Latent Structural SVM Learning for Lower Linear Envelope Potentials in Binary Markov Random Fields

Chang Li

A subthesis submitted in partial fulfillment of the degree of
Master of Science (Honours) at
The Department of Computer Science
Australian National University

October 2016

Typeset in Palatino by TEX and LATEX 2ε.

Except where otherwise indicated, this thesis is my own original work.

Chang Li
27 October 2016

To my parents.

Acknowledgements

I would like to express my very great appreciation to my supervisor Stehpen Gould for his valuable and constructive suggestions during the planning and development of this research work. His great patience and willingness to give his time so generously has been very much appreciated.

I would also like to express my very great appreciation to the open source community (Python, OpenCV and Chun-Nam Yu etc.). I cannot finish this work without their generous contributions.

Abstract

The lower linear envelope potentials, one of higher-order potentials used in Markov Random Fields (MRFs) are raising interests in recent years due to its capability of enforcing consistency over large sets of random variables. Previous researches in this area are only able to learn the lower linear envelope function approximately thus losses a rich class of representations which may harm the model's performance.

In this thesis we propose an exact formulation of the lower linear envelope function which can also be inferred exactly and learned using the latent structural SVM. We first show the implicit equivalence between the quadratic pseudo-Boolean formulation and linear combination formulation of the lower linear envelope. Then, with the exact inference developed by previous research [9], we propose the learning algorithm based on the linear combination formulation within a latent structural SVM framework.

We experiment our algorithm on two experiments and demonstrate advantages and disadvantages between our new formulation and previous formulation [8, 9]. The results show that despite computational expensive during training, our new method is more efficient during testing, able to learn the lower linear envelope exactly and performs better on harder problems than the previous method.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One interesting task in machine learning is labeling over complex and structured objects. Many applications such as image segmentation, motif finding and noun-phrase parsing involved with representing jointly correlated variables. Encoding consistency constraints over large number of random variables, for example, is central to the problem of image segmentation. Algorithm frameworks like Markov Random Field (MRF) containing higher order energy functions and max margin method for solving learning problem are raising interests recently due to their capability of representing structural dependencies of variables and ensuring computationally feasible approximation.

Lower linear envelope potentials is one of higher order energy functions defined on MRF which becomes popular due to their ability to encode consistency relationship between labels in clique. Gould [9] investigated the submodularity of lower linear envelope potentials and developed a graph-cuts algorithm to perform exact inference on them. Then they proposed a Max-Margin framework to optimize potentials' parameters. However, in order to write the energy function into a linear combination, they sampled the lower linear envelope potentials using a set of fixed space points. Althought this formulation can be globally optimized by using the Max-Margin framework, it lost a rich class of representations of energy function due to the fixed space sampling. Removing the equally spaced constraint and introduce their auxiliary variables back will result in a latent SVM formulation. Under this formulation the algorithm can learn the lower linear envelope exactly. Our main goal in this thesis is focused on this extension. The difficulty is how to learn parameters of energy function together with latent information.

In practical, many information providing useful cues for prediction is not directly observable from data. For motif (repeated patterns in DNA sequences) finding problem, as an example, the task if to find motifs from a set of DNA sequences where the location of these motifs are unknown. Thus the information of position can be treated as hidden variable and is important to be considered in the model though it is not directly observable. Issues like this have been well studied by many researchers and latent SVMs, which can explicitly model hidden variables with joint feature vectors, outperforms many other methods.

The latent SVM was developed by Felzenszwalb et al. [6] and Yu and Joachims [26] independently in different ways. The main idea is introducing a latent variable to

extend the feature vector, which results in an arbitrary loss function, e.g. Hinge Loss, with an upper bound. Then the optimization was done by using Concave-Convex Procedure (CCCP) algorithm, which is guaranteed to decrease the objective function to a local minimum.

In this thesis, we are aiming at exploring a variant formulation of Gould [9] by rewriting the lower linear envelope function directly into a linear combination and developing the learning algorithm using the latent structural SVM. The rest of the thesis is structured as follows:

Chapter 2 describes work related to MRFs and latent structural SVM. We also provide some background about our experiments.

Chapter 3 describes our main contributions. We first introduce the concept of the lower linear envelope and the exact inference method developed by Gould [9]. We then propose the exact formulation of the lower linear envelope and develop the learning algorithm basing on that formulation.

Chapter 4 describes two experiments we use to compare our new method to previous method [9]. We also give brief explanations and summary at the end of each experiments.

Chapter 5 summarizes our work and point out advantages and disadvantages of our new formulation. We also provide some insights for future work.

Chapter 2

Related Work and Background

2.1 Related Work

2.1.1 Markov Random Fields

Markov Random Fields are also known as *undirected graphical model* can be seen as a regularized joint log-probability distribution of arbitrary non-negative functions over a set of maximal cliques of the graph [3]. Let C denotes a maximal clique in one graph and $\boldsymbol{y}_C$ denotes the set of variables in that clique. Then the joint distribution can be written as:

$$p(\boldsymbol{y}) = \frac{1}{Z} \prod_C \Psi_C(\boldsymbol{y}_C) \tag{2.1}$$

where Ψ is called *potential functions* which can be defined as any non-negative functions and $Z = \sum_{\boldsymbol{y}} \prod_C \Psi_C(\boldsymbol{y}_C)$ which is a normalization constant. To infer labels which best explains input data set, we can find the *maximum a posteriori* (MAP) labels by solving $\boldsymbol{y}^* = \text{argmax}_{\boldsymbol{y}}\, p(\boldsymbol{y})$. Because potential functions are restricted to be non-negative, it gives us more flexible representations by taking exponential of those terms. Thus the joint distribution becomes:

$$p(\boldsymbol{y}) = \frac{1}{Z} exp(-\sum_C E_C(\boldsymbol{y}_C)) \tag{2.2}$$

where E is called *energy functions* which can be arbitrary functions. Therefore, *maximum a posteriori* problem is equivalent to *energy minimization* problem, which is also known as *inference*:

