An Optimization-based Framework to Learn Conditional Random Fields for Multi-label Classification

Mahdi Pakdaman Naeini∗ Iyad Batal† Zitao Liu‡ Charmgil Hong‡ Milos Hauskrecht‡

Abstract
This paper studies multi-label classification problem in which data instances are associated with multiple, possibly high-dimensional, label vectors. This problem is especially challenging when labels are dependent and one cannot decompose the problem into a set of independent classification problems. To address the problem and properly represent label dependencies we propose and study a pairwise conditional random Field (CRF) model. We develop a new approach for learning the structure and parameters of the CRF from data. The approach maximizes the pseudo likelihood of observed labels and relies on the fast proximal gradient descend for learning the structure and limited memory BFGS for learning the parameters of the model. Empirical results on several datasets show that our approach outperforms several multi-label classification baselines, including recently published state-of-the-art methods.

1 Introduction
In standard binary classification problem data instances are assigned to one of the two classes (or labels). In multi-label classification (MLC) problems data instances are associated with subsets of labels selected from a set of possible labels \( L \). Multi-label classification problems may arise in text categorization, where text documents are associated with topics the document covers; or in image analysis and retrieval where individual image can be associated with multiple labels reflecting, for example, objects shown in the image.

The multi-label classification problem can be cast as an \(|L|\) dimensional classification problem where each instance \( x \) is mapped to an output \( y \) defined by a \( \{0, 1\} \) vector of values of length \(|L|\), such that the \( i \)-th component of the \( y \) vector reflects whether the \( i \)-th label in \( L \) is absent or present when labeling \( x \).

In this work, we study the problem of learning multi-label classifiers from data. This problem is challenging for two reasons; the number of possible label assignments is exponential in the number of labels, and the occurrence of different labels and their combinations is not arbitrary and some label combinations may be more or less likely reflecting the dependencies among labels. This prompts us to find ways of representing the dependencies among labels in a compact and tractable form, and to devise algorithms capable of learning such dependencies from data.

Many different multi-label classification models and methods have been devised in recent years. In this work we propose, develop and test a model based on the Conditional Markov Random field (CRFs) [16]. Briefly, CRFs define a class of conditional probability models representing \( P(Y|X) \), where \( Y \) is the output label space and \( X \) is the input feature space. The CRF model is based on the Markov random field (MRF) representation [14] to model dependencies/interactions among components of \( Y \) using potential functions over subsets of variables. However, CRF extends (unconditional) MRF by permitting to condition the potential functions on features.

In this work we limit the dependencies expressed by our model to a pairwise conditional random Field (CRF) model, where each potential function depends on at most two label variables. We develop a new approach for learning the structure and parameters of such a CRF from data. The approach maximizes the pseudo likelihood of observed labels and relies on the fast proximal gradient descend [2] for learning the structure of the model, and limited memory Broyden-Fletcher-Goldfarb-Shanno algorithm (BFGS) [18] for learning the parameters of the model. Empirical results on several datasets from different domains show that our approach outperforms several multi-label classification baselines, including recently published state-of-the-art methods.

1.1 Related work The majority of existing methods for solving the multi-label classification (MLC)
problems fall into two main categories: dimensionality-reduction, and probabilistic approaches. Both of these methods try to find the inherent structure in data and represent the output label space more compactly.

The dimensionality reduction methods, such as singular value decomposition (SVD), attempt to reduce the dimensionality of the output space Y by defining an intermediate lower dimensional space \( Y' \) that is sufficient to model the relations between input features and output labels. Multiple approaches fall into this category including the work by Tai and Lin [22] who applied principal component analysis to find the intermediate lower dimensional space, or the work by Zhang and Schneider [31] who applied canonical correlation analysis. They also proposed a maximum margin formulation to find a low dimensional subspace by optimizing the subspace from both the feature side and the label side [32]. That is, the subspace should be easy to predict from the features and at the same time should discriminate well between the labels.

