Un algoritmo de clasificación ordinal basado en el modelo de recubrimiento secuencial

Juan Carlos Gámez[1], David García[2], Antonio González[2], and Raúl Pérez[2]

[1] Dpto. Arq. de Computadores, E y TE
Escuela Politécnica Superior de Córdoba
Universidad de Córdoba (Spain)
jcgamez@uco.es,
[2] Dpto. Ciencias de la Comp. e IA
E. T. S. de Ingeniería Informática y Telecomunicaciones
Universidad de Granada (Spain)
{dgarcia,A.Gonzalez,fgr}@decsai.ugr.es,

Abstract. La *clasificación ordinal* es un problema de clasificación supervisada cuyo objetivo es predecir la categoría a la que pertenece un patrón teniendo presente que hay una relación de orden entre dichas categorías. En este trabajo presentamos un algoritmo de aprendizaje de reglas difusas basado en la estrategia de recubrimiento secuencial para clasificación ordinal. Este algoritmo es la extensión a clasificación ordinal de NSLV, un algoritmo para clasificación. Dicha extensión se basa en una nueva definición del concepto de ejemplo negativo, que implica un nuevo modelo de evaluación de reglas. Finalmente se presentan un estudio experimental que muestra el buen funcionamiento del algoritmo propuesto en comparación con otros algoritmos conocidos para clasificación ordinal.

1 Introducción

Los problemas de clasificación ordinal [10] son un tipo de problema de aprendizaje supervisado que se encuentra situado entre la clasificación nominal y la regresión. De la primera se diferencia en que existe un orden preestablecido entre las clases, mientras que de la segunda se distingue en que el conjunto de etiquetas es finito y las diferencias entre los valores de las etiquetas no están definidas. Por ello se hace necesario utilizar técnicas que nos permitan llevar a cabo el aprendizaje explotando el conocimiento existente sobre la distribución de patrones en el espacio de entrada.

La clasificación ordinal tiene aplicación en multitud de áreas como la evaluación de la enseñanza [9] o predicción de la velocidad del viento [6] entre otras.

Se han utilizado distintos métodos para abordar la clasificación ordinal. En [3] se plantea una metodología basada en árboles de decisión binarios, transformando un problema ordinal con k clases en k-1 problemas de clasificación binaria. En [1, 7, 8] se muestra una adaptación de las máquinas de soporte vectorial para abordar este problema, y en [12] se hace uso del análisis discriminante.

Debido a las diferencias y similitudes con la clasificación nominal y la regresión, en la evaluación del clasificador se puede utilizar la combinación de métricas de clasificación y regresión, así como métricas específicas [2].

La estrategia de recubrimiento secuencial ha sido capaz de demostrar su utilidad como base para el desarrollo de algoritmos de aprendizaje muy conocidos. Esta estrategia aprende un conjunto de reglas mediante un proceso iterativo en el que en cada paso se añade una regla al conjunto de reglas, penalizando los ejemplos de las reglas aprendidas para que no sean considerados nuevamente en la siguiente iteración. Este proceso se repite hasta que no se pueden encontrar más reglas útiles.

En este trabajo se presenta una propuesta basada en la estrategia de recubrimiento secuencial para abordar la clasificación ordinal. Para ello presentamos en este trabajo la extensión a este tipo del problemas del algoritmo NSLV [4], un algoritmo de aprendizaje de reglas difusas para clasificación nominal.

El resto del trabajo está organizado como sigue. En la sección 2 presentamos el algoritmo de aprendizaje NSLV; en la sección 3 se expone las modificaciones propuestas para la adaptación de este algoritmo a problemas de clasificación ordinal; en la sección 4 exponemos los resultados experimentales obtenidos, finalizando la sección 5 con las conclusiones.

