Noname manuscript No.
(will be inserted by the editor)

Improving the ϵ-approximate Algorithm for Probabilistic Classifier Chains

Miriam Fdez-Díaz · Laura Fdez-Díaz · Deiner Mena · Elena Montañés · José Ramón Quevedo · Juan José del Coz

Received: date / Accepted: date

Abstract Probabilistic Classifier Chains are a multi-label classification method on which has gained the attention of researchers in recent years. This is because of their ability to optimally estimate the entire joint conditional probability of a label combination through the product rule of probability. Their main drawback is that they require performing an exhaustive search in order to obtain Bayes-optimal predictions. This means computing this probability for all possible label combinations before taking a label combination with the highest value of probability. This is the reason why several works have been published in recent years that avoid exploring all combinations, while

This research has been funded by MINECO (the Spanish Ministerio de Economía y Competitividad) and ERDF (European Regional Development Fund) under grant TIN2015-65069-C2-2-R (MINECO/FEDER).

Miriam Fdez-Díaz
Artificial Intelligence Center. University of Oviedo at Gijón, 33204 Asturias, Spain
E-mail: uo231492@uniovi.es

Laura Fdez-Díaz
Artificial Intelligence Center. University of Oviedo at Gijón, 33204 Asturias, Spain
E-mail: uo231493@uniovi.es

Deiner Mena
Dept. de Ingeniería en Telecomunicaciones e Informática, Universidad Tecnológica del Chocó, Quibdó - Chocó, Colombia
E-mail: deiner.mena@utch.edu.co

Elena Montañés
Artificial Intelligence Center. University of Oviedo at Gijón, 33204 Asturias, Spain
E-mail: montaneselena@uniovi.es

José Ramón Quevedo
Artificial Intelligence Center. University of Oviedo at Gijón, 33204 Asturias, Spain
E-mail: quevedo@uniovi.es

Juan José del Coz
Artificial Intelligence Center. University of Oviedo at Gijón, 33204 Asturias, Spain
E-mail: juanjo@uniovi.es

maintaining optimality. Approaches such as greedy search, beam search and Monte Carlo reduce the computational cost, but at the cost of not ensuring Bayes-optimal predictions (although, in general, they provide close to optimal solutions). Methods based on a heuristic search provide optimal predictions, but the computational time has not been as good as expected. In this respect, the ϵ-approximate algorithm has been found to be the best inference approach among those that provide Bayes-optimal predictions, not only for its optimality, but also for its computational time. However, this paper both theoretically and experimentally shows that it sometimes performs some backtracking during the search for optimal predictions which may prolong the prediction time. The aim of this paper is thus to improve this algorithm by achieving a more direct search. Specifically, it enhances the criterion under which the next node to be expanded is chosen by adding heuristic information, although it is only applicable for linear based models. The experiments carried out confirm that the improved ϵ-approximate algorithm explores fewer nodes and reduces the computational time of the original version.

Keywords Multi-label · Classifier Chains · Inference · ϵ-approximate algorithm

1 Introduction

In multi-label classification [11], more than one label from a finite subset of labels can be assigned to an instance, unlike single-label classification in which just one class is assigned to an instance. Problems of this kind commonly appear in daily life: for instance, in medical diagnosis, where a patient may perhaps suffer from more than one pathology; also, in weather forecasting, where one day can be labelled with several weather aspects (fog, rain, etc...).

An important issue within multi-label classification is that the algorithms to cope with problems of this kind must present good scalability, since the problem becomes complex as the number of labels increases. In addition to scalability, algorithms need to deal with the correlations that usually exist among labels. Unfortunately, in fact, it is well-known that a patient who suffers from a certain pathology may have more probability of developing others or that a cold and cloudy day may end up becoming a rainy day. Hence, these relationships are actually a promising source of information that algorithms must be able to exploit in order to improve the performance of the classification.

Different strategies have been followed by state-of-the-art algorithms. On the one hand, some algorithms have adapted and extended other algorithms originating from multi-class classification. This is the case, for instance, of the decision tree algorithm C4.5, which has been modified to determine several functions for gene description [3]. Likewise, an instance-based learning approach that uses statistical information gained from the label sets and the maximum a posteriori (MAP) principle has been proposed [37]. Furthermore, there exists a back propagation neural network that employs a novel error function capturing the characteristics of multi-label learning [36]. Another proposed approach is based on a large margin ranking system that shares a lot of common properties with support vector machines [7]. The Naive Bayes approach has also been adapted for multi-label classification [17], while conditional random fields have been employed to create models that directly parameterize label co-occurrences in a multi-label context [10].

At the same time, new approaches have been developed. Some methods of this kind make use of relations in sets of different sizes. The RAndom k-labELsets (RAkEL) algorithm [34] considers a small random subset of labels and learns a single-label classifier to predict of each element in the powerset of this subset. An extension of this method, called Progressive RAndom k-labELsets (PRAkEL) [35], inherits the efficiency of RakEL but can handle arbitrary example-based evaluation criteria by progressively transforming a cost-sensitive multilabel classification problem into a series of cost-sensitive single-label classification problems. A label ranking approach [8] introduces an artificial calibration label that, in each example, separates the relevant from the irrelevant labels. Another approach simultaneously unifies a binary classification for the labels and a correlation between them in a single step [25]. The Pruned Sets (PS) method treats a set of labels as single labels, like RAkEL, but after a pruning process, only the relevant labels are considered [29]. Also, an algorithm that includes a boosting strategy [32] is a new approach that takes into account relations among set of labels of different size. The Hierarchy Of Mul-tilabel classifiERs (HOMER) [33] method is an effective and computationally efficient multi-label classifier for domains with large label sets. It constructs a hierarchy of multi-label classifiers, each one dealing with a much smaller set of labels and with a more balanced example distribution. The use of fuzzy-rule-based classifiers as base learners when the multi-label problem is transformed into a set of single-label problemas has also been exploited [24]. The interest in building fuzzy rules lies in the interpretability they offer for the classification and the vagueness among the labels boundaries they are able to capture.

Other new approaches make use of label dependence [5]. What is known as unconditional dependence is considered in some methods. For instance, a method that merges instance-based learning and logistic regression (IBLR) [2] falls within this kind of methods. So-called conditional dependence has been exploited by many other methods. For instance, an approach that aggregates both independent and dependent (AID) classifiers has also been put forward [22]. Similarly, a stacking approach [12], which stacks independent and dependent classifiers, also exploits conditional dependence. Another approach of this kind is Dependence Binary Relevance (DBR) [23], which combines chaining and stacking strategies; in fact, it trains both independent and fully-dependent classifiers (the chain) and then stacks both. Also, Classifier Chains (CC) [30, 31] establish a chain of classifiers from an order of the labels, in such way that one label depends on the labels placed before it in the chain. Related to CC, the method Probabilistic Classifier Chains (PCC) [4] method trains a classifier chain and performs an exhaustive search to obtain Bayes-optimal predictions.

Similarly, Classifier Trellis (CT) [28] has arisen for scalable multi-label classification. This method uses a heuristic to build a structure of the label (trellis) in an order that maximizes label dependence between parents and children, subsequently capturing the main label dependencies and maintaining scalability. Moreover, Viterbi Classifier Chains (VCC) [26] have recently been proposed, which are similar to PCC. However, the chain structure differs, since, each label in VCC only depends on the previous label and not on all the labels placed before it in the chain. This simplification allows a polynomial order for the search for a solution, which means considerably reducing the computational time. In general, however, VCC is not able to estimate the highest entire joint conditional probability, as PCC does. In fact, this is the property of PCC that has led several researchers to focus their attention on PCC in recent years. The main drawback of the method is its computational cost, because it performs an exhaustive search over all possible combinations of labels.

Major efforts have been made to maintain the promising property of optimality of PCC, while trying to improve the computational cost of the inference process. In fact, methods based on greedy search [30,31], beam search [14,15] and Monte Carlo sampling [6,27] do not consume so much time when they perform inference, but they do not always provide optimal predictions. Heuristic-search-based methods [18,20,21] have been found to provide optimal solutions, but their computational cost is sometimes still high, just being competitive for complex problems, such as those that include noise. The ϵ-approximate algorithm [6] has proven to be the most promising method so far, as it ensures Bayes-optimal predictions in the fastest way [19,20,21]. However, in spite of these two promising properties, this algorithm may perform some backtracking during the search that can hinder obtaining an optimal solution in a more direct way. This paper proposes modifying the criterion of the ϵ-approximate algorithm that decides the node to be expanded each time. To this end, the criterion includes heuristic information in order to accelerate the process of reaching a Bayes-optimal solution, not merely in terms of computational time, but also in the number of nodes explored. The result is a new algorithm that, although it is limited to linear base models, maintains the property of ensuring optimal predictions. However, it also guarantees following a more direct path; hence, it explores fewer nodes. Furthermore, it reduces the computational time in spite of having to compute the heuristic.

The rest of the paper is organized as follows. Section 2 formally states the multi-label framework, the principles of PCC and the process of inference in PCC. Section 3 describes and discusses the properties and behavior of the ϵ-approximate algorithm for performing inference in PCC. Section 4 details the contribution of this paper that modifies the ϵ-approximate algorithm in order to shorten the search path to reach optimal predictions. Exhaustive experiments are presented and discussed in Section 5. Finally, Section 6 lays out some conclusions and includes new directions for future work.

2 Inference in Probabilistic Classifier Chains for multi-label classification

Let us first formally define the problem of multi-label classification. The starting point is a finite and non-empty set of m labels, $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_m\}$, and a training set, $S = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$ independently and randomly drawn according to an unknown probability distribution, $\mathbf{P}(\mathbf{X}, \mathbf{Y})$, on $\mathcal{X} \times \mathcal{Y}$. The input space, $\mathcal{X}$, is the space of the instance description whereas the output space, $\mathcal{Y}$, is the power set of $\mathcal{L}$, $\mathcal{P}(\mathcal{L})$, or equivalently $\mathcal{Y} = \{0,1\}^m$. This equivalence means that each $\boldsymbol{y}_i = (y_{i,1}, y_{i,2}, \ldots, y_{i,m})$ is the observation of a corresponding random vector $\mathbf{Y} = (\mathbf{Y_1}, \mathbf{Y_2}, \ldots, \mathbf{Y_m})$, where $y_{i,j} = 1$ indicates the presence (relevance) and $y_{i,j} = 0$ the absence (irrelevance) of ℓ_j in the labeling of $\boldsymbol{x}_i$.

From the aforementioned framework, the goal in multi-label classification can be stated as the induction from S of a hypothesis, $\boldsymbol{f} : \mathcal{X} \longrightarrow \mathcal{Y}$, that provides a combination of relevant labels $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$ for unlabelled instances, $\boldsymbol{x}$, 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, $\mathbf{P}(\mathbf{X}, \mathbf{Y})$, i.e.,

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

therefore, the so-called risk minimizer, $\boldsymbol{f}^*$, can be expressed by

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

where $\mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x})$ is the conditional distribution $\mathbf{Y} = \boldsymbol{y}$ given $\mathbf{X} = \boldsymbol{x}$.

With regard to the loss functions [11], this paper focuses only on the subset 0/1 loss, because it is the measure that PCC is able to optimize (see [6] for details and proof of this point). This measure oversees whether the predicted and relevant label subsets are equal or not, is defined by

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

in which the expression $[\![p]\!]$ evaluates to 1 when the predicate, p, is true and to 0 otherwise. This measure simplifies the risk minimizer, since it can be obtained by simply taking the mode of the entire joint conditional distribution, i.e.

