Learning Weighted Lower Linear Envelope Potentials in Binary Markov Random Fields

Stephen Gould, *Member, IEEE*

Abstract—Markov random fields containing higher-order terms are becoming increasingly popular due to their ability to capture complicated relationships as soft constraints involving many output random variables. In computer vision an important class of constraints encode a preference for label consistency over large sets of pixels and can be modeled using higher-order terms known as *lower linear envelope potentials*. In this paper we develop an algorithm for learning the parameters of binary Markov random fields with weighted lower linear envelope potentials. We first show how to perform exact energy minimization on these models in time polynomial in the number of variables and number of linear envelope functions. Then, with tractable inference in hand, we show how the parameters of the lower linear envelope potentials can be estimated from labeled training data within a max-margin learning framework. We explore three variants of the lower linear envelope parameterization and demonstrate results on both synthetic and real-world problems.

Index Terms—higher-order MRFs, lower linear envelope potentials, max-margin learning

1 Introduction

Markov random field (MRF) parameter learning is a challenging task that has advanced considerably in the past several years with the introduction of the max-margin principle for structured prediction [36, 37]. The standard max-margin approach is to learn model parameters by constraining the prediction rule to favour the ground-truth assignment over all other joint assignments to the variables. Since the set of all possible joint assignments can be prohibitively large (exponential in the number of the variables), constraints are introduced incrementally by finding the most violated ones (with respect to the current parameter settings) during each iteration of the learning algorithm.

Despite this advance, learning the parameters of an MRF remains a notoriously difficult task due to the problem of finding the most violated constraints, which requires performing exact maximum a-posteriori (MAP) inference. Except in a few special cases, such as tree-structured graphs or binary pairwise MRFs with submodular potentials [21], exact inference is intractable and the max-margin framework cannot be applied. When substituting approximate inference routines to generate constraints, the max-margin framework is not guaranteed to learn the optimal parameters and often performs poorly [10].

Recently, models with structured higher-order terms have become of interest to the machine learning community with many applications in computer vision, particularly for encoding consistency constraints over large sets of pixels, e.g., [26, 27, 33]. A rich class of higher-order models, known as *lower linear envelope potentials*, was proposed by Kohli and Kumar [17]. The class defines a concave function of label cardinality (i.e., number of variables taking each label) and includes the generalized Potts model [19] and its variants. While efficient approximate inference algorithms based on message-passing or move-making exist for these models, parameter learning remains an unsolved problem.

In this paper we focus on learning the parameters of weighted lower linear envelope potentials for binary MRFs. We present an exact MAP inference algorithm for these models that is polynomial in the number of variables and number of linear envelope functions. This opens the way for max-margin parameter learning. However, to encode the max-margin constraints we require a linear relationship between model parameters and the features that encode each problem instance.

Our key insight is that we can represent the weighted lower linear envelope in two different ways. The first way encodes the envelope as the minimum over a set of linear functions and admits tractable algorithms for MAP inference, which is required during constraint generation. The second representation encodes the envelope by linearly interpolating between a sequence of sample points. This representation allows us to treat the potential as a linear combination of features and weights as required for max-margin parameter learning. By mapping between these two representations we can learn model parameters efficiently. Indeed, other linear parameterizations are also possible and we explore these together with the corresponding feature representations.

We evaluate our approach on synthetic data as well as two real-world problems—a variant of the "GrabCut" interactive image segmentation problem [32] and segmentation of horses from the Weizmann Horse dataset [2]. Our experiments show that models with learned higher-order terms can result in improved pixelwise segmentation accuracy.

• *S. Gould is with the Research School of Computer Science, Australian National University, ACT 0200, Australia.*
E-mail: stephen.gould@anu.edu.au

2 Related Work

Our work focuses on a class of higher-order potentials known as lower linear envelope potentials, which can be used to represent arbitrary concave functions over the number of variables (in a clique) taking a given assignment. Kohli and Kumar [17] show how such potentials can be represented in an energy-minimization setting by introducing a multi-valued auxiliary variable to select each linear function in the envelope. In principle, the optimal assignment can be found by jointly minimizing the energy function over the original variables and this auxiliary variable. However, this is non-trivial, in general, and Kohli and Kumar [17] only show how the resulting energy function can be approximately optimized.

Earlier research, on MRFs with restricted variants of the lower linear envelope potential, showed how exact inference can be performed in the binary case. Kohli et al. [19] introduced the P^n-model for encoding consistency constraints. This was later extended to the robust P^n-model by Kohli et al. [20] who also describe an efficient move-making inference algorithm based on graph-cuts [5, 6]. The robust P^n-model is a lower linear envelope potential with only two terms per label—one increasing and one constant. Multiple robust P^n-models can be added to form a non-decreasing concave envelope. However, these works did not address the problem of parameter learning. Ladicky et al. [25] used this model for improving the quality of multi-class image labeling with parameters set by hand. The model is still considered state of the art for the multi-class image labeling task (i.e., dense semantic segmentation). For segmenting large foreground classes (like those in the PASCAL VOC Challenge [9]) methods based on proposing whole-object segments are showing promise (e.g., [7]).

In contrast to these works, we propose an algorithm for exactly optimizing binary MRFs with *arbitrary* lower linear envelope potentials and show how to learn their parameters. We extend our previous work [12] to allow each variable within the potential to be assigned an arbitrary non-negative weight. Our work and the previous approaches described above are related to a number of methods in discrete optimization that transform higher-order or multi-label energy functions into quadratic pseudo-Boolean functions (e.g., [14, 15, 33]). These functions have been studied extensively in the operations research literature (for a survey see Boros and Hammer [4]). Under certain conditions, the resulting pseudo-Boolean function can be minimized exactly by finding the minimum-cut in a suitably constructed graph [11, 13]. Our work and other graph-cut approaches make use of this result.

More general methods—falling under the class of submodular energy minimization algorithms—can also be used to optimize binary MRFs with arbitrary lower linear envelope potentials. These include the strongly polynomial algorithm of Orlin [29] for general submodular functions and the recently proposed submodular flows method of Kolmogorov [22] for minimizing the sum of submodular functions. However, the availability of efficient software implementations and strong empirical performance of graph-cut approaches [5] makes graph-cuts the most appropriate for our problem.

Our max-margin learning framework is based on the approaches introduced by Tsochantaridis et al. [37, 38] and Taskar et al. [36], which have been successfully applied within many application domains (see Joachims et al. [16] for a recent survey and the "1-slack" reformulation). Szummer et al. [35] showed how this framework could be adapted to learn pairwise MRF parameters using graph-cuts for inference. Unlike their approach, our method applies to models with higher-order terms—specifically, weighted lower linear envelopes.

3 Lower Linear Envelope MRFs

We begin with a brief overview of higher-order Markov random fields (MRFs). We then introduce the lower linear envelope potential and show how to perform exact inference in models with these potentials. In Section 4 we will discuss learning the parameters of the models.

3.1 Higher-order MRFs

The *energy function* for a higher-order MRF over discrete random variables $\boldsymbol{y} = \{y_1, \ldots, y_n\}$ can be written as:

$$E(\boldsymbol{y}) = \underbrace{\sum_{i=1}^{n} \psi_i^U(y_i)}_{\text{unary}} + \underbrace{\sum_{ij \in \mathcal{E}} \psi_{ij}^P(y_i, y_j)}_{\text{pairwise}} + \underbrace{\sum_{c \in \mathcal{C}} \psi_c^H(\boldsymbol{y}_c)}_{\text{higher-order}} \quad (1)$$

where the *potential* functions ψ_i^U, ψ_{ij}^P and ψ_c^H encode preferences for unary, pairwise and k-ary variable assignments, respectively. The pairwise terms, ψ_{ij}^P, also called *edge potentials*, are usually only defined over a sparse subset $\mathcal{E}$ of possible variable pairs (y_i, y_j). The latter terms, ψ_c^H, are defined over arbitrary subsets of variables (or *cliques*), $\boldsymbol{y}_c = \{y_i : i \in c\}$ where $c \subseteq \{1, \ldots, n\}$ is a subset of variable indices. When $|c| > 2$ these are known as *higher-order potentials*. The higher-order potentials can be defined over multiple overlapping cliques. We denote the set of all cliques for which a higher-order potential is defined by $\mathcal{C}$.

In this paper, we will be concerned with inference and learning of higher-order binary MRFs (i.e., $y_i \in \{0, 1\}$) with weighted lower linear envelope potentials. These have been attracting much interest in computer vision applications for encoding consistency constraints over large subsets of pixels in an image [19, 26, 28]. A weighted lower linear envelope potential over a subset (clique) of binary variables $\boldsymbol{y}_c$ is a piecewise linear function defined as the minimum over a set of K linear functions as

$$\psi_c^H(\boldsymbol{y}_c) \triangleq \min_{k=1,\ldots,K} \left\{ a_k \sum_{i \in c} w_i^c y_i + b_k \right\} \quad (2)$$

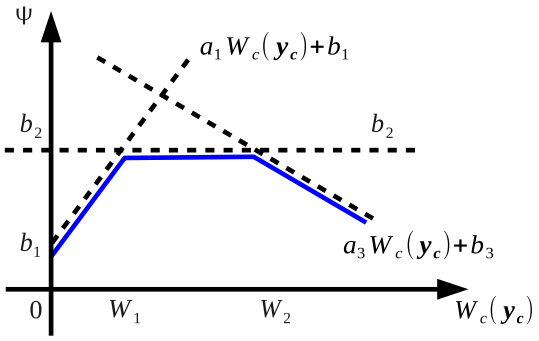


Fig. 1: Example lower linear envelope $\psi_c^H(\boldsymbol{y}_c)$ (shown solid) with three terms (dashed) as a function of $W_c(\boldsymbol{y}_c) = \sum_{i \in c} w_i^c y_i$. When $W_c(\boldsymbol{y}_c) \leq W_1$ the first linear function is active, when $W_1 < W_c(\boldsymbol{y}_c) \leq W_2$ the second linear function is active, otherwise the third linear function is active.

where $w_i^c \geq 0$ are non-negative per-variable weights for each clique with $\sum_{i \in c} w_i^c = 1$, and $(a_k, b_k) \in \mathbb{R}^2$ are the linear function parameters.[1] To simplify notation in the sequel, we will define the *weight* of the clique for a given assignment $\boldsymbol{y}_c$ as $W_c(\boldsymbol{y}_c) = \sum_{i \in c} w_i y_i$. Using this definition we can re-write Equation 2 as

$$\psi_c^H(\boldsymbol{y}_c) = \min_{k=1,\ldots,K} \{a_k W_c(\boldsymbol{y}_c) + b_k\}. \tag{3}$$

Note that if all variables are assigned the same weight $w_i^c = \frac{1}{|c|}$, then $W_c(\boldsymbol{y}_c)$ is simply the proportion of variables in the clique taking assignment one. This restriction (of uniform weights) was assumed in Gould [12].

