Algoritmo evolutivo para optimizar ensembles de clasificadores multi-etiqueta

Jose M. Moyano, Eva L. Gibaja, Alberto Cano, Jose M. Luna, Sebastián Ventura

Resumen— **En este artículo se propone un método evolutivo para el diseño automático de *ensembles* en aprendizaje multi-etiqueta. En este modelo, cada uno de los elementos base son clasificadores multi-etiqueta donde cada uno está especializado en un subconjunto de etiquetas. De esta forma se reduce la complejidad del clasificador en problemas en los que el número de etiquetas es elevado. Los resultados de los experimentos que se han llevado a cabo sobre un amplio conjunto de *datasets* de referencia indican que existen diferencias significativas respecto RA*k*EL, su competidor más directo.**

Palabras clave—**clasificación, multi-etiqueta, algoritmo evolutivo, ensemble**

I. Introducción

El aprendizaje multi-etiqueta se caracteriza, a diferencia del aprendizaje supervisado clásico, en que un patrón puede tener asociadas varias clases o etiquetas de manera simultánea, de modo que no se cumple la restricción clásica de *una etiqueta por patrón*. Así, dado un conjunto de patrones de entrenamiento d-dimensional $\mathcal{X} = X_1 \times \cdots \times X_d$ donde cada patrón tiene asociada una o varias etiquetas binarias $Y \subseteq \mathcal{L} = \{\lambda_1, \lambda_2, ..., \lambda_q\}, q > 1$, se pretende obtener un modelo capaz de predecir, para cada nuevo patrón desconocido, su conjunto de etiquetas asociado, basándose en los atributos del conjunto de datos. Este modelo de aprendizaje supervisado se ha utilizado con éxito en tareas como la clasificación de textos o multimedia (ej. una imagen puede tener asociadas varias etiquetas a la vez) [1], [2], en biología (ej. predicción de la función de proteínas y genes) [3], [4] o problemas como el marketing directo (ej. a un cliente potencial es útil recomendarle varios productos) [5] entre otros.

El hecho de considerar múltiples salidas y que el espacio de etiquetas sea generalmente de gran dimensionalidad hace necesario desarrollar modelos que no sean costosos ni en memoria ni en tiempo de predicción. Una de las técnicas más utilizadas para este fin consiste en desarrollar *ensembles*, que combinan varios clasificadores base débiles. La idea es elevar el nivel de acierto del clasificador bajo la suposición de que para que una predicción sea errónea, ha debido ser errónea en varios clasificadores base. Particularmente, en aprendizaje multi-etiqueta los clasificadores base son también clasificadores multi-etiqueta.

Se han desarrollado ya modelos de clasificación multi-etiqueta basados en *ensembles* [6]. Uno de los más utilizados es RA*k*EL [7], basado en proyecciones aleatorias del espacio de etiquetas. Así, RA*k*EL crea un *ensemble* donde cada uno de sus clasificadores base tiene en cuenta solo un subconjunto aleatorio de k etiquetas binarias, y la decisión final vendrá dada por la votación de los clasificadores base. De este modo, tiene en cuenta las relaciones entre etiquetas con un coste computacional relativamente bajo. La selección aleatoria de subconjuntos de etiquetas que realiza RA*k*EL puede producir una reducción del rendimiento del clasificador. Para evitar este efecto, en este trabajo proponemos diseñar automáticamente el *ensemble*, mediante algoritmos evolutivos, teniendo en cuenta tanto las combinaciones de etiquetas de los clasificadores base como qué clasificadores base lo formarán. El rendimiento del modelo será evaluado utilizando un amplio conjunto de *datasets* y métricas de referencia y en este primer estudio se comparará con RA*k*EL, pues antes de compararlo con un conjunto más amplio de algoritmos, debemos saber si supera a su competidor más directo.

El resto del artículo está organizado de las siguiente manera: la Sección 2 incluye los antecedentes, la Sección 3 muestra el funcionamiento del algoritmo, en la Sección 4 se presenta el conjunto de experimentos realizados, y en la Sección 5 los resultados y discusión de los mismos. Finalmente, la Sección 6 presenta las conclusiones del trabajo y las futuras mejoras.

II. Antecedentes

En este apartado se describen los distintos tipos de algoritmos para clasificación multi-etiqueta existentes y las métricas de evaluación para aprendizaje multi-etiqueta.

A. Algoritmos de clasificación multi-etiqueta

Los algoritmos de aprendizaje multi-etiqueta se suelen categorizar en: transformación de problemas, adaptación de algoritmos y *ensembles* de clasificadores multi-etiqueta [8].

