, es una aproximación de Juan tiene 20 años. Como la variable numerable base cambia continuamente no hay una manera con sentido de delimitar los límites entre los valores de la variable lingüística. La función de membresía describe la compatibilidad entre una variable lingüística y un valor numérico (como decir que la compatibilidad entre la etiqueta joven y la edad 20 años es 0.7).

Las FP se pueden representar:

- explícitamente.
- Analíticamente.

- Gráficamente.

Representación explicita

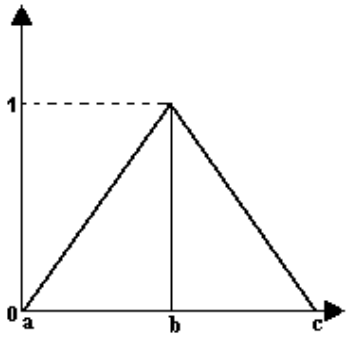
A= {(1, 0), (2,0),..., (5, 0.1),..., (10, 0.5),..., (12, 0.8),..., (16,1),..., (19, 0.9),..., (35, 0.07)}

A= 0/1 + 0/2+...+ 0.1/5 +...+ 0.5/10 +... + 0.8/12 +... + 1/16 +...+ 0.9/19+... + 0.07/35

Representación analítica y grafica

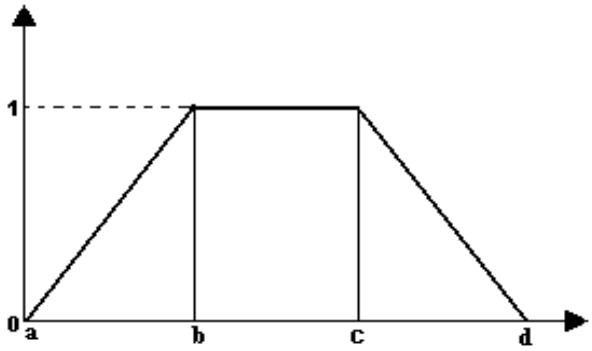
1. **Triangular.**

$$triangle(x;a,b,c)=\begin{cases}0, & x\le a,\\ \dfrac{x-a}{b-a}, & a\le x\le b,\\ \dfrac{c-x}{c-b}, & b\le x\le c,\\ 0, & c\le x\end{cases}$$



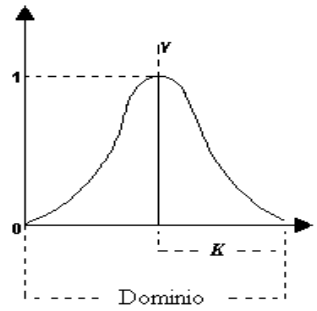
2. Trapezoide.

$$\text{trapezoid}\left(x; a, b, c, d\right) = \begin{cases} 0, & x \le a, \\ \dfrac{x-a}{b-a}, & a \le x \le b, \\ 1, & b \le x \le c \\ \dfrac{d-x}{d-c}, & c \le x \le d, \\ 0, & d \le x \end{cases}$$



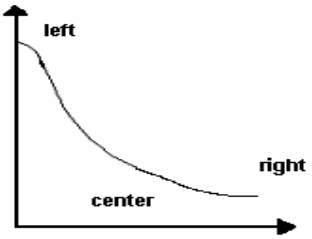
3. Gaussiana.

$$\text{gaussian}\left(x; c, \sigma\right) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2}$$



4. Sigmoidea.

$$sig(x; a, c) = \frac{1}{1+e^{[-a(x-c)]}}$$



La teoría de los conjuntos difusos es cada vez más frecuentemente usada en sistemas expertos y controladores [SET00] [FAB00] [NAU00] debido a su simplicidad y similaridad con el razonamiento humano. Es por ello que pueden ser herramientas muy útiles en la representación de conocimiento pues logran una caracterización matemática bastante acertada de términos del lenguaje natural.

En [PIÑ03] se presentan diversas estrategias que se pueden seguir para construir FP: evaluación subjetiva y construcción a partir de expertos, frecuencias convertidas o probabilidades, medición física y aprendizaje y adaptación. Esta última variante permite construir de forma automática a partir de datos numéricos y simbólicos FP.

En general las técnicas de aprendizaje automatizado se han empleado para eliminar el cuello de botella que constituye la ingeniería del conocimiento en los SBC que requieren conocimiento experto [MIT97]. Obtener los términos y las FP asociadas es una tarea que requiere un conocimiento del experto en el área de aplicación. Queda claro que adaptar manualmente una FP a un problema concreto no es fácil, pequeñas modificaciones pueden posiblemente causar cambios drásticos en la eficiencia.

En [HON96] se propone un método de aprendizaje para derivar automáticamente reglas borrosas y FP deSBe un conjunto dado de casos de entrenamiento facilitando la adquisición de conocimiento. Detallaremos loss primeros pasos del algoritmo que tienen como objetivo obtener las FP de los rasgos. Para ello primeramente se agrupan los datos que aparecen el rasgo que se está modelando en los ejemplos que se disponen. El método de agrupamiento empleado considera la pertenencia a una clase o no (*crisp set*), comparando la similaridad entre los datos adyacentes con un parámetro de entrada alfa (*α*). Si la similaridad entre estos valores excede el valor de *α*, entonces estos se ubican en clases diferentes, sino se ubican en la misma clase; es decir, este valor determina el umbral de pertenencia a una misma clase (para mayor valor de *α* se tiene un menor número de grupos). A cada clase obtenida se asocia un término lingüístico, por tanto, la pericia para proponer un valor adecuado para este parámetro influirá en la modelación que se está haciendo de la variable lingüística.

Para encontrar el valor de similitud entre datos adyacentes, primeramente se ordena el rasgo y se convierte cada diferencia a un número real entre 0 y 1, donde se tiene en cuenta la desviación estándar (σ_s) y otro parámetro de control (*C*). La heurística que se propone para determinar los tres parámetros que caracterizan una FP de tipo triangular es la siguiente:

$$b_j = \frac{y_i * S_i + y_{i+1} * \frac{S_i + S_{i+1}}{2} + ... + y_{k-1} * \frac{S_{k-2} + S_{k-1}}{2} + y_k * S_{k-1}}{S_i + \frac{S_i + S_{i+1}}{2} + ... + \frac{S_{k-2} + S_{k-1}}{2} + S_{k-1}} \quad (1)$$

$$\mu_j(y_i) = \mu_j(y_k) = \min\{S_i, S_{i+1}, .., S_{k-1}\} \quad (2)$$

$$S_i = 1 - \frac{diff_i}{C * \sigma_s} \qquad \text{Donde: } \mathrm{diff_i} = \mathrm{y_{i+1}} - \mathrm{y_i} \quad (3)$$

$$a_j = b_j - \frac{b_j - y_i}{1 - u_j(y_i)} \qquad c_j = b_j - \frac{y_i - b_j}{1 - u_j(y_k)} \quad (4)$$

Donde:
j: representa el j-ésimo intervalo [A_j, B_j]
i: representa el primer índice de los datos en [A_j, B_j]

k: representa el ultimo índice de los datos en $[A_j, B_j]$
y_i : valor de *i-esimo* dato en $[A_j, B_j]$
S_i: similaridad entre y_i y y_{i+1}

Nótese que la FP resultante no es necesariamente simétrica. La similitud calculada influirá en la forma de las FP a estimar, en lo cual influye otro parámetro que se debe proponer. Además el conjunto de FP obtenidos para cada rasgo se consideran propuestas iniciales que se reducen, en el resto del algoritmo con la finalidad de obtener las reglas.

Este proceso de agrupamiento es muy primitivo, tal vez fuera más conveniente utilizar más información disponible en el propio conjunto de entrenamiento como por ejemplo la relación del rasgo que se está analizando con el rasgo objetivo que se está considerando en el problema. Este algoritmo propone solamente FP de tipo triangular. Aunque los modelos borrosos son raramente sensibles a la descripción de los conjuntos borrosos, en problemas reales no todos los rasgos son descritos con un mismo tipo de función donde sí es probable que para algún rasgo la función triangular no sea la apropiada para describirlo, esto provoque que los valores de la FP no sean los deseados y esto pueda influir en el desempeño del sistema.

En [PIÑ03] se compara esta variante de obtener FP con un método que proponen sus autores utilizando AG, mostrando las potencialidades de los AG en la obtención de FP tanto en datos numéricos como simbólicos.. Como caso particular se analizó el uso de los Algoritmos Genéticos en la obtención de FP tipo campana Beta y Triángulo. El diseño del algoritmo genético desarrollado posibilita la resolución y el planteamiento del problema de construcción automática de las funciones de pertenencia como un problema multiobjetivo.

Una idea similar se presenta en [BOT01] se explica un método de uso del llamado “algoritmo bacteriano” para extraer las reglas de un sistema borroso. La clase de FP utilizadas se restringe a la trapezoidal, pues es bastante general y ampliamente utilizada. El algoritmo contiene el paso bacteriano de la mutación, permitiendo el cambio de más de una función de membresía a la vez, y el ajuste de parámetros.

En [HIL95] se expone que una de las principales aplicaciones de la lógica borrosa ha sido el diseño de sistemas de control que, a partir de unas entradas, deben generar unas salidas para actuar sobre determinados mecanismos. Un ejemplo podría ser el sistema de control para regular la velocidad de un ventilador en función de la temperatura ambiente. En 1980, se utilizó por primera vez un SB para supervisar el funcionamiento de un horno de cemento.

Los motivos por los que se empieza a utilizar esta tecnología en los controladores se deben sobre todo a su simplicidad, ya que no requieren constructores matemáticos complejos. Además tiene mayor suavidad en el control respecto a los sistemas convencionales; y mayor posibilidad de combinación con tecnologías clásicas ya establecidas y otras más modernas como las RNA.

1.3. Los sistemas Neuro-borrosos

Entre la lógica borrosa y las RNA puede establecerse una relación bidireccional, ya que es posible, utilizar redes para optimizar ciertos parámetros de los sistemas borrosos, pero también se puede aplicar la lógica borrosa para modelar un nuevo tipo de neurona especializada en el procesamiento de información de este tipo [ZUR95] [DIA00] [HIL95].

Utilizando los llamados Conjuntos Borrosos el humano trata la información inexacta, mientras que las RNA son modelos de la arquitectura física y funcionamiento del cerebro de los humanos [CHI96]. En particular, estos han sido campos donde ha crecido el interés de los investigadores de varias áreas. Aunque las inspiraciones fundamentales de estos dos campos son bastante diferentes, hay muchos puntos en común que demuestran sus similitudes [CHI96].

Los sistemas que utilizan la teoría de los conjuntos borrosos, llamados sistemas difusos o borrosos (SB), y los que se desarrollan utilizando las RNA se pueden utilizar como estimadores numéricos y son sistemas dinámicos. Tienen en común la habilidad para mejorar la inteligencia de los sistemas que trabajan con incertidumbre, imprecisión, y en ambientes ruidosos. Además ambos han mostrado su habilidad de modelación en procesos complejos no lineales a un grado arbitrario de precisión.

Las diferencias radican en que los SB son estimadores numéricos estructurados. Ellos parten de vistas bastante formalizadas de la estructura de categorías encontradas en el mundo real, y entonces articular reglas borrosas para producir comportamientos complejos no lineales. Por otro lado las RNA son sistemas dinámicos entrenables, los cuales aprenden, tienen tolerancia al ruido, y habilidades generales de crecer fuera de su estructura conexionista y representación distribuida de los datos. En particular las RNA tienen un gran número de elementos de procesamiento altamente conectados, los cuales demuestran la habilidad para aprender y generalizar a partir de ejemplos de entrenamiento, y así simples elementos de procesamiento también colectivamente producen comportamientos no lineales complejos.

En [GAR03] se analizan varias tendencias de esta unión. Una tendencia es aplicar esta teoría solo a las componentes de un modelo de RNA, por ejemplo la posibilidad de incluir incertidumbre en la topología de las neuronas, básicamente, en su función de activación [KOS88]. Esta fuzzyficación puede ser parcial o total. En la fuzzyficación parcial se intenta preservar la topología original de la red; modificando, solamente, las capas de

entrada o las de entrada y de salida. Esta forma de fuzzyficación hace un acercamiento de las redes neuronales a los sistemas de expertos pues logra que los rasgos de entrada que sean fuzzyficados tomen la forma de los conceptos que manipulan los expertos. Otra tendencia que se considera una variante parcial sería solamente aplicar esto a las neuronas asociadas a los rasgos de entrada (sensoras) y las que representen la salida. Una neurona sensora [PAL92] sería sustituida por una subred borrosa que implementaría la variable lexicológica que modela al rasgo borroso, la cual tendría la estructura de una red simple con una neurona de entrada y n de salida (una por cada conjunto borroso).

Otra interpretación de esta fusión se presenta en [NAU97]. La idea general es combinar RNA y SB con el objetivo de optimizar un sistema de inferencia borroso.

En un primer enfoque, la RNA y el SB trabajan independiente uno del otro. La combinación radica en la determinación de ciertos parámetros de un SB por una RNA, o por un algoritmo de aprendizaje de esta. Este proceso puede llevarse a cabo durante el uso del sistema borroso (*online*) o fuera de esto (*offline)*. A este tipo de combinación se le llama sistema neuronal borroso cooperativo (*Cooperative Neuro-Fuzzy systems*). Se emplea este término porque la RNA coopera con el SB en el sentido de que ayuda a encontrar parámetros adecuados para este.

A su vez se distinguen cuatro formas de obtener un sistema neuronal borroso cooperativo (ver figura 1).

