Multi-Label Classification with Cutset Networks

Nicola Di Mauro NICOLA.DIMAURO@UNIBA.IT
Antonio Vergari ANTONIO.VERGARI@UNIBA.IT
Floriana Esposito FLORIANA.ESPOSITO@UNIBA.IT
Department of Computer Science, University of Bari
70125 Bari (Italy)

Abstract

In this work, we tackle the problem of Multi-Label Classification (MLC) by using Cutset Networks (CNets), weighted probabilistic model trees, recently proposed as *tractable* probabilistic models for discrete distributions. We employ CNets to perform Most Probable Explanation (MPE) inference exactly and efficiently and we improve a state-of-the-art structure learning algorithm for CNets by explicitly taking advantage of label dependencies. We achieve this by forcing the tree inner nodes to represent only feature variables and by exploiting structural heuristics while learning the leaf models. A thorough experimental evaluation on ten real-world datasets shows how the proposed approach improves several metrics for MLC, proving it to be competitive with problem transformation methods like classifier chains.

Keywords: Multi-label classification; tractable inference; structure learning; cutset networks.

1. Introduction

Many real world classification problems involve multiple label classes. The problem of *Multi-Label Classification* (MLC) concerns learning a mapping from an example to a set of *relevant* labels. This issue has recently attracted significant attention due to an increasing number of applications, such as image and video annotation, functional genomics in bioinformatics, text categorization, and others (Madjarov et al., 2012). A common approach to MLC is to adopt a *problem transformation* technique, where a multi-label problem is transformed into one or more single-label problems. A popular problem transformation method to deal with MLC is *binary relevance* (BR) (Boutell et al., 2004; Tsoumakas et al., 2010). A BR classifier decomposes the MLC problem into a set of single label classification problems, one for each different label. The classifiers are learned independently, thus possibly losing the dependencies among the label variables. However, it is very well known that exploiting these dependencies can significantly improve the classification performance in a MLC scenario, as reported by Dembczyński et al. (2012). In order to exploit potential label correlations, the *classifier chain* approach (CC) (Read et al., 2009) transforms a MLC problem into a chain of binary classification problems, where subsequent binary classifiers in the chain are built upon the predictions of preceding ones.

Probabilistic Graphical Models (PGMs) (Koller and Friedman, 2009), such as Bayesian Networks (BNs) and Markov Networks, provide a powerful formalism to model and reason about MLC problems. Indeed, they are able to capture the conditional independence assumptions among random variables into a graph-based representation. Exploiting inference tools in PGMs results in answering different query types such as conditional probability queries and Most Probable Explanation (MPE) queries. Indeed, exploiting MPE inference is one of the approaches to solve MLC (Antonucci et al., 2013; Corani et al., 2014). Nevertheless, learning and inference with PGMs can be challenging.

In works like (Waal and Gaag, 2007; Rodríguez and Lozano, 2008; Bielza et al., 2011) the concept of multi-dimensional Bayesian network classifiers (MBCs) for MLC has been introduced. The structure of this kind of networks is learned by partitioning the arcs of the graphs into three sets: links among the label variables (*label graph*), links among features (*features graph*) and links between label and features variables (*bridge graph*). Each subgraph is separately learned and different families of MBCs could be derived by imposing restrictions on the graphical structure of the label and feature subgraphs, e.g. empty graphs, trees, polytrees, and so on. In (Antonucci et al., 2013), an ensemble of BN classifiers has been used for MLC, thus extending to the multi-label case the idea of averaging over a constrained family of classifiers. The label subgraph is assumed to be a tree, and a different classifier for each label is instantiated. Independence of the features given the labels is assumed, thus generalizing to the multi-label case the naïve Bayes assumption. Thanks to this assumption, the optimal bridge subgraph, linking labels and features, can be identified in polynomial time. Due to the high complexity of the inference regarding the MPE joint configuration of all the labels, the same authors switched from the ensemble approach to the single model approach (Corani et al., 2014). To compensate the losing in accuracy of a single model when compared to an ensemble of models, they introduced a more sophisticated structural learning procedure for the label subgraph. Still, the bottleneck remains the high complexity of the inference task during learning and classification.

The need for exact and efficient inference procedures has lead to the introduction of *Tractable Probabilistic Models* (TPMs). An instance of TPMs, tractable PGMs usually trade off expressiveness in exchange for a tractable inference guarantee, see for instance the mixture of tree distributions (Meilă and Jordan, 2000). TPMs include, among the others, the recently introduced Sum-Product Networks (SPNs) (Poon and Domingos, 2011) as deep architectures encoding probability distributions by layering hidden variables as mixtures of independent components. The greater expressive capacity of SPNs poses new challenges: learning the structure of SPNs involves balancing inference times, hence the whole learning time, and the model accuracy in the terms of a likelihood score (Vergari et al., 2015). To cope with similar issues, *Cutset Networks* (CNets) have been recently proposed as easy-to-learn TPMs (Rahman et al., 2014). 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, i.e., conditioning OR nodes, are associated to random variables and outgoing branches represent conditioning on the values for those variables domains. Structure learning algorithmic variants for CNets have been proven to be both accurate and scalable, given the decomposability property guaranteed by the tree structure (Di Mauro et al., 2015b,a).

