Extremely Randomized CNets for Multi-label Classification

Teresa M.A. Basile[2,3](✉), Nicola Di Mauro[1], and Floriana Esposito[1]
teresamaria.basile@uniba.it

[1] Department of Computer Science, University of Bari "Aldo Moro", Bari, Italy
[2] Department of Physics, University of Bari "Aldo Moro", Bari, Italy
[3] National Institute for Nuclear Physics (INFN), Bari Division, Bari, Italy

Abstract. Multi-label classification (MLC) is a challenging task in machine learning consisting in the prediction of multiple labels associated with a single instance. Promising approaches for MLC are those able to capture label dependencies by learning a single probabilistic model—differently from other competitive approaches requiring to learn many models. The model is then exploited to compute the most probable label configuration given the observed attributes. Cutset Networks (CNets) are density estimators leveraging context-specific independencies providing exact inference in polynomial time. The recently introduced Extremely Randomized CNets (XCNets) reduce the structure learning complexity making able to learn ensembles of XCNets outperforming state-of-the-art density estimators. In this paper we employ XCNets for MLC by exploiting efficient Most Probable Explanations (MPE). An experimental evaluation on real-world datasets shows how the proposed approach is competitive w.r.t. other sophisticated methods for MLC.

Keywords: Multi-label classification; Cutset Networks; Tractable Probabilistic Models.

1 Introduction

Many real world classification problems, such as image and video annotation, functional genomics in bioinformatics, text categorization, and others [16], involve multiple label classes. Multi-Label Classification (MLC) aims at learning a mapping from an instance to a set of relevant labels. A quite common approach to MLC is to adopt a problem transformation technique—the multi-label problem is transformed into one or more single-label problems. *Binary relevance* (BR) [2,25] is a popular problem transformation method that decomposes the MLC problem into a set of single label classification problems, learning the classifiers independently, thus possibly losing the dependencies among the label variables. On the contrary, as reported in [8], it is well known that exploiting the label dependencies can significantly improve the classification performance. For instance, the *classifier chain* approach (CC) [22] exploits potential label correlations by transforming a MLC problem into a chain of binary classification problems—subsequent binary classifiers in the chain are built upon the predictions of preceding ones.

Density estimators like Probabilistic Graphical Models (PGMs) [13], such as Bayesian Networks (BNs), represent a powerful formalism to model and reason about MLC problems, since they are able to capture the conditional independence assumptions among random variables (RVs) into a graph-based representation. Inference routines in PGMs, such as conditional probability inference and Most Probable Explanation (MPE) inference can be exploited to solve MLC [1,6]. In a MLC scenario, we assume to have a set of N training instances $\mathcal{D} = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^{N}$, where each instance has a component $\mathbf{x}^i$ as a vector of M feature values $x_k^i \in \mathbb{R}$. The set $\mathcal{L} = \{y_1, \dots, y_L\}$ denotes the output domain of possible labels. Each vector $\mathbf{x}^i$ is associated to a subset $\mathcal{Y}_i \subseteq \mathcal{L}$ of these labels, represented by an vector of L binary values $\mathbf{y}^i = [y_1^i, \dots, y_L^i]$. The instances are assumed to to be i.i.d. according to a probability distribution $P(\mathbf{X}, \mathbf{Y})$. A common way for probabilistic classifiers to tackle the MLC problem is to learn a model h able to compute a prediction $\hat{\mathbf{y}} = [\hat{y}_1, \dots, \hat{y}_L] = h(\hat{\mathbf{x}})$ for a given attribute instance $\hat{\mathbf{x}}$, usually solved by computing the MPE assignment for the $\mathbf{Y}$—solving $\hat{\mathbf{y}} = \mathrm{argmax}_{\mathbf{y} \in \{0,1\}^L} P(\mathbf{y}|\mathbf{x}) = \mathrm{argmax}_{\mathbf{y} \in \{0,1\}^L} P(\mathbf{y}, \mathbf{x})$, where the $\mathbf{X}$ is assumed to be observed as evidence.

However, learning and inference with PGMs can be challenging—computing exact inference is an NP-Hard problem and some approximate inference routines can be intractable in practice [23]. The need for exact and efficient inference procedures has lead to the introduction of *Tractable Probabilistic Models* (TPMs). An example of TPMs are tractable PGMs, such as mixture of tree distributions [17], trading off expressiveness in exchange of tractable inference. Recent TPMs include, among the others, Sum-Product Networks (SPNs) [19]: deep architectures encoding probability distributions by layering hidden variables as mixtures of independent components. Similarly, *Cutset Networks* (CNets), a particular kind of SPNs, have been recently proposed as easy-to-learn TPMs [21]. CNets are weighted probabilistic model trees in the form of OR-trees having tree-structured probabilistic models as leaves, and positive weights on inner edges. Inner nodes—conditioning OR nodes—are associated to random variables and outgoing branches represent conditioning on the values for those variables. Structure learning algorithmic variants for CNets have been proven to be both accurate and scalable [11,9].

