FUZZYCN2: UN ALGORITMO DE EXTRACCIÓN DE LISTAS DE REGLAS DIFUSAS

Pablo Martín Muñoz[1] **Francisco José Moreno Velo**[2]

[1]Unidad para la Dirección Estratégica, Universidad de Huelva, pablo.martin@sc.uhu.es
[2]Departamento de Tecnologías de la Información, Universidad de Huelva, francisco.moreno@dti.uhu.es

Resumen

La mayoría de los algoritmos de extracción de reglas de clasificación difusas producen reglas con antecedentes conjuntivos en los que aparecen todos los atributos del sistema. Con este tipo de antecedentes, el número de reglas crece exponencialmente en función del número de atributos. Este trabajo presenta un nuevo algoritmo de extracción de reglas de clasificación difusas llamado FuzzyCN2. Se trata de una versión difusa del conocido algoritmo CN2 y genera una lista ordenada de reglas. El algoritmo produce reglas donde no todos los atributos están presentes y permite, además, utilizar modificadores lingüísticos en estas reglas. Esto permite reducir el número de reglas con respecto a los resultados de los algoritmos convencionales.

Palabras Clave: Clasificadores Difusos, Extracción de Reglas Difusas, Listas Ordenadas de Reglas Difusas, Modificadores Lingüísticos.

1 INTRODUCCIÓN

Los sistemas basados en reglas difusas han demostrado ser una opción muy adecuada en problemas de clasificación. Por un lado y frente a alternativas como las redes neuronales, las máquinas de vectores soporte o los vecinos más cercanos, el uso de reglas difusas permite la inducción de modelos interpretables. Por otro lado y frente a los sistemas expertos clásicos, la salida de un clasificador difuso no se limita a la clase seleccionada, sino que ofrece el grado de activación de cada clase. Esto resulta especialmente útil en problemas con atributos continuos. En estos casos los sistemas expertos clásicos se basan en consultas del tipo *atributo* $\geq$ *valor* [9], por lo que estos sistemas presentan cambios abruptos alrededor de estos valores. Por su parte, el comportamiento de los clasificadores difusos presenta cambios suaves en el grado de activación de cada clase, lo que permite detectar situaciones en las que la decisión es más arriesgada al involucrar a clases con grados de activación muy próximos.

El algoritmo de extracción de reglas difusas más conocido es el propuesto por Wang y Mendel [13]. Este algoritmo se basa, en primer lugar, en definir las etiquetas lingüísticas asociadas a cada atributo y, a continuación, generar la regla difusa conjuntiva más activa para cada uno de las instancias del conjunto de datos. Cada una de las reglas generadas en este algoritmo contiene un término para cada atributo del sistema. Esto provoca un crecimiento exponencial en el número de reglas generadas en función del número de atributos del sistema. Existen muchas propuestas para reducir el número de reglas generadas, pero la mayoría mantiene la forma de las reglas, es decir, utilizan reglas conjuntivas que contienen a todos los atributos (por ejemplo [7][10]).

Este enfoque contrasta con el seguido habitualmente en los algoritmos clásicos de extracción de reglas. El objetivo de estos algoritmos es minimizar el número de reglas generadas. Para ello tratan de inducir las reglas más generales posible, esto es, reglas en las que no aparezcan todos los atributos. Entre estos algoritmos los más conocidos son ID3 [8], que genera árboles de decisión, y la familia de algoritmos AQ [5], que genera conjuntos de reglas de clasificación. Tomando ideas de estos algoritmos, Clark y Niblett propusieron en 1989 el algoritmo CN2 [4], que induce listas ordenadas de reglas de clasificación. Posteriormente Clark y Boswell plantearon algunas mejoras a este algoritmo [3].

Algunas propuestas para inducir reglas de clasificación difusas se basan en versiones difusas de los algoritmos anteriores. Existen numerosas adaptaciones del algoritmo ID3 para generar árboles de decisión difusos [2][11][12][14]. El algoritmo CN2 también ha sido adaptado al caso difuso [16], pero en esta caso se trata de la versión del algoritmo que induce conjuntos desordenados de reglas.