2 Estrategia de recubrimiento secuencial

NSLV [5] es un algoritmo de aprendizaje de reglas difusas basado en la estrategia de recubrimiento secuencial y que utiliza como mecanismo de búsqueda un algoritmo genético. La estrategia de recubrimiento secuencial es una estrategia de descomposición, en la que en cada paso se va obteniendo de forma iterativa el conjunto de reglas. El algoritmo de recubrimiento secuencial implementado en NSLV es el siguiente:

```
SequentialCovering(E: ExampleSet, f: fitnessClass): RuleClassSet
  var
    LearnedRules: RuleClassSet;
    Rule: RuleClass
  begin
    1. LearnedRules:= empty;
    2. Rule:= LearnOneRule(E,f);
    3. While (Performance(Rule,E) > 0) do
      3.1. LearnedRules:= LearnedRules + Rule;
      3.2. E:= Penalize(LearnedRules,E);
      3.3 Rule:= LearnOneRule(E,f);
    done
    4. return LearnedRules;
  end
```

Este algoritmo toma como entradas E, un conjunto de ejemplos y f, una función de evaluación de reglas, y ofrece como salida un conjunto de reglas que

tratan de describir el comportamiento de E. Inicialmente el conjunto de reglas está vacío (paso 1). En cada iteración se obtiene una nueva regla (paso 2 y 3.3) que se añade al conjunto de reglas aprendidas (paso 3.1) y se penalizan los ejemplos cubiertos para posibilitar el aprendizaje de nuevas reglas con el mismo o diferente consecuente (paso 3.2). El proceso se repite mientras que la función $Performance$ de la regla extraída sea mayor que cero (paso 3), siendo $Performance$ una función que mide el incremento de completitud que produce el incluir esta regla al conjunto de reglas.

LearnOneRule es la función que tiene como objetivo buscar la mejor regla y se implementa mediante un algoritmo genético de estado estacionario. La función multiobjetivo que guía la búsqueda es:

$$fitness(R_B(A)) = [\Psi(R_B(A)), svar(R_B(A)), sval(R_B(A))] \; . \tag{1}$$

en donde $R_B(A)$ es la regla con antecedente A y consecuente B, $\Psi(R_B(A))$ define una valoración de la regla en base a los conceptos de *consistencia* y *completitud*, $svar(R_B(A))$ y $sval(R_B(A))$ hacen referencia a la *simplicidad* y *comprensibilidad* de la regla respectivamente. Una descripción más detallada de todos los elementos de NSLV puede encontrarse en [4].

NSLV resuelve este problema multiobjetivo utilizando una función de evaluación lexicográfica, es decir, estos tres criterios se ordenan en función de su importancia siendo el orden de relevancia el que se muestra en la fórmula (1), siendo el primero el más importante.

Como acabamos de decir, $\Psi(R_B(A))$ es el criterio de mayor relevancia de la función de evaluación de NSLV. La idea subyacente de una buena regla en NSLV es aquella que cubre el mayor número de ejemplos de una clase cubriendo el menor número de ejemplos del resto de las clases. Para ello combinan los criterios de consistencia y completitud [4] de la siguiente forma:

$$\Psi(R_B(A)) = \frac{n_E^+(R_B(A)) - n_E^-(R_B(A))}{n_{E_B}} \; . \tag{2}$$

donde E es el conjunto de ejemplos, $n_E^+(R_B(A))$ y $n_E^-(R_B(A))$ son el número de ejemplos cubiertos positivamente y negativamente por la regla respectivamente, y n_{E_B} representa el número de ejemplos de la clase B existentes en E.

Las reglas difusas obtenidas de la ejecución del algoritmo NSLV son del tipo DNF, cuya expresión general es:

SI C_1 y C_2 y ... y C_m **ENTONCES** Y ES B con **peso w**

en donde la condición C_i podría ser:

- X_h es A, siendo A un valor de el dominio de la variable X_h, o
- X_h es $\hat{A}$, siendo $\hat{A}$ un subconjunto del dominio de la variable X_h

siendo $X = \{X_1, X_2, \ldots, X_n\}$ el conjunto de variables antecedente donde para cada variable se define su universo $U_1, U_2, \ldots, U_n$. Y es la variable consecuente

sobre el universo V, siendo B un subconjunto difuso sobre el dominio de la variable consecuente. Finalmente w es una medida de peso.

