Learning Efficiently with Approximate Inference via Dual Losses

Ofer Meshi MESHI@CS.HUJI.AC.IL
David Sontag DSONTAG@CSAIL.MIT.EDU
Tommi Jaakkola TOMMI@CSAIL.MIT.EDU
Amir Globerson GAMIR@CS.HUJI.AC.IL

Abstract

Many structured prediction tasks involve complex models where inference is computationally intractable, but where it can be well approximated using a linear programming relaxation. Previous approaches for learning for structured prediction (e.g., cutting-plane, subgradient methods, perceptron) repeatedly make predictions for some of the data points. These approaches are computationally demanding because each prediction involves solving a linear program to optimality. We present a scalable algorithm for learning for structured prediction. The main idea is to instead solve the dual of the structured prediction loss. We formulate the learning task as a convex minimization over both the weights and the dual variables corresponding to each data point. As a result, we can begin to optimize the weights even before completely solving any of the individual prediction problems. We show how the dual variables can be efficiently optimized using coordinate descent. Our algorithm is competitive with state-of-the-art methods such as stochastic subgradient and cutting-plane.

1. Introduction

In many prediction problems we are interested in predicting multiple labels $y_1, \ldots, y_d$ from an input $\boldsymbol{x}$ rather than a single label as in multiclass prediction. This setting is referred to as structured prediction, and has found many applications in various domains, from natural language processing to computational biology (Bakir et al., 2007). A naive approach to the problem is to predict each label y_i individually, ignoring possible correlations between the labels. A better approach would be to explicitly model the interactions between the labels, which then results in the labels being jointly predicted. Structured prediction models do this by using classifiers of the form $\boldsymbol{y} = \arg\max_{\hat{\boldsymbol{y}}} \boldsymbol{w} \cdot f(\boldsymbol{x}, \hat{\boldsymbol{y}})$, where $f(\boldsymbol{x}, \boldsymbol{y})$ is a given function and $\boldsymbol{w}$ are weights to be learned from data.

Much of the early work on structured prediction (Lafferty et al., 2001; Taskar et al., 2004) focused on the case where prediction (i.e., maximization over $\boldsymbol{y}$) could be done using efficient combinatorial algorithms such as dynamic programming or maximum-weight matching. However, this restricted the types of interactions that these models were capable of capturing to tractable structures such as tree graphs. Recent work on graphical models has shown that even when the maximization over $\boldsymbol{y}$ is not known a priori to be tractable, linear programming (LP) relaxations often succeed at finding the true maximum, even giving certificates of optimality (Sontag et al., 2008). This strongly motivates learning structured prediction models which use LP relaxations for prediction, and indeed several recent works show that this yields empirically effective results (Finley and Joachims, 2008; Martins et al., 2009).

Learning with large scale data necessitates efficient algorithms for finding the optimal weight vector $\boldsymbol{w}$. Although several such algorithms have been proposed for structured prediction, these have primarily focused on settings where the maximization over $\boldsymbol{y}$ is performed using combinatorial optimization. Some examples are structured perceptron (Collins, 2002), stochastic subgradient (Ratliff et al., 2007), extra-gradient (Taskar et al., 2006), and cutting-plane algorithms (Joachims et al., 2009). All of these approaches require making a prediction at every iteration. When LP relaxations are used, this corresponds to repeatedly solving an LP to optimality, significantly reducing the scalability of the overall learning algorithm.

These earlier approaches have two potential sources of inefficiency. First, it is likely not necessary to solve the LPs to optimality to obtain an approximately cor-

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

rect update for the weights, particularly in the early iterations of the algorithms. Second, these approaches typically re-solve the LPs from scratch at each iteration. However, particularly in later iterations when there are only small changes to the weight vector, we would like to be able to "warm start" using the previous iteration's solution to the LP for a data point.

In this paper we introduce a novel method for learning structured prediction models using LP relaxations. Whereas previous learning approaches involved repeatedly solving the computationally intensive LP problem per data point, our new formulation replaces the standard LP with its dual. This turns the entire problem into a minimization over the weights $\boldsymbol{w}$ and auxiliary dual variables δ. The latter can be updated via a simple closed form message passing scheme that decreases the overall objective at each iteration. We combine these with stochastic subgradient updates on $\boldsymbol{w}$ and thus our scheme has an online flavor similar to Shalev-Shwartz et al. (2007).

We show empirically that avoiding the LP solution indeed results in much improved convergence time when compared to previous methods. This effect becomes more pronounced the larger the label space is, and the method is thus expected to enhance performance on many large scale structured prediction problems.

2. Problem Formulation

We begin by reviewing the maximum margin Markov network formulation (M^3N) (Taskar et al., 2004), and its LP relaxation. We consider a labelled dataset $\{\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)}\}_{i=1}^{n}$ containing n samples. We seek a function $h(\boldsymbol{x}; \boldsymbol{w})$ that will predict $\boldsymbol{y}$ from $\boldsymbol{x}$. It is assumed to be of the form

$$h(\boldsymbol{x}; \boldsymbol{w}) = \arg\max_{\boldsymbol{y}} \boldsymbol{w} \cdot f(\boldsymbol{x}, \boldsymbol{y}) \tag{1}$$

where $f(\boldsymbol{x}, \boldsymbol{y})$, the feature vector, is given by a fixed, known, vector-valued function of both $\boldsymbol{x}$ and $\boldsymbol{y}$. In what follows we assume that $\boldsymbol{y}$ is multivariate and has d variables denoted by $y_1, \ldots, y_d$. Furthermore, f is assumed to decompose into pairwise and singleton factors on $\boldsymbol{y}$, so that:

$$\boldsymbol{w} \cdot f(\boldsymbol{x}, \boldsymbol{y}) = \sum_{ij \in E} f_{ij}(y_i, y_j, \boldsymbol{x}) \cdot \boldsymbol{w}_{ij} + \sum_i f_i(y_i, \boldsymbol{x}) \cdot \boldsymbol{w}_i \tag{2}$$

where E is a set of edges in a graph G, and the vectors $\boldsymbol{w}_{ij}, \boldsymbol{w}_i$ are the elements of the weight vector corresponding to each one of the factors (some weights may be shared across edges).[1]

[1] We use factors of size one and two for notational convenience only; our approach generalizes to larger factors.