Recently, both SPNs [14] and *Restricted CNets* (RCNets) [12] have been successfully applied to solve MLC problems obtaining state-of-the-art results—learning a single model and exploiting its (exact) tractable inference routines to obtain the correct label predictions. In order to increase the predictive accuracy one can leverage the very well know statistical tool for robust parameter estimation: *bagging*. However, learning an optimal single CNet is a costly operation. *Extremely Randomized CNets* (XCNets) have been recently introduced in [10], as CNets that can be learned in a simple, fast and yet effective approach by performing random conditioning to grow the OR tree. While the likelihood of a single XCNet is not greater than an optimally learned CNet, ensembles of XCNets outperformed state-of-the-art density estimators.

In this work we show *Extremely Randomized Restricted CNets* (XRCNets) combining RCNets and XCNets for solving MLC problems. In particular we focus on learning *ensembles of random RCNets* (XRCNets), and then we prove them to be very competitive against more sophisticated approaches like RAkEL [24], CC [22] and SPNs based. In a thorough empirical comparison on many real-world benchmark datasets, we show our model effectiveness under commonly used metrics for MLC, like accuracy, Hamming and exact match scores.

2 Cutset Networks

Let RVs be denoted by upper-case letters, e.g. X, and their values as the corresponding lower-case letters, e.g. x. Let set of RVs be denoted by $\mathbf{X}$ and their values as $\mathbf{x}$. Given a set of RVs $\mathbf{X}$, $\mathbf{X}_{\setminus i}$ will denote $\mathbf{X} \setminus \{X_i\}$, while $\mathbf{X}_{|\mathbf{Y}}$ the restriction of $\mathbf{X}$ to $\mathbf{Y} \subseteq \mathbf{X}$. Before describing the multi-label scenario, here we assume $\mathcal{D}$ to be a set of N n-dimensional i.i.d. samples drawn from an unknown joint probability distribution $p(\mathbf{X})$. The primary goal is to learn a model $\mathcal{M}$ from $\mathcal{D}$ estimating a density $p_{\mathcal{M}}(\mathbf{X})$ as close as possible to $p(\mathbf{X})$.

2.1 Tree-structured models

A *directed tree-structured model* [17] over a set of RVs $\mathbf{X}$ is a BN in which each node $X_i \in \mathbf{X}$ has at most one parent. It is a tractable probabilistic model—less expressive than general BNs but performing exact complete and marginal inference in $O(n)$ [17]—encoding a distribution that factorizes as:

$$p(\mathbf{x}) = \prod_{i=1}^{n} \mathsf{p}(x_i|\mathrm{Pa}_{x_i}), \tag{1}$$

where Pa_{x_i} denotes the projection of the assignment $\mathbf{x}$ on the parent of X_i.

To learn a tree-structured model $\mathcal{M}$, one has to estimate both a tree structure $\mathcal{T}$ and the corresponding conditional probabilities $\theta_{i|\mathrm{Pa}_{X_i}} = \mathsf{p}_{\mathcal{M}}(X_i|\mathrm{Pa}_{X_i})$. An optimal model, according to the KL-divergence, can be obtained by employing the classical result from Chow and Liu [4]. We will refer to tree-structured models as Chow-Liu trees, or CLtrees, assuming the Chow-Liu algorithm has been employed to learn them. CLTrees have been employed as the core components of many tractable probabilistic models ranging from mixtures of them [17], SPNs [26] and CNets [21,11,9].

2.2 Cutset Networks

CNets, introduced in [21], are TPMs represented as a hybrid of OR trees and CLTrees as the tree leaves. Their definition has been generalized to comprise generic TPMs as leaf distributions in [10]. In particular, a CNet $\mathcal{C}$ over a set of

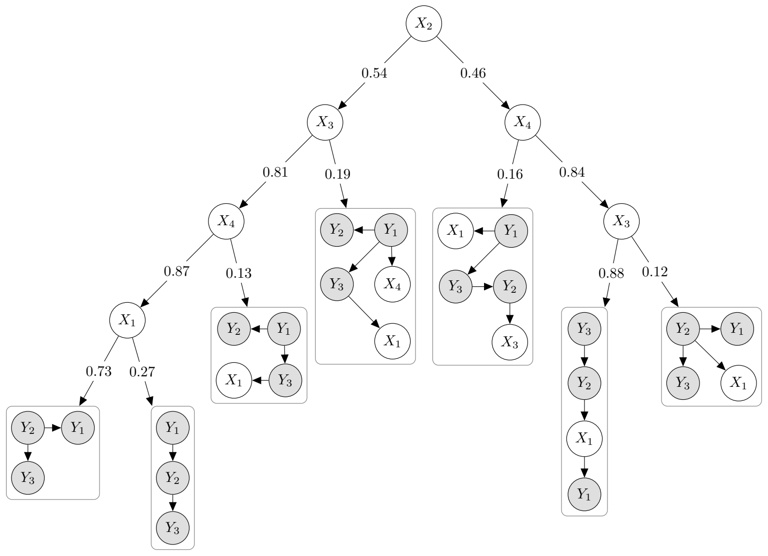

Fig. 1. Example of a CNet over the binary RVs $X_1, \dots, X_4, Y_1, \dots, Y_3$. Inner nodes, with weighted outgoing edges, on variables X_i are OR nodes, while leaf nodes—RVs grouped in a plate—represent CLtrees.