Los métodos de transformación de problemas se basan en transformar un problema multi-etiqueta en uno o varios problemas multi-clase, que se resuelven con algoritmos de clasificación clásicos. Uno de los métodos más conocidos es *Binary Relevance* (BR) [9], que genera un clasificador binario para cada etiqueta, donde los patrones positivos serán los que tengan asociada dicha etiqueta y los negativos el resto. Dado un patrón desconocido, se etiquetará teniendo en cuenta la salida para cada uno de estos clasificado-

Departamento de Informática y Análisis Numérico. Universidad de Córdoba. E-mail: sventura@uco.es.

res binarios. Este método es eficiente, pero trata las etiquetas de modo independiente. El algoritmo *Label Powerset* (LP) [10] convierte el problema multi-etiqueta en otro de clasificación multi-clase, creando una clase por cada combinación de etiquetas posible. Este método tiene en cuenta las combinaciones de etiquetas, pero su complejidad crece exponencialmente con el número de etiquetas. Por otro lado, *Classifier Chain* (CC) [11], genera q clasificadores binarios encadenados de tal manera que cada clasificador incluye como entradas las etiquetas predichas por los clasificadores anteriores.

La adaptación de algoritmos consiste en extender algoritmos clásicos para trabajar directamente en clasificación multi-etiqueta. Se han adaptado prácticamente todos los tipos de técnicas de clasificación, por ejemplo, árboles de decisión [12], redes neuronales [13] o algoritmos basados en instancias [14], [15].

Existe un tercer grupo de métodos, denominados *ensembles* de clasificadores multi-etiqueta, cuyos clasificadores base son multi-etiqueta. RAkEL (*RAndom k-labELsets*) [7], como se comentó en la sección anterior, construye un *ensemble* de clasificadores LP, donde cada uno entrena únicamente un subconjunto aleatorio de etiquetas, y la decisión final vendrá dada por votación de las decisiones de cada clasificador base. Por otro lado, ECC (*Ensemble of Classifier Chains*) [16] entrena un conjunto de clasificadores CC, cada uno con un orden de encadenamiento aleatorio y una muestra aleatoria con reemplazo de patrones de entrenamiento. Por último, EBR (*Ensemble of BR classifiers*) [16] es un *ensemble* de clasificadores BR entrenados con una muestra de patrones obtenida igual que en ECC.

B. Métricas de evaluación

La evaluación de los clasificadores multi-etiqueta se debe hacer de manera diferente a como se hace en los problemas de clasificación tradicionales, pues hay que tener en cuenta todas las etiquetas. Por tanto, la predicción de un patrón puede ser totalmente correcta (se predicen todas las etiquetas), parcialmente correcta (se predicen algunas etiquetas) o completamente incorrecta (ninguna de las etiquetas que realmente son positivas son predichas). Estas métricas se clasifican en métricas basadas en etiquetas y métricas basadas en ejemplos [9].

B.1 Métricas basadas en etiquetas

Cualquier métrica de evaluación para clasificación binaria se puede calcular por el enfoque basado en etiquetas, como *precision*, *recall*, *specificity* o *F-measure*. La idea es calcularla basándose en el número de *true positives* (tp), *true negatives* (tn), *false positives* (fp) y *false negatives* (fn) de las matrices de confusión de todas las etiquetas (Figura 1).

Dada una métrica de evaluación binaria, la aproximación micro primero une todas las matrices de confusión y luego calcula el valor para la métrica,

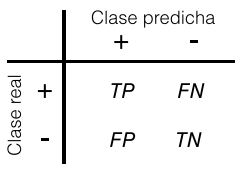

Fig. 1. Matriz de confusión.

mientras que la aproximación macro se obtiene calculando su valor para cada etiqueta y luego promediando entre el número de etiquetas [17].

En la Tabla I se definen las aproximaciones macro y micro de las distintas métricas. Se define *recall* como la fracción de patrones predichos correctamente como positivos del conjunto de patrones realmente positivos y *precision* como los patrones realmente positivos de todos los predichos como positivos. El *F-measure* es una combinación de ambas métricas, y por último, *specificity* indica los patrones correctamente predichos como negativos de los que realmente son negativos. También se incluye el área bajo la curva ROC (AUC), siendo la curva ROC una representación del ratio de *tp* frente a los *fn* de un clasificador binario, dependiendo de diferentes valores de umbral, siendo el AUC el área bajo esta curva.

B.2 Métricas basadas en ejemplos

Las métricas basadas en etiquetas ignoran la relación entre dichas etiquetas, en cambio, las métricas basadas en ejemplos o instancias, calculan la métrica para cada patrón o instancia, y luego promedia entre el número de instancias totales. Estas métricas pueden evaluar tanto biparticiones como *rankings* de etiquetas. Para las notaciones posteriores, se tendrá en cuenta que t es el número de instancias del conjunto de *test*, Y_i y Z_i el conjunto de etiquetas reales y predichas, respectivamente, τ el *ranking* de etiquetas predicho, Δ indica la diferencia simétrica entre dos conjuntos, y para un predicado π, $[\![\pi]\!]$ es 1 si el predicado es verdadero y 0 en caso contrario.

En la Tabla II se muestran las fórmulas para calcular estas métricas. *Hamming loss* [18] evalúa cuantas veces, en media, una etiqueta no se predice correctamente, teniendo en cuenta tanto los errores de predicción (se predice una etiqueta incorrecta) como los errores de omisión (una etiqueta correcta no se predice), normalizando sobre el número total de etiquetas y de instancias. Mide el error, por tanto, será una métrica a minimizar. Por otro lado, *subset accuracy* [19] indica el porcentaje de instancias cuyas etiquetas predichas son exactamente las mismas que las reales. Es una métrica muy estricta, ya que necesita que coincidan todas las etiquetas predichas con las etiquetas que son positivas realmente en el patrón. En cuanto a la evaluación de *rankings* de etiquetas, cabe destacar *average precision* [20] que mide la proporción de etiquetas reales que tienen mejor *ranking* que otra etiqueta de Y. Es una métrica a maximizar, siendo perfecto el rendimiento cuando su valor es 1.

