Accelerated Training of Max-Margin Markov Networks with Kernels

Xinhua Zhang*

Department of Computing Science, University of Alberta, Edmonton, AB T6G2E8, Canada

Ankan Saha

Department of Computer Science, University of Chicago, Chicago, IL 60637, USA

S.V.N. Vishwanathan

Department of Statistics and Computer Science, Purdue University, West Lafayette, IN 47907, USA

Abstract

Structured output prediction is an important machine learning problem both in theory and practice, and the max-margin Markov network ($\mathsf{M^3N}$) is an effective approach. All state-of-the-art algorithms for optimizing $\mathsf{M^3N}$ objectives take at least $O(1/\epsilon)$ number of iterations to find an ϵ accurate solution. Nesterov [1] broke this barrier by proposing an excessive gap reduction technique (EGR) which converges in $O(1/\sqrt{\epsilon})$ iterations. However, it is restricted to Euclidean projections which consequently requires an intractable amount of computation for each iteration when applied to solve $\mathsf{M^3N}$. In this paper, we show that by extending EGR to Bregman projection, this faster rate of convergence can be retained, and more importantly, the updates can be performed efficiently by exploiting graphical model factorization. Further, we design a kernelized procedure which allows all computations per iteration to be performed at the same cost as the state-of-the-art approaches.

Keywords: Convex optimization, max-margin models, kernel method, graphical models

*Corresponding author

Email addresses: `xinhua2@cs.ualberta.ca` (Xinhua Zhang), `ankans@cs.uchicago.edu` (Ankan Saha), `vishy@stat.purdue.edu` (S.V.N. Vishwanathan)

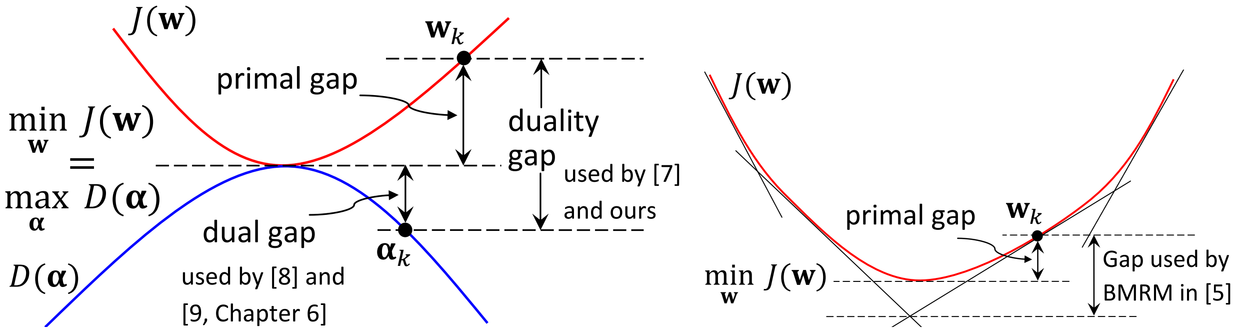
1. Introduction

In the supervised learning setting, one is given a training set of labeled data points and the aim is to learn a function which predicts labels on unseen data points. Sometimes the label space has a rich internal structure which characterizes the combinatorial or recursive inter-dependencies of the application domain. It is widely believed that capturing these dependencies is critical for effectively learning with *structured output*. Examples of such problems include sequence labeling, context free grammar parsing, and word alignment. However, parameter estimation is generally hard even for simple linear models, because the size of the label space is potentially exponentially large (see *e.g.* [2]). Therefore it is crucial to exploit the underlying conditional independence assumptions for the sake of computational tractability. This is often done by defining a graphical model on the output space, and exploiting the underlying graphical model factorization to perform efficient computations.

Research in structured prediction can broadly be categorized into two tracks: Optimizing conditional likelihood in an exponential family results in conditional random fields [CRFs, 3], and a maximum margin approach leads to max-margin Markov networks [M^{3}Ns, 4]. Unsurprisingly, these two approaches share many commonalities: First, they both minimize a regularized risk with a square norm regularizer. Second, they assume that there is a joint feature map $\boldsymbol{\phi}$ which maps $(\mathbf{x}, \mathbf{y})$ to a feature vector in $\mathbb{R}^p$.[1] Third, they assume a label loss $\ell(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i)$ which quantifies the loss of predicting label $\mathbf{y}$ when the correct label of input $\mathbf{x}^i$ is $\mathbf{y}^i$. Finally, they assume that the space of labels $\mathcal{Y}$ is endowed with a graphical model structure and that $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$ and $\ell(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i)$ factorize according to the cliques of this graphical model. The main difference is in the loss function employed. CRFs minimize the L_2-regularized logistic loss:

$$J(\mathbf{w}) = \frac{\lambda}{2} \|\mathbf{w}\|_2^2 + \frac{1}{n} \sum_{i=1}^{n} \log \sum_{\mathbf{y} \in \mathcal{Y}} \exp \left(\ell(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i) - \left\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}^i) - \boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}) \right\rangle \right), \tag{1}$$

[1] We discuss kernels and associated feature maps into a Reproducing Kernel Hilbert Space (RKHS) in Section 4.3.



(a) Primal gap, dual gap, and duality gap

(b) BMRM gap (and similarly for SVM-Struct)

Figure 1: Illustration of stopping criterion monitored by various algorithms; convergence rates are stated with respect to these stopping criterion. $D(\boldsymbol{\alpha})$ is the Lagrange dual of $J(\mathbf{w})$, and $\min_{\mathbf{w}} J(\mathbf{w}) = \max_{\boldsymbol{\alpha}} D(\boldsymbol{\alpha})$. Neither the primal gap nor the dual gap is actually measurable in practice since $\min_{\mathbf{w}} J(\mathbf{w})$ (and $\max_{\boldsymbol{\alpha}} D(\boldsymbol{\alpha})$) is unknown. BMRM (right) therefore uses a measurable upper bound of the primal gap. SVM-Struct monitors constraint violation, which can be also be translated to an upper bound on the primal gap.

where all log in this paper stands for natural basis. In contrast, the M^{3}Ns minimize the L_2-regularized hinge loss

$$J(\mathbf{w}) = \frac{\lambda}{2} \|\mathbf{w}\|_2^2 + \frac{1}{n} \sum_{i=1}^{n} \max_{\mathbf{y} \in \mathcal{Y}} \left\{ \ell(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i) - \left\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}^i) - \boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}) \right\rangle \right\}. \tag{2}$$

A large body of literature exists on efficient algorithms for minimizing the above objective functions. A summary of existing methods, and their convergence rates (iterations needed to find an ϵ accurate solution) can be found in Table 1. The ϵ accuracy of a solution can be measured in many different ways and different algorithms employ different but somewhat related stopping criterion (see Figure 1). Some produce iterates $\mathbf{w}_k$ in the primal space and bound the *primal gap* $J(\mathbf{w}_k) - \min_{\mathbf{w}} J(\mathbf{w})$. Some solve the dual problem $D(\boldsymbol{\alpha})$ with iterates $\boldsymbol{\alpha}_k$ and bound the *dual gap* $\max_{\boldsymbol{\alpha}} D(\boldsymbol{\alpha}) - D(\boldsymbol{\alpha}_k)$. Some bound the *duality gap* $J(\mathbf{w}_k) - D(\boldsymbol{\alpha}_k)$, and still others bound $J(\mathbf{w}_k) - \min_{\mathbf{w}} J_k(\mathbf{w})$ where J_k is a uniform lower bound of J. This must be borne in mind when interpreting the convergence rates in Table 1.

Since (1) is a smooth convex objective, classical methods such as L-BFGS can directly be applied [11]. Specialized solvers also exist. For instance a primal algorithm based on bundle methods was proposed by [5], while a dual algorithm for the same problem was

Optimization algorithm	Primal/dual	Type of gap	Oracle for $\mathsf{M^3N}$	Convergence rate	
				CRF	$\mathsf{M^3N}$
BMRM [5]	primal	primal gap	max	$O\left(\frac{1}{\lambda}\log\frac{1}{\epsilon}\right)$	$O\left(\frac{G^2}{\lambda\epsilon}\right)$
SVM-Struct [6]	primal-dual	constraint violation	max	n/a	$O\left(\frac{G^2}{\lambda\epsilon}\right)$
Extragradient [7]	primal-dual	duality gap	exp	n/a	$O\left(\frac{\log\lvert\mathcal{Y}\rvert}{\epsilon}\right)$
Exponentiated gradient [8]	dual	dual gap	exp	$O\left(\frac{1}{\lambda}\log\frac{1}{\epsilon}\right)$	$O\left(\frac{G^2\log\lvert\mathcal{Y}\rvert}{\lambda\epsilon}\right)$
SMO [9, Chapter 6]	dual	dual gap	max	n/a	psd: $O\left(n\lvert\mathcal{Y}\rvert\frac{1}{\lambda\epsilon}\right)$ pd: $O\left(n\lvert\mathcal{Y}\rvert\log\frac{1}{\epsilon}\right)$
Our algorithm	primal-dual	duality gap	exp	n/a	$O\left(G\sqrt{\frac{\log\lvert\mathcal{Y}\rvert}{\lambda\epsilon}}\right)$

Table 1: Comparison of specialized optimization algorithms for training structured prediction models. Primal-dual methods maintain estimation sequences in both primal and dual spaces. Details of the oracle will be discussed in Section 5. The convergence rate highlights the dependence on both ϵ and some "constants" that are often hidden in the O notation: n, λ, and the size of the label space $|\mathcal{Y}|$. The convergence rate of SMO on $\mathsf{M^3N}$ is derived from [10, Corollary 17], noting the dual problem (26) is so-called pairable. It enjoys linear convergence $O(\log\frac{1}{\epsilon})$ when the dual objective is positive definite (pd), and $O(\frac{1}{\epsilon})$ when it is positive semi-definite (psd). The term G in the convergence rate denotes the maximum L_2 norm of the features vectors $\boldsymbol{\phi}(\mathbf{x}^i,\mathbf{y})$. The convergence rate of Extragradient depends on λ in an indirect way.

proposed by [8]. Both algorithms converge at $O(\frac{1}{\lambda}\log(1/\epsilon))$ rates to an ϵ accurate solution, and, remarkably, their convergence rates are independent of n (the number of data points), and $|\mathcal{Y}|$ (the size of the label space). It is widely believed in optimization (see *e.g.* Section 9.3 of [12]) that unconstrained smooth strongly convex objective functions can be minimized in $O(\log(1/\epsilon))$ iterations, and these specialized optimizers also achieve this rate. Although interior point methods can converge in quadratic rates $\log(\log(1/\epsilon))$ which are even faster than $O(\log(1/\epsilon))$, its cost per step is prohibitively high.

On the other hand, since (2) is a non-smooth convex function, efficient algorithms are harder to come by. SVM-Struct was one of the first specialized algorithms to tackle this problem, and [6] derived an $O(G^2/\lambda\epsilon^2)$ rate of convergence, which was later improved to $O(G^2/\lambda\epsilon)$ by more refined analysis [13]. Here G denotes the maximum L_2 norm of the feature vectors $\boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y})$. In a slightly more general setting, [5] proposed a bundle method for regularized risk minimization (BMRM) which also attained $O(G^2/\lambda\epsilon)$ rate of convergence. At first glance, it looks like the rates of convergence of these algorithms are independent of $|\mathcal{Y}|$. This is somewhat misleading because, although the dependence is not direct, the convergence rates depend on G, which is in turn implicitly related to the size of $\mathcal{Y}$.

Algorithms which optimize (2) in the dual have also been developed. For instance, the algorithm proposed by [8] performs exponentiated gradient descent in the dual and converges at $O\left(\frac{\log|\mathcal{Y}|}{\lambda\epsilon}\right)$ rate. Again, these rates of convergence are not surprising given the well established lower bounds of [14] who show that, in general, non-smooth optimization problems cannot be solved in fewer than $\Omega(1/\epsilon)$ iterations by solvers which treat the objective function as a black box.

In this paper, we present an algorithm that provably converges to an ϵ accurate solution of (2) in $O\left(\sqrt{\frac{\log|\mathcal{Y}|}{\lambda\epsilon}}\right)$ iterations. This does not contradict the lower bound because our algorithm is not a general purpose black box optimizer. In fact, it exploits the special form of the objective function (2). Before launching into the technical details we would like to highlight some important features of our algorithm. First, compared to existing algorithms our convergence rates are better in terms of $|\mathcal{Y}|$, λ, and ϵ. Second, our convergence analysis is tighter in that our rates are with respect to the duality gap. Not only is the duality gap computable, it also upper bounds the primal and dual gaps used by other algorithms. Finally, our cost per iteration is comparable with other algorithms.

