A Family of Admissible Heuristics for A* to perform Inference in Probabilistic Classifier Chains

Deiner Mena
Elena Montañés
José Ramón Quevedo
Juan José del Coz

Abstract Probabilistic Classifiers Chains (PCCs) have recently gained interest in multi-label classification, due to their ability to optimally estimate the joint probability of a set of labels. The main hindrance is the excessive computational cost of performing inference in the prediction stage. This pitfall has opened the door to propose efficient inference alternatives that avoid exploring all the possible solutions. The $\epsilon-$approximate algorithm, beam search and Monte Carlo sampling are appropriate techniques, but only $\epsilon-$approximate algorithm with $\epsilon = 0$ theoretically guarantees reaching an optimal solution in terms of subset 0/1 loss. This paper offers another alternative based on heuristic search that keeps such optimality. It consists of applying the A* algorithm providing an admissible heuristic able to explore fewer nodes than the $\epsilon-$approximate algorithm with $\epsilon = 0$. A preliminary study has already coped with this goal, but at the expense of the high computational time of evaluating the heuristic and only for linear models. In this paper, we propose a family of heuristics defined by a parameter that controls the trade-off between the

Deiner Mena
Artificial Intelligence Center, Campus de Viesques, s/n, 33204, University of Oviedo, Gijón, Asturias, Spain
Universidad Tecnológica del Chocó, Colombia
E-mail: deiner@aic.uniovi.es/deiner.mena@utch.edu.co

Elena Montañés
Artificial Intelligence Center, Campus de Viesques, s/n, 33204, University of Oviedo, Gijón, Asturias, Spain
E-mail: elena@aic.uniovi.es

José Ramón Quevedo
Artificial Intelligence Center, Campus de Viesques, s/n, 33204, University of Oviedo, Gijón, Asturias, Spain
E-mail: quevedo@aic.uniovi.es

Juan José del Coz
Artificial Intelligence Center, Campus de Viesques, s/n, 33204, University of Oviedo, Gijón, Asturias, Spain
E-mail: juanjo@aic.uniovi.es

number of nodes explored and the cost of computing the heuristic. Besides, a certain value of the parameter provides a method that is also suitable for the non-linear case. The experiments reported over several benchmark datasets show that the number of nodes explored remains quite steady for different values of the parameter, although the time considerably increases for high values. Hence, low values of the parameter give heuristics that theoretically guarantee exploring fewer nodes than the $\epsilon-$approximate algorithm with $\epsilon=0$ and show competitive computational time. Finally, the results exhibit the good behavior of the A* algorithm using these heuristics in complex situations such as the presence of noise.

Keywords Multi-label classification · Probabilistic classifier chains · Inference · A* · Admissible heuristics

1 Introduction

Multi-label classification (MLC) is a machine learning problem in which models are sought that assign a subset of (classes) labels to each instance, unlike conventional (single-class) classification that involves predicting only a single class. Multi-label classification problems are ubiquitous and naturally occur, for instance, in assigning keywords to a paper, tags to resources in a social network, objects to images or emotional expressions to human faces.

In general, the problem of multi-label learning comes with two fundamental challenges. The first refers to the computational complexity of the algorithms. If the number of labels is large, then a complex approach might not be applicable in practice. Therefore, the scalability of algorithms is a key issue in this field. The second problem is related to the 'own nature' of multi-label data. Not only is the number of labels typically large, but each instance also belongs to a variable-sized subset of labels simultaneously. Moreover, and perhaps even more importantly, the labels will normally not occur independently of each other; instead, there are statistical dependencies between them. From a learning and prediction point of view, these relationships constitute a promising source of information, in addition to that coming from the mere description of the instances. Thus, it is hardly surprising that research on MLC has very much focused on the design of new methods that are able to detect—and benefit from—interdependencies among labels.

Several approaches have been proposed in the literature to cope with MLC. Firstly, researchers tried to adapt and extend different state-of-the-art binary or multi-class classification algorithms [6], [11] and [25]. Secondly, they further analyzed in depth the label dependence and attempt to design new approaches that exploit label correlations [4]. In this regard, two kinds of label dependence have been formally distinguished, namely, conditional dependence [3], [14], [15] and [20] and marginal (unconditional) dependence [2]. Also, pairwise relations [6], relations in sets of different sizes [20] and [24], or relations in the whole set of labels [2] and[14] have also been exploited.

Regarding conditional label dependence, the approach called Probabilistic Classifier Chains (PCC) has aroused great interest among the multi-label community, since it offers the excellent property of being able to estimate the conditional joint distribution of the labels. However, the original PCC algorithm [3] suffers from high computational cost, since it performs an exhaustive search as inference strategy to obtain optimal solutions in terms of a given loss function. Several efforts that use different searching and sampling strategies in order to overcome this drawback are being made currently. This includes uniform-cost search [5], beam search [8] and [9] and Monte Carlo sampling [5] and [18]. All of these algorithms successfully estimate an optimal solution reached by the original PCC [3], at the same time that they reduce the computational cost in terms of both the number of candidate solutions explored and execution time. The main contribution of this paper is to propose an alternative based on an heuristic search strategy. In particular, the proposal consists of obtaining admissible heuristics for the well-known A* algorithm [7]. In this respect, we have already started to fill this gap in the literature with a recent published preliminary work [13], concluding that the proposal guarantees, not only optimal predictions in terms of subset 0/1 loss, but also that it explores fewer nodes than all previous methods that also provide optimal predictions. Unfortunately, and after studying in depth this heuristic, two main drawbacks can be stated: i) it could only be defined for linear models and ii) its computation is moderately high. In this direction, the goal of this paper is twofold. On the one hand, this work goes further by defining a family of heuristics through a parameter that controls the trade-off between the number of nodes explored and the cost of computing the heuristic. Besides, a special value of the parameter leads to an heuristic suitable for non-linear models. On the other hand, this work also studies and analyzes situations in which the computation of the heuristic compensates the whole computational cost, showing a steady behavior with regard to other algorithms. All these methods are analyzed and experimentally compared over a wide range of multi-label datasets.

The rest of the paper is organized as follows. Section 2 formally describes the multi-label framework and the principles of PCC. Section 3 describes and discusses the properties and behavior of the different state-of-the-art approaches for inference in PCCs. Section 4 details the heuristic search framework and defines admissible heuristics for the A* algorithm. Exhaustive experiments are shown and discussed in Section 5. Finally, Section 6 offers some conclusions and includes new directions for future work.

2 Probabilistic Classifier Chains in Multi-label Classification

This section formally describes the MLC task and the PCC methods.

2.1 Formal Settings of Multi-label Classification and Loss Functions

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 is the space of the instance description, 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 hypothesis $\boldsymbol{f} : \mathcal{X} \longrightarrow \mathcal{Y}$ that minimizes the risk in terms of certain loss function $L(\cdot)$ when it provides a vector of relevant labels $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$ for unlabeled query instances $\boldsymbol{x}$. This risk can be defined as the expected loss over the joint distribution $\mathbf{P}(\mathbf{X}, \mathbf{Y})$, that is,

$$R_L(\boldsymbol{f}) = \mathbb{E}_{\mathbf{X},\mathbf{Y}} L(\mathbf{Y}, \boldsymbol{f}(\mathbf{X})). \tag{1}$$

therefore, denoted by $\mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x})$ the conditional distribution $\mathbf{Y} = \boldsymbol{y}$ given $\mathbf{X} = \boldsymbol{x}$, then the so-called risk minimizer $\boldsymbol{f}^*$ can be expressed by

$$\boldsymbol{f}^*(\boldsymbol{x}) = \arg\min_{\boldsymbol{f}} \sum_{\boldsymbol{y} \in \mathcal{Y}} \mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x}) L(\boldsymbol{y}, \boldsymbol{f}(\boldsymbol{x})). \tag{2}$$

Let us comment that the conditional distribution $\mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x})$ presents different properties which are crucial for optimizing different loss functions. In this respect, the strategy followed by a certain MLC algorithm for modeling label dependence determines the loss function that is optimized. But, unfortunately, for most of the algorithms it is quite complex and confusing to discover the loss function they attempt to optimize.

With regard to the loss functions, several performance measures have been taken for evaluating MLC. The most specific ones are the subset 0/1 loss and the Hamming loss, but there exist other measures that have been taken from other research fields, as such F_1 or the *Jaccard* index. Here, we will focus on just the subset 0/1 loss, since it is the measure PCCs are able to optimize. The subset 0/1 loss checks if the predicted and relevant label subsets are equal or not and is defined by[1]

$$L_{S_{0/1}}(\boldsymbol{y}, \boldsymbol{f}(\boldsymbol{x})) = [\![\boldsymbol{y} \neq \boldsymbol{f}(\boldsymbol{x})]\!]. \tag{3}$$

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

$$\boldsymbol{f}^*(\boldsymbol{x}) = \arg\max_{\boldsymbol{y} \in \mathcal{Y}} \mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x}). \tag{4}$$

[1] For a predicate p, the expression $[\![p]\!]$ evaluates to 1 if p is true and 0 otherwise.

2.2 Probabilistic Classifier Chains

PCCs [3] (such as CC [19] and [20]) are based on learning a chain of classifiers. These methods take an order of the label set and train 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. Hence, the probabilistic model obtained for predicting label ℓ_j, denoted by f_j is of the form

$$f_j : \mathcal{X} \times \{0,1\}^{j-1} \longrightarrow [0,1]. \tag{5}$$

The training data for this classifier is the set $S_j = \{(\overline{\boldsymbol{x}}_1, y_{1,j}), \ldots, (\overline{\boldsymbol{x}}_n, y_{n,j})\}$ where $\overline{\boldsymbol{x}}_i = (\boldsymbol{x}_i, y_{i,1}, \ldots, y_{i,j-1})$, 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 the label ℓ_j.

