An Efficient Probabilistic Framework for Multi-Dimensional Classification

Iyad Batal
Computer Science Department
University of Pittsburgh
iyad@cs.pitt.edu

Charmgil Hong
Department of Computer Science
University of Pittsburgh
charmgil@cs.pitt.edu

Milos Hauskrecht
Department of Computer Science
University of Pittsburgh
milos@cs.pitt.edu

ABSTRACT

The objective of multi-dimensional classification is to learn a function that accurately maps each data instance to a vector of class labels. Multi-dimensional classification appears in a wide range of applications including text categorization, gene functionality classification, semantic image labeling, etc. Usually, in such problems, the class variables are not independent, but rather exhibit conditional dependence relations among them. Hence, the key to the success of multi-dimensional classification is to effectively model such dependencies and use them to facilitate the learning. In this paper, we propose a new probabilistic approach that represents class conditional dependencies in an effective yet computationally efficient way. Our approach uses a special tree-structured Bayesian network model to represent the conditional joint distribution of the class variables given the feature variables. We develop and present efficient algorithms for learning the model from data and for performing exact probabilistic inferences on the model. Extensive experiments on multiple datasets demonstrate that our approach achieves highly competitive results when it is compared to existing state-of-the-art methods.

Categories and Subject Descriptors

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

Keywords

Multi-dimensional classification, Bayesian network, MAP inference

1. INTRODUCTION

In traditional classification learning, each data instance is associated with a single class variable and the goal is to predict the class label for future instances. However, there are many real-world applications where each data instance is naturally associated with multiple class variables (a vector of labels). To name a few, in text categorization [8, 17], a document can simultaneously cover multiple topics, such as *politics* and *economy*; In bioinformatics [4, 17], each gene may be associated with several functional classes, such as *protein synthesis*, *transcription* and *metabolism*; In semantic scene and video classification [2, 10], a scene object can be classified as *urban*, *building* and *road*; In clinical informatics, a patient may become resistant to multiple drugs for HIV treatment.

Multi-Dimensional Classification (MDC) learning deals with the above mentioned situations and assumes each instance is associated with d discrete-valued class variables $Y_1, ..., Y_d$. The goal is to learn a function that assigns to each instance, represented by its feature vector $\mathbf{x} = (x_1, ..., x_m)$, the most probable assignment of the class variables $\mathbf{y} = (y_1, ..., y_d)$. Learning such a function can be very challenging due to the exponential number of possible class labelings that may be assigned to each instance.

The simplest solution to MDC is to completely ignore class correlations by constructing a classifier that predicts each class variable independently from the other classes [2, 4, 12]. This approach often fails because it does not take advantage of the dependency relations between the classes, which is the key to facilitate the learning of MDC. For example, a document that is related to *politics* is unlikely to be labeled as *sports*. To overcome this limitation, several methods have been proposed in the literature. These methods account for class dependencies by using different strategies, such as instance-based learning [18, 3], output coding [7, 19, 20], multi-dimensional Bayesian networks [14, 1] and classifier chains [11, 5, 16, 15].

In this paper, we propose a new probabilistic method called Conditional Tree-structured Bayesian Network (CTBN) which models the conditional dependencies between classes in an effective yet computationally efficient way. Briefly, CTBN learns a Bayesian network to represent the distribution of all class variables *conditioned* on the feature variables $P(Y_1, ..., Y_d|\mathbf{X})$. The network structure is defined by making the feature vector $\mathbf{X}$ a common parent for all class variables. Besides that, each class variable Y_i is allowed to have at most one other class variable $Y_{\pi(i)}$ as a parent (in addition to $\mathbf{X}$). In other words, the conditional dependency relations between class variables follow a *directed tree*. The conditional probability distribution of each class conditioned on its parents $P(Y_i|\mathbf{X}, Y_{\pi(i)})$ is modeled using a probabilistic classifier, such as logistic regression. An example of CTBN with four class variables is depicted in Figure 1.

To learn CTBN structure from data, we present an efficient algorithm for finding the structure that has the highest conditional log likelihood score. To classify new instances, we present an exact inference algorithm, which computes the most probable assignment of the classes in linear time in the number of classes.

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

http://dx.doi.org/10.1145/2505515.2505594.

