# Heurístico Exhaustivo de Profundidad Limitada para CC Probabilístico*

Deiner Mena, Elena Montañés, José Ramón Quevedo, Juan José del Coz
{deiner,elena,quevedo,juanjo}@aic.uniovi.es

Centro de Inteligencia Artificial, Universidad de Oviedo en Gijón (Spain)

**Resumen** El método Probabilístico de Cadenas de Clasificadores (PCC) ofrece interesantes propiedades para solucionar tareas de clasificación multietiqueta debido a su habilidad para estimar la probabilidad conjunta de las etiquetas. Su mayor inconveniente es el alto coste computacional del proceso de inferencia requerido para predecir nuevos ejemplos. Recientemente se han propuesto algunos métodos para solucionar este problema, como *Beam Search* o el algoritmo $\epsilon$-*Approximate* basado en búsqueda de coste uniforme. En este artículo se propone un heurístico admisible para el algoritmo A* basado en búsqueda exhaustiva que garantiza, no solamente que alcanza una solución óptima en términos de *subset 0/1 loss*, sino que también explora menos nodos que el algoritmo $\epsilon$-*Approximate*. Los experimentos muestran que, como se esperaba teóricamente, el número de nodos que explora nuestro método siempre es menor. Sin embargo, la diferencia en el número de nodos explorados no es lo suficientemente grande como para compensar el tiempo invertido en el cálculo del heurístico. En ese sentido, el algoritmo $\epsilon$-*Approximate* supone una mejor alternativa y quizá el uso del algoritmo A* con el heurístico aquí propuesto sea una buena opción para problemas multietiqueta complejos, como pueden ser aquellos que presenten ruido.

**Palabras clave** Clasif. multietiqueta, Inferencia, Búsqueda heurística

## 1. Introducción

La clasificación multietiqueta (MLC, *Multi-label Classification*) es un problema de aprendizaje consistente en asignar un subconjunto de etiquetas (clases) a cada instancia. Los problemas de MLC ocurren de manera natural en múltiples aplicaciones, por ejemplo, en la asignación de etiquetas a contenidos.

Tal vez el problema más interesante de aprendizaje multietiqueta es el hecho de que exista dependencia estadística entre las etiquetas. Esa relación constituye una gran fuente de información y por ello no es de extrañar que la investigación sobre MLC se haya centrado en este tema. En este sentido, se distinguen formalmente dos tipos de dependencia, la condicional y la marginal (incondicional) [4]. Existen ya varios métodos capaces de detectar las interdependencias entre las etiquetas [2,8,9,11,12].

* Trabajo financiado parcialmente por el MINECO, proyecto TIN2011-23558.

En cuanto a la dependencia condicional, el método Probabilístico de Cadenas de Clasificadores (PCC) ha despertado gran interés en los últimos años debido a que estima la distribución conjunta condicional de las etiquetas. Sin embargo, el algoritmo PCC original [3] tiene un alto coste computacional ya que usa búsqueda exhaustiva (ES) como estrategia de inferencia para seleccionar predicciones óptimas. El algoritmo $\epsilon$-*Approximate* [5] es un algoritmo más sofisticado basado en búsqueda de coste uniforme que puede realizar predicciones óptimas en términos de *subset 0/1 loss* reduciendo significativamente el coste computacional. Finalmente, un enfoque reciente basado en *Beam Search* [6] también presenta buen comportamiento en rendimiento y coste computacional, a pesar de no garantizar predicciones óptimas en términos de *subset 0/1 loss*.

Una alternativa a estos métodos es aplicar el algoritmo A* con un heurístico admisible. La principal contribución de este artículo es la propuesta de un heurístico basado en búsqueda exhaustiva, pero que permite limitar la profundidad de búsqueda. Esta propuesta garantiza, no solamente predicciones óptimas en términos de *subset 0/1 loss*, sino que también explora menos nodos que todos los métodos anteriores que también realizan predicciones óptimas.

## 2. Clasificación multietiqueta y PCC

Sea $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_m\}$ un conjunto finito y no vacío de $m$ etiquetas y $S = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$ un conjunto de ejemplos de entrenamiento independientes y aleatorios construído según una distribución de probabilidad desconocida $\mathbf{P}(\mathbf{X}, \mathbf{Y})$ sobre $\mathcal{X} \times \mathcal{Y}$, donde $\mathcal{X}$ y $\mathcal{Y}$ son las entradas (espacio de instancias) y las salidas (conjunto de partes de $\mathcal{L}$, $\mathcal{P}(\mathcal{L})$) respectivamente. Para facilitar la notación definimos $\boldsymbol{y}_i$ como un vector binario $\boldsymbol{y}_i = (y_{i,1}, y_{i,2}, \ldots, y_{i,m})$ en el que $y_{i,j} = 1$ indica la presencia (relevancia) y $y_{i,j} = 0$ la ausencia (irrelevancia) de $\ell_j$ en el etiquetado de $\boldsymbol{x}_i$. Por lo tanto, $\boldsymbol{y}_i$ es la realización del vector aleatorio correspondiente $\mathbf{Y} = (\mathbf{Y_1}, \mathbf{Y_2}, \ldots, \mathbf{Y_m})$. Con esta conversión, el espacio de salida se puede definir como $\mathcal{Y} = \{0, 1\}^m$. El objetivo en MLC es inducir un clasificador $\boldsymbol{f} : \mathcal{X} \longrightarrow \mathcal{Y}$ a partir de $S$ que minimice el riesgo en términos de cierta función de pérdida $L(\cdot)$. Este riesgo se puede definir como la pérdida esperada sobre la distribución conjunta $\mathbf{P}(\mathbf{X}, \mathbf{Y})$, es decir,

