Journal of Machine Learning Research 7 (2006) 1627–1653 Submitted 10/05; Published 7/06

Structured Prediction, Dual Extragradient and Bregman Projections

Ben Taskar TASKAR@CS.BERKELEY.EDU
Simon Lacoste-Julien SLACOSTE@CS.BERKELEY.EDU
Computer Science
University of California
Berkeley, CA 94720, USA

Michael I. Jordan JORDAN@CS.BERKELEY.EDU
Computer Science and Statistics
University of California
Berkeley, CA 94720, USA

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

We present a simple and scalable algorithm for maximum-margin estimation of structured output models, including an important class of Markov networks and combinatorial models. We formulate the estimation problem as a convex-concave saddle-point problem that allows us to use simple projection methods based on the dual extragradient algorithm (Nesterov, 2003). The projection step can be solved using dynamic programming or combinatorial algorithms for min-cost convex flow, depending on the structure of the problem. We show that this approach provides a memory-efficient alternative to formulations based on reductions to a quadratic program (QP). We analyze the convergence of the method and present experiments on two very different structured prediction tasks: 3D image segmentation and word alignment, illustrating the favorable scaling properties of our algorithm.[1]

Keywords: Markov networks, large-margin methods, structured prediction, extragradient, Bregman projections

1. Introduction

Structured prediction problems are classification or regression problems in which the output variables (the class labels or regression responses) are interdependent. These dependencies may reflect sequential, spatial, recursive or combinatorial structure in the problem domain, and capturing these dependencies is often as important for the purposes of prediction as capturing input-output dependencies. In addition to modeling output correlations, we may wish to incorporate hard constraints between variables. For example, we may seek a model that maps descriptions of pairs of structured objects (shapes, strings, trees, etc.) into alignments of those objects. Real-life examples of such problems include bipartite matchings in alignment of 2D shapes (Belongie et al., 2002) and word alignment of sentences from a source language to a target language in machine translation (Matusov et al., 2004) or non-bipartite matchings of residues in disulfide connectivity prediction for proteins (Baldi et al., 2005). In these examples, the output variables encode presence of edges in the matching and may obey hard one-to-one matching constraints. The prediction problem in such situ-

1. Preliminary versions of some of this work appeared in the proceedings of Advances in Neural Information Processing Systems 19, 2006 and Empirical Methods in Natural Language Processing, 2005.

ations is often solved via efficient combinatorial optimization such as finding the maximum weight matching, where the model provides the appropriate edge weights.

Thus in this paper we define the term *structured output model* very broadly, as a compact scoring scheme over a (possibly very large) set of combinatorial structures and a method for finding the highest scoring structure. For example, when a probabilistic graphical model is used to capture dependencies in a structured output model, the scoring scheme is specified via a factorized probability distribution for the output variables conditional on the input variables, and the search involves some form of generalized Viterbi algorithm. More broadly, in models based on combinatorial problems, the scoring scheme is usually a simple sum of weights associated with vertices, edges, or other components of a structure; these weights are often represented as parametric functions of the inputs. Given training data consisting of instances labeled by desired structured outputs and a set of features that parameterize the scoring function, the (discriminative) learning problem is to find parameters such that the highest scoring outputs are as close as possible to the desired outputs.

In the case of structured prediction based on graphical models, which encompasses most work to date on structured prediction, two major approaches to discriminative learning have been explored: (1) maximum conditional likelihood (Lafferty et al., 2001, 2004) and (2) maximum margin (Collins, 2002; Altun et al., 2003; Taskar et al., 2004b). Both approaches are viable computationally for restricted classes of graphical models. In the broader context of the current paper, however, only the maximum-margin approach appears to be viable. In particular, it has been shown that maximum-margin estimation can be formulated as a tractable convex problem — a polynomial-size quadratic program (QP) — in several cases of interest (Taskar et al., 2004a, 2005a); such results are not available for conditional likelihood. Moreover, it is possible to find interesting subfamilies of graphical models for which maximum-margin methods are provably tractable whereas likelihood-based methods are not. For example, for the Markov random fields that arise in object segmentation problems in vision (Kumar and Hebert, 2004; Anguelov et al., 2005) the task of finding the most likely assignment reduces to a min-cut problem. In these prediction tasks, the problem of finding the highest scoring structure is tractable, while computing the partition function is #P-complete. Essentially, maximum-likelihood estimation requires the partition function, while maximum-margin estimation does not, and thus remains tractable. Polynomial-time sampling algorithms for approximating the partition function for some models do exist (Jerrum and Sinclair, 1993), but have high-degree polynomial complexity and have not yet been shown to be effective for conditional likelihood estimation.

While the reduction to a tractable convex program such as a QP is a significant step forward, it is unfortunately not the case that off-the-shelf QP solvers necessarily provide practical solutions to structured prediction problems. Indeed, despite the reduction to a polynomial number of variables, off-the-shelf QP solvers tend to scale poorly with problem and training sample size for these models. The number of variables is still large and the memory needed to maintain second-order information (for example, the inverse Hessian) is a serious practical bottleneck.

To solve the largest-scale machine learning problems, researchers have often found it expedient to consider simple gradient-based algorithms, in which each individual step is cheap in terms of computation and memory (Platt, 1999; LeCun et al., 1998). Examples of this approach in the structured prediction setting include the Structured Sequential Minimal Optimization algorithm (Taskar et al., 2004b; Taskar, 2004) and the Structured Exponentiated Gradient algorithm (Bartlett et al., 2005). These algorithms are first-order methods for solving QPs arising from low-treewidth Markov random fields and other decomposable models. In these restricted settings these methods can be used to solve significantly larger problems than can be solved with off-the-shelf QP solvers. These methods are, however, limited in scope in that they rely on dynamic programming to compute essential quantities such as gradients. They do not extend to models where dynamic programming is not applicable, for example, to problems such as matchings and min-cuts. Another line of work in learning structured prediction models aims to approximate the arising QPs via constraint generation (Altun et al., 2003; Tsochantaridis et al., 2004). This approach only requires finding the highest scoring structure in the inner loop and incrementally solving a growing QP as constraints are added.

In this paper, we present a solution methodology for structured prediction that encompasses a broad range of combinatorial optimization problems, including matchings, min-cuts and other network flow problems. There are two key aspects to our methodology. The first is that we take a novel approach to the formulation of structured prediction problems, formulating them as saddle-point problems. This allows us to exploit recent developments in the optimization literature, where simple gradient-based methods have been developed for solving saddle-point problems (Nesterov, 2003). Moreover, we show that the key computational step in these methods—a certain projection operation—inherits the favorable computational complexity of the underlying optimization problem. This important result makes our approach viable computationally. In particular, for decomposable graphical models, the projection step is solvable via dynamic programming. For matchings and min-cuts, projection involves a min-cost quadratic flow computation, a problem for which efficient, highly-specialized algorithms are available.

The paper is organized as follows. In Section 2 we present an overview of structured prediction, focusing on three classes of tractable optimization problems. Section 3 shows how to formulate the maximum-margin estimation problem for these models as a saddle-point problem. In Section 4 we discuss the dual extragradient method for solving saddle-point problems and show how it specializes to our setting. We derive a memory-efficient version of the algorithm that requires storage proportional to the number of parameters in the model and is independent of the number of examples in Section 5. In Section 6 we illustrate the effectiveness of our approach on two very different large-scale structured prediction tasks: 3D image segmentation and word alignment in natural language translation. Finally, Section 7 presents our conclusions.

2. Structured Output Models

We begin by discussing three special cases of the general framework that we present subsequently: (1) tree-structured Markov networks, (2) Markov networks with submodular potentials, and (3) a bipartite matching model. Despite significant differences in the formal specification of these models, they share the property that in all cases the problem of finding the highest-scoring output can be formulated as a linear program (LP).

2.1 Tree-Structured Markov Networks

For simplicity of notation, we focus on tree networks, noting in passing that the extension to hypertrees is straightforward. Given N variables, $\mathbf{y} = \{y_1, \ldots, y_N\}$, with discrete domains $y_j \in \mathcal{D}_j = \{\alpha_1, \ldots, \alpha_{|\mathcal{D}_j|}\}$, we define a joint distribution over $\mathcal{Y} = \mathcal{D}_1 \times \ldots \times \mathcal{D}_N$ via

$$P(\mathbf{y}) \propto \prod_{j \in \mathcal{V}} \phi_j(y_j) \prod_{jk \in \mathcal{E}} \phi_{jk}(y_j, y_k),$$

where $(\mathcal{V} = \{1, \ldots, N\}, \mathcal{E} \subset \{jk : j < k, j \in \mathcal{V}, k \in \mathcal{V}\})$ is an undirected graph, and where $\{\phi_j(y_j), j \in \mathcal{V}\}$ are the node potentials and $\{\phi_{jk}(y_j, y_k), jk \in \mathcal{E}\}$ are the edge potentials. We can find the most likely assignment, $\arg\max_{\mathbf{y}} P(\mathbf{y})$, using the Viterbi dynamic programming algorithm for trees. We can also find it using a standard linear programming formulation as follows. We introduce variables $z_{j\alpha}$ to denote indicators $\mathbb{1}(y_j = \alpha)$ for all variables $j \in \mathcal{V}$ and their values $\alpha \in \mathcal{D}_j$. Similarly, we introduce variables $z_{jk\alpha\beta}$ to denote indicators $\mathbb{1}(y_j = \alpha, y_k = \beta)$ for all edges $jk \in \mathcal{E}$ and the values of their nodes, $\alpha \in \mathcal{D}_j, \beta \in \mathcal{D}_k$. We can formulate the problem of finding the maximal probability configuration as follows:

$$\max_{0 \le \mathbf{z} \le 1} \quad \sum_{j \in \mathcal{V}} \sum_{\alpha \in \mathcal{D}_j} z_{j\alpha} \log \phi_j(\alpha) + \sum_{jk \in \mathcal{E}} \sum_{\alpha \in \mathcal{D}_j, \beta \in \mathcal{D}_k} z_{jk\alpha\beta} \log \phi_{jk}(\alpha, \beta) \tag{1}$$

$$\text{s.t.} \quad \sum_{\alpha \in \mathcal{D}_j} z_{j\alpha} = 1, \ \forall j \in \mathcal{V}; \qquad \sum_{\alpha \in \mathcal{D}_j, \beta \in \mathcal{D}_k} z_{jk\alpha\beta} = 1, \quad \forall jk \in \mathcal{E}; \tag{2}$$

$$\sum_{\alpha \in \mathcal{D}_j} z_{jk\alpha\beta} = z_{k\beta}, \ \forall jk \in \mathcal{E}, \beta \in \mathcal{D}_k; \qquad \sum_{\beta \in \mathcal{D}_k} z_{jk\alpha\beta} = z_{j\alpha}, \ \forall jk \in \mathcal{E}, \alpha \in \mathcal{D}_j, \tag{3}$$

where (2) expresses normalization constraints and (3) captures marginalization constraints. This LP has integral optimal solutions if $\mathcal{E}$ is a forest (Chekuri et al., 2001; Wainwright et al., 2002; Chekuri et al., 2005). In networks of general topology, however, the optimal solution can be fractional (as expected, since the problem is NP-hard). Other important exceptions can be found, however, specifically by focusing on constraints on the potentials rather than constraints on the topology. We discuss one such example in the following section.

2.2 Markov Networks with Submodular Potentials

We consider a special class of Markov networks, common in vision applications, in which inference reduces to a tractable min-cut problem (Greig et al., 1989; Kolmogorov and Zabih, 2004). We assume that (1) all variables are binary ($\mathcal{D}_j = \{0,1\}$), and (2) all edge potentials are "regular" (i.e., submodular):

$$\log \phi_{jk}(0,0) + \log \phi_{jk}(1,1) \ge \log \phi_{jk}(1,0) + \log \phi_{jk}(0,1), \qquad \forall jk \in \mathcal{E}. \tag{4}$$

Such potentials prefer assignments where connected nodes have the same label, that is, $y_j = y_k$. This notion of regularity can be extended to potentials over more than two variables (Kolmogorov and Zabih, 2004). These assumptions ensure that the LP in Eq. (1) has integral optimal solutions (Chekuri et al., 2001; Kolmogorov and Wainwright, 2005; Chekuri et al., 2005). Similar kinds of networks (defined also for non-binary variables and non-pairwise potentials) were called "associative Markov networks" by Taskar et al. (2004a) and Anguelov et al. (2005), who used them for object segmentation and hypertext classification.