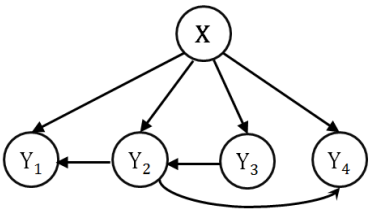

Figure 1: An example of a CTBN model that defines the conditional distribution $P(Y_1, Y_2, Y_3, Y_4|\mathbf{X})$.

Note that CTBN is related to some recent MDC methods. In particular, it shares some similarities with the multi-dimensional Bayesian network classifier (MBC) approach [14, 1], in that both use Bayesian networks to solve the MDC problem. The main difference, however, is that MBC attempts to model the full *joint* distribution over both features and classes $P(\mathbf{Y}, \mathbf{X})$ (a generative model), while CTBN directly models the *conditional* distribution of the classes given the features $P(\mathbf{Y}|\mathbf{X})$ without wasting effort on modeling the features (a discriminative model). As a result, learning MBC structure from data is much harder than learning CTBN[1]. Furthermore, MBC can only handle discrete features, while CTBN can handle both numeric and discrete features. Note that CTBN is also related to the classifier chains (CC) approach [11, 5, 16, 15] as both use classifiers that incorporate other classes as input. However, the classification process is very different: CC-based methods apply an ad-hoc heuristic by simply propagating class predictions along the chain, while CTBN performs proper probabilistic inference to find the optimal output.

2. PROBLEM DEFINITION

Multi-Dimensional Classification (MDC) [1] are classification problems in which each data instance is associated with d discrete-valued class variables $Y_1, ..., Y_d$. We are given labeled training data $D = \{\mathbf{x}^{(k)}, \mathbf{y}^{(k)}\}_{k=1}^{n}$, where $\mathbf{x}^{(k)} = (x_1^{(k)}, ..., x_m^{(k)})$ is the m-dimensional feature vector of the k-th instance (the input) and $\mathbf{y}^{(k)} = (y_1^{(k)}, ..., y_d^{(k)})$ is its d-dimensional class vector (the output). The goal is to learn from D a function h that assigns to each instance, represented by a vector of m features, a vector of d class values:

$$h : \mathbb{R}^m \rightarrow \Omega(Y_1) \times ... \times \Omega(Y_d)$$

where $\Omega(Y_i)$ denotes the sample space of class variable Y_i and $\Omega(Y_1) \times ... \times \Omega(Y_d)$ denotes the space of joint configurations of all class variables.

Note that Multi-Label Classification (MLC) can be seen as a special case of MDC. In the MLC setting, each instance is associated with a subset of labels from a set of d labels. By using the notation above, each instance $\mathbf{x}^{(k)}$ would be associated with a *binary* class vector $\mathbf{y}^{(k)} \in \{0, 1\}^d$, where $y_i^{(k)} = 1$ if $\mathbf{x}^{(k)}$ is associated with the i-th label and $y_i^{(k)} = 0$ otherwise.

Developing a probabilistic approach to MDC requires modeling and learning 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. Under the 0-1 loss function, the Bayes optimal classifier h^* should assign to instance $\mathbf{x}$ the most probable assignment of the class variables, known as the maximum a posterior (MAP) estimation:

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

In general, solving Equation 1 has exponential complexity because it requires evaluating all possible value assignments to the class variables.

The goal of this paper is to develop a parametric model that allows us to effectively estimate $P(\mathbf{Y}|\mathbf{X})$ from data and to perform MAP inference (i.e., classification) in a computationally efficient way.

Notation: From hereafter, we will sometimes abbreviate the expressions by omitting variable names, for example we write $P(Y_1 = y_1, ..., Y_d = y_d|\mathbf{X} = \mathbf{x})$ simply as $P(y_1, ..., y_d|\mathbf{x})$.

3. WHY NOT LEARNING INDEPENDENT CLASSIFIERS

The simplest solution to MDC is to learn an independent classifier for each class variable [2, 4, 12], which is known as the *binary relevance* (BR) approach. More specifically, BR learns a separate classifier to predict each class $Y_i : i \in \{1, ..., d\}$ and determines the output of a new instance $\mathbf{x}$ by simply aggregating the predictions of all classifiers.

