

NUEVOS ALGORITMOS DE CLASIFICACIÓN INTEGRADOS EN XFUZZY 3

Francisco José Moreno Velo¹ Iluminada Baturone² Santiago Sánchez Solano² Ángel Barriga²

¹ Dpto Tecnologías de la Información (Universidad de Huelva), francisco.moreno@dti.uhu.es

² Instituto de Microelectrónica de Sevilla (CNM-CSIC), {lumi,santiago,barriga}@imse.cnm.es

Resumen

El entorno Xfuzzy 3 está formado por un amplio conjunto de herramientas dedicadas a dar soporte a las diferentes etapas del desarrollo de sistemas difusos. Entre estas herramientas se encuentra *Xfdm*, dedicada a la extracción de conocimiento difuso a partir de conjuntos de datos. Este trabajo presenta las últimas modificaciones realizadas en esta herramienta, que consisten en la integración de los algoritmos de clasificación difusos FuzzyID3 y FuzzConRI.

Palabras Clave: clasificadores difusos, algoritmos, herramientas CAD.

1 INTRODUCCIÓN

El entorno Xfuzzy [2] es una plataforma de desarrollo de sistemas difusos complejos. El entorno está formado por un conjunto de herramientas que dan soporte a las diferentes etapas de diseño de los sistemas difusos: edición, visualización, adquisición de conocimiento, optimización, simplificación, simulación, monitorización, síntesis software, síntesis hardware. Todas estas herramientas comparten una representación común del sistema en diseño basada en un lenguaje de especificación formal llamado XFL3. Xfuzzy soporta sistemas difusos complejos, con posibilidad de definir sistemas jerárquicos que pueden incluir módulos no difusos y expresiones lógicas complejas con modificadores lingüísticos y funciones difusas (operadores, funciones de pertenencia, métodos de *defuzzificación*) definidas por el usuario.

La herramienta *Xfdm* [1] está dedicada a la extracción automática de reglas difusas a partir de conjuntos de datos. Esta herramienta desarrolla los dos enfoques más utilizados en adquisición de reglas difusas: los basados en *clustering* y los basados en *grid*. El enfoque basado en

clustering tiene como objetivo buscar grupos de datos entre los ejemplos, generando reglas difusas a partir de estos *clusters*. Las funciones de pertenencia de estas reglas se obtienen por proyección de los *clusters* sobre los universos de discurso de cada variable del sistema. *Xfdm* incluye algoritmos para generar *clusters* de forma esférica [3] y de forma elíptica con ejes libres [9] o paralelos a los ejes de coordenadas [8]. También incluye un algoritmo para generar los *clusters* de forma incremental [4]. Este enfoque tiene como ventaja que el número de reglas generadas es pequeño mientras que su desventaja principal es que las funciones de pertenencia generadas por proyección son difícilmente interpretables.

El enfoque basado en *grid* consiste en definir previamente las funciones de pertenencia de cada variable y encontrar las reglas utilizando estas funciones. El más conocido de los algoritmos que siguen este enfoque es el propuesto por Wang y Mendel [18] que recorre el conjunto de datos incluyendo entre las reglas a aquella que mejor cubre cada ejemplo. Los algoritmos basados en *grid* favorecen la interpretabilidad al utilizar funciones de pertenencia con un significado fijado previamente. Su principal desventaja es el gran número de reglas que generan. Para reducir el número de reglas algunos autores proponen escoger tan sólo las más significativas de las generadas por el algoritmo de Wang y Mendel [11][14], disminuyendo el tamaño del sistema generado al precio de aumentar el número de errores de clasificación del resultado.

La obtención de reglas lógicas para clasificar un conjunto de datos ha sido ampliamente tratada en el campo no difuso. Una de los enfoques en este campo es la inducción de árboles de decisión. Estos árboles permiten representar de forma compacta los procesos de decisión involucrados en las tareas de clasificación. En estos árboles las hojas representan la decisión final del proceso de clasificación (la clase seleccionada), los nodos internos describen la consulta acerca de un cierto atributo y las ramas de ese nodo se refieren a los diferentes valores que puede tomar dicho atributo. Los algoritmos de inducción de árboles de decisión se basan en seleccionar el atributo más adecuado para generar un nodo del árbol, en base al conjunto de ejemplos de entrada. El algoritmo CART utiliza el índice

Gini como medida para seleccionar el atributo. Por su parte, el algoritmo CHAID utiliza como medida el test de Pearson. El más conocido de estos algoritmos es ID3 [12], que utiliza como medida la entropía de información de cada atributo. El éxito de ID3 ha llevado a diversos autores a proponer versiones difusas de este algoritmo [5][15][16][19].