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\text{argmax}}\, p(\boldsymbol{y}) = \underset{\boldsymbol{y}}{\text{argmin}}(-\sum_C E_C(\boldsymbol{y}_C)) \tag{2.3}$$

To optimize the performance we can also consider a weighted version of energy functions. In order to do this we can decompose energy functions over nodes $\mathcal{N}$, edges $\mathcal{E}$ and higher order cliques $\mathcal{C}$ [23] then add weights on them accordingly. Let $\boldsymbol{w}$ be the vector of parameters and ϕ be arbitrary feature function, then the energy can be decomposed as a set linear combinations of weights and feature vectors:

$$E(\boldsymbol{y}; \boldsymbol{w}) = \sum_{i \in \mathcal{N}} \boldsymbol{w}_i^U \phi^U(\boldsymbol{y}_i) + \sum_{(i,j) \in \mathcal{E}} \boldsymbol{w}_{ij}^P \phi^P(\boldsymbol{y}_i, \boldsymbol{y}_j) + \sum_{\boldsymbol{y}_C \in \mathcal{C}} \boldsymbol{w}_C^H \phi^H(\boldsymbol{y}_C) \tag{2.4}$$

where U denotes *unary* terms, P denotes *pairwise* terms and H denotes *higher order* terms (when $|C| > 2$ namely each clique contains more than two variables).

A weight vector $\boldsymbol{w}$ is more preferable if it gives the ground-truth assignments $\boldsymbol{y}_t$ less than or equal to energy value than any other assignments y:

$$E(y_t, w) \leq E(y, w) \, , \forall y \neq y_t, y \in \mathcal{Y} \tag{2.5}$$

Thus the goal of *learning* MRFs is to learn the parameter vector $\boldsymbol{w}^*$ which returns the lowest energy value for the ground-truth labels y_t relative to any other assignments y [23]:

$$\boldsymbol{w}^* = argmax_w(E(y_t, w) - E(y, w)) \, , \forall y \neq y_t, y \in \mathcal{Y} \tag{2.6}$$

We have introduced three main research topics of MRFs: definition of *energy function* (potential functions), *inference* problem (MAP or energy minimization) and *learning* problem. As for energy function, our work focus on a class of higher-order potentials defined as a concave piecewise linear function which is known as lower linear envelope potentials over a clique of binary variables. It has been raising much interest due to its capability of encoding consistency constraints over large subsets of pixels in an image [15, 20].

Kohli et al. [17] proposed a method to represent a class of higher order potentials with lower (upper) linear envelope potentials. By introducing auxiliary variables [13], they reduced the linear representation to a pairwise form and proposed an approximate algorithm with standard linear programming methods. However, they only show an exact inference algorithm on at most three terms. Following their routine, Gould [9] extended their method to a weighted lower linear envelope with arbitrary many terms which can be solved with an efficient algorithm. They showed the energy function with auxiliary variables is submodular by transforming it into a quadratic pseudo-Boolean form [4] and how *graph-cuts* [5, 7, 10] like algorithm can be applied to do exact *inference*.

Gould [9] solved *learning* problem of lower linear envelope under the max margin framework [25]. In their work they pointed out the potential relationship between their auxiliary representation and latent SVM [26]. Our work is closely based on their research. We continue to use the higher order energy function and inference algorithm developed in their previous work [8] and extend their max margin learning algorithm to include latent variables. The learning algorithm we use is an extension of max margin framework which is known as "latent structural SVM" [26].

2.1.2 Latent Structural SVMs

The max-margin framework [24, 25] is a principled approach to learn the weights of pairwise MRFs. Szummer et al. [23] adapted this framework to optimize parameters of pairwise MRFs inferred by graph-cuts method. In our previous work Gould [9] extended this framework with additional linear constraints which enforces concavity on weights thus can be used for learning lower linear envelope potentials.

In this section we introduce *latent structural SVM* [26] which extends the max-margin framework by encoding latent information in feature vector. In section 3.2 we will show how this framework can be adapted to learn parameters for higher order energy function with latent variables.

Given an a linear combination of features vector $\phi(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^m$ and weights $\boldsymbol{\theta} \in \mathbb{R}^m$, and a set of n training examples $\{\boldsymbol{y}_i\}_{i=1}^n$ max-margin framework can be used to solve optimized solution $\boldsymbol{\theta}^*$. To include unobserved information in the model, Yu[26] extended the joint feature function[25] $\phi(\mathbf{x}, \mathbf{y})$ with a latent variable $\mathbf{h} \in \mathcal{H}$ to $\phi(\mathbf{x}, \mathbf{y}, \mathbf{h})$. So the inference problem becomes

$$f_\theta(x) = \underset{(\mathbf{y}\times\mathbf{h})\in\mathcal{Y}\times\mathcal{H}}{\operatorname{argmax}} \; \theta \cdot \phi(\mathbf{x}, \mathbf{y}, \mathbf{h}) \tag{2.7}$$

Accordingly, the loss function can be extended as

$$\Delta((\mathbf{y}_i, \mathbf{h}_i^*(\theta)), (\hat{\mathbf{y}}_i(\theta), \hat{\mathbf{h}}_i(\theta)))$$

where

$$(\hat{\mathbf{y}}_i(\theta), \hat{\mathbf{h}}_i(\theta)) = \underset{(\mathbf{y}\times\mathbf{h})\in\mathcal{Y}\times\mathcal{H}}{\operatorname{argmax}} \; \theta \cdot \phi(\mathbf{x}_i, \mathbf{y_i}, \mathbf{h}) \tag{2.8}$$

$$\mathbf{h}_i^*(\theta) = \underset{\mathbf{h}\in\mathcal{H}}{\operatorname{argmax}}\, \theta \cdot \phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}) \tag{2.9}$$