Probabilistically, BR relies on the simplifying assumption that all class variables are conditionally independent of each other given $\mathbf{x}$:

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

Under this assumption, the optimal prediction of BR (solving Equation 1) is simply the class vector that has the highest conditional *marginal* probability for each element:

$$h_{BR}(\mathbf{x}) = \Big(\arg\max_{y_1} P(y_1|\mathbf{x}), ..., \arg\max_{y_d} P(y_d|\mathbf{x}) \Big) \quad (2)$$

However, this simple approach does not always produce correct results, as we show in the following example.

EXAMPLE 1. *Assume the conditional joint distribution of class variables Y_1 and Y_2 for a specific instance $\mathbf{x}$ is shown in Table 1. The optimal classification for $\mathbf{x}$ (according to Equation 1) is $h^*(\mathbf{x}) = (Y_1 = 1, Y_2 = 0)$. However, the result of BR (according to Equation 2) is $h_{BR}(\mathbf{x}) = (Y_1 = 0, Y_2 = 0)$.*

$P(Y_1, Y_2|\mathbf{X} = \mathbf{x})$	$Y_1 = 0$	$Y_1 = 1$	$P(Y_2|\mathbf{X} = \mathbf{x})$
$Y_2 = 0$	0.2	**0.45**	**0.65**
$Y_2 = 1$	0.35	0	0.35
$P(Y_1|\mathbf{X} = \mathbf{x})$	**0.55**	0.45	

Table 1: The joint distribution of class variables Y_1 and Y_2 conditioned on instance $\mathbf{x}$. The optimal (MAP) prediction is $h^*(\mathbf{x}) = (Y_1 = 1, Y_2 = 0)$.

In the following, we propose the Conditional Tree-structured Bayesian Network (CTBN) model to allow more elaborate conditional dependency relations between the class variables.

4. THE CONDITIONAL TREE-STRUCTURED BAYESIAN NETWORK MODEL

[1]The search space of MBC (increases with the dimensionality of the input m) is much larger than that of CTBN (does not depend on m).

In the CTBN model, the feature vector $\mathbf{X}$ is defined to be a common parent for all class variables (similar to BR). In addition to $\mathbf{X}$, each class variable can have at most another class variable as a parent (without creating a cycle). That is, the conditional dependency relations between class variables follow a *directed tree*. We chose to restrict the dependency structure to a tree for the following reasons:

1. The optimal structure can be learned using a simple and efficient learning algorithm.
2. The prediction can be done efficiently using exact inference.

4.1 Representation and Parametrization

Representation: Let T be a CTBN model and let $\pi(i,T)$ be the parent class of class Y_i in T (by convention, $\pi(i,T) = \phi$ if Y_i does not have a parent class). The joint distribution of class vector $(y_1, ..., y_d)$ conditioned on feature vector $\mathbf{x}$ is now expressed as follows:

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

In Figure 1, we showed an example CTBN with four class variables (Y_1, Y_2, Y_3, Y_4). The conditional joint distribution of class assignment (y_1, y_2, y_3, y_4) given $\mathbf{x}$ according to this network is defined as follows:

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

Parametrization: The parametrization of the CTBN model corresponds to specifying the conditional probability distribution (CPD) of each class variable Y_i conditioned on its parents: $P(Y_i|\mathbf{X}, Y_{\pi(i,T)})$. The standard parametrization of Bayesian networks uses conditional probability tables (CPT) to define the distribution of each variable conditioned on every possible configuration of its parents. However, the CPT style parametrization is not feasible for the CTBN model. The reason is that the feature vector $\mathbf{X}$, which is a common parent for all variables, can be a high-dimensional vector of continuous values, discrete values or a mixture of both (we cannot enumerate all possible configurations of $\mathbf{X}$).

To overcome this difficulty, we represent the CPDs using probabilistic prediction functions. More specifically, for each class $Y_i : i \in \{1, ..., d\}$, we approximate its CPD by learning a different probabilistic classifier $f_{iv}(\mathbf{X})$ for each possible value v of the parent class:

$$\tilde{P}(Y_i|\mathbf{X}=\mathbf{x}, Y_{\pi(i,T)}=v) = f_{iv}(\mathbf{x}) \quad (3)$$

Note that we can use several standard probabilistic classifiers in the CTBN model, such as logistic regression, naïve Bayes or the maximum entropy model. In our experiments, we use logistic regression with L_2 regularization.

