Using A* for Inference in Probabilistic Classifier Chains*

Deiner Mena[a,b], **Elena Montañés**[a], **José R. Quevedo**[a], **Juan José del Coz**[a]
[a] Artificial Intelligence Center, University of Oviedo at Gijón, (Asturias) Spain
[b] Universidad Tecnológica del Chocó, Colombia
{deiner,quevedo,elena,juanjo}@aic.uniovi.es, deiner.mena@utch.edu.co

Abstract

Probabilistic Classifiers Chains (PCC) offers interesting properties to solve multi-label classification tasks due to its ability to estimate the joint probability of the labels. However, PCC presents the major drawback of having a high computational cost in the inference process required to predict new samples. Lately, several approaches have been proposed to overcome this issue, including beam search and an ϵ-Approximate algorithm based on uniform-cost search. Surprisingly, the obvious possibility of using heuristic search has not been considered yet. This paper studies this alternative and proposes an admisible heuristic that, applied in combination with A* algorithm, guarantees, not only optimal predictions in terms of subset 0/1 loss, but also that it always explores less nodes than ϵ-Approximate algorithm. In the experiments reported, the number of nodes explored by our method is less than two times the number of labels for all datasets analyzed. But, the difference in explored nodes must be large enough to compensate the overhead of the heuristic in order to improve prediction time. Thus, our proposal may be a good choice for complex multi-label problems.

1 Introduction

Multi-label classification (MLC) is a machine learning problem in which models are sought that assign a subset of (class) labels to each instance. Multi-label classification problems are ubiquitous and naturally occur in many applications, for instance, in assigning tags to content.

Maybe the most interesting problem in multi-label learning is that the labels shows statistical dependencies between them. These relationships constitute a key source of information, thus, it is hardly surprising that research on MLC has very much focused on this topic. In this regard, two kinds of label dependence have been formally distinguished, namely, conditional dependence and marginal (unconditional) dependence [Dembczyński *et al.*, 2012a]. Several methods are able to detect interdependencies among labels, see [Godbole and Sarawagi, 2004; Tsoumakas and Vlahavas, 2007; Read *et al.*, 2008; Cheng and Hüllermeier, 2009; Montañés *et al.*, 2011; Read *et al.*, 2011; Montañés *et al.*, 2014].

Regarding conditional label dependence, Probabilistic Classifier Chains (PCC) has aroused great interest since it estimates the conditional joint distribution of the labels. However, the original PCC algorithm [Dembczyński *et al.*, 2010] suffers from high computational cost because it uses exhaustive search as the inference strategy to select optimal predictions given a loss function. Then, several efforts of overcoming this drawback have being proposed. Monte Carlo sampling [Dembczynski *et al.*, 2012b; Read *et al.*, 2014] is a simple alternative to control the computational cost of the inference process. ϵ-Approximate [Dembczynski *et al.*, 2012b] is a more sophisticated algorithm based on uniform-cost search. This method can output optimal predictions in terms of subset 0/1 loss reducing significantly the computational cost. Finally, a more recently approach based on beam search [Kumar *et al.*, 2012; 2013] also presents good behavior both in terms of performance and computational cost.

An alternative to these (uninformed) approaches is to apply other search algorithms. Among them, heuristic search is at least worthy of consideration. However, to the best of our knowledge this possibility has never been studied before. The main contribution of this paper is to fill this gap in the literature, proposing a new approach for inference in linear PCC models based on the combination of A* algorithm [Hart *et al.*, 1968] and an admisible heuristic. Our proposal guarantees, not only optimal predictions in terms of subset 0/1 loss, but also that it explores less nodes than all previous methods that also provide optimal predictions.

The paper is organized as follows. Section 2 formally describes MLC and PCC. Recent approaches for inference in PCC are discussed in Section 3. Section 4 details the new method based on A*. Finally, our experiments are discussed in Section 5 and some conclusions are exposed in Section 6.

2 Multi-label classification and PCC

Let be $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_m\}$ a finite and non-empty set of m labels and $S = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$ a training set independently and randomly drawn according to an unknown probability distribution $\mathbf{P}(\mathbf{X}, \mathbf{Y})$ on $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are the input and the output space, respectively. The former

*Research supported by MINECO, grant TIN2011-23558.

is the instance space, whereas the latter is given by the power set $\mathcal{P}(\mathcal{L})$ of $\mathcal{L}$. To ease notation, we define $\boldsymbol{y}_i$ as a binary vector $\boldsymbol{y}_i = (y_{i,1}, y_{i,2}, \ldots, y_{i,m})$ in which $y_{i,j} = 1$ indicates the presence (relevance) and $y_{i,j} = 0$ the absence (irrelevance) of ℓ_j in the labeling of $\boldsymbol{x}_i$. Hence, $\boldsymbol{y}_i$ is the realization of a corresponding random vector $\mathbf{Y} = (\mathbf{Y_1}, \mathbf{Y_2}, \ldots, \mathbf{Y_m})$. Using this convention, the output space can also be defined as $\mathcal{Y} = \{0,1\}^m$. The goal in MLC is to induce from S a classifier $\boldsymbol{f} : \mathcal{X} \longrightarrow \mathcal{Y}$ that minimizes the risk in terms of certain loss function $L(\cdot)$. This risk can be defined as the expected loss over the joint distribution $\mathbf{P}(\mathbf{X}, \mathbf{Y})$, that is,

