A Mixtures-of-Trees Framework for Multi-Label Classification

Charmgil Hong
Computer Science Dept.
University of Pittsburgh
Pittsburgh, PA, USA
charmgil@cs.pitt.edu

Iyad Batal
GE Global Research
San Ramon, CA, USA
iyad.batal@ge.com

Milos Hauskrecht
Computer Science Dept.
University of Pittsburgh
Pittsburgh, PA, USA
milos@cs.pitt.edu

ABSTRACT

We propose a new probabilistic approach for multi-label classification that aims to represent the class posterior distribution $P(\mathbf{Y}|\mathbf{X})$. Our approach uses a mixture of tree-structured Bayesian networks, which can leverage the computational advantages of conditional tree-structured models and the abilities of mixtures to compensate for tree-structured restrictions. We develop algorithms for learning the model from data and for performing multi-label predictions using the learned model. Experiments on multiple datasets demonstrate that our approach outperforms several state-of-the-art multi-label classification methods.

Categories and Subject Descriptors

I.2.6 [**LEARNING**]: General

Keywords

Multi-label classification, Bayesian network, Mixture of trees

1. INTRODUCTION

In many real-world applications, a data instance is naturally associated with multiple class labels. For example, a document can cover multiple topics [21, 42], an image can be annotated with multiple tags [6, 29] and a single gene may be associated with several functional classes [9, 42]. Multi-label classification (MLC) formulates such situations by assuming each data instance is associated with a subset of d labels. Alternatively, this problem can be defined by associating each instance with d binary class variables $Y_1, ...Y_d$, where Y_i denotes whether or not the i-th label is present in the instance. The goal is to learn a function that assigns to each instance, represented by a feature vector $\mathbf{x} = (x_1, ..., x_m)$, the most probable assignment of the class variables $\mathbf{y} = (y_1, ..., y_d)$. However, learning of such a function can be very challenging because the number of possible label configurations is exponential in d.

CIKM'14, November 3–7, 2014, Shanghai, China.

ACM 978-1-4503-2598-1/14/11 ...$15.00.
http://dx.doi.org/10.1145/2661829.2661989.

A simple solution to the above problem is to assume that all class variables are conditionally independent of each other and learn d functions to predict each class separately [9, 6]. However, this may not suffice for many real-world problems where dependences among output variables exist. To overcome this limitation, multiple machine learning methods that model class relations have been proposed in recent years. These include two-layer classification models [14, 8], classifier chains [31, 41, 10], output coding methods [18, 34, 44, 45] and multi-dimensional Bayesian network classifiers [38, 5, 1].

In this work, we develop and study a new probabilistic approach for modeling and learning an MLC. Our approach aims to represent the class posterior distribution $P(Y_1, ..., Y_d|\mathbf{X})$ such that it captures multivariate dependences among features and labels. Our proposed model is defined by a mixture of Conditional Tree-structured Bayesian Networks (CTBNs) [2]. A CTBN defines $P(Y_1, ..., Y_d|\mathbf{X})$ using a directed tree structure to model the relations among the class variables conditioned on the feature variables. The main advantage of CTBN is that it allows efficient learning and inference. A mixture of CTBNs leverages the computational advantages of CTBNs and the ability of a mixture to compensate for the tree-structure restriction.

Our new mixture model extends the work by [26] that models and learns the joint distribution over many variables using tree-structured distributions and their mixtures, to learn conditional distributions where the multivariate relations among $\mathbf{Y}$ components are conditioned on inputs $\mathbf{X}$. To support learning and inference in the new model, we develop and test new algorithms for: (1) learning the parameters of conditional trees mixtures, (2) selecting individual tree structures and (3) inferring the maximum a posteriori (MAP) output label configurations.

An important advantage of our method compared to existing MLC methods is that it gives a well-defined model of posterior class probabilities. That is, our model lets us calculate $P(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{x})$ for any $(\mathbf{x}, \mathbf{y})$ input-output pair. This is extremely useful not only for prediction, but also for decision making [30, 3], conditional outlier analysis [15, 16, 17], or for performing any inference over subsets of output class variables. In contrast to our approach, the majority of existing MLC methods aim to only identify the best output configuration for the given $\mathbf{x}$.

2. PROBLEM DEFINITION

In Multi-Label Classification (MLC), each instance is associated with d binary class variables $Y_1, ...Y_d$. We are given

labeled training data $D = \{\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^{N}$, where $\mathbf{x}^{(n)} = (x_1^{(n)}, ..., x_m^{(n)})$ is a m-dimensional feature vector representing the n-th instance (the input) and $\mathbf{y}^{(n)} = (y_1^{(n)}, ..., y_d^{(n)})$ is its corresponding d-dimensional class vector (the output). We want to learn a function h (from D) that assigns to each instance, represented by its feature vector, a class vector:

$$h : \mathbb{R}^m \rightarrow \{0, 1\}^d$$

One way to approach this task is to model and learn the *conditional joint distribution* $P(\mathbf{Y}|\mathbf{X})$, where $\mathbf{Y} = (Y_1, ..., Y_d)$ is a random variable for the class vector and $\mathbf{X}$ is a random variable for the feature vector. Assuming the 0-1 loss function, the optimal classifier h^* assigns to each instance $\mathbf{x}$ the maximum a posteriori (MAP) assignment of class variables:

$$\begin{aligned} h^*(\mathbf{x}) &= \arg\max_{\mathbf{y}} P(\mathbf{Y}=\mathbf{y}|\mathbf{X}=\mathbf{x}) \\ &= \arg\max_{y_1,...,y_d} P(Y_1=y_1, ..., Y_d=y_d|\mathbf{X}=\mathbf{x}) \end{aligned} \quad (1)$$

A key challenge for modeling and learning $P(\mathbf{Y}|\mathbf{X})$ from data, as well as for defining the corresponding MAP classifier, is that the number of all possible class assignments one has to consider is 2^d. The goal of this paper is to develop a new, efficient model and methods for its learning and inference that overcome this difficulty.

Notation: For notational convenience, we will omit the index superscript $^{(n)}$ when it is not necessary. We may also abbreviate the expressions by omitting variable names; e.g., $P(Y_1 = y_1, ..., Y_d = y_d|\mathbf{X}=\mathbf{x}) = P(y_1, ..., y_d|\mathbf{x})$.

3. RELATED RESEARCH

In this section, we briefly review the research work related to our approach and pinpoint the main differences.

MLC method based on learning independent classifiers was studied by [9, 6]. Zhang and Zhou [43] presented a multi-label k-nearest neighbor method, which learns a classifier for each class by combining k-nearest neighbor with Bayesian inference. To model possible class dependences, [14, 8] proposed adding a second layer of classifiers that combine input features with the outputs of independent classifiers. The limitation of these early approaches is that class dependences are either not modeled at all, or modeled in a very limited way.

