Deterministic Annealing for Semi-Supervised Structured Output Learning

Paramveer S. Dhillon
Computer & Information Science
University of Pennsylvania
Philadelphia, PA
dhillon@cis.upenn.edu

S. Sathiya Keerthi[1], **Kedar Bellare**[1], **Olivier Chapelle**[1], **S. Sundararajan**[2]
Yahoo! Labs
[1] Santa Clara, CA, [2] Bangalore, India
{selvarak|kedar|chap|ssrajan@yahoo-inc.com}

Abstract

In this paper we propose a new approach for semi-supervised structured output learning. Our approach uses relaxed labeling on unlabeled data to deal with the combinatorial nature of the label space and further uses domain constraints to guide the learning. Since the overall objective is non-convex, we alternate between the optimization of the model parameters and the label distribution of unlabeled data. The alternating optimization coupled with deterministic annealing helps us achieve better local optima and as a result our approach leads to better constraint satisfaction during inference. Experimental results on sequence labeling benchmarks show superior performance of our approach compared to Constraint Driven Learning (CoDL) and Posterior Regularization (PR).

1 Introduction

Consider a structured output problem in which a data instance consists of an input vector $\mathbf{x}$ and a label vector $\mathbf{y}$. 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

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} s(\mathbf{x}, \mathbf{y}, \theta). \tag{1}$$

Appearing in Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume 11 of JMLR: W&CP 11. Copyright 2012 by the authors.

In sequence labeling $f(\mathbf{x}, \mathbf{y})$ is a vector of factors associated with a linear chain structure on the $\mathbf{y}$ sequence. 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 $p(\mathbf{y}|\mathbf{x}, \theta)$ assuming independence of the $\mathbf{y}$'s. For ease of notation we will simply refer to these quantities as $p_\theta(\mathbf{Y})$ and $p_\theta(\mathbf{y})$.

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

$$\min_\theta S(\theta) = R(\theta) + \mathcal{L}(\mathbf{Y}_L; \mathbf{X}_L, \theta), \tag{2}$$

where $R(\theta) = \|\theta\|^2/2\sigma^2$ is a regularizer and

$$\mathcal{L}(\mathbf{Y}_L; \mathbf{X}_L, \theta) = \frac{1}{n_L} \sum_{\mathbf{x}_L} \mathcal{L}_{\mathbf{x}_L}(\mathbf{y}_L; \mathbf{x}_L, \theta) \tag{3}$$

is the loss term and n_L is the number of labeled instances. $\mathcal{L}_{\mathbf{x}_L}$ is the instance level loss; in the probabilistic model

$$\mathcal{L}_{\mathbf{x}_L}(\mathbf{y}_L; \mathbf{x}_L, \theta) = -\log p_\theta(\mathbf{y}_L). \tag{4}$$

We are interested in semi-supervised learning where labeled data is limited and we have access to a (large) set of unlabeled data instances $\mathbf{X}$; let $\mathbf{Y}$ denote the corresponding unknown set of label vectors. Recently several effective methods have been proposed for this problem (Mann & McCallum, 2010; Gärtner et al., 2005; Ganchev et al., 2010; Bellare et al., 2009; Liang et al., 2009; Chang et al., 2007). Constraints that come from domain knowledge play a key role in these methods, and consist of hard or soft restrictions on the labels $\mathbf{Y}$, possibly in a way that is dependent on $\mathbf{X}$. See (Chang et al., 2007; Ganchev et al., 2010) for many examples of constraints in sequence labeling and other structured output tasks. These constraints are expressed through a vector of constraint functions $\phi(\mathbf{X},\mathbf{Y})$. For example, domain knowledge might say that the constraints $\phi(\mathbf{X},\mathbf{Y}) - \mathbf{c} \leq 0$ hold approximately. This information can be used in semi-supervised learning by defining a constraint loss function $\mathcal{C}(\phi(\mathbf{X},\mathbf{Y}) - \mathbf{c})$ which is set up to penalize violations of $\phi(\mathbf{X},\mathbf{Y}) - \mathbf{c} \leq 0$. Enforcing constraints exactly corresponds to putting infinite penalty on violations. Different methods use different forms of $\mathcal{C}$; see (Ganchev et al., 2010) for a detailed description of several ways of doing constraint modeling.

Related Semi-Supervised Methods.

To motivate our approach and put it in the right perspective let us briefly review the key semi-supervised methods. *Constraint driven learning (CoDL)* (Chang et al., 2007) was one of the first methods that interweaved constrained inference and perceptron style learning steps, and was based on hard labeling of unlabeled data.

Since then, several methods have been proposed which are more general than CoDL. *Generalized Expectation (GE)* (Mann & McCallum, 2010) is a method in which learning is done by including the expected constraint loss term with respect to p_θ in the training objective:

$$\min_{\theta} S(\theta) + \mathcal{C}(\mathbb{E}_{p_\theta}[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{c}). \tag{5}$$

GE training is expensive because the gradient of the constraint loss term involves the computation of the covariance between model features f and constraint features ϕ with respect to p_θ. The *Posterior Regularization (PR)* method (Ganchev et al., 2010; Liang et al., 2009; Bellare et al., 2009; Gärtner et al., 2005) was proposed to overcome this difficulty. This is done by defining an auxiliary distribution $q(\mathbf{Y})$ and replacing the constraint loss term by two terms, one for expressing closeness of p_θ with q and the other for minimizing the expected constraint violation with respect to q:

$$\min_{\theta,q} S(\theta) + KL(q||p_\theta) + \mathcal{C}(\mathbb{E}_q[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{c}). \tag{6}$$

where KL denotes KL divergence. This problem is solved using alternating optimization of θ and q. Each of the two subproblems is simple to solve; also, the computation of correlations between feature functions and constraint functions is avoided.