Entre los algoritmos clásicos dedicados a generar reglas de clasificación destaca la familia de algoritmos AQ [10]. La base de estos algoritmos consiste en seleccionar un ejemplo de clasificación positivo y utilizar sus valores para construir una regla consistente (es decir, que no cubra ningún ejemplo negativo) lo más general posible, repitiendo este procedimiento hasta cubrir a todos los ejemplos positivos. La consistencia de las reglas generadas provoca, sin embargo, que estos algoritmos sean muy sensibles al ruido. Para evitar estos problemas se propuso el algoritmo CN2 [6][7] en el que, en lugar de exigir consistencia a las reglas, estas se eligen utilizando criterios de representatividad estadística y de calidad. En el ámbito de los sistemas difusos, un enfoque similar se utiliza en el algoritmo FuzzConRI [17].

El algoritmo de Wang y Mendel y sus derivados generan reglas conjuntivas cuyos antecedentes incluyen valores de todas las variables de entrada del sistema. Esto provoca que el número de reglas crezca exponencialmente con el número de entradas. Por el contrario, los algoritmos FuzzyID3 y FuzzConRI potencian la generación de reglas en el que no todas las variables estén incluidas. Estas reglas cubren un espacio mucho mayor por lo que el resultado final es un sistema formado por un número de reglas más reducido, lo que mejora su interpretabilidad.

La estructura de este trabajo es la siguiente. El apartado 2 describe el algoritmo FuzzyID3, que genera árboles de decisión difusos. El apartado 3 describe el algoritmo FuzzConRI, que desarrolla una versión difusa del algoritmo clásico CN2. El apartado 4 presenta la implementación de los algoritmos anteriores en Xfuzzy. El apartado 5 muestra algunos resultados obtenidos por estos algoritmos sobre conjuntos de datos estándar. Las conclusiones de este trabajo se presentan en el apartado 6.

2 EL ALGORITMO FUZZYID3

El algoritmo ID3 forma parte de un conjunto de algoritmos de inducción de árboles de decisión conocidos como TDIDT (*Top Down Induction of Decision Trees*). El esquema general de estos algoritmos se presenta en la Figura 1.

Los algoritmos TDIDT construyen el árbol de decisión de forma recursiva. El caso base se produce cuando todos los ejemplos son de la misma clase (en cuyo caso se genera

```

function TDIDT(ejemplos, atributos, defecto)
returns Tree
  if ( atributos= $\emptyset$  )
    return Hoja(clase-mayoritaria(ejemplos))
  if ( ejemplos= $\emptyset$  )
    return Hoja(defecto)
  if ( $\forall e \in$  ejemplos, e.clase = C)
    return Hoja(C)

  mejor  $\leftarrow$  escoger-atributo(ejemplos, atributos)
  nodo  $\leftarrow$  crear-nodo(mejor)
  resto-atr  $\leftarrow$  eliminar(atributos, mejor)
  foreach v  $\in$  valores(mejor)
    ejemplos-v  $\leftarrow$  extraer-ejemplos(ejemplos, mejor, v)
    rama  $\leftarrow$  TDIDT(ejemplos-v, resto-atr, defecto)
    añadir-rama(nodo, rama)
  return nodo

```

Figura 1. Esquema general de los algoritmos TDIDT

una hoja con dicha clase), cuando no quedan atributos (se genera una hoja con la clase mayoritaria) o cuando no existen ejemplos para clasificar (se genera una hoja con un valor por defecto). El caso general consiste en escoger el mejor atributo, atendiendo a una cierta función de evaluación, y generar un nodo interno con dicho atributo. Para generar las ramas de este nodo se aplica de nuevo el algoritmo al subconjunto de ejemplos de cada valor del atributo seleccionado. El algoritmo ID3 utiliza la ganancia de información para seleccionar el atributo:

$$Ganancia(A, S) = Entropía(S) - Entropía(A, S) \quad (1)$$

$$Entropía(S) = - \sum_{c \in Clase} p_c \cdot \log_2 p_c \quad (2)$$

$$Entropía(A, S) = \sum_{a \in A} p_a \cdot Entropía(a, A, S_a) \quad (3)$$

$$Entropía(a, A, S_a) = - \sum_{c \in Clase} p_{ac} \cdot \log_2 p_{ac} \quad (4)$$

$$p_c = |S_c|/|S| \quad (5)$$

$$p_a = |S_a|/|S| \quad (6)$$

$$p_{ac} = |S_{ac}|/|S_a| \quad (7)$$

Donde S es el conjunto de ejemplos a estudiar, A es el atributo cuya ganancia de información se va a calcular, a se refiere a cada uno de los valores del atributo A , $Clase$ es el atributo de clasificación, c se refiere a cada uno de los valores del atributo $Clase$, S_c es el subconjunto de S formado por los ejemplos de la clase c , S_a es el subconjunto de S formado por los ejemplos en los que el atributo A toma el valor a , S_{ac} es el subconjunto de S formado por los ejemplos en los que el atributo A toma el