In this work we show how to employ CNets for MLC problems. As we will show, CNets, differently from MBCs, offer exact and tractable MPE inference, thus alleviating a major issue for MLC. Additionally, we propose a CNet structure learning algorithmic variant for MLC. We modify the searching procedure proposed in (Di Mauro et al., 2015b) by focusing on label dependencies, as it has been proven to be advantageous for MLC (Dembczyński et al., 2012). First we force each tree inner node to condition on feature variables only. Then, we apply some heuristics while learning the leaf tree models to put more prominence on the links among label variables while seeking for independence among feature variables. In this way a model can exploit more feature contributions while making predictions about the label variables. We focus on learning a single model and then we prove it to be very competitive against more sophisticated approaches like BR and CC. In a thorough empirical comparison on 10 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

In order to introduce the basics about CNets, and before switching to the multi-label case in Section 3, now we assume $\mathcal{D}$ to be a set of N i.i.d. instances over the discrete variables $\mathbf{X} = \{X_1, \ldots, X_m\}$, whose domains are the sets $\text{Val}(X_i) = \{x_i^j\}_{j=1}^{k_i}, i = 1, \ldots, m$.

2.1 Tree-structured probabilistic graphical models

A *directed tree-structured model* (Meilă and Jordan, 2000) is a BN in which each variable has at most one parent. The joint probability distribution over a set of discrete variables $\mathbf{X}$ represented by such a model can be factorized as $P(\mathbf{X}) = \prod_{i=1}^{m} P(X_i|\text{Pa}_i)$, where Pa_i stands for the parent variable of X_i, if present. It follows immediately that inference for complete or marginal queries has complexity linear in the number of variables, hence the tractability of tree-structured models.

The classic algorithm for learning tree-structured models is that presented in (Chow and Liu, 1968). There it is shown that maximizing the Mutual Information (MI) among random variables in $\mathbf{X}$ leads to the best tree, in an information-theoretic sense, approximating the underlying probability distribution of $\mathcal{D}$ in terms of the Kullback-Leibler divergence. The learning process for obtaining a tree-structured probabilistic model (CLtree in the following) proceeds as follows. Firstly, for each pair of variables in $\mathbf{X}$, their MI is estimated from $\mathcal{D}$; then a maximum spanning tree is built on the weighted graph induced by the MI as an adjacency matrix. Rooting the tree in a randomly chosen variable and traversing it leads to the learned tree-structured BN.

2.2 Structure and Parameter Learning of Cutset Networks

As introduced in (Rahman et al., 2014) and then extended in (Di Mauro et al., 2015b,a), CNets are a hybrid of rooted OR trees and CLtrees, with OR nodes as internal nodes and CLtrees as leaves (see Figure 1). More formally, a CNet is a pair $\langle \mathcal{G}, \boldsymbol{\gamma} \rangle$, where $\mathcal{G} = \mathcal{O} \cup \{\mathcal{T}_1, \ldots, \mathcal{T}_L\}$ is the graphical structure, composed by a rooted OR tree, $\mathcal{O}$, and by leaf trees $\mathcal{T}_l$; and $\boldsymbol{\gamma} = \boldsymbol{w} \cup \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L\}$ is the parameter set containing the OR tree weights $\boldsymbol{w}$ and the leaf tree parameters $\boldsymbol{\theta}_l$. The *scope* of a CNet $\mathcal{G}$ (resp. a leaf tree $\mathcal{T}_l$), denoted as $\text{scope}(\mathcal{G})$ (resp. $\text{scope}(\mathcal{T}_l)$), is the set of random variables that appear in it. Each node in the OR tree is labeled by a variable X_i, and each edge emanating from it represents the conditioning of X_i by a value $x_i^j \in Val(X_i)$, weighted by the conditional probability $w_{i,j}$ of conditioning the variable X_i to the value x_i^j. A CNet can be thought of a model tree associating to each instance a weighted probabilistic leaf model.

In (Di Mauro et al., 2015b) a recursive definition of CNets has been proposed along with the proof of the decomposability of their log-likelihood and Bayesian Information Criterion (BIC) (Friedman et al., 1997) scores. This lead to a principled algorithm, dCSN, to learn the structure of CNets. A CNet is defined as follows: a) a CLtree, with scope $\mathbf{X}$, is a CNet; b) given $X_i \in \mathbf{X}$ a variable with $|Val(X_i)| = k$, graphically conditioned in an OR node, a weighted disjunction of k CNets $\mathcal{G}_i$ with same scope $\mathbf{X}_{\setminus i}$ is a CNet, where all weights $w_{i,j}$, $j = 1, \ldots, k$, sum up to one, and $\mathbf{X}_{\setminus i}$ denotes the set $\mathbf{X}$ minus the variable X_i. Following this definition, the computation of the log-likelihood of a CNet can be decomposed as follows (Di Mauro et al., 2015b). Given a CNet $\langle \mathcal{G}, \boldsymbol{\gamma} \rangle$ over variables $\mathbf{X}$ and a set of instances $\mathcal{D}$, its log-likelihood $\ell_{\mathcal{D}}(\langle \mathcal{G}, \boldsymbol{\gamma} \rangle)$ can be computed as:

$$\ell_{\mathcal{D}}(\langle \mathcal{G}, \boldsymbol{\gamma} \rangle) = \begin{cases} \sum_{\xi \in \mathcal{D}} \sum_{i=1}^{m} \log P(\xi[X_i]|\xi[\text{Pa}_i]) & \text{if } \mathcal{G} = \{\mathcal{T}\} \\ \sum_{j=1}^{k} M_j \log w_{i,j} + \ell_{\mathcal{D}_j}(\langle \mathcal{G}_j, \boldsymbol{\gamma}_{\mathcal{G}_j} \rangle) & \text{otherwise,} \end{cases}$$

where the first equation refers to the case of a CNet composed by a single CLtree ($\xi[X_i]$ is the value assumed by an instance ξ in correspondence of a particular variable X_i), while the second one specifies the case of an OR tree rooted on the variable X_i, with $|Val(X_i)| = k$, where, for each $j = 1, \dots, k$, $\mathcal{G}_j$ is the CNet involved in the disjunction with parameters $\boldsymbol{\gamma}_{\mathcal{G}_j}$, $\mathcal{D}_j = \{\xi \in \mathcal{D} : \xi[X_i] = x_i^j\}$ is the slice of $\mathcal{D}$ after conditioning on X_i, and $M_j = |\mathcal{D}_j|$ its cardinality. $\ell_{\mathcal{D}_j}(\langle \mathcal{G}_j, \boldsymbol{\gamma}_{\mathcal{G}_j}\rangle)$ denotes the log-likelihood of the sub-CNet $\mathcal{G}_j$ on $\mathcal{D}_j$.

By exploiting the log-likelihood decomposition, in dCSN a CNet is grown top-down allowing further expansion, i.e., substituting a CLtree with an OR node only if it improves the log-likelihood. In detail, dCSN starts with a single CLtree, for variables $\mathbf{X}$, learned from $\mathcal{D}$ and it checks whether there is a decomposition, i.e., an OR node applied on as many CLtrees as the values of the best variable X_i, providing a better log-likelihood than that scored by the initial tree. If a such decomposition exists, than the decomposition process is recursively applied to the sub-slices $\mathcal{D}_i$, testing each leaf for a possible substitution. In order to penalize complex structures and thus avoiding overfitting, a BIC score has been adopted. The BIC score of a CNet $\langle \mathcal{G}, \boldsymbol{\gamma}\rangle$ on data $\mathcal{D}$ is defined as:

$$\text{score}_{\text{BIC}}(\langle \mathcal{G}, \boldsymbol{\gamma}\rangle) = \log P_{\mathcal{D}}(\langle \mathcal{G}, \boldsymbol{\gamma}\rangle) - \frac{\log N}{2}\text{Dim}(\mathcal{G}),$$

where $\text{Dim}(\mathcal{G})$ is the model dimension, set to the number of OR nodes appearing in $\mathcal{G}$, and N is the size of the dataset $\mathcal{D}$. A Bayesian approach to learn a CLtrees $\mathcal{T}$ with parameters $\boldsymbol{\theta}$ from data $\mathcal{D}$ has been adopted, by exploiting as scoring function $P(\boldsymbol{\theta}|\mathcal{D}) \approx P(\mathcal{D}|\boldsymbol{\theta})P(\boldsymbol{\theta})$. Considering Dirichlet priors, the regularized model parameter estimates are:

$$\theta_{x_i|\text{Pa}_i} \approx E_{P(\theta_{x_i|\text{Pa}_i}|\mathcal{D},\mathcal{T})}[\theta_{x_i|\text{Pa}_i}] = \frac{M_{x_i,\text{Pa}_i} + \alpha_{x_i|\text{Pa}_i}}{M_{\text{Pa}_i} + \alpha_{\text{Pa}_i}},$$

where $M_{\mathbf{z}}$ is the number of entries in a dataset $\mathcal{D}_{\mathbf{z}}$ having the set of variables $\mathbf{Z}$ instatiated to $\mathbf{z}$.

dCSN, whose source code is public available[1], has been extended to the MLC scenario as described in the next section.

3. Restricted CNets for MLC

In this section we will explain how to exploit CNets for MLC. In the following we assume to have a set of training data $\mathcal{D}$ of N instances, $\mathcal{D} = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^N$. Each training instance has a component $\mathbf{x}^i \in \mathcal{X}$ as a vector of M feature values $\mathbf{x}^i = [x_1^i, \dots, x_M^i]$, where each $x_k^i \in \mathbb{R}$. The set $\mathcal{L} = \{y_1, \dots, y_L\}$ denotes the output domain of possible labels. Each vector of attribute values $\mathbf{x}^i$ for each instance is associated to a subset $\mathcal{Y}_i \subseteq \mathcal{L}$ of these labels, represented by an vector of L binary values $\mathbf{y}^i = [y_1^i, \dots, y_L^i]$, where $y_j^i = 1$, if the label y_j is assigned to (i.e., is relevant for) the instance, and $y_j^i = 0$ otherwise (i.e., y_j^i represents the binary relevance of the jth label associated to the ith instance). We assume the instances to be i.i.d. according to a probability distribution $P(\mathbf{X}, \mathbf{Y})$ on $\mathcal{X} \times \mathcal{Y}$. We are interested to learn a model h, s.t., given a test attribute instance $\hat{\mathbf{x}}$, we can compute a prediction:

$$\hat{\mathbf{y}} = [\hat{y}_1, \dots, \hat{y}_L] = h(\hat{\mathbf{x}}) = \underset{\mathbf{y}\in\{0,1\}^L}{\text{argmax}}\, P(\hat{\mathbf{x}}, \mathbf{y}).$$