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

thus, denoting by $\mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x})$ the conditional distribution $\mathbf{Y} = \boldsymbol{y}$ given $\mathbf{X} = \boldsymbol{x}$, the risk minimizer $\boldsymbol{f}^*$ can be expressed by

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

Let us comment that the conditional distribution $\mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x})$ presents different properties which are crucial for optimizing different loss functions. Despite several loss functions have been taken for evaluating MLC, like Hamming loss or F_1, here we will focus on just the subset 0/1 loss. This measure looks if predicted and relevant label subsets are equal or not:

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

In case of this evaluation measure, it suffices taking the mode of the entire joint conditional distribution for optimizing this loss. Formally, the risk minimizer adopts the simplified following form

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

Among MLC algorithms, let us focus on PCC methods. This group of algorithms is based on learning a CC model [Read *et al.*, 2011] so the training phase consists on 1) taking an order of the label set and 2) training a probabilistic binary classifier for estimating $\mathbf{P}(y_j \,|\, \boldsymbol{x}, y_1, \ldots, y_{j-1})$ for each label ℓ_j following this order. The probabilistic model f_j obtained for predicting label ℓ_j is of the form

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

The training data for classifier f_j is the set $S_j = \{(\overline{\boldsymbol{x}}_1, y_{1,j}), \ldots, (\overline{\boldsymbol{x}}_n, y_{n,j})\}$ where $\overline{\boldsymbol{x}}_i = (\boldsymbol{x}, y_{i,1}, \ldots, y_{i,j-1})$, that is, the features are $\boldsymbol{x}_i$ supplemented by the relevance of the labels $\ell_1, \ldots, \ell_{j-1}$ preceding ℓ_j in the chain and the category is the relevance of label ℓ_j.

The great advantage of PCC approaches is that they allow to perform inference for each instance, which consists of estimating the risk minimizer for a given loss function over the estimated entire joint conditional distribution. The idea revolves around repeatedly applying the product rule of probability to the joint distribution of the labels $\mathbf{Y} = (\mathbf{Y_1}, \mathbf{Y_2}, \ldots, \mathbf{Y_m})$, that is, computing

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

Notice that from a theoretical point of view, this expression holds for any order considered for the labels.

[1]The expression $[\![p]\!]$ evaluates to 1 if p is true and 0 otherwise.

3 Inference in Probabilistic Classifier Chains

Several methods have been previously proposed for inference in PCC. Before proceeding to cope with the particularities of some of them, let us recall that their training phases are equal, thus, the models f_j will be the same. So, in what follows we will just describe their respective prediction phases.

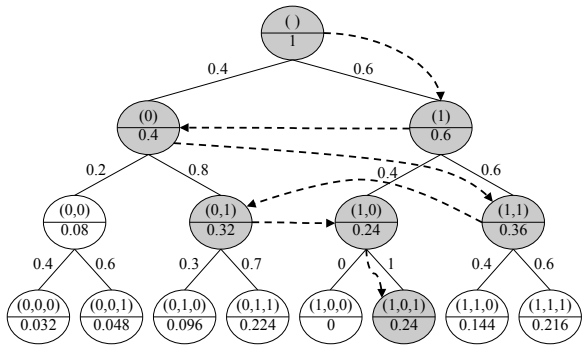
In this sense, let now consider the task of performing different inference procedures as different manners of exploring a probability binary tree. In such tree, the root node is labeled by the empty set whereas a generic node k of level $j < m$ with $k \leq 2^j$, is labeled by $y_j^k = (v_1, v_2, \ldots, v_j)$ with $v_i \in \{0,1\}$ for $i = 1, \ldots, j$. This node has two children respectively labeled as $y_{j+1}^{2k-1} = (v_1, v_2, \ldots, v_j, 0)$ and $y_{j+1}^{2k} = (v_1, v_2, \ldots, v_j, 1)$ and with marginal joint conditional probability $\mathbf{P}(y_1 = v_1, \ldots, y_j = v_j, y_{j+1} = 0 \,|\, \boldsymbol{x})$ and $\mathbf{P}(y_1 = v_1, \ldots, y_j = v_j, y_{j+1} = 1 \,|\, \boldsymbol{x})$. The weights of the edges between both father and children are respectively $\mathbf{P}(y_{j+1} = 0 \,|\, \boldsymbol{x}, y_1 = v_1, \ldots, y_j = v_j)$ and $\mathbf{P}(y_{j+1} = 1 \,|\, \boldsymbol{x}, y_1 = v_1, \ldots, y_j = v_j)$, which are respectively estimated by $1 - f_{j+1}(\boldsymbol{x}, v_1, \ldots, v_j)$ and $f_{j+1}(\boldsymbol{x}, v_1, \ldots, v_j)$. The marginal joint conditional probability of the children are computed by the product rule of probability: $\mathbf{P}(y_1 = v_1, \ldots, y_j = v_j, y_{j+1} = v_{j+1} \,|\, \boldsymbol{x}) = \mathbf{P}(y_{j+1} = v_{j+1} \,|\, \boldsymbol{x}, y_1 = v_1, \ldots, y_j = v_j) \cdot \mathbf{P}(y_1 = v_1, \ldots, y_j = v_j \,|\, \boldsymbol{x})$. See the examples in Figure 1.