Este trabajo presenta una adaptación diferente del algoritmo CN2 para extraer listas ordenadas de reglas difusas.

La propuesta contiene no sólo un modo de adaptar los diferentes procedimientos y métricas de CN2 al caso difuso, sino que también añade la capacidad de utilizar modificadores lingüísticos como parte de las reglas inducidas, lo que permite extraer reglas más compactas y expresivas que las inducidas en la mayoría de algoritmos de clasificación difusos.

2 EL ALGORITMO CN2

El algoritmo CN2 fue propuesto sobre la base de los algoritmos AQ tratando de introducir la capacidad de tratamiento de ruido de los algoritmos TDIDT (Top Down Induction of Decision Trees). El algoritmo CN2 toma de los algoritmos AQ la idea de encontrar el mejor conjunto de reglas por medio de un conjunto de búsquedas en haz en paralelo (lo que se conoce como una búsqueda en estrella) y de los algoritmos TDIDT la idea de finalizar la búsqueda cuando las reglas encontradas no superan cierto umbral de significancia estadística (técnicas de poda).

El algoritmo se basa en un bucle externo en el que, dado un conjunto de ejemplos de clasificación, se encuentra la mejor regla para dicho conjunto y se eliminan los ejemplos cubiertos por dicha regla. El bucle finaliza cuando el conjunto de ejemplos queda vacío o cuando no se encuentra ninguna regla con el mínimo nivel de significancia exigido. Puesto que las reglas se generan sobre conjuntos de ejemplos en los que se han eliminado aquellos que han sido cubierto por reglas anteriores, el resultado debe interpretarse como una lista ordenada de reglas, es decir que una regla sólo debe ser considerada si ninguna de las reglas precedentes se encuentra activa.

La búsqueda de la mejor regla asociada a un conjunto de ejemplos se basa en una búsqueda en estrella. En cada iteración las reglas candidatas son especializadas añadiéndoles un nuevo selector (una consulta que compara un atributo con uno de sus valores). En los algoritmos AQ, los términos a añadir se obtienen a partir de uno de los ejemplos positivos del conjunto (*semilla*). El algoritmo CN2 amplía este conjunto a todos los posibles selectores. Para estudiar la bondad de las reglas el algoritmo utiliza dos heurísticas. Por un lado, la entropía de información se utiliza para medir la calidad de las reglas candidatas. Por otro lado, a las reglas se les exige un mínimo de significancia estadística que se mide por medio de una heurística conocida como *likelihood ratio*. En un trabajo posterior, presentado en 1991, se propuso la utilización de la estimación del error laplaciano como heurística de calidad, en sustitución de la entropía de información. Este trabajo contiene también una versión del algoritmo que genera listas de reglas desordenadas.

El algoritmo FuzzyCN2 presentado en este trabajo se basa en una versión difusa del algoritmo CN2 original, que genera listas de reglas ordenadas. La heurística utilizada como medida de calidad es una versión difusa de la estimación del error laplaciano y la medida de la significancia estadística se basa en un valor mínimo de la cobertura de las reglas sobre el conjunto de ejemplos.

3 LISTAS ORDENADAS DE REGLAS DIFUSAS

El algoritmo FuzzyCN2 genera una lista ordenada de reglas difusas en lugar de un conjunto desordenado de reglas. Ambas representaciones tienen sus respectivas ventajas y desventajas en cuanto a interpretabilidad. En una lista ordenada de reglas la interpretación de cada regla depende de cuales son las reglas que le preceden, ya que dicha regla sólo se activa en la medida en que las predecesoras no lo hacen. Por su parte, los conjuntos de reglas desordenadas pueden contener reglas que cubren el mismo espacio de entrada, es decir, reglas que se activan al mismo tiempo. En este caso es necesario algún tipo de mecanismo adicional que decide cual de las reglas se impone cuando surgen estos conflictos, de manera que las reglas no pueden considerarse estrictamente independientes de las demás si permiten la existencia de estos conflictos.