To derive our algorithm we extend the recent excessive gap technique of Nesterov [1] to Bregman projections and establish rates of convergence (Section 2). This extension is important because the original gradient based algorithm for strongly convex objectives by Nesterov [1] does not admit graphical model factorizations, which are crucial for efficiency in structured prediction problems. We apply our resulting algorithm to the M^{3}N objective in Section 3. A straightforward implementation requires $O(|\mathcal{Y}|)$ computational cost per iteration, which makes it prohibitively expensive. We show that by exploiting the graphical model structure of $\mathcal{Y}$ the cost per iteration can be reduced to $O(\log |\mathcal{Y}|)$ (Section 4). Finally we contrast our algorithm with existing techniques in Section 5.

2. Excessive Gap Technique with Bregman Projection

The excessive gap technique proposed by Nesterov [1] achieves accelerated rate of convergence only when the Euclidean projection is used. This prevents the algorithm from being applied to train $\mathsf{M}^3\mathsf{N}$ efficiently, and the aim of this section is to extend the approach to Bregman projection. We start by recapping the algorithm.

The following three concepts from convex analysis will be extensively used in the sequel. Define $\overline{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$. For any convex function f, its domain is defined as the set of $\mathbf{x}$ where $f(\mathbf{x})$ is not infinity: $\operatorname{dom} f = \{\mathbf{x} : f(\mathbf{x}) < \infty\}$. Denote its continuous domain as $\operatorname{cont} f = \{\mathbf{x} \in \operatorname{dom} f : f \text{ is continuous at } \mathbf{x}\}$. The dual norm of a norm $\|\cdot\|$ is denoted as $\|\cdot\|_*$.

Definition 1. *A convex function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is strongly convex with respect to a norm $\|\cdot\|$ if there exists a constant $\rho > 0$ such that $f - \frac{\rho}{2}\|\cdot\|^2$ is convex. The maximum of such constants, ρ^* is called the modulus of strong convexity of f, and for brevity we will call f ρ^*-strongly convex.*

Definition 2. *Suppose a function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is differentiable on $Q \subseteq \mathbb{R}^n$. Then f is said to have Lipschitz continuous gradient (l.c.g) with respect to a norm $\|\cdot\|$ if there exists a constant L such that*

$$\|\nabla f(\mathbf{w}) - \nabla f(\mathbf{w}')\|_* \leq L\|\mathbf{w} - \mathbf{w}'\| \qquad \forall\ \mathbf{w}, \mathbf{w}' \in Q. \tag{3}$$

For brevity, we will call f L-l.c.g.

An important consequence of *l.c.g* is to upper bound a convex function by a quadratic.

Lemma 1 ([15, Eq 3.1])**.** *A convex function f is L-l.c.g if, and only if, it is differentiable everywhere in its domain and*

$$f(\mathbf{w}') \leq f(\mathbf{w}) + \langle \nabla f(\mathbf{w}), \mathbf{w}' - \mathbf{w}\rangle + \frac{L}{2}\|\mathbf{w}' - \mathbf{w}\|^2, \qquad \forall\ \mathbf{w}, \mathbf{w}' \in \operatorname{dom} f.$$

Definition 3. *The Fenchel dual of a function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is a function $f^\star : \mathbb{R}^n \to \overline{\mathbb{R}}$ defined by*

$$f^\star(\mathbf{w}^\star) = \sup_{\mathbf{w} \in \mathbb{R}^n} \left\{ \langle \mathbf{w}, \mathbf{w}^\star \rangle - f(\mathbf{w}) \right\}. \tag{4}$$

The definition of Fenchel duality gives a convenient characterization of the gradient of $f^\star$.

Lemma 2 ([16, Corollary X.1.4.4])**.** *If $f^\star$ is differentiable at $\mathbf{w}^\star$, then*

$$\nabla f^\star(\mathbf{w}^\star) = \operatorname*{argsup}_{\mathbf{w} \in \mathbb{R}^n} \left\{ \langle \mathbf{w}, \mathbf{w}^\star \rangle - f(\mathbf{w}) \right\},$$

and the argsup *must be attainable and unique. Thus* argsup *can be replaced by* argmax*.*

Strong convexity and *l.c.g* are related by Fenchel duality according to the following lemma:

Lemma 3 ([16, Theorem 4.2.1 and 4.2.2])**.**

1. *If $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is ρ-strongly convex, then $f^\star$ is finite on $\mathbb{R}^n$ and $f^\star$ is differentiable and $\frac{1}{\rho}$-l.c.g.*
2. *If $f : \mathbb{R}^n \to \mathbb{R}$ is convex on $\mathbb{R}^n$ and L-l.c.g, then $f^\star$ is $\frac{1}{L}$-strongly convex.*

Let Q_1 and Q_2 be subsets of Euclidean spaces and A be a linear map from Q_1 to Q_2. Suppose f and g are convex functions defined on Q_1 and Q_2 respectively. We are interested in the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{w} \in Q_1} \; J(\mathbf{w}) \\ &\text{where } J(\mathbf{w}) := f(\mathbf{w}) + g^\star(A\mathbf{w}) = f(\mathbf{w}) + \max_{\boldsymbol{\alpha} \in Q_2} \left\{ \langle A\mathbf{w}, \boldsymbol{\alpha} \rangle - g(\boldsymbol{\alpha}) \right\}. \end{aligned} \tag{5}$$

We will make the following standard assumptions: a) Q_2 is compact; b) with respect to a certain norm on Q_1, the function f defined on Q_1 is ρ-strongly convex ($\rho > 0$) but not necessarily *l.c.g*, and c) with respect to a certain norm on Q_2 (which can be different from that on Q_1), the function g defined on Q_2 is L_g-*l.c.g* and convex, but not necessarily strongly convex. If we identify $f(\mathbf{w})$ with the regularizer and $g^\star(A\mathbf{w})$ with the loss function, then it can be noted that (5) has the same form as (1) and (2). This observation will be made clear and exploited in Section 3.

The key difficulty in solving (5) arises because $g^\star$ and hence J may potentially be non-smooth. Our aim is to uniformly approximate $J(\mathbf{w})$ with a smooth and strongly convex function. Towards this end let d be a σ strongly convex smooth function ($\sigma > 0$) with the following properties:

$$\min_{\boldsymbol{\alpha} \in Q_2} d(\boldsymbol{\alpha}) = 0, \quad d(\boldsymbol{\alpha}_0) = 0, \text{ and } \mathcal{D} := \max_{\boldsymbol{\alpha} \in Q_2} d(\boldsymbol{\alpha}).$$

In optimization parlance, d is called a prox-function. Let $\mu \in \mathbb{R}$ be an arbitrary positive constant, and

$$(g + \mu d)^\star(\mathbf{w}) = \sup_{\boldsymbol{\alpha} \in Q_2} \left\{ \langle \boldsymbol{\alpha}, \mathbf{w} \rangle - g(\boldsymbol{\alpha}) - \mu\, d(\boldsymbol{\alpha}) \right\}. \tag{6}$$

If $\mathcal{D} < \infty$ then it is easy to see that $(g + \mu\, d)^\star$ is uniformly close to $g^\star$:

$$g^\star(\mathbf{w}) - \mu \mathcal{D} \leq (g + \mu d)^\star(\mathbf{w}) \leq g^\star(\mathbf{w}). \tag{7}$$

we will use $(g + \mu d)^\star$ to define a new objective function

$$\begin{aligned} J_\mu(\mathbf{w}) &:= f(\mathbf{w}) + (g + \mu d)^\star(A\mathbf{w}) \\ &= f(\mathbf{w}) + \max_{\boldsymbol{\alpha} \in Q_2} \left\{ \langle A\mathbf{w}, \boldsymbol{\alpha} \rangle - g(\boldsymbol{\alpha}) - \mu\, d(\boldsymbol{\alpha}) \right\}. \end{aligned} \tag{8}$$

One can write the dual $D(\boldsymbol{\alpha})$ of $J(\mathbf{w})$ by using $A^\top$ (the transpose of A):

$$D(\boldsymbol{\alpha}) := -g(\boldsymbol{\alpha}) - f^\star(-A^\top \boldsymbol{\alpha}) = -g(\boldsymbol{\alpha}) - \max_{\mathbf{w} \in Q_1} \left\{ \langle -A\mathbf{w}, \boldsymbol{\alpha} \rangle - f(\mathbf{w}) \right\}. \tag{9}$$

By Theorem 3.3.5 of [17], if $A \operatorname{dom} f \cap \operatorname{cont} g^\star \neq \emptyset$, then[2]

$$\min_{\mathbf{w} \in Q_1} J(\mathbf{w}) = \max_{\boldsymbol{\alpha} \in Q_2} D(\boldsymbol{\alpha}), \quad \text{and} \quad J(\mathbf{w}) \geq D(\boldsymbol{\alpha}) \quad \forall\ \mathbf{w} \in Q_1, \boldsymbol{\alpha} \in Q_2. \tag{10}$$

[2] Strictly speaking, [17] used inf and sup in (10). Since they are both attainable in $\mathsf{M}^3\mathsf{N}$, we just use min and max for simplicity.

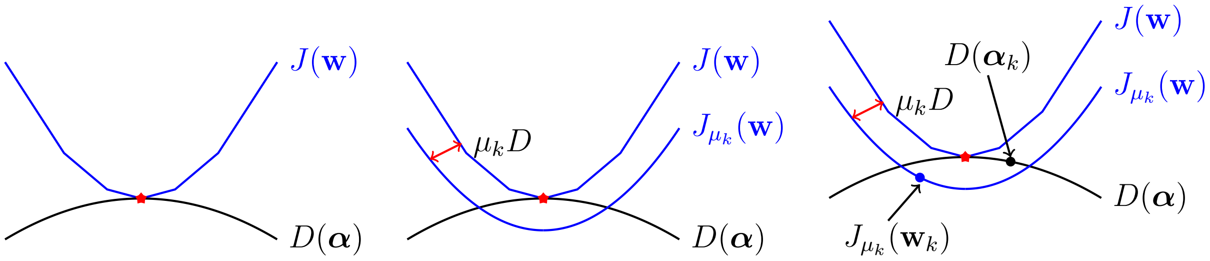


Figure 2: Illustration of excessive gap. When μ_k decreases to 0, the "overlap" of $J_{\mu_k}(\mathbf{w})$ and $D(\boldsymbol{\alpha})$ becomes narrower and narrower. And both $J_{\mu_k}(\mathbf{w}_k)$ and $D(\boldsymbol{\alpha}_k)$ need to lie in this "narrow tube".

As we will see in (24), $\operatorname{dom} f$ and $\operatorname{cont} g^{\star}$ are both the whole Euclidean space for $\mathsf{M}^3\mathsf{N}$. Hence they both contain $\mathbf{0}$ and $A \operatorname{dom} f \cap \operatorname{cont} g^{\star} \neq \emptyset$.

The key idea of excessive gap minimization pioneered by Nesterov [1] is to maintain two estimation sequences $\{\mathbf{w}_k\}$ and $\{\boldsymbol{\alpha}_k\}$, together with a diminishing sequence $\{\mu_k\}$ such that

$$\boxed{J_{\mu_k}(\mathbf{w}_k) \leq D(\boldsymbol{\alpha}_k), \text{ and } \lim_{k \to \infty} \mu_k = 0.} \tag{11}$$

The idea is illustrated in Figure 2. In conjunction with (10) and (7), it is not hard to see that $\{\mathbf{w}_k\}$ and $\{\boldsymbol{\alpha}_k\}$ approach the solution of $\min_{\mathbf{w}} J(\mathbf{w}) = \max_{\boldsymbol{\alpha}} D(\boldsymbol{\alpha})$. Using (7), (8), and (11), we can derive the rate of convergence of this scheme:

$$J(\mathbf{w}_k) - D(\boldsymbol{\alpha}_k) \leq J_{\mu_k}(\mathbf{w}_k) + \mu_k \mathcal{D} - D(\boldsymbol{\alpha}_k) \leq \mu_k \mathcal{D}. \tag{12}$$

In other words, the duality gap is reduced at the same rate at which μ_k approaches 0. All that remains to turn this idea into an implementable algorithm is to answer the following two questions:

1. How to efficiently find initial points $\mathbf{w}_1$, $\boldsymbol{\alpha}_1$ and μ_1 that satisfy (11).
2. Given $\mathbf{w}_k$, $\boldsymbol{\alpha}_k$, and μ_k, how to *efficiently* find $\mathbf{w}_{k+1}$, $\boldsymbol{\alpha}_{k+1}$, and μ_{k+1} which maintain (11).