The classifier chains (CC) method [31] models the class posterior distribution $P(\mathbf{Y}|\mathbf{X})$ by decomposing the relations among class variables using the chain rule:

$$P(Y_1, ..., Y_d|\mathbf{X}) = \prod_{i=1}^{d} P(Y_i|\mathbf{X}, Y_1, ..., Y_{i-1}) \quad (2)$$

Each component in the chain is a classifier that is learned separately by incorporating the predictions of preceding classifiers as additional features. Zhang and Zhang [41] realized that the performance of CC is influenced by the order of classes in the chain and presented a method to learn such ordering from data. Dembczynski et al. [10] discussed the suboptimality of CC and presented probabilistic classifier chains to estimate the entire posterior distribution of classes. However, this method has to evaluate exponentially many label configurations, which greatly limits its applicability.

Another approach for modeling $P(\mathbf{Y}|\mathbf{X})$ relies on conditional random fields (CRFs) [24]. Ghamrawi and McCallum [13] presented a method called collective multi-label with features classifier (CMLF) that captures label co-occurrences conditioned on features. However, CMLF assumes a fully connected CRF structure which results in a high computational cost. Later, Shahaf et al. [32] and Bradley et al. [7] proposed to learn tractable (low-treewidth) structures of class variables for CRFs using conditional mutual information. More recently, Pakdaman et al. [28] used pairwise CRFs to model the class dependences and presented L_2-optimization-based structure and parameter learning algorithms. Although the later methods share similarities with our approach by modeling the conditional dependences in $\mathbf{Y}$ space using restricted structures, their optimization of the likelihood of data is computationally more costly. To alleviate this, CRF-based methods often resort to optimization of a surrogate objective function (e.g., the pseudo-likelihood of data [28]) or include specific assumptions (e.g., features are assumed to be discrete [13]; relevant features for each class are assumed to be known [32, 7]), which complicate the application of the methods.

Multi-dimensional Bayesian network classifiers (MBC) [38, 5, 1] build a generative model of $P(\mathbf{X}, \mathbf{Y})$ using special Bayesian network structures that assume all class variables are top nodes and all feature variables are their descendants. Although our approach can be compared to MBC, there are significant differences and advantages: (1) MBC only handles discrete features and, thus, all features should be a priori discretized; while we handle both continuous and discrete features. (2) MBC defines a joint distribution over both feature and class variables and the search space of the model increases with the input dimensionality m; while our search space does not depend on m. (3) Feature selection in MBC is done explicitly by learning the individual relationships between features and class variables; while we perform feature selection by regularizing the base classifiers. (4) MBC requires expensive marginalization to obtain class conditional distribution $P(\mathbf{Y}|\mathbf{X})$; while we directly estimate $P(\mathbf{Y}|\mathbf{X})$.

An alternative approach for MLC is based on output coding. The idea is to compress the output into a codeword, learn how to predict the codeword and then recover the correct output from the noisy predictions. A variety of approaches have been devised by using different compression techniques, such as compressed sensing [18], principal component analysis [34] and canonical correlation analysis [44]. The state-of-the-art in output coding utilizes a maximum margin formulation [45] that promotes both discriminative and predictable codes. The limitation of output coding methods is that they can only predict the single "best" output for a given input, and they cannot compute probabilities for different input-output pairs.

Several researchers proposed using ensemble methods for MLC. Read et al. [31] presented a simple method that averages the predictions of multiple random classifier chains trained on a random subset of the data. Antonucci et al. [1] proposed an ensemble of multi-dimensional Bayesian networks combined via simple averaging. These networks represent different $\mathbf{Y}$ relations (the structures are set a priori and not learned) and all of the networks adopt the naïve Bayes assumption (the features are independent given the classes). Unlike these methods, our approach learns the structures in the mixture, its parameters and mixing coefficients from data in a principled way.

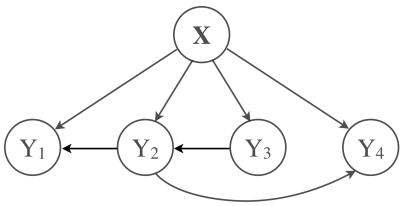


Figure 1: An example CTBN

4. PRELIMINARY

The MLC solution we propose in this work combines multiple base MLC classifiers using the *mixtures-of-trees* (MT) [26, 39] framework, which uses a mixture of multiple trees to define a generative model of $P(\mathbf{Y})$ for discrete multi-dimensional domains. The base classifiers we use are based on the *conditional tree-structured Bayesian networks* (CTBN) [2]. To begin with, we briefly review the basics of MT and CTBN.

MT consists of a set of *trees* that are combined using *mixture coefficients* λ_k to represent the joint distribution $P(\mathbf{y})$. The model is defined by the following decomposition:

$$P(\mathbf{y}) = \sum_{k=1}^{K} \lambda_k P(\mathbf{y}|T_k), \tag{3}$$

where $P(\mathbf{y}|T_k)$ are called *mixture components* that represent the distribution of outputs defined by the k-th tree T_k. Note that a mixture can be understood as a soft-multiplexer, where we have a hidden selector variable which takes a value $k \in \{1, ..., K\}$ with probability λ_k. That is, by having a convex combination of mutually complementary tree-structured models, MT aims at achieving a more expressive and accurate model.

While MT is not as computationally efficient as individual trees, it has been considered as a useful approximation at a fraction of the computational cost learning general graphical models [22]. MT has been successfully adopted in a range of applications, including modeling of handwriting patterns, medical diagnostic network, automated application screening, gene classification and identification [26], face detection [20], video tracking [19], road traffic modeling [39] and climate modeling [22].

In this work, we apply the MT framework in context of MLC. In particular, we combine MT with CTBN to model individual trees. CTBN is a recently proposed probabilistic MLC method that has been shown to be competitive and efficient on a range of domains. CTBN defines $P(\mathbf{Y}|\mathbf{X})$ using a collection of classifiers modeling relations in between features and individual labels that are tied together using a special Bayesian network structure that approximates the dependence relations among the class variables. In modeling of the dependences, it allows each class variable to have at most one other class variable as a parent (without creating a cycle) besides the feature vector $\mathbf{X}$.

A CTBN T defines the joint distribution of class vector $(y_1, ..., y_d)$ conditioned on feature vector $\mathbf{x}$ as:

$$P(y_1, ..., y_d|\mathbf{x}, T) = \prod_{i=1}^{d} P(y_i|\mathbf{x}, y_{\pi(i,T)}), \tag{4}$$

where $\pi(i, T)$ denotes the parent class of class Y_i in T (by convention, $\pi(i, T) = \{\}$ if Y_i does not have a parent class).

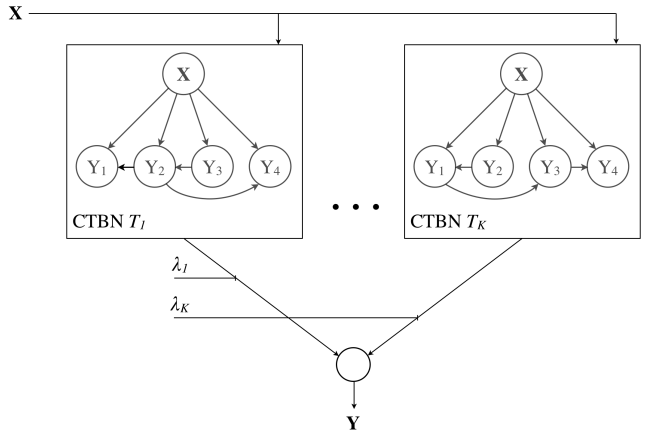


Figure 2: An example MC

For example, the conditional joint distribution of class assignment (y_1, y_2, y_3, y_4) given $\mathbf{x}$ according to the network T in Figure 1 is defined as:

$$P(y_1, y_2, y_3, y_4|\mathbf{x}, T) = P(y_3|\mathbf{x}) \cdot P(y_2|\mathbf{x}, y_3) \cdot P(y_1|\mathbf{x}, y_2) \cdot P(y_4|\mathbf{x}, y_2)$$

Although our proposed method is motivated by MT, there are significant extensions and differences. We summarize the key distinctions below.