4.2 Learning the Structure

In the previous section, we described the CTBN model and how to learn its parameters when the structure is known. In this section, we describe how to automatically learn the structure from data.

Our objective is to find the tree structure that best approximates the conditional joint distribution $P(\mathbf{Y}|\mathbf{X})$, which corresponds to the tree that maximizes the conditional log likelihood (CLL) of the data. To do this, we partition the data into two parts: training data D_t and hold-out data D_h. Given a CTBN T, we use D_t to train its parameters and we use D_h to compute its score, which corresponds to the CLL of D_h given T (adopting the standard iid assumption):

$$Score(T) = \sum_{(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) \in D_h} \sum_{i=1}^{d} \log\left(\tilde{P}(y_i^{(k)}|\mathbf{x}^{(k)}, y_{\pi(i,T)}^{(k)})\right) \quad (4)$$

In the following, we provide an algorithm to efficiently obtain the optimal CTBN T^* (the model that has the maximum score) without having to explicitly evaluate all of the exponentially many possible tree structures.

We start by defining a weighted directed graph $G = (V, E)$ as follows:

- There is one vertex V_i for each class variable Y_i.
- There is a directed edge $E_{j \to i}$ from each vertex V_j to each vertex V_i (G is complete). Furthermore, each vertex V_i has a self loop $E_{i \to i}$.
- The weights of the edges are defined as follows: The weight of edge $E_{j \to i}$, denoted as $W_{j \to i}$, is the CLL of class Y_i conditioned on $\mathbf{X}$ and Y_j:

$$W_{j \to i} = \sum_{(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) \in D_h} \log\left(\tilde{P}(y_i^{(k)}|\mathbf{x}^{(k)}, y_j^{(k)})\right) \text{ if } i \neq j$$

  The weight of self-loop $E_{i \to i}$, denoted as $W_{\phi \to i}$, is the CLL of Y_i conditioned only on $\mathbf{X}$.

By using this definition of edge weights and switching the order of the summations in Equation 4, we can rewrite the score of T simply as the sum of its edge weights (by convention, a node without a parent has a self loop):

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

Now we have transformed the problem of finding the optimal CTBN into the problem of finding the directed tree in G that has the maximum sum of edge weights. The solution can be obtained by solving the maximum branching (arborescence) problem [13], which finds the maximum weight directed tree in a weighted directed graph.

Complexity: Computing the edge weights for the complete graph G requires estimating $\tilde{P}(Y_i|\mathbf{X}, Y_j)$ for all d^2 pairs of classes. Finding the maximum branching in G can be done in $O(d^2)$ using Tarjan's implementation [13]. Therefore, the overall complexity is $O(d^2)$ times the complexity of learning the probabilistic classifiers (e.g., logistic regression).

4.3 Prediction

In order to make a prediction for a new instance $\mathbf{x}$, we should find the MAP assignment of class variables according to the CTBN model (solve Equation 1). This problem is NP-hard for general structure Bayesian networks. However, since we have restricted our structure to a tree, we can solve the problem efficiently using exact inference.

In particular, we perform inference using a variant of the max-sum algorithm [9] that we design for the CTBN model. This algorithm first computes the local CPTs for each node Y_i by applying the corresponding classifier for each possible value of the parent class (see Equation 3). After that, it performs two phases to obtain the optimal prediction. In the first phase, the algorithm sends messages upward (from the leaves to the root) where each node Y_i applies the following steps: i) it computes the sum of the logarithm of its local

CPT and all messages sent from its children, ii) maximizes the result over its value and iii) sends it to the parent node. In the second phase, the algorithm propagates the optimal assignments downward.

Complexity: The inference algorithm described above runs in $O(d)$, where d is the number of class variables.