In M^3N the weight vector is found by minimizing the following regularized hinge loss:

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{n} \sum_m \ell_h(\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)}; \boldsymbol{w}) \tag{3}$$

where:

$$\ell_h(\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)}; \boldsymbol{w}) = \max_{\boldsymbol{y}} \boldsymbol{w} \cdot \Delta f^{(m)}(\boldsymbol{y}) + e^{(m)}(\boldsymbol{y}) \tag{4}$$

Here $e^{(m)}(\boldsymbol{y})$ is the discrepancy between the true labelling $\boldsymbol{y}^{(m)}$ and $\boldsymbol{y}$, and is assumed to decompose as $e^{(m)}(\boldsymbol{y}) = \sum_i e_i^{(m)}(\boldsymbol{y})$.[2] We also define $\Delta f^{(m)}(\boldsymbol{y}) = f(\boldsymbol{x}^{(m)}, \boldsymbol{y}) - f(\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)})$.

The problems in Eq. 1 and Eq. 4 involve finding an assignment $\boldsymbol{y}$ that maximizes a function of the form $\sum_{ij} \theta_{ij}(y_i, y_j) + \sum_i \theta_i(y_i)$. This problem, commonly referred to as the MAP problem in the graphical models literature, is intractable (NP hard) for general graphs G, and tractable only in isolated cases such as tree structured graphs. However, linear programming (LP) relaxations are often effective as approximations, and have been incorporated into M^3N by several authors, which we review further in Section 4.

In MAP-LP relaxations one replaces maximization over $\boldsymbol{y}$ of $\sum_{ij} \theta_{ij}(y_i, y_j) + \sum_i \theta_i(y_i)$ with the following linear program:[3] $\max_{\boldsymbol{\mu} \in \mathcal{M}_L(G)} \boldsymbol{\mu} \cdot \boldsymbol{\theta}$, where $\mathcal{M}_L(G)$ enforces consistency between the pairwise distributions $\mu_{ij}(y_i, y_j)$:

$$\mathcal{M}_L(G) = \left\{ \boldsymbol{\mu} \geq 0 \;\middle|\; \begin{array}{l} \sum_{y_j} \mu_{ij}(y_i, y_j) = \mu_i(y_i) \\ \sum_{y_i} \mu_{ij}(y_i, y_j) = \mu_j(y_j) \\ \sum_{y_i} \mu_i(y_i) = 1 \end{array} \right\} . \tag{5}$$

To introduce the LP relaxation into M^3N, we use the notation $\boldsymbol{\theta}(f, e, \boldsymbol{w})$ to denote the vector of parameters, where the pairwise elements are $\theta_{ij}(y_i, y_j) = f_{ij}(y_i, y_j) \cdot \boldsymbol{w}_{ij}$ and the singleton elements are $\theta_i(y_i) = f_i(y_i) \cdot \boldsymbol{w}_i + e_i(y_i)$. We shall also use: $\boldsymbol{\theta}^{(m)}(\boldsymbol{w}) = \boldsymbol{\theta}(\Delta f^{(m)}, e^{(m)}, \boldsymbol{w})$. Finally, we will denote the singleton and pairwise elements of $\boldsymbol{\theta}^{(m)}(\boldsymbol{w})$ by $\boldsymbol{\theta}^{(m)}(y_i; \boldsymbol{w})$ and $\boldsymbol{\theta}^{(m)}(y_i, y_j; \boldsymbol{w})$ respectively.

We then have the following approximation of the loss ℓ_h from Eq. 4:

$$\hat{\ell}_h(\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)}; \boldsymbol{w}) = \max_{\boldsymbol{\mu} \in \mathcal{M}_L(G)} \boldsymbol{\mu} \cdot \boldsymbol{\theta}^{(m)}(\boldsymbol{w}) \tag{6}$$

[2] The loss could also be defined along the edges, but we omit this for notational convenience.

[3] We use the notation $\boldsymbol{\mu} \cdot \boldsymbol{\theta} = \sum_{ij \in E} \sum_{y_i, y_j} \mu_{ij}(y_i, y_j) \theta_{ij}(y_i, y_j) + \sum_i \sum_{y_i} \mu_i(y_i) \theta_i(y_i)$

It is straightforward to show that the relaxed loss $\hat{\ell}_h$ provides an upper bound on the true loss ℓ_h (Finley and Joachims, 2008).