RVs $\mathbf{X}$, is a probabilistic weighted model tree defined via a rooted OR tree $\mathcal{G}$ and a set of TPMs $\{\mathcal{M}_i\}_{i=1}^{L}$ as leaves encoding distributions $\mathsf{p}_{\mathcal{M}_i}$ over a subset of $\mathbf{X}$, called *scope* and denoted as $\mathsf{sc}(\mathcal{M}_i)$. The scope of a CNet $\mathcal{C}$, $\mathsf{sc}(\mathcal{C})$, is the set of RVs appearing in it. Figure 1 shows a CNet over binary RVs where each circled node is an OR tree node labeled by a variable X_i. Outgoing edges are weighted by the probability w_i^0, resp.w_i^1, of conditioning X_i to the value 0 (left), resp. 1 (right). In order to encode a probability distribution it must hold $w_i^0 + w_i^1 = 1$.

Definition 1 (Cutset network [10]). *Given binary RVs $\mathbf{X}$, a CNet is: 1) a TPM $\mathcal{M}$, with $\mathsf{sc}(\mathcal{M}) = \mathbf{X}$; or 2) a weighted disjunction of two CNets $\mathcal{C}_0$ and $\mathcal{C}_1$ graphically represented as an OR node conditioned on RV $X_i \in \mathbf{X}$, with associated weights w_i^0 and w_i^1 s.t. $w_i^0 + w_i^1 = 1$, where $\mathsf{sc}(\mathcal{C}_0) = \mathsf{sc}(\mathcal{C}_1) = \mathbf{X}_{\setminus i}$.*

A CNet $\mathcal{C}$ encodes a distribution factorizing as follows:

$$\mathsf{p}(\mathbf{x}) = \mathsf{p}_l(\mathbf{x}_{|\mathsf{sc}(\mathcal{C})\setminus\mathsf{sc}(\mathcal{M}_l)})\mathsf{p}_{\mathcal{M}_l}(\mathbf{x}_{|\mathsf{sc}(\mathcal{M}_l)}), \tag{2}$$

where $\mathsf{p}_l(\mathbf{x}_{|\mathsf{sc}(\mathcal{C})\setminus\mathsf{sc}(\mathcal{M}_l)}) = \prod_i (w_i^0)^{1-x_i}(w_i^1)^{x_i}$ is a factor obtained by multiplying all the weights attached to the edges of the path in the OR tree starting from the root of $\mathcal{C}$ and reaching a unique leaf node l; while, $\mathsf{p}_{\mathcal{M}_l}(\mathbf{x}_{|\mathsf{sc}(\mathcal{M}_l)})$ is the distribution encoded by the reached leaf l.

Learning CNets. Learning the structure and parameters of a CNet from data equals to perform searching in the space of probabilistic weighted model trees, requiring an exponential time w.r.t. its height k. The learning problem is tackled in a two-stage greedy fashion by: first performing a top-down search in the space of weighted OR trees, and then learning TPMs as leaf distributions according to a conditioned subset of the data.

In [21] has been introduced the first structure learning algorithm for CNets, leveraging a heuristic approach to induce the OR tree and then pruning it to overcome the overfitting. Then in [11] a new approach has been presented, growing the OR tree by a principled Bayesian search maximizing the data likelihood. The general learning schema to learn CNets is shown in Algorithm 1. The procedure tries to select a variable X_i on the input data slice $\mathcal{D}$, and if a such a variable exists, it then recursively tries to decompose the two new slices $\mathcal{D}_0$ and $\mathcal{D}_1$ over $\mathbf{X}_{\setminus i}$. When the slice $\mathcal{D}$ has few instances, or it is defined on few variables, then a leaf distribution is learned.

Algorithm 1 LearnCNet($\mathcal{D}$, $\mathbf{X}$, α, δ, σ) [10]

```
1: Input: a dataset $\mathcal{D}$ over RVs $\mathbf{X}$; $\alpha$: Laplace smoothing factor; $\delta$ min number of
   samples to split; $\sigma$ min number of features to split
2: Output: a CNet $\mathcal{C}$ encoding $\mathsf{p}_{\mathcal{C}}(\mathbf{X})$ learned from $\mathcal{D}$
3: if $|\mathcal{D}| > \delta$ and $|\mathbf{X}| > \sigma$ then
4:     $X_i, \mathsf{success} \leftarrow \mathsf{select}(\mathcal{D}, \mathbf{X}, \alpha)$
5:     if success then
6:         $\mathcal{D}_0 \leftarrow \{\xi \in \mathcal{D} : \xi[X_i] = 0\}$, $\mathcal{D}_1 \leftarrow \{\xi \in \mathcal{D} : \xi[X_i] = 1\}$
7:         $w_0 \leftarrow |\mathcal{D}_0|/|\mathcal{D}|$, $w_1 \leftarrow |\mathcal{D}_1|/|\mathcal{D}|$
8:         $\mathcal{C} \leftarrow w_0 \cdot \mathsf{LearnCNet}(\mathcal{D}_0, \mathbf{X}_{\setminus i}, \alpha, \delta, \sigma) + w_1 \cdot \mathsf{LearnCNet}(\mathcal{D}_1, \mathbf{X}_{\setminus i}, \alpha, \delta, \sigma)$
9: else
10:    $\mathcal{C} \leftarrow \mathsf{learnDistribution}(\mathcal{D}, \mathbf{X}, \alpha)$
11: return $\mathcal{C}$
```