El número de ejemplos cubiertos positivamente por una regla $n_E^+(R_B(A))$ se define como:

$$n_E^+(R_B(A)) = \sum_{e \in \bigtriangledown_B^+} (U(e,A) \times U(e,B) \times \omega(R_B(A))) \tag{3}$$

donde $U(e,A)$ y $U(e,B)$ denota las adaptaciones del ejemplo al antecedente y consecuente respectivamente; $\omega(R_B(A))$ es el peso de la regla; y $\bigtriangledown_B^+$ representa el conjunto de ejemplos de la clase B que no estaban cubiertos con las reglas anteriores ($Rules$) pero están cubiertos de forma positiva con esta nueva regla. $\bigtriangledown_B^+$ se define como:

$$\bigtriangledown_B^+ = \{e \in E_B / e \notin covering^+(E_B, Rules) \wedge e \in covering^+(E_B, Rules \cup R_B(A)) \tag{4}$$

donde $covering^+(E, Rules)$ es el conjunto de ejemplos de E que están correctamente clasificados por $Rules$.

De forma similar el número de ejemplos cubiertos negativamente por una regla $n_E^-(R_B(A))$ se define como:

$$n_E^-(R_B(A)) = \sum_{e \in \bigtriangledown_B^-} (U(e,A) \times U(e,\overline{B}) \times \omega(R_B(A))) \tag{5}$$

donde $\bigtriangledown_B^-$ representa el conjunto de ejemplos de la clase B que pudiendo estar cubiertos o no por las reglas anteriores ($Rules$), están cubiertos de forma negativa con la introducción de esta nueva regla ($R_B(A)$). Este conjunto puede definirse de la forma:

$$\bigtriangledown_B^- = \{e \in E_B / e \in covering^-(E_B, Rules \cup R_B(A)) \ . \tag{6}$$

donde $covering^-(E, Rules)$ es el conjunto de ejemplos de E que se clasifican de forma incorrecta por $Rules$.

En la siguiente sección se describe una extensión de NSLV para el problema de la clasificación ordinal.

3 Un algoritmo de clasificación ordinal

NSLV ha demostrado ser un algoritmo de aprendizaje de reglas difusas muy competitivo para resolver problemas de clasificación nominal [4]. En este trabajo pretendemos extender este algoritmo a problemas de clasificación ordinal.

Desde un punto de vista abstracto podemos considerar que la clasificación ordinal es una clasificación nominal donde los errores cometidos en la clasificación tienen distinta magnitud dependiendo de la distancia a la clase correcta. Así, extender NSLV a clasificación ordinal siguiendo esta idea consistiría en redefinir

la ponderación de los errores que se cometen. Básicamente dicha extensión implica establecer una nueva definición del concepto de ejemplo negativo usado en NSLV.

En las siguientes subsecciones describiremos **NSLVOrd**, la extensión de NSLV al problema de clasificación ordinal. En la primera de ellas se propone una estrategia de recubrimiento secuencial adaptada a este problema, mientras que en la segunda se describe la nueva función fitness en la que se explota la potencialidad de la ordenación de las clases.

3.1 Modificación del algoritmo de recubrimiento secuencial

En primer lugar se ha modificado el algoritmo de recubrimiento secuencial en el sentido de explotar la característica de orden de la clase como se describe a continuación:

```
SeqCoveringNSLVOrd(E: ExampleSet, f: fitnessClass): RuleClassSet
  var
    LearnedRules: RuleClassSet;
    Rule: RuleClass;
    RemovedRules: boolean;
  begin
    1. RemovedRules= true;
    2. LearnedRules:= empty + SetDefaultRule(E,f);
    3. While (RemovedRules) do
      3.1. Rule:= LearnOneOrdRule(E,f);
       4. While (PerformanceOrd(Rule,E)) do
         4.1. LearnedRules:= LearnedRules + Rule;
         4.2. E:= Penalize(LearnedRules,E);
         4.3. Rule:= LearnOneOrdRule(E,f);
      done
      3.2. RemovedRules:= RemoveRules(LearnedRules,E,f);
    done
    4. return LearnedRules;
  end
```

Hay cuatro diferencias fundamentales entre esta propuesta de recubrimiento secuencial y la que se mostró en la sección anterior de NSLV.

1. Al principio del proceso (paso 2) se incluye en la base de reglas una regla por defecto. El valor del consecuente de esta regla es la clase mayoritaria.
2. La función *LearnOneOrdRule*, que se implementa mediante un algoritmo genético que tiene en consideración las peculiaridades de los problemas ordinales (pasos 3.1 y 4.3), y devuelve la mejor regla obtenida.
3. La función $PerformanceOrd$ (paso 4) toma el valor verdadero siempre que la última regla encontrada provoque una mejora sustancial en términos de consistencia y completitud sobre el conjunto de reglas.