1. *Model*: Our model represents $P(\mathbf{Y}|\mathbf{X})$, the class posterior distribution for MLC, using CTBNs that each consists of a collection of logistic regression models, linked together by a directed tree; on the other hand, the MT model [26] represents the joint distribution $P(\mathbf{Y})$ using standard tree-structured Bayesian networks.

2. *Structure learning*: Our structure learning algorithm optimizes $P(\mathbf{Y}|\mathbf{X})$ using weighted conditional log-likelihood criterion; while MT relies on the standard Chow-Liu algorithm [23] that optimizes $P(\mathbf{Y})$ using mutual information.

3. *Parameter learning*: Not surprisingly, both our parameter learning method and that of MT rely on the EM algorithm. However, the criteria and how to optimize them are very different. For example, the M-step of our algorithm corresponds to learning of instance-weighted logistic regression classifiers; while that of MT is based on simple (weighted) counting.

5. OUR METHOD

In this section, we describe *Mixture of Conditional Tree-structured Bayesian Networks* (MC), which uses the MT framework in combination with the CTBN classifiers to improve the classification accuracy of MLC tasks, and develop algorithms for its learning and predictions. In section 5.1, we describe the mixture defined by the MC model. In section 5.2 through 5.4, we present the learning and prediction algorithms for the MC model.

5.1 Representation

By following the definition of MT in Equation (3), MC defines the multivariate posterior distribution of class vector

$\mathbf{y} = (y_1, ..., y_d)$ as:

$$P(\mathbf{y}|\mathbf{x}) = \sum_{k=1}^{K} \lambda_k P(\mathbf{y}|\mathbf{x}, T_k), \tag{5}$$

where $\lambda_k \geq 0, \forall k$; and $\sum_{k=1}^{K} \lambda_k = 1$. Here each *mixture component* $P(\mathbf{y}|\mathbf{x}, T_k)$ is the distribution defined by CTBN T_k (as in Equation (4)) and *mixture coefficients* are denoted by λ_k. Figure 2 depicts an example MC model, which consists of K CTBNs and the mixture coefficients λ_k.

5.2 Parameter Learning

In this section, we describe how to learn the parameters of MC by assuming the structures of individual CTBNs are known and fixed. The parameters of the MC model are the mixture coefficients $\{\lambda_1, ..., \lambda_K\}$ as well as the parameters of each CTBN in the mixture $\{\theta_1, ..., \theta_K\}$.

Given training data $D = \{\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\} : n \in 1, ..., N$, the objective is to optimize the log-likelihood of D, which we refer to as the *observed log-likelihood*.

$$\sum_{n=1}^{N} \log P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \lambda_k P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, T_k)$$

However, this is very difficult to directly optimize because it contains the log of the sum. Hence, we cast this optimization in the expectation-maximization (EM) framework. Let us associate each instance $(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})$ with a hidden variable $z^{(n)} \in \{1, ..., K\}$ indicating which CTBN it belongs. The *complete log-likelihood* (assuming $z^{(n)}$ are observed) is:

$$\sum_{n=1}^{N} \log P(\mathbf{y}^{(n)}, z^{(n)}|\mathbf{x}^{(n)}) \tag{6}$$

$$= \sum_{n=1}^{N} \log \prod_{k=1}^{K} P\left(\mathbf{y}^{(n)}, T_k|\mathbf{x}^{(n)}\right)^{\mathbb{1}[z^{(n)}=k]} \tag{7}$$

$$= \sum_{n=1}^{N} \log \prod_{k=1}^{K} \left[\lambda_k P\left(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, T_k\right)\right]^{\mathbb{1}[z^{(n)}=k]}$$

$$= \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{1}[z^{(n)} = k] \left[\log \lambda_k + \log P\left(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, T_k\right)\right],$$

where $\mathbb{1}[z^{(n)} = k]$ is the indicator function, which is one if the n-th instance belongs to the k-th CTBN and zero otherwise; and λ_k is the mixture coefficient of CTBN T_k, which can be interpreted as its prior probability in the data.

The EM algorithm iteratively optimizes the *expected complete log-likelihood*, which is always a lower bound to the observed log-likelihood [27]. In the *E-step*, the expectation is computed with the current set of parameters; in the *M-step*, the parameters of the mixture $(\lambda_k, \theta_k : k = \{1, ..., K\})$ are relearned to maximize the expected complete log-likelihood. In the following, we describe our parameter learning algorithm by deriving the E-step and the M-step for MC.

5.2.1 E-step

In the E-step, we compute the expectation of the hidden variables. Let $\gamma_k(n)$ denote $P(z^{(n)} = k|\mathbf{y}^{(n)}, \mathbf{x}^{(n)})$, the posterior of the hidden variable $z^{(n)}$ given the observations and the current parameters. Using Bayes rule, we write:

$$\gamma_k(n) = \frac{\lambda_k P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, T_k)}{\sum_{k'} \lambda_{k'} P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, T_{k'})} \tag{8}$$

5.2.2 M-step

In the M-step, we learn the model parameters $\{\lambda_1, ..., \lambda_K, \theta_1, ..., \theta_K\}$ that maximize the expected complete log-likelihood, which is a lower bound of the observed log-likelihood. Let us first define the following two quantities:

$$\Gamma_k = \sum_{n=1}^{N} \gamma_k(n), \qquad w_k(n) = \frac{\gamma_k(n)}{\Gamma_k}$$

Γ_k can be interpreted as the number of observations that belongs to the k-th CTBN (hence, $\sum_{k=1}^{K} \Gamma_k = N$), and $w_k(n)$ is the renormalized posterior $\gamma_k(n)$, which can be interpreted as the weight of the n-th instance on the k-th CTBN.

