REVISTA COLOMBIANA DE COMPUTACIÓN
Volumen 3, número 1
Págs. 7-20

Método de Inducción de Reglas de Clasificación Oblicuas mediante un Algoritmo Evolutivo

Álvarez, J.L. * Mata, J.* Riquelme, J.C. **

Resumen

En este artículo presentamos un nuevo método, denominado OBLIC, para inducción de reglas de clasificación oblicuas no jerárquicas a partir de un conjunto de datos etiquetados. La base del método es un algoritmo evolutivo con codificación real para los individuos y basado en la estrategia de Pittsburgh. Así, cada individuo está compuesto por un conjunto de reglas de clasificación que dividen el espacio de búsqueda en regiones para cada una de las clases del conjunto de datos. La función de bondad determina la exactitud de cada individuo mediante la exploración de estas regiones. El modelo de clasificación es deducido a partir del mejor individuo obtenido durante el proceso evolutivo.

Para analizar los resultados se ofrece una comparativa entre OBLIC, C4.5 y OC1 sobre un conjunto de bases de datos del UCI Repository.

Palabras Claves: *Minería de Datos, Aprendizaje Supervisado, Clasificación, Algoritmos Evolutivos.*

Abstract

This paper presents a new method, called OBLIC, to induce no-hierarchical oblique classification rules from a labeled dataset. The core of the method is an evolutionary algorithm with real-coded using the Pittsburgh approach. Each individual is compound for a set of classification rules. These rules split the search space into regions for each class of the dataset. The fitness function obtains the accuracy of the individuals by means of the exploration of these regions. The induced classification model is deduced from the best individual obtained during the evolutionary process.

Tenfold cross validation measure of the performance of OBLIC by a comparison of the results with C4.5 and OC1, using datasets from UCI Repository.

Keywords: *Data Mining, Supervised Learning, Classification, Evolutionary Algorithm.*

1. Introducción

El auge de la técnicas de adquisición de datos y el abaratamiento de los sistemas de almacenamiento, producidos en los últimos años, ha provocado un enorme volumen de información intratable mediante procedimientos manuales. Por otro lado, la riqueza que supone disponer de la información oculta en estos datos, para las organizaciones, ha llevado a la búsqueda de nuevos métodos y herramientas de aprendizaje automático y minería de datos. El objetivo

*Universidad de Huelva, Campus de la Rábida. {alvarez, mata}@uhu.es
**Universidad de Sevilla, Avda. Reina Mercedes, s/n. riquelme@lsi.us.es

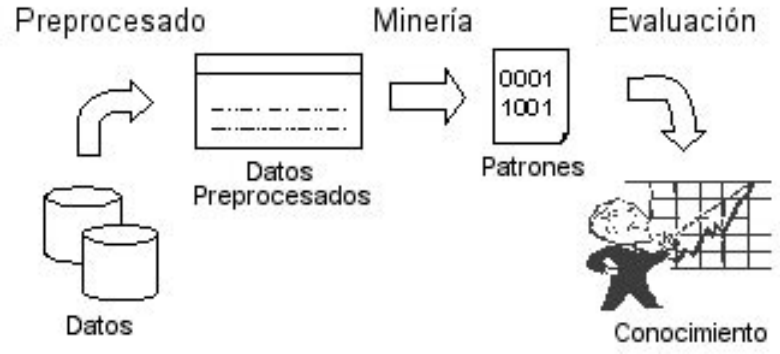

Figura 1: Proceso de extracción de conocimiento

de estás técnicas es la extracción del conocimiento útil, *a priori* desconocido, oculto en los datos para su posterior utilización en diferentes tareas de decisión.

La figura 1 presenta, de forma resumida, el proceso de extracción de conocimiento [8]. Inicialmente, los datos sufren un proceso de transformación y preprocesado que permitirá la adecuación de los mismos para el posterior tratamiento de minería. Tras la etapa de minería se obtienen un conjunto de patrones que aportarán al experto el conocimiento previamente oculto en los datos.

Atendiendo al objetivo deseado y a las características de los datos a analizar, existen diferentes técnicas para minería de datos: clasificación, clustering, reglas de asociación, etc. En concreto, la clasificación tiene como objetivo inducir un conjunto o modelo de reglas, a partir de un conjunto de datos compuesto por un conjunto de atributos o características $(x_1, x_2, ..., x_d)$ y su clase (c_i), donde cada regla establecerá una determinada clase en función de los valores de los atributos [4]. En este trabajo se considerará que todos los atributos $x_i \in \Re$, esta decisión no supone necesariamente una restricción del uso del método, ya que numerosos trabajos [3, 10, 20, 21, 22] han tratado la conversión de dominios simbólicos o numéricos y viceversa.