4. Antes de finalizar el algoritmo (paso 3.2), se incluye un proceso de filtrado de reglas *RemoveRules*. Este proceso elimina del conjunto de reglas aquellas que no aportan en el ajuste de los ejemplos, indicando como resultado si se he producido eliminación de reglas o no.

3.2 Modificación de la función de evaluación

El objetivo que se persigue con la extensión del algoritmo NSLV es adaptarlo a las peculiaridades de los problemas de clasificación ordinal. Para ello proponemos modificar el criterio de mejor regla. Así, *LearnOneOrdRule* usa una función de evaluación multicriterio guiada por una función de evaluación lexicográfica e implementada mediante un algoritmo genético, al igual que en NSLV. La diferencia estriba en la necesidad de considerar distintos grados entre los errores. Por ejemplo, dado un problema con 5 clases ordenadas y un ejemplo i de la clase "2", el error cometido no debe ser considerado de igual forma si el ejemplo es clasificado de la clase "3" como si es clasificado de la clase "5".

En este sentido, proponemos dos cambios. El primero consiste en la inclusión de un criterio adicional para explotar esta característica de ordenación de las clases quedando dicha función de la forma:

$$fitness(R_B(A)) = [\Phi(R_B(A)), \Psi(R_B(A)), svar(R_B(A)), sval(R_B(A))] \; . \quad (7)$$

El nuevo criterio tiene en cuenta tanto los aciertos como la ponderación de los fallos cometidos por la regla que estamos analizando y se define como:

$$\Phi(R_B(A)) = \frac{\frac{n_E^+(R_B(A))}{n_E}(1 - \frac{n_E^-(R_B(A))}{n_E * (Q-1)})}{2} \quad (8)$$

donde n_E indica el número de ejemplos y Q el número de clases del problema.

El segundo cambio implica tener en cuenta los distintos grados de error, lo que hace necesario redefinir "el número de ejemplos cubiertos negativamente $(n_E^-(R_B(A)))$ para una regla" de la siguiente forma:

$$n_E^-(R_B(A)) = \sum_{e \in \bigtriangledown_B^-} (U(e, A) \times \widetilde{U(e, \overline{B})} \times \omega(R_B(A))) \; . \quad (9)$$

donde $\widetilde{U(e, \overline{B})} = (1 - U(e, B)) * |B - Class(e))|$ y $\bigtriangledown_B^-$ representa el subconjunto de ejemplos definido en (6) de la sección anterior.

Este segundo cambio implica una alteración en los valores obtenidos por $\Psi(R)$.

Una vez presentada la propuesta, en la sección siguiente mostramos un estudio experimental con el algoritmo desarrollado.

4 Resultados experimentales

En este estudio experimental utilizamos dos medidas para establecer la calidad desde el punto de vista de la clasificación: CCR que mide el acierto global, MS que mide el acierto en la clase peor clasificada. También usamos dos medidas para evaluar los resultados desde el punto de vista de la regresión: OMAE que mide la dispersión global del error y MMAE que mide el error de la clase peor ajustada.

En la tabla 1 se muestran las bases de datos pertenecientes a un repositorio de ejemplos de problemas de clasificación ordinal[1] utilizadas en la experimentación.

Cada base de datos está compuesta por 30 pares de ficheros "entrenamiento-prueba", lo que facilita la posible reproducción de los experimentos que aquí se presentan.

Nombre	Atributos (Num/Nom)	Ejemplos	Clases
automobile	25 (15/1)	4590	6
balance-scale	4 (4/0)	14040	3
bondrate	10 (4/6)	1260	5
contact-lenses	5 (0/4)	540	3
ERA	4 (4/0)	22500	9
ESL	4 (4/0)	10980	9
eucalyptus	19 (14/5)	16560	5
LEV	4 (4/0)	22500	5
newthyroid	5 (5/0)	4830	3
pasture	22 (21/1)	810	3
squash-stored	24 (21/3)	1170	3
squash-unstored	24 (21/3)	1170	3
SWD	10 (10/0)	22500	4
tae	5 (1/4)	3390	3
toy	2 (2/0)	6750	5
winequality-red	11 (11/0)	35970	6

Table 1. Bases de datos consideradas en las comparativas donde *Atributos(Num/Nom)* es el número de atributos totales, atributos numéricos y atributos nominales, *Ejemplos* es el número de ejemplos y *Clases* es el número de clases.