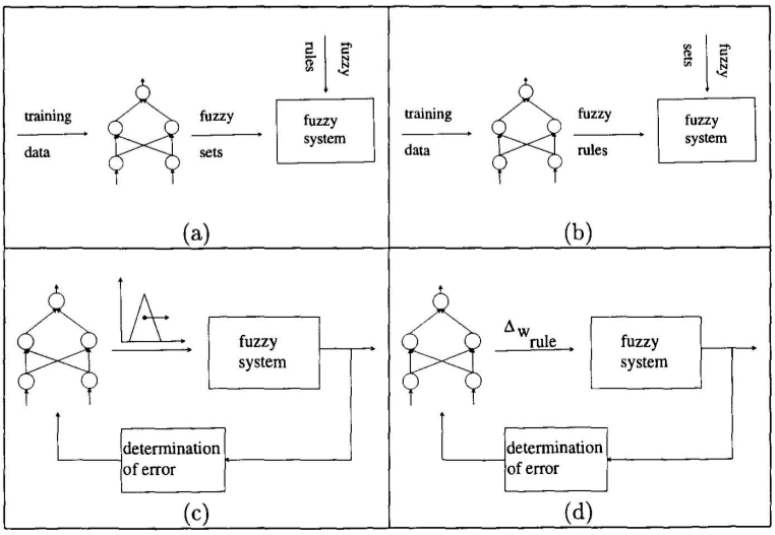


Figura 1. tipos de sistemas neuronales borrosos cooperativos

a) Una RNA determina las FP a partir de datos de entrenamiento. Esto se puede hacer sobre la base de parámetros adecuados, o aproximando las FP con la RNA. Los conjuntos borrosos que se especifican (*offline*) por esta vía se usan conjuntamente con reglas borrosas dadas por separado para implementar un SB. Los datos de entrenamiento deben especificar grados de pertenencia para los valores de los datos de entrada.

b) Una RNA determina reglas borrosas a partir de datos de entrenamiento. Se hace este proceso a través del agrupamiento implementada usualmente por arquitecturas auto-organizativas (*Self Organizing Maps*) o similares. La RNA se usa antes de la implementación del Separa aprender las reglas. Tienen que especificarse por separado las FP correspondientes. Otra alternativa es usar algoritmos de agrupamiento borroso en vez de una RNA.

c) El sistema aprende parámetros durante la explotación del SB. Por ejemplo, durante el uso del SB adaptar las FP. Para este modelo, se deben conocer tanto las reglas borrosas como las FP iniciales. Este proceso debe ser regido por una medida de error que guíe el aprendizaje de la RNA.

d) Una RNA determina los pesos de las reglas borrosas, antes o durante la explotación del SB. Tales factores pueden ser interpretados como la "influencia" de una regla, y son multiplicados por la salida de las reglas.

El segundo tipo de combinación define una arquitectura homogénea, usualmente similar a la estructura de una RNA. Puede hacerse interpretando un SB como un tipo especial de RNA, o implementando un SB mediante el uso de una RNA. A esta combinación se le denomina sistema híbrido neuronal borroso (*Hybrid Neuro-Fuzzy Systems)*, ya que el sistema resultante puede verse como una RNA o como un SB, pero es un único sistema que no es divisible en dos (ver figura 2). A su vez, la idea de un enfoque híbrido es interpretar la base de reglas de un FS en términos de una RNA. Los FS pueden verse como pesos y las variables de entrada y salida, así como las reglas, pueden interpretarse como neuronas. De esta forma, un FS puede modelarse como un caso especial de una RNA o puede ser emulado por una red dirigida hacia delante. El algoritmo de aprendizaje modifica la estructura y/o los parámetros, incluyendo o eliminando neuronas y adaptando los pesos.

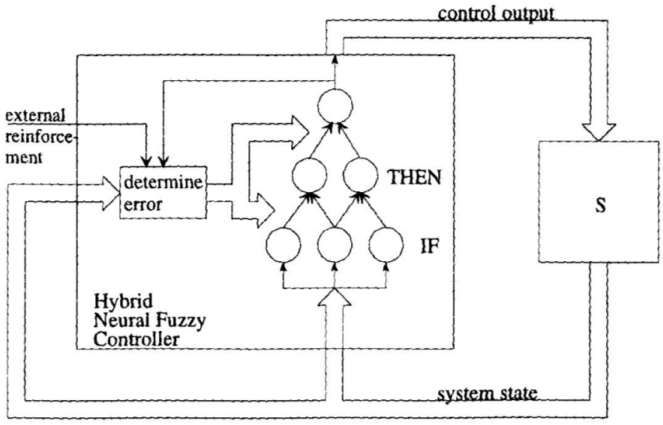


Figura 2. Sistemas híbridos neuronales borrosos

El número y variedad de aplicaciones en estos campos de la Inteligencia Artificial se ha venido incrementando. En [NAU00] se describen varios sistemas que extraen reglas desde conjuntos de datos, y como parte de este procedimiento obtienen las FP asociadas a los términos lingüísticos que consideran en los antecedentes y/o consecuentes de éstas. A continuación se describirán brevemente, focalizándonos en los procedimientos empleados para proponer las FP.

El sistema NEFCLASS es un algoritmo que detecta, primeramente, todos los antecedentes de la regla que cubre algunos datos de entrenamiento y crea una lista de antecedentes. Al principio esta lista está vacía o contiene antecedentes de una regla dada como conocimiento previo. Cada vez que un patrón de entrenamiento es entrado y no es cubierto por un antecedente de la lista, un nuevo antecedente es creado y almacenado. Después, el algoritmo selecciona un consecuente apropiado para cada antecedente y crea una lista base de reglas candidatas. Para cada regla es calculada una medida de su funcionamiento, indicando su precisión (no ambigüedad).

El sistema NEFPROX actúa similar al algoritmo anterior, pero para determinar los conjuntos borrosos y consecuentes, éste calcula un promedio pesado de la salida de los valores objetivos para todos los patrones que la función de membresía no es cero con un antecedente descubierto en los datos. El algoritmo de aprendizaje del NEXPROX necesita recorrer dos veces el conjunto de entrenamiento, una para obtener los antecedentes de las reglas y otra para calcular la media y varianza para todos los antecedentes, requiere después, seleccionar los consecuentes para todos los antecedentes y completar los cálculos estadísticos.

En [JYH98] se presenta el ANFIS, que es un sistema híbrido neuronal borroso que implementa un SB tipo Sugeno como una arquitectura de RNA hacia delante de cinco capas. La base de reglas debe conocerse previamente, ANFIS solo determina las FP de los antecedentes y parámetros lineales de los consecuentes de la regla, aplicando procedimientos estándares de aprendizaje de las RNA como el algoritmo de propagación de los errores hacia atrás (*Backpropagation*). El aprendizaje tiene dos fases, en la primera se estiman los parámetros óptimos del consecuente, asumiendo fijos los parámetros de los antecedentes; mientras que en la segunda fase los patrones se propagan nuevamente con el *Backpropagation* se modifican los parámetros de las FP, asumiendo fijos los consecuentes.

En [HAT00] la generación de neuroreglas es un buen ejemplo de la integración entre RNA y la lógica difusa. Una neuroregla se considera como una unidad de la red adaline, donde los pesos representan factores de significación. Cada factor representa cuán significativa es la condición asociada en el dibujo de la conclusión. Se dispara una regla cuando la salida correspondiente del adaline llega a activarse. Modelar reglas simbólicas borrosas tradicionales como neuroreglas reduce el tamaño de la base de reglas y aumenta la eficacia de las inferencias.

Castellano y Fanelli [CAS00] propusieron una red neuronal para construir y optimizar modelos borrosos. La red se puede mirar como un sistema borroso adaptativo con la capacidad de aprender reglas borrosas a partir de los datos, y a la vez como una arquitectura conexionista provista de significado lingüístico. Las reglas borrosas son extraídas de ejemplos del entrenamiento por un esquema de aprendizaje híbrido compuesto de dos fases: la fase de la generación de datos usando un aprendizaje competitivo modificado, y la fase que ajusta los parámetros usando aprendizaje del gradiente descendente. Esto permite la definición simultánea de la estructura y los parámetros de la base de reglas borrosas. Después de aprender, la red codifica en su topología los parámetros de diseño esenciales de un sistema de inferencia borroso.

Rudy Setiono demuestra con orgullo una aplicación que genera reglas exactas de clasificación para el diagnóstico del cáncer de pecho. La idea principal usada en [SET00] es extraer reglas de clasificación usando el algoritmo NeuroRule a partir de redes neuronales entrenadas, y hacer dichas reglas tan exactas como sea posible, agregando una etapa simple de preprocesamiento a los datos. El preprocesamiento de los datos implica seleccionar los atributos de entrada más relevantes y quitar aquellas muestras con valores desconocidos.

En [BOD01] trata el problema de aprendizaje adaptativo en línea en una red neuroborrosa basado en el modelo de inferencia tipo Sugeno. Se propone un nuevo algoritmo que ajusta tanto los antecedentes como los consecuentes de las reglas borrosas. El algoritmo se deriva del bien conocido procedimiento de Marquardt y utiliza la aproximación de la matriz hessiana. Una característica del algoritmo propuesto es que no requiere operaciones matriciales que consuman gran cantidad de tiempo.

1.4. Un modelo de RNA asociativo utilizando conjuntos borrosos

En [GAR96] se presenta un modelo que combina las RNA y el RBC, en el cual la RNA resuelve el problema (ante la descripción de un patrón P valoriza los rasgos definidos como objetivos); y la componente basada en casos justifica la solución en el contexto de los k casos más similares al problema P, utilizando una función de similitud que toma la información requerida de los pesos de la RNA. De esta forma se aprovecha las ventajas que las RNA tienen en cuanto a la capacidad de aprendizaje y se elimina su limitación de no permitir explicar la solución encontrada.

Los modelos de RNA implementados son del tipo asociativo: el modelo IAC, que explicamos anteriormente, y un modelo similar a este que referenciaremos como SIAC. La diferencia de este modelo con respecto al anterior radica fundamentalmente en la etapa de explotación, pues este no establece competencia entre las neuronas, sino que propagan las entradas y se analiza la activación resultante en las neuronas correspondientes a los rasgos objetivos.

Sin embargo, en la mayoría de las ocasiones no es necesario considerar todos los valores que aparecen en la base para el rasgo, basta con tomar valores que puedan ser representativos de todo un grupo de valores próximos a él. Para los rasgos con esta característica se definen valores representativos, que son a los que se les coloca una neurona en el grupo. Por ejemplo, para el rasgo "Estatura" se pudieran seleccionar tres valores: 110, 155 y 170, como representantes de los intervalos de edades que se consideran para este ejemplo. Es decir, el grupo que corresponde a este rasgo en la RNA tendría entonces tres neuronas asociadas a los valores representativos: bajo, medio, alto; pero con la limitación de que cada uno de estos representa un valor de este rasgo en forma booleana (lo representa o no) y no se considera en cuanto lo aproxima.

Una extensión de este modelo para representar los rasgos lineales (continuos o discretos) utilizando conjuntos borrosos, se presenta en [GAR00]. En este nuevo modelo los valores representativos se corresponden con los términos lingüísticos definidos cuando el rasgo lineal se modela como variable lingüística. Si para un rasgo de este tipo se definen intervalos o el rasgo es simbólico se procede de igual forma que en el modelo original para conformar la topología de la RNA. Es por ello que en este nuevo modelo se requiere añadir a la RNA una capa de preprocesamiento, donde cada neurona tiene definida una función f, donde si v es el valor dado a un atributo x que tiene asociado un valor representativo αi, este activará la neurona asociada con αi en la RNA en la medida del resultado de la función $f_{\alpha i}(v)$ que se define a continuación.

$$f_{\alpha_i}(v) = 1 \qquad si \;\; \alpha_i \subseteq v \,, \;\; \forall \alpha_i \in T_x \tag{5}$$

$$f_{\alpha_i}(v) = \begin{cases} 1 & si \quad Inf_{\alpha_i} \le v \lhd Sup_{\alpha_i} \,, \forall \alpha_i \in T_x \\ 0 & en \quad otro \quad caso \end{cases} \tag{6}$$

$$f_{\alpha_i}(v) = \begin{cases} \mu_{\alpha_i}(v) & si \quad \mu_{\alpha_i}(v) = \max \mu_{\alpha_i}(v), \forall \alpha_i \in T_x \\ 0 & en \quad otro \quad caso \end{cases} \tag{7}$$

$$f_{\alpha_i}(v) = \begin{cases} \mu_{\alpha_i}(v) & si \quad \mu_{\alpha_i}(v) \ge \alpha_0, \forall \alpha_i \in T_x \quad y \quad \alpha_0 \in [0,1] \\ 0 & en \quad otro \quad caso \end{cases} \tag{8}$$

$$f_{\alpha_i}(v) = \mu_{\alpha_i}(v) \tag{9}$$

donde:
T_x : conjunto de valores representativos del atributo c
$\mu_{\alpha_i}(v)$: pertenencia de v al valor representativo α_i (término lingüístico) según la función μ

La primera de estas expresiones se aplica cuando el atributo c es nominal (o simbólico), y la segunda para atributos lineales discretizados (Inf y Sup se refieren a los extremos del intervalo αi). Cuando se usan conjuntos difusos para modelar el atributo c, cualquiera de las tres últimas expresiones se pueden utilizar.

Por ejemplo, retomando el ejemplo de la "estatura de una persona", supongamos que un problema a resolver presenta para este rasgo el valor 154. Si la modelación borrosa de este rasgo da como resultado el conjunto de términos lingüísticos {bajo, medio, alto} con las correspondientes FP tal que: $\mu_{bajo}(154)=0.1$, $\mu_{medio}(154)=0.9$ y $\mu_{alto}(154)=0.05$; entonces las neuronas de este grupo serán activadas con 0.1, 0.9 y 0.05 respectivamente si consideramos todos los términos lingüísticos (expresión 9).

Esta combinación de RNA y conjuntos borrosos se clasifica en [BEL] como "fuzzificación parcial", y lo denominamos modelo RNA-Borroso. Ver figura 3, que ilustra la adaptación realizada a la topología del modelo original.

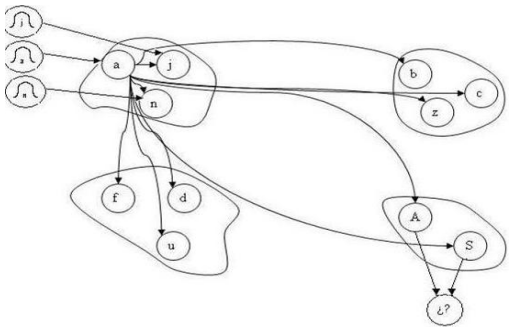


Figura 3. Topología del modelo RNA asociativo usando conjuntos borrosos en las neuronas sensoras.