In figure-ground segmentation (see Fig. 1a), the node potentials capture local evidence about the label of a pixel or range scan point. Edges usually connect nearby pixels in an image, and serve to correlate their labels. Assuming that such correlations tend to be *positive* (connected nodes tend to have the same label) leads us to consider simplified edge potentials of the form $\phi_{jk}(y_j, y_k) = \exp\{-s_{jk}\mathbb{1}(y_j \neq y_k)\}$, where s_{jk} is a nonnegative penalty for assigning y_j and y_k different labels. Note that such potentials are regular if $s_{jk} \ge 0$. Expressing node potentials as $\phi_j(y_j) = \exp\{s_j y_j\}$, we have $P(\mathbf{y}) \propto \exp\left\{\sum_{j \in \mathcal{V}} s_j y_j - \sum_{jk \in \mathcal{E}} s_{jk}\mathbb{1}(y_j \neq y_k)\right\}$. Under this restriction on the potentials, we can obtain the following (simpler) LP:

$$\max_{0 \le \mathbf{z} \le 1} \quad \sum_{j \in \mathcal{V}} s_j z_j - \sum_{jk \in \mathcal{E}} s_{jk} z_{jk} \tag{5}$$

$$\text{s.t.} \quad z_j - z_k \le z_{jk}, \ z_k - z_j \le z_{jk}, \ \forall jk \in \mathcal{E},$$

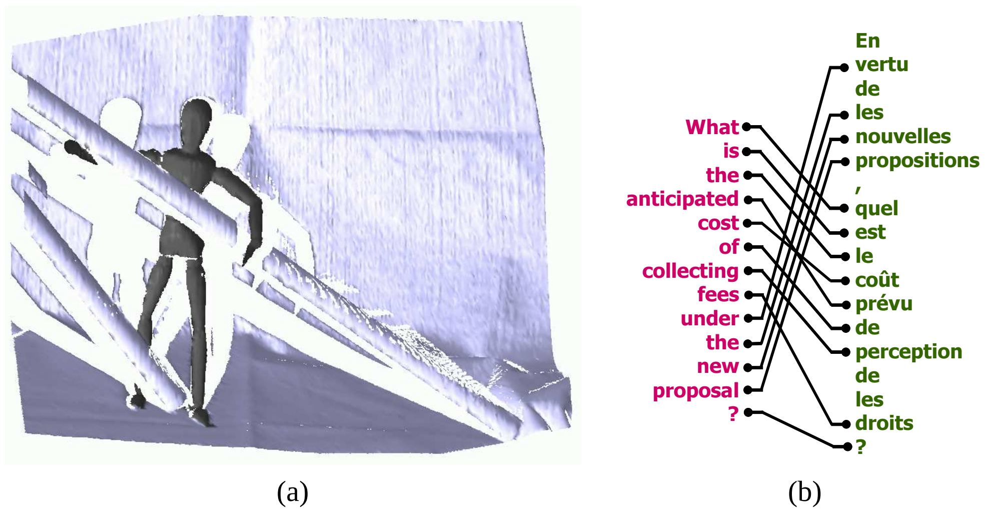


Figure 1: Structured prediction applications: (a) 3D figure-ground segmentation; (b) Word alignment in machine translation.

where the continuous variables z_j correspond to a relaxation of the binary variables y_j, and the constraints encode $z_{jk} = \mathbb{1}(z_j \neq z_k)$. To see this, note that the constraints can be equivalently expressed as $|z_j - z_k| \leq z_{jk}$. Because s_{jk} is positive, $z_{jk} = |z_k - z_j|$ at the maximum, which is equivalent to $\mathbb{1}(z_j \neq z_k)$ if the z_j, z_k variables are binary. An integral optimal solution always exists, since the constraint matrix is totally unimodular (Schrijver, 2003).

We can parameterize the node and edge potentials in terms of user-provided features $\mathbf{x}_j$ and $\mathbf{x}_{jk}$ associated with the nodes and edges. In particular, in 3D range data, $\mathbf{x}_j$ might involve spin-image features or spatial occupancy histograms of a point j, while $\mathbf{x}_{jk}$ might include the distance between points j and k, the dot-product of their normals, etc. The simplest model of dependence is a linear combination of features: $s_j = \mathbf{w}_n^\top \mathbf{f}_n(\mathbf{x}_j)$ and $s_{jk} = \mathbf{w}_e^\top \mathbf{f}_e(\mathbf{x}_{jk})$, where $\mathbf{w}_n$ and $\mathbf{w}_e$ are node and edge parameters, and $\mathbf{f}_n$ and $\mathbf{f}_e$ are node and edge feature mappings, of dimension d_n and d_e, respectively. To ensure non-negativity of s_{jk}, we assume that the edge features $\mathbf{f}_e$ are nonnegative and we impose the restriction $\mathbf{w}_e \geq 0$. This constraint is incorporated into the learning formulation we present below. We assume that the feature mappings $\mathbf{f}$ are provided by the user and our goal is to estimate parameters $\mathbf{w}$ from labeled data. We abbreviate the score assigned to a labeling $\mathbf{y}$ for an input $\mathbf{x}$ as $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_j y_j \mathbf{w}_n^\top \mathbf{f}_n(\mathbf{x}_j) - \sum_{jk \in \mathcal{E}} y_{jk} \mathbf{w}_e^\top \mathbf{f}_e(\mathbf{x}_{jk})$, where $y_{jk} = \mathbb{1}(y_j \neq y_k)$.

2.3 Matchings

Consider modeling the task of word alignment of parallel bilingual sentences (Fig. 1b) as a maximum weight bipartite matching problem in a graph, where the nodes $\mathcal{V} = \mathcal{V}^s \cup \mathcal{V}^t$ correspond to the words in the "source" sentence $(\mathcal{V}^s)$ and the "target" sentence $(\mathcal{V}^t)$ and the edges $\mathcal{E} = \{jk : j \in \mathcal{V}^s, k \in \mathcal{V}^t\}$ correspond to possible alignments between the words. For simplicity, assume that each word aligns to one or zero words in the other sentence. The edge weight s_{jk} represents the degree to which word j in one sentence can translate into the word k in the other sentence. Our objective is to find an alignment that maximizes the sum of edge scores. We represent a matching using a set of binary variables y_{jk} that are set to 1 if word j is assigned to word k in the other sentence, and 0 otherwise. The score of an assignment is the sum of edge scores: $s(\mathbf{y}) = \sum_{jk\in\mathcal{E}} s_{jk}y_{jk}$. The maximum weight bipartite matching problem, $\arg\max_{\mathbf{y}\in\mathcal{Y}} s(\mathbf{y})$, can be found by solving the following LP:

$$\begin{aligned} \max_{0\leq\mathbf{z}\leq 1} \quad & \sum_{jk\in\mathcal{E}} s_{jk}z_{jk} \qquad\qquad (6)\\ \text{s.t.} \quad & \sum_{j\in\mathcal{V}^s} z_{jk} \leq 1,\ \forall k\in\mathcal{V}^t; \qquad \sum_{k\in\mathcal{V}^t} z_{jk} \leq 1,\ \forall j\in\mathcal{V}^s. \end{aligned}$$

where again the continuous variables z_{jk} correspond to the relaxation of the binary variables y_{jk}. As in the min-cut problem, this LP is guaranteed to have integral solutions for any scoring function $s(\mathbf{y})$ (Schrijver, 2003).

For word alignment, the scores s_{jk} can be defined in terms of the word pair jk and input features associated with $\mathbf{x}_{jk}$. We can include the identity of the two words, the relative position in the respective sentences, the part-of-speech tags, the string similarity (for detecting cognates), etc. We let $s_{jk} = \mathbf{w}^\top\mathbf{f}(\mathbf{x}_{jk})$ for a user-provided feature mapping $\mathbf{f}$ and abbreviate $\mathbf{w}^\top\mathbf{f}(\mathbf{x},\mathbf{y}) = \sum_{jk} y_{jk}\mathbf{w}^\top\mathbf{f}(\mathbf{x}_{jk})$.

2.4 General Structure

More generally, we consider prediction problems in which the input $\mathbf{x}\in\mathcal{X}$ is an arbitrary structured object and the output is a vector of values $\mathbf{y} = (y_1,\ldots,y_{L_\mathbf{x}})$ encoding, for example, a matching or a cut in the graph. We assume that the length $L_\mathbf{x}$ and the structure encoded by $\mathbf{y}$ depend deterministically on the input $\mathbf{x}$. In our word alignment example, the output space is defined by the length of the two sentences. Denote the output space for a given input $\mathbf{x}$ as $\mathcal{Y}(\mathbf{x})$ and the entire output space as $\mathcal{Y} = \bigcup_{\mathbf{x}\in\mathcal{X}} \mathcal{Y}(\mathbf{x})$.

Consider the class of structured prediction models $\mathcal{H}$ defined by the linear family:

$$h_\mathbf{w}(\mathbf{x}) = \arg\max_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \mathbf{w}^\top\mathbf{f}(\mathbf{x},\mathbf{y}), \qquad (7)$$

where $\mathbf{f}(\mathbf{x},\mathbf{y})$ is a vector of functions $\mathbf{f}: \mathcal{X}\times\mathcal{Y} \mapsto \mathrm{I\!R}^n$. This formulation is very general. Indeed, it is too general for our purposes—for many $(\mathbf{f},\mathcal{Y})$ pairs, finding the optimal $\mathbf{y}$ is intractable. We specialize to the class of models in which the optimization problem in Eq. (7) can be solved in polynomial time via convex optimization; this is still a very large class of models. Beyond the examples discussed here, it includes weighted context-free grammars and dependency grammars (Manning and Schütze, 1999) and string edit distance models for sequence alignment (Durbin et al., 1998).

3. Large Margin Estimation

We assume a set of training instances $S = \{(\mathbf{x}_i,\mathbf{y}_i)\}_{i=1}^m$, where each instance consists of a structured object $\mathbf{x}_i$ (such as a graph) and a target solution $\mathbf{y}_i$ (such as a matching). Consider learning the parameters $\mathbf{w}$ in the conditional likelihood setting. We can define $P_\mathbf{w}(\mathbf{y}\mid\mathbf{x}) = \frac{1}{Z_\mathbf{w}(\mathbf{x})}\exp\{\mathbf{w}^\top\mathbf{f}(\mathbf{x},\mathbf{y})\}$, where $Z_\mathbf{w}(\mathbf{x}) = \sum_{\mathbf{y}'\in\mathcal{Y}(\mathbf{x})}\exp\{\mathbf{w}^\top\mathbf{f}(\mathbf{x},\mathbf{y}')\}$, and maximize the conditional log-likelihood $\sum_i \log P_\mathbf{w}(\mathbf{y}_i \mid \mathbf{x}_i)$, perhaps with additional regularization of the parameters $\mathbf{w}$. As we have noted earlier, however, the problem of computing the partition function $Z_\mathbf{w}(\mathbf{x})$ is computationally intractable for many of the problems we are interested in. In particular, it is #P-complete for matchings and min-cuts (Valiant, 1979; Jerrum and Sinclair, 1993).

We thus retreat from conditional likelihood and consider the max-margin formulation developed in several recent papers (Collins, 2002; Altun et al., 2003; Taskar et al., 2004b). In this formulation, we seek to find parameters $\mathbf{w}$ such that:

$$\mathbf{y}_i = \arg\max_{\mathbf{y}'_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}(\mathbf{x}_i, \mathbf{y}'_i), \qquad \forall i,$$

where $\mathcal{Y}_i = \mathcal{Y}(\mathbf{x}_i)$. The solution space $\mathcal{Y}_i$ depends on the structured object $\mathbf{x}_i$; for example, the space of possible matchings depends on the precise set of nodes and edges in the graph.

As in univariate prediction, we measure the error of prediction using a loss function $\ell(\mathbf{y}_i, \mathbf{y}'_i)$. To obtain a convex formulation, we upper bound the loss $\ell(\mathbf{y}_i, h_{\mathbf{w}}(\mathbf{x}_i))$ using the hinge function: $\max_{\mathbf{y}'_i \in \mathcal{Y}_i}[\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) + \ell_i(\mathbf{y}'_i) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i)]$, where $\ell_i(\mathbf{y}'_i) = \ell(\mathbf{y}_i, \mathbf{y}'_i)$, and $\mathbf{f}_i(\mathbf{y}'_i) = \mathbf{f}(\mathbf{x}_i, \mathbf{y}'_i)$. Minimizing this upper bound will force the true structure $\mathbf{y}_i$ to be optimal with respect to $\mathbf{w}$ for each instance i:

$$\min_{\mathbf{w} \in \mathcal{W}} \quad \sum_i \max_{\mathbf{y}'_i \in \mathcal{Y}_i} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) + \ell_i(\mathbf{y}'_i)] - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i), \tag{8}$$

where $\mathcal{W}$ is the set of allowed parameters $\mathbf{w}$. We assume that the parameter space $\mathcal{W}$ is a convex set, typically a norm ball $\{\mathbf{w} : ||\mathbf{w}||_p \leq \gamma\}$ with $p = 1, 2$ and a regularization parameter γ. In the case that $\mathcal{W} = \{\mathbf{w} : \; ||\mathbf{w}||_2 \leq \gamma\}$, this formulation is equivalent to the standard large margin formulation using slack variables ξ and slack penalty C (cf. Taskar et al., 2004b), for some suitable values of C depending on γ. The correspondence can be seen as follows: let $\mathbf{w}^*(C)$ be a solution to the optimization problem with slack penalty C and define $\gamma(C) = ||\mathbf{w}^*(C)||$. Then $\mathbf{w}^*$ is also a solution to Eq. (8). Conversely, we can invert the mapping $\gamma(\cdot)$ to find those values of C (possibly non-unique) that give rise to the same solution as Eq. (8) for a specific γ. In the case of submodular potentials, there are additional linear constraints on the edge potentials. In the setting of Eq. (5), we simply constrain $w_e \geq 0$. For general submodular potentials, we can parameterize the log of the edge potential using four sets of edge parameters, $\mathbf{w}_{e00}, \mathbf{w}_{e01}, \mathbf{w}_{e10}, \mathbf{w}_{e11}$, as follows: $\log \phi_{jk}(\alpha, \beta) = \mathbf{w}_{e\alpha\beta}^\top \mathbf{f}(\mathbf{x}_{jk})$. Assuming, as before, that the edge features are nonnegative, the regularity of the potentials can be enforced via a linear constraint: $\mathbf{w}_{e00} + \mathbf{w}_{e11} \geq \mathbf{w}_{e10} + \mathbf{w}_{e01}$, where the inequality should be interpreted componentwise.