3.1 Greedy Search and Exhaustive Search

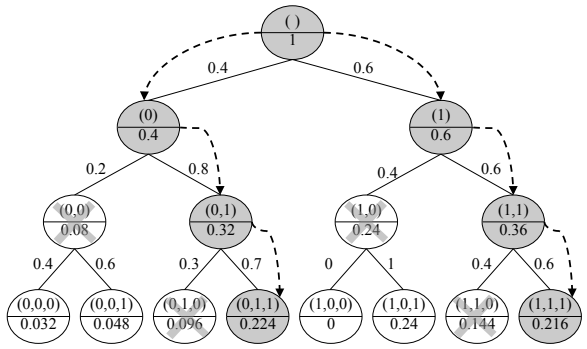
The Greedy Search (GS) strategy defines the original CC method [Read *et al.*, 2009; 2011]. CC provides an output $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_m)$ for a new unlabeled instance $\boldsymbol{x}$ by successively querying each classifier f_j that estimates the probability $\mathbf{P}(y_j \,|\, \boldsymbol{x}, y_1, \ldots, y_{j-1})$. Notice that when $\hat{y}_j$ is estimated, f_j uses the predictions of the previous labels, $\hat{y}_1, \ldots, \hat{y}_{j-1}$. This means that, in terms of the probability binary tree defined above, GS only explores one path. In the example depicted in Figure 1, GS would select the rightmost leaf, so the optimal solution is not reached.

Concerning the optimization of subset 0/1 loss, a rigorous analysis [Dembczynski *et al.*, 2012b] establishes bounds for the performance of GS, concluding that it offers quite poor performance for this loss function. However, the work concludes stating that GS tries to optimize subset 0/1 loss, since it is more suitable for estimating the mode of the joint rather of the marginal conditional distribution.

Unlike GS that explores only one path, Exhaustive Search (ES) explores all possible paths in the tree. Thus, ES estimates the entire joint conditional distribution $\mathbf{P}(\cdot \,|\, \boldsymbol{x})$, providing Bayes optimal inference. For each combination of labels $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$ it computes $\mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x})$ and $L(\boldsymbol{y}, \boldsymbol{f}(\boldsymbol{x}))$ for all $f(\boldsymbol{x})$ and outputs the combination $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_m) = \boldsymbol{f}^*(\boldsymbol{x})$ with minimum risk for the given loss $L(\cdot, \cdot)$. By doing so, it generally improves in terms of performance, since it perfectly estimates the risk minimizer, albeit at the cost of a higher computational cost, as it comes down to summing over an exponential (2^m) number of label combinations. ES is the inference process used by the original PCC method [Dembczyński *et al.*, 2010].



(a) ϵ-Aproximate with $\epsilon = .0$



(b) Beams Search with $b = 2$

Figure 1: An example of the paths followed by ϵ-A ($\epsilon = .0$) and BS ($b = 2$). The dotted arrows show the path followed by the algorithm. In the case of BS, the nodes with a cross are not explored anymore because they have not any of the highest marginal joint conditional probabilities at their level

3.2 ϵ-Approximate algorithm

The ϵ-Approximate (ϵ-A) algorithm [Dembczynski *et al.*, 2012b] arises as an alternative to the high computational cost of ES and to the poor performance of GS. In terms of the probability tree defined above, it expands only those nodes whose marginal joint conditional probability, $\mathbf{P}(y_1, \ldots, y_j \mid \boldsymbol{x})$, exceeds the threshold $\epsilon = 2^{-k}$, with $1 \leq k \leq m$. In fact, the nodes are expanded in the decreasing order established by this probability (see Figure 1(a)). Two situations can be found: i) the node expanded is a leaf or ii) there are not more nodes that exceed the threshold ϵ. If the former situation happens, the prediction for the unlabeled instance will be the label combination of such leaf. Conversely, if situation ii) takes place, then GS is applied to the those nodes whose children do not exceed the threshold, and the prediction will be that with the highest entire joint conditional probability $\mathbf{P}(y_1, \ldots, y_m \mid \boldsymbol{x})$.