The final classifier is obtained by solving the maximization $\arg\max_{\boldsymbol{\mu}\in\mathcal{M}_L(G)} \boldsymbol{\mu}\cdot\boldsymbol{\theta}(f,0,\boldsymbol{w})$ and returning $y_i = \arg\max_{\hat{y}_i} \mu_i(\hat{y}_i)$. To summarize the above, we are interested in solving the optimization problem:

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{n}\sum_m \max_{\boldsymbol{\mu}\in\mathcal{M}_L(G)} \boldsymbol{\mu}\cdot\boldsymbol{\theta}^{(m)}(\boldsymbol{w}) \quad (7)$$

3. Optimization via Dual Losses

In this section we present our algorithm for solving the optimization problem in Eq. 7. We begin by using convex duality to replace the internal maximization in Eq. 7 with minimization of a piecewise-linear objective. Numerous duals have been suggested for the MAP LP relaxation problem (e.g., Globerson and Jaakkola, 2008; Komodakis et al., 2007; Werner, 2007). We use the formulation discussed in Werner (2007). The dual of $\max_{\boldsymbol{\mu}\in\mathcal{M}_L(G)} \boldsymbol{\mu}\cdot\boldsymbol{\theta}$ is thus:

$$\begin{aligned}\min_\delta \quad & \sum_i \max_{y_i}\Big[\theta_i(y_i) + \textstyle\sum_{k\in N(i)} \delta_{ki}(y_i)\Big] + \\ & \sum_{ij} \max_{y_i,y_j}\Big[\theta_{ij}(y_i,y_j) - \delta_{ij}(y_j) - \delta_{ji}(y_i)\Big]\end{aligned} \quad (8)$$

Denote the objective of the above dual by $g(\delta;\boldsymbol{\theta})$. Then we have that Eq. 7 equals:

$$\min_{\boldsymbol{w},\delta^{(1)},\ldots,\delta^{(n)}} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{n}\sum_m g(\delta^{(m)};\boldsymbol{\theta}^{(m)}(\boldsymbol{w})), \quad (9)$$

where we now minimize over the dual variables $\delta^{(m)}$ in addition to the weights $\boldsymbol{w}$. Because the dual always upper bounds the primal, the function $g(\delta^{(m)};\boldsymbol{\theta}^{(m)}(\boldsymbol{w}))$ is a convex upper bound on the relaxed loss $\hat{\ell}_h(\boldsymbol{x}^{(m)},\boldsymbol{y}^{(m)};\boldsymbol{w})$ for every value of $\delta^{(m)}$. This bound can be tightened by minimizing it over $\delta^{(m)}$. By LP duality, the minimal $\delta^{(m)}$ value gives us exactly $\hat{\ell}_h(\boldsymbol{x}^{(m)},\boldsymbol{y}^{(m)};\boldsymbol{w})$.

The key advantage of Eq. 9 is that it removes the difficult inner maximization from Eq. 7. Moreover, Eq. 9 is jointly convex in its variables (namely $\boldsymbol{w}$ and the $\delta^{(m)}$s), and furthermore is unconstrained. Thus, we can employ a variety of minimization algorithms to it without the need for a "black-box" solver of the maximization problem.

In the next three sections we describe our algorithm for optimizing Eq. 9. Our approach has two components: one is to decrease the objective via message passing updates on δ, corresponding to coordinate descent on δ. The other is stochastic subgradient updates on $\boldsymbol{w}$ that process each example separately and are thus ideally suited for large data sets.

3.1. Dual minimization via coordinate descent

Notice that, unlike the $\boldsymbol{w}$ variables, the $\delta^{(m)}$ variables are only dependent on the m^{th} sample in the dual formulation Eq. 9. Block coordinate descent of $g(\delta^{(m)};\boldsymbol{\theta}^{(m)}(\boldsymbol{w}))$ can be performed in closed form, as has been noted by several authors (e.g., Globerson and Jaakkola, 2008; Werner, 2007), and as we review next.

Suppose that at iteration t we have a given value of the $\delta^{(m)}$ variables, denoted by $\delta^{(m,t)}$. Now assume we fix all $\delta^{(m)}$ variables except $\delta^{(m)}_{ij}, \delta^{(m)}_{ji}$ and seek the optimal value of $\delta^{(m)}_{ij}, \delta^{(m)}_{ji}$. The closed form solution is given by:

$$\begin{aligned}\delta^{(m,t+1)}_{ij}(y_j) = \quad & -\tfrac{1}{2}\boldsymbol{\theta}^{(m)}(y_j;\boldsymbol{w}) - \tfrac{1}{2}\textstyle\sum_{k\in N(j)\setminus i} \delta^{(m,t)}_{kj}(y_j) \\ & +\tfrac{1}{2}\max_{y_i}\Big[\boldsymbol{\theta}^{(m)}(y_i,y_j;\boldsymbol{w}) - \delta^{(m,t)}_{ji}(y_i)\Big],\end{aligned} \quad (10)$$

and analogously for $\delta^{(m,t+1)}_{ji}(y_i)$. This update is commonly referred to as max-sum diffusion, or MSD (see Werner, 2007, and references within).

We use a more efficient block coordinate descent step where we simultaneously update all of the dual variables $\delta_{ij}(y_j)$ going into variable y_j (see Globerson and Jaakkola, 2008, for a similar update). It is equivalent to iterating the MSD updates for the corresponding edges until convergence, and is given by (we drop the m superscript for brevity):

$$\delta_{ij}(y_j) = -\frac{1}{1+d_j}\boldsymbol{\theta}^{(m)}(y_j;\boldsymbol{w}) - \frac{1}{1+d_j}\gamma_j(y_j) + \gamma_{ij}(y_j) \quad (11)$$

where d_j is the degree of node j in the graph and:

$$\gamma_{ij}(y_j) = \max_{y_i}\Big[\boldsymbol{\theta}^{(m)}(y_i,y_j;\boldsymbol{w}) - \delta_{ji}(y_i)\Big] \quad (12)$$

and $\gamma_j(y_j) = \sum_{k\in N(j)} \gamma_{kj}(y_j)$. The messages $\delta_{ij}(y_j)$ need to be updated simultaneously for all neighbors of j. The derivation is very similar to that in Globerson and Jaakkola (2008) and is not repeated here. We note that even more efficient updates can be performed, for example by simultaneously updating all δ's that correspond to a tree (Sontag and Jaakkola, 2009).