$$r_L(\boldsymbol{f}) = \mathbb{E}_{\mathbf{X},\mathbf{Y}} L(\mathbf{Y}, \boldsymbol{f}(\mathbf{X})),$$

Así, si denotamos por $\mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x})$ la distribución condicional $\mathbf{Y} = \boldsymbol{y}$ dada $\mathbf{X} = \boldsymbol{x}$, el minimizador de riesgo $\boldsymbol{f}^*$ se puede expresar mediante

$$\boldsymbol{f}^*(\boldsymbol{x}) = \arg\min_{\boldsymbol{f}} \sum_{\boldsymbol{y} \in \mathcal{Y}} \mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x}) L(\boldsymbol{y}, \boldsymbol{f}(\boldsymbol{x})).$$

Pese a que se han adoptado varias funciones de pérdida para evaluar la MLC, como *Hamming loss* o $F_1$, este artículo se centra en *subset 0/1 loss*. Esta medida mira si el subconjunto de etiquetas predichas y relevantes coincide o no:

$$L_{S_{0/1}}(\boldsymbol{y}, \boldsymbol{f}(\boldsymbol{x})) = [\![\boldsymbol{y} \neq \boldsymbol{f}(\boldsymbol{x})]\!]^1.$$

[1] La expresión $[\![p]\!]$ evalúa 1 si $p$ es verdadera y 0 de lo contrario.

En el caso de esta medida de evaluación, basta tomar la distribución condicional conjunta para optimizarla. Formalmente, el minimizador de riesgo adopta la siguiente forma

$$\boldsymbol{f}^*(\boldsymbol{x}) = \arg\max_{\boldsymbol{y}\in\mathcal{Y}} \mathbf{P}(\boldsymbol{y} \mid \boldsymbol{x}).$$

Entre los algoritmos de MLC, nos centraremos en el método PCC y sus variantes. Este grupo de algoritmos está basado en el algoritmo Classifier Chains (CC) [11], así su fase de entrenamiento consiste en 1) establecer un orden para el conjunto de etiquetas, y 2) el entrenamiento de un clasificador binario probabilístico para estimar $\mathbf{P}(y_j \mid \boldsymbol{x}, y_1, \ldots, y_{j-1})$ para cada etiqueta $\ell_j$ siguiendo este orden. Por lo tanto, el modelo probabilístico $f_j$ obtenido para predecir la probabilidad de la etiqueta $\ell_j$ tiene la forma:

$$f_j : \mathcal{X} \times \{0,1\}^{j-1} \longrightarrow [0,1].$$

El conjunto de entrenamiento para $f_j$ es $S_j = \{(\overline{\boldsymbol{x}}_1, y_{1,j}), \ldots, (\overline{\boldsymbol{x}}_n, y_{n,j})\}$ donde $\overline{\boldsymbol{x}}_i = (\boldsymbol{x_i}, y_{i,1}, \ldots, y_{i,j-1})$, es decir, $\boldsymbol{x}_i$ suplementadas por la relevancia de las etiquetas $\ell_1, \ldots, \ell_{j-1}$ que preceden a $\ell_j$ en la cadena. La categoría es la relevancia de la etiqueta $\ell_j$.

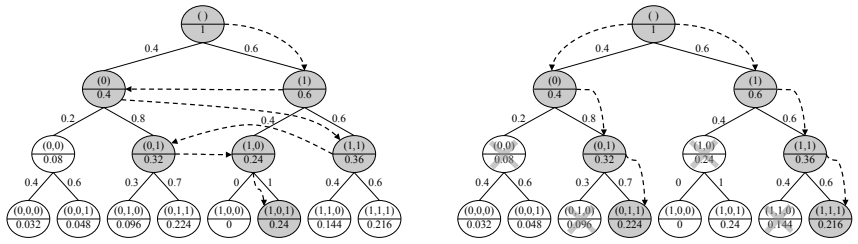
La gran ventaja del método PCC es que permite realizar la inferencia para cada instancia, que consiste en la estimación del minimizador de riesgo para una función de pérdida dada sobre toda la distribución condicional conjunta. Para ello se aplica la regla del producto de la probabilidad de la distribución conjunta de las etiquetas $\mathbf{Y} = (\mathbf{Y_1}, \mathbf{Y_2}, \ldots, \mathbf{Y_m})$, es decir:

$$\mathbf{P}(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{j=1}^{m} \mathbf{P}(y_j \mid \boldsymbol{x}, y_1, \ldots, y_{j-1}).$$