Para este estudio experimental usaremos 5 algoritmos, 3 de ellos son versiones del algoritmo SVM de la librería LIBSVM adaptadas a clasificación ordinal, SVC, C-RNK y SVOR-IM, otro es una implementación para clasificación ordinal de C4.5 llamado ASAOR, y el último es NSLVOrd la extensión de NSLV que se describe en este trabajo.

- SVC: *Cost Support Vector Classification* [7]. Es usado en clasificación nominal. Para tratar con multiclases utiliza la aproximación uno-contra-uno.
- C-RNK: es el kernel utilizado en SVM-Rank [8]. SVM-Rank es un caso particular de SVM para entrenamiento eficiente de ranking con SVM.
- SVOR-IM: *Support Vector Ordinal Regression with Implicit Constraints* [1], una caso particular de SVM con restricciones implicitas.

[1] http://www.uco.es/grupos/ayrna/datasets/datasets-orreview.zip

CCR Test	SVC	C-RNK	SVORIM	ASAOR	NSLVOrd
automobile - CCR	0,322 (3)	0,265 (4,5)	0,265 (4,5)	0,696 (2)	**0,747 (1)**
automobile - MS	0,000 (3,5)	0,000 (3,5)	0,000 (3,5)	0,000 (3,5)	**0,474 (1)**
automobile - OMAE	1,109 (5)	1,011 (3,5)	1,011 (3,5)	0,400 (2)	**0,362 (1)**
automobile - MMAE	3,000 (5)	2,967 (3,5)	2,967 (3,5)	1,088 (2)	**0,783 (1)**
balance-scale - CCR	0,895 (3)	**0,939 (1)**	0,938 (2)	0,757 (5)	0,760 (4)
balance-scale - MS	0,000 (5)	0,710 (2)	**0,712 (1)**	0,141 (3)	0,013 (4)
balance-scale - OMAE	0,126 (3)	**0,066 (1)**	0,067 (2)	0,300 (4)	0,359 (5)
balance-scale - MMAE	1,000 (5)	0,290 (2)	**0,287 (1)**	0,859 (3)	0,987 (4)
bondrate - CCR	0,578 (2)	0,578 (2)	0,578 (2)	0,533 (4)	0,484 (5)
bondrate - MS	0,000 (3)	0,000 (3)	0,000 (3)	0,000 (3)	0,000 (3)
bondrate - OMAE	0,624 (2)	0,624 (2)	0,624 (2)	0,624 (4)	0,680 (5)
bondrate - MMAE	2,900 (4)	2,900 (4)	2,900 (4)	2,367 (2)	**2,150 (1)**
contact-lenses - CCR	0,634 (3)	0,617 (5)	0,628 (4)	0,750 (2)	**0,756 (1)**
contact-lenses - MS	0,000 (4)	0,000 (4)	0,000 (4)	0,330 (2)	**0,456 (1)**
contact-lenses - OMAE	0,533 (5)	0,439 (4)	0,400 (2)	**0,367 (1)**	0,400 (3)
contact-lenses - MMAE	2,000 (5)	1,333 (4)	1,167 (3)	**0,878 (1)**	0,908 (2)
ERA - CCR	0,261 (2)	0,243 (5)	0,243 (4)	**0,262 (1)**	0,260 (3)
ERA - MS	**0,008 (1)**	0,000 (4)	0,000 (4)	0,000 (4)	0,004 (2)
ERA - OMAE	1,333 (5)	1,324 (4)	1,314 (2)	**1,222 (1)**	1,316 (3)
ERA - MMAE	2,158 (2)	2,282 (4)	2,290 (5)	2,160 (3)	**2,134 (1)**
ESL - CCR	**0,684 (1)**	0,663 (3)	0,665 (2)	0,640 (4)	0,625 (5)
ESL - MS	0,000 (3)	0,000 (3)	0,000 (3)	0,000 (3)	0,000 (3)
ESL - OMAE	**0,337 (1)**	0,372 (3)	0,365 (2)	0,384 (4)	0,423 (5)
ESL - MMAE	1,473 (2)	2,333 (5)	2,122 (4)	**1,361 (1)**	1,618 (3)
eucalyptus - CCR	0,343 (3)	0,238 (5)	0,238 (4)	**0,639 (1)**	0,545 (2)
eucalyptus - MS	0,032 (3)	0,007 (4,5)	0,007 (4,5)	**0,370 (1)**	0,343 (2)
eucalyptus - OMAE	1,200 (5)	1,082 (4)	1,082 (3)	**0,383 (1)**	0,566 (2)
eucalyptus - MMAE	2,640 (5)	1,808 (3)	1,810 (4)	**0,674 (1)**	0,818 (2)
LEV - CCR	0,629 (3)	**0,629 (1,5)**	**0,629 (1,5)**	0,613 (5)	0,623 (4)
LEV - MS	0,006 (2)	0,000 (4)	0,000 (4)	0,000 (4)	**0,021 (1)**