The probabilistic multi-label classification methods attempt to model the relation in between the input space X and output space Y probabilistically, by modeling the conditional distribution \( P(Y|X) \). The existing methods include classifier chains (CC) methods [20, 27]. Briefly, both of CC methods attempt to model \( P(Y|X) \) by decomposing it to a product of conditional densities based on the chain rule. The difference is that [20] approached the problem by using a random ordering of output variables, while [27] tried to optimize their ordering first. Other probabilistic MLC methods include conditional tree structured Bayesian networks (CTBN) [1], and multi-dimensional Bayesian networks [25]. CTBN represents the distribution of \( P(X|Y) \) by modeling the relations among the labels using a tree structured Bayesian network. Multi-dimensional Bayesian networks define the joint probability distribution \( P(X,Y) \) using a Bayesian network with a restricted structure that puts all class variables as ancestors of all feature variables.

Other approaches to multi-label classification include work by Clare et. al. [6] and Boutell et. al. [4] who studied methods for learning a set of independent classifiers. Godbole et. al. [10] and Cheng et. al. [5] improved these early attempts by introducing a second layer of classifiers that combine input features with the outputs of independent classifiers. Zhang et. al. proposed several multi-label prediction algorithms, such as RBF neural networks [28], multi-label lazy learning approach [30], multi-instance multi-label RBF neural networks [26], multi-label BP neural networks [29]. Ji et. al. [13] used matrix decomposition to generate orthogonal linear prediction functions. Hsu et. al. [12] proposed to use compressed sensing in the label space for multi-label prediction. Liu et. al. [19] formulated the problem as a constrained Non-negative Matrix Factorization (NMF) problem. Although these methods work well for a small number of labels, they do not scale-up well to higher dimensional input and output space.

In this paper, we propose a new MLC algorithm that learns a prediction model by modeling the structure of the output space using Conditional Random Fields (CRF). Our method learns jointly the structure and parameters of the CRF from data. CRF has been used for multi-label classification of text and image data [9, 21, 11]. Compared to our approach, these applications make additional simplifying assumptions; for example, they rely on an a priori fixed structure of the output space, and/or assume a discrete set of features.

## 2 Proposed Model

We first describe the parametrization of the proposed multi-label classification model in Section 2.1. Section 2.2 defines the log-pseudo likelihood as the objective function and formulates the optimization problem. Finally, Section 2.3 describes the optimization methods we use to learn the model.

### 2.1 Model Parametrization

Our multi-label classification model uses pairwise potential conditional random fields (CRF). CRF directly models the conditional distribution \( p(y|x) \) with an associated graphical structure [16]. A CRF under graph \( G = (V,E) \) with node potentials and pairwise potentials can be characterized as follows:

\[
\begin{align*}
\log p(y|x) &= \log Z(x) + \sum_{v \in V} \phi_i(y_v, x) + \sum_{(i,j) \in E} \phi_{ij}(y_i, y_j, x) \\
&= \sum_{y \in \mathbb{R}^m} \phi_i(y, x) \prod_{(i,j) \in E} \phi_{ij}(y_i, y_j, x)
\end{align*}
\]

where \( x \in \mathbb{R}^m \) is the feature vector for the input instance of multi-label classification model and \( y = (y_1, \ldots, y_d) \) is its corresponding output labels. Also, \( \phi_i \) is the node potential for every node (label) in V and \( \phi_{ij} \) is the edge potential for each edge (pair of labels) in E. Let \( d \) denotes the number of ys and \( Z \) denotes the partition (normalization) function \( Z(x) = \sum_{y_1, \ldots, y_d} \prod_{v \in V} \phi_i(y_v, x) \prod_{(i,j) \in E} \phi_{ij}(y_i, y_j, x) \).

