# Discretización Supervisada No Paramétrica Orientada a la Obtención de Reglas de Decisión

Raúl Giráldez, Jesús S. Aguilar y José C. Riquelme
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Sevilla
Avda. Reina Mercedes s/n
41012 – Sevilla

**Resumen:** *Muchos de los algoritmos de aprendizaje supervisado necesitan un espacio de atributos discreto, lo que hace imprescindible la aplicación de algún método de discretización que disminuya la cardinalidad de los valores que los atributos continuos pueden tomar. Algunos de los métodos propuestos en la literatura enfocan la discretización hacia la generación de reglas de decisión. Este trabajo aporta un nuevo algoritmo de discretización denominado USD (Unparametrized Supervised Discretization), el cual transforma el espacio infinito de valores de los atributos continuos en un conjunto finito de intervalos con el propósito de usar dichos intervalos en la generación de reglas de decisión, y además que cubran el máximo número posible de ejemplos sin que se produzca pérdida de bondad por parte de dichas reglas. Se han comparado los resultados con un método similar muy conocido, 1R [Hol93]. Destaca el hecho de que a diferencia de otros métodos, incluido 1R, USD no necesita parametrización.*

**Claves:** Discretización, Aprendizaje Supervisado

## 1 Introducción

Existen en la literatura gran cantidad de estudios centrados en el desarrollo de métodos de obtención de reglas de decisión en el contexto del Aprendizaje Supervisado. En general, las reglas pueden ser evaluadas según dos criterios: su precisión en la clasificación y su complejidad. La relación entre ambos criterios ha sido objeto de numerosos estudios [Agu01, Hol93].

Otro aspecto importante dentro del área en que nos encontramos es la disminución de la cardinalidad de los atributos continuos de una base de datos, especialmente en

procesos de minería de datos, donde muchos algoritmos de aprendizaje sólo operan en espacios finitos de atributos discretos o nominales [BB95]. Para trabajar con atributos continuos donde el rango de valores es infinito, puede ser recomendable la discretización de dichos atributos. Este problema también ha sido estudiado ampliamente en la literatura [AH96, BB95, DSK95, Hol93, KS96]. Los métodos de discretización se dividen en dos tipos: métodos supervisados (si la clase de los ejemplos es considerada en el proceso de discretización como atributo de decisión) y métodos no supervisados (si la clase no es considerada o bien es tratada como un atributo normal). Este trabajo se centra en el estudio de la discretización dentro del campo del aprendizaje supervisado. Numerosos algoritmos con tal propósito han sido estudiados y desarrollados en la literatura, entre los que destaca el propuesto en 1993 por Robert C. Holte y que denominó 1R (1-Rules)[Hol93]. Este método es considerado como clasificador más que como método de discretización, ya que su objetivo es la generación de reglas de decisión a partir de intervalos de valores calculados para los atributos continuos de una base de datos.

Este trabajo presenta un nuevo método de discretización de atributos continuos denominado USD (Unparametrized Supervised Discretization). USD disminuye la cardinalidad de los atributos continuos obteniendo intervalos que conservan la información de la base de datos original en el sentido de que el método pretende mantener la máxima bondad en los intervalos. Análogamente al método 1R, USD está enfocado para la generación de reglas de decisión, por lo que la maximización de la bondad de los intervalos constituye un aspecto fundamental del método. Aunque existen en la literatura numerosos métodos de discretización, dado que el propósito del algoritmo 1R es similar al de USD en tanto que enfoca la discretización hacia la obtención de reglas de decisión, los resultados de este trabajo serán comparados con los resultados generados por 1R.

Una característica a destacar de USD y que supone una ventaja frente a otros métodos de similar propósito es que no necesita parametrización por parte del usuario. Por ejemplo, antes de ejecutar 1R, es necesario establecer un parámetro que Holte denomina SMALL y que es fundamental para la ejecución del algoritmo [Hol93].

La descripción detallada del método así como las pruebas realizadas sobre el mismo se detallan en las secciones siguientes.

# 2 Propuesta

Dentro del área en que nos encontramos, aprendizaje supervisado, las bases de datos tienen un atributo denominado *clase* o *etiqueta*, la cual asigna a cada ejemplo un tipo. Esta clase tiene a su vez una cardinalidad que habitualmente es dos, aunque ese valor no influirá en el método que se va a presentar.