Note that when taking the expectation of the complete log-likelihood (Equation (6)), only the indicator $\mathbb{1}[z^{(n)} = k]$ is affected by the expectation. By using the notations introduced above, we rewrite the expected complete log-likelihood:

$$\sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_k(n) \left[\log \lambda_k + \log P\left(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, T_k\right)\right]$$

$$= \sum_{k=1}^{K} \Gamma_k \log \lambda_k + \sum_{k=1}^{K} \Gamma_k \sum_{n=1}^{N} w_k(n) \log P\left(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, T_k\right) \tag{9}$$

We wish to maximize (9) with respect to $\{\lambda_1, ..., \lambda_K, \theta_1, ..., \theta_K\}$ subject to the constraint $\sum_{k=1}^{K} \lambda_k = 1$. Notice that (9) consists of two terms and each term has a disjoint subset of parameters – which allows us to maximize (9) term by term. By maximizing the first term with respect to λ_j (the mixture coefficient of T_j), we obtain:

$$\lambda_j = \frac{\Gamma_j}{\sum_{k=1}^{K} \Gamma_k} = \frac{\Gamma_j}{N}$$

To maximize the second term, we train θ_j (the parameters of T_j) to maximize:

$$\theta_j = \arg\max \sum_{n=1}^{N} w_j(n) \log P(y^{(n)}|x^{(n)}, T_j) \tag{10}$$

It turns out (10) is the instance-weighted log-likelihood, and we use instance-weighted logistic regression to optimize it. Algorithm 1 outlines our parameter learning algorithm.

5.2.3 Complexity

E-step: We compute $\gamma_k(n)$ for each instance on every CTBN. To compute $\gamma_k(n)$, we should estimate $P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, T_k)$, which requires applying the logistic regression classifiers for each node of T_k, which requires $O(md)$ multiplications. Hence, the complexity of the E-step is $O(KNmd)$.

M-step: The major computational cost of the M-step is to learn the instance-weighted logistic regression models for the nodes of every CTBN. Hence, the complexity is $O(Kd)$ times the complexity of learning logistic regression.

5.3 Structure Learning

In this section, we describe how to automatically learn multiple CTBN structures from data. We apply a sequential

Algorithm 1 learn-MC-parameters

Input: Training data D; base CTBNs $T_1, ..., T_K$
Output: Model parameters $\{\theta_1, ..., \theta_K, \lambda_1, ..., \lambda_K\}$

repeat
  E-step:
  for $k = 1$ **to** K, $n = 1$ **to** N **do**
    Compute $\gamma_k(n)$ using Equation (8)
  end for
  M-step:
  for $k = 1$ **to** K **do**
    $\Gamma_k = \sum_{n=1}^{N} \gamma_k(n)$
    $w_k(n) = \gamma_k(n)/\Gamma_k$
    $\lambda_k = \Gamma_k/N$
    $\theta_k = \arg\max \sum_{n=1}^{N} w_k(n) \log P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, T_k)$
  end for
until *convergence*

boosting-like heuristic, where in each iteration we learn the structure that focuses on the instances that are not well predicted by the previous structures (i.e., the MC model learned so far). In the following, we first describe how to learn a single CTBN structure from instance-weighted data. After that, we describe how to re-weight the instances and present our algorithm for learning the overall MC model.

5.3.1 Learning a Single CTBN Structure on Weighted Data

The goal here is to discover the CTBN structure that maximizes the weighted conditional log-likelihood (WCLL) on $\{D, \Omega\}$, where $D = \{\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^{N}$ is the data and $\Omega = \{\omega^{(n)}\}_{n=1}^{N}$ is the weight for each instance. We do this by partitioning D into two parts: training data D_{tr} and hold-out data D_h. Given a CTBN structure T, we train its parameters using D_{tr} and the corresponding instance weights. On the other hand, we use WCLL of D_h to score T.

$$Score(T) = \sum_{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)}) \in D_h} \omega^{(n)} \log P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, T) \quad (11)$$

$$= \sum_{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)}) \in D_h} \sum_{i=1}^{d} \omega^{(n)} \log P(y_i^{(n)}|\mathbf{x}^{(n)}, y_{\pi(i,T)}^{(n)})$$

In the following, we describe our algorithm for obtaining the CTBN structure that optimizes Equation (11) without having to evaluate all of the exponentially many possible tree structures.

Let us first define a weighted directed graph $G = (V, E)$, which has one vertex V_i for each class label Y_i and a directed edge $E_{j \to i}$ from each vertex V_j to each vertex V_i (i.e., G is complete). In addition, each vertex V_i has a self-loop $E_{i \to i}$. The weight of edge $E_{j \to i}$, denoted as $W_{j \to i}$, is the WCLL of class Y_i conditioned on $\mathbf{X}$ and Y_j:

$$W_{j \to i} = \sum_{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)}) \in D_h} \omega^{(n)} \log P(y_i^{(n)}|\mathbf{x}^{(n)}, y_j^{(n)})$$

The weight of self-loop $E_{i \to i}$, denoted as $W_{\phi \to i}$, is the WCLL of class Y_i conditioned only on $\mathbf{X}$. Using the definition of edge weights, Equation (11) can be simplified as the sum of the edge weights:

$$Score(T) = \sum_{n=1}^{d} W_{\pi(i,T) \to i}$$

Now we have transformed the problem of finding the optimal tree structure into the problem of finding the tree in G that has the maximum sum of edge weights. The solution can be obtained by solving the maximum branching (arborescence) problem [11], which finds the maximum weight tree in a weighted directed graph.

5.3.2 Learning Multiple CTBN Structures

In order to obtain multiple CTBN structures for the MC model, we apply the algorithm described above multiple times with different sets of instance weights. We assign the weights such that we give higher weights for poorly predicted instances and lower weights for well-predicted instances.

We start with assigning all instances uniform weights (i.e., all instances are equally important a priori).

$$\omega^{(n)} = 1/N : n = 1, ..., N$$

Using this initial set of weights, we find the initial CTBN structure T_1 (and its parameters θ_1) and set the current model M to be T_1. We then estimate the prediction error margin $\omega^{(n)} = 1 - P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, M)$ for each instance and renormalize such that $\sum_{n=1}^{N} \omega^{(n)} = 1$. We use $\{\omega^{(n)}\}$ to find the next CTBN structure T_2. After that, we set the current model to be the MC model learned by mixing T_1 and T_2 according to Algorithm 1.

We repeat the process by incrementally adding trees to the mixture. To stop the process, we use internal validation approach. Specifically, the data used for learning are split to internal train and test sets. The structure of the trees and parameters are always learned on the internal train set. The quality of the current mixture is evaluated on the internal test set. The mixture growth stops when the log-likelihood on the internal test set for the new mixture is worse than for the previous mixture. The trees included in the previous mixture are then fixed, and the parameters of the mixture are relearned on the full training data.

5.3.3 Complexity

In order to learn a single CTBN structure, we compute edge weights for the complete graph G, which requires estimating $P(Y_i|\mathbf{X}, Y_j)$ for all d^2 pairs of classes. Finding the maximum branching in G can be obtained in $O(d^2)$ using [35]. To learn K CTBN structures for the mixture, we repeat these steps K times. Therefore, the overall complexity is $O(d^2)$ times the complexity of learning logistic regression.

5.4 Prediction

In order to make a prediction for a new instance $\mathbf{x}$, we want to find the MAP assignment of the class variables (see Equation (1)). In general, this requires to evaluate all possible assignments of values to d class variables, which is exponential in d.

One important advantage of the CTBN model is that the MAP inference can be done more efficiently by avoiding blind enumeration of all possible assignments. More specifically, the MAP inference on a CTBN is linear in the number of classes ($O(d)$) when implemented using a variant of the max-sum algorithm [23] on a tree structure.