The loss function under this formulation measures difference between the inferred result pair $(\hat{\mathbf{y}}_i(\theta), \hat{\mathbf{h}}_i(\theta))$ and the pair $(\mathbf{y}_i(\theta), \mathbf{h}_i^*(\theta))$ which best explains the training data. However, under this formulation the "loss augmented inference" used in structural SVMs[25] to remove the complexity cannot be performed due to the dependence of loss function Δ on hidden variables $\mathbf{h}_i^*(\theta)$. Yu and Joachims [26] argued that in real world applications hidden variables are usually intermediate results and are not required as an output[26]. Therefore, the loss function can only focus on the inferenced hidden variables $\hat{\mathbf{h}}_i(\theta)$ which leads to:

$$\Delta((\mathbf{y}_i, \mathbf{h}_i^*(\theta)), (\hat{\mathbf{y}}_i(\theta), \hat{\mathbf{h}}_i(\theta))) = \Delta(\mathbf{y}_i, \hat{\mathbf{y}}_i(\theta), \hat{\mathbf{h}}_i(\theta))$$

Thus the upper bound used in standard structural SVMs[25] can be extended to:

$$\Delta((\mathbf{y}_i, \mathbf{h}_i^*(\theta)), (\hat{\mathbf{y}}_i(\theta), \hat{\mathbf{h}}_i(\theta))) \leq \left(\max_{(\hat{\mathbf{y}} \times \hat{\mathbf{h}}) \in \mathcal{Y} \times \mathcal{H}} [\theta \cdot \Psi(\mathbf{x}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}}) + \Delta(\mathbf{y}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}})] \right) \tag{2.10}$$

$$- \max_{\mathbf{h} \in \mathcal{H}} \theta \cdot \Psi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}) \tag{2.11}$$

Hence the optimization problem for Structural SVMs with latent variables becomes

$$\min_{\theta} \left(\frac{1}{2} \|\theta\|^2 + C \sum_{i=1}^{n} \left(\max_{(\hat{\mathbf{y}} \times \hat{\mathbf{h}}) \in \mathcal{Y} \times \mathcal{H}} [\theta \cdot \Psi(\mathbf{x}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}}) + \Delta(\mathbf{y}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}})] \right) \right) \tag{2.12}$$

$$- C \sum_{i=1}^{n} \left(\max_{\mathbf{h} \in \mathcal{H}} \theta \cdot \Psi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}) \right)$$

which is a difference of two convex functions. Problem of this formulation can be solved using the Concave-Convex Procedure (CCCP)[27] which is guaranteed to converge to a local minimum. Yu and Joachims [26] proposed a two stages algorithm. In the first step the latent variable $\boldsymbol{h}_i^*$ which best explains training pair $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ is found by solving equation (2.9). This step is also called the "latent variable completion" problem. In the second step $\boldsymbol{h}_i^*$ is used as completely observed to substitute $\boldsymbol{h}$ in equation (2.12). Therefore, solving equation (2.12) is equivalent to solve the standard structural SVM problem.

Auxiliary variables was introduced to help representing lower linear envelope potentials in an energy-minimization setting [13]. In order to adapt the energy function to max margin framework, Gould [8] approximated the energy function using equally spaced break-points thus removed those auxiliary variables. In this thesis we try to optimize the energy function exactly by introducing auxiliary variables back into the feature vector and solving the learning problem using the latent structural SVM framework. We will present this in detail in section 3.2.

2.2 Background for Experiments

2.2.1 Superpixel

Markov Random Fields are often defined on grids of pixels or "cliques". However, artificial pixel-grids are usually computational expensive and not natrual representations of images. A much preferable representation is to group images in a both perceptual and semantic meaningful way. Superpixel algorithms, by capturing redundancy in images, can group pixels into semantically meaningful homogeneous regions and reduce hundreds of thousands of pixels into several hundreds of superpixels (see figure 2.1). Such a superpixel map can greatly increase the efficiency of images processing tasks by providing an efficient primitive on which MRF can compute Energies[1].

Figure 2.1: Picture on the left is the original picture which contains 154,400 pixels. Picture on the right is the same picture represented by 268 superpixels

There are various superpixel segmentation algorithms. When choosing the desirable algorithm we mainly consider the following two aspects[1]:

1. Computational Efficiency: Superpixels should be fast to generate. Algorithms with less complexity are much preferable.
2. Adherence to boundaries: Algorithms which can generate superpixels with higher boundary recall (less true edges are missed) are much preferable.

Basing on those two considerations the superpixel algorithm we use in this thesis is the state-of-the-art *Simple Linear Iterative Clustering* (SLIC) algorithm which outperforms other algorithms in nearly all aspects[1]. SLIC in essence is an adaptation of k-means clustering algorithm. It mainly differs from k-means with two distinctions[1]:

1. Searching space: In the assignment step k-means clustering computes distances from each cluster to every pixel in the image. On the other hand SLIC narrow the search space into a $2S \times 2S$ region (see figure 2.2), where $S = \sqrt{\frac{N}{K}}$. N is the number of total pixels in images and k is the only parameter of the algorithm which defines the number of superpixels (clusterings). Therefore, the SLIC algorithm reduces the complexity from k-means' $\mathcal{O}(N^k)$ to a linear complexity $\mathcal{O}(N)$.
2. Distance measurement: Distance in k-means algorithm is measured by Euclidean norm of color vectors between pairs of pixels. SLIC algorithm also takes spatial distance into account by redefining the distance as $D = \sqrt{d_c^2 + (\frac{d_s}{S})^2 m^2}$ where d_c is the Euclidean norm of color vectors and d_s is the Euclidean norm of pixels'