Antes de plantear la descripción del algoritmo, es necesario establecer varias definiciones con el fin de fijar la notación utilizada en este trabajo.

## 2.1 Definiciones

**Definición (valor puro)**. Decimos que un valor es puro, para un atributo cualquiera, cuando tiene la misma clase para todas las apariciones en los distintos ejemplos del conjunto de datos. En caso contrario, es decir, cuando dos ejemplos tienen el mismo valor para un atributo pero distinta clase, se denominará valor impuro.

**Definición (corte)**. Definimos los cortes como los valores que delimitan los intervalos calculados a lo largo del proceso de discretización. Es decir, para un determinado atributo $a_j$, el i-ésimo corte ($c_i$) será el límite superior abierto del intervalo $I_i$ y el límite inferior cerrado del intervalo $I_{i+1}$. Cada corte $c_i$ es calculado como la semisuma del mayor valor del atributo $a_j$ para los ejemplos contenidos en el intervalo $I_i$ y el menor valor del atributo $a_j$ para los ejemplos contenidos en el intervalo $I_{i+1}$. La figura 1 muestra gráficamente esta idea.

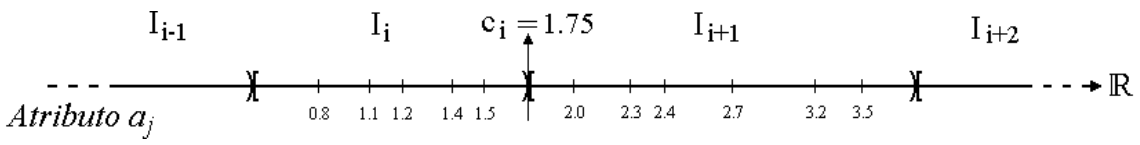


*Figura 1*. Ejemplo de cálculo de un corte simple.

**Definición (intervalo puro)**. Decimos que un intervalo es puro, para un atributo cualquiera, cuando todos los ejemplos que contiene pertenecen a la misma clase. En caso contrario decimos que el intervalo es impuro.

**Definición (clase mayoritaria)**. La clase mayoritaria de un intervalo es la clase con más apariciones dentro del intervalo. Así, el número de apariciones de la clase mayoritaria en un intervalo puro será igual al número de ejemplos que contenga el intervalo.

**Definición (bondad de un intervalo)**. Se define la bondad de un intervalo como la relación entre los aciertos y los errores de dicho intervalo. La ecuación que define la bondad puede variar dependiendo de la penalización por error que queramos considerar. Una posible expresión de la bondad es expresada en la ecuación 1, en la que la penalización por error es alta, al encontrarse el número de errores en el denominador.

$$bondad = \frac{aciertos}{1 + errores} \tag{1}$$

donde los aciertos de un intervalo es el número de ejemplos contenidos cuya clase es igual a la clase mayoritaria. Y análogamente, los errores de un intervalo es el número de ejemplos del intervalo cuya clase es diferente a la clase mayoritaria.

## 2.2 Algoritmo

El objetivo que el algoritmo USD persigue es dividir los atributos continuos en intervalos de máxima bondad, de forma que la bondad media de todos los intervalos finales para un determinado atributo sea lo más alta posible. Todo el proceso se realiza sin la necesidad de que el usuario introduzca ningún parámetro ni información adicional. La forma en que se calculan los intervalos hacen que el algoritmo sea determinista.

```
1. Procedimiento USD (BD: Base de datos)
2.    InicializaCortes(BD)
3.    Para cada atributo continuo de BD
4.      Para cada intervalo I_i menos el último
5.        Si CondiciónDeUnión(I_i, I_i+1)=CIERTO
6.              MarcarPosibleUnión(I_i, I_i+1) y su bondad
7.        Fin Si
8.      Fin Para
9.      Mientras haya posibles uniones
10.       i = Unir posible unión con máxima bondad
11.       Si CondiciónDeUnión(I_i, I_i+1)=CIERTO
12.             MarcarPosibleUnión(I_i, I_i+1) y su bondad
13.       Fin Si
14.       Si CondiciónDeUnión(I_i-1, I_i)=CIERTO
15.             MarcarPosibleUnión(I_i-1, I_i) y su bondad
16.       Fin Si
17.      Fin Mientras
18.   Fin Para
19.Fin Proc.

20. CondiciónDeUnión(I_i, I_i+1) =
21.  [(I_i tiene la misma clase mayoritaria que I_i+1)O(hay empate en I_i o I_i+1)]
22.    Y [la bondad de la unión de I_i y I_i+1 es mayor o igual a la media de la
23.      bondad de I_i y la bondad de I_i+1]
```

*Figura 2.* Algoritmo USD.