The *Entropy Regularization method (ER)* (Grandvalet & Bengio, 2003) includes an entropy term on p_θ:

$$\min_{\theta} S(\theta) + ER(\theta), \quad ER(\theta) = -\sum_{\mathbf{Y}} p_\theta(\mathbf{Y}) \log p_\theta(\mathbf{Y}). \tag{7}$$

Although the original entropy regularization method (Grandvalet & Bengio, 2003) does not use domain constraints, these constraints are crucial for getting good performance. Mann and McCallum (2010) try the inclusion of the constraint loss term:

$$\min_{\theta} S(\theta) + ER(\theta) + \mathcal{C}(\mathbb{E}_{p_\theta}[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{c}). \tag{8}$$

This can also be viewed as GE with the entropy term added. A brief set of experiments on multi-class problems by (Mann & McCallum, 2010) shows that, while (7) gives very poor performance, (8) gives a performance better than GE. Thus entropy regularization is useful in the presence of constraints.

A general issue with the probabilistic methods discussed above is that constraints are enforced only in an expected sense. Thus it is possible that constraints are violated badly when inference (1) is done. Note that such a violation can happen even on the training examples, $\mathbf{X}$.

Large margin structured output models have also been suggested for semi-supervised learning. (Zien et al., 2007) extend the TSVM model of binary classification (Joachims, 1999) to structured outputs by solving

$$\min_{\theta,\mathbf{Y}} S(\theta) + \mathcal{L}(\mathbf{Y};\mathbf{X},\theta) + \mathcal{C}(\phi(\mathbf{X},\mathbf{Y}) - \mathbf{c}), \tag{9}$$

where, like (3),

$$\mathcal{L}(\mathbf{Y};\mathbf{X},\theta) = \frac{1}{n}\sum_{\mathbf{x}} \mathcal{L}_{\mathbf{x}}(\mathbf{y};\mathbf{x},\theta) \tag{10}$$

is the loss term corresponding to unlabeled data and n is the number of unlabeled instances.

Unlike binary classification the solution of (9) is hard due to each $\mathbf{y}$ in $\mathbf{Y} = \{\mathbf{y}\}$ having combinatorial possibilities. To deal with the combinatorial issue, (Zien et al., 2007) use the fact that, in large margin models, the number of active $\mathbf{y}$'s is usually small. They keep track of a small set of possible candidates for each $\mathbf{y}$, use smoothing of the large margin loss function and employ gradient based methods to optimize θ. In their experiments on sequence labeling they do not include constraints in any significant way; possibly as a result of this, their experiments do not show their semi-supervised method to be very useful.

In this paper we propose a new approach for semi-supervised structured output learning that is close in spirit to (9) and has the following properties:

- it ensures that constraints are closely satisfied during inference;
- it applies to general loss functions;
- it deals with the combinatorial issue in $\mathbf{Y}$ in a computationally tractable way while avoiding poor local minima;

- and, it keeps the attractive property of avoiding computation of correlations between model and constraint features.

The paper is organized as follows. In section 2 we describe the main idea behind our approach. Section 3 describes the use of deterministic annealing as applied to general loss functions. In section 4 we specialize the ideas for probabilistic models. Experiments comparing our method with other methods are given in section 5. Section 6 concludes the paper.

2 Our Approach

Our model training is a relaxed version of (9) in which, instead of a single $\mathbf{Y}$ we work with a distribution $\{a(\mathbf{Y})\}$ and the loss function can be general:

$$\min_{\theta,a} S(\theta) + \mathbb{E}_a\mathcal{L}(\mathbf{Y};\mathbf{X},\theta) + \mathcal{C}(\mathbb{E}_a[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{c}). \quad (11)$$

The relaxation from $\mathbf{Y}$ to $\{a(\mathbf{Y})\}$ is for computational tractability reasons, similar to LP-relaxation based inference of complex graphical models (Rush et al., 2010). Whereas dealing with $\mathbf{Y}$ brings combinatorial issues, dealing with $\{a(\mathbf{Y})\}$ leads to simplified computations. For example, when deterministic annealing is used to solve (11) (in sections 3 and 4), $\{a(\mathbf{Y})\}$ takes a standard exponential form that is easy to deal with; see, for example, the expressions for $\{a(\mathbf{Y})\}$ derived later, in (20) and (27).

Even though $\{a(\mathbf{Y})\}$ is a distribution, it plays a role quite different from p_θ and q that are used to handle constraints in GE, PR and ER methods. To appreciate this, consider an example scenario in which there are only hard constraints: $\phi(\mathbf{X},\mathbf{Y}) - \mathbf{c} \leq 0$ (this corresponds to defining: $\mathcal{C}(z) = 0$ if $z \leq 0$ and ∞ otherwise) and the model is probabilistic, i.e., the instance-level loss is $\mathcal{L}_\mathbf{x}(\mathbf{y};\mathbf{x},\theta) = -\log p_\theta(\mathbf{y}) = -\theta \cdot f(\mathbf{x},\mathbf{y}) + \log Z(\mathbf{x})$. Now consider the problem of optimizing a keeping θ fixed. For the example scenario this problem reduces to:

$$\min_a \frac{1}{n}\mathbb{E}_a \sum_\mathbf{x} \theta \cdot f(\mathbf{x},\mathbf{y}) \quad s.t.\ \mathbb{E}_a[\phi(\mathbf{X},\mathbf{Y})] - \mathbf{c}] \leq 0. \quad (12)$$

This is a linear programming problem in $\{a(\mathbf{Y})\}$. (Note that $\mathbb{E}_a[\cdot]$ is linear in a and the distribution $\{a(\mathbf{Y})\}$ needs to satisfy distribution polytope constraints.) This problem is nothing but LP-relaxation based inference (Rush et al., 2010) on the unlabeled data[1]; the resulting $\{a(\mathbf{Y})\}$ tends to be sparse (only a few $a(\mathbf{Y})$'s are non-zero); most often $a(\mathbf{Y}) = 1$ for a single $\mathbf{Y}$. Thus the training objective is in tune with constraint satisfaction at the inference stage.