Because the objective g is not strictly convex, the MSD updates may get trapped in a sub-optimal point (Kolmogorov, 2006). This problem does not occur for binary variables however, as shown in e.g., Globerson and Jaakkola (2008). One way to avoid this is to replace the max function in Eq. 8 with a *soft-max func-*

tion, namely:

$$\max_{y_i} f(y_i) \leq \frac{1}{K} \log \sum_{y_i} e^{Kf(y_i)} \tag{13}$$

The upper bound becomes tight as $K \to \infty$. Note that g with the soft-max is also a convex upper bound on the original loss. Thus we can use g with a sufficiently high K and the technical issue of non-optimal fixed points is alleviated. It also turns out (Johnson et al., 2007) that the MSD updates when the soft-max is used are exactly as in Eq. 10 and Eq. 11, only with the soft-max replacing the max function. The convergence rate of such updates has recently been analyzed in the context of LDPC codes (Burshtein, 2009). In practice we have found that using the original max does not sacrifice optimality, so we do not use the soft-max in practice.

3.2. Subgradient optimization over $\boldsymbol{w}$

The previous section showed how to update the $\delta^{(m)}$ variables such that the objective is decreased at every iteration. We now turn to the update steps on the weight vector $\boldsymbol{w}$. One method that has proven very useful for losses as in Eq. 9 is stochastic subgradient descent (SSD) (Shalev-Shwartz et al., 2007; Ratliff et al., 2007). In SSD, the vector $\boldsymbol{w}$ is changed in the direction of the subgradient of $g(\delta^{(m)}; \boldsymbol{\theta}^{(m)}(\boldsymbol{w}))$ for each sample m. This strategy is especially effective for large datasets since the exact subgradient involves summation over all the sample points.

In the Pegasos method (Shalev-Shwartz et al., 2007; Shalev-Shwartz and Srebro, 2009), the stochastic subgradient is followed by a projection onto a ball of radius $\sqrt{C}$. This is justified by the fact that the optimal $\boldsymbol{w}$ is known to be inside this ball, and results in improved rates of convergence. We follow the same procedure here, since $\boldsymbol{w}$ in our case satisfies the same condition (assuming that the label loss is upper bounded by one, which it is in our case since we use the normalized Hamming loss). The key difference between Pegasos and the method we propose is that we introduce additional variables $\delta^{(m)}$ and minimize with respect to these as well. In the original Pegasos algorithm, one views the objective as a function of $\boldsymbol{w}$ alone, and therefore has to calculate exact subgradients w.r.t. $\boldsymbol{w}$, which requires solving an LP problem at every sample point. As we show in the experiments, this can have a major effect on runtime.

3.3. The DLPW algorithm and convergence

To summarize the above two sections, we propose to solve the structured prediction problem in Eq. 7 by casting it as a joint minimization problem over $\delta^{(m)}$ and $\boldsymbol{w}$ (Eq. 9) and performing coordinate descent updates on $\delta^{(m)}$ together with stochastic subgradient updates on $\boldsymbol{w}$. The overall algorithm, which we call DLPW for *Dual Loss Primal Weights* is described in Algorithm 1. When processing the m^{th} sample point, the algorithm first updates its $\delta^{(m)}$ variables by improving the objective using coordinate descent updates (Eq. 11). Each $\delta_{ij}^{(m)}(y_j)$ should be updated at least once, but in practice it is preferable to perform R passes over the graph, where R is a small number (we use $R = 10$ in our experiments).

Our scheme combines two minimization approaches: stochastic subgradient and coordinate descent. Each is known to converge to the global optimum if used alone (under appropriate conditions on the objective). Although it is not obvious that using them together would have the same guarantees, we show in Appendix A that the combined method will in fact converge to the global optimum. Since the MSD updates for non-binary y_i may get trapped in suboptimal $\delta^{(m)}$, we show convergence for either binary y_i or a soft-max with any K (which in the limit is equivalent to the max function).

Algorithm 1 The DLPW algorithm

Initialize: Choose $\boldsymbol{w}_1$ s.t. $\|\boldsymbol{w}_1\| \leq \sqrt{C}$
for $t = 1$ to T **do**
  Pick a sample point m
  Perform R coordinate descent iterations on all variables $\delta^{(m,t)}$ via the updates in Eq. 11. Denote the new values by $\delta^{(m,t+1)}$.
  Set: $\boldsymbol{w}_{t+\frac{1}{2}} = \boldsymbol{w}_t - \frac{1}{t} \partial_{\boldsymbol{w}_t} g_m(\delta^{(m,t+1)}, \boldsymbol{\theta}^{(m)}(\boldsymbol{w}))$
  Set: $\boldsymbol{w}_{t+1} = \min\left\{1, \frac{\sqrt{C}}{\|\boldsymbol{w}_{t+\frac{1}{2}}\|}\right\} \boldsymbol{w}_{t+\frac{1}{2}}$
end for

4. Previous Approaches

Most algorithmic effort in structured prediction has focused on the case where maximization over the label space y is tractable. Key examples are when the graph G corresponds to a tree (Taskar et al., 2004; Collins, 2002), or where labels correspond to a combinatorial object such as graphs or matchings (Taskar et al., 2006). Below we review these approaches, and highlight their applicability to LP approximations.