The algorithm proposed in [21], performs a greedy top-down search in the OR-trees space. It implements the select function as a procedure to determine the RV X_i that maximizes a reformulation of the information gain from decision tree theory. To cope with the systematic overfitting, then a post-pruning method on a validation set is introduced. As shown in [10], growing a full binary OR tree using this approach when learning a CNet on $\mathcal{D}$ over RVs $\mathbf{X}$ has time complexity $O(kmn^2)$, where $m = |\mathcal{D}|$, $n = |\mathbf{X}|$, and k is the height of the OR tree.

The approach proposed in [11] to learn CNets exploits a different approach from that in [21], by avoiding decision tree heuristics while choosing the best variable directly maximizing the data log-likelihood. As already reported in [11], the log-likelihood function of a CNet may be recursively decomposed. By exploiting the recursive nature of CNets, a CNet is grown top-down, allowing further expansion—the substitution of a CLtree with an OR node—only if it improves the structure log-likelihood. In particular, one starts with a single CLtree, learned from $\mathcal{D}$ over $\mathbf{X}$, and then checks whether there is a decomposition—an OR node on the best variable X_i applied on two CLtrees—providing a better log-likelihood than that scored by the initial tree. If such a decomposition exists, than the decomposition process is recursively applied to the sub-slices $\mathcal{D}_0$ and $\mathcal{D}_1$ over $\mathbf{X}_{\setminus i}$, testing each leaf for a possible substitution. Growing a full binary OR tree on $\mathcal{D}$ over RVs $\mathbf{X}$ with this new proposed approach has time complexity $O(kmn^3)$.

Extremely Randomized CNets *Extremely Randomized CNets* (XCNets), proposed in [10], are CNets that are built following the procedure sketched in the Algorithm 1, where the OR split node procedure—the select function—is simplified in the most straightforward way: selecting a RV uniformly at random. As a consequence, the cost of the new select function does not directly depend anymore on the number of features and can be considered to be constant. For XCNets, growing a full binary OR tree on $\mathcal{D}$ over $\mathbf{X}$ has time complexity $O(km)$.

Ensembles of CNets. To improve the accuracy of a single model, in [21] CNets have been employed as components of a mixture of the form: $\mathsf{p}(\mathbf{X}) = \sum_{i=1}^{c} \lambda_i \mathcal{C}_i(\mathbf{X})$, being $\lambda_i \geq 0 : \sum_{i=1}^{c} \lambda_i = 1$ the mixture coefficients. Learning such a mixture can be obtained employing EM to alternatively learn both the weights and the mixture components. A more efficient method to learn Mixtures of CNets, presented in [11], adopts bagging. For bagged CNets, mixture coefficients are set equally probable and the mixture components can be learned independently on different bootstrapped data samples. An approach adding random subspace projection to bagged CNets has been introduced in [9]. While its worst case complexity is the same as for bagging, the cost of growing the OR tree reduced by random sub-spacing is effective in practice. Finally, mixtures of CNets have been learned by exploiting some boosting approaches proposed in [20], having time complexity equals to that for bagging or even worst.

3 Sum-Product Networks for MLC

Recently, in [14] has been investigated the use of SPNs for multi-label classification, showing that SPN-based multi-label classifiers are competitive against state-of-the-art classifiers.

Sum-Product Networks [19] are deep probabilistic-models that have obtained impressive results in many tasks. An SPN represents a probability distribution over a set of random variables by a rooted directed-acyclic graph with tractable distributions as leaves, sums and products as inner nodes and weighted edges. In order to deal with MLC different approaches have been proposed in [14]. The two most effective ones that we are reporting here are one based on MPE inference and the other on pool of sequential classification.

In particular, since learning an SPN is a costly operation—differently from learning an XCNet—the general approach is to learn a single SPN $\mathcal{S}$ from the whole training set $\mathcal{D}$, using a structure learner algorithm such as those reported in [26,15], and then perform the classification on an instance $\hat{\mathbf{x}}$ by solving:

$$\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y}} P_{\mathcal{S}}(\mathbf{y}|\mathbf{x}).$$

The first approach ($\mathsf{SPN_{mpe}}$) solves the problem, as done in [12] for CNets, by computing the MPE inference as proposed in [19]. However, differently from CNets computing exact MPE in SPNs is NP-hard in general [18,5]. Being the adopted MPE inference approximate, in order to improve the quality of the predictions another approach ($\mathsf{SPN_{psc}}$) inspired by the classifier chain [22] one has been proposed. Let θ be an ordering of the labels, then the θ_j-th label could be predicted by solving:

$$y_{\theta_j} = \underset{y_{\theta_j}}{\operatorname{argmax}} P_{\mathcal{S}}(Y_k = 1 | \mathbf{x}, y_{\theta_1}, \ldots, y_{\theta_{j-1}}).$$

Instead of fixing a single ordering, $\mathsf{SPN_{psc}}$ uses an ensemble approach, by considering a set of label orderings $\{\theta^1, \ldots, \theta^k\}$, and then computing the set $\mathcal{CC} = \{(y_1^i, \ldots, y_L^i)\}_{i=1}^k$ of classifications obtained for each order. Finally, a method to aggregate the results for producing the single prediction is adopted such as majority voting: $y_j = 1$ iff $1/k \sum_{i=1}^k y_j^i \geq 0.5$.