[1]Constraints cause the combined inference of all unlabeled instances. If each constraint involves only one instance then (12) decouples into separate constrained inference problems, one for each instance.

The terms $\mathbb{E}_a\mathcal{L}(\mathbf{Y};\mathbf{X},\theta)$ and $\mathcal{L}(\mathbf{Y};\mathbf{X},\theta)$ in (11) and (9) are the losses on unlabeled data corresponding to the labeling given by $\{a(\mathbf{Y})\}$ and $\mathbf{Y}$ respectively. They play an important role since, without them unlabeled data has no effect on θ. In classification problems these terms represent the geometric intuition that the classifier boundary passes through regions where data is sparse. Like in entropy regularization, the unlabeled loss terms play effective role in the presence of good domain constraints.

The entropy term in (7) and (8) can also be viewed as the expected loss (negative log-likelihood) of labels chosen according to p_θ. Note that (8) is an instance of (11) with the additional constraint $a = p_\theta$. Thus, for the probabilistic model, the $\mathbb{E}_a\mathcal{L}(\mathbf{Y};\mathbf{X},\theta)$ term in (11) can be viewed as relaxed entropy regularization. Leaving out the $a = p_\theta$ constraint allows for computational simplicity. As we will see in section 5, the generalization performance is excellent in spite of this relaxation.

The solution of (11) will be addressed in the following sections.

3 Deterministic Annealing Solution

The objective function in (11) is non-convex, making it susceptible to local minima issues. We employ deterministic annealing (DA) (Peterson & Soderberg, 1989; Sindhwani et al., 2006), which is a well established tool for finding better local minima of non-convex objective functions. DA proceeds by adding the negative entropy of the distribution $\{a(\mathbf{Y})\}$ and controlling this term using a temperature parameter T:

$$\min_{\theta,a} S(\theta) + \mathbb{E}_a\mathcal{L}(\mathbf{Y};\mathbf{X},\theta) + \mathcal{C}(\mathbb{E}_a[\phi(\mathbf{X},\mathbf{Y}) - \mathbf{c}])$$
$$+\frac{T}{n}\sum_\mathbf{Y} a(\mathbf{Y})\log a(\mathbf{Y}) \quad \text{s.t.} \sum_\mathbf{Y} a(\mathbf{Y}) = 1. \quad (13)$$

We have omitted the non-negativity constraints on the $a(\mathbf{Y})$'s since the presence of the negative entropy term ensures that these variables naturally come out positive in the optimization. The denominator coefficients n_L in (3) and n in (10) and the negative entropy term in (13) are simple normalization factors that make sure various parts of the objective get equal weightage.

DA solves the problem by slowly decreasing (annealing) the temperature T from a reasonably big value towards zero; at each T the optimization problem (13) is solved using the (θ, a) obtained from the previous T as the starting point. When T is large, the negative entropy term plays a dominant role, allowing the variables to move around easily; this corresponds to the exploration phase of the method. As T moves across a critical temperature the variables settle in values that are in the region of attraction of a good local minimum. As T is further decreased towards zero, the effect of the negative entropy term diminishes and (θ, a) moves towards a good local minimum of (11); this is the exploitation phase of the method. In practice, T is changed according to the annealing schedule, $T_{next} = \alpha T$ where $0 < \alpha < 1$, say $\alpha = 2/3$.

DA has effectively been used for solving binary TSVM in (Sindhwani et al., 2006). The application of DA to structured output training such as (11) is more complex because $\{a(\mathbf{Y})\}$ has a very large number of variables and so $\{a(\mathbf{Y})\}$ cannot be handled in a direct fashion. Our approach to handling this issue will become clear below.

At each T we use alternating optimization steps on θ and $\{a(\mathbf{Y})\}$ to solve (13). This is attractive since both subproblems turn out to have convex programming structure and, more importantly, the computation of the covariance between model and constraint features is avoided.

Consider the subproblem corresponding to fixing $\{a(\mathbf{Y})\}$ and optimizing θ (with terms not dependent on θ omitted):

$$\min_{\theta} \mathcal{P}_1(\theta; a) = S(\theta) + \mathbb{E}_a \mathcal{L}(\mathbf{Y}; \mathbf{X}, \theta). \tag{14}$$

This is a convex programming problem if the loss $\mathcal{L}$ is convex in θ. Depending on the type of loss, a suitable optimization method can be used to solve (14). Later, in section 4 we will look at the details for the probabilistic model using the loss in (4).

Next, consider the subproblem corresponding to fixing θ and optimizing $\{a(\mathbf{Y})\}$ (with terms not dependent on $\{a(\mathbf{Y})\}$ omitted):

$$\min_{a} \mathbb{E}_a \mathcal{L}(\mathbf{Y}; \mathbf{X}, \theta) + \mathcal{C}(\mathbb{E}_a[\phi(\mathbf{X}, \mathbf{Y}) - \mathbf{c}]) + \frac{T}{n} \sum_{\mathbf{Y}} a(\mathbf{Y}) \log a(\mathbf{Y}) \quad \text{s.t.} \quad \sum_{\mathbf{Y}} a(\mathbf{Y}) = 1. \tag{15}$$