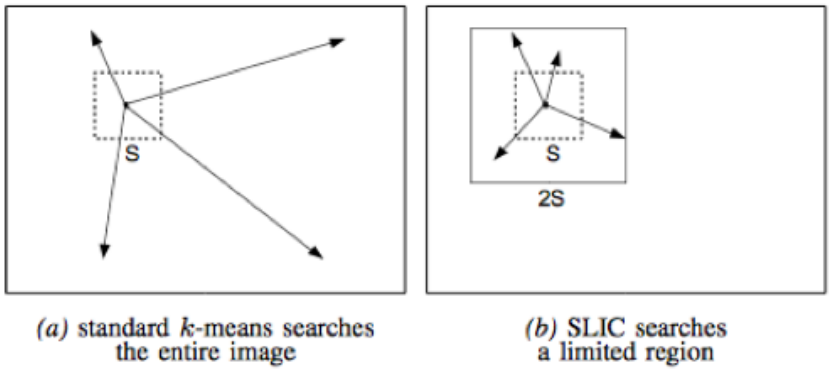


Figure 2.2: SLIC narrows the superpixel search regions.

positions. m is a constant which indicates the relative importance between color distance and spatial distance and thus balances the compactness and adherence to boundries in generated superpixels.

In this thesis we use SLIC as our cliques generating algorithm because it is fast to compute and out-performs other algorithms on adherence to image boundries while keeping results compact.

2.2.2 GrabCut

GrabCut algorithm was proposed by Rother et al. [21] in order to solve background foreground segmentation problem (see figure 2.3). They first defined MRFs over an labeled image and then use *graph-cuts* [5] method to do the inference. In this section we mainly focus on two of their contributions: estimating color distribution (foreground and background) using *Gaussian Mixture Models* (GMMs) and an *EM* like two-step algorithm to train their model.

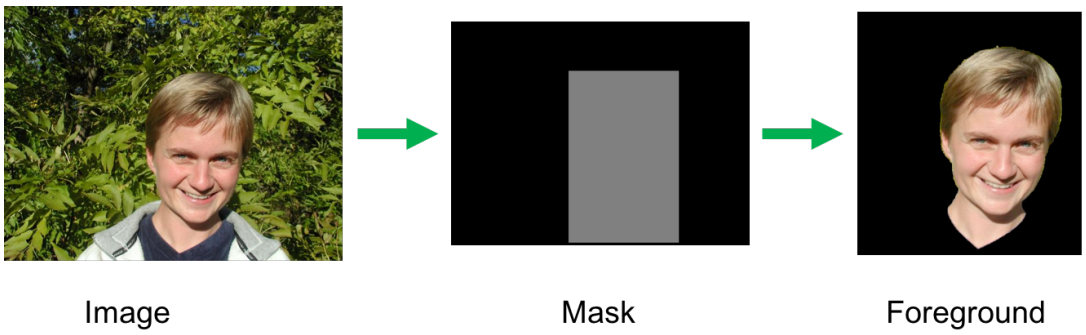


Figure 2.3: Picture on the left is the original picture. Picture on the middle is a user defined mask. The task is to extract foreground pixels within that rectangle. On the right is the ground truth foreground.

Algorithm 1 GrabCut training algorithm

1: **repeat**
2: *Assign GMM components* to pixels:
$\boldsymbol{k}_i^* = \text{argmin}_{\boldsymbol{k}_i} \phi^U(\alpha_i, \boldsymbol{k}_i, \boldsymbol{\theta}, \boldsymbol{z}_i)$
3: *Learn GMM parameters* from data z:
$\boldsymbol{\theta} = \text{argmin}_{\boldsymbol{\theta}} \sum_{i \in \mathcal{N}} \phi^U(\alpha_i, \boldsymbol{k}_i, \boldsymbol{\theta}, \boldsymbol{z}_i)$
4: *Estimate segmentation*: graph-cuts inference:
$\min_{\alpha} \min_{\boldsymbol{k}} E(\alpha, \boldsymbol{k}, \boldsymbol{\theta}, \boldsymbol{z})$
5: **until** convergence

Suppose there are N pixels in an image. In order to construct MRFs, they first defined an energy function (2.4) with unary and pairwise terms:

$$E(\alpha, \boldsymbol{k}, \boldsymbol{\theta}, \boldsymbol{z}) = \sum_{i \in \mathcal{N}} \phi^U(\alpha_i, \boldsymbol{k}_i, \boldsymbol{\theta}, \boldsymbol{z}_i) + \sum_{(i,j) \in \mathcal{E}} \phi^P(\alpha_i, \boldsymbol{z}_i) \tag{2.13}$$

where i is the index of pixels, $\alpha \in 0, 1$ is the label for pixel i. 0 is for the background and 1 is for the foreground. $\boldsymbol{z}$ denotes the pixel vector in RGB color space. $\boldsymbol{k}$ and $\boldsymbol{\theta}$ are all parameters vectors and will be explained in the next paragraph.

The first contribution is using *Gaussian Mixture Models* (GMMs) with K components (typically $K = 5$) for generating unary terms. They used two GMMs in their model, one for background and one for foreground. $\boldsymbol{k} = \boldsymbol{k}_1, \ldots, \boldsymbol{k}_i, \ldots, \boldsymbol{k}_N$ with $\boldsymbol{k}_i \in 1, \ldots, K$ assigns each pixel i to a unique GMMs component. The component is either belonging to background's GMMs or foreground's GMMs, which is depended on the label $\alpha_i \in 0, 1$. $\boldsymbol{\theta}$ is the parameter vector which contains parameters of standard GMMs plus *mixture weighting coefficients* [21].

The pairwise function ϕ^P is defined as a smoothness indicator which measures both color space and spatial distances simultaneously. It is used to encourage coherence of similar pixel pairs. This energy function was later used to construct an *st min-cut* graph which can be inferred efficiently using *graph-cuts* [5] algorithm. This gives some insights to their second contribution.

