Overview

An overview of inference methods in probabilistic classifier chains for multilabel classification

Deiner Mena,[1,2] Elena Montañés,[1] José Ramón Quevedo[1] and Juan José del Coz[1]*

This study presents a review of the recent advances in performing inference in probabilistic classifier chains for multilabel classification. The interest of performing such inference arises in an attempt of improving the performance of the approach based on greedy search (the well-known CC method) and simultaneously reducing the computational cost of an exhaustive search (the well-known PCC method). Unlike PCC and as CC, inference techniques do not explore all the possible solutions, but they increase the performance of CC, sometimes reaching the optimal solution in terms of subset 0/1 loss, as PCC does. The ε-approximate algorithm, the method based on a beam search and Monte Carlo sampling are those techniques. An exhaustive set of experiments over a wide range of datasets are performed to analyze not only to which extent these techniques tend to produce optimal solutions, otherwise also to study their computational cost, both in terms of solutions explored and execution time. Only ε-approximate algorithm with ε=.0 theoretically guarantees reaching an optimal solution in terms of subset 0/1 loss. However, the other algorithms provide solutions close to an optimal solution, despite the fact they do not guarantee to reach an optimal solution. The ε-approximate algorithm is the most promising to balance the performance in terms of subset 0/1 loss against the number of solutions explored and execution time. The value of ε determines a degree to which one prefers to guarantee to reach an optimal solution at the expense of increasing the computational cost. © 2016 John Wiley & Sons, Ltd

How to cite this article:
WIREs Data Mining Knowl Discov 2016, 6:215–230. doi: 10.1002/widm.1185

INTRODUCTION

Multilabel classification[1] (MLC) is a machine learning problem in which models are able to assign a subset of (class) labels to each instance, unlike conventional (single-class) classification that involves predicting only a single class. MLC 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 multilabel learning comes with two fundamental challenges. The first one bears on the computational complexity of the algorithms. A complex approach might not be applicable in practice when the number of labels is large. Therefore, the scalability of algorithms is a key issue in this field. The second problem is related to the own nature of multilabel data. Not only the number of labels is typically large, otherwise each instance also belongs to a variable-sized subset of labels simultaneously. Moreover, and perhaps even more important, 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

*Correspondence to: juanjo@uniovi.es

[1]Artificial Intelligence Center, University of Oviedo at Gijón, Gijón, Spain

[2]Dept. de Ingeniería en Telecomunicaciones e Informática, Universidad Tecnológica del Chocó, Chocó, Colombia

Conflict of interest: The authors have declared no conflicts of interest for this article.

promising source of information, in addition to the information 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. First, researchers tried to adapt and extend different state-of-the-art binary or multiclass classification algorithms, including methods using decision trees,[2] neural networks,[3] support vector machines,[4] naive Bayes,[5] conditional random fields,[6] and boosting.[7] Secondly, they further analyzed in depth the label dependence and attempted to design new approaches that exploit label correlations.[8] In this regard, two kinds of label dependence have been formally distinguished: conditional dependence[6,9–13] and unconditional dependence.[3,14,15] Also, pairwise relations,[3,4,7,16,17] relations in sets of different sizes,[12,18,19] or relations in the whole set of labels[10,14,15] have also been exploited.

Regarding conditional label dependence, the approach called probabilistic classifier chains (PCC) has aroused great interest among the multilabel community, because it offers the nice property of being able to estimate the conditional joint distribution of the labels. However, the original PCC algorithm[9] suffers from high computational cost, as it performs an exhaustive search (ES) as inference strategy to obtain optimal solutions in terms of a given loss function. Then, several efforts that use different searching and sampling strategies in order to overcome this drawback are being made just now. This includes uniform-cost (UC) search,[20] beam search (BS),[21,22] and Monte Carlo sampling (MS).[20,23,24] All of these algorithms successfully estimate the optimal solution reached by the original PCC,[9] at the same time that they reduce the computational cost in terms of both the number of candidate solutions explored and execution time. This article studies in depth the behavior and the properties of all these algorithms, comparing their strategies and establishing their differences and similarities, paying special attention to the meaning of their parameters and the effect of the different values they can take. The methods are experimentally compared over a wide range of multilabel datasets, concluding that even those that do not theoretically guarantee obtaining optimal solutions also reach good performance. However, the ε-approximate algorithm shows to be a promising alternative even for values of ε that do not guarantee reaching optimal solutions. For this algorithm, it happens that renouncing to reach optimal solutions leads to reduce the computational cost in terms of candidate solutions explored and execution time and viceversa.

The rest of the article is organized as follows. *PCCs in MLC* section formally describes multilabel framework and the principles of PCC. *Inference in PCCs* section discusses the properties and behavior of the different existing approaches for inference in PCC. Exhaustive experiments are shown and discussed in *Experiments* section. Finally, *Conclusions* section exposes some conclusions and includes new directions of future work.

PCCs IN MLC

This section formally describes the MLC task. Let $\mathscr{L} = \{\ell_1, \ell_2, \dots, \ell_m\}$ be a finite and nonempty set of m labels and $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$ 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 $\mathcal{P}(\mathscr{L})$ of $\mathscr{L}$. To ease notation, we define y_i as a binary vector $y_i = (y_{i,1}, y_{i,2}, \dots, y_{i,m})$ 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 x_i. Hence, y_i is the observation of a corresponding random vector $\mathbf{Y} = (\mathbf{Y_1}, \mathbf{Y_2}, \dots, \mathbf{Y_m})$. 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 $h : \mathcal{X} \to \mathcal{Y}$ that minimizes the risk in terms of certain loss function $L(\cdot)$ when it provides a vector of relevant labels $y = h(x) = (h_1(x), h_2(x), \dots, h_m(x))$ for unlabeled query instances x. This risk can be defined as the expected loss over the joint distribution $\mathbf{P}(\mathbf{X}, \mathbf{Y})$, i.e.,

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

therefore, denoting by $\mathbf{P}(y \mid x)$ the conditional distribution $\mathbf{Y} = y$ given $\mathbf{X} = x$, the so-called risk minimizer h^* can be expressed by

$$h^*(x) = \operatorname*{argmin}_h \sum_{y \in \mathcal{Y}} \mathbf{P}(y|x) L(y, h(x)). \qquad (2)$$

Let us comment that the conditional distribution $\mathbf{P}(y \mid x)$ presents different properties which are crucial for optimizing different loss functions. At this respect, the strategy followed by a certain MLC algorithm for modeling label dependence determines the optimized loss function. Unfortunately, it is quite complex and confusing to find out the loss function optimized by several algorithms.

In 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, such as the F_1 measure or the *Jaccard* index. Here, we will focus on just the subset 0/1 loss because it is the measure that PCC is able to optimize. The subset 0/1 loss looks whether the predicted and relevant label subsets are equal or not. It is defined by

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

in which the expression $[[p]]$ evaluates to 1 when the predicate, p, is true and to 0 otherwise. Notice that it suffices taking the mode of the entire joint conditional distribution for optimizing this loss function. Formally, the risk minimizer adopts the simplified following form

$$h^*(x) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \mathbf{P}(y \mid x). \tag{4}$$

Among the methods that cope with MLC, the simplest straightforward approach is Binary Relevance (BR).[25] This method assumes independence among the labels and provides optimal prediction for subset 0/1 loss when such independence actually exists. Hence, it estimates $\mathbf{P}(y_j \mid x)$ for label ℓ_j just considering the description of the instances and using a probabilistic model $h_j : \mathcal{X} \to [0,1]$. The joint distribution of labels $\mathbf{Y} = (\mathbf{Y_1}, \mathbf{Y_2}, \ldots, \mathbf{Y_m})$ is computed using the product rule of probability assuming independence among labels, i.e., $\mathbf{P}(y \mid x) = \prod_{j=1}^{m} \mathbf{P}(y_j \mid x)$.