In the testing stage of the methods based on learning a chain of classifiers, the goal is 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 general 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

$$\mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x}) = \prod_{j=1}^{m} \mathbf{P}(y_j \,|\, \boldsymbol{x}, y_1, \ldots, y_{j-1}). \tag{6}$$

Before analyzing this issue in the next section, note that from a theoretical point of view, this expression holds for any order considered for the labels. But, in practice, these methods are label order dependent for several reasons. On the one hand, it is not possible to assure that the models obtained in the training stage perfectly estimate the joint conditional probability $P(\boldsymbol{y} \,|\, \boldsymbol{x})$. On the other hand, predicted values instead of true values are successively taken in the testing stage. This is more serious if the highest errors occur at the beginning of the chain, since error predictions are successively propagated [15], [22] and [23]. In any case, in this paper we assume the order of the labels in the chain to be given, since the goal is just to analyze the performance of the methods, without taking into account the effect of different orders. Hence, we do not include any study about which order can be the best.

Before going into the detailed description of the state-of-the-art of the inference approaches and of our proposal, we note that the training phase is common to all of them, thus, the models f_j induced by the binary classifiers will be the same. So, in what follows we will focus just on the testing stage.

3 Inference in Probabilistic Classifier Chains

First of all, let us consider the task of performing different inference procedures as different manners of exploring a probability binary tree in order to facilitate the explanation and analysis of the inference approaches in the next section.

In such a tree, the k-th node 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 parent 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 is computed by the product rule of probability. Then, $\mathbf{P}(y_1 = v_1, \ldots, y_j = v_j, y_{j+1} = 0 \,|\, \boldsymbol{x}) = \mathbf{P}(y_{j+1} = 0 \,|\, \boldsymbol{x}, y_1 = v_1, \ldots, y_j = v_j) \cdot \mathbf{P}(y_1 = v_1, \ldots, y_j = v_j \,|\, \boldsymbol{x})$ and $\mathbf{P}(y_1 = v_1, \ldots, y_j = v_j, y_{j+1} = 1 \,|\, \boldsymbol{x}) = \mathbf{P}(y_{j+1} = 1 \,|\, \boldsymbol{x}, y_1 = v_1, \ldots, y_j = v_j) \cdot \mathbf{P}(y_1 = v_1, \ldots, y_j = v_j \,|\, \boldsymbol{x})$. The root node is labeled by the empty set. Figure 1 illustrates this.

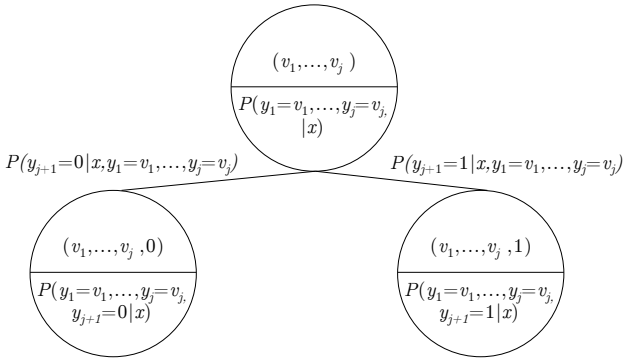


Fig. 1 A generic node and its children of the probability binary tree. The top part of each node contains the combination of labels and the bottom part includes the joint probability of such a combination. The edges are labeled with the conditional probability.

Several approaches have been proposed for inference in PCCs. The method the first proposed is the one based on greedy search (GS), being the integral part of the original CC method [19]. Its successor is the exhaustive search (ES), called the PCC method [3]. The $\epsilon-$approximate ($\epsilon-$A) algorithm [5] is a uniform-cost (UC) search algorithm that can output optimal predictions in terms of subset 0/1 loss and also reduces significantly the computational cost of ES. A more recent approach based on beam search [8] and [9] (BS) presents good behavior both in terms of performance and computational cost. Finally, Monte Carlo sampling [5] is an appealing and simpler alternative [5] and [18] to overcome the high computational cost of ES.

This section copes with the particularities of these inference methods, except the Monte Carlo sampling approaches. We have already studied Monte Carlo approaches [5] and [18] and compared them with the $\epsilon-$A algorithm and BS techniques [12]. The conclusions of that work were that, although they are well suited for minimization of other losses, e.g., Hamming loss or example-

based F-measure, i) they need to explore many more nodes to be closer to the optimal, despite the fact that their performance could sometimes be better, ii) they enlarge as the size of the sample drawn grows and iii) they are quite slow even for low values of the sample. Hence, we do not consider them as competitive methods in the present work, although they could sometimes be appealing.

3.1 Greedy Search

At the testing stage, the GS strategy, originally called CC [19] and [20], 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 conditional probability $\mathbf{P}(y_j \mid \boldsymbol{x}, y_1, \ldots, y_{j-1})$. This means exploring just one node in each level j. Given that only the two children of the explored node in level j are taken, their marginal joint conditional probability only differs in the marginal conditional probability $\mathbf{P}(y_j \mid \boldsymbol{x}, y_1, \ldots, y_{j-1})$, since both children have the same parent. Thus, the path selected is that of the child with the highest marginal conditional probability $\mathbf{P}(y_j \mid \boldsymbol{x}, y_1, \ldots, y_{j-1})$ and the prediction for an instance $\boldsymbol{x}$ is of the form

$$\hat{\boldsymbol{y}} = (\, f_1(\boldsymbol{x}),\, f_2(\boldsymbol{x}, f_1(\boldsymbol{x})),\, f_3(\boldsymbol{x}, f_1(\boldsymbol{x}), f_2(\boldsymbol{x}, f_1(\boldsymbol{x}))), \ldots). \tag{7}$$

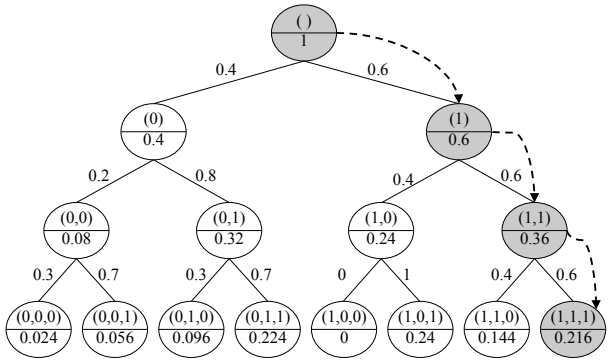
Figure 2(a) shows the path followed by an instance using this strategy. In this example, only the right node is explored in each level. The optimal solution is not reached, since the optimal solution is that which ends in the sixth leaf, whereas the method falls in the last leaf.

Concerning the optimization of subset 0/1 loss, a rigorous analysis [5] establishes bounds for the performance of the GS, showing the poor performance of the method for this loss, although it tends to optimize it.

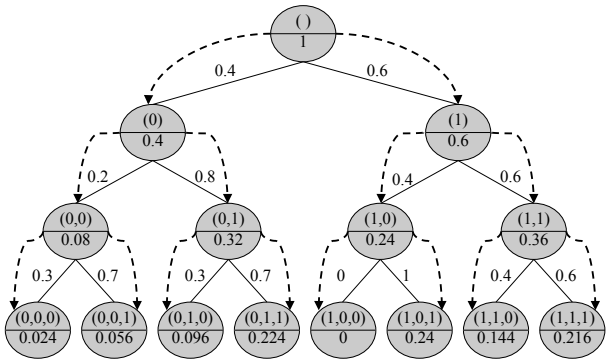
3.2 Exhaustive Search

Unlike GS that explores only a single label combination, ES estimates the entire joint conditional distribution $\mathbf{P}(\cdot \mid \boldsymbol{x})$ for a new unlabeled instance $\boldsymbol{x}$, since it provides a Bayes optimal inference. Hence, it explores all possible paths in the tree. Then, for each $h(\boldsymbol{x})$, it computes $\mathbf{P}(\boldsymbol{y} \mid \boldsymbol{x})$ and $L(\boldsymbol{y}, \boldsymbol{f}(\boldsymbol{x}))$ for all combination of labels $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$ 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 for each $h(\boldsymbol{x})$.

Figure 2(b) illustrates this approach where, by exploring all paths, the optimal solution is always reached.



(a) Greedy Search (CC)



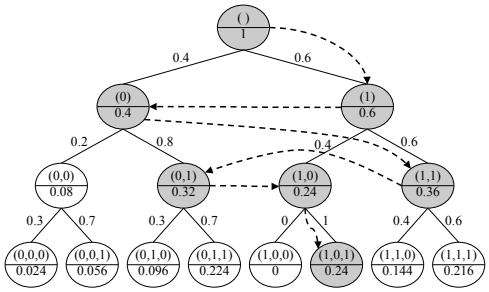
(b) Exhaustive Search (PCC)

Fig. 2 An example of paths followed by an instance using (a) Greedy Search (CC) and (b) Exhaustive Search (PCC). The dotted arrows show the path followed by the algorithm.

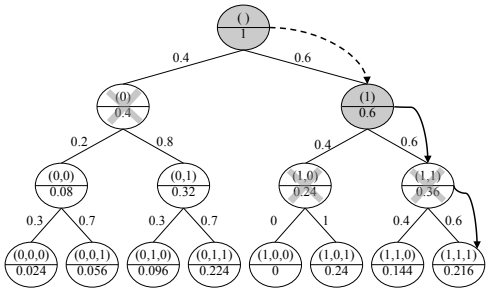
3.3 $\epsilon-$Approximate Algorithm

The $\epsilon-$Approximate ($\epsilon-$A) algorithm [5] arises as an alternative to the high computational cost of ES and to the poor performance of CC. In terms of the probability tree defined above, it expands only the nodes whose marginal joint conditional probability exceeds the threshold $\epsilon = 2^{-k}$ with $1 \leq k \leq m$ (notice that $\epsilon = 0$ and all values of ϵ between 0 and 2^{-m} are in fact the same case of $\epsilon = 2^{-m}$). This marginal joint conditional probability for a node in level j, which deals with the label ℓ_j and for an unlabeled instance $\boldsymbol{x}$ is

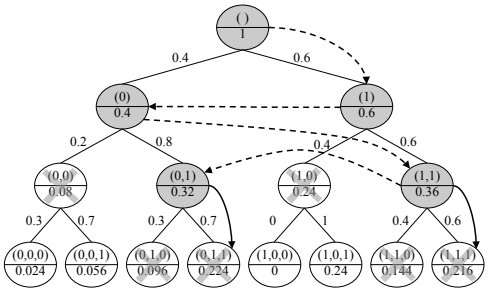
$$\mathbf{P}(y_1, \ldots, y_j \,|\, \boldsymbol{x}) = \prod_{i=1}^{j} \mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1}), \tag{8}$$



(a) $\epsilon-$A algorithm with ϵ=0 ($k = m$)



(b) $\epsilon-$A algorithm with $\epsilon = 0.5$ ($k = 1$)



(c) $\epsilon-$A algorithm with $\epsilon = 0.25$ ($k = 2$)

Fig. 3 Several examples of the paths followed by the $\epsilon-$A algorithm for different values of ϵ. The nodes with a cross are those that have a marginal joint conditional probability lower than ϵ and, hence, they are not explored any more. The dotted arrows show the path followed by the algorithm. The solid arrows indicate the path followed by the algorithm when the marginal joint conditional probability does not exceed the value of ϵ, and, hence a GS is applied to this node from here to the bottom of the tree.

where $\mathbf{P}(y_i, | \boldsymbol{x}, y_1, \ldots, y_{i-1})$ is estimated by $f_i(\boldsymbol{x}, y_1, \ldots, y_{i-1})$.