Figure 1 depicts an example lower envelope for three linear functions. Kohli and Kumar [17] showed that this representation can encode arbitrary concave functions of $W_c(\boldsymbol{y}_c)$ given sufficiently many linear functions. The parameterization, however, is not unique.

Definition 3.1 (Active). *We say that the k-th linear function is* active *with respect to an assignment $\boldsymbol{y}_c$ if $\psi_c^H(\boldsymbol{y}_c) = a_k W_c(\boldsymbol{y}_c) + b_k$.*

Note that more than one linear function can be active for the same assignment (e.g., at points where two or more linear functions intersect). Clearly, however, if a linear function is never active, or only active whenever another linear function is also active, it can be removed from the potential without changing the energy function.

Definition 3.2 (Redundant). *We say that the k-th linear function is* redundant *if it is not active for any assignment to $\boldsymbol{y}_c$ in any clique $c \in \mathcal{C}$ or is only active whenever another linear function is also active.*

Although not strictly necessary, in the following, we assume that our potentials do not contain redundant linear functions. Furthermore, we assume that the parameters $\{(a_k, b_k)\}_{k=1}^K$ are sorted in decreasing order of a_k. Clearly, this implies that $a_k > a_{k+1}$ and $b_k < b_{k+1}$ since if not then the k-th linear function will lie above the $(k+1)$-th linear function for all configurations of $\boldsymbol{y}_c$, i.e., the k-th linear function will never be active.

One may ask what conditions on $\{(a_k, b_k)\}_{k=1}^K$ ensure that the potentials do not have any redundant linear functions. The following proposition provides such a characterization.

Proposition 3.1. : Let $f : [0,1] \rightarrow \mathbb{R}$ be defined by $f(x) = \min_{k=1,\ldots,K} \{a_k x + b_k\}$. Assume the a_k are sorted in decreasing order (so $a_k > a_{k+1}$). Then the k-th linear function is not redundant if

$$0 < \frac{b_k - b_{k-1}}{a_{k-1} - a_k} < \frac{b_{k+1} - b_k}{a_k - a_{k+1}} < 1. \tag{4}$$

Proof: The k-th linear function is active (and no other linear function is active) if there exists $x \in (0,1)$ such that the following two inequalities hold

$$a_{k-1} x + b_{k-1} > a_k x + b_k$$
$$a_{k+1} x + b_{k+1} > a_k x + b_k$$

Rearranging for x and adding the constraint that $0 < x < 1$ gives the result. □

Finally, we note that arbitrarily shifting each linear function in the lower-linear envelope potential up or down by the same fixed constant does not change the energy-minimizing assignment $\boldsymbol{y}^\star$ as captured by the following observation.

Observation 3.2.: Let $\psi_c^H(\boldsymbol{y}_c)$ be defined as in Equation 3 and let $\widetilde{\psi}_c^H(\boldsymbol{y}_c) = \min_{k=1,\ldots,K} \{a_k W_c(\boldsymbol{y}_c) + b_k + b^{\text{const}}\}$. Then

$$\operatorname*{argmin}_{\boldsymbol{y}_c} \psi_c^H(\boldsymbol{y}_c) = \operatorname*{argmin}_{\boldsymbol{y}_c} \widetilde{\psi}_c^H(\boldsymbol{y}_c). \tag{5}$$

This property will be useful when considering different learning algorithms and will allow us to make convenient assumptions such as $b_1 = 0$ (and therefore all the b_k are non-negative), without loss of generality.

3.2 Exact Inference

The goal of inference is to find an energy-minimizing assignment $\boldsymbol{y}^\star \in \operatorname{argmin}_{\boldsymbol{y}} E(\boldsymbol{y})$. As we will show our energy function is submodular so can be solved in time polynomial in the number of variables by general-purpose submodular minimization algorithms [22, 29]. However, the special form of the lower linear envelope higher-order terms admits the use of much faster graph-cut based methods.

We follow the approach of a number of works that address the problem of inference in certain classes of higher-order MRFs by transforming the inference problem to that of minimizing a quadratic pseudo-Boolean function, i.e., pairwise MRF (e.g., [4, 11, 15]). For example, Kohli et al. [20] showed that exact inference can be performed using graph-cuts when the potential is a

1. The condition that $\sum_{i \in c} w_i^c = 1$ is assumed so that the same parameters (a_k, b_k) can be used for cliques with different per-variable weights and still produce the same shaped lower linear envelope. Note that if $Z = \sum_{i \in c} w_i^c \neq 1$ we can always normalize the w_i^c by dividing throughout by Z.

concave piecewise linear function of at most three terms (one increasing, one constant, and one decreasing). Arbitrary concave functions can be handled by decomposing them into a sum of piecewise linear functions of two or three terms. Gould [12] showed an alternative graph-cut method for minimizing potentials with arbitrary many terms. We now develop a weighted version of that method.

Consider, again the weighted lower linear envelope potential represented by Equation 2. Introducing $K-1$ auxiliary binary variables $\boldsymbol{z} = (z_1, \ldots, z_{K-1})$, we define the quadratic pseudo-Boolean function

$$E^c(\boldsymbol{y}_c, \boldsymbol{z}) = a_1 W_c(\boldsymbol{y}_c) + b_1 + \sum_{k=1}^{K-1} z_k \left((a_{k+1} - a_k) W_c(\boldsymbol{y}_c) + b_{k+1} - b_k \right) \quad (6)$$

for a single clique $c \in \mathcal{C}$.

The advantage of this formulation is that minimizing over $\boldsymbol{z}$, subject to some constraints, selects (one of) the active function(s) from $\psi_c^H(\boldsymbol{y})$ as we will now show.

Proposition 3.3. : Minimizing the function $E^c(\boldsymbol{y}_c, \boldsymbol{z})$ over $\boldsymbol{z}$ subject to $z_{k+1} \leq z_k$ for all k is equivalent to $\min_{k=1,\ldots,K} \{a_k W_c(\boldsymbol{y}_c) + b_k\}$, i.e.,

$$\psi_c^H(\boldsymbol{y}_c) = \min_{\boldsymbol{z}: z_{k+1} \leq z_k} E^c(\boldsymbol{y}_c, \boldsymbol{z}).$$

Proof: The constraints ensure that $\boldsymbol{z}$ takes the form of a vector of all ones followed by all zeros. There are K such vectors and for $k = \mathbf{1}^T \boldsymbol{z} + 1$ we have $E^c(\boldsymbol{y}_c, \boldsymbol{z}) = a_k W_c(\boldsymbol{y}_c) + b_k$. Therefore, minimizing over $\boldsymbol{z}$ is the same as minimizing over $k \in \{1, \ldots, K\}$. □

In Gould [12] we showed that the constraints on $\boldsymbol{z}$ can be enforced by adding $M z_{k+1}(1 - z_k)$ for $k = 1, \ldots, K-2$ to the energy function with M sufficiently large. We now show that it is not necessary to add these terms as the constraints are either automatically satisfied (with $M = 0$) or violations of the constraints do not affect the value of the energy function or the optimal assignment of $\boldsymbol{y}_c$.

Lemma 3.4.: Unconstrained (binary) minimization of the function $E^c(\boldsymbol{y}_c, \boldsymbol{z})$ over $\boldsymbol{z}$ is equivalent to minimization of $E^c(\boldsymbol{y}_c, \boldsymbol{z})$ subject to the constraints $z_{k+1} \leq z_k$.

Before proving the lemma, we make two observations.

Observation 3.5.: Assume that for some assignment $\boldsymbol{y}_c$ we have $a_{k+1} W_c(\boldsymbol{y}_c) + b_{k+1} = a_k W_c(\boldsymbol{y}_c) + b_k$. Then, for any assignment to $\boldsymbol{z} \in \{0, 1\}^{K-1}$, flipping the value of z_k does not change $E^c(\boldsymbol{y}_c, \boldsymbol{z})$.

Observation 3.6. : For any assignment $\boldsymbol{y}_c$ such that $(a_{k+1} - a_k) W_c(\boldsymbol{y}_c) + b_{k+1} - b_k \leq 0$ we have that setting $z_k = 1$ will result in a lower energy than setting $z_k = 0$.

We now proceed with the proof of Lemma 3.4.