Dependiendo del formato de las reglas inducidas, existen dos tipos de clasificadores: ortogonales y oblicuos. En los clasificadores ortogonales, cada regla afecta sólo a un atributo, originando divisiones del espacio de búsqueda paralelas u ortogonales a los ejes. Así, cada regla tendrá una condición sobre un atributo de la forma $x_i < k$, donde x_i es un atributo y k un valor constante. Por otro lado, en los clasificadores oblicuos una regla contiene una combinación lineal de atributos, de la forma $\sum_{i=1}^{d} a_i x_i < k$, donde, de nuevo x_i representa a cada atributo, k una constante y a_i coeficientes reales. Este división del espacio de búsqueda se corresponderá con un hiperplano oblicuo a los ejes.

Los sistemas de clasificación más utilizados son los basados en árboles de decisión. Estos sistemas utilizan una estructura de árbol para inducir el modelo de clasificación, donde los nodos intermedios del árbol corresponden a atributos, para un clasificador ortogonal, o una combinación lineal de atributos, si es un clasificador oblicuo; los arcos están etiquetados con una condición sobre los atributos y las hojas se corresponde con una de las clases.

Para la inducción del árbol, estos sistemas disponen de un criterio que permite determinar cual es el valor para un atributo que produce la mejor división del espacio de búsqueda inicial, y posteriormente, iteran este procedimiento con los subespacios generados en la división anterior.

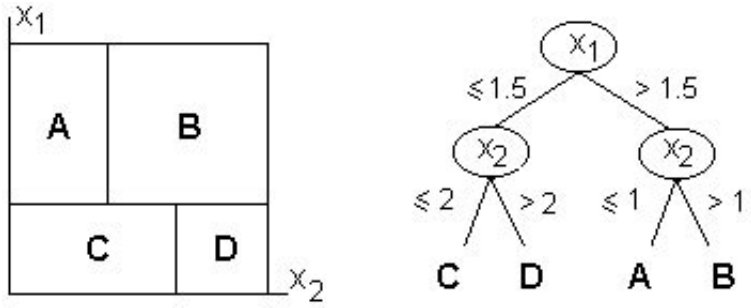

Figura 2: Clasificación mediante un árbol de decisión

$$
\begin{array}{l}
Si\ x_1 \leq 1{,}5 \\
|\ \ Si\ x_2 \leq 2 \Rightarrow \text{"}C\text{"} \\
|\ \ Sino \Rightarrow \text{"}D\text{"} \\
Sino \\
|\ \ Si\ x_2 \leq 1 \Rightarrow \text{"}A\text{"} \\
|\ \ Sino \Rightarrow \text{"}B\text{"}
\end{array}
\tag{1}
$$

Las reglas son obtenidas mediante la interpretación de este árbol, comenzando en la raíz y siguiendo cada una de las trayectorias hasta las hojas. En la figura 2 se ofrece un ejemplo de clasificación mediante árbol de decisión cuyas reglas son mostradas en la ecuación 1. Entre los clasificadores basados en árboles de decisión se encuentran C4.5 (clasificador ortogonal) [17] y OC1 (clasificador oblicuo) [16].

En este artículo, presentamos un método alternativo a los árboles de clasificación, denominado OBLIC (OBLIque Classifier - Clasificador Oblicuo). Este método utiliza un algoritmo evolutivo para la búsqueda del modelo de clasificación, de tal forma que cada potencial clasificación (cada individuo) es analizado en virtud de la exactitud de ese modelo para clasificar el conjunto de datos inicial. El modelo de reglas correspondiente al mejor individuo obtenido en el proceso evolutivo es presentado como resultado del algoritmo. Este conjunto de reglas divide el espacio de búsqueda en regiones donde cada una de ellas se corresponde con una de las posibles clases. Para clasificar futuras instancias se analiza a que región pertenece y se devuelve la clase de esa región.

Para la presentación del método el resto del artículo se ha estructurado de la siguiente forma. En la siguiente sección se ofrece una breve descripción de un algoritmo evolutivo, que servirá de introducción a la sección 3 donde se ofrecen los detalles del algoritmo de nuestro método. La sección 4 se ofrecen una breve ilustración de la ejecución del algoritmo sobre un conjunto de datos artificial. La sección 5 muestra una comparativa entre OBLIC y los sistemas de clasificación C4.5 y OC1, y finalmente, en la sección 6 se presentan las conclusiones sobre nuestro método.

2. Algoritmos Evolutivos

Un Algoritmo Evolutivo (AE) es una técnica de búsqueda y optimización inspirada en los principios de la selección natural y la reproducción genética de la naturaleza [9] o teoría de la evolución de Darwin.