To achieve the best possible convergence rate it is desirable to anneal μ_k as fast as possible while still allowing $\mathbf{w}_k$ and $\boldsymbol{\alpha}_k$ to be updated efficiently. Nesterov [1] gave a solution based on Euclidean projections, where μ_k decays at $1/k^2$ rate and all updates can be computed in closed form. We now extend his ideas to updates based on Bregman projections[3], which will be the key to our application to structured prediction problems later. Since d is differentiable, we can define a Bregman divergence based on it:[4]

$$\Delta(\bar{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) := d(\bar{\boldsymbol{\alpha}}) - d(\boldsymbol{\alpha}) - \langle \nabla d(\boldsymbol{\alpha}), \bar{\boldsymbol{\alpha}} - \boldsymbol{\alpha} \rangle . \tag{13}$$

Given a point $\boldsymbol{\alpha}$ and a direction $\mathbf{g}$, we can define the Bregman projection as:

$$V(\boldsymbol{\alpha}, \mathbf{g}) := \operatorname*{argmin}_{\bar{\boldsymbol{\alpha}} \in Q_2} \{\Delta(\bar{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) + \langle \mathbf{g}, \bar{\boldsymbol{\alpha}} - \boldsymbol{\alpha} \rangle\} = \operatorname*{argmin}_{\bar{\boldsymbol{\alpha}} \in Q_2} \{d(\bar{\boldsymbol{\alpha}}) - \langle \nabla d(\boldsymbol{\alpha}) - \mathbf{g}, \bar{\boldsymbol{\alpha}} \rangle\}.$$

For notational convenience, we define the following two maps:

$$\mathbf{w}(\boldsymbol{\alpha}) := \operatorname*{argmax}_{\mathbf{w} \in Q_1} \{\langle -A\mathbf{w}, \boldsymbol{\alpha} \rangle - f(\mathbf{w})\} = \nabla f^\star(-A^\top \boldsymbol{\alpha}) \tag{14a}$$

$$\boldsymbol{\alpha}_\mu(\mathbf{w}) := \operatorname*{argmax}_{\boldsymbol{\alpha} \in Q_2} \{\langle A\mathbf{w}, \boldsymbol{\alpha} \rangle - g(\boldsymbol{\alpha}) - \mu d(\boldsymbol{\alpha})\} = \nabla (g + \mu d)^\star (A\mathbf{w}), \tag{14b}$$

where the last step of both (14a) and (14b) are due to Lemma 2. It should be noted that $\boldsymbol{\alpha}_{\mu_k}$ is a different sequence from the sequence of dual variables α_k used in Algorithm 1 and the two should not be confused. Here $\nabla f^\star(-A^\top \boldsymbol{\alpha})$ and $\nabla (g + \mu d)^\star (A\mathbf{w})$ stand for the the gradient of $f^\star$ evaluated at $-A^\top \boldsymbol{\alpha}$, and the gradient of $(g + \mu d)^\star$ evaluated at $A\mathbf{w}$. Since both f and $g + \mu d$ are strongly convex, the above maps are unique and well defined. By Lemma 2,

$$\nabla D(\boldsymbol{\alpha}) = -\nabla g(\boldsymbol{\alpha}) + A\mathbf{w}(\boldsymbol{\alpha}). \tag{15}$$

Since f is assumed to be ρ-strongly convex, it follows from Lemma 3 that $-D(\boldsymbol{\alpha})$ is *l.c.g.* If we denote its *l.c.g* modulus as L, then an easy calculation [*e.g.* Eq. (7.2) 1] shows that

$$L = \frac{1}{\rho} \|A\|^2 + L_g, \qquad \text{and } \|A\| := \max_{\|\mathbf{w}\| = \|\boldsymbol{\alpha}\| = 1} \langle A\mathbf{w}, \boldsymbol{\alpha} \rangle . \tag{16}$$

With this notation in place we now describe our excessive gap minimization method in Algorithm 1. Unrolling the recursive update for μ_{k+1} yields $\mu_{k+1} = \frac{6}{(k+3)(k+2)} \frac{L}{\sigma}$. Plugging

[3] [1] did discuss updates based on Bregman projections, but just for the case where f is convex rather than strongly convex. Here, we show how to improve the convergence rate from $O(1/\epsilon)$ to $O(1/\sqrt{\epsilon})$ when f is strongly convex.

[4] This paper applies ∇ only to differentiable functions; it never refers to subgradient.

Algorithm 1: Excessive gap minimization

```
Input: Function $f$ which is strongly convex, convex function $g$ which is $l.c.g.$
Output: Sequences $\{\mathbf{w}_k\}$, $\{\boldsymbol{\alpha}_k\}$, and $\{\mu_k\}$ that satisfy (11), with $\lim_{k\to\infty} \mu_k = 0$.
Initialize: $\boldsymbol{\alpha}_0 \leftarrow \mathrm{argmin}_{\mathbf{u}\in Q_2} d(\mathbf{u})$, $\mu_1 \leftarrow \frac{L}{\sigma}$, $\mathbf{w}_1 \leftarrow \mathbf{w}(\boldsymbol{\alpha}_0)$, $\boldsymbol{\alpha}_1 \leftarrow V\left(\boldsymbol{\alpha}_0, \frac{-1}{\mu_1}\nabla D(\boldsymbol{\alpha}_0)\right)$.
for $k = 1, 2, \ldots$ do
    $\tau_k \leftarrow \frac{2}{k+3}$.
    $\hat{\boldsymbol{\alpha}} \leftarrow (1-\tau_k)\boldsymbol{\alpha}_k + \tau_k \boldsymbol{\alpha}_{\mu_k}(\mathbf{w}_k)$.
    $\mathbf{w}_{k+1} \leftarrow (1-\tau_k)\mathbf{w}_k + \tau_k \mathbf{w}(\hat{\boldsymbol{\alpha}})$.
    $\tilde{\boldsymbol{\alpha}} \leftarrow V\left(\boldsymbol{\alpha}_{\mu_k}(\mathbf{w}_k), \frac{-\tau_k}{(1-\tau_k)\mu_k}\nabla D(\hat{\boldsymbol{\alpha}})\right)$.
    $\boldsymbol{\alpha}_{k+1} \leftarrow (1-\tau_k)\boldsymbol{\alpha}_k + \tau_k \tilde{\boldsymbol{\alpha}}$.
    $\mu_{k+1} \leftarrow (1-\tau_k)\mu_k$.
```

this into (12) and using (16) immediately yields a $O(1/\sqrt{\epsilon})$ rate of convergence of our algorithm.

Theorem 4 (Rate of convergence for duality gap). *The sequences $\{\mathbf{w}_k\}$ and $\{\boldsymbol{\alpha}_k\}$ in Algorithm 1 satisfy*

$$J(\mathbf{w}_k) - D(\boldsymbol{\alpha}_k) \leq \frac{6L\mathcal{D}}{\sigma(k+1)(k+2)} = \frac{6\mathcal{D}}{\sigma(k+1)(k+2)}\left(\frac{1}{\rho}\|A\|^2 + L_g\right). \tag{17}$$

All that remains is to show that

Theorem 5. *The updates in Algorithm 1 guarantee (11) is satisfied for all $k \geq 1$.*

To prove Theorem 5, we begin with a technical lemma.

Lemma 6. *(Lemma 7.2 of [1]) For any $\boldsymbol{\alpha}$ and $\bar{\boldsymbol{\alpha}}$, we have*

$$D(\boldsymbol{\alpha}) + \langle \nabla D(\boldsymbol{\alpha}), \bar{\boldsymbol{\alpha}} - \boldsymbol{\alpha}\rangle \geq -g(\bar{\boldsymbol{\alpha}}) + \langle A\mathbf{w}(\boldsymbol{\alpha}), \bar{\boldsymbol{\alpha}}\rangle + f(\mathbf{w}(\boldsymbol{\alpha})).$$

Proof. By directly applying (15) and using the fact that a convex function is always above its first order Taylor approximation [16, Corollary 2.1.4], we obtain

$$\begin{aligned} D(\boldsymbol{\alpha}) + \langle \nabla D(\boldsymbol{\alpha}), \bar{\boldsymbol{\alpha}} - \boldsymbol{\alpha}\rangle &= -g(\boldsymbol{\alpha}) + \langle A\mathbf{w}(\boldsymbol{\alpha}), \boldsymbol{\alpha}\rangle + f(\mathbf{w}(\boldsymbol{\alpha})) + \langle -\nabla g(\boldsymbol{\alpha}) + A\mathbf{w}(\boldsymbol{\alpha}), \bar{\boldsymbol{\alpha}} - \boldsymbol{\alpha}\rangle \\ &\geq -g(\bar{\boldsymbol{\alpha}}) + \langle A\mathbf{w}(\boldsymbol{\alpha}), \bar{\boldsymbol{\alpha}}\rangle + f(\mathbf{w}(\boldsymbol{\alpha})). \end{aligned}$$ ■

Furthermore, because d is σ-strongly convex, it follows that

$$\Delta(\bar{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) = d(\bar{\boldsymbol{\alpha}}) - d(\boldsymbol{\alpha}) - \langle \nabla d(\boldsymbol{\alpha}), \bar{\boldsymbol{\alpha}} - \boldsymbol{\alpha} \rangle \geq \frac{\sigma}{2} \left\| \bar{\boldsymbol{\alpha}} - \boldsymbol{\alpha} \right\|^2 . \tag{18}$$

As $\boldsymbol{\alpha}_0$ minimizes d over Q_2, we have

$$\langle \nabla d(\boldsymbol{\alpha}_0), \boldsymbol{\alpha} - \boldsymbol{\alpha}_0 \rangle \geq 0 \qquad \forall\ \boldsymbol{\alpha} \in Q_2. \tag{19}$$

Proof. (of Theorem 5) We first show that the initial $\mathbf{w}_1$ and $\boldsymbol{\alpha}_1$ satisfy the excessive gap condition (11). Since $-D$ is L-$l.c.g$, so by Lemma 1

$$\begin{aligned}
D(\boldsymbol{\alpha}_1) &\geq D(\boldsymbol{\alpha}_0) + \langle \nabla D(\boldsymbol{\alpha}_0), \boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_0 \rangle - \frac{1}{2} L \left\| \boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_0 \right\|^2 \\
\text{(by defn. of } \mu_1 \text{ and (18))} &\geq D(\boldsymbol{\alpha}_0) + \langle \nabla D(\boldsymbol{\alpha}_0), \boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_0 \rangle - \mu_1 \Delta(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_0) \\
\text{(by defn. of } \boldsymbol{\alpha}_1\text{)} &= D(\boldsymbol{\alpha}_0) - \mu_1 \min_{\boldsymbol{\alpha} \in Q_2} \left\{ - \mu_1^{-1} \langle \nabla D(\boldsymbol{\alpha}_0), \boldsymbol{\alpha} - \boldsymbol{\alpha}_0 \rangle + \Delta(\boldsymbol{\alpha}, \boldsymbol{\alpha}_0) \right\} \\
\text{(by (19) and } d(\boldsymbol{\alpha}_0) = 0\text{)} &\geq D(\boldsymbol{\alpha}_0) - \mu_1 \min_{\boldsymbol{\alpha} \in Q_2} \left\{ - \mu_1^{-1} \langle \nabla D(\boldsymbol{\alpha}_0), \boldsymbol{\alpha} - \boldsymbol{\alpha}_0 \rangle + d(\boldsymbol{\alpha}) \right\} \\
&= \max_{\boldsymbol{\alpha} \in Q_2} \left\{ D(\boldsymbol{\alpha}_0) + \langle \nabla D(\boldsymbol{\alpha}_0), \boldsymbol{\alpha} - \boldsymbol{\alpha}_0 \rangle - \mu_1 \, d(\boldsymbol{\alpha}) \right\} \\
\text{(by Lemma 6)} &\geq \max_{\boldsymbol{\alpha} \in Q_2} \left\{ -g(\boldsymbol{\alpha}) + \langle A\mathbf{w}(\boldsymbol{\alpha}_0), \boldsymbol{\alpha} \rangle + f(\mathbf{w}(\boldsymbol{\alpha}_0)) - \mu_1 \, d(\boldsymbol{\alpha}) \right\} \\
&= J_{\mu_1}(\mathbf{w}_1),
\end{aligned}$$

which shows that our initialization indeed satisfies (11).