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

2.1 Probabilistic Classifier Chains

PCC are based on the earlier CC method [30,31]. In fact, both share the training process, but differ in the way they carry out the testing stage.

During training, both methods define an order of the label set ($\ell_1 \prec \ell_2 \prec \ldots \prec \ell_{j-1} \prec \ell_j \prec \ell_{j+1} \prec \ldots \prec \ell_m$) and, following this order, they train a probabilistic binary classifier for each label ℓ_j to estimate $\mathbf{P}(y_j \,|\, \boldsymbol{x}, y_1, \ldots, y_{j-1})$.

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 each classifier, f_j, is the set $S_j = \{(\overline{\boldsymbol{x}}_1, y_{1,j}), \dots, (\overline{\boldsymbol{x}}_n, y_{n,j})\}$, where $\overline{\boldsymbol{x}}_i = (\boldsymbol{x}_i, y_{i,1}, \dots, y_{i,j-1})$; i.e., the features are the description of the instance $\boldsymbol{x}_i$ supplemented by the relevance of the labels $\ell_1, \dots, \ell_{j-1}$ preceding ℓ_j in the chain and the category is the relevance of the label ℓ_j.

In the testing stage, the goal is to perform inference for each instance, which consists in estimating the risk minimizer for a given loss function over the estimated entire joint conditional distribution [6]. As already shown, this means computing Equation (4) when the loss function is the subset 0/1, which is our case. PCC estimates the entire joint conditional probability, $\mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x})$, of Equation (4) using the product rule of probability; i.e., it estimates

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

Before analyzing this issue in the following section, notice that, according to Bayes, this expression theoretically holds for any order considered for the labels. Although these methods are label-order dependent in practice and have already been studied [13,15], the study of this issue falls beyond the scope of this paper. Hence, an order is set beforehand (the original one), given that the aim of the paper is to improve the original ϵ-approximate algorithm regardless of the label order considered.

2.2 Inference in Probabilistic Classifier Chains

Several approaches have been proposed for inference in PCC to carry out the testing stage. As already stated, this task entails obtaining a combination of labels, $\boldsymbol{y}$, for a given instance, $\boldsymbol{x}$, that maximizes the probability, $\mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x})$. This task may also be seen as the different ways of exploring a probability binary tree of depth m. The root of the tree is labelled by the empty set of labels with a joint probability of 1, whereas each leaf is labelled by one of the possible combinations of labels $(v_1, v_2, \dots, v_m)$ with $v_i \in \{0,1\}$ for $i = 1, \dots, m$, together with the entire joint conditional probability of this combination (notice that a binary tree of depth m has as many leaves as the number of possible combinations of labels). The intermediate nodes are labelled by a partial combination of labels and by the marginal joint conditional probability. That is, a node of level $j < m$ is labelled by $(v_1, v_2, \dots, v_j)$ with the probability $\mathbf{P}(y_1 = v_1, \dots, y_j = v_j \,|\, \boldsymbol{x})$. In turn, the children of this node are respectively labelled by $(v_1, v_2, \dots, v_j, 0)$ and $(v_1, v_2, \dots, v_j, 1)$ with a respective marginal joint conditional probability $\mathbf{P}(y_1 = v_1, \dots, y_j = v_j, y_{j+1} = 0 \,|\, \boldsymbol{x})$ and $\mathbf{P}(y_1 = v_1, \dots, y_j = v_j, y_{j+1} = 1 \,|\, \boldsymbol{x})$ (see Figure 1). These marginal joint conditional probabilities are estimated through the product rule of probability from the marginal joint conditional probability of their parent and

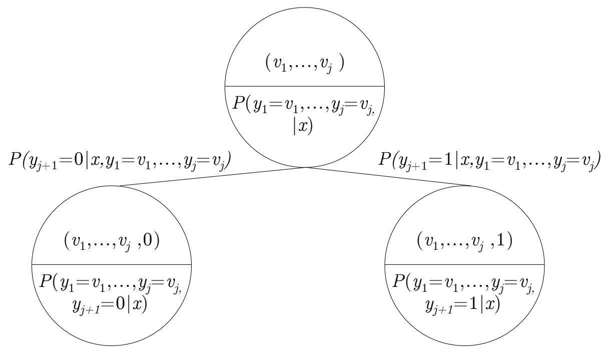
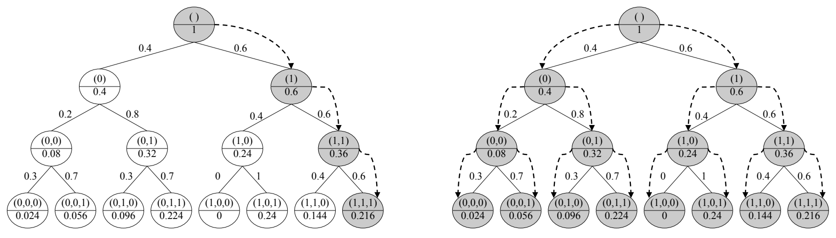

$(v_1,\ldots,v_j)$
$P(y_1{=}v_1,\ldots,y_j{=}v_j, |x)$
$P(y_{j+1}{=}0|x,y_1{=}v_1,\ldots,y_j{=}v_j)$
$P(y_{j+1}{=}1|x,y_1{=}v_1,\ldots,y_j{=}v_j)$
$(v_1,\ldots,v_j,0)$
$P(y_1{=}v_1,\ldots,y_j{=}v_j, y_{j+1}{=}0|x)$
$(v_1,\ldots,v_j,1)$
$P(y_1{=}v_1,\ldots,y_j{=}v_j, y_{j+1}{=}1|x)$

Fig. 1 A generic node and its children in 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 labelled with the conditional probability.



(a) Greedy search (b) Exhaustive search

Fig. 2 Example of the path followed by (a) the original CC (greedy search) and (b) the original PCC method (exhaustive search) for three labels ($m = 3$). The dashed arrows indicate the path followed by the algorithm.

from the conditional probabilities $\mathbf{P}(y_{j+1} = 0 \,|\, \boldsymbol{x}, y_1 = v_1, \ldots, y_j = v_j)$ and $\mathbf{P}(y_{j+1} = 1 \,|\, \boldsymbol{x}, y_1 = v_1, \ldots, y_j = v_j)$ of the edges between the parent and the children; i.e., $\mathbf{P}(y_1 = v_1, \ldots, y_j = v_j, y_{j+1} = v_{j+1} \,|\, \boldsymbol{x}) = \mathbf{P}(y_{j+1} = v_{j+1} \,|\, \boldsymbol{x}, y_1 = v_1, \ldots, y_j = v_j) \cdot \mathbf{P}(y_1 = v_1, \ldots, y_j = v_j \,|\, \boldsymbol{x})$. The conditional probabilities of the edges that joint a node of level j and its children are estimated using the trained classifiers, f_{j+1}.

Several methods are already available in the literature that address inference in PCC. The method first proposed was one based on a greedy search (the original CC method [30,31]), which simply follows one path of the tree, namely the path with the highest conditional probability. Its successor was the method based on an exhaustive search (the original PCC method [4]), which follows all possible paths and takes the one with the highest joint conditional probability. Figure 2 shows an example for three labels ($m = 3$) of the path followed by the original CC (Figure 2(a)), which provides the non-optimal combination of labels $(1, 1, 1)$ with a joint conditional probability equal to 0.216, and by the original PCC (Figure 2(b)), which provides the optimal combination of labels $(1, 0, 1)$ with a joint conditional probability equal to 0.24. The ϵ-approximate algorithm [6] subsequently arose as an alternative for the poor performance of CC and the high computational cost of PCC. It is a uniform-cost search algorithm that includes a pruning process which can output optimal predictions in terms of subset 0/1 loss, like the method based on exhaustive search (PCC), but significantly reducing its computational cost, like the greedy-search-based method (CC). A more recent approach based on a beam search [14,15] has been found to behave well both in terms of performance and computational cost. It explores only a certain predefined number of nodes in each level, and depending of this number, called beam width and denoted by b, it performs an exhaustive search until a certain level of the tree. In fact, the extreme value of $b = 2^m$ means that a complete exhaustive search is performed. In turn, the extreme value of $b = 1$ makes this method follow the same path as the original CC. Its main drawback is that it does not ensure optimal predictions for any value $1 < b < 2^m$. Monte Carlo sampling [6, 27] is an appealing and simpler alternative that randomly explores a certain number of paths (combinations of labels) drawn using a probability distribution induced by the trained models, which are finally aggregated to provide a definitive prediction. This strategy overcomes the high computational cost of the exhaustive-search-based method, but has the drawback of not ensuring an optimal combination of labels. Finally, some methods based on a heuristic search [18,20,21] have recently been proposed. These have theoretically been shown to provide Bayes-optimal solutions, as they employ admissible[1] heuristics for the A* algorithm. Their main drawback, however, is the computational time spent on computing the heuristic, which, in general, does not compensate the reduction achieved in the number of nodes explored. This paper focuses on the ϵ-approximate algorithm, as it is able to ensure Bayes-optimal predictions and it has been experimentally shown to obtain them in the fastest way [19,20, 21]. The next section provides details and in-depth analysis of this approach, since it will be the starting point for the contribution of this paper. The idea is to provide heuristic information to the ϵ-approximate algorithm in order to obtain an even faster algorithm, because the search will be more direct, as it will be subsequently shown.

3 The ϵ-approximate algorithm

The ϵ-approximate algorithm [6] is based on a uniform-cost search, which, applied to inference in PCC, involves expanding the node with the highest marginal joint probability. This marginal joint conditional probability for a

[1] A heuristic is admissible if it never underestimates the gain of reaching the goal, i.e. the gain it estimates to reach the goal is not lower than the highest possible gain. This property means that the A* algorithm ensures an optimal solution.

node in the level j for an unlabelled $\boldsymbol{x}$ is

$$\mathbf{P}(y_1, \ldots, y_j \,|\, \boldsymbol{x}) = \prod_{i=1}^{j} \mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1}), \tag{7}$$

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

Algorithm 1 Pseudocode of the ϵ-approximate algorithm

```
1: function ϵ-APPROXIMATE
   Input: $\boldsymbol{x}$, $[\mathbf{W}, \boldsymbol{\beta}]$ a CC Linear Model, $m$, and $\epsilon \geq 0$
   Output: Label combination with highest probability for $\boldsymbol{x}$
2:     $WX \leftarrow \mathbf{W}_x * \boldsymbol{x}$    // Computes $\langle \boldsymbol{w}_x, \boldsymbol{x} \rangle$ for all labels
3:     $Q \leftarrow \{[\,], 1, 1\}$    // $\{$root node, $level, g\}$
4:     $K \leftarrow \emptyset$    // list of non-survived parents
5:     repeat
6:         $Node \leftarrow Max(Q)$    // node in $Q$ with highest $g$ value
7:         if $Node.Level = m$ then    // Node is a leaf
8:             return $Node.Labels$
9:         $level \leftarrow Node.Level + 1$
10:        $P \leftarrow 1/(1 + exp(-(WX[level] + \beta[level] + \langle \boldsymbol{w}_y[level], Node.Labels \rangle)))$
11:        $e\_left \leftarrow Node.g * (1 - P)$
12:        if $e\_left \geq \epsilon$ then    // Left child
13:            $Insert(Q, \{[Node.Labels\ 0], level, e\_left\})$
14:        $e\_right \leftarrow Node.g * P$
15:        if $e\_right \geq \epsilon$ then    // Right child
16:            $Insert(Q, \{[Node.Labels\ 1], level, e\_right\})$
17:        if $e\_left < \epsilon$ and $e\_right < \epsilon$ then
18:            $Insert(K, Node)$
19:    until $Q = \emptyset$
20:    $\epsilon \leftarrow 0$
21:    repeat
22:        $Node' \leftarrow$ pop first element in $K$
23:        $LeafProb \leftarrow GreedySearch(Node')$    // apply GS to reach a leaf
24:        if $LeafProb \geq \epsilon$ then
25:            $Best \leftarrow Node'$
26:            $\epsilon \leftarrow LeafProb$
27:    until $K \neq \emptyset$
28:    return $Best.Labels$
```

For this purpose, the algorithm maintains a list of candidate nodes to be expanded, Q, that initially only contains the root node (empty combination of labels) with a probability equal to 1 (see line 3 of Algorithm 1). The candidate nodes are expanded in the order established by this probability (see line 6 of Algorithm 1). Once a node is chosen to be expanded, the marginal joint conditional probability for its children is obtained (see lines 10, 11 and 14 of Algorithm 1).