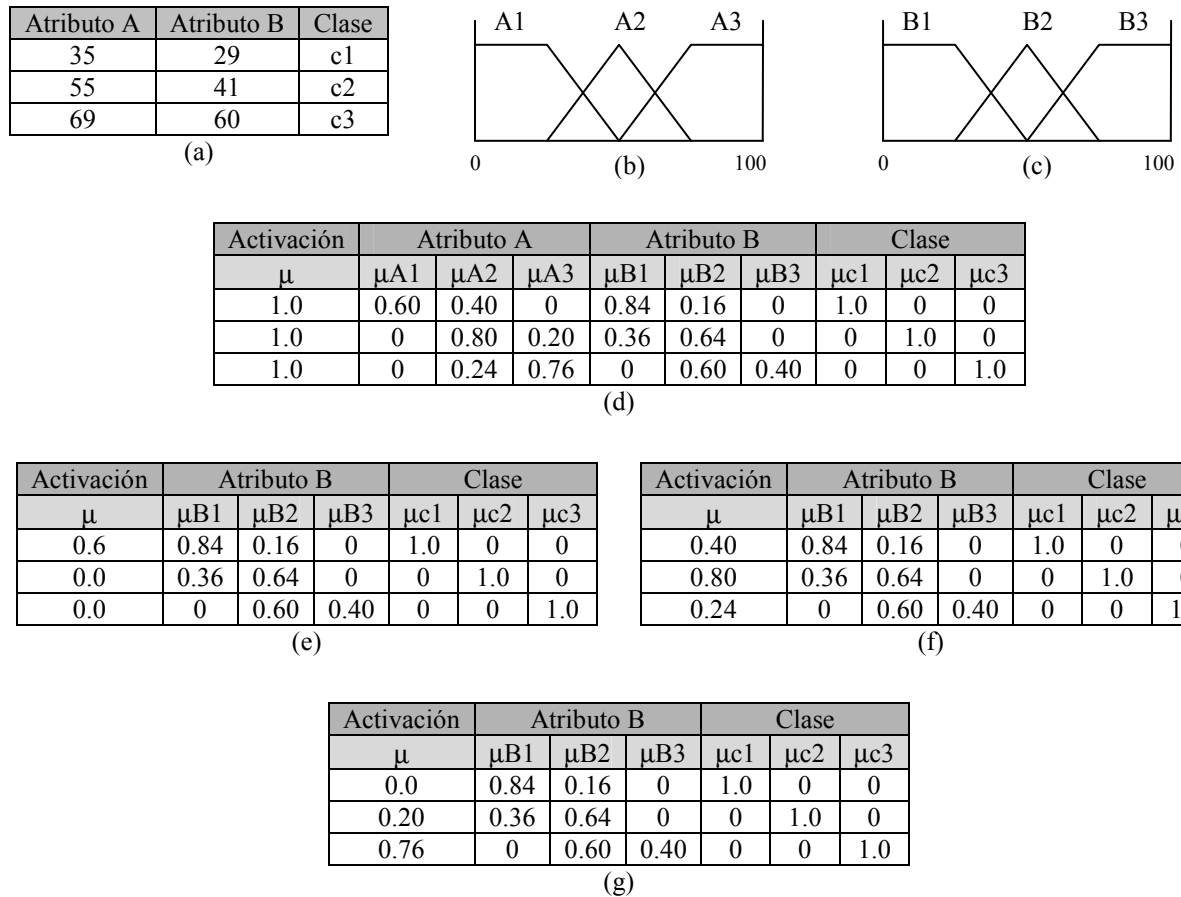


Figura 2: (a) Conjunto de datos original; (b,c) Funciones de pertenencia de los atributos A y B; (d) Conjunto de datos difusos; (e,f,g) Subconjuntos de datos difusos para los valores A1, A2 y A3 del atributo A.

valor a y el atributo *Clase* toma el valor c , y $|\cdot|$ indica el número de elementos de un conjunto.

Una de las limitaciones más serias del algoritmo ID3 es que sólo está definido para atributos nominales. Cuando los atributos están definidos sobre dominios continuos es necesario discretizarlos definiendo intervalos en dichos dominios. La evolución del algoritmo ID3, denominada C4.5, discretiza cada atributo continuo A en los intervalos $(A \leq t)$ y $(A > t)$ siendo t el umbral que conduce al mejor valor de cierto criterio de división [13].

El algoritmo FuzzyID3 es una extensión del algoritmo ID3 orientado a problemas de clasificación que utilicen atributos continuos. El algoritmo se basa en describir los valores nominales de los atributos continuos por medio de conjuntos difusos en lugar de utilizar intervalos. Los conjuntos de datos a clasificar se convierten entonces en conjuntos de datos difusos (Figura 2) que contienen para cada instancia el grado de pertenencia a cada etiqueta de cada parámetro.