Concerning methods that take into account the possible dependence among labels, let us focus on those based on learning a chain of classifiers (as CC[12,26] or PCC[9]). First, these methods define an order of the label set. Then, following such order, they train a probabilistic binary classifier for each label ℓ_j to estimate $\mathbf{P}(y_j \mid x, y_1, \ldots, y_{j-1})$. Hence, the probabilistic model obtained for predicting label ℓ_j, denoted by h_j, is of the form

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

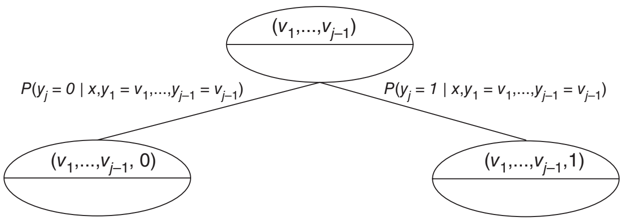
The training data for this classifier are the set $S_j = \{(\overline{x}_1, y_{1,j}), \ldots, (\overline{x}_n, y_{n,j})\}$ where $\overline{x}_i = (x, y_{i,1}, \ldots, y_{i,j-1})$, i.e., the features are x_i supplemented by the relevance of the labels $\ell_1, \ldots, \ell_{j-1}$ preceding ℓ_j in the 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 $\mathbf{Y} = (\mathbf{Y_1}, \mathbf{Y_2}, \ldots, \mathbf{Y_m})$, i.e., computing

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

Before analyzing this issue in the following section, notice 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 $P(y \mid x)$. 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, as error predictions are successively propagated.[11,27,28] In any case, in this article we assume an order of the labels in the chain in order to better analyze all these methods in insolation without taking into account other factors. Hence, we do not include any study about which order can be the best.

Finally, let us now 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 next section. In such tree, a node of the $(j-1)$-th level is labeled by $(v_1, \ldots, v_{j-1})$ with $v_i \in \{0, 1\}$ for $i = 1, \ldots, j-1$. This node has two children respectively labeled as $(v_1, \ldots, v_{j-1}, 0)$ and $(v_1, \ldots, v_{j-1}, 1)$ with marginal joint conditional probability $\mathbf{P}(y_1 = v_1, \ldots, y_{j-1} = v_{j-1}, y_j = 0 \mid x)$ and $\mathbf{P}(y_1 = v_1, \ldots, y_{j-1} = v_{j-1}, y_j = 1 \mid x)$, respectively. The weights of the edges between both parent and children are respectively $\mathbf{P}(y_j = 0 \mid x, y_1 = v_1, \ldots, y_{j-1} = v_{j-1})$ and $\mathbf{P}(y_j = 1 \mid x, y_1 = v_1, \ldots, y_{j-1} = v_{j-1})$, which are respectively estimated by $1 - h_j(x, v_1, \ldots, v_{j-1})$ and $h_j(x, v_1, \ldots, v_{j-1})$. The marginal joint conditional probability of the children is computed using the product rule of probability. Then, $\mathbf{P}(y_1 = v_1, \ldots, y_{j-1} = v_{j-1}, y_j = 0 \mid x) = \mathbf{P}(y_j = 0 \mid x, y_1 = v_1, \ldots, y_{j-1} = v_{j-1}) \cdot \mathbf{P}(y_1 = v_1, \ldots, y_{j-1} = v_{j-1} \mid x)$ and $\mathbf{P}(y_1 = v_1, \ldots, y_{j-1} = v_{j-1}, y_j = 1 \mid x) = \mathbf{P}(y_j = 1 \mid x, y_1 = v_1, \ldots, y_{j-1} = v_{j-1}) \cdot$



$(v_1,\ldots,v_{j-1})$
$P(y_j = 0 \mid x, y_1 = v_1,\ldots,y_{j-1} = v_{j-1})$
$P(y_j = 1 \mid x, y_1 = v_1,\ldots,y_{j-1} = v_{j-1})$
$(v_1,\ldots,v_{j-1}, 0)$
$(v_1,\ldots,v_{j-1}, 1)$

FIGURE 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 combination. The edges are labeled with the conditional probability.

$\mathbf{P}(y_1 = v_1, \ldots, y_{j-1} = v_{j-1} \mid x)$. The root node is labeled by the empty set. Figure 1 illustrates it.

INFERENCE IN PCCs

Several approaches have been proposed for inference in PCC. The method firstly proposed was that one based on greedy search (GS), being the integral part of the original CC method.[26] Its successor is the ES, called the PCC method.[9] The ε-approximate (ε-A) algorithm[20] is an UC search algorithm that can output optimal predictions in terms of subset 0/1 loss, reducing significantly the computational cost of ES. A more recent approach based on BS[21,22] presents good behavior both in terms of performance and computational cost. Finally, MS[20] is an appealing and simpler alternative[20,23] to overcome the high computational cost of ES. Before proceeding to cope with the specificity of all these inference methods, notice that the training phase is common to all of them, thus, the models h_j induced by the binary classifiers will be the same. So, in what follows we will focus just on the testing stage for a given unseen example x.

Greedy Search

At the testing stage, the GS strategy, originally called CC,[12,26] provides an output $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_m)$ for a new unlabeled instance x by successively querying each classifier h_j that estimates the conditional probability $\mathbf{P}(y_j \mid x, y_1, \ldots, y_{j-1})$. This means to explore 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 probabilities only differ in the marginal conditional probability term, $\mathbf{P}(y_j \mid x, y_1, \ldots, y_{j-1})$, because both children have the same parent. Thus, the path selected is the one of the child with the highest marginal conditional probability $\mathbf{P}(y_j \mid x, y_1, \ldots, y_{j-1})$. Notice that when $\hat{y}_j$ is estimated, h_j is applied, which needs both the feature vector x and the values for the labels from ℓ_1 to ℓ_{j-1}. In this regard, let us remind that $y_1, \ldots, y_{j-1}$ are not available in the testing stage, hence, the respective predictions $\hat{y}_1 = h_1(x), \hat{y}_2 = h_2(x, \hat{y}_1), \ldots, \hat{y}_{j-1} = h_{j-1}(x, \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{j-2})$ are taken instead. Thus, the prediction for an instance x is of the form $\hat{y} = (h_1(x), h_2(x, h_1(x)), h_3(x, h_1(x), h_2(x, h_1(x))), \ldots)$.

Figure 2(a) shows the path followed by an instance using this strategy. In this example, only the right node is explored at each level. The optimal solution is not reached, because the optimal solution is that which ends in the sixth leaf, whereas the method falls in the last leaf. The pseudocode of the GS method can be seen in Algorithm 1 (left) in which $v_R = \emptyset$ is the root node of the probability tree, and $(\Pi(lc(v)), \Pi(rc(v)))$ are the marginal conditional probabilities of the left and right child of a node v, respectively. For instance, if v is a node of j-th level, $\Pi(lc(v)) = \mathbf{P}(y_j = 0 \mid x, y_1 = v_1, \ldots, y_{j-1} = v_{j-1})$ and $\Pi(rc(v)) = \mathbf{P}(y_j = 1 \mid x, y_1 = v_1, \ldots, y_{j-1} = v_{j-1})$.

Algorithm 1 Greedy search (left) and Exhaustive search (right)

```
1: function GreedySearch
   Input: CC Model $\{h_j : j = 1, ..., m\}$
   Output: $v = (v_1, ..., v_m)$ with $v_j \in \{0,1\}$
2:     $v \leftarrow v_R$    // the root of the probability tree
3:     while $v$ is not a leaf do
4:         $lc(v), rc(v) \leftarrow$ left and right child of $v$
5:         if $\Pi(lc(v)) \geq \Pi(rc(v))$ then
6:             $v \leftarrow lc(v)$
7:         else
8:             $v \leftarrow rc(v)$
9:     return $v$
```

```
1: function ExhaustiveSearch
   Input: CC Model $\{h_j : j = 1, ..., m\}$
   Output: $v = (v_1, ..., v_m)$ with $v_j \in \{0,1\}$
2:     $Q \leftarrow \{(v_R, \overline{\Pi}(v_R) = 1)\}$    // {(root node, $\overline{\Pi}$(root node))}
3:     $maxP \leftarrow 0$
4:     while $Q \neq \emptyset$ do
5:         $w \leftarrow$ pop an element in $Q$
6:         $lc(w), rc(w) \leftarrow$ left and right child of $w$
7:         compute $\overline{\Pi}(lc(w)), \overline{\Pi}(rc(w))$ recursively from $\overline{\Pi}(w)$
8:         if $lc(w)$ and $rc(w)$ are not leaves then
9:             $Q \leftarrow Q \cup \{(lc(w), \overline{\Pi}(lc(w))), (rc(w), \overline{\Pi}(rc(w)))\}$
10:        else if $max(\overline{\Pi}(lc(w)), \overline{\Pi}(rc(w))) > maxP$ then
11:            $v \leftarrow argmax_{w' \in \{lc(w), rc(w)\}}(\overline{\Pi}(w')$
12:            $maxP \leftarrow max(\overline{\Pi}(lc(w)), \overline{\Pi}(rc(w)))$
13:    return $v$
```