However, our MC model consists of multiple CTBNs and the MAP solution may, at the end, require enumeration of exponentially many class assignments. To address this problem, we rely on approximate MAP inference. Two commonly applied MAP approximation approaches are convex

programming relaxation via dual decomposition [33], and simulated annealing using a Markov chain [40]. In this work, we use the latter approach. Briefly, we search the space of all assignments by defining a Markov chain that is induced by local changes to individual class labels. The annealed version of the exploration procedure [40] is then used to speed up the search. We initialize our MAP algorithm using the following heuristic: first, we identify the MAP assignments for each CTBN in the mixture individually, and after that, we pick the best assignment from among these candidates. We have found this (efficient) heuristic to work very well and it often results in the true MAP assignment.

6. EXPERIMENTS

We perform experiments on ten publicly available multi-label datasets. These datasets are obtained from different domains such as music recognition (emotions [36]), semantic image labeling (scene [6] and image [10]), biology (yeast [12]) and text classification (enron [4] and RCV1 [25] datasets). Table 1 summarizes the characteristics of the datasets. We show the number of instances (N), number of feature variables (m) and number of class variables (d). In addition, we show two statistics: label cardinality (LC), which is the average number of labels per instance, and distinct label set (DLS), which is the number of all distinct configurations of classes that appear in the data. Note that, for RCV1 datasets, we have used the ten most common labels.

6.1 Methods

We compare the performance of our proposed mixture-of-CTBNs (MC) model with simple binary relevance (BR) independent classification [9, 6] as well as several state-of-the-art MLC methods. These methods include classification with heterogeneous features (CHF) [14], multi-label k-nearest neighbor (MLKNN) [43], instance-based learning by logistic regression (IBLR) [8], classifier chains (CC) [31], ensemble of classifier chains (ECC) [31], probabilistic classifier chains (PCC) [10], ensemble of probabilistic classifier chains (EPCC) [10], multi-label conditional random fields (ML-CRF) [28], and maximum margin output coding (MMOC) [45]. We also compare MC with a single CTBN (SC) [2] model without creating a mixture.

For all methods, we use the same parameter settings as suggested in their papers: For MLKNN and IBLR, which use the k-nearest neighbor (KNN) method, we use Euclidean distance to measure similarity of instances and we set the number of nearest neighbors to 10 [43, 8]; for CC, we set the order of classes to $Y_1 < Y_2, ... < Y_d$ [31]; for ECC and EPCC, we use 10 CCs in the ensemble [31, 10]; finally for MMOC, we set the decoding parameter to 1 [45]. Also note that all of these methods except MLKNN and MMOC are considered as meta-learners because they can work with several base classifiers. To eliminate additional effects that may bias the results, we use L_2-penalized logistic regression for all of these methods and choose their regularization parameters by cross validation. For our MC model, we decide the number of mixture components using our stopping criterion (Section 5.3.2) and we use 150 iterations of simulated annealing for prediction.

Table 1: Datasets characteristics
(N: number of instances, m: number of features, d: number of labels, LC: label cardinality, DLS: distinct label set)

DATASET	N	m	d	LC	DLS	DOMAIN
Emotions	593	72	6	1.87	27	music
Yeast	2,417	103	14	4.24	198	biology
Scene	2,407	294	6	1.07	15	image
Image	2,000	135	5	1.24	20	image
Enron	1,702	1,001	53	3.38	753	text
RCV1_subset1	6,000	8,394	10	1.31	69	text
RCV1_subset2	6,000	8,304	10	1.21	70	text
RCV1_subset3	6,000	8,328	10	1.22	74	text
RCV1_subset4	6,000	8,332	10	1.22	79	text
RCV1_subset5	6,000	8,367	10	1.31	76	text

6.2 Evaluation Measures

Evaluating the performance of MLC methods is more difficult than evaluating simple classification methods. The most suitable performance measure is the *exact match accuracy* (EMA), which computes the percentage of instances whose predicted label vectors are exactly the same as their true label vectors.

$$EMA = \sum_{n=1}^{N} \delta(\mathbf{y}^{(n)}, h(\mathbf{x}^{(n)}))$$

However, this measure could be too harsh, especially when the output dimensionality is high. Another very useful measure is the *conditional log-likelihood loss* (CLL-loss), which computes the negative conditional log-likelihood of the test instances:

$$CLL\text{-}loss = \sum_{n=1}^{N} -\log\left(P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)})\right)$$

CLL-loss evaluates how much probability mass is given to the true label vectors (the higher the probability, the smaller the loss).

Other evaluation measures used commonly in MLC literature are based on F1 scores. *Micro F1* aggregates the number of true positives, false positives and false negatives for all classes and then calculates the overall F1 score. On the other hand, *macro F1* computes the F1 score for each class separately and then averages these scores. Note that both measures are not the best for MLC because they do not account for the correlations between classes (see [10] and [41]). However, we report them in our performance comparisons as they have been used in other MLC literature [37].

6.3 Results

6.3.1 Performance Comparisons

We have performed *ten-fold cross validation* for all of our experiments. To evaluate the statistical significance of performance difference, we apply paired t-tests at 0.05 significance level. We use markers $*/\circledast$ to indicate whether MC is significantly better/worse than the compared method.

Tables 2, 3, 4 and 5 show the performance of the methods in terms of EMA, CLL-loss, micro F1 and macro F1, respectively. We only show the results of MMOC on four datasets (emotions, yeast, scene and image) because it did not finish on the remaining data (MMOC did not finish one round of the learning within a 24 hours time limit). For the same reason, we do not report the results of PCC, EPCC and MLCRF on the enron dataset. Also note that we do not report CLL-loss for MMOC, ECC and EPCC because they do not compute a probabilistic score for a given class assignment.

Table 2: Performance of each method on the benchmark datasets in terms of exact match accuracy
Marker $*/\circledast$ indicates whether MC is statistically superior/inferior to the compared method (using paired t-test at 0.05 significance level). The last row shows the total number of win/tie/loss for MC against the compared method (e.g., #win is how many times MC significantly outperforms that method).