La tabla 1 muestra estos dos tipos de listas de reglas difusas. El mecanismo de inferencia en una lista de reglas desordenadas considera el grado de activación del consecuente de cada regla, $\mu(C_i)$, como el grado de activación de su antecedente, $\mu(A_i)$. En el caso de las listas ordenadas de reglas el mecanismo de inferencia es diferente. En este caso hay una condición *else* implícita respecto a las reglas precedentes por lo que el grado de activación del consecuente de la regla, $\mu(C_i)$, depende tanto del grado de activación de su antecedente, $\mu(A_i)$, como del grado de activación de sus predecesoras, $\Pi(1-\mu(A_i))$.

Tabla 1: Diferentes tipos de conjuntos de reglas

Conjunto	Lista ordenada

if(A1) then C1
if(A2) then C2
if(A3) then C3
	if(A1) then C1
	elseif(A2) then C2
	elseif(A3) then C3

μ(C1) = μ(A1) μ(C2) = μ(A2) μ(C3) = μ(A3)	μ(C1) = μ(A1) μ(C2) = (1-μ(A1))·μ(A2) μ(C3) = (1-μ(A1))·(1-μ(A2))·μ(A3)

4 EL ALGORITMO FUZZYCN2

El algoritmo FuzzyCN2 consiste en una versión difusa del algoritmo CN2 clásico. Para realizar esta traducción es necesario resolver algunas cuestiones:

a) ¿Cómo se eliminan los ejemplos cubiertos por una regla difusa? Puesto que las reglas difusas sólo cubren los ejemplos en un cierto grado, estos ejemplos deberían ser eliminados sólo parcialmente.

b) ¿Cómo se representan conjuntos de ejemplos que han sido eliminados parcialmente?

c) ¿Cómo se calcula el grado de cubrimiento de una regla difusa sobre un ejemplo eliminado parcialmente?

d) ¿Cómo se traducen las heurísticas de búsqueda (estimación del error laplaciano y test de significancia estadística) al caso difuso?

Las siguientes definiciones permiten resolver estas cuestiones:

Definición 1. Un selector difuso es un término difuso que relaciona un atributo y una de sus etiquetas lingüísticas. Esta relación puede ser la igualdad, es decir, *atributo igual a etiqueta*, o puede utilizar modificadores lingüísticos, como *atributo mayor que etiqueta*. La Figura 1 muestra los diferentes selectores considerados en FuzzyCN2.

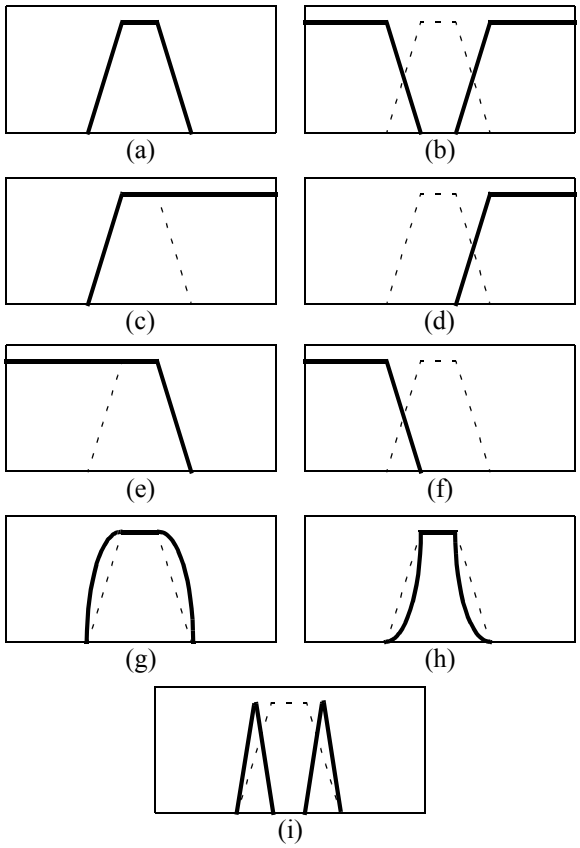


Figura 1: Selectores utilizados en FuzzyCN2 por cada atributo (*attr*) y etiqueta (*lb*). (a) *attr es igual a lb*; (b) *attr es distinto de lb*; (c) *attr es mayor o igual que lb*; (d) *attr es mayor que lb*; (e) *attr es menor o igual que lb*; (f) *attr es menor que lb*; (g) *attr es aproximadamente igual a lb*; (h) *attr es fuertemente igual a lb*; (i) *attr es casi igual a lb*.