TABLA I

MÉTRICAS BASADAS EN ETIQUETAS.

	Micro	Macro
↑ Recall	$\frac{\sum_{i=1}^{q} tp_i}{\sum_{i=1}^{q} tp_i + \sum_{i=1}^{q} fn_i}$	$\frac{1}{q}\sum_{i=1}^{q} \frac{tp_i}{tp_i+fn_i}$
↑ Precision	$\frac{\sum_{i=1}^{q} tp_i}{\sum_{i=1}^{q} tp_i + \sum_{i=1}^{q} fp_i}$	$\frac{1}{q}\sum_{i=1}^{q} \frac{tp_i}{tp_i+fp_i}$
↑ F-measure	$2 \times \frac{precision_{mic} \times recall_{mic}}{precision_{mic}+recall_{mic}}$	$2 \times \frac{precision_{mac} \times recall_{mac}}{precision_{mac}+recall_{mac}}$
↑ Specificity	$\frac{\sum_{i=1}^{q} tn_i}{\sum_{i=1}^{q} tn_i + \sum_{i=1}^{q} fp_i}$	$\frac{1}{q}\sum_{i=1}^{q} \frac{tn_i}{tn_i+fp_i}$

TABLA II

MÉTRICAS BASADAS EN EJEMPLOS.

↓ Hamming loss	$\frac{1}{t}\sum_{i=1}^{t} \frac{1}{q} \lvert Z_i \Delta Y_i \rvert$
↑ Subset accuracy	$\frac{1}{t}\sum_{i=1}^{t} [\![Z_i = Y_i]\!]$
↑ Average precision	$\frac{1}{t}\sum_{i=1}^{t} \frac{1}{\lvert Y_i \rvert} \sum_{\lambda \in Y_i} \frac{\lvert \{\lambda' \in Y_i \lvert \tau_i(\lambda') \leq \tau_i(\lambda)\} \rvert}{\tau_i(\lambda)}$

III. ALGORITMO

En este apartado se presentará el algoritmo desarrollado, centrándonos en la exposición de su esquema de codificación, operadores genéticos y función de *fitness* utilizados. Por último, se detallan algunos detalles de implementación orientados a optimizar el rendimiento del algoritmo.

A. Algoritmo evolutivo

El algoritmo evolutivo está basado en un algoritmo generacional con elitismo, donde cada generación estará compuesta por todos los nuevos individuos, y si el mejor de los padres es mejor a todos los nuevos individuos, reemplazará al peor de los hijos. Finalmente, se obtiene el mejor individuo, es decir, la mejor solución encontrada durante la ejecución del algoritmo. En el Algoritmo 1 se detallan los pasos del algoritmo evolutivo.

Algoritmo 1 Pasos del algoritmo evolutivo.

1: Generar una población inicial aleatoria de N individuos
2: Crear los *ensembles* correspondientes a cada individuo
3: Evaluar los individuos de la población inicial
4: **para** G generaciones **hacer**
5: Seleccionar individuos y aplicar operadores de cruce y mutación
6: Generar los *ensembles* correspondientes a los nuevos individuos
7: Evaluar los nuevos individuos, seleccionar la nueva población y reemplazarla por la anterior
8: **fin para**

B. Individuos

Cada individuo representa un *ensemble* completo, formado por n clasificadores base, que son también clasificadores multi-etiqueta. Todos estos clasificadores base multi-etiqueta serán del mismo tipo, pero cada uno clasificará un subconjunto pequeño de k etiquetas (el valor de k se fija en los parámetros y es el mismo para todos los clasificadores base), al estilo del clasificador RAkEL. Así, los individuos serán vectores binarios de tamaño $n \times q$, siendo q el número de etiquetas del problema. Los individuos de la población inicial se generan seleccionando k bits aleatorios a 1 para cada uno de los clasificadores base. En la Figura 2 se muestra el genotipo de un individuo de la población (representado como matriz para una visualización más intuitiva). Cada fila representa un clasificador base del *ensemble*. Cada uno de los bits indica qué etiquetas tendrá en cuenta cada clasificador base, siendo los bits a 1 las etiquetas que sí tendrá en cuenta y los bits a 0 las que no. Por ejemplo, el MLL_1 clasificaría las etiquetas λ_1, λ_2 y λ_5.

	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
MLL_1	1	1	0	0	1	0
MLL_2	0	0	1	1	0	1
MLL_3	1	0	1	0	0	1
MLL_4	1	0	1	1	0	0
MLL_5	0	1	0	1	1	0
MLL_6	1	0	0	0	1	1
MLL_7	0	1	1	1	0	0
MLL_8	0	1	0	1	0	1

Fig. 2. Ejemplo del individuo representado como matriz.