EMA	BR	CHF	MLKNN	IBLR	CC	ECC	PCC	EPCC	MLCRF	MMOC	SC	MC
Emotions	0.265 $*$	0.300 $*$	0.283 $*$	0.335	0.268 $*$	0.288 $*$	0.317	0.344	0.303 $*$	0.332	0.322	0.346
Yeast	0.151 $*$	0.163 $*$	0.179 $*$	0.204 $*$	0.193 $*$	0.204 $*$	0.230	0.219	0.180 $*$	0.219	0.192 $*$	0.235
Scene	0.541 $*$	0.605 $*$	0.629 $*$	0.644 $*$	0.632 $*$	0.658 $*$	0.666	0.671	0.583 $*$	0.664	0.625 $*$	0.680
Image	0.280 $*$	0.360 $*$	0.346 $*$	0.387 $*$	0.426 $*$	0.413 $*$	0.449	0.442	0.377 $*$	0.448	0.414 $*$	0.463
Enron	0.164 $*$	0.170 $*$	0.078 $*$	0.163 $*$	0.173 $*$	0.180	-	-	-	-	0.167 $*$	0.187
Rcv1_subset1	0.334 $*$	0.357 $*$	0.205 $*$	0.279 $*$	0.429 $*$	0.410 $*$	0.432 $*$	0.420 $*$	0.344 $*$	-	0.441 $*$	0.457
Rcv1_subset2	0.439 $*$	0.465 $*$	0.288 $*$	0.417 $*$	0.516 $*$	0.509 $*$	0.523 $*$	0.516 $*$	0.475 $*$	-	0.531	0.536
Rcv1_subset3	0.466 $*$	0.486 $*$	0.327 $*$	0.446 $*$	0.539 $*$	0.539 $*$	0.548 $*$	0.544 $*$	0.489 $*$	-	0.560	0.561
Rcv1_subset4	0.510 $*$	0.531 $*$	0.354 $*$	0.491 $*$	0.579 $*$	0.569 $*$	0.588	0.576 $*$	0.550 $*$	-	0.592	0.591
Rcv1_subset5	0.439 $*$	0.456 $*$	0.276 $*$	0.411 $*$	0.497 $*$	0.494 $*$	0.519 $*$	0.513 $*$	0.457 $*$	-	0.539	0.540
#win/#tie/#loss	**10/0/0**	**10/0/0**	**10/0/0**	**9/1/0**	**10/0/0**	**9/1/0**	**4/5/0**	**5/4/0**	**9/0/0**	**0/4/0**	**5/5/0**	

Table 3: Performance of each method in terms of conditional log-likelihood loss
Marker $*/\circledast$ indicates whether MC is statistically superior/inferior to the compared method (using paired t-test at 0.05 significance level). The last row shows the total number of win/tie/loss for MC against the compared method.

CLL-loss	BR	CHF	MLKNN	IBLR	CC	PCC	MLCRF	SC	MC
Emotions	153.5 $*$	147.5 $*$	151.7 $*$	143.0 $*$	169.6 $*$	134.9	139.2 $*$	147.4 $*$	128.8
Yeast	1500.3 $*$	1491.7 $*$	1464.9 $*$	1434.2 $*$	2303.8 $*$	932.1 $\circledast$	1175.4 $*$	1097.0 $*$	1000.0
Scene	344.7 $*$	318.4 $*$	310.9 $*$	283.9 $*$	395.0 $*$	258.9	313.2 $*$	306.3 $*$	260.1
Image	432.5 $*$	415.9 $*$	425.3 $*$	395.6 $*$	480.3 $*$	354.7	401.4 $*$	388.4 $*$	347.1
Enron	1287.3 $*$	1272.5 $*$	1301.2 $*$	1287.4 $*$	1293.5 $*$	-	-	1437.9 $*$	1224.4
Rcv1_subset1	1443.8 $*$	2144.2 $*$	1873.7 $*$	1379.5 $*$	1701.3 $*$	1034.3 $*$	1369.4 $*$	962.7	951.1
Rcv1_subset2	1207.4 $*$	2223.6 $*$	1687.8 $*$	1172.6 $*$	1398.8 $*$	923.0 $*$	1123.6 $*$	893.5 $*$	855.8
Rcv1_subset3	1207.4 $*$	2156.0 $*$	1674.6 $*$	1168.2 $*$	1500.5 $*$	896.7 $*$	1116.4 $*$	939.7 $*$	837.2
Rcv1_subset4	1072.9 $*$	1759.9 $*$	1532.9 $*$	1034.8 $*$	1282.1 $*$	823.0 $*$	951.4 $*$	790.7 $*$	770.6
Rcv1_subset5	1267.0 $*$	2283.6 $*$	1795.5 $*$	1234.7 $*$	1422.0 $*$	1009.0 $*$	1192.4 $*$	924.0 $*$	894.3
#win/#tie/#loss	**10/0/0**	**10/0/0**	**10/0/0**	**10/0/0**	**10/0/0**	**5/3/1**	**9/0/0**	**9/1/0**	

Table 4: Performance of each method in terms of micro F1
Marker $*/\circledast$ indicates whether MC is statistically superior/inferior to the compared method (using paired t-test at 0.05 significance level). The last row shows the total number of win/tie/loss for MC against the compared method.

Micro_F1	BR	CHF	MLKNN	IBLR	CC	ECC	PCC	EPCC	MLCRF	MMOC	SC	MC
Emotions	0.645 $*$	0.672	0.656 $*$	0.692	0.621 $*$	0.652 $*$	0.664 $*$	0.688	0.684	0.687	0.678	0.693
Yeast	0.635	0.637	0.646	0.661 $\circledast$	0.628	0.631	0.645	0.650	0.619 $*$	0.651	0.631	0.640
Scene	0.696 $*$	0.722 $*$	0.736	0.758	0.697 $*$	0.724 $*$	0.722 $*$	0.743	0.713 $*$	0.711 $*$	0.717 $*$	0.745
Image	0.479 $*$	0.541 $*$	0.504 $*$	0.573	0.550	0.563	0.565	0.577	0.558	0.572	0.561	0.573
Enron	0.551	0.569 $\circledast$	0.450 $*$	0.566 $\circledast$	0.577 $\circledast$	0.583 $\circledast$	-	-	-	-	0.552	0.556
Rcv1_subset1	0.503 $*$	0.516	0.257 $*$	0.459 $*$	0.511 $*$	0.525	0.510 $*$	0.529	0.505 $*$	-	0.512 $*$	0.525
Rcv1_subset2	0.568 $*$	0.584	0.317 $*$	0.546 $*$	0.586	0.589	0.588	0.591	0.582 $*$	-	0.591	0.587
Rcv1_subset3	0.576 $*$	0.592	0.364 $*$	0.564 $*$	0.594	0.610 $\circledast$	0.594	0.613 $\circledast$	0.590	-	0.596	0.599
Rcv1_subset4	0.622 $*$	0.637	0.404 $*$	0.606 $*$	0.640	0.646 $\circledast$	0.644 $\circledast$	0.650 $\circledast$	0.635	-	0.638	0.635
Rcv1_subset5	0.582 $*$	0.597	0.314 $*$	0.566 $*$	0.595	0.603	0.600	0.605 $\circledast$	0.589 $*$	-	0.598	0.597
#win/#tie/#loss	**8/2/0**	**2/7/1**	**8/2/0**	**5/3/2**	**3/6/1**	**2/5/3**	**3/5/1**	**0/6/3**	**5/4/0**	**1/3/0**	**2/8/0**	