## 3. Inferencia en el método PCC

La inferencia en PCC puede verse como distintas maneras de explorar un árbol binario de probabilidades. En dicho árbol, el nodo raíz está etiquetado por un conjunto vacío, mientras, un nodo genérico $k$ de nivel $j<m$ con $k \leq 2^j$, está etiquedato por $y_j^k=(v_1, v_2, \ldots, v_j)$ con $v_i \in \{0,1\}$ for $i=1, \ldots, j$. Este nodo tiene dos hijos etiquetados respectivamente como $y_{j+1}^{2k-1}=(v_1, v_2, \ldots, v_j, 0)$ y $y_{j+1}^{2k}=(v_1, v_2, \ldots, v_j, 1)$ y con una probabilidad condicional conjunta marginal $\mathbf{P}(y_1{=}v_1, \ldots, y_j{=}v_j, y_{j+1}{=}0 \mid \boldsymbol{x})$ y $\mathbf{P}(y_1{=}v_1, \ldots, y_j{=}v_j, y_{j+1}{=}1 \mid \boldsymbol{x})$. Los pesos de los extremos entre el padre y los hijos son, respectivamente $\mathbf{P}(y_{j+1}{=}0 \mid \boldsymbol{x}, y_1{=} v_1, \ldots, y_j{=}v_j)$ y $\mathbf{P}(y_{j+1}{=}1 \mid \boldsymbol{x}, y_1{=}v_1, \ldots, y_j{=}v_j)$, que se estiman respectivamente por $1 - f_{j+1}(\boldsymbol{x}, v_1, \ldots, v_j)$ y $f_{j+1}(\boldsymbol{x}, v_1, \ldots, v_j)$. La probabilidad condicional conjunta marginal de los hijos se calcula mediante la regla del producto de la probabilidad: $\mathbf{P}(y_1{=}v_1, \ldots, y_j{=}v_j, y_{j+1}{=}v_{j+1} \mid \boldsymbol{x}){=}\mathbf{P}(y_{j+1}{=}v_{j+1} \mid \boldsymbol{x}, y_1{=}v_1, \ldots, y_j{=} v_j) \cdot \mathbf{P}(y_1{=}v_1, \ldots, y_j{=}v_j \mid \boldsymbol{x})$. La Figura 1 ilustra este tipo de árboles.

Veamos ahora los principales métodos de inferencia de la literatura.




(a) $\epsilon$-Aproximate con $\epsilon = .0$ (b) Beams Search con $b = 2$

**Figura 1.** Un ejemplo del camino recorrido por $\epsilon$-A ($\epsilon = .0$) y BS ($b = 2$). Las flechas punteadas muestran el camino que sigue el algoritmo. En el caso de BS, los nodos con una cruz no son explorados porque no tienen ninguna de las probabilidades condicionales conjuntas marginales más altas en su nivel

### 3.1. Greedy Search

La estrategia *Greedy Search* (GS) define el método original CC [11]. GS produce una salida $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_m)$ para una nueva instancia sin etiquetar $\boldsymbol{x}$ estimando sucesivamente la probabilidad $\mathbf{P}(y_j \mid \boldsymbol{x}, y_1, \ldots, y_{j-1})$ a través de cada clasificador $f_j$. Cuando $\hat{y}_j$ se estima, $f_j$ usa las predicciones de las etiquetas anteriores, $\hat{y}_1, \ldots, \hat{y}_{j-1}$. Esto significa que, en términos de la probabilidad del árbol binario definido anteriormente, GS solo explora un camino. En el ejemplo representado en la Figura 1, GS seleccionaría la hoja del extremo derecho, por lo tanto no se alcanza la solución óptima.

En cuanto a la optimización de la *subset 0/1 loss*, un análisis riguroso [5] establece límites para el método GS y determina que ofrece resultados bastantes pobres para esta función de pérdida, a pesar de ser su medida objetivo.

### 3.2. Algoritmo $\epsilon$-Approximate

El algoritmo $\epsilon$-*Approximate* ($\epsilon$-A) [5] surge como una alternativa al alto coste computacional de la ES y el pobre rendimiento de GS. En términos del árbol de probabilidad definido anteriormente, este método expande solamente aquellos nodos cuya probabilidad condicional conjunta marginal, $\mathbf{P}(y_1, \ldots, y_j \mid \boldsymbol{x})$, excede el umbral $\epsilon = 2^{-k}$, con $1 \leq k \leq m$. De hecho, los nodos se expanden en el orden decreciente establecido por esta probabilidad (ver la Figura 1(a)). Se pueden presentar dos situaciones: 1) el nodo que se expande es una hoja o 2) no hay más nodos que excedan el umbral $\epsilon$. Si la primera situación ocurre, la predicción para las instancias sin etiquetar será la combinación de etiquetas de dicha hoja. Si ocurre la segunda situación, se aplica GS a aquellos nodos cuyas hijos no excedan el umbral, y la predicción será aquella con la probabilidad condicional conjunta $\mathbf{P}(y_1, \ldots, y_m \mid \boldsymbol{x})$ más alta.