To optimize the performance, they developed a two-step learning algorithm. The algorithm first re-assign GMMs components ($\boldsymbol{k}$) to each pixel then update parameters $\boldsymbol{\theta}$ with new assignments. The result of the trained GMMs are used directly into *graph-cuts* algorithm as unary terms. Finally the label α_i for each pixel i is inferred jointly using *graph-cuts* algorithm. This whole procedure is repeated until convergence (or reaches termination conditions). We briefly summarized this procedure in Algorithm 1

In this thesis we also use GMMs trained by GrabCut algorithm for our unary terms.

Chapter 3

Methodology

3.1 Lower Linear Envelope MRFs

We begin with extending standard Markov Random Fields (see equation (2.4)) to include the lower linear envelope potential. We then show how to perform exact inference in models with these potentials. In 3.2 we will discuss learning the parameters of the models. Major work in this section is done by Gould [9].

3.1.1 Lower Linear Envelope Potentials

From section 2.1.1 we have already introduced that an *energy function* may contain *unary*, *pairwise* and *higher-order* potentials (see equation (2.4)). In this section we mainly focus on one class of higher-order potentials ϕ^H defined as a concave piecewise linear function which is known as *lower linear envelope potentials*. This has been studied extensively in Markov Random Fields area for encouraging consistency over large cliques [8, 15, 20].

Let $\mathcal{C}$ denotes the set of all maximal cliques in an image and $\boldsymbol{y}_c = \{y_i | \text{for } i \in c\}$ denotes set of random variables in the clique c, a weighted lower linear envelope potential [9] over $\boldsymbol{y}_c$ is defined as the minimum over a set of K linear functions as:

$$\psi_c^H(\boldsymbol{y}_c) = \min_{k=1,\dots,K} \{a_k W_c(\boldsymbol{y}_c) + b_k\}. \tag{3.1}$$

where $W_c(\boldsymbol{y}_c) = \sum_{i \in c} w_i y_i$ with $w_i^c \geq 0$ and $\sum_{i \in c} w_i^c = 1$ which are weights for each clique. $(a_k, b_k) \in \mathbb{R}^2$ are the linear function parameters. We illustrate an example [9] with three linear functions in Figure 3.1.

Inference on energy function contains lower linear potentials is the same as the standard equation (2.4) and is given by:

$$\boldsymbol{y}^* = \operatorname{argmin} E(\boldsymbol{y}) \tag{3.2}$$

Suppose that parameters $\{(a_k, b_k)\}_{k=1}^K$ are sorted in decreasing order of a_k. From *Definition 3.1* [9] we know that the k-th linear function is said to be *active* if there exists

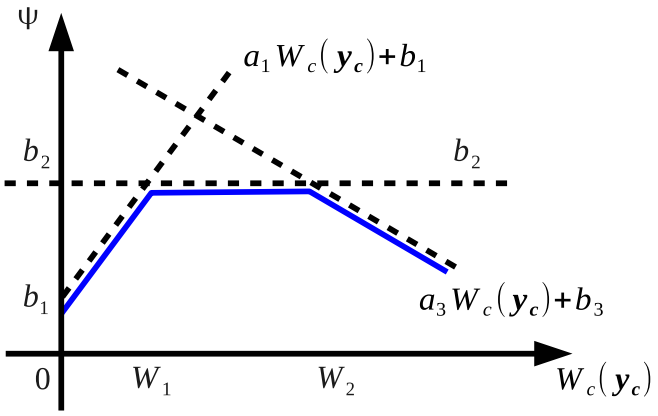


Figure 3.1: Example lower linear envelope $\psi_c^H(\boldsymbol{y}_c)$ (shown solid) with three terms (dashed). When $W_c(\boldsymbol{y}_c) \leq W_1$ the first linear function is active, when $W_1 < W_c(\boldsymbol{y}_c) \leq W_2$ the second linear function is active, otherwise the third linear function is active.

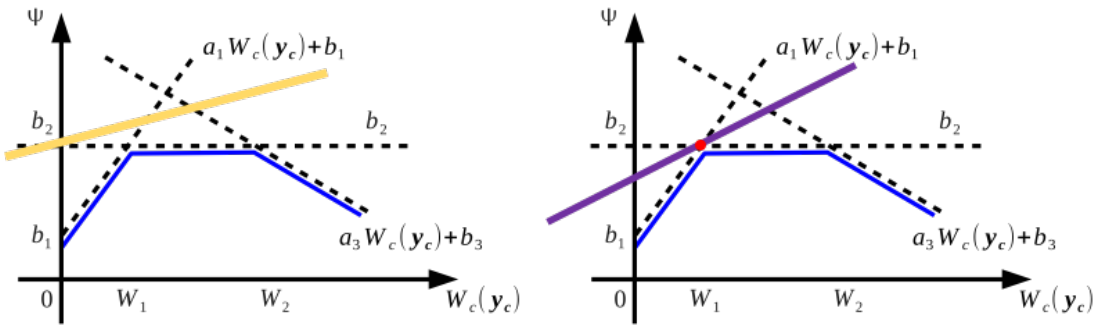


Figure 3.2: Example lower linear envelope with redundant linear functions. On the left figure, the solid yellow line is always inactive. On the right figure, the solid purple line intersects *line* 1 and *line* 2 at the red point. It's only active when *line* 1 and *line* 2 are both active. Both solid lines are redundant linear functions hence can be removed without changing their energy function.

$x \in (0, 1)$ such that the following two inequalities hold

$$\begin{aligned} a_{k-1}x + b_{k-1} &> a_k x + b_k \\ a_{k+1}x + b_{k+1} &> a_k x + b_k \end{aligned} \tag{3.3}$$

The k-th linear function is said to be *redundant* (*Definition 3.2* [9]) 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. Figure 3.2 depicts such conditions. As a result, removing redundant functions from the potential does not chang the energy function.