Second, we prove by induction that the updates in Algorithm 1 maintain (11). We begin with two useful observations. Using $\mu_{k+1} = \frac{6}{(k+3)(k+2)} \frac{L}{\sigma}$ and the definition of τ_k, one can bound

$$\mu_{k+1} = \frac{6}{(k+3)(k+2)} \frac{L}{\sigma} \geq \tau_k^2 \frac{L}{\sigma}. \tag{20}$$

Let $\boldsymbol{\beta} := \boldsymbol{\alpha}_{\mu_k}(\mathbf{w}_k)$. The optimality conditions for (14b) imply

$$\langle \mu_k \nabla d(\boldsymbol{\beta}) - A\mathbf{w}_k + \nabla g(\boldsymbol{\beta}), \boldsymbol{\alpha} - \boldsymbol{\beta} \rangle \geq 0. \tag{21}$$

By using the update equation for $\mathbf{w}_{k+1}$ and the convexity of f, we have

$$\begin{aligned}
J_{\mu_{k+1}}(\mathbf{w}_{k+1}) &= f(\mathbf{w}_{k+1}) + \max_{\boldsymbol{\alpha}\in Q_2}\left\{\langle A\mathbf{w}_{k+1}, \boldsymbol{\alpha}\rangle - g(\boldsymbol{\alpha}) - \mu_{k+1}d(\boldsymbol{\alpha})\right\} \\
&= f((1-\tau_k)\mathbf{w}_k + \tau_k\mathbf{w}(\hat{\boldsymbol{\alpha}})) + \max_{\boldsymbol{\alpha}\in Q_2}\left\{(1-\tau_k)\langle A\mathbf{w}_k, \boldsymbol{\alpha}\rangle + \right. \\
&\qquad \left. \tau_k\langle A\mathbf{w}(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha}\rangle - g(\boldsymbol{\alpha}) - (1-\tau_k)\mu_k d(\boldsymbol{\alpha})\right\} \\
&\le \max_{\boldsymbol{\alpha}\in Q_2}\left\{(1-\tau_k)T_1 + \tau_k T_2\right\},
\end{aligned}$$

where $T_1 = -\mu_k d(\boldsymbol{\alpha}) + \langle A\mathbf{w}_k, \boldsymbol{\alpha}\rangle - g(\boldsymbol{\alpha}) + f(\mathbf{w}_k)$ and $T_2 = -g(\boldsymbol{\alpha}) + \langle A\mathbf{w}(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha}\rangle + f(\mathbf{w}(\hat{\boldsymbol{\alpha}}))$. T_1 can be bounded as follows

$$\begin{aligned}
\text{(by defn. of } \Delta\text{)} \quad T_1 &= -\mu_k\left\{\Delta(\boldsymbol{\alpha},\boldsymbol{\beta}) + d(\boldsymbol{\beta}) + \langle\nabla d(\boldsymbol{\beta}), \boldsymbol{\alpha}-\boldsymbol{\beta}\rangle\right\} \\
&\quad + \langle A\mathbf{w}_k, \boldsymbol{\alpha}\rangle - g(\boldsymbol{\alpha}) + f(\mathbf{w}_k) \\
\text{(by (21))} &\le -\mu_k\Delta(\boldsymbol{\alpha},\boldsymbol{\beta}) - \mu_k d(\boldsymbol{\beta}) + \langle -A\mathbf{w}_k + \nabla g(\boldsymbol{\beta}), \boldsymbol{\alpha}-\boldsymbol{\beta}\rangle \\
&\quad + \langle A\mathbf{w}_k, \boldsymbol{\alpha}\rangle - g(\boldsymbol{\alpha}) + f(\mathbf{w}_k) \\
&= -\mu_k\Delta(\boldsymbol{\alpha},\boldsymbol{\beta}) - \mu_k d(\boldsymbol{\beta}) + \langle A\mathbf{w}_k, \boldsymbol{\beta}\rangle - g(\boldsymbol{\alpha}) + \\
&\quad \langle\nabla g(\boldsymbol{\beta}), \boldsymbol{\alpha}-\boldsymbol{\beta}\rangle + f(\mathbf{w}_k) \\
\text{(by convexity of g)} &\le -\mu_k\Delta(\boldsymbol{\alpha},\boldsymbol{\beta}) - \mu_k d(\boldsymbol{\beta}) + \langle A\mathbf{w}_k, \boldsymbol{\beta}\rangle - g(\boldsymbol{\beta}) + f(\mathbf{w}_k) \\
\text{(by defn. of } \boldsymbol{\beta}\text{)} &= -\mu_k\Delta(\boldsymbol{\alpha},\boldsymbol{\beta}) + J_{\mu_k}(\mathbf{w}_k) \\
\text{(by induction assumption)} &\le -\mu_k\Delta(\boldsymbol{\alpha},\boldsymbol{\beta}) + D(\boldsymbol{\alpha}_k) \\
\text{(by concavity of } D\text{)} &\le -\mu_k\Delta(\boldsymbol{\alpha},\boldsymbol{\beta}) + D(\hat{\boldsymbol{\alpha}}) + \langle\nabla D(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha}_k - \hat{\boldsymbol{\alpha}}\rangle,
\end{aligned}$$

while T_2 can be bounded by using Lemma 7.2 of [1]:

$$T_2 = -g(\boldsymbol{\alpha}) + \langle A\mathbf{w}(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha}\rangle + f(\mathbf{w}(\hat{\boldsymbol{\alpha}})) \le D(\hat{\boldsymbol{\alpha}}) + \langle\nabla D(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha}-\hat{\boldsymbol{\alpha}}\rangle.$$

Putting the upper bounds on T_1 and T_2 together, we obtain the desired result.

$$\begin{aligned}
J_{\mu_{k+1}}(\mathbf{w}_{k+1}) &\leq \max_{\boldsymbol{\alpha}\in Q_2}\{\tau_k\,[D(\hat{\boldsymbol{\alpha}}) + \langle \nabla D(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}\rangle] \\
&\quad + (1-\tau_k)\,[-\mu_k \Delta(\boldsymbol{\alpha}, \boldsymbol{\beta}) + D(\hat{\boldsymbol{\alpha}}) + \langle \nabla D(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha}_k - \hat{\boldsymbol{\alpha}}\rangle]\} \\
&= \max_{\boldsymbol{\alpha}\in Q_2}\{-\mu_{k+1}\Delta(\boldsymbol{\alpha}, \boldsymbol{\beta}) + D(\hat{\boldsymbol{\alpha}}) + \\
&\qquad \langle \nabla D(\hat{\boldsymbol{\alpha}}), (1-\tau_k)\boldsymbol{\alpha}_k + \tau_k \boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}\rangle\} \\
\text{(by defn. of } \hat{\boldsymbol{\alpha}}) &= \max_{\boldsymbol{\alpha}\in Q_2}\{-\mu_{k+1}\Delta(\boldsymbol{\alpha}, \boldsymbol{\beta}) + D(\hat{\boldsymbol{\alpha}}) + \tau_k \langle \nabla D(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha} - \boldsymbol{\beta}\rangle\} \\
&= -\min_{\boldsymbol{\alpha}\in Q_2}\{\mu_{k+1}\Delta(\boldsymbol{\alpha}, \boldsymbol{\beta}) - D(\hat{\boldsymbol{\alpha}}) - \tau_k \langle \nabla D(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha} - \boldsymbol{\beta}\rangle\} \\
\text{(by defn. of } \tilde{\boldsymbol{\alpha}}) &= -\mu_{k+1}\Delta(\tilde{\boldsymbol{\alpha}}, \boldsymbol{\beta}) + D(\hat{\boldsymbol{\alpha}}) + \tau_k \langle \nabla D(\hat{\boldsymbol{\alpha}}), \tilde{\boldsymbol{\alpha}} - \boldsymbol{\beta}\rangle \\
\text{(by (18))} &\leq -\tfrac{\sigma}{2}\mu_{k+1}\left\|\tilde{\boldsymbol{\alpha}} - \boldsymbol{\beta}\right\|^2 + D(\hat{\boldsymbol{\alpha}}) + \tau_k \langle \nabla D(\hat{\boldsymbol{\alpha}}), \tilde{\boldsymbol{\alpha}} - \boldsymbol{\beta}\rangle \\
\text{(by (20))} &\leq -\tfrac{1}{2}\tau_k^2 L\left\|\tilde{\boldsymbol{\alpha}} - \boldsymbol{\beta}\right\|^2 + D(\hat{\boldsymbol{\alpha}}) + \tau_k \langle \nabla D(\hat{\boldsymbol{\alpha}}), \tilde{\boldsymbol{\alpha}} - \boldsymbol{\beta}\rangle \\
\text{(by defn. of } \boldsymbol{\alpha}_{k+1}) &= -\tfrac{1}{2}L\left\|\boldsymbol{\alpha}_{k+1} - \hat{\boldsymbol{\alpha}}\right\|^2 + D(\hat{\boldsymbol{\alpha}}) + \langle \nabla D(\hat{\boldsymbol{\alpha}}), \boldsymbol{\alpha}_{k+1} - \hat{\boldsymbol{\alpha}}\rangle \\
\text{(by } L\text{-}l.c.g \text{ of } -D \text{ and Lemma 1)} &\leq D(\boldsymbol{\alpha}_{k+1}).
\end{aligned}$$

■

When stated in terms of the dual gap (as opposed to the duality gap) our convergence results can be strengthened slightly.

Corollary 7 (Rate of convergence for dual gap)**.** *The sequence* $\{\boldsymbol{\alpha}_k\}$ *in Algorithm 1 satisfy*

$$\max_{\boldsymbol{\alpha}\in Q_2} D(\boldsymbol{\alpha}) - D(\boldsymbol{\alpha}_k) \leq \frac{6\;L\,d(\boldsymbol{\alpha}^*)}{\sigma(k+1)(k+2)} = \frac{6\;d(\boldsymbol{\alpha}^*)}{\sigma(k+1)(k+2)}\left(\frac{\|A\|_{1,2}^2}{\rho} + L_g\right), \tag{22}$$

where $\boldsymbol{\alpha}^* := \operatorname{argmax}_{\boldsymbol{\alpha}\in Q_2} D(\boldsymbol{\alpha})$*. Note* $d(\boldsymbol{\alpha}^*)$ *is tighter than the* D *in* (17)*.*

Proof.

$$\begin{aligned}
D(\boldsymbol{\alpha}_{k+1}) &\geq J_{\mu_{k+1}}(\mathbf{w}_{k+1}) = f(\mathbf{w}_{k+1}) + \max_{\boldsymbol{\alpha}}\{\langle A\mathbf{w}_{k+1}, \boldsymbol{\alpha}\rangle - g(\boldsymbol{\alpha}) - \mu_{k+1}d(\boldsymbol{\alpha})\} \\
&\geq f(\mathbf{w}_{k+1}) + \langle A\mathbf{w}_{k+1}, \boldsymbol{\alpha}^*\rangle - g(\boldsymbol{\alpha}^*) - \mu_{k+1}d(\boldsymbol{\alpha}^*) \\
&\geq -g(\boldsymbol{\alpha}^*) + \min_{\mathbf{w}}\{f(\mathbf{w}) + \langle A\mathbf{w}, \boldsymbol{\alpha}^*\rangle\} - \mu_{k+1}d(\boldsymbol{\alpha}^*) \\
&= D(\boldsymbol{\alpha}^*) - \mu_{k+1}d(\boldsymbol{\alpha}^*).
\end{aligned}$$

■

3. Training Max-Margin Markov Networks

In the max-margin Markov network ($\mathsf{M^3N}$) setting [4], we are given n labeled data points $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^n$, where $\mathbf{x}^i$ are drawn from some space $\mathcal{X}$ and $\mathbf{y}^i$ belong to some space $\mathcal{Y}$. We assume that there is a feature map $\boldsymbol{\phi}$ which maps $(\mathbf{x}, \mathbf{y})$ to a feature vector in $\mathbb{R}^p$. Furthermore, for each $\mathbf{x}^i$, there is a label loss $\ell_{\mathbf{y}}^i := \ell(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i)$ which quantifies the loss of predicting label $\mathbf{y}$ when the correct label is $\mathbf{y}^i$. Given this setup, the objective function minimized by $\mathsf{M^3N}$s can be written as

$$J(\mathbf{w}) = \frac{\lambda}{2} \|\mathbf{w}\|_2^2 + \frac{1}{n} \sum_{i=1}^n \max_{\mathbf{y} \in \mathcal{Y}} \left\{ \ell_{\mathbf{y}}^i - \langle \mathbf{w}, \boldsymbol{\psi}_{\mathbf{y}}^i \rangle \right\}, \tag{23}$$

where $\|\mathbf{w}\|_2 = (\sum_j w_j^2)^{1/2}$ is the L_2 norm and we used the shorthand $\boldsymbol{\psi}_{\mathbf{y}}^i := \boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}^i) - \boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y})$. To write (23) in the form of (5), let $Q_1 = \mathbb{R}^p$, A be a $(n\,|\mathcal{Y}|)$-by-p matrix whose $(i, \mathbf{y})$-th row is $(-\boldsymbol{\psi}_{\mathbf{y}}^i)^\top$,

$$f(\mathbf{w}) = \frac{\lambda}{2} \|\mathbf{w}\|_2^2, \quad \text{and} \quad g^\star(\mathbf{u}) = \frac{1}{n} \sum_i \max_{\mathbf{y}} \left\{ \ell_{\mathbf{y}}^i + u_{\mathbf{y}}^i \right\}. \tag{24}$$