Esto es, un AE es un método para la resolución de problemas de búsqueda y optimización que hace uso de los principios de la evolución de las especies: selección, reproducción y

```
Algoritmo Evolutivo
1.    P_i ← Inicializar población
2.    Repetir
3.        Para Cada I ϵ P_i
4.            Calcular bondad de I : f(I)
5.        Seleccionar el mejor de P_i para P_{i+1}
6.        Seleccionar % de P_i para P_{i+1}
7.        Cruzar individuos de P_i para P_{i+1}
8.        Mutar sobre Pi+1
9.        P_i ← P_{i+1}
10.   Hasta alcanzar número de generaciones
11.   Devolver mejor individuo obtenido
Fin
```

Figura 3: Algoritmo Evolutivo

mutación [23]. La figura 3 muestra el esquema de un algoritmo evolutivo simple [15, 14]. El objetivo del algoritmo es alcanzar la mejor solución para un determinado problema mediante la evolución de poblaciones codificadas como posibles soluciones para ese problema.

La representación de las posibles soluciones al problema, o de los individuos, consiste en la elección de un estructura de datos que represente el cromosoma que va a ser manipulado por el algoritmo y cuya decodificación ofrecerá una solución al problema que se desea resolver.

Esta elección, obviamente, es una de las principales decisiones durante la implementación del algoritmo, ya que de ella dependerá el resto de los operadores genéticos a implementar. A pesar de ello, no hay nada formalmente establecido sobre la elección de la representación más adecuada ya que depende fuertemente del problema a solucionar; eso sí, toda representación puede ser evaluada en función de una serie de propiedades que determinan su perfección. Estas propiedades son:

Completitud: Capacidad de representar todos los posibles casos del espacio de búsqueda.

Coherencia: Necesidad de que ningún caso no deseado sea representado.

Uniformidad: Necesidad de que todos los casos estén representados con el mismo porcentaje, a ser posible una sola vez.

Simplicidad: Capacidad de codificación y decodificación fácil.

Localidad: Necesidad de que las modificaciones en la estructura afecten en igual proporción en la solución representada.

Consistencia: Capacidad de no representar soluciones imposibles.

Minimalidad: Necesidad de ofrecer una representación con el menor tamaño posible.

Existen, básicamente, dos tipos de representación de los individuos: binaria [9] y real [7], aunque de forma menos extendida se ha realizado implementaciones híbridas.

Otro de los elementos fundamentales del proceso evolutivo es la evaluación de los individuos. Este procedimiento se realiza mediante una función, denominada función de bondad o

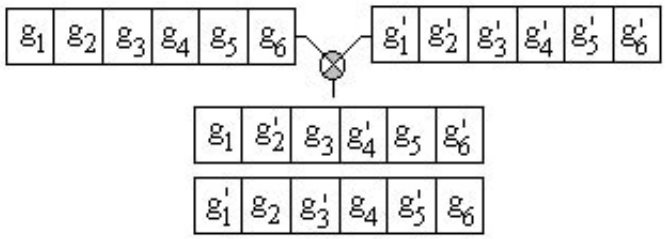

Figura 4: Cruce uniforme

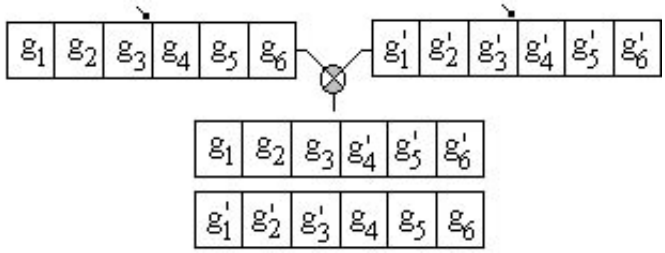

Figura 5: Cruce monopuntual (1-punto)

de evaluación, cuyo objetivo es suministrar una medida de aptitud de cada individuo, como solución para el problema planteado, dentro de la población actual. Como similitud con las leyes de la naturaleza, la función de bondad dentro del proceso evolutivo es como el medio para los seres vivos.

Generalmente, el proceso de evaluación de los individuos suele estar compuesto por una función que determina la bondad del mismo, por una función que establece una determinada penalización sobre el individuo y una función de ajuste que normaliza los valores a un intervalo determinado, normalmente [0,1].

Finalmente, los operadores genéticos: selección, cruce y mutación son decisiones vitales para la convergencia del algoritmo hacia una solución óptima.

El operador de selección tiene como objetivo elegir individuos de una población de los cuales todo o parte de su "material genético" pasará a la siguiente población. El método más utilizado se conoce como ruleta de la fortuna, donde una vez la bondad, f_i, de cada individuo ha sido ajustada al intervalo [0,1], se calculan la bondad acumulada de cada individuo $a_i = \sum_{j=1}^{i} f_j$ y se genera un número aleatorio b uniformemente distribuido en el intervalo [0,1]. El individuo seleccionado será aquel donde $b < a_i$.

El operador de cruce o recombinación consiste en el intercambio de material genético para generar los individuos de la siguiente población utilizando individuos de la población anterior, previamente elegidos con el operador de selección anterior. Los operadores de cruce más utilizados son el cruce uniforme y el cruce n-puntual.