La figura 2 presenta el algoritmo USD. El procedimiento principal se divide en dos partes diferenciadas. La primera parte calcula los intervalos iniciales que posteriormente serán refinados en la segunda parte dependiendo de las bondades que se obtengan tras realizar las acciones posibles: unir dos intervalos o dejarlos independientes.

## 2.3 Cálculo de los intervalos iniciales

El cálculo de los intervalos iniciales (figura 2, línea 2) podría constituir un método de discretización simple por sí solo. Este proceso maximiza la pureza de los intervalos con el propósito de obtener las mejores bondades posibles, es decir, obtiene intervalos tan puros como sea posible independientemente de los ejemplos que contenga. Esto hace que el número de intervalos sea relativamente elevado, aunque no supone una desventaja puesto que, posteriormente, el proceso de refinamiento reducirá considerablemente dicho número.

Para facilitar la comprensión del proceso de cálculo de los intervalos iniciales, planteamos un ejemplo sencillo, tomado de [Agu01].

**Ejemplo.** Supongamos que tenemos los datos representados en la tabla I, donde destacamos los valores de un atributo particular, con valores reales, su frecuencia de aparición y la clase. Nótese que el conjunto de datos ha sido ordenado previamente por el atributo en cuestión.

*Tabla I.* Ejemplo de base de datos con atributos reales y sólo dos clases.

| N | $a_k$ | $frec(C_A)$ | $frec(C_B)$ | Clase Mayoritaria |
|---|---|---|---|---|
| 1 | 1.0 | 5 | 0 | A |
| 2 | 1.2 | 4 | 0 | A |
| 3 | 1.4 | 0 | 3 | B |
| 4 | 1.6 | 0 | 4 | B |
| 5 | 1.8 | 6 | 0 | A |
| 6 | 2.0 | 5 | 1 | A |
| 7 | 2.2 | 5 | 2 | A |
| 8 | 2.4 | 0 | 4 | B |
| 9 | 2.6 | 1 | 6 | B |
| 10 | 3.0 | 1 | 5 | B |
| 11 | 3.2 | 0 | 4 | B |
| 12 | 3.4 | 6 | 2 | A |
| 13 | 3.6 | 1 | 3 | B |
| 14 | 3.8 | 8 | 1 | A |
| 15 | 4.0 | 6 | 0 | A |
| 16 | 4.2 | 2 | 7 | B |
| 17 | 4.4 | 8 | 0 | A |

La tabla I recoge todas las posibles situaciones que se pueden dar en una base de datos, concretamente ésta tiene 100 ejemplos. Las situaciones corresponden a todas las combinaciones entre valores consecutivos que tienen clase igual y distinta y, a su vez, el valor es puro e impuro. En general, el número de posibilidades viene dado por la expresión $4|C|^2$, donde $|C|$ es el número de clases diferentes de la base de datos. A partir de la tabla I calcularemos los *cortes*. Los valores por parejas son tratados de forma consecutiva, es decir, se analizan el primero y el segundo; a continuación el segundo y el tercero; y así sucesivamente.

La tabla II muestra las acciones a realizar, fijar un corte o no, en cada una de las situaciones. La primera columna (Pareja) indica qué ejemplos están considerándose; las dos columnas siguientes (Clase) muestran sus clases mayoritarias; las dos siguientes (Puro) indican si el valor es puro o no; y la última (Acción), precisa la acción a realizar con la pareja de ejemplos.