In Taskar et al. (2004) the authors noted that although the primal hinge loss involves maximizing over an exponentially large set, its dual has a simpler structure. Specifically, for singly connected graphs G the dual involves only a polynomial number of constraints. They

suggested a dual algorithm similar to the SMO algorithm in Platt (1998), where the dual variables are updated by switching probability mass between two labels y_1, y_2. We note that this dual is different from ours since it also involves dualizing over $\boldsymbol{w}$. This method can also be applied to the LP relaxation case as noted in Taskar et al. (2004, Section 4).

Another dual approach is to use exponentiated gradient steps (Collins et al., 2008). However, these are tailored for the case when marginal inference is tractable and do not seem easily transferable to LP approximations. It is also possible to adapt the perceptron update to the LP case (Kulesza and Pereira, 2008) but this again requires solving the LP at every iteration, and is only exact in the separable case.

Primal methods (such as the one we propose here) operate by updating $\boldsymbol{w}$ directly, and seem to have been used more frequently for structured prediction with LP approximations. One natural approach is to use stochastic (or incremental) subgradient descent on the objective in Eq. 3 (e.g., Shalev-Shwartz and Srebro, 2009; Ratliff et al., 2007). The main drawback of this approach is that calculating the subgradient requires solving the LP approximation after every sample point. This can be quite costly, and is precisely what we avoid doing in the current paper. Another popular primal approach is based on cutting planes (Joachims et al., 2009; Finley and Joachims, 2008). Here one incrementally adds constraints that correspond to vertices of the relaxed polytope. It can be shown that a polynomial number of constraints are sufficient to achieve a given optimization accuracy, but in practice this number may be large. The method we present here avoids this growth in problem size.

The work closest to ours is Taskar et al. (2005), where the dual of only the LP over $\boldsymbol{\mu}$ is taken and the $\boldsymbol{w}$ is kept intact. However, this is done only for problems where the LP is exact (has only integral vertices) and the authors suggest solving the problem via a standard QP solver, as opposed to the efficient coordinate descent message passing procedure we employ here.

5. Experiments

To evaluate our proposed algorithm, we compare its performance on multi-label classification tasks to some of the alternative approaches discussed in Section 4. We will show that DLPW often outperforms the other algorithms, and that it scales well with problem size.

In this multi-label classification setting, each label y_i is a binary random variable indicating whether the i'th label is 'on', and these form a fully connected graph over all label variables. Our model is equivalent to the one used by Finley and Joachims (2008) except that we use an overcomplete representation for feature functions f ($f_i(y_i, \boldsymbol{x})$ is defined for all values y_i and similarly for $f_{ij}(y_i, y_j, \boldsymbol{x})$). The inputs $\boldsymbol{x}$ are vectors in $\mathbb{R}^s$. The feature $f_i(y_i, \boldsymbol{x})$ is a $|y_i| * s$ dimensional vector, i.e., $|y_i|$ concatenated vectors of dimension s. The value of $f_i(y_i, \boldsymbol{x})$ will be $\boldsymbol{x}$ for the vector corresponding to the label y_i and zero elsewhere. The edge feature functions $f_{ij}(y_i, y_j, \boldsymbol{x})$ are indicator vectors, so the length of $\boldsymbol{w}_{ij}$ is $|y_i| * |y_j|$ (4 in the binary case). We use a normalized Hamming loss with $e_i^{(m)}(\boldsymbol{y}) = 1\{y_i^{(m)} \neq y_i\}/d$.

We focus on three datasets of real-world domains taken from the LIBSVM collection (Chang and Lin, 2001) including **Yeast** [14 labels, 1500 training samples, 103 features in $\boldsymbol{x}$] (Elisseeff and Weston, 2001), **Scene** [6 labels, 1211 samples, 294 features] (Boutell et al., 2004), and **Reuters** (subset 1 [3000 samples, 47236 features]) (Lewis et al., 2004). We use a reduction of the Reuters dataset to the 30 most frequent labels. For each dataset, we train a classifier using DLPW and two other algorithms. The first is a cutting-plane algorithm (Finley and Joachims, 2008) and the second is the Pegasos algorithm which uses an LP solver to obtain the approximate MAP at each iteration.[4] The results are shown in Fig. 1(a-c).

As we mentioned in Section 3.3, we limited the number of iterations (MSD updates to all nodes) in DLPW to $R = 10$. We tried a couple of other values for R in this regime and found that these gave similar overall performance. Generally, decreasing R results in faster iterations, but each has a smaller improvement in the objective. On the other hand, if we set no limit on R and allow the MSD algorithm to converge, we get similar performance to that of Pegasos. This makes sense as the LP would be solved completely at each iteration by both algorithms.

Figure 1 shows the objective of Eq. 7, evaluated using the weights found at each iteration, as a function of runtime for each algorithm.[5] For the Yeast dataset we can first see that both subgradient algorithms converge to the optimum much faster than the cutting-plane algorithm. Furthermore, we see that DLPW is

[4] Implementation details: we have implemented all algorithms in C++. The cutting-plane code is taken from `svm_struct` (`http://svmlight.joachims.org/svm_struct.html`) and adapted for the LP case. We use the `GLPK` library (`http://www.gnu.org/software/glpk/glpk.html`) to solve the relaxed LP in both cutting-plane and Pegasos algorithms. We run the experiments on a Dual-Core AMD 2.6 GHz Linux machine.