1. `https://github.com/nicoladimauro/dcsn`

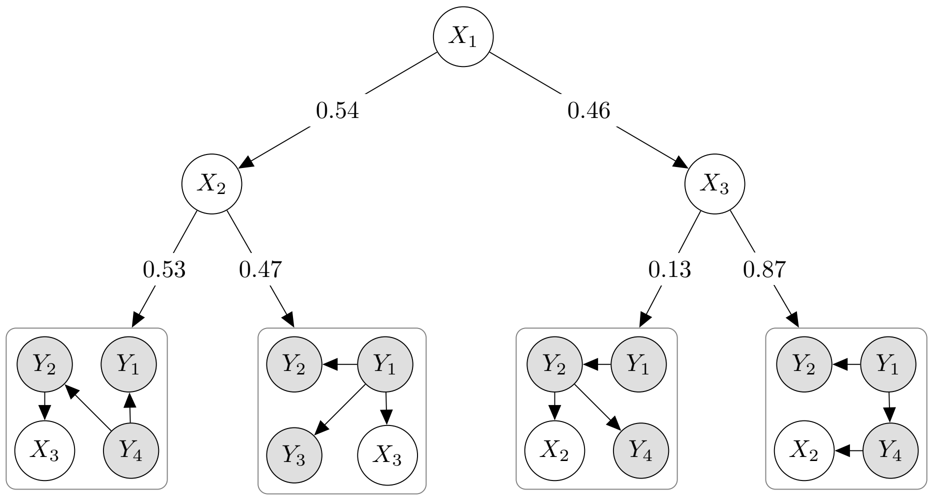

Figure 1: Example of a binary CNet model. Internal nodes on variables X_i are OR nodes, while leaf nodes are CLtrees encoding a direct graphical model.

First we will describe how to perform exact MPE inference by the means of CNets, then we will introduce how we adapt the structure learning algorithm proposed in (Di Mauro et al., 2015b) for MLC. Computing the MPE assignment for the $\mathbf{Y}$ is a common way for probabilistic classifiers to tackle the MLC problem (Dembczyński et al., 2012). This equals to solve:

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}\in\{0,1\}^L} P(\mathbf{y}|\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}\in\{0,1\}^L} P(\mathbf{y},\mathbf{x}).$$

where the $\mathbf{X} = \mathbf{x}$ are assumed to be observed as evidence.

As already stated, CNets and their leaf tree structures favor tractable probabilistic inference for complete evidences. Here we show how even MPE queries can be answered in time linear to the size of a CNet. In order to do so, one can compute the MPE state for each leaf node and then continue visiting all the OR nodes up to the root, obtaining a complete assignment. For the leaf nodes the MPE state associated to their scope and its corresponding probability can be computed by employing the max-out variant of the Variable Elimination Algorithm, which is guaranteed to be linear in the size of the trees (Koller and Friedman, 2009). For each OR node, defined on the variable Z_i, and with outgoing weights $w_1, \ldots, w_k$, one would have k possible MPE states, $\mathbf{s}_1, \ldots, \mathbf{s}_k$, corresponding to each sub-tree. Each state, alongside its MPE probability $p_i, i = 1, \ldots, k$, has been computed when exploring the child nodes. If Z_i appears as evidence, i.e. $Z_i \in \mathbf{X}$, then one simply chooses the child branch corresponding to its value. Otherwise, the branch $j^* = \operatorname{argmax}_{j=1,\ldots,k} w_j p_j$ and the corresponding MPE state are chosen. This is also linear in the number of child sub-trees.

3.1 Restricted Structure Learning

Probabilistic multi-label classifiers can be learned by optimizing one particular loss function, e.g. subset 0/1 loss or hamming loss. While doing this, they may try to elicit the marginal (resp. conditional) label dependencies if they focus on modeling $p(\mathbf{Y})$ (resp. $p(\mathbf{Y}|\mathbf{X})$) (Dembczyński et al., 2012). Our approach consists in learning a CNet while optimizing the joint likelihood, but guiding the structure learning algorithm to focus on the label dependence relationships. We employ several heuristics to achieve this, both during the CNet structure growing phase and the leaf tree learning steps. The common idea behind all of them is to prefer modeling more accurately the dependencies among the $\mathbf{Y}$ than the ones over the $\mathbf{X}$.

First, we limit the OR split tests to be taken on the $\mathbf{X}$ variables only while growing a CNet. This leads to particular networks which we call *Restricted CNets* (RCNets). By doing this we force the $\mathbf{Y}$ to be strictly dependent from the $\mathbf{X}$ appearing in the internal nodes. This is pretty evident when using an RCNet as a generative model. This also helps 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: e.g., the leftmost leaf models the interactions among all label variables but on the subset of the dataset where X_1 and X_2 variables are set to 0.