El resultado de FuzzyID3 es un árbol de decisión difuso. El funcionamiento de un árbol de decisión difuso es el siguiente. Cada nodo interno representa una consulta respecto al valor de un atributo. Las ramas se refieren a las distintas etiquetas lingüísticas del atributo, de manera que cada rama tiene un cierto grado de activación. Ante unos determinados valores de entrada, los grados de activación se van propagando por las ramas del árbol hasta determinar el grado de activación de cada hoja. Mientras que en un árbol de decisión no difuso sólo existe una hoja activa en cada caso, en un árbol de decisión difuso aparecen muchas hojas activas. Para obtener una salida concreta del clasificador difuso se toma la clase de la hoja más activa o se calcula el grado de activación agregado de cada clase y se toma la clase más activa.

Para transformar el algoritmo ID3 en un algoritmo difuso es necesario definir la ganancia de información difusa y la forma en la que se divide el conjunto de datos entre los valores de un atributo. Para definir la ganancia de información sobre un conjunto de datos difuso es necesario rescribir la forma en la que se calculan las diferentes proporciones con datos difusos:

$$p_c = \sum_e \mu \otimes \mu_c / \sum_{\theta \in Clase} \sum_e \mu \otimes \mu_\theta \quad (8)$$

$$p_a = \sum_e \mu \otimes \mu_a / \sum_{\alpha \in A} \sum_e \mu \otimes \mu_\alpha \quad (9)$$

$$p_{ac} = \sum_e \mu \otimes \mu_a \otimes \mu_c / \sum_{\theta \in Clase} \sum_e \mu \otimes \mu_a \otimes \mu_\theta \quad (10)$$

Donde e se refiere a cada uno de los ejemplos del conjunto de datos, μ es el grado de activación de cada ejemplo, μ_c es el grado de pertenencia del ejemplo a la clase c , μ_θ es el grado de pertenencia del ejemplo a la clase θ , μ_a es el grado de pertenencia del ejemplo a la etiqueta a del atributo A , μ_α es el grado de pertenencia del ejemplo a la etiqueta α del atributo A , y el operador \otimes se refiere a la T-norma escogida para la conjunción.

Al dividir un conjunto de datos entre los valores de un atributo, cada elemento e del conjunto de datos es incluido en el subconjunto del valor a con el siguiente grado de activación:

$$\mu_{ea} = \mu \otimes \mu_a \quad (11)$$

Esta forma de dividir los conjuntos de datos tiende a generar subconjuntos con numerosas instancias con grado de activación pequeño. Para evitar esto se introduce un cierto umbral de activación por debajo del cual las instancias son eliminadas de los conjuntos de datos.

3 EL ALGORITMO FUZZCONRI

El algoritmo CN2 fue propuesto como forma de mejorar los problemas de los algoritmos AQ frente al ruido. El algoritmo realiza una búsqueda en estrella en el espacio de reglas conjuntivas considerando como potenciales selectores a todos los posibles valores de cada atributo. CN2 utiliza dos heurísticas para seleccionar las reglas. La primera es un test de significancia estadística que elimina aquellas reglas que no cubren un número suficiente de ejemplos. La segunda heurística es un test de bondad que selecciona a las reglas con mayor proporción de acierto. Inicialmente el algoritmo generaba listas de reglas ordenadas y utilizaba como heurísticas el factor de probabilidad (*likelihood ratio*) y la entropía de información. Posteriormente se propuso modificaciones al algoritmo para generar listas de reglas desordenadas y la entropía de información se sustituyó por el error laplaciano, con una considerable mejora del comportamiento del algoritmo. La Figura 3 muestra el esquema general de la versión mejorada de CN2.

El algoritmo tiene dos parámetros de control: el umbral de significancia y el tamaño máximo de la estrella, es decir, el número máximo de candidatos que se almacenan en la búsqueda.

```

function CN2(ejemplos,clases)
returns lista_de_reglas
lista_de_reglas  $\leftarrow \emptyset$ 
foreach c  $\in$  clases
    P  $\leftarrow$  Positivos(ejemplos,c)
    N  $\leftarrow$  Negativos(ejemplos,c)
    reglas  $\leftarrow$  CN2ParaUnaClase(P,N,c)
    lista_de_reglas  $\leftarrow$  lista_de_reglas  $\cup$  reglas
return lista_de_reglas

function CN2ParaUnaClase(P,N,c)
returns lista_de_reglas
lista_de_reglas  $\leftarrow \emptyset$ 
regla  $\leftarrow$  EncontrarMejorRegla(P,N,c)
while regla  $\neq$  null
    lista_de_reglas  $\leftarrow$  lista_de_reglas  $\cup$  regla
    P  $\leftarrow$  EliminarCubiertos(P,regla)
    regla  $\leftarrow$  EncontrarMejorRegla(P,N,c)
return lista_de_reglas

function EncontrarMejorRegla(P,N,c)
returns regla
mgr  $\leftarrow$  "if(true) then c"
star  $\leftarrow \{ mgr \}$ 
mejor  $\leftarrow$  null
while star  $\neq \emptyset$ 
    newstar  $\leftarrow \emptyset$ 
    foreach regla  $\in$  star, sel  $\in$  selectores
        newr  $\leftarrow$  AñadirSelector(regla,sel)
        if (Significancia(P,N,newr) > umbral
            & Bondad(P,N,newr) > Bondad(P,N,mejor))
            then
                mejor  $\leftarrow$  newr
                newstar  $\leftarrow$  newstar  $\cup$  newr
        if (Tamaño(newstar) > maximo)
            then newstar  $\leftarrow$  EliminarPeor(newstar)
    star  $\leftarrow$  newstar
return mejor