The nodes are expanded in the order established by this probability, calculating the marginal joint conditional probability for their children. So, the algorithm does not follow a specific path, otherwise it changes from one path to another depending on the marginal joint conditional probabilities. In the

end, two situations can be found: i) the node expanded is a leaf or ii) there are no more nodes that exceed the threshold. If the former situation happens, the prediction for the unlabeled instance $\boldsymbol{x}$ will be $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_m)$ corresponding to the combination of the leaf reached (see Figure 3(a)). Conversely, if situation ii) takes place, then GS is applied to the nodes whose children do not exceed the threshold, and the prediction $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_m)$ for the unlabeled instance $\boldsymbol{x}$ in this case will be that with the highest entire joint conditional probability $\mathbf{P}(y_1, \ldots, y_m \,|\, \boldsymbol{x})$ (see Figures 3(b) and 3(c)).

The parameter ϵ plays an important role in the algorithm. The particular case of $\epsilon = 0$ (or any value in the interval $[0,\, 2^{-m}]$, that is, $k = m$) is of special interest, since the algorithm performs a UC that always finds the optimal solution. Figure 3(a) illustrates this situation.

Conversely, the method is looking to GS as ϵ grows, being the GS in the case of $\epsilon = 0.5$ (or equivalently $\epsilon = 2^{-1}$, that is, $k = 1$). This is so because in this case two situations are possible: i) only one node has a marginal joint conditional probability greater than ϵ, in which case the algorithm follows one path, or ii) no nodes have a marginal joint conditional probability greater than ϵ, in which case a GS is applied from here to the bottom of the tree. Figure 3(b) shows an example of this particular case.

Notice that this method provides an optimal prediction if the entire joint conditional probability of the corresponding label combination is greater than ϵ. The interpretation of the method for a generic value of $\epsilon = 2^{-k}$ is that the method guarantees reaching a partial optimal solution at least until the $k-th$ level on the tree. Figure 3(c) shows the particular case of $\epsilon = 0.25$.

Consequently, this algorithm estimates the risk minimizer for the subset 0/1 loss to a greater or lesser extend, depending on the value of ϵ. Moreover, a theoretical analysis of this estimation [5] allows bounding its goodness as a function of the number of iterations, which in turn depends on ϵ.

3.4 Beam Search

Beam Search (BS) [8] and [9] also explores more than one path in the probabilistic tree. This method includes a parameter b called the beam width that limits the number of combinations of labels explored. The idea is to explore b possible candidate sets of labels at each level of the tree. Hence, depending on such a value, a certain number of the top levels are exhaustively explored, particularly a total of $k^* - 1$ levels, k^* being the lowest integer such that $b < 2^{k^*}$. Then, only b number of possibilities are explored for each of the remainder levels. The combinations explored from the level k^* to the bottom are those with the highest marginal joint conditional probability seen thus far. This marginal joint conditional probability for a node of level j for an unlabeled instance $\boldsymbol{x}$ is the same as for the $\epsilon-$A algorithm. Hence, such a probability is

$$\mathbf{P}(y_1, \ldots, y_j \,|\, \boldsymbol{x}) = \prod_{i=1}^{j} \mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1}), \tag{9}$$

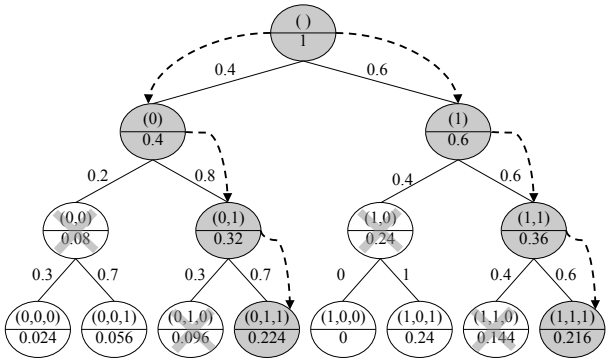


Fig. 4 An example of paths followed by an instance using Beam Search (BS) with $b = 2$. The nodes with a cross mean that this node has none of the highest marginal joint conditional probabilities and, hence, it is not explored any more. The dotted arrows show the path followed by the algorithm.

where $\mathbf{P}(y_i, | \boldsymbol{x}, y_1, \ldots, y_{i-1})$ is estimated by $f_i(\boldsymbol{x}, y_1, \ldots, y_{i-1})$.

In the end, the algorithm outputs $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_m)$ with the highest entire joint conditional probability $\mathbf{P}(y_1, \ldots, y_m \,|\, \boldsymbol{x})$.

BS differs from GS in that i) BS explores more than one combination whereas GS just explores one and also in ii) the probability taken for deciding a path to follow in the tree is different from one method to the other. Concerning ii), both take the marginal joint conditional probability $\mathbf{P}(y_1, \ldots, y_j|\boldsymbol{x})$, but in the case of GS, that is equivalent to taking the marginal conditional probability $\mathbf{P}(y_j, | \boldsymbol{x}, y_1, \ldots, y_{j-1})$ since the two nodes explored in each level of the tree have the same parent, as explained before. However, in the case of BS, the b nodes explored do not have to have the same parent, even in the case of $b = 2$. Of course, if $b > 2$, this is impossible to happen in a binary tree.

Let consider the case of $b = 1$. In this case, BS expands just one node in each level, the one with the highest marginal joint conditional probability that coincides with the highest marginal conditional probability, so BS when $b = 1$ follows just one path that coincides with the one followed by GS. Also, if $b = 2^m$, BS performs an ES. Hence, BS encapsulates both GS and ES respectively considering $b = 1$ and $b = 2^m$. This makes it possible to control the trade-off between computational cost and performance of the method by tuning b between 1 and 2^m.

As a 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. The authors of the method [8] and [9] do not include any theoretical analysis about the goodness of the estimation of the risk minimizer, but they empirically show that by taking certain values of b ($b < 15$), the risk minimizer provided by the method converges to the one obtained using ES.

Figure 4 shows an example of the paths explored by the BS algorithm when $b = 2$.

4 A* Algorithm for Inference in PCC

The algorithm A* is the most widely-known form of best-first search [16], in which the *best* node at each iteration, according to an evaluation function e, is expanded. The particularity of the A* algorithm is that e can be seen as a function E of the 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 obtain an optimal solution, h must not overestimate the actual cost of reaching a solution, that is, it must be an admissible heuristic. These kinds of heuristics are optimistic, because they estimate that the cost of obtaining a solution is less than it actually is. Also, the A* algorithm is optimally efficient for any heuristic, because no other optimal algorithm using the same heuristic guarantees to expand fewer nodes than A*.

4.1 Building an Admissible Heuristic

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, that is, h must be an admissible heuristic[2] 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 us say 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

$$\begin{aligned} \mathbf{P}(y_1, \ldots, y_m \,|\, \boldsymbol{x}) = \mathbf{P}(y_1, \ldots, y_j \,|\, \boldsymbol{x}) \times \\ \mathbf{P}(y_{j+1}, \ldots, y_m \,|\, \boldsymbol{x}, y_1, \ldots, y_j). \end{aligned} \tag{10}$$

Hence, for a node at level j, let us 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 us remember that the values of $y_1, \ldots, y_j$ are known at level j.

Before discussing the heuristic h proposed here, let us notice that g is the same marginal joint conditional probability that both the ϵ-A algorithm

[2] An heuristic h is admissible for our search problem if and only if it satisfies that $h^*(k) \leq h(k)$ for any node k, where $h^*(k)$ is the highest (and unknown) probability from the node k to a leaf.

and the BS calculate to select the nodes to be expanded. Even more, ϵ-A with $\epsilon = 0$ is not only equivalent to the UC search, but also to A* with the constant heuristic $h = 1$. This heuristic is admissible too, since no probability is greater than 1. But, on the other hand, it is also the worst admissible heuristic, since any other admissible heuristic will dominate it[3] and consequently the A* algorithm using such an heuristic will never expand more nodes than the A* algorithm using $h = 1$.