[5] Not including the time to evaluate Eq. 7.

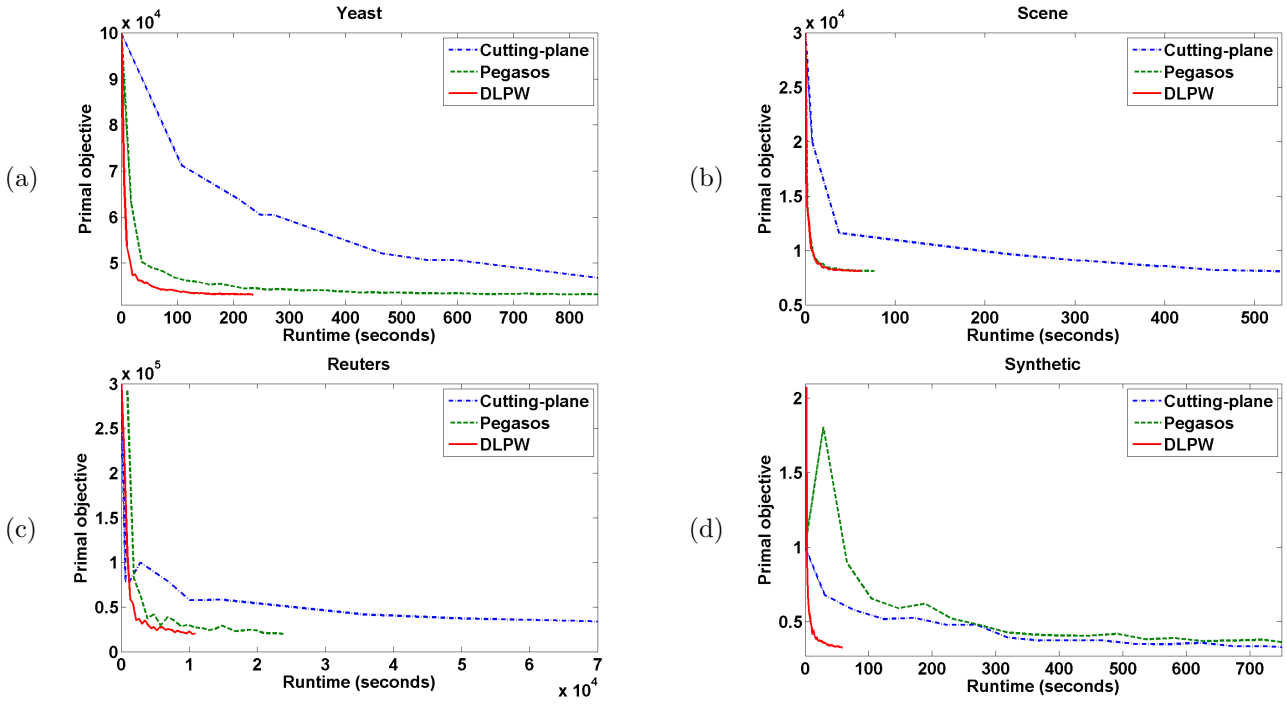

Figure 1. Comparing quality of solution as a function of runtime for various datasets. The x-axis in each subfigure represents the runtime in seconds while the y-axis represents the objective of Eq. 7. Each subfigure shows a line for each of the tested algorithms. Some of the traces were truncated to show more details in the interesting range.

significantly more efficient than Pegasos, which solves the primal LP at each iteration. Specifically, on this dataset DLPW runs 6 times faster than Pegasos and 43 times faster than cutting-plane. In the Scene dataset we see that again the subgradient methods converge much faster than cutting-plane, but here there is only a small advantage for DLPW over Pegasos. This is presumably because the graph is rather small (6 nodes vs. 14 nodes in Yeast) so the LP solver becomes quite efficient. Finally, for the Reuters dataset we observe once more the improved efficiency of the subgradient algorithms. Here DLPW takes less than half the time to converge than Pegasos. We note that when reducing the Reuters dataset to only the 10 most frequent labels, rather than 30, DLPW converges only slightly faster than Pegasos (not shown), which demonstrates the improved scalability of our method as the graph grows in size.

For the multi-label binary models, one could also use graph cut based methods for inference, which would typically be much faster than a general LP solver (e.g., see Finley and Joachims, 2008). However, our method generalizes to the non-binary setting where cut based methods are less effective at solving LPs. Indeed, Figure 1(d) shows results for such a case. For this we use synthetic data similar to the multi-label setting, but with $f_i(y_i, \boldsymbol{x})$ holding x_i in position y_i rather than the whole vector $\boldsymbol{x}$ ($\boldsymbol{x}$ and $\boldsymbol{y}$ are assumed to be of the same length). We run all algorithms on a fully connected graph with $d = 20$, each y_i has 4 states, and the training set consists of 100 samples (these were generated by randomly sampling $\boldsymbol{w}, \boldsymbol{x}$ and obtaining $\boldsymbol{y}$ via iterative conditional modes). We can see that in this setting the cutting-plane algorithm outperforms Pegasos, however DLPW is significantly faster than both.

6. Discussion

We have presented an algorithm for efficient scalable optimization of structured prediction problems that employ approximate inference. Our algorithm dualizes the LP approximation and thus avoids the need to completely solve it at each iteration. The dual can be viewed as an upper bound on the hinge loss (hence the term dual-loss) which can be tightened via auxiliary variables $\delta^{(m)}$. An interesting future direction would be to further explore this notion of tunable surrogate losses further, and study its effects on generalization.