Definición 2. Un complejo es una conjunción de selectores difusos. Se utilizan como antecedentes en las reglas inducidas por CN2.

Definición 3. Un conjunto difuso de instancias es un conjunto de ejemplos de clasificación en el que cada ejemplo tiene asociado un grado de activación. Un conjunto concreto (*crisp*) de instancias puede interpretarse como un conjunto difuso de instancias en el que el grado de activación de todos los ejemplos es la unidad. Los conjuntos difusos de instancias pueden ser truncados, eliminando aquellos ejemplos cuyo grado de activación no alcance un cierto nivel de corte.

Definición 4. Dado un complejo C y un ejemplo e con grado de activación $\mu(e)$, el soporte de C sobre e se define como el producto entre el grado de activación del complejo al aplicarse sobre el ejemplo y el grado de activación del ejemplo, es decir,

$$soporte(C, e) = \mu(C, e) \cdot \mu(e) \qquad (1)$$

Definición 5. El soporte de un complejo C sobre un conjunto difuso de instancias S se define como la suma del soporte de C sobre cada uno de los ejemplos de S.

$$soporte(C, S) = \sum_{e \in S} soporte(C, e) \qquad (2)$$

Definición 6. El soporte de un complejo C sobre una clase c y un conjunto difuso de instancias S se define como la suma del soporte de C sobre aquellos ejemplos de S que pertenecen a la clase c.

$$soporte(C, S, c) = \sum_{e \in S,\, clase(e) = c} soporte(C, e) \qquad (3)$$

Definición 7. El soporte relativo de un complejo C sobre un conjunto difuso de ejemplos S se define como

$$soporte\text{-}relativo(C, S) = \frac{soporte(C, S)}{\sum_{e \in S} \mu(e)} \qquad (4)$$

Definición 8. La desactivación de un conjunto difuso de ejemplos S con respecto a un complejo C consiste en actualizar el grado de activación de cada ejemplo e de la siguiente forma:

$$\mu_{desactivado}(e) = \mu(e) \cdot (1 - \mu(C, e)) \qquad (5)$$

Definición 9. Se dice que un complejo C es significativo con respecto a un conjunto difuso de ejemplos S si el soporte relativo de C sobre S es mayor que un cierto valor

(*soporte relativo mínimo*) y el soporte de C sobre S es mayor que un cierto valor (*soporte absoluto mínimo*).

Definición 10. La estimación difusa del error laplaciano de un complejo C es una medida de la calidad de dicho complejo definida como

$$Laplaciano(C, S) = \frac{soporte(C, S, c) + 1}{soporte(C, S) + k} \quad (6)$$

donde c es la clase de mayor soporte y k es el número de clases.

La siguiente tabla muestra el pseudocódigo del algoritmo FuzzyCN2. El algoritmo toma como entrada un conjunto de ejemplos de clasificación y genera una lista ordenada de reglas difusas.

Tabla 2: El algoritmo FuzzyCN2

```
procedure FuzzyCN2(InstanceSet)
returns RuleList
  RuleList ← ∅
  FuzzyInstSet ← AddActivationDegree(InstanceSet)
  Complex ← FindBestComplex(FuzzyInstSet)
  while IsNotNil(Complex) and IsNotEmpty(FuzzyInstSet)
    Conseq ← MostCoveredClass(FuzzyInstSet,Complex)
    NewRule ← "if( Complex ) then class = Conseq"
    RuleList ← Add(RuleList,NewRule)
    FuzzyInstSet ← Deactivate(FuzzyInstSet,Complex)
    FuzzyInstSet ← Truncate(FuzzyInstSet)
    Complex ← FindBestComplex(FuzzyInstSet)
  endwhile
  return RuleList
end
```