Capitulo 2 Estimación de parámetros de Funciones de Pertenencia

En el desarrollo de un sistema inteligente que emplee conjuntos borrosos en la representación de sus datos, los expertos humanos juegan un rol importante. Los ingenieros del conocimiento pudieran establecer un diálogo con los expertos humanos con el objetivo de modelar el conocimiento experto, y en particular, definir las FP de pertenencia para cada rasgo del problema. Si dependemos de expertos humanos y no están disponibles entonces las FP pueden no ser cuidadosamente definidas; aunque no ocurra lo anterior, el tiempo de desarrollo puede incrementarse, o los sistemas difusos desarrollados pueden no tener un buen desempeño. A continuación se propone un procedimiento para proponer de forma automática, a partir de ejemplos de entrenamiento, FP apropiadas y entendibles por el usuario.

2.1. Heurísticas para proponer los parámetros de las FP a partir de ejemplos

El proceso por el cual se determina si los elementos de un conjunto X (Universo de discurso de la variable lingüística) pertenecen o no a un intervalo (*crisp set*) se puede definir mediante una función característica o discriminante [HON88]. La misma se puede extender no solamente para indicar lo anterior, sino cuantificar en que medida utilizando el grado de pertenencia (*fuzzy set*).

El procedimiento para obtener los conjuntos borrosos asociados a una variable consta de dos etapas. En la primera etapa se definen los términos lingüísticos, particionando el universo de la variable lingüística (*X*) en grupos. Luego para cada uno de estos términos se estiman los parámetros del tipo de FP seleccionado. Los conjuntos borrosos asociados a un atributo resultante de este procedimiento deben ser fáciles de entender por los expertos del dominio de aplicación.

2.1.1. Selección de términos lingüísticos

El universo de discurso (discreto o continuo) de una variable lingüística *X* asociada a un atributo lineal esta formada por los valores que aparecen para ese atributo en el conjunto de entrenamiento. Este se particiona en grupos, asociando un termino lingüístico a cada uno de los grupos obtenidos. El *j*-ésimo grupo (*Gj*) se representa por [*Aj*, *Bj*], donde *Aj* se corresponde con el menor valor del intervalo asociado al grupo *Gj* y *Bj* con el mayor.

Esto se logra aplicando métodos de discretización o agrupamiento, y con esta finalidad se han propuesto varios métodos. Fueron seleccionados como métodos de discretización el Chi2 [LIU97] y el CAIM [KUR04], y el *k-means* [JYH98] para agrupar los datos.

El método Chi^2 tiene como objetivo discretizar atributos numéricos basados en el estadístico x^2, y además permite eliminar los atributos redundantes y chequear inconsistencias. Se inicia considerando cada valor como un intervalo y se realiza un proceso de mezclado hacia arriba (*Button up*) hasta que se cumpla la condición de parada. Una discretización excesiva introduciría muchas inconsistencias, y la permisibilidad de la medida de este valor se introduce por un factor.

El algoritmo CAIM discretiza un atributo en el menor número posible de intervalos y maximiza la interdependencia entre el atributo y el rasgo objetivo (clase). Es el propio algoritmo quien selecciona de manera automática el número de intervalos discretos en los que quedará finalmente particionado el atributo. Este método parte de un conjunto inicial que contiene un único intervalo: todo el dominio del atributo, y calcula los posibles valores a ser añadidos de uno en uno (*top-down*), hasta formar el esquema final de discretización. La adición de un nuevo valor al esquema se efectúa si dicho valor satisface el criterio de alcanzar la mejor interdependencia clase-atributo posible.

El algoritmo *K-means*, también conocido como *C-means clustering*, particiona una colección de N vectores xj, j = 1,…, N. en c grupos o clusters Gi, i = 1,…, c y encuentra un centro de cluster en cada grupo de forma tal que se minimice una cierta función de costo. Dicha función de costo se calcula a partir de una distancia entre dos vectores, para lo cual proponemos dos medidas de distancia: la euclidiana y la de Manhatan [WIL97]. Inicialmente se pide al usuario especificar la cantidad de clusters deseada. Se inicializan aleatoriamente los centros de clusters y se calculan las distancias de todos los vectores a cada centro de cluster, ubicando al vector x_k en el cluster más cercano a él. Posteriormente, se recalculan todos los centros de cluster y se vuelve a repetir los pasos anteriores, hasta que se cumpla una condición de parada.

Los métodos seleccionados consideran la similaridad entre los valores, en contraste con otros métodos que definen intervalos de igual amplitud o igual frecuencia, lo cual propicia definir términos relacionados con los datos que se están procesando. Los primeros se aplican solamente cuando hay solo un rasgo objetivo definido y puede tomar un solo valor en cada caso (problemas de clasificación). Para el resto de las situaciones se emplearía el *k-means*.

Retomando el ejemplo anterior para el atributo “estatura”, la variable lingüística tiene como universo de discurso los números reales entre 0 y 200, y como resultado de aplicar métodos de discretización se obtendrían tres intervalos ((0; 137), (137; 178.5), (178.5; 200)). Luego se obtiene el conjunto T_{estatura} de términos lingüísticos con los tres elementos que definimos en el epígrafe 1.4.

2.1.2. Estimación de los parámetros de las FP

Para estimar los parámetros de una FP μ_j asociada al *j*-ésimo término lingüístico (*T*j), se considera el conjunto *X* definido anteriormente ordenado en orden ascendente. Esta se corresponde con el *j*-ésimo

conjunto borroso a ser obtenido, cuyo soporte [15] será el conjunto de todos los puntos {z} en [A_j, B_j], para los cuales el grado de pertenencia a la misma será mayor que cero ($\mu_j(z)>0$).

A continuación se describen las heurísticas propuestas para estimar, a partir de ejemplos, los parámetros de las FP de tipo Triangular, Trapezoidal, Gaussiana y Sigmoidal [JYH98].

FP de tipo Triangular

Una FP Triangular se especifica por tres parámetros {a, b, c}, donde a<b<c. Para estimar estos parámetros se aplica una generalización de la variante propuesta por Hong y explicada anteriormente en el epígrafe 1.2, asumiendo la forma que él emplea para calcular la similaridad entre dos valores adyacentes a partir de la diferencia entre ellos como caso particular ($\beta_i = S_i$). Si no se considera la similaridad entre datos adyacentes (β_i =1 para todo i), entonces el centro seria la media de los valores del intervalo. La misma idea propuesta por Hong es aplicable al resto de los parámetros. En los extremos del intervalo (A_j y B_j), la FP tiene un valor igual al mínimo de la similaridad entre todos los puntos en [A_j, B_j], y luego se obtienen los parámetros a_j y c_j de la FP por interpolación.

FP de tipo Trapezoidal

La FP Trapezoidal es especificada por cuatro parámetros {a, b, c, d}, donde a<b<c<d. Note que esta función es reducida a la función triangular cuando b es igual a c. Por esta razón, el siguiente procedimiento es aplicado para estimar los parámetros:

a) A partir del intervalo [A_j, B_j] calculamos a_1, b_1, c_1 por la heurística para FP Triangulares
b) De la misma forma con [A_j, b_1] obtenemos a_2, b_2, c_2
c) De la misma forma con [b_1,B_j] obtenemos a_3, b_3, c_3

Finalmente definimos la función trapezoidal con los parámetros (a_2, b_2, b_3, c_3)

FP de tipo Gausiana

La FP Gausiana esta definida por dos parámetros {c, σ}, donde c representa el centro de la FP y σ determina el ancho de la FP ($\sigma>0$). Se aplican las ideas anteriores para la estimación de los parámetros de la FP Triangular. El valor de c_j debe ser considerado como la media de los valores de los puntos en [A_j, B_j], considerando el caso particular de la ecuación 4 donde β_i es igual a uno para todos los puntos y_i en [A_j, B_j]. El parámetro ancho esta relacionado con c_j (ver expresión 5), asumiendo las propiedades de simetría de esta función.

$$\sigma = \frac{Ln(2)}{|c - v|} \qquad (10)$$

Donde
v= A_j si (|Centro-Aj|,|Centro-Bj|) = |Centro-Aj|
v= B_j en otro caso

FP de tipo Sigmoidal

La FP sigmoidal está definida por dos parámetros {c, α}. Dependiendo del signo del parámetro α ($\alpha \neq 0$), esta función abre a la derecha (positivo) o a la izquierda (negativo) y es apropiada para representar conceptos como "muy largo" (término lingüístico Tk, 0<j<k) o "muy negativo" (término lingüístico T0). El parámetro *c* es el mismo que el de la anterior FP, y el valor absoluto del parámetro α se estima por la fórmula 6.

$$a = \ln(0.25) * |v - c| \qquad (11)$$

Donde
v= A_j si (|Centro-Aj|,|Centro-Bj|) = |Centro-Aj|
v= B_j en otro caso

2.2. Algoritmo para modificar los parámetros de las FP en un modelo RNA borroso

Determinar los términos lingüísticos y las FP es una tarea que requiere conocimiento experto del área de aplicación. Adaptar manualmente los parámetros de una FP a un problema no es fácil, pequeñas modificaciones pueden causar cambios considerables en el desempeño del modelo.

Cuando se emplean las FP que se obtienen aplicando las heurísticas anteriores el modelo RNA-Borroso que explicamos en el epígrafe 1.4 muestra resultados satisfactorios. No obstante es deseable tener un algoritmo para modificar los parámetros de las FP a partir de los errores cometidos por el modelo cuando resuelve un problema del conjunto de entrenamiento.

En general el algoritmo propuesto sigue ideas similares a los sistemas NEFCLASS y ANFIS referenciados en el epígrafe 1.3. También tiene semejanzas con los algoritmos de entrenamiento que se emplean en las RNA (se aplica cíclicamente sobre el conjunto de entrenamiento); y en especial con el *Backpropagation,* porque las modificaciones se realizan en la dirección del gradiente del error. Se puede considerar como un algoritmo supervisado y correccional.

2.2.1. Notaciones y Condiciones

Una muestra de *N* ejemplos caracterizados por *M* atributos, se puede representar como una matriz *NxM*, que denominaremos *MA*. El preprocesamiento de ésta (que denominaremos *MA*'), a partir de la especificación de una función *f* para cada atributo siguiendo alguna de las variantes previamente definidas, da como resultado una matriz de dimensión *NxM*' donde:

(12)

$$M' = \sum_{i=1}^{/M/} m_i$$

$$M = C_p \cup C_o$$

$$m_i = |Tx_i|, \forall x_i \in M$$

Donde:

C_p: conjunto de rasgos predictores
C_o: conjunto de rasgos objetivos
$|Tx|$: representa la cantidad de valores representativos que tiene asociado el rasgo x

El valor $w_{\alpha i, \alpha j}$ representa el peso del enlace entre las neuronas *i* y *j* correspondientes a los valores representativos α_i y α_j [1].

Seleccionar un patrón *P*

El método de selección de un patrón puede ser por alguna de las siguientes variantes: Aleatoria, secuencial, o repetir hasta aprender. La segunda variante asume el orden que los patrones tienen en el conjunto de entrenamiento (CE), mientras que la tercera significa que se utiliza el mismo patrón hasta que sea correctamente resuelto. Por supuesto esta variante debe combinarse con una de las dos anteriores, y solamente sería aplicable a problemas de clasificación.

Conjunto de restricciones

El conjunto de restricciones es un elemento muy importante en el algoritmo. Solo si estas se cumplen considerando las modificaciones propuestas a los parámetros de las FP, estas se pueden efectuar. Las restricciones definidas son:

a) Los parámetros de las FP deben ser válidos.
b) El centro de las FP tiene que pertenecer al soporte de la FP correspondiente. No tiene sentido que un valor que inicialmente tenía máximo nivel de membresía quede fuera de soporte de la FP final.
c) La intersección máxima entre dos FP adyacentes es equivalente al 25 % de la suma de sus áreas. Esto garantiza que las FP no se solapen.
d) La intersección entre dos FP tiene que ser mayor que cero, es decir, para todo {x} en *X* existe al menos una FP μ_i tal que $\mu_i(x)>0$. Esto garantiza cubrir todo el universo de discurso de la variable.

Criterio de parada "A"

El algoritmo termina si se cumple alguna de las situaciones siguientes:

[1] Ver [GAR00] para encontrar como calcular esta medida.

a) Se llegó a la iteración (o época) que especificó el usuario *MaxIter*. Una iteración significa recorrer todos los ejemplos del CE.
b) El error medio de la época es menor o igual que el especificado por el usuario. Este se calcula promediando el error cometido en cada ejemplo del CE en esa época, que a su vez se estima por el promedio de los errores cometidos en las neuronas que asociadas a los valores de los rasgos considerados como objetivos.
c) El desempeño es mayor o igual que el deseado. La expresión para calcular esta medida es la que aparece en la ecuación 13.
d) Transcurrió una época completa y no se efectuaron cambios en las FP. Esta condición sería equivalente a la de la estabilidad en los algoritmos de aprendizaje de las RNA.

$$SP(CE) = \frac{\sum_{P \in MA'} \delta\left(P.o, \overline{P.o}\right)}{N} \tag{13}$$

$$\delta\left(P.o, \overline{P.o}\right) = 1 - \frac{D(P.o, \overline{P.o})}{M'} \tag{14}$$

$$D(x, y) = \sum_{i=1}^{M'} |x_i - y_i|$$

Donde:

$P.o, \overline{P.o}$: representan la salida deseada y la obtenida para el Patrón P respectivamente, siendo estos vectores de M' componentes considerando M= Co en la expresión 12.

D(x,y): distancia de Manhattan

Criterio de parada "B"

El algoritmo termina si se cumple alguna de las situaciones siguientes:

a) Se llegó a la iteración (o época) que especificó el usuario *MaxIter.* Una iteración significa considerar todos los rasgos del problema como rasgo objetivo, uno cada vez.
b) El error medio de la época es menor o igual que el especificado por el usuario. Este se calcula a partir de la media de los valores de error retornados por el algoritmo *Modificar_Parametros*.
c) El desempeño es mayor o igual que el deseado. La expresión para calcular esta medida es la que aparece en la ecuación 13

2.2.2. El algoritmo

Este algoritmo requiere una definición inicial de las funciones *f* para todos los rasgos, que significa para los rasgos lineales modelados como variables lingüísticas proponer FP iniciales y definir el criterio a seguir para determinar la pertenencia de un valor a un término lingüístico (expresiones 7, 8, 9). Estas pueden ser sugeridas por el usuario, u obtenidas de forma automática siguiendo las heurísticas explicadas anteriormente.