4 Cutset Networks for MLC

We start to explain how to exploit CNets for MLC as reported in [12], where Restricted CNets have been introduced. As already stated in the introduction, in MLC we assume to have a set of N training instances $\mathcal{D} = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^N$, where each instance has a component $\mathbf{x}^i$ as a vector of M feature values $x_k^i \in \mathbb{R}$. We assume the instances to be i.i.d. according to a probability distribution $P(\mathbf{X}, \mathbf{Y})$. We want to learn a model h able to compute a prediction $\hat{\mathbf{y}}$ for a given attribute instance $\hat{\mathbf{x}}$: $\hat{\mathbf{y}} = [\hat{y}_1, \ldots, \hat{y}_L] = h(\hat{\mathbf{x}}) = \operatorname{argmax}_{\mathbf{y} \in \{0,1\}^L} P(\hat{\mathbf{x}}, \mathbf{y})$.

CNets tackle the MLC problem computing the MPE assignment for the $\mathbf{Y}$—solving $\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y} \in \{0,1\}^L} P(\mathbf{y}|\mathbf{x})$. As showed in [12], differently from SPNs, even MPE queries can be answered in time linear to the size of the network. After having computed the MPE assignment for each leaf node, one can continue visiting all the OR nodes up to the root, obtaining a complete assignment. The leaf node MPE assignment associated to their scope can be computed by employing the max-out variant of the Variable Elimination Algorithm—guaranteed to be linear in the size of the trees [13].

4.1 Restricted CNets

Probabilistic multi-label classifiers can be learned by optimizing one particular loss function, trying to elicit the marginal (resp. conditional) label dependencies if they focus on modeling $p(\mathbf{Y})$ (resp. $p(\mathbf{Y}|\mathbf{X})$) [8]. The approach proposed in [12] consists in learning a CNet while optimizing the joint likelihood, but guiding the structure learning algorithm to focus on the label dependence relationships. This has been obtained by i) limiting the OR split tests to be taken on the $\mathbf{X}$ variables only while growing a CNet, and by ii) constraining both label variables and feature variables in the BNs in the leaves to have as a parent a label variable. This leads to particular networks called *Restricted CNets* (RCNets). The former constraint forces the $\mathbf{Y}$ to be strictly dependent from the $\mathbf{X}$ appearing in the internal nodes, helping the Chow-Liu algorithm to better focus on the $\mathbf{Y}$ variable interactions. The CNet in Figure 1 is an RCNet as well, since label variables are represented only in the leaves. The latter constraint implies that features variables shall be independent given the label variables. In this way the algorithm is forced to model the dependencies among the class variables, and giving the feature to independently contribute in the MPE inference evaluation.

4.2 Extremely Randomized Restricted CNets

Here, in order to improve the classification results for a multi-label classification problem we combine the Extremely Randomized CNets (XCNet) and the Restricted CNets leading to *Extremely Randomized Restricted CNets* (XRCNets). Since the score of a single XCNet is lesser than that obtained with a regular CNet, due to its random process, we learned a mixture of XCNets adopting bagging. However, while for a single CNet the MPE inference is exact, for ensemble of CNets this in not longer true, like for SPNs—being the mixture modeled as a sum node over k models, where k is the number of the components. Hence, we have to use a kind of approximate MPE inference.

In particular, let k be the number of XRCNets used to build the mixture model. Given a test instance $\hat{\mathbf{x}}$, each XRCNet in the mixture can be queried to compute an exact MPE assignment $\hat{\mathbf{y}}_i = [\hat{y}_{i1}, \dots, \hat{y}_{iL}], i = 1, \dots, k$. Then, the final prediction $\hat{\mathbf{y}}$ for the instance $\hat{\mathbf{x}}$ can be computed using a simple aggregation function as follows:

$$\hat{\mathbf{y}} = \left[\mathbb{1}\left(\sum_{i=1}^{k} \hat{y}_{i1} > L/2 \right), \dots, \mathbb{1}\left(\sum_{i=1}^{k} \hat{y}_{iL} > L/2 \right) \right].$$

Other more sophisticated approaches to better approximate the exact MPE assignment for mixtures of CNets deserves a further study.

5 Experimental Results

Here we detail the performance evaluation metrics, the datasets, the algorithms and their experimental settings. The source code of our algorithm and all the scripts to reproduce the experiments reported in this paper are made publicly available[4].