The parameter ϵ plays a key role. The particular case of ϵ=0 (or any value in the interval $[0, 2^{-m}]$, that is, $k = m$) has special interest, since the algorithm performs a uniform-cost search (UC) and always finds the optimal solution. This is so because the marginal joint conditional probabilities for at least one leaf is guaranteed to be higher than ϵ. Even more, this leaf is the optimal solution, since the marginal joint conditional probabilities decrease in the successive steps of the algorithm. Figure 1(a) illustrates this situation. The dotted arrows show the path followed by the algorithm and at the end the optimal label combination in terms of subset 0/1 loss is reached $(1, 0, 1)$. Conversely, the method is looking to GS as ϵ grows, being GS in case of $k = 1, \epsilon = 2^{-1} = 0.5$.

Notice that this method provides an optimal prediction whenever its corresponding joint probability is greater or equal than ϵ. A more general interpretation for a generic value of $\epsilon = 2^{-k}$ is that the algorithm guarantees to reach an optimal solution until the k-th level on the tree. Unfortunately, this optimal combination of labels until k-th level can be different from the ones of the optimal solution, since such solution also depends on what happens hereinafter. From level $k+1$, if the entire joint conditional probability of the optimal solution is greater than ϵ, then none of its nodes is discarded and hence this optimal solution is reached. Conversely, if the entire joint conditional probability of the optimal solution is lower than ϵ, then the algorithm never reaches a leaf, and hence, GS is applied starting at each discarded node. Therefore, the optimal solution is reached only if GS follows the optimal path, which is not guaranteed.

Consequently, this algorithm estimates the risk minimizer for the subset 0/1 loss a greater or lesser extent depending on the value of ϵ. Even more, a theoretical analysis of this estimation [Dembczynski *et al.*, 2012b] allows to bound its goodness as a function of ϵ.

3.3 Beam search

Beam Search (BS) [Kumar *et al.*, 2012; 2013] also explores more than one path of the probabilistic tree. This method includes a parameter b called beam width that limits the number of label combinations explored. The idea is to explore b nodes at each level. Hence, depending on such value certain number of the top levels are exhaustively explored, particularly a total of k^*-1 levels, being k^* the lowest integer such that $b < 2^{k^*}$. Then, only b nodes are explored for each of the remainder levels. The nodes explored from the level k^* to the bottom are those with the highest marginal joint conditional probability seen thus far. At the end, the algorithm outputs the label combination with highest joint conditional probability.

Figure 1(b) shows an example of the BS algorithm with $b = 2$. Hence, all nodes of the first level are explored. For the rest of levels, just two nodes are explored, those with highest marginal joint conditional probability among the children whose parents have been previously explored. At this respect, notice that the node with the optimal solution is not explored since its parent was discarded. This example confirms that BS can not guarantee to reach optimal solutions.

In the case of b=1, BS expands just one node at each level, that with highest marginal joint conditional probability, so BS follows just one path and is equivalent to GS. Also, if $b = 2^m$, BS performs an ES. Hence, BS encapsules both GS and ES. This makes possible to control the trade-off between computational cost and performance of the method by tuning b.

As final remark, the fact that BS considers marginal joint conditional probabilities makes the method tend to estimate the risk minimizer for the subset 0/1 loss. In any case, it would be possible to consider other loss functions. At this respect, the authors of this proposal do not include any theoretical analysis about the goodness of the estimation of the risk minimizer. However, they empirically show that taking certain values of b ($b < 15$), the risk minimizer provided by the method converges to the one obtained using ES.

4 A* algorithm for inference in PCC

The algorithm A* is the most widely-known form of best-first search [Pearl, 1984] in which, at each iteration, the *best* node according to an evaluation function e is expanded. The particularity of A* algorithm is that e can be seen as a function E of other two functions g and h, $e(k) = E(g(k), h(k))$, where $g(k)$ evaluates the cost of reaching node k from the root and $h(k)$ evaluates the cost of reaching a solution (a leaf) from k. Hence, $e(k)$ evaluates the total cost of reaching a solution from the root through k. In general, it is possible to obtain the exact value of the known information (g), but the unknown information must be estimated through an heuristic (h). To get an optimal solution, h must not overestimate the actual cost of reaching a solution, that is, it must be an admisible heuristic. This kind of heuristics are optimistic, because they estimate that the cost of getting a solution is less than it actually is. Also, A* algorithm is optimally efficient for any heuristic, because no other optimal algorithm using the same heuristic guarantees to expand fewer nodes than A*.

In order to adapt A* for inference in PCC, we must take into account that we have probabilities instead of costs. So, 1) A* must select the node with the highest estimated probability, 2) h must not underestimate the probability from the node to a leaf, and 3) E must be the product function, $e = g \cdot h$. Considering all these aspects, e will provide an estimation of the entire joint conditional probability $\mathbf{P}(y_1, \ldots, y_m \,|\, \boldsymbol{x})$ for optimizing subset 0/1 loss. In order to derive g and h, let see that the product rule of probability to the joint distribution of the labels $\mathbf{Y} = (\mathbf{Y_1}, \mathbf{Y_2}, \ldots, \mathbf{Y_m})$ can be rewritten as

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

Hence, for a node at level j, let consider g to be the marginal joint conditional probability $\mathbf{P}(y_1, \ldots, y_j \,|\, \boldsymbol{x})$ and h an heuristic that does not underestimate the marginal joint conditional probability $\mathbf{P}(y_{j+1}, \ldots, y_m \,|\, \boldsymbol{x}, y_1, \ldots, y_j)$. Let remember that the values of $y_1, \ldots, y_j$ are known at level j.