The key to solving Eq. (8) efficiently is the *loss-augmented inference problem*,

$$\max_{\mathbf{y}'_i \in \mathcal{Y}_i} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) + \ell_i(\mathbf{y}'_i)]. \tag{9}$$

This optimization problem has precisely the same form as the prediction problem whose parameters we are trying to learn—$\max_{\mathbf{y}'_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i)$—but with an additional term corresponding to the loss function. Tractability of the loss-augmented inference thus depends not only on the tractability of $\max_{\mathbf{y}'_i \in \mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i)$, but also on the form of the loss term $\ell_i(\mathbf{y}'_i)$. A natural choice in this regard is the Hamming distance, which simply counts the number of variables in which a candidate solution $\mathbf{y}'_i$ differs from the target output $\mathbf{y}_i$. In general, we need only assume that the loss function decomposes over the variables in $\mathbf{y}_i$.

In particular, for word alignment, we use weighted Hamming distance, which counts the number of variables in which a candidate matching $\mathbf{y}'_i$ differs from the target alignment $\mathbf{y}_i$, with different cost for false positives (c^+) and false negatives (c^-):

$$\begin{aligned} \ell(\mathbf{y}_i, \mathbf{y}'_i) &= \sum_{jk \in \mathcal{E}_i} \left[c^- y_{i,jk}(1 - y'_{i,jk}) + c^+ \, y'_{i,jk}(1 - y_{i,jk})\right] \qquad (10) \\ &= \sum_{jk \in \mathcal{E}_i} c^- y_{i,jk} + \sum_{jk \in \mathcal{E}_i} [c^+ - (c^- + c^+) y_{i,jk}] y'_{i,jk}, \end{aligned}$$

where $y_{i,jk}$ indicates the presence of edge jk in example i and $\mathcal{E}_i$ is the set of edges in example i. The loss-augmented matching problem can then be written as an LP similar to Eq. (6) (without the constant term $\sum_{jk} c^- y_{i,jk}$):

$$\begin{aligned}\max_{0\le \mathbf{z}_i \le 1} \quad & \sum_{jk\in\mathcal{E}_i} z_{i,jk}[\mathbf{w}^\top \mathbf{f}(\mathbf{x}_{i,jk}) + c^+ - (c^- + c^+)y_{i,jk}] \\ \text{s.t.} \quad & \sum_{j\in\mathcal{V}_i^s} z_{i,jk} \le 1, \ \forall k \in \mathcal{V}_i^t; \qquad \sum_{k\in\mathcal{V}_i^t} z_{i,jk} \le 1, \ \forall j \in \mathcal{V}_i^s,\end{aligned}$$

where $\mathbf{f}(\mathbf{x}_{i,jk})$ is the vector of features of the edge jk in example i and $\mathcal{V}_i^s$ and $\mathcal{V}_i^t$ are the nodes in example i. As before, the continuous variables $z_{i,jk}$ correspond to the binary values $y'_{i,jk}$.

Generally, suppose we can express the prediction problem as an LP:

$$\max_{\mathbf{y}'_i\in\mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) = \max_{\mathbf{z}_i\in\mathcal{Z}_i} \mathbf{w}^\top \mathbf{F}_i \mathbf{z}_i,$$

where

$$\mathcal{Z}_i = \{\mathbf{z}_i : \mathbf{A}_i \mathbf{z}_i \le \mathbf{b}_i,\ 0 \le \mathbf{z}_i \le 1\}, \tag{11}$$

for appropriately defined $\mathbf{F}_i, \mathbf{A}_i$ and $\mathbf{b}_i$. Then we have a similar LP for the loss-augmented inference for each example i:

$$\max_{\mathbf{y}'_i\in\mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) + \ell_i(\mathbf{y}'_i) = d_i + \max_{\mathbf{z}_i\in\mathcal{Z}_i} (\mathbf{F}_i^\top \mathbf{w} + \mathbf{c}_i)^\top \mathbf{z}_i, \tag{12}$$

for appropriately defined d_i and $\mathbf{c}_i$. For the matching case, $d_i = \sum_{jk} c^- y_{i,jk}$ is the constant term, $\mathbf{F}_i$ is a matrix that has a column of features $\mathbf{f}(\mathbf{x}_{i,jk})$ for each edge jk in example i, and $\mathbf{c}_i$ is the vector of the loss terms $c^+ - (c^- + c^+)y_{i,jk}$. Let $\mathbf{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ and $\mathcal{Z} = \mathcal{Z}_1 \times \ldots \times \mathcal{Z}_m$. With these definitions, we have the following saddle-point problem:

$$\min_{\mathbf{w}\in\mathcal{W}} \max_{\mathbf{z}\in\mathcal{Z}} \sum_i \left(\mathbf{w}^\top \mathbf{F}_i \mathbf{z}_i + \mathbf{c}_i^\top \mathbf{z}_i - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i) \right). \tag{13}$$

where we have omitted the constant term $\sum_i d_i$. The only difference between this formulation and our initial formulation in Eq. (8) is that we have created a concise continuous optimization problem by replacing the discrete $\mathbf{y}'_i$'s with continuous $\mathbf{z}_i$'s.

When the prediction problem is intractable (for example, in general Markov networks or tripartite matchings), we can use a convex relaxation (for example, a linear or semidefinite program) to upper bound $\max_{\mathbf{y}'_i\in\mathcal{Y}_i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i)$ and obtain an approximate maximum-margin formulation. This is the approach taken in Taskar et al. (2004b) for general Markov networks using the LP in Eq. (1).

To solve (13), we could proceed by making use of Lagrangian duality. This approach, explored in Taskar et al. (2004a, 2005a), yields a joint convex optimization problem. If the parameter space $\mathcal{W}$ is described by linear and convex quadratic constraints, the result is a convex quadratic program which can be solved using a generic QP solver.

We briefly outline this approach below, but in this paper, we take a different tack, solving the problem in its natural saddle-point form. As we discuss in the following section, this approach allows us to exploit the structure of $\mathcal{W}$ and $\mathcal{Z}$ *separately*, allowing for efficient solutions for a wider range of parameterizations and structures. It also opens up alternatives with respect to numerical algorithms.

Before moving on to solution of the saddle-point problem, we consider the joint convex form when the feasible set has the form of (11) and the loss-augmented inference problem is a LP, as in (12). Using commercial convex optimization solvers for this formulation will provide us with a comparison point for our saddle-point solver. We now proceed to present this alternative form.

To transform the saddle-point form of (13) into a standard convex optimization form, we take the dual of the individual loss-augmented LPs (12):

$$\max_{\mathbf{z}_i \in \mathcal{Z}_i} (\mathbf{F}_i^\top \mathbf{w} + \mathbf{c}_i)^\top \mathbf{z}_i = \min_{(\lambda_i, \mu_i) \in \Lambda_i(\mathbf{w})} \mathbf{b}_i^\top \lambda_i + \mathbf{1}^\top \mu_i \tag{14}$$

where $\Lambda_i(\mathbf{w}) = \{(\lambda_i, \mu_i) \geq 0 : \mathbf{F}_i^\top \mathbf{w} + \mathbf{c}_i \leq \mathbf{A}_i^\top \lambda_i + \mu_i\}$ defines the feasible space for the dual variables λ_i and μ_i. Substituting back in equation (13) and writing $\lambda = (\lambda_1, \ldots, \lambda_m)$, $\mu = (\mu_1, \ldots, \mu_m)$, we obtain (omitting the constant $\sum_i d_i$):

$$\begin{aligned} \min_{\mathbf{w} \in \mathcal{W}, (\lambda, \mu) \geq 0} \quad & \sum_i \left(\mathbf{b}_i^\top \lambda_i + \mathbf{1}^\top \mu_i - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i) \right) \\ \text{s.t.} \quad & \mathbf{F}_i^\top \mathbf{w} + \mathbf{c}_i \leq \mathbf{A}_i^\top \lambda_i + \mu_i \quad i = 1, \ldots, m. \end{aligned} \tag{15}$$

If $\mathcal{W}$ is defined by linear and convex quadratic constraints, the above optimization problem can be solved using standard commercial solvers. The number of variables and constraints in this problem is linear in the number of the parameters *and* the training data (for example nodes and edges).

4. Saddle-Point Problems and the Dual Extragradient Method

We begin by establishing some notation and definitions. Denote the objective of the saddle-point problem in (13) by:

$$\mathcal{L}(\mathbf{w}, \mathbf{z}) \equiv \sum_i \mathbf{w}^\top \mathbf{F}_i \mathbf{z}_i + \mathbf{c}_i^\top \mathbf{z}_i - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i).$$

$\mathcal{L}(\mathbf{w}, \mathbf{z})$ is bilinear in $\mathbf{w}$ and $\mathbf{z}$, with gradient given by: $\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \mathbf{z}) = \sum_i \mathbf{F}_i \mathbf{z}_i - \mathbf{f}_i(\mathbf{y}_i)$ and $\nabla_{\mathbf{z}_i} \mathcal{L}(\mathbf{w}, \mathbf{z}) = \mathbf{F}_i^\top \mathbf{w} + \mathbf{c}_i$.

We can view this problem as a zero-sum game between two players, $\mathbf{w}$ and $\mathbf{z}$. Consider a simple iterative improvement method based on gradient projections:

$$\mathbf{w}^{t+1} = \pi_{\mathcal{W}}(\mathbf{w}^t - \eta \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}^t, \mathbf{z}^t)); \qquad \mathbf{z}_i^{t+1} = \pi_{\mathcal{Z}_i}(\mathbf{z}_i^t + \eta \nabla_{\mathbf{z}_i} \mathcal{L}(\mathbf{w}^t, \mathbf{z}^t)), \tag{16}$$

where η is a step size and $\pi_{\mathcal{V}}(\mathbf{v}) = \arg\min_{\mathbf{v}' \in \mathcal{V}} ||\mathbf{v} - \mathbf{v}'||_2$ denotes the Euclidean projection of a vector $\mathbf{v}$ onto a convex set $\mathcal{V}$. In this simple iteration, each player makes a small best-response improvement without taking into account the effect of the change on the opponent's strategy. This usually leads to oscillations, and indeed, this method is generally not guaranteed to converge for bilinear objectives for any step size (Korpelevich, 1976; He and Liao, 2002). One way forward is to attempt to average the points $(\mathbf{w}^t, \mathbf{z}^t)$ to reduce oscillation. We pursue a different approach that is based on the dual extragradient method of Nesterov (2003). In our previous work (Taskar et al., 2006), we used a related method, the extragradient method due to Korpelevich (1976). The dual extragradient is, however, a more flexible and general method, in terms of the types of projections and feasible sets that can be used, allowing a broader range of structured problems and parameter regularization schemes. Before we present the algorithm, we introduce some notation which will be useful for its description.

Let us combine $\mathbf{w}$ and $\mathbf{z}$ into a single vector, $\mathbf{u} = (\mathbf{w}, \mathbf{z})$, and define the joint feasible space $\mathcal{U} = \mathcal{W} \times \mathcal{Z}$. Note that $\mathcal{U}$ is convex since it is a direct product of convex sets.

We denote the (affine) gradient operator on this joint space as

$$\begin{pmatrix} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \mathbf{z}) \\ -\nabla_{\mathbf{z}_1} \mathcal{L}(\mathbf{w}, \mathbf{z}) \\ \vdots \\ -\nabla_{\mathbf{z}_m} \mathcal{L}(\mathbf{w}, \mathbf{z}) \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & \mathbf{F}_1 \cdots \mathbf{F}_m \\ -\mathbf{F}_1^\top & \\ \vdots & 0 \\ -\mathbf{F}_m^\top & \end{pmatrix}}_{\mathbf{F}} \underbrace{\begin{pmatrix} \mathbf{w} \\ \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_m \end{pmatrix}}_{\mathbf{u}} - \underbrace{\begin{pmatrix} \sum_i \mathbf{f}_i(\mathbf{y}_i) \\ \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_m \end{pmatrix}}_{\mathbf{a}} = \mathbf{F}\mathbf{u} - \mathbf{a}.$$

4.1 Dual Extragradient

We first present the dual extragradient algorithm of Nesterov (2003) using the Euclidean geometry induced by the standard 2-norm, and consider a non-Euclidean setup in Sec. 4.2.

As shown in Fig. 2, the dual extragradient algorithm proceeds using very simple gradient and projection calculations.

```
Initialize: Choose $\hat{\mathbf{u}} \in \mathcal{U}$, set $\mathbf{s}^{-1} = 0$.
Iteration $t$, $0 \le t \le \tau$:
    $\mathbf{v} = \pi_{\mathcal{U}}(\hat{\mathbf{u}} + \eta \mathbf{s}^{t-1})$;
    $\mathbf{u}^t = \pi_{\mathcal{U}}(\mathbf{v} - \eta(\mathbf{F}\mathbf{v} - \mathbf{a}))$;        (17)
    $\mathbf{s}^t = \mathbf{s}^{t-1} - (\mathbf{F}\mathbf{u}^t - \mathbf{a})$.
Output: $\bar{\mathbf{u}}^\tau = \frac{1}{\tau+1} \sum_{t=0}^{\tau} \mathbf{u}^t$.
```

Figure 2: Euclidean dual extragradient.