Proof: Consider minimizing Equation 6 over $\boldsymbol{z}$ for fixed $\boldsymbol{y}_c$. Clearly, from our observations above, $z_k = 1$ if $(a_{k+1} - a_k) W_c(\boldsymbol{y}_c) + b_{k+1} - b_k < 0$. Likewise $z_k = 0$ if $(a_{k+1} - a_k) W_c(\boldsymbol{y}_c) + b_{k+1} - b_k > 0$. Now assume $z_{k+1} = 1$. Then

$$a_{k+2} W_c(\boldsymbol{y}_c) + b_{k+2} \leq a_{k+1} W_c(\boldsymbol{y}_c) + b_{k+1}$$

Since none of the linear functions are redundant, we must have (by Proposition 3.1)

$$a_{k+1} W_c(\boldsymbol{y}_c) + b_{k+1} \leq a_k W_c(\boldsymbol{y}_c) + b_k$$

otherwise the $(k+1)$-th function will never be active. If the above holds with equality then the value of z_k does not affect the value of $E^c(\boldsymbol{y}_c, \boldsymbol{z})$. Otherwise $z_k = 1$ and the constraint $z_{k+1} \leq z_k$ is satisfied. □

Rewriting the quadratic pseudo-Boolean function of Equation 6 in *posiform* [4], we have

$$E^c(\boldsymbol{y}_c, \boldsymbol{z}) = b_1 - (a_1 - a_K) + \sum_{i \in c} a_1 w_i^c y_i + \sum_{k=1}^{K-1} (b_{k+1} - b_k) z_k + \sum_{k=1}^{K-1} (a_k - a_{k+1}) \bar{z}_k + \sum_{k=1}^{K-1} \sum_{i \in c} (a_k - a_{k+1}) w_i^c \bar{y}_i z_k \quad (7)$$

where $\bar{z}_k = 1 - z_k$ and $\bar{y}_i = 1 - y_i$, and all coefficients (apart from the constant term) are positive.[2]

Importantly, $E^c(\boldsymbol{y}_c, \boldsymbol{z})$ is a *submodular* energy function, which allows us to perform efficient inference by minimizing jointly over both variables $\boldsymbol{y}_c$ and auxiliary variables $\boldsymbol{z}$.

Proposition 3.7.: The energy function $E^c(\boldsymbol{y}_c, \boldsymbol{z})$ defined by Equation 7 is submodular.

Proof: Follows from the fact that all the bi-linear terms in Equation 7 are of the form $\lambda \bar{u} v$ with $\lambda \geq 0$. See Boros and Hammer [4]. □

It is well known that submodular pairwise energy functions can be minimized exactly in time polynomial in the number of variables by finding the minimum-st-cut on a suitably constructed graph [13, 23]. We illustrate one possible construction for $E^c(\boldsymbol{y}_c, \boldsymbol{z})$ in Figure 2.

Using this fact, we can show that an energy function containing arbitrary weighted lower linear envelope potentials can be minimized in polynomial time.

Theorem 3.8.: For binary variables $\boldsymbol{y} \in \{0, 1\}^n$, let $E^0(\boldsymbol{y})$ be a submodular energy function, and let

$$E(\boldsymbol{y}) = E^0(\boldsymbol{y}) + \sum_{c \in \mathcal{C}} \psi_c^H(\boldsymbol{y}_c),$$

where $\psi_c^H(\boldsymbol{y}_c)$ are arbitrary weighted lower linear envelope higher-order potentials. Then $E(\boldsymbol{y})$ can be minimized in time polynomial in the number of variables n and total number of linear envelope functions.

2. Here we have assumed that $a_1 \geq 0$. If $a_1 < 0$ then the term $\sum_{i \in c} a_1 w_i^c y_i$ should be replaced with $a_1 + \sum_{i \in c} |a_1| w_i^c \bar{y}_i$. For all other terms, recall we have $a_k > a_{k+1}$ and $b_k < b_{k+1}$.

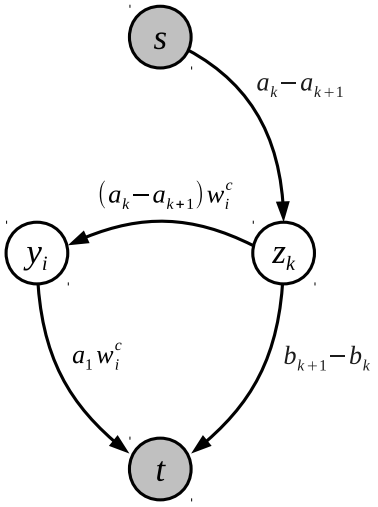


Fig. 2: Construction of an st-graph for minimizing energy functions with arbitrary weighted lower linear envelope potentials. Every cut corresponds to an assignment to the random variables, where variables associated with nodes in the $\mathcal{S}$ set take the value one, and those associated with nodes in the $\mathcal{T}$ set take the value zero. With slight abuse of notation, we use the variables to denote nodes in our graph. For each lower linear envelope potential edges are added as follows: for each $i \in c$, add an edge from y_i to t with weight $a_1 w_i^c$; for each $i \in c$ and $k = 1, \ldots, K-1$, add an edge from z_k to y_i with weight $(a_k - a_{k+1}) w_i^c$; and for $k = 1, \ldots, K-1$, add an edge from s to z_k with weight $a_k - a_{k+1}$ and edge from z_k to t with weight $b_{k+1} - b_k$. Other edges may be required to represent unary and pairwise potentials (see [23]).

Proof: By Proposition 3.3 we have $\operatorname{argmin}_{\boldsymbol{y}} E(\boldsymbol{y}) = \operatorname{argmin}_{\boldsymbol{y}} (E^0(\boldsymbol{y}) + \sum_c \min_{\boldsymbol{z}_c} E^c(\boldsymbol{y}_c, \boldsymbol{z}_c))$. By Proposition 3.7 we have that the $E^c(\boldsymbol{y}_c, \boldsymbol{z}_c)$ are submodular. The sum of submodular energy functions is submodular. Each higher-order term adds $K-1$ auxiliary variables so the total number of variables in the augmented energy function is less than n plus the total number of linear functions. □

3.3 Relationship to Binary MRFs

From a graphical models perspective, we note that $E^c(\boldsymbol{y}_c, \boldsymbol{z}_c)$ is nothing more than a pairwise binary Markov random field (MRF). Evidently, we can express Equation 7 as

$$E^c(\boldsymbol{y}_c, \boldsymbol{z}) = \text{const.} + \sum_{i \in c} \psi_i^Y(y_i) + \sum_{k=1}^{K-1} \psi_k^Z(z_k) + \sum_{(i,k)} \psi_{ik}^P(y_i, z_k) \quad (8)$$

where, for example, $\psi_k^Z(z_k) = (b_{k+1} - b_k)$ if $z_k = 1$ and $(a_k - a_{k+1})$ otherwise. For brevity, we omit details of the remaining potential functions, which can be trivially constructed by considering the corresponding unary and pairwise terms in y_i and z_k between the two forms.

4 Learning the Lower Linear Envelope

We now show how the max-margin framework can be used to learn parameters of our weighted lower linear envelope potentials. For simplicity of exposition we consider a single higher-order term $\psi_c^H(\boldsymbol{y}_c)$ and drop the subscript c for brevity. The extension to multiple higher-order terms defined over different subsets of variables is straightforward.

We begin by reviewing a variant of the max-margin framework introduced by Tsochantaridis et al. [37] and Taskar et al. [36]. We then show how alternative representations of the weighted lower linear envelope potential can be learned using the framework.

4.1 Max-margin Learning

Given an energy function $E(\boldsymbol{y}; \boldsymbol{\theta}) = \boldsymbol{\theta}^T \phi(\boldsymbol{y})$ parameterized as a linear combination of features $\phi(\boldsymbol{y}) \in \mathbb{R}^m$ and weights $\boldsymbol{\theta} \in \mathbb{R}^m$, and a set of T training examples $\{\boldsymbol{y}_t\}_{t=1}^T$ the max-margin framework is a principled approach to learning the weights of the model.

In our formulation we will allow additional linear constraints to be imposed on the weights of the form $\boldsymbol{G\theta} \geq \boldsymbol{h}$, where $\boldsymbol{G} \in \mathbb{R}^{d \times m}$ and $\boldsymbol{h} \in \mathbb{R}^d$. This is not typically necessary for max-margin learning, but, as we will see below, is required for enforcing concavity when learning lower linear envelope potentials.

Now, let $\mathcal{Y}_t = \{0,1\}^n$ be the set of all possible assignments for the t-th training example. The (margin-rescaling) max-margin approach formulates learning as a quadratic programming optimization problem, $\textsc{MaxMarginQP}(\{\boldsymbol{y}_t, \mathcal{Y}_t\}_{t=1}^T, \boldsymbol{G}, \boldsymbol{h})$:

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}\|\boldsymbol{\theta}\|^2 + \tfrac{C}{T} \textstyle\sum_{t=1}^T \xi_t && (9) \\ &\text{subject to} \\ &\quad \boldsymbol{\theta}^T \delta\phi_t(\boldsymbol{y}) + \xi_t \geq \Delta(\boldsymbol{y}, \boldsymbol{y}_t), && \forall t, \boldsymbol{y} \in \mathcal{Y}_t, \\ &\quad \xi_t \geq 0, && \forall t, \\ &\quad \boldsymbol{G\theta} \geq \boldsymbol{h} \end{aligned}$$

where $\delta\phi_t(\boldsymbol{y}) \triangleq (\phi_t(\boldsymbol{y}) - \phi_t(\boldsymbol{y}_t))$ is the difference between feature representations for some assignment $\boldsymbol{y}$ and the t-th ground-truth assignment $\boldsymbol{y}_t$, $C > 0$ is a regularization constant, and $\Delta(\boldsymbol{y}, \boldsymbol{y}_t)$ measures the loss between a ground-truth assignment $\boldsymbol{y}_t$ and any other assignment. In our work we use the Hamming loss, which measures the proportion of variables whose corresponding assignments disagree. More formally, the Hamming loss is defined as $\Delta(\boldsymbol{y}, \boldsymbol{y}') = \frac{1}{n} \sum_{i=1}^n [\![y_i \neq y_i']\!]$, where $[\![P]\!]$ is the indicator function taking value one when P is true and zero otherwise.