EXAMPLE 2. *Consider the example in Figure 2, where we show the conditional probability tables of a CTBN model for a specific instance* $\mathbf{x}$ *(obtained by applying the classifiers on* $\mathbf{x}$*). The optimal prediction for* $\mathbf{x}$ *is* $(Y_3 = 0, Y_2 = 1, Y_1 = 0, Y_4 = 0)$ *(has a probability of 0.2016), which can be obtain by our exact inference algorithm. On the other hand, if we apply a CC-based method [11, 16, 15] using the topological order of classes in the tree, we get the suboptimal prediction* $(Y_3 = 1, Y_2 = 0, Y_1 = 1, Y_4 = 0)$ *(has a probability of 0.1512). The reason CC fails is that it starts incorrectly by predicting* $Y_3 = 1$ *and propagates this error down the tree.*

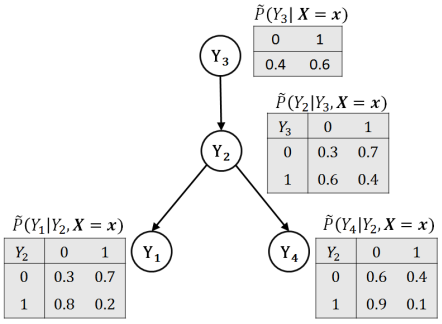

Figure 2: An example showing the CPTs of a CTBN model for a specific instance x.

5. EXPERIMENTS

5.1 Data

We use ten publicly available datasets that are obtained from different domains such as music recognition (emotions), biology (yeast), semantic image labeling (scene) and text classification (enron, TMC 2007 and RCV1 datasets). The RCV1 datasets are obtained from manually categorized news articles made available by Reuters. The characteristics of the datasets are summarized in Table 2. For each dataset, we show the number of instances, number of feature variables (input dimensionality), number of class variables (output dimensionality). In addition, we show two statistics: 1) label cardinality (LC), which is the average number of class variables per instance that have value one and 2) distinct label set (DLS), which is the number of all distinct configurations of classes that appear in the data.

5.2 Methods

We compare the performance of CTBN to several recently proposed methods:

- *Binary Relevance (BR)* [2, 4]. This simple baseline method learns to classify each class variable independently from the other class variables.
- *Classification with heterogeneous features (CHF)* [6]. This method stacks the two-levels of classifiers. The first level learns to classify each class using the original features (same as BR), while the second level learns to classify each class using the original features plus the output of the first level.
- *Classifier chains (CC)* [11]. This method defines a chain of classifiers, such that each classifier incorporates the predictions of all previous classifiers in the chain. As in [11], we set the order of classes to $Y_1 < Y_2, ... < Y_d$ and perform classification by propagating predictions along the chain (see Example 2).
- *Multi-label k-nearest neighbor (MLKNN)* [18]. This method learns a classifier for each class by combining KNN with Bayesian inference. As suggested in [18], the number of neighbors is set to 10 and Euclidean distance is used to measure similarity of instances.
- *Instance-based learning by logistic regression (IBLR)* [3]. This method combines instance-based and model-based classification by using class information of the neighbors as additional features for logistic regression. Similar to MLKNN, the number of neighbors is set to 10 and Euclidean distance is used.
- *Maximum margin output coding (MMOC)* [20]. This method is the state-of-the-art in multi-label output coding. MMOC applies a maximum margin formulation to encode the output, which promotes both discriminative and predictable codes. The decoding parameter λ is set to 1 as suggested by the authors.

Note that BR, CHF, CC, IBLR and CTBN are all considered meta-learners because they can apply different classifiers (e.g., different probabilistic classifiers can be used to estimate the CPDs in CTBN). So to eliminate additional effects that may bias the results, we use L_2-penalized logistic regression for all of these methods and choose the regularization parameters by cross validation.

5.3 Evaluation measures

We consider three different measures to evaluate the success of MDC:

- *Exact match accuracy (EMA)* computes the percentage of the instances whose predicted class vector is exactly the same as the true class vector (all classes are predicted correctly).
- *Conditional log likelihood loss (CLL-loss)* is only defined for probabilistic methods as follows:

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

  We can see that the "loss" for a test instance $\mathbf{x}^{(k)}$ is small if the predicted probability of the true class vector $\mathbf{y}^{(k)}$ is close to 1; and the loss is large if the probability of $\mathbf{y}^{(k)}$ is close to 0.
- *Micro F1* computes the number of the true positives, false positives and false negatives for each class variable independently and then aggregates them using micro-averaging to compute an overall F1 score.