To ensure potentials do not contain redundant linear functions, Gould [9] rearranged conditions 3.3 in terms of x and proposed a condition on parameters of the envelope. The k-th linear function is not redundant if the following condition is satis-

fied:

$$0 < \frac{b_k - b_{k-1}}{a_{k-1} - a_k} < \frac{b_{k+1} - b_k}{a_k - a_{k+1}} < 1. \tag{3.4}$$

Another important property of equation (3.2) is shift invariant [9] (vertically). We write $\widetilde{\psi}_c^H(\boldsymbol{y}_c)$ by shift equation (3.1) vertically with an abitrary amount $b^{const} \in R$

$$\widetilde{\psi}_c^H(\boldsymbol{y}_c) = \min_{k=1,\dots,K} \left\{ a_k W_c(\boldsymbol{y}_c) + b_k + b^{\text{const}} \right\}$$

Then we have

$$\underset{\boldsymbol{y}_c}{\operatorname{argmin}}\, \psi_c^H(\boldsymbol{y}_c) = \underset{\boldsymbol{y}_c}{\operatorname{argmin}}\, \widetilde{\psi}_c^H(\boldsymbol{y}_c). \tag{3.5}$$

Therefore, in the following discussion without loss of generality we assume $b_1 = 0$ thus $b_k \geq 0$ for $k = 1, \dots, n$.

3.1.2 Exact Inference

Exact inference on MRFs has been extensively studied in past years. Researchers found that, energy functions which can be transformed into quadratic pseudo-Boolean functions [11, 12, 22] are able to be minimized exactly using *graph-cuts* like algorithms [7, 10] when they satisfy submodularity condition [4]. Kohli et al. [16] and Gould [8] adapted those results to perform exact inference on lower linear envelope potentials. In this section we mainly focus on describing the *st min cut* graph constructed by Gould [8, 9] for exact inference (3.2) of energy function containing lower linear envelope potentials.

Following the approach of Kohli and Kumar [13], Gould [8, 9] transformed the weighted lower linear envelope potential (3.1) into a quadratic pseudo-Boolean function by introducing $K - 1$ auxiliary variables $\boldsymbol{z} = (z_1, \dots, z_{K-1})$ with $z_k \in \{0, 1\}$:

$$E^c(\boldsymbol{y}_c, \boldsymbol{z}) = a_1 W_c(\boldsymbol{y}_c) + b_1 + \sum_{k=1}^{K-1} z_k \left((a_{k+1} - a_k) W_c(\boldsymbol{y}_c) + b_{k+1} - b_k \right) \tag{3.6}$$

for a single clique $c \in \mathcal{C}$. Under this formulation, Gould [8, 9] showed that minimizing the pseudo-Boolean function over $\boldsymbol{z}$ is equivalent to selecting (one of) the active functions(s) from equation (3.1). Another important property of optimized $\boldsymbol{z}$ under this formulation is that it automatically satisfies the constraint [9]:

$$z_{k+1} \leq z_k \tag{3.7}$$

this property give rise to further development of parameter vector (3.11) and feature vector (3.12) which are used in latent structural SVM.

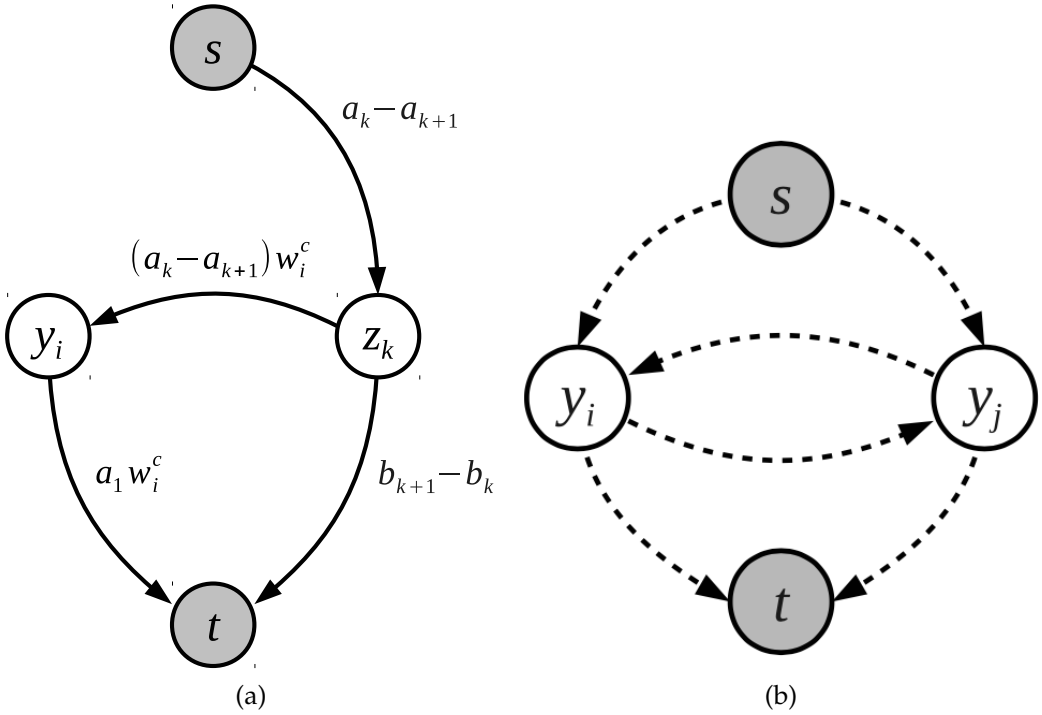


Figure 3.3: *st*-graph construction [9] for equation (3.8), unary and pairwise terms. Every cut corresponds to an assignment to the random variables, where variables associated with nodes in the $\mathcal{S}$ set take the value one, and those associated with nodes in the $\mathcal{T}$ set take the value zero. With slight abuse of notation, we use the variables to denote nodes in our graph.

