Tractable Semi-Supervised Learning of Complex Structured Prediction Models

Kai-Wei Chang[1], S. Sundararajan[2], and S. Sathiya Keerthi[3]

[1]Dept. of Computer Science, University of Illinois at Urbana-Champaign, IL USA
[2]Microsoft Research India, Bangalore, INDIA
[3]Cloud and Information Services Lab, Microsoft, Mountain View, CA
kchang10@illinois.edu,{ssrajan,keerthi}@microsoft.com

Abstract. Semi-supervised learning has been widely studied in the literature. However, most previous works assume that the output structure is simple enough to allow the direct use of tractable inference/learning algorithms (e.g., binary label or linear chain). Therefore, these methods cannot be applied to problems with complex structure. In this paper, we propose an approximate semi-supervised learning method that uses piecewise training for estimating the model weights and a dual decomposition approach for solving the inference problem of finding the labels of unlabeled data subject to domain specific constraints. This allows us to extend semi-supervised learning to general structured prediction problems. As an example, we apply this approach to the problem of multi-label classification (a fully connected pairwise Markov random field). Experimental results on benchmark data show that, in spite of using approximations, the approach is effective and yields good improvements in generalization performance over the plain supervised method. In addition, we demonstrate that our inference engine can be applied to other semi-supervised learning frameworks, and extends them to solve problems with complex structure.

1 Introduction

Over the past decade, a variety of semi-supervised learning methods have been suggested in the literature; these methods use unlabeled data to yield a good lift in generalization performance when labeled data is sparse. One particular model, initially proposed by Joachims [12] for binary support vector machines, has given impressive results on problems involving large feature spaces, such as those encountered in text classification and natural language processing. This model has been nicely extended to multi-class classification, ordinal regression and structured output problems [31, 5, 2]. The key ideas behind these methods are: (a) using the labels of the unlabeled data ($\mathbf{Y}_U$) as extra variables and the associated loss function in training; (b) optimizing the model weight vector (θ) and $\mathbf{Y}_U$ via alternating optimization steps; (c) using constraints on $\mathbf{Y}_U$ that come from domain knowledge to effectively guide the training towards good solutions; and (d) employing annealing to avoid getting caught in local minima.

The probabilistic structured output model in [5] is useful only when the output structure is simple, e.g., linear chains, that allows inference and learning computations to be done in a tractable fashion. In this paper we go beyond and focus on problems with more complex output structure. For such problems, making approximations in inference and

learning is inevitable. In many practical situations, the complex output structure is such that: (a) the output has several components with sparse inter-component intersections; and (b) learning computations involving each component viewed as a separate piece is tractable (known as piecewise training [32]). Thus, if we replace the likelihood terms in the semi-supervised training objective function with composite likelihood [20] terms involving the components, then the θ determination step becomes tractable; we do not obtain the component marginals from the underlying intractable joint distribution. The components-based structure also lets us to use dual decomposition based inference techniques [16, 14] which have matured over recent years. The use of these approximations in the semi-supervised learning method is neat since alternating optimization allows the two approximations to be used independently in an iterative loop without crossing each other.

We illustrate our approach by applying it to the problem of multi-label classification and developing all the details for it. Detailed empirical analysis on several benchmark datasets shows the effectiveness of the approach. Despite the approximations, the semi-supervised method gives good lift in performance over the plain supervised method. Also, when tested on small datasets where exact inference and learning are feasible, the generalization performance of our method is competitive with that obtained by the semi-supervised learning method using exact inference and exact learning. Furthermore, our inference engine could be used with other semi-supervised learning methods; experimental comparison with such methods show that the proposed method performs significantly better.

The rest of the paper is organized as follows. We review the related work in Section 2. In Section 3 we provide a background on semi-supervised learning, label assignment problem, and composite likelihood. Section 4 gives the details behind the determination of $\mathbf{Y}_U$ subject to domain constraints. Section 5 develops the details of the proposed method for the multi-label classification problem. Experimental results are given in Section 6. Section 7 concludes the paper.

2 Related Work

Some related works have a resemblance to the method proposed in this paper. Composite likelihood methods [20], Piecewise training [32], message-passing algorithm [10], and large-margin methods with approximate inference [30, 6, 18] have been proposed for approximate learning. However, they have been used only in the supervised learning. Dual decomposition methods have been combined with training in the supervised learning of large margin models [26, 15], but the ideas do not transfer to semi-supervised learning and probabilistic models.

Approximate inference algorithms for general structure (e.g., [9, 14, 16, 28, 25]) have been widely studied in the literature. However, most of them do not consider solving the inference problem with constraints from prior knowledge. On the other hand, many studies have been conducted to solve specific inference problems with constraints (e.g., [17, 3]). Recently, Martins et al. [24] proposed a decomposition method to solve general label assignment problems with first-order logic constraints. However, it is not clear how to inject the corpus-level constraints into their framework. In this paper, we extend

a dual decomposition method [16] to solve general constrained inference problem involved in our semi-supervised learning framework.

Several semi-supervised learning algorithms [2, 23, 36, 40, 1, 34, 7] have been proposed in the literature. However, with a few exceptions, they all focus on the cases that exact inference and learning are tractable. In such cases, there is no need to use an approximate algorithm. Note that for some problems with a special structure, efficient exact algorithms have been developed (e.g., Viterbi algorithm for linear chain structure). They might be faster than a general approximate inference/learning tool. We refer the readers to an example in [3, Section 6.1], where they showed that by using a dynamic programming algorithm to leverage the problem structure, a Lagrangian relaxation method is more efficient than a general approximate inference solver. In addition, some works have been conducted to study semi-supervised learning approach for multi-label classification problems [4, 8, 22, 38]. However, they do not use constraints from domain knowledge and the settings are different from ours.

In the following, we briefly discuss the connection of our method to other semi-supervised learning frameworks. Our work is closely related to [5] and [29]. These probability models have been shown to generate the state-of-the-art results on problems with a tractable structure, but they are not directly applicable to problems with complex structured outputs. When exact inference and learning are tractable, our model is reduced to a probability model similar to them. Without corpus level constraints, our model is also related to CoDL [2]. However, CoDL only demonstrate the results using exact inference algorithm and Perceptron style learning steps, while our probabilistic method is more general. The posterior regularization [7] and Generalized Exception [23] are probabilistic methods that has several key differences from our method: (a) they enforce domain constraints only in an expectation sense; and (b) it is unclear if they applied to problems with complex structured outputs. Transductive SVM proposed in [40] considers extending structured SVM to a semi-supervised setting. The method is complicated and the study is only conducted on problems with simple structures (linear chain and multi-class). Moreover, in their experiments they do not incorporate prior knowledge via constraints, which are crucial to get good performance in semi-supervised learning. Yu [36] also considers an extension of a large-margin method and regularizes the model with the labels assigned to the unlabeled data. Their framework is different from us, and it is unclear how to train their model on problems with complex structure. Lee et al. [19] proposed a semi-supervised discriminative random field algorithm with approximate inference. However, their method doesn't incorporate constraints. We will show the value of using constraints in Section 6.2.

Our primary focus is on semi-supervised structured prediction problems whose output structure is too complex to allow exact inference and learning. Therefore, comparing different semi-supervised learning framework on the problems with tractable structure is not the focus of this paper. Nevertheless, we show that our inference engine can be plugged in other semi-supervised learning frameworks such as transductive SVM [40] and CoDL [2] in Section 6.5.