Este algoritmo da como salida los nuevos parámetros de las FP (μ) iniciales, que mejoran el desempeño de la RNA-Borroso con respecto a las FP iniciales; y por tanto también variarían las funciones *f* correspondientes en la capa de preprocesamiento (Figura 3). A continuación se describe el algoritmo *Modificar_Parametros*.

I) Inicializar el conjunto *Co*

II) Inicializar *N* con la cantidad de ejemplos del CE.

III) Para todos los patrones *P* del CE

1- Seleccionar un patrón *P.* Decrementar *N* en uno

2- Calcular el grado de activación ($Output_{k+i}$) para todas las neuronas n_{k+i}; donde *k* representa el índice de un atributo *x* en *Co*, y n_{k+i} la neurona asociada con el *i-ésimo* valor representativo de este atributo.

3- Para todas las neuronas n_{k+i} calcular el error cometido al resolver el patrón *P*, de la siguiente forma:

$\delta_{p,k+i} = f_{\alpha i.}(P.o_k) - Output_{k+i}$, donde $P.o_k$ es el valor deseado para el k-ésimo rasgo objetivo en el patrón P. Esta es una medida de si la activación de una neurona debió ser mayor o menor que la obtenida.

4- Asignar a E_p la media de los valores de $\delta_{p,k+i}$

5- Para cada neurona n_{k+i} con $\delta_{p,k+i} > 0$

a. Sea *D* el conjunto de todas las neuronas *n* correspondientes al atributo lineal *x* en *Cp*, tal que *x* se halla modelado utilizando conjuntos borrosos. Seleccionar *n'* en *D* tal que:

$$Output_{n'} * w_{n'k+i} = \min \{Output_n * w_{nk+i}\}$$

b. Determinar el error cometido por *n*' cuando se presenta el patrón *P* mediante la siguiente expresión (esto es una medida de como la salida de esta neurona influye en el error cometido por cada neurona n_{k+i})

$$\delta_{pn'} = \left(\sum_k \sum_i \left(\delta_{pk+i} * w_{n'k+i} \right) \right) * F$$

Donde F es la primera derivada de la función que se utiliza como modelo de la neurona en el modelo de RNA que se este utilizando. Si se emplea el modelo SIAC entonces F es igual a uno, mientras que con el modelo IAC se calcula mediante el siguiente la siguiente expresión:

$$F = \begin{cases} \text{Max} - \text{OutPut n}' & \text{si Max} > \text{OutPut n}' \\ \text{OutPut n}' - \text{Min} & \text{si Min} >= \text{OutPut n}' \\ 0 & \text{e.o.c.} \end{cases}$$

Donde Max y Min son parámetros del modelo IAC que se inicializan durante el entrenamiento (epígrafe 1.1).

c. Proponer las modificaciones para los parámetros de la FP (para los parámetros de la FP de tipo Triangular):

$$\Delta b = \sigma * \delta_{pn'} * (c - a) * \text{sgn}(\text{Output}_{n'} - b) \quad (14)$$

$$\Delta a = -\sigma * \delta_{pn'} * (c - a) + \Delta b \quad (15)$$

$$\Delta c = \sigma * \delta_{pn'} * (c - a) + \Delta b \quad (16)$$

Donde:
σ: razón de aprendizaje (σ > 0)
sgn: función signo

d. Aplicar los cambios a los parámetros si estos no violan el Conjunto de Restricciones.

6- Incrementar el valor de E en Ep.

7- Si el Criterio de Parada "A" se cumple, entonces Terminar.

8- Si una época no se ha completado (N > 0), entonces volver al paso 1

9- Calcular Ē = Ep/N (error medio de una época)

IV) Si no se cumple el Criterio de Parada "A", entonces volver al paso II.

En el algoritmo anterior (5-c) las modificaciones son de dos tipos: modificar el centro utilizando la ecuación 8 (tipo I) y modificar el soporte aplicando ecuaciones 9 y 10 (tipo II). Aunque la especificación del algoritmo es solo para FP triangulares, para modificar los parámetros de los restantes tipos de MF que se consideran en el epígrafe 2.1, se requiere solamente redefinir estos dos tipos de modificaciones, considerando nuevas ecuaciones (en lugar de 8, 9, 10) de acuerdo al nuevo tipo de MF.

Por las características de los modelos de RNA utilizados, se pueden presentar dos tipos de situaciones a considerar en el aprendizaje de las FP: el conjunto de rasgos está dividido en predictores y objetivos, o la variedad de problemas que se pueden presentar hace que esto no se haya definido a priori. La primera

variante debe arrojar un aprendizaje con un mejor desempeño aunque menos general. Estas situaciones a los efectos de aplicar el algoritmo anteriormente especificado se generalizan de la siguiente forma:

I. Para cada atributo lineal ordenar las FP según su centro
II. Formar la matriz MA.
III. Obtener *MA',* Hacer *Iter* = 0
IV. Entrenar la RNA. Se obtiene un conjunto de pesos que no varia durante el algoritmo
V. Si *Co* es vacío,

Entonces, para todo *x* en *M* (conjunto de rasgos) hacer:

a) Seleccionar un rasgo x_j (puede ser aleatorio o secuencial)
b) Hacer Co = { x_j } y ejecutar Modificar_Parametros (*MaxIter* = 1*)*
c) Hacer Co = {}

Sino ejecutar *Modificar_Parametros (MaxIter*). Terminar

VI. Incrementar *Iter*
VII. Calcular el error medio de la época
VIII. Calcular el desempeño
IX. Si no se cumple el Criterio de Parada "B" volver al paso V

2.3 Implementación computacional del módulo para procesar rasgos usando conjuntos borrosos

En este epígrafe brindaremos una información detallada, a modo de guía al desarrollador, y con menos detalle un manual de usuario simple de la herramienta que nos permitirá editar, estimar y ajustar los parámetros de FP (ver figura 4). La herramienta fue totalmente programada en el lenguaje de alto nivel Visual Basic.Net, cuenta con una interfaz de usuario cómoda y de fácil maniobrabilidad por parte del usuario y realiza un manejo eficiente de la información contenida en los casos de la base de datos.

Se detallarán a continuación los requerimientos del sistema sobre el cual se monte esta herramienta, y luego veremos los pasos generales para el trabajo con la misma. Finalmente, se hará una explicación exhaustiva de cómo usar los componentes de esta herramienta.

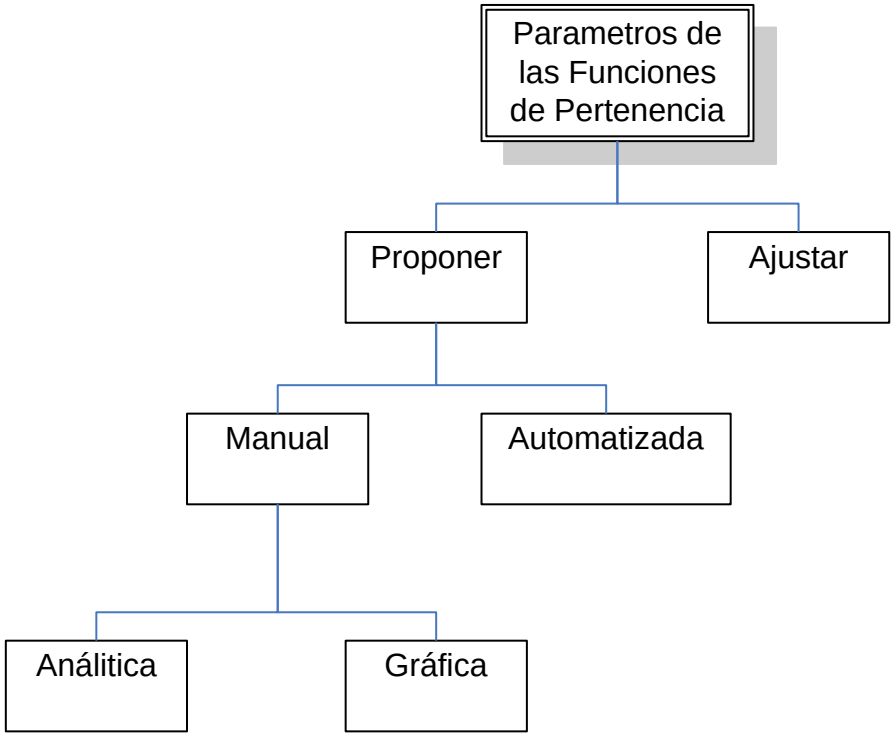


Figura 4. Funcionalidades del módulo de preprocesamiento

2.3.1 ¿Cómo interactuar con el módulo de preprocesamiento?

La aplicación desarrollada debe ser capaz de manejar un gran cúmulo de información, proveniente de la conjunción entre los tipos de rasgos existentes en la base de datos y cada uno de los casos de la misma. No serían extrañas ciertas bases de casos que tuvieran 30, 40 ó 50 rasgos, y como promedio entre 1500 ó 2000 casos. Algunas incluso, pudieran ser mucho más complicadas.

Todo esto requiere un uso de memoria y de velocidad de procesamiento. Como mínimo, la herramienta puede desempeñar sus funciones en un Intel 586 Pentium II con 64 MB de memoria y 300 MHz de CPU. Sin embargo, si desea un rendimiento óptimo por parte de la misma, recomendamos correrla sobre un Intel 586 Celeron Pentium II o superior, con 128 MB de RAM y procesador a 566 MHz o superior. En estas condiciones podemos asegurar que ofrecerá una rápida maniobrabilidad de los datos.

Esta herramienta fue diseñada para ser corrida sobre el Framework 1.1 de .Net por lo que no depende de una plataforma específica.

Conceptos generales para el trabajo con la herramienta

Deberemos definir ahora algunos conceptos importantes que serán imprescindibles para el trabajo con el software. Dejaremos clara la estructura de ciertos tipos de ficheros de entrada y salida que se necesitan y genera.

Fichero de Datos o Base de Casos

Quizás de todos los ficheros con los cuales tiene que lidiar nuestra herramienta, el más importante sea aquel donde se definen los datos con que va a trabajar la herramienta. Este fichero esta compuesto por varias filas que representarían un caso y a su vez, separados por uno o varios espacios los rasgos del caso. Es muy importante que todos los casos tengan la misma cantidad de rasgos.

Figura 5 Estructura interna de la base de casos Iris.data

Fichero de DEFINICIÓN de rasgos (*.DEF)

En el fichero .DEF, cada línea representa la información completa de un rasgo. Se comienza con el nombre del rasgo, seguido de la categoría a la que pertenece, y luego un listado de parámetros, que detallamos seguidamente.

$Valor_1, Valor_2, \ldots, Valor_N$	Si el atributo es de tipo discreto.
$(Inf_1 ; Sup_1), (Inf_2 ; Sup_2), \ldots, (Inf_N ; Sup_N)$ ó :Método (Min ; Max)	Si el atributo es de tipo continuo (intervalo).
$(Termino_1 : Funcion_1 : Params_1), \ldots, (Termino_1 : Funcion_1 : Params_1)$	Si el atributo es de tipo difuso (fuzzy).

En el caso del rasgo borroso, se especifica el nombre del término lingüístico al que se hace referencia, el tipo de función de membresía a la que pertenece, y los parámetros de la misma.

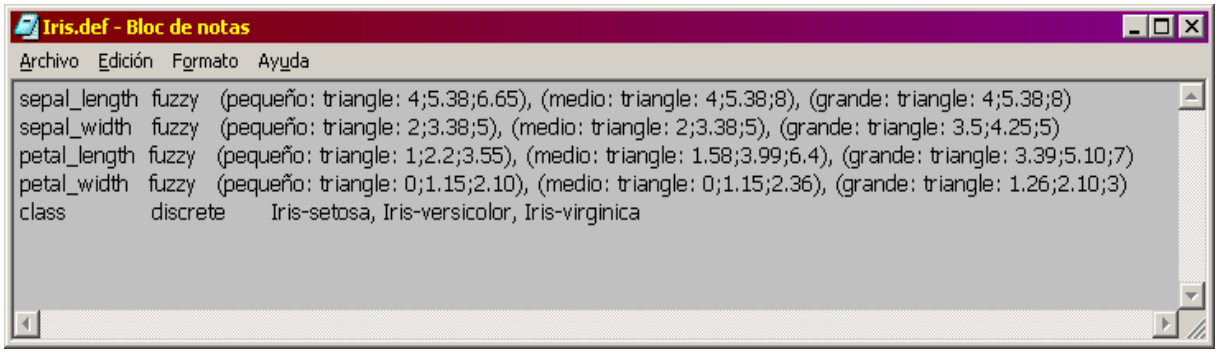


Figura 6 Estructura interna del fichero de definición de rasgos de la base Iris

Fichero de datos preprocesado (*.prep)

En este fichero se almacena la información referente a cada caso, por filas, y por columnas, separados por espacios un numero entre cero y uno; existe una relación uno a uno con respecto a la cantidad de FP que existan en total para los casos.

Funcionamiento de la herramienta

Nuestra herramienta tiene como misión principal garantizar con eficiencia la edición, estimación y ajuste de los parámetros de las FP que el usuario cree, así como la visualización de los resultados de dicha ejecución. Todo esto puede realizarse cómodamente mediante la selección de diversas opciones que aparecen en el menú principal. Procederemos a explicar cada una de ellas:

Menú Base de Casos (Cases Base)

Cases Base| Load CB

Mediante esta opción el usuario le indica a la herramienta cual será la base de casos que usará para todas las posibles operaciones. Es obligatorio seleccionar una.

Cases base| Load definition

Carga y visualiza una definición de rasgos previamente salvada. Es muy importante que esta se corresponda con la base de casos actual.

Cases base| Save definition

Salva la definición de datos actual para un fichero con el nombre de la base de casos con extensión ".def"

Por ejemplo, si la base de casos se llama "iris.data" el fichero generado se llamará "iris.data.def"

Cases base| Save Preprocess CB

Genera a partir de la base de casos que fue cargada previamente y salva el fichero de datos preprocesados con el nombre de la base de casos con extensión ".prep"

La generación de este fichero depende del modo seleccionado en el menú Evaluate que puede ser uno de los siguientes:

- Standard
- Max
- Max1
- Epsilon

Menú Variables (Variables)

Variabes| Change Variable Type

Esta opción es para seleccionar cuales variables van a ser tratadas como Simbólicas (No marcadas) o Lineales (Marcadas). La herramienta hace un reconocimiento de rasgos cuando carga la base de casos.