Before discussing the heuristic h proposed here, let notice that g is the same marginal joint conditional probability that both ϵ-A algorithm and BS calculate to select the nodes to be expanded. Even more, ϵ-A with $\epsilon = 0$ is not only equivalent to UC, but also to A* with the constant heuristic $h = 1$. This heuristic is admisible too, since none probability is greater than 1. But, on the other hand, it is also the worst admisible heuristic, since any other admisible heuristic will dominate it and consequently the A* algorithm using such heuristic will never expand more nodes than A* algorithm using $h = 1$.

Let go now to discuss the heuristic proposed here for guaranteeing optimal predictions. Since our heuristic should not underestimate the marginal joint conditional probability $\mathbf{P}(y_{j+1}, \ldots, y_m \,|\, \boldsymbol{x}, y_1, \ldots, y_j)$ in order to be admisible, it is quite straightforward to pick the maximum value of such probability for obtaining an optimal heuristic h^*:

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

However, obtaining such maximum is, in fact, applying an ES over the set of labels $\mathcal{L} = \{\ell_{j+1}, \ldots, \ell_m\}$. Hence, this optimal heuristic is not computationally applicable. So, we need to obtain a tractable heuristic in exchange of renouncing to such optimality. This leads to design an heuristic $\hat{h}$ also admisible, but less dominant than h^*. For this purpose, let consider $(\overline{y}_{j+1}, \ldots, \overline{y}_m)$ the values that define h^*, that is

$$h^* = \prod_{i=j+1}^{m} \mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, \overline{y}_{j+1}, \ldots, \overline{y}_{i-1}),$$

and let notice that values $(\overline{y}_{j+1}, \ldots, \overline{y}_m)$ that maximize the product do not have to maximize each individual term, then

$$\begin{aligned}&\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, \overline{y}_{j+1}, \ldots, \overline{y}_{i-1}) \leq \\ &\max_{\substack{(y_{j+1}, \ldots, y_{i-1}) \\ \in \{0,1\}^{i-1-j}}} \mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1}).\end{aligned}$$

Hence, defining $\hat{h}$ as

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

it is easy to deduce that $\hat{h}$ is admisible and $h^* \leq \hat{h}$. But again, $\hat{h}$ is not computationally applicable in general. However, this is not the case if we restrict ourselves to the case of using linear models. Remember that $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1})$ is estimated through the model $f_i(\boldsymbol{x}, y_1, \ldots, y_{i-1})$. Particularly, $\mathbf{P}(y_i = 1 \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1}) = f_i(\boldsymbol{x}, y_1, \ldots, y_{i-1})$ and $\mathbf{P}(y_i = 0 \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1}) = 1 - f_i(\boldsymbol{x}, y_1, \ldots, y_{i-1})$.

Hence, if f_i is a linear model, it adopts the following form

$$f_i(\boldsymbol{x}, y_1, \ldots, y_{i-1}) = \langle \boldsymbol{w}_x, \boldsymbol{x} \rangle + \langle \boldsymbol{w}_y, (y_1, \ldots, y_{i-1}) \rangle + \beta,$$

that splitting the second term in the known part (from ℓ_1 to ℓ_j) and unknown part (from ℓ_{j+1} to ℓ_i) it leads to

$$f_i(\boldsymbol{x}, y_1, \ldots, y_{i-1}) = \langle \boldsymbol{w}_x, \boldsymbol{x} \rangle + \sum_{k=1}^{j} w_{y,k} y_k + \sum_{k=j+1}^{i-1} w_{y,k} y_k + \beta.$$

Since $\boldsymbol{x}$ is given and $y_1, \ldots, y_j$ are fixed, the last summation contains the variables for which the maximum must be obtained. Then, let denote by $K^+ = \{k \,|\, j+1 \leq k \leq i-1,\ w_{y,k} \geq 0\}$ the indexes of the positive coefficients and by $K^- = \{k \mid j+1 \leq k \leq i-1,\ w_{y,k} < 0\}$ the indexes of the negatives. Hence $f_i(\boldsymbol{x}, y_1, \ldots, y_{i-1})$ is maximum when

$$y_k = \begin{cases} 1 & \text{if } k \in K^+ \\ 0 & \text{if } k \in K^-, \end{cases}$$

and similarly $1 - f_i(\boldsymbol{x}, y_1, \ldots, y_{i-1})$ is maximum if

$$y_k = \begin{cases} 1 & \text{if } k \in K^- \\ 0 & \text{if } k \in K^+. \end{cases}$$