The algorithm includes a parameter, ϵ, which controls the candidate nodes to be expanded: only those nodes whose marginal joint conditional probability, $\mathbf{P}(y_1, \ldots, y_j \,|\, \boldsymbol{x})$, is greater than or equal to ϵ. As ϵ grows, fewer nodes are expanded, thereby limiting the computational cost of the algorithm. In this respect, if the marginal joint conditional probability of the children exceeds the value of ϵ, then these children are added to the list of nodes to be expanded. So, it could occur that just one of the children is added to the list, both of them are added to the list or none of them are added to the list (see lines 13 and 16 of Algorithm 1). If this last situation takes place, i.e., none of the children are added to the list, the node is included in the list of non-survived parents, K (see line 18). This way of taking a node to be expanded means the algorithm does not follow a strictly downwards path (it does not just visit nodes of single branch of the tree), otherwise it may change from one node of a certain branch to one node of another, different branch, depending on the marginal joint conditional probabilities. Figures 3 and 4 respectively show two examples of paths followed by the algorithm when $\epsilon = 0$ and $\epsilon = 0.25$ for three labels ($m = 3$). In both figures, each node is labelled with the combination of labels (at the top) and with the marginal joint conditional probability (at the bottom). Besides, the right arches are labelled with the conditional probability, $\mathbf{P}(y_i|\,\boldsymbol{x}, y_1, \ldots, y_{i-1})$, estimated from $f_i(\boldsymbol{x}, y_1, \ldots, y_{i-1})$ when $y_i = 1$. Analogously, the left arches are labelled with this conditional probability, but when $y_i = 0$. Notice that a value of $\epsilon = 0$ means that all nodes are candidates to be explored, as all nodes have a marginal joint conditional probability greater than $\epsilon = 0$. Nonetheless, this does not mean that all nodes are explored, given that each time the candidate to be explored is always the one with the highest marginal joint conditional probability. Moreover, only the nodes with a marginal joint conditional probability higher than or equal to an optimal leaf are explored.

Eventually, two situations can arise: i) the expanded node is a leaf, or ii) there are no more nodes that exceed the threshold, and hence there are no candidate nodes to be expanded. If the former situation occurs (see Figure 3), the prediction for the unlabelled instance, $\boldsymbol{x}$, will be $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_m)$, corresponding to the combination of the leaf reached (see line 8 of Algorithm 1). Conversely, if situation ii) arises (see Figure 4), then a greedy search strategy is applied to the non-survived nodes; i.e., to those nodes whose children do not exceed the threshold (see lines from 19 to 25 of Algorithm 1). In this case, the prediction $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_m)$ for the unlabelled instance, $\boldsymbol{x}$, will be the one with the highest entire joint conditional probability, $\mathbf{P}(y_1, \ldots, y_m \,|\, \boldsymbol{x})$ (see line 26 of Algorithm 1). Notice that if situation ii) arises, the algorithm does not necessarily expand all nodes, even in the case of $\epsilon = 0$.

The parameter ϵ plays an important role in the algorithm. The particular case of $\epsilon = 0$ (or any value in the interval $[0,\, 2^{-m}]$) is of special interest, as the algorithm always finds an optimal solution. The fact is that an optimal leaf has a joint conditional probability equal to or greater than 2^{-m}; so any value for ϵ below 2^{-m} does not filter an optimal leaf. The reason that an optimal leaf has a joint conditional probability equal to or greater than 2^{-m} is because the arches that joint a node and its two children have probabilities p and $1-p$ respectively. If one follows a path that always has the highest probability between p and $1-p$, then these highest probabilities are greater than or equal

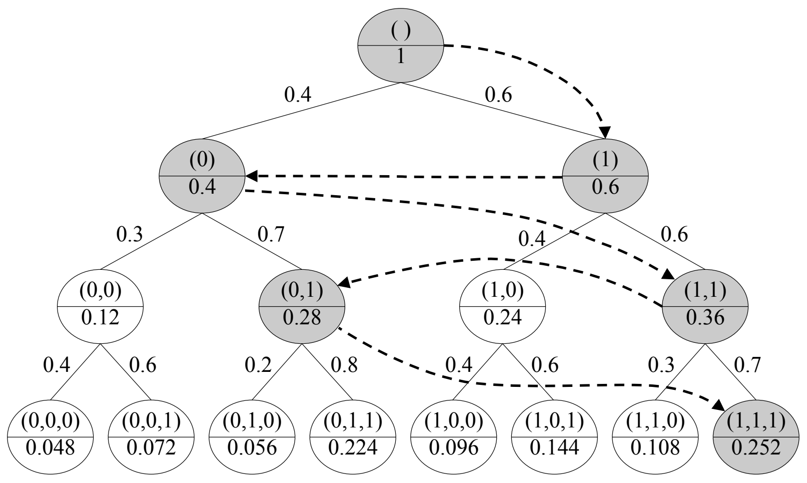


Fig. 3 An example of paths followed by ϵ-approximate algorithm when $\epsilon = 0$ for three labels ($m = 3$). The dotted arrows show the path followed by the algorithm.

to 1/2. Said path leads to a leaf with a joint conditional probability greater than or equal to $1/2^m$ or, equivalently, 2^{-m}. An optimal leaf may be a leaf with a probability of 2^{-m}, or at least this value. Figure 3 shows a situation of this kind, specifically, for the case of $\epsilon = 0$. In this case, all the nodes are candidates to be explored, since all of them have a marginal joint conditional probability greater than $\epsilon = 0$. First, the right node of the first level (the (1) node) has the highest probability (0.6); hence it is explored first. Its children (the (1,0) and (1,1) nodes) have a marginal joint conditional probability (respectively 0.24 and 0.36) lower than their uncle (the (0) node), which has a probability of 0.4. So their uncle is explored before them and then the children of that uncle are included as candidates to be explored. After that, the child more to the right in the second level (the (1,1) node) is explored next because it has the highest marginal joint conditional probability (0.36) among its brother (the (1,0) node, with a probability of 0.24) and cousins (the (0,0) and (0,1) nodes with respective probabilities 0.12 and 0.28). The following node to explore is the right cousin of the (1,1) node (the (0,1) node), with a probability of 0.28, which is higher than the probabilities of its children (the (1,1,0) and the (1,1,1) nodes with respective probabilities 0.108 and 0.252), the probability of its brother (the (1,0) node, with probability 0.24) and the probability of its left cousin (the (0,0) node, with probability 0.12). Finally, the right child of the right cousin of the (0,1) node (the (1,1,1) node, with a probability of 0.252) is expanded before the other nodes (the (0,0), (1,0), (0,1,0), (0,1,1) and (1,1,0) nodes with respective probabilities 0.12, 0.24, 0.056, 0.224 and 0.108) due to

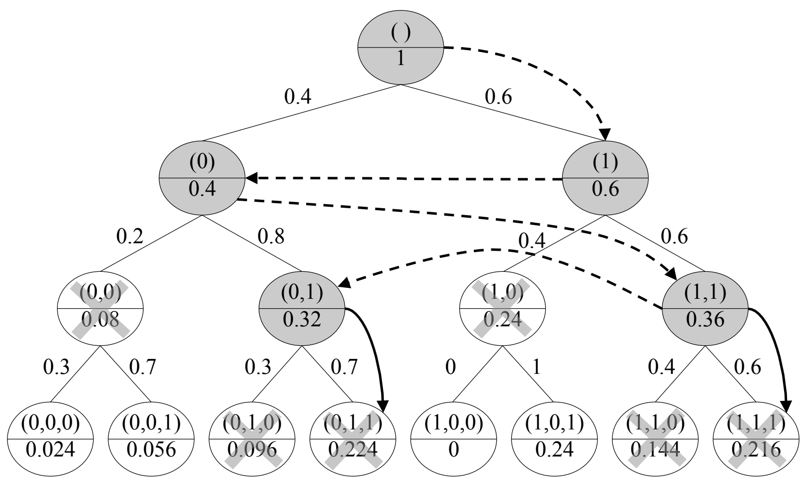


Fig. 4 An example of paths followed by the ϵ-approximate algorithm when $\epsilon = 0.25$ for three labels ($m = 3$). The dotted arrows show the path followed by the algorithm.

having the highest probability. This node is already a leaf and, as expected, a leaf of an optimal solution (which, in this case, is the sole optimal solution).

Conversely, the algorithm tends to be the method based on greedy search (CC) as ϵ grows, being the actual greedy search method (CC) in the case of $\epsilon = 0.5$ (or, equivalently, $\epsilon = 2^{-1}$). This is so because two situations are possible in this case: i) only one node has a marginal joint conditional probability greater than ϵ, in which case the algorithm follows one path; or ii) no node has a marginal joint conditional probability greater than ϵ, in which case a greedy search is applied from here to the bottom of the tree (given that the aforementioned latter situation takes place).

Let us now study the intermediate value $\epsilon = 0.25$ in Figure 4, which can be considered a value without any loss of generality. This example illustrates the aforementioned three possible situations: i) both children of the root have a probability greater than ϵ, and hence, both will be included in the list of candidate nodes to be expanded, Q (in fact, both are expanded); ii) they only have one child whose probability is greater than ϵ (the right child in both cases) and hence, the right children of both nodes are included in the list of candidate nodes to be expanded, Q, and the left children are discarded and finally, iii) both children of these two right nodes (the ones labelled by (0,1) and (1,1)) have a probability lower than ϵ, so they are included in the non-survival node list, K. Eventually, as there are no more candidate nodes to be expanded, a greedy search is applied to the last two right nodes included in the non-survival node list, K (the ones labelled (0,1) and (1,1)). Subsequently, the solution provided is the combination of labels of the fourth leaf, whose probability is 0.224, as it is greater than the probability of the last leaf, 0.216. As can be seen, this example also shows that the algorithm may not provide the optimal solution corresponding to the probability of the sixth leaf, 0.24.

The interpretation of the method for a generic value of $\epsilon = 2^{-k}$ is that the method ensures a partial optimal solution at least until the k-th level of the tree. Moreover, the solution remains optimal in levels below the k-th level if the highest marginal joint conditional probability remains higher than 2^{-k} in these levels. As a particular case, if this situation persits until reaching a leaf, then the algorithm obtains an optimal solution. Also notice that the partial combination of labels which is optimal at least until the k-th level or which may be optimal until levels below the k-th level can be different from that of the global optimal solution until such levels, as the global optimal solution also depends on what happens subsequently. In this respect, from $(k + 1)$-th level onwards, if the joint conditional probability of an optimal solution is greater than ϵ, then none node is discarded and hence this global optimal solution is reached. Conversely, if the entire joint conditional probability of an optimal solution is lower than ϵ, then there exists a node in the path of this leaf with a marginal conditional probability lower than ϵ. If so, this node will be discarded so that optimal solution will not be obtained. However, if this situation arises, a greedy search is applied starting at this discarded node. Therefore, the aforementioned optimal solution is reached if the path followed by the greedy search from this node coincides with the path leading to that optimal solution.