The number of constraints in the QP is exponential in the number of variables, and a standard approach to solving the max-margin QP is by adding constraints incrementally. Briefly, at each iteration the algorithm checks for the most violated constraint (for each training example), using *loss-augmented inference*, and, if found, adds it to the constraint set. The algorithm terminates

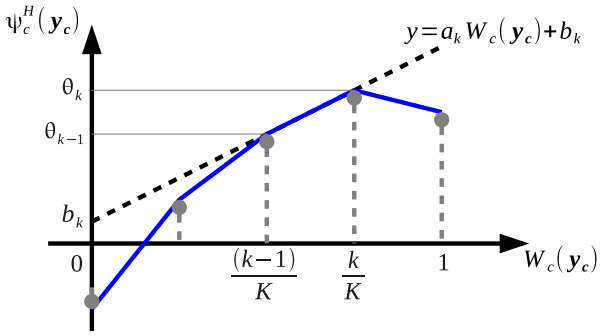


Fig. 3: Example piecewise-linear concave function of $W_c(\boldsymbol{y}_c) = \sum_{i \in c} w_i^c y_i$. The function can be represented as the minimum over a set of linear functions (lower linear envelope) or as a set of sampled points θ_k with curvature constraint.

when no more violated constraints are found (see Algorithm 1).

Note that while we use the Hamming loss in this work, the loss function $\Delta(\boldsymbol{y}, \boldsymbol{y}_t)$ in Equation 9 can be more general. For example, Pletscher and Kohli [31] recently showed that certain higher-order losses can be reduced to binary pairwise supermodular functions. In this way the loss function factors over the same terms as in the energy function with the addition of auxiliary variables. Since the loss function is subtracted from the energy function during loss-augmented inference, the supermodular loss becomes a submodular objective and therefore admits tractable minimization.

4.2 Transforming Between Representations

The max-margin formulation (see Equation 9) requires that the energy function be expressed as a linear combination of features and weights, however, our higher-order potential is represented as the minimum over a set of linear functions. One simple way to re-parameterize the energy function for learning is to sample the higher-order potential at regular intervals between zero and one.[3] This provides a piecewise linear approximation of the weighted lower linear envelope and the number of points sampled lets us trade-off tightness of the approximation with efficiency of inference. Let $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_K) \in \mathbb{R}^{K+1}$ be the sampled values. Then, we can retrieve the equivalent weighted lower linear envelope representation as

$$a_k = (\theta_k - \theta_{k-1})K \tag{10}$$

$$b_k = \theta_k - a_k \frac{k}{K} = k\theta_{k-1} - (k-1)\theta_k \tag{11}$$

for $k = 1, \ldots, K$ as illustrated in Figure 3.[4] The corresponding feature vector $\phi(\boldsymbol{y}) = (\phi_0, \ldots, \phi_K) \in \mathbb{R}^{K+1}$, under this representation, is a $(K+1)$-length vector with n-th entry

$$\phi_{n-1} = \begin{cases} W(\boldsymbol{y}) \cdot K - n + 2 & \text{if } \frac{n-2}{K} \leq W(\boldsymbol{y}) < \frac{n-1}{K} \\ n - W(\boldsymbol{y}) \cdot K & \text{if } \frac{n-1}{K} \leq W(\boldsymbol{y}) < \frac{n}{K} \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

so that $\boldsymbol{\theta}^T \phi(\boldsymbol{y})$ linearly interpolates between the samples corresponding to the active linear function (see Figure 4). For example, assume $K = 3$ and $W(\boldsymbol{y}) = \frac{1}{2} + \epsilon$ then $\phi(\boldsymbol{y}) = (0, \frac{1}{2} - 3\epsilon, \frac{1}{2} + 3\epsilon, 0)$. Note that $\mathbf{1}^T \phi(\boldsymbol{y}) = 1$. This representation is independent of clique size and so can be used without modification for applications where clique size varies between instantiations of the higher-order potentials, e.g., when cliques are derived from superpixels generated from an over-segmentation algorithm.

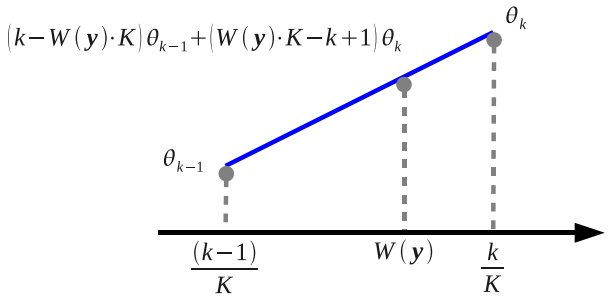


Fig. 4: Illustration of how the feature vector $\phi(\boldsymbol{y})$ interpolates between samples to produce the correct value for the active linear function.

It is instructive to observe that under this representation, with weighted lower linear envelope sampled at uniform intervals between 0 and 1, we can compute $k^\star = \text{argmin}_{k=1,\ldots,K} \{a_k W(\boldsymbol{y}) + b_k\}$ and hence $\boldsymbol{z}$ in closed-form as

$$k^\star = \begin{cases} 1 & \text{if } W(\boldsymbol{y}) = 0 \\ \lceil W(\boldsymbol{y}) \cdot K \rceil & \text{otherwise} \end{cases} \tag{13}$$

which is the key insight behind the feature representation in Equation 12.

We now have a representation of our higher-order potentials which is linear in the parameters $\boldsymbol{\theta}$. It remains to ensure that $\boldsymbol{\theta}$ represents a concave function. We do this by adding the second-order curvature constraint $\boldsymbol{D}^2\boldsymbol{\theta} \geq \mathbf{0}$ where $\boldsymbol{D}^2 \in \mathbb{R}^{(K-1)\times(K+1)}$ is the (negative) discrete second-derivative operator:

$$\boldsymbol{D}^2 = \begin{bmatrix} -1 & 2 & -1 & 0 & \cdots \\ & & \ddots & & \\ \cdots & 0 & -1 & 2 & -1 \end{bmatrix}. \tag{14}$$

Our optimization follows the standard max-margin approach and is summarized in Algorithm 1.[5]

3. Recall from Section 3 that we have assumed, without loss of generality, that $\sum_{i \in c} w_i^c = 1$.

4. Note that if $a_k = a_{k-1}$ then the k-th linear function is redundant and can be omitted from the energy function.

5. To jointly learn the unary and pairwise weights, we augment the parameter vector $\boldsymbol{\theta}$ with a weight θ^{unary} for the unary terms and non-negative weight θ^{pair} for the pairwise terms, and add the corresponding features $\phi^{\text{unary}} = \sum_i \psi_i^U(y_i)$ and $\phi^{\text{pair}} = \sum_{ij} \psi_{ij}^P(y_i, y_j)$ to the feature vector $\phi(\boldsymbol{y})$. The non-negativity of θ^{pair} ensures that the energy function remains submodular.

Algorithm 1 Learning lower linear envelope MRFs.

1: **input** training set $\{\boldsymbol{y}_t\}_{t=1}^T$, regularization constant $C > 0$, and tolerance $\epsilon \geq 0$
2: **initialize** active constraints set $\mathcal{A}_t = \{\}$ for all t
3: **repeat**
4: solve $\textsc{MaxMarginQP}\big(\{\boldsymbol{y}_t, \mathcal{A}_t\}_{t=1}^T, \boldsymbol{D}^2, \boldsymbol{0}\big)$ to get $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\xi}}$
5: convert from $\hat{\boldsymbol{\theta}}$ to (a_k, b_k) representation
6: **for** each training example, $t = 1, \ldots, T$ **do**
7: compute $\boldsymbol{y}_t^\star = \operatorname{argmin}_{\boldsymbol{y}} E(\boldsymbol{y}; \hat{\boldsymbol{\theta}}) - \Delta(\boldsymbol{y}, \boldsymbol{y}_t)$
8: **if** $\hat{\xi}_t + \epsilon < \Delta(\boldsymbol{y}_t^\star, \boldsymbol{y}_t) - E(\boldsymbol{y}_t^\star; \hat{\boldsymbol{\theta}}) + E(\boldsymbol{y}_t; \hat{\boldsymbol{\theta}})$ **then**
9: $\mathcal{A}_t \leftarrow \mathcal{A}_t \cup \{\boldsymbol{y}_t^\star\}$
10: **end if**
11: **end for**
12: **until** no more violated constraints
13: **return** parameters $\hat{\boldsymbol{\theta}}$

Theorem 4.1. : For the setting $\epsilon = 0$, Algorithm 1 terminates with the optimal parameters $\boldsymbol{\theta}^\star$ for $\textsc{MaxMarginQP}\big(\{\boldsymbol{y}_t, \mathcal{Y}_t\}_{t=1}^T, \boldsymbol{D}^2, \boldsymbol{0}\big)$.