```

Figura 3. Esquema general del algoritmo CN2.

El algoritmo FuzzConRI es una adaptación de CN2 para tratar conjuntos de datos difusos. El algoritmo introduce un corte- α para discriminar entre los ejemplos positivos y negativos de cada clase, de manera que para cada clase c todos los ejemplos cuyo μ_c sea menor que el corte- α se consideran negativos. Este corte- α se utiliza también para eliminar los elementos cubiertos por una regla, que serían aquellos para los que dicha regla alcanza un grado de activación superior a dicho corte- α . FuzzConRI requiere también adaptar las heurísticas de significancia y bondad de las reglas generadas. Los autores consideran la significancia estadística como una heurística opcional y en sus trabajos utilizan para esta función el porcentaje de

ejemplos positivos cubiertos por la regla. Por su parte, para evaluar la bondad de las reglas proponen utilizar la función de precisión difusa (*fuzzy accuracy*) que viene dada por la siguiente expresión.

$$A(r) = \frac{\sum_{p \in P} \mu_r(p) - \sum_{n \in N} \mu_r(n)}{\sum_{p \in P} \mu_r(p) + \sum_{n \in N} \mu_r(n)} \quad (12)$$

Donde $\mu_r(e)$ representa el grado de activación de la regla r para la instancia e .

4 LA IMPLEMENTACIÓN EN XFUZZY

Los algoritmos FuzzyID3 y FuzzConRI se han añadido al entorno Xfuzzy como parte de los algoritmos integrados en la herramienta Xfdm. La Figura 4 muestra la ventana principal de esta herramienta. Desde esta ventana es posible seleccionar el algoritmo a utilizar en el proceso de inducción de reglas, el conjunto de datos a utilizar, la forma de las variables de entrada de la base de reglas a extraer y el estilo del sistema a generar.

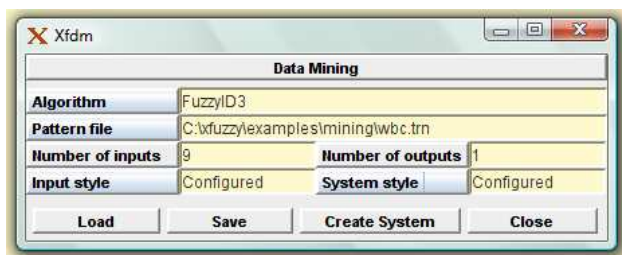


Figura 4: Ventana principal de Xfdm.

Cada algoritmo de extracción de reglas difusas dispone de una ventana de configuración en la que se introducen los valores de los parámetros de control del algoritmo. La Figura 5 muestra la ventana de selección del algoritmo FuzzyID3 que contiene como único parámetro de configuración el umbral de activación mínimo de las instancias a considerar por el algoritmo.



Figura 5: Configuración del algoritmo FuzzyID3.

Por su parte, el algoritmo FuzzConRI tiene como parámetros de control el tamaño máximo de la estrella (conjunto de reglas que se mantienen en memoria), el corte- α (que se utiliza para construir N y P y para eliminar las instancias cubiertas por una regla) y el

umbral de significancia. La Figura 6 muestra la ventana de configuración de este algoritmo.

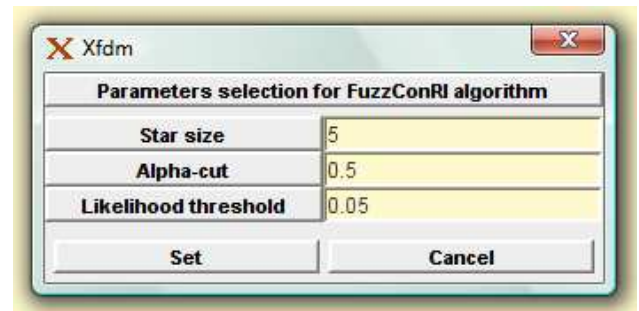


Figura 6: Configuración del algoritmo FuzzConRI.