Consequently, this algorithm estimates the risk minimizer for the subset 0/1 loss to a greater or lesser extent depending on the value of $\epsilon = 2^{-k}$. Moreover, a theoretical analysis of this estimation [6] enables bounding its goodness as a function of the number of iterations, which in turn depends on ϵ. Particularly, and in the same direction followed for the method based on greedy search (CC), this analysis establishes that this algorithm needs fewer than $\mathbf{O}(m2^k)$ iterations to find a prediction that allows upper bounding the regret, $r_L(f)$, of the classifier f by $2^{-k} - 2^{-m}$ for the subset 0/1 loss and $k \leq m$, once again under the assumption that a perfect estimate of the joint conditional probability, $P(\boldsymbol{y} \,|\, \boldsymbol{x})$, is obtained. Hence, if the probability distribution for which the entire joint mode has a probability mass greater than 2^{-k}, then the algorithm needs fewer than $m2^k$ iterations to find a prediction that corresponds to this mode. Note that the particular case of $\epsilon = 0$ (or $k = m$) makes the bound becomes 0.

4 The improved ϵ-approximate algorithm

The methods based on uniform-cost search, such as the ϵ-approximate algorithm, only consider already known information, disregarding other information susceptible to being exploited. In terms of search algorithms, this means that for a certain node, only information on the path from the root to the node (the known path) is taken and information on the path from the node to a leaf (the unknown path) is ignored, leading to a non-informed search strategy. Unlike this kind of search, informed search methods, such as the well-known A* algorithm, consider the unknown information via a heuristic. Methods of this kind have already been studied for performing inference in PCC [18,20,21], demonstrating their power when the inference is not highly direct. However, the computational time spent on calculating the heuristic does not offset the reduction in the number of nodes explored when the inference is highly direct. This is the reason why the ϵ-approximate algorithm can be considered the best alternative for inference in PCC so far. The contribution of the present paper is to transform the ϵ-approximate algorithm into an informed search method in order to find optimal solutions in a more direct way (avoiding unnecessary backtracking). For this purpose, a heuristic able to provide information for a certain node to a leaf is included in the original ϵ-approximate algorithm. At the same time, however, the resulting method must be fast enough to avoid the drawbacks of the heuristics designed for A* [18,20,21], whose computation slows down the A* algorithm.

Formally, the idea is to provide an estimation of the entire joint conditional probability of a solution closer to the actual value, which is given by the product rule of probability theory (Equation (6)). The fact is that only some terms in Equation (6) can be estimated, whereas the proposal for the others consists in obtaining an upper bound as close as possible to the actual value. In this respect, the ϵ-approximate algorithm computes the marginal joint conditional probability of the labels from ℓ_1 to ℓ_j (see Equation (7)) and upper bounds the marginal joint conditional probability of the labels from ℓ_{j+1} to ℓ_m by 1, that is

$$\begin{aligned}
\mathbf{P}(\boldsymbol{y}\,|\,\boldsymbol{x}) &= \prod_{i=1}^{m} \mathbf{P}(y_i\,|\,\boldsymbol{x}, y_1, \ldots, y_{i-1}) \\
&= \prod_{i=1}^{j} \mathbf{P}(y_i\,|\,\boldsymbol{x}, y_1, \ldots, y_{i-1}) \cdot \prod_{i=j+1}^{m} \mathbf{P}(y_i\,|\,\boldsymbol{x}, y_1, \ldots, y_{i-1}) \\
&\leq \prod_{i=1}^{j} \mathbf{P}(y_i\,|\,\boldsymbol{x}, y_1, \ldots, y_{i-1}) \cdot \prod_{i=j+1}^{m} 1 = \prod_{i=1}^{j} \mathbf{P}(y_i\,|\,\boldsymbol{x}, y_1, \ldots, y_{i-1}). \quad (8)
\end{aligned}$$

Although ϵ-approximate is an appealing approach, the upper bound of 1 for the probabilities from the labels ℓ_{j+1} to ℓ_m is quite clumsy, as any probability is upper bounded by 1. The proposal of this paper is to obtain a tighter bound; i.e. one as close as possible to the actual value. Hence, let us now focus on obtaining a fast-to-compute bound for the conditional probability, $\mathbf{P}(y_i\,|\,\boldsymbol{x}, y_1, \ldots, y_{i-1})$, for $j+1 \leq i \leq m$.

4.1 The maximum as a bound

Before continuing, let us recall that the values for labels $\ell_1, \dots, \ell_j$ are known. Hence, a straightforward bound for the conditional probability $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \dots, y_{i-1})$ (or, equivalently, $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \dots, y_j,\ y_{j+1}, \dots, y_{i-1})$, expanding the expression for j) can be the maximum value this probability reaches when the combination of labels from ℓ_{j+1} to ℓ_{i-1} varies, that is

$$\mathbf{P}(y_i \,|\boldsymbol{x}, y_1,\dots, y_j, y_{j+1},\dots, y_{i-1}) \leq \max_{\substack{(y_{j+1},\dots,y_{i-1})\\ \in\{0,1\}^{i-1-j}}} \mathbf{P}(y_i \,|\boldsymbol{x}, y_1,\dots, y_j, y_{j+1},\dots, y_{i-1}). \quad (9)$$

Since it is quite common to employ logistic regression for the models, f_i, in PCC in order to generate probabilities, the expression of the conditional probability $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1,\dots, y_j, y_{j+1},\dots, y_{i-1})$ from the model $f_i(\boldsymbol{x}, y_1,\dots, y_j, y_{j+1},\dots, y_{i-1})$ is

$$\mathbf{P}(y_i|\boldsymbol{x}, y_1,\dots, y_j, y_{j+1},\dots, y_{i-1}) = \begin{cases} \frac{1}{1+e^{-f_i(\boldsymbol{x},y_1,\dots,y_j,y_{j+1},\dots,y_{i-1})}} & \text{if } y_i=1 \\ 1-\frac{1}{1+e^{-f_i(\boldsymbol{x},y_1,\dots,y_j,y_{j+1},\dots,y_{i-1})}} & \text{if } y_i=0, \end{cases} \quad (10)$$

where it can be seen that

- Each conditional probability $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \dots, y_j, y_{j+1}, \dots, y_{i-1})$ is estimated by means of the model $f_i(\boldsymbol{x}, y_1, \dots, y_j, y_{j+1}, \dots, y_{i-1})$ and a sigmoid function.
- The expression for $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \dots, y_j, y_{j+1}, \dots, y_{i-1})$ is a piecewise function and differs depending on the value that the label ℓ_i takes. Hence, the maximum over the labels $\ell_{j+1}, \dots, \ell_{i-1}$ must be computed under both assumptions; i.e., the assumption of the label ℓ_i is present ($y_i = 1$) and the assumption of the label ℓ_i is absent ($y_i = 0$).

Therefore, taking into account these two points, especially the expression of $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \dots, y_j, y_{j+1}, \dots, y_{i-1})$ in Equation (10), computing the maximum of $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \dots, y_j, y_{j+1}, \dots, y_{i-1})$ over the labels $\ell_{j+1}, \dots, \ell_{i-1}$ can be reduced to estimating the maximum and minimum of $f_i(\boldsymbol{x}, y_1, \dots, y_j, y_{j+1}, \dots, y_{i-1})$ over the labels $\ell_{j+1}, \dots, \ell_{i-1}$. Specifically,

i) if the label ℓ_i is present ($y_i = 1$), then, the probability $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \dots, y_j, y_{j+1}, \dots, y_{i-1})$ will be maximum over the labels $\ell_{j+1}, \dots, \ell_{i-1}$ if $f_i(\boldsymbol{x}, y_1, \dots, y_j, y_{j+1}, \dots, y_{i-1})$ is maximum over the labels $\ell_{j+1}, \dots, \ell_{i-1}$ (see the definition of $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \dots, y_j,\ y_{j+1}, \dots, y_{i-1})$, where $y_i = 1$ in Equation (10)) and
ii) if the label ℓ_i is absent ($y_i = 0$), then, the probability $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \dots, y_j, y_{j+1}, \dots, y_{i-1})$ will be maximum over the labels $\ell_{j+1}, \dots, \ell_{i-1}$ if $f_i(\boldsymbol{x}, y_1, \dots, y_j, y_{j+1}, \dots, y_{i-1})$ is minimum over the labels $\ell_{j+1}, \dots, \ell_{i-1}$ (see the definition of $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \dots, y_j,\ y_{j+1}, \dots, y_{i-1})$, where $y_i = 0$ in Equation (10)) .

The following subsection addresses the computation of these maximum and minimum of $f_i(\boldsymbol{x}, y_1, \dots, y_j, y_{j+1}, \dots, y_{i-1})$, for which the assumption of each model $f_i(\boldsymbol{x}, y_1, \dots, y_j, y_{j+1}, \dots, y_{i-1})$ being linear is stated.

4.2 Assuming linearity

Inspired by the deduction employed in the heuristic for A* [18,20] and in order to compute the maximum and minimum of $f_i(\boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1})$ over the labels $\ell_{j+1}, \ldots, \ell_{i-1}$, let us continue under the assumption that $f_i(\boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1})$ is a linear model. In this case, the expression of $f_i(\boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1})$ can be written (already split into the labels from ℓ_i to ℓ_j and from ℓ_{j+1} to ℓ_{i-1}) as

$$f_i(\boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1}) = \langle \boldsymbol{w}_x^i, \boldsymbol{x} \rangle + \sum_{k=1}^{j} w_{y,k}^i y_k + \sum_{k=j+1}^{i-1} w_{y,k}^i y_k + \beta^i, \quad (11)$$

or, equivalently,

$$f_i(\boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1}) = C_i(\boldsymbol{x}, y_1, \ldots, y_j) + g_i(y_{j+1}, \ldots, y_{i-1}), \quad (12)$$

where C_i can be considered as constant, since the maximum and the minimum values are simply computed over the labels $\ell_{j+1}, \ldots, \ell_{i-1}$; that is,

$$C_i(\boldsymbol{x}, y_1, \ldots, y_j) = \langle \boldsymbol{w}_x^i, \boldsymbol{x} \rangle + \sum_{k=1}^{j} w_{y,k}^i y_k + \beta^i. \quad (13)$$

and g_i is the function from which the maximum and minimum must be obtained; that is,

$$g_i(y_{j+1}, \ldots, y_{i-1}) = \sum_{k=j+1}^{i-1} w_{y,k}^i y_k. \quad (14)$$

Now, in order to compute the maximum and the minimum values of g_i when the aforementioned two situations occur, namely when $y_i = 1$ and when $y_i = 0$, let us respectively 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,

$$\begin{array}{l} K_{i,j}^+ = \{k \mid j+1 \leq k \leq i-1,\ w_{y,k}^i \geq 0\} \\ K_{i,j}^- = \{k \mid j+1 \leq k \leq i-1,\ w_{y,k}^i < 0\}. \end{array} \quad (15)$$

Thus, g_i is maximum when y_k for $j+1 \leq k \leq i-1$ are

$$y_k = \begin{cases} 1 \text{ if } k \in K_{i,j}^+ \\ 0 \text{ if } k \in K_{i,j}^-, \end{cases} \quad (16)$$

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

$$y_k = \begin{cases} 1 \text{ if } k \in K_{i,j}^- \\ 0 \text{ if } k \in K_{i,j}^+. \end{cases} \quad (17)$$

Hence,

i) Let $y_{j+1}^{i,1}, \ldots, y_{i-1}^{i,1} \in \{0,1\}$ be the values of Equation (16) that maximize g_i, i.e., the values that maximize $f_i(\boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1})$ when $y_i = 1$ (see the superindex 1 in these values), and hence the values which make $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1})$ be maximum when $y_i = 1$ and

ii) Let $y_{j+1}^{i,0}, \ldots, y_{i-1}^{i,0} \in \{0,1\}$ be the values of Equation (17) that minimize g_i, i.e., the values that minimize $f_i(\boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1})$ when $y_i = 0$ (see the superindex 0 in these values), and hence, the values which make $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1})$ be maximum when $y_i = 0$.