In order to construct the *st-min-cut* graph, Gould [9] rewrote equation (3.6) into *posiform* [4]:

$$E^c(\boldsymbol{y}_c, \boldsymbol{z}) = b_1 - (a_1 - a_K) + \sum_{i \in c} a_1 w_i^c y_i + \sum_{k=1}^{K-1} (b_{k+1} - b_k)\, z_k \qquad (3.8)$$
$$+ \sum_{k=1}^{K-1} (a_k - a_{k+1})\, \bar{z}_k + \sum_{k=1}^{K-1} \sum_{i \in c} (a_k - a_{k+1})\, w_i^c \bar{y}_i z_k$$

where $\bar{z}_k = 1 - z_k$ and $\bar{y}_i = 1 - y_i$. a_1 is assumed to be greater than 0 so that all coefficients are positive (recall we assume $b_1 = 0$ in section 3.1.1 and we have $a_k > a_{k+1}$ and $b_k < b_{k+1}$). After proving *submodularity* of the energy function (3.8), Gould [9] constructed the *st-min-cut* graph based on equation (3.8).

The construction is explained in Figure 3.3. Figure (a) denotes construction for equation (3.8). 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 $b_{k+1} - b_k$. Figure (b) denotes construction for unary and pairwise

terms (see [18]). For unary edges (4 edges on both sides), weights on each edge are corresponding to values in input unary terms accordingly. For pairwise edges (2 edges in the middle), both edges share the same weight which equals to the input pairwise term.

3.2 Learning the Lower Linear Envelope with Latent Information

With the inference algorithm in hand, we now can develop the learning algorithm for weighted lower linear envelope potentials using the latent structural SVM framework. We begin by transforming the equation (3.6) into a linear combination of parameter vector and feature vector. Then a two-step algorithm was developed to solve the latent structural SVM.

3.2.1 Transforming Between Representations

The latent structural SVM formulation (see equation (2.7)) requires that the energy function be expressed as a linear combination of features and weights while our higher-order potential is represented as the minimum over a set of linear functions. However, in 3.1.2 we reformulated the piesewise linear functions into a quadratic pseudo-Boolean function (3.6) by introducing auxiliary variables. Now we show function (3.6) itself is an inner product of parameter vector and feature vector with latent information. We first noticed that the function can be expanded as a summation of $2K-1$ terms:

$$\begin{aligned} E^c(y_c, z) &= a_1 W_c(y_c) + b_1 + \sum_{k=1}^{K-1} z_k((a_{k+1} - a_k)W_c(y_c) + b_{k+1} - b_k) \\ &= a_1 W_c(y_c) + \sum_{k=1}^{K-1} (a_{k+1} - a_k) z_k W_c(y_c) + \sum_{k=1}^{K-1} (b_{k+1} - b_k) z_k \end{aligned} \tag{3.9}$$

Here we use the fact of equation (3.5) and let $b_1 = 0$. Now we can reparameterize the energy function as

$$E^c(\boldsymbol{y}_c, \boldsymbol{z}; \boldsymbol{\theta}) = \boldsymbol{\theta}^T \boldsymbol{\phi}(\boldsymbol{y}_c, \boldsymbol{z}) \tag{3.10}$$

where:

$$\theta_k = \begin{cases} a_1 & \text{for } k = 1 \\ a_k - a_{k-1} & \text{for } 1 < k \leq K \\ b_{k+1-K} - b_{k-K} & \text{for } K < k \leq 2K-1 \end{cases} \tag{3.11}$$

$$\phi_k = \begin{cases} W_c(\boldsymbol{y}_c) & \text{for } k = 1 \\ W_c(\boldsymbol{y}_c)\boldsymbol{z}_k & \text{for } 1 < k \leq K \\ \boldsymbol{z}_k & \text{for } K < k \leq 2K - 1 \end{cases} \tag{3.12}$$

Under this formulation, inference problems (2.8) and (2.9) introduced in section 2.1.2 can be written as:

$$(\hat{\mathbf{y}}_k(\boldsymbol{\theta}), \hat{\mathbf{z}}_k(\theta)) = \underset{(\mathbf{y}\times\mathbf{z})\in\mathcal{Y}\times\mathcal{Z}}{\operatorname{argmin}} \boldsymbol{\theta}^T \cdot \phi(\mathbf{y}_k, \mathbf{z}_k) \tag{3.13}$$

and

$$\mathbf{z}_k^*(\boldsymbol{\theta}) = \underset{\mathbf{z}\in\mathcal{Z}}{\operatorname{argmin}} \boldsymbol{\theta}^T \cdot \phi(\mathbf{y}_k, \mathbf{z}_k) \tag{3.14}$$

There are 2 facts worth to mention. The first fact is that in our previous construction of minimum-*st*-cut graph the latent variable $\boldsymbol{z}$ is already included. Therefore, we can apply our inference algorithm directly on our 2 new formulations.