Proof: By Theorem 3.8, our test for the most violated constraints (lines 7 and 8) can be performed exactly ($\Delta(\boldsymbol{y}, \boldsymbol{y}_t)$ decomposes as a sum of unary terms). If the test succeeds, then $\boldsymbol{y}_t^\star$ cannot already be in $\mathcal{A}_t$. It is now added (line 9). Since there are only finitely many constraints, this happens at most $2^n - 1$ times (per training example), and the algorithm must eventually terminate. On termination there are no more violated constraints, hence the parameters are optimal. □

Unfortunately, as our proof suggests, it may take exponential time for the algorithm to reach convergence with $\epsilon = 0$. Tsochantaridis et al. [38] showed, however, that for $\epsilon > 0$ and no additional linear constraints (i.e., $\boldsymbol{G} = \boldsymbol{0}$, $\boldsymbol{h} = \boldsymbol{0}$) max-margin learning within a dual optimization framework will terminate in a polynomial number of iterations. Their result can be extended to the case of additional linear constraints (see the Appendix for details).

4.3 Alternative QP Formulations

Our quadratic program above is just one possible formulation that is based on a particular choice for representing the weighted lower linear envelope and corresponding feature vectors. An alternative representation may encode the slope of the weighted lower linear envelope directly, that is,

$$\theta'_i = \begin{cases} b_1 & \text{for } i = 0 \\ a_i = \theta_i - \theta_{i-1} & \text{for } i = 1, \ldots, K \end{cases} \tag{15}$$

The i-th component in the corresponding feature vector is then $\phi'_i = \sum_{j \geq i} \phi_j$. And instead of a second-order constraint $\boldsymbol{D}^2\boldsymbol{\theta}' \geq \boldsymbol{0}$, we have a first-order constraint $\boldsymbol{D}\boldsymbol{\theta}' \geq \boldsymbol{0}$. Here we can retrieve the b_k recursively as

$$b_{k+1} = (a_k - a_{k+1})\frac{k}{K} + b_k \tag{16}$$

One of the advantages of this formulation is that the regularization term does not penalize flat envelopes (i.e., $a_k = 0$). Moreover, it is interesting to note that under this formulation the optimal θ'_0 is always zero, i.e., $b_1 = 0$, which is not surprising in light of Observation 3.2.

We can take this process one step further and represent the higher-order potential as

$$\theta''_k = \begin{cases} b_1 & \text{for } k = 0 \\ a_1 & \text{for } k = 1 \\ a_{k-1} - a_k & \text{for } k = 2, \ldots, K \end{cases} \tag{17}$$

with non-negativity constraints on θ''_k for $k = 2, \ldots, K$, and appropriate feature vectors, i.e.,

$$\phi''_k = \begin{cases} 1 & \text{if } k = 0 \\ W(\boldsymbol{y}) & \text{if } k = 1 \\ \Big(\frac{(k-1)}{K} - W(\boldsymbol{y})\Big) \Big[\!\Big[W(\boldsymbol{y}) > \frac{k-1}{K} \Big]\!\Big] & k = 2, \ldots, K \end{cases} \tag{18}$$

Here we are encoding the coefficients of the pseudo-Boolean function used during inference directly into the learning problem. Like the previous formulation we know that the optimal b_1 is zero so can simply drop θ_0 and ϕ_0 from the optimization.

It is interesting to note the resemblance of the latter QP formulation with latent-variable structural SVM learning [39]. In our formulation the auxiliary variables $\boldsymbol{z}$ (see Section 3.2) can be determined directly from the ground-truth or inferred labels $\boldsymbol{y}$. Moreover, since we have fixed the piecewise-linear approximation to have equally spaced break-points, the auxiliary variables are independent of the parameters (a_k, b_k) given $\boldsymbol{y}$. We also have that the b_k are a function of the a_k (by Equation 16). Removing the restriction of equally spaced break-points (and introducing the b_k into the optimization) results in a latent-variable SVM. The main difficulty is that the latent variables $\boldsymbol{z}$ now depend on the parameters making the optimization problem non-convex.

A number of other variants can be considered by linearly constraining $\boldsymbol{\theta}$ (or alternatively re-defining $\phi(\boldsymbol{y})$). For example, the parameters of the P^n-model can be learned by constraining $\theta_0 \leq \theta_1$ and forcing $\theta_i = \theta_{i-1}$ for $i = 2, \ldots, K-1$. Although this case is somewhat uninteresting as there is only one parameter to learn (since by Observation 3.2 we can set $\theta_1 = \ldots = \theta_K = 0$ without changing the shape of the potential function), which can often be done more efficiently by other means, e.g., cross-validation over a range of values.

5 Experimental Results

We conduct experiments on synthetic and real-world data, comparing baseline MRF models with ones that include higher-order terms learned by our method.

5.1 Synthetic Checkerboard

Our synthetic experiments are designed to explore the different QP formulations for learning the lower linear

envelope model parameters and provide intuition into the real-world experiments that follow. They involve an 8×8 checkerboard pattern of alternating white ($y_i = 1$) and black ($y_i = 0$) squares. Each square contains 256 pixels. We associate one variable in the model with each pixel giving our MRF a total of $8 \times 8 \times 256 = 16,384$ variables. We generate a noisy version of the checkerboard as input by the following method. Let $\boldsymbol{y}^\star$ be the ground-truth checkerboard, then our input is generated as $x_i = \eta_0 [\![y_i^\star = 0]\!] - \eta_1 [\![y_i^\star = 1]\!] + \delta_i$ where η_0 and η_1 are the signal-to-noise ratios for the black and white squares, respectively, and $\delta_i \sim \mathcal{U}(-1, 1)$ is additive i.i.d. uniform noise. Our task is to recover the checkerboard pattern from the noisy input.

We consider three difference MRF models involving: (i) unary and pairwise terms, (ii) unary and higher-order terms, and (iii) unary, pairwise, and higher-order terms. Our unary terms are constructed for each pixel as $\psi_i^U(y_i) = \theta^{\text{unary}} x_i$ where θ^{unary} is an arbitrary weight. The pairwise terms take the form $\psi_{ij}^P(y_i, y_j) = \theta^{\text{pair}} [\![y_i \neq y_j]\!]$, where i and j are neighbouring pixels, and $\theta^{\text{pair}} \geq 0$ weights the strength of the pairwise term relative to the unary term.

For models including higher-order terms, we add one lower linear envelope potential term $\psi_c^H(\boldsymbol{y}_c) = \min_{k=1,\ldots,K} \left\{ a_k \sum_{i \in c} \frac{1}{|c|} y_i + b_k \right\}$ for each square in the checkerboard, so each higher-order potential contains 256 variables and the terms are disjoint. Intuitively, we would like the potential to favour label consistency within the square. We learn θ^{unary}, θ^{pair} (when included), and $\{(a_k, b_k)\}_{k=1}^K$ for $K = 10$ linear functions using Algorithm 1. For the baseline model with unary and pairwise terms we set $\theta^{\text{unary}} = 1$ and choose θ^{pair} to give best the Hamming loss by evaluating 101 uniformly spaced values in the range $[0, 1]$.

We report results on two different problem instances. The first has symmetric signal-to-noise ratios $\eta_0 = \eta_1 = 0.1$, and the second has five times less noise on the black squares ($\eta_0 = 0.5$) than on the white ($\eta_1 = 0.1$). Figure 5 shows the ground-truth checkerboard patterns and the noisy input. For both instances we set $C = 1000$ in Equation 9. Learning is run to convergence, taking 27 iterations for the first instance and 19 iterations for the second instance on the model with unary, pairwise and higher-order terms. Each training iteration took under 1s with inference taking about 120ms on a 2.66GHz quad-core Intel CPU.

Figure 5 shows the inferred checkerboard patterns for the pairwise MRF baseline, and for our higher-order model after the third and after the final training iterations ((c), (d), and (e), respectively). We see that after just three iterations our higher-order model is already performing well on both problem instances, and by the final iteration we can perfectly recover the checkerboard unlike the pairwise model. This is not surprising given that our higher-order cliques are defined on the checkerboard squares. Below we run further experiments with misspecified cliques.

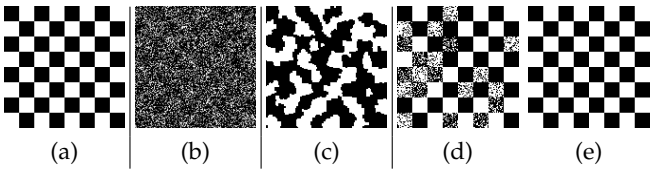


Fig. 5: Inferred output from our synthetic experiments. Shown are (a) the ground-truth labels, (b) noisy inputs, (c) best inferred labels using a pairwise model, (d)-(e) inferred output from the model containing higher-order terms after three training iterations and at convergence, respectively. Matlab source code for reproducing these results is available from the author's homepage.

We also compared the shape of the learned lower linear envelope for different problem instances comparing the different QP formulations (as described in Section 4.3). All formulations were able to learn parameters that perfectly reconstruct the known checkerboard pattern. The learned linear envelope parameters (relative to the unary weight) are shown in Figure 6. Note that for the second instance (with asymmetric noise), our algorithm is able to learn an asymmetric potential.

Next we evaluate our algorithm on synthetic data with partially overlapping as well as misspecified cliques. Here we generate multiple cliques (between one and five) for each checkerboard square then randomly remove between 5% and 50% of the pixels from each clique. We also introduce a number of completely misspecified cliques composed of pixels chosen at random from anywhere in the image—on expectation half the pixels in these cliques have ground-truth label one and half have ground-truth label zero. We learn a model with unary and higher-order terms only.

Inferred checkerboard patterns are shown in Figure 7. Increasing from left to right is the percentage of pixels randomly removed from the cliques (i.e., reduction in clique size). Increasing from top to bottom is the number of overlapping cliques. As expected our model performs poorly when many of the pixels are not covered by a higher-order clique, for example in Figure 7(i)(d). Multiple partially overlapping cliques addresses this problem. Moreover, our method is robust to a reasonable number of misspecified cliques (10% in the case of the results shown in Figure 7).