3 Semi-Supervised Learning

3.1 Learning Problem

Consider a structured output problem in which a data instance consists of an input vector $\mathbf{x} \in \mathcal{X}$ and a label vector $\mathbf{y} \in \mathcal{Y}$. For example, in sequence labeling, $\mathbf{x}$ is a sequence of tokens $\{x^1, \ldots, x^l\}$ and $\mathbf{y}$ is a sequence of scalar labels $\{y^1, \ldots, y^l\}$. We are interested in discriminative models to determine $\mathbf{y}$ for given $\mathbf{x}$. This is done by using a feature vector $f(\mathbf{x}, \mathbf{y})$ and a parameter vector θ which define a scoring function $s(\mathbf{x}, \mathbf{y}, \theta) = \theta \cdot f(\mathbf{x}, \mathbf{y})$. Then inference is done as $\mathbf{y}^* = \arg\max_{\mathbf{y}} s(\mathbf{x}, \mathbf{y}, \theta)$. We assume that the scoring function $s(\mathbf{x}, \mathbf{y}, \theta)$ can be written as:

$$s(\mathbf{x}, \mathbf{y}, \theta) = \sum\nolimits_c \phi_c(\mathbf{y}_{\pi_c}), \tag{1}$$

where c is some break-up of $s(\cdot)$ into sub-scores, $\pi_c \subset \{1, \ldots, N\}$ is an index set associated with c, and $\mathbf{y}_{\pi_c} = \{y_j : j \in \pi_c\}$ is the label assignment on c. For this paper c can be taken as a break up into sub-problems such that each sub-problem, $\arg\max_{\mathbf{y}_{\pi_c}} \phi_c(\mathbf{y}_{\pi_c})$ is easy to solve, and the dependency among the variables $\mathbf{y}_{\pi_c}$ is considered only within the component c. For example, when single variables are considered, we are ignoring the label dependency and using a simple model for marginal probabilities instead of computing them using the entire graph. More examples can be found in, for example, [39]. Note that we have suppressed the dependency of the potentials on $\mathbf{x}$ and θ in (1) for ease of notation.

For probabilistic models, we can define conditional probability using the scoring function: $p(\mathbf{y}|\mathbf{x}, \theta) \propto \exp(s(\mathbf{x}, \mathbf{y}, \theta))$. If $(\mathbf{X}, \mathbf{Y})$ is a set of data instances $\{(\mathbf{x}, \mathbf{y})\}$, then $p(\mathbf{Y}|\mathbf{X}, \theta)$ can be re-written as the product of $p(\mathbf{y}|\mathbf{x}, \theta)$ assuming samples are drawn i.i.d from a fixed distribution. For ease of notation we will simply refer to these quantities as $p_\theta(\mathbf{Y})$ and $p_\theta(\mathbf{y})$.

Let $(\mathbf{X}_L, \mathbf{Y}_L) = \{(\mathbf{x}_L, \mathbf{y}_L)\}$ denote the set of all labeled instances. Consider the supervised learning problem of determining θ by maximizing the regularized log-likelihood:

$$\max_\theta S(\theta) = R(\theta) + \mathcal{L}(\mathbf{Y}_L; \mathbf{X}_L, \theta),$$

where $R(\theta) = -\|\theta\|^2/2\sigma^2$ is a regularizer and $\mathcal{L}(\mathbf{Y}_L; \mathbf{X}_L, \theta) = \frac{1}{n_L} \sum_{\mathbf{x}_L} \mathcal{L}_{\mathbf{x}_L}(\mathbf{y}_L; \mathbf{x}_L, \theta)$ is the log likelihood term and n_L is the number of labeled instances. $\mathcal{L}_{\mathbf{x}_L}$ is the instance level log likelihood; in the probabilistic model

$$\mathcal{L}_{\mathbf{x}_L}(\mathbf{y}_L; \mathbf{x}_L, \theta) = \log \frac{\exp(s(\mathbf{x}_L, \mathbf{y}_L, \theta))}{\sum_{\mathbf{y} \in \mathcal{Y}} \exp(s(\mathbf{x}_L, \mathbf{y}, \theta))}.$$

In semi-supervised learning, θ is learned from both labeled data $(\mathbf{X}_L, \mathbf{Y}_L)$ and unlabeled data $\mathbf{X}_U$, and we consider the following optimization problem instead:

$$\max_{\theta, \mathbf{Y}_U} S(\theta) + \mathcal{L}(\mathbf{Y}_U; \mathbf{X}_U, \theta) \ \text{ s.t. } \mu(\mathbf{X}_U, \mathbf{Y}_U) \geq \mathbf{c}, \tag{2}$$

where $\mathbf{Y}_U$ is the labels assigned to the unlabeled data during the learning. The vector valued domain constraints $\mu(\mathbf{X}_U, \mathbf{Y}_U) \geq \mathbf{c}$ (discussed below) are included to guide the semi-supervised learning algorithm towards good solutions.

We can perform alternating optimization over θ and $\mathbf{Y}_U$ to solve (2). While the optimization of θ can be done using a standard gradient based optimization routine such as LBFGS, optimization of $\mathbf{Y}_U$ can be done by solving a label assignment problem with constraints. If the structure is complex, both, optimizing θ and optimizing $\mathbf{Y}_U$ will be difficult. In the following, we will discuss how to use approximate inference and learning algorithms to learn θ and $\mathbf{Y}_U$.

3.2 Label Assignment Problem (Inference)

To simplify notations let us use $\mathbf{Y}$, $\mathbf{X}$, $\mathbf{y}$ and $\mathbf{x}$ to represent $\mathbf{Y}_U$, $\mathbf{X}_U$, $\mathbf{y}_U$ and $\mathbf{x}_U$. The label assignment problem subject to inequality constraints is given by:

$$\max_{\mathbf{Y}} \mathcal{L}(\mathbf{Y}; \mathbf{X}, \theta) \quad \text{s.t.} \;\; \mu(\mathbf{X}, \mathbf{Y}) \geq \mathbf{c}. \tag{3}$$

We can use (1) to rewrite (3) as:

$$\max_{\mathbf{Y}} \frac{1}{n_u} \sum\nolimits_i \sum\nolimits_c \phi_{i,c}(\mathbf{y}_{i,\pi_c}) \quad \text{s.t.} \;\; \mu(\mathbf{X}, \mathbf{Y}) \geq \mathbf{c}, \tag{4}$$

where the index i refers to i^{th} unlabeled example, c refers to a clique and n_u denotes the number of unlabeled examples. We assume that the constraint function μ can be written as: $\mu(\mathbf{X}, \mathbf{Y}) = \frac{1}{n_u} \sum_i \mu_i(\mathbf{x}_i, \mathbf{y}_i)$. We will also assume that, for each example, say the i-th, μ_i decomposes clique-wise, in the same way as (1). Thus, $\mu_i(\mathbf{x}, \mathbf{y})$ can be written as: $\sum_c \gamma_{ic} \mu_{ic}(\mathbf{x}, \mathbf{y}_c)$ where some γ_cs can be zero. These assumptions hold in most practical structured output scenarios; see [2, 7, 5] for many examples of constraints arising in different problems. Later we will describe constraints arising in multi-label classification. A constraint could be *instance level* (e.g., a particular label has to occur only once in an example) or *corpus level* (e.g., the number of occurrences of a particular label in all the examples is some number). Both these types of constraints fall in the general constraint function format described above. Note that an instance level constraint is a special case of a corpus level constraint and is obtained by setting $\mu_i(\mathbf{x}_i, \mathbf{y}_i) = 0$ for all i except one of them.