However, for equation (3.14) there exists more efficient algorithm. At training stage the ground-truth labels y_i is a function input thus completely observed. Therefore, the term $((a_{k+1} - a_k)W_c(\boldsymbol{y}_c) + b_{k+1} - b_k)$ in equation 3.9 becomes constant. So we can infer latent variable $\boldsymbol{z}$ explicitly by:

$$z_k^c = \begin{cases} 0 & \text{if } ((a_{k+1} - a_k)W_c(y_c) + b_{k+1} - b_k) \geq 0 \\ 1 & \text{otherwise.} \end{cases} \tag{3.15}$$

To show the equivalence between equation (3.6) and equation (3.10) we consider the example illustrated in figure 3.4. Assume the inferred latent vector $\boldsymbol{z}^c = (1, 1, 0, 0)$. Plug it into equation (3.12) the energy function can be written as:

$$\begin{aligned} E^c(\boldsymbol{y}_c, \boldsymbol{z}; \boldsymbol{\theta}) &= \begin{bmatrix} a_1 \\ a_2 - a_1 \\ a_3 - a_2 \\ a_4 - a_3 \\ b_2 \\ b_3 - b_2 \\ b_4 - b_3 \end{bmatrix}^T \begin{bmatrix} W_c(\boldsymbol{y}_c) \\ W_c(\boldsymbol{y}_c) \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ &= a_1 W_c(\boldsymbol{y}_c) + (a_2 - a_1)W_c(\boldsymbol{y}_c) + b_2 \\ &= a_2 W_c(\boldsymbol{y}_c) + b_2 \end{aligned}$$

Therefore, assignments inferred by graph-cut algorithm can be directly encoded into a linear combination by using our latent structural SVM formulation for learning purpose. The remaining task is to ensure the concavity of $\boldsymbol{\theta}$. We do this by adding the following constraint:

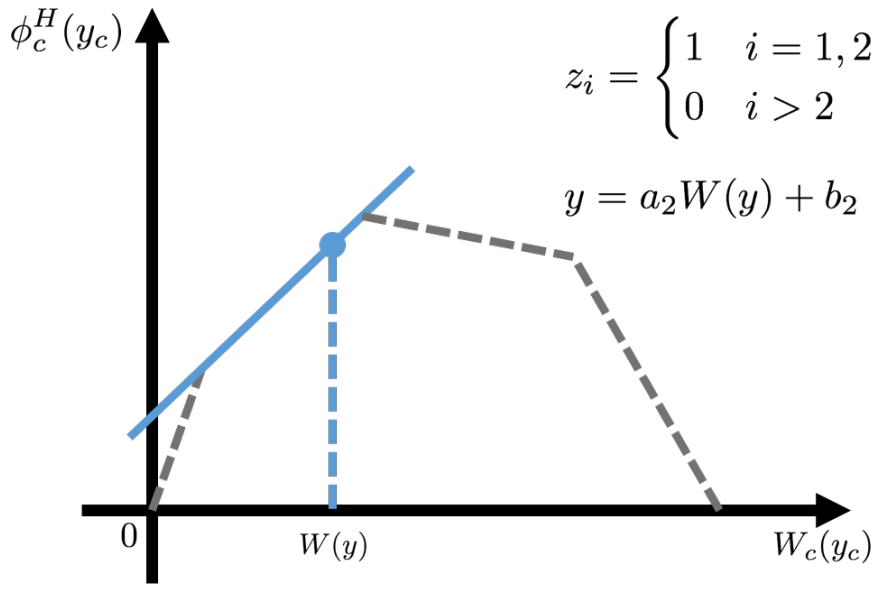


Figure 3.4: Example piecewise-linear concave function of $W_c(\boldsymbol{y}_c) = \sum_{i \in c} w_i^c y_i$. Assume the second linear function is active namely $\boldsymbol{z}^c = (1, 1, 0, 0)$. The result of linear combination of parameter vector and feature vector is same as quadratic psuedo-Boolean function.

$$A\boldsymbol{\theta} \geq 0, \quad A = \begin{bmatrix} 1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{P} \end{bmatrix} \in \mathbb{R}^{(2K-1)\times(2K-1)} \tag{3.16}$$

where $-\mathbf{1}$ is a matrix of size $(K-1) \times (K-1)$ and $\mathbf{P}$ is an identity matrix of size $(K-1) \times (K-1)$.

One subtle problem we found during experiments is that the algorithm can be stuck small numerical value. To avoid this we add small slack variables ϵ on those constraints:

$$A\boldsymbol{\theta} \geq \epsilon, \quad A = \begin{bmatrix} 1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{P} \end{bmatrix} \in \mathbb{R}^{(2K-1)\times(2K-1)} \tag{3.17}$$

where ϵ usually equal to $\mathbf{1}^{-15}$ in our experiments.

3.2.2 Latent Structural SVM Learning

With the inner product formulation (equation (3.10)) of higher order energy function in hand, we now able to develop our latent structural SVM learning algorithm. The energy function (higher order function together with unary and pairwise functions)

can be written as:

$$E_{all}(y,z) = \begin{bmatrix} \theta^{H} \\ \theta^{unary} \\ \theta^{pairwise} \end{bmatrix}^{T} \cdot \begin{bmatrix} \phi^{H} \\ \phi^{unary} \\ \phi^{pairwise} \end{bmatrix} = \theta_{all}^{T} \cdot \phi_{all} \tag{3.18}$$

where $\theta^{H} \in$ is the parameter vector in higher order equation (3.10) of size $2K-1$. θ^{unary} and $\theta^{pairwise}$ are both scalars. $\phi^{\text{unary}} = \sum_i \psi_i^{U}(y_i)$ and $\phi^{\text{pairwise}} = \sum_{ij} \psi_{ij}^{P}(y_i, y_j)$. Therefore, the size of θ_{all} is $2K+1$.

Plug equation (3.13) and equation (3.14) into object function (2.12), the latent structural SVM object function for our problem can be derived as a difference of two convex functions:

$$\min_{\theta} \left(\frac{1}{2}\|\theta\|^2 + C \sum_{i=1}^{n} \left(\max_{(\hat{\mathbf{y}} \times \hat{\mathbf{z}}) \in \mathcal{Y} \times \mathcal{Z}} [\theta \cdot \phi(\hat{\mathbf{y}}, \hat{\mathbf{z}}) + \Delta(\mathbf{y}_i, \hat{\mathbf{y}}, \hat{\mathbf{z}})] \right) \right) \tag{3.19}$$
$$-C \sum_{i=1}^{n} \left(\max_{\mathbf{z} \in \mathcal{Z}} \theta \cdot \phi(\mathbf{y}