Now, g can be verified to be:

$$g(\boldsymbol{\alpha}) = -\sum_i \sum_{\mathbf{y}} \ell_{\mathbf{y}}^i \alpha_{\mathbf{y}}^i \qquad \text{if } \alpha_{\mathbf{y}}^i \geq 0, \text{ and } \sum_{\mathbf{y}} \alpha_{\mathbf{y}}^i = \frac{1}{n}, \ \forall\, i \tag{25}$$

and ∞ otherwise. The domain of g is $Q_2 = \mathcal{S}^n := \left\{ \boldsymbol{\alpha} \in [0,1]^{n|\mathcal{Y}|} : \sum_{\mathbf{y}} \alpha_{\mathbf{y}}^i = \frac{1}{n}, \forall i \right\}$, which is convex and compact. Using the L_2 norm on Q_1, f is clearly λ-strongly convex. Similarly, if we use the L_1 norm on Q_2 (*i.e.*, $\|\boldsymbol{\alpha}\|_1 = \sum_i \sum_{\mathbf{y}} |\alpha_{\mathbf{y}}^i|$), then g is 0-*l.c.g.* By noting that $f^\star(-A^\top \boldsymbol{\alpha}) = \frac{1}{2\lambda} \boldsymbol{\alpha}^\top A A^\top \boldsymbol{\alpha}$, one can write the dual form $D(\boldsymbol{\alpha}) : \mathcal{S}^n \mapsto \mathbb{R}$ of $J(\mathbf{w})$ as

$$D(\boldsymbol{\alpha}) = -g(\boldsymbol{\alpha}) - f^\star(-A^\top \boldsymbol{\alpha}) = -\frac{1}{2\lambda} \boldsymbol{\alpha}^\top A A^\top \boldsymbol{\alpha} + \sum_i \sum_{\mathbf{y}} \ell_{\mathbf{y}}^i \alpha_{\mathbf{y}}^i, \ \ \boldsymbol{\alpha} \in \mathcal{S}^n. \tag{26}$$

3.1. Rates of Convergence

A natural prox-function to use in our setting is the relative entropy with respect to the uniform distribution, which is defined as:

$$d(\boldsymbol{\alpha}) = \sum_{i=1}^n \sum_{\mathbf{y}} \alpha_{\mathbf{y}}^i \log \alpha_{\mathbf{y}}^i + \log n + \log |\mathcal{Y}|. \tag{27}$$

The relative entropy is 1-strongly convex in $\mathcal{S}^n$ with respect to the L_1 norm [e.g., 18, Proposition 5.1]. Furthermore, $d(\boldsymbol{\alpha}) \leq D = \log|\mathcal{Y}|$ for $\boldsymbol{\alpha} \in \mathcal{S}^n$, and the norm of A can be computed via

$$\|A\| = \max_{\mathbf{w}\in\mathbb{R}^p, \mathbf{u}\in\mathbb{R}^{n|\mathcal{Y}|}} \left\{ \langle A\mathbf{w}, \mathbf{u}\rangle : \sum_{j=1}^{p} w_j^2 = 1, \sum_{i=1}^{n}\sum_{\mathbf{y}\in\mathcal{Y}} \left|u_{\mathbf{y}}^i\right| = 1\right\} = \max_{i,\mathbf{y}} \left\|\boldsymbol{\psi}_{\mathbf{y}}^i\right\|_2,$$

where $\left\|\boldsymbol{\psi}_{\mathbf{y}}^i\right\|_2$ is the Euclidean norm of $\boldsymbol{\psi}_{\mathbf{y}}^i$. Since f is λ-strongly convex and $L_g = 0$, plugging this expression of $\|A\|$ into (17) and (22), we obtain the following rates of convergence for our algorithm:

$$\begin{aligned} J(\mathbf{w}_k) - D(\boldsymbol{\alpha}_k) &\leq \frac{6\log|\mathcal{Y}|}{(k+1)(k+2)} \frac{\max_{i,\mathbf{y}} \left\|\boldsymbol{\psi}_{\mathbf{y}}^i\right\|_2^2}{\lambda} \\ \text{and} \quad \max_{\boldsymbol{\alpha}\in Q_2} D(\boldsymbol{\alpha}) - D(\boldsymbol{\alpha}_k) &\leq \frac{6\mathrm{KL}(\boldsymbol{\alpha}^*||\boldsymbol{\alpha}_0)}{(k+1)(k+2)} \frac{\max_{i,\mathbf{y}} \left\|\boldsymbol{\psi}_{\mathbf{y}}^i\right\|_2^2}{\lambda}, \end{aligned}$$

where $\mathrm{KL}(\boldsymbol{\alpha}^*||\boldsymbol{\alpha}_0)$ denotes the KL divergence between $\boldsymbol{\alpha}^*$ and the uniform distribution $\boldsymbol{\alpha}_0$. Recall that for distributions $\mathbf{p}$ and $\mathbf{q}$ the KL divergence is defined as $\mathrm{KL}(\mathbf{p}||\mathbf{q}) = \sum_i p_i \ln \frac{p_i}{q_i}$.

Therefore to reduce the duality gap and dual gap below ϵ, it suffices to take

$$2 + \max_{i,\mathbf{y}} \left\|\boldsymbol{\psi}_{\mathbf{y}}^i\right\|_2 \sqrt{\frac{6\log|\mathcal{Y}|}{\lambda\epsilon}} \quad \text{and} \quad \max_{i,\mathbf{y}} \left\|\boldsymbol{\psi}_{\mathbf{y}}^i\right\|_2 \sqrt{\frac{6\mathrm{KL}(\boldsymbol{\alpha}^*||\boldsymbol{\alpha}_0)}{\lambda\epsilon}} \tag{28}$$

steps respectively.

3.2. Computing the Approximation $J_\mu(\mathbf{w})$ and Connection to CRFs

In this section we show how to compute $J_\mu(\mathbf{w})$. Towards this end, we first compute $(g + \mu d)^\star(\mathbf{u})$.

Lemma 8. *The Fenchel dual of $(g + \mu d)$ is given by*

$$(g+\mu d)^\star(\mathbf{u}) = \frac{\mu}{n}\sum_{i=1}^{n} \log \sum_{\mathbf{y}\in\mathcal{Y}} \exp\left(\frac{u_{\mathbf{y}}^i + \ell_{\mathbf{y}}^i}{\mu}\right) - \mu\log|\mathcal{Y}|, \tag{29}$$

and the $(i, \mathbf{y})$-th element of its gradient can be written as

$$(\nabla(g+\mu d)^\star(\mathbf{u}))_{\mathbf{y}}^i = \frac{1}{n}\exp\left(\frac{u_{\mathbf{y}}^i + \ell_{\mathbf{y}}^i}{\mu}\right) \Big/ \sum_{\mathbf{y}'} \exp\left(\frac{u_{\mathbf{y}'}^i + \ell_{\mathbf{y}'}^i}{\mu}\right). \tag{30}$$

Proof. Using (25) and (27) we can write $(g+\mu d)^\star(\mathbf{u})$ as

$$\begin{aligned}
&\sup_{\boldsymbol{\alpha}\in\mathcal{S}^n}\{\langle\mathbf{u},\boldsymbol{\alpha}\rangle - g(\boldsymbol{\alpha}) - \mu d(\boldsymbol{\alpha})\} \\
&= \sup_{\boldsymbol{\alpha}\in\mathcal{S}^n}\sum_i\sum_{\mathbf{y}} u^i_{\mathbf{y}}\alpha^i_{\mathbf{y}} + \sum_i\sum_{\mathbf{y}} \ell^i_{\mathbf{y}}\alpha^i_{\mathbf{y}} - \mu\sum_i\sum_{\mathbf{y}}\alpha^i_{\mathbf{y}}\log\alpha^i_{\mathbf{y}} - \mu\log n - \mu\log|\mathcal{Y}| \\
&= \sup_{\boldsymbol{\alpha}\in\mathcal{S}^n}\sum_i\sum_{\mathbf{y}}(u^i_{\mathbf{y}} + \ell^i_{\mathbf{y}} - \mu\log\alpha^i_{\mathbf{y}})\alpha^i_{\mathbf{y}} - \mu\log n - \mu\log|\mathcal{Y}|
\end{aligned}$$

By introducing non-negative Lagrange multipliers σ_i we can write the partial Lagrangian of the above maximization problem:

$$L(\boldsymbol{\alpha},\sigma) = \sup_{\boldsymbol{\alpha}\in\mathcal{S}^n}\left\{\sum_i\sum_{\mathbf{y}}(u^i_{\mathbf{y}} + \ell^i_{\mathbf{y}} - \mu\log\alpha^i_{\mathbf{y}})\alpha^i_{\mathbf{y}} - \mu\log n - \mu\log|\mathcal{Y}| - \sum_i\sigma_i\left(\sum_{\mathbf{y}}\alpha^i_{\mathbf{y}} - \frac{1}{n}\right)\right\}.$$

Taking partial derivative with respect $\alpha^i_{\mathbf{y}}$ and setting it to 0, we get

$$u^i_{\mathbf{y}} + \ell^i_{\mathbf{y}} - \mu\log\alpha^i_{\mathbf{y}} - \mu - \sigma_i = 0.$$

Therefore

$$\alpha^i_{\mathbf{y}} = \frac{\exp\left(\frac{u^i_{\mathbf{y}}+\ell^i_{\mathbf{y}}}{\mu}\right)}{nZ_i}, \qquad \text{where } Z_i := \sum_{\mathbf{y}}\exp\left(\frac{u^i_{\mathbf{y}}+\ell^i_{\mathbf{y}}}{\mu}\right).$$

Plugging this back to the Lagrangian, we can eliminate both $\boldsymbol{\alpha}$ and σ_i and write out the solution of the optimization problem in closed form

$$\begin{aligned}
&\sum_{i,\mathbf{y}}(\mu\log Z_i + \mu\log n)\alpha^i_{\mathbf{y}} - \mu\log n - \mu\log|\mathcal{Y}| \\
&= \frac{\mu}{n}\sum_{i=1}^n\log\sum_{\mathbf{y}\in\mathcal{Y}}\exp\left(\frac{u^i_{\mathbf{y}}+\ell^i_{\mathbf{y}}}{\mu}\right) - \mu\log|\mathcal{Y}|.
\end{aligned}$$ ■

Using the above lemma, plugging in the definition of A and $\boldsymbol{\psi}^i_{\mathbf{y}}$, and assuming that $\ell^i_{\mathbf{y}^i} = 0$, we get

$$\begin{aligned}
J_\mu(\mathbf{w}) &= f(\mathbf{w}) + (g+\mu d)^\star(A\mathbf{w}) \\
&= \frac{\lambda}{2}\|\mathbf{w}\|_2^2 - \frac{\mu}{n}\sum_{i=1}^n\log p(\mathbf{y}^i|\mathbf{x}^i;\mathbf{w}) - \mu\log|\mathcal{Y}|, \qquad (31)
\end{aligned}$$

$$\text{where} \qquad p(\mathbf{y}|\mathbf{x}^i;\mathbf{w}) \propto \exp\left(\frac{\ell^i_{\mathbf{y}} + \langle\mathbf{w},\boldsymbol{\phi}(\mathbf{x}^i,\mathbf{y})\rangle}{\mu}\right).$$

This interpretation clearly shows that the approximation $J_\mu(\mathbf{w})$ essentially converts the maximum margin estimation problem (2) into a CRF estimation problem (1). Here μ determines the quality of the approximation; when $\mu \to 0$, $p(\mathbf{y}|\mathbf{x}^i; \mathbf{w})$ tends to the delta distribution with the probability mass concentrated on $\operatorname{argmax}_{\mathbf{y}} \ell^i_{\mathbf{y}} + \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}) \rangle$. Besides, the loss $\ell^i_{\mathbf{y}}$ rescales the distribution.

Given the above interpretation, it is tempting to argue that every non-smooth problem can be solved by computing a smooth approximation $J_\mu(\mathbf{w})$, and applying a standard smooth convex optimizer to minimize $J_\mu(\mathbf{w})$. Unfortunately, this approach is fraught with problems. In order to get a close enough approximation of $J(\mathbf{w})$ the μ needs to be set to a very small number which makes $J_\mu(\mathbf{w})$ ill-conditioned and leads to numerical issues in the optimizer. The excessive gap technique adaptively changes the μ in each iteration in order to avoid these problems.