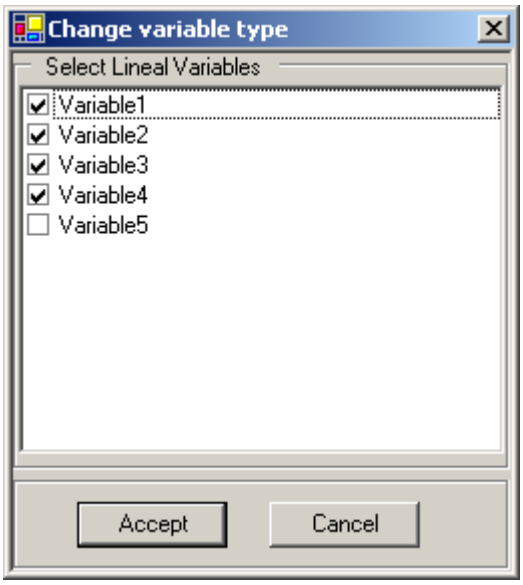


Figura 7 Ventana de selección del tipo de rasgo

Variables| Auto| Current Variable

Con esta opción se construye proponen de modo automático los parámetros para las FP del rasgo seleccionado. Es necesario seleccionar una de las variantes de construcción.

1. Todas las funciones de pertenencia del tipo Triangular
2. Todas las funciones de pertenencia del tipo Triangular con membresía uno en los extremos
3. Funciones de tipo Triangular con membresía uno en los extremos, Trapezoidales en el centro.
4. Todas las Funciones Gaussianas
5. Funciones de tipo Sigmoidales en los extremos con Gausianas en el centro

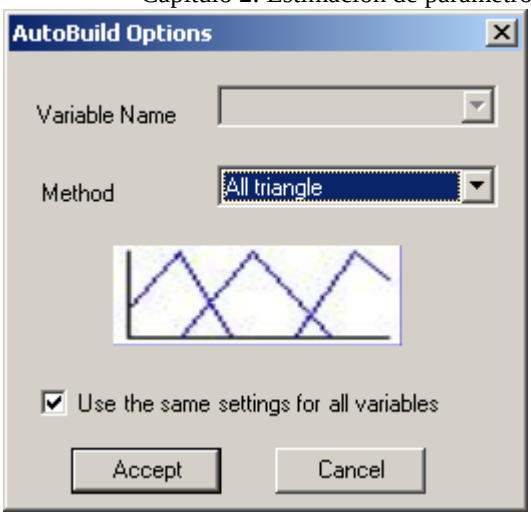


Figura 8 Selección del método de construcción de los rasgos

La opción de usar esta configuración para todas las variables en este caso será ignorada (**Use the same settings for all variables**)

Encima de esta opción se le mostrara al usuario una imagen alegórica al método seleccionado para darle una noción de cómo van a quedar las FP construidas por este método.

Variables| Auto| All Variables

Es igual que la opción explicada anteriormente, la única diferencia es que si esta marcada la opción (**Use the same settings for all variables**) todos los rasgos van a ser construidos siguiendo el mismo método, en caso de desmarcarse, se puede seleccionar un método diferente para cada rasgo

Variables| Add FP

Adiciona una FP de tipo triangular al rasgo seleccionado. El rasgo tiene que ser de tipo Lineal y debe tener un valor valido para el soporte, más adelante se explicará donde cambiar este valor

Variables| Delete FP

Elimina la FP seleccionada en el Plotter.

Menú Evaluar (Evaluate)

Evaluate| Objetive Traits

Con esta opción se seleccionan cuales de los rasgos van a ser considerados como objetivos (Marcados) o predictores (No marcados).

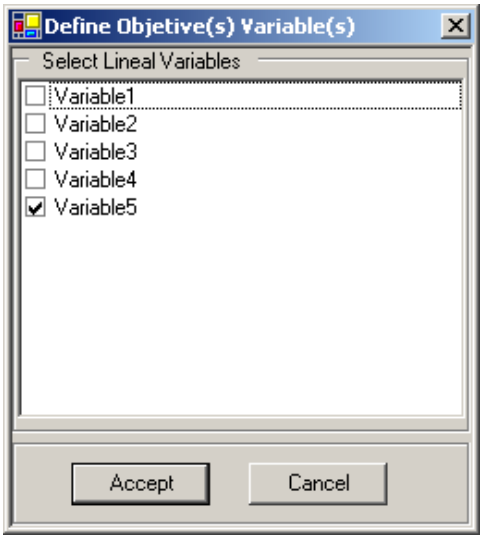


Figura 9 Selección de los rasgos objetivos

Evaluate| Standard

Selecciona el método de evaluación Standard, por este método son consideradas todas las FP de cada rasgo para cada caso

Evaluate| Max

Selecciona el método de evaluación Máximo, por este método solo es considerado el máximo valor de todas las FP de cada rasgo, el resto se considera cero, para cada caso

Evaluate| Max1

Igual que la anterior con la diferencia que el máximo será igualado a uno

Evaluate| Epsilon

Selecciona el método de evaluación Epsilon, por este método se consideran todos los valores de FP que sean mayores o iguales a epsilon, de cada rasgo, el resto se considera cero, para cada caso. Epsilon está entre cero y uno.

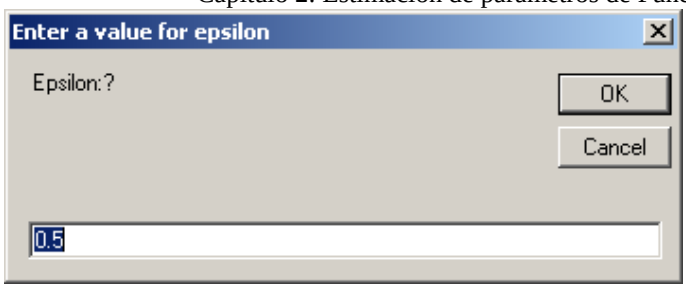


Figura 10 Elección del valor de Epsilon

Evaluate| SIAC

Selecciona el modelo de red SIAC

Evaluate| IAC

Selecciona el modelo de red IAC

Evaluate| SIAC

Selecciona el modelo de red SIAC

Evaluate| Evaluate Model

Evalúa la base de casos usando la definición de FP, el modo de evaluación (Standard, Max, Max1, Epsilon) con el modelo de red seleccionado (SIAC, IAC)

Tune

Ejecuta el algoritmo de ajuste de parámetros a las FP

Ventana Principal

Al ejecutar la herramienta, se muestra una ventana como la figura 3.7, a continuación explicaremos brevemente las partes fundamentales de esta.

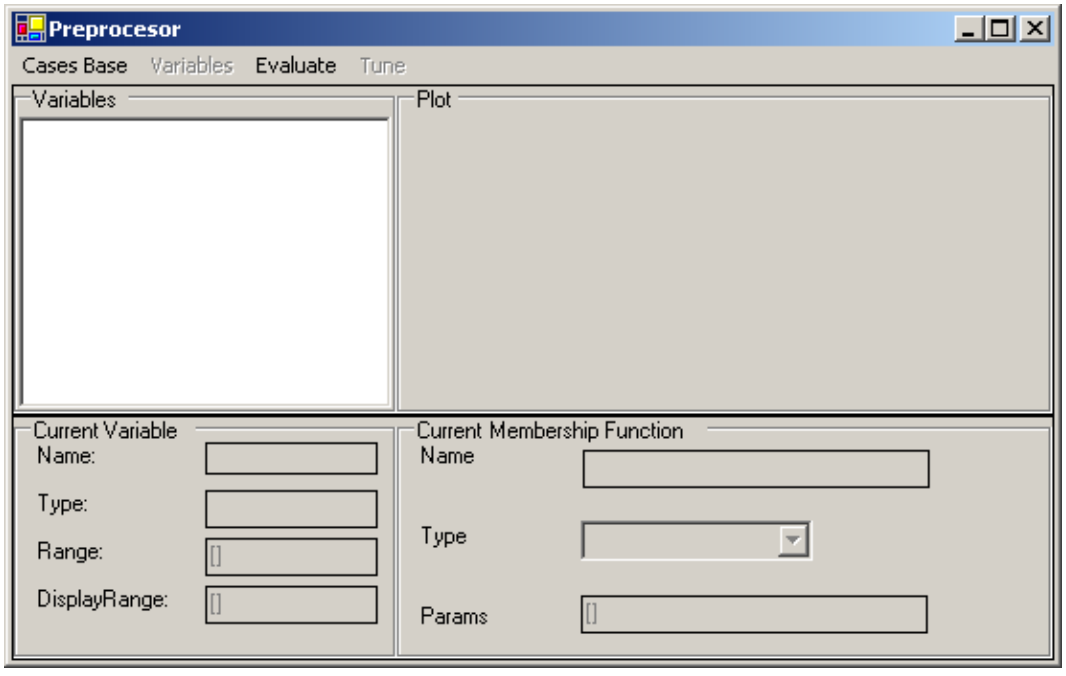


Figura 11 Ventana principal de la aplicación

Variables

Una vez cargada la base de casos, se mostrará un listado de los rasgos disponibles, así como un icono que identifica si el rasgo es Lineal  o Simbólico  .

Se le puede cambiar el nombre al rasgo, solo hay que seleccionar el rasgo deseado y dar clic sobre el, y escribir el nuevo nombre.

Current Variable

Aquí aparecerán algunos datos del rasgo seleccionado, tales como el Nombre, tipo, y en caso de ser lineal, el soporte.

En este puede ser modificado el valor del soporte, así como la escala del Plot, siempre respetando el formato [A, B] donde B tiene que ser mayor que A y son los extremos del intervalo deseado.

Plot

Este es el componente de la interfaz gráfica más importante, es aquí donde se mostrarán todas las FP correspondientes al rasgo seleccionado en Variables, y adicionalmente permite seccionar una FP así como cambiar sus parámetros desplazando a la izquierda o a la derecha, cambiar uno de los puntos sensibles de esta, y automáticamente ajustará los parámetros de la FP para que en ese punto tenga el valor predeterminado.

Los puntos sensibles de las FP solo se pueden cambiar por el eje de las “x”

Current Membership Function

Aquí aparecerán algunos datos de la FP seleccionada en el Plot. Los datos que aparecen son, Nombre, Tipo y parámetros. Nombre, admite cualquier cadena de caracteres. El tipo puede ser cambiado, seleccionando otro en la lista de los disponibles, la FP cambiará al tipo seleccionado. Los parámetros pueden ser cambiados manualmente, siempre respetando la cantidad y con el siguiente formato [p1, p2,…, pn] donde pi son los parámetros de la FP.

2.3.2 Manual del programador

Antes de entrar en detalles de cómo se diseñaron cada una de las partes de la herramienta, describiremos los principales conceptos de la Programación Orientada a Objetos (**POO**), pues sobre ella se soporta toda la estructura interna de la aplicación.

Conceptos básicos de la Programación Orientada a Objetos (POO).

El siguiente resumen ha sido extraído de [ARC01] y de [BOO91].

Según el diccionario filosófico, es obvio que el concepto de objeto no coincide completamente con ninguna de las especificaciones posibles. Las cosas, los cuerpos físicos, las entidades lógicas y

matemáticas, los valores, así como los estados psíquicos son todos objetos especificados o especificables por procedimientos regulares.

Los objetos tienen características particulares, propiedades que los identifican como se muestra a continuación. Según [MED97], los objetos poseen:

- Estado: Se define a partir de los valores que en un momento dado tienen los atributos del objeto. La estructura del objeto se define como el conjunto de todos los atributos o propiedades. Además, un objeto puede conocer o contener a otros objetos, estas relaciones son también parte de su estado.
- Comportamiento: Define cómo actúan los objetos frente a estímulos externos en términos de cambio de estados. Aquí se introduce el concepto de mecanismo como proceso de comunicación entre objetos, donde un objeto cliente actúa sobre uno servidor. Si un objeto servidor genera otros estímulos como parte de un mecanismo, entonces se le llama agente. Los mecanismos necesitan que el objeto servidor ofrezca una interfaz o protocolo de comunicación conocida por el cliente.
- Identidad: Esta es la propiedad de un objeto que lo distingue del conjunto de todos los demás objetos del universo al que pertenece. Los modelos de POO son representaciones abstractas de este tipo.

  El protocolo de objeto define la envoltura del comportamiento admitido por el objeto, representa todas las vistas estáticas y dinámicas del objeto. Para la mayoría de las abstracciones no triviales, es útil dividir los protocolos en grupos lógicos de comportamiento. Las colecciones que dividen el espacio del comportamiento de un objeto denotan los roles que un objeto puede jugar. Un rol es una máscara con la cual se presenta y define un contrato entre la abstracción y sus clientes.

El marco de referencia conceptual en un sistema orientado a objeto es el modelo de objetos que incluye cuatro conceptos fundamentales: la abstracción, el encapsulamiento, la modularidad y la jerarquía. También existen otros conceptos secundarios dignos a tener en cuenta, como los tipos, la concurrencia y la persistencia.

A continuación describiremos con más profundidad algunos de los conceptos a los que nos referimos con anterioridad.

Abstracción: Denota las características esenciales de un objeto que lo distinguen de todos los demás tipos de objeto y proporciona así fronteras conceptuales nítidamente definidas respecto a la perspectiva del observador.

Encapsulamiento: Es uno de los principios más importantes de la POO. Ha permitido a los programadores construir flexibles y poderosos objetos de software que pueden ser reusados fácilmente por otros programadores. Constituye el proceso de almacenar en un mismo compartimiento los elementos de una abstracción que constituyen su estructura y su comportamiento. Permite que los clientes se interesen sólo en el comportamiento del objeto, es decir, lo que hace y no cómo lo hace.

Modularidad: Es la propiedad que tiene un sistema que ha sido descompuesto en un conjunto de módulos cohesivos y débilmente acoplados.

Jerarquía: Es una clasificación u ordenamiento de abstracciones.

Concurrencia: Es la propiedad que distingue un objeto activo de uno que no está activo.

Persistencia: Es la propiedad de un objeto por la que su existencia trasciende el tiempo, el espacio, o ambos.

En [MED97] se definen los conceptos de clase y tipo como a continuación se muestran:

Clase: Es una representación abstracta que define la estructura y el comportamiento que le son comunes a un grupo de objetos. Estas se organizan jerárquicamente en el proceso de clasificación. Elementos comunes a una o más clases se generalizan en otra más abstracta, a las relaciones de generalización se les conoce como de herencia, existiendo herencia simple y múltiple.