LEV - OMAE	**0,406 (1)**	0,407 (2,5)	0,407 (2,5)	0,424 (5)	0,412 (4)
LEV - MMAE	1,246 (4)	1,240 (2,5)	1,240 (2,5)	1,252 (5)	**1,192 (1)**
newthyroid - CCR	0,748 (3)	0,720 (4,5)	0,720 (4,5)	0,916 (2)	**0,953 (1)**
newthyroid - MS	0,000 (4)	0,000 (4)	0,000 (4)	0,771 (2)	**0,848 (1)**
newthyroid - OMAE	0,252 (3)	0,280 (4,5)	0,280 (4,5)	0,084 (2)	**0,048 (1)**
newthyroid - MMAE	1,000 (4)	1,000 (4)	1,000 (4)	0,229 (2)	**0,157 (1)**
pasture - CCR	0,311 (5)	0,333 (3,5)	0,333 (3,5)	**0,752 (1)**	0,745 (2)
pasture - MS	0,000 (4)	0,000 (4)	0,000 (4)	**0,500 (1)**	0,467 (2)
pasture - OMAE	0,867 (4)	1,000 (5)	0,667 (3)	**0,248 (1)**	0,263 (2)
pasture - MMAE	1,533 (4)	2,000 (5)	1,000 (3)	**0,500 (1)**	0,556 (2)
squash-stored - CCR	0,421 (4)	0,421 (4)	0,421 (4)	**0,602 (1,5)**	**0,602 (1,5)**
squash-stored - MS	0,000 (4)	0,000 (4)	0,000 (4)	**0,324 (1)**	0,306 (2)
squash-stored - OMAE	0,733 (4)	0,733 (4)	0,733 (4)	0,444 (2)	**0,421 (1)**
squash-stored - MMAE	2,000 (4)	2,000 (4)	2,000 (4)	0,888 (2)	**0,813 (1)**
squash-unstored - CCR	0,515 (3)	0,462 (4,5)	0,462 (4,5)	**0,774 (1)**	0,661 (2)
squash-unstored - MS	0,000 (4)	0,000 (4)	0,000 (4)	**0,456 (1)**	0,106 (2)
squash-unstored - OMAE	0,510 (3)	0,615 (5)	0,538 (4)	**0,239 (1)**	0,354 (2)
squash-unstored - MMAE	1,333 (4)	2,000 (5)	1,000 (3)	**0,561 (1)**	0,906 (2)
SWD - CCR	0,570 (4)	0,572 (2,5)	0,572 (2,5)	**0,574 (1)**	0,568 (5)
SWD - MS	0,000 (3,5)	0,000 (3,5)	0,000 (3,5)	0,000 (3,5)	**0,096 (1)**
SWD - OMAE	0,453 (4)	**0,442 (1,5)**	**0,442 (1,5)**	0,447 (3)	0,462 (5)
SWD - MMAE	1,083 (2)	1,088 (4)	1,088 (4)	1,088 (4)	**1,008 (1)**
tae - CCR	0,508 (3)	0,512 (2)	0,502 (4)	0,395 (5)	**0,566 (1)**
tae - MS	0,366 (2)	0,259 (3)	0,237 (4)	0,000 (5)	**0,442 (1)**
tae - OMAE	0,706 (5)	**0,523 (1)**	0,526 (2)	0,686 (4)	0,555 (3)
tae - MMAE	1,045 (4)	0,824 (2)	0,830 (3)	1,246 (5)	**0,740 (1)**
toy - CCR	0,334 (5)	0,565 (3)	0,559 (4)	0,694 (2)	**0,853 (1)**
toy - MS	0,000 (5)	0,253 (3)	0,247 (4)	0,284 (2)	**0,675 (1)**
toy - OMAE	0,884 (5)	0,435 (3)	0,441 (4)	0,356 (2)	**0,169 (1)**
toy - MMAE	1,744 (5)	0,747 (2)	0,753 (3)	0,878 (4)	**0,438 (1)**
winequality-red - CCR	0,575 (3)	0,562 (5)	0,563 (4)	0,603 (2)	**0,612 (1)**
winequality-red - MS	0,000 (3)	0,000 (3)	0,000 (3)	0,000 (3)	0,000 (3)
winequality-red - OMAE	0,486 (3)	0,489 (5)	0,488 (4)	**0,441 (1)**	0,456 (2)
winequality-red - MMAE	2,463 (3)	2,554 (5)	2,546 (4)	**2,107 (1)**	2,418 (2)
MEDIA - CCR	0,520 (3)	0,520 (4)	0,520 (5)	0,638 (2)	**0,647 (1)**
MEDIA - MS	0,026 (5)	0,077 (3)	0,075 (4)	0,199 (2)	**0,266 (1)**
MEDIA - OMAE	0,660 (5)	0,615 (4)	0,587 (3)	**0,441 (1)**	0,453 (2)
MEDIA - MMAE	1,789 (5)	1,710 (4)	1,562 (3)	1,133 (2)	**1,102 (1)**
RNK - CCR	3,125 (3)	3,500 (5)	3,438 (4)	**2,469 (1)**	**2,469 (1)**
RNK - MS	3,375 (3)	3,531 (4)	3,594 (5)	2,625 (2)	**1,875 (1)**
RNK - OMAE	3,625 (5)	3,312 (4)	2,875 (3)	**2,375 (1)**	2,812 (2)
RNK - MMAE	3,875 (5)	3,688 (4)	3,438 (3)	2,375 (2)	**1,625 (1)**