In this paper we assume that node potentials and the edge potentials have the following form:

\[
\phi_i(y, x) = \langle e^{f_i(x)}v^o_i, e^{f_i(x)}v^i_i \rangle
\]
\begin{equation}
\phi_{ij}(\ldots, x) = \left( \begin{array}{c}
e f_{ij}(x)w_{ij}^{0,0} \\
e f_{ij}(x)w_{ij}^{0,1} \\
e f_{ij}(x)w_{ij}^{1,1} \end{array} \right),
\end{equation}

where \((v_i^0, v_i^1)\) are the node parameters for node \(i\) and \((w_{ij}^{0,0}, w_{ij}^{0,1}, w_{ij}^{1,1})\) are the edge parameters for edge \((i, j)\). Also, \(f_i(x)\) and \(f_{ij}(x)\) are row vectors of local features corresponding to node \(i\) and edge \((i, j)\). These features can be extracted using any feature extraction method such as (kernel) linear discriminant analysis, etc. However, for the simplicity we used all input features \(x\) as both local node and edge features. In addition, for the identifiability of the model we set \(v_i^1 = w_{ij}^{1,1} = 0\).

2.2 Objective Function The most common objective function used in probabilistic modeling is the log-likelihood function of training data. However, learning CRF parameters using log-likelihood requires inference in computing the gradient of the log-likelihood. If exact inference is used, it would be exponential in the tree width which is intractable for dense graphs [15]. To address this problem, one can either use approximate inference [17] or optimize a surrogate function that is easier to deal with.

The other objective function that is commonly used in probabilistic modeling is the log-pseudo likelihood of training data, which is known to give a consistent estimator of model parameters when the number of training data is sufficiently large [3]. Assume we have a set of \(N\) multi-label training instances \(D = \{(x_i^n; y_{1i}^n, \ldots, y_{N_i}^n)\}_{i=1}^N\). Defining \(\theta = \{v_i, w_{ij}\}\) as the set of all node and edge parameters of the model, the log-pseudo likelihood of training data is defined as follow:

\begin{equation}
lpl(D; \theta) = \sum_{n=1}^N \sum_{i=1}^d \log(P(y_i^n | y_{N(i)}^n, x^n))
\end{equation}

Using our model parameterizations in Equations 2.2, 2.3 we have:

\[P(y_i^n | y_{N(i)}^n, x^n) = \frac{\exp\{f_i(x^n)v_i^{y_i^n} + \sum_{j \in N(i)} f_{ij}(x^n)w_{ij}^{y_i^n, y_j^n}\}}{Z_i},\]

where \(Z_i\) is the local partition function which normalizes with respect to \(y_i\). Consequently, \(lpl\) is computed as follow:

\begin{equation}
lpl(D; \theta) = \sum_{i=1}^d \sum_{n=1}^N \left( f_i(x^n)v_i^{y_i^n} + \sum_{j \in N(i)} f_{ij}(x^n)w_{ij}^{y_i^n, y_j^n} \right)
\end{equation}

\[\begin{align*}
&- \log \left\{ \exp(f_i(x^n)v_i^{y_i^n}) + \sum_{j \in N(i)} f_{ij}(x^n)w_{ij}^{y_i^n, y_j^n} \right\} \\
&+ \exp\left( \sum_{j \in N(i)} f_{ij}(x^n)w_{ij}^{1,1} \right) \right) \right) \right)
\]

We can see in the above equation that \(lpl(D; \theta)\) is a concave function in both node and edge parameters.

To prevent model overfitting we use a regularization term \(R(\theta)\) that penalizes model complexity. In our method \(R(\theta)\) is defined as follow:

\begin{equation}
R(\theta) = \sum_{i=1}^d \lambda_v ||v_i, 0||^2_2
\end{equation}

\[+ \lambda_w \left( (1 - \eta) \sum_{e \in E} ||w_e||^2_2 + \eta \sum_{e \in E} ||w_e||_2 \right),\]

where \(w_e = (w_e^{0,0}; w_e^{0,1}; w_e^{1,1})\). The first term in \(R(\theta)\) is the ridge penalty (\(l_2\) norm regularization) over node parameters. The second term (inside the parentheses) is the elastic-net penalty [34], which is a compromise between the ridge regression penalty (\(\eta = 0\)) and the group lasso penalty (\(\eta = 1\)). This penalty imposes block sparsity effect over the edge parameters of the CRF model. That is, for a specific edge in the graph, either all its parameters go to zero (the edge is absent from the graph) or not (the edge exists in the graph). For setting the regularization parameters \(\lambda_v, \lambda_w, \eta\), we apply the following sequential cross validation process. We first set \(\lambda_v\) separately for each node \((y_i)\) using cross-validation over independent logistic regression models. After that, we apply the LARS (or homotopy) method [8] to set \(\lambda_w\) and \(\eta\), using a k-fold cross validation and a warm start procedure.