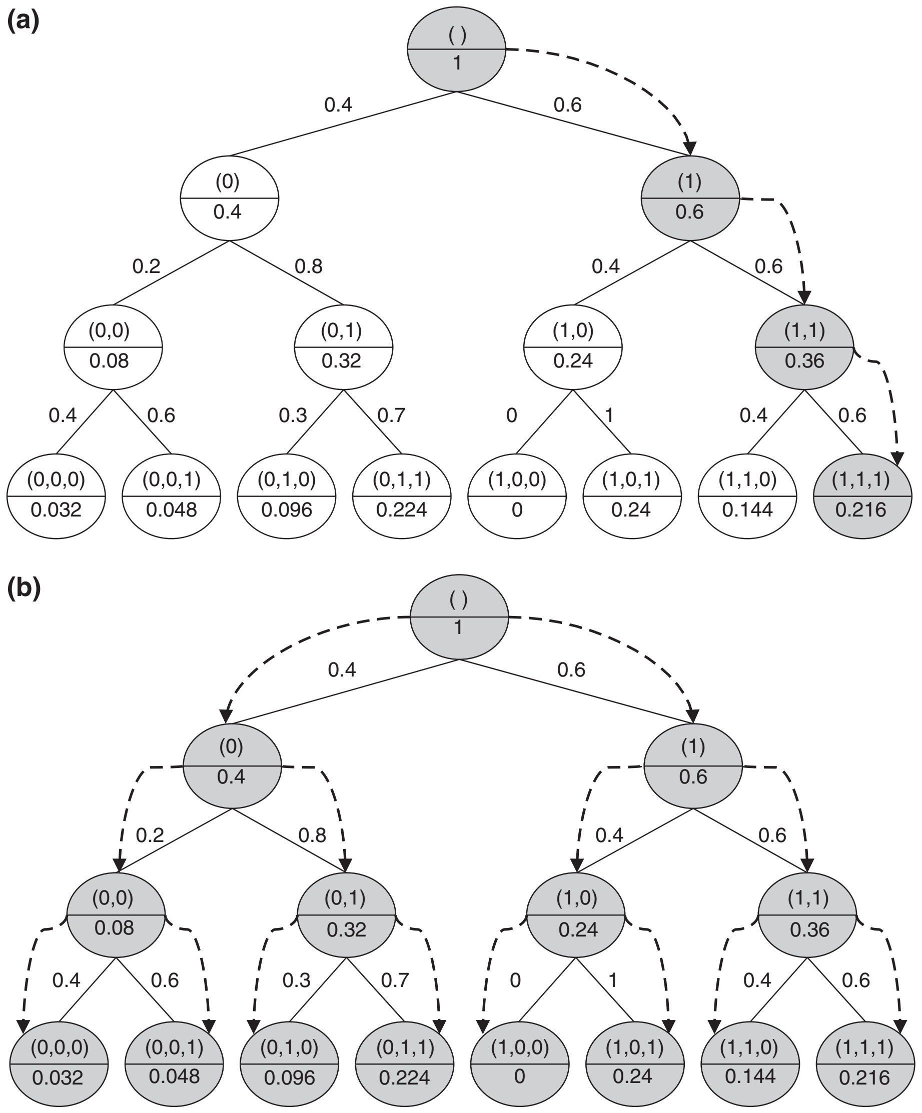


FIGURE 2 | An example of paths followed by an instance using Greedy Search (CC). The dotted arrows show the path followed by the algorithm. (a) Greedy Search (CC) and (b) Exhaustive Search (PCC).

Concerning the optimization of subset 0/1 loss, a rigorous analysis[20] establishes bounds for the performance of GS. For this purpose, the authors define the regret r of a classifier h for a given loss $L(\cdot,\cdot)$ as the difference between the risk of that classifier and the Bayes-optimal classifier h^B, i.e.,

$$r_L(h) = \mathbb{E}_{\mathbf{X},\mathbf{Y}} L(\mathbf{Y}, h(\mathbf{X})) - \mathbb{E}_{\mathbf{X},\mathbf{Y}} L\left(\mathbf{Y}, h^B(\mathbf{X})\right) \qquad (7)$$

In case of the subset 0/1 loss, one can just analyze the expectation over $\mathbf{Y}$ given x, because this loss is decomposable with regard to individual instances. Hence, the regret becomes

$$r_L(h) = \mathbb{E}_{\mathbf{Y}} L(\mathbf{Y}, h(x)) - \mathbb{E}_{\mathbf{Y}} L\left(\mathbf{Y}, h^B(x)\right) \qquad (8)$$

Then, assuming that perfect estimate of the joint conditional probability $\mathbf{P}(y \mid x)$ has been obtained, they establish an upper bound equal to $2^{-1} - 2^{-m}$ for the regret of the classifier h induced by the GS method for the subset 0/1 loss. This bound shows that this method offers quite poor performance for this loss, in other words, the method does not manage to provide a good estimation of the risk minimizer for this loss. However, the work concludes stating that GS tries to optimize the subset 0/1 loss rather than, e.g., the Hamming loss, because it is more suitable for estimating the mode of the joint conditional distribution rather than the mode of the marginal conditional distribution.

Exhaustive Search

Unlike GS that explores only one label combination, ES analyzes all possible paths in the tree, estimating the entire joint conditional distribution $\mathbf{P}(y|x)$ for a new instance x. Hence, it provides Bayes optimal inference. Then, for each $h(x)$, it computes $\mathbf{P}(y \mid x)$ and $L(y, h(x))$ for all combination of labels $y = (y_1, y_2, \ldots, y_m)$ and outputs the combination $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_m) = h^*(x)$ with minimum risk for the given loss $L(\cdot,\cdot)$. By doing so, it generally improves in terms of performance, as 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 $h(x)$.

Figure 2(b) illustrates this approach; all paths are explored and the optimal solution is always reached. Algorithm 1 (right) shows a possible pseudocode for this method. In this case, $\overline{\Pi}(v) = P(y_1 = v_1, \ldots, y_j = v_j | x)$ is recursively obtained by $\overline{\Pi}(v) = \Pi(v) \cdot \overline{\Pi}(pa(v))$ in which v is a node of the j-th level and $pa(v)$ denotes the parent of node v.

ε-Approximate Algorithm

The ε-approximate (ε-A) algorithm[20] 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 $\varepsilon = 2^{-k}$ with $1 \le k \le m$. This marginal joint conditional probability for a node in level j for an unlabeled x is

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

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

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. At the end, two situations can be found: (1) the node expanded is a leaf or (2) there are not more nodes that exceed the threshold. If the former situation occurs, the prediction for the unlabeled instance x will be $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_m)$ corresponding to the combination of the leaf reached (see Figure 3(a)). Conversely, if situation (2) takes place, then GS is applied to the nodes whose children do not exceed the threshold, and the prediction $y = (\hat{y}_1, \ldots, \hat{y}_m)$ for the unlabeled instance x in this case will be that with highest entire joint conditional probability $\mathbf{P}(y_1, \ldots, y_m | x)$ (see Figure 3(b) and (c)). Algorithm 2 entails the pseudocode of this method.