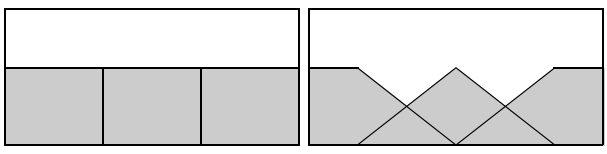
```
procedure FindBestComplex(FuzzyInstSet)
returns BestComplex
  BestComplex ← nil
  Selectors ← CreateAllSelectors()
  Star ← InitializeStar()
  while IsNotEmpty(Star)
    NewStar ← SpecializeStar(Star, Selectors)
    NewStar ← SignificanceTest(NewStar,FuzzInstSet)
    NewStar ← QualityTest(NewStar,FuzzyInstSet)
    BestComplex ← BetterQuality(NewStar,BestComplex)
    Star ← NewStar
  endwhile
  return BestComplex
end
```

El primer paso del algoritmo FuzzyCN2 consiste en transformar el conjunto de ejemplos de clasificación en un conjunto de ejemplos difusos añadiendo a cada ejemplo un grado de activación de la unidad (método *AddActivationDegree*). Al igual que el algoritmo CN2, el algoritmo FuzzyCN2 está dividido en dos bucles. El bucle externo consiste en buscar una nueva regla difusa (el método *FindBestComplex* genera el antecedente de la nueva regla y el método *MostCoveredClass* el consecuente), añadirla a la lista ordenada (método *Add*), eliminar parcialmente los ejemplos cubiertos por la regla (método *Deactivate*) y truncar el conjunto difuso de ejemplos para eliminar aquellos que estén poco activos (método *Truncate*). El bucle termina cuando no quedan ejemplos por cubrir o no se encuentra ninguna regla con los niveles de calidad y de significancia exigida.

El bucle interno se desarrolla en el método *FindBestComplex*. Este método toma un conjunto difuso de ejemplos y busca el mejor complejo en términos de significancia y calidad. El procedimiento considera un conjunto de selectores entre los que se incluyen los modificadores lingüísticos escogidos por el usuario (método *CreateAllSelectors*). La estrella contiene un conjunto de complejos candidatos. Inicialmente la estrella contiene un complejo vacío. En cada iteración se estudian todos los complejos obtenidos añadiendo uno de los selectores a cada uno de los complejos de la estrella (método *SpecializeStar*). De estos complejos se eliminan aquellos en los que algún atributo aparece más de dos veces o en los que se repite algún selector. También se eliminan aquellos en los no se alcanza el nivel mínimo de significancia (Definición 9). Tras esto se calcula la heurística de calidad sobre los complejos (Definición 10) y se genera una nueva estrella del tamaño fijado (método *QualityTest*). El bucle termina cuando la estrella queda vacía, devolviéndose el complejo de mayor calidad encontrado en la búsqueda.

5 ESTUDIO EXPERIMENTAL

A continuación se presenta un estudio comparativo del funcionamiento del algoritmo FuzzyCN2 frente a otros algoritmos de clasificación, tanto difusos como clásicos. Para ello se han escogido 11 conjuntos de datos de clasificación del repositorio UCI [1] con atributos continuos. En el caso de los algoritmos difusos, los atributos han sido descritos por medio de tres etiquetas lingüísticas, tal y como muestra la figura 2b. Con respecto a los algoritmos



(a) (b)

Figura 2: Discretización empleada en el estudio experimental (a) en los algoritmos clásicos; (b) en los algoritmos difusos.

Tabla 3: Número de reglas promedio generadas por los distintos algoritmos

	ID3	C4.5	Ripper	W-M	FuzzyID3	FuzzyCN2 v1	FuzzyCN2 v2	FuzzyCN2 v3
haberman	17.6	3.4	2.1	17.6	15.3	10.1	11.5	17.7
iris	8.9	4.8	3.3	20.4	5.8	5.0	5.8	6.4
bupa	40.4	8.8	2.1	52.8	16.9	14.7	16.3	22.1
ecoli	51.0	19.6	8.6	71.4	20.7	12.1	11.7	11.9
pima	115.7	16.0	3.3	152.6	30.0	24.6	28.3	34.4
glass	41.4	22.2	7.2	55.2	17.5	9.4	10.3	12.6
wbc	54.0	10.6	5.9	231.4	19.3	15.4	14.8	15.4
wine	25.7	12.2	6.0	152.1	18.4	8.7	9.0	8.5
cleveland	143.4	44.6	3.8	249.1	60.2	22.5	24.8	25.4
vehicle	283.7	90.4	12.5	462.9	38.2	28.6	33.3	36.1
ion	56.4	23.4	7.7	239.4	28.2	18.7	18.6	18.6
media	**76.2**	**23.3**	**5.7**	**155.0**	**24.6**	**15.4**	**16.8**	**19.0**