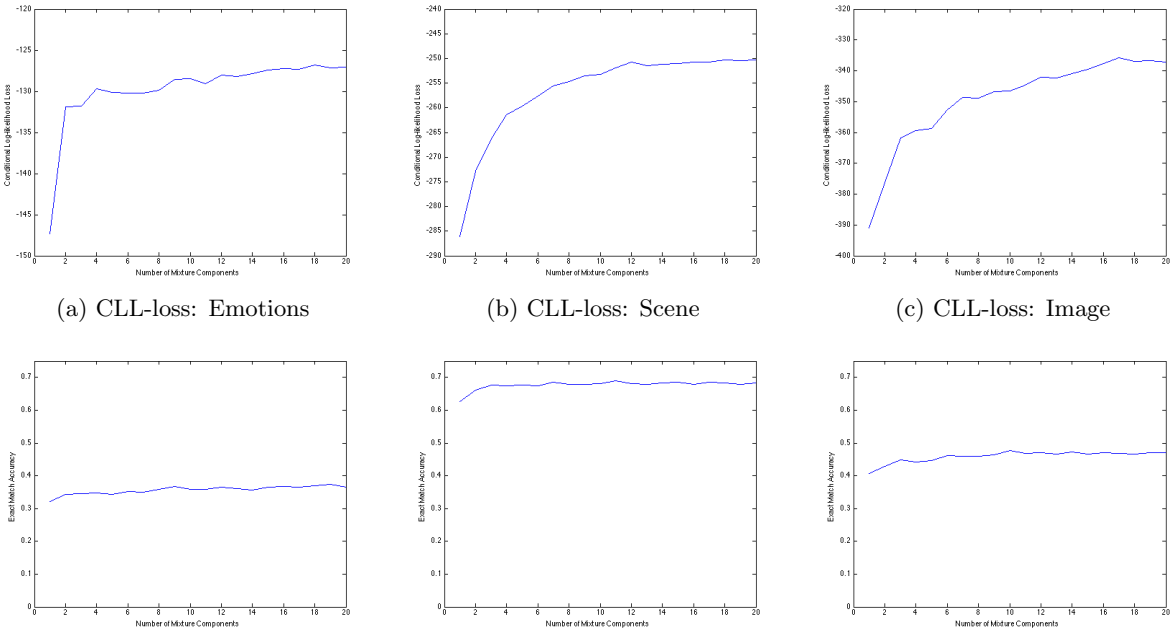
In terms of EMA (Table 2), MC clearly outperforms the other methods on most datasets. MC is significantly better than BR, CHF, MLKNN and CC on all ten datasets, significantly better than IBLR, ECC and MLCRF on nine datasets, significantly better than EPCC and SC on five datasets and significantly better than PCC on four datasets (see the last row of Table 2). Although not statistically significant, MC performs better than MMOC on all datasets MMOC is able to finish. MLKNN and IBLR perform poorly on the high-dimensional ($m > 1,000$) datasets because Euclidean distances between data instances become indiscernible in high dimensions.

Interestingly, MC shows significant improvements over SC (a single CTBN) on five datasets, while SC produces competitive results as well. We attribute the improved performance of MC to the ability of mixtures to compensate for the restricted dependences modeled by CTBNs, and that of individual CTBNs to better fit the data with different weight sets. On the contrary, ECC and EPCC do not show consistent improvements over their base methods (CC and PCC, respectively) and sometimes even deteriorate the accuracy. This is due to the ad-hoc nature of their ensemble learning and prediction (see Section 3) that limits the potential improvement and disturbs the prediction of the ensemble classifiers.

Table 3 compares MC to other probabilistic MLC methods using CLL-loss. The results show that MC outperforms all other methods. This is expected because MC is tailored to optimize the conditional log-likelihood. Among the compared probabilistic methods, only PCC produces comparable results with MC because PCC explicitly evaluates all possible class assignments to compute the entire class conditional distribution. On the other hand, CC greedily seeks the mode of the class conditional distribution (Equation (2)) and results in large losses. In addition, CHF and MLKNN perform very poorly because they apply ad-hoc classification heuristics without performing proper probabilistic inference. Again, MC shows consistent improvements over SC because

Table 5: Performance of each method in terms of macro F1
Marker $*/\circledast$ indicates whether MC is statistically superior/inferior to the compared method (using paired t-test at 0.05 significance level). The last row shows the total number of win/tie/loss for MC against the compared method.

Macro_F1	BR	CHF	MLKNN	IBLR	CC	ECC	PCC	EPCC	MLCRF	MMOC	SC	MC
Emotions	0.632 $*$	0.667	0.656	0.690	0.620 $*$	0.643 $*$	0.659	0.683	0.667	0.679	0.670	0.686
Yeast	0.457 $*$	0.461 $*$	0.478	0.498 $\circledast$	0.467	0.477	0.486	0.496 $\circledast$	0.451 $*$	0.473	0.467	0.477
Scene	0.703 $*$	0.730 $*$	0.743	0.765	0.709 $*$	0.740	0.729 $*$	0.753	0.721 $*$	0.721 $*$	0.728 $*$	0.755
Image	0.486 $*$	0.546 $*$	0.516 $*$	0.581	0.562	0.571	0.575	0.586	0.560 $*$	0.578	0.572	0.584
Enron	0.478 $\circledast$	0.479	0.411 $*$	0.475	0.484 $\circledast$	0.482 $\circledast$	-	-	-	-	0.470	0.470
Rcv1_subset1	0.495 $*$	0.511	0.273 $*$	0.463 $*$	0.506 $*$	0.516	0.504 $*$	0.521	0.500 $*$	-	0.507	0.517
Rcv1_subset2	0.503 $*$	0.526	0.264 $*$	0.475 $*$	0.531	0.539	0.531	0.538	0.516 $*$	-	0.536	0.531
Rcv1_subset3	0.513 $*$	0.536	0.278 $*$	0.497 $*$	0.547	0.558 $\circledast$	0.548	0.561 $\circledast$	0.531	-	0.543	0.542
Rcv1_subset4	0.499 $*$	0.519	0.269 $*$	0.477 $*$	0.534 $\circledast$	0.540 $\circledast$	0.534 $\circledast$	0.539 $\circledast$	0.515	-	0.526	0.522
Rcv1_subset5	0.500 $*$	0.526	0.257 $*$	0.487 $*$	0.536	0.538 $\circledast$	0.534	0.538 $\circledast$	0.513 $*$	-	0.536	0.527
#win/#tie/#loss	**9/0/1**	**3/7/0**	**7/3/0**	**5/4/1**	**3/5/2**	**1/5/4**	**2/6/1**	**0/5/4**	**6/3/0**	**1/3/0**	**1/9/0**	



(a) CLL-loss: Emotions (b) CLL-loss: Scene (c) CLL-loss: Image

(d) EMA: Emotions (e) EMA: Scene (f) EMA: Image

Figure 3: Conditional log-likelihood loss and exact match accuracy of MC with different number of mixture components

mixing multiple CTBNs allows us to account for different patterns in the data and, hence, improves the generalization of the model.

Lastly, Tables 4 and 5 show that MC is also very competitive in terms of micro and macro F1 scores, although optimizing them was not our immediate objective. One noteworthy observation is that ECC and EPCC do particularly well in terms of F1 scores. We consider averaging out the predictions on each class variable enhances BR-like characteristics in their ensemble decision. In the future, we will crossbreed these two different ensemble approaches (e.g., MCC/MPCC by applying our mixture framework and algorithms to CC/PCC; ECTBN using randomly structured CTBNs and simple averaging) and compare the performances.

6.3.2 *Effect of the Number of Mixture Components*

In the second part of our experiments, we investigate the effect of different number of mixture components in the MC model. Using three of the benchmark datasets (emotions, scene and image), we study how the performance of MC changes while we increase the number of trees in a model from 1 to 20. In particular, we use ten-fold cross validation and trace the average CLL-loss and EMA across the folds.