Para evaluar un individuo hay que obtener su fenotipo a partir del genotipo. Se construye el *ensemble* de clasificadores multi-etiqueta teniendo en cuenta que, cada clasificador base estará especializado únicamente en un subconjunto de k etiquetas. Una vez generado el *ensemble*, se calculará el valor de la función de *fitness* descrita en el apartado D, teniendo en cuenta que la predicción para cada patrón desconocido se realizará mediante la votación de los clasifica-

dores base del *ensemble*. En la Figura 3, se muestra un ejemplo del proceso de votación en el que cada clasificador base proporciona la predicción para un subconjunto de etiquetas. Por ejemplo, el clasificador MLL1 que estaba especializado en λ_1, λ_2 y λ_5 predice para estas etiquetas 1, 0 y 0 respectivamente. Finalmente, si la suma de votos totales para una determinada etiqueta supera cierto umbral (en este caso 0.5, que representa que esa etiqueta ha sido asociada al patrón por al menos la mitad de los clasificadores base), la salida del *ensemble* será que ese patrón está etiquetado con dicha etiqueta. En nuestro ejemplo, la etiqueta λ_2 obtuvo solo un voto de los cuatro posibles, por lo que la salida será negativa, mientras que la etiqueta λ_1 obtuvo 4 votos de 4 positivos, siendo la salida positiva.

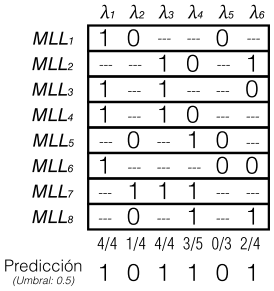

Fig. 3. Proceso de votación del *ensemble*.

C. Operadores genéticos

C.1 Operador de cruce

Para cada individuo de la población se determina en base a una probabilidad de cruce si se incluye en el conjunto de padres de la generación actual para aplicarles el cruce. De este conjunto se escogen pares de individuos aleatorios para cruzarlos. El operador de cruce intercambia dos clasificadores base del *ensemble* seleccionados al azar entre dos individuos (Figura 4). Así, se buscan nuevas combinaciones de clasificadores base con los que se puedan obtener mejores resultados.

Los individuos generados al cruzar serán siempre válidos, ya que el número de bits activos (k) siempre será el mismo.

C.2 Operador de mutación

Para determinar qué individuos se mutarán, a cada individuo se le aplica una probabilidad de mutación en base a la cual se determinará los individuos a los que se les aplicará este operador.

El operador de mutación selecciona al azar dos bits distintos en cada clasificador base del *ensemble* y los intercambia (Figura 5). Así, cada clasificador base dejará de clasificar una de las etiquetas para clasificar otra. De este modo se buscan nuevas combinaciones de etiquetas para un clasificador base, buscando un mejor rendimiento del mismo y por tanto, un posible mejor rendimiento del *ensemble*.

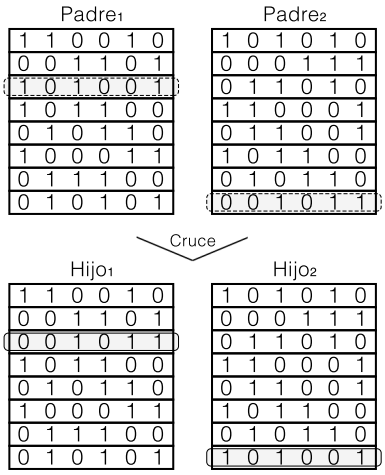

Fig. 4. Operador de cruce.

Al intercambiarse dos bits distintos dentro del mismo clasificador base, todos los individuos mutados serán válidos, pues el número de bits activos (k) no varía.

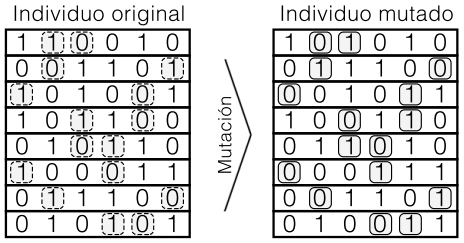

Fig. 5. Operador de mutación.

D. Función de fitness

La función de *fitness* en base a la que se evaluarán los individuos será el *macro-averaged F-measure*. Se ha escogido esta métrica porque representa una combinación de *precision* y *recall*, y porque la aproximación macro da el mismo peso a todas las etiquetas, por tanto está más influenciada por categorías raras.

E. Optimización

Dentro de la ejecución del algoritmo evolutivo, la función más costosa computacionalmente es la evaluación de los individuos. Además, puesto que el algoritmo necesita del entrenamiento y evaluación de muchos clasificadores, se usarán dos estructuras que faciliten la optimización en tiempo del algoritmo.