Tabla 4: Porcentaje de error promedio sobre el conjunto de entrenamiento

	ID3	C4.5	Ripper	W-M	FuzzyID3	FuzzyCN2 v1	FuzzyCN2 v2	FuzzyCN2 v3
haberman	24.1%	25.4%	25.4%	24.8%	24.7%	24.8%	24.2%	21.5%
iris	2.1%	2.1%	2.1%	2.7%	3.2%	4.4%	2.4%	2.6%
bupa	33.7%	37.9%	39.0%	36.9%	41.3%	37.0%	33.3%	29.7%
ecoli	12.1%	17.2%	17.5%	15.1%	19.5%	20.7%	12.9%	12.3%
pima	18.6%	23.2%	25.4%	22.2%	27.7%	21.5%	19.3%	16.4%
glass	22.9%	29.3%	31.2%	26.0%	48.2%	37.7%	31.7%	21.8%
wbc	0.4%	3.1%	3.0%	1.4%	2.6%	2.6%	1.6%	1.1%
wine	0.1%	5.2%	6.5%	0.0%	4.6%	3.5%	1.4%	1.2%
cleveland	1.4%	26.5%	40.9%	4.1%	29.9%	27.1%	22.5%	18.4%
vehicle	13.2%	25.7&	38.4%	23.9%	40.1%	30.3%	25.7%	20.4%
ion	0.6%	5.2%	7.0%	1.4%	6.9%	3.2%	1.6%	1.3%
media	**11.7%**	**18.3%**	**21.5%**	**14.4%**	**22.6%**	**19.3%**	**16.1%**	**13.3%**

Tabla 5: Porcentaje de error promedio sobre el conjunto de test

	ID3	C4.5	Ripper	W-M	FuzzyID3	FuzzyCN2 v1	FuzzyCN2 v2	FuzzyCN2 v3
haberman	28.4%	26.1%	26.5%	26.1%	29.4%	28.1%	28.1%	30.4%
iris	13.3%	12.0%	12.0%	5.3%	3.3%	4.7%	4.7%	2.7%
bupa	52.4%	44.0%	39.8%	42.6%	44.6%	41.3%	38.9%	37.4%
ecoli	32.1%	27.7%	29.7%	19.3%	24.5%	23.8%	15.2%	15.1%
pima	43.2%	27.8%	28.6%	27.2%	29.5%	27.5%	27.1%	25.5%
glass	56.8%	52.6%	50.3%	34.5%	61.1%	41.4%	39.7%	30.6%
wbc	5.0%	4.2%	4.4%	5.4%	3.5%	4.5%	4.5%	4.1%
wine	10.2%	12.4%	13.5%	10.7%	9.5%	6.7%	3.9%	4.5%
cleveland	57.5%	44.0%	46.1%	60.0%	46.4%	46.7%	46.3%	48.0%
vehicle	46.4%	44.3%	50.6%	38.2%	42.3%	35.4%	31.7%	32.0%
ion	12.9%	13.6%	13.7%	7.1%	12.2%	12.5%	11.3%	8.5%
media	**32.6%**	**28.1%**	**28.7%**	**25.1%**	**27.8%**	**24.8%**	**22.9%**	**21.7%**

clásicos, los datos han sido discretizados en tres intervalos del mismo tamaño con un significado equivalente al caso difuso, tal y como muestra la figura 2a. Los resultados del estudio utilizando una validación cruzada en 10 trozos se muestran en las tablas 3, 4 y 5. En el caso de los algoritmos clásicos, se han estudiado los algoritmos ID3, C4.5 y RIPPER utilizando para ello la herramienta Weka [15] y la configuración por defecto de estos algoritmos. En cuanto a los algoritmos difusos, se han estudiado los algoritmos de Wang y Mendel y FuzzyID3 realizando el estudio con el entorno Xfuzzy 3 [6].