Thus, $\max_{\substack{(y_{j+1},\ldots,y_{i-1}) \\ \in\{0,1\}^{i-1-j}}} \mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1})$ in Equation (9) will be

$$\max_{v\in\{0,1\}} \mathbf{P}(y_i = v \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}^{i,v}, \ldots, y_{i-1}^{i,v}), \tag{18}$$

and hence Equation (9) becomes

$$\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1,\ldots, y_j, y_{j+1},\ldots, y_{i-1}) \leq \max_{v\in\{0,1\}} \mathbf{P}(y_i = v \,|\, \boldsymbol{x}, y_1,\ldots, y_j, y_{j+1}^{i,v},\ldots, y_{i-1}^{i,v}). \tag{19}$$

According to Equation (10), which that involves the sigmoid function, $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1,\ldots, y_j, y_{j+1}^{i,v},\ldots, y_{i-1}^{i,v})$ reaches the maximum over the labels $\ell_{j+1}, \ldots, \ell_{i-1}$ when $y_i = 1$ if

$$f_i(\boldsymbol{x}, y_1,\ldots, y_j, y_{j+1}^{i,1},\ldots, y_{i-1}^{i,1}) \geq 1 - f_i(\boldsymbol{x}, y_1,\ldots, y_j, y_{j+1}^{i,0},\ldots, y_{i-1}^{i,0}), \tag{20}$$

and it reaches the maximum over the same labels when $y_i = 0$, otherwise. Notice that the values of $y_{j+1}, \cdots, y_{i-1}$ differ if $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1,\ldots, y_j, y_{j+1},\ldots, y_{i-1})$ reaches the maximum when $y_i = 1$ or if $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1},\ldots, y_{i-1})$ reaches the maximum when $y_i = 0$. In fact, these values are complementary to each other; i.e. $y_k^{i,1} = 1 - y_k^{i,0} \in [0,1]$ for $j+1 \leq k \leq i-1$ (see Equations (16) and (17)). This is so thanks to the assumption of $f_i(\boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}, \ldots, y_{i-1})$ being linear (see Equation (14)).

Therefore, the bound in Equation (8) of the entire joint conditional probability for the improved ϵ-approximate algorithm will be

$$\begin{aligned}\mathbf{P}(\boldsymbol{y} \,|\, \boldsymbol{x}) &= \prod_{j=1}^{m} \mathbf{P}(y_j \,|\, \boldsymbol{x}, y_1, \ldots, y_{j-1}) \\ &\leq \prod_{i=1}^{j} \mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_{i-1}) \cdot \\ &\quad \prod_{i=j+1}^{m} \max_{v\in\{0,1\}} \mathbf{P}(y_i = v \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y_{j+1}^{i,v}, \ldots, y_{i-1}^{i,v}).\end{aligned} \tag{21}$$

In view of Equation (21), the computation of the unknown part (the part that involves labels from ℓ_{j+1} to ℓ_m for which the maximum must be computed to obtain a bound) is expected to be considerably fast, since, given $\boldsymbol{x}$, it suffices to obtain the maximums for all levels once, at the beginning of the inference.

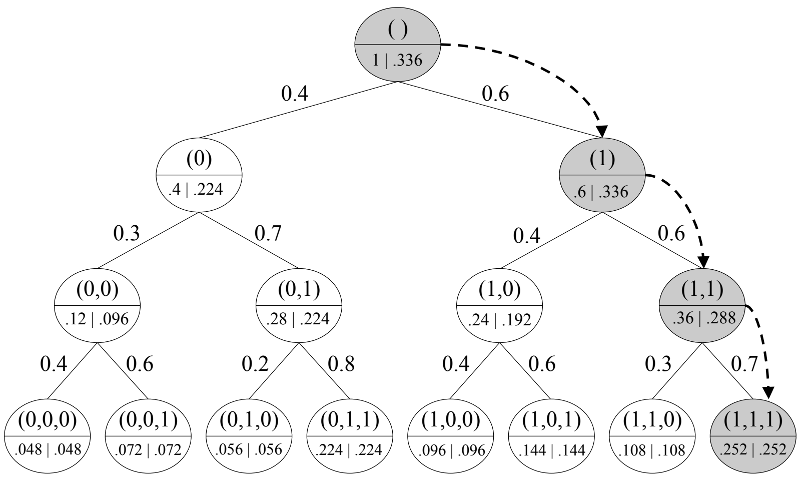


Fig. 5 An example of paths followed by the improved ϵ-approximate algorithm when $\epsilon = 0$ for three labels ($m = 3$). The dotted arrows show the path followed by the algorithm.

In fact, the cost of computing the maximum values shown in Equation (21) is of the order of m^2. This is so because there are m models (one per level) for which the respective maximum must be computed. In turn, for each model and due to the assumption of linearity, it suffices to decide the values for the m labels according to Equations (15), (16) and (17).

This bound for the unknown part (the part that involves labels from ℓ_{j+1} to ℓ_m) presents similarities with the heuristic proposed for A* [18,20]. The difference lies in the way it is computed. The heuristic for A* evaluates the maximum for each node to be expanded only considering the nodes of the subtree of this node. Here, however, the maximum is computed beforehand, just before starting to explore the whole tree. Hence, the estimation of the maximum in the case of A* is more accurate, which makes the heuristic more dominant[2]. Hence, A* will explore fewer nodes than the proposed algorithm. The ϵ-approximate algorithm goes to the other extreme, as it uses the bound of 1; i.e., the highest possible bound dealing with probabilities. Consequently, the proposed algorithm is situated somewhere between the A* and the ϵ-approximate algorithms. Hence, it is expected to take the property of A* of exploring few nodes and the property of the ϵ-approximate algorithm of being fast.

[2] A heuristic is more dominant than another heuristic if it estimates the gain of reaching the goal closer to the actual value. A consequence of this property is that it makes the former algorithm explore fewer nodes than the latter.

Algorithm 2 Pseudocode of the improved version of the ϵ-approximate algorithm

```
1: function $Imp$-$\epsilon$-APPROXIMATE
   Input: $\boldsymbol{x}$, $[\mathbf{W}, \boldsymbol{\beta}]$ a CC Linear Model, $m$, and $\epsilon \geq 0$
   Output: Label combination with highest probability for $\boldsymbol{x}$
2:     $WX \leftarrow \mathbf{W}_x * \boldsymbol{x}$    // Computes $\langle \boldsymbol{w}_x, \boldsymbol{x} \rangle$ for all labels
3:     $LevelHighProb \leftarrow \mathbf{1}_{m+1}$    // vector of $m+1$ ones
4:     for $label = [m : -1 : 2]$ do    // No label attributes in the 1st model
5:         $\boldsymbol{y}^+ \leftarrow [\![\mathbf{w}_y[label] >= 0]\!]$
6:         $P^+ \leftarrow 1/(1 + exp(-(WX[label] + \beta[label] + \langle \boldsymbol{w}_y[label], \boldsymbol{y}^+ \rangle)))$
7:         $\boldsymbol{y}^- \leftarrow [\![\mathbf{w}_y[label] < 0]\!]$
8:         $P^- \leftarrow 1/(1 + exp(-(WX[label] + \beta[label] + \langle \boldsymbol{w}_y[label], \boldsymbol{y}^- \rangle)))$
9:         $LevelHighProb(label) \leftarrow max(P^+, P^-) * LevelHighProb(label+1)$
10:    $Q \leftarrow \{[\ ], 1, 1, LevelHighProb(1)\}$    // $\{$(root node, $level, g, e\}$
11:    $K \leftarrow \emptyset$    // list of non-survived parents
12:    repeat
13:        $Node \leftarrow Max(Q)$    // node in $Q$ with highest $e$ value
14:        if $Node.Level = m$ then    // Node is a leaf
15:            return $Node.Labels$
16:        $level \leftarrow Node.Level + 1$
17:        $P \leftarrow 1/(1 + exp(-(WX[level] + \beta[level] + \langle \boldsymbol{w}_y[level], Node.Labels \rangle)))$
18:        $e\_left \leftarrow Node.g * (1 - P) * LevelHighProb(level)$
19:        if $e\_left \geq \epsilon$ then    // Left child
20:            $Insert(Q, \{[Node.Labels\ 0], level, Node.g * (1 - P), e\_left\})$
21:        $e\_right \leftarrow Node.g * P * LevelHighProb(level)$
22:        if $e\_right \geq \epsilon$ then    // Right child
23:            $Insert(Q, \{[Node.Labels\ 1], level, Node.g * P, e\_right\})$
24:        if $e\_left < \epsilon$ and $e\_right < \epsilon$ then
25:            $Insert(K, Node)$
26:    until $Q = \emptyset$
27:    $\epsilon \leftarrow 0$
28:    repeat
29:        $Node' \leftarrow$ pop first element in $K$
30:        $LeafProb \leftarrow GreedySearch(Node')$    // apply GS to reach a leaf
31:        if $LeafProb \geq \epsilon$ then
32:            $Best \leftarrow Node'$
33:            $\epsilon \leftarrow LeafProb$
34:    until $K \neq \emptyset$
35:    return $Best.Labels$
```

4.3 Theoretical analysis

Let us now compare the original and the improved version of the ϵ-approximate algorithm both theoretically and graphically. This means comparing the bound in Equation (8) that the original version of the ϵ-approximate algorithm takes and the bound in Equation (21), deduced above, that the improved version of the algorithm will take. It is clear that, the bound of the improved ϵ-approximate algorithm is more in line with the actual entire joint conditional probability than that of the ϵ-approximate algorithm. This is so because it is closer to it, as any probability is upper bounded by 1. For this reason, it may be stated that the improved ϵ-approximate algorithm follows a more direct path not only when it ensures a Bayes-optimal solution ($\epsilon = 0$), but also for any value of $\epsilon \geq 0$, as it is proved in the following theorem:

Theorem 1 *The improved ϵ-approximate algorithm always expands fewer than or an equal number of nodes to the original ϵ-approximate algorithm whatever the value of ϵ.*

Proof Let us first provide the proof for the particular case of $\epsilon = 0$. In this case, it is clear that the successive probabilities of the nodes that are chosen to be expanded decrease from the root node (the first node expanded) to an optimal solution of a leaf. Given that the probabilities provided by the improved ϵ-approximate algorithm are always lower than those provided by the original version of the algorithm, it is not possible for a node expanded by the improved version not be also expanded by the original version. This is so due to using the maximum as an heuristic, which is more informative than the heuristic of the constant 1. Hence, the original version expands at least the same number of nodes as the improved version and usually more.