Figure 3 summarizes the results. Figures 3(a), 3(b) and 3(c) show how CLL-loss changes on emotions, scene and image, respectively. On all three datasets, adding first few trees brings the CLL-loss of a mixture model in a rapid improvement. Then the growth becomes slower until it reaches its first peak. After it passes the first peak, CLL-loss stops improving and becomes stable.

Figures 3(d), 3(e) and 3(f) show the performance changes in EMA. Notice that EMA is closely correlated with CLL-loss on all three datasets, and our stopping criteria is useful in optimizing EMA as well as CLL-loss. That is, EMA improves significantly while CLL-loss increases rapidly. Once CLL-loss becomes stable, EMA also seems to be stable and does not show any signs of fluctuation or overfitting.

7. CONCLUSION

In this work, we proposed a new probabilistic approach to multi-label classification based on the mixture of Conditional Tree-structured Bayesian Networks. We devised and presented algorithms for learning the parameters of the mixture, finding multiple tree structures and inferring the maximum a posteriori (MAP) output label configurations for the model. Our experimental evaluation on a range of datasets shows that our approach outperforms the state-of-the-art multi-label classification methods in most cases.

8. ACKNOWLEDGMENTS

This work was supported by grants R01LM010019 and R01GM088224 from the NIH. Its content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

9. REFERENCES

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

[2] I. Batal, C. Hong, and M. Hauskrecht. An efficient probabilistic framework for multi-dimensional classification. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, CIKM '13, pages 2417–2422. ACM, 2013.

[3] J. Berger. *Statistical decision theory and Bayesian analysis*. Springer series in statistics. Springer, New York, NY, 2. ed edition, 1985.

[4] U. Berkeley. Enron email analysis. http://bailando.sims.berkeley.edu/enron_email.html. Accessed: 2014-8-16.

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

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

[7] J. K. Bradley and C. Guestrin. Learning tree conditional random fields. In *International Conference on Machine Learning (ICML 2010)*, Haifa, Israel, 2010.

[8] W. Cheng and E. Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225, 2009.

[9] A. Clare and R. D. King. Knowledge discovery in multi-label phenotype data. In *In: Lecture Notes in Computer Science*, pages 42–53. Springer, 2001.

[10] K. Dembczynski, W. Cheng, and E. Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 279–286. Omnipress, 2010.

[11] J. Edmonds. Optimum branchings. *Research of the National Bureau of Standards*, 71B:233–240, 1967.

[12] A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In *NIPS*, pages 681–687, 2001.

[13] N. Ghamrawi and A. McCallum. Collective multi-label classification. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 195–200. ACM, 2005.

[14] S. Godbole and S. Sarawagi. Discriminative methods for multi-labeled classification. In *PAKDD'04*, pages 22–30, 2004.

[15] M. Hauskrecht, I. Batal, M. Valko, S. Visweswaran, G. F. Cooper, and G. Clermont. Outlier detection for patient monitoring and alerting. *Journal of Biomedical Informatics*, 46(1):47–55, Feb. 2013.

[16] M. Hauskrecht, M. Valko, I. Batal, G. Clermont, S. Visweswaram, and G. Cooper. Conditional outlier detection for clinical alerting. *Annual American Medical Informatics Association Symposium*, 2010.

[17] M. Hauskrecht, M. Valko, B. Kveton, S. Visweswaram, and G. Cooper. Evidence-based anomaly detection. In *Annual American Medical Informatics Association Symposium*, pages 319–324, November 2007.

[18] D. Hsu, S. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In *NIPS*, pages 772–780, 2009.

[19] S. Ioffe and D. Forsyth. Human tracking with mixtures of trees. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 690–695 vol.1, 2001.

[20] S. Ioffe and D. A. Forsyth. Mixtures of trees for object recognition. In *CVPR (2)*, pages 180–185. IEEE Computer Society, 2001.

[21] H. Kazawa, T. Izumitani, H. Taira, and E. Maeda. Maximal margin labeling for multi-topic text categorization. In *Advances in Neural Information Processing Systems 17*, pages 649–656. MIT Press, 2005.

[22] S. Kirshner and P. Smyth. Infinite mixtures of trees. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 417–423, New York, NY, USA, 2007. ACM.

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

[24] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, 2001.

[25] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, Dec. 2004.

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

[27] T. Moon. The expectation-maximization algorithm. *Signal Processing Magazine, IEEE*, 13(6):47–60, 1996.

[28] M. Pakdaman, I. Batal, Z. Liu, C. Hong, and M. Hauskrecht. An optimization-based framework to learn conditional random fields for multi-label classification. In *SDM*. SIAM, 2014.

[29] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, T. Mei, and H.-J. Zhang. Correlative multi-label video annotation. In *Proceedings of the 15th international conference on Multimedia*, MULTIMEDIA '07, pages 17–26. ACM, 2007.

[30] H. Raiffa. *Decision Analysis: Introductory Lectures on Choices Under Uncertainty*. Mcgraw-Hill College, Jan. 1997.

[31] 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*, ECML PKDD '09. Springer-Verlag, 2009.

[32] D. Shahaf and C. Guestrin. Learning thin junction trees via graph cuts. In *AISTATS*, volume 5 of *JMLR Proceedings*, pages 113–120. JMLR.org, 2009.

[33] D. Sontag. *Approximate Inference in Graphical Models using LP Relaxations*. PhD thesis, Massachusetts Institute of Technology, 2010.

[34] F. Tai and H.-T. Lin. Multi-label classification with principle label space transformation. In *the 2nd International Workshop on Multi-Label Learning*, 2010.

[35] R. E. Tarjan. Finding optimum branchings. *Networks*, 7(1):25–35, 1977.

[36] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. P. Vlahavas. Multi-label classification of music into emotions. In *ISMIR*, pages 325–330, 2008.

[37] G. Tsoumakas, M.-L. Zhang, and Z.-H. Zhou. Learning from multi-label data. ECML PKDD Tutorial, 2009.

[38] L. C. van der Gaag and P. R. de Waal. Multi-dimensional bayesian network classifiers. In *Probabilistic Graphical Models*, pages 107–114, 2006.

[39] T. Šingliar and M. Hauskrecht. Modeling highway traffic volumes. In *Proceedings of the 18th European Conference on Machine Learning*, ECML '07, pages 732–739. Springer-Verlag, 2007.

[40] C. Yuan, T.-C. Lu, and M. J. Druzdzel. Annealed map. In *UAI*, pages 628–635. AUAI Press, 2004.

[41] M.-L. Zhang and K. Zhang. Multi-label learning by exploiting label dependency. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 999–1008. ACM, 2010.

[42] M.-L. Zhang and Z.-H. Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.

[43] M.-L. Zhang and Z.-H. Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recogn.*, 40(7):2038–2048, July 2007.

[44] Y. Zhang and J. Schneider. Multi-label output codes using canonical correlation analysis. In *AISTATS 2011*, 2011.

[45] Y. Zhang and J. Schneider. Maximum margin output coding. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1575–1582, 2012.