En primer lugar, se utilizará una estructura que almacene los clasificadores base entrenados a lo largo de la ejecución del algoritmo, pues con alta probabilidad habrá clasificadores base dentro de los *ensembles* que se repitan entre los distintos individuos de la población. Así, al construir un *ensemble* a partir del genotipo de un individuo, antes de crear y entrenar un nuevo clasificador base, se consulta en la tabla si ya se ha entrenado con anterioridad, obteniéndolo en caso de ser posible. Almacenando todos

estos clasificadores base, se ahorra el tiempo que se invertiría en entrenar estos modelos repetidos. Por otro lado, se incluye otra estructura, que almacena para cada individuo su valor de *fitness*, ya que de una generación a otra puede haber individuos que no se modifiquen, o que con el paso de las generaciones se repitan individuos, con lo que se evita tener que evaluar al individuo completo de nuevo.

IV. Experimentación

La finalidad del proceso de experimentación es comparar nuestro algoritmo con RAkEl, que genera el *ensemble* de manera aleatoria.

A. Datasets

Los experimentos se realizarán sobre un conjunto de 10 *datasets* de referencia en clasificación multi-etiqueta de varios dominios, como clasificación de imágenes y sonidos, biología y categorización de textos. Los *datasets* utilizados son: *emotions* [21], *birds* [22], *yeast* [23], *scene* [10], *plant* [24], *human* [24], *genbase* [25], *medical* [26], *slashdot* [27] y *enron* [28].

En la Tabla III se muestran características de todos los *datasets* utilizados. Se incluye el dominio, el número de patrones (p), número de etiquetas (q), número de atributos (d), cardinalidad, densidad y el número de combinaciones de etiquetas distintas presentes (*distinct*) [9]. Se define la cardinalidad como el número medio de etiquetas asociadas a cada patrón, y la densidad como la cardinalidad entre el número de etiquetas. En la tabla se encuentran ordenados por complejidad ($p \times q \times d$) [27], y se puede observar que varían en cuanto a complejidad, número de etiquetas, características y patrones.

B. Sección experimental

La implementación del algoritmo se ha realizado en Java, haciendo uso de las librerías JCLEC [29] y Mulan [30]. JCLEC es una librería para computación evolutiva, que proporciona un entorno de alto nivel para realizar cualquier algoritmo evolutivo, dando soporte para algoritmos genéticos (codificación binaria, entera y real), programación genética y programación evolutiva. Mulan es una librería para aprendizaje multi-etiqueta, que proporciona una API para los usuarios, que permite usar sus funcionalidades desde el propio código Java.

Para la realización de los experimentos se han dividido los *datasets* en 5 particiones distintas en *5-folds*, y los algoritmos se ejecutarán con 5 semillas distintas, realizando así 125 ejecuciones sobre cada *dataset*. Ambos algoritmos se ejecutarán con las mismas particiones y mismas semillas aleatorias, para poder comparar su rendimiento y ver si hay diferencias significativas entre ambos. Las métricas utilizadas son las descritas anteriormente en el apartado B de la sección II.

La configuración recomendada para RAkEL es LP con J48 como clasificador base, número de etiquetas activas $k = 3$, el número de clasificadores base del *ensemble* será $2q$, y el umbral para la votación 0.5 [7]. Los parámetros del algoritmo evolutivo son los que se resumen en la Tabla IV. Para los parámetros comunes con RAkEL se utilizarán los mismos valores, para así poder compararlos en las mismas condiciones.

TABLA IV
Parámetros del algoritmo evolutivo.

Tamaño de la población	100
Número máximo de generaciones	50
Probabilidad de cruce	0.9
Probabilidad de mutación	0.2
Número de clasificadores base	$2q$
Tipo de clasificador base	LP(J48)
Número de etiquetas activas (k)	3
Umbral para la votación	0.5

V. Resultados y discusión

En la Tabla V se muestran los resultados obtenidos tras la ejecución de los algoritmos, para todos los *datasets* (ordenados por complejidad) y distintas métricas evaluadas.

En primer lugar, nos fijamos en las métricas basadas en ejemplos. Teniendo en cuenta que el *Hamming loss* es una métrica a minimizar, vemos como nuestro algoritmo tiene mejor rendimiento que RAkEL en 8 de los 10 *datasets*. Observamos también como para *subset accuracy*, en la mayoría de los *datasets* nuestro algoritmo supera a RAkEL, haciéndolo con mayor claridad en los más complejos como *genbase*, *medical*, *slashdot* o *enron*. Esta superioridad en los *datasets* más complejos se produce en la mayoría de los casos. En cuanto a *average precision*, nuestro algoritmo tiene mejor rendimiento en 8 de los 10 *datasets*.

En el caso de las métricas basadas en etiquetas, en las aproximaciones macro y micro de cada métrica nuestro algoritmo supera a RAkEL en los mismos *datasets*. Para *precision* lo hace en 8 ocasiones, para *recall* en 4 y para *F-measure* en 6 de los *datasets*, en todos incluyendo los *datasets* más complejos. Por otro lado, aunque el rendimiento entre ambos está más igualado, para *specificity* nuestro algoritmo supera a RAkEL en 7 *datasets*. Por último, para *micro-averaged AUC*, nuestro algoritmo tiene un mejor rendimiento en 8 de los 10 *datasets*, incluyendo de nuevo los más complejos.