*Tabla II.* Análisis de la coloración de un corte o unión de dos valores de un intervalo.

| **Pareja** | ***Clase*$_i$** | ***Clase*$_{i+1}$** | ***Puro(valor*$_i$*)*** | ***Puro(valor*$_{i+1}$*)*** | **Acción** |
|---|---|---|---|---|---|
| 1,2 | A | A | S | S | unir |
| 2,3 | A | B | S | S | corte |
| 3,4 | B | B | S | S | unir |
| 4,5 | B | A | S | S | corte |
| 5,6 | A | A | S | N | corte |
| 6,7 | A | A | N | N | unir |
| 7,8 | A | B | N | S | corte |
| 8,9 | B | B | S | N | corte |
| 9,10 | B | B | N | N | unir |
| 10,11 | B | B | N | S | corte |
| 11,12 | B | A | S | N | corte |
| 12,13 | A | B | N | N | corte |
| 13,14 | B | A | N | N | corte |
| 14,15 | A | A | N | S | corte |
| 15,16 | A | B | S | N | corte |
| 16,17 | B | A | N | S | corte |

Mediante un mapa de Karnaugh se obtiene la expresión lógica de la figura 3, que indica qué condiciones se deben cumplir para que se fije un corte, es decir, la condición de corte en el proceso de cálculo de los intervalos iniciales.

***Si***$((Clase_i \neq Clase_{i+1})$ *O* $(Puro(valor_i) \neq Puro(valor_{i+1})))$
*"colocar un corte en la semisuma de valores"*
***si no***
*"unir ambos valores en un intervalo"*

*Figura 3.* Condición de corte o unión de valores en un intervalo.

## 2.4 Refinamiento de los intervalos

Una vez obtenidos los intervalos mediante las condiciones establecidas en la figura 3, USD procesa estos intervalos para reducir la cardinalidad de los mismos sin que se produzca pérdida de la bondad global. En la figura 2, el proceso de refinamiento es realizado entre las líneas 3 y 18. Básicamente, el refinamiento consiste en recorrer los intervalos para cada atributo y evaluar, para cada par de intervalos consecutivos, si es o no posible unirlos dependiendo de la condición de unión expresada entre las líneas 20 y 23 de la figura 2. En esta condición, el primer término de la conjunción evita que dos intervalos con diferente clase mayoritaria sean unidos, pues dicha unión no podría ser ventajosa en conceptos de bondad. El segundo término de la conjunción compara la bondad de la unión con la semisuma de las bondades de los intervalos que intervienen en el par. Esta semisuma representa la bondad media que obtendríamos si no se produce la unión de los dos intervalos. Si un par de intervalos satisface la condición de unión, se marca una posible unión, almacenando también la bondad de dicha unión (figura 2, línea 6)

Una vez calculadas todas las posibles uniones para el atributo en estudio, Se pasa a ejecutar el bucle "*mientras*" (figura 2, línea 9), uniéndose en cada iteración sólo aquellos intervalos cuya posible unión sea máxima (figura 2, línea 10). Esta unión produce un nuevo conjunto de intervalos donde se calculan las nuevas posibles uniones y bondades de los dos pares de intervalos en los que interviene el nuevo intervalo resultado de la unión anterior (figura 2, líneas 11 a 16). El proceso se ejecuta mientras que el conjunto de posibles uniones no sea vacío. Cuando en una iteración no se ha producido unión, se pasa a tratar el siguiente atributo continuo.

## 2.5 Ejemplo

La tabla III muestra los intervalos obtenidos para la base de datos del ejemplo de la tabla I. Las dos primeras columnas ($N$, $a_k$) son similares a las de la tabla I. Las tres columnas siguientes corresponden a los intervalos iniciales (*I.I.*), la clase mayoritaria (*C.M.*) de dichos intervalos y la bondad de éstos (*Bond.*) aplicando la expresión de la *bondad* (ecuación 1). Los tres siguientes grupos de columnas corresponden a las iteraciones que el proceso de refinamiento de intervalos lleva a cabo para este ejemplo, indicando en cada caso los nuevos intervalos (*N.I.*), su clase mayoritaria y su bondad, respectivamente.