Let us now generalize the proof for any value of ϵ. The difference lies in the existence of non-survival nodes, in whose case a leaf is not reached and it is necessary to apply a greedy search to the non-survival nodes. If the improved version of the algorithm explores more nodes than the original one, this is because the improved version generates more non-survival nodes than the original version. In this case, if the improved version has to perform a greedy search for a certain node, then the original version also has to do so for the same node or for some of its descendants, as there will be at least one descendant whose probability does not reach the ϵ value. This is true because the marginal joint probability for a node computed by the improved version is always greater than or equal to the actual marginal joint probability of the descendants of such a node. Thus, it is certain that at least all the leaves of the corresponding subtree have a joint probability less than ϵ. If the greedy search applied to that node entails exploring more nodes than the greedy search applied to any descendant, this will be offset at least by the expansion of the nodes on the path from the node to the descendant, since all of them will have a probability greater than ϵ. $\square$

Figure 5 shows the path followed by the improved ϵ-approximate algorithm when $\epsilon = 0$ for the same example in Figure 3, which shows the path followed by the original ϵ-approximate algorithm. The bottom of each node is labelled by the marginal joint conditional probability on the left (the known part of the path, which is the information taken by the ϵ-approximate algorithm) and by the estimated entire joint conditional probability (both the known and unknown parts of the path, see Equation (21)) on the right. Notice in this respect that the maximum conditional probabilities estimated by the models f_i for the first, second and third levels are respectively 0.6, 0.7 and 0.8. Hence, the improved ϵ-approximate algorithm upper bounds the unknown part of the root node by the product of these values; i.e., by $0.6 \cdot 0.7 \cdot 0.8 = 0.336$, which, multiplied by the known part (equal to 1), leads to 0.336. The known part of the first-level nodes are 0.4 for the left node and 0.6 for the right node, while the unknown part is upper bounded by the product of the maximum probabilities of the second and third levels; i.e., by $0.7 \cdot 0.8 = 0.56$. Hence, the left and right nodes of this level are respectively labelled on the bottom-right by $0.4 \cdot 0.56 = 0.224$ and $0.6 \cdot 0.56 = 0.336$. Analogously, the unknown part of the second-level nodes is upper bounded by 0.8, so the bottom-right of the nodes of this level is labelled by the product of 0.8 and the respective bottom-left value, which is the known part. Finally, all the information of the nodes on the last level (the leaves) is known. As expected, the path followed by the improved ϵ-approximate algorithm is more direct than that followed by the original ϵ-approximate algorithm. The reason is that the probabilities to decide that a node is expanded in the improved version are closer to the actual values and so are lower than those of the probabilities taken in the original version of the algorithm. This example for $\epsilon = 0$ suggests that the original version may find a better solution, as it may perform a greedy search over more nodes. It is true that these nodes may not be the most promising ones, but the possibility exists that they may lead to a leaf with a higher joint conditional probability. Furthermore, the improved version may provide a solution of the same quality or even better with an equivalent computational cost, simply decreasing the value of ϵ.

4.4 Implementation details

Obviously, the improved ϵ-approximate algorithm requires more calculations than the original ϵ-approximate algorithm, which only uses the unit constant to bound the conditional probability. Fortunately, most of the computations, especially the hardest ones, can be calculated beforehand. On the one hand, the labels whose values of $w^i_{y,k}$ are positive $(K^+_{i,j})$ or negative $(K^-_{i,j})$ are previously computed (see lines 5 and 7 of Algorithm 2). On the other hand, the maximum conditional probabilities, $\mathbf{P}(y_i \,|\, \boldsymbol{x}, y_1, \ldots, y_j, y^{i,v}_{j+1}, \ldots, y^{i,v}_{i-1})$, for $v = 1$ (when $y_i = 1$) and for $v = 0$ (when $y_i = 0$) over the values of the labels from ℓ_{j+1} to ℓ_{i-1} for all $i = 1, \ldots, m$ are obtained (see lines 6 and 8 of Algorithm 2). Finally, the maximum among these two probabilities is computed and accumulated in order to compute the entire joint conditional probability (see line 9 of Algorithm 2). The rest of the algorithm works just like the original ϵ-approximate algorithm does. It only differs from it when the value of the probability of the left and right children of the node susceptible to being expanded is computed. In this case, the improved ϵ-approximate algorithm makes use of the previously calculated probabilities (see lines 18 and 21 of Algorithm 2) and computes and additional product. Subsequently, it will be checked in the experiments whether the additional calculations that need to be computed in the improved version of the algorithm offset the savings of following a more direct path in the search for a solution.

5 Experiments

The experiments carried out have the goal of empirically confirming the theoretically promising properties of the improved version of the ϵ-approximate algorithm versus the original version. In addition, the study includes an analysis of whether the reduction in the number of nodes explored in the improved version offsets the time spent on computing the heuristic.

The ϵ-approximation algorithm is a benchmark method for inference in PCC that has been exhaustively compared with other state-of-the-art inference methods [19,20,21], proving to be one of the best options so far. Specifically, it has been compared with greedy search, beam search, Monte Carlo sampling and heuristic search by A*. Although this paper focuses on providing an improved version of the ϵ-approximation algorithm, the results obtained for these state-of-the-art methods are also shown. In case of A*, only the results of the algorithm with the most dominant admissible heuristic existing so far for inference in PCC, called h^∞, is presented as benchmark [20]. The already existing families of admissible heuristics [20,21] subsequently arose due to the excessive time spent on computing the h^∞ heuristic. These families include a parameter that controls the trade-off between the number of nodes explored and the computational time spent on evaluating the heuristic.

As deduced above, the improved ϵ-approximate algorithm has been limited to linear base models. Hence, logistic regression with probabilistic output [16] was used to build the binary classifiers, f_i, for all the methods. Only the regularization parameter, C, was optimized. This was done using a grid search over the values of $C \in \{10^{-3}, 10^{-2}, \ldots, 10^2, 10^3\}$ using the Brier Loss score [1] as the target function estimated via a 2-fold cross validation with five repetitions. The Brier Loss [1] is a proper score that measures the accuracy of probabilistic predictions, as logistic regression does. The expression is as follows

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{p_i} - a_i)^2, \qquad (22)$$

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).

The experiments were performed over the same benchmark multi-label datasets as in previous studies [19,20,21]. The main properties of all these datasets are shown in Table 1. As can be seen, there are differences in the number of attributes, instances and labels. The cardinality—the average number of labels per instance—varies from 1.07 to 4.27. As to the number of labels, there are some datasets with just 5, 6 or 7 labels, whereas others have more than 100 and one of them even has almost 400 labels.

Table 2 shows the subset 0/1 loss for the improved ϵ-approximate algorithm (Imp-ϵ-A), the original ϵ-approximate algorithm (ϵ-A) with different values of ϵ (0, 0.25 and 0.5), the A* algorithm with h^∞ (A*(h^∞)), greedy search (GS), beam search (BS) with different values of the beam (1, 2, 3, and 10) and Monte Carlos sampling (MC) with different values of the sample (10,

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

Table 2 Subset 0/1 loss. The first column corresponds to the methods that theoretically ensure an optimal solution. Those scores that are equal to or better than the optimal predictions obtained by Imp-ϵ-A, ϵ-A (both with $\epsilon = 0$) and A*(h^∞) are shown in bold.

Datasets	Imp-ϵ-A(.0) ϵ-A(.0) A*(h^∞)	Imp-ϵ-A(.25) ϵ-A(.25)	Imp-ϵ-A(.5) ϵ-A(.5) GS/BS(1)	BS (2)	BS (3)	BS (10)	MC (10)	MC (50)	MC (100)
bibtex	**81.92**	81.95	82.19	**81.88**	**81.92**	**81.92**	84.02	82.85	82.46
corel5k	**97.48**	98.62	98.90	98.30	98.04	**97.48**	99.64	98.98	98.26
emotions	**71.16**	71.82	72.83	72.16	71.32	**71.16**	80.77	73.68	73.84
enron	**83.14**	84.26	85.43	83.43	83.37	**83.14**	92.95	85.90	84.61
flags	**87.13**	87.16	**86.13**	88.21	**87.13**	**87.13**	96.39	91.21	89.24
image	**68.35**	**68.35**	69.75	**68.35**	**68.35**	**68.35**	69.20	**65.25**	**62.65**
mediamill*	**83.86**	84.58	85.80	84.10	**83.86**	**83.86**	90.90	85.88	84.88
medical	**30.37**	**30.37**	**30.67**	**30.37**	**30.37**	**30.37**	31.19	**30.16**	30.67
reuters	**22.73**	**22.70**	23.60	**22.69**	22.73	22.73	25.37	23.18	22.83
scene	**31.86**	**31.86**	33.28	31.90	**31.86**	**31.86**	33.53	**30.28**	**29.70**
slashdot	**51.80**	52.22	54.49	**51.77**	**51.80**	**51.80**	56.45	52.96	52.70
yeast	**76.95**	77.62	79.77	**76.83**	77.08	**76.95**	85.52	79.93	77.62

50 and 100). The first column corresponds to the methods that theoretically ensure Bayes-optimal solutions (Imp-ϵ-A and ϵ-A with $\epsilon = 0$ and A*(h^∞)), highlighted in bold. Those scores that are equal to or better than the optimal predictions reached by Imp-ϵ-A, ϵ-A (both with $\epsilon = 0$) and A*(h^∞) are likewise shown in bold. It is a fact that some methods do not guarantee an optimal label combination (e.g. BS and MC). Hence, they may predict another label combination with a lower joint conditional probability than the optimal joint conditional probability. However, they may possible provide better subset 0/1 scores for certain testing samples. This is due to a number of reasons, mainly a) the relatively small size of the testing sample from which it is not possible to perfectly estimate the subset 0/1 loss because it does not perfectly represent the population, and b) the models, f_i, obtained to estimate the joint conditional probability, $P(\boldsymbol{y} \,|\, \boldsymbol{x})$, do not usually return actual estimations, because they are obtained from a training sample that may not perfectly represent the population. Theoretically, under perfect conditions (large test sets and perfect models), Imp-ϵ-A and ϵ-A with $\epsilon = 0$ and A* with admissible heuristics would obtain the best scores. In general, the performance of Imp-ϵ-A and ϵ-

A decreases as the value of ϵ increases, the performance of the BS method increases as the value of the beam increases, and the performance of MC improves as the size of the sample grows.

Focusing now on the Imp-ϵ-A and ϵ-A algorithms, both provide an optimal solution for $\epsilon = 0$, which is the case of the greatest interest. For a value $\epsilon > 0$, the solutions may differ. This is because the improved version uses a more informative heuristic, which ensures exploring fewer nodes than the original version. Hence, the original version may explore more solutions, applying the final greedy search step to a higher number of nodes and, in such cases, it could sometimes perform better than the improved version. However, the same result can be achieved using the improved version with a lower value of ϵ. Thus, it is meaningless to compare the performance of both when $\epsilon > 0$, because it is always true that the improved version needs to explore fewer nodes than the original one to achieve the same level of performance. Notice that this is all simply a consequence of Theorem 1.

Table 3 Average number of explored nodes (top) and prediction time in milliseconds (bottom) averaged per instance. The table was split in three groups of columns: the first one refers to Imp-ϵ-A and ϵ-A, both with $\epsilon = 0$ together with A*(h^{∞}) (those that obtain optimal solutions); the second one reports the scores for Imp-ϵ-A and ϵ-A, both with $\epsilon = 0.25$, and the last one shows the scores for Imp-ϵ-A and ϵ-A, both with $\epsilon = 0.5$ (in fact, for this value, both methods are the same algorithm). The best scores within the same group are highlighted in bold. In addition, the rank for both the number of nodes explored and the computational time, though only comparing Imp-ϵ-A and ϵ-A with the same value of ϵ is shown in brackets. The averaged rank for all datasets is also shown in brackets.