To relate this generic algorithm to our setting, recall that $\mathbf{u}$ is composed of subvectors $\mathbf{w}$ and $\mathbf{z}$; this induces a commensurate decomposition of the $\mathbf{v}$ and $\mathbf{s}$ vectors into subvectors. To refer to these subvectors we will abuse notation and use the symbols $\mathbf{w}$ and $\mathbf{z}_i$ as indices. Thus, we write $\mathbf{v} = (\mathbf{v}_{\mathbf{w}}, \mathbf{v}_{\mathbf{z}_1}, \ldots, \mathbf{v}_{\mathbf{z}_m})$, and similarly for $\mathbf{u}$ and $\mathbf{s}$. Using this notation, the generic algorithm in Eq. (17) expands into the following dual extragradient algorithm for structured prediction (where the brackets represent gradient vectors):

$$\mathbf{v}_{\mathbf{w}} = \pi_{\mathcal{W}}(\hat{\mathbf{u}}_{\mathbf{w}} + \eta \mathbf{s}_{\mathbf{w}}^{t-1}); \qquad \mathbf{v}_{\mathbf{z}_i} = \pi_{\mathcal{Z}_i}(\hat{\mathbf{u}}_{\mathbf{z}_i} + \eta \mathbf{s}_{\mathbf{z}_i}^{t-1}), \ \forall i;$$

$$\mathbf{u}_{\mathbf{w}}^t = \pi_{\mathcal{W}}\left(\mathbf{v}_{\mathbf{w}} - \eta \left[\sum_i \mathbf{F}_i \mathbf{v}_{\mathbf{z}_i} - \mathbf{f}_i(\mathbf{y}_i)\right]\right); \qquad \mathbf{u}_{\mathbf{z}_i}^t = \pi_{\mathcal{Z}_i}\left(\mathbf{v}_{\mathbf{z}_i} + \eta \left[\mathbf{F}_i^\top \mathbf{v}_{\mathbf{w}} + \mathbf{c}_i\right]\right), \ \forall i;$$

$$\mathbf{s}_{\mathbf{w}}^t = \mathbf{s}_{\mathbf{w}}^{t-1} - \left[\sum_i \mathbf{F}_i \mathbf{u}_{\mathbf{z}_i}^t - \mathbf{f}_i(\mathbf{y}_i)\right]; \qquad \mathbf{s}_{\mathbf{z}_i}^t = \mathbf{s}_{\mathbf{z}_i}^{t-1} + \left[\mathbf{F}_i^\top \mathbf{u}_{\mathbf{w}}^t + \mathbf{c}_i\right], \ \forall i.$$

In the convergence analysis of dual extragradient (Nesterov, 2003), the stepsize η is set to the inverse of the Lipschitz constant (with respect to the 2-norm) of the gradient operator:

$$1/\eta = L \equiv \max_{\mathbf{u}, \mathbf{u}' \in \mathcal{U}} \frac{||\mathbf{F}(\mathbf{u} - \mathbf{u}')||_2}{||\mathbf{u} - \mathbf{u}'||_2} \le ||\mathbf{F}||_2,$$

where $||\mathbf{F}||_2$ is the largest singular value of the matrix $\mathbf{F}$. In practice, various simple heuristics can be considered for setting the stepsize, including search procedures based on optimizing the gap merit function (see, e.g., He and Liao, 2002).

4.1.1 CONVERGENCE

One measure of quality of a saddle-point solution is via the gap function:

$$\mathcal{G}(\mathbf{w},\mathbf{z}) = \left[\max_{\mathbf{z}'\in\mathcal{Z}} \mathcal{L}(\mathbf{w},\mathbf{z}') - \mathcal{L}^*\right] + \left[\mathcal{L}^* - \min_{\mathbf{w}'\in\mathcal{W}} \mathcal{L}(\mathbf{w}',\mathbf{z})\right], \tag{18}$$

where the optimal loss is denoted $\mathcal{L}^* = \min_{\mathbf{w}'\in\mathcal{W}} \max_{\mathbf{z}\in\mathcal{Z}} \mathcal{L}(\mathbf{w},\mathbf{z})$. For non-optimal points $(\mathbf{w},\mathbf{z})$, the gap $\mathcal{G}(\mathbf{w},\mathbf{z})$ is positive and serves as a useful merit function, a measure of accuracy of a solution found by the extragradient algorithm. At an optimum we have

$$\mathcal{G}(\mathbf{w}^*,\mathbf{z}^*) = \max_{\mathbf{z}'\in\mathcal{Z}} \mathcal{L}(\mathbf{w}^*,\mathbf{z}') - \min_{\mathbf{w}'\in\mathcal{W}} \mathcal{L}(\mathbf{w}',\mathbf{z}^*) = 0.$$

Define the Euclidean divergence function as

$$d(\mathbf{v},\mathbf{v}') = \frac{1}{2}||\mathbf{v}-\mathbf{v}'||_2^2,$$

and define a restricted gap function parameterized by positive divergence radii $D_{\mathbf{w}}$ and $D_{\mathbf{z}}$

$$\mathcal{G}_{D_{\mathbf{w}},D_{\mathbf{z}}}(\mathbf{w},\mathbf{z}) = \max_{\mathbf{z}'\in\mathcal{Z}} \left[\mathcal{L}(\mathbf{w},\mathbf{z}') : d(\hat{\mathbf{z}},\mathbf{z}') \le D_{\mathbf{z}}\right] - \min_{\mathbf{w}'\in\mathcal{W}} \left[\mathcal{L}(\mathbf{w}',\mathbf{z}) : d(\hat{\mathbf{w}},\mathbf{w}') \le D_{\mathbf{w}}\right],$$

where the point $\hat{\mathbf{u}} = (\hat{\mathbf{u}}_{\mathbf{w}},\hat{\mathbf{u}}_{\mathbf{z}}) \in \mathcal{U}$ is an arbitrary point that can be thought of as the "center" of $\mathcal{U}$. Assuming there exists a solution $\mathbf{w}^*,\mathbf{z}^*$ such that $d(\hat{\mathbf{w}},\mathbf{w}^*) \le D_{\mathbf{w}}$ and $d(\hat{\mathbf{z}},\mathbf{z}^*) \le D_{\mathbf{z}}$, this restricted gap function coincides with the unrestricted function defined in Eq. (18). The choice of the center point $\hat{\mathbf{u}}$ should reflect an expectation of where the "average" solution lies, as will be evident from the convergence guarantees presented below. For example, we can take $\hat{\mathbf{u}}_{\mathbf{w}} = 0$ and let $\hat{\mathbf{u}}_{\mathbf{z}_i}$ correspond to the encoding of the target $\mathbf{y}_i$.

By Theorem 2 of Nesterov (2003), after τ iterations, the gap of $(\bar{\mathbf{w}}^\tau,\bar{\mathbf{z}}^\tau) = \bar{\mathbf{u}}^\tau$ is upper bounded by:

$$\mathcal{G}_{D_{\mathbf{w}},D_{\mathbf{z}}}(\bar{\mathbf{w}}^\tau,\bar{\mathbf{z}}^\tau) \le \frac{(D_{\mathbf{w}}+D_{\mathbf{z}})L}{\tau+1}. \tag{19}$$

This implies that $O(\frac{1}{\varepsilon})$ steps are required to achieve a desired accuracy of solution ε as measured by the gap function. Note that the exponentiated gradient algorithm (Bartlett et al., 2005) has the same $O(\frac{1}{\varepsilon})$ convergence rate. This sublinear convergence rate is slow compared to interior point methods, which enjoy superlinear convergence (Boyd and Vandenberghe, 2004). However, the simplicity of each iteration, the locality of key operations (projections), and the linear memory requirements make this a practical algorithm when the desired accuracy ε is not too small, and, in particular, these properties align well with the desiderata of large-scale machine learning algorithms. We illustrate these properties experimentally in Section 6.

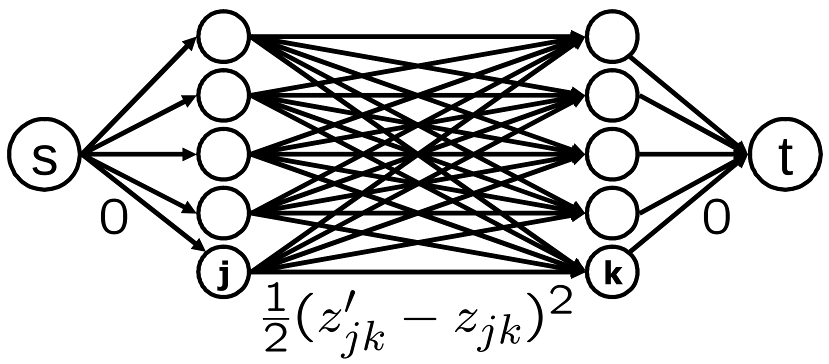


Figure 3: Euclidean projection onto the matching polytope using min-cost quadratic flow. Source s is connected to all the "source" nodes and target t connected to all the "target" nodes, using edges of capacity 1 and cost 0. The original edges jk have a quadratic cost $\frac{1}{2}(z'_{jk} - z_{jk})^2$ and capacity 1.

4.1.2 PROJECTIONS

The efficiency of the algorithm hinges on the computational complexity of the Euclidean projection onto the feasible sets $\mathcal{W}$ and $\mathcal{Z}_i$. In the case of $\mathcal{W}$, projections are cheap when we have a 2-norm ball $\{\mathbf{w} : ||\mathbf{w}||_2 \leq \gamma\}$: $\pi_{\mathcal{W}}(\mathbf{w}) = \gamma\mathbf{w}/\max(\gamma, ||\mathbf{w}||_2)$. Additional non-negativity constraints on the parameters (e.g., $\mathbf{w}_e \geq 0$) can also be easily incorporated by clipping negative values. Projections onto the 1-norm ball are not expensive either (Boyd and Vandenberghe, 2004), but may be better handled by the non-Euclidean setup we discuss below.

We turn to the consideration of the projections onto $\mathcal{Z}_i$. The complexity of these projections is the key issue determining the viability of the extragradient approach for our class of problems. In fact, for both alignment and matchings these projections turn out to reduce to classical network flow problems for which efficient solutions exist. In case of alignment, $\mathcal{Z}_i$ is the convex hull of the bipartite matching polytope and the projections onto $\mathcal{Z}_i$ reduce to the much-studied minimum cost quadratic flow problem (Bertsekas, 1998). In particular, the projection problem $\mathbf{z} = \pi_{\mathcal{Z}_i}(\mathbf{z}'_i)$ can be computed by solving

$$\begin{aligned}
\min_{0 \leq \mathbf{z}_i \leq 1} \quad & \sum_{jk \in \mathcal{E}_i} \frac{1}{2}(z'_{i,jk} - z_{i,jk})^2 \\
\text{s.t.} \quad & \sum_{j \in \mathcal{V}_i^s} z_{i,jk} \leq 1, \ \forall j \in \mathcal{V}_i^t; \qquad \sum_{k \in \mathcal{V}_i^t} z_{i,jk} \leq 1, \ \forall k \in \mathcal{V}_i^s.
\end{aligned}$$

We use a standard reduction of bipartite matching to min-cost flow (see Fig. 3) by introducing a source node s connected to all the words in the "source" sentence, $\mathcal{V}_i^s$, and a target node t connected to all the words in the "target" sentence, $\mathcal{V}_i^t$, using edges of capacity 1 and cost 0. The original edges jk have a quadratic cost $\frac{1}{2}(z'_{i,jk} - z_{i,jk})^2$ and capacity 1. Since the edge capacities are 1, the flow conservation constraints at each original node ensure that the (possibly fractional) degree of each node in a valid flow is at most 1. Now the minimum cost flow from the source s to the target t computes projection of $\mathbf{z}'_i$ onto $\mathcal{Z}_i$.

The reduction of the min-cut polytope projection to a convex network flow problem is more complicated; we present this reduction in Appendix A. Algorithms for solving convex network flow problems (see, for example, Bertsekas et al., 1997) are nearly as efficient as those for solving linear min-cost flow problems, bipartite matchings and min-cuts. In case of word alignment, the running time scales with the cube of the sentence length. We use standard, publicly-available code for solving this problem (Guerriero and Tseng, 2002).[2]

4.2 Non-Euclidean Dual Extragradient

Euclidean projections may not be easy to compute for many structured prediction problems or parameter spaces. The non-Euclidean version of the algorithm of Nesterov (2003) affords flexibility to use other types of (Bregman) projections. The basic idea is as follows. Let $d(\mathbf{u},\mathbf{u}')$ denote a suitable divergence function (see below for a definition) and define a proximal step operator:

$$T_\eta(\mathbf{u},\mathbf{s}) \equiv \arg\max_{\mathbf{u}'\in\mathcal{U}} [\mathbf{s}^\top(\mathbf{u}'-\mathbf{u}) - \frac{1}{\eta}d(\mathbf{u},\mathbf{u}')].$$

Intuitively, the operator tries to make a large step from $\mathbf{u}$ in the direction of $\mathbf{s}$ but not too large as measured by $d(\cdot,\cdot)$. Then the only change to the algorithm is to switch from using a Euclidean projection of a gradient step $\pi_{\mathcal{U}}(\mathbf{u}+\frac{1}{\eta}\mathbf{s})$ to a proximal step in a direction of the gradient $T_\eta(\mathbf{u},\mathbf{s})$ (see Fig. 4):

```
Initialize: Choose $\hat{\mathbf{u}} \in \widetilde{\mathcal{U}}$, set $\mathbf{s}^{-1} = 0$.
Iteration $t$, $0 \le t \le \tau$:

    $\mathbf{v}_\mathbf{w} = T_\eta(\hat{\mathbf{u}}_\mathbf{w}, \mathbf{s}_\mathbf{w}^{t-1})$;    $\mathbf{v}_{\mathbf{z}_i} = T_\eta(\hat{\mathbf{u}}_{\mathbf{z}_i}, \mathbf{s}_{\mathbf{z}_i}^{t-1}),\ \forall i$;
    $\mathbf{u}_\mathbf{w}^t = T_\eta(\mathbf{v}_\mathbf{w}, -\left[\sum_i \mathbf{F}_i\mathbf{v}_{\mathbf{z}_i} - \mathbf{f}_i(\mathbf{y}_i)\right])$;    $\mathbf{u}_{\mathbf{z}_i}^t = T_\eta(\mathbf{v}_{\mathbf{z}_i}, \left[\mathbf{F}_i^\top\mathbf{v}_\mathbf{w} + \mathbf{c}_i\right]),\ \forall i$;
    $\mathbf{s}_\mathbf{w}^t = \mathbf{s}_\mathbf{w}^{t-1} - \left[\sum_i \mathbf{F}_i\mathbf{u}_{\mathbf{z}_i}^t - \mathbf{f}_i(\mathbf{y}_i)\right]$;    $\mathbf{s}_{\mathbf{z}_i}^t = \mathbf{s}_{\mathbf{z}_i}^{t-1} + \left[\mathbf{F}_i^\top\mathbf{u}_\mathbf{w}^t + \mathbf{c}_i\right],\ \forall i$.

Output: $\bar{\mathbf{u}}^\tau = \frac{1}{\tau+1}\sum_{t=0}^{\tau}\mathbf{u}^t$.
```

Figure 4: Non-Euclidean dual extragradient.