*Tabla III*. Ejemplo de aplicación del algoritmo USD.

| N | $a_k$ | I.I. | C.M. | Bond. | Iteración 1 | | | Iteración 2 | | | Iteración 3 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | N.I. | C.M. | Bond. | N.I. | C.M. | Bond. | N.I. | C.M. | Bond. |
| 1 | 1.0 | $I_1$ | A | 9.0 | $I_1$ | A | 9.0 | $I_1$ | A | 9.0 | $I_1$ | A | 9.0 |
| 2 | 1.2 | | | | | | | | | | | | |
| 3 | 1.4 | $I_2$ | B | 7.0 | $I_2$ | B | 7.0 | $I_2$ | B | 7.0 | $I_2$ | B | 7.0 |
| 4 | 1.6 | | | | | | | | | | | | |
| 5 | 1.8 | $I_3$ | A | 6.0 | $I_3$ | A | 6.0 | $I_3$ | A | 6.0 | $I_3$ | A | 6.0 |
| 6 | 2.0 | $I_4$ | A | 2.5 | $I_4$ | A | 2.5 | $I_4$ | A | 2.5 | $I_4$ | A | 2.5 |
| 7 | 2.2 | | | | | | | | | | | | |
| 8 | 2.4 | $I_5$ | B | 4.0 | $I_5$ | B | 4.0 | $I_5$ | B | 5.0 | $I_5$ | B | 6.33 |
| 9 | 2.6 | $I_6$ | B | 3.66 | $I_6$ | B | 3.66 | | | | | | |
| 10 | 3.0 | | | | | | | | | | | | |
| 11 | 3.2 | $I_7$ | B | 4.0 | $I_7$ | B | 4.0 | $I_6$ | B | 4.0 | | | |
| 12 | 3.4 | $I_8$ | A | 2.0 | $I_8$ | A | 2.0 | $I_7$ | A | 2.0 | $I_6$ | A | 2.0 |
| 13 | 3.6 | $I_9$ | B | 1.5 | $I_9$ | B | 1.5 | $I_8$ | B | 1.5 | $I_7$ | B | 1.5 |
| 14 | 3.8 | $I_{10}$ | A | 4.0 | $I_{10}$ | A | 7.0 | $I_9$ | A | 7.0 | $I_8$ | A | 7.0 |
| 15 | 4.0 | $I_{11}$ | A | 6.0 | | | | | | | | | |
| 16 | 4.2 | $I_{12}$ | B | 2.33 | $I_{11}$ | B | 2.33 | $I_{10}$ | B | 2.33 | $I_9$ | B | 2.33 |
| 17 | 4.4 | $I_{13}$ | A | 8.0 | $I_{12}$ | A | 8.0 | $I_{11}$ | A | 8.0 | $I_{10}$ | A | 8.0 |

Partiendo de los intervalos iniciales se va aplicando el proceso de refinamiento a los intervalos resultantes de cada iteración. En tabla III se muestra sombreados los interva-

los resultantes de las uniones en cada iteración. Así, en la *iteración 1* son unidos los intervalos $I_{10}$ e $I_{11}$ formando el nuevo intervalo $I_{10}$. Sobre el nuevo conjunto de intervalos se aplica nuevamente el proceso de refinamiento, dando como resultado la unión de los intervalos $I_5$ e $I_6$ en la *iteración 2*. Por último, en la *iteración 3* son unidos los intervalos $I_5$ e $I_6$ obtenidos tras la *iteración 2*, dando como resultado el conjunto final de intervalos: *{[1,1.3), [1.3,1.7), [1.7,1.9), [1.9 2.3), [2.3,3.3), [3.3,3.5), [3.5,3.7),[3.7,4.1), [4.1,4.3), [4.3,4.4]}*. Con este sencillo ejemplo se pretende recoger tanto la unión de intervalos iniciales como la de un intervalo fruto de la unión de dos con otro cualquiera.

# 3 Pruebas

Se han realizado pruebas exhaustivas del método, comparándolo con el algoritmo 1R, usando bases de datos de UCI [BM98]. El proceso de prueba ha consistido en la generación aleatoria de reglas de decisión a partir de los intervalos obtenidos por USD y 1R, comparando la relación entre aciertos y errores de dichas reglas. Las reglas de decisión generadas tienen la siguiente notación:

$$Regla: a_1 \in [li_1, ls_1] \; Y \; a_2 \in [li_2, ls_2] Y \ldots Y \; a_k \in [li_k, ls_k]$$

donde $a_i \in [li_i, ls_i]$ representa que el atributo i-ésimo ($a_i$) pertenece al intervalo calculado $[li_i, ls_i]$ con límite superior $li_i$ y límite inferior $ls_i$. Los intervalos que intervienen en las reglas son calculados a partir de los intervalos obtenidos por cada algoritmo de forma que cada regla generada contenga al menos un ejemplo de la base de datos. En general, los ejemplos que contendrá la regla serán aquellos cuyos valores de cada atributo $a_i$ estén incluidos entre los límites $li_i$ y $ls_i$.