If all the constraints are *instance level*, the solution to (4) can be obtained by solving the inference problem on each example independently. Otherwise, joint inference is required. We will discuss the joint inference approach later. The inference problem can be solved exactly and efficiently only for restricted structured output types (e.g., linear chain, tree with low-width). For general structured output problems, *Master-Slave* type methods have been proposed to solve the inference problem.

3.3 Composite Likelihood Maximization (Learning)

In general structured prediction problems with graphs having cycles, learning algorithms for probabilistic models are intractable due to the need to handle the partition function. To alleviate the computational intractability of parameter estimation in supervised learning, *composite marginal or conditional likelihood* maximization [20] and *piecewise training* methods [32] have been proposed. Such models are useful not only to reduce computational complexity but also to provide robustness to model misspecification via using the simpler interactions. We make use of the piecewise training approach to learn the model parameter vector θ in our semi-supervised learning

setting. In this approach, the likelihood is approximated by the composition of likelihoods of the components. That is, $p_\theta(\mathbf{y}) = \exp\{\mathcal{L}(\mathbf{y};\theta,\mathbf{x})\} \approx \prod_c p(\mathbf{y}_{\pi_c};\theta_c)$, where $p(\mathbf{y}_{\pi_c};\theta_c) = \frac{\exp(\phi_c(\mathbf{y}_{\pi_c};\theta_c))}{\mathcal{Z}_c}$ and $\mathcal{Z}_c = \sum_{\mathbf{y}_{\pi_c}} \exp(\phi_c(\mathbf{y}_{\pi_c};\theta_c))$. We assume that each component c is tractable (e.g., $\mathbf{y}_{\pi_c}$ is a small subset of variables or c is a tree). Note that we are not marginalizing any underlying intractable joint distribution $p(\mathbf{y};\theta)$.

Thus we replace the log-likelihood term $\mathcal{L}(\mathbf{Y};\mathbf{X},\theta)$ with $\mathcal{L}_C(\mathbf{Y};\mathbf{X},\theta) = \sum_c \mathcal{L}_C^c$, $\mathcal{L}_C^c = \sum_i \phi_c(\mathbf{y}_{i,c}) - \sum_i \log(\mathcal{Z}_{i,c})$. Note that the first term (which is critical for inference) remains the same as in the *full* likelihood maximization of general structured prediction models. It is in the second term (involving the partition function) where we make a tractable simplification. It is possible to learn the model parameters of components independently in situations where there is no overlap of parameters between components. In general, we allow overlaps (i.e., share parameters across the components) keeping tractability in mind so that the parameters of components are optimized together. However, unlike [32] where the components considered are factors of the graphical model, we allow general user specified components involving more than one factor, as long as inference on them is tractable. (In section 5, we illustrate this approach on the multi-label classification problem, using trees as components sharing model parameters in a fully connected graph.)

With this approach, the semi-supervised learning problem (2) can be written as:

$$\max_{\theta,\mathbf{Y}_U} R(\theta) + \mathcal{L}_C(\mathbf{Y}_L;\mathbf{X}_L,\theta) + C_m\mathcal{L}_C(\mathbf{Y}_U;\mathbf{X}_U,\theta) \quad \text{s.t.} \quad \mu(\mathbf{X}_U,\mathbf{Y}_U) \geq \mathbf{c}. \quad (5)$$

where C_m is a regularization parameter introduced to provide annealing capability, which we discuss next.

3.4 Annealing steps for solving (5)

The objective function in (5) (and in (2)) is a non-concave function. In practice, we can apply annealing steps using the parameter C_m to avoid being trapped in a bad local minimum. The optimization procedure has a double loop. In the outer loop, we gradually increase C_m from a small positive value (e.g., tripling the value in every iteration starting from 10^{-4}) to one. This allows the unlabeled data to gradually influence the modeling process in achieving a better optimum. In the inner loop, we alternatively update the label assignment $\mathbf{Y}_U$ and the model θ, where θ is updated using the LBFGS routine [21]. For efficiency sake we set the maximal number of LBFGS iterations to be 25 and stop the inner loop after five rounds; we found this sufficient to get good solutions.

4 Solving Label Assignment Problems With Constraints

We begin by discussing a master-slave approach [16] for the inference problem without constraints. Then, we describe a joint approximate inference algorithm for the label assignment problem with corpus level constraints.

4.1 Master-Slave Approach

Let $\mathbf{y} = \{y_1, \ldots, y_N\}$ be a vector of random variables associated with an example; $y_j \in \mathcal{Y}_j$ where $\mathcal{Y}_j$ is a discrete set. Consider the inference problem for one example without any constraints.

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \sum\nolimits_c \phi_c(\mathbf{y}_{\pi_c}). \tag{6}$$

Note that to avoid notational complexity, we use the same notations as in (1) and assume that the score is re-written as a composition of scores on sub-problems such that each sub-problem c is tractable. Let $\pi_j^{-1} = \{c : j \in \pi_c\}$. For each $a \in \mathcal{Y}_j$, introduce a binary integer variable $z_{j,a}$ indicating whether y_j assumes the value a. Let $\mathbf{z}_j = (z_{j,1}, \ldots, z_{j,|\mathcal{Y}_j|})$, a binary vector. Let Z_j be the set of vector values taken by $\mathbf{z}_j$ and $\mathbf{z}$ be a single vector that collects all $\mathbf{z}_j$. So, there exists an invertible mapping between $\mathbf{z}$ and $\mathbf{y}$. Denote it by $\mathbf{y} = \text{bip}(\mathbf{z})$ and $\mathbf{z} = \text{bip}^{-1}(\mathbf{y})$. Let $\mathbf{z}_{\pi_c} = \{z_j : j \in \pi_c\}$. We slightly abuse notations and write $\phi_c(\mathbf{z}_{\pi_c}) = \phi_c(\text{bip}(\mathbf{z}_{\pi_c}))$. Let $Z_{\pi_c} = \prod_{j \in \pi_c} Z_j$. With these notations, we can rewrite (6) as

$$\max_{\mathbf{z}} \sum\nolimits_c \phi_c(\mathbf{z}_{\pi_c}) \quad \text{s.t.} \quad z_j \in Z_j, \;\; j = 1, \ldots, N. \tag{7}$$

In the master-slave approach, new variable vectors $\{\mathbf{z}^c_{\pi_c}\}$ are introduced for each sub-problem and constraints connecting them are introduced via a variable vector $\bar{\mathbf{z}}$ controlled by the master that coordinates an iterative optimization process. Using these variables we can rewrite (7) as

$$\max_{\bar{\mathbf{z}}, \{\mathbf{z}^c_{\pi_c}\}} \sum\nolimits_c \phi_c(\mathbf{z}^c_{\pi_c}) \quad \text{s.t.} \quad \mathbf{z}^c_{\pi_c} \in Z_{\pi_c} \;\; and \;\; \mathbf{z}^c_{\pi_c} = \bar{\mathbf{z}}_{\pi_c} \;\; \forall c.$$

Then, the Lagrangian min-max dual problem can be written as [16]

$$\min_{\nu} \;\; \sum\nolimits_c \max_{\mathbf{z}^c_{\pi_c}} \Big(\phi_c(\mathbf{z}^c_{\pi_c}) + \sum\nolimits_{j \in \pi_c} \langle \nu_{c,j}, (\mathbf{z}^c_{\pi_c})_j \rangle \Big) \quad \text{s.t.} \sum\nolimits_{c \in \pi_j^{-1}} \nu_{c,j} = 0 \;\; \forall j.$$

This problem can be solved using methods such as the projected sub-gradient method [16] or the accelerated dual method [14]. Since we are discussing the case without constraints, the inference problem for each example is independent and so we simply have to repeat the above described method to all examples.