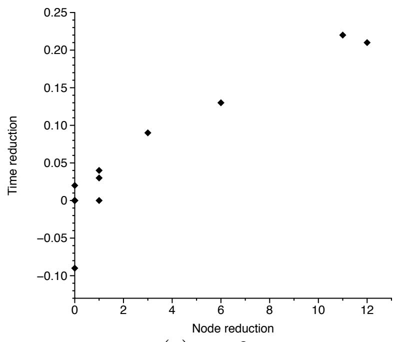
Dataset	A*(h^{∞})	Imp-ϵ-A(.0)	ϵ-A(.0)	Imp-ϵ-A(.25)	ϵ-A(.25)	Imp-ϵ-A(.5) ϵ-A(.5)
bibtex (159)	**215.91**	278.04 (1)	289.27 (2)	**181.53** (1)	184.00 (2)	160
corel5k (374)	**1338.62**	1470.83 (1)	1474.17 (2)	**437.47** (1)	517.11 (2)	375
emotions (6)	**7.68**	9.66 (1)	10.67 (2)	**8.19** (1)	10.78 (2)	7
enron (53)	**75.51**	109.07 (1)	114.81 (2)	**69.29** (1)	77.29 (2)	54
flags (7)	**8.79**	10.53 (1)	22.56 (2)	**8.00** (1)	16.33 (2)	8
image (5)	**6.11**	6.32 (1)	7.33 (2)	**6.15** (1)	7.66 (2)	6
mediamill *(101)	**178.89**	191.68 (1)	191.76 (2)	**142.22** (1)	142.37 (2)	102
medical (45)	**46.40**	46.60 (1)	46.64 (2)	**46.61** (1)	46.65 (2)	46
reuters (7)	**8.13**	8.24 (1.5)	8.24 (1.5)	**8.25** (1)	8.26 (2)	8
scene (6)	**7.15**	7.25 (1.5)	7.25 (1.5)	**7.25** (1.5)	**7.25** (1.5)	7
slashdot (22)	**24.84**	25.28 (1)	25.29 (2)	**24.87** (1)	24.89 (2)	23
yeast (14)	**21.01**	25.48 (1)	26.09 (2)	**24.32** (1)	26.02 (2)	25
avg. rank		(1.08)	(1.92)	(1.04)	(1.96)	
bibtex (159)	840.75	**15.98** (1)	16.20 (2)	**9.90** (1)	9.91 (2)	6.77
corel5k (374)	23834.37	**135.91** (1)	136.00 (2)	**15.79** (1)	16.06 (2)	16.62
emotions (6)	1.06	**0.60** (1)	0.63 (2)	**0.51** (1)	0.56 (2)	0.29
enron (53)	108.38	**7.06** (1)	7.19 (2)	**2.96** (1)	2.98 (2)	2.21
flags (7)	1.48	**0.98** (1)	1.19 (2)	**0.54** (1)	0.72 (2)	0.32
image (5)	0.71	**0.43** (1)	0.47 (2)	**0.42** (1)	0.46 (2)	0.25
mediamill* (101)	484.70	11.61 (2)	**11.52** (1)	5.56 (2)	**5.54** (1)	4.07
medical (45)	49.46	**2.70** (1.5)	**2.70** (1.5)	2.67 (2)	**2.66** (1)	1.89
reuters (7)	1.31	**0.51** (1.5)	**0.51** (1.5)	**0.51** (1.5)	**0.51** (1.5)	0.33
scene (6)	0.98	**0.46** (1.5)	**0.46** (1.5)	0.46 (2)	**0.45** (1)	0.29
slashdot (22)	12.42	**1.47** (1)	1.49 (2)	**1.43** (1.5)	**1.43** (1.5)	0.93
yeast (14)	7.47	**1.47** (1.5)	**1.47** (1.5)	**1.03** (1)	1.04 (2)	0.60
avg. rank		(1.25)	(1.75)	(1.33)	(1.67)	

Table 4 Pairs of methods with significant differences according to Bergmann-Hommel's test at a level of significance of 90% or 99%

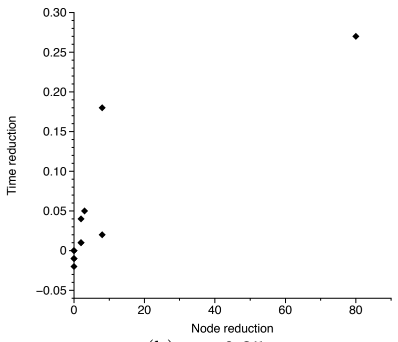
	Number of nodes explored	Computational time
Imp-ϵ-A(0) $\succ$ ϵ-A(0)	99%	90%
Imp-ϵ-A(0.25) $\succ$ ϵ-A(0.25)	99%	—

Table 3 shows the number of nodes explored together with the time spent on the inference process, measured in milliseconds averaged per instance. The table was split into three groups of columns: the first one refers to Imp-ϵ-A and ϵ-A both with $\epsilon = 0$ together with A*(h^∞) (those that reach optimal solutions); the second one reports the scores for Imp-ϵ-A and ϵ-A, both with $\epsilon = 0.25$; and the last one shows the scores for Imp-ϵ-A and ϵ-A, both with $\epsilon = 0.5$ (in fact, for this value, both methods are the same algorithm). The best scores within the same group are highlighted in bold. For the case of $\epsilon = 0.5$ (GS and BS(1)), both algorithms are clearly the fastest methods and the ones which explore the least number of nodes (only exploring the number of labels plus one nodes). However, they do so at the cost of failing in performance, since, in fact, there is not guarantee that they will obtain optimal solutions (see Table 2). In this respect, the number of nodes and the inference process time for BS and MC are not given because their respective values soar, especially for high values of their respective parameters, or at least for values needed to achieve accurate performance [19]. Besides, as stated before, both methods do not ensure optimal solutions. However, the number of nodes explored and the inference process time for A*(h^∞) is reported as a benchmark of the heuristic search methods. It ensures optimal solutions, as the heuristic h^∞ is admissible, and the method clearly explores the least number of nodes among the methods that ensure reaching optimal solutions, but it clearly reports the worst computational time, as stated previously, because of the time spent on computing the heuristic.

As to Imp-ϵ-A and ϵ-A, Table 3 also reports the rank concerning the number of nodes explored and the computational time, though only for these two methods, as they are the target methods of the paper. The averaged rank for all datasets is also shown. All the ranks are shown in brackets. Furthermore, Table 4 shows the Bergmann-Hommel procedure, which is one of the most powerful among existing tests [9], once again only for Imp-ϵ-A and ϵ-A for being the target of the paper. As was expected in theory, the Imp-ϵ-A always expands fewer nodes than ϵ-A for the same value of ϵ, or at least an equal number of nodes. The differences are not so high in general, but sufficient to obtain a lower averaged rank and to be statistically significant even at level of significance of 99% and for the two values of $\epsilon = 0$ and $\epsilon = 0.25$. Especially worth noting is the result obtained for the flags dataset, whose number of nodes is reduced by half. In fact, the number of nodes explored almost reaches the depth of the tree using Imp-ϵ-A. This fact leads us to think that most of the arches with the maximum probability in each level (the heuristic is calculated from these probabilities) belong to a path that leads to an optimal solution. Curiously, the computational time is always slightly lower for most datasets (or even equal in some situations) for Imp-ϵ-A, despite the fact it needs to compute the heuristic. Hence, the hypothesis of the heuristic of being considerably fast to compute is also empirically verified. Only some exceptions appear in scene and medical for $\epsilon = 0.25$ and in mediamill for both values, although the differences are barely perceptible. In any case, the averaged rank for the computational time is lower for the improved version for both values of ϵ. However, only for $\epsilon = 0$ it is statistically significant and just at the 90% of significance level.



(a) $\epsilon = 0$



(b) $\epsilon = 0.25$

Fig. 6 Reduction in the number of nodes versus reduction in computational time of Imp-ϵ-A with regard to ϵ-A.

Figure 6 presents the reduction in the number of nodes versus the reduction in computational time of Imp-ϵ-A with regard to ϵ-A for $\epsilon = 0$ (Figure 6(a)) and for $\epsilon = 0.25$ (Figure 6(b)). There is clearly a direct relation between the reduction in the number of nodes explored and computational time. It can also be seen that as the value of ϵ increases, the points on the graph tend to be concentrated at the $(0, 0)$ point (recall that for $\epsilon = 0.5$, both algorithms are identical).

Given that the differences between Imp-ϵ-A and ϵ-A are not so high and that the number of nodes in most cases are close to the depth of the tree, some experiments including noise in the datasets were carried out to achieve a more in-depth analysis of the power of the Imp-ϵ-A for inference in PCC. In fact, for these datasets, both algorithms need to perform few backtracking steps in the tree. Presumably, this is so maybe because the probabilities provided by the models may not be so close to 0.5 or that they may even be close to 0 or 1. Thus, by adding noise the probabilities are expected to be closer to 0.5, making the inference problem more complex. Seeing as, the heuristic of Imp-ϵ-A estimates the probabilities more accurate (see Section 4.2), the algorithm is expected to follows a more direct path in the tree and hence will be substancially faster than ϵ-A.

Certain grades of noise in terms of percentage were added to the datasets. Noise needs only be included in the labels of the instances (the $\boldsymbol{y}$ part) rather than in the description of the instances (the $\boldsymbol{x}$ part). For this purpose, a certain percentage of the values of the labels was swapped from being relevant to being irrelevant and vice versa. This means that the value of either all their labels or only some of them or even none of them were swapped for each instance. The total percentage of noise included ranged from 0% to 26% for datasets with fewer than 22 labels. However, the maximum percentage considered for medical (45 labels), enron (53 labels), mediamill (101 labels) and bibtex (159 labels) was respectively 18%, 10%, 8% and 8%. The reason for this decision was the excessive experimental time needed to complete the experiments. The corel5k dataset was excluded from these experiments for the same reason, given its large number of labels (374).

Figures 7 and 8 show the computational time spent on the inference and averaged per instance for Imp-ϵ-A and ϵ-A for $\epsilon = 0$ and $\epsilon = 0.25$ when the aforementioned percentages of noise were included in the datasets. It can be seen that the improved version of the algorithm always spent less time than the original version. This is quite noticeable as the percentage of noise grows, even more so when $\epsilon = 0$ and for datasets with few or a medium number of labels. However, the differences between the two methods for datasets with a high number of labels (mediamill 101 and bibtex 159) are quite small, even for the cases of more noise. Furthermore, the computational time spent by the improved ϵ-approximate algorithm with $\epsilon = 0$ is seen to be quite similar to that of the original version with $\epsilon = 0.25$. This fact reinforces the hypothesis already mentioned in Section 4 regarding the possibility of employing the improved version with a lower value of ϵ and spending a comparable computational time.

6 Conclusions and future work

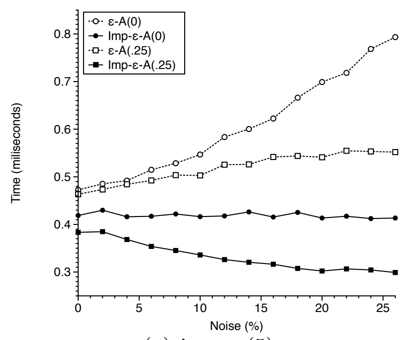
This paper presents an improvement based on heuristic search on the promising and well-known ϵ-approximate algorithm for inference in PCC. The starting hypothesis is that adding heuristic information to the criterion under a node is expanded when the algorithm searches for a solution avoids backtracking and makes the algorithm follow a more direct path. This hypothesis was theoretically proven and empirically confirmed in a series of experiments. Not only does the improved ϵ-approximate algorithm explore fewer nodes than the original version, but it is also faster in terms of computational time in spite of the additional computation of the heuristic that the improved version of the algorithm needs to compute with regard to the original version. This additional computation of the heuristic is actually extremely fast, as it is computed beforehand for each instance and it remains constant throughout the inference process.