Algorithm 2 Pseudocode of the ϵ−approximate algorithm

```
function $\epsilon$-APPROXIMATE
Input: CC Model $\{h_j : j = 1, \ldots, m\}$, and $\epsilon \ge 0$
Output: $v = (v_1, \ldots, v_m)$ with $v_j \in \{0, 1\}$
    an ordered list $Q \leftarrow \{(v_R, \overline{\Pi}(v_R) = 1)\}$    // $\{$(root node, $\overline{\Pi}$(root node))$\}$
    an ordered list $K \leftarrow \emptyset$    // list of non-survived parents
    repeat
        $v \leftarrow$ pop the first element in $Q$
        if $v$ is not a leaf then
            $lc(v), rc(v) \leftarrow$ left and right child of $v$
            compute $\overline{\Pi}(lc(v))$ and $\overline{\Pi}(rc(v))$ recursively from $\overline{\Pi}(v)$
            if $\overline{\Pi}(lc(v)) \ge \epsilon$ then
                insert $(lc(v), \overline{\Pi}(lc(v)))$ in list $Q$ sorted according to $\overline{\Pi}$
            if $\overline{\Pi}(rc(v)) \ge \epsilon$ then
                insert $(rc(v), \overline{\Pi}(rc(v)))$ in list $Q$ sorted according to $\overline{\Pi}$
            if $lc(v)$ and $rc(v)$ are not inserted in the list $Q$ then
                insert $(v, \overline{\Pi}(v))$ in list $K$ sorted according to $\overline{\Pi}$
    until $v$ is a leaf or $Q = \emptyset$
    if $v$ is not a leaf then
        $\epsilon \leftarrow 0$
        while $K \neq \emptyset$ do
            $w \leftarrow$ pop first element in $K$ and apply GreedySearch to obtain $\overline{\Pi}(w)$
            if $\overline{\Pi}(w) \ge \epsilon$ then
                $v \leftarrow w$ and $\epsilon \leftarrow \overline{\Pi}(w)$
    return $v$
```

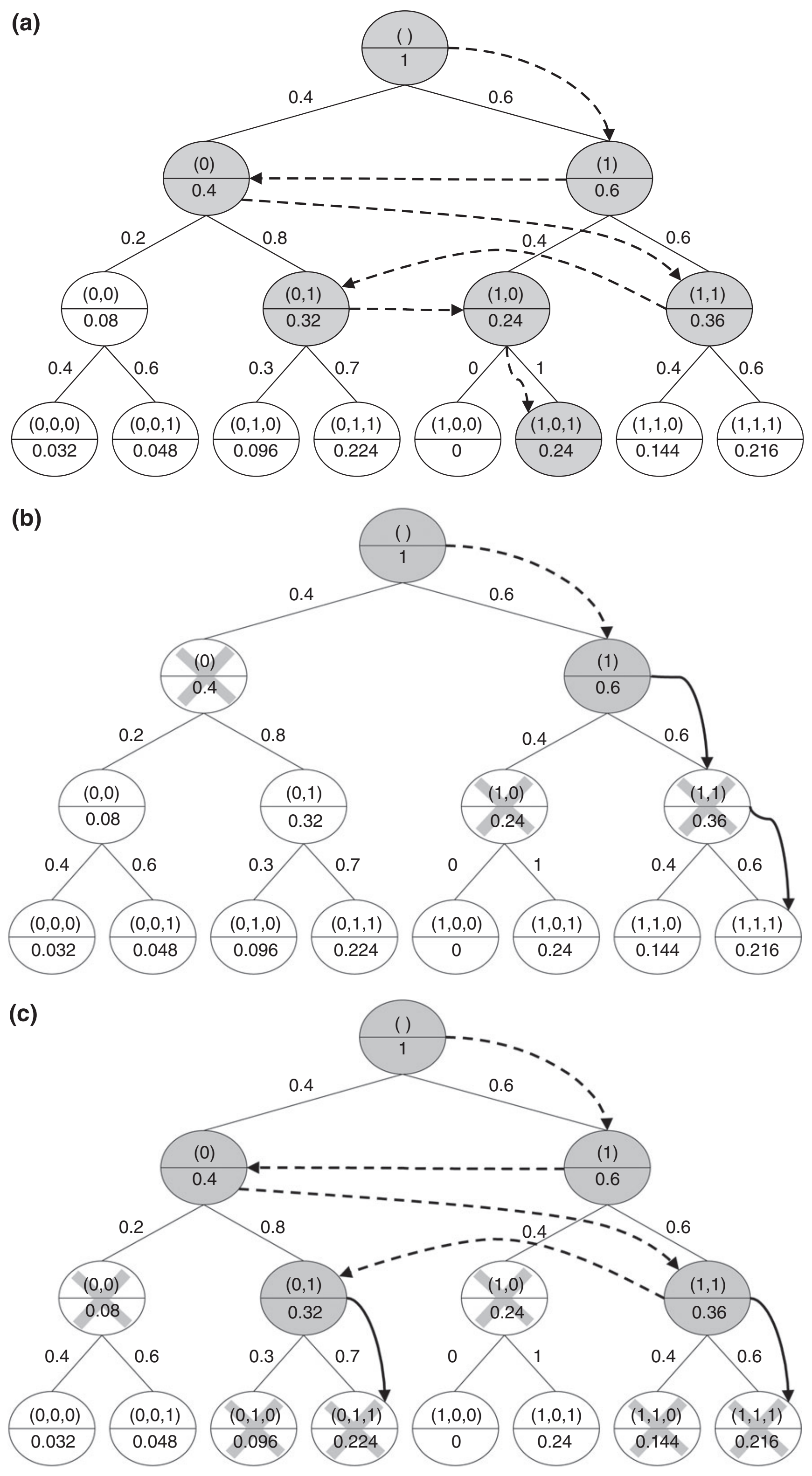


FIGURE 3 | Several examples of the paths followed by ε-A algorithm for different values of ε. The nodes with a cross are those that have a marginal joint conditional probability lower than ε and, hence, they are not explored anymore. The dotted arrows show the path followed by the algorithm. The solid 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. (a) ε-A algorithm with ε=.0(k = m), (b) ε-A algorithm with ε=.5 (k = 1), (c) ε-A algorithm with ε=.25(k = 2).

The parameter ε plays an important role in the algorithm. The particular case of ε=.0 (or any value in the interval $[0, 2^{-m}]$, i.e.,, $k = m$) has special interest, because the algorithm performs a UC search that always finds the optimal solution. This is so because the marginal joint conditional probabilities for at least one leaf is guaranteed to be higher than ε, as any probability is positive (or null in the worst case). Even more, this leaf is an optimal solution, because the marginal joint conditional probabilities dismiss as it goes down on the tree. Figure 3a illustrates this situation. In this case, all nodes are candidates to be explored, because all of them will have a marginal joint conditional probability greater than ε. First, the right node of the first level is explored. Their children have a marginal joint conditional probability (0.24 and 0.36, respectively) lower than their uncle (0.4), so their uncle is explored before them. After that, the child positioned closer to the right of the first node is explored because it has the highest marginal joint conditional probability (0.36) among their brother and cousins. The following node to explore is its right cousin with a probability of 0.32, which is higher than their children (0.144 and 0.216). Then, its brother (0.24) is explored and finally its right nephew 0.24, which is already a leaf and as expected the leaf of the optimal solution.

Conversely, the method is looking to GS as ε grows, being the GS in case of ε=.5 (or equivalently $\varepsilon = 2^{-1}$, i.e.,, $k = 1$). This is so because in this case two situations are possible: (1) only one node has a marginal joint conditional probability greater than ε, in whose case the algorithm follows one path, or (2) none node has a marginal joint conditional probability greater than ε, in whose case a GS is applied from here to the bottom of the tree. Figure 3 (b) shows an example of this particular case. In the first level, only the right node has a marginal joint conditional probability that exceeds the value of ε. However, in the second level, none marginal joint conditional probability of the two nodes is greater than ε. In both cases, a step of GS algorithm is applied. Besides, the path followed leads to a non optimal solution.

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 $\varepsilon = 2^{-k}$ is that the method guarantees to reach a partial optimal solution at least until the k-th level on the tree. Even more, 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 remains 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 k-th level or which can be optimal until levels below the k-th level can be different from the one of the global optimal solution until such levels, as the global optimal solution also depends on what happens hereinafter. At this respect and from $(k + 1)$-th level, if the joint conditional probability of the 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 the optimal solution is lower than ε, then some nodes will be discarded because their marginal joint conditional probability will fall below ε, and hence, it is guarantee that one of these nodes would have lead to the optimal solution. However, if this situation takes place, i.e., if a node is discarded, then the algorithm never reaches a leaf, and hence, GS is applied starting at each discarded node. Therefore, the optimal solution is reached if GS follows the optimal path, which is not guaranteed. Figure 3 (c) shows the particular case of ε=.25. In the figure, some nodes do not exceed the constraint of having their marginal joint conditional probability greater than ε. Even more, none leaf is reached without applying the GS strategy and the optimal solution is not reached.