Note that EMA is appropriate in the MDC settings because it evaluates the success of the method in finding the mode of the conditional joint distribution $P(\mathbf{Y}|\mathbf{X})$ (see Section 2). However, EMA may become overly stringent when the output dimensionality is large. CLL-loss is very useful for probabilistic methods because it evaluates how much probability mass the method assigns to the true class vector. For example, if two methods misclassify an instance

Dataset	#Instances	#Features	#Classes	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
Enron	1,702	1,001	53	3.38	753	text
TMC 2007	28,596	30,438	22	2.16	1,102	text
RCV1_top10 (S1)	6,000	8,394	10	1.31	69	text
RCV1_top10 (S2)	6,000	8,304	10	1.21	70	text
RCV1_top10 (S3)	6,000	8,328	10	1.22	74	text
RCV1_top10 (S4)	6,000	8,332	10	1.22	79	text
RCV1_top10 (S5)	6,000	8,367	10	1.31	76	text

Table 2: Datasets characteristics (LC: label cardinality, DLS: distinct label set)

Dataset	BR	CHF	CC	MLKNN	IBLR	MMOC	CTBN
Emotions	0.266 ∗	0.315	0.272 ∗	0.283 ∗	0.332	0.336	0.335
Yeast	0.147 ∗	0.162 ∗	0.194	0.179 ∗	0.204	0.214 ⊛	0.195
Scene	0.521 ∗	0.610 ∗	0.633	0.629	0.644	0.684 ⊛	0.626
Enron	0.162	0.169	0.173	0.078 ∗	0.163	-	0.168
TMC 2007	0.315 ∗	0.322 ∗	0.323 ∗	0.165 ∗	0.316 ∗	-	0.329
RCV1_top10 (S1)	0.278 ∗	0.357 ∗	0.429 ∗	0.205 ∗	0.279 ∗	-	0.448
RCV1_top10 (S2)	0.420 ∗	0.466 ∗	0.517 ∗	0.288 ∗	0.417 ∗	-	0.531
RCV1_top10 (S3)	0.442 ∗	0.485 ∗	0.540 ∗	0.327 ∗	0.446 ∗	-	0.561
RCV1_top10 (S4)	0.494 ∗	0.532 ∗	0.579 ∗	0.354 ∗	0.491 ∗	-	0.590
RCV1_top10 (S5)	0.412 ∗	0.457 ∗	0.497 ∗	0.276 ∗	0.411 ∗	-	0.538
#win/#tie/#loss	**9/1/0**	**8/2/0**	**7/3/0**	**9/1/0**	**6/4/0**	**0/1/2**	

Table 3: Performance of each method on the benchmark datasets in terms of exact match accuracy. Marker ∗/⊛ indicates whether CTBN 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 CTBN against the compared method (e.g, #win is how many times CTBN significantly outperforms that method).

Dataset	BR	CHF	CC	MLKNN	IBLR	CTBN
Emotions	154.2 ∗	146.6 ∗	170.8 ∗	151.7 ∗	143.4 ∗	136.2
Yeast	1,497 ∗	1,491 ∗	2,284 ∗	1,464 ∗	1,434 ∗	1,112
Scene	342.1 ∗	316.8 ∗	386.5 ∗	310.9 ∗	284.5 ⊛	291.2
Enron	1,290 ∗	1,272 ∗	1,295 ∗	1,301 ∗	1,285 ∗	1,239
TMC 2007	8,685 ∗	8,809 ∗	8,808 ∗	13,249 ∗	8,651 ∗	8,388
RCV1_top10 (S1)	1,386 ∗	2,201 ∗	1,684 ∗	1,873 ∗	1,379 ∗	960
RCV1_top10 (S2)	1,181 ∗	2,221 ∗	1,418 ∗	1,687 ∗	1,172 ∗	894
RCV1_top10 (S3)	1,177 ∗	2,157 ∗	1,500 ∗	1,674 ∗	1,171 ∗	941
RCV1_top10 (S4)	1,051 ∗	1,722 ∗	1,273 ∗	1,532 ∗	1,036 ∗	794
RCV1_top10 (S5)	1,240 ∗	2,272 ∗	1,426 ∗	1,795 ∗	1,234 ∗	925
#win/#tie/#loss	**10/0/0**	**10/0/0**	**10/0/0**	**10/0/0**	**9/0/1**	

Table 4: Performance of each method on the benchmark datasets in terms of conditional log likelihood loss.