Our empirical results show that our DLPW algorithm improves on methods that employ LP solvers as a black box, and that the improvement is increased as the number of labels grow. A natural question might

be why we could not have simply used the MSD updates within these solvers to replace the black-box LP. There are two problems with this approach. First, we would still be required to iterate the MSD to convergence (since the LP needs to be solved exactly in these methods). Second, there would be an extra step of obtaining a primal solution from the $\delta^{(m)}$ variables, which requires extra work for non-binary labels.

In many structured prediction problems, part of the instance may have special structure for which more efficient inference algorithms are known. In these cases we can make global moves to optimize the $\delta^{(m)}$ variables, e.g. tree block coordinate descent (Sontag and Jaakkola, 2009). Our techniques can also be applied to structured prediction problems other than graphical models, such as parsing, by varying the dual decomposition of the optimization problem used for prediction.

The convergence results presented here are asymptotic. It would be desirable to also derive rate of convergence results. It is possible to show that if the $\delta^{(m)}$ are updated at each iteration until they minimize the dual loss then a rate of $O(\frac{1}{\epsilon})$ is obtained. A similar result can be given for minimization up to a certain accuracy. It thus seems likely we can obtain rate results by employing convergence rates for the $\delta^{(m)}$ updates. Recent work (Burshtein, 2009) has analyzed convergence for a related variant of MSD, and can probably be used in this context.

Finally, our results are easily extended to functions $f(x, y)$ that involve larger cliques on y. This can be done via generalizations of MSD type updates to this case (e.g., the GMPLP algorithm in Globerson and Jaakkola, 2008). Furthermore, such cliques can be introduced to improve the accuracy of the LP approximation (Sontag et al., 2008). We thus expect DLPW to allow improved prediction accuracy for a variety of large scale structured prediction problems.

Acknowledgements

This work was supported by BSF grant 2008303. D.S. was supported by a Google PhD Fellowship.

A. Convergence Proof

To simplify the derivation we assume we only have two delta variables per sample point, and denote those by $\delta_1^{(m)}, \delta_2^{(m)}$. All arguments generalize to $\delta^{(m)}$ with more variables. Our objective thus has the form:

$$h(\boldsymbol{w}, \boldsymbol{\delta}) = \sum_m h_m(\boldsymbol{w}, \delta_1^{(m)}, \delta_2^{(m)}) \tag{14}$$

where h includes the $L2$ regularization on $\boldsymbol{w}$. We assume that h is strictly convex w.r.t. its variables. Strict convexity w.r.t. the δ variables is obtained if we use the soft-max loss (see Eq. 13) for any finite value of K. h is strictly convex w.r.t. $\boldsymbol{w}$ because of the $L2$ regularization. The proof also applies (with minor modifications) to the case where y_i are binary, since in this case the max-sum-diffusion updates do not have non-optimal fixed-points and the strict convexity w.r.t. δ is then not needed in the proof.

We wish to show that our algorithm converges to the global minimum of $h(\boldsymbol{w}, \boldsymbol{\delta})$. The updates of our algorithm are as follows. At iteration t choose the next sample point m (for simplicity we shall assume that the sample points are picked according to the same order at every iteration. We also implicitly assume that m is dependent on t but for brevity drop it from the notation) and:

- Choose $\delta_1^{(m,t+1)}$ to minimize $f(\boldsymbol{w}_t, \delta_1^{(m)}, \delta_2^{(m,t)})$
- Choose $\delta_2^{(m,t+1)}$ to minimize $f(\boldsymbol{w}_t, \delta_1^{(m,t+1)}, \delta_2^{(m)})$
- Set $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha_t \partial_{\boldsymbol{w}} h_m(\boldsymbol{w}_t, \delta_1^{(m,t+1)}, \delta_2^{(m,t+1)})$

Note that the update on $\boldsymbol{w}$ may be viewed as a subgradient on the function $\partial_{\boldsymbol{w}} h_m(\boldsymbol{w}_t, \delta_1^{(m,t+1)}, \delta_2^{(m,t+1)})$ when the latter is understood as a function of $\boldsymbol{w}$.

Using the same derivations as in Nedic and Bertsekas (2001, Lemma 2.1 therein) we arrive at the following inequality, which holds at iteration t for every $\boldsymbol{w}$:

$$\begin{aligned}\|\boldsymbol{w}_{t+1} - \boldsymbol{w}\|^2 \leq \quad & \|\boldsymbol{w}_t - \boldsymbol{w}\|^2 + \alpha_t^2 D^2 \\ & -2\alpha_t \left[h(\boldsymbol{w}_t, \delta^{(m,t+1)}) - h(\boldsymbol{w}, \delta^{(m,t+1)})\right]\end{aligned} \tag{15}$$