5.2 Interactive Figure-Ground Segmentation

We also ran experiments on the real-world "GrabCut" problem introduced by Rother et al. [32]. Here the aim is to segment a foreground object from an image given a user-annotated bounding box of the object (see Figure 8(a) for some examples). To solve this problem the GrabCut algorithm associates a binary random variable y_i with each pixel in the image indicating whether the pixel belongs to the "background" (binary label 0) or the "foreground" (binary label 1). Variables corresponding to pixels outside of the user-annotated bounding box are

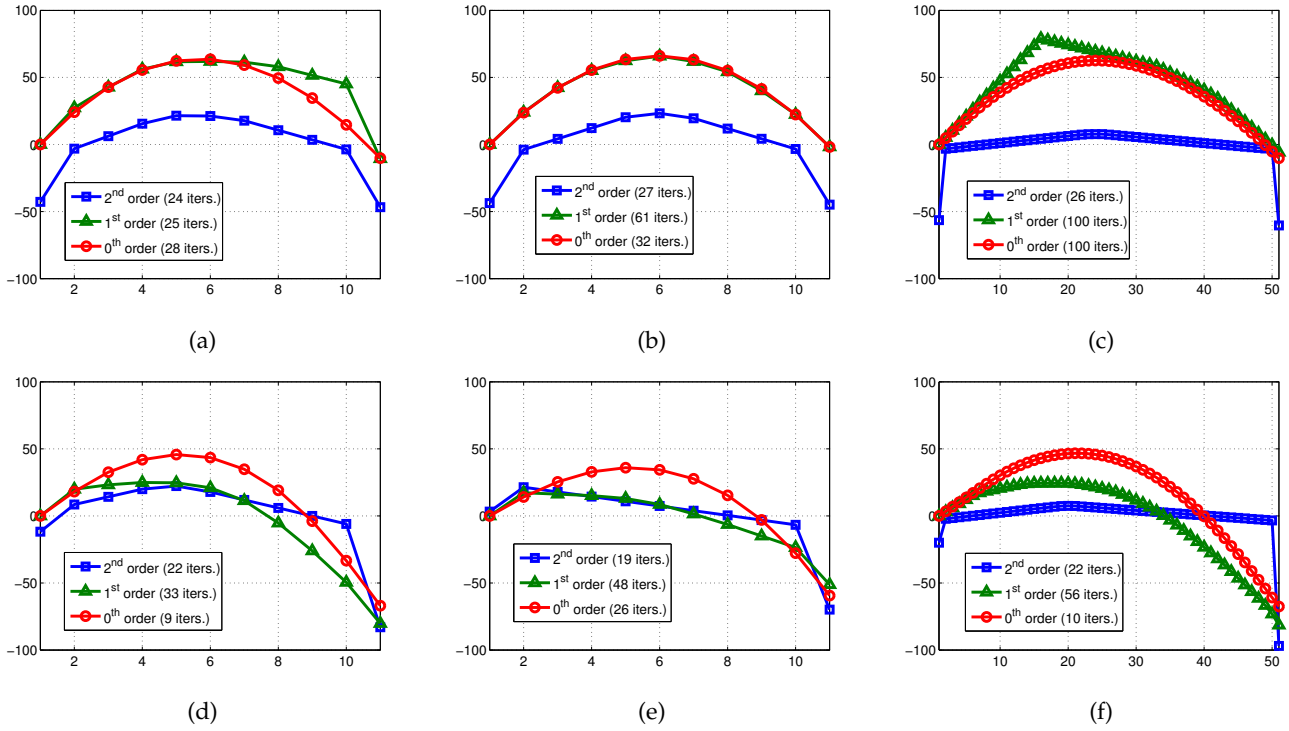


Fig. 6: Learned linear envelopes (parameters are normalized by the unary weight) for synthetic experiments. The first row (a)-(c) shows results with symmetric noise ($\eta_0 = \eta_1 = 0.1$) while the second row (d)-(f) shows results with asymmetric noise ($\eta_0 = 0.5$ and $\eta_1 = 0.1$). Compared are models with unary and higher-order potentials with $K = 10$ linear terms ((a) and (d)), unary, pairwise and higher-order potentials ((b) and (e)), and unary and higher-order potentials with $K = 50$ linear terms ((c) and (f)).

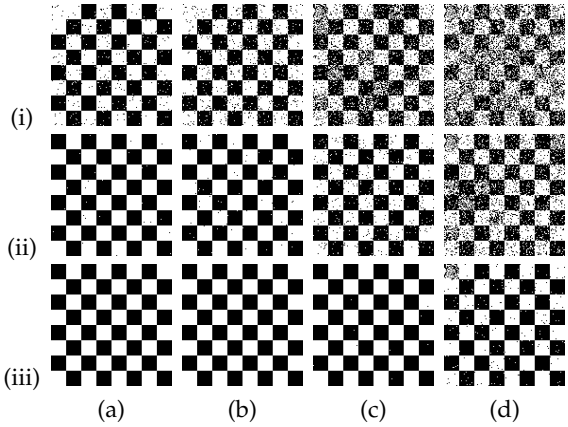


Fig. 7: Inferred output from our synthetic experiments with misspecified cliques. Shown are inferred outputs from the model at convergence. Rows (i)–(ii) correspond to partially covering each grid square with 1, 2 and 5 higher-order cliques, respectively. Columns (a)-(d) correspond to the size of each clique (95%, 90%, 75%, 50% grid square coverage, respectively). In addition, 10% of the cliques were generated to contain random a random mix of pixels.

automatically assigned a label of zero (i.e., background). The assignment for the remaining variables, i.e., those within the bounding box, is inferred.

We compare a model with learned higher-order terms against the baseline GrabCut model by performing leave-one-out cross-validation on a standard set of 50 images from Lempitsky et al. [26]. Following the approach of Rother et al. [32], our baseline model contains unary and pairwise terms. The unary terms are defined as the log-likelihood from foreground and background Gaussian mixture models (GMMs) over pixel colour and are image-specific. Briefly, the GMMs are initialized by learning foreground and background models from pixels inside and outside the user-annotated bounding box, respectively. Next, the GMMs are used to relabel pixels (within the bounding box) as either foreground or background by taking the label with highest likelihood according to the current parameter settings. Next the parameters of the foreground and background colour models are re-estimated given the new labeling. This process of parameter estimation and re-labeling is repeated until convergence (or a maximum number of iterations is reached). The final GMMs are used to construct the unary terms.

The pairwise terms encode smoothness between each pixel and its eight neighbours, and are defined as

$$\psi_{ij}^{P}(y_i, y_j) = \frac{\lambda}{d_{ij}} [\![y_i \neq y_j]\!] \exp\left\{ -\frac{\|x_i - x_j\|^2}{2\beta} \right\} \qquad (19)$$

where d_{ij} is the distance between pixels i and j, x_i and x_j are the RGB colour vectors for pixels i and j, β is the average squared-distance between adjacent RGB colour vectors in the image, and λ determines the strength of the pairwise smoothness term. It is the only free

Fig. 8: Example results from our GrabCut experiments. Shown are: (a) the image and bounding box, (b) ground-truth segmentation, (c) baseline model output, and (d) output from model with higher-order terms.

MODEL	ACCURACY
Baseline	90.9
Higher-Order (1 Seg.)	91.4
Higher-Order (2 Segs.)	**91.6**
Higher-Order (3 Segs.)	91.2

TABLE 1: Results from our Weizmann Horse experiments. The baseline model includes unary and pairwise terms. The higher-order model includes unary, pairwise and lower-linear envelope terms defined by multiple over-segmentations. See text for details.

parameter in the baseline model and learned by cross-validation.

To construct the higher-order terms, we adopt a similar superpixel-based approach to Ladicky et al. [25]. First, we over-segment the image into a few hundred superpixels. Here we use the mean-shift segmentation algorithm of Comaniciu and Meer [8] but our method does not depend on this choice. The pixels within each superpixel then define a higher-order term, much like the checkerboard squares in our synthetic experiments. Here, however, the higher-order terms are over different sized cliques and there is no guarantee that they should be labeled homogeneously.

We learn the weights for the unary and pairwise potentials and the parameters for a lower linear envelope potential with $K = 10$ terms using Algorithm 1. We set $C = 1000$ and ran for a maximum of 100 iterations, however, for most cross-validation folds, the algorithm converged before the maximum number of iterations was reached. The parameters determined at the last iteration were used for testing. Learning took approximately 3 hours per cross-validation fold with the majority of the time spent generating violated constraints for the 49 training images (each typically containing 640×480 pixels).

Some example results are shown in Figure 8. The first row shows that our higher-order terms can capture some fine structure such as the cheetah's tail but it also segments part of the similarly-appearing rock. In the second example, we are able to correctly segment the person's legs. The third example shows that we are able to segment the petals at the lower part of the rightmost flower, which the baseline model does not. The final example (fourth row) shows that our model is able to remove background regions that have similar appearance to the foreground. However, we can also make mistakes such as parts of the sculpture's robe. Quantitatively, our method achieves 91.5% accuracy compared to 90.0% for the strong baseline.

5.3 Weizmann Horses

We also ran experiments on the 328-image Weizmann Horse dataset [2, 3]. The task here is a supervised machine learning one with the goal being to segment horses from the background in unseen images. As with the previous experiments, our model consists of unary, pairwise and higher-order lower linear envelope terms.

We divide the dataset into three subsets of size 100, 64 and 164. The first subset of 100 images is used to learn a classifier for predicting horse pixels from local features. The second subset of 64 images is used to learn the weights for the unary and pairwise terms, and the parameters of the higher-order lower linear envelope potentials. The final subset of 164 images is used for testing.