Para el cruce uniforme los genes de los individuos seleccionados para cruce compiten con el objetivo de formar parte del individuo hijo [1, 19]. La figura 4 ofrece una representación gráfica de éste tipo de cruce.

Para el cruce n-puntual, los individuos involucrados en él son seccionados por n-puntos, generando individuos compuestos por los diferentes retales producidos por los cortes anteriores sobre los individuos padres [11, 5]. En la figura 5 se presenta un ejemplo de cruce mono-puntual.

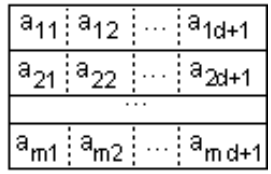

Figura 6: Representación de los individuos en OBLIC

Por último, el operador de mutación tiene como objetivo producir pequeñas alteraciones de algunos de los genes de los individuos, según una determinada probabilidad.

Otras consideraciones adicionales sobre los AE son la inicialización de la primera población y el ajuste de los diferentes parámetros que intervienen en el proceso, como: número de generaciones, número de individuos, probabilidad de cruce, probabilidad de mutación.

3. Algoritmo OBLIC

La base del algoritmo OBLIC es un AE (ver figura 3) con codificación real de individuos. Existen un gran número de trabajos donde se utilizan los algoritmos evolutivos para tareas de aprendizaje [6, 12, 13, 24]. En nuestro trabajo cada individuo (potencial clasificación) está compuesto por un número no fijo de vectores de valores reales, donde cada vector representa una regla de clasificación. Así, una regla de clasificación estará compuesta por un vector de números reales que representan la ecuación de un hiperplano en el espacio de búsqueda.

Para determinar la exactitud de una posible clasificación, la función de bondad analiza cada una de las regiones que forman los hiperplanos o reglas del individuo según la clase de las instancias, del conjunto de datos, en cada región.

Los operadores de selección, cruce y mutación generan las siguientes poblaciones del proceso evolutivo, y el resultado del algoritmo es la clasificación ofrecida por el mejor individuo obtenido durante dicho proceso.

En las siguientes subsecciones abordamos los detalles más significativos del algoritmo.

3.1. Representación de los individuos

Cada individuo o potencial clasificación está compuesto por un conjunto de vectores de valores reales; o sea, una matriz bidimensional donde el i^{th} vector representa la i^{th} regla de clasificación y cada j^{th} columna representa el j^{th} coeficiente para la combinación lineal de los atributos. Así, para un espacio d-dimensional $(x_1, x_2, ..., x_d)$ el i^{th} vector equivale a una regla de clasificación de ecuación 2, donde a_{ij} son los coeficientes del vector.

$$a_{i1}X_1 + a_{i2}X_2 + \cdots + a_{id}X_d + a_{d+1} \tag{2}$$

La representación de los individuos se basa en la estrategia de Pittsburgh, así, un individuo está compuesto por un conjunto no fijo de reglas del mismo tamaño. La figura 6 ofrece una representación gráfica de un individuo, para un espacio d-dimensional, compuesto por m reglas. La ecuación 3 muestra la interpretación de ese individuo, presentando el conjunto de reglas de clasificación que se deducen del mismo.

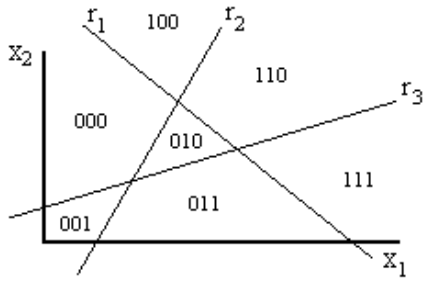

Figura 7: Codificación Binaria de Regiones

$$\begin{array}{c} a_{11}X_1 + a_{12}X_2 + \cdots + a_{1d}X_d + a_{1d+1} = 0 \\ a_{21}X_1 + a_{22}X_2 + \cdots + a_{2d}X_d + a_{2d+1} = 0 \\ \vdots \\ a_{m1}X_1 + a_{m2}X_2 + \cdots + a_{md}X_d + a_{md+1} = 0 \end{array} \quad (3)$$

3.2. Población inicial y sucesivas

La población inicial de individuos es seleccionada aleatoriamente, de tal forma, que para cada individuo se selecciona un valor entero m en el rango $[1, MR]$, donde MR es una constante que determina el máximo de reglas para un individuo, y $m \times (d+1)$ valores reales en el rango $[mc, Mc]$, donde mc y Mc es el valor mínimo y máximo para los coeficientes. El primer valor representa el número de reglas que tendrá el individuo y cada conjunto de $d+1$ valores representa la ecuación para cada una de las reglas. Los valores para los parámetros MR, mc, Mc y el resto de parámetros del algoritmo son detallados en la sección siguiente.

Las sucesivas poblaciones hasta alcanzar el número de generaciones deseadas, se obtienen a partir de la generación anterior, mediante la duplicación del mejor individuo obtenido en ella (criterio elitista), un porcentaje de individuos elegidos con el operador de selección y el resto mediante recombinación de individuos mediante el operador de cruce. Además, los individuos de la nueva población pueden verse afectados por alteraciones mediante el operador de mutación.