By defining \(nlpl \triangleq -lpl\) as the corresponding loss function, we can learn the structure and parameters of the model by minimizing the regularized loss function as following:

\begin{equation}
\theta^* = \operatorname{argmin}_\theta (nlpl(D; \theta) + R(\theta))
\end{equation}

Since both \(nlpl(D; \theta)\) and \(R(\theta)\) are convex functions, the above optimization program is a convex optimization program in which all local minima are global minima.
2.3 Optimization

In this section we describe the proposed optimization method for solving the optimization program in Equation 2.7. In order to use any gradient-based optimization, we should be able to compute the gradient of \( nplp \). Taking derivative with respect to \( \psi_i^0 \) (the parameters of node \( i \)), we would have the following:

\[
\nabla_{\psi_i^0} \ nplp = \sum_{n=1}^{N} -f_i(x^n)^T p(y^n_i = 0) + \sum_{n=1}^{N} f_i(x^n)^T P(y^n_i = 0 | y^n_{N(i)}, x^n),
\]

where superscript \( T \) denotes the transpose. Also taking the derivative of the \( nplp \) with respect to \( w_{ij} \) (the parameters of edge \((i, j)\) and values \((b_i, b_j)\)), we would have the following:

\[
\nabla_{w_{ij}} \ nplp = \sum_{n=1}^{N} -2f_{ij}(x^n)^T p(y^n_i = b_i, y^n_j = b_j) + \sum_{n=1}^{N} f_{ij}(x^n)^T \left( P(y^n_i = b_i | y^n_{N(i)}, x^n) p(y^n_j = b_j) + P(y^n_j = b_j | y^n_{N(j)}, x^n) p(y^n_i = b_i) \right)
\]

Fast Proximal Gradient Descent

Although the minimization problem in equation 2.7 is a convex optimization problem, the objective function is non-smooth due to the non-smoothness of the \( f_{1,2} \)-norm induced by the elastic-net regularization term in \( R(\theta) \). To overcome this difficulty, we use the fast proximal gradient decent algorithm \([2]\) to solve this non-smooth optimization problem.

The unconstrained problem in Equation 2.7 can be optimized with two approaches. The first approach directly defines proximal mapping for the non-smooth part of the elastic-net penalty i.e. the \( L_2 \) penalty over the edge variables. The second approach converts the unconstrained program to an equivalent constraint optimization program by defining slack variables \( \alpha_e \) as in Equation 2.10. These two approaches are very similar and they both have the same order of computations. However, we use the latter one mainly because in this case we can use the slack variables \( \alpha_e \) for monitoring existence of edges in the CRF model and for early termination of the structure learning phase (having \( \alpha_e = 0 \) means that edge \( e \) is not present in the graph structure). In the following, for the simplicity of representation, we use \( \theta_k \) to denote the parameters vector \( \{\alpha, w, v\} \) at iteration \( k \) and \( f(\theta_k) \) to denote its corresponding objective function.

\[
\text{minimize} \quad \left( nplp(\theta) + \sum_{i=1}^{d} \lambda_{\psi_i} \|\psi_i^0\|_2^2 \right)
\]

\[
\text{subject to} \quad \|w_e\|_2 \leq \alpha_e, \quad e \in E.
\]