El parámetro $\epsilon$ juega un papel clave. El caso particular de $\epsilon$=0 (o cualquier valor en el intervalo [0, $2^{-m}$], es decir, $k = m$) tiene un especial interés, porque

el algoritmo realiza una búsqueda de coste uniforme (UC) y siempre encuentra una solución óptima. La Figura 1(a) ilustra esta situación. Las flechas punteadas muestran el camino seguido por el algoritmo y al final la combinación de etiquetas óptima en términos de *subset 0/1* se alcanza. Por otro lado, el método tiende a ser GS a medida que $\epsilon$ crece, siendo GS en el caso de $k=1, \epsilon=2^{-1}=0.5$.

Este algoritmo estima el riesgo del minimizador para la *subset 0/1 loss* en mayor o menor grado en función del valor de $\epsilon$. Incluso, un análisis teórico de esta estimación [5] permite conocer sus límites en función $\epsilon$.

### 3.3. Beam Search

*Beam Search* (BS) [6] también explora más de un camino del árbol probabilístico. Este método incluye un parámetro $b$ llamado *beam width* que limita el número de combinaciones exploradas. La idea es explorar $b$ nodos en cada nivel. Por lo tanto, dependiendo de dicho valor se exploran exhaustivamente cierto número de niveles superiores, particularmente un total de $k^* - 1$ niveles, siendo $k^*$ el número entero más bajo tal que $b < 2^{k^*}$. Para cada uno de los niveles restantes únicamente se exploran $b$ nodos. Los nodos explorados desde el nivel $k^*$ hasta las hojas son aquellos con la probabilidad condicional conjunta marginal más alta vista hasta entonces. Al final, el algoritmo produce como salida la combinación con la probabilidad condicional conjunta más alta. La Figura 1(b) muestra un ejemplo del algoritmo BS con $b=2$. En dicho ejemplo se exploran todos los nodos del primer nivel. Para el resto de niveles, solo se exploran dos nodos, aquellos con la mayor probabilidad condicional conjunta marginal entre los hijos cuyos padres se hayan explorado previamente. En este sentido, el nodo con la solución optima no se explora ya que el padre se descarta antes. Este ejemplo confirma que BS no garantiza que se alcance la solución óptima.

En el caso de $b = 1$, BS explora sólo un nodo por cada nivel, aquel con la mayor probabilidad condicional conjunta marginal, así BS sigue solo un camino y es equivalente a GS. Del mismo modo, si $b=2^m$ BS equivale a ES. Por lo tanto, BS encapsula a GS y ES. Esto hace posible controlar el equilibrio entre el coste computacional y el rendimiento del método mediante el ajuste de $b$.

Como observación final, el hecho de que BS considere probabilidades condicionales conjuntas marginales hace que este método tienda a estimar el riesgo del minimizador para la *subset 0/1 loss*.

## 4. Algoritmo A* para inferencia en PCC

El algoritmo A* es la forma más conocida de búsqueda primero el mejor [10]. En cada iteración se explora el mejor nodo de acuerdo a una función de evaluación $e$. La particularidad del algoritmo A* es que $e$ puede verse como una función $E$ de dos funciones $g$ y $h$, $e(k) = E(g(k), h(k))$, donde $g(k)$ evalúa el coste de llegar a un nodo $k$ desde la raíz y $h(k)$ evalúa el coste de alcanzar una solución desde el nodo $k$. Por lo tanto, $e(k)$ evalúa el coste total de alcanzar una solución desde la raíz pasando por el nodo $k$. En general, es posible obtener exactamente

el valor de la información conocida ($g$), pero la información desconocida debe estimarse utilizando un heurístico ($h$). Para obtener una solución óptima, $h$ no debe sobreestimar el coste real de alcanzar una solución, es decir, debe ser un heurístico admisible. Esta clase de heurísticos son optimistas, porque estiman que el coste de obtener una solución es menor del que realmente es. También, el algoritmo A* es óptimamente eficiente para cualquier heurístico, porque ningún otro algoritmo que use el mismo heurístico expandirá menos nodos que A*.