4.2 Joint Inference with Constraints

For corpus level constraints joint inference over all examples is needed. Notations become a bit clumsy: when dealing with all examples we need to use $\mathbf{y}_{i,\pi_c}, \bar{\mathbf{z}}_{i,\pi_c}$ etc., instead of $\mathbf{y}_{\pi_c}, \bar{\mathbf{z}}_{\pi_c}$ etc. We also use $\mathbf{z}_{i,\pi_c}$ as a shorthand for $\mathbf{z}^c_{i,\pi_c}$. Let C_i be the set of components associated with example i. The m-th constraint can be written as: $\sum_i \sum_{c \in \mathcal{C}_i} \gamma_{m,i,c} \mu_{m,i,c}(\mathbf{y}_{i,\pi_c}) \geq c_m$. Then the joint optimization problem is given by:

$$\begin{aligned} \max_{\{\bar{\mathbf{z}}_i\}, \{\mathbf{z}_{i,\pi_c}\}} \quad & \sum\nolimits_i \sum\nolimits_{c \in \mathcal{C}_i} \phi_{i,c}(\mathbf{z}_{i,\pi_c}) \\ \text{s.t.} \quad & \mathbf{z}_{i,\pi_c} \in Z_{\pi_c} \;\; and \;\; \mathbf{z}_{i,\pi_c} = \bar{\mathbf{z}}_{i,\pi_c} \;\; \forall i, c \\ & \sum\nolimits_i \sum\nolimits_{c \in \mathcal{C}_i} \gamma_{m,i,c} \mu_{m,i,c}(\mathbf{z}_{i,\pi_c}) \geq c_m, \forall m. \end{aligned}$$

Let us define the following functions: $\tilde{\phi}_{i,c}(\mathbf{z}_{i,\pi_c}) = \phi_{i,c}(\mathbf{z}_{i,\pi_c}) + g(\mathbf{z}_{i,\pi_c})$, $g(\mathbf{z}_{i,\pi_c}) = \sum_m \eta_m(\gamma_{m,i,c}\mu_{m,i,c}(\mathbf{z}_{i,\pi_c}) - c_m)$, $h(\nu_c^{(i)}; \mathbf{z}_{i,\pi_c}) = \sum_{j\in\pi_{i,c}} \langle \nu_{c,j}^{(i)}, (\mathbf{z}_{i,\pi_c})_j \rangle$. Then, the corresponding Lagrangian min-max problem is given by:

$$\min_{\{\{\nu^{(i)}\},\{\eta_m\}\}} \max_{\{\mathbf{z}_{i,\pi_c}\}} \sum_i \sum_{c\in\mathcal{C}_i} \left(\tilde{\phi}_{i,c}(\mathbf{z}_{i,\pi_c}) + h(\nu_c^{(i)}; \mathbf{z}_{i,\pi_c})\right)$$
$$\text{s.t.} \quad \mathbf{z}_{i,\pi_c} \in Z_{\pi_c} \quad \sum_{c\in\pi_{i,j}^{-1}} \nu_{c,j}^{(i)} = 0 \ \ \forall i, j, \eta_m \geq 0, \ \ \forall m.$$

We use the projected sub-gradient method to solve this problem since the inner *maximization* is not differentiable. Note that for fixed $\{\eta_m\}$, the dual variables $\{\nu^{(i)}\}$ can be solved independently for each i; this is possible due to the decomposable nature of the constraint functions across the examples. The examples get coupled only via the domain constraint dual variables. Therefore, we use an alternate optimization strategy of optimizing $\{\eta_m\}$ and $\{\nu^{(i)}\}$. Essentially, we run a projected sub-gradient based algorithm over several iterations in an inner loop, and run a similar algorithm in an outer loop.

Optimizing $\{\nu^{(i)}\}$ with fixed $\{\eta_m\}$. Assume that $\{\eta_m\}$ is fixed and consider the sub-problem involving the *i-th* example given below:

$$\min_{\{\{\nu^{(i)}\}} \max_{\{\mathbf{z}_{i,\pi_c}\}} \sum_{c\in\mathcal{C}_i} \mathcal{U}(\nu_c^{(i)}, \mathbf{z}_{i,\pi_c}; \{\eta_m\}) \quad \text{s.t.} \ \ \mathbf{z}_{i,\pi_c} \in Z_{\pi_c}, \ \ \sum_{c\in\pi_{i,j}^{-1}} \nu_{c,j}^{(i)} = 0 \ \forall j$$

where $\mathcal{U}(\nu_c^{(i)}, \mathbf{z}_{i,\pi_c}; \{\eta_m\}) = \tilde{\phi}_{i,c}(\mathbf{z}_{i,\pi_c}) + h(\nu_c^{(i)}; \mathbf{z}_{i,\pi_c})$. We solve the inner maximization problem (i.e., $\hat{\mathbf{z}}_{i,\pi_c} = \arg\max_{\mathbf{z}_{i,\pi_c}} \mathcal{U}(\nu_c^{(i)}, \mathbf{z}_{i,\pi_c}; \{\eta_m\})$ for fixed $\nu_c^{(i)}$. Then, the sub-gradient of $\mathcal{U}(\cdot)$ with respect to $\nu_{c,j}^{(i)}$ is $(\hat{\mathbf{z}}_{i,\pi_c})_j$. (Note that $h(\nu_c^{(i)}; \mathbf{z}_{i,\pi_c})$ is linear in $\nu_c^{(i)}$.) Using the prediction, we make the update: $\nu_{c,j}^{(i)}(t) \leftarrow \nu_{c,j}^{(i)}(t-1) - \gamma_t \Delta\nu_{c,j}^{(i)}$ where t denotes the t-th step and γ_t is the learning rate. Assuming that we start with $\nu^{(i)}$ satisfying the equality constraint, the constraint will be satisfied if $\sum_{c\in\bar{\mathcal{C}}_j} \Delta\nu_{c,j}^{(i)} = 0$ where $\bar{\mathcal{C}}_j = \{c : c \in \pi_{i,j}^{-1}\}$. This can be ensured by setting $\Delta\nu_{c,j}^{(i)} = (\hat{\mathbf{z}}_{i,\pi_c})_j - \frac{1}{|\bar{\mathcal{C}}_j|} \sum_{c\in\bar{\mathcal{C}}_j} (\hat{\mathbf{z}}_{i,\pi_c})_j$ (i.e., removing the *mean* from each component's optimal assignment). This update for ν is indeed the Euclidean projection on the feasible set. By assumption, each component c is tractable; therefore, the optimal assignment $\hat{\mathbf{z}}_{i,\pi_c}$ can be easily found. For example, in simple cases involving only a single node or a pair nodes in the graph, the optimal assignment can be found by enumeration. For more complex component such as trees, the *max-product* algorithm [27] can be used to find the optimal assignment.