Consequently, this algorithm estimates the risk minimizer for the subset 0/1 loss a greater or lesser extent depending on the value of ε. Moreover, a theoretical analysis of this estimation[20] allows to bound 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 GS approach, this analysis establishes that this algorithm needs less than $\mathbf{O}(m2^k)$ iterations to find a prediction that allows to upper bound the regret $r_L(h)$ of the classifier h by $2^{-k} - 2^{-m}$ for the subset 0/1 loss and $k \leq m$, again under the assumption that a perfect estimate of the joint conditional probability $P(y \mid x)$ is obtained. As a consequence, if the probability distribution for which the joint mode has a probability mass bigger than 2^{-k}, then, the algorithm needs less than $m2^k$ iterations to find a prediction that corresponds to this mode. Let notice that the particular case of ε=.0 (or $k = m$) makes the bound becomes 0.

Beam Search

Beam Search (BS)[21,22] also explores more than one path in the probabilistic tree. This method includes a parameter b called 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, a certain number of the top levels are exhaustively explored depending on the value of b, particularly a total of $k^* -1$ levels, being k^* the lowest integer such that $b < 2^{k^*}$. Then, only b possibilities are explored for each of the remaining 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 x is the same as for the ε-A algorithm. Hence, such probability is given by Eq. (9). At the end, the algorithm outputs $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_m)$ with the highest entire joint conditional probability $\mathbf{P}(y_1, \ldots, y_m | x)$.

BS differs from GS in that (1) BS explores more than one combination and also in (2) the probability taken for deciding a path to follow in the tree. Concerning (2), both take the marginal joint conditional probability $\mathbf{P}(y_1, \ldots, y_j | x)$, but in the case of GS, that is equivalent to take the marginal conditional probability $\mathbf{P}(y_j, | x, y_1, \ldots, y_{j-1})$ because the two nodes explored in each level of the tree have the same parent as explained before. However, in case of BS, the b nodes explored do not have to have the same parent (except for the root node) even in the case of $b = 2$.

In the case of $b = 1$, BS expands just one node in each level, that with 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 encapsules both GS and ES respectively by 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. The number of nodes expanded by BS is bounded by $\mathbf{O}(bm)$.

As final remark, the fact that BS considers marginal joint conditional probabilities makes the method tend to estimate the risk minimizer for the subset 0/1 loss. In any case, it would be possible to include other loss functions into the search algorithm. At this respect, the authors who proposed BS for inference in PCC in MLC[21,22] do not include any theoretical analysis about the goodness of the estimation of the risk minimizer. However, they empirically show that taking certain values of b ($b < 15$), the risk minimizer provided by the method converges to the one obtained using ES.

Figure 4 shows an example of the paths explored by the BS algorithm when $b = 2$. Hence, all nodes of the first level are explored. From there, just two nodes are explored, those with highest marginal joint conditional probability among the children whose parents have been previously explored. At this respect, notice that the node with the optimal solution is not explored as its parent has not been previously explored. This example confirms that BS cannot guarantee to reach optimal solutions unless $b = 2^m$. The pseudocode of the algorithm is detailed in Algorithm 3.

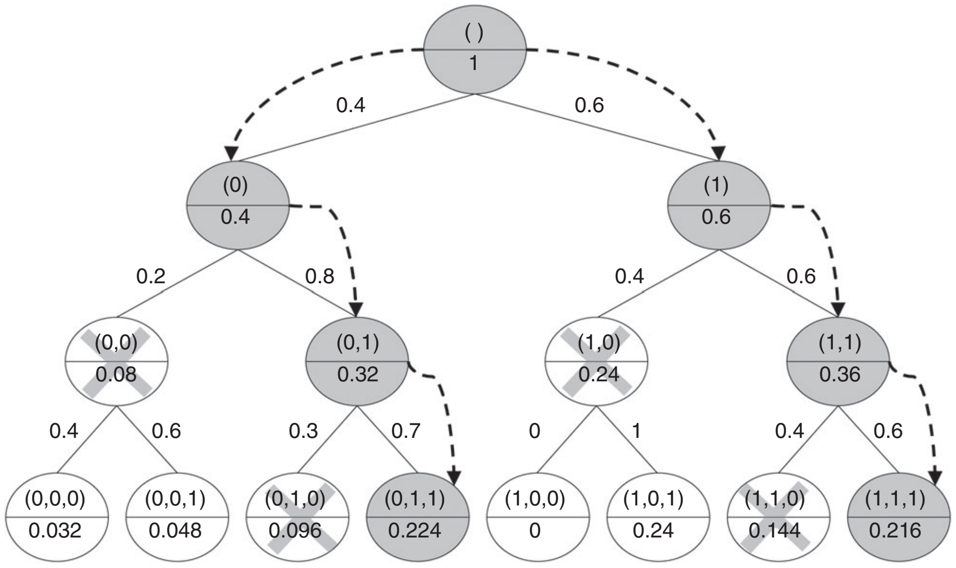


FIGURE 4 | An example of paths followed by an instance using Beam Search (BS) with $b = 2$. The cross out nodes with a cross mean that this node has not any of the highest marginal joint conditional probabilities and, hence, it is not explored anymore. The dotted arrows show the path followed by the algorithm.

Algorithm 3 Beam Search

```
1:  function BEAMSEARCH
    Input: CC Model $\{h_j : j = 1, ..., m\}$, and $b \geq 1$    // $b$ is the beam width
    Output: $v = (v_1, ..., v_m)$ with $v_j \in \{0, 1\}$
2:    $B^{(0)} \leftarrow \{(v_R, \overline{\Pi}(v_R) = 1)\}$    // $\{(\text{root node}, \overline{\Pi}(\text{root node}))\}$
3:    for $j = 1, \ldots, m$ do
4:      $B^{(j)} \leftarrow \emptyset$
5:      for $w \in B^{(j-1)}$ do
6:        $lc(w), rc(w) \leftarrow$ left and right child of $w$
7:        compute $\overline{\Pi}(lc(w))$ and $\overline{\Pi}(rc(w))$ recursively from $\overline{\Pi}(w)$
8:        if $\overline{\Pi}(lc(w)) > \overline{\Pi}(last(B^j))$ then
9:          insert $(lc(w), \overline{\Pi}(lc(w)))$ in $B^{(j)}$ sorted according to $\overline{\Pi}$ and $B^{(j)} \leftarrow Top_b(B^{(j)})$
10:       if $\overline{\Pi}(rc(w)) > \overline{\Pi}(last(B^j))$ then
11:         insert $(rc(w), \overline{\Pi}(rc(w)))$ in $B^{(j)}$ sorted according to $\overline{\Pi}$ and $B^{(j)} \leftarrow Top_b(B^{(j)})$
12:   return $v \leftarrow Top(B^{(m)})$
```

Monte Carlo Sampling

Monte Carlo sampling (MS) is a technique based on repeating random sampling in order to obtain the distribution of an unknown probabilistic distribution. There are several ways of implementing a Monte Carlo method, but they tend to follow a particular pattern: (1) they define a domain of possible values, (2) they generate values randomly from a probability distribution over the domain, (3) they perform a deterministic computation on the values, and finally, (4) they aggregate the results.

Regarding PCC for MLC, two different Monte Carlo algorithms were recently proposed.[20,23] Both use the domain of the 0/1 vectors of dimension m. In both approaches, the random values are drawn using each classifier h_j, which estimates the probability $\mathbf{P}(y_j | x, y_1, \ldots, y_{j-1})$ of being the label ℓ_j relevant for a new unlabeled instance x. Hence, both algorithms take the previously obtained $\hat{y}_1^{(i)}, \ldots, \hat{y}_{j-1}^{(i)}$ in the chain, for predicting $\hat{y}_j^{(i)}$ in iteration i. Hence, the conditional probability $\mathbf{P}\left(y_j | x, \hat{y}_1^{(i)}, \ldots, \hat{y}_{j-1}^{(i)}\right)$ estimated by $h_j\left(x, \hat{y}_1^{(i)}, \ldots, \hat{y}_{j-1}^{(i)}\right)$ is calculated when the random process takes place for the label ℓ_j. The difference between Monte Carlo approaches and GS, e.g., is that, in the latter the prediction $\hat{y}_j$ for ℓ_j is directly obtained from the evaluation of h_j, whereas in the former such prediction is obtained after a random process drawn using the distribution induced by h_j. This difference makes it possible to repeat the process a certain number of times, producing different predictions, although some of them may be repeated several times. Algorithm 4 shows the pseudocode of both approaches. The key element is the process to draw observations of the conditional distribution (line 4 in the pseudocode). Figure 5 depicts an example of one observation. In each node v, a random number is generated to decide whether the next label is relevant or not. If the random number is lower or equal to $\Pi(lc(v))$ then the left child is selected and the next label is irrelevant; otherwise the label is relevant. The process moves down in the tree according to this rule until a leaf is reached, obtaining a new observation. Notice that this implies that the number of nodes expanded by Monte Carlo approaches is always mq, being q the number of observations of the sample.