Dataset	BR	CHF	CC	MLKNN	IBLR	MMOC	CTBN
Emotions	0.646 ∗	0.674	0.623 ∗	0.656 ∗	0.690	0.687	0.684
Yeast	0.636	0.638	0.629 ∗	0.646	0.662 ⊛	0.649 ⊛	0.642
Scene	0.682 ∗	0.724	0.697 ∗	0.736	0.757 ⊛	0.724	0.725
Enron	0.566	0.570	0.577 ⊛	0.449 ∗	0.567	-	0.568
TMC 2007	0.688	0.698 ⊛	0.690 ⊛	0.505 ∗	0.689 ⊛	-	0.687
RCV1_top10 (S1)	0.451 ∗	0.516	0.511	0.298 ∗	0.459 ∗	-	0.519
RCV1_top10 (S2)	0.550 ∗	0.584	0.586	0.317 ∗	0.546 ∗	-	0.590
RCV1_top10 (S3)	0.561 ∗	0.592 ∗	0.593	0.364 ∗	0.566 ∗	-	0.598
RCV1_top10 (S4)	0.609 ∗	0.637	0.640	0.404 ∗	0.608 ∗	-	0.635
RCV1_top10 (S5)	0.565 ∗	0.596	0.596	0.314 ∗	0.566 ∗	-	0.596
#win/#tie/#loss	**7/3/0**	**1/8/1**	**3/5/2**	**8/2/0**	**5/2/3**	**0/2/1**	

Table 5: Performance of each method on the benchmark datasets in terms of micro F1.

(according to EMA), CLL-loss still favors the one that assigns higher probability to the correct output. Finally, note that micro F1 is not very suitable for MDC because it does not consider the correlations between classes (see [5, 16]). However, we report it in our results as it has been used in several other papers such as [20, 21].

5.4 Results

All experimental results (Tables 3 to 5) are obtained using *ten-fold cross validation*. To evaluate the statistical significance of performance difference, we apply paired t-tests at 0.05 significance level. We use markers ∗/⊛ to indicate wether CTBN is significantly better/worse than the compared method.

Table 3 shows the EMA of the methods. We show the results of MMOC for only three datasets (emotions, yeast and scene) because it did not finish on the remaining data[2]. We

[2]MMOC did not finish one round of the learning within a 24 hours time limit.

Dataset	BR	CHF	CC	MLKNN	IBLR	MMOC	CTBN
Emotions	0.4	0.9	0.4	0.4	4.8	744.1	2.0
Yeast	7	16	7	7	89	5,405	43
Scene	12	25	12	10	133	2,718	39
Enron	33	80	35	21	152	-	517
TMC 2007	785	1,874	890	12,859	66,698	-	5,596
RCV1_top10 (S1)	185	375	176	3,476	14,769	-	744
RCV1_top10 (S2)	181	374	180	3,460	14,173	-	759
RCV1_top10 (S3)	176	385	179	3,563	14,267	-	756
RCV1_top10 (S4)	193	391	184	3,460	14,244	-	808
RCV1_top10 (S5)	159	355	167	3,019	13,135	-	691

Table 6: Learning time (in seconds) of each method on the benchmark datasets.

can see that CTBN outperforms the other methods for most datasets. For example, CTBN is significantly better than CC on seven datasets, significantly better than MLKNN on nine datasets and significantly better than IBLR on six datasets (see the last row of Table 3). The only exception is the MMOC method, which outperforms CTBN on Yeast and Scene datasets. Although very accurate, MMOC is computationally very expensive (see Table 6) and does not scale up to large data.

Table 4 shows the CCL-loss of the methods. Note that we could not compute CLL-loss for MMOC because it is not a probabilistic method. We can see that CTBN clearly outperforms all other methods in terms of CLL-loss. The reason is that CTBN is learned to optimize the conditional log likelihood. Furthermore, it applies proper probabilistic inference for prediction, which produces very accurate probabilistic estimates. On the other hand, CC performs very poorly (it provides a bad estimate of the conditional distribution as noted by [5]) because of its ad-hoc classification heuristic. Table 5 shows that CTBN also produces competitive results in terms of the micro F1 score.