Let us go now to discuss the heuristic proposed in this paper. Since our heuristic must 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 admissible, it is quite straightforward to pick the maximum value of such probability for obtaining an optimal heuristic h^*:

$$\begin{aligned} h^* &= \max_{(y_{j+1},\ldots,y_m)\in\{0,1\}^{m-j}} \mathbf{P}(y_{j+1}, \ldots, y_m \,|\, \boldsymbol{x}, y_1, \ldots, y_j) \\ &= \max_{\substack{(y_{j+1},\ldots,y_m)\\ \in\{0,1\}^{m-j}}} \prod_{i=j+1}^{m} \mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1}). \end{aligned} \tag{11}$$

However, obtaining such a 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 for renouncing such optimality. This leads to design an heuristic $\hat{h}$ also admissible, but less dominant than h^* ($h^* \prec \hat{h}$). For this purpose, let us consider $(\overline{y}_{j+1}, \ldots, \overline{y}_m)$ the values that define h^*, that is

$$h^* = \prod_{i=j+1}^{m} \mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, \overline{y}_{j+1}, \ldots, \overline{y}_{i-1}), \tag{12}$$

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

$$\begin{aligned} &\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, \overline{y}_{j+1}, \ldots, \overline{y}_{i-1}) \leq \\ &\quad \max_{\substack{(y_{j+1},\ldots,y_{i-1})\\ \in\{0,1\}^{i-1-j}}} \mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1}). \end{aligned} \tag{13}$$

Hence, by defining $\hat{h}$ as

$$\hat{h} = \prod_{i=j+1}^{m} \max_{\substack{(y_{j+1},\ldots,y_{i-1})\\ \in\{0,1\}^{i-1-j}}} \mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1}). \tag{14}$$

it is easy to deduce that $\hat{h}$ is admissible and less dominant than h^* ($h^* \prec \hat{h}$). But again, $\hat{h}$ is not computationally applicable in general. However, this is not the case if we restrict ourselves to the case of linear models, for instance using *logistic regression*. Remember that $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1})$ is estimated through

[3] In our case, an heuristic h_1 dominates another heuristic h_2 (denoted by $h_1 \prec h_2$) if and only if it satisfies that $h_1(k) \leq h_2(k)$ for any node k.

the model $f_i(\boldsymbol{x}, y_1, \ldots, y_{i-1})$ using a sigmoid function to transform the output of $f_i(\boldsymbol{x}, y_1, \ldots, y_{i-1})$ into a probability. Particularly,

$$\mathbf{P}(y_i = 1 \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1}) = \frac{1}{1 + \exp^{-f_i(\boldsymbol{x}, y_1, \ldots, y_{i-1})}} \tag{15}$$

$$\mathbf{P}(y_i = 0 \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1}) = 1 - \frac{1}{1 + \exp^{-f_i(\boldsymbol{x}, y_1, \ldots, y_{i-1})}}. \tag{16}$$

In order to obtain the maximum value between $\mathbf{P}(y_i = 1 \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1})$ and $\mathbf{P}(y_i = 0 \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1})$, we need first to obtain the maximum value of both terms on their own. In this direction and according to the above expressions of both probabilities, $\mathbf{P}(y_i = 1 \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1})$ will be maximum if $f_i(\boldsymbol{x}, y_1, \ldots, y_{i-1})$ is the maximum and analogously $\mathbf{P}(y_i = 0 \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1})$ will be maximum when $f_i(\boldsymbol{x}, y_1, \ldots, y_{i-1})$ is the minimum. Hence, let us now focus on obtaining the maximum and the minimum of $f_i(\boldsymbol{x}, y_1, \ldots, y_{i-1})$.

From now on, let us consider that f_i is a linear model, then f_i adopts the following form

$$f_i(\boldsymbol{x}, y_1, \ldots, y_{i-1}) = \langle \boldsymbol{w}_x^i, \boldsymbol{x} \rangle + \langle \boldsymbol{w}_y^i, (y_1, \ldots, y_{i-1}) \rangle + \beta^i, \tag{17}$$

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

$$f_i(\boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1}) = \langle \boldsymbol{w}_x^i, \boldsymbol{x} \rangle + \sum_{k=1}^{j} w_{y,k}^i y_k + \sum_{k=j+1}^{i-1} w_{y,k}^i y_k + \beta^i. \tag{18}$$

Since $\boldsymbol{x}$ is given and $y_1, \ldots, y_j$ are fixed, the second summation contains the variables for which the maximum and the minimum must be obtained. Let C_i be the constant part of f_i with regard obtaining the maximum and the minimum, that is,

$$C_i(\boldsymbol{x}, y_1, \ldots, y_j) = \langle \boldsymbol{w}_x^i, \boldsymbol{x} \rangle + \sum_{k=1}^{j} w_{y,k}^i y_k + \beta^i. \tag{19}$$

Then f_i can be rewritten as

$$f_i(\boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1}) = C_i(\boldsymbol{x}, y_1, \ldots, y_j) + \sum_{k=j+1}^{i-1} w_{y,k}^i y_k. \tag{20}$$

Consequently, the function whose maximum must be obtained is

$$\sum_{k=j+1}^{i-1} w_{y,k}^i y_k = w_{y,j+1}^i y_{j+1} + \ldots + w_{y,i-1}^i y_{i-1}. \tag{21}$$

Let us now denote by $K_{i,j}^+$ and $K_{i,j}^-$ the positive and negative indexes of the coefficients $w_{y,k}^i$ with $j+1 \leq k \leq i-1$, that is,

$$\begin{aligned} K_{i,j}^+ &= \{k \mid j+1 \leq k \leq i-1,\ w_{y,k}^i \geq 0\} \\ K_{i,j}^- &= \{k \mid j+1 \leq k \leq i-1,\ w_{y,k}^i < 0\}. \end{aligned} \tag{22}$$

Hence, (21) is maximum when y_k for $j+1 \leq k \leq i-1$ are

$$y_k = \begin{cases} 1 \text{ if } k \in K^+_{i,j} \\ 0 \text{ if } k \in K^-_{i,j} \end{cases} \tag{23}$$

and (21) is minimum when y_k for $j+1 \leq k \leq i-1$ are

$$y_k = \begin{cases} 1 \text{ if } k \in K^-_{i,j} \\ 0 \text{ if } k \in K^+_{i,j} \end{cases} \tag{24}$$

Hence,

i) Let $y^{i,1}_{j+1}, \ldots, y^{i,1}_{i-1} \in \{0,1\}$ be the values which maximize (21), that is, the values that maximize $f_i(\boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1})$ and hence, the values which makes $\mathbf{P}(y_i = 1 \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1})$ be maximum and,
ii) Let $y^{i,0}_{j+1}, \ldots, y^{i,0}_{i-1} \in \{0,1\}$ be the values that minimize (21), that is, the values that minimize $f_i(\boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1})$ and hence, the values which makes $\mathbf{P}(y_i = 0 \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1})$ be maximum.

Hence, $\max_{\substack{(y_{j+1},\ldots,y_{i-1}) \\ \in \{0,1\}^{i-1-j}}} \mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1})$ will be

$$\max_{v \in \{0,1\}} \{\mathbf{P}(y_i = v \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y^{i,v}_{j+1}, \ldots, y^{i,v}_{i-1})\}. \tag{25}$$

Notice that $y^{i,1}_k = 1 - y^{i,0}_k$ for $j+1 \leq k \leq i-1$ and that according to the above definition of the sigmoid function, if $f_i(\boldsymbol{x}, y_1, \ldots, y_j, y^{i,1}_{j+1}, \ldots, y^{i,1}_{i-1}) \geq -f_i(\boldsymbol{x}, y_1, \ldots, y_j, y^{i,0}_{j+1}, \ldots, y^{i,0}_{i-1})$, then the maximum of $\mathbf{P}(y_i = v \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y^{i,v}_{j+1}, \ldots, y^{i,v}_{i-1})$ is reached when $v = 1$ and otherwise when $v = 0$.
Therefore, the final expression for the heuristic will be

$$\hat{h} = \prod_{i=j+1}^{m} \max_{v \in \{0,1\}} \{\mathbf{P}(y_i = v \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y^{i,v}_{j+1}, \ldots, y^{i,v}_{i-1})\}. \tag{26}$$

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, which 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 5 exemplifies $\hat{h}$. Let us focus on the root node of the subtree in Figure 5. 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 $f_2(\boldsymbol{x}, y_1)$ and $P(y_3 \,|\, \boldsymbol{x}, y_1 = 1, y_2)$ provided by $f_3(\boldsymbol{x}, y_1, y_2)$ which respectively are 0.6 and 1.0. Then we carry out their product by 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

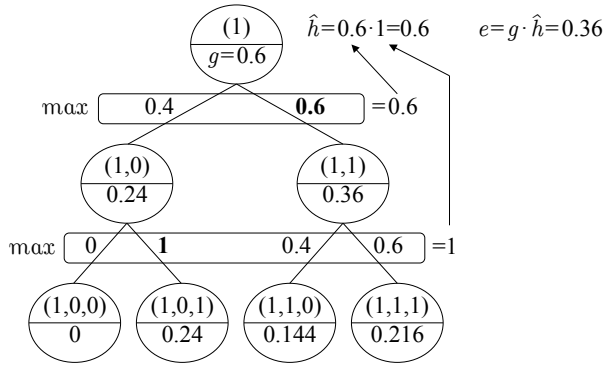


Fig. 5 An example of the computation of heuristic $\hat{h}$

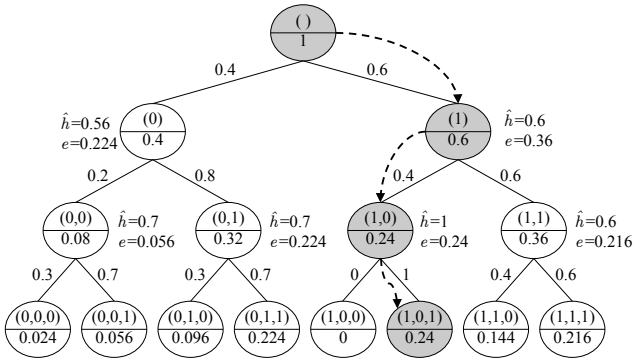


Fig. 6 An example of A* using the heuristic $\hat{h}$. The dotted arrows show the path followed by the algorithm. The values of g are provided inside each node

corresponds 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 h^*.

Figure 6 shows the path followed by A* using $\hat{h}$. It reaches the optimal leaf as is theoretically expected. Comparing this graph with the one in Figure 3(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 a 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 fewer nodes, since $\hat{h}$ is more dominant than the heuristic $h = 1$ ($\hat{h} \prec h = 1$). Obviously, computing $\hat{h}$ is more costly than computing $h = 1$ or applying just a UC search. The question is then if this

additional computing time compensates the theoretical guarantee it has of expanding fewer nodes.