To define the range of possible divergence functions and to state convergence properties of the algorithm, we will need some more definitions. We follow the development of Nesterov (2003). Given a norm $||\cdot||^{\mathcal{W}}$ on $\mathcal{W}$ and norms $||\cdot||^{\mathcal{Z}_i}$ on $\mathcal{Z}_i$, we combine them into a norm on $\mathcal{U}$ as

$$||\mathbf{u}|| = \max(||\mathbf{w}||^{\mathcal{W}}, ||\mathbf{z}_1||^{\mathcal{Z}_1}, \ldots, ||\mathbf{z}_m||^{\mathcal{Z}_m}).$$

We denote the dual of $\mathcal{U}$ (the vector space of linear functions on $\mathcal{U}$) as $\mathcal{U}^*$. The norm $||\cdot||$ on the space $\mathcal{U}$ induces the dual norm $||\cdot||_*$ for all $\mathbf{s} \in \mathcal{U}^*$:

$$||\mathbf{s}||_* \equiv \max_{\mathbf{u}\in\mathcal{U}, ||\mathbf{u}||\le 1} \mathbf{s}^\top\mathbf{u}.$$

2. Available from http://www.math.washington.edu/∼tseng/netflowg_nl/.

The Lipschitz constant with respect to this norm (used to set $\eta = 1/L$) is

$$L \equiv \max_{\mathbf{u},\mathbf{u}' \in \mathcal{U}} \frac{||\mathbf{F}(\mathbf{u}-\mathbf{u}')||_*}{||\mathbf{u}-\mathbf{u}'||}.$$

The dual extragradient algorithm adjusts to the geometry induced by the norm by making use of Bregman divergences. We assume a strongly convex function $h(\mathbf{u})$:

$$h(\alpha\mathbf{u}+(1-\alpha)\mathbf{u}') \le \alpha h(\mathbf{u}) + (1-\alpha)h(\mathbf{u}') - \alpha(1-\alpha)\frac{\sigma}{2}||\mathbf{u}-\mathbf{u}'||^2, \qquad \forall \mathbf{u},\mathbf{u}',\alpha \in [0,1],$$

for some $\sigma > 0$, the convexity parameter of $h(\cdot)$. This function is constructed from strongly convex functions on each of the spaces $\mathcal{W}$ and $\mathcal{Z}_i$ by a simple sum: $h(\mathbf{u}) = h(\mathbf{w}) + \sum_i h(\mathbf{z}_i)$. Its conjugate is defined as:

$$h^*(\mathbf{s}) \equiv \max_{\mathbf{u}\in\mathcal{U}} [\mathbf{s}^\top \mathbf{u} - h(\mathbf{u})].$$

Since $h(\cdot)$ is strongly convex, $h^*(\mathbf{u})$ is well-defined and differentiable at any $\mathbf{s} \in \mathcal{U}^*$. We define

$$\widetilde{\mathcal{U}} \equiv \{\nabla h^*(\mathbf{s}) : \mathbf{s} \in \mathcal{U}^*\}.$$

We further assume that $h(\cdot)$ is differentiable at any $\mathbf{u} \in \widetilde{\mathcal{U}}$; since it is also strongly convex, for any two points $\mathbf{u} \in \widetilde{\mathcal{U}}$ and $\mathbf{u}' \in \mathcal{U}$ we have

$$h(\mathbf{u}') \ge h(\mathbf{u}) + \nabla h(\mathbf{u})^\top(\mathbf{u}'-\mathbf{u}) + \frac{\sigma}{2}||\mathbf{u}'-\mathbf{u}||^2,$$

and we can define the Bregman divergence:

$$d(\mathbf{u},\mathbf{u}') = h(\mathbf{u}') - h(\mathbf{u}) - \nabla h(\mathbf{u})^\top(\mathbf{u}'-\mathbf{u}).$$

Note that when $||\cdot||$ is the 2-norm, we can use $h(\mathbf{u}) = \frac{1}{2}||\mathbf{u}||_2^2$, which has convexity parameter $\sigma = 1$, and induces the usual squared Euclidean distance $d(\mathbf{u},\mathbf{u}') = \frac{1}{2}||\mathbf{u}-\mathbf{u}'||_2^2$. When $||\cdot||$ is the 1-norm, we can use the negative entropy $h(\mathbf{u}) = -\mathrm{H}(\mathbf{u})$ (say if $\mathcal{U}$ is a simplex), which also has $\sigma = 1$ and recovers the Kullback-Leibler divergence $d(\mathbf{u},\mathbf{u}') = \mathrm{KL}(\mathbf{u}'||\mathbf{u})$.

With these definitions, the convergence bound in Eq. (19) applies to the non-Euclidean setup, but now the divergence radii are measured using Bregman divergence and the Lipschitz constant is computed with respect to a different norm.

EXAMPLE 1: L_1 REGULARIZATION

Suppose $\mathcal{W} = \{\mathbf{w} : ||\mathbf{w}||_1 \le \gamma\}$. We can transform this constraint set into a simplex constraint by the following variable transformation. Let $\mathbf{w} = \mathbf{w}^+ - \mathbf{w}^-$, $v_0 = 1 - ||\mathbf{w}||_1/\gamma$, and $\mathbf{v} \equiv (v_0, \mathbf{w}^+/\gamma, \mathbf{w}^-/\gamma)$. Then $\mathcal{V} = \{\mathbf{v} : \mathbf{v} \ge 0; \mathbf{1}^\top\mathbf{v} = 1\}$ corresponds to $\mathcal{W}$. We define $h(\mathbf{v})$ as the negative entropy of $\mathbf{v}$:

$$h(\mathbf{v}) = \sum_d v_d \log v_d.$$

The resulting conjugate function and its gradient are given by

$$h^*(\mathbf{s}) = \log \sum_d e^{s_d}; \qquad \frac{\partial h^*(\mathbf{s})}{\partial s_d} = \frac{e^{s_d}}{\sum_d e^{s_d}}.$$

Hence, the gradient space of $h^*(\mathbf{s})$ is the interior of the simplex, $\widetilde{\mathcal{V}} = \{\mathbf{v} : \mathbf{v} > 0; \mathbf{1}^\top \mathbf{v} = 1\}$. The corresponding Bregman divergence is the standard Kullback-Leibler divergence

$$d(\mathbf{v}, \mathbf{v}') = \sum_d v'_d \log \frac{v'_d}{v_d}, \quad \forall \mathbf{v} \in \widetilde{\mathcal{V}}, \mathbf{v}' \in \mathcal{V},$$

and the Bregman proximal step or projection, $\tilde{\mathbf{v}} = T_\eta(\mathbf{v}, \mathbf{s}) = \arg\max_{\mathbf{v}' \in \mathbf{v}} [\mathbf{s}^\top \mathbf{v}' - \frac{1}{\eta} d(\mathbf{v}, \mathbf{v}')]$ is given by a multiplicative update:

$$\tilde{v}_d = \frac{v_d e^{\eta s_d}}{\sum_d v_d e^{\eta s_d}}.$$

Note that we cannot choose $\hat{\mathbf{u}}_\mathbf{v} = (1, \mathbf{0}, \mathbf{0})$ as the center of $\widetilde{\mathcal{V}}$—given that the updates are multiplicative the algorithm will not make any progress in this case. In fact, this choice is precluded by the constraint that $\hat{\mathbf{u}}_\mathbf{v} \in \widetilde{\mathcal{V}}$, not just $\hat{\mathbf{u}}_\mathbf{v} \in \mathcal{V}$. A reasonable choice is to set $\hat{\mathbf{u}}_\mathbf{v}$ to be the center of the simplex $\mathcal{V}$, $\hat{\mathbf{u}}_{v_d} = \frac{1}{|\mathcal{V}|} = \frac{1}{2|\mathcal{W}|+1}$.

EXAMPLE 2: TREE-STRUCTURED MARGINALS

Consider the case in which each example i corresponds to a tree-structured Markov network, and $\mathcal{Z}_i$ is defined by the normalization and marginalization constraints in Eq. (2) and Eq. (3) respectively. These constraints define the space of valid marginals. For simplicity of notation, we assume that we are dealing with a single example i and drop the explicit index i. Let us use a more suggestive notation for the components of $\mathbf{z}$: $z_j(\alpha) = z_{j\alpha}$ and $z_{jk}(\alpha, \beta) = z_{jk\alpha\beta}$. We can construct a natural joint probability distribution via

$$P_\mathbf{z}(\mathbf{y}) = \prod_{jk \in \mathcal{E}} z_{jk}(y_j, y_k) \prod_{j \in \mathcal{V}} (z_j(y_j))^{1-q_j},$$

where q_j is the number of neighbors of node j. Now $\mathbf{z}$ defines a point on the simplex of joint distributions over $\mathcal{Y}$, which has dimension $|\mathcal{Y}|$. One natural measure of complexity in this enlarged space is the 1-norm. We define $h(\mathbf{z})$ as the negative entropy of the distribution represented by $\mathbf{z}$:

$$h(\mathbf{z}) = \sum_{jk \in \mathcal{E}} \sum_{\alpha \in \mathcal{D}_j, \beta \in \mathcal{D}_k} z_{jk}(\alpha, \beta) \log z_{jk}(\alpha, \beta) + (1 - q_j) \sum_{j \in \mathcal{V}} \sum_{\alpha \in \mathcal{D}_j} z_j(\alpha) \log z_j(\alpha).$$

The resulting $d(\mathbf{z}, \mathbf{z}')$ is the Kullback-Leibler divergence $\mathrm{KL}(P_{\mathbf{z}'}||P_\mathbf{z})$. The corresponding Bregman step or projection operator, $\tilde{\mathbf{z}} = T_\eta(\mathbf{z}, \mathbf{s}) = \arg\max_{\mathbf{z}' \in \mathcal{Z}} [\mathbf{s}^\top \mathbf{z}' - \frac{1}{\eta} \mathrm{KL}(P_{\mathbf{z}'}||P_\mathbf{z})]$ is given by a multiplicative update on the space of distributions:

$$P_{\tilde{\mathbf{z}}}(\mathbf{y}) = \frac{1}{Z} P_\mathbf{z}(\mathbf{y}) e^{\eta[\sum_{jk} s_{jk}(y_j, y_k) + \sum_j s_j(y_j)]} = \frac{1}{Z} \prod_{jk} z_{jk}(y_j, y_k) e^{\eta s_{jk}(y_j, y_k)} \prod_j (z_j(y_j))^{1-q_j} e^{\eta s_j(y_j)},$$

where we use the same indexing for the dual space vector $\mathbf{s}$ as for $\mathbf{z}$ and Z is a normalization constant. Hence, to obtain the projection $\tilde{\mathbf{z}}$, we compute the node and edge marginals of the distribution $P_{\tilde{\mathbf{z}}}(\mathbf{y})$ via the standard sum-product dynamic programming algorithm using the node and edge potentials defined above. Note that the form of the multiplicative update of the projection resembles that of exponentiated gradient. As in the example above, we cannot let $\hat{\mathbf{u}}_\mathbf{z}$ be a corner (or any boundary point) of the simplex since $\widetilde{\mathcal{Z}}$ does not include it. A reasonable choice for $\hat{\mathbf{u}}_\mathbf{z}$ would be either the center of the simplex or a point near the target structure but in the interior of the simplex.

5. Memory-Efficient Formulation

Consider the memory requirements of the algorithm. The algorithm maintains the vector $\mathbf{s}^\tau$ as well as the running average, $\bar{\mathbf{u}}^\tau$, a total dimensionality of $|\mathcal{W}|+|\mathcal{Z}|$. Note, however, that these vectors are related very simply by:

$$\mathbf{s}^\tau = -\sum_{t=0}^{\tau}(\mathbf{F}\mathbf{u}^t - \mathbf{a}) = -(\tau+1)(\mathbf{F}\bar{\mathbf{u}}^\tau - \mathbf{a}).$$

So it suffices to only maintain the running average $\bar{\mathbf{u}}^\tau$ and reconstruct $\mathbf{s}$ as needed.