Una vez tenemos los resultados, y viendo que en la mayoría de los casos el rendimiento de nuestro algoritmo está por encima de RAkEL, queremos saber si realmente las diferencias en rendimiento son significativas. Para ello realizaremos un test de Wilcoxon, que es un test no paramétrico recomendado en el estudio de Demšar [31], que nos permite saber si hay diferencias significativas entre el rendimiento de dos muestras. La hipótesis nula de este test indica que

TABLA III

Características de los conjuntos de datos utilizados.

Dataset	Dominio	p	q	d	Cardinalidad	Densidad	*Distinct*
Emotions	Audio	593	6	72	1.869	0.311	27
Birds	Audio	645	12	260	1.014	0.053	133
Yeast	Biología	2417	14	103	4.237	0.303	198
Scene	Imágenes	2407	6	294	1.074	0.179	15
Plant	Biología	948	12	440	1.080	0.089	32
Human	Biología	3108	14	440	1.190	0.084	85
Genbase	Biología	662	27	1186	1.252	0.046	32
Medical	Texto	978	45	1449	1.245	0.028	94
Slashdot	Texto	3782	22	1079	1.180	0.053	156
Enron	Texto	1702	53	1001	3.378	0.064	753

TABLA V

Resultados de los experimentos.

		Emotions	**Birds**	**Yeast**	**Scene**	**Plant**	**Human**	**Genbase**	**Medical**	**Slashdot**	**Enron**
Hamming loss	Evolutivo	0.22583	**0.04723**	**0.21884**	0.10886	**0.10395**	**0.09358**	**0.00099**	**0.01003**	**0.04129**	**0.04800**
	RAkEL	**0.21949**	0.05572	0.26998	**0.10310**	0.13418	0.12453	0.01128	0.01861	0.04414	0.05736
Subset accuracy	Evolutivo	0.24129	**0.49829**	**0.11707**	0.53766	**0.12118**	**0.15302**	**0.97479**	**0.66567**	**0.31822**	**0.12427**
	RAkEL	**0.26028**	0.46425	0.05276	**0.56698**	0.12036	0.12628	0.70936	0.33775	0.26430	0.04695
Average Precision	Evolutivo	0.76121	**0.55892**	**0.71556**	0.81946	**0.47292**	**0.52242**	**0.99169**	**0.83553**	**0.57295**	**0.62744**
	RAkEL	**0.77212**	0.43977	0.64163	**0.83450**	0.42387	0.46934	0.78727	0.48676	0.48580	0.35092
Recall$_{\text{micro}}$	Evolutivo	0,59744	0,32426	0,56903	0,62128	0,15587	0,22297	**0,98564**	**0,79840**	**0,35775**	**0,49208**
	RAkEL	**0,64219**	**0,34509**	**0,58082**	**0,66312**	**0,25302**	**0,35262**	0,76398	0,41513	0,30085	0,30573
Precision$_{\text{micro}}$	Evolutivo	0,64980	**0,60910**	**0,66090**	0,73118	**0,33613**	**0,40650**	**0,99292**	**0,83297**	**0,73828**	**0,66791**
	RAkEL	**0,65020**	0,47257	0,55103	**0,73542**	0,25443	0,30039	0,99050	0,81980	0,70640	0,60388
F-measure$_{\text{micro}}$	Evolutivo	0,62215	**0,42240**	**0,61145**	0,67148	0,21183	0,28711	**0,98922**	**0,81495**	**0,48181**	**0,56652**
	RAkEL	**0,64554**	0,39665	0,56400	**0,69714**	**0,25135**	**0,32235**	0,86104	0,54158	0,41505	0,40110
Specificity$_{\text{micro}}$	Evolutivo	**0,85409**	**0,98821**	**0,87322**	**0,94998**	**0,96916**	**0,96963**	**0,99965**	0,99542	0,99280	0,98332
	RAkEL	0,84308	0,97807	0,79478	0,94788	0,92634	0,92382	0,99964	**0,99752**	**0,99301**	**0,98600**
Recall$_{\text{macro}}$	Evolutivo	0.58731	0.25621	0.36583	0.63424	0.08698	0.09560	**0.93611**	**0.65618**	**0.32436**	**0.21570**
	RAkEL	**0.63172**	**0.28306**	**0.41200**	**0.67337**	**0.15921**	**0.17006**	0.75570	0.48978	0.28337	0.15711
Precision$_{\text{macro}}$	Evolutivo	0.64386	**0.43045**	**0.49162**	0.73746	**0.18982**	**0.20836**	**0.93168**	**0.66319**	**0.54802**	**0.30176**
	RAkEL	**0.64534**	0.34638	0.37093	**0.74805**	0.17430	0.15166	0.75080	0.49268	0.44903	0.17646
F-measure$_{\text{macro}}$	Evolutivo	0.60860	**0.29934**	**0.39248**	0.67911	0.11152	0.12070	**0.93293**	**0.65133**	**0.36936**	**0.23473**
	RAkEL	**0.63145**	0.29070	0.37778	**0.70557**	**0.14958**	**0.14666**	0.75230	0.48667	0.31882	0.16075
Specificity$_{\text{macro}}$	Evolutivo	**0.84979**	**0.98783**	**0.79191**	**0.94931**	**0.96500**	**0.96339**	**0.99966**	0.99502	0.99228	0.97552
	RAkEL	0.83891	0.97744	0.72279	0.94710	0.91912	0.91250	0.99964	**0.99732**	**0.99250**	**0.98023**
AUC$_{\text{micro}}$	Evolutivo	0.80912	**0.78367**	**0.79746**	0.89547	**0.71235**	**0.75540**	**0.99290**	**0.91827**	**0.71279**	**0.82891**
	RAkEL	**0.82229**	0.68344	0.71553	**0.91153**	0.65812	0.69587	0.88194	0.70681	0.65212	0.64764

no hay diferencias significativas entre el rendimiento de cada clasificador, mientras que la hipótesis alternativa indica que sí existen.