5.1 Evaluation Metrics and Datasets

Given a multi-label dataset consisting of N multi-label instances $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^{N}$, where each $\mathbf{y}_i$ is a vector of L binary values $\mathbf{y}_i \in \{0,1\}^L$. Let h a multi-label classifier and $\hat{\mathbf{y}}^i = h(\mathbf{x}_i) \in \{0,1\}^L$ be the set of label memberships predicted by h

[4] `https://github.com/nicoladimauro/mlxcnet`.

	Domain	M	N	L	LCard	LDens	LDist
Arts-Yahoo	Text	500	7484	26	1.653	0.063	599
Business-Yahoo	Text	500	11214	30	1.598	0.053	233
CAL500	Music	68	502	174	26.043	0.149	502
Emotions	Music	72	593	6	1.868	0.311	27
Flags	Images	19	194	7	3.391	0.484	54
Health-Yahoo	Text	500	9205	32	1.644	0.051	335
Human	Biology	440	3106	14	1.185	0.084	85
Plant	Biology	440	978	12	1.078	0.089	32
Scene	Images	294	2407	6	1.073	0.178	15
Yeast	Biology	103	2417	14	4.237	0.302	198

Table 1. Datasets: number of attributes (M), instances (N), and labels (L).

for the instance $\mathbf{x}_i$. In order to assess the classifier performance, different metrics focus on different dependency relationships among the labels, and are better optimized by taking into account those dependencies [8]. We employ three different metrics to assess the performance of the considered models, namely: accuracy score $= \frac{1}{N}\sum_{i=1}^{N}\frac{|\mathbf{y}^i \wedge \hat{\mathbf{y}}^i|}{|\mathbf{y}^i \vee \hat{\mathbf{y}}^i|}$, Hamming score $= \frac{1}{NL}\sum_{i=1}^{N}\sum_{j=1}^{L}\mathbb{1}(y_j^i = \hat{y}_j^i)$, and exact match score $= \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(\mathbf{y}^i = \hat{\mathbf{y}}^i)$, where $\mathbb{1}(C)$ is the indicator function, while $\wedge$ and $\vee$ are the bitwise logical AND and OR operations, respectively [16], applied vector-wise. The accuracy score is a label set-based measure defined by the Jaccard similarity coefficients between the predicted and true set of labels. The Hamming score rewards methods for predicting individual labels well, while the exact match score computes the percentage of instances whose predicted set of labels $\hat{\mathbf{y}}$ matches the true set of labels $\mathbf{y}$ *exactly*.

We considered 10 numerical traditional multi-label datasets—accessible from the MULAN[5], MEKA[6], and LABIC[7] websites—belonging to a wide variety of application domains with their labels ranging from 6 to 174, the number of attributes ranging from 19 to 500, and the number of examples ranging from 194 to 11214. Table 1 reports the information about the adopted datasets, where M, N and L represent the number of attributes, instances, and possible labels respectively. Furthermore, for each dataset $\mathcal{D}$ the following statistics are also reported: *Label Cardinality*: $LCard(\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{L} y_j^i$, *Label Density*: $LDens(\mathcal{D}) = \frac{LCard(\mathcal{S})}{L}$ and *Distinct Labels*: $LDist(\mathcal{D}) = |\{\mathbf{y}|\exists(\mathbf{x},\mathbf{y}) \in \mathcal{D}\}|$.

As reported in [12], we discretized all the numeric features for each dataset implementing the Label-Attribute Interdependence Maximization (LAIM) [3] discretization method for multi-label data. All the algorithms have been ran on the datasets preprocessed by LAIM.

[5] `http://mulan.sourceforge.net/`.
[6] `http://meka.sourceforge.net/`.
[7] `http://computer.njnu.edu.cn/Lab/LABIC/LABIC_Software.html`.

	CNets		competitors			
dataset	XRCN	RCN	$\mathsf{SPN}_{\mathsf{psc}}$	$\mathsf{SPN}_{\mathsf{mpe}}$	RAkEL	CC
arts	0.429	0.425	0.334↑	0.379↑	0.293↑	0.402↑
business	0.720	0.728	0.732↓	0.720	0.729	0.729
cal	0.172	0.184	0.220↓	0.217↓	0.001↑	0.199↓
emotions	0.584	0.524	0.586	0.555↑	0.532↑	0.493↑
flags	0.548	0.544	0.537	0.541	0.576	0.554
health	0.586	0.609	0.627↓	0.619↓	0.560↑	0.616↓
human	0.330	0.335	0.180↑	0.209↑	0.276↑	0.318
plants	0.337	0.322	0.146↑	0.218↑	0.301↑	0.298↑
scene	0.705	0.680	0.682↑	0.634↑	0.687	0.606↑
yeast	0.471	0.441	0.472	0.459	0.508↓	0.444↑
↑ / ↓			4/3	5/2	6/1	5/2
Avrg score	0.488	0.479	0.452	0.455	0.446	0.466

Table 2. Accuracy scores of CNets and the competitors on the ten datasets.

5.2 Algorithms

For the experimental evaluation, we compared both RCNets (RCN) and an ensemble of 10 XRCNets (XRCN)[8], to different algorithms. First, we include in the comparison the RAndom k-labELsets (RAkEL) algorithm [24], an ensemble method for multi-label classification that constructs each member of the ensemble by considering a small random subset of labels and learning a single-label classifier for the prediction of each element in the powerset of this subset. Another competitive algorithm included is Classifier Chains (CC) [22], a chaining method that can model label correlations while maintaining acceptable computational complexity. Both RAkEL and CC have been run using their openly available implementations in MEKA[9] (release 1.9.2), with parameters set as default values[10].