3.3. Función de evaluación

Para determinar la bondad de un individuo o clasificación, el algoritmo realiza una exploración de cada una de las regiones formadas por las reglas de clasificación. De esta forma, se analiza cada una de las instancias que existen en una determinada región, se asigna a la región la clase mayoritaria en ella, y se contabilizan las instancias de la misma clase como positivas y las de clase diferente como negativas.

Así, el algoritmo mantiene una ordenación de las reglas tal y como están ubicadas en el individuo para realizar una codificación binaria de las regiones que permitirá su identificación. La codificación de regiones estará compuesta por un conjunto de bits, uno por cada regla, donde el i^{th} bit se corresponde con la i^{th} regla. El valor de el i^{th} bit será 0 si la región está a la izquierda de la i^{th} regla y será 1 si la región está a la derecha de la misma.

La figura 7 presenta un ejemplo de esta codificación para un espacio bidimensional con tres reglas.

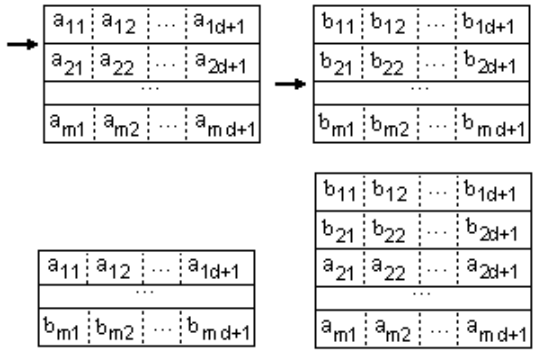

Figura 8: Operador de Cruce para OBLIC

A partir de la codificación anterior, las instancias del conjunto de datos son analizadas comprobando en que región están ubicadas. Para ello, la instancia es substituida en la ecuación de cada regla y si el valor resultante para una regla es mayor que 0 entonces la instancia está a la derecha de esa regla y sino entonces está a la izquierda. Repitiendo este proceso para cada regla se identificarán las instancias que están en cada una de las regiones, y por ello, se conocen cuantas instancias de cada clase existen en cada región, asignando a cada región la clase mayoritaria en ella y calculando el número de instancias con esa clase (aciertos) y el número de instancias con clase diferente (errores).

$$f(n) = \sum_{r=1}^{m} Pos_r(c) - \sum_{r=1}^{m} Neg_r(n) - Reglas(n) * FIR \tag{4}$$

Para determinar la bondad de una determinada clasificación c, el algoritmo utiliza la ecuación 4 donde $Pos_r(c)$ son las instancias de la misma clase que la región r, $Neg_r(c)$ el número de instancias de otras clases en la región r, $Reglas(c)$ el número de reglas de la clasificación y FIR es el factor de influencia del número de reglas. El efecto del número de reglas en la función de bondad tiene como objetivo premiar clasificación con menos reglas y el FIR es un valor en el rango [0,1] que tiene como objetivo relajar el número de reglas, de tal forma que valores cercanos a 0 provocan que tenga una influencia baja y valores cercanos a 1 que la influencia sea alta.

3.4. Operadores genéticos

Los operadores genéticos utilizados en el algoritmo son: selección, cruce y mutación. El operador de selección utilizado es el conocido como ruleta de la fortuna, su objetivo es seleccionar el porcentaje de individuos que serán duplicados para la siguiente generación y los individuos que forman parte de cada cruce.

El operador de cruce producirá nuevos individuos para las sucesivas poblaciones a partir de individuos de las poblaciones anteriores. El operador de cruce utilizado en el cruce mono-puntual, así, para cada cruce dos individuos son seleccionados y para cada individuo se elige aleatoriamente un punto de corte, que se corresponde con una regla, y se generan dos nuevos individuos formados por cada una de las partes producidas. Si en alguna ocasión el número de reglas de un individuo supera el máximo permitido las últimas reglas no son consideradas.

La figura 8 ofrece una representación gráfica del efecto del operador de cruce.

Cuadro 1: Valores de los parámetros del proceso evolutivo

Parámetro	Valor
Núm. de individuos	100
Núm de generaciones	200
Porcentaje de mutación	80 %
Probabilidad de mutación	70 %
Núm. máximo de reglas	12
Rango para coeficientes	[-10,10]
FIR	10 %

El operador de mutación permite realizar variaciones sobre los individuos de una población en función de una probabilidad de mutación, de tal forma que para cada individuo se elige un valor aleatorio y si este supera la probabilidad de mutación el individuo será mutado. El algoritmo utiliza tres operadores de mutación, que actúan aleatoriamente:

Alteración: Modifica un determinado coeficiente un porcentaje de mutación según la ecuación 5, donde $coef$ es el coeficiente, $Cant$ es la cantidad a mutar elegida aleatoriamente, $PMut$ es el porcentaje de mutación.

$$coef = coef \pm Cant * coef * PMut \tag{5}$$

Ampliación: Añade una nueva regla al individuo.