In problems in which the number of examples, m, is large we can take advantage of the fact that the memory needed to store the target structure $\mathbf{y}_i$ is often much smaller than the corresponding vector $\mathbf{z}_i$. For example, for word alignment, we need $O(|\mathcal{V}_i^s|\log|\mathcal{V}_i^t|)$ bits to encode a matching $\mathbf{y}_i$ by using roughly $\log\mathcal{V}_i^t$ bits per node in $\mathcal{V}_i^s$ to identify its match. By contrast, we need $|\mathcal{V}_i^s||\mathcal{V}_i^t|$ floating numbers to maintain $\mathbf{z}_i$. The situation is worse in context-free parsing, where a parse tree $\mathbf{y}_i$ requires space linear in the sentence length and logarithmic in grammar size, while $|\mathcal{Z}_i|$ is the product of the grammar size and the cube of the sentence length.

Note that from $\bar{\mathbf{u}}^\tau = (\bar{\mathbf{u}}^\tau_{\mathbf{w}}, \bar{\mathbf{u}}^\tau_{\mathbf{z}})$, we only care about $\bar{\mathbf{u}}^\tau_{\mathbf{w}}$, the parameters of the model, while the other component, $\bar{\mathbf{u}}^\tau_{\mathbf{z}}$, maintains the state of the algorithm. Fortunately, we can eliminate the need to store $\bar{\mathbf{u}}_{\mathbf{z}}$ by maintaining it implicitly, at the cost of storing a vector of size $|\mathcal{W}|$. This allows us to essentially have the same small memory footprint of online-type learning methods, where a single example is processed at a time and only a vector of parameters is maintained. In particular, instead of maintaining the entire vector $\bar{\mathbf{u}}^t$ and reconstructing $\mathbf{s}^t$ from $\bar{\mathbf{u}}^t$, we can instead store only $\bar{\mathbf{u}}^t_{\mathbf{w}}$ and $\mathbf{s}^t_{\mathbf{w}}$ between iterations, since

$$\mathbf{s}^t_{\mathbf{z}_i} = (t+1)(\mathbf{F}_i^\top \bar{\mathbf{u}}^t_{\mathbf{w}} + \mathbf{c}_i).$$

The diagram in Fig. 5 illustrates the process and the algorithm is summarized in Fig. 6. We use two "temporary" variables $\mathbf{v}_{\mathbf{w}}$ and $\mathbf{r}_{\mathbf{w}}$ of size $|\mathcal{W}|$ to maintain intermediate quantities. The additional vector $\mathbf{q}_{\mathbf{w}}$ shown in Fig. 5 is introduced only to allow the diagram to be drawn in a simplified manner; it can be eliminated by using $\mathbf{s}_{\mathbf{w}}$ to accumulate the gradients as shown in Fig. 6. The total amount of memory needed is thus four times the number of parameters plus memory for a single example $(\mathbf{v}_{\mathbf{z}_i}, \mathbf{u}_{\mathbf{z}_i})$. We assume that we do not need to store $\hat{\mathbf{u}}_{\mathbf{z}_i}$ explicitly but can construct it efficiently from $(\mathbf{x}_i, \mathbf{y}_i)$.

Note that in case the dimensionality of the parameter space is much larger than the dimensionality of $\mathcal{Z}$, we can use a similar trick to only store variables of the size of $\mathbf{z}$. In fact, if $\mathcal{W} = \{\mathbf{w} : ||\mathbf{w}||_2 \le \gamma\}$ and we use Euclidean projections onto $\mathcal{W}$, we can exploit kernels to define infinite-dimensional feature spaces and derive a kernelized version of the algorithm.

6. Experiments

In this section we describe experiments focusing on two of the structured models we described earlier: bipartite matchings for word alignments and restricted potential Markov nets for 3D segmentation.[3] We compared three algorithms: the dual extragradient (`dual-ex`), the averaged projected gradient (`proj-grad`) defined in Eq. (16), and the averaged perceptron (Collins, 2002). For

3. Software implementing our dual extragradient algorithm can be found at `http://www.cs.berkeley.edu/∼slacoste/research/dualex`.

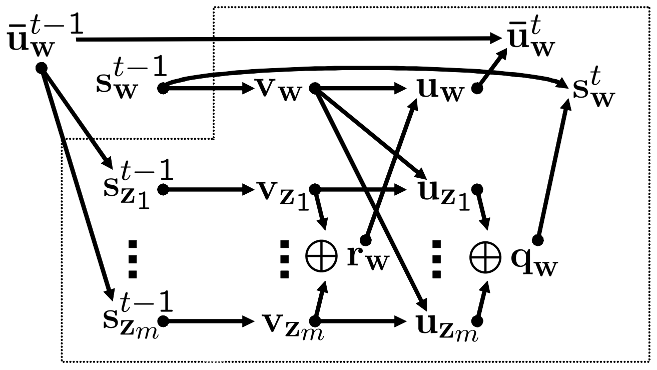

$\bar{\mathbf{u}}_{\mathbf{w}}^{t-1}$
$\bar{\mathbf{u}}_{\mathbf{w}}^{t}$
$\mathbf{s}_{\mathbf{w}}^{t-1}$
$\mathbf{v}_{\mathbf{w}}$
$\mathbf{u}_{\mathbf{w}}$
$\mathbf{s}_{\mathbf{w}}^{t}$
$\mathbf{s}_{\mathbf{z}_1}^{t-1}$
$\mathbf{v}_{\mathbf{z}_1}$
$\mathbf{u}_{\mathbf{z}_1}$
$\oplus$ $\mathbf{r}_{\mathbf{w}}$
$\oplus$ $\mathbf{q}_{\mathbf{w}}$
$\mathbf{s}_{\mathbf{z}_m}^{t-1}$
$\mathbf{v}_{\mathbf{z}_m}$
$\mathbf{u}_{\mathbf{z}_m}$

Figure 5: Dependency diagram for memory-efficient dual extragradient. The dotted box represents the computations of an iteration of the algorithm. Only $\bar{\mathbf{u}}_{\mathbf{w}}^t$ and $\mathbf{s}_{\mathbf{w}}^t$ are kept between iterations. Each example is processed one by one and the intermediate results are accumulated as $\mathbf{r}_{\mathbf{w}} = \mathbf{r}_{\mathbf{w}} - \mathbf{F}_i\mathbf{v}_{\mathbf{z}_i} + \mathbf{f}_i(\mathbf{y}_i)$ and $\mathbf{q}_{\mathbf{w}} = \mathbf{q}_{\mathbf{w}} - \mathbf{F}_i\mathbf{u}_{\mathbf{z}_i} + \mathbf{f}_i(\mathbf{y}_i)$. Details shown in Fig. 6, except that intermediate variables $\mathbf{u}_{\mathbf{w}}$ and $\mathbf{q}_{\mathbf{w}}$ are only used here for pictorial clarity.

```
Initialize: Choose $\hat{\mathbf{u}} \in \widetilde{\mathcal{U}}$, $\mathbf{s}_{\mathbf{w}} = 0$, $\bar{\mathbf{u}}_{\mathbf{w}} = 0$, $\eta = 1/L$.
Iteration $t$, $0 \le t \le \tau$:
    $\mathbf{v}_{\mathbf{w}} = T_\eta(\hat{\mathbf{u}}_{\mathbf{w}}, \mathbf{s}_{\mathbf{w}})$;     $\mathbf{r}_{\mathbf{w}} = 0$.
    Example $i$, $1 \le i \le m$:
        $\mathbf{v}_{\mathbf{z}_i} = T_\eta(\hat{\mathbf{u}}_{\mathbf{z}_i}, t(\mathbf{F}_i^\top \bar{\mathbf{u}}_{\mathbf{w}} + \mathbf{c}_i))$;     $\mathbf{r}_{\mathbf{w}} = \mathbf{r}_{\mathbf{w}} - \mathbf{F}_i\mathbf{v}_{\mathbf{z}_i} + \mathbf{f}_i(\mathbf{y}_i)$;
        $\mathbf{u}_{\mathbf{z}_i} = T_\eta(\mathbf{v}_{\mathbf{z}_i}, \mathbf{F}_i^\top \mathbf{v}_{\mathbf{w}} + \mathbf{c}_i)$;     $\mathbf{s}_{\mathbf{w}} = \mathbf{s}_{\mathbf{w}} - \mathbf{F}_i\mathbf{u}_{\mathbf{z}_i} + \mathbf{f}_i(\mathbf{y}_i)$.

    $\bar{\mathbf{u}}_{\mathbf{w}} = \frac{t\bar{\mathbf{u}}_{\mathbf{w}} + T_\eta(\mathbf{v}_{\mathbf{w}}, \mathbf{r}_{\mathbf{w}})}{t+1}$.
Output $\mathbf{w} = \bar{\mathbf{u}}_{\mathbf{w}}$.
```

Figure 6: Memory-efficient dual extragradient.

`dual-ex` and `proj-grad`, we used Euclidean projections, which can be formulated as min-cost quadratic flow problems. We used $\mathbf{w} = 0$ and $\mathbf{z}_i$ corresponding to $\mathbf{y}_i$ as the centroid $\hat{\mathbf{u}}$ in `dual-ex` and as the starting point of `proj-grad`.

In our experiments, we consider standard L_2 regularization, $\{||\mathbf{w}||_2 \le \gamma\}$. A question which arises in practice is how to choose the regularization parameter γ. The typical approach is to run the algorithm for several values of the regularization parameter and pick the best model using a validation set. This can be quite expensive, though, and several recent papers have explored techniques for obtaining the whole regularization path, either exactly (Hastie et al., 2004), or approximately using path following techniques (Rosset, 2004). Instead, we run the algorithm without regularization ($\gamma = \infty$) and track its performance on the validation set, selecting the model with best performance. For comparison, whenever feasible with the available memory, we used commercial software to compute points on the regularization path. As we discuss below, the dual extragradient algorithm approximately follows the regularization path in our experiments (in terms of the training objective and test error) in the beginning and the end of the range of γ and often performs better in terms of generalization error in the mid-range.

6.1 Object Segmentation

We tested our algorithm on a 3D scan segmentation problem using the class of Markov networks with regular potentials that were described above. The dataset is a challenging collection of cluttered scenes containing articulated wooden puppets (Anguelov et al., 2005). It contains eleven different single-view scans of three puppets of varying sizes and positions, with clutter and occluding objects such as rope, sticks and rings. Each scan consists of around $7,000$ points. Our goal was to segment the scenes into two classes—puppet and background. We use five of the scenes for our training data, three for validation and three for testing. Sample scans from the training and test set can be seen at `http://www.cs.berkeley.edu/~taskar/3DSegment/`. We computed spin images of size 10×5 bins at two different resolutions, then scaled the values and performed PCA to obtain 45 principal components, which comprised our node features. We used the surface links output by the scanner as edges between points and for each edge only used a single feature, set to a constant value of 1 for all edges. This results in all edges having the same potential. The training data contains approximately $37,000$ nodes and $88,000$ edges. We used standard Hamming distance for our loss function $\ell(\mathbf{y}_i, \mathbf{y}'_i)$.

We compared the performance of the dual extragradient algorithm along its unregularized path to solutions of the regularized problems for different settings of the norm.[4] For dual extragradient, the stepsize is set to $\eta = 1/||\mathbf{F}||_2 \approx 0.005$. We also compared to a variant of the averaged perceptron algorithm (Collins, 2002), where we use the batch updates to stabilize the algorithm, since we only have five training examples. We set the learning rate to 0.0007 by trying several values and picking the best value based on the validation data.

In Fig. 7(a) we track the hinge loss on the training data:

$$\sum_i \max_{\mathbf{y}'_i \in \mathcal{Y}_i} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}'_i) + \ell_i(\mathbf{y}'_i)] - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}_i). \tag{20}$$

The hinge loss of the regularization path (`reg-path`) is the minimum loss for a given norm, and hence is always lower than the hinge loss of the other algorithms. However, as the norm increases and the model approaches the unregularized solution, `dual-ex` loss tends towards that of `reg-path`. Note that `proj-grad` behaves quite erratically in the range of the norms shown. Fig. 7(b) shows the growth of the norm as a function of iteration number for `dual-ex` and `ave-perc`. The unregularized dual extragradient seems to explore the range of models (in terms on their norm) on the regularization path more thoroughly than the averaged perceptron and eventually asymptotes to the unregularized solution, while `proj-grad` quickly achieves very large norm.

Fig. 7(c) and Fig. 7(d) show validation and test error for the three algorithms. The best validation and test error achieved by the `dual-ex` and `ave-perc` algorithms as well as `reg-path` are fairly close, however, this error level is reached at very different norms. Since the number of scenes in the validation and test data is very small (three), because of variance, the best norm on validation is not very close to the best norm on the test set. Selecting the best model on the validation set leads to test errors of 3.4% for `dual-ex`, 3.5% for `ave-perc`, 3.6% for `reg-path` and 3.8% for `proj-grad`

4. We used CPLEX to solve the regularized problems and also to find the projections onto the min-cut polytope, since the min-cost quadratic flow code we used (Guerriero and Tseng, 2002) does not support negative flows on edges, which are needed in the formulation presented in Appendix A.