Finally, we included the two algorithms based on SPNs [14], $\mathsf{SPN}_{\mathsf{mpe}}$ and $\mathsf{SPN}_{\mathsf{psc}}$[11]. After having learned an SPN on the training dataset $(\mathbf{x}_i, \mathbf{y}_i)$, $\mathsf{SPN}_{\mathsf{mpe}}$ performs classification on an instance $\mathbf{x}$ by computing the approiximate MPE $\text{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x})$, while $\mathsf{SPN}_{\mathsf{psc}}$ combines by majority voting the predictions of an ensemble of sequential classifications.

All experiments have been run on a 8-core Intel Xeon E5-1620 @3.5 GHz with 16Gb of RAM and Linux kernel 4.4.0-59.

[8] Both have be run with `-d 0.1`, leaving all the other parameters set to default value.

[9] Available at `http://meka.sourceforge.net`.

[10] `RAkEL`, resp. CC, has been executed with Support Vector Machines with polynomial kernel, resp. with C4.5 decision trees, as base classifier.

[11] We executed the code avalible at `https://github.com/giulianavll/MLC-SPN` to reproduce the results reported in this paper. The algorithm used for learning the structure of SPNs corresponds to that reported in [26].

	CNets		competitors			
dataset	XRCN	RCN	SPN_{psc}	SPN_{mpe}	RAkEL	CC
arts	0.939	0.937	0.945↓	0.942↓	0.947↓	0.936↑
business	0.972	0.974	0.976↓	0.975↓	0.976↓	0.975↓
cal	0.860	0.854	0.856↑	0.833↑	0.850↑	0.845↑
emotions	0.811	0.770	0.799	0.783↑	0.773↑	0.748↑
flags	0.707	0.701	0.689	0.697	0.714	0.705
health	0.963	0.966	0.969↓	0.968↓	0.965↓	0.967↓
human	0.914	0.890	0.917↓	0.912↑	0.886↑	0.890↑
plants	0.913	0.881	0.910	0.899	0.870↑	0.884↑
scene	0.902	0.882	0.903	0.892	0.895	0.863↑
yeast	0.790	0.753	0.770↑	0.757↑	0.779↑	0.752↑
↑ / ↓			2/4	4/3	5/3	7/2
Avrg score	0.877	0.861	0.873	0.866	0.865	0.857

Table 3. Hamming scores of CNets and the competitors on the ten datasets.

5.3 Results and discussion

The results over a 10-fold cross validation for each evaluation metric and for each classification algorithm on all the datasets are reported in Table 2 for the accuracy score, Table 3 for the Hamming score, and Table 4 for the exact match score. In order to assess whether the differences of the scores reported in Table 2, 3 and 4 are statistically significant, a *t*-test has been adopted comparing the means with a significance level $p = 0.05$. In each column of the tables, a ↑ (resp. ↓) denotes that XRCN (resp. the competitor) outperforms the competitor (resp. XRCN) with a difference statistically significant.

First of all, as we can see, the ensemble of XRCNets obtains on average scores better that those obtained with a single RCNet, proving the validity of the ensemble approach—even if each components of the ensemble are learned completely at random, their aggregation provides more precise predictions when compared to a single RCNet. Furthermore, for each metric the average score over all the datasets is always greater than that obtained by other competitors.

As regards the accuracy score, XRCN obtains better results when compared to all the competitors, while for the Hamming and exact match scores the values seems to be comparable to that of SPN_{psc}. Indeed, XRCN outperforms the competitors in 6 (RAkEL), 5 (SPN_{mpe} and CC), and 4 (SPN_{psc}) in terms of the accuracy score. In terms of the Hamming score, XRCN is superior to the others in 7 (CC), 5 (RAkEL), 4 (SPN_{mpe}), and 2 (SPN_{pcs}) cases. Regarding the exact match score, XRCN is superior to the competitors in 5 (RAkEL), 3 (SPN_{psc} and CC), and 2 (SPN_{mpe}) cases.

It is important to note that, even if we used a fixed inference procedure, i.e. MPE, it is robust for each score. Indeed, as reported in [7], MPE inference is maximizer of the exact match score, while marginal inference for each label is a maximizer for the Hamming score.

	CNets		competitors			
dataset	XRCN	RCN	SPN$_{psc}$	SPN$_{mpe}$	RAkEL	CC
arts	0.304	0.327	0.278↑	0.309	0.222↑	0.319
business	0.558	0.571	0.584↓	0.570↓	0.570	0.575
cal	0.000	0.000	0.000	0.000	0.000	0.000
emotions	0.336	0.246	0.352	0.325	0.260↑	0.265↑
flags	0.155	0.140	0.139	0.144	0.170	0.170
health	0.436	0.473	0.514↓	0.504↓	0.387↑	0.496↓
human	0.275	0.251	0.161↑	0.186↑	0.176↑	0.254
plants	0.316	0.292	0.139↑	0.207↑	0.183↑	0.267↑
scene	0.612	0.569	0.644↓	0.599	0.589	0.561↑
yeast	0.147	0.138	0.160	0.158	0.141	0.156
↑ / ↓			3/3	2/2	5/0	3/1
Avrg score	0.314	0.301	0.297	0.300	0.270	0.306

Table 4. Exact match scores of CNets and the competitors on the ten datasets.