Optimizing $\{\eta_m\}$ with fixed $\{\nu^{(i)}\}$. Consider the sub-problem of optimizing $\{\eta_m\}$ given by:

$$\min_{\{\eta_m\}} \max_{\{\mathbf{z}_{i,\pi_c}\}} \sum_i \sum_{c\in\mathcal{C}_i} \mathcal{U}(\{\eta_m\}, \mathbf{z}_{i,\pi_c}; \nu_c^{(i)}) \quad \text{s.t.} \ \ \mathbf{z}_{i,\pi_c} \in Z_{\pi_c} \quad \eta_m \geq 0 \ \ \forall m.$$

We solve this problem using the projected sub-gradient method; the parameters $\eta_m^{(t)}$ are updated as: $\eta_m^{(t)} \leftarrow [\eta_m^{(t-1)} - \tilde{\gamma}_t \Delta\eta_m]^+$ where $^+$ indicates projection on the non-negative orthant, and $\Delta\eta_m = \sum_{i,c\in\mathcal{C}_i} (\gamma_{m,i,c}\mu_{m,i,c}(\hat{\mathbf{z}}_{i,\pi_c}) - c_m)$. For fixed $\eta_m^{(t)}$, the

optimal assignments $\hat{\mathbf{z}}_{i,\pi_c}, \forall c, i$ can be found as earlier. Once the optimal assignments are obtained, we follow [16, Section IV.B] to obtain the final primal solution.

5 Multi-Label Classification Example

The method proposed in Sections 3-4 is applicable to any general structured prediction problem. To demonstrate the usefulness of this method, we show how our method can be applied to the multi-label classification problem and conduct related experiments. Here we use one formulation, given in [6]. We consider a pair-wise fully connected graph (to take care of all label correlations). Then, the scoring function $s(\mathbf{x}, \mathbf{y}; \theta)$ can be written as:

$$s(\mathbf{x}, \mathbf{y}; \theta) = \sum\nolimits_p s_p(\mathbf{z}_p; \theta_p(\mathbf{z}_p)) + \sum\nolimits_{p,q \neq p} s_{pq}((\mathbf{z}_{pq}); \theta_{pq}(\mathbf{z}_{pq})) \tag{8}$$

where the indices p and q run over the nodes (classes), $s_p(\cdot)$, $s_{pq}(\cdot)$ denote the node and edge scores computed using the class and label dependent model parameters $\theta_p(\mathbf{z}_p)$ and $\theta_{pq}(\mathbf{z}_{pq})$; $\mathbf{y} = \{-1, 1\}^K$ is a K-dimensional vector, where K is the number of classes. With binary label assignment for each class, $\mathbf{z}_p \in \mathcal{Z}_p$ is a 2-dimensional vector, and $\mathcal{Z}_p = \{(1,0), (0,1)\}$. $\mathbf{z}_{pq} \in \mathcal{Z}_{pq}$ is a 4-dimensional binary vector with only one *unit* element. There exists a mapping between $\mathbf{y}$ and $\mathbf{z}$ as described in Section 4.1: $\mathbf{z}_p = \text{bip}^{-1}(y_p)$ and $\mathbf{z}_{pq} = \text{bip}^{-1}(y_p, y_q)$. For linear models, the scores are computed as: $s_p(\mathbf{z}_p; \theta_p(\mathbf{z}_p)) = \theta_p(\mathbf{z}_p) \cdot \mathbf{x}$ and $s_{pq}(\mathbf{z}_{pq}; \theta_{pq}(\mathbf{z}_{pq})) = \theta_{pq}(\mathbf{z}_{pq}) \cdot \mathbf{x}$, where $\mathbf{x}$ is the feature vector. This setting has been used to study approximate learning or inference in the supervised learning setting [6, 26, 39]. However, we are not aware of any existing work studying this formulation in the semi-supervised setting.

Composite likelihood. Given the score having the form given in (8), composite likelihood can be defined in many different ways (e.g., [39]). In this paper, we define a composite likelihood function composed of K *spanning* tree models where each tree has one class at its *root* and the remaining classes as leaf nodes. Then, we can write the score for each tree as:

$$s_k(\mathbf{x}, \mathbf{y}; \bar{\theta}_k) = \frac{1}{K} \sum\nolimits_p \theta_p(\mathbf{z}_p) \cdot \mathbf{x} + \frac{1}{2} \sum\nolimits_{q \neq k} \theta_{pq}(\mathbf{z}_{pq}) \cdot \mathbf{x}$$

and the likelihood function as $p_\theta(\mathbf{Y}) = \prod_k p_k(\mathbf{Y}; \bar{\theta}_k)$ where $\bar{\theta}_k = \{\theta_p, \theta_{p,q} : q \neq k, p = 1, \ldots, K\}$, $p_k(\mathbf{Y}; \bar{\theta}_k) = \frac{\exp(s_k(\mathbf{x}, \mathbf{y}; \bar{\theta}_k))}{\sum_{\mathbf{y}} \exp(s_k(\mathbf{x}, \mathbf{y}; \bar{\theta}_k))}$. Note that $s(\mathbf{x}, \mathbf{y}; \theta) = \sum_k s_k(\mathbf{x}, \mathbf{y}; \theta)$ (scaling factors ensure the equality), and potentials are shared across the models. Therefore, all the models are learned jointly. [1] Since each sub-problem is a tree, the partition function and its gradient can be easily computed[2]; also, inference can be efficiently done. [3] See Komodakis et al, [16] for a discussion on the choice of sub-problems used in the decomposition.

[1] For other complex structured outputs, if the potentials are not shared then components could be learned independently.

[2] Evaluating the partition function of each sub-problem can be done in $O(Kln)$, where K, l, n are the numbers of classes, features, instances, respectively. Therefore, compute the composite likelihood requires $O(K^2 ln)$.

[3] Our inference algorithm is an iterative process. If the number of iterations is fixed, labeling l instances cost $O(K^2 ln)$.

Table 1. Data statistics (l: total number of samples, l_{tri}: number of train samples, l_{tst}: number of test samples, n: number of features, and K: number of classes). The train set is further split into two parts: labeled train and unlabeled train (see text for details).

Data set	l	l_{tri}	l_{tst}	n	K
scene	2,407	1,684	723	294	6
yeast	2,417	1,691	726	103	14
emotions	593	415	178	72	6
rcv	6,000	4,200	1,800	47,236	30
tmc2007	28,596	20,018	8,578	30,438	22