Algorithm 4 Pseudocode of Monte Carlo Sampling (left) and Efficient Monte Carlo Sampling (right)

```
1:  function MONTECARLOSAMPLING
    Input: CC Model $\{h_j : j = 1, ..., m\}$, and $q$    // $q$ sample size
    Output: $v = (v_1, ..., v_m)$ with $v_j \in \{0, 1\}$
2:    $Q = \emptyset$
3:    for $t = 1, \ldots, q$ do
4:      draw $w$ according to $\Pi$
5:      insert $w$ in $Q$
6:    return $v \leftarrow$ mode of $Q$
```

```
1:  function EFFICIENTMONTECARLOSAMPLING
    Input: CC Model $\{h_j : j = 1, ..., m\}$, and $q$    // $q$ sample size
    Output: $v = (v_1, ..., v_m)$ with $v_j \in \{0, 1\}$
2:    $\overline{\Pi}(u_0) \leftarrow 0$
3:    for $t = 1, \ldots, q$ do
4:      draw $w$ according to $\Pi$
5:      if $\overline{\Pi}(w) > \overline{\Pi}(u_{t-1})$ then
6:        $u_t \leftarrow w$
7:      else
8:        $u_t \leftarrow u_{t-1}$
9:    return $v \leftarrow u_q$
```

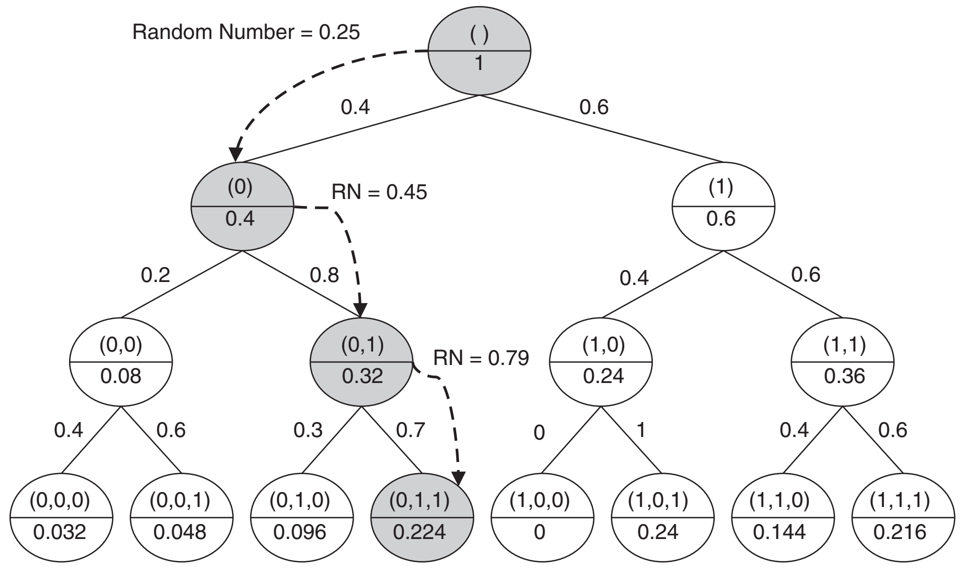


FIGURE 5 | An example of one observation drawn using Monte Carlo sampling. In each node v, a biased coin is flipped to decide whether the label is relevant or not. The probability of tails and heads are given, respectively, by the weights $\pi(lc(v))$ and $\pi(rc(v))$ of the left and right child of a node v. If the random number is lower or equal than $\pi(lc(v))$ then the left child is selected and the label is irrelevant; the right child is selected otherwise. One observation of the conditional distribution is obtained when a leaf is reached.

Finally, different aggregation approaches of the predictions obtained through the iteration procedure can be performed depending on the target loss to be optimized. The risk minimizer in this case is estimated over the subset of $\mathcal{Y}$ formed by the predictions obtained in the iterations instead of over the whole $\mathcal{Y}$. Hence, the estimation can be poorer than in case of ES, and, hence the optimal solution is not guarantee to be reached. However, the fact of using the distribution induced by h_j in the random process guarantees that the aggregated final prediction $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_m)$ of these Monte Carlo methods converges to the risk minimizer when the sample drawn is large enough and when an adequate aggregation procedure according to the loss to be minimized is taken. It is in this point where the two Monte Carlo approaches differ. The first one proposed (we will refer to it as MC)[20] takes the most frequent combination of $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_m)$, i.e., the mode. This approach not only has been used for performing inference in classifier chains, otherwise it was also used for performing inference in classifier trellisers.[29] Conversely, the most recent proposal (we will refer to it as EMC, efficient MC)[23] takes the combination with the highest joint conditional probability. Their authors tag this approach as efficient, because this aggregation procedure allows the method to converge faster. Besides, it is not necessary to store all the combinations, consuming less memory. In any case, both tend to optimize the subset 0/1 loss, despite it is not possible to guarantee to reach it.

EXPERIMENTS

The experiments were performed over several benchmark multilabel 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 a hundred, even one of them has almost four hundred labels. The approaches for inference in PCC compared were those discussed along the paper,

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

except ES. No experiment was carried out with the ES method due to its computational cost. Hence, the methods compared were GS, ε-A algorithm for different values of ε (.0,.25,.5), BS for different values of the beam width b (1, 2, 3, 10) and the MS approaches, MC and EMC, with samples of different size (10, 50, 100), as suggested in Ref 23. Let us remember that the ε-A algorithm with ε=.5 is equivalent to GS and to BS with b = 1.

The results are presented in terms of the example-based subset 0/1 loss estimated by means of a 10-fold cross-validation. The base learner employed to obtain the binary classifiers that compose all these multilabel models was *logistic regression*[30] 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 twofold cross validation repeated five times. The Brier loss[31] is a proper score that measures the accuracy of probabilistic predictions. The expression is $\frac{1}{n}\sum_{i=1}^{n}\left(\hat{p}_i - a_i\right)^2$, where for an instance i, p_i is the predicted probability that label i is relevant, and a_i is the actual label value (0 or 1).

Tables 2–4 respectively show the subset 0/1 loss, the number of nodes explored, and the averaged computational time (in seconds) per test instance for the different methods compared. The number of nodes explored and the computational time for the Monte Carlo approaches should be the same as those of GS multiplied by the size of the sample drawn. Maybe slight differences can occur in time due to initialization or other implementation issues.

Before discussing the results of the tables, let us remember that only ε-A algorithm with ε=.0 provides Bayes optimal inference, like ES does. This means that it always predicts the label combination with the highest joint conditional probability. Despite other methods may predict other label combination with lower joint conditional probability for some examples, they obtain better subset 0/1 scores in some few cases. This fact may be due to several reasons, mainly: (1) the relatively small size of the testing sets, and (2) the models h_j obtained to estimate the joint conditional probability $P(y \mid x)$ do not return true estimations usually. Theoretically, under perfect conditions (large test sets and perfect models), ε-A with ε=.0 would obtain the best scores.

Then, looking at Table 2 and as it is theoretically expected, the performance of ε-A algorithm decreases as the value of ε increases. As commented before, some exceptions can occur. In this case, it happens for flag and reuters datasets.