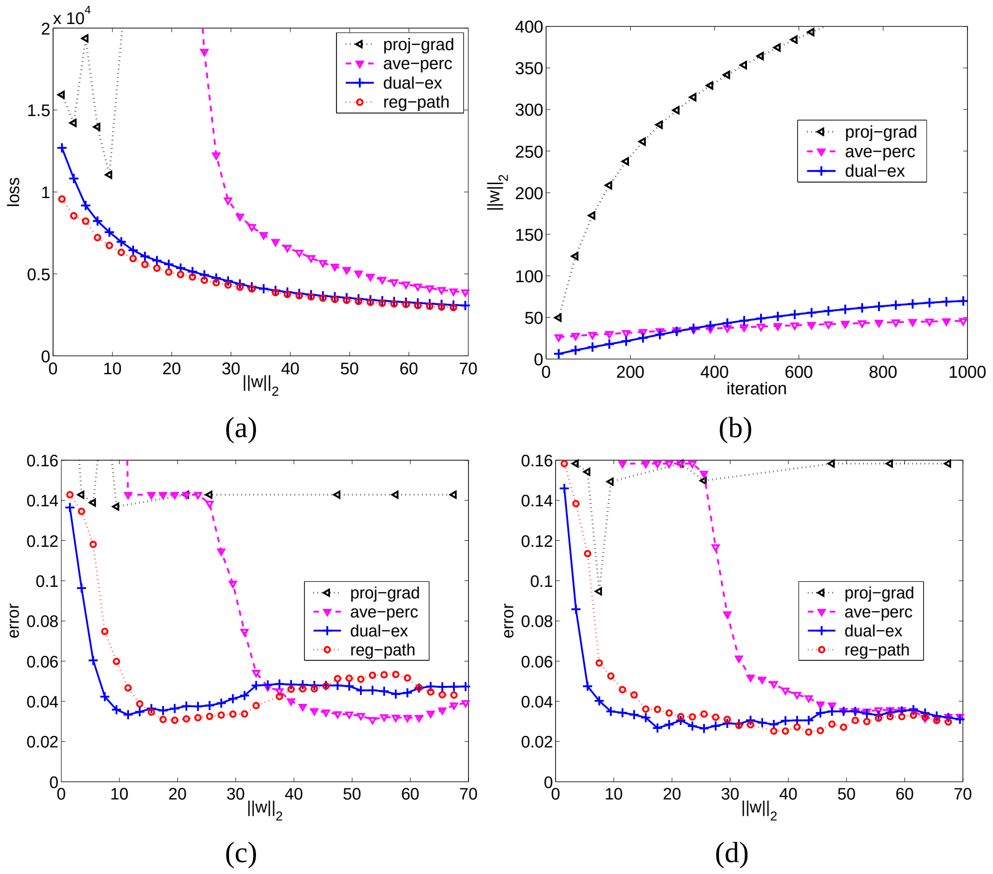


Figure 7: Object segmentation results: (a) Training hinge loss for the regularization path (`reg-path`), the averaged projected gradient (`proj-grad`), the averaged perceptron (`ave-perc`) and unregularized dual extragradient (`dual-ex`) vs. the norm of the parameters. (b) Norm of the parameters vs. iteration number for the three algorithms. (c) Validation error vs. the norm of the parameters. (d) Test error vs. the norm of the parameters.

(`proj-grad` actually improves performance after the model norm is larger than 500, which is not shown in the graphs).

6.2 Word Alignment

We also tested our algorithm on word-level alignment using a data set from the 2003 NAACL set (Mihalcea and Pedersen, 2003), the English-French Hansards task. This corpus consists of 1.1M pairs of sentences, and comes with a validation set of 37 sentence pairs and a test set of 447 word-aligned sentences. The validation and test sentences have been hand-aligned (see Och and Ney, 2003) and are marked with both *sure* and *possible* alignments. Using these alignments, the *alignment error rate* (AER) is calculated as:

$$AER(A,S,P) = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|},$$

where A is a set of proposed alignment pairs, S is the set of sure gold pairs, and P is the set of possible gold pairs (where $S \subseteq P$).

We experimented with two different training settings. In the first one, we split the original test set into 100 training examples and 347 test examples—this dataset is called the 'Gold' dataset. In the second setting, we used GIZA++ (Och and Ney, 2003) to produce IBM Model 4 alignments for the unlabeled sentence pairs. We took the intersection of the predictions of the English-to-French and French-to-English Model 4 alignments on the first 5000 sentence pairs from the 1.1M sentences in order to experiment with the scaling of our algorithm (training on 500, 1000 and 5000 sentences). The number of edges for 500, 1000 and 5000 sentences of GIZA++ were about 31,000, 99,000 and 555,000 respectively. We still tested on the 347 Gold test examples, and used the validation set to select the stopping point. The stepsize for the dual extragradient algorithm was chosen to be $1/||\mathbf{F}||_2$.

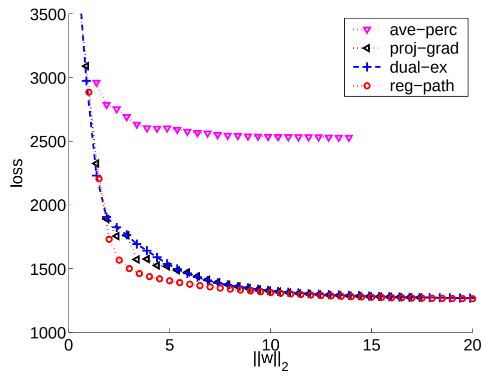
We used statistics computed on the 1.1M sentence pairs as the edge features for our model. A detailed analysis of the constructed features and corresponding error analysis is presented in Taskar et al. (2005b). Example features include: a measure of mutual information between the two words computed from their co-occurrence in the aligned sentences (Dice coefficient); the difference between word positions; character-based similarity features designed to capture cognate (and exact match) information; and identity of the top five most frequently occurring words. We used the structured loss $\ell(\mathbf{y}_i, \mathbf{y}'_i)$ defined in Eq. (10) with $(c^+, c^-) = (1, 3)$ (where 3 was selected by testing several values on the validation set). We obtained low recall when using equal cost for both type of errors because the number of positive edges is significantly smaller than the number of negative edges, and so it is safe (precision-wise) for the model to predict fewer edges, hurting the recall. Increasing the cost for false negatives solves this problem.

Fig. 8(a) and Fig. 8(e) compare the hinge loss of the regularization path with the evolution of the objective for the unregularized dual extragradient, averaged projected gradient and averaged perceptron algorithms when trained on the Gold data set, 500 sentences and 1000 sentences of the GIZA++ output respectively.[5] The dual extragradient path appears to follow the regularization path closely for $||\mathbf{w}|| \leq 2$ and $||\mathbf{w}|| \geq 12$. Fig. 8(b) compares the AER on the test set along the dual extragradient path trained on the Gold dataset versus the regularization path AER. The results on the validation set for each path are also shown. On the Gold data set, the minimum AER was reached roughly after 200 iterations.

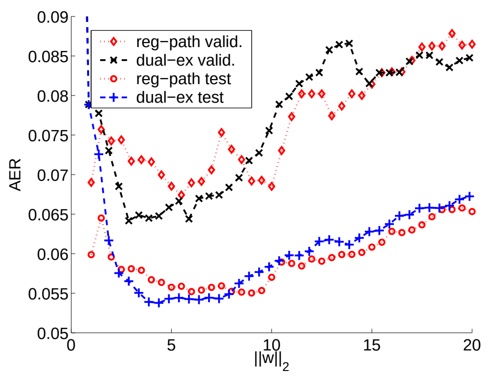
Interestingly, the unregularized dual extragradient path seems to give better performance on the test set than that obtained by optimizing along the regularization path. The dominance of the dual extragradient path over the regularization path is more salient in figure 8(f) for the case where both are trained on 1000 sentences from the GIZA++ output. We conjecture that the dual extragradient method provides additional statistical regularization (compensating for the noisier labels of the GIZA++ output) by enforcing local smoothness of the path in parameter space.

The averaged projected gradient performed much better for this task than segmentation, getting somewhat close to the dual extragradient path as is shown in Fig. 8(c). The online version of the averaged perceptron algorithm varied significantly with the order of presentation of examples (up to five points of difference in AER between two orders). To alleviate this, we randomize the order of the points at each pass over the data. Fig. 8(d) shows that a typical run of averaged perceptron does somewhat worse than dual extragradient. The variance of the averaged perceptron performance for

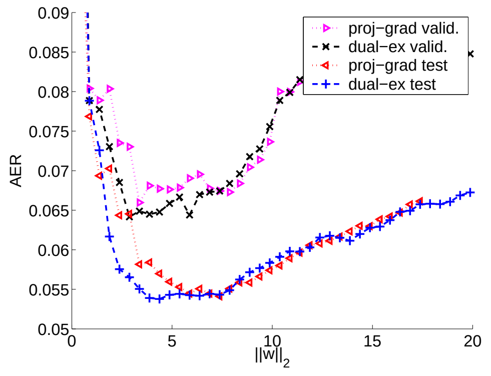
5. The regularization path is obtained by using the commercial optimization software Mosek with the QCQP formulation of Eq. (15). We did not obtain the path in the case of 5000 sentences, as Mosek runs out of memory.



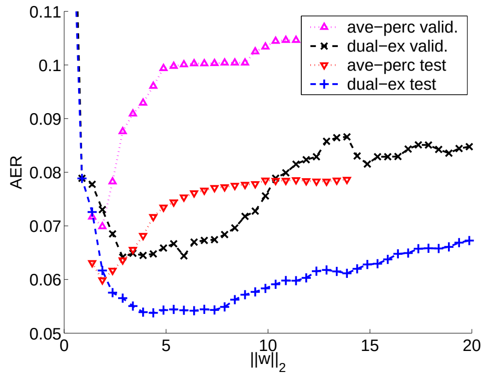
(a)



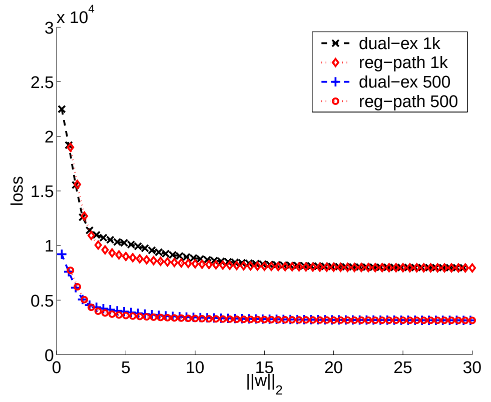
(b)



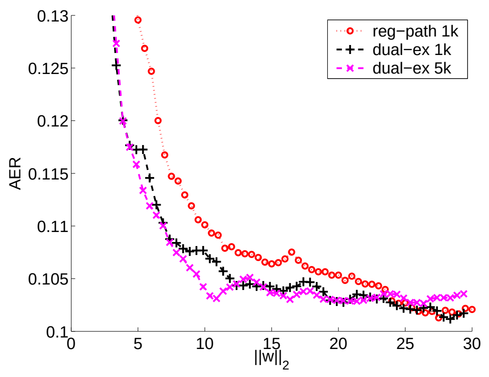
(c)



(d)



(e)



(f)

Figure 8: Word alignment results: (a) Training hinge loss for the three different algorithms and the regularization path on the Gold dataset. (b) AER for the unregularized dual extragradient (`dual-ex`) and the regularization path (`reg-path`) on the 347 Gold sentences (`test`) and the validation set (`valid`) when trained on the 100 Gold sentences; (c) Same setting as in (b), comparing `dual-ex` with the averaged projected-gradient (`proj-grad`); (d) Same setting as in (b), comparing `proj-grad` with the averaged perceptron (`ave-perc`); (e) Training hinge loss for `dual-ex` and `reg-path` on 500 and 1000 GIZA++ labeled sentences. (f) AER for `dual-ex` and `reg-path` tested on the Gold test set and trained on 1000 and 5000 GIZA++ sentences. The graph for 500 sentences is omitted for clarity.

different datasets and learning rate choices was also significantly higher than for dual extragradient, which is more stable numerically. The online version of the averaged perceptron converged very quickly to its minimum AER score; converging in as few as five iterations for the Gold training set. Selecting the best model on the validation set leads to test errors of 5.6% for `dual-ex`, 5.6% for `reg-path`, 5.8% for `proj-grad` and 6.1% for `ave-perc` on the Gold data training set.

The running time for 500 iterations of dual extragradient on a 3.4 Ghz Intel Xeon CPU with 4G of memory was roughly 10 minutes, 30 minutes and 3 hours for 500, 1000 and 5000 sentences, respectively, showing the favorable linear scaling of the algorithm (linear in the number of edges). Note, by way of comparison, that Mosek ran out of memory for more than 1500 training sentences.

The framework we have presented here supports much richer models for word alignment; for example, allowing finer-grained, feature-based fertility control (number of aligned words for each word) as well as inclusion of positive correlations between adjacent edges in alignments. These extensions are developed in Lacoste-Julien et al. (2006).

7. Conclusions

We have presented a general and simple solution strategy for large-scale structured prediction problems. Using a saddle-point formulation of the problem, we exploit the dual extragradient algorithm, a simple gradient-based algorithm for saddle-point problems (Nesterov, 2003). The factoring of the problem into optimization over the feasible parameter space $\mathcal{W}$ and feasible structured output space $\mathcal{Z}$ allows easy integration of complex parameter constraints that arise in estimation of restricted classes of Markov networks and other models.