To proceed further we need to specify the model used for $\mathcal{C}$. As a running example let us consider the case where constraints correspond to the domain knowledge that $\phi(\mathbf{X}, \mathbf{Y}) - \mathbf{c} \leq 0$ holds approximately.[2] This can be handled by defining a slack vector ξ, using the constraints $\phi(\mathbf{X}, \mathbf{Y}) - \mathbf{c} \leq \xi$ and setting the constraint loss to be proportional to $\|\xi\|^2$. The same constraint model will be used in all the experiments reported in section 5. Using this model in our relaxed solution approach, we get the following specific version of (15):

$$\min_{a,\xi} \mathcal{P}_2(a, \xi; \theta) = \mathbb{E}_a \mathcal{L}(\mathbf{Y}; \mathbf{X}, \theta) + \frac{T}{n} \sum_{\mathbf{Y}} a(\mathbf{Y}) \log a(\mathbf{Y}) + \frac{1}{2\epsilon} \|\xi\|^2$$
$$\text{s.t. } \mathbb{E}_a[\phi(\mathbf{X}, \mathbf{Y}) - \mathbf{c}] \leq \xi; \ \sum_{\mathbf{Y}} a(\mathbf{Y}) = 1. \tag{16}$$

Note that hard inequality constraints can be handled by taking ϵ towards zero.

We also make the assumption that the constraint function decomposes over the instances, $\{\mathbf{x}\}$ as sum, i.e., there exists a set of functions $\phi(\mathbf{x}, \mathbf{y})$ such that

$$\phi(\mathbf{X}, \mathbf{Y}) = \frac{1}{n} \sum_{\mathbf{x}} \phi(\mathbf{x}, \mathbf{y}). \tag{17}$$

For efficient computation it is also necessary that each $\phi(\mathbf{x}, \mathbf{y})$ has a structure similar to the model feature vector $f(\mathbf{x}, \mathbf{y})$. For example, in sequence labeling, we require that $\phi(\mathbf{x}, \mathbf{y})$ is a vector of factors associated with a linear chain structure on the $\mathbf{y}$ sequence.

In practice, the above assumption on the structure of constraint function is not restrictive. Let us look at some constraint functions. Domain constraints are of two types: instance level and corpus level. An instance level constraint is some restriction placed on each individual instance $(\mathbf{y}, \mathbf{x})$, and so it easily falls into the form (17); all that we need to do is set $\phi(\mathbf{x}, \mathbf{y}) = 0$ for all but one sequence. Examples of this type of constraint in sequence labeling tasks are: in a sequence, a label (e.g., *Paper Title* of a citation) must occur at most once as a contiguous list of words; in postal address extraction, the label *Street Number* immediately precedes the label *Street Name* with high probability. Typically, in the case of an instance level constraint, we will write one separate constraint for each instance. A corpus level constraint involves labels across instances. In sequence labeling, a good example is a soft/hard specification of the number of contiguous occurrences of a given label averaged over all sequences. Note that, if we define $\phi(\mathbf{x}, \mathbf{y})$ as $k(\mathbf{x}, \mathbf{y})/n$ where $k(\mathbf{x}, \mathbf{y})$ is the number of contiguous occurrences of the given label in sequence $\mathbf{y}$ and n is the total number of sequences, then this constraint is expressible via a constraint function of the form (17).

As $\{a(\mathbf{Y})\}$ consists of a very large number of variables we turn to the dual of (16) to get a tractable solution. This is similar to the dual ideas employed in other methods such as PR. The Lagrangian associated with

[2] Equality constraints can be easily handled in a way very similar to inequality constraints. Details are given further below.

(16) is given by

$$\mathbb{L}(a, \xi, \delta, \gamma) = \mathbb{E}_a \mathcal{L}(\mathbf{Y}; \mathbf{X}, \theta) + \frac{T}{n} \sum_{\mathbf{Y}} a(\mathbf{Y}) \log a(\mathbf{Y}) + \frac{1}{2\epsilon}\|\xi\|^2 + \delta \cdot (\mathbb{E}_a[\phi(\mathbf{X}, \mathbf{Y}) - \mathbf{c}] - \xi) + \gamma(\sum_{\mathbf{Y}} a(\mathbf{Y}) - 1),$$

where the vector $\delta \geq 0$ and γ are Lagrange multipliers. The dual problem is:

$$\max_{\delta \geq 0, \gamma} \min_{a, \xi} \mathbb{L}(a, \xi, \delta, \gamma). \qquad (18)$$

The nice structure in $\mathbb{L}$ (in conjunction with (17)) allows us to solve the inner minimization problem in closed form and eliminate $\{a(\mathbf{Y})\}$ and ξ as well as γ by expressing them in terms of δ:

$$a(\mathbf{Y}) = \prod_{\mathbf{y}} a(\mathbf{y}), \qquad (19)$$

$$a(\mathbf{y}) = \frac{\exp((-\mathcal{L}_{\mathbf{x}}(\mathbf{y}; \mathbf{x}, \theta) - \delta \cdot \phi(\mathbf{x}, \mathbf{y}))/T)}{Z_\delta(\mathbf{x})}, \qquad (20)$$

$$\xi = \epsilon\delta, \quad \gamma = \frac{T}{n}(\log \sum_{\mathbf{x}} Z_\delta(\mathbf{x}) - 1), \qquad (21)$$

where

$$Z_\delta(\mathbf{x}) = \sum_{\mathbf{y}} \exp((-\mathcal{L}_{\mathbf{x}}(\mathbf{y}; \mathbf{x}, \theta) - \delta \cdot \phi(\mathbf{x}, \mathbf{y}))/T). \quad (22)$$

These expressions can be used to simplify (18) as

$$\max_{\delta \geq 0} \mathcal{D}(\delta) = -\frac{T}{n} \sum_{\mathbf{x}} \log Z_\delta(\mathbf{x}) - \mathbf{c} \cdot \delta - \frac{\epsilon}{2} \cdot \|\delta\|^2. \quad (23)$$