Con el fin de adaptar el algoritmo A* para ser utilizado como método de inferencia en PCC, se debe tener en cuenta que tenemos ganancias (probabilidades) en lugar de costes. Por lo tanto, 1) A* debe seleccionar el nodo con la probabilidad estimada más alta , 2) $h$ no debe subestimar la probabilidad desde el nodo hasta una hoja, y 3) $E$ debe ser una función del producto, $e = g \cdot h$. Considerando todos estos aspectos, $e$ suministrará una estimación de la probabilidad condicional conjunta $\mathbf{P}(y_1, \ldots, y_m \,|\, \boldsymbol{x})$ para optimizar la *subset 0/1 loss*. Para obtener $g$ y $h$, se reescribe la regla del producto de la probabilidad para la distribución conjunta de las etiquetas $\mathbf{Y} = (\mathbf{Y_1}, \mathbf{Y_2}, \ldots, \mathbf{Y_m})$ como

$$\mathbf{P}(y_1, \ldots, y_m \,|\, \boldsymbol{x}) = \mathbf{P}(y_1, \ldots, y_j \,|\, \boldsymbol{x}) \times \mathbf{P}(y_{j+1}, \ldots, y_m \,|\, \boldsymbol{x}, y_1, \ldots, y_j).$$

Por lo tanto, para un nodo en el nivel $j$, sea $g$ la probabilidad condicional conjunta marginal $\mathbf{P}(y_1, \ldots, y_j \,|\, \boldsymbol{x})$ y $h$ un heurístico que no subestima la probabilidad condicional conjunta marginal $\mathbf{P}(y_{j+1}, \ldots, y_m \,|\, \boldsymbol{x}, y_1, \ldots, y_j)$. Los valores de $y_1, \ldots, y_j$ son conocidos en el nivel $j$. La función $g$ es la misma probabilidad condicional conjunta marginal que el algoritmo $\epsilon$-A y BS calculan para seleccionar un nodo a expandir. Incluso, $\epsilon$-A con $\epsilon = 0$ no es solamente equivalente a UC, sino también a A* con heurístico $h = 1$. Este heurístico es admisible también, porque ninguna probabilidad es mayor que 1. Sin embargo, este es el peor heurístico admisible, porque cualquier otro heurístico admisible lo dominará y consecuentemente el algoritmo A* usando dicho heurístico nunca expandirá más nodos que el algoritmo A* usando $h = 1$.

Veamos ahora el heurístico propuesto en este artículo. Dado que nuestro heurístico no debe subestimar la probabilidad condicional conjunta marginal $\mathbf{P}(y_{j+1}, \ldots, y_m \,|\, \boldsymbol{x}, y_1, \ldots, y_j)$ para ser admisible, es bastante sencillo seleccionar el valor máximo de dicha probabilidad para obtener un heurístico óptimo $h^*$:

$$\begin{aligned}
h^* &= \max_{(y_{j+1},\ldots,y_m)\in\{0,1\}^{m-j}} \mathbf{P}(y_{j+1}, \ldots, y_m \,|\, \boldsymbol{x}, y_1, \ldots, y_j) \\
&= \max_{(y_{j+1},\ldots,y_m)\in\{0,1\}^{m-j}} \prod_{i=j+1}^{m} \mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1}) \\
&\leq \max_{(y_{j+1},\ldots,y_{j+k})\in\{0,1\}^{k}} \prod_{i=j+1}^{j+k} \mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1}) \cdot \prod_{i=j+k+1}^{m} 1.
\end{aligned}$$

Sin embargo, obtener dicho máximo significa, de hecho, aplicar una ES sobre el conjunto de etiquetas $\mathcal{L} = \{\ell_{j+1}, \ldots, \ell_m\}$, lo cual no es computacionalmente aplicable. Por lo tanto, debemos renunciar a obtener un heurístico óptimo a

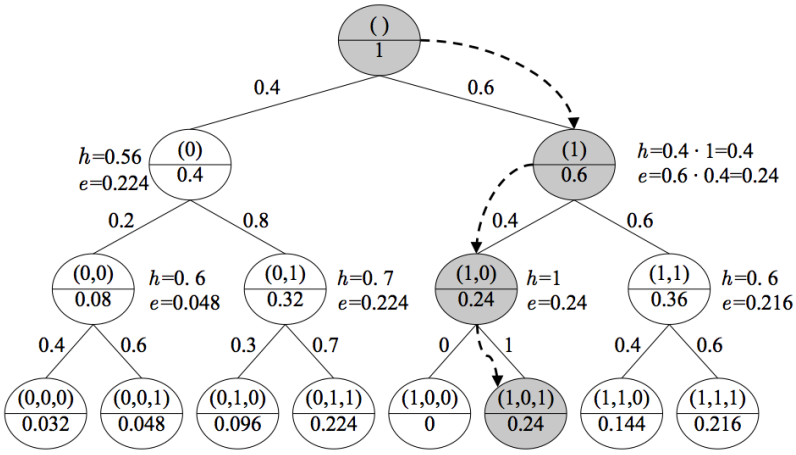



**Figura 2.** Camino recorrido por A* usando $h^k$ con $k = 2$ niveles de profundidad. Las flechas punteadas muestran el camino que sigue el algoritmo