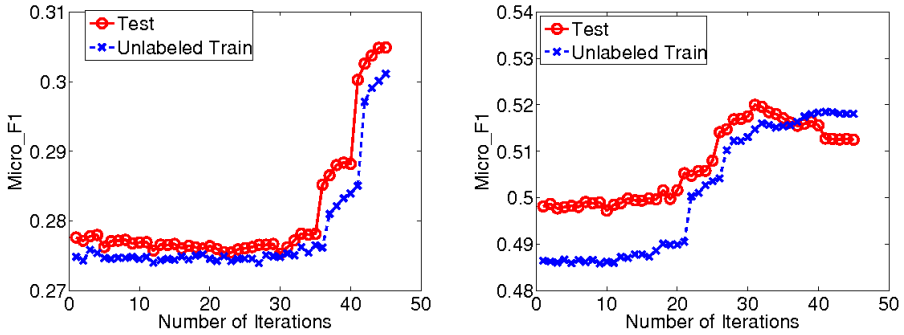
Constraints. Following the discussions in Section 3.2, we consider two types of corpus-level constraints: label distribution constraint (LDC) and label correlation constraint (LCC). LDC constrains the number of times a given label occurs throughout the entire data set and LCC constrains the number of times one label co-occurs with another throughout the data set. In the context of multi-label classification, LDC can be written in the following forms $\sum_{\mathbf{y}\in\mathbf{Y}} \delta(\text{bip}^{-1}(y_p), \mathbf{z}'_p) = n(\mathbf{z}'_p), \forall p = 1, ...K, \mathbf{z}'_p \in \mathcal{Z}_p$, where δ is the Kronecker delta function. LCC is $\sum_{\mathbf{y}\in\mathbf{Y}} \delta(\text{bip}^{-1}(y_p, y_q), \mathbf{z}'_{pq}) = n(\mathbf{z}'_{pq}), \forall p, q \neq p, \mathbf{z}'_{pq} \in \mathcal{Z}_{pq}$. $n(\mathbf{z}'_p)$ and $n(\mathbf{z}'_{pq})$ are given and estimated by counting the occurrence/co-occurrence of labels in the data. There are many other possible constraints that can be used in multi-label problems. For example, we can restrict the number of assigned labels for each sample. Local correlations [11] can be also incorporated in our model via instance level constraints. In the experiments we use LDC and LCC only. We will show the value of these constraints in Section 6.2.

6 Experiments

In this section, we do experiments on multi-label classification to understand the following: (a) how well the proposed semi-supervised learning framework improves over the supervised classifier; (b) the role of different constraints on the performance; (c) the impact of the inference and learning approximations on the performance; and (d) differences between transductive and semi-supervised solutions.

Datasets and setting. We considered five multi-label datasets from various applications[4]. Table 2 lists the data statistics. For rcv, we only used the first set of data. Because several labels in rcv have only a few positive examples, we removed such labels and only considered the 30 most frequent labels. For each data set, we constructed 10 random train and test splits of 70% and 30%. Then, we performed experiments on different degrees of labeling (d) by further splitting the train set into two parts: labeled train data ($d\%$) and unlabeled train data ($(100-d)\%$). In the training phase, only the labels of labeled train data are used. Because there are only few labels in the training set when the degree of labeling is low, we fixed the regularization parameter to a default value ($\sigma^2 = 1$) instead of tuning it. We checked in our experiments that the conclusions are still valid when σ is tuned using a validation set. We evaluate performance in terms of the Micro-F1 score [35] and conduct Wilcoxon sign-rank test with the significance level

[4] Data is available at `http://mulan.sourceforge.net`.



(a) rcv (b) emotions

Fig. 1. Performance on test set (red) and unlabeled train set (blue) along iterations for $d = 1\%$.

of 5%. All results in the tables are reported in terms of the mean and standard deviation over the 10 splits. Unless stated otherwise all results correspond to the case where both, label distribution and label correlation constraints are used. Test set inference is done collectively by treating it as a sample and applying collective inference with the constraints.

6.1 Performance of the Proposed Method

We begin by showing the performance (as a function of optimization iterations) of the proposed semi-supervised learning algorithm with approximate inference and learning. As we mentioned in Section 3.4, we increase the weight of unlabeled data C_m from 10^{-4} to 1 by a factor of 3 and for each C_m we conduct five inner iterations. Therefore, there are 45 total iterations for each trial. Figure 1 shows Micro-F1 scores evaluated on unlabeled data and test data along iterations when $d = 1\%$. We omit the corresponding plots for scene, yeast, and tmc2007 because they are somewhat similar to that on rcv. In most cases, the model achieves better performance as the number of iterations grows. This result is consistent with the observations made in the semi-supervised learning literature, e.g., [2, 5]. However, a key difference is that, our results are demonstrated on problems with complex structured outputs. The test set curve of emotions dips a bit at the end in Figure 1(b). This because, in the annealing step, we increase C_m along iterations. As the amount of unlabeled data in emotions is small, large C_m may overemphasize the unlabeled data, resulting in slightly inferior performance. When degree of labeling is 2% or higher, the performance on emotions does not drop in the end. In general, when the amount of unlabeled data is small, over-emphasizing it via large C_m leads to over-fitting.

Next, we show that our semi-supervised classifier (Semi-Sup) is better than the supervised classifier (Sup) that is trained only on the labeled data. We compare the results with various degrees of labeling $d \in \{1, 2, 4, 8\}$. For a fair comparison, when evaluating Sup on test set, the constraints are used. Table 2 shows the results. Overall, the semi-supervised learning algorithm outperforms the baseline classifier. The improvement is statistically significant. As d increases, the Micro-F1 score of the methods increases while the standard deviation decreases, as expected.

Table 2. Comparison of semi-supervised (Semi-Sup) and supervised (Sup) learning algorithms. Boldface indicates significant improvement of Semi-Sup over Sup.

Dataset	Degree of labeling (d)							
	1%		2%		4%		8%	
	Semi-Sup	Sup	Semi-Sup	Sup	Semi-Sup	Sup	Semi-Sup	Sup
scene	**55.2±3.7**	51.5±2.7	**59.3±2.1**	56.9±2.4	**63.8±1.4**	61.8±2.5	**66.9±1.4**	66.0±1.6
yeast	**42.9±2.1**	42.5±1.8	**43.4±1.0**	43.2±1.2	**45.2±1.2**	44.5±1.0	**45.2±0.9**	44.8±0.9
emotions	**51.3±5.9**	49.9±4.5	**55.5±4.3**	53.8±3.9	**58.8±3.8**	58.3±4.3	**62.2±2.5**	60.5±2.7
rcv	**30.5±2.1**	27.8±2.0	**33.6±1.8**	32.2±1.8	**36.4±0.9**	34.2±1.4	**36.3±1.3**	34.0±1.4
tmc2007	**42.0±1.2**	41.3±1.2	**43.0±1.1**	42.4±1.2	**44.1±0.6**	41.4±1.7	**43.8±0.7**	41.4±1.6

Table 3. Columns 1-4 compare various situations with constraints: without incorporating any constraint (No), using label distribution constraints (LDC), using label correlation constraints (LCC) and using both constraints (Both). Columns 4-7 investigate approximate/exact learning/inference. We use abbreviations to represent the combinations (e.g., ALEI stands for (A)pproximate (L)earning with (E)xact (I)nference). Results are in Micro-F1 (%). The best result (mean) of each column is boldfaced.