A second constraint has been imposed on the structure of the BNs in the leaves: each label variable and each feature variable can only have as a parent a label variable. This implies that features variables shall be independent given the label variables. From the perspective of a MBC, this constraint translates into the label subgraph to be a tree, and the feature sub-graph to be empty. However, differently from a single MBC, in an RCNet we have several possible variations of such templated trees, one for each leaf and modeling different dependencies for different views of the data. In this way we force to model the dependencies among the class variables, and giving the feature to independently contribute in the MPE inference evaluation. In order to fulfill this constraint, the classical Chow-Liu tree algorithm described in Section 2 has been modified as reported in Algorithm 1 leading to restricted tree-structured PGMs. After having computed the mutual information matrix $\mathbf{MI}$ between all the variables, we forcibly alter some of its values. Let the $\mathbf{MI}$ matrix be represented as a block matrix:

$$\mathbf{MI} = \begin{pmatrix} \mathbf{MI}_{\mathbf{X},\mathbf{X}} & \mathbf{MI}_{\mathbf{X},\mathbf{Y}} \\ \mathbf{MI}^{T}_{\mathbf{X},\mathbf{Y}} & \mathbf{MI}_{\mathbf{Y},\mathbf{Y}} \end{pmatrix}$$

where $\mathbf{MI}_{\mathbf{X},\mathbf{X}}$ (resp. $\mathbf{MI}_{\mathbf{Y},\mathbf{Y}}$) denotes the sub-matrix comprising the mutual information values among variables in $\mathbf{X}$ (resp. $\mathbf{Y}$) and $\mathbf{MI}_{\mathbf{X},\mathbf{Y}}$ denotes the sub-matrix comprising the values among one feature and one label variables. We first set $\mathbf{MI}_{\mathbf{X},\mathbf{X}} = \mathbf{0}$ to enforce the label conditional independence of the $\mathbf{X}$, then we boost the mutual information values for the label variables by setting $\mathbf{MI}_{\mathbf{Y},\mathbf{Y}} = \mathbf{MI}_{\mathbf{Y},\mathbf{Y}} + \max(\mathbf{MI})$, thus forcing the insertion of all the edges over the $\mathbf{Y}$ in the spanning tree before considering the interactions with the $\mathbf{X}$.

4. Experimental Evaluation

In this section we present the experimental setup to detail the performance evaluation metrics, the multi-label datasets, the algorithms and their experimental settings. The source code of our algorithm, scripts and datasets are made publicly available[2].

4.1 Evaluation Metrics

Recall that given a multi-label dataset consisting of N instances, each one associated with a set of labels $\mathcal{Y}_i \subseteq \mathcal{L}$, denoted by a vector of L binary values $\mathbf{y}^i$, a classifier predicts a set of labels $\hat{\mathcal{Y}}$, denoted by a vector of L binary values $\hat{\mathbf{y}}^i$. In MLC there is no one single way to assess the

2. `https://github.com/nicoladimauro/dcsn`

Algorithm 1 LearnRestrictedCLT($\mathcal{D}$, $\mathbf{X}$, $\mathbf{Y}$)

```
1: Input: a set of instances $\mathcal{D}$ over a set of features $\mathbf{X}$ and labels $\mathbf{Y}$
2: Output: $\langle \mathcal{T}, \boldsymbol{\theta} \rangle$, a tree $\mathcal{T}$ with parameters $\boldsymbol{\theta}$ encoding a pdf over $\mathbf{X} \cup \mathbf{Y}$
3: $\mathbf{MI} \leftarrow \mathbf{0}_{|\mathbf{X}\cup\mathbf{Y}|\times|\mathbf{X}\cup\mathbf{Y}|}$
4: for each $V_i, V_j \in \mathbf{X} \cup \mathbf{Y}$ do
5:     $MI_{ij} \leftarrow$ estimateMutualInformation$(V_i, V_j, \mathcal{D})$
6: $\mathbf{MI}_{\mathbf{X},\mathbf{X}} \leftarrow \mathbf{0}$                         ▷ label conditional independence of the $\mathbf{X}$s
7: $\mathbf{MI}_{\mathbf{Y},\mathbf{Y}} \leftarrow \mathbf{MI}_{\mathbf{Y},\mathbf{Y}} + \max(\mathbf{MI})$      ▷ forcing dependencies among the $\mathbf{Y}$s
8: $T \leftarrow$ maximumSpanningTree$(\mathbf{MI})$
9: $\mathcal{T} \leftarrow$ traverseTree$(T)$
10: $\boldsymbol{\theta} \leftarrow$ computeFactors$(\mathcal{D}, \mathcal{T})$
11: return $\langle \mathcal{T}, \boldsymbol{\theta} \rangle$
```

classifier performance. It is well known in the literature that different metrics focus on different dependency relationships among the label variables, and are better optimized by taking into account those dependencies (Dembczyński et al., 2012). We employ three different metrics to assess the overall performance of CNets and the improvement gained by RCNets. We want to prove that even a single CNet can be competitive against more sophisticated models and that the proposed heuristics for RCNets are indeed meaningful for MLC. Namely, the three employed metrics are:

$$\begin{aligned}
\text{HAMMING SCORE} &= \frac{1}{NL}\sum_{i=1}^{N}\sum_{j=1}^{L}\mathbb{I}(y_j^i = \hat{y}_j^i),\\
\text{EXACT MATCH} &= \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(\mathbf{y}^i = \hat{\mathbf{y}}^i),\\
\text{ACCURACY} &= \frac{1}{N}\sum_{i=1}^{N}\frac{|\mathbf{y}^i \wedge \hat{\mathbf{y}}^i|}{|\mathbf{y}^i \vee \hat{\mathbf{y}}^i|},
\end{aligned}$$

where $\mathbb{I}(C)$ is the indicator function, while $\wedge$ and $\vee$ are the bitwise logical AND and OR operations, respectively (Madjarov et al., 2012).