Las clases más abstractas tienen mayor extensión por representar una población mayor de objetos, mientras que las más específicas tienen mayor contenido.

Tipo: Es un protocolo usado en los mecanismos de comunicación e interacción entre objetos. Tienen identidad y generalmente están más relacionados a los mecanismos de comunicación que a la propia naturaleza de los objetos. Estos establecen los atributos y las operaciones esenciales para un proceso de comunicación determinado.

No definen cómo se comportan las operaciones de un objeto, sino qué operaciones pueden ser usadas en determinado contexto. El uso de los tipos ofrece otro nivel de abstracción en un modelo. Las relaciones entre las clases son, en lo conceptual, muy fuertes, su sentido está ligado a la naturaleza de los objetos que representan. Los tipos, por su parte, ofrecen mayor libertad al depender fundamentalmente de la naturaleza de los mecanismos.

Interfaces.

El uso de las interfaces en la Programación Orientada a Objetos es una tendencia actual. Ellas esclarecen el diseño, lo hacen más cercano a la realidad y facilitan la implementación.

El nombre de una **interfaz** suele ser precedido por una I, por convenio (aunque la real identidad de una interfaz está dada por su GUID, el cual es usado para las operaciones entre ellas). Cuando

un objeto “implementa una interfaz”, ese objeto implementa cada función miembro de la interfaz. Los objetos pueden, por supuesto, soportar simultáneamente múltiples interfaces.

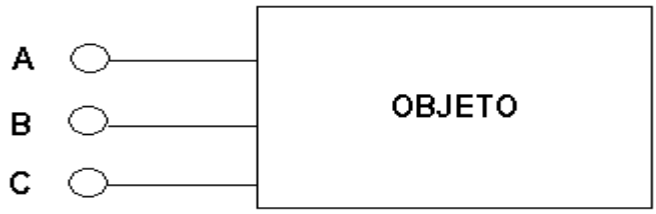


Figura 12 Un objeto que soporta las interfaces A, B y C.

La figura 2.1 muestra el esquema de un objeto típico que soporta tres interfaces A, B y C, para las cuales tiene que implementar cada uno de los métodos por ellas definidos.

Las interfaces son inmutables, debido a que nunca pueden ser versionadas, por lo que esto elimina los problemas de versionamiento. Una nueva versión de una interfaz, creada por la adición o eliminación de funciones, o cambios semánticos, es enteramente una nueva interfaz a la cual se le asigna un nuevo identificador (ID) único; por lo que la nueva interfaz no va a crear “conflictos” con la vieja.

La funcionalidad de encapsular los objetos accedidos a través de interfaces hace al sistema abierto y extensible. Es abierto en el sentido de que cualquiera puede proveer una implementación de una interfaz definida y cualquiera puede desarrollar una aplicación que use dichas interfaces, y es extensible en el sentido de que interfaces nuevas, o extendidas puedan ser definidas sin cambiar las aplicaciones existentes y estas aplicaciones extienden las nuevas interfaces y las explotan mientras continúan inter-operando con las viejas aplicaciones a través de las interfaces viejas.

Implementación computacional

A continuación damos una exhaustiva explicación de cada uno de los principales componentes del sistema

Funciones de pertenencia

- **TMembershipFunctionBase**

Es la clase base para modelar las FP. Ver Jerarquía de clases en el anexo 1

Método	**Función**
BeginUpdate	Pone en falso la bandera de chequeo de parámetros
EndUpdate	Pone en verdadero la bandera de chequeo de parámetros
Evaluate	Evalúa un valor, el resultado es un numero entre cero y uno
SetParameters	Aplica el valor especificado al parámetro dado
Propiedad	
InstanceName	Nombre de la instancia. Ejem. *FP1*
Name	Nombre del tipo de la función de membresía. Ejem. *Triangle*
ParamsList	Lista de parámetros con el formato [p_1, p_2,…]
ParametersCount	Indica la cantidad de parámetros de la FP
Sección Protegida	
_Update	Bandera de chequeo de parámetros
TMemberShipFunctionParameter()	Arreglo con los parámetros de la FP

- **TMembershipFunction**

Es la clase base para las funciones que son dibujables por el componente plot. Además contiene todos los métodos necesarios para el ajuste de los parámetros y el cambio de una FP a otra. Es una especialización de *TMembershipFunctionBase*

Método	**Función**
Move	Desplaza la FP a la derecha o a la izquierda
isValidParams	Retorna verdadero si los parámetros de la FP son validos
MoveFPoint	Mueve un punto sensible de la FP, ajusta los parámetros para que la FP pase por este punto
SetFPoint	Fija un punto sensible, ajusta los parámetros para que la FP pase por este punto
Area	Calcula el área de la FP
Adjust	Ajusta los parámetros de la FP
Propiedades	
FPointsCount	Cantidad de puntos sensibles
FPoints	Puntos sensibles
FCenter	Centro de la FP
Left	Extremo izquierdo de la FP
Rigth	Extremo derecho de la FP
Sección Protegida	
_Points	Arreglo con los puntos sensibles de la FP

¿Cómo adicionar una nueva FP?

Hay dos posibilidades, la primera es heredando de *TMembershipFunctionBase*, en este caso la FP resultante no sería dibujable. Ver ejemplo 1 anexo 2. La otra posibilidad es heredar de *TMembershipFunction*. Ver ejemplo 2 anexo 2

- **TBuildMFBase**

Es la clase base para los algoritmos de estimación de los parámetros de las FP

Método	**Función**
Run	Ejecuta el algoritmo de estimación de los parámetros de las FP
GetName	Retorna el nombre del algoritmo
Propiedades	
Data	Arreglo con los datos del rasgo
Interval	Intervalo correspondiente al termino lingüístico
GetMF	Retorna la FP construida
Parameters	Parámetros del algoritmo
Sección Protegida	
_MF	FP Generada
_Interval	Intervalo correspondiente al termino lingüístico
_Parameters	Parámetros del algoritmo
_Data	Arreglo con los datos del rasgo

¿Cómo adicionar un nuevo algoritmo de estimación de los parámetros de las FP?

Para esto es necesario heredar de *TBuildMFBase* e implementar el método *run*, finalmente, en este método es necesario construir la función deseada y asignar este valor a *_mf.* Ver Ejemplo 3 Anexo 2

- **TAbSBiscretizer**

Es la clase base para los métodos de discretizacion.

Métodos	**Función**
Run	Ejecuta el método de discretización
GetIntervals	Retorna los intervalos generados
Propiedades	
TargetTrait	Rasgo objetivo
SelectedTrait	Rasgo a discretizar
CasesBase	Nombre del fichero de la base de casos
Sección Protegida	
_missingInfoChar	Carácter que indica ausencia de información
_intervals	Intervalos generados por el algoritmo de discretización
_params	Tabla hash con los parámetros del algoritmo (si tiene)
_targetData	Arreglo con los datos del rasgo objetivo
_traitData	Arreglo con los datos del rasgo a discretizar
_casesBase	Nombre de la base de casos
_selectedTrait	Índice en la base de casos del rasgo a discretizar
_targetTrait	Índice en la base de casos del rasgo objetivo

¿Cómo adicionar un nuevo discretizador?

Para esto es necesario heredar de la clase *TAbSBiscretizer*, e implementar el método *run*, este debe colocar los intervalos resultantes en *_intervals*. Si el nuevo discretizador tiene parámetros adicionalmente debe heredar de la interfase *IParameterizableDiscretizer*. Ver ejemplo 4 anexo 2

- **TVariable**

Esta clase se usa para modelar un rasgo. Puede ser de tipo lineal, o simbólico; se le pueden adicionar y eliminar FP.

Métodos	**Función**
Evaluate	Evalúa un valor en una o más FP
RemoveMemberShipFunction	Elimina una FP
SortMembershipFunctions	Ordena las FP por su centro(si la variable es lineal)
AddMemberShipFunction	Adiciona una FP
Intersect	Calcula la intersección entre una FP y sus adyacentes, o entre todas las FP(si la variable es lineal)
CheckArea	Chequea que la intersección entre cada FP y su adyacente sea menor que el 25 % de la suma de sus áreas, y que esta intersección no sea cero
Propiedades	
MembershipFunctions	Funciones de membresía contenidas en la variable
Support	Soporte de la variable(si la variable es lineal)
Name	Nombre de la variable
MemberShipsFunctionsCount	Cantidad de FP

- **Plot**

Este componente permite cambiar de modo visual, usando los puntos sensibles de las FP, los parámetros, así como desplazarla a la izquierda y a la derecha. Para lograr esto es necesario pasarle un objeto de tipo variable, que tiene que ser de tipo lineal, que contiene las FP.

Propiedades	
X	Cantidad de “marcas” por el eje de las X
LoxX	Mínimo valor por el eje de las X
LowY	Máximo valor por el eje de las X
Variable	Variable de la cual se mostraran sus FP
ActiveMembershipFunction	Objeto de tipo TCurrentMembershipFunction

- **Variables**

Esta clase maneja una lista de las variables, y hace distinción entre las variables lineales y simbólicas (se diferencian por el icono). Es aquí donde se le puede cambiar el nombre a las variables

Propiedades	**Función**
Plotter	Objeto del tipo Plot
CurrentVariable	Objeto del tipo CurrentVariable
Variable	Lista de las variables
ActiveVariable	Variable que esta seleccionada, si no hay ninguna tiene valor nothing (null)

- **CurrentVariable**

Con esta clase se puede ver el nombre y el tipo de la variable seleccionada, además de dar la posibilidad de cambiar el soporte de esta y la escala del Ploter (si la variable seleccionada es lineal)

Propiedades	**Función**
Plot	Objeto del tipo Plot
Variable	Variable de la que se mostrarán los datos (cuando esta asociado al componente variables, este actualiza automáticamente esta propiedad)

- **CurrentMembershipFunction**

Con este componente se pueden editar manualmente los parámetros de la FP activa el ploter, también permite cambiar de un tipo a otro de FP usando la clase Transform. Y cambiar el nombre de la FP

Propiedades	**Función**
Variable	Variable activa en el ploter
FunctionIndex	Índice de la FP activa en el ploter

- **TTrnasform**

Esta clase esta diseñada para mutar una FP a cualquier tipo de los disponibles (de los predefinidos, Triangular, Trapezoidal, Gausiana, Sigmoidal) para extender esta funcionalidad a otras

FP es necesario hacer lo siguiente para cada FP que se quiera adicionar. Es muy importante adicionar este tipo de FP a las que se reconocen en el componente *CurrentMembershipFunction*

Métodos	
Convert	Convierte de un tipo de FP a otro
Propiedades	
FMToTransform	FP a convertir

Case "FP"
FP = New FP
FP.InstanceName = FPToTransform.InstanceName
FP.BeginUpdate()
FP.Left = FPToTransform.Left
FP.Rigth = FPToTransform.Rigth
FP.FCenter = FPToTransform.FCenter
FP.EndUpdate ()

Capitulo 3 Resultados experimentales

Para demostrar la factibilidad de aplicar el módulo de preprocesamiento implementado, los experimentos a realizar se dividirán en dos etapas. La primera tiene como objetivo mostrar que aplicando las heurísticas propuestas se obtienen automáticamente FP apropiadas a partir de ejemplos. La segunda está orientada a mostrar los resultados obtenidos con el algoritmo propuesto para modificar los parámetros en las FP de tipo triangular. Para ello los datos preprocesados se probaron con el modelo RNA-Borroso (epígrafe 1.4). En ambos casos se realizan experimentos utilizando los archivos de datos de la UCIMLR (ver anexo 2).

Además el procedimiento implementado se emplea en preprocesar los datos para una aplicación real, comparando estos resultados con los que se obtienen en [ROD03] utilizando el criterio de los expertos humanos.

Se definen cinco variantes para obtener automáticamente los parámetros, a partir de las heurísticas para los cuatro tipos de FP que se consideran:

1. Todos los términos lingüísticos asociados a un rasgo como FP Triangulares
2. Similar al anterior, pero el primer y último término son FP Trapezoidales
3. Todos los términos lingüísticos asociados a un rasgo como FP Trapezoidales
4. Todos los términos lingüísticos asociados a un rasgo como FP Gausianas
5. Similar al anterior, pero el primer y último término como FP Sigmoidales

La tabla 1 y 2 muestran el desempeño del modelo seleccionado con cada una de estas variantes y el CE, resaltando la variante de mejores resultados en cada archivo de datos. En particular los resultados de aplicar todas las variantes a *"Iris.data"* se muestran en el anexo 4.

Otra tabla similar a la uno aparece en el anexo 5, pero utilizando Chi^2 como discretizador. Nótese que en cuatro archivos de datos no fue posible aplicar las heurísticas por limitaciones de la implementación computacional del Chi^2 que no permite más de seis clases; y que estos resultados difieren de los obtenidos aplicando CAIM como discretizador.

En general, estas tablas muestran que no hay una heurística superior en todos los casos, sino que esta depende del problema a solucionar. En particular la variante 4 de la tabla 1 (todas gausianas) muestran mejores resultados en *"Iris.data"* y *"WBC.data"*, y similares a los reportados en [NAU97] utilizando el sistema de inferencia borroso NEFCLASS (96.67% y 96.05% respectivamente).