El proceso de generación de una regla consiste en elegir al azar un ejemplo de la base de datos y para cada atributo se genera el límite inferior y el límite superior de forma pseudoaleatoria a partir de los cortes obtenidos por cada algoritmo. El límite inferior es calculado eligiendo al azar un corte menor que el valor que el ejemplo toma en el atributo en estudio. Análogamente, el límite superior se obtiene eligiendo al azar un corte mayor que el valor del dicho atributo para el ejemplo. Este cálculo es repetido para cada atributo de forma que se asegura que la regla generada contiene al menos un ejemplo de la base de datos, que es aquel que fue elegido de forma aleatoria. La figura 4 ilustra este proceso. Como muestra la figura 4, el intervalo para el atributo $a_i$ en la $Regla_j$ tomará un valor para el límite inferior en el conjunto {-5.6,-3.8,-1.9,2.0} y para el límite superior en el conjunto {4.2, 6.9, 11.2, 15.4}.

Siguiendo el proceso de generación de reglas descrito anteriormente, han sido generadas 100.000 reglas de forma aleatoria para cada algoritmo y para cada base de datos. El propósito de estas pruebas es comparar la bondad de los intervalos obtenidos por USD frente a los obtenidos por 1R para diferentes valores del parámetro SMALL que

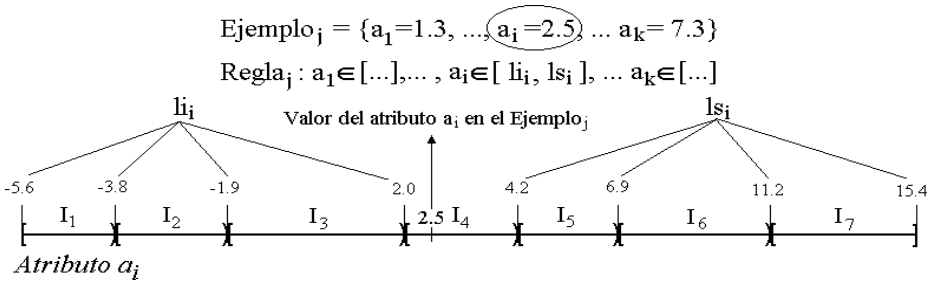


*Figura 4*. Ejemplo de cálculo de regla.

1R necesita para su funcionamiento. Esta bondad se basa fundamentalmente en el porcentaje de aciertos de las reglas obtenidas a partir de cada método. La justificación del método de prueba elegido radica en que a mayor bondad de los intervalos obtenidos, mejores serán las reglas calculadas a partir de dichos intervalos. De este modo, una medida importante es el porcentaje de aciertos de las reglas generadas por cada método.

La tabla VI muestra los resultados obtenidos aplicando las pruebas al método USD y 1R para valores del parámetro SMALL desde 1 a 7 (valor entre paréntesis en la columna de métodos *MÉT*). Las bases de datos utilizadas para la realización de las pruebas son *iris*, *pima*, *heart*, *ionosphere*, *wine*, *glass*, *vehicle y letter*, indicadas en la tabla por las dos primeras letras del nombre. Las dos columnas de valores bajo cada una de las bases de datos indican el porcentaje de aciertos (*%AC*) y el número medio de cortes (*NC*) respectivamente para cada método.