If, in (16), one of the constraints, say the k-th, is an equality instead of an inequality, then, in (23) and (18) δ_k will be unconstrained instead of being non-negative. The appendix gives the details behind the derivation of (20)-(23). $\mathcal{D}$ is differentiable; its gradient is given by

$$\frac{\partial \mathcal{D}}{\partial \delta} = -\mathbf{c} + \frac{1}{n} \sum_{\mathbf{x}} \mathbb{E}_a[\phi(\mathbf{x}, \mathbf{y})] - \epsilon \cdot \delta. \qquad (24)$$

Any good gradient based method for minimizing a differentiable function with non-negativity constraints, such as LBFGS-B (Zhu et al., 1997) can be used to solve (23). However care is needed in terminating the solution. For the alternating optimization steps on θ and (a, ξ) to work well it is important to ensure that the primal $\mathcal{P}_2(a, \xi; \theta)$ decreases. We are solving the dual to achieve this. If (23) is solved till optimality then it is assured that the (a, ξ) resulting from the optimal δ via (20) achieves a decrease in $\mathcal{P}_2(a, \xi; \theta)$. However, during the solution of (23) the value of $\mathcal{P}_2(a, \xi; \theta)$ tends to oscillate. Thus, when terminating the solution of (23) approximately, it is important to check that a decrease in $\mathcal{P}_2(a, \xi; \theta)$ has been achieved; if not, more optimization steps need to be taken.

Note from (20) that $a(\mathbf{y}) > 0$ for all $\mathbf{y}$. However, it should be clear that, as $T \to 0$, only those $\mathbf{y}$ that yield the maximum value for $(-\mathcal{L}_{\mathbf{x}}(\mathbf{y}; \mathbf{x}, \theta) - \delta \cdot \phi(\mathbf{x}, \mathbf{y}))$ will have positive $a(\mathbf{y})$ values. This is consistent with the comment on sparsity of a that we made in section 2.

There is still one incomplete element in the solution of (23) outlined above. Computation of $Z_\delta(\mathbf{x})$ involves $\mathcal{L}_{\mathbf{x}}$; if $\mathcal{L}_{\mathbf{x}}(\mathbf{y}; \mathbf{x}, \theta)$ does not decompose over edge and node factors this computation can become unwieldy. It turns out that, for the probabilistic model and the corresponding loss in (4) this decomposition happens cleanly; we take up the details of this in the next section. For other losses such as the large margin loss, this requires more intricate handling. For example, in the case of large margin loss we need to maintain a small set of active candidates for each $\mathbf{y}$ in order to deal with $\mathcal{L}_{\mathbf{x}}$ in the $Z_\delta(\mathbf{x})$ computation.

We will take this up in future work.

4 Specialization to probabilistic models

The loss in (4) can be expanded as:

$$\mathcal{L}_{\mathbf{x}}(\mathbf{y}; \mathbf{x}, \theta) = -\theta \cdot f(\mathbf{x}, \mathbf{y}) + \log Z(\mathbf{x}), \qquad (25)$$

where $Z(\mathbf{x})$ is an instance specific normalization constant. This allows $Z_\delta(\mathbf{x})$ in (22) to be rewritten as $Z_\delta(\mathbf{x}) = Z(\mathbf{x})^{-\frac{1}{T}} \tilde{Z}_\delta(\mathbf{x})$, where

$$\tilde{Z}_\delta(\mathbf{x}) = \sum_{\mathbf{y}} \exp \frac{(\theta \cdot f(\mathbf{x}, \mathbf{y}) - \delta \cdot \phi(\mathbf{x}, \mathbf{y}))}{T}. \qquad (26)$$

The simplified expression for $a(\mathbf{y})$ follows:

$$a(\mathbf{y}) = \frac{1}{\tilde{Z}_\delta(\mathbf{x})} \exp\left(\frac{\theta \cdot f(\mathbf{x}, \mathbf{y}) - \delta \cdot \phi(\mathbf{x}, \mathbf{y})}{T}\right). \qquad (27)$$

Now the dual objective in (23) simplifies to:

$$\mathcal{D}(\delta) = -\frac{T}{n} \sum_{\mathbf{x}} \log \tilde{Z}_\delta(\mathbf{x}) - \mathbf{c} \cdot \delta - \frac{\epsilon}{2} \cdot \|\delta\|^2 + z, \quad (28)$$

where $z = \sum_{\mathbf{x}} \log Z(\mathbf{x})$ is a constant that does not depend on δ. The gradient in (24) remains unchanged. Because of the simplified a in (27) standard CRF-style computations can be used to compute $\mathcal{D}(\delta)$ and its gradient.

Also, with (25), the θ-subproblem in (14) can be solved using a gradient based technique such as L-BFGS. The problem is only slightly more complicated than standard CRF training: the unlabeled term involves a label distribution instead of a single label. Given the simplified expression for $a(\mathbf{y})$ in (27), this becomes a minor issue as the ingredients in CRF computations for function and gradient can be easily tweaked to handle this. Specifically, computation of expectations with respect to exponential distributions of the form (27) can be easily done via standard forward-backward operations.

From (27) we can get

$$\arg\max_{\mathbf{y}} a(\mathbf{y}) = \arg\max_{\mathbf{y}} (\theta \cdot f(\mathbf{x}, \mathbf{y}) - \delta \cdot \phi(\mathbf{x}, \mathbf{y})) . \tag{29}$$

We can extend this as a pseudo-inductive inference rule for application to unseen data:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} (\theta \cdot f(\mathbf{x}, \mathbf{y}) - \delta \cdot \phi(\mathbf{x}, \mathbf{y})) . \tag{30}$$

An advantage of this rule over the inductive inference rule in (1) is that it also makes use of the constraint functions through δ.