La aplicación de estos algoritmos requiere definir las funciones de pertenencia de las variables de entrada de las reglas. La Figura 7 muestra la ventana de configuración de las variables de entrada. En esta ventana se puede seleccionar el número de funciones de pertenencia de las entradas, la forma de estas funciones o el universo de discurso de las variables.

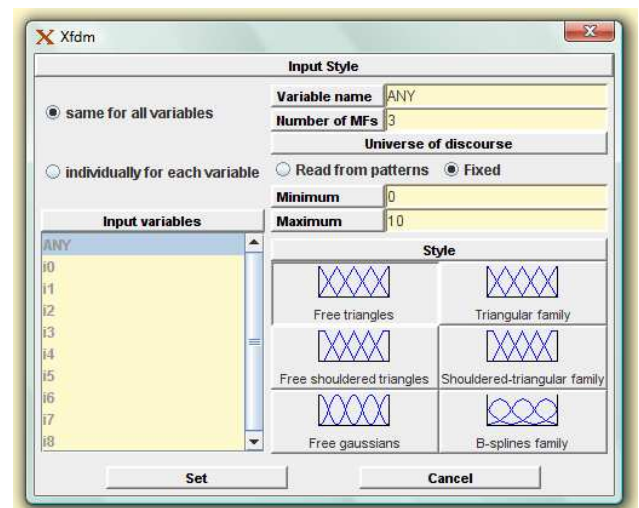


Figura 7: Configuración de las variables de entrada.

Para finalizar el proceso de configuración de los algoritmos es necesario definir el tipo de base de reglas que se va a generar (de clasificación en nuestro caso) y la T-norma a utilizar como conjunción. La Figura 8 muestra la ventana de Xfdm dedicada a seleccionar estos aspectos.

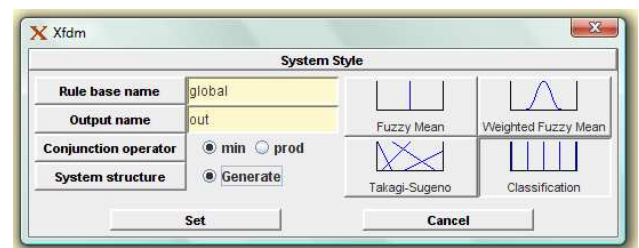


Figura 8: Configuración del estilo del sistema.

5 ALGUNOS EJEMPLOS

A continuación se presentan los resultados de un estudio preliminar sobre el comportamiento de estos dos algoritmos. Para desarrollar este estudio se han utilizado varias bases de datos de clasificación con atributos de entrada numéricos. Todas estas bases de datos han sido extraídas del repositorio UCI. En el estudio se han considerado los resultados de los algoritmos sobre las bases de datos completas. Para obtener conclusiones más firmes sería necesario realizar un estudio más profundo basado en evaluación por validación cruzada.

En primer lugar se ha realizado un estudio sobre el efecto de los diferentes parámetros de control de los algoritmos FuzzyID3 y FuzzConRI. Para ello se han utilizado dos bases de datos conocidas: Pima, que contiene 768 patrones de clasificación entre pacientes de diabetes de esta tribu indígena en base a 8 atributos numéricos, y Winsconsin Breast Cancer, que contiene 683 patrones de clasificación entre tumores malignos y benignos en base a 9 atributos numéricos. En este estudio se ha representado cada atributo de entrada por medio de 3 etiquetas como las mostradas en Fig. 2.b y se ha escogido el mínimo como T-norma.

La Tabla 1 muestra los resultados obtenidos por el algoritmo FuzzyID3 sobre estas bases de datos para distintos valores del umbral de activación. Este parámetro se utiliza para eliminar de los subconjuntos de datos aquellos ejemplos cuyo grado de activación no supere este umbral. Un valor pequeño del umbral provoca que la mayoría de los ejemplos no sean eliminados y que un mismo ejemplo se utilice en los subconjuntos de datos de diferentes ramas del árbol de decisión. Por el contrario, un valor alto del umbral provoca que muchos ejemplos sean eliminados de los subconjuntos y que algunos de ellos sean eliminados de todos los subconjuntos de datos de las diferentes ramas del árbol. En la tabla se observa que los mejores resultados se obtienen en el entorno de un umbral 0.5. Con respecto al número de reglas se observa que al aumentar el umbral de activación el número de reglas generadas disminuye. Esto se debe a que un umbral mayor conduce a subconjuntos de datos más pequeños que generan árboles de decisión menos profundos.