Concerning BS method, the performance increases as b increases (as theoretically expected), although some exceptions appear in bibtex, reuters, slashdot and yeast, due to the same reasons discussed above. Values from 1 to 3 were taken into account, because the performance gets steady when b = 3 for most of the datasets. Even more, it reaches steadiness when b = 3 for the half of the datasets and when b = 2 for both image and medical. In any case, a value equal to 10 was also considered to show some cases in which the predictions of the algorithm

TABLE 2 | Subset 0/1 Loss for the Different Methods

Datasets	ES ε-A (.0)	ε-A (.25)	GS BS(1) ε-A(.5)	BS(2)	BS(3)	BS (10)	MC (10)	MC (50)	MC (100)	EMC (10)	EMC (50)	EMC (100)
bibtex	81.92	81.95	82.19	**81.88**	**81.92**	**81.92**	**84.02**	82.85	82.46	82.73	82.22	82.11
corel5k	97.48	98.62	98.90	98.30	98.04	**97.48**	**99.64**	98.98	98.26	98.72	97.70	**97.42**
emotions	71.16	71.82	72.83	72.16	71.32	**71.16**	**80.77**	73.68	73.84	72.83	72.33	72.50
enron	83.14	84.26	85.43	83.43	83.37	**83.14**	**92.95**	85.90	84.61	87.43	83.61	83.26
flags	87.13	87.16	**86.13**	88.21	**87.13**	**87.13**	**96.39**	91.21	89.24	91.29	90.71	90.18
image	68.35	**68.35**	69.75	**68.35**	**68.35**	**68.35**	**69.20**	**65.25**	**62.65**	**64.95**	**65.70**	**62.95**
mediamill*	83.86	84.58	85.80	84.10	**83.86**	**83.86**	**90.90**	85.88	84.88	85.34	**83.70**	**83.40**
medical	30.37	**30.37**	**30.67**	**30.37**	**30.37**	**30.37**	**31.19**	**30.16**	**30.67**	**30.16**	**30.16**	**30.16**
reuters	22.73	**22.70**	23.60	**22.69**	22.73	**22.73**	**25.37**	23.18	22.83	22.97	22.83	22.83
scene	31.86	**31.86**	33.28	31.90	**31.86**	**31.86**	**33.53**	**30.28**	**29.70**	**29.24**	**29.24**	**29.29**
slashdot	51.80	52.22	54.49	**51.77**	**51.80**	**51.80**	**56.45**	52.96	52.70	52.22	52.35	52.35
yeast	76.95	77.62	79.77	**76.83**	**77.08**	**76.95**	**85.52**	79.93	79.35	79.06	77.53	77.62

Those scores that are equal or better than optimal predictions reached by ε-A and ES are shown in bold.

TABLE 3 | Number of Explored Nodes for the Different Methods

Datasets	ε-A(.0)	ε-A(.25)	GS BS(1) ε-A(.5)	BS(2)	BS(3)	BS(10)	MC/EMC(10)	MC/EMC(50)	MC/EMC(100)
bibtex	289.3	184.0	160.0	319.0	477.0	1575.0	1590.0	7950.0	15900.0
corel5k	1474.2	517.1	375.0	749.0	1122.0	3725.0	3740.0	18700.0	37400.0
emotions	10.7	10.8	7.0	13.0	18.0	45.0	60.0	300.0	600.0
enron	114.8	77.3	54.0	107.0	159.0	515.0	530.0	2650.0	5300.0
flags	22.6	16.3	8.0	15.0	21.0	55.0	70.0	350.0	700.0
image	7.3	7.7	6.0	11.0	15.0	35.0	50.0	250.0	500.0
mediamill*	191.8	142.4	102.0	203.0	303.0	995.0	1010.0	5050.0	10100.0
medical	46.6	46.6	46.0	91.0	135.0	435.0	450.0	2250.0	4500.0
reuters	8.2	8.3	8.0	15.0	21.0	55.0	70.0	350.0	700.0
scene	7.2	7.3	7.0	13.0	18.0	45.0	60.0	300.0	600.0
slashdot	25.3	24.9	23.0	45.0	66.0	205.0	220.0	1100.0	2200.0
yeast	26.1	26.0	15.0	29.0	42.0	125.0	140.0	700.0	1400.0

TABLE 4 | Average Prediction Time (in Seconds) Per Example for All Methods

Datasets	ES ε-A(.0)	ε-A (.25)	GS BS (1) ε-A (.5)	BS(2)	BS(3)	BS(10)	MC (10)	MC (50)	MC (100)	EMC (10)	EMC (50)	EMC (100)
bibtex	0.0162	0.0099	0.0079	0.0110	0.0156	0.0507	0.3220	1.6099	3.2256	0.3207	1.6021	3.2065
corel5k	0.1360	0.0161	0.0110	0.0278	0.0404	0.1361	0.7607	3.8076	7.6294	0.7563	3.7815	7.5781
emotions	0.0006	0.0006	0.0004	0.0005	0.0006	0.0013	0.0120	0.0590	0.1177	0.0117	0.0585	0.1172
enron	0.0072	0.0030	0.0019	0.0039	0.0055	0.0172	0.1046	0.5227	1.0457	0.1039	0.5190	1.0388
flags	0.0012	0.0007	0.0004	0.0005	0.0007	0.0016	0.0139	0.0691	0.1381	0.0136	0.0680	0.1360
image	0.0005	0.0005	0.0004	0.0004	0.0005	0.0009	0.0100	0.0491	0.0978	0.0098	0.0485	0.0970
mediamill*	0.0115	0.0055	0.0037	0.0072	0.0105	0.0333	0.1992	0.9963	1.9937	0.1984	0.9910	1.9821
medical	0.0027	0.0027	0.0027	0.0033	0.0046	0.0144	0.0886	0.4422	0.8857	0.0881	0.4404	0.8823
reuters	0.0005	0.0005	0.0005	0.0005	0.0007	0.0016	0.0138	0.0684	0.1373	0.0137	0.0682	0.1368
scene	0.0005	0.0005	0.0004	0.0005	0.0006	0.0013	0.0119	0.0587	0.1172	0.0118	0.0584	0.1171
slashdot	0.0015	0.0014	0.0012	0.0016	0.0022	0.0063	0.0433	0.2159	0.4317	0.0429	0.2149	0.4296
yeast	0.0015	0.0010	0.0006	0.0010	0.0014	0.0040	0.0277	0.1380	0.2759	0.0274	0.1369	0.2737

converge to those of ε-A with ε=.0 or ES. Notice that this is at the cost of exploring much more solutions.

Both Monte Carlo approaches improve their performance as the size of the sample increases. However, there are some exceptions. In this sense, it happens that MC reaches slightly better subset 0/1 loss for a sample size of 50 than for a sample size of 100 in case of emotions and medical. Unfortunately, EMC has more exceptions, but this is quite logical, because EMC is more sensitive to the size of the sample. This is so because it takes the maximum joint conditional probability, which is more likely to change as the sample size increases. In case of MC, it would be necessary to enlarge enough the sample size to make the mode change. Comparing both approaches, it is clear that EMC converges faster than MC, because the subset 0/1 of EMC is quite lower than that of MC for the same size of the sample drawn, especially in case of size equal to 10. Even more, MC with a sample of size equal to 50 is hardly

TABLE 5 | Subset 0/1 Loss (With the Standard Deviation) for the Compared Methods Considering a Sample of Different Label Orders

Datasets	ε-A(.0)	ε-A(.25)	ε-A(.5)	BS(2)	MC(10)	EMC(10)
image (24)	**64.44 ± 2.62**	65.02 ± 2.45	66.69 ± 1.93	64.61 ± 2.72	71.05 ± 1.97	66.03 ± 2.69
emotions (144)	**71.55 ± 1.06**	71.71 ± 1.00	73.16 ± 1.17	71.75 ± 1.02	78.81 ± 1.38	72.36 ± 1.07
scene (144)	32.09 ± 1.26	32.11 ± 1.27	34.35 ± 1.68	32.09 ± 1.26	35.63 ± 1.50	**32.00 ± 1.30**
reuters (500)	**22.78 ± 0.29**	**22.78 ± 0.29**	23.69 ± 0.25	**22.78 ± 0.29**	25.29 ± 0.33	22.87 ± 0.28

The number of label orders are in parentheses and represent the 20% of the possible label orders, except in the case of reuters dataset which represent the 10%.

able to reach EMC with a sample of size equal to 10. Also, EMC with a sample size of 50 outperforms all MC for most of the datasets. Just flags and image are exceptions. EMC is an appealing approach to perform inference in PCC, however, sometimes, even with a large enough sample, EMC is able to reach the precision of ε –A with ε=.0 in some datasets.

With regard to the number of nodes explored (see Table 3), GS (equivalent to ε-A algorithm with ε=.5 and to BS(1)) is the method which explores the smallest number of nodes, as it only goes over one path in the tree. In fact, such number corresponds to the number of labels plus one, because the root of the tree is considered as an explored node. It follows the ε-A algorithm with ε=.25, because the BS(b) with b from 2 rapidly increases the number of nodes explored. However, let us remember that none of those methods guarantee to reach an optimal solution. Then, focusing on the method that theoretically reaches the optimum (the ε-A algorithm with ε=.0), it occurs that this particular case of the ε-A algorithm explores the greatest amount of nodes among the same algorithm with other values for ε, especially as the number of labels grows. However, the ε-A algorithm clearly outperforms the BS technique in number of nodes explored, even for ε=.0 and for low values of the beam width b. Regarding EMC and MC, the number of nodes considerably increases as the size of the sample drawn increases. Hence, they require to explore much more nodes to be closer to the optimal, despite sometimes their performance could be better.