Tras la ejecución del test, obtenemos los resultados que vemos en la Tabla VI, siendo R^+ la suma de rangos donde el rendimiento del algoritmo evolutivo es mejor que el de RAkEL, y R^-, la suma de rangos donde el rendimiento de RAkEL es superior al del algoritmo evolutivo.

Como vemos, para 8 de las 12 métricas existen diferencias significativas en el rendimiento de ambos algoritmos, siendo siempre nuestro algoritmo evolutivo superior a RAkEL. Para *Hamming loss*, *subset accuracy*, *average precision*, *micro-averaged precision*, *macro-averaged precision* y *micro-averaged AUC* podemos afirmar que existen diferencias significativas con un 95 % de confianza ($p-value < 0.05$), y con el 90 % de confianza ($p - value < 0.1$) para *micro-averaged specificity* y *macro-averaged specificity*. Para el resto, aunque no existen diferencias significativas, sí que la suma de rangos positivos es mayor en todos los casos y favorable a nuestro algoritmo. Cabe destacar que *precision* y *recall* son métricas contrapuestas y cuando se obtiene buen resultado en una, suele no obtenerse un buen valor en la otra, por lo que nuestro algoritmo no obtiene diferencias significativas con RAkEL para *recall* (tanto aproximación micro como macro) pero sí para *precision*. De la misma manera, al estar el valor de *F-measure* influenciado por *recall*, hace que no se obtengan diferencias significativas tampoco en esos casos.

TABLA VI

Resultados del test de Wilcoxon.

Métrica	R^+	R^-	*p-value*	Confianza
Hamming loss	50.0	5.0	0.022	95 %
Subset accuracy	49.0	6.0	0.028	95 %
Average precision	52.0	3.0	0.013	95 %
Recall$_{\text{micro}}$	32.0	23.0	0.646	-
Precision$_{\text{micro}}$	51.0	4.0	0.017	95 %
F-measure$_{\text{micro}}$	43.0	12.0	0.114	-
Specificity$_{\text{micro}}$	44.5	10.5	0.083	90 %
Recall$_{\text{macro}}$	28.0	27.0	0.959	-
Precision$_{\text{macro}}$	52.0	3.0	0.013	95 %
F-measure$_{\text{macro}}$	37.0	18.0	0.333	-
Specificity$_{\text{macro}}$	44.0	11.0	0.093	90 %
AUC$_{\text{micro}}$	52.0	3.0	0.013	95 %

VI. Conclusiones

En este artículo se ha presentado un algoritmo evolutivo capaz de diseñar automáticamente un *ensemble* para clasificación multi-etiqueta, encontrando diferencias significativas en 8 de las 12 métricas evaluadas con respecto a RAkEL, un algoritmo de clasificación multi-etiqueta basado en *ensembles*, con una propuesta similar al nuestro pero cuyos clasificadores base son escogidos al azar.

Como líneas de trabajo futuro, pretendemos utilizar otro tipo de clasificadores base distintos en lugar de LP con J48. También se pretende tener un número de etiquetas activas (k) variable entre los distintos clasificadores base y usar otros métodos de votación para el *ensemble*. Por último, ya que se han obtenido diferencias significativas de nuestro algoritmo con su competidor más directo, el próximo paso sería compararlo con otros algoritmos de referencia en clasificación multi-etiqueta.

Agradecimientos

Este trabajo ha sido financiado por el proyecto TIN-2011-22408 del Ministerio de Ciencia y Tecnología y fondos FEDER.

Referencias

[1] Loza E and Fürnkranz J, "Efficient multilabel classification algorithms for large-scale problems in the legal domain," in *Semantic Processing of Legal Texts*, 2010, vol. 6036, pp. 192–215.

[2] Nasierding G and Kouzani A, "Image to text translation by multi-label classification," in *Advanced Intelligent Computing Theories and Applications with Aspects of Artificial Intelligence*, 2010, vol. 6216, pp. 247–254.

[3] Barutcuoglu Z, Schapire RE, and Troyanskaya OG, "Hierarchical multi-label prediction of gene function," *Bioinformatics*, vol. 22, pp. 830–836, 2006.

[4] Alves RT, Delgado MR, and Freitas AA, "Multi-label hierarchical classification of protein functions with artificial immune systems," in *Brazilian Symposium in Bioinformatics (BSB-2008)*, 2008, vol. 5167, pp. 1–12.