Tabla 1. Resultados de FuzzyID3

Umbral	Pima		WBC	
	# Reglas	%Error	# Reglas	%Error
0.0	1649	25.78	961	2.20
0.2	901	24.09	239	1.02
0.4	423	21.61	105	0.34
0.5	299	19.27	105	0.34
0.6	193	26.95	41	2.63
0.8	55	36.20	17	5.42

La Tabla 2 muestra los resultados que se obtienen con el algoritmo FuzzConRI para distintos valores de sus parámetros de configuración. Con respecto al tamaño de la estrella la tabla muestra que su influencia sobre el resultado del algoritmo es pequeña. Un tamaño de la estrella inferior a 5 puede provocar un aumento de la tasa de error, mientras que valores más altos en el tamaño de la estrella no consiguen mejorar los resultados. Por su parte, los valores superiores a 0.5 en el corte- α producen un aumento considerable en el número de reglas generadas y en la tasa de error. Al considerar valores altos de este parámetro la mayoría de los ejemplos se consideran no cubiertos por las reglas generadas lo que obliga a seguir buscando reglas. Un valor en el corte- α inferior a 0.5 provoca que se eliminen muchos ejemplos del conjunto P y que el número de reglas generadas sea menor, aunque también produce una tasa de error más alta al perder ejemplos que aportan información al conjunto de datos. Por último, los valores altos del umbral de significancia (10%) provocan una disminución del número de reglas a costa de un leve incremento en la tasa de error. La razón es que a las reglas generadas por el algoritmo se les exige un grado de significancia superior al umbral de manera que si este umbral es alto son pocas las reglas que consiguen superarlo.

Tabla 2. Resultados de FuzzConRI

Star	Corte- α	Sign.	Pima		WBC	
			# R.	%Err	# R.	%Err
5	0.5	0.10	34	27.99	28	7.47
5	0.5	0.05	42	27.86	34	7.47
5	0.5	0.01	59	27.86	34	7.47
5	0.3	0.05	46	28.51	24	7.47
5	0.5	0.05	42	27.86	34	7.47
5	0.7	0.05	72	31.38	40	50.95
3	0.5	0.05	42	28.25	34	7.47
5	0.5	0.05	42	27.86	34	7.47
10	0.5	0.05	43	27.86	33	7.47

La Tabla 3 contiene los resultados obtenidos por los algoritmos FuzzyID3, FuzzConRI y Wang&Mendel sobre 7 bases de datos de atributos numéricos extraídas del repositorio UCI. En todos los casos se ha considerado que cada atributo se describe por medio de 3 etiquetas como las mostradas en la Figura 2.b y se ha escogido el mínimo como T-norma. Para la aplicación del algoritmo FuzzyID3 se ha utilizado un umbral de activación de 0.5. Las pruebas realizadas con el algoritmo FuzzConRI utilizan una estrella de tamaño 5, un valor de 0.5 en el corte- α y un umbral de significancia de 0.05.

Comparado los resultados del algoritmo FuzzyID3 con los obtenidos por el algoritmo de Wang y Mendel se observa

Tabla 3. Resultados sobre varias bases de datos

Base de datos	Atributos	Instancias	FuzzyID3		FuzzConRI		Wang & Mendel	
			#Reglas	%Error	#Reglas	%Error	#Reglas	%Error
Glass	9	214	173	23.8%	51	28.0%	68	29.9%
Heart-statlog	13	270	199	4.1%	46	12.6%	234	4.1%
Pima	8	768	299	19.3%	42	27.9%	173	23.81
Iris	4	150	21	4.7%	8	2.7%	22	4.7%
WBC	9	683	105	0.34%	34	7.5%	254	1.0%
Sonar	60	208	91	0.0%	37	14.9%	207	0.0%
Vehicle	18	946	1844	15.0%	174	29.1%	496	28.4%

una mejora apreciable de la precisión (una tasa de error menor). Con respecto al número de reglas, en muchos casos se observa que el número de reglas generado por FuzzyID3 es mayor que el que obtiene el algoritmo de Wang y Mendel. Esto se debe a que FuzzyID3 genera reglas que cubren todo el espacio, mientras que el algoritmo de Wang y Mendel genera reglas que solo cubren los ejemplos del conjunto de datos. En estos casos, un proceso de podado de reglas que no se activen significativamente para ningún ejemplo suele reducir apreciablemente el número de reglas del sistema generado por FuzzyID3 (por ejemplo, para la base de datos Glass el número de reglas se reduce a 52, mientras que para la base de datos Vehicle se reduce a 323).

Con respecto al algoritmo FuzzConRI, como norma general se observa que el número de reglas generadas es bastante menor que el de otros algoritmos, a costa de una tasa de error más alta. Esto se debe a la elección de la precisión difusa como medida de bondad, ya que esta medida premia a las reglas con mayor número de ejemplos positivos cubiertos, incluso aunque el número de errores de clasificación sea alto.