Consequently, if $y_{j+1}^{i,1}, \ldots, y_{i-1}^{i,1}$ are the values that maximize $f_i(\boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1})$ and if $y_{j+1}^{i,0}, \ldots, y_{i-1}^{i,0}$ are the values that maximize $1 - f_i(\boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1})$, then

$$y_k^i = \begin{cases} 1 & \text{if } 1 - f_i(\boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}^{i,0}, \ldots, y_{i-1}^{i,0}) \leq \\ & \quad f_i(\boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}^{i,1}, \ldots, y_{i-1}^{i,1}) \\ 0 & \text{otherwise.} \end{cases}$$

Notice that $y_k^{i,1} = 1 - y_k^{i,0}$. Also, if $y_{j+1}^i, \ldots, y_{i-1}^i$ equals either $y_{j+1}^{i,1}, \ldots, y_{i-1}^{i,1}$ or $y_{j+1}^{i,0}, \ldots, y_{i-1}^{i,0}$ respectively depending if $1 - f_i(\boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}^{i,0}, \ldots, y_{i-1}^{i,0})$ is lower than $f_i(\boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}^{i,1}, \ldots, y_{i-1}^{i,1})$ or not, then the heuristic $\hat{h}$ for a node at level j in the linear case will be

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

Remember that h^* and $\hat{h}$ only differ on the values of $y_{j+1}, \ldots, y_{i-1}$. In the former, the same values are common for all the factors of the product, whereas in the latter these values depend on each term i of the product, and hence, they can be different, what makes $\hat{h}$ not be an optimal heuristic. On the other hand, the cost of computing $\hat{h}$ is of polynomial order, unlike h^* which is of exponential order. Figure 2 exemplifies $\hat{h}$. Let focus on the node labeled by (1). The marginal joint conditional probability until this node is $g = 0.6$. For computing $\hat{h}$, first we evaluate the maximum marginal conditional probabilities $P(y_2 \,|\, \boldsymbol{x}, y_1 = 1)$ provided by $h_2(\boldsymbol{x}, y_1)$ and $P(y_3 \,|\, \boldsymbol{x}, y_1 = 1, y_2)$ provided by $h_3(\boldsymbol{x}, y_1, y_2)$ which respectively are 0.6 and 1.0. Then we carry out their product as applying the product rule of the probability to estimate the marginal joint conditional probability from that node to a leaf. Notice that the maximum at each level does not correspond to the same branch of the tree. In other words, the maximum in the first level correspond to $y_2 = 1$, whereas the maximum in the second level is obtained by $P(y_3 \,|\, \boldsymbol{x}, y_1 = 1, y_2 = 0)$ when $y_3 = 1$ fixing $y_2 = 0$. This is what makes the heuristic $\hat{h}$ not to be the optimal one h^*.

Figure 3 shows the path followed by A* using $\hat{h}$. It reaches the optimal leaf as theoretically is expected. Comparing this graph with the one in Figure 1(a) that illustrates ϵ-A with $\epsilon = .0$, and taking into account the properties of the heuristics related to the dominance, ϵ-A, $\epsilon = .0$ (equivalent to A* using $h = 1$ or UC search) explores more nodes than A* using $\hat{h}$.

As final remark, the A* algorithm using $\hat{h}$ perfectly estimates the risk minimizer for the subset 0/1 loss, as both ϵ-A with $\epsilon = .0$ and ES do it. Even more, A* with $\hat{h}$ expands equal or less nodes, since $\hat{h}$ is more dominant than the heuristic $h = 1$. Obviously, computing $\hat{h}$ is more costly than computing $h = 1$ or applying just uniform-cost search. The question is then if this additional computing time compensates the theoretical guarantee it has of expanding less nodes.

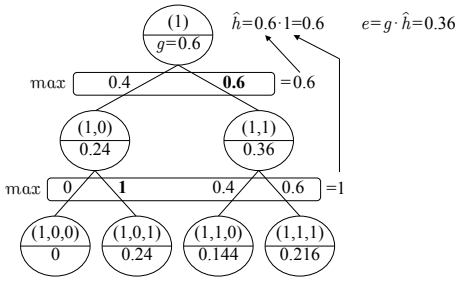


Figure 2: Computation of heuristic $\hat{h}$

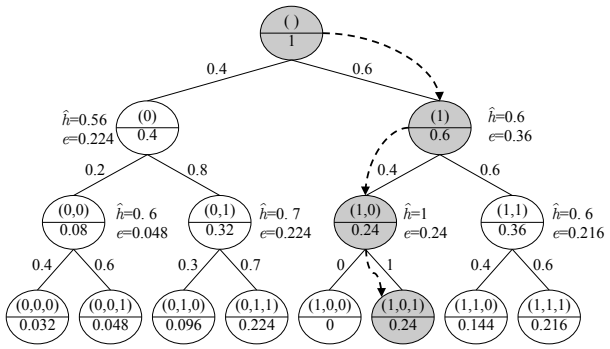


Figure 3: An example of A* using the heuristic $\hat{h}$. The dotted arrows show the path followed by the algorithm. The values of g are provided inside each node

5 Experimental results

The experiments reported here were performed over several benchmark datasets previously used in many MLC papers. The main characteristic for these experiments is the number of labels, which varies between 5 and 101. The methods compared were those discussed along the paper, except ES because it is computationally intractable. We do not include other approaches, like Monte Carlo sampling, due to the lack of space to properly discuss them in the paper. Hence, the methods compared were GS, uniform-cost search, ϵ-A for different values of $\epsilon \in \{.0, .25, .5\}$, BS for several values of beam width $b \in \{1, 2, 3\}$ and A* with the linear heuristic $\hat{h}$. Recall that ϵ-A($\epsilon = .0$) is equivalent to UC, whereas ϵ-A($\epsilon = .5$) is equivalent to GS and to BS with $b = 1$. The results will be presented in terms of the subset 0/1 loss estimated by means of a 10-fold cross-validation.