cambio de obtener un heurístico computacionalmente tratable. Será admisible, pero menos dominante que $h^*$. Para lograr dicho heurístico consideramos una variante de la ES que incluya un parámetro $k$ que limite su profundidad de búsqueda. Llamaremos a este heurístico ES de $k$ niveles de profundidad y lo denotaremos por $h^k$. Por consiguiente, para un nodo en el nivel $j$ y un valor de $k$ tal que $0 \leq k \leq m - j$, las probabilidades $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, \overline{y}_{j+1}, \ldots, \overline{y}_{i-1})$ se evalúan 1) aplicando ES mediante el cálculo de la probabilidad condicional conjunta marginal de todas las ramas del subárbol y quedándose con la rama de máxima probabilidad para $i$ desde $j+1$ hasta $j+k$ y 2) mediante la constante 1 para $i$ desde $j+k+1$ hasta $m$. Luego, $h^k$ estará definido por la siguiente ecuación:

$$h^k = \max_{(y_{j+1},\ldots,y_{j+k})\in\{0,1\}^k} \prod_{i=j+1}^{j+k} \mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1}).$$

De esta forma, $h^k$ es menos dominante que $h^*$ ($h^* \prec h^k$) pero continua siendo admisible. Nótese que $h^k$ con $k = 0$ ($h^0$) no es más que $h = 1$. A su vez, $h^k$ es más dominante que $h^0$ ($h^k \prec h = 1$). Por lo tanto, el algoritmo A* usando el heurístico $h^k$ explora más nodos que $h^*$, pero menos que $h^0$. De hecho el heurístico $h^k$ explora menos nodos conforme $k$ aumenta, aunque también incrementa su tiempo computacional. Respecto a la optimalidad, el algoritmo A* usando $h^k$ estima perfectamente el riesgo del minimizador para la *subset 0/1 loss*, de igual manera que lo hace $\epsilon$-A con $\epsilon = .0$.

La Figura 2 muestra la ruta seguida por A* usando $h^k$ con $k = 2$. En este caso el algoritmo alcanza la solución óptima como teóricamente se esperaba. Comparando este gráfico con el de la Figura 1(a) que ilustra $\epsilon$-A con $\epsilon = 0$ y teniendo en cuenta las propiedades de los heurísticos relacionadas con la dominancia, $\epsilon$-A con $\epsilon = 0$ explora más nodos que A* usando $h^k$.

**Tabla 1.** Resultados de subset 0/1. Aquellos resultados que son iguales o mejores que las predicciones óptimas se muestran en negrita

| | UC/$h^{1,2,3}$ | | GS/BS(1) | BS(2) | BS(3) |
|---|---|---|---|---|---|
| Conjuntos | $\epsilon$-A(.0) | $\epsilon$-A(.25) | $\epsilon$-A(.5) | | |
| bibtex | **81.92** | 81.95 | 82.19 | **81.88** | **81.92** |
| corel5k | **97.48** | 98.62 | 98.90 | 98.30 | 98.04 |
| emotions | **71.16** | 71.82 | 72.83 | 72.16 | 71.32 |
| enron | **83.14** | 84.26 | 85.43 | 83.43 | 83.37 |
| flags | **87.13** | 87.16 | **86.13** | 88.21 | **87.13** |
| image | **68.35** | **68.35** | 69.75 | **68.35** | **68.35** |
| mediamill* | **83.86** | 84.58 | 85.80 | 84.10 | **83.86** |
| medical | **30.37** | **30.37** | 30.67 | **30.37** | **30.37** |
| reuters | **22.73** | **22.70** | **23.60** | **22.69** | **22.73** |
| scene | **31.86** | **31.86** | 33.28 | 31.90 | **31.86** |
| slashdot | **51.80** | 52.22 | 54.49 | **51.77** | **51.80** |
| yeast | **76.95** | 77.62 | 79.77 | **76.83** | 77.08 |

## 5. Experimentos y resultados

Se realizaron experimentos sobre varios conjuntos de datos de referencia en MLC. La principal característica de los conjuntos es el número de etiquetas, que varía entre 5 y 374. Los métodos que se compararon son GS, búsqueda de coste uniforme (UC), $\epsilon$-A para diferentes valores de $\epsilon \in \{.0, .25, .5\}$, BS para varios valores de *beam width* $b \in \{1, 2, 3\}$ y $h^k$ con diferentes valores $k \in \{1, 2, 3\}$. Notemos que GS es equivalente a $\epsilon$-A (.5) y a BS con $b = 1$ y que $\epsilon$-A (.0) es equivalente a UC.

El algoritmo base empleado para obtener los clasificadores binarios $f_i$ fue regresión logística [7] con salida probabilística. El parámetro de regularización $C$ se estableció para cada clasificador binario realizando un *grid search* sobre los valores $C \in \{10^{-3}, 10^{-2}, \ldots, 10^2, 10^3\}$ optimizando la medida *brier loss* [1] estimada mediante una validación cruzada de 2 particiones con 5 repeticiones.