For solving the optimization program in Equation 2.10, we use FPGD with backtracking line search. The high level idea of proximal gradient descend methods for minimizing a non-smooth function \( f \) is to write it as two separate parts \( f = g + h \), in which \( g \) is a smooth differentiable function and \( h \) is a simple non-differentiable convex function. In our case, \( g = nplp(D; \theta) + \lambda_e \sum_{i=1}^{d} \|\psi_i^n\|_2^2 + \lambda_w \left( (1 - \eta) \sum_{e \in E} \|w_e\|_2^2 + \eta \sum_{e \in E} \alpha_e \right) \) and the non-smooth function \( h = I_{\Sigma}(\theta) \), where \( \Sigma = \{ (\alpha, w, v) | \forall e \in E : \|w_e\|_2 \leq \alpha_e \} \) is the set of feasible solutions (a convex set) and \( I_{\Sigma} \) is the indicator function defined as follows:

\[
I_{\Sigma}(\theta) = \begin{cases} 
0 & \text{if } \theta \in \Sigma \\
\infty & \text{otherwise}
\end{cases}
\]

Proximal gradient based optimization methods use a proximal function defined as \( \text{prox}_\rho(x) = \text{argmin}_{z \in \mathbb{R}^n} \frac{1}{2\rho} \|x - z\| + h(z) \) to optimize a second order approximation of the objective function \( f \). In our case, the proximal function is the projection operator \( \Pi(\alpha) \) of the parameters onto the convex set of feasible solutions \( \Sigma \) (defined above). Interestingly, the above projection operator can be decoupled and solved separately over different edges of CRF model which gives following simple program for each edge:

\[
\text{minimize} \quad \| (w_e', \alpha_e') - (w_e, \alpha_e) \|_2^2
\]

\[
\text{subject to} \quad \|w_e'|_2 \leq \alpha_e'
\]

For solving the above optimization program we used the linear-time projection algorithm proposed in \([24]\). Algorithm 1 outlines the optimization method. In this algorithm the termination condition is defined based on the maximum number of iterations (50 in our experiments), the norm of the changes in value of \( \alpha \), or the change in the function values \( |f(\theta_k) - f(\theta_{k-1})| \).

995

Copyright © SIAM.
Unauthorized reproduction of this article is prohibited.
can work with several base classifiers. We use $L_2$-penalized logistic regression for all of these methods and choose their regularization parameters by cross-validation.

### 3.2 Evaluation Measures

Evaluating the performance of MLC methods is more difficult than evaluation of 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. However, this measure could be too harsh, specially when the output dimensionality is high. The other evaluation measure is the **log-likelihood loss** ($LL$) which computes the negative conditional log-likelihood of the test instances:

$$LL = \sum_{k=1}^{n} -\log \left( P(y^{(k)}|x^{(k)}) \right)$$

$LL$ is very insightful in evaluating the performance of probabilistic MLC methods because it evaluate how much probability mass is given to the true label vectors (the higher the probability, the smaller the loss). Micro F1 and Macro F1 are two other intuitive measures for evaluation MLC methods [23]. Macro F1 computes the F1 score for each label separately and averages them over all labels. On the other hand, micro F1 aggregates the true positives, false positives, true negatives and false negatives over all labels, and then calculates the overall F1 score. Note that these two measures do not account for the correlations between labels (see [7, 27]). However, we report them in our results because they have been used in several other papers [32, 33].

### 4 Experimental Results

We perform two sets of experiments to evaluate the proposed model. First, in order to evaluate the predictive performance of MLCRF, which is the main point of this paper, we perform experiments on several real-world data. In addition, we perform some experiments on synthetic data to verify that the FPGD-based structure learning algorithm of MLCRF also performs reasonably well in finding the true conditional dependencies among output labels.

#### 4.1 Real data
We use five real-world benchmark datasets to evaluate the performance of the MLC models. These datasets are obtained from different domains including image labeling (image and scene), biology (yeast), music recognition (emotions) and text classification (rcv1). Note that many labels in rcv1 data are rare, so we select the 10 most common labels to study. Table 1 shows the characteristics of these datasets. For
each multi-label dataset, we use $N$, $m$, $d$ to represent the number of instances, number of features (input dimension) and number of labels (output dimension), respectively. In addition, we show two statistics of the data: 1) label cardinality (LC), which measures the average number of labels per instance and 2) distinct label set (DLS), which is the number of all possible label combinations that appear in data.