where D is an upper bound on the norm of the subgradient (which is indeed bounded by $\sqrt{C} + \hat{R}$ where $\hat{R}$ is the maximum norm of a feature vector $f(\boldsymbol{x}, \boldsymbol{y})$ as in Shalev-Shwartz et al., 2007). Specifically, the above holds for $\boldsymbol{w} = \boldsymbol{w}(\delta^{(m,t+1)}) = \arg\min_{\hat{\boldsymbol{w}}} h_m(\hat{\boldsymbol{w}}, \delta^{(m,t+1)})$ (this $\boldsymbol{w}$ is unique due to the strict convexity of $h(\boldsymbol{w}, \boldsymbol{\delta})$ in $\boldsymbol{w}$ as a result of the L2 regularization). For this $\boldsymbol{w}$ the difference in h objectives above is always non-negative so that by Eq. 15 every iteration brings $\boldsymbol{w}_t$ closer to its optimal values for the current $\delta^{(m,t)}$, provided that the stepsize α_t is sufficiently small. We now wish to take $t \to \infty$ and analyze the limit point $\bar{\boldsymbol{w}}, \bar{\delta}$. It can be shown via standard methods that such a point exists. Furthermore, using similar arguments to Correa and Lemaréchal (1993, Proposition 1.2) we can conclude that: $h(\boldsymbol{w}(\bar{\delta}), \bar{\delta}) = h(\bar{\boldsymbol{w}}, \bar{\delta})$ and thus $\bar{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} h(\boldsymbol{w}, \bar{\delta})$.

We now wish to prove that $\bar{\delta}$ is optimal for $\bar{\boldsymbol{w}}$. This is done as in standard coordinate descent analyses (e.g.,

Bertsekas, 1995, page 274). The update of $\delta_1^{(m,t+1)}$ requires $h_m(\boldsymbol{w}_t, \delta_1^{(m,t+1)}, \delta_2^{(m,t)}) \leq h_m(\boldsymbol{w}_t, \delta_1^{(m)}, \delta_2^{(m,t)})$ for all values of $\delta_1^{(m)}$. Taking $t \to \infty$ we have $h_m(\bar{\boldsymbol{w}}, \bar{\delta}_1^{(m)}, \bar{\delta}_2^{(m)}) \leq h_m(\bar{\boldsymbol{w}}, \delta_1^{(m)}, \bar{\delta}_2^{(m)})$. This implies that the limit value $\bar{\delta}_1^{(m)}$ is optimal with respect to all other coordinate (δ and $\boldsymbol{w}$). We can show this *local optimality* for all coordinates of δ, and by the properties of strictly convex functions we obtain that $\bar{\delta} = \arg\min_\delta h(\bar{\boldsymbol{w}}, \delta)$.

Taken together, the above arguments show that $\bar{\boldsymbol{w}}, \bar{\delta}$ are the minimizers of $h(\boldsymbol{w}, \delta)$ when one of the two variables is fixed to either $\bar{\boldsymbol{w}}$ or $\bar{\delta}$. Strict convexity of h then implies that $\bar{\boldsymbol{w}}, \bar{\delta}$ is the global optimum of $h(\boldsymbol{w}, \boldsymbol{\delta})$.

References

G. H. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data*. The MIT Press, 2007.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.

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

D. Burshtein. Iterative approximate linear programming decoding of ldpc codes with linear complexity. *IEEE Trans. on Information Theory*, 55(11):4835–4859, 2009.

C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

M. Collins. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8, 2002.

M. Collins, A. Globerson, T. Koo, X. Carreras, and P. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *JMLR*, 9:1775–1822, 2008.

R. Correa and C. Lemaréchal. Convergence of some algorithms for convex minimization. *Math. Program.*, 62(2): 261–275, 1993.

A. Elisseeff and J. Weston. Kernel methods for multi-labelled classification and categorical regression problems. In *NIPS 14*, pages 681–687, 2001.

T. Finley and T. Joachims. Training structural svms when exact inference is intractable. In *ICML 25*, pages 304–311, New York, NY, USA, 2008. ACM.

A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *NIPS 20*, pages 553–560. 2008.

T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

J. K. Johnson, D. Malioutov, and A. S. Willsky. Lagrangian relaxation for map estimation in graphical models. In *45th Annual Allerton Conference on Communication, Control and Computing*, September 2007.

V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on PAMI*, 28(10):1568–1583, 2006.

N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *ICCV*, 2007.

A. Kulesza and F. Pereira. Structured learning with approximate inference. In *NIPS 20*, pages 785–792. 2008.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML 18*, pages 282–289, 2001.

D. Lewis, , Y. Yang, T. Rose, and F. Li. RCV1: a new benchmark collection for text categorization research. *JMLR*, 5:361–397, 2004.

A. F. T. Martins, N. A. Smith, and E. P. Xing. Polyhedral outer approximations with application to natural language parsing. In *ICML 26*, pages 713–720, 2009.

A. Nedic and D. P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM J. on Optimization*, 12(1):109–138, 2001.

J. C. Platt. Fast training of Support Vector Machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

N. Ratliff, J. A. D. Bagnell, and M. Zinkevich. (Online) subgradient methods for structured prediction. In *AISTATS*, 2007.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *ICML 24*, pages 807–814, New York, NY, 2007. ACM Press.

S. Shalev-Shwartz and N. Srebro. Theory and practice of support vector machines optimization. In J. Keshet and S. Bengio, editors, *Automatic Speech and Speaker Recognition: Large Margin and Kernel Methods*. 2009.

D. Sontag and T. Jaakkola. Tree block coordinate descent for MAP in graphical models. In *AISTATS 12*, 2009.

D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening LP relaxations for MAP using message passing. In *UAI 24*, pages 503–510, 2008.

B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: a large margin approach. In *ICML 22*, pages 896–903, 2005.

B. Taskar, C. Guestrin, and D. Koller. Max margin Markov networks. In *NIPS 16*, pages 25–32. 2004.

B. Taskar, S. Lacoste-Julien, and M. Jordan. Structured prediction, dual extragradient and Bregman projections. *JMLR*, pages 1627–1653, 2006.

T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1165–1179, 2007.