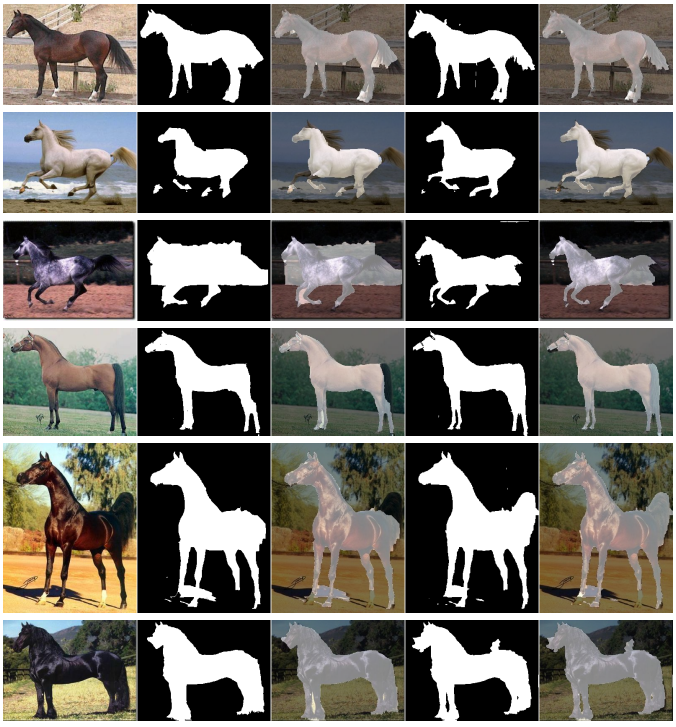
Concretely, our unary terms contain the log-probability from a learned boosted decision tree classifier, which estimates the probability of each pixel belonging to a horse given colour and texture features surrounding the pixel. We use the 17-dimensional "texton" filterbank of Shotton et al. [34] for describing texture.

Again for the higher-order terms we over-segment the images using the mean-shift segmentation algorithm [8] to produce superpixels. However, instead of a single over-segmentation we compute multiple over-segmentations by varying the spatial and colour bandwidth parameters of the mean-shift algorithm. Superpixels containing less than 64 pixels are discarded. The remaining set of (overlapping) superpixels are used to define cliques for the higher-order terms with $K = 10$ linear functions.

Training the boosted classifier on the colour and texture features for the unary potentials took approximately 7 minutes on a 2.66GHz quadcore Intel CPU. Cross-validating the strength of the pairwise term for the model without higher-order terms took a further 15 minutes. When training the model with higher-order terms we learn all parameters simultaneously. This took approximately 6 hours with the bulk of the time spent running loss-augmented inference.

We compare a baseline model with unary and pairwise terms against a model that also includes the lower linear envelope potentials. Results showing average pixel accuracy over the set of test images are shown in Table 1. Once again the baseline model is very strong but our method with higher-order terms is able to achieve a slight improvement.

Example horse/background segmentation results are shown in Figure 9. Qualitatively our method performs better than the baseline on regions where the contrast between the foreground horse and background is low (e.g., in the third row of Figure 9). This is not surprising when we consider that low contrast boundaries are exactly where the pairwise smoothness term is expected to perform poorly.



(a) (b) (c) (d) (e)

Fig. 9: Example segmentations produced by our Weizmann Horse experiments. Shown are: (a) the image, (b) baseline foreground mask, (c) baseline model foreground overlay, (d) higher-order model foreground mask, and (e) higher-order model foreground overlay.

6 Discussion

This paper has shown how to perform efficient inference and learning for lower linear envelope binary MRFs, which are becoming popular for enforcing higher-order consistency constraints over large sets of random variables, particularly in computer vision.

Our formulation allows arbitrary non-negative weights to be assigned to each variable in the higher-order term. These weights allow different size cliques to share the same higher-order parameters (resulting in the same shape lower linear envelope). In addition, the weights can be used to place more importance on some variables than others, e.g., pixels further away from the boundary of a superpixel.

Our work suggests a number of directions for future research. Perhaps the most obvious is extending our approach to multi-label MRFs. An initial exploration of this extension was done by Park and Gould [30] using our inference method within the inner loop of move-making algorithms such as α-expansion or $\alpha\beta$-swap [6] for generating constraints. However, the question of efficient learning remains open since inference in this regime is only approximate.

Other straightforward extensions include the introduction of features for modulating the higher-order terms and the use of dynamic graph cuts [18] for accelerating loss-augmented inference within our learning framework. We could also consider other optimization schemes for solving our learning problem, e.g., dual-decomposition [24] or the subgradient method [1, 28].

More interesting is the implicit relationship between structured higher-order models and latent-variable SVMs [39] as suggested by the introduction of auxiliary variables for inference and our alternative QP formulations. Exploring this relationship further may provide insights into both models.

From an application perspective, we hope that the ability to efficiently learn higher-order potentials from data will encourage researchers to more readily adopt these more expressive models for their applications.

Acknowledgments

This research was supported under Australian Research Council's *Discovery Projects* funding scheme (project number DP110103819).

References

[1] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2004.

[2] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2002.

[3] E. Borenstein, E. Sharon, and S. Ullman. Combining top-down and bottom-up segmentation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[4] E. Boros and P. L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123:155–225, 2002.

[5] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 26:1124–1137, 2004.

[6] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *Proc. of the International Conference on Computer Vision (ICCV)*, 1999.

[7] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2012.

[8] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 24: 603–619, 2002.

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results, 2012.

[10] T. Finley and T. Joachims. Training structural SVMs when exact inference is intractable. In *Proc. of the International Conference on Machine Learning (ICML)*, 2008.

[11] D. Freedman and P. Drineas. Energy minimization via graph cuts: Settling what is possible. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[12] S. Gould. Max-margin learning for lower linear envelope potentials in binary Markov random fields. In *Proc. of the International Conference on Machine Learning (ICML)*, 2011.

[13] P. L. Hammer. Some network flow problems solved with psuedo-boolean programming. *Operations Research*, 13:388–399, 1965.

[14] H. Ishikawa. Exact optimization for Markov random fields with convex priors. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 25: 1333–1336, 2003.

[15] H. Ishikawa. Higher-order clique reduction in binary graph cut. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[16] T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77: 27–59, 2009.

[17] P. Kohli and M. P. Kumar. Energy minimization for linear envelope MRFs. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[18] P. Kohli and P. H. S. Torr. Dynamic graph cuts for efficient inference in markov random fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2007.

[19] P. Kohli, M. P. Kumar, and P. H. S. Torr. P3 & beyond: Solving energies with higher order cliques. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[20] P. Kohli, L. Ladicky, and P. H. S. Torr. Graph cuts for minimizing higher order potentials. Technical report, Microsoft Research, 2008.

[21] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[22] V. Kolmogorov. Minimizing a sum of submodular functions. *Discrete Applied Mathematics*, 160(14): 2246–2258, Oct 2012.

[23] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 26:65–81, 2004.

[24] N. Komodakis. Efficient training for pairwise or higher order crfs via dual decomposition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[25] L. Ladicky, C. Russell, P. Kohli, and P. H. Torr. Associative hierarchical CRFs for object class image segmentation. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2009.

[26] V. Lempitsky, P. Kohli, C. Rother, and T. Sharp. Image segmentation with a bounding box prior. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2009.

[27] S. Nowozin and C. H. Lampert. Global connectivity potentials for random field models. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[28] S. Nowozin and C. H. Lampert. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6(3–4):185–365, 2011.

[29] J. B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251, 2009.

[30] K. Park and S. Gould. On learning higher-order consistency potentials for multi-class pixel labeling. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2012.

[31] P. Pletscher and P. Kohli. Learning low-order models for enforcing high-order statistics. In *Proc. of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

[32] C. Rother, V. Kolmogorov, and A. Blake. GrabCut: Interactive foreground extraction using iterated graph cuts. In *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2004.

[33] C. Rother, P. Kohli, W. Feng, and J. Jia. Minimizing sparse higher order energy functions of discrete variables. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[34] J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2006.

[35] M. Szummer, P. Kohli, and D. Hoiem. Learning CRFs using graph-cuts. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2008.

[36] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *Proc. of the International Conference on Machine Learning (ICML)*, 2005.

[37] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector learning for interdependent and structured output spaces. In *Proc. of the International Conference on Machine Learning (ICML)*, 2004.

[38] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6:1453–1484, 2005.

[39] C.-N. Yu and T. Joachims. Learning structural SVMs with latent variables. In *Proc. of the International Conference on Machine Learning (ICML)*, 2009.

Stephen Gould is a Fellow in the Research School of Computer Science in the College of Engineering and Computer Science at the Australian National University. He received his BSc degree in mathematics and computer science and BE degree in electrical engineering from the University of Sydney in 1994 and 1996, respectively. He received his MS degree in electrical engineering from Stanford University in 1998. He then worked in industry for a number of years before returning to PhD studies in 2005. He earned his PhD degree from Stanford University in 2010. His research interests are in computer and robotic vision, machine learning, probabilistic graphical models, and optimization. He is a member of the IEEE.

APPENDIX

In this section we show that the polynomial time cutting-plane method of Tsochantaridis et al. [38] can be extended to handle linear inequality constraints on the parameters. Our argument follows their $\text{SVM}_1^{\Delta m}$ formulation of the max-margin structured prediction problem.