4.2 A General Admissible Heuristic for Trading off the Number of Nodes Explored and its Computing Time

The previous section pointed out that both $\hat{h}$ and $h = 1$ are admissible heuristics. Besides, using the former theoretically guarantees that the A* algorithm explores fewer nodes than using the latter. But the cost of computing $\hat{h}$ could be high in comparison to just considering a constant heuristic $h = 1$. This trade-off sheds light on including a parameter d for limiting the depth of the heuristic. Hence, for a node of level j and a value of d with $0 \leq d \leq m - j$, only the terms $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}^i, \ldots, y_{i-1}^i)$ of $\hat{h}$ are evaluated using f_i for nodes from level $j+1$ to level $j+k$ whereas these terms are estimated by the constant 1 for nodes from level $j+k+1$ to level m, that is,

$$\hat{h}^d = \prod_{i=j+1}^{j+d} \mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}^i, \ldots, y_{i-1}^i) \cdot \prod_{i=j+d+1}^{m} 1, \tag{27}$$

or equivalently

$$\hat{h}^d = \prod_{i=j+1}^{j+d} \mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}^i, \ldots, y_{i-1}^i). \tag{28}$$

It is clear that $\hat{h}$ is more dominant than $\hat{h}^d$ ($\hat{h} \prec \hat{h}^d$) and it continues being admissible. In turn, $\hat{h}^d$ is more dominant than $h = 1$ ($\hat{h}^d \prec h = 1$). Hence, it is expected that using the heuristic $\hat{h}^d$ with $0 \leq d \leq m - j$ and $1 \leq j \leq m - 1$, the A* algorithm explores more number or equal number of nodes than $\hat{h}$, but less number or equal number of nodes than $h = 1$. In fact, $\hat{h}^d$ encapsulates both heuristics, since taking the extreme values of d leads to them. On the one hand, taking the maximum value of d leads to $\hat{h}^{m-j}$ with $1 \leq j \leq m - 1$ (we will denote this heuristic as $\hat{h}^\infty$), which is in fact the heuristic $\hat{h}$ detailed in Section 4.1. On the other hand, taking the minimum value of d leads to $\hat{h}^0$ which is indeed the heuristic $h = 1$. In general, $\hat{h}^{d_1}$ is more dominant than $\hat{h}^{d_2}$ if $d_1 > d_2$. However, the computational time of obtaining the values of $\hat{h}^d$ increases as d increases. Hence, tuning d adequately one can obtain a balance between the number of nodes explored and the computational time employed to evaluate the heuristic.

The case of $d = 1$ has especial interest, since the heuristic is also valid for non-linear models f_j. The form of the heuristic is

$$\hat{h}^1 = \prod_{i=j+1}^{j+1} \mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}^i, \ldots, y_{i-1}^i) \cdot \prod_{i=j+2}^{m} 1, \tag{29}$$

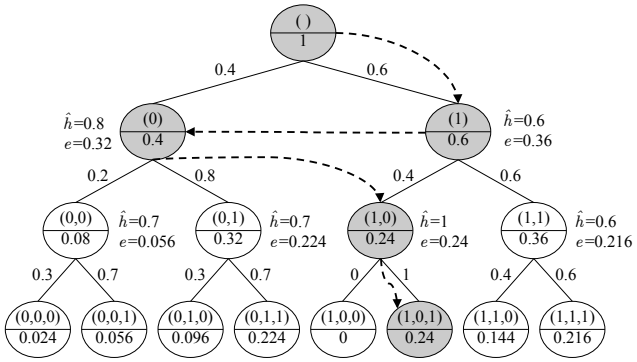


Fig. 7 An example of A* using the heuristic $\hat{h}^1$. The dotted arrows show the path followed by the algorithm. The values of g are provided inside each node

that simplifying leads to

$$\hat{h}^1 = \mathbf{P}(y_{j+1} \,|\, \boldsymbol{x}, y_1, \ldots, y_j). \tag{30}$$

As seen, this heuristic $\hat{h}^1$ means just evaluating the model of the node $f_j(\boldsymbol{x}, y_1, \ldots, y_j)$, hence without making any restriction on the linearity of f_j.

Figure 7 shows an example of the path followed by the A* algorithm when $\hat{h}^1$ is taken as an heuristic. As seen, the A* algorithm explores more nodes using this heuristic than using $\hat{h}$, but the computational time of evaluating the heuristic is considerably reduced, as we will show later on in the experiments.

4.3 Implementation Details

Here we significantly extend the results of A* using $\hat{h}$ presented in the preliminary work [13]. However, they present differences because we have carried out an improvement in the implementation of the algorithms in order to become more efficient. In fact, the time results reported there in comparison with the number of nodes explored was the key to make us reconsider the implementation of the algorithms. Algorithm 1 shows the pseudocode of A* using $\hat{h}^d$.

Particularly, the differences between this version and that of the preliminary work [13] are that now:

1. The models f_i (parameters $\boldsymbol{w}_x^i$, $\boldsymbol{w}_y^i$ and $\boldsymbol{\beta}^i$) are ordered according to the label order of the chain.
2. All repeated computations are calculated just once and stored:
   (a) The evaluation of $\langle \boldsymbol{w}_x^i, \boldsymbol{x} \rangle$ for each linear model f_i, which is common for all nodes corresponding to f_i, is computed once and saved in variable WX (line 2).

Algorithm 1 Pseudocode of the implementation of A* algorithm using $\hat{h}^d$

```
1: function A*
   Input: x, [W, β] a CC Linear Model, m, d, BlockSize
   Output: Label combination with highest probability for x
2:    WX ← W_x * x    // Computes ⟨w_x, x⟩ for all labels
3:    K+ ← AllocMemory(m)
4:    K− ← AllocMemory(m)
5:    for label = [2 : m] do    // Starts at 2nd level, no label attributes in the 1st model
6:       K+[label] ← [[W_y[label] >= 0]]
7:       K−[label] ← [[W_y[label] < 0]]
8:    end for
9:    Q ← AllocMemory(BlockSize)    // tuples {Labels, Level, e, g}
10:   Q[1] ← {[ ], 0, 1, 1}    // root node, empty label set, level 0, e and g = 1
11:   last ← 1    // last element used in Q
12:   while true do
13:      [Best, Position] ← Max(Q, last)
14:      if Best.Level = m then
15:         return Best.Labels    // Leaf node
16:      end if
17:      level ← Best.Level + 1
18:      P ← 1/(1 + exp(−(WX[level] + β[level] + ⟨w_y[level], Best.Labels⟩)))
19:         // Left child
20:      ĥ^d ← Heuristic(d, level, [Best.Labels 0], K+, K−, WX, W_y, x)    // Eq(28)
21:      Q[Position] ← {[Best.Labels 0], level, Best.g * (1 − P) * ĥ^d, Best.g * (1 − P)}
22:         // Right child
23:      last ← last + 1
24:      if last > Q.size then
25:         Q ← resize(Q, BlockSize)
26:      end if
27:      ĥ^d ← Heuristic(d, level, [Best.Labels 1], K+, K−, WX, W_y, x)    // Eq(28)
28:      Q[last] ← {[Best.Labels 1], level, Best.g * P * ĥ^d, Best.g * P}
29:   end while
30: end function
```

(b) Sets K^+ and K^-, that is the best values of y_k for the unknown part of the heuristic, are computed only once before starting the main loop of A* (lines 3-8).

3. Open set, Q, stores tuples of four elements: label combination, level, e and g. Q is stored in a resizable vector whose positions are reused. The left child is stored in the position of its parent (line 21). Notice that parent information is not required to obtain the final solution, since Q contains all the required information for each node.
4. The design of Q allows us to optimize the function Max, the operation to obtain the best node to be expanded. The function Max only needs to evaluate the first elements (1..*last*) of Q (line 13), because the rest of the vector is unused.
5. Algorithm 1 contains several auxiliary functions to make the pseudocode shorter and more easily understood. However, in the actual program all these functions were coded inline, making the code faster because all the calls to functions with large parameters, like the function *Heuristic*, are avoided.

4.4 Complexity analysis

Despite the theoretical optimality properties of the A* algorithm, it might not be useful in some problems because it may explore an exponential number of nodes with regard to the depth of the tree (see [17] and [21]). Only under certain conditions over the heuristic taken, is it possible to obtain an algorithm with a theoretical complexity less than exponential. This is the case if one is able to prove that the heuristic h satisfies the following condition for any level j with regard to the optimal heuristic h^* [21]

$$|h^*(j) - h(j)| \leq \mathcal{O}(\log h^*(j)). \tag{31}$$

In general, it occurs that that error is at least linear with regard to the path cost, hence leading to exponential growth. However, by taking care of designing good heuristics, one can obtain huge profit in relation to not providing any kind of heuristic information. That is the case with our heuristic $\hat{h}^d$ with regard to the heuristic $h = 1$, which does not provide any kind of heuristic information, in spite of being an improvement of the ES.

On what follows, let us focus on the heuristic $\hat{h}^\infty$, since it is the most dominant heuristic among the heuristics proposed. According to the results where the noise is increasing (see Figures 9 and 10 of Section 5.3), it is not clear that $\hat{h}^\infty$ is exponential at all as the rest of the algorithms clearly show, at least for the datasets taken. This behavior of $\hat{h}^\infty$ sheds light on performing a deeper analysis over other situations. In this sense, let us consider a theoretical case when the difference between $\hat{h}^\infty$ and h^* is maximum. For instance, this situation occurs (see Figure 8) when the first level has probability 0 in the left branch and 1 in the right branch, all the probabilities of the left subtree are 0 (for the left branches) and 1 (for the right branches) and all the probabilities of the right subtree are 1/2 (for all the branches). In this case, the error can be bounded as follows for any level j

$$|h^*(j) - \hat{h}^\infty(j)| \leq |1/2^{j-1} - 1|. \tag{32}$$

This bound tends to 1 as the level j grows, hence, in this case, it is not possible to guarantee that the theoretical complexity is less than exponential. However, these extreme cases are hardly likely to occur. Let us remember that the probabilities are evaluated from the models, where the description $\boldsymbol{x}$ of the examples plays an important role. This description is equal for obtaining the probabilities of all the nodes of the same level, hence, such probabilities will probably not differ so much among them as in the extreme case considered.