Reducción: Elimina aleatoriamente una de las reglas del individuo.

4. Ilustración Práctica

Una vez ha sido presentado el algoritmo OBLIC, en esta sección se ofrece un ejemplo práctico cuyo objetivo es presentar los resultados del algoritmo. Para llevar a cabo esta ilustración práctica, así como para obtener los resultados presentados en la siguiente sección, se han utilizado para cada uno de los parámetros del algoritmo los valores mostrados en la tabla 1.

Para presentar los resultados de OBLIC se ha generado una base de datos artificial compuesta por dos atributos (x_1, x_2) y dos clases (A,B). La distribución de las clases se ha realizado según la ecuación 6. En la figura 9 se ofrece una representación gráfica del conjunto de datos generado.

$$\begin{array}{l} Si \;\; x_2 \leq (x_1 - 5)^2 \Rightarrow \;\; "A" \\ Sino \;\; "B" \end{array} \tag{6}$$

La figura 10 muestra los resultados obtenidos para este conjunto de datos. Los resultados muestran cada una de las regiones inducidas, su tipo, el número de aciertos, el número de errores y las regla de clasificación inducidas. También, se ofrece el nombre del conjunto de datos analizado, el número de instancias, el número de atributos, el número de clases, el número de reglas inducido y el número y porcentaje de error. En la figura 11 se muestra gráficamente la clasificación obtenida por OBLIC.

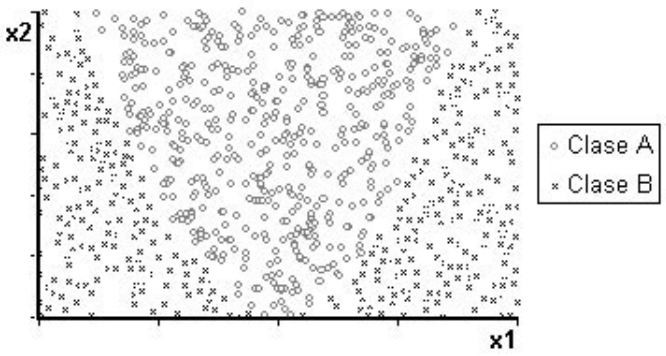

Figura 9: Conjunto de datos artificial

Training set: synthetic.dat
500 cases. 2 attributes. 2 class.
Classification
Regions
Region 0 ⟶ Class B. Pos: 155. Neg: 1
Region 2 ⟶ Class A. Pos: 184. Neg: 5
Region 3 ⟶ Class B. Pos: 154. Neg: 1
Rules
r1: (-6.00)X1 + (-3.00)X2 + (30.00) = 0
r2: (-6.00)X1 + (3.00)X2 + (30.00)= 0
#Reglas: 2 Errors: 7 (1.4 %)

Figura 10: Resultado inducido por OBLIC para la base de datos artificial

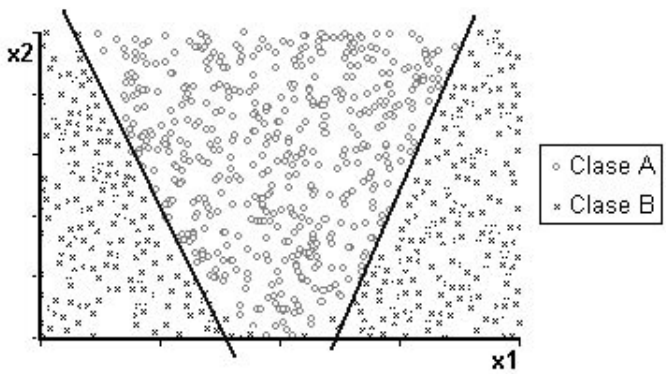

Figura 11: Representación gráfica de la clasificación obtenida por OBLIC

Cuadro 2: Bases de datos de UCI Repository

Databases	#Attr.	#Cases	#Cases-test	#Classes
BUPA	7	345	69	2
IRIS	4	150	27	3
TROID	5	215	43	3
PIMA	8	768	153	2
GLASS	10	214	42	6

Cuadro 3: Resultados de la comparativa

	C45		OC1		OBLIC	
	#Ru.	%Er.	#Ru.	%Er.	#Ru.	%Er.
BUPA	26.1	12.9	8.5	16.1	11.9	12.7
IRIS	4.2	3.1	3.7	2.3	3.8	0.9
TROID	8.3	1.8	5.6	1.5	6.2	0.7
PIMA	22.3	18.4	21.9	13.6	12.0	20.7
GLASS	5.0	1.3	5.0	1.3	5.0	1.3
Av.	13.18	7.50	8.94	6.96	7.78	7.26

5. Evaluación Empírica

La sección anterior ofrece una ilustración sobre la puesta en funcionamiento de nuestro método cuyo objetivo es presentar los resultados obtenidos con la herramienta pero que no permite demostrar que este método es un buen método de clasificación, ya que el ejemplo ha sido diseñado para la ocasión. En este apartado ofrecemos una comparativa, mediante validación cruzada, con dos de los sistemas de clasificación más utilizados entre la comunidad investigadora, como son C4.5 y OC1.