Data	d	ALAI			ALAI	ALEI	ELAI	ELEI
		No	LDC	LCC	Both	Both		
scene	1	51.6±2.4	54.6±3.5	53.5±3.6	55.2±3.7	**56.8±5.9**	53.7±5.4	56.6±5.5
	2	58.0±1.5	59.1±2.1	59.1±1.9	59.3±2.1	**62.6±2.0**	60.1±2.6	61.8±2.4
	4	61.2±3.2	63.5±1.6	63.2±1.8	63.8±1.4	**65.2±1.8**	64.2±1.6	64.7±1.9
yeast	1	42.3±2.7	42.8±2.1	42.8±2.3	42.9±2.1	42.1±2.0	**42.9±1.7**	42.2±1.8
	2	42.8±1.5	43.2±1.1	43.0±0.8	**43.4±1.0**	42.5±1.4	43.1±1.3	43.0±1.5
	4	45.0±1.8	**45.2±1.2**	**45.2±1.4**	45.1±1.2	44.5±1.3	44.5±1.6	44.0±1.4
emotions	1	48.3±6.2	51.0±5.9	51.1±5.7	51.2±5.9	**54.0±5.0**	51.7±6.9	52.9±5.6
	2	53.2±4.5	55.2±3.9	54.7±4.4	**55.5±4.3**	55.2±4.8	54.0±3.7	54.6±5.7
	4	58.0±3.9	**59.0±4.0**	**59.0±3.9**	58.8±3.8	58.4±3.1	57.0±3.2	58.1±3.1

In section 3.4, we mentioned the use of annealing to deal with the non-concavity of the objective function in (5). Experiments show the importance of annealing. For example, without using annealing, the mean performance of Semi-Sup on scene ($d = 1$) drops from 55.2% to 54.2%. In addition, incorporating constraints during testing improves the performance of the classifier. For example, without using constraints, the mean Micro-F1 scores of Semi-Sup and Sup on scene ($d = 1$) dropped to 54.9% and 49.6% from 55.2% and 51.5% respectively (see Table 2).

6.2 Using Different Sets of Constraints

The first four columns in Table 3 show the results for different uses of constraints. Without using constraints, the performance of Semi-Sup is suboptimal. With label distribution constraint (LDC) the performance is significantly better. In most cases, using label correlation constraints (LCC) obtains similar results as using LDC. However, when both constraints are used (Both), the performance gain is enhanced even further. These results show the importance of using constraints to improve the model.

6.3 Exact versus Approximation

Next, we investigate the performance difference between using an exact algorithm and an approximate algorithm during training and inference. We show the results on three small data sets, scene, yeast and emotions. The number of labels in these data sets is less than 15. For such cases, exact learning and inference algorithms are tractable. For exact inference, we explicitly enumerated all possible assignments of labels and chose the one with the highest objective function value in (6) as the solution. For exact learning, instead of computing composite likelihood, we compute the full likelihood. Computation of the partition function is the main bottleneck associated with exact learning. However, when the size of labels is small, the partition function can be computed exactly using scores for all enumerated label assignments. Columns 4-7 of in Table 3 show the results. In some cases, the approximate algorithm even achieves better performance than the exact one (e.g., ALAI achieves the best result on yeast). However, except on yeast ($d = 1\%, 4\%$), the difference between ELEI and the best result are not statistically significant. In general, the performance of approximate inference/learning is competitive with that of exact inference/learning. Regarding the running time, ALAI takes 4,167 seconds to train a model using yeast data set, while ELEI takes 9,175 seconds. Therefore, the approximation is faster than the exact method, especially when the number of labels is large. This shows the effectiveness of our semi-supervised learning framework with approximate inference and training.

6.4 Transductive Setting Experiments

If the test data is known in advance, it can be used in the semi-supervised learning process as additional unlabeled data. This has the potential to yield better performance on the test data. We refer this as the transductive setting, as opposed to the original inductive setting. Comparing the results of the transductive setting with those in Table 2 for the inductive setting, we found that the transductive setting is statistically significantly better on emotions data but achieves similar performance as the inductive setting on the other two data sets. For example, when degree of labeling is 1%, the mean performances of the transductive setting are 55.1%, 42.9% and 51.7% on scene, yeast, and emotions, respectively; compare this with 55.2%, 42.9% and 51.3% for the inductive setting. The key reason for this is that the original unlabeled train set of emotions is small, and therefore, including test data in training helps it do better.

Moreover, the transductive setting improves the supervised learning setting. For example, using the full train set as labeled data and the test set as the unlabled data to train the model improves the plain supervised learning setting from 74.3% to 74.8% in Micro-F1 on scene data set.

6.5 Incorporate the Inference Engine Proposed in Sec. 4 with Existing Methods

As we mentioned in Section 2, the proposed method is related to several methods in semi-supervised learning literature. However, most existing papers focus on problems with tractable structure outputs (e.g., linear chains), for which there is no need to use an approximate algorithm. Therefore, we are not aware of any existing method that we

Table 4. Comparison of semi-supervised (Semi-Sup) and supervised (Sup) learning algorithms of TSVM+ and CoDL+. Boldface indicates significant improvement of Semi-Sup over Sup or vice versa. We reproduce the results of our model from Table 2 for ease of reference.

Dataset	Degree of labeling (d)							
	1%		2%		4%		8%	
	Semi-Sup	Sup	Semi-Sup	Sup	Semi-Sup	Sup	Semi-Sup	Sup
TSVM+								
scene	**45.7±4.0**	42.1±2.6	**50.7±2.4**	45.5±3.2	**60.4±1.5**	55.9±1.9	**64.9±1.2**	63.0±1.2
yeast	40.1±2.1	**41.1±2.2**	40.3±1.3	**41.8±1.6**	41.2±1.5	**43.3±1.2**	40.4±1.7	**43.2±1.6**
emotions	46.8±6.4	50.0±5.7	51.9±4.3	51.5±4.2	54.2±3.5	55.3±5.1	56.7±2.9	57.6±1.8
CoDL+								
scene	**50.2±9.1**	33.8±6.8	**56.0±3.5**	37.8±3.8	**60.8±1.9**	45.4±4.7	**65.3±1.8**	55.4±3.8
yeast	40.9±3.3	39.6±3.0	39.8±2.4	**41.5±1.5**	40.9±1.7	**43.2±1.9**	40.5±1.3	**44.3±1.9**
emotions	**52.6±4.3**	43.0±8.2	**55.9±6.5**	44.8±6.3	**58.6±3.8**	52.5±4.7	**62.0±4.0**	55.8±4.2
The proposed method								
scene	**55.2±3.7**	51.5±2.7	**59.3±2.1**	56.9±2.4	**63.8±1.4**	61.8±2.5	**66.9±1.4**	66.0±1.6
yeast	**43.9±2.1**	42.5±1.8	**43.4±1.0**	43.2±1.2	**45.2±1.2**	44.5±1.0	**45.2±0.9**	44.8±0.9
emotions	**51.3±5.9**	49.9±4.5	**55.5±4.3**	53.8±3.9	**58.8±3.8**	58.3±4.3	**62.2±2.5**	60.5±2.7

can directly compare with. However, our inference method introduced in Section 4 can be applied to other semi-supervised models. In the following, we show two examples, where we combine our inference engine with CoDL [2] and transductive structured SVM [40, 6]. We refer the combined methods as CoDL+ and TSVM+, respectively.

TSVM+. TSVM [40] extends a binary transductive SVM model [12] to deal with structured outputs. However, it cannot deal with complex structured outputs, because it relies on an exact inference solver. In addition, the model does not use constraints from domain knowledge to guide learning. To extend TSVM, we modify the learning algorithm with an approximate structured SVM approach [6]. The augmented inference problems involved during the learning are solved approximately using our inference algorithm. Then, we add prior constraints over $\mathbf{y}$ in [40, Eq. OP3] to incorporate with corpus-level constraints. The resulted optimization problem is solved by a CCCP procedure [37]. We implemented TSVM+ based upon the Matlab version of structured SVM [33, 13].