As future work, it would be interesting to overcome the main drawback of the improved ϵ-approximate algorithm, namely, it is only applicable to linear base models in PCC. In this respect, bounding the maximum of the non-linear models if it is not possible to compute the exact maximum may be a possible future line of research.

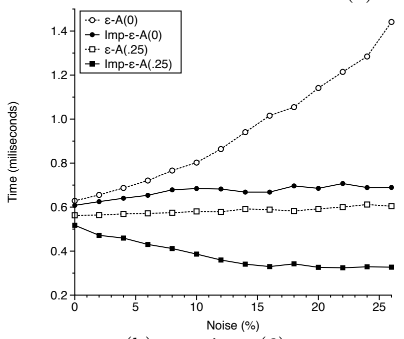
References

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 multi-label classification. Machine Learning **76**(2-3), 211–225 (2009)
3. Clare, A., King, R.D.: Knowledge discovery in multi-label phenotype data. In: European Conference on Data Mining and Knowledge Discovery (2001), pp. 42–53 (2001)
4. Dembczyński, K., Cheng, W., Hüllermeier, E.: Bayes optimal multilabel classification via probabilistic classifier chains. In: ICML, 2010, pp. 279–286 (2010)
5. 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)
6. Dembczynski, K., Waegeman, W., Hüllermeier, E.: An analysis of chaining in multi-label classification. In: ECAI, *Frontiers in Artificial Intelligence and Applications*, vol. 242, pp. 294–299. IOS Press (2012)
7. 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)
8. Fürnkranz, J., Hüllermeier, E., Loza Mencía, E., Brinker, K.: Multilabel classification via calibrated label ranking. Machine Learning **73**, 133–153 (2008)
9. García, S., Herrera, F.: An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. Journal of Machine Learning Research **9**(12), 2677 – 2694 (2008)
10. Ghamrawi, N., McCallum, A.: Collective multi-label classification. In: ACM Int. Conf. on Information and Knowledge Management, pp. 195–200. ACM (2005)
11. Gibaja, E., Ventura, S.: A tutorial on multilabel learning. ACM Comput. Surv. **47**(3), 52:1–52:38 (2015). DOI 10.1145/2716262
12. Godbole, S., Sarawagi, S.: Discriminative methods for multi-labeled classification. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining (2004), pp. 22–30 (2004)
13. Goncalves, E.C., Plastino, A., Freitas, A.A.: A genetic algorithm for optimizing the label ordering in multi-label classifier chains. In: 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, pp. 469–476 (2013). DOI 10.1109/ICTAI.2013.76
14. 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)
15. Kumar, A., Vembu, S., Menon, A.K., Elkan, C.: Beam search algorithms for multi-label learning. Mach. Learn. **92**(1), 65–89 (2013)
16. 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)
17. McCallum, A.K.: Multi-label text classification with a mixture model trained by em. In: AAAI 99 Workshop on Text Learning (1999)
18. Mena, D., Montañés, E., Quevedo, J.R., Del Coz, J.J.: Using A* for inference in probabilistic classifier chains. In: Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI'15, pp. 3707–3713. AAAI Press (2015)
19. Mena, D., Montañés, E., Quevedo, J.R., del Coz, J.J.: An overview of inference methods in probabilistic classifier chains for multilabel classification. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **6**(6), 215–230 (2016). DOI 10.1002/widm.1185
20. Mena, D., Montañés, E., Quevedo, J.R., del Coz, J.J.: A family of admissible heuristics for A* to perform inference in probabilistic classifier chains. Machine Learning **106**(1), 143–169 (2017). DOI 10.1007/s10994-016-5593-5
21. Mena, D., Quevedo, J.R., Montañés, E., del Coz, J.J.: A heuristic in A* for inference in nonlinear probabilistic classifier chains. Knowl.-Based Syst. **126**, 78–90 (2017). DOI 10.1016/j.knosys.2017.03.015

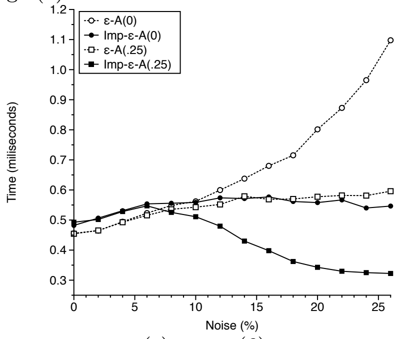
22. Montañés, E., Quevedo, J., del Coz, J.J.: Aggregating independent and dependent models to learn multi-label classifiers. In: ECML'11, pp. 484–500 (2011)
23. Montañés, E., Senge, R., Barranquero, J., Quevedo, J., del Coz, J.J., Hüllermeier, E.: Dependent binary relevance models for multi-label classification. Pattern Recognition **47**(3), 1494 – 1508 (2014)
24. Prati, R.: Fuzzy rule classifiers for multi-label classification (2015). DOI 10.1109/FUZZ-IEEE.2015.7337815
25. Qi, G.J., Hua, X.S., Rui, Y., Tang, J., Mei, T., Zhang, H.J.: Correlative multi-label video annotation. In: Proceedings of the International Conference on Multimedia, pp. 17–26. ACM, NY, USA (2007)
26. Read, J., Martino, L., Hollmén, J.: Multi-label methods for prediction with sequential data. CoRR **abs/1609.08349** (2016)
27. Read, J., Martino, L., Luengo, D.: Efficient monte carlo methods for multi-dimensional learning with classifier chains. Pattern Recognition **47**(3), 1535 – 1546 (2014)
28. Read, J., Martino, L., Olmos, P.M., Luengo, D.: Scalable multi-output label prediction: From classifier chains to classifier trellises. Pattern Recognition **48**(6), 2096 – 2109 (2015). DOI https://doi.org/10.1016/j.patcog.2015.01.004
29. Read, J., Pfahringer, B., Holmes, G.: Multi-label classification using ensembles of pruned sets. In: IEEE Int. Conf. on Data Mining, pp. 995–1000. IEEE (2008)
30. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. In: ECML/PKDD'09, LNCS, pp. 254–269. Springer (2009)
31. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Machine Learning **85**(3), 333–359 (2011)
32. Schapire, R.E., Singer, Y.: Boostexter: A boosting-based system for text categorization. Machine Learning pp. 135–168 (2000)
33. Tsoumakas, G., Katakis, I., Vlahavas, I.: Effective and efficient multilabel classification in domains with large number of labels (2008)
34. Tsoumakas, G., Vlahavas, I.: Random k-Labelsets: An ensemble method for multi-label classification. In: ECML/PKDD'07, pp. 406–417. Springer (2007)
35. Wu, Y.P., Lin, H.T.: Progressive random k-labelsets for cost-sensitive multi-label classification. Machine Learning **106**(5), 671–694 (2017). DOI 10.1007/s10994-016-5600-x
36. Zhang, M.L., Zhou, Z.H.: Multilabel neural networks with applications to functional genomics and text categorization. IEEE Trans. on Knowl. and Data Eng. **18**, 1338–1351 (2006)
37. Zhang, M.L., Zhou, Z.H.: Ml-knn: A lazy learning approach to multi-label learning. Pattern Recognition **40**(7), 2038–2048 (2007)



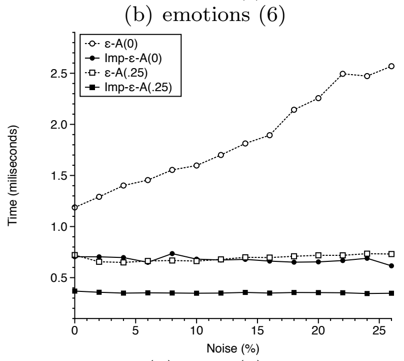
(a) image (5)



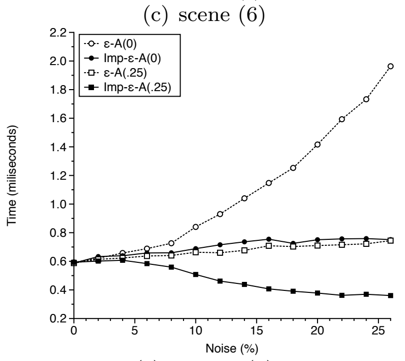
(b) emotions (6)



(c) scene (6)



(d) flags (7)



(e) reuters (7)

Fig. 7 Computational time employed (ms) for both versions of ϵ-A with $\epsilon = 0$ and $\epsilon = 0.25$ averaged per instance. Different percentages of noise are considered. All the datasets have few labels (from 5 to 7).

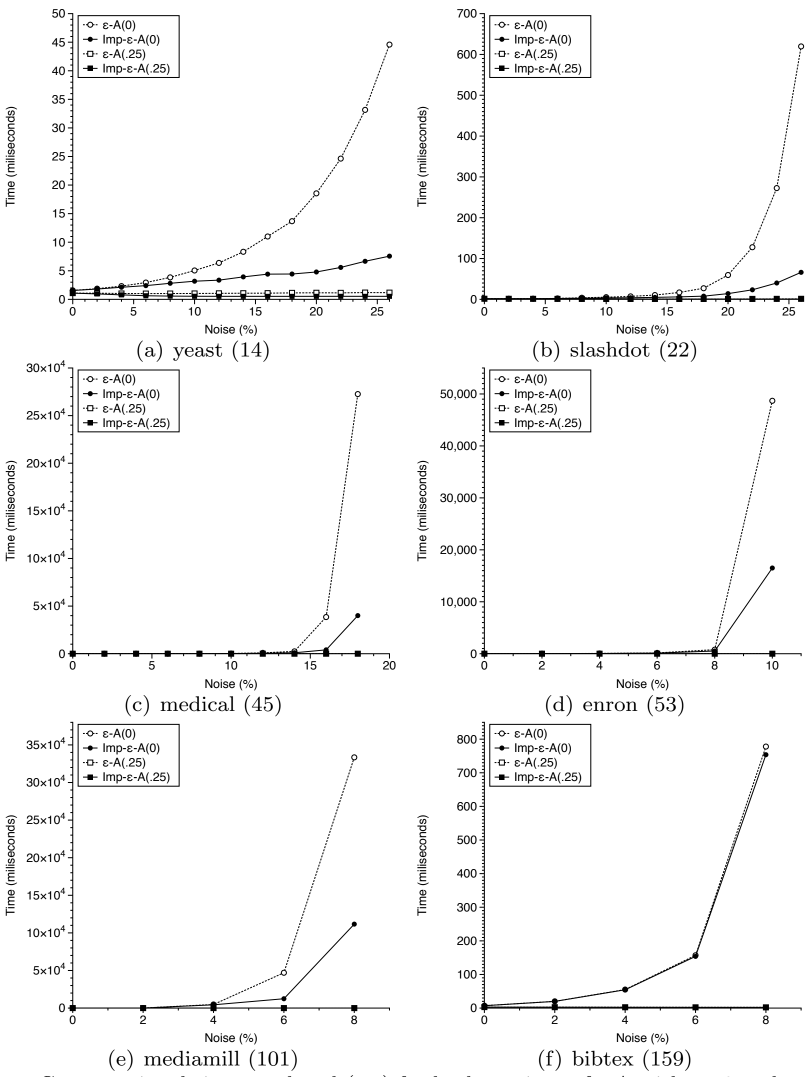


Fig. 8 Computational time employed (ms) for both versions of ϵ-A with $\epsilon = 0$ and $\epsilon = 0.25$ averaged per instance. Different percentages of noise are considered. The number of labels range from 14 to 159.