Para realizar la comparativa se han utilizado cinco bases de datos de UCI repository [2]. En la tabla 2 se muestran las principales características de estas bases de datos.

En la tabla 3 se presentan los resultados de la comparativa. La comparativa consiste en una validación cruzada, para la que se han realizado 10 ejecuciones sobre diferentes conjuntos de entrenamiento y test. Para cada base de datos y cada herramienta se muestran el valor medio del número de reglas (#Ru.) y el porcentaje de error (%Er.) obtenido en la fase de test.

La tabla 4 muestra una medida de la mejora [18] entre las herramientas y las figuras 12 y 13 ofrecen una comparativa gráfica de esta mejora.

De estos resultados, se puede deducir que OBLIC reduce en un 27 % el numero de reglas de la clasificación y en un 25 % el porcentaje de error con respecto a C4.5. Mientras que frente a OC1, OBLIC induce, aproximadamente, el mismo número de reglas, pero mejora en error en un 18 %.

En la comparativa, es necesario resaltar que el número de reglas máximo permitido en OBLIC tiene una influencia negativa sobre aquellas bases de datos de difícil clasificación, como ocurre con PIMA[1], ya que debido a la distribución de la clase necesita un mayor número de reglas para una adecuada clasificación. Esto provocará, como puede apreciarse en la tabla, que el porcentaje de error sea muy elevado para estos casos.

[1] Se ha llegado a esta conclusión tras analizar PIMA con algoritmos de clasificación basadas en los vecinos más cercanos. Estas herramientas han inducido sobre PIMA modelos de clasificación con aproximadamente un 25 % de error.

Cuadro 4: Comparativa general entre C4.5, OC1 y OBLIC

Databases	$R_{oblic}/R_{c4,5}$	$E_{oblic}/E_{c4,5}$	R_{oblic}/R_{OC1}	E_{oblic}/E_{OC1}
BUPA	0.45	0.98	1.4	0.78
IRIS	0.90	0.29	1.02	0.39
TROID	0.74	0.38	1.10	0.46
PIMA	0.57	1.12	0.54	1.52
GLASS	1.00	1.00	1.00	1.00
Av.	0.73	0.75	1.01	0.82