4. Efficient Computation by Clique Decomposition

In the structured large margin setting, the number of labels $|\mathcal{Y}|$ could potentially be exponentially large. For example, if a sequence has l nodes and each node has two states, then $|\mathcal{Y}| = 2^l$. A naive implementation of the excessive gap reduction algorithm described in the previous section requires maintaining and updating $O(|\mathcal{Y}|)$ coefficients at every iteration, which is prohibitively expensive. With a view to reducing the computational complexity, and also to take into account the inherent conditional independence properties of the output space, it is customary to assume that $\mathcal{Y}$ is endowed with a graphical model structure; we refer the reader to [2] for an in-depth treatment of this issue. For our purposes it suffices to assume that $\ell(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i)$ and $\boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y})$ decompose according to the cliques[5] of an undirected graphical model, and hence can be written (with some abuse of notation) as

$$\ell^i_{\mathbf{y}} = \ell(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i) = \sum_{c \in \mathcal{C}} \ell(y_c, y^i_c; \mathbf{x}^i) = \sum_{c \in \mathcal{C}} \ell^i_{y_c},$$

$$\boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}) = \bigoplus_{c \in \mathcal{C}} \boldsymbol{\phi}(\mathbf{x}^i, y_c), \quad \text{and} \quad \boldsymbol{\psi}^i_{\mathbf{y}} = \bigoplus_{c \in \mathcal{C}} \boldsymbol{\psi}^i_{y_c}. \tag{32}$$

[5] Any fully connected subgraph of a graph is called a clique.

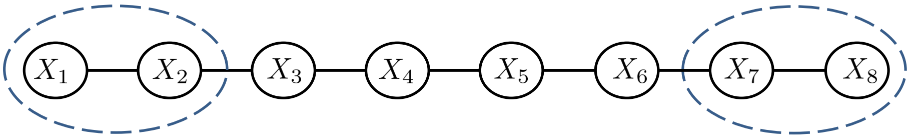


Figure 3: Graphical model of a linear chain. Circles indicate cliques.

Here $\mathcal{C}$ denotes the set of all cliques of the graphical model and $\oplus$ denotes vector concatenation. More explicitly, $\boldsymbol{\psi}^i_{\mathbf{y}}$ is the vector on the graphical model obtained by accumulating the vector $\boldsymbol{\psi}^i_{y_c}$ on all the cliques c of the graph.

Let $h_c(y_c)$ be an arbitrary real valued function on the value of $\mathbf{y}$ restricted to clique c. Graphical models define a distribution $p(\mathbf{y})$ on $\mathbf{y} \in \mathcal{Y}$ whose density takes the following factorized form:

$$p(\mathbf{y}) \propto q(\mathbf{y}) = \prod_{c \in \mathcal{C}} \exp\left(h_c(y_c)\right). \tag{33}$$

The key advantage of a graphical model is that the marginals on the cliques can be efficiently computed:

$$m_{y_c} := \sum_{\mathbf{z}:\mathbf{z}|_c = y_c} q(\mathbf{z}) = \sum_{\mathbf{z}:\mathbf{z}|_c = y_c} \prod_{c' \in \mathcal{C}} \exp\left(h_{c'}(z_{c'})\right).$$

where the summation is over all the configurations $\mathbf{z}$ in $\mathcal{Y}$ whose restriction on the clique c equals y_c. Although $\mathcal{Y}$ can be exponentially large, efficient dynamic programming algorithms exist that exploit the factorized form (33), *e.g.* belief propagation [19]. The computational cost is $O(s^\omega N)$ where s is the number of states of each node, ω is the maximum size of the cliques, and N is the number of cliques. For example, a linear chain as shown in Figure 3 has $\omega = 2$ and the cliques are just edges between consecutive nodes. When ω is large, approximate algorithms also exist [20, 21, 22]. In the sequel we will assume that our graphical models are tractable, *i.e.*, ω is low. The key technique that keeps our algorithm tractable is to reformulate all updates in terms of the marginal distribution on the cliques, which is similar in vein to the exponentiated gradient algorithm [8].

4.1. Basics

At each iteration of Algorithm 1, we need to compute four quantities: $\mathbf{w}(\boldsymbol{\alpha})$, $\nabla D(\boldsymbol{\alpha})$, $\boldsymbol{\alpha}_\mu(\mathbf{w})$, and $V(\boldsymbol{\alpha}, \mathbf{g})$. Below we rewrite them by taking into account the factorization (32), and postpone to Section 4.2 the discussion on how to compute them efficiently. Since $\alpha^i_{\mathbf{y}} \geq 0$ and $\sum_{\mathbf{y}} \alpha^i_{\mathbf{y}} = \frac{1}{n}$, the $\left\{\alpha^i_{\mathbf{y}} : \mathbf{y} \in \mathcal{Y}\right\}$ form an unnormalized distribution, and we denote its (unnormalized) marginal distribution on clique c by

$$\alpha^i_{y_c} := \sum\nolimits_{\mathbf{z}:\mathbf{z}|_c = y_c} \alpha^i_{\mathbf{z}}. \tag{34}$$

The feature expectations on the cliques with respect to the unnormalized distributions $\boldsymbol{\alpha}$ are important:

$$\mathbb{F}\left[\boldsymbol{\psi}^i_c; \boldsymbol{\alpha}\right] := \sum_{y_c} \alpha^i_{y_c} \boldsymbol{\psi}^i_{y_c}, \quad \text{and} \quad \mathbb{F}[\boldsymbol{\psi}_c; \boldsymbol{\alpha}] := \sum_i \mathbb{F}\left[\boldsymbol{\psi}^i_c; \boldsymbol{\alpha}\right]. \tag{35}$$

Clearly, if for all i the marginals of $\boldsymbol{\alpha}$ on the cliques (*i.e.*, $\left\{\alpha^i_{y_c} : i, c, y_c\right\}$ in (34)) are available, then these two expectations can be computed efficiently.

- $\mathbf{w}(\boldsymbol{\alpha})$**:** As a consequence of (32) we can write $\boldsymbol{\psi}^i_{\mathbf{y}} = \bigoplus_{c \in \mathcal{C}} \boldsymbol{\psi}^i_{y_c}$. Plugging this into (14a) and recalling that $\nabla f^\star(-A^\top \boldsymbol{\alpha}) = \frac{-1}{\lambda} A^\top \boldsymbol{\alpha}$ yields the following expression for $\mathbf{w}(\boldsymbol{\alpha}) = \frac{-1}{\lambda} A^\top \boldsymbol{\alpha}$:

$$\begin{aligned}\mathbf{w}(\boldsymbol{\alpha}) &= \frac{1}{\lambda} \sum_i \sum_{\mathbf{y}} \alpha^i_{\mathbf{y}} \boldsymbol{\psi}^i_{\mathbf{y}} = \frac{1}{\lambda} \sum_i \sum_{\mathbf{y}} \alpha^i_{\mathbf{y}} \left(\bigoplus_{c \in \mathcal{C}} \boldsymbol{\psi}^i_{y_c} \right) \\ &= \frac{1}{\lambda} \bigoplus_{c \in \mathcal{C}} \left(\sum_i \mathbb{F}\left[\boldsymbol{\psi}^i_c; \boldsymbol{\alpha}\right] \right) = \frac{1}{\lambda} \bigoplus_{c \in \mathcal{C}} \mathbb{F}[\boldsymbol{\psi}_c; \boldsymbol{\alpha}].\end{aligned} \tag{36}$$

- $\nabla D(\boldsymbol{\alpha})$**:** Using (26) and the definition of $\mathbf{w}(\boldsymbol{\alpha})$, the $(i, \mathbf{y})$-th element of $\nabla D(\boldsymbol{\alpha})$ can be written as

$$\begin{aligned}(\nabla D(\boldsymbol{\alpha}))^i_{\mathbf{y}} &= \ell^i_{\mathbf{y}} - \frac{1}{\lambda} \left(AA^\top \boldsymbol{\alpha}\right)^i_{\mathbf{y}} = \ell^i_{\mathbf{y}} - \left\langle \boldsymbol{\psi}^i_{\mathbf{y}}, \mathbf{w}(\boldsymbol{\alpha}) \right\rangle \\ &= \sum_c \left(\ell^i_{y_c} - \frac{1}{\lambda} \left\langle \boldsymbol{\psi}^i_{y_c}, \mathbb{F}[\boldsymbol{\psi}_c; \boldsymbol{\alpha}] \right\rangle \right).\end{aligned} \tag{37}$$

- $\boldsymbol{\alpha}_\mu(\mathbf{w})$**:** Using (14b) and (30), the $(i, \mathbf{y})$-th element of $\boldsymbol{\alpha}_\mu(\mathbf{w})$ given by $(\nabla(g+\mu d)^\star(A\mathbf{w}))^i_{\mathbf{y}}$ can be written as

$$
\begin{aligned}
(\boldsymbol{\alpha}_\mu(\mathbf{w}))^i_{\mathbf{y}} &= \frac{1}{n} \frac{\exp\left(\mu^{-1}\left(\ell^i_{\mathbf{y}} - \left\langle \boldsymbol{\psi}^i_{\mathbf{y}}, \mathbf{w}\right\rangle\right)\right)}{\sum_{\mathbf{y}'} \exp\left(\mu^{-1}\left(\ell^i_{\mathbf{y}'} - \left\langle \boldsymbol{\psi}^i_{\mathbf{y}'}, \mathbf{w}\right\rangle\right)\right)} \\
&= \frac{1}{n} \frac{\prod_c \exp\left(\mu^{-1}\left(\ell^i_{y_c} - \left\langle \boldsymbol{\psi}^i_{y_c}, \mathbf{w}_c\right\rangle\right)\right)}{\sum_{\mathbf{y}'} \prod_c \exp\left(\mu^{-1}\left(\ell^i_{y'_c} - \left\langle \boldsymbol{\psi}^i_{y'_c}, \mathbf{w}_c\right\rangle\right)\right)}.
\end{aligned} \tag{38}
$$

- $V(\boldsymbol{\alpha}, \mathbf{g})$**:** Since the prox-function d is the relative entropy, by using the same derivation as in the proof of Lemma 8, the $(i, \mathbf{y})$-th element of $V(\boldsymbol{\alpha}, \mathbf{g})$ is

$$
(V(\boldsymbol{\alpha}, \mathbf{g}))^i_{\mathbf{y}} = \frac{1}{n} \frac{\alpha^i_{\mathbf{y}} \exp(-g^i_{\mathbf{y}})}{\sum_{\mathbf{y}'} \alpha^i_{\mathbf{y}'} \exp(-g^i_{\mathbf{y}'})}. \tag{39}
$$

4.2. Efficient Computation

We now show how the algorithm can be made efficient by taking into account (32). As an example, we will give the detailed computational complexity for linear chains as shown in Figure 3, where we assume that there are l nodes and the clique-wise features are all p dimensional (except for the case of kernels in Section 4.3). Key to our efficient implementation are the following four observations from Algorithm 1 when applied to the structured large margin setting. In particular, we will exploit the fact that the marginals of $\boldsymbol{\alpha}_k$ can be updated iteratively.

- **The marginals of $\boldsymbol{\alpha}_{\mu_k}(\mathbf{w}_k)$ and $\hat{\boldsymbol{\alpha}}$ can be computed efficiently**. From (38) it is easy to see that $\boldsymbol{\alpha}_{\mu_k}(\mathbf{w}_k)$ can be written as a product of factors over cliques, that is, in the form of (33). Therefore, the marginals of $\boldsymbol{\alpha}_{\mu_k}(\mathbf{w}_k)$ can be computed efficiently. As a result, if we keep track of the marginal distributions of $\boldsymbol{\alpha}_k$, then it is trivial to compute the marginals of $\hat{\boldsymbol{\alpha}} = (1 - \tau_k)\boldsymbol{\alpha}_k + \tau_k \boldsymbol{\alpha}_{\mu_k}(\mathbf{w}_k)$ over all the combinations of example index i and clique index c. In the case of linear chain, there are nl combinations. It costs $O(s^2 p)$ to compute $\left\langle \boldsymbol{\psi}^i_{y'_c}, \mathbf{w}_c \right\rangle$ for each (i, c), and $O(s^2 l)$ to compute all the marginal distributions of $\left\{(\boldsymbol{\alpha}_\mu(\mathbf{w}))^i_c : c\right\}$ for each i.