The base learner employed to obtain the binary classifiers f_i was *logistic regression* [Lin *et al.*, 2008] with probabilistic output. The regularization parameter C was established for each *individual* binary classifier performing a grid search over the values $C \in \{10^{-3}, 10^{-2}, \ldots, 10^2, 10^3\}$ optimizing the brier loss [Brier, 1950] estimated by means of a balanced 2-fold cross validation repeated 5 times. This loss function is a proper score that measures the accuracy of probabilistic predictions, as those provided by *logistic regression*.

Table 1 shows subset 0/1 scores. UC, ϵ-A(.0) and A*($\hat{h}$)

Datasets	UC/A*($\hat{h}$) ϵ-A(.0)	ϵ-A(.25)	GS/BS(1) ϵ-A(.5)	BS(2)	BS(3)
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**
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**
slashdot	**51.80**	52.22	54.49	**51.77**	**51.80**
yeast	**76.95**	77.62	79.77	**76.83**	77.08

Table 1: Subset 0/1 scores. Those scores that are equal or better than optimal predictions are shown in bold

provide optimal predictions in terms 0/1 subset loss, given the f_i classifiers learned. They are shown all together in the first column. As expected, the performance of ϵ-A algorithm decreases as the value of ϵ increases (.25 or .5). These two versions combined perform worse than UC, ϵ-A(.0) and A*($\hat{h}$) in 13 out of 16 cases. On the other hand, the performance of BS converges to the optimal one as the value of b increases. For BS(2) there are four datasets in which its scores are worse than those of A*($\hat{h}$), UC and ϵ-A(.0) and in the other four its results are equal or better. Notice that the differences are larger when BS(2) performs worse. In the case of BS(3), in five datasets its scores are exactly the same than those of UC, ϵ-A(.0) and A*($\hat{h}$), showing the convergence of the method. We do not run further experiments to study this aspect, because as we shall analyze later, the number of nodes explored by BS(3) is greater than those of the optimal methods.

Table 2 contains the number of nodes explored by each method and average prediction times. GS (or ϵ-A(.5) and BS(1)) is, of course, the method which explores the least number of nodes, since it only goes over one path in the tree. Such number corresponds to the number of labels plus one, because the root node is also explored. In the comparison between those methods that do not return optimal predictions, it seems that ϵ-A(.25) tends to explore less nodes than BS(2), but it is worse in terms of subset 0/1 loss. BS(3) is the algorithm that expands the most number of nodes.

Focusing on the methods that return optimal predictions, A*($\hat{h}$) explores less nodes, as we proved theoretically before. In fact, it explores a number of nodes lower than two times the number of labels (those of BS(2)), and sometimes it is quite close to the number of nodes explored by GS. This suggest that $\hat{h}$ seems a quite good heuristic. However, the difference with respect to ϵ-A(.0) or UC is surprisingly small. The number of nodes explored by these two algorithms is lower than expected. The reason may be that the predictions of classifiers f_i along the optimal path are usually close to 1 or to 0. There is not much uncertainty, so the level of backtracking is low. All these results suggest that the inference in PCC over these concrete MLC datasets, analyzed as a searching problem, is a quite easy task. In this sense, it is worth noting that those approaches based on sampling techniques, like Monte Carlo sampling, should use very small samples in order to be competitive over these benchmark datasets, further considering that such methods do not guarantee optimal predictions.

Datasets	A*($\hat{h}$)	UC ϵ-A(.0)	ϵ-A(.25)	GS/BS(1) ϵ-A(.5)	BS(2)	BS(3)
emotions(6)	7.5	10.7	10.8	7.0	13.0	18.0
enron(53)	78.2	114.8	77.3	54.0	107.0	159.0
flags(7)	9.0	22.6	16.3	8.0	15.0	21.0
mediamill(101)	180.9	191.8	142.4	102.0	203.0	303.0
medical(45)	46.1	46.6	46.6	46.0	91.0	135.0
reuters(7)	8.2	8.2	8.3	8.0	15.0	21.0
slashdot(22)	24.8	25.3	24.9	23.0	45.0	66.0
yeast(14)	21.2	26.1	26.0	15.0	29.0	42.0
emotions(6)	.0011	.0006	.0006	.0004	.0005	.0006
enron(53)	.1084	.0072	.0030	.0019	.0039	.0055
flags(7)	.0015	.0012	.0007	.0004	.0005	.0007
mediamill(101)	.4847	.0115	.0055	.0037	.0072	.0105
medical(45)	.0495	.0027	.0027	.0027	.0033	.0046
reuters(7)	.0013	.0005	.0005	.0005	.0005	.0007
slashdot(22)	.0124	.0015	.0014	.0012	.0016	.0022
yeast(14)	.0075	.0015	.0010	.0006	.0010	.0014