[5] Zhang Y, Burer S, Street WN, Bennett K, and Parrado hern E, "Ensemble pruning via semi-definite programming," *Journal of Machine Learning Research*, vol. 7, pp. 1315–1338, 2006.

[6] Eva Gibaja and Sebastian Ventura, "Multi-label learning: a review of the state of the art and ongoing research," WIREs Data Mining Knowl Discov 2014. doi: 10.1002/widm.1139.

[7] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas, "Random k-labelsets for multi-label classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 7, pp. 1079–1089, 2011.

[8] Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Deroski, "An extensive experimental comparison of methods for multi-label learning," *Pattern Recognition*, vol. 45, no. 9, pp. 3084–3104, 2012.

[9] Tsoumakas G., Katakis I., and Vlahavas I., *Data Mining and Knowledge Discovery Handbook, Part 6*, chapter Mining Multi-label Data, pp. 667–685, Springer, 2010.

[10] Boutell M., Luo J., Shen X., and Brown C., "Learning multi-label scene classification," *Pattern recognition*, vol. 37, pp. 1757–1771, 2004.

[11] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank, "Classifier chains for multi-label classification," *Machine Learning*, vol. 85, no. 3, pp. 335–359, 2011.

[12] Amanda Clare, A Clare, and Ross D. King, "Knowledge discovery in multi-label phenotype data," in *Lecture Notes in Computer Science*. 2001, pp. 42–53, Springer.

[13] Zhou Z.H. Zhang M.L., "Multi-label neural networks with applications to functional genomics and text categorization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, pp. 1338–1351, 2006.

[14] Min-Ling Zhang and Zhi-Hua Zhou, "Ml-knn: A lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[15] Weiwei Cheng and Eyke Hullermeier, "Combining instance-based learning and logistic regression for multilabel classification," *Machine Learning*, vol. 76, no. 2-3, pp. 211–225, 2009.

[16] Antenreiter M, Ortner R, and Auer P., "Combining classifiers for improved multilabel image classification," in *1st Workshop on Learning from Multilabel Data (MLD) Held in Conjunction with ECML/PKDD*, 2009, pp. 16–27.

[17] Yiming Yang, "An evaluation of statistical approaches to text categorization," *Information Retrieval*, , no. 1, pp. 69–90, 1999.

[18] Robert E. Schapire and Yoram Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learningl*, , no. 37, pp. 297–336, 1999.

[19] Shenghuo Zhu, Xiang Ji, Wei Xu, and Yihong Gong, "Multi-labelled classification using maximum entropy method," in *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, 2005, pp. 274–281.

[20] Robert E. Schapire and Yoram Singer, "Boostexter: A boosting-based system for text categorization," *Machine Learning*, vol. 39, pp. 135–168, 2000.

[21] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas, "Multilabel classification of music into emotions," *Proc. 2008 International Conference on Music Information Retrieval (ISMIR 2008)*, pp. 325–330, 2008.

[22] F. Briggs, Yonghong Huang, R. Raich, K. Eftaxias, Zhong Lei, W. Cukierski, S. Hadley, A. Hadley, M. Betts, X. Fern, J. Irvine, L. Neal, A. Thomas, G. Fodor, G. Tsoumakas, Hong Wei Ng, Thi Ngoc Tho Nguyen, H. Huttunen, P. Ruusuvuori, T. Manninen, A. Diment, T. Virtanen, J. Marzat, J. Defretin, D. Callender, C. Hurlburt, K. Larrey, and M. Milakov, "The 9th annual mlsp competition: New methods for acoustic classification of multiple simultaneous bird species in a noisy environment," in *2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2013.

[23] A. Elisseeff and J. Weston, "A kernel method for multilabelled classification," in *T.G. Dietterich, S. Becker, and Z. Ghahramani, (eds), Advances in Neural Information Processing Systems 14)*, 2001.

[24] Jianhua Xu, "Fast multi-label core vector machine," *Pattern Recognition*, , no. 46, pp. 885–898, 2013.

[25] S. Diplaris, G. Tsoumakas, P. Mitkas, and I. Vlahavas, "Protein classification with multiple algorithms," in *Proc. 10th Panhellenic Conference on Informatics (PCI 2005)*, 2005, pp. 448–456.

[26] John P. Pestian, Christopher Brew, Pawel Matykiewicz, D. J. Hovermale, Neil Johnson, K. Bretonnel Cohen, and Wodzislaw Duch, "A shared task involving multi-label classification of clinical free text," in *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing (BioNLP '07)*, 2007, pp. 97–104.

[27] Jesse Read, "Scalable multi-label classification," *PhD Thesis, University of Waikato*, 2010.

[28] Jesse Read, "A pruned problem transformation method for multi-label classification," in *Proceedings of the NZ Computer Science Research Student Conference*, 2008.

[29] "Jclec: Java class library for evolutionary computation," http://jclec.sourceforge.net/, Ultimo acceso: 21-10-2014.

[30] Tsoumakas G., Spyromitros-Xioufis E., Vilcek J., and Vlahavas I., "Mulan: A java library for multi-label learning," *Journal of Machine Learning Research*, vol. 12, pp. 2411–2414, 2011.

[31] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, , no. 7, pp. 1–30, 2006.