- **The marginals of $\tilde{\boldsymbol{\alpha}}$ can be computed efficiently**. Define $\eta = \frac{-\tau_k}{(1-\tau_k)\mu_k}$. By plugging in (37) and (38) into (39) and observing that $\nabla D(\boldsymbol{\alpha})$ can be written as a sum of terms over cliques obtains:

$$\begin{aligned}\tilde{\boldsymbol{\alpha}}^i_{\mathbf{y}} &= \left(V\left(\boldsymbol{\alpha}_{\mu_k}(\mathbf{w}_k), \eta\nabla D(\hat{\boldsymbol{\alpha}})\right)\right)^i_{\mathbf{y}} \propto \left(\boldsymbol{\alpha}_{\mu_k}(\mathbf{w}_k)\right)^i_{\mathbf{y}} \exp\left(-\eta\left(\nabla D(\hat{\boldsymbol{\alpha}})\right)^i_{\mathbf{y}}\right) \\ &= \prod_c \exp\left(\mu_k^{-1}\left(\ell^i_{y_c} - \left\langle \boldsymbol{\psi}^i_{y_c}, (\mathbf{w}_k)_c \right\rangle\right) - \eta\ell^i_{y_c} + \eta\lambda^{-1}\left\langle \boldsymbol{\psi}^i_{y_c}, \mathbb{F}[\boldsymbol{\psi}_c; \hat{\boldsymbol{\alpha}}]\right\rangle\right). \qquad (40)\end{aligned}$$

  Clearly, $\tilde{\boldsymbol{\alpha}}$ factorizes and has the form of (33). Hence its marginals can be computed efficiently. For linear chains, given $\hat{\boldsymbol{\alpha}}$ it costs $O(s^2np)$ to compute $\mathbb{F}[\boldsymbol{\psi}_c; \hat{\boldsymbol{\alpha}}]$ for each c. With all $\left\langle \boldsymbol{\psi}^i_{y_c}, (\mathbf{w}_k)_c \right\rangle$ and $\mathbb{F}[\boldsymbol{\psi}_c; \hat{\boldsymbol{\alpha}}]$ available, it costs $O(s^2l)$ to compute the marginals $\{(\tilde{\boldsymbol{\alpha}}^i)_c : c\}$ for each i.

- **The marginals of $\boldsymbol{\alpha}_k$ can be updated efficiently**. Given the marginals of $\tilde{\boldsymbol{\alpha}}$, it is trivial to update the marginals of $\boldsymbol{\alpha}_{k+1}$ clique by clique since $\boldsymbol{\alpha}_{k+1} = (1-\tau_k)\boldsymbol{\alpha}_k + \tau_k\tilde{\boldsymbol{\alpha}}$. This costs $O(s^2)$ for each clique c corresponding to a particular example i in the case of linear chain. For convenience, define $\boldsymbol{\alpha}_c := \{\alpha^i_{y_c} : i, y_c\}$.

- **$\mathbf{w}_k$ can be updated efficiently**. According to step 5 of Algorithm 1, by using (36) we have

$$(\mathbf{w}_{k+1})_c = (1-\tau_k)(\mathbf{w}_k)_c + \tau_k(\mathbf{w}(\hat{\boldsymbol{\alpha}}))_c = (1-\tau_k)(\mathbf{w}_k)_c + \tau_k\lambda^{-1}\mathbb{F}[\boldsymbol{\psi}_c; \hat{\boldsymbol{\alpha}}]. \qquad (41)$$

  For a linear chain, it costs $O(pl)$ to compute $\{(\mathbf{w}_{k+1})_c : c\}$ given $\{\mathbb{F}[\boldsymbol{\psi}_c; \hat{\boldsymbol{\alpha}}] : c\}$.

Leveraging these observations, Algorithm 2 provides a complete listing of how to implement the excessive gap technique with Bregman projections for training $\mathsf{M}^3\mathsf{N}$. It focuses on clarifying the ideas; a practical implementation can be sped up in many ways. Its memory cost of is $O(nls^2 + pl)$, due to the storage of the marginals and feature means. The last issue to be addressed is the computation of the primal and dual objectives $J(\mathbf{w}_k)$ and $D(\boldsymbol{\alpha}_k)$, so as to monitor the duality gap. Indeed, this is viable without incurring higher order of computations and we leave the details to the reader. In Section 4.3, we will give the formula for kernelized versions.

Algorithm 2: Max-margin structured learning using clique factorization

```
Input: Loss functions $\{\ell^i_{\mathbf{y}}\}$ and features $\{\boldsymbol{\psi}^i_{\mathbf{y}}\}$, a regularization parameter $\lambda$, a
       tolerance level $\epsilon > 0$.
Output: A pair $\mathbf{w}$ and $\boldsymbol{\alpha}$ that satisfy $J(\mathbf{w}) - D(\boldsymbol{\alpha}) < \epsilon$.
Initialize: $k \leftarrow 1$, $\mu_1 \leftarrow \frac{1}{\lambda} \max_{i,\mathbf{y}} \left\|\boldsymbol{\psi}^i_{\mathbf{y}}\right\|_2^2 = \frac{1}{\lambda} \max_{i,\mathbf{y}} \sum_c \left\|\boldsymbol{\psi}^i_{\mathbf{y}_c}\right\|^2$ by (32),
$\boldsymbol{\alpha}_0 \leftarrow \left(\frac{1}{n|\mathcal{Y}|}, \ldots, \frac{1}{n|\mathcal{Y}|}\right)^\top \in \mathbb{R}^{n|\mathcal{Y}|}$, or equivalently set $(\boldsymbol{\alpha}_0)_c$ as a uniform distribution
(divided by $n$) for all clique $c$.
Update $\mathbf{w}_1 \leftarrow \mathbf{w}(\boldsymbol{\alpha}_0) = \frac{1}{\lambda} \oplus_{c\in\mathcal{C}} \mathbb{F}[\boldsymbol{\psi}_c; \boldsymbol{\alpha}_0]$, $\boldsymbol{\alpha}_1 \leftarrow V\left(\boldsymbol{\alpha}_0, -\frac{1}{\mu_1}\nabla D(\boldsymbol{\alpha}_0)\right)$ by computing
its marginals on all cliques via (40).
while $J(\mathbf{w}_k) - D(\boldsymbol{\alpha}_k) \geq \epsilon$. do    /* Terminate when duality gap falls below $\epsilon$ */
    $\tau_k \leftarrow \frac{2}{k+3}$.
    Compute the marginals of $\boldsymbol{\alpha}_{\mu_k}(\mathbf{w}_k)$ by exploiting (38).
    forall the cliques $c \in \mathcal{C}$ do
        Compute the marginals $\hat{\boldsymbol{\alpha}}_c$ by convex combination:
        $\hat{\boldsymbol{\alpha}}_c \leftarrow (1-\tau_k)(\boldsymbol{\alpha}_k)_c + \tau_k(\boldsymbol{\alpha}_{\mu_k}(\mathbf{w}_k))_c$.
        Update the weight on clique $c$: $(\mathbf{w}_{k+1})_c \leftarrow (1-\tau_k)(\mathbf{w}_k)_c + \frac{\tau_k}{\lambda} \sum_i \mathbb{F}[\boldsymbol{\psi}^i_c; \hat{\boldsymbol{\alpha}}_c]$.
    Compute the marginals of $\tilde{\boldsymbol{\alpha}}$ by using (40) and the marginals $\{\hat{\boldsymbol{\alpha}}_c\}$.
    forall the cliques $c \in \mathcal{C}$ do
        Update the marginals $(\boldsymbol{\alpha}_k)_c$ by convex combination:
        $(\boldsymbol{\alpha}_{k+1})_c \leftarrow (1-\tau_k)(\boldsymbol{\alpha}_k)_c + \tau_k \tilde{\boldsymbol{\alpha}}_c$.
    Update $\mu_{k+1} \leftarrow (1-\tau_k)\mu_k$, $k \leftarrow k+1$.
return $\{(\mathbf{w}_k)_c : c \in \mathcal{C}\}$ and $\{(\boldsymbol{\alpha}_k)_c : c \in \mathcal{C}\}$.
```

4.3. Kernelization

When nonlinear kernels are used, the feature vectors $\boldsymbol{\phi}^i_{\mathbf{y}}$ are not expressed explicitly and only their inner products can be evaluated via kernels on the cliques:

$$\left\langle \boldsymbol{\psi}^i_{\mathbf{y}}, \boldsymbol{\psi}^j_{\mathbf{y}'} \right\rangle := k((\mathbf{x}^i, \mathbf{y}), (\mathbf{x}^j, \mathbf{y}')) = \sum_c k_c((\mathbf{x}^i, y_c), (\mathbf{x}^j, y'_c)),$$

where $k_c((\mathbf{x}^i, y_c), (\mathbf{x}^j, y'_c)) := \left\langle \boldsymbol{\psi}^i_{y_c}, \boldsymbol{\psi}^j_{y'_c} \right\rangle$. Algorithm 2 is no longer applicable because no explicit expression of $\mathbf{w}$ is available. However, by rewriting $\mathbf{w}_k$ as the feature expectations with respect to some distribution $\beta_k \in \mathcal{S}^n$, then we only need to update $\mathbf{w}_k$ implicitly via β_k, and the inner product between $\mathbf{w}_k$ and any feature vector can also be efficiently calculated. We formalize and prove this claim by induction.

Theorem 9. *For all $k \geq 0$, there exists $\boldsymbol{\beta}_k \in \mathcal{S}^n$, such that $(\mathbf{w}_k)_c = \frac{1}{\lambda}\mathbb{F}[\boldsymbol{\psi}_c; \boldsymbol{\beta}_k]$, and $\boldsymbol{\beta}_k$ can be updated by $\boldsymbol{\beta}_{k+1} = (1-\tau_k)\boldsymbol{\beta}_k + \tau_k \hat{\boldsymbol{\alpha}}_k$.*

Proof. First, $\mathbf{w}_1 = \mathbf{w}(\boldsymbol{\alpha}_0) = \frac{1}{\lambda} \oplus_{c \in \mathcal{C}} \mathbb{F}[\boldsymbol{\psi}_c; \boldsymbol{\alpha}_0]$, so $\boldsymbol{\beta}_1 = \boldsymbol{\alpha}_0$. Suppose the claim holds for all $1, \ldots, k$, then

$$\begin{aligned}(\mathbf{w}_{k+1})_c &= (1-\tau_k)(\mathbf{w}_k)_c + \frac{\tau_k}{\lambda}\mathbb{F}[\boldsymbol{\psi}_c; (\hat{\boldsymbol{\alpha}}_k)_c] = (1-\tau_k)\frac{1}{\lambda}\mathbb{F}[\boldsymbol{\psi}_c; \boldsymbol{\beta}_k] + \frac{\tau_k}{\lambda}\mathbb{F}[\boldsymbol{\psi}_c; (\hat{\boldsymbol{\alpha}}_k)_c] \\ &= \frac{1}{\lambda}\mathbb{F}[\boldsymbol{\psi}_c; (1-\tau_k)(\boldsymbol{\beta}_k)_c + \tau_k(\hat{\boldsymbol{\alpha}}_k)_c].\end{aligned}$$

Therefore, we can set $\boldsymbol{\beta}_{k+1} = (1-\tau_k)\boldsymbol{\beta}_k + \tau_k \hat{\boldsymbol{\alpha}}_k \in \mathcal{S}^n$. ∎

In general $\hat{\boldsymbol{\alpha}}_k \neq \tilde{\boldsymbol{\alpha}}_k$, hence $\boldsymbol{\beta}_k \neq \boldsymbol{\alpha}_k$. To compute $\langle \boldsymbol{\psi}^i_{y_c}, (\mathbf{w}_k)_c \rangle$ required by (40), we have

$$\langle \boldsymbol{\psi}^i_{y_c}, (\mathbf{w}_k)_c \rangle = \left\langle \boldsymbol{\psi}^i_{y_c}, \frac{1}{\lambda}\sum_j \sum_{y'_c} \beta^j_{y'_c} \boldsymbol{\psi}^j_{y'_c} \right\rangle = \frac{1}{\lambda}\sum_j \sum_{y'_c} \beta^j_{y'_c} k_c((\mathbf{x}^i, y_c), (\mathbf{x}^j, y'_c)).$$

And by using this trick, all the iterative updates in Algorithm 2 can be done efficiently. So is the evaluation of $\|\mathbf{w}_k\|^2$ and the primal objective. The dual objective (26) is also easy since

$$\sum_i \sum_{\mathbf{y}} \ell^i_{\mathbf{y}} (\boldsymbol{\alpha}_k)^i_{\mathbf{y}} = \sum_i \sum_{\mathbf{y}} \sum_c \ell^i_{y_c} (\boldsymbol{\alpha}_k)^i_{\mathbf{y}} = \sum_i \sum_c \sum_{y_c} \ell^i_{y_c} \sum_{\mathbf{y}:\mathbf{y}|_c = y_c} (\boldsymbol{\alpha}_k)^i_{\mathbf{y}} = \sum_{i,c,y_c} \ell^i_{y_c} (\boldsymbol{\alpha}_k)^i_{y_c},$$

and the marginals of $\boldsymbol{\alpha}_k$ are available. Finally, the quadratic term in $D(\boldsymbol{\alpha}_k)$ can be computed by

$$\begin{aligned}\left\|A^\top \boldsymbol{\alpha}_k\right\|_2^2 &= \Big\| \sum_{i,\mathbf{y}} \boldsymbol{\psi}^i_{\mathbf{y}} (\boldsymbol{\alpha}_k)^i_{\mathbf{y}} \Big\|_2^2 = \sum_c \Big\| \sum_{i,y_c} \boldsymbol{\psi}^i_{y_c} (\boldsymbol{\alpha}_k)^i_{y_c} \Big\|_2^2 \\ &= \sum_c \sum_{i,j,y_c,y'_c} (\boldsymbol{\alpha}_k)^i_{y_c} (\boldsymbol{\alpha}_k)^j_{y'_c} k_c((\mathbf{x}^i, y_c), (\mathbf{x}^j, y'_c)),\end{aligned}$$

where the inner term is the same as the unnormalized expectation that can be efficiently calculated. The last formula is only for nonlinear kernels.