5 Experiments

This section deals with the experiments carried out with all the approaches. Before discussing the experimental results in Sections 5.2 and 5.3, let us describe in Section 5.1 the common settings of all experiments: datasets, learning

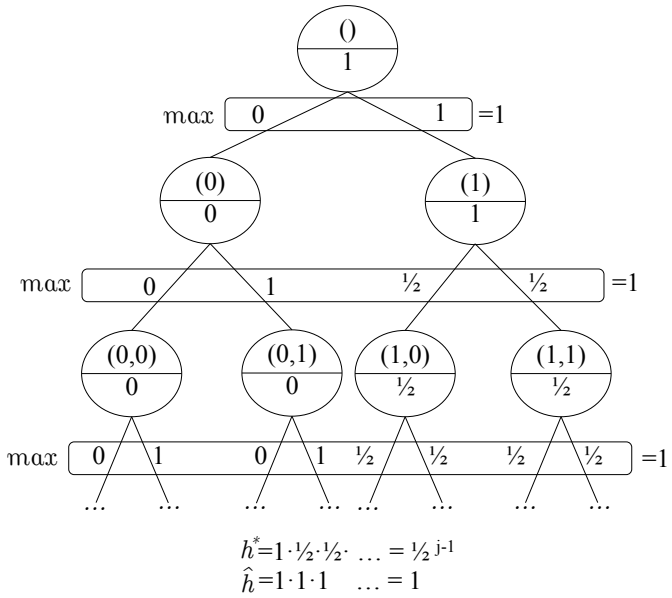


Fig. 8 An example of worst cases for heuristic $\hat{h}$

algorithms and parameter selection procedures. Finally, Section 5.3 reports the computational results of the methods for more complex problems, such as those that include noise.

5.1 Settings

The experiments were performed over several benchmark multi-label datasets whose main properties are shown in Table 1. As can be seen, there are significant differences in the number of attributes, instances and labels. The cardinality—number of labels per instance—varies from 1.07 to 4.27. Concerning the number of labels, there are some datasets with just 5, 6 or 7 labels, whereas others have more than 100, one of them even has almost 400 labels.

The approaches for inference in PCC compared with our proposal were those discussed throughout the paper, except for the ES. No experiment was carried out with the ES method due to its computational cost. Hence, the methods compared with the A* algorithm with the heuristic $\hat{h}$ for different values of the parameter d $(1, 2, 3, \infty)$ were GS, $\epsilon-$A algorithm for different values of ϵ $(.0, .25, .5)$ and BS for different values of beam width b $(1, 2, 3, 10)$. Let us remember that the $\epsilon-$A algorithm with $\epsilon = 0.5$ is equivalent to GS and to BS with $b = 1$.

The results will be presented in terms of the example-based subset 0/1 loss estimated by means of a 10-fold cross-validation.

Table 1 Properties of the datasets

Datasets	Instances	Attributes	Labels	Cardinality
bibtex	7395	1836	159	2.40
corel5k	5000	499	374	3.52
emotions	593	72	6	1.87
enron	1702	1001	53	3.38
flags	194	19	7	3.39
image	2000	135	5	1.24
mediamill*	5000	120	101	4.27
medical	978	1449	45	1.25
reuters	7119	243	7	1.24
scene	2407	294	6	1.07
slashdot	3782	1079	22	1.18
yeast	2417	103	14	4.24

The base learner employed to obtain the binary classifiers that compose all these multi-label models was *logistic regression* [10] with probabilistic output. The regularization parameter C was established for each *individual* binary classifier performing a grid search over the values $C \in \{10^{-3}, 10^{-2}, \ldots, 10^{3}\}$ optimizing the brier loss estimated by means of a balanced 2-fold cross validation repeated 5 times. The brier loss [1] is a proper score that measures the accuracy of probabilistic predictions, as *logistic regression* does. The expression is as follows

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{p}_i - a_i)^2, \tag{33}$$

where for an instance i, p_i is the predicted probability of a certain label and a_i is the actual value of the label (0 or 1).

Table 2 Subset 0/1 loss for the different methods. Those scores that are equal to or better than optimal predictions reached by $\epsilon-$A and ES are shown in bold.

Datasets	$\hat{h}$ (*) $\epsilon-$A(.0)	$\epsilon-$A(.25)	GS/BS(1) $\epsilon-$A(.5)	BS (2)	BS (3)	BS (10)
bibtex	**81.92**	81.95	82.19	**81.88**	**81.92**	**81.92**
corel5k	**97.48**	98.62	98.90	98.30	98.04	**97.48**
emotions	**71.16**	71.82	72.83	72.16	71.32	**71.16**
enron	**83.14**	84.26	85.43	83.43	83.37	**83.14**
flags	**87.13**	87.16	**86.13**	88.21	**87.13**	**87.13**
image	**68.35**	**68.35**	69.75	**68.35**	**68.35**	**68.35**
mediamill*	**83.86**	84.58	85.80	84.10	**83.86**	**83.86**
medical	**30.37**	**30.37**	30.67	**30.37**	**30.37**	**30.37**
reuters	**22.73**	**22.70**	23.60	**22.69**	**22.73**	**22.73**
scene	**31.86**	**31.86**	33.28	31.90	**31.86**	**31.86**
slashdot	**51.80**	52.22	54.49	**51.77**	**51.80**	**51.80**
yeast	**76.95**	77.62	79.77	**76.83**	77.08	**76.95**

Table 3 Number of explored nodes for the different methods.

Datasets	$\hat{h}^1$	$\hat{h}^2$	$\hat{h}^3$	$\hat{h}^\infty$	$\epsilon-$A(.0)	$\epsilon-$A(.25)	GS/BS(1) $\epsilon-$A(.5)	BS(2)	BS(3)	BS(10)
bibtex	283.31	277.97	273.23	215.91	289.27	184.00	160.00	319.00	477.00	1575.00
corel5k	1456.92	1443.08	1431.57	1338.62	1474.17	517.11	375.00	749.00	1122.00	3725.00
emotions	9.12	8.28	7.80	7.68	10.67	10.78	7.00	13.00	18.00	45.00
enron	110.32	106.50	103.28	75.51	114.81	77.29	54.00	107.00	159.00	515.00
flags	16.45	12.69	10.43	8.79	22.56	16.33	8.00	15.00	21.00	55.00
image	6.64	6.28	6.14	6.11	7.33	7.66	6.00	11.00	15.00	35.00
mediamill*	188.18	185.64	183.68	178.89	191.76	142.37	102.00	203.00	303.00	995.00
medical	46.61	46.58	46.55	46.40	46.64	46.65	46.00	91.00	135.00	435.00
reuters	8.15	8.14	8.13	8.13	8.24	8.26	8.00	15.00	21.00	55.00
scene	7.16	7.15	7.15	7.15	7.25	7.25	7.00	13.00	18.00	45.00
slashdot	24.98	24.88	24.86	24.84	25.29	24.89	23.00	45.00	66.00	205.00
yeast	23.60	22.80	22.38	21.01	26.09	26.02	15.00	29.00	42.00	125.00

Table 4 Average prediction time (in seconds) per example for all methods

Datasets	$\hat{h}^1$	$\hat{h}^2$	$\hat{h}^3$	$\hat{h}^\infty$	$\epsilon-$A(.0)	$\epsilon-$A(.25)	GS/BS(1) $\epsilon-$A(.5)	BS(2)	BS(3)	BS(10)
bibtex	0.0195	0.0314	0.0411	0.8407	0.0162	0.0099	0.0079	0.0110	0.0156	0.0507
corel5k	0.1684	0.2578	0.3333	23.8344	0.1360	0.0161	0.0110	0.0278	0.0404	0.1361
emotions	0.0006	0.0008	0.0009	0.0011	0.0006	0.0006	0.0004	0.0005	0.0006	0.0013
enron	0.0085	0.0141	0.0185	0.1084	0.0072	0.0030	0.0019	0.0039	0.0055	0.0172
flags	0.0011	0.0014	0.0015	0.0015	0.0012	0.0007	0.0004	0.0005	0.0007	0.0016
image	0.0005	0.0006	0.0007	0.0007	0.0005	0.0005	0.0004	0.0004	0.0005	0.0009
mediamill*	0.0144	0.0242	0.0326	0.4847	0.0115	0.0055	0.0037	0.0072	0.0105	0.0333
medical	0.0033	0.0057	0.0077	0.0495	0.0027	0.0027	0.0027	0.0033	0.0046	0.0144
reuters	0.0006	0.0008	0.0010	0.0013	0.0005	0.0005	0.0005	0.0005	0.0007	0.0016
scene	0.0005	0.0007	0.0009	0.0010	0.0005	0.0005	0.0004	0.0005	0.0006	0.0013
slashdot	0.0018	0.0029	0.0039	0.0124	0.0015	0.0014	0.0012	0.0016	0.0022	0.0063
yeast	0.0016	0.0026	0.0034	0.0075	0.0015	0.0010	0.0006	0.0010	0.0014	0.0040

5.2 Results over Benchmark Datasets

Tables 2, 3 and 4 respectively show the subset 0/1 loss, the number of nodes explored and the computational time (in seconds) averaged per test instances for the different methods compared.

Before discussing the results of the tables, let us remember that only the A* algorithm using $\hat{h}^d$ (and consequently also the $\epsilon-$A with $\epsilon = 0$, since it is the A* algorithm with $h = 1$ or $\hat{h}^0$) provides a Bayes optimal inference, as the ES does. This means that they always predict the label combination with the highest joint conditional probability. Despite the fact that other methods may predict other label combinations with lower joint conditional probability for some examples, in a few cases they obtain better subset 0/1 scores. This fact is due to several reasons, mainly a) the relatively small size of testing sets, and b) that the models f_j obtained to estimate the joint conditional probability $P(\boldsymbol{y} \,|\, \boldsymbol{x})$ do not usually return true estimations. Theoretically, under perfect conditions (large test sets and perfect models), the A* algorithm using $\hat{h}^d$ would obtain the best scores. In general, the performance of the $\epsilon-$A algorithm decreases as the value of ϵ increases and the performance of the BS method increases as b increases. The BS method reaches stability for low values of the beam b, it even converges to the performance of the A* algorithm but at the cost of exploring many more nodes.