It is interesting to note that, if, in the deterministic annealing solution for probabilistic models outlined in this section, we set $T = 1$ and do not do any annealing, then the solution is identical to posterior regularization. This is a mere coincidence and does not mean any systematic connection between the two approaches. As $T \to 0$ the deterministic annealing solution becomes quite different from posterior regularization.

5 Experiments

In this section we evaluate the empirical performance of our approach on two standard real world sequence labeling datasets, *citations* and *apartment listings* (we will refer to this as the *apartments* dataset), and demonstrate the effectiveness of our approach. These two datasets, which originated in (Grenager et al., 2005), have been used to benchmark the performance of sequence labeling methods in the context of information extraction. We compare the labeling accuracy obtained from our solution with state of the art approaches.

5.1 Data Description

The apartments dataset contains 300 hand-labeled ads (sequences) from `craigslist.org`. Each token of a sequence takes one of 12 possible labels (e.g. FEATURES, RENT, SIZE, NEIGHBORHOOD, UTILITIES). The average sequence length is 119 tokens with 8.7 labels.

The citations data consists of 500 hand annotated citations of computer science papers. Each token of a citation takes one of 13 possible labels (e.g. AUTHOR, TITLE, JOURNAL, PAGES). The average sequence length in this case is 35 tokens with 5.5 labels. The description and sizes of splits of the datasets used are given in Table 1. In order to ensure fairness of comparison, the train, development, test, unlabeled splits and also the tokenization are exactly the same as was used by all the previous works which used these datasets, namely (Grenager et al., 2005; Chang et al., 2007; Mann & McCallum, 2010; Bellare et al., 2009).

5.2 Types of Constraints

The constraints that we used[3] for the two datasets can be broadly classified into two types: corpus level and instance level; the former places constraints on statistics accumulated over the instances while the latter puts constraints at each instance level. In the training objective function a corpus level constraint leads to a single term that is the square of a sum of terms while an instance level constraint leads to the sum of squares of several terms. Some constraints can be implemented either as corpus level constraints or instance level constraints. Corpus level constraints are computationally attractive since the number of ϕ elements (and so the number of δ variables) does not grow with the size of the unlabeled data set. However, instance level constraints allow us to specify background knowledge at a more fine-grained level. For example, if we know that all citation sequences *may not* contain any VOLUME label (i.e. it is an optional label), we can inject this domain knowledge into our learning procedure easily as an instance level constraint. Let us now describe the two types of constraints in more detail using examples.

Corpus Level Constraints: These constrain the accumulated statistics across the entire unlabeled corpus. In our experiments, we use three types of corpus level constraints:

1. **Token Level Constraints:** These are the simplest kind of constraints and constrain certain tokens to have a particular label (or a set of labels), e.g. *"The words CA, NY, Australia are* LOCATION*"*. They are implemented as corpus level constraints; for the example mentioned above, we set the constraint function as *the fraction of occurences of CA, NY and Australia that take the label* LOCATION and set the corresponding component of $\mathbf{c}$ to 1. Token level constraints can also be easily handled at the time of inference by constraining the Viterbi solution in (1); however this procedure would not help with generalization since it would not effect the model parameters θ. We set the $\mathbf{c}$ value of such constraints to be 1.

[3]The competing methods also employed exactly the same constraints.

Dataset	# Train	# Dev.	# Unlab.	# Test.
Citations	5, 20, 300	100	1000	100
Apartments	5, 20,100	100	1000	100

Table 1: Data Statistics

2. **Label Regularization Constraints:** These constraints are similar in spirit to the label balance constraints that are employed in TSVMs (Joachims, 1999). They constrain the fraction of total number of tokens which can have a particular label, e.g. *"30% of tokens should be labeled* AUTHORS*"*. The values of these constraints (30% in the example above) are estimated from the training data.

3. **Edge Level Constraints:** Such constraints consider the labels of two consecutive tokens and constrain them. An example is: *"Label transitions should only occur on punctuation marks"*. We implement this constraint as: *the fraction of label transitions that happen at non-punctuation characters is 0.01.*

In our experiments, all corpus constraints described above are implemented as equality constraints (satisfied in a least squares sense as described in section 4).

Instance Level Constraints: These constrain aggregate statistics only at the level of a single instance, e.g. *"*AUTHOR *can only appear at most once in each citation sequence and should be a contiguous list of tokens"*. We implement this constraint as an inequality constraint with its value set to be less than or equal to 1. This is because we do not expect all instances to have all the labels present; so, an equality constraint saying *"*NOTE *can occur only once"* in a citation which doesn't have the label NOTE at all is problematic.

The detailed list of constraints we used are given in Table 2.

Citations Dataset
Each label must be a consecutive list of words, and can appear at most once in a citation.
Label transitions must occur on punctuation marks.
Words *pp., pages* are PAGE.
Four digits starting numbers 20XX, 19XX are DATE.
Quotations can only appear in TITLE.
The words *note, submitted, appear* are NOTE.
The words *CA, Australia, NY* are LOCATION.
The words *tech, tech_report* are TECH_REPORT.
The words *proc, journal, proceedings, ACM* are JOURNAL or BOOKTITLE.
The words *ed, editors* are EDITOR.

Apartments Dataset
Label transitions must occur on punctuation marks or newline symbol.
Words *laundry, kitchen, parking* are FEATURES.
Words *sq, ft, bdrm* are SIZE.
The words $, *money* are RENT.
The words *close, near, shopping* are NEIGHBORHOOD.
The (normalized) words *phone, email* are CONTACT.
The words *smoking, dogs, cats* are RESTRICTIONS.
The words *http, image, link* are PHOTOS.
The words *address, carlmont, st, cross* are ADDRESS.
The words *utilities, pays, electricity* are UTILITIES.

Table 2: List of constraints used for the Citations and Apartments datasets.