*Tabla IV*. Resultados.

| **MÉT** | **IR** | | **PI** | | **HE** | | **IO** | | **WI** | | **GL** | | **VE** | | **LE** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **%AC** | **NC** | **%AC** | **NC** | **%AC** | **NC** | **%AC** | **NC** | **%AC** | **NC** | **%AC** | **NC** | **%AC** | **NC** | **%AC** | **NC** |
| USD: | 78,2 | 7,2 | 76,1 | 52,6 | 85,0 | 11,7 | 99,9 | 68,4 | 99,6 | 36,0 | 65,6 | 54,4 | 69,8 | 37,7 | 25,4 | 13,1 |
| 1R(1): | 73,3 | 6,0 | 75,6 | 56,1 | 82,8 | 13,3 | 99,9 | 68,5 | 99,4 | 34,4 | 65,8 | 52,2 | 69,6 | 38,7 | 24,4 | 13,6 |
| 1R(2): | 73,7 | 5,7 | 75,9 | 45,0 | 83,1 | 11,8 | 99,9 | 40,1 | 99,5 | 23,6 | 67,4 | 30,1 | 70,9 | 34,3 | 24,3 | 13,6 |
| 1R(3): | 73,9 | 5,7 | 75,8 | 36,4 | 83,1 | 10,5 | 99,9 | 30,8 | 99,4 | 19,4 | 69,4 | 20,8 | 71,3 | 31,1 | 24,4 | 13,6 |
| 1R(4): | 75,8 | 5,2 | 75,9 | 32,5 | 83,2 | 9,1 | 99,9 | 25,9 | 99,1 | 15,8 | 69,1 | 17,4 | 73,0 | 28,7 | 24,1 | 13,6 |
| 1R(5): | 75,2 | 5,2 | 76,5 | 28,5 | 83,0 | 8,1 | 99,9 | 23,3 | 99,0 | 13,8 | 68,7 | 14,5 | 72,8 | 25,6 | 24,5 | 13,6 |
| 1R(6): | 75,1 | 4,7 | 76,5 | 25,7 | 82,7 | 7,7 | 99,9 | 20,9 | 98,9 | 12,5 | 68,3 | 12,8 | 73,5 | 23,7 | 23,8 | 13,6 |
| 1R(7): | 75,4 | 4,7 | 76,2 | 23,1 | 82,5 | 7,1 | 99,9 | 19,3 | 98,5 | 11,0 | 67,9 | 11,3 | 74,4 | 22,1 | 24,4 | 13,6 |

Como se puede observar, para la mitad de las bases de datos el porcentaje de acierto que USD consigue supera al obtenido por 1R para cualquier valor de SMALL estudiado. Con respecto al número medio de cortes obtenido en cada caso, USD obtiene un número de cortes sensiblemente mayor que 1R, sin embargo, con USD no tenemos que preocuparnos de establecer el valor de SMALL más favorable y, por tanto, prescindimos de la realización de múltiples experimentos.

# 4 Conclusiones

Como resultado de este trabajo se ha obtenido un algoritmo de discretización supervisado no paramétrico que disminuye la cardinalidad de los atributos continuos de una base de datos etiquetada. USD divide el espacio de búsqueda de los atributos continuos en intervalos, intentando que estos intervalos conserven la máxima bondad posible para un número de intervalos aproximadamente del mismo orden. Una ventaja importante de USD frente a otros algoritmos de discretización es la nula parametrización, es decir, USD no necesita ningún parámetro proporcionado por el usuario.

# Referencias

[Agu01] J. S. Aguilar. *Generación de Reglas Jerárquicas de Decisión con Algoritmos Evolutivos en Aprendizaje Supervisado.* Tesis Doctoral, Univ. de Sevilla.

[AH96] C. Apte and S. Hong. 1996. *Predicting equity returns from securities data*. chapter 22, 542 -- 560. In (Fayyad et al. 1996).

[BB95] P. Berka and I. Bruha. *Empirical comparison of various discretization procedures.* Technical Report LISP-95-04, Laboratory of Intelligent Systems, Prage, 1995.

[BM98] C. Blake and E.K. Merz. Uci repository of machine learning databases, 1998.

[DKS95] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and Unsupervised Discretization of Continuous Features. *Proceedings of the Twelfth International Conference on Machine Learning*. pág. 194--202, 1995.

[FI93] U. Fayyad and K.B. Irani (1993), Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. *Proceedings of 13th International Joint Conference on Artificial Intelligence,* pág. 1022-1027.

[Hol93] R. C. Holte. Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. *Machine Learning*, 11:63-91, 1993.

[KS96] R. Kohavi and M. Sahami. Error-based and entropy-based discretization of continuous features. *Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining*, pág. 114-119, 1996. AAAI Press.