With regard to the number of nodes explored (see Table 3), GS (equivalent to $\epsilon-$A algorithm with $\epsilon = 0.5$ and to BS(1)) is the method which explores the least number of nodes, since it only goes over one path in the tree. In fact, such a number corresponds to the number of labels (plus one if the root is considered as an explored node). It follows the A* algorithm using our heuristic $\hat{h}^d$ for any value of d, although it is exceeded by the $\epsilon-$A algorithm with $\epsilon > 0$ and by BS with $b > 1$ in some cases. However, let us remember that neither GS nor the $\epsilon-$A algorithm with $\epsilon > 0$ nor BS with any value of the beam b guarantee reaching an optimal solution as the A* algorithm using heuristic $\hat{h}^d$ does. So, it is not surprising that they explore fewer nodes. What it is actually surprising is that for most of the cases they explore more nodes even without reaching an optimal solution.

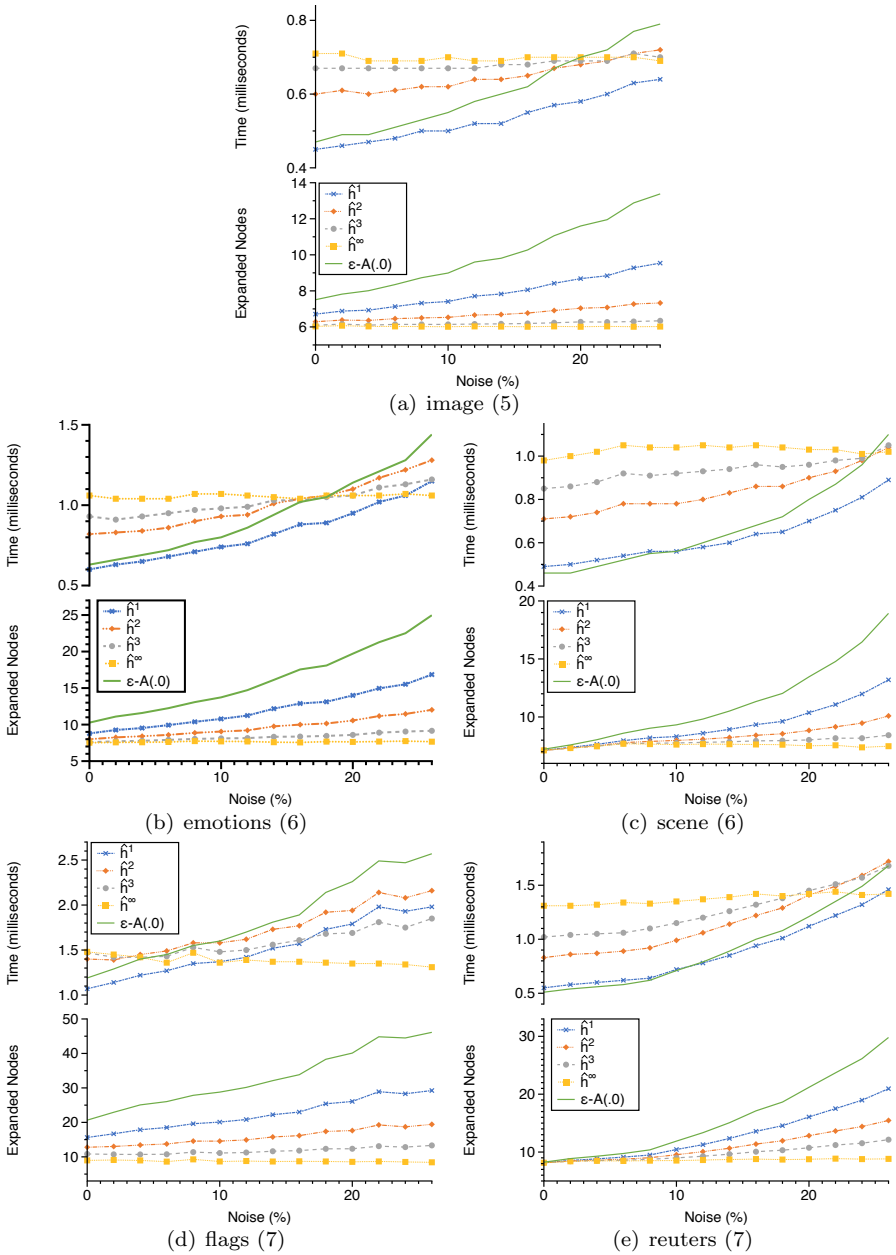
Let us now focus on the methods that theoretically reach the optimum (the A* algorithm with the heuristic $\hat{h}^d$ or the $\epsilon-$A algorithm with $\epsilon=0$). As it has theoretically been shown, the A* algorithm with the heuristic $\hat{h}^\infty$ explores the least number of nodes, followed by the same algorithm but with the heuristic $\hat{h}^3$, then with the heuristic $\hat{h}^2$, after that with the heuristic $\hat{h}^1$ and finally the $\epsilon-$A algorithm with $\epsilon=0$.

The computational time is expected to be higher as more nodes are explored, but looking at Table 4 one can see that this does not happen at all. This is true for GS, the $\epsilon-$A algorithm and the BS technique, but the A* algorithm obtains higher computational time in spite of exploring considerably fewer nodes. The reason for that is the time spent in computing the heuristic. In this respect, the A* algorithm is faster as the parameter d diminishes, although this implies exploring more nodes. Hence, considering the methods that reach the optimum, the $\epsilon-$A approximation algorithm with $\epsilon=0$ is the fastest method, followed by A* using the heuristic $\hat{h}^1$, then $\hat{h}^2$ and so on until ending with the heuristic $\hat{h}^\infty$.

As a conclusion, the A* algorithm with the heuristic h^1 can be considered a good alternative for guaranteeing optimal performance in terms of subset 0/1 and balancing the trade-off between the number of nodes explored and the computational time. Besides, it is also applicable for non-linear models.

5.3 Results with Noise

We have also performed experiments including noise in the datasets in order to analyze more in depth the power of the A* algorithm for inference in PCCs. This study arises from the fact that the number of nodes explored by A* is quite low in comparison with the number of labels (depth of the tree). This is so even using the heuristic $h = 1$ ($\epsilon-$A with $\epsilon = 0$) which does not provide information as other heuristics do, for instance, $\hat{h}$. This means that the algorithm A* performs few backtracking in the tree, hence, the probabilities provided by the models may be not so close from 0.5 or, even more, they may be close to 0 or 1. In this way, by adding noise we expect the probabilities to



(a) image (5)

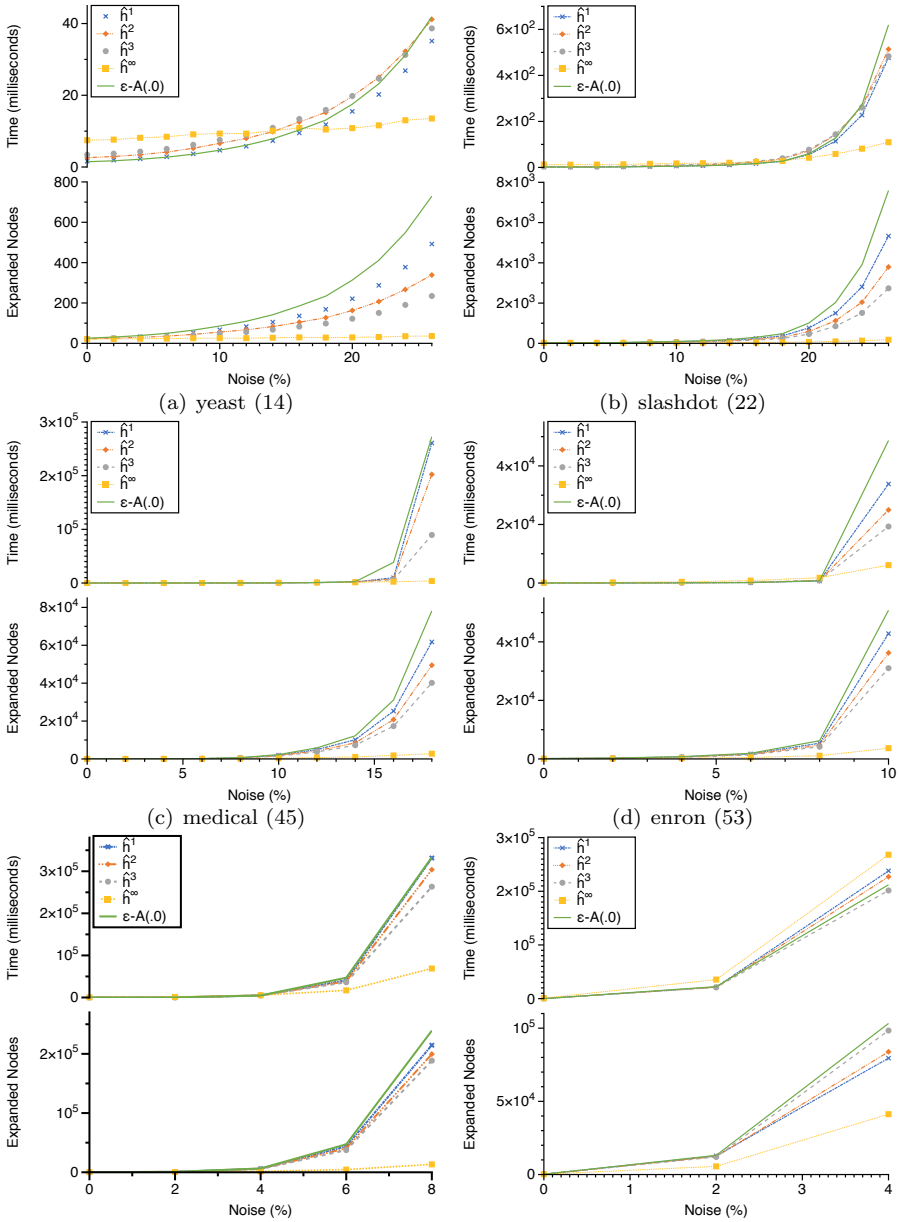
(b) emotions (6)

(c) scene (6)

(d) flags (7)

(e) reuters (7)

Fig. 9 Number of nodes expanded and computational time employed (ms) for the algorithm A* with the heuristic $\hat{h}$ for different values of the parameter d $(1, 2, 3, \infty)$ and for the $\epsilon-$A with $\epsilon=0$, for different percentages of noise and for datasets with few labels (from 5 to 7).