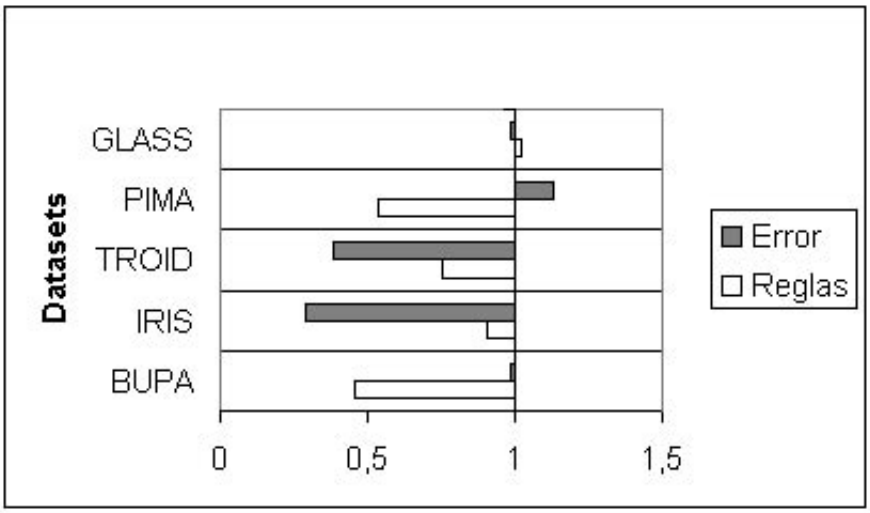

Figura 12: Comparativa gráfica general entre OBLIC y C4.5

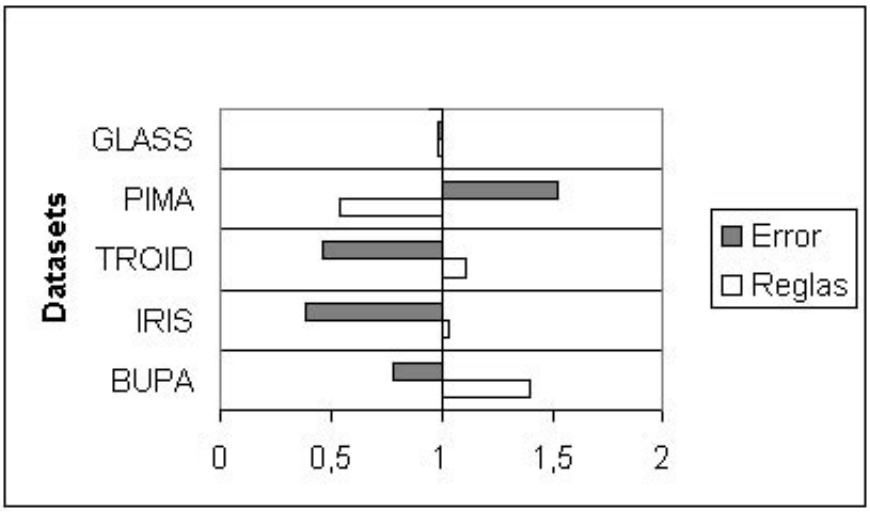

Figura 13: Comparativa gráfica general entre OBLIC y OC1

6. Conclusiones

En este artículo se ha presentado un nuevo método para inducción de reglas de clasificación oblicuas alternativo a los árboles de decisión. El método utiliza un algoritmo evolutivo para encontrar el mejor conjunto de reglas que divida el espacio de búsqueda en regiones para cada una de las diferentes clases.

Este método ofrece mejores resultados que los clasificadores ortogonales para la mayoría de las bases de datos en dos aspectos: reduciendo el número de reglas y el porcentaje de error. Además, el método tiene otras dos ventajas a resaltar, como son la inducción de reglas no jerárquicas y, a diferencia de los sistemas de clasificación basados en árboles de decisión, no necesita ningún criterio a priori para realizar las divisiones.

Referencias

[1] D. Ackley. *A Connectionist Machine for Genetic Hillclimbing*. Kluver Acad. Pub. 1987.

[2] C.L. Blake y C.J Merz. *[http://www.ics.uci.edu/∼mlearn/MLRepository.html]*. Irvine, CA: University of California, Department of Information and Computer Science. 1998.

[3] Breiman; Friedman; Olshen y Stone. *Classification and Regression Trees*. Wadsworth International Group. 1984.

[4] M.S. Chen; J. Han y P.S. Yu. Data Mining: An Overview from Database perspective. *IEEE Transactions on knowledge and Data Engineering*, 8(6), 866–883. 1996.

[5] K.A. De Jong. *An analysis of the behavoir of a class of genetic adaptive systems*. PhD thesis, University of Michigan. 1975

[6] K.A. De Jong. Using Genetic Algorithms for Concepts Learning. *Machine Learning*, 13 161–188, 1993.

[7] L.J. Eshelman y J.D. Schaffer. Real-Coded Genetic Algorithms and Interval Schemata. *Foundations of Genetic Algorithms 2*. Morgan Kaufmann Pub. 1993.

[8] Fayyad; Piatetsky-Shapiro y Smyth. From Data Mining to Knowledge Discovery: An Overview. *Advances in Knowledge Discovery and Data Mining*, 1–34, 1996.

[9] D.E. Goldberg *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Pub. Company, inc., 1989.

[10] S. Hampson y D. Volper. Linear function neurons: Structure and training. *Biological Cybernetics*, 53, 203–217, 1986.

[11] J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI. 1975.

[12] C.Z. Janikov. A Knowleged-Intensive Genetic Algorithm for Supervised Learning. *Machine Learning*, 13 189–228, 1993.

[13] J.R. Koza. *Concept Formation and Decision Tree Induction using the Genetic Programming Paradimg*. Springer-Verlag, 1991.

[14] Z. Michalewicz. *Genetics Algorithms + Data Structures = Evolution Programs, Third Edition*. Springer-Verlag, 1999.

[15] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press. 1996.

[16] S.K. Murthy; S. Kasif y S. Salzberg. A System for Induction of Obliques Decision Tress. *Journal or Artificial Intelligence Research*, 2 1–32, 1994.

[17] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Pub., 1993.

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

[19] G. Syswerda. Uniform crossover in Genetic Algorithms. *3th Int. Conference on Genetic Algorithms*, 2–9, 1989.

[20] P.E. Utgoff y C.E. Brodley. An incremental method for finding multivariate splits for decision trees. *7th International Conference on Machine Learning*, 58–65, 1990.

[21] T. Van de Merckt. NFDT: A system that learns flexible concepts based on decision trees for numerical attributes. *9th Int. Workshop on Machine Learning*, pp. 322–331, 1992.

[22] T. Van de Merckt. Decision trees in numerical attribute spaces. *13th International Joint Conference on Artificial Intelligence*, pp. 1016–1021, 1993.

[23] D. Whitley. *A Genetic Algorithm Tutorial*. Technical Report CS-93-103. Colorado State University. 1993.

[24] A.H. Wright. *Genetic Algorithm for Real Parameter Optimization*. Morgan Kaufmann Pub., 1991.