Table 2. CCR / MS / OMAE / MMAE Test

– ASAOR: *A Simple Approach to Ordinal Classification* [3]. Este método transforma un problema ordinal de k clases en k-1 problemas de dos clases. Para la predicción se estima la probabilidad de la clase ordinal k combinando los k-1 modelos aprendidos.

Los parámetros utilizados en NSLVOrd para este experimento han sido los parámetros por defecto de NSLV, exceptuando que el tamaño de la población genética se ha fijado a 20 por el número de clases. El resto de los algoritmos se han ejecutado utilizando los parámetros por defecto definidos por sus autores.

En la tabla 2 se muestran los resultados obtenidos sobre las cuatro medidas, las 16 bases de datos y los 5 algoritmos. Cada valor representa la media sobre los 30 conjuntos de test que hay por cada base de datos.

Para realizar el análisis del comportamiento de más de un algoritmo se aconseja utilizar el test de Shaffer [11] tomando un valor de $\alpha = 0,05$ o $\alpha = 0,10$. En este test, para comprobar si un algoritmo es significativamente mejor que otro se tiene que comprobar si se acepta o rechaza la hipótesis nula de igualdad de medias. Esta hipótesis se rechaza si el $p - value$ calculado por este test es menor o igual que el valor α considerado. Con estos parámetros, según se puede observar en la tabla 3 podemos deducir que,

– con respecto a CCR y OMAE no aparecen diferencias entre los algoritmos con $\alpha = 0,05$, observándose un empate en primer puesto en CCR entre NSLVOrd y ASAOR, y siendo NSLVOrd el primero respecto a OMAE, y
– con respecto a MS y MMAE, NSLVOrd obtiene el mejor resultado siendo mejor significativamente que SVC, C-RNK y SVORIM.