Las últimas columnas de las tablas muestran el resultado del estudio para tres configuraciones del algoritmo FuzzyCN2. En la primera de ellas (v1) no se han utilizado modificadores lingüísticos en los selectores estudiados. En la segunda configuración (v2) se han incluido los modificadores lingüísticos mayor, menor, mayor o igual y menor o igual. En la tercera configuración (v3) se han considerado todos los modificadores lingüísticos presentados en la Figura 1. En todos los casos, los valores de los parámetros de control del algoritmo han sido de 0.5 para el corte alfa, 5 para el tamaño de la estrella, 5 para el soporte absoluto mínimo y 0.05 para el soporte relativo mínimo.

En cuanto al número de reglas generadas, las tres configuraciones muestran un resultado parecido aunque la versión v3 suele dar un número de reglas algo mayor. En los tres casos el número de reglas resulta significativamente menor que los obtenidos por el resto de algoritmos estudiados salvo RIPPER y C4.5. En general, el algoritmo RIPPER genera un número de reglas muy pequeño a costa de un porcentaje de error mayor que el del resto de algoritmos. Con respecto a C4.5, el número de reglas parece menor en conjuntos de datos con muchos atributos y más alto en conjuntos de datos con menos atributos.

En cuanto al error promedio sobre los conjuntos de test, los resultados para FuzzyCN2 resultan mejores que el del resto de los algoritmos estudiados, particularmente para la configuración v3. En este último caso y a pesar de que el número de conjuntos de datos estudiado no es demasiado grande, el test no paramétrico de Wilcoxon (Wilcoxon´s Signed Rank Test) permite afirmar que el error promedio del algoritmo FuzzyCN2 es menor que el del resto de los algoritmos estudiados con un nivel de confianza del 95%.

Agradecimientos

Este trabajo ha sido financiado parcialmente por los Proyectos de Investigación TEC2008-04920/TEC y TIN2008-06681-C06-06 de la CICYT y los Proyectos de Excelencia P07-TIC-03179 y P08-TIC-03674 de la Junta de Andalucía.

Referencias

[1] A. Asuncion, D.J. Newman,. UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. University of California, School of Information and Computer Science.

[2] K.J. Cios, L.M. Sztandera. Continuous ID3 algorithm with fuzzy entropy measures. *Proc. IEEE Int. Conference on Fuzzy Systems*, Pág. 469-476, 1992.

[3] P. Clark, R. Boswell. Rule induction with CN2: some recent improvements. *Proc. 5th European Working Session on Learning (EWSL-91)*, Pág. 151-163, 1991.

[4] P. Clark, T. Niblett. The CN2 Induction Algorithm. *Machine Learning*, 3, Pág. 261-283, 1989.

[5] 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.

[6] F.J. Moreno-Velo, I. Baturone, S. Sánchez-Solano, A. Barriga. Nuevos algoritmos de clasificación integrados en Xfuzzy. *Actas del XIV Congreso Español de Tecnologías y Lógica Fuzzy (ESTYLF 2008)*, Pág. 277-284, 2008.

[7] D. Nauck, R. Kruse. NEFCLASS - A Neuro-Fuzzy Approach for the Classificatioin of Data. *Proc. 1995 ACM Symposium on Applied Computing*, Pág. 461-465, 1995.

[8] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1, Pág. 81-106, 1986.

[9] J.R. Quinlan. Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, 4, Pág. 77-90, 1996.

[10] 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.

[11] 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.

[12] M. Umano, 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.

[13] L.Wang, J.M. Mendel. Generation of Rules by Learning from Examples. *IEEE Transactions on Systems, Man and Cybernetics*, 22, Pág. 1414-1427, 1992.

[14] R. Weber. Fuzzy ID3: a class of methods for automatic knowledge acquisition. *Proc. 2nd International Conference on Fuzzy Logic and Neural Networks*, Pág. 265-268, 1992.

[15] I.H. Witten, E. Frank. *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[16] 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.