Tables 2, 3, 4 and 5 show the results of experiments in terms of different evaluation metrics. Standard ten-fold cross validation is applied for all methods and the mean metric value is reported. Furthermore, to statistically measure the significance of performance difference, paired t-tests at 0.05 significance level are conducted between MLCRF and the other methods. Specifically, whenever MLCRF achieves significantly better/worse performance than the compared method on any dataset, a marker */$*$ is shown in the table. Note that the results of MLOCC are not shown on rcv1 data because it did not finish in the time limit of 24 hours for one iteration of learning.

Table 2 shows the exact match accuracy of the methods on the benchmark datasets. We can see that MLCRF outperforms all the other methods for most datasets.

Table 3 shows the performance of each method in terms of log-likelihood loss (LL). Since the output dimensionality of the datasets is relatively small, we use exact inference in the prediction time to obtain accurate probabilistic estimates$^1$. We can see that MLCRF is significantly superior to all other methods in terms of LL. This shows that the pseudo likelihood, which MLCRF optimizes, is a good surrogate for the likelihood of the data. Notice that CC performs poorly because of its ad-hoc classification heuristic$^7$. Also, note that we cannot compute LL for MLOCC because it is not a probabilistic method.

Tables 4 and 5 show that MLCRF also produces competitive results in terms of the micro and macro F1 scores.

### 4.2 Synthetic data
We perform simple experiments on synthetic data to verify that the structure learning part of the proposed method is performing reasonably well. In the following, we describe four different scenarios, each corresponds to a different graph topology with three output nodes, as shown in Figure 1. We generate $500$ data instances: $D = \{x^{(n)}, y^{(n)}\}_{n=1}^{500}$, where $x^{(n)} \in \mathbb{R}^6$ and $y^{(n)} \in \{0, 1\}^2$. The feature vectors are generated uniformly from $[-1, 1]^6$. On the other hand, the class vectors are generated based on the following strategies:

Independent (SD1): For each instance $x^{(n)}$, we first generate random variables $\mu_1, \mu_2, \mu_3$ that depends on the features as follows: $\mu_1 = \sigma(x_1^{(n)} + x_2^{(n)}), \mu_2 = \sigma(3x_3^{(n)} + x_4^{(n)}), \mu_3 = \sigma(5x_5^{(n)} + 5x_6^{(n)})$ where $\sigma$ is the logistic (sigmoid) function. $y_1^{(n)}, y_2^{(n)}, y_3^{(n)}$ will be bernoulli random variables generated using $\mu_1, \mu_2, \mu_3$. Note that all three classes are conditionally independent of each other.

Single Edge (SD2): For each instance $x^{(n)}$, we first generate random variables $\mu_1, \mu_2$ that depend on the features as follows: $\mu_1 = \sigma(x_1^{(n)} + 5x_2^{(n)}), \mu_2 = \sigma(3x_3^{(n)} + 3x_4^{(n)})$. $y_3$ will be bernoulli random variables generated using $\mu_1, \mu_2$. If the value of $y_3$ is zero, we assign $(y_1, y_2)^{(n)} = (0, 0)$ (both classes are absent). Otherwise (the value of $z$ is one), we assign $(y_1, y_2)^{(n)}$ to either $(1, 0)$ or $(0, 1)$ with equal probability (only one of the two classes is present). Note that we do not have any instance where both classes are present.

Chain Structure (SD3): For each instance $x^{(n)}$, we first generate random variables $\mu_1, \mu_2$ that depends on the features as follows: $\mu_1 = \sigma(x_1^{(n)} + x_2^{(n)}), \mu_2 = \sigma(3x_3^{(n)} + 3x_4^{(n)}). y_1^{(n)}, y_2^{(n)}$ will be bernoulli random variables generated using $\mu_1, \mu_2$. If both $y_1^{(n)}$ and $y_2^{(n)}$ are equal to one we set $y_3^{(n)} = 0$; Otherwise, we set $y_3^{(n)} = 1$.