Let us begin by writing out the Lagrangian for the quadratic program $\textsc{MaxMarginQP}\big(\{\boldsymbol{y}_t, \mathcal{Y}_t\}_{t=1}^T, \boldsymbol{G}, \boldsymbol{h}\big)$ defined in Equation 8. Introducing dual variables $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, we have

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \frac{1}{2}\|\boldsymbol{\theta}\|^2 + \frac{C}{T}\sum_{t=1}^T \xi_t \\ - \sum_{t=1}^T \sum_{\boldsymbol{y}\in\mathcal{Y}_t} \alpha_{t,\boldsymbol{y}} \Big(\boldsymbol{\theta}^T \delta\phi_t(\boldsymbol{y}) + \xi_t - \Delta(\boldsymbol{y}, \boldsymbol{y}_t)\Big) \\ - \boldsymbol{\beta}^T(\boldsymbol{G}\boldsymbol{\theta} - \boldsymbol{h}) - \sum_{t=1}^T \gamma_t \xi_t \quad (20)\end{aligned}$$

subject to $\boldsymbol{\alpha} \succeq \mathbf{0}$, $\boldsymbol{\beta} \succeq \mathbf{0}$, and $\boldsymbol{\gamma} \succeq \mathbf{0}$ where "$\boldsymbol{a} \succeq \boldsymbol{b}$" denotes componentwise inequality between the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

Setting $\frac{\partial \mathcal{L}}{\partial \xi_i} = \frac{C}{T} - \sum_{\boldsymbol{y}\in\mathcal{Y}_t} \alpha_{t,\boldsymbol{y}} - \gamma_t = 0$ and substituting for γ_t we can re-write Equation 20 as

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{\theta}\|^2 \\ - \sum_{t=1}^T \sum_{\boldsymbol{y}\in\mathcal{Y}_t} \alpha_{t,\boldsymbol{y}} \Big(\boldsymbol{\theta}^T \delta\phi_t(\boldsymbol{y}) - \Delta(\boldsymbol{y}, \boldsymbol{y}_t)\Big) \\ - \boldsymbol{\beta}^T(\boldsymbol{G}\boldsymbol{\theta} - \boldsymbol{h}) \quad (21)\end{aligned}$$

subject to constraints $\sum_{\boldsymbol{y}\in\mathcal{Y}_t} \alpha_{t,\boldsymbol{y}} \le \frac{C}{T}$ for all $t = 1, \ldots, T$.

Now

$$\nabla_{\boldsymbol{\theta}} \mathcal{L} = \boldsymbol{\theta} - \sum_{t=1}^T \sum_{\boldsymbol{y}\in\mathcal{Y}_t} \alpha_{t,\boldsymbol{y}} \delta\phi_t(\boldsymbol{y}) - \boldsymbol{G}^T\boldsymbol{\beta}. \quad (22)$$

Eliminating $\boldsymbol{\theta}$ by setting $\nabla_{\boldsymbol{\theta}}\mathcal{L} = 0$ we have

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = -\frac{1}{2}\sum_{\substack{t=1,\\ \boldsymbol{y}\in\mathcal{Y}_t}}^T \sum_{\substack{t'=1,\\ \boldsymbol{y}'\in\mathcal{Y}_t'}}^T \alpha_{t,\boldsymbol{y}}\alpha_{t',\boldsymbol{y}'} \delta\phi_t(\boldsymbol{y})^T \delta\phi_{t'}(\boldsymbol{y}') \\ - \sum_{\substack{t=1,\\ \boldsymbol{y}\in\mathcal{Y}_t}}^T \alpha_{t,\boldsymbol{y}} \delta\phi_t(\boldsymbol{y})^T \boldsymbol{G}^T\boldsymbol{\beta} - \frac{1}{2}\boldsymbol{\beta}^T\boldsymbol{G}\boldsymbol{G}^T\boldsymbol{\beta} \\ + \sum_{\substack{t=1,\\ \boldsymbol{y}\in\mathcal{Y}_t}}^T \alpha_{t,\boldsymbol{y}} \Delta(\boldsymbol{y}, \boldsymbol{y}_t) + \boldsymbol{h}^T\boldsymbol{\beta} \quad (23)\end{aligned}$$

where we have written the double summations over t and $\boldsymbol{y}$ more succinctly. This is a quadratic equation that can be written more compactly as

$$\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = -\frac{1}{2}\begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{bmatrix}^T \begin{bmatrix}\boldsymbol{J}_{\alpha\alpha} & \boldsymbol{J}_{\alpha\beta} \\ \boldsymbol{J}_{\beta\alpha} & \boldsymbol{G}\boldsymbol{G}^T\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{bmatrix} + \begin{bmatrix}\boldsymbol{\Delta}\\ \boldsymbol{h}\end{bmatrix}^T\begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{bmatrix} \quad (24)$$

where $\boldsymbol{\alpha}$ is a vector containing the $\alpha_{t,\boldsymbol{y}}$, $\boldsymbol{\Delta}$ is a vector with entries $\Delta(\boldsymbol{y}, \boldsymbol{y}_t)$ corresponding to the entries in $\boldsymbol{\alpha}$.

The dual optimization problem to $\textsc{MaxMarginQP}\big(\{\boldsymbol{y}_t, \mathcal{Y}_t\}_{t=1}^T, \boldsymbol{G}, \boldsymbol{h}\big)$ is to maximize $\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ subject to constraints $\boldsymbol{\alpha} \succeq \mathbf{0}$, $\boldsymbol{\beta} \succeq \mathbf{0}$, and $\sum_{\boldsymbol{y}\in\mathcal{Y}_t} \alpha_{t,\boldsymbol{y}} \le \frac{C}{T}$ for $t = 1, \ldots, T$.

Lemmas 10, 11, 12, and 13 from Tsochantaridis et al. [38] apply directly to Equation 23 on the joint variables $(\boldsymbol{\alpha}, \boldsymbol{\beta})$. Specifically, we have the following bounds

$$\begin{aligned}\max_{0<\lambda\le D} \{\mathcal{L}(\hat{\boldsymbol{\alpha}} + \lambda\eta, \hat{\boldsymbol{\beta}})\} - \mathcal{L}(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}) \\ \ge \frac{1}{2}\min\left\{D, \frac{\eta^T\nabla_\alpha\mathcal{L}(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}})}{\eta^T\boldsymbol{J}_{\alpha\alpha}\eta}\right\}\eta^T\nabla_\alpha\mathcal{L}(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}) \quad (25)\end{aligned}$$

and

$$\begin{aligned}\max_{0<\lambda\le D} \{\mathcal{L}(\hat{\boldsymbol{\alpha}} + \lambda\boldsymbol{e}_{t,\boldsymbol{y}}, \hat{\boldsymbol{\beta}})\} - \mathcal{L}(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}) \\ \ge \frac{1}{2}\min\left\{D, \frac{\frac{\partial\mathcal{L}}{\partial\alpha_{t,\boldsymbol{y}}}(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}})}{\|\delta\phi_t(\boldsymbol{y})\|^2}\right\}\frac{\partial\mathcal{L}}{\partial\alpha_{t,\boldsymbol{y}}}(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}) \quad (26)\end{aligned}$$

from Lemmas 12 and 13, respectively.

Now for a given pair of primal parameters $(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\xi}})$ and corresponding dual variables $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}})$, consider the adding an example $\boldsymbol{y}_t^\star$ to the constraint set $\mathcal{A}_t$ in Line 9 of Algorithm 1. Fixing $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}} \ge \mathbf{0}$ we can write

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}; \hat{\boldsymbol{\beta}}) = -\frac{1}{2}\sum_{\substack{t=1,\\ \boldsymbol{y}\in\mathcal{Y}_t}}^T \sum_{\substack{t'=1,\\ \boldsymbol{y}'\in\mathcal{Y}_t'}}^T \alpha_{t,\boldsymbol{y}}\alpha_{t',\boldsymbol{y}'} \delta\phi_t(\boldsymbol{y})^T \delta\phi_{t'}(\boldsymbol{y}') \\ + \sum_{\substack{t=1,\\ \boldsymbol{y}\in\mathcal{Y}_t}}^T \alpha_{t,\boldsymbol{y}} \Big(\Delta(\boldsymbol{y}, \boldsymbol{y}_t) - \delta\phi_t(\boldsymbol{y})^T\boldsymbol{G}^T\hat{\boldsymbol{\beta}}\Big) + \kappa(\hat{\boldsymbol{\beta}}) \quad (27)\end{aligned}$$

where κ is independent of $\boldsymbol{\alpha}$. Then recognizing that

$$\hat{\boldsymbol{\theta}} = \sum_{\substack{t=1,\\ \boldsymbol{y}\in\mathcal{Y}_t}}^T \hat{\alpha}_{t,\boldsymbol{y}} \delta\phi_t(\boldsymbol{y})^T + \boldsymbol{G}^T\hat{\boldsymbol{\beta}} \quad (28)$$

and

$$\Delta(\boldsymbol{y}_t^\star, \boldsymbol{y}_t) - \hat{\boldsymbol{\theta}}^T \delta\phi_t(\boldsymbol{y}_t^\star) > \hat{\xi}_t + \epsilon \ge \epsilon \quad (29)$$

we arrive at the same bound for improvement in $\mathcal{L}$ as [Proposition 17, 34] for $\text{SVM}_1^{\Delta m}$.

Finally, noticing that for the case of $\boldsymbol{h} = \mathbf{0}$ we have as a primal feasible point $\boldsymbol{\theta} = \mathbf{0}$. Therefore we can upper bound $\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ by $C\bar{\Delta}$ where $\bar{\Delta} = \max_{t,\boldsymbol{y}\in\mathcal{Y}_t} \Delta(\boldsymbol{y}, \boldsymbol{y}_t)$ and so Theorem 18, which bounds the number of iterations of the dual optimization algorithm of Tsochantaridis et al. [38], applies. We conclude that for $\epsilon > 0$ our algorithm will converge in a polynomial number of iterations.