The EXACT MATCH measure, or 0/1 LOSS as a loss measure, computes the percentage of instances whose predicted set of labels $\hat{\mathbf{y}}$ matches the true set of labels $\mathbf{y}$ *exactly*. On the other side, the HAMMING SCORE rewards methods for predicting individual labels well. While the EXACT MATCH is a very strict evaluation measure, the HAMMING SCORE tends to be very lenient. Finally, ACCURACY is a label set-based measure defined by the Jaccard similarity coefficients between the predicted and true set of labels.

4.2 Datasets

A set of 10 numerical traditional multi-label datasets (accessible from the MULAN[3], MEKA[4], and LABIC[5] websites) has been used in the experiments. These datasets belong to a wide variety of ap-

3. `http://mulan.sourceforge.net/`.
4. `http://meka.sourceforge.net/`.
5. `http://computer.njnu.edu.cn/Lab/LABIC/LABIC_Software.html`.

	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: Dataset descriptions: number of attributes (M), instances (N), and labels (L).

plication domains and their labels range from 6 to 174, while the number of attributes ranges from 19 to 500, and the number of examples ranges 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*: $LDist(\mathcal{D}) = |\{\mathbf{y}|\exists(\mathbf{x},\mathbf{y}) \in \mathcal{D}\}|$.

Since our current implementation of CNets works on binary data we discretized all the numeric features for each dataset implementing the Label-Attribute Interdependence Maximization (LAIM) (Cano et al., 2016) discretization method for multi-label data. We run all the algorithms in these experiments on the datasets preprocessed by LAIM.

4.3 Algorithms and setup

For the experimental evaluation, we compared both CNet and RCNet implementations, `CNET` and `RCNET`[6], to different algorithms. Both `CNET` (obtained by dropping lines 6-7 in Algorithm 1) and `RCNET` limit OR nodes on $\mathbf{X}$ variables, while in addition `RCNET` modify the $\mathbf{MI}$ matrix in order to fulfill the (in)dependency constraints. Other algorithms have been run using their openly available implementations in MEKA[7] (release 1.9.0). First, due to the lack of openly available implementations of BNs structure learning algorithms for MLC, we include in the comparison the Bayesian Chain Classifier (`BCC`) (Zaragoza et al., 2011), which builds a probabilistic CC after learning the structure of a BN for MLC. As reported in (Zaragoza et al., 2011), BCC is highly competitive against other Bayesian multi-dimensional classifiers such as those reported in (Waal and Gaag, 2007; Bielza et al., 2011). Then, even if they are not directly comparable to CNets, as they are ensembles as problem transformation methods, we included both binary relevance method (`BR`) and the classifier chains method (`CC`). As base classifiers we used Naïve Bayes (`NB`) and Tree Augmented Naïve Bayes (`TAN`), leading respectively to six different multi-label classifier variants: `BRNB`, `CCNB`, `BRTAN` and `CCTAN`. Parameters were set as suggested by the MEKA software[8].

6. Both have be run with `-d 0.6`, leaving all the other parameters set to default value.
7. Available at `http://meka.sourceforge.net`.
8. `TAN` with `SimpleEstimator` with `alpha=1`, and `scoreType = BAYES`. `BCC` with `NaiveBayes` as classifier and a default value for `dependencyType`. All the scripts for reproduce the MEKA results reported in this paper are available at `https://github.com/nicoladimauro/dcsn`. All experiments have been run on a 4-core Intel Xeon E312xx (Sandy Bridge) @2.0 GHz with 8Gb of RAM and Linux kernel 3.13.0-39.

Dataset	RCNET	CCTAN	BRTAN	CNET	CCNB	BRNB	BCC
Arts	0.432 (6)	0.372 (4)	0.391 (3)	0.399 (2)	0.376 (4)	0.358 (7)	0.331 (5)
Business	0.728 (1)	0.703 (4)	0.704 (3)	0.719 (2)	0.68 (6)	0.660 (7)	0.658 (5)
CAL500	0.203 (3)	0.188 (5)	0.251 (2)	0.187 (5)	0.184 (7)	0.252 (1)	0.287 (4)
Emotions	0.554 (2)	0.563 (1)	0.542 (5)	0.499 (6)	0.553 (3)	0.512 (6)	0.475 (4)
Flags	0.563 (2)	0.565 (1)	0.544 (4)	0.526 (6)	0.542 (5)	0.545 (3)	0.550 (6)
Health	0.603 (3)	0.611 (1)	0.610 (2)	0.587 (4)	0.578 (6)	0.582 (5)	0.521 (6)
Human	0.361 (1)	0.301 (3)	0.249 (5)	0.306 (2)	0.258 (4)	0.197 (7)	0.265 (6)
Plant	0.397 (1)	0.313 (3)	0.308 (4)	0.354 (2)	0.298 (5)	0.278 (7)	0.229 (6)
Scene	0.669 (1)	0.658 (2)	0.634 (3)	0.530 (4)	0.530 (4)	0.528 (6)	0.563 (7)
Yeast	0.464 (2)	0.452 (3)	0.479 (1)	0.442 (4)	0.404 (7)	0.429 (5)	0.453 (6)
Avrg rank	**1.7**	**2.8**	**3.2**	**3.7**	**5.1**	**5.4**	**5.6**

Table 2: The performance of the multi-label learning approaches in terms of the ACCURACY measure. In parenthesis the rank of the algorithm for a given dataset.