6 CONCLUSIONES

Los algoritmos de inducción de reglas difusas se dividen en dos grupos: los que generan etiquetas difusas asociadas a cada regla (*clustering*) y los que utilizan etiquetas definidas previamente (*grid*). Estos últimos son mejores en cuanto a interpretabilidad de las etiquetas, pero suelen generar un número de reglas demasiado alto. La adaptación de algunos algoritmos de clasificación clásicos al mundo difuso nos permite obtener clasificadores difusos con reglas más compactas lo que mejora significativamente la interpretabilidad. Dos de estos algoritmos (FuzzyID3 y FuzzConRI) han sido incorporados al entorno de desarrollo Xfuzzy, lo que permite integrar estos algoritmos al ciclo de diseño desarrollado por este entorno.

Agradecimientos

Este trabajo ha sido parcialmente financiado gracias al proyecto de investigación TEC2005-04359/MIC del Ministerio de Educación y Ciencias y a los proyectos de investigación TIC2006-635, TEP2006-375 y P07-TIC-03179 de la Junta de Andalucía.

Referencias

- [1] I. Baturone, F.J. Moreno Velo, A. Gersnoviez. Identifying fuzzy systems from numerical data with Xfuzzy. *Proc. 4th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2005)*, Pág 1257-1262, 2005.
- [2] I. Baturone, F.J. Moreno-Velo, S. Sanchez-Solano, A. Barriga, P. Brox, A. Gersnoviez, M. Brox. Using Xfuzzy Environment for the Whole Design of Fuzzy Systems. *Proc. IEEE International Conference on Fuzzy Systems*, Pág 1-6, 2007.
- [3] J.C. Bezdek, Pattern recognition with fuzzy objective function algorithms, *Plenum Press*, 1981.
- [4] S. L. Chiu. A Cluster Estimation Method with Extension to Fuzzy Model Identification. *Proc. 3rd IEEE Conference on Fuzzy Systems*, Pág. 1240-1245, 1994.
- [5] K.J. Cios, L.M. Sztandera. Continuous ID3 algorithm with fuzzy entropy measures. *Proc. IEEE Int. Conf. on Fuzzy Systems*, Pág. 469-476, 1992.
- [6] P. Clark, T. Niblett. The CN2 Induction Algorithm. *Machine Learning*, vol. 3, Pág. 261-283, 1989.
- [7] P. Clark, R. Boswell. Rule induction with CN2: some recent improvements. *Proc. Fifth European Working Session on Learning (EWSL-91)*, Pág. 151-163, 1991.
- [8] I. Gath, A. B. Geva. Unsupervised Optimal Fuzzy Clustering. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 11, Pág. 773- 781, 1989.

- [9] D. E. Gustafson, W. C. Kessel. Fuzzy Clustering with a Covariance Matrix. *Proc. IEEE Conference on Decision and Control*, Pág. 761-766, 1978.
- [10] R.S. Michalski, I. Mozetic, J. Hong, N. Lavrac. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. *Proc. 5th National Conference on Artificial Intelligence (AAAI86)*, Pág. 1041-1047, 1986.
- [11] D. Nauck, R. Kruse. NEFCLASS – A Neuro-Fuzzy Approach for the Classification of Data. *Proc. 1995 ACM Symposium on Applied Computing*, Pág. 461-465, 1995.
- [12] J.R. Quinlan. Induction of decision trees. *Machine Learning*, vol. 1, Pág. 81-106, 1986.
- [13] J. R. Quinlan. Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, vol. 4, Pág. 77-90, 1996.
- [14] R. Senhadji, S. Sánchez-solano, A. Barriga, I. Baturone, F.J. Moreno-Velo. NORFREA: An algorithm for non-redundant fuzzy rule extraction. *Proc. IEEE International Conference on Systems, Man and Cybernetics*, Pág. 604-608, 2002.
- [15] T. Tani, M. Sakoda, K. Tanaka, Fuzzy Modeling by ID3 algorithm and its application to prediction, *Proc IEEE International Conference on Fuzzy Systems*, Pág. 923-930, 1992.
- [16] M. Umamo, H. Okamoto, I. Hatono, H. Tamura, F. Kawachi, S. Umedzu, J. Kinoshita. Fuzzy Decisión Trees by Fuzzy ID3 Algorithm and Its Application to Diagnosis Systems. *Proc IEEE International Conference on Fuzzy Systems*, Pág. 2113-2118, 1994.
- [17] J. van Zyl, I. Cloete. An inductive algorithm for learning conjunctive fuzzy rules. *Proc. 3rd IEEE International Conference on Machine Learning and Cybernetics*, Pág. 4181-4187, 2004.
- [18] L.Wang, J.M. Mendel. Generation of Rules by Learning from Examples. *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 22, Pág. 1414-1427, 1992.
- [19] R. Weber. Fuzzy ID3: a class of methods for automatic knowledge acquisition. *Proc. 2nd Int. Conf. on Fuzzy Logic and Neural Networks*, Pág. 265-268, 1992.