CoDL+. CoDL [2] proposed a general framework to incorporate declarative constraints for structured learning. However, they did not show results on problems with complex structured output. We implement CoDL using an Averaged Structured Perceptron algorithm and our inference engine. Specifically, Steps 4-7 in [2, Algorithm 2] and Step 4 in [2, Algorithm 3] are replaced by our inference algorithm. We use an L+I setting with $\rho_k = \infty$. CoDL uses a smoothing parameter to combine the models learned from labeled and unlabeled data instead of using annealing steps. For a better comparison, we implement the same annealing steps introduced in 3.4 for TSVM+ and CODL+.

Comparison and Discussion. Table 4 shows the performance in micro-F1. Results show that TSVM+ significantly improves the plain supervised setting on scene, but achieves a sub-optimal solution on yeast. We suspect that the performance dip is due to over-fitting (a similar situation is shown in Figure 1(b)). In fact, TSVM+ achieves

better generative performance at early iterations on yeast. For example, it achieves 42.4% F1 at iteration 17 when $C_m = 0.005$ and the performance goes down afterwards. Regarding CODL+, it significantly improves its supervised counterpart on both scene and emotions. However, it also suffers from over-fitting on yeast data set.

The results in this section are mainly to demonstrate that our inference method can be incorporated with other semi-supervised models. We noted that extending existing semi-supervised learner for complex structured output problems is nontrivial. Therefore, a careful study might further improve the performance of CoDL+ or TSVM+. Nevertheless, results in Tables 4 show that our method achieves better or competitive performance with CODL+ and TSVM+ in all cases.

7 Conclusion and Discussion

In summary, we presented an effective semi-supervised learning framework for structured prediction problems with complex output structure. The proposed framework is general and can be easily applied to problems other than multi-label classification. Evaluating the framework on more complex problems is an important direction. A detailed and thoughtful comparison with other state-of-the-art semi-supervised multi-label classification methods is an interesting topic for future study.

References

1. Brefeld, U., Scheffer, T.: Semi-supervised learning for structured output variables. In: ICML. (2006)
2. Chang, M.W., Ratinov, L.A., Roth, D.: Structured learning with constrained conditional models. Machine Learning **88**(3) (2012) 399–431
3. Chang, Y.W., Collins, M.: Exact decoding of phrase-based translation models through lagrangian relaxation. In: EMNLP. (2011)
4. Chen, G., Song, Y., Wang, F., Zhang, C.: Semi-supervised multi-label learning by solving a sylvester equation. In: SDM. (2008) 410–419
5. Dhillon, P.S., Keerthi, S.S., Bellare, K., Chapelle, O., Sellamanickam, S.: Deterministic annealing for semi-supervised structured output learning. In: AISTATS. (2012)
6. Finley, T., Joachims, T.: Training structural SVMs when exact inference is intractable. In: ICML. (2008) 304–311
7. Ganchev, K., Graca, J., Gillenwater, J., Taskar, B.: Posterior regularization for structured latent variable models. JMLR **11** (2010)
8. Guo, Y., Schuurmans, D.: Semi-supervised multi-label classification - a simultaneous large-margin, subspace learning approach. In: ECML/PKDD. (2012) 355–370
9. Hazan, T., Shashua, A.: Norm-product belief propagation: Primal-dual message-passing for approximate inference. CoRR (2009)
10. Hazan, T., Urtasun, R.: Efficient learning of structured predictors in general graphical models. CoRR (2012)
11. Huang, S.J., Zhou, Z.H., Zhou, Z.H.: Multi-label learning by exploiting label correlations locally. In: AAAI. (2012)
12. Joachims, T.: Transductive inference for text classification using support vector machines. In: ICML, Morgan Kaufmann (1999) 200–209
13. Joachims, T., Finley, T., Yu, C.N.J.: Cutting-plane training of structural SVMs. Machine Learning **77**(1) (2009) 27–59

14. Jojic, V., Gould, S., Koller, D.: Accelerated dual decomposition for MAP inference. In: ICML. (2010)
15. Komodakis, N.: Efficient training for pairwise or higher order crfs via dual decomposition. In: CVPR. (2011)
16. Komodakis, N., Paragios, N., Tziritas, G.: MRF energy minimization and beyond via dual decomposition. PAMI **33**(3) (2011) 531–552
17. Koo, T., Rush, A.M., Collins, M., Jaakkola, T., Sontag, D.: Dual decomposition for parsing with non-projective head automata. In: EMNLP. (2010)
18. Kulesza, A., Pereira, F.: Structured learning with approximate inference. In: NIPS. (2008)
19. Lee, C.H., Jiao, F., Wang, S., Schuurmans, D., Greiner, R.: Learning to model spatial dependency: Semi-supervised discriminative random fields. In: NIPS. (2006)
20. Lindsay, B.G.: Composite likelihood methods. Contemporary Mathematics **80** (1988) 221–239
21. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Math. Program. **45**(3) (1989) 503–528
22. Liu, Y., Jin, R., Yang, L.: Semi-supervised multi-label learning by constrained non-negative matrix factorization. In: AAAI. (2006) 421–426
23. Mann, G.S., McCallum, A.: Generalized expectation criteria for semi-supervised learning with weakly labeled data. JMLR **11** (2010) 955–984
24. Martins, A.F.T., Figueiredo, M.A.T., Aguiar, P.M.Q., Smith, N.A., Xing, E.P.: Alternating directions dual decomposition. CoRR (2012)
25. Meshi, O., Globerson, A.: An alternating direction method for dual MAP LP relaxation. In: ECML/PKDD. (2011)
26. Meshi, O., Sontag, D., Jaakkola, T., Globerson, A.: Learning efficiently with approximate inference via dual losses. In: ICML. (2010)
27. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. (1988)
28. Pletscher, P., Wulff, S.: LPQP for MAP: Putting LP solvers to better use. In: ICML. (2012)
29. Samdani, R., Chang, M., Roth, D.: Unified expectation maximization. In: NAACL. (2012)
30. Samdani, R., Roth, D.: Efficient decomposed learning for structured prediction. In: ICML. (2012)
31. Seah, C.W., Tsang, I.W., Ong, Y.S.: Transductive ordinal regression. In: TNNLS. (2012) 1074–1086
32. Sutton, C., McCallum, A.: Piecewise training for structured prediction. Machine Learning **77**(2–3) (2009) 165–194
33. Vedaldi, A.: A MATLAB wrapper of $\mathrm{SVM}^{\mathrm{struct}}$. `http://www.vlfeat.org/~vedaldi/code/svm-struct-matlab.html` (2011)
34. Xu, L., Wilkinson, D., Schuurmans, D.: Discriminative unsupervised learning of structured predictors. In: ICML. (2006)
35. Yang, Y.: An evaluation of statistical approaches to text categorization. Information Retrieval **1** (1999) 69–90
36. Yu, C.N.: Transductive learning of structural SVMs via prior knowledge constraints. In: AISTATS. (2012)
37. Yuille, A.L., Rangarajan, A.: The concave-convex procedure. Neural Computation (2003)
38. Zha, Z.J., Mei, T., Wang, J., Wang, Z., Hua, X.S.: Graph-based semi-supervised learning with multiple labels. J. Visual Communication and Image Representation **20**(2) (2009) 97–103
39. Zhang, Y., Schneider, J.: A composite likelihood view for multi-label classification. In: AISTATS. (2012)
40. Zien, A., Brefeld, U., Scheffer, T.: Transductive support vector machines for structured variables. In: ICML. (2007)