Lastly, Table 6 shows the learning times of the different methods. BR, CC and CHF are all very fast because they do not involve any structure learning. CTBN is also efficient and it can scale up to large data. We can see that on all of the RCV1_top10 datasets, CTBN is more than four times faster than MLKNN and fifteen times faster than IBLR. Finally, although very accurate, MMOC has a very high computational cost even on the smallest datasets (on the first three datasets, CTBN is around two orders of magnitude faster than MMOC).

6. CONCLUSION

In this paper, we proposed a novel probabilistic approach to multi-dimensional classification. Our approach encodes the conditional dependence relations between classes using a special tree-structured Bayesian network, whose conditional distributions are defined using probabilistic classifiers. We presented an efficient algorithm to learn the tree structure that maximizes the conditional log likelihood. Furthermore, we presented an efficient exact inference algorithm that has a linear complexity in the number of class variables. Our experimental evaluation on a broad range of datasets showed that our approach outperforms several state-of-the-art methods and produces much more reliable probabilistic estimates. Moreover, it is efficient and can scale up to large data.

7. ACKNOWLEDGMENT

This research work was supported by grants 1R21LM009102-01A1 and 1R01LM010019-01A1 from the NIH. Its content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

8. REFERENCES

[1] C. Bielza, G. Li, and P. Larrañaga. Multi-dimensional Classification with Bayesian Networks. *International Journal of Approximate Reasoning*, 52(6):705–727, 2011.

[2] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning Multi-label Scene Classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

[3] W. Cheng and E. Hüllermeier. Combining Instance-based Learning and Logistic Regression for Multilabel Classification. *Machine Learning*, 76(2):211–225, 2009.

[4] A. Clare and R. D. King. Knowledge Discovery in Multi-label Phenotype Data. In *PKDD*, 2001.

[5] K. Dembczynski, W. Cheng, and E. Hüllermeier. Bayes Optimal Multilabel Classification via Probabilistic Classifier Chains. In *ICML*, 2010.

[6] S. Godbole and S. Sarawagi. Discriminative Methods for Multi-labeled Classification. In *PAKDD*, 2004.

[7] D. Hsu, S. Kakade, J. Langford, and T. Zhang. Multi-label Prediction via Compressed Sensing. In *NIPS*, 2009.

[8] H. Kazawa, T. Izumitani, H. Taira, and E. Maeda. Maximal margin labeling for multi-topic text categorization. In *NIPS*, 2005.

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

[10] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, T. Mei, and H.-J. Zhang. Correlative Multi-label Video Annotation. In *the International Conference on Multimedia*, 2007.

[11] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier Chains for Multi-label Classification. In *ECML*, 2009.

[12] R. Schapire and Y. Singer. Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, pages 80–91, 1999.

[13] R. E. Tarjan. Finding Optimum Branchings. *Networks*, 7:22–35, 1977.

[14] L. van der Gaag and P. de Waal. Multi-dimensional Bayesian Network Classifiers. In *Probabilistic Graphical Models*, 2006.

[15] J. H. Zaragoza, L. E. Sucar, E. F. Morales, C. Bielza, and P. Larrañaga. Bayesian Chain Classifiers for Multidimensional Classification. In *IJCAI*, 2011.

[16] M.-L. Zhang and K. Zhang. Multi-label Learning by Exploiting Label Dependency. In *SIGKDD*, 2010.

[17] 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.

[18] M.-L. Zhang and Z.-H. Zhou. ML-KNN: A Lazy Learning Approach to Multi-label Learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

[19] Y. Zhang and J. Schneider. Multi-label Output Codes using Canonical Correlation Analysis. In *AISTATS*, 2011.

[20] Y. Zhang and J. Schneider. Maximum Margin Output Coding. In *ICML*, 2012.

[21] S. Zhu, X. Ji, W. Xu, and Y. Gong. Multi-labelled Classification using Maximum Entropy Method. In *SIGIR*, 2005.