Complete Graph (SD4): For each instance $x^{(n)}$, we first generate a random variable $\mu_1$ that depends on the features as follows: $\mu_1 = \sigma(x_1^{(n)} + x_2^{(n)}). z$ will be bernoulli random variables generated using $\mu_1$. If $z$ is equal to one we set $(y_1, y_2, y_3)^{(n)} = (0, 0, 0)$; Otherwise, we set $(y_1, y_2, y_3)^{(n)}$ randomly (with equal probability) to one of the three configurations $(1, 0, 0); (0, 1, 0); (0, 0, 1)$.

![Figure 1: Graphical model representation of four synthetic data sets.](image-url)
Table 1: Datasets characteristics.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>N</th>
<th>m</th>
<th>d</th>
<th>LC</th>
<th>DLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emotions</td>
<td>593</td>
<td>72</td>
<td>6</td>
<td>1.868</td>
<td>27</td>
</tr>
<tr>
<td>Yeast</td>
<td>2417</td>
<td>103</td>
<td>14</td>
<td>4.237</td>
<td>198</td>
</tr>
<tr>
<td>Scene</td>
<td>2407</td>
<td>294</td>
<td>6</td>
<td>1.074</td>
<td>15</td>
</tr>
<tr>
<td>Image</td>
<td>2000</td>
<td>135</td>
<td>5</td>
<td>1.24</td>
<td>20</td>
</tr>
<tr>
<td>rcv1_top10</td>
<td>6000</td>
<td>8367</td>
<td>10</td>
<td>1.310</td>
<td>76</td>
</tr>
</tbody>
</table>

Table 2: Performance of each method on the benchmark datasets in terms of exact match accuracy. Marker */non-star indicates whether MLCRF is statistically superior/inferior to the compared method (using paired t-test at 0.05 significance level).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>BR</th>
<th>CHF</th>
<th>CC</th>
<th>PCC</th>
<th>MLKNN</th>
<th>IBLR</th>
<th>MMOC</th>
<th>MLCRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>0.281*</td>
<td>0.360*</td>
<td>0.426*</td>
<td>0.449</td>
<td>0.346*</td>
<td>0.387*</td>
<td>0.448</td>
<td>0.452</td>
</tr>
<tr>
<td>scene</td>
<td>0.541*</td>
<td>0.605*</td>
<td>0.632*</td>
<td>0.666*</td>
<td>0.629*</td>
<td>0.644*</td>
<td>0.664*</td>
<td>0.687</td>
</tr>
<tr>
<td>yeast</td>
<td>0.151*</td>
<td>0.163*</td>
<td>0.193*</td>
<td>0.230</td>
<td>0.179*</td>
<td>0.204*</td>
<td>0.219</td>
<td>0.225</td>
</tr>
<tr>
<td>emotions</td>
<td>0.265*</td>
<td>0.300*</td>
<td>0.268*</td>
<td>0.317*</td>
<td>0.283*</td>
<td>0.335*</td>
<td>0.332*</td>
<td>0.371</td>
</tr>
<tr>
<td>rcv1_top10</td>
<td>0.439*</td>
<td>0.456*</td>
<td>0.497*</td>
<td>0.519*</td>
<td>0.276*</td>
<td>0.411*</td>
<td>-</td>
<td>0.541</td>
</tr>
</tbody>
</table>