5.3 Experimental Setup and Results

Our model $p_\theta(\mathbf{y}|\mathbf{x})$ is a simple linear chain CRF with feature functions of the form $f(y_{t-1}, y_t, \mathbf{x})$. We used a very simple set of model features: (1) Identity of the token; (2) Start, End features indicating the start and end of a sequence; (3) Regular expression features (*Capitalization, hyphenation, number, suffixes, prefixes*) in a window of 1 around the current token; and, (4) Brown Clustering (100 clusters) (Brown et al., 1992) features for the tokens.

As mentioned in (3) and (10), we normalize the labeled and unlabeled losses by the total number of labeled and unlabeled instances, to ensure that the unlabeled data term does not exert undue influence in the objective function. The values of the hyperparameters i.e. σ (associated with the ℓ_2 norm regularizer on θ) and ϵ (weight of constraint term) were tuned on the development set. We used different ϵ's for different types of constraints as each of them encodes a different kind of prior information.

At each temperature we performed 10 alternating optimizations of θ and $\{a(\mathbf{Y})\}$ and, within each alternating step the θ and $\{a(\mathbf{Y})\}$ optimizations were done for 40 and 100 LBFGS iterations respectively. In our experiments we found that this combination was good enough to achieve good convergence of the overall objective function. The annealing temperature T was started at 2 and decreased slowly ($T_{next} = 0.9T$) for 10 iterations.

We compare our approach against the following approaches[4]: **Supervised Baseline**: A CRF trained on just the labeled examples; **Constraint Driven Learning (CoDL)** (Chang et al., 2007); and, **Posterior Regularization (PR)** (Ganchev et al., 2010;

[4]The hyperparameters of all these approaches were tuned in the same way as in our approach.

Citations Dataset	N	Sup. Baseline	CoDL	PR (I)	PR (T)	DASO (I)	DASO (T)
	5	0.631	0.710	0.730	0.741	**0.752**	**0.773**
	20	0.791	0.794	0.827	0.844	**0.849**	**0.860**
	300	0.899	0.888	0.901	0.906	**0.911**	**0.921**
Apartments Dataset	N	Sup. Baseline	CoDL	PR (I)	PR (T)	DASO (I)	DASO (T)
	5	0.651	0.660	0.665	0.671	**0.679**	**0.685**
	20	0.727	0.746	0.749	0.761	**0.762**	**0.769**
	100	0.764	0.786	0.790	0.799	**0.800**	**0.804**

Table 3: Comparison of methods. N denotes the number of labeled examples. **Note:** (1) DASO is our approach i.e. Deterministic Annealing for Structured Outputs. (2) (I), (T) denote whether it is the inductive performance or transductive accuracy. (3) Bold numbers indicate that the results are significant at 5% level in a paired t-test. Significance was tested among corresponding numbers only i.e. inductive against inductive and transductive against transductive.

Bellare et al., 2009).

A fair comparison against GE (Mann & McCallum, 2010) was not feasible since handling edge level and instance level constraints requires approximate inference for computing the covariance matrix of constraint and model features. This is still a subject of ongoing research.[5] Additionally, GE currently does not allow inequality constraints that we use in our experiments. Hence, the comparison against this approach is left as future work.

For PR and our approach, test set labels were found either transductively (**T**) (including the test set as a part of unlabeled data so that constraints are applied on them, and then using (30)[6]) or inductively (**I**) (not using the test data as part of the unlabeled data and then applying (1)). The accuracy is measured as token labeling accuracy on test data. The results averaged over 5 random subsets of the data are shown in Table 3.

As can be seen from the results, our approach performs significantly better than the other methods in a variety of settings. The gain over competing approaches is particularly good when the number of labeled examples is small.

The relative improvements in constraint satisfication of our approach over PR (averaged over all the constraints) on the Citations test data are 4.4%, 3.5% and 3.1% respectively for $N = 5, 20, 300$, where N is the number of labeled examples. Here, constraint violation is computed by summing $(\phi_k(\mathbf{X}, \mathbf{Y}_{viterbi}) - \mathbf{c}_k)^2$ and $\max(0, (\phi_k(\mathbf{X}, \mathbf{Y}_{viterbi}) - \mathbf{c}_k))^2$ for equality and inequality constraints respectively; here $\mathbf{Y}_{viterbi}$ is the output obtained via (1). As we can see our approach results in better constraint satisfaction (closer to the Viterbi solution) on test data.

5.3.1 Effect of Annealing Temperature

The annealing schedule ($T_{i+1} = \alpha T_i$) is an important parameter. It has been shown that for binary classification any reasonable schedule performs well (Sindhwani et al., 2006). We experimented with a variety of different schedules i.e., starting temperatures and annealing rates (α). Particularly, we experimented with starting temperatures of 1, 2 and 5, with α of 0.9, 0.7 and 0.5. In addition to this, we also tested the hypothesis, *Do we need annealing at all?*, by evaluating the performance of the method that directly starts at the stop temperature and does no further annealing. The results are shown in Table 4.

Start T	α	Stop T	DASO (UA)	DASO (A)
1	0.9	0.3486	0.721	**0.771**
1	0.7	0.0282	0.729	**0.769**
1	0.5	0.0009	0.707	**0.761**
2	0.9	0.6973	0.742	**0.773**
2	0.7	0.0564	0.738	**0.770**
2	0.5	0.0019	0.728	**0.762**
5	0.5	0.0012	0.725	**0.755**

Table 4: Effect of annealing on Citations dataset (5 labeled examples averaged over 5 subsets). **Note:** (1) These experiments were performed in the transductive setting. (2) *DASO (A)* is the performance with annealing and *DASO (UA)* is the performance if we start at the stop temperature and do not do any annealing. (3) The bold numbers indicate that they are significantly better than the un-annealed version in a paired t-test at 5% significance level.