Table 2: Average number of explored nodes (top) and average prediction time (in seconds) per example (bottom). The number of labels of each dataset is between parentheses

Looking at prediction times, interestingly ϵ-A(.0) and UC are not much slower than ϵ-A(.25) and ϵ-A(.5) and similar to BS(2) and BS(3). Moreover, UC and ϵ-A(.0) are faster than A*($\hat{h}$) despite they explore more nodes. The reason is obviously the computational cost of $\hat{h}$, which is due to the number of probabilities that must be computed $\{\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1}) | i = j+1..m\}$, see (1). An alternative is to reduce the number of probabilities calculated, for instance, only the next d probabilities. Parameter d would allow us to control computational complexity.

6 Conclusions and future work

This paper proposes a new approach based on A* algorithm with an admisible heuristic for inferring predictions in PCC. Hence, our proposal A*($\hat{h}$) provides optimal predictions in terms of subset 0/1 loss and explores less nodes than previous approaches that also produce optimal predictions. A*($\hat{h}$) is limited to linear models, but this is not a major drawback because it is quite common to employ linear models. This is usually the case of MLC problems with many labels in which the examples of some classes may be scarce, since using non-linear models in such problems can lead to overfitting.

In the experiments performed, uniform-cost search (or ϵ-A($\epsilon = .0$)) is a better choice than A*($\hat{h}$) since the computational cost of evaluating the heuristic does not compensate the reduction in nodes explored. In any case, the benchmark datasets employed here seem quite simple and these methods should be tested over more complex problems. At this respect, approaches like beam search, Monte Carlo sampling and our proposal begin to have sense for more complex and difficult –as inference problems– MLC tasks.

In our opinion, heuristic search is a promising line of research for inference in PCC. As future work, new heuristics can be devised, keeping in mind that we should look for a trade off between their computational complexity and the quality of the estimations. For instance, limiting the depth of the heuristic may be a plausible alternative.

References

[Brier, 1950] Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Mon. Wea. Rev.*, 78(1):1–3, January 1950.

[Cheng and Hüllermeier, 2009] W. Cheng and E. Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225, 2009.

[Dembczyński *et al.*, 2010] K. Dembczyński, W. Cheng, and E. Hüllermeier. Bayes Optimal Multilabel Classification via Probabilistic Classifier Chains. In *ICML, 2010*, pages 279–286, 2010.

[Dembczyński *et al.*, 2012a] 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.

[Dembczynski *et al.*, 2012b] 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.

[Godbole and Sarawagi, 2004] S. Godbole and S. Sarawagi. Discriminative methods for multi-labeled classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (2004)*, pages 22–30, 2004.

[Hart *et al.*, 1968] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, SSC-4(2):100–107, 1968.

[Kumar *et al.*, 2012] Abhishek Kumar, Shankar Vembu, Aditya Krishna Menon, and Charles Elkan. Learning and inference in probabilistic classifier chains with beam search. In *ECML/PKDD 2012*, pages 665–680, 2012.

[Kumar *et al.*, 2013] Abhishek Kumar, Shankar Vembu, Aditya Krishna Menon, and Charles Elkan. Beam search algorithms for multilabel learning. *Mach. Learn.*, 92(1):65–89, July 2013.

[Lin *et al.*, 2008] 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.

[Montañés *et al.*, 2011] E. Montañés, J.R. Quevedo, and J. J. del Coz. Aggregating independent and dependent models to learn multi-label classifiers. In *ECML/PKDD'11 - Volume Part II*, pages 484–500. Springer-Verlag, 2011.

[Montañés *et al.*, 2014] 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.

[Pearl, 1984] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984.

[Read *et al.*, 2008] J. Read, B. Pfahringer, and G. Holmes. Multi-label classification using ensembles of pruned sets. In *IEEE Int. Conf. on Data Mining*, pages 995–1000. IEEE, 2008.

[Read *et al.*, 2009] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. In *ECML/PKDD'09*, LNCS, pages 254–269. Springer, 2009.

[Read *et al.*, 2011] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359, 2011.

[Read *et al.*, 2014] Jesse Read, Luca Martino, and David Luengo. Efficient monte carlo methods for multi-dimensional learning with classifier chains. *Pattern Recognition*, 47(3):1535 – 1546, 2014.

[Tsoumakas and Vlahavas, 2007] G. Tsoumakas and I. Vlahavas. Random k-Labelsets: An Ensemble Method for Multilabel Classification. In *ECML/PKDD'07*, LNCS, pages 406–417. Springer, 2007.