(a) yeast (14) (b) slashdot (22)

(c) medical (45) (d) enron (53)

(e) mediamill (101) (f) bibtex (159)

Fig. 10 Number of nodes expanded and computational time employed (ms) for the algorithm A* with the heuristic $\hat{h}$ for different values of the parameter d $(1, 2, 3, \infty)$ and for the $\epsilon-$A with $\epsilon = 0$, for different percentages of noise and for datasets with more labels (from 14 to 159)

be closer to 0.5, making the problem more complex where more informative heuristics can show their strength.

Certain grades of noise in terms of percentage were added to the datasets. No kinds of noise were included in the description of the examples, that is, in the $\boldsymbol{x}$ part, otherwise, the noise was only introduced in the labels of the examples, that is, in the $\boldsymbol{y}$ part. In this sense, a percentage of the values of the labels was swapped from being relevant to become irrelevant and vice versa for the whole dataset. This means that for a given example the relevance of either all their labels or only some of them or even none of them was changed. The percentage of noise included ranges from 0% to 25% for datasets with fewer than 22 labels. However, the experiments did not finish in a prudential time by using all the percentages of this range for datasets with more than 45 labels. In particular, the maximum percentage considered in this respect for medical (45 labels), enron (53 labels), mediamill (101 labels) and bibtex (159 labels) respectively was 18%, 10%, 8% and 4%.

Figures 9 and 10, respectively show for datasets with few labels (from 5 to 7) and for datasets with more labels (from 14 to 159), the number of nodes expanded and the computational time employed by the A* algorithm with the heuristic $\hat{h}$ for different values of the parameter d $(1, 2, 3, \infty)$ and by the $\epsilon-$A with $\epsilon=0$, all of them for different percentages of noise.

Obviously, both the number of nodes explored and the computational time increases as the percentage of noise increases. Regarding the number of nodes explored, it decreases as d increases as theoretically expected. In this respect, the number of nodes expanded using the heuristic $\hat{h}^{\infty}$ keeps quite steady as the noise increases in comparison to using the rest of the heuristics. In fact, by using these other heuristics, the number of nodes rockets from a certain percentage of noise. This is so in general, but it especially happens for datasets with more than 14 labels.

Concerning the computational time and for datasets with few labels, A* using $\hat{h}^{d}$ from $d > 1$ is steadier than using $\hat{h}^{1}$ or $\epsilon-$A with $\epsilon=0$ as the noise increases. In fact, these two last approaches are the best options, $\hat{h}^{1}$ being clearly better than $\epsilon-$A with $\epsilon = 0$. However, for high percentages of noise the other options begin to show their strength. Only flags among the datasets with few labels presents a different behavior. In this case, the behavior of the computational time is similar to that of the datasets with more labels, which coincides with the behavior of the number of nodes. So, the computational time increases as d decreases and $\hat{h}^{\infty}$ becomes clearly the best option. In any case, we are measuring the computational time in milliseconds, so the worth of the heuristics becomes crucial for datasets with more than 22 labels.

6 Conclusions

This paper proposes a family of admissible heuristics for the A* algorithm for performing inference in PCCs. In this way, providing admissible heuristics leads obtaining optimal predictions in terms of subset 0/1 loss and exploring

fewer nodes than previous approaches that also produce optimal predictions. The family of heuristics ($\hat{h}^d$) is defined by a parameter d which determines the depth in the tree for evaluating the heuristic and, hence, controls the trade-off between the number of nodes explored and the computational time of the algorithm due to the evaluation of the heuristic. In this manner, the algorithm A*($\hat{h}^d$) explores fewer nodes but it expends more time in reaching a solution as d increases. So far, only uniform-cost search (or ϵ-A($\epsilon = .0$), which in turn is the particular case of the A*($\hat{h}^0$)) has been shown to provide optimal subset 0/1 loss, unlike other approaches such as GS, ϵ-A($\epsilon > 0$) or BS that only estimate it. The algorithm A*($\hat{h}^d$) with $d > 1$ 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 for 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. Only the particular case of $d = 1$ is also suitable for non-linear models, since it just involves evaluating the models from all the known values of the labels.

In the experiments performed over the benchmark datasets, the algorithm A*($\hat{h}^1$) or uniform-cost search (or ϵ-A($\epsilon = .0$) or A*($\hat{h}^0$)) are better choices than A*($\hat{h}^d$) for $d > 1$, since the computational cost of evaluating the heuristic does not seem to compensate for the reduction in nodes explored. In this respect, adding noise to the benchmark datasets, and, hence, obtaining more complex problems, allows us to show the strength of A*($\hat{h}^d$) for $d > 1$. In this sense, the behavior of A*($\hat{h}^\infty$) is quite steady as the percentage of noise increases, unlike A*($\hat{h}^d$) for the rest of the values of d, including $d = 0$ (ϵ-A($\epsilon = .0$)). Furthermore, only for small datasets in a number of labels, $\hat{h}^1$ becomes the best alternative.

In our opinion, the heuristic search is a promising line of research for inference in PCCs. As future work, new admissible 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.

Acknowledgements This research has been supported by the Spanish Ministerio de Economía y Competitividad (grants TIN2011-23558, TIN2015-65069).

References

1. Brier, G.W.: Verification of forecasts expressed in terms of probability. Mon. Wea. Rev. **78**(1), 1–3 (1950)
2. Cheng, W., Hüllermeier, E.: Combining instance-based learning and logistic regression for multilabel classification. Machine Learning **76**(2-3), 211–225 (2009)
3. Dembczyński, K., Cheng, W., Hüllermeier, E.: Bayes Optimal Multilabel Classification via Probabilistic Classifier Chains. In: ICML, 2010, pp. 279–286 (2010)
4. Dembczyński, K., Waegeman, W., Cheng, W., Hüllermeier, E.: On label dependence and loss minimization in multi-label classification. Machine Learning **88**(1-2), 5–45 (2012)
5. Dembczynski, K., Waegeman, W., Hllermeier, E.: An analysis of chaining in multi-label classification. In: L.D. Raedt, C. Bessire, D. Dubois, P. Doherty, P. Frasconi, F. Heintz, P.J.F. Lucas (eds.) ECAI, *Frontiers in Artificial Intelligence and Applications*, vol. 242, pp. 294–299. IOS Press (2012)

6. Elisseeff, A., Weston, J.: A Kernel Method for Multi-Labelled Classification. In: ACM Conf. on Research and Develop. in Information Retrieval (2005), pp. 274–281 (2005)
7. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics **SSC-4(2)**, 100–107 (1968)
8. Kumar, A., Vembu, S., Menon, A.K., Elkan, C.: Learning and inference in probabilistic classifier chains with beam search. In: ECML/PKDD 2012, pp. 665–680 (2012)
9. Kumar, A., Vembu, S., Menon, A.K., Elkan, C.: Beam search algorithms for multilabel learning. Mach. Learn. **92**(1), 65–89 (2013)
10. Lin, C.J., Weng, R.C., Keerthi, S.S.: Trust region Newton method for logistic regression. Journal of Machine Learning Research **9**(Apr), 627–650 (2008)
11. McCallum, A.K.: Multi-label text classification with a mixture model trained by em. In: AAAI 99 Workshop on Text Learning (1999)
12. Mena, D., Montañés, E., Quevedo, J.R., del Coz, J.J.: An overview of inference methods in probabilistic classifier chains for multi-label classification. Tech. rep., Artificial Intelligence Center, University of Oviedo, Spain, Campus de Viesques, Gijón (2015)
13. Mena, D., Montañés, E., Quevedo, J.R., del Coz, J.J.: Using a* for inference in probabilistic classifier chains. In: IJCAI 2015, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, July 25-31, pp. 3707–3713 (2015)
14. Montañés, E., Quevedo, J.R., del Coz, J.J.: Aggregating independent and dependent models to learn multi-label classifiers. In: ECML/PKDD'11 - Volume Part II, pp. 484–500. Springer-Verlag (2011)
15. Montañés, E., Senge, R., Barranquero, J., Quevedo, J.R., del Coz, J.J., Hüllermeier, E.: Dependent binary relevance models for multi-label classification. Pattern Recognition **47**(3), 1494 – 1508 (2014). Handwriting Recognition and other {PR} Applications
16. Pearl, J.: Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1984)
17. Pearl, J.: Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1984)
18. Read, J., Martino, L., Luengo, D.: Efficient monte carlo methods for multi-dimensional learning with classifier chains. Pattern Recognition **47**(3), 1535 – 1546 (2014)
19. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. In: ECML/PKDD'09, LNCS, pp. 254–269. Springer (2009)
20. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Machine Learning **85**(3), 333–359 (2011)
21. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 2 edn. Pearson Education (2003)
22. Senge, R., Barranquero, J., del Coz, J.J., Hüllermeier, E.: Rectifying classifier chains for multi-label classification. Tech. rep. (2012)
23. Senge, R., del Coz, J.J., Hüllermeier, E.: On the problem of error propagation in classifier chains for multi-label classification. In: Conference of the German Classification Society on Data Analysis, Machine Learning and Knowledge Discovery, 2012 (2012)
24. Tsoumakas, G., Vlahavas, I.: Random k-Labelsets: An Ensemble Method for Multilabel Classification. In: ECML/PKDD'07, LNCS, pp. 406–417. Springer (2007)
25. Zhang, M.L., Zhou, Z.H.: Ml-knn: A lazy learning approach to multi-label learning. Pattern Recognition **40**(7), 2038–2048 (2007)