Key to our approach is the recognition that the projection step in the extragradient algorithm can be solved by network flow algorithms for matchings and min-cuts (and dynamic programming for decomposable models). Network flow algorithms are among the most well-developed algorithms in the field of combinatorial optimization, and yield stable, efficient algorithmic platforms.

One of the key bottlenecks of large learning problems is the memory requirement of the algorithm. We have derived a version of the algorithm that only uses storage proportional to the number of parameters in the model, and is independent of the number of examples. We have exhibited the favorable scaling of this overall approach in two concrete, large-scale learning problems. It is also important to note that the general approach extends and adopts to a much broader class of problems by allowing the use of Bregman projections suitable to particular problem classes.

Acknowledgments

We would like to thank Paul Tseng for kindly answering our questions about the minimum cost quadratic flow implementation. We also wish to acknowledge support from the Defense Advanced Research projects Agency under contract NBCHD030010, and grants from Google and Microsoft Research.

Appendix A. Min-Cut Polytope Projections

Consider projection for a single example i:

$$\begin{aligned}
&\min_{\mathbf{z}} \quad \sum_{j\in\mathcal{V}} \frac{1}{2}(z'_j - z_j)^2 + \sum_{jk\in\mathcal{E}} \frac{1}{2}(z'_{jk} - z_{jk})^2 \\
&\text{s.t.} \quad 0 \le z_j \le 1, \quad \forall j \in \mathcal{V}; \quad z_j - z_k \le z_{jk}, \quad z_k - z_j \le z_{jk}, \quad \forall jk \in \mathcal{E}.
\end{aligned} \tag{21}$$

Let $h_j^+(z_j) = \frac{1}{2}(z'_j - z_j)^2$ if $0 \le z_j$, else ∞. We introduce non-negative Lagrangian variables $\lambda_{jk}, \lambda_{kj}$ for the two constraints for each edge jk and λ_{j0} for the constraint $z_j \le 1$ each node j.

The Lagrangian is given by:

$$\begin{aligned}
L(\mathbf{z},\lambda) &= \sum_{j\in\mathcal{V}} h_j^+(z_j) + \sum_{jk\in\mathcal{E}} \frac{1}{2}(z'_{jk} - z_{jk})^2 - \sum_{j\in\mathcal{V}} (1 - z_j)\lambda_{j0} \\
&\quad - \sum_{jk\in\mathcal{E}} (z_{jk} - z_j + z_k)\lambda_{jk} - \sum_{jk\in\mathcal{E}} (z_{jk} - z_k + z_j)\lambda_{kj}
\end{aligned}$$

Letting $\lambda_{0j} = \lambda_{j0} + \sum_{k:jk\in\mathcal{E}}(\lambda_{jk} - \lambda_{kj}) + \sum_{k:kj\in\mathcal{E}}(\lambda_{jk} - \lambda_{kj})$, note that

$$\sum_{j\in\mathcal{V}} z_j \lambda_{0j} = \sum_{j\in\mathcal{V}} z_j \lambda_{j0} + \sum_{jk\in\mathcal{E}} (z_j - z_k)\lambda_{jk} + \sum_{jk\in\mathcal{E}} (z_k - z_j)\lambda_{kj}.$$

So the Lagrangian becomes:

$$L(\mathbf{z},\lambda) = \sum_{j\in\mathcal{V}} \left[h_j^+(z_j) + z_j\lambda_{0j} - \lambda_{j0}\right] + \sum_{jk\in\mathcal{E}} \left[\frac{1}{2}(z'_{jk} - z_{jk})^2 - z_{jk}(\lambda_{jk} + \lambda_{kj})\right].$$

Now, minimizing $L(\mathbf{z},\lambda)$ with respect to $\mathbf{z}$, we have

$$\min_{\mathbf{z}} L(\mathbf{z},\lambda) = \sum_{jk\in\mathcal{E}} q_{jk}(\lambda_{jk} + \lambda_{kj}) + \sum_{j\in\mathcal{V}} [q_{0j}(\lambda_{0j}) - \lambda_{j0}],$$

where $q_{jk}(\lambda_{jk} + \lambda_{kj}) = \min_{z_{jk}} \left[\frac{1}{2}(z'_{jk} - z_{jk})^2 - z_{jk}(\lambda_{jk} + \lambda_{kj})\right]$ and $q_{0j}(\lambda_{0j}) = \min_{z_j}[h_j^+(z_j) + z_j\lambda_{0j}]$. The minimizing values of $\mathbf{z}$ are:

$$\begin{aligned}
z_j^* &= \arg\min_{z_j} \left[h_j^+(z_j) + z_j\lambda_{0j}\right] = \begin{cases} 0 & \lambda_{0j} \ge z'_j; \\ z'_j - \lambda_{0j} & \lambda_{0j} \le z'_j; \end{cases} \\
z_{jk}^* &= \arg\min_{z_{jk}} \left[\frac{1}{2}(z'_{jk} - z_{jk})^2 - z_{jk}(\lambda_{jk} + \lambda_{kj})\right] = z'_{jk} + \lambda_{jk} + \lambda_{kj}.
\end{aligned}$$

Hence, we have:

$$\begin{aligned}
q_{jk}(\lambda_{jk} + \lambda_{kj}) &= -z'_{jk}(\lambda_{jk} + \lambda_{kj}) - \frac{1}{2}(\lambda_{jk} + \lambda_{kj})^2 \\
q_{0j}(\lambda_{0j}) &= \begin{cases} \frac{1}{2}{z'_j}^2 & \lambda_{0j} \ge z'_j; \\ z'_j\lambda_{0j} - \frac{1}{2}\lambda_{0j}^2 & \lambda_{0j} \le z'_j. \end{cases}
\end{aligned}$$

The dual of the projection problem is thus:

$$\begin{aligned}\max_{\lambda} \quad & \sum_{j\in\mathcal{V}} [q_{0j}(\lambda_{0j}) - \lambda_{j0}] + \sum_{jk\in\mathcal{E}} \left[-z'_{jk}(\lambda_{jk}+\lambda_{kj}) - \frac{1}{2}(\lambda_{jk}+\lambda_{kj})^2\right] \qquad (22)\\ \text{s.t.} \quad & \lambda_{j0} - \lambda_{0j} + \sum_{jk\in\mathcal{E}} (\lambda_{jk}-\lambda_{kj}) = 0, \ \forall j \in \mathcal{V};\\ & \lambda_{jk}, \lambda_{kj} \ge 0, \ \forall jk \in \mathcal{E}; \ \ \lambda_{j0} \ge 0, \ \forall j \in \mathcal{V}.\end{aligned}$$

Interpreting λ_{jk} as flow from node j to node k, and λ_{kj} as flow from k to j and $\lambda_{j0}, \lambda_{0j}$ as flow from and to a special node 0, we can identify the constraints of Eq. (22) as conservation of flow constraints. The last transformation we need is to address the presence of cross-terms $\lambda_{jk}\lambda_{kj}$ in the objective. Note that in the flow conservation constraints, λ_{jk}, λ_{kj} always appear together as $\lambda_{jk} - \lambda_{kj}$. Since we are minimizing $(\lambda_{jk}+\lambda_{kj})^2$ subject to constraints on $\lambda_{jk} - \lambda_{kj}$, at least one of λ_{jk}, λ_{kj} will be zero at the optimum and the cross-terms can be ignored. Note that all λ variables are non-negative except for λ_{0j}'s. Many standard flow packages support this problem form, but we can also transform the problem to have all non-negative flows by introducing extra variables. The final form has a convex quadratic cost for each edge:

$$\begin{aligned}\min_{\lambda} \quad & \sum_{j\in\mathcal{V}} [-q_{0j}(\lambda_{0j}) + \lambda_{j0}] + \sum_{jk\in\mathcal{E}} \left[z'_{jk}\lambda_{jk} + \frac{1}{2}\lambda_{jk}^2\right] + \sum_{jk\in\mathcal{E}} \left[z'_{jk}\lambda_{kj} + \frac{1}{2}\lambda_{kj}^2\right] \qquad (23)\\ \text{s.t.} \quad & \lambda_{j0} - \lambda_{0j} + \sum_{jk\in\mathcal{E}} (\lambda_{jk}-\lambda_{kj}) = 0, \ \forall j \in \mathcal{V};\\ & \lambda_{jk}, \lambda_{kj} \ge 0, \ \forall jk \in \mathcal{E}; \ \ \lambda_{j0} \ge 0, \ \forall j \in \mathcal{V}.\end{aligned}$$

References

Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. Hidden Markov support vector machines. In *International Conference on Machine Learning*. ACM Press, New York, 2003.

Drago Anguelov, Ben Taskar, Vassil Chatalbashev, Daphne Koller, Dinkar Gupta, Geremey Heitz, and Andrew Y. Ng. Discriminative learning of Markov random fields for segmentation of 3D scan data. In *International Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Washington, DC, 2005.

Pierre Baldi, Jianlin Cheng, and Alessandro Vullo. Large-scale prediction of disulphide bond connectivity. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2005.

Peter L. Bartlett, Michael Collins, Ben Taskar, and David McAllester. Exponentiated gradient algorithms for large-margin structured classification. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2005.

Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machince Intelligence*, 24:509–522, 2002.

Dimitri P. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, Belmont, MA, 1998.

Dimitri P. Bertsekas, Lazaros C. Polymenakos, and Paul Tseng. An ε-relaxation method for separable convex cost network flow problems. *SIAM Journal on Optimization*, 7(3):853–870, 1997.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Chandra Chekuri, Sanjeev Khanna, Joseph Naor, and Leonid Zosin. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, Washington, DC, 2001.

Chandra Chekuri, Sanjeev Khanna, Joseph Naor, and Leonid Zosin. A linear programming formulation and approximation algorithms for the metric labeling problem. *SIAM Journal on Discrete Mathematics*, 18(3):606–635, 2005.

Michael Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Philadelphia, PA, 2002.

Richard Durbin, Sean Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis*. Cambridge University Press, UK, 1998.

D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society B*, 51:271–279, 1989.

Francesca Guerriero and Paul Tseng. Implementation and test of auction methods for solving generalized network flow problems with separable convex cost. *Journal of Optimization Theory and Applications*, 115:113–144, 2002.

Trevor Hastie, Saharon Rosset, Robert Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

Bingsheng He and Li-Zhi Liao. Improvements of some projection methods for monotone nonlinear variational inequalities. *Journal of Optimization Theory and Applications*, 112:111–128, 2002.

Mark Jerrum and Alistair Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing*, 22:1087–1116, 1993.

Vladimir Kolmogorov and Martin Wainwright. On the optimality of tree-reweighted max-product message passing. In *Uncertainty in Artificial Intelligence: Proceedings of the Twenty-First Conference*. Morgan Kaufmann, San Mateo, CA, 2005.

Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized using graph cuts? *IEEE Transactions on Pattern Analysis and Machince Intelligence*, 26:147–159, 2004.

G. M. Korpelevich. The extragradient method for finding saddle points and other problems. *Ekonomika i Matematicheskie Metody*, 12:747–756, 1976.

Sanjiv Kumar and Martial Hebert. Discriminative fields for modeling spatial dependencies in natural images. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2004.

Simon Lacoste-Julien, Ben Taskar, Dan Klein, and Michael I. Jordan. Word alignment via quadratic assignment. In *Human Language Technology–North American Association for Computational Linguistics*, New York, NY, 2006.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*. Morgan Kaufmann, San Mateo, CA, 2001.

John Lafferty, Xiaojin Zhu, and Yan Liu. Kernel conditional random fields: Representation and clique selection. In *International Conference on Machine Learning*. ACM Press, New York, 2004.

Yan LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998.

Christopher Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999.

Evgeny Matusov, Richard Zens, and Herman Ney. Symmetric word alignments for statistical machine translation. In *Proceedings of the Twentieth International Conference on Computational Linguistics*, Geneva, Switzerland, 2004.

Rada Mihalcea and Ted Pedersen. An evaluation exercise for word alignment. In *Human Language Technology–North American Association for Computational Linguistics*, Edmonton, Canada, 2003.

Yurii Nesterov. Dual extrapolation and its application for solving variational inequalites and related problems. Technical report, CORE, Catholic University of Louvain, 2003.

Franz Och and Herman Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–52, 2003.

John C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods—Support Vector Learning*, pages 185–208. MIT Press, Cambridge, MA, 1999.

Saharon Rosset. Tracking curved regularized optimization solution paths. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2004.

Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, Berlin, 2003.

Ben Taskar. *Learning Structured Prediction Models: A Large Margin Approach*. PhD thesis, Stanford University, 2004.

Ben Taskar, Vassil Chatalbashev, and Daphne Koller. Learning associative Markov networks. In *International Conference on Machine Learning*. ACM Press, New York, 2004a.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max margin Markov networks. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2004b.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning structured prediction models: a large margin approach. In *International Conference on Machine Learning*. ACM Press, New York, 2005a.

Ben Taskar, Simon Lacoste-Julien, and Dan Klein. A discriminative matching approach to word alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Vancouver, CA, 2005b.

Ben Taskar, Simon Lacoste-Julien, and Michael I. Jordan. Structured prediction via the extragradient method. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2006.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning*. ACM Press, New York, 2004.

Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8: 189–201, 1979.

Martin Wainwright, Tommi Jaakkola, and Alan Willsky. Map estimation via agreement on (hyper)trees: Message-passing and linear programming approaches. In *Proceedings of the Allerton Conference on Communication, Control and Computing*, Allerton, IL, 2002.