As can be seen annealing helps in all the settings when compared to starting at the end temperature and not doing any annealing. Annealing is reasonably robust to the starting temperature and annealing rate; however, if the annealing is rapid ($\alpha = 0.5$), accuracy suffers due to the optimization following a bad contour and reaching poor optima. This phenomenon was not observed for binary classification (Sindhwani et al., 2006); in sequence labeling, this possibly happens due to the sequential nature of the data and errors getting propagated within a sequence.

Our method is slower (5-10 times) than CoDL/PR as it needs to solve the optimization problem at several T values while they solve it only once. However, doing so allows us to get better local optima compared to other approaches. As future work, it would be interesting to investigate if annealing can even help other methods i.e. CoDL/ PR/GE.

[5]Personal communication with Gregory Druck.

[6]Recall that setting $T = 1$ in our approach gives PR.

6 Appendix (Supplementary Material)

Here we derive (20)-(23). Consider the solution of (18). Setting $\partial\, \mathbb{L}/\partial\, a(\mathbf{Y}) = 0$ we get

$$\mathcal{L}(\mathbf{Y};\mathbf{X},\theta) + \frac{T}{n} + \frac{T}{n}\log a(\mathbf{Y}) + \delta\cdot\phi(\mathbf{X},\mathbf{Y}) + \gamma\; = 0.$$

This leads to

$$T\log a(\mathbf{Y}) = -(n\mathcal{L}(\mathbf{Y};\mathbf{X},\theta) + T + n\delta\cdot\phi(\mathbf{X},\mathbf{Y}) + n\gamma),$$

which gives

$$\sum_{\mathbf{Y}} a(\mathbf{Y}) = Z_\delta(\mathbf{X})\exp((-T-n\gamma)/T), \qquad (31)$$

where

$$\begin{aligned} Z_\delta(\mathbf{X}) &= \sum_{\mathbf{Y}} \exp((-n\mathcal{L}(\mathbf{Y};\mathbf{X},\theta) - n\delta\cdot\phi(\mathbf{X},\mathbf{Y}))/T) \\ &= \prod_{\mathbf{x}} Z_\delta(\mathbf{x}) \end{aligned}$$

and

$$Z_\delta(\mathbf{x}) = \sum_{\mathbf{y}} \exp((-\mathcal{L}_{\mathbf{x}}(\mathbf{y};\mathbf{x},\theta) - \delta\cdot\phi_{\mathbf{x}}(\mathbf{x},\mathbf{y}))/T).$$

Setting $\partial\ \mathbb{L}/\partial\ \xi = 0$ we get $\xi = \epsilon\cdot\delta$. Use of the derivations above allows us to simplify (18) to

$$\max_{\delta\geq 0,\gamma} -\frac{T}{n}\sum_{\mathbf{Y}} a(\mathbf{Y}) - \mathbf{c}\cdot\delta - \gamma - \frac{\epsilon}{2}\cdot\|\delta\|^2.$$

Using (31) this further simplifies to

$$\max_{\delta\geq 0,\gamma} -\frac{T}{n}Z_\delta(\mathbf{X})\exp((-T-n\gamma)/T) - \mathbf{c}\cdot\delta - \gamma - \frac{\epsilon}{2}\cdot\|\delta\|^2. \qquad (32)$$

Optimizing γ first with fixed δ yields $\gamma = \frac{T}{n}(\log Z_\delta(\mathbf{X}) - 1)$. Putting this in (32) reduces the lagrangian dual problem (18) to (23). The expressions in (20)-(22) follow easily from the above derivations.

References

Bellare, K., Druck, G., & McCallum, A. (2009). Alternating projections for learning with expectation constraints. *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* (pp. 43–50). Arlington, Virginia, United States: AUAI Press.

Brown, P., deSouza, P., Mercer, R., Pietra, V. D., & Lai, J. (1992). Class-based n-gram models of natural language. *Comput. Linguist.*, *18*, 467–479.

Chang, M.-W., Ratinov, L.-A., & Roth, D. (2007). Guiding semi-supervision with constraint-driven learning. *ACL*.

Ganchev, K., Graca, J., Gillenwater, J., & Taskar, B. (2010). Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, *11*.

Gärtner, T., Le, Q. V., Burton, S., Smola, A. J., & Vishwanathan, S. V. N. (2005). Large-scale multiclass transduction. *NIPS*.

Grandvalet, Y., & Bengio, Y. (2003). Semi-supervised learning by entropy minimization. *NIPS*.

Grenager, T., Klein, D., & Manning, C. D. (2005). Unsupervised learning of field segmentation models for information extraction. *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics* (pp. 371–378). Ann Arbor, Michigan: Association for Computational Linguistics.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. *ICML* (pp. 200–209).

Liang, P., Jordan, M. I., & Klein, D. (2009). Learning from measurements in exponential families. *Proceedings of the 26th Annual International Conference on Machine Learning* (pp. 641–648). New York, NY, USA: ACM.

Mann, G. S., & McCallum, A. (2010). Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of Machine Learning Research*, *11*, 955–984.

Peterson, C., & Soderberg, B. (1989). A new method for mapping optimization problems onto neural networks. *International Journal of Neural Systems*, *1*, 3–22.

Rush, A. M., Sontag, D., Collins, M., & Jaakkola, T. (2010). On dual decomposition and linear programming relaxations for natural language processing. *In Proc. EMNLP*.

Sindhwani, V., Keerthi, S. S., & Chapelle, O. (2006). Deterministic annealing for semi-supervised kernel machines. *ICML* (pp. 841–848).

Zhu, C., Byrd, R. H., & Nocedal, J. (1997). Algorithm 778: L-BFGS-B, fortran routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, *23*, 550–560.

Zien, A., Brefeld, U., & Scheffer, T. (2007). Transductive support vector machines for structured variables. *ICML*.