Dataset	RCNET	BRTAN	CCTAN	CNET	BRNB	BCC	CCNB
Arts	0.937 (2)	0.931 (3)	0.940 (1)	0.937 (2)	0.926 (4)	0.936 (5)	0.923 (6)
Business	0.973 (1)	0.965 (2)	0.965 (2)	0.974 (1)	0.957 (4)	0.969 (5)	0.942 (6)
CAL500	0.854 (1)	0.814 (4)	0.823 (3)	0.854 (1)	0.813 (5)	0.833 (2)	0.684 (6)
Emotions	0.783 (4)	0.797 (1)	0.786 (3)	0.737 (6)	0.787 (2)	0.760 (6)	0.768 (5)
Flags	0.709 (4)	0.710 (3)	0.709 (4)	0.680 (6)	0.723 (1)	0.716 (2)	0.708 (6)
Health	0.965 (1)	0.963 (3)	0.964 (2)	0.964 (2)	0.959 (4)	0.956 (5)	0.955 (6)
Human	0.894 (1)	0.890 (2)	0.870 (4)	0.894 (1)	0.884 (3)	0.891 (6)	0.827 (5)
Plant	0.892 (3)	0.894 (2)	0.901 (1)	0.888 (5)	0.890 (4)	0.862 (5)	0.884 (5)
Scene	0.879 (3)	0.898 (1)	0.888 (2)	0.838 (5)	0.874 (4)	0.867 (6)	0.816 (5)
Yeast	0.773 (1)	0.765 (2)	0.749 (3)	0.765 (2)	0.732 (4)	0.754 (5)	0.720 (6)
Avrg rank	**2.2**	**2.8**	**2.9**	**3.1**	**4.1**	**5.5**	**6.4**

Table 3: The performance of the multi-label learning approaches in terms of the HAMMING SCORE measure. In parenthesis the rank of the algorithm for a given dataset.

.

4.4 Results and discussion

The results for each evaluation metric and for each classification algorithm on all the datasets are reported in Table 2, 3 and 4. RCNET outperforms the other methods in 10 (CCNB), 9 (BRNB and BCC), and 7 (CCTAN and BRTAN) cases in terms of the ACCURACY measure (Table 2). In terms of the HAMMING SCORE measure, RCNET is superior to the others in 10 (CCNB and BCC), 8 (BRNB), 6 (BRTAN) and 5 (CCTAN) cases (Table 3). Regarding the EXACT MATCH measure, RCNET is ranked better in 9 (BRNB, CCNB and BCC), 8 (BRTAN), 6 (CCTAN) cases (Table 4). By looking at the average ranks, even CNET consistently outperforms BCC, BRNB and CCNB for all the three measures. We can see from these results that even if RCNET learns a single model, it is competitive to other approaches. It performs better that other methods in terms of the ACCURACY and EXACT MATCH measure.

Dataset	RCNET	CCTAN	CNET	BRTAN	CCNB	BRNB	BCC
Arts	0.313 (1)	0.275 (2)	0.302 (1)	0.247 (4)	0.251 (3)	0.234 (6)	0.187 (5)
Business	0.567 (1)	0.548 (2)	0.565 (1)	0.545 (3)	0.520 (5)	0.518 (6)	0.410 (4)
CAL500	0.000 (1)	0.000 (1)	0.000 (1)	0.000 (1)	0.000 (1)	0.000 (1)	0.000 (1)
Emotions	0.295 (1)	0.282 (3)	0.260 (1)	0.295 (1)	0.265 (5)	0.280 (4)	0.207 (6)
Flags	0.170 (2)	0.212 (1)	0.185 (1)	0.108 (6)	0.129 (3)	0.114 (5)	0.109 (3)
Health	0.456 (2)	0.478 (1)	0.448 (1)	0.437 (3)	0.407 (6)	0.425 (4)	0.247 (5)
Human	0.263 (1)	0.139 (3)	0.260 (1)	0.141 (2)	0.085 (5)	0.100 (4)	0.150 (6)
Plant	0.344 (1)	0.247 (2)	0.333 (1)	0.207 (3)	0.185 (4)	0.181 (5)	0.110 (6)
Scene	0.554 (1)	0.498 (3)	0.492 (1)	0.518 (2)	0.248 (5)	0.375 (4)	0.422 (6)
Yeast	0.147 (2)	0.169 (1)	0.129 (1)	0.120 (3)	0.117 (4)	0.101 (5)	0.074 (6)
Avrg rank	**1.4**	**2.3**	**2.7**	**3.5**	**4.9**	**5.2**	**5.6**

Table 4: The performance of the multi-label learning approaches in terms of the EXACT MATCH measure. In parenthesis the rank of the algorithm for a given dataset.

.