Table 3: Performance of each method on the benchmark datasets in terms of log-likelihood loss. Marker */non-star indicates whether MLCRF is statistically superior/inferior to the compared method (using paired t-test at 0.05 significance level).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>BR</th>
<th>CHF</th>
<th>CC</th>
<th>PCC</th>
<th>MLKNN</th>
<th>IBLR</th>
<th>MMOC</th>
<th>MLCRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>0.479*</td>
<td>0.541*</td>
<td>0.550*</td>
<td>0.565*</td>
<td>0.504*</td>
<td>0.573</td>
<td>0.572</td>
<td>0.576</td>
</tr>
<tr>
<td>scene</td>
<td>0.696*</td>
<td>0.722</td>
<td>0.697*</td>
<td>0.722*</td>
<td>0.736</td>
<td>0.758*</td>
<td>0.711*</td>
<td>0.733</td>
</tr>
<tr>
<td>yeast</td>
<td>0.635</td>
<td>0.637</td>
<td>0.628</td>
<td>0.645@</td>
<td>0.646@</td>
<td>0.661@</td>
<td>0.651@</td>
<td>0.629</td>
</tr>
<tr>
<td>emotions</td>
<td>0.645*</td>
<td>0.672*</td>
<td>0.621*</td>
<td>0.664*</td>
<td>0.656*</td>
<td>0.692</td>
<td>0.687</td>
<td>0.697</td>
</tr>
<tr>
<td>rcv1_top10</td>
<td>0.582*</td>
<td>0.597</td>
<td>0.595</td>
<td>0.600</td>
<td>0.314*</td>
<td>0.566*</td>
<td>-</td>
<td>0.598</td>
</tr>
</tbody>
</table>

Table 4: Performance of each method on the benchmark datasets in terms of micro F1. Marker */non-star indicates whether MLCRF is statistically superior/inferior to the compared method (using paired t-test at 0.05 significance level).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>BR</th>
<th>CHF</th>
<th>CC</th>
<th>PCC</th>
<th>MLKNN</th>
<th>IBLR</th>
<th>MMOC</th>
<th>MLCRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>0.486*</td>
<td>0.546*</td>
<td>0.562*</td>
<td>0.575*</td>
<td>0.516*</td>
<td>0.581</td>
<td>0.578</td>
<td>0.585</td>
</tr>
<tr>
<td>scene</td>
<td>0.703*</td>
<td>0.730*</td>
<td>0.709*</td>
<td>0.729*</td>
<td>0.743</td>
<td>0.765@</td>
<td>0.721*</td>
<td>0.746</td>
</tr>
<tr>
<td>yeast</td>
<td>0.457</td>
<td>0.461</td>
<td>0.467</td>
<td>0.486</td>
<td>0.478</td>
<td>0.498</td>
<td>0.473</td>
<td>0.476</td>
</tr>
<tr>
<td>emotions</td>
<td>0.632*</td>
<td>0.667*</td>
<td>0.620*</td>
<td>0.659*</td>
<td>0.656*</td>
<td>0.690</td>
<td>0.679*</td>
<td>0.695</td>
</tr>
<tr>
<td>rcv1_top10</td>
<td>0.500*</td>
<td>0.526</td>
<td>0.537</td>
<td>0.534</td>
<td>0.257*</td>
<td>0.487*</td>
<td>-</td>
<td>0.525</td>
</tr>
</tbody>
</table>

Table 5: Performance of each method on the benchmark datasets in terms of macro F1. Marker */non-star indicates whether MLCRF is statistically superior/inferior to the compared method (using paired t-test at 0.05 significance level).

The results on synthetic data are shown in Figure 2. For each dataset, we profile the value of the auxiliary edge variables (the $\alpha_i$ variables in Equation 2.10) over the course of the optimization algorithm. Note that these variables imply the graph structure because having a positive $\alpha_{i,j}$ means having an edge between class $Y_i$ and class $Y_j$ in the graph. As we can see, for all four datasets, the algorithm succeeds in finding the correct dependency relations among the outputs.
direction for future work is to use a max margin loss function (e.g. Hinge loss) in order to define a max-margin CRF for multi-label classification [23, 24, 26]. We conjecture that the max margin approach for learning of a CRF will lead to a more accurate MLC model when considering the exact label match, while our CRF model would perform better in terms of the conditional log-likelihood loss.

6 Acknowledgement
This work was supported by grants 1R01GM088224 and 1R01LM010019 from the NIH. Its content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

References