The computational time is expected to be higher as more nodes are explored, and looking at Table 4 one can confirm that this is what indeed happens. At this respect, ε-A algorithm is actually quite faster than BS technique and MS. In the same way, considering each method separately and varying their parameters is shown that ε-A algorithm is faster as ε increases, whereas BS is faster as the beam width b dismisses. However, this is not surprising, because increasing the value of ε or dismissing the value of the beam width b means to explore less nodes of the tree. Analogously, the time needed for the Monte Carlo approaches enlarge as the size of the sample drawn grows.

Additional experiments were performed in order to analyze the possible influence of taking different label orders. Table 5 contains the average of the subset 0/1 scores for a set of random label orders. Particularly, it was taken 20% of the possible label orders for the datasets with five or six labels (24 and 144 different label orders, respectively) and 10% for reuters dataset that has seven labels (500 different label orders). The results confirm that ε-A algorithm obtains the best scores. The only exception is the scene dataset in which EMC performs slightly better.

As a conclusion, the ε-A algorithm can be considered the best alternative if one desires to guarantee good performance in terms of subset 0/1, taking care of not exceeding the number of nodes explored and the execution time, even in the case of taking ε=.0 for which an optimal solution is guaranteed to reach.

CONCLUSIONS

This study analyzes the methods that have been proposed so far for performing inference in probabilistic classifiers chains for MLC. The ε-approximate algorithm with ε=.0 is theoretically shown to reach an optimal solution in terms of subset 0/1 loss, unlike other approaches that only estimate it, like the method based on a BS, MS or even the same algorithm with $\varepsilon > 0$. Besides, it offers good behavior both in number of nodes explored and in computational time. Even, the ε-approximate algorithm with values of ε greater than .0 offers good behavior, because it considerably reduces both the number of nodes explored and execution time with regard to either beam search or MS, keeping similar performance in terms of subset 0/1. Although beam search and MS seem worse to optimize subset 0/1 loss, they offer some interesting benefits. First, these methods can be adapted to optimize other loss functions. Additionally, MS succeeds in discovering efficiently good label orders.

ACKNOWLEDGMENTS

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

REFERENCES

1. Gibaja E, Ventura S. Multi-label learning: a review of the state of the art and ongoing research. *Wiley Interdiscip Rev Data Mining Knowl Discov* 2014, 4:411–444.
2. Clare A, King RD. Knowledge discovery in multi-label phenotype data. In: *European Conference on Data Mining and Knowledge Discovery (2001)*, Freiburg, Germany, 2001, 42–53.
3. Zhang M-L, Zhou Z-H. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Trans Knowl Data Eng* 2006, 18:1338–1351.
4. Elisseeff A, Weston J. A Kernel method for multi-labelled classification. In: *Advances in Neural Information Processing Systems (NIPS 2001)* Vancouver, Canada. 2001, 681–687.
5. McCallum AK. Multi-label text classification with a mixture model trained by EM. In: *AAAI 99 Workshop on Text Learning*, Orlando, Florida, 1999.
6. Ghamrawi N, McCallum A. Collective multi-label classification. In: *ACM International Conference on Information and Knowledge Management*, Bremen, Germany, 2005, 195–200. New York, NY, USA: ACM.
7. Schapire RE, Singer Y. Boostexter: a boosting-based system for text categorization. *Mach Learn* 2000, 39:135–168.
8. Dembczyński K, Waegeman W, Cheng W, Hüllermeier E. On label dependence and loss minimization in multi-label classification. *Mach Learn* 2012, 88:5–45.
9. Dembczyński K, Cheng W, Hüllermeier E. Bayes optimal multilabel classification via probabilistic classifier chains. In: *ICML (2010)*, Haifa, Israel, 2010, 279–286.
10. Montañés E, Quevedo JR, del Coz JJ. Aggregating independent and dependent models to learn multi-label classifiers. In: *ECML/PKDD'11—Volume Part II*, Athens, Greece, 2011, 484–500. Berlin Heidelberg: Springer-Verlag.
11. Montañés E, Senge R, Barranquero J, Quevedo JR, del Coz JJ, Hüllermeier E. Dependent binary relevance models for multi-label classification. *Pattern Recogn* 2014, 47:1494–1508.
12. Read J, Pfahringer B, Holmes G, Frank E. Classifier chains for multi-label classification. *Mach Learn* 2011, 85:333–359.
13. Tsoumakas G, Katakis I, Vlahavas I. Mining multi-label data. In: *Data Mining and Knowledge Discovery Handbook*. New York, US: Springer; 2010, 667–685.
14. Cheng W, Hüllermeier E. Combining instance-based learning and logistic regression for multilabel classification. *Mach Learn* 2009, 76:211–225.
15. Godbole S, Sarawagi S. Discriminative methods for multi-labeled classification. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining (2004)*, Sydney, Australia, 2004, 22–30.
16. Fürnkranz J, Hüllermeier E, Loza Menca E, Brinker K. Multilabel classification via calibrated label ranking. *Mach Learn* 2008, 73:133–153.
17. Qi GJ, Hua XS, Rui Y, Tang J, Mei T, Zhang HJ. Correlative multi-label video annotation. In: *Proceedings of the International Conference on Multimedia*, Augsburg, Germany, 2007, 17–26. New York: ACM.
18. Read J, Pfahringer B, Holmes G. Multi-label classification using ensembles of pruned sets. In: *IEEE International Conference on Data Mining*, Pisa, Italy, 2008, 995–1000.
19. Tsoumakas G, Vlahavas I. Random k-labelsets: an ensemble method for multilabel classification. In: *ECML/PKDD'07*, LNCS, Warsaw, Poland, 2007, 406–417. Berlin Heidelberg: Springer.
20. Dembczynski K, Waegeman W, Hüllermeier E. An analysis of chaining in multi-label classification. In: Raedt LD, Bessière C, Dubois D, Doherty P, Frasconi P, Heintz F, Lucas PJF, eds. *ECAI: Frontiers in Artificial Intelligence and Applications*, Montpellier, France, vol. 242. Amsterdam, Netherlands: IOS Press; 2012, 294–299.
21. Kumar A, Vembu S, Menon AK, Elkan C. Learning and inference in probabilistic classifier chains with beam search. In: *ECML/PKDD (2012)*, Bristol, UK, 2012, 665–680.
22. Kumar A, Vembu S, Menon AK, Elkan C. Beam search algorithms for multilabel learning. *Mach Learn* 2013, 92:65–89.
23. Read J, Martino L, Luengo D. Efficient monte carlo methods for multi-dimensional learning with classifier chains. *Pattern Recogn* 2014, 47:1535–1546.
24. Read J, Martino L, Olmos PM, Luengo D. Scalable multi-output label prediction: from classifier chains to classifier trellises. *Pattern Recogn* 2015, 48:2096–2109.

25. Luaces Ó, Dez J, Barranquero J, del Coz JJ, Bahamonde A. Binary relevance efficacy for multilabel classification. *Prog Artif Intell* 2012, 4:303–313.

26. Read J, Pfahringer B, Holmes G, Frank E. Classifier chains for multi-label classification. In: *ECML/PKDD'09*, LNCS, Bled, Slovenia, 2009, 254–269. Berlin Heidelberg: Springer.

27. Senge R, del Coz JJ, 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)*, Hildesheim, Germany, 2012.

28. Senge R, del Coz JJ, Hüllermeier E. Rectifying classifier chains for multi-label classification. In: *LWA 2013: Lernen, Wissen & Adaptivität, Workshop Proceedings Bamberg*, Bamberg, Germany, 2013, 151–158.

29. Read J, Martino L, Olmos PM, Luengo D. Scalable multi-output label prediction: from classifier chains to classifier trellises. *Pattern Recogn* 2015, 48:2096–2109.

30. Lin C-J, Weng RC, Keerthi SS. Trust region Newton method for logistic regression. *J Mach Learn Res* 2008, 9:627–650.

31. Brier GW. Verification of forecasts expressed in terms of probability. *Mon Weather Rev* 1950, 78:1–3.