Note for nonlinear kernels, the computational cost can be shown to be $O(n^2 l s^2)$.

5. Discussion

Structured output prediction is an important learning task in both theory and practice. The main contribution of our paper is twofold. First, we identified an efficient algorithm by Nesterov [1] for solving the optimization problems in structured prediction. We proved the $O(1/\sqrt{\epsilon})$ rate of convergence for the Bregman projection based updates in excessive gap optimization, while Nesterov [1] showed this rate only for projected gradient style updates. In $\mathsf{M}^3\mathsf{N}$ optimization, Bregman projection plays a key role in factorizing the computations, while technically such factorizations are not applicable to projected gradient. Second, we designed a nontrivial application of the excessive gap technique to $\mathsf{M}^3\mathsf{N}$ optimization, in which the computations are kept efficient by using the graphical model decomposition. Kernelized objectives can also be handled by our method, and we proved superior convergence and computational guarantees than existing algorithms.

When $\mathsf{M}^3\mathsf{N}$s are trained in a batch fashion, we can compare the convergence rate of dual gap between our algorithm and the exponentiated gradient method [ExpGrad, 8]. Assume $\boldsymbol{\alpha}_0$, the initial value of $\boldsymbol{\alpha}$, is the uniform distribution and $\boldsymbol{\alpha}^*$ is the optimal dual solution. Then by (28), we have

$$\text{Ours:}\quad \max_{i,\mathbf{y}} \left\|\boldsymbol{\psi}^i_{\mathbf{y}}\right\|_2 \sqrt{\frac{6\mathrm{KL}(\boldsymbol{\alpha}^*||\boldsymbol{\alpha}_0)}{\lambda\epsilon}}, \qquad \mathsf{ExpGrad:}\quad \max_{i,\mathbf{y}} \left\|\boldsymbol{\psi}^i_{\mathbf{y}}\right\|_2^2 \frac{\mathrm{KL}(\boldsymbol{\alpha}^*||\boldsymbol{\alpha}_0)}{\lambda\epsilon}.$$

It is clear that our iteration bound is almost the square root of ExpGrad, and has much better dependence on ϵ, λ, $\max_{i,\mathbf{y}} \left\|\boldsymbol{\psi}^i_{\mathbf{y}}\right\|_2$, as well as the divergence from the initial guess to the optimal solution $\mathrm{KL}(\boldsymbol{\alpha}^*||\boldsymbol{\alpha}_0)$.

In addition, the cost per iteration of our algorithm is almost the same as ExpGrad, and both are governed by the computation of the expected feature values on the cliques (which we call exp-oracle), or equivalently the marginal distributions. For graphical models, exact inference algorithms such as belief propagation can compute the marginals via dynamic programming [19]. Finally, although both algorithms require marginalization, they are calculated in very different ways. In ExpGrad, the dual variables $\boldsymbol{\alpha}$ correspond to a factorized distribution, and in each iteration its potential functions on the cliques are updated using the exponentiated gradient rule. In contrast, our algorithm explicitly updates the marginal distributions of $\boldsymbol{\alpha}_k$ on the cliques, and marginalization inference is needed only for $\hat{\boldsymbol{\alpha}}$ and $\tilde{\boldsymbol{\alpha}}$. Indeed, the joint distribution $\boldsymbol{\alpha}$ does *not* factorize, which can be seen from step 7 of Algorithm 1: the convex combination of two factorized distributions is not necessarily factorized.

Marginalization is just one type of query that can be answered efficiently by graphical models, and another important query is the max a-posteriori inference (which we call max-oracle): given the current model $\mathbf{w}$, find the argmax in (2). Max-oracle has been used by greedy algorithms such as cutting plane (BMRM and SVM-Struct) and sequential minimal optimization [SMO, 9, Chapter 6]. SMO picks the steepest descent coordinate in the dual and greedily optimizes the quadratic analytically, but its convergence rate is linear in $|\mathcal{Y}|$ which can be exponentially large for M^{3}N (ref Table 1). The max-oracle again relies on graphical models for dynamical programming [22], and many existing combinatorial optimizers can also be used, such as in the applications of matching [23] and context free grammar parsing [24]. Furthermore, this oracle is particularly useful for solving the slack rescaling variant of M^{3}N proposed by [6]:

$$J(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} \max_{\mathbf{y}\in\mathcal{Y}} \left\{ \ell(\mathbf{y}, \mathbf{y}^i; \mathbf{x}^i) \left(1 - \left\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}^i) - \boldsymbol{\phi}(\mathbf{x}^i, \mathbf{y}) \right\rangle \right) \right\}. \tag{42}$$

Here two factorized terms get multiplied, which causes additional complexity in finding the maximizer. [25, Section 1.4.1] solved this problem by a modified dynamic program. Nevertheless, it is not clear how ExpGrad or our method can be used to optimize this objective.

In the quest for faster optimization algorithms for M^{3}Ns, the following three questions are important: how hard is it to optimize M^{3}N intrinsically, how informative is the oracle which is the only way for the algorithm to access the objective function, and how well does the algorithm make use of such information. In the oracle-optimizer model proposed in [14], a solver can access the target objective only through an oracle, *e.g.*, the function value and its derivative at a given point. Cutting plane methods [6] and bundle methods [5] use a max-oracle, *i.e.* given a query point $\mathbf{w}_k$, the only information available to the solver about the objective (2) is the linear piece determined by the label $\mathbf{y}$ that "wins" in the $\max_{\mathbf{y}\in\mathcal{Y}}$, while all other linear pieces are ignored. In contrast, our algorithm *essentially* endows a distribution over all labels $\mathbf{y}$ as in (29), and it aggregates the information of all labels $\mathbf{y}$ with respect to this distribution. Hence we call it exp-oracle ('exp' for expectation).

The superiority of our algorithm suggests that the exp-oracle provides more information about the function than the max-oracle does, and a deeper explanation is that the max-oracle is local [14, Section 1.3], *i.e.* it depends only on the value of the function in the neighborhood of the querying point $\mathbf{w}_k$. In contrast, the exp-oracle is not local and uses the global structure of the function. Hence there is no surprise that the less informative max-oracle is easier to compute, which makes it applicable to a wider range of problems such as (42). Moreover, the comparison between ExpGrad and our algorithm shows that even if the exp-oracle is used, the algorithm still needs to make good use of it in order to converge faster.

For future research, it is interesting to study the lower bound complexity for optimizing M^{3}N, including the dependence on ϵ, n, λ, $\mathcal{Y}$, and probably even on the graphical model topology. Empirical evaluation of our algorithm is also important, especially regarding the numerical stability of the additive update of marginal distributions $\boldsymbol{\alpha}_k$ under fixed precision. Broader applications are possible in sequence labeling, word alignment, context free grammar parsing, etc.

Acknowledgements. We thank the anonymous reviewers of ALT 2011 for many insightful comments and helpful suggestions. Xinhua Zhang would like to acknowledge support from the Alberta Innovates Centre for Machine Learning. Ankan Saha would like to acknowledge the fellowship support of the University of Chicago. The work of S.V. N. Vishwanathan is partially supported by a grant from Google and NSF grant IIS-1117705.

References

[1] Y. Nesterov, Excessive Gap Technique in Nonsmooth Convex Minimization, SIAM Journal on Optimization 16 (1) (2005) 235–249.

[2] G. Bakir, T. Hofmann, B. Schölkopf, A. Smola, B. Taskar, S. V. N. Vishwanathan, Predicting Structured Data, MIT Press, 2007.

[3] J. D. Lafferty, A. McCallum, F. Pereira, Conditional Random Fields: Probabilistic Modeling for Segmenting and Labeling Sequence Data, in: Proceedings of International Conference on Machine Learning, 2001.

[4] B. Taskar, C. Guestrin, D. Koller, Max-Margin Markov Networks, in: Advances in Neural Information Processing Systems 16, 2004.

[5] C. Teo, S. Vishwanthan, A. Smola, Q. Le, Bundle Methods for Regularized Risk Minimization, Journal of Machine Learning Research 11 (2010) 311–365.

[6] I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, Large Margin Methods for Structured and Interdependent Output Variables, Journal of Machine Learning Research 6 (2005) 1453–1484.

[7] B. Taskar, S. Lacoste-Julien, M. Jordan, Structured Prediction, Dual Extragradient and Bregman Projections, Journal of Machine Learning Research 7 (2006) 1627–1653.

[8] M. Collins, A. Globerson, T. Koo, X. Carreras, P. Bartlett, Exponentiated Gradient Algorithms for Conditional Random Fields and Max-Margin Markov Networks, Journal of Machine Learning Research 9 (2008) 1775–1822.

[9] B. Taskar, Learning Structured Prediction Models: A Large Margin Approach, Ph.D. thesis, Stanford University, 2004.

[10] N. List, H. U. Simon, SVM-Optimization and Steepest-Descent Line Search, in: Proceedings of the Annual Conference on Computational Learning Theory, 2009.

[11] F. Sha, F. Pereira, Shallow Parsing with Conditional Random Fields, in: Proceedings of HLT-NAACL, 2003.

[12] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, Cambridge, England, 2004.

[13] T. Joachims, T. Finley, C.-N. J. Yu, Cutting-Plane Training of Structural SVMs, Machine Learning 77 (1) (2009) 27–59.

[14] A. Nemirovski, D. Yudin, Problem Complexity and Method Efficiency in Optimization, John Wiley and Sons, 1983.

[15] Y. Nesterov, Smooth minimization of non-smooth functions, Mathematical Programming 103 (1) (2005) 127–152.

[16] J. Hiriart-Urruty, C. Lemaréchal, Convex Analysis and Minimization Algorithms, I and II, vol. 305 and

306, Springer-Verlag, 1993.

[17] J. M. Borwein, A. S. Lewis, Convex Analysis and Nonlinear Optimization: Theory and Examples, Canadian Mathematical Society, 2000.

[18] A. Beck, B. Teboulle, Mirror Descent and Nonlinear Projected Subgradient Methods for Convex Optimization, Operations Research Letters 31 (3) (2003) 167–175.

[19] S. L. Lauritzen, Graphical Models, Oxford University Press, Oxford, UK, 1996.

[20] M. Wainwright, M. Jordan, Graphical Models, Exponential Families, and Variational Inference, Foundations and Trends in Machine Learning 1 (1–2) (2008) 1–305.

[21] C. Andrieu, N. de Freitas, A. Doucet, M. I. Jordan, An Introduction to MCMC for Machine Learning, Machine Learning 50 (2003) 5–43.

[22] F. Kschischang, B. J. Frey, H. Loeliger, Factor Graphs and the Sum-Product Algorithm, IEEE Trans. on Information Theory 47 (2) (2001) 498–519.

[23] B. Taskar, S. Lacoste-Julien, D. Klein, A Discriminative Matching Approach to Word Alignment, in: Empirical Methods in Natural Language Processing, 2005.

[24] B. Taskar, D. Klein, M. Collins, D. Koller, C. Manning, Max-Margin Parsing, in: Empirical Methods in Natural Language Processing, 2004.

[25] Y. Altun, T. Hofmann, I. Tsochandiridis, Support Vector Machine Learning for Interdependent and Structured Output Spaces, in: G. Bakir, T. Hofmann, B. Schölkopf, A. Smola, B. Taskar, S. V. N. Vishwanathan (Eds.), Predicting Structured Data, chap. 5, MIT Press, 85–103, 2007.