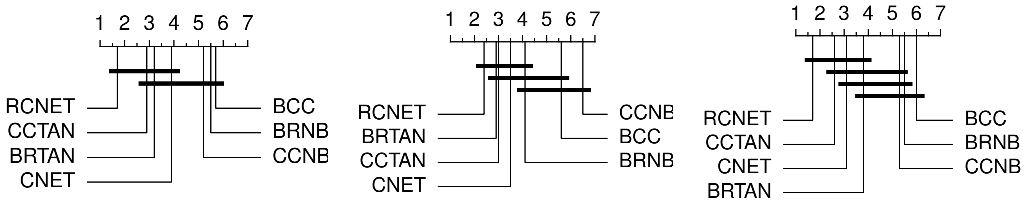

Figure 2: The critical diagrams for the ACCURACY (left), HAMMING SCORE (center) and EXACT MATCH (right) evaluation measures: results from the Nemenyi post-hoc test at p=0.05 significance level.

In order to assess whether the overall differences in performance are statistically significant a post-hoc Nemenyi test (Demšar, 2006) has been conducted comparing all the classifiers to each others. With a Nemenyi test the performances of two classifiers are significantly different if their average ranks differ by more than some critical distance (CD), obtained by taking into account the number of algorithms, the number of datasets and a critical value for a given significance level p. Figure 2 reports the results from the Nemenyi post-hoc test with average rank diagrams, where the average ranks of the algorithms are drawn on an enumerated axis. The best ranking algorithms are at the left-most side of the diagram, and the horizontal bold lines connect the vertical lines for the average ranks of the algorithms that do not differ significantly. As we can see, the differences are not statistically different at $p = 0.05$ significance level when compared to those obtained by the methods BRTAN and CCTAN. However, note that RCNETs are single models and as such they could be incorporated into sophisticated problem transformation schemes as CC and BR. Devising a way to blend CNet structure learning into these schemes is an interesting future line of work.

5. Conclusions

In this paper, we tackled the MLC problem by employing CNets, a recently proposed tractable probabilistic model for representing multi-dimensional discrete distributions. We exploited CNets to efficiently and exactly solve an MPE formulation of MLC. We devised a structure learning algorithmic variant to cope with the prominence of label dependencies, a crucial issue in MLC. The experimental evaluation on 10 real-world datasets showed how our approach can effectively improve the accuracy, exact and hamming scores, proving itself to be highly competitive against more complex ensemble approaches.

Acknowledgments

Work partially supported by the European Commission through the project MAESTRA - Learning from Massive, Incompletely annotated, and Structured Data (Grant number ICT-2013-612944), and by the project PUGLIA@SERVICE (PON02 00563 3489339) financed by the Italian Ministry of University and Research (MIUR). Experiments executed on the resources made available by two projects financed by the MIUR: ReCaS (PONa3_00052) and PRISMA (PON04_a2A).

References

A. Antonucci, G. Corani, D. D. Mauá, and S. Gabaglio. An ensemble of bayesian networks for multilabel classification. In *Proceedings of IJCAI*, pages 1220–1225, 2013.

C. Bielza, G. Li, and P. Larrañaga. Multi-dimensional classification with bayesian networks. *International Journal of Approximate Reasoning*, 52(6):705–727, 2011.

M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

A. Cano, J. M. Luna, E. L. Gibaja, and S. Ventura. LAIM discretization for multi-label data. *Information Sciences*, 330:370–384, 2016.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

G. Corani, A. Antonucci, D. D. Mauá, and S. Gabaglio. Trading off speed and accuracy in multilabel classification. In *Proceedings of PGM*, pages 145–159. Springer, 2014.

K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier. On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88(1):5–45, 2012.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

N. Di Mauro, A. Vergari, and T. M. Basile. Learning bayesian random cutset forests. In *Proceedings of ISMIS*, pages 122–132. Springer, 2015a.

N. Di Mauro, A. Vergari, and F. Esposito. Learning accurate cutset networks by exploiting decomposability. In *Proceedings of AIXIA*, pages 221–232. Springer, 2015b.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine learning*, 29 (2-3):131–163, 1997.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Deroski. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45(9):3084–3104, 2012.

M. Meilă and M. I. Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48, 2000.

H. Poon and P. Domingos. Sum-Product Network: a New Deep Architecture. *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

T. Rahman, P. Kothalkar, and V. Gogate. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of Chow-Liu trees. In *Machine Learning and Knowledge Discovery in Databases*, volume 8725 of *LNCS*, pages 630–645. Springer, 2014.

J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, pages 254–269. Springer-Verlag, 2009.

J. D. Rodríguez and J. A. Lozano. Multi-objective learning of multi-dimensional Bayesian classifiers. In *Proceedings of the Eighth International Conference on Hybrid Intelligent Systems*, pages 501–506, 2008.

G. Tsoumakas, I. Katakis, and I. P. Vlahavas. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook, 2nd ed.*, pages 667–685. Springer, 2010.

A. Vergari, N. Di Mauro, and F. Esposito. Simplifying, Regularizing and Strengthening Sum-Product Network Structure Learning. In *ECML-PKDD 2015*, 2015.

P. R. Waal and L. C. Gaag. Inference and learning in multi-dimensional bayesian network classifiers. In *9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 501–511. Springer, 2007.

J. H. Zaragoza, L. E. Sucar, E. F. Morales, C. Bielza, and P. Larrañaga. Bayesian chain classifiers for multidimensional classification. In *Proceedings of the 22nd IJCAI*. AAAI Press, 2011.