La Tabla 1 muestra los resultados de *subset 0/1 loss* en una validación cruzada de 10 particiones. UC, $\epsilon$-A(.0) y $h^k$ que suministran predicciones óptimas en términos de *subset 0/1 loss* se muestran todas juntas en la primera columna. Como se esperaba el rendimiento del algoritmo $\epsilon$-A disminuye a medida que los valores de $\epsilon$ se incrementan (.25 o .5). Para estos dos valores, el algoritmo $\epsilon$-A obtiene peores resultados que UC, $\epsilon$-A(.0) y $h^k$ en 18 de los 24 casos. Por otro lado, el rendimiento de BS converge al óptimo conforme el valor de $b$ aumenta. Para BS(2) hay 6 conjuntos de datos cuyos resultados son peores que los obtenidos por el óptimo, mientras que en otros 6 sus resultados son iguales o mejores. Las diferencias son grandes cuando los resultados de BS(2) son peores. En el caso de BS(3), en 8 conjuntos de datos los resultados son exactamente los mismos que los de UC, $\epsilon$-A(.0) y $h^k$ mostrando la convergencia del método al óptimo. No ejecutamos más experimentos con más valores de $b$ para estudiar este aspecto, ya que como analizaremos más adelante, el número de nodos explorados por BS(3) empieza a ser considerablemente mayor que el de los métodos óptimos.

**Tabla 2.** Promedio de número de nodos explorados (arriba) y de tiempo (ms) de predicción por ejemplo (abajo). El número de etiquetas se muestra entre paréntesis

| Conjuntos | $h^1$ | $h^2$ | $h^3$ | $\epsilon$-A(.0) | $\epsilon$-A(.25) | GS/BS(1) $\epsilon$-A(.5) | BS(2) | BS(3) |
|---|---|---|---|---|---|---|---|---|
| bibtex (159) | 228.3 | 225.7 | 223.3 | 231.2 | 180.5 | 180.3 | 318.0 | 477.0 |
| corel5k (374) | 1577.1 | 1559.1 | 1.542.1 | 1597.0 | 534.3 | 525.4 | 748.0 | 1.122.0 |
| emotions (6) | 8.6 | 7.8 | 7.4 | 10.3 | 10.5 | 9.1 | 12.0 | 18.0 |
| enron (53) | 116.1 | 111.5 | 107.1 | 121.6 | 78.5 | 77.7 | 106.0 | 159.0 |
| flags (7) | 13.1 | 10.5 | 8.8 | 20.7 | 17.3 | 12.0 | 14.0 | 21.0 |
| image (5) | 6.3 | 6.1 | 6.0 | 7.5 | 7.8 | 6.9 | 10.0 | 15.0 |
| mediamill* (101) | 189.5 | 186.1 | 182.8 | 193.2 | 140.3 | 140.3 | 202.0 | 303.0 |
| medical (45) | 46.1 | 46.1 | 46.1 | 46.1 | 46.1 | 46.1 | 90.0 | 135.0 |
| reuters (7) | 8.2 | 8.1 | 8.1 | 8.3 | 8.3 | 8.3 | 14.0 | 21.0 |
| scene (6) | 7.1 | 7.1 | 7.0 | 7.2 | 7.2 | 7.2 | 12.0 | 18.0 |
| slashdot (22) | 25.0 | 24.7 | 24.5 | 25.2 | 24.9 | 24.9 | 44.0 | 66.0 |
| yeast (14) | 23.1 | 21.1 | 19.6 | 25.6 | 25.9 | 25.1 | 28.0 | 42.0 |
| bibtex (159) | 22.7 | 24.8 | 30.2 | 16.2 | 9.9 | 7.9 | 11.0 | 15.6 |
| corel5k (374) | 175.4 | 190.8 | 229.4 | 136.0 | 16.1 | 11.0 | 27.8 | 40.4 |
| emotions (6) | 0.7 | 0.7 | 0.7 | 0.6 | 0.6 | 0.4 | 0.5 | 0.6 |
| enron (53) | 10.2 | 14.5 | 18.4 | 7.2 | 3.0 | 1.9 | 3.9 | 5.5 |
| flags (7) | 1.1 | 0.9 | 0.9 | 1.2 | 0.7 | 0.4 | 0.5 | 0.7 |
| image (5) | 0.5 | 0.5 | 0.6 | 0.5 | 0.5 | 0.4 | 0.4 | 0.5 |
| mediamill* (101) | 17.1 | 18.9 | 23.3 | 11.5 | 5.5 | 3.7 | 7.2 | 10.5 |
| medical (45) | 4.1 | 4.6 | 5.7 | 2.7 | 2.7 | 2.7 | 3.3 | 4.6 |
| reuters (7) | 0.7 | 0.7 | 0.8 | 0.5 | 0.5 | 0.5 | 0.5 | 0.7 |
| scene (6) | 0.6 | 0.6 | 0.7 | 0.5 | 0.5 | 0.4 | 0.5 | 0.6 |
| slashdot (22) | 2.2 | 2.4 | 2.9 | 1.5 | 1.4 | 1.2 | 1.6 | 2.2 |
| yeast (14) | 2.0 | 2.0 | 2.3 | 1.5 | 1.0 | 0.6 | 1.0 | 1.4 |