Estos resultados muestran que la versión propuesta presenta una capacidad de ajuste global, desde el punto de vista de clasificación (CCR) como de regresión (OMAE) semejante a la que ofrecen otros métodos conocidos en este tipo de problemas, pero mejora con respecto a ellos la clasificación/regresión (MS/MMAE) con relación a la peor clase.

hypothesis	$p_{Shaf_{CCR}}$	$p_{Shaf_{MS}}$	$p_{Shaf_{OMAE}}$	$p_{Shaf_{MMAE}}$
NSLVOrd vs ASAOR	2.00934	0.71885	2.22656	0.71885
NSLVOrd vs SVORIM	0.65073	**0.021079**	2.22656	**0.007114**
NSLVOrd vs C-RNK	0.65073	**0.021079**	2.22656	**0.001348**
NSLVOrd vs SVC	1.442525	**0.043742**	0.876603	**0.00057**

Table 3. Resultados del test de Shaffer de NSVLOrd vs resto. En negrita se indica la diferencia significativa.

5 Conclusiones

En este trabajo se ha presentado NSLVOrd, un algoritmo para clasificación ordinal basado en la técnica de recubrimiento secuencial que expresa el conocimiento mediante un conjunto de reglas difusas y tiene como mecanismo de búsqueda un algoritmo genético.

NSLVOrd es una extensión para clasificación ordinal de NSLV. Dicha extensión implica una redefinición del concepto de ejemplo negativo propuesto en NSLV, lo que define una función objetivo para el algoritmo genético.

Para comparar el comportamiento de la propuesta, se ha presentado un estudio experimental en el que se han incluido 4 algoritmos conocidos para clasificación ordinal. Los resultados muestran que NSLVOrd presenta un buen comportamiento en todos los parámetros estudiados, mejorando en relación a los algoritmos incluidos el ajuste por clase.

Acknowledgments

Este trabajo ha sido parcialmente financiado por el Ministerio de Economía y Competitividad bajo el proyecto TIN2012-38969 y cofinanciado por los fondos FEDER de la Unión Europea.

References

1. Sathiya Keerthi Chu wei. Support vector ordinal regression. *Neural computation*, 19(3):792–815, 2007.
2. Manuel Cruz-Ramírez, César Hervás-Martínez, Javier Sánchez-Monedero, and Pedro Antonio Gutiérrez. Metrics to guide a multi-objective evolutionary algorithm for ordinal classification. *Neurocomputing*, 135:21–31, 2014.
3. Eibe Frank and Mark Hall. A simple approach to ordinal classification. *ECML 2001*, page 145, 2001.
4. David García, Antonio González, and Raúl Pérez. Overview of the SLAVE learning algorithm: A review of its evolution and prospects. *International Journal of Computational Intelligence Systems*, 7(6):1194–1221, 2014.
5. A. González and R. Pérez. Improving the genetic algorithm of SLAVE. *Mathware & Soft Computing*, 16(1):59–70, 2009.
6. Pedro Antonio Gutiérrez, Sancho Salcedo-Sanz, César Hervás-Martínez, Leopoldo Carro-Calvo, Javier Sánchez-Monedero, and Luis Prieto. Ordinal and nominal classification of wind speed from synoptic pressurepatterns. *Engineering Applications of Artificial Intelligence*, 26(3):1008–1015, 2013.
7. Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002.
8. Tzu-Ming Kuo, Ching-Pei Lee, and Chih-Jen Lin. *Large-scale Kernel RankSVM*, chapter 91, pages 812–820. 2014.
9. Tjen-Sien Lim, Wei-Yin Loh, and Yu-Shan Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine learning*, 40(3):203–228, 2000.
10. Ivy Liu and Alan Agresti. The analysis of ordered categorical data: An overview and a survey of recent developments. *Test*, 14(1):1–73, 2005.
11. Juliet Popper Shaffer. Modified sequentially rejective multiple test procedures. *Journal of the American Statistical Association*, 81(395):826, 1986.
12. Bing-Yu Sun, Jiuyong Li, D.D. Wu, Xiao-Ming Zhang, and Wen-Bo Li. Kernel discriminant learning for ordinal regression. *Knowledge and Data Engineering, IEEE Transactions on*, 22(6):906–910, June 2010.