Archivo de datos	Valor del Desempeño del modelo RNA-Borroso con el conjunto de entrenamiento (%)				
	1	2	3	4	5
1-Iris.data	94.67	94.00	**95.33**	**95.33**	92.00
2-WBC	92.84	94.99	94.99	**96.85**	No
3-Cleveland	82.51	83.83	83.83	**85.81**	No
4-Credit-app	**85.22**	84.92	84.92	83.91	83.48
5-Hepatitis	79.35	83.87	**83.87**	63.23	No
6- Glass	69.63	73.36	**92.36**	70.56	No
7- Vehicle	82.03	82.15	82.21	**82.33**	No
8- Vowel	**84.76**	85.27	86.09	86.57	**86.76**
9- Anneal	92.42	**92.87**	91.84	92.65	No
10- Segmentation	93.10	93.48	93.09	**93.56**	No
11- Letter	97.19	96.33	96.32	**97.23**	No
12- Hypothyroid	60.09	60.18	61.07	60.07	**61.29**
13- Diabetes	75.78	72.66	72.66	73.18	**74.48**
14- Ionosphere	64.39	**70.37**	**70.37**	65.81	No
15- Liver-disorders	59.42	61.45	61.45	**65.22**	62.60
16- Sonar	73.08	73.08	73.08	**87.5**	80.77
17- Zoo	**99.13**	**99.13**	99.03	99.11	No
18- Wine	94.76	94.76	**96.25**	94.01	94.38
22-Acancer (Aplicación real)	78.04	78.04	78.04	78.15	**78.46**

Tabla 1. Resultados con las FP propuestas automáticamente (CAIM)

Archivo de datos	Valor del Desempeño del modelo RNA-Borroso con el conjunto de entrenamiento (%)				
	1	2	3	4	5
20- Flag(Mult)	71.61	68.40	68.35	**74.62**	71.55
21- Horse-colic	94.21	94.19	94.28	94.30	**94.32**
22- Solar-flare	**93.14**	89.60	90.59	91.05	90.79

Tabla 2. Resultados con las FP propuestas automáticamente (K-Means)

A continuación se realiza una comparación de los resultados de aplicar las heurísticas propuestas (la variante de mejores resultados en cada uno de los archivos de datos según la tabla 1), y la heurística utilizada por Hong [HON96]. Para ello se utilizó la técnica de validación cruzada con 10 particiones. Observe en la tabla 3 los resultados del desempeño promedio en cada variante. Nótese que en cinco de los archivos de datos

utilizados los resultados de la heurística propuesta son superiores, a partir de comparar el desempeño medio con la heurística de Hong y el intervalo de confianza que se muestra.

Además, utilizando el paquete estadístico SPSS, se compararon los valores medios del desempeño del modelo con cada variante. Esto se realizó utilizando el test de Wilcoxon (por el tamaño de la muestra se utilizan pruebas no paramétricas), cuyos resultados muestran diferencias significativas (Anexo 6).

Archivo de datos	Con una de las heurísticas propuestas		Con la heurística de Hong	Observaciones
	Media	Intervalo de confianza (α=0.05)	Media	
Ionosphere	70.37	(68.27,72.48)	**79.67**	No lo supera
Iris	**94.22**	(92.34,96.09)	77.85	Supera
Sonar	**83.33**	(81.72,84.94)	56.9	Supera
Liver_disorders	63.07	(61.10,65.05)	**72.12**	No lo supera
Diabetes	**73.76**	(73.12,74.41)	44.24	Supera
Credit-app	**83.91**	(83.31,84.50)	69.88	Supera
Cleveland	**83.18**	(79.72,86.62)	52.03	Supera

Tabla 3. Resultados de la comparación con la heurística de Hong (Se emplea el modelo SIAC)

La tabla 4 muestra los resultados de aplicar el algoritmo para modificar los parámetros de una FP, de tipo triangular, en el modelo RNA-Borroso. Nótese que en todos los casos se mejora el desempeño del modelo.

Además se realizaron diferentes corridas con *"Iris.data"* utilizando las FP iniciales por diferentes vías. Los resultados (tres primeras filas de la tabla 4) muestran que partiendo de las FP obtenidas aplicando la heurística propuesta, se obtienen mejores resultados y en menor número de épocas que si se parten de FP propuestas manualmente.

La aplicación de este algoritmo al archivo de datos *"Flag"* (tabla 4) muestra que este algoritmo es aplicable, aunque no haya una definición a priori del conjunto de rasgos objetivos. Aunque en ambos casos se mejora el desempeño, este resultó mejor en la variante donde esta definición se hizo previa a su aplicación. Para estas corridas se fijó un máximo de iteraciones igual a 25 épocas.

Archivo de datos	Criterio para seleccionar FP iniciales	Valor del Desempeño del modelo SIAC con el conjunto de entrenamiento (%)		
		Utilizando FP iniciales	Con las FP modificadas	Cantidad de épocas
Iris	Automática	93.3	**94.67**	4
	Manual	72.00	**77.33**	10
	Manual	82.00	**86.66**	25
Cleveland	Automática	82.83	**83.82**	3
Flag (Sin definir Co)	Automática	74.69	**77.11**	25
Flag (Co = {4,5,11,12})	Automática	74.69	**81.07**	25

Tabla 4. Resultados con en el algoritmo Modificar_Parametros (con FP Triangulares)

En la aplicación real de determinar las propiedades anticancerígenas en el diseño de un fármaco se cuenta con una muestra de 961 ejemplos, donde cada uno se describe por 11 rasgos predictores, que toman valores continuos, y la clase a la que pertenece. En el trabajo que se presenta en [ROD03] se aplica el modelo hibrido que se presenta en [GAR00] para desarrollar un sistema inteligente que a partir de las características de un fármaco determine la clase correspondiente.

En esta aplicación se emplean FP de tipo sigmoidal y campana para representar los conjuntos borrosos, cuyos parámetros se definieron utilizando el criterio de expertos en el dominio de aplicación. La tabla 5 muestra los resultados que se obtienen en el trabajo referenciado y los de aplicar las heurísticas propuestas.

Criterio para seleccionar FP		Desempeño
Automáticamente	1	78.04 %
	2	78.14%
	3	78.14%
	4	78.56%
Criterio experto		77.86%

Tabla 5. Resultados con la aplicación real

Los resultados del desempeño del modelo utilizando las FP obtenidas de forma automática son mejores en todos los casos que la efectividad de este reportada en [ROD03]. Además aplicando el algoritmo propuesto para modificar los parámetros partiendo de las FP obtenidas por la variante 1, se hacen pequeñas modificaciones a los parámetros hasta la época 10, pero que no repercuten en el desempeño del modelo.

Conclusiones

Con este trabajo se obtiene como resultado el diseño e implementación de un módulo que automatiza la definición de las funciones de las neuronas de preprocesamiento, asociadas a rasgos lineales que son modelados utilizando conjuntos borrosos, en un modelo RNA-Borroso. También le brinda facilidades al usuario para editar de forma manual (gráfica y analíticamente) las FP. Los resultados fundamentales son:

- Un módulo dentro del preprocesador que facilita la construcción manual de las FP; brindándole al usuario facilidades de graficación, modificación y cambios de tipos de FP.
- Se propusieron e implementaron 4 heurísticas que construyen automáticamente las FP a partir de ejemplos de entrenamiento. Los resultados experimentales muestran la factibilidad de las mismas, y qué la definición de las FP depende del problema e influye en el desempeño del modelo utilizado. Su implementación orientada a componentes permite que puedan ser fácilmente incorporadas a otros sistemas que requieran modelar rasgos lineales utilizando conjuntos borrosos.
- Se implementó y evaluó un algoritmo que ajusta los parámetros de una FP de tipo triangular para mejorar el desempeño del modelo RNA-Borroso que se utiliza. La implementación realizada hace extensible este procedimiento a otros tipos de FP.
- Los experimentos realizados utilizando archivos de datos internacionalmente reconocidos, y una base de casos suministrada por el CBQ y que representa un problema real de clasificación de propiedades anticancerígenas de compuestos químicos; muestran resultados satisfactorios.

Recomendaciones

A partir de los resultados obtenidos recomendamos que:

1- Incorporar el módulo de preprocesamiento a las herramientas: “NeuroEvaluator” y “SISIFuzzy”.

2- Implementar otro método de agrupamiento que no requiera como entrada el número de grupos, para automatizar completamente el procedimiento de obtener los términos lingüísticos en problemas donde estos se recomiendan.

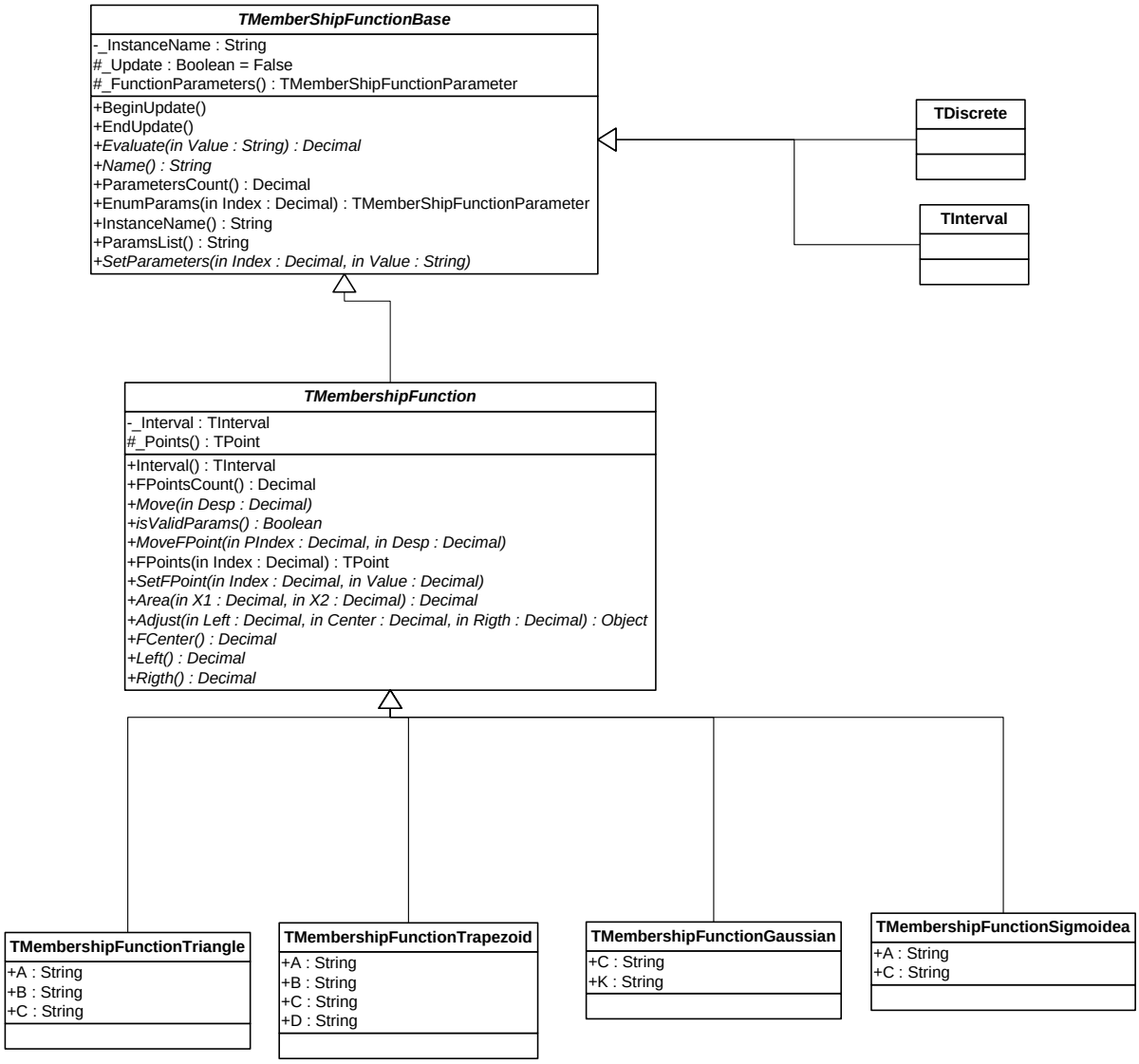
3- Validar el algoritmo para modificar los parámetros de las FP de tipo trapezoidal, Gaussiana y Sigmoidal.

Bibliografía

[BEL02]	Bello, P.R. et. al, Aplicaciones de la Inteligencia Artificial. Ed. Universidad de Guadalajara. 2002
[BOD01]	Bodyanskiy, Yevgeniy; Kolodyazhniy, Vitaliy; Stephan, Andreas. "An Adaptive Learning Algorithm for a Neuro-Fuzzy Network", 7th Fuzzy Days in Dortmund, 2001.
[BOT01]	Botzheim, J.; Extracting Trapezoidal Membership Functions of a Fuzzy Rule System by Bacterial Algorithm. Proceeding in LNCS Computational Intelligence (2001).
[CAS00]	Castellano G. et. al.: Fuzzy inference and rule extraction using a neural network. Neural Network World, Volume 10 (2000).
[CHE95]	Chen, Joseph E. Otto, Kevin N. Constructing membership functions using interpolation and measurement theory. Fuzzy sets and systems 73 pp. 313-327. 1995
[CHI96]	Chin-Teng Lin y C.S George Lee Neural Fuzzy Systems: a neuro-fuzzy synergism to intelligent systems. Prentice Hall. 1996
[FAB00]	Fabri J.A.: Aplicao de um sistema especialista fuzzy a classificao de flores (IRIS). Simposio brasileiro de IA. (2000)
[FAL03]	Falcón, R.J. Herramienta para el desarrollo de sistemas conexionistas en presencia de rasgos difusos. Trabajo de Diploma. 2003.
[GAR00]	García, et. al. Usando conjuntos borrosos para implementar un modelo para sistemas basados en casos interpretativos. In Proceedings of IBERAMIA-SBIA 2000, Eds por M. C. Monard y J.S. Sichman, Sao Paulo, Brasil, Nov. 2000.
[GAR03]	García, María M., et. al, Redes Neuronales Artificiales. Edit. Universidad de Guadalajara. México. 2003.
[GAR96]	García, MM and Bello, P.R.. A model and its different applications to case-based reasoning. Knowledge-based systems 9 (1996) 465-473.
[HAT00]	Hatzilygeroudis, I. et. al.: Neurules: Improving the performance of symbolic rules. International Journal on Artificial Intelligence Tools (IJAIT), vol. 9, No. 1 (2000).
[HIL95]	Hilera, José R. y Martínez, Victor J.,Redes Neuronales Artificiales. Fundamentos, Modelos y Aplicaciones. Madrid, España.1995.
[HON88]	Hong, J. On the handling of fuzziness for continuous-valued attributes in decision tree generation. Fuzzy Sets and Systems 99. 1988.
[HON96]	Hong, Tzung – Pei. Lee, Chai – Ying. Induction of fuzzy rules and membership functions from training examples. Fuzzy sets and systems 84. pp 33-47. 1996.
[HON98]	Hong, Tzung-Pei. Lee, Chai-Ying. Learning Fuzzy Knowledge from Training Examples. CIKM. pp 161-166. 1998.
[JAN93]	Jang, J.S. ANFIS: Adaptative Network-Based Fuzzy Inference Systems. IEEE Trans. Systems, Man & Cybernetics 23 pp. 665-685. 1993
[JYN98]	Jyh-Shing, R. et. al. Neuro-Fuzzy and Soft Computing. Prentice Hall (1998)
[KUN98]	Kunchiva, Liudmila I. et. al. Fuzzy diagnosis. AI in medicine. 1998