En la Tabla 2 se muestra el número de nodos explorados por cada método y el promedio de tiempo de predicción en milisegundos. GS (o $\epsilon$-A(.5) y BS(1)) es, por supuesto, el método que explora menos nodos, ya que recorre un solo camino del árbol, o sea, explora tantos nodos como número de etiquetas más uno, ya que el nodo raíz también se explora. En los métodos que no devuelven predicciones óptimas, parece que $\epsilon$-A(.25) tiende a expandir menos nodos que BS(2), pero con peores resultados en términos de *subset 0/1 loss*. BS(3) es el algoritmo que expande más número de nodos. Respecto a los métodos que devuelven predicciones óptimas, $h^k$ es el método que explora menos nodos, para todos los valores de $k$. Incluso se exploran menos nodos a medida que aumenta el nivel de profundidad de la búsqueda. La diferencia en nodos explorados entre $h^k$ y GS en algunos casos es pequeña. Por otro lado la diferencia en nodos explorados entre $h^k$ con $k > 0$ y $\epsilon$-A(.0), UC o $h^0$ no es lo suficientemente alta. La razón podría ser que las predicciones de los clasificadores $f_i$ a lo largo del camino óptimo están generalmente cerca de 1 o de 0. No hay mucha incertidumbre, por lo que el nivel de *backtracking* es bajo y por tanto la búsqueda es muy dirigida.

En cuanto a los tiempos de predicción, curiosamente $\epsilon$-A(.0) y UC no son mucho más lentos que $\epsilon$-A(.25) y $\epsilon$-A(.5) y son similares a BS(2) y BS(3). Además, UC, $\epsilon$-A(.0) o $h^0$ son más rápidos que $h^k$ con $k > 0$ pese a explorar más nodos. La razón es el coste computacional de $h^k$ con $k > 0$.

## 6. Conclusiones

Este artículo propone un método basado en el algoritmo A* con un heurístico admisible para inferir predicciones en PCC. Este heurístico $h^k$ se basa en el método de búsqueda exhaustiva con un parámetro $k$ que controla el nivel de profundidad de la búsqueda y el equilibrio entre el número de nodos explorados y el tiempo computacional. El método produce predicciones óptimas en términos de *subset 0/1 loss* y explora menos nodos que otros métodos de la literatura que también realizan predicciones óptimas.

En los experimentos realizados, la búsqueda de coste uniforme ($\epsilon$-A(.0) o A* con $h^0$) es mejor opción que $h^k$ con $k > 0$, ya que el coste computacional de evaluar el heurístico no compensa la reducción en nodos explorados. En cualquier caso, los conjuntos de datos empleados aquí generan búsquedas muy dirigidas. Si hubiese más incertidumbre en la predicción de las etiquetas, el heurístico propuesto tendría mejores resultados. Por ello, creemos que puede resultar una alternativa interesante en problemas más difíciles, incluyendo aquellos con ruido.

## Referencias

1. Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Mon. Wea. Rev.*, 78(1):1–3, January 1950.
2. W. Cheng and E. Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225, 2009.
3. K. Dembczyński, W. Cheng, and E. Hüllermeier. Bayes Optimal Multilabel Classification via Probabilistic Classifier Chains. In *ICML, 2010*, pages 279–286, 2010.
4. K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier. On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88(1-2):5–45, 2012.
5. K. Dembczynski, W. Waegeman, and E. Hüllermeier. An analysis of chaining in multi-label classification. In *ECAI*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 294–299. IOS Press, 2012.
6. Abhishek Kumar, Shankar Vembu, Aditya Krishna Menon, and Charles Elkan. Beam search algorithms for multilabel learning. *Mach. Learn.*, 92(1):65–89, 2013.
7. C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region Newton method for logistic regression. *Journal of Machine Learning Research*, 9(Apr):627–650, 2008.
8. E. Montañés, J.R. Quevedo, and J. J. del Coz. Aggregating independent and dependent models to learn multi-label classifiers. In *ECML'11*, pages 484–500.
9. E. Montañés, R. Senge, J. Barranquero, J.R. Quevedo, J. J. del Coz, and E. Hüllermeier. Dependent binary relevance models for multi-label classification. *Pattern Recognition*, 47(3):1494 – 1508, 2014.
10. Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984.
11. J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359, 2011.
12. G. Tsoumakas and I. Vlahavas. Random k-Labelsets: An Ensemble Method for Multilabel Classification. In *ECML/PKDD'07*, pages 406–417. Springer, 2007.