Overall, while XRCN outperforms problem transformation schemes such a RAkEL and CC, it is competitive with respect to the approaches based on SPNs for the Hamming and exact match scores and outperforming them for the accuracy score. Adopting sophisticated schemes to infer the correct label predictions such those used in SPN$_{psc}$ represent an interesting future work.

6 Conclusion

In this paper, we employed the recently introduced tractable probabilistic model XCNets to tackle the MLC problem. XCNets reduce the structure learning complexity making able to learn ensembles of XCNets outperforming state-of-the-art density estimators. The experimental evaluation on real-world datasets showed how our approach can effectively improve the accuracy, exact match and Hamming scores, proving itself to be highly competitive against complex approaches.

References

1. Antonucci, A., Corani, G., Mauá, D.D., Gabaglio, S.: An ensemble of bayesian networks for multilabel classification. In: IJCAI. pp. 1220–1225 (2013)
2. Boutell, M.R., Luo, J., Shen, X., Brown, C.M.: Learning multi-label scene classification. Pattern Recognition 37(9), 1757–1771 (2004)
3. Cano, A., Luna, J.M., Gibaja, E.L., Ventura, S.: LAIM discretization for multi-label data. Information Sciences 330, 370–384 (2016)
4. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory 14(3), 462–467 (1968)
5. Conaty, D., de Campos, C.P., Mauá, D.D.: Approximation complexity of maximum A posteriori inference in sum-product networks. In: UAI (2017)

6. Corani, G., Antonucci, A., Mau, D.D., Gabaglio, S.: Trading off speed and accuracy in multilabel classification. In: PGM, pp. 145–159 (2014)
7. Dembczyński, K., Cheng, W., Hüllermeier, E.: Bayes optimal multilabel classification via probabilistic classifier chains. In: ICML. pp. 279–286 (2010)
8. Dembczyński, K., Waegeman, W., Cheng, W., Hüllermeier, E.: On label dependence and loss minimization in multi-label classification. Machine Learning 88(1), 5–45 (2012)
9. Di Mauro, N., Vergari, A., Basile, T.M.: Learning bayesian random cutset forests. In: ISMIS, pp. 122–132 (2015)
10. Di Mauro, N., Vergari, A., Basile, T.M.A., Esposito, F.: Fast and accurate density estimation with extremely randomized cutset networks. In: ECML/PKDD. pp. 203–219 (2017)
11. Di Mauro, N., Vergari, A., Esposito, F.: Learning accurate cutset networks by exploiting decomposability. In: AIXIA, pp. 221–232 (2015)
12. Di Mauro, N., Vergari, A., Esposito, F.: Multi-label classification with cutset networks. In: PGM. vol. 52, pp. 147–158 (2016)
13. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
14. Llerena, J.V., Mau, D.D.: On using sum-product networks for multi-label classification. In: BRACIS. pp. 25–30 (2017)
15. Lowd, D., Rooshenas, A.: The Libra Toolkit for Probabilistic Models. JMLR 16, 2459–2463 (2015)
16. Madjarov, G., Kocev, D., Gjorgjevikj, D., Deroski, S.: An extensive experimental comparison of methods for multi-label learning. Pattern Recognition 45(9), 3084–3104 (2012)
17. Meila, M., Jordan, M.I.: Learning with mixtures of trees. JMLR 1, 1–48 (2000)
18. Peharz, R., Gens, R., Pernkopf, F., Domingos, P.: On the latent variable interpretation in sum-product networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 39(10), 2030–2044 (2017)
19. Poon, H., Domingos, P.: Sum-Product Network: a New Deep Architecture. NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning (2011)
20. Rahman, T., Gogate, V.: Learning ensembles of cutset networks. In: AAAI (2016)
21. Rahman, T., Kothalkar, P., Gogate, V.: Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of Chow-Liu trees. In: ECML/PKDD, pp. 630–645 (2014)
22. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. In: ECML/PKDD. pp. 254–269 (2009)
23. Roth, D.: On the hardness of approximate reasoning. Artificial Intelligence 82(12), 273–302 (1996)
24. Tsoumakas, G., Katakis, I., Vlahavas, I.: Random k-labelsets for multilabel classification. IEEE Transactions on Knowledge and Data Engineering 23(7), 1079–1089 (2011)
25. Tsoumakas, G., Katakis, I., Vlahavas, I.P.: Mining multi-label data. In: Data Mining and Knowledge Discovery Handbook, 2nd ed., pp. 667–685. Springer (2010)
26. Vergari, A., Di Mauro, N., Esposito, F.: Simplifying, Regularizing and Strengthening Sum-Product Network Structure Learning. In: ECML-PKDD. pp. 343–358 (2015)