[KUR04]	Kurgan L.; et. al.: CAIM Discretization Algorithm. IEEE Transactions on Knowledge and Data Engineering. Vol 16. No. 2 (2004)
[LIU95]	Liu, Huan. Setiono, Rudy.Chi2: Feature selection and discretization of numeric attributes. In proceedings of the IEEE 7th International Conference on Tools with Artificial Intelligence, Washington D.C. pp. 388-391. November, 1995.
[LIU97]	Liu H. and Setiono R.: Chi2: Attribute selection and discretization of numeric attributes. In Proceedings of the IEEE 7th Conference on tolls with AI.(1997)
[MAT01]	Matich, Damián Jorge. Redes neuronales. Conceptos básicos y aplicaciones. Universidad Tecnológica Nacional. Marzo, 2001
[MCC89]	McClelland, J.L. y Rumelhart, D.E. Explorations in parallel distributed processing.MIT Press, 1989.
[NAU97]	Nauck D. et. al, Foundations of neuro-fuzzy systems, John Wiley & Sons Ltd. 1997.
[PIÑ03]	Piñeiro, P. et. al.: Algoritmos genéticos en la construcción de funciones de pertenencia. Revista Iberoamericana de Inteligencia Artificial (AEPIA) No. 18, Vol. 2, Invierno (2003). ISSN 1137-3601.
[SAN00]	Sankar K.Pal, et. al. Soft Computing in Case Based Reasoning. Edit. Springer-Verlag London. 2001
[SET00]	Rudy Setiono. "Generating concise and accurate classification rules for breast cancer diagnosis", Journal of Artificial Intelligence in Medicine. Vol. 18, pp. 205-219, 2000.
[WIL97]	Wilson, Randall and Martinez, Tony R. Improved Heterogeneus Distance Functions. Journal of AI Research No 6. 1997
[ZAD76]	Zadeh L.. The concept of a linguistic variable and its application to approximate reasoning. Information Science 9, page. 43-80,1976.
[ZUR92]	Zurada, J. M.: Introduction to Artificial Neural Systems, West Publishing (1992)

Anexo 1: Jerarquía de clases



Anexo 2: Ejemplos de extensión de los componentes

Ficheros requeridos: *MemberShipFunctionBase.dll*

Ejemplo 1

```
Imports MembershipFunctionBase
Public Class TDiscrete
  Inherits TMemberShipFunctionBase
  Public Overrides Function Evaluate (ByVal Value As String) As Decimal
    Implementar
  End Function
 Public Overrides ReadOnly Property Name() As String
   Get
     Name = Nombre de la función
   End Get
 End Property
 Public Overrides Sub SetParameters(ByVal Index As Decimal, ByVal Value As String)
    Implementar
 End Sub
 Sub New()
   Es muy importante inicializar el arreglo de parámetros con la cantidad de elementos necesaria para la nueva función.
    ReDim _FunctionParameters(Cantidad de parámetros)
 End Sub
End Class
```

Ficheros requeridos: *MemberShipFunctionBase.dll*

Ejemplo 2

```
Imports MembershipFunctionBase
Public NotInheritable Class TMemberShipFunctionTriangle
 Inherits TMembershipFunction
Public Overrides ReadOnly Property Name () As String
Get
        Name = Nombre de la función
End Get
End Property
Public Overrides Property FCenter() As Decimal Implementar
 Public Overrides Property Left() As Decimal Implementar
Public Overrides Property Rigth() As Decimal Implementar
        Implementar los métodos siguientes
Public Overrides Function Evaluate (ByVal Value As String) As Decimal
Public Overrides Sub MoveFPoint(ByVal PIndex As Decimal, ByVal Desp As Decimal)
Public Overrides Sub Move(ByVal Desp As Decimal)        Implementar
Public Overrides Sub SetFPoint(ByVal Index As Decimal, ByVal Value As Decimal)
Aquí se debe hacer un chequeo de los parámetros si _Update es falso, si es necesario.
Public Overrides Sub SetParameters(ByVal Index As Decimal, ByVal Value As String)
Public Overrides Function Area (ByVal X1 As Decimal, ByVal X2 As Decimal) As Decimal
Public Overrides Function Adjust (ByVal Left As Decimal, ByVal Center As Decimal, ByVal Rigth As Decimal) As
Object
        Chequear los parámetros, si están correctos retornar verdadero
Public Overrides Function isValidParams() As Boolean
End Class
```

Ficheros requeridos: *MemberShipFunctionBase.dll*

BuiltMemberShipFunctionBase.dll

Dll de la función a construir

Ejemplo 3

```
Imports NeuroDeveloper.MFBuildersBase
Public Class TBuidMemberShipFunctionGaussian
 Inherits TBuildMFBase
Public Overrides Function GetName() As String  Implementar
Public Overrides Sub Run()
Implementar
_MF = New Tipo de la FP(Parametros)
End Sub
Default Public Overrides Property Parameters(ByVal ParamName As String) As String Implementar
End Class
```

Ficheros requeridos: *DiscretizerUtils.dll*

Ejemplo 4

```
using System;
using System.Collections;
using NeuroEvaluator.Discretizers.Utils;
public sealed class  myNewDiscretizer: AbSBiscretizer, IParameterizableDiscretizer (si tiene parametros)
{ CONSTRUCTORS
public myNewDiscretizer(char miss): base(miss){InitParams();}
public myNewDiscretizer(string casesBase, char miss): base(casesBase, miss){InitParams();}
public myNewDiscretizer(string casesBase, ushort selectedTrait, char miss): base(casesBase, selectedTrait,
miss){InitParams();}
public myNewDiscretizer(string casesBase, ushort selectedTrait, ushort targetTrait, char miss): base(casesBase,
selectedTrait, targetTrait, miss){InitParams();}
public myNewDiscretizer(string [] traitData, char miss): base(traitData, miss) {InitParams(); }
public myNewDiscretizer(string [] traitData, string [] targetData, char miss): base(traitData, targetData, miss)
{InitParams(); }
public override void Run() Implementar
public void InitParams()  // interface method to initialize the collection of discretizer's params
{
_params = new Hashtable();
_params.Add("A", new DiscretizerParam("A", Descripcion de A, 5, DiscretizerParamType.Tipo));
}
public static string Name Implementar
public Hashtable params                    // interface method to retrieve the Hashtable collection with discretizer's
params
{
get { return _params; }
}

}
```

Anexo 3: Descripción de los archivos de datos utilizados en los experimentos

Archivo de datos	Cantidad de ejemplos	Características de los rasgos predictores		Tiene ausencia de información	Cantidad de clases
		Simbólicos	Lineales		
1-Iris	150	0	4	No	3
2-WBC	699	0	9	No	2
3-Cleveland	303	9	5	Si	2
4-Credit-app	690	9	6	Si	2
5- Hepatitis	155	13	6	Si	2
6- Glass	214	0	9	No	7
7-Vehicle	846	0	18	No	4
8- Vowel	990	1	10	No	11
9-Anneal	898	25	5	Si	6
10-Segmentation	2310	0	19	No	7
11-Letter	20 000	0	16	No	26
12-Hypothyroid	3772	16	6	Si	4
13-Diabetes	768	0	8	No	2
14-Ionosphere	351	0	34	No	2
15-Liver-disorders	345	0	6	No	2
16- Sonar	208	0	60	No	2
17- Zoo	101	16	1	No	7
18- Wine	178	0	13	No	3
19- Flag(multi)	194	18	6	No	*
20-Horse-colic	368	20	7	30.0	*
21-Solar-flare	1389	3	10	No	*

(* Significa que no está predefinida una partición de los rasgos en predictores y objetivos)

Anexo 4: Variantes de construcción automática de funciones de membresía

1. Todas triangulares
2. Todas triangulares con *
3. Triangular-Trapezoidal-triangular
4. Todas Gausianas
5. Sigmoidea-Gausiana-Sigmoidea

Base de casos: **Iris.data**

Total de casos: 150

Casos reconocidos

1	2	3	4	5
140	140	141	143	136

Definiciones

Variante 1:

Sepal_length: Fuzzy

Triangle [3.68422352, 5.03888852, 6.06762028]

Triangle [5.19238389, 5.86911772, 6.65908959]

Triangle [6.20000000, 6.69439119, 7.90000000]

Sepal_width: Fuzzy

Triangle [2.00000000, 2.66516758, 3.00000000]

Triangle [2.83989959, 2.99855060, 3.16186822]

Triangle [3.00000000, 3.37673301, 4.40000000]

Petal_length: Fuzzy

Triangle [1.00000000, 1.46120308, 3.00000000]

Triangle [1.90000000, 4.23283397, 4.80000000]

Triangle [3.89005334, 5.44781104, 8.47285136]

Petal_width: Fuzzy

Triangle [0.10000000, 0.23410187, 1.00000000]

Triangle [0.60000000, 1.34083363, 1.80000000]

Triangle [1.38712955, 2.06482351, 2.87320473]

Class: Discrete Iris-setosa, Iris-versicolor, Iris-virginica

Variante 2

Sepal_length: Fuzzy

Triangle [*, 5.03888852, 6.06762028]

Triangle [5.19238389, 5.86911772, 6.65908959]

Triangle [6.20000000, 6.69439119, *]

Sepal_width: Fuzzy

Triangle [*, 2.66516758, 3.00000000]

Triangle [2.83989959, 2.99855060, 3.16186822]

Triangle [3.00000000, 3.37673301, *]

Petal_length: Fuzzy

Triangle [*, 1.46120308, 3.00000000]

Triangle [1.90000000, 4.23283397, 4.80000000]

Triangle [3.89005334, 5.44781104, *]

Petal_width: Fuzzy

Triangle [*, 0.23410187, 1.00000000]

Triangle [0.60000000, 1.34083363, 1.80000000]

Triangle [1.38712955, 2.06482351, *]

Class: Discrete Iris-setosa, Iris-versicolor, Iris-virginica

Variante 3:

Sepal_length: Fuzzy

Triangle [*, 5.03888852, 6.06762028]

Trapezoid [5.51175946, 5.70588245, 6.06010378, 6.49992523]

Triangle [6.20000000, 6.69439119, *]

Sepal_width: Fuzzy

Triangle [*, 2.66516758, 3.00000000]

Trapezoid [2.95, 2.97252571, 3.00000000, 3.16098432]

Triangle [3.00000000, 3.37673301, *]

Petal_length: Fuzzy

Triangle [*, 1.46120308, 3.00000000]

Trapezoid [2.64671358, 3.90882990, 4.52896661, 5.65814051]

Triangle [3.89005334, 5.44781104, *]

Petal_width: Fuzzy

Triangle [*, 0.23410187, 1.00000000]

Trapezoid [0.83816150, 1.18871226, 1.49828100, 2.05875240]

Triangle [1.38712955, 2.06482351, *]

Class: Discrete Iris-setosa, Iris-versicolor, Iris-virginica

Variante 4:

Sepal_length: Fuzzy

Gaussian Bell [5.01864407, 2.45501638]

Gaussian Bell [5.86341463, 7.05646057]

Gaussian Bell [6.74117647, 2.87309742])

Sepal_width: Fuzzy

Gaussian Bell [2.64035088, 7.22912531]

Gaussian Bell [2.99629630, 323.39474856]

Gaussian Bell [3.40597015, 5.47013821])

Petal_length: Fuzzy

Gaussian Bell [1.464, 3.21950794]

Gaussian Bell [4.12826087, 1.79311878]

Gaussian Bell [5.46181818, 1.36800381]

Petal_width: Fuzzy

Gaussian Bell [0.244, 33.42723672]

Gaussian Bell [1.32363636, 3.81298543]

Gaussian Bell [2.05652174, 7.37739266]

Class: Discrete Iris-setosa, Iris-versicolor, Iris-virginica

Variante 5:

Sepal_length: Fuzzy

Sigmoidea [-2.60897503, 5.01864407]

Gaussian Bell [5.86341463, 7.05646057]

Sigmoidea [2.82239571, 6.74117647]

Sepal_width: Fuzzy

Sigmoidea [-4.47698462, 2.64035088]

Gaussian Bell [2.99629630, 323.39474856]

Sigmoidea [3.89441183, 3.40597015]

Petal_length: Fuzzy

Sigmoidea [-2.98770336, 1.464]

Gaussian Bell [4.12826087, 1.79311878]

Sigmoidea [1.94753997, 5.46181818]

Petal_width: Fuzzy

Sigmoidea [-9.62704417, 0.244]

Gaussian Bell [1.32363636, 3.81298543]

Sigmoidea [4.52266245, 2.05652174]

Class: Discrete Iris-setosa, Iris-versicolor, Iris-virginica

Anexo 5: Resultados con las FP propuestas automáticamente utilizando Chi² como discretizador

Archivo de datos	Valor del Desempeño del modelo SIAC con el conjunto de entrenamiento (%)				
	1	2	3	4	5
1-Iris.data	78.22	**81.77**	**81.77**	80.44	80.44
2-WBC	81.12	**95.71**	**95.71**	85.98	No
3-Cleveland	85.15	85.81	**86.13**	85.48	85.81
4-Credit-app	**86.81**	83.77	83.33	84.64	79.57
5-Hepatitis	83.87	83.87	**84.52**	**84.52**	**84.52**
6- Glass	90.50	89.56	92.52	93.15	**94.08**
7- Vehicle	82.03	79.91	80.32	**82.27**	80.14
8- Vowel **	Falla	Falla	Falla	Falla	Falla
9- Anneal	90.20	90.56	91.09	**92.29**	91.26
10- Segmentation **	Falla	Falla	Falla	Falla	Falla
11- Letter **	Falla	Falla	Falla	Falla	Falla
12- Hypothyroid	62.32	62.50	61.81	62.54	**62.76**
13- Diabetes	**74.87**	64.84	61.98	70.18	56.25
14- Ionosphere	81.48	86.89	80.63	86.04	**92.02**
15- Liver-disorders	82.61	80.87	80.87	**84.64**	79.42
17- Zoo **	Falla	Falla	Falla	Falla	Falla
18- Wine	**78.04**	35.93	37.89	53.79	13.46
22-Acancer (Aplicación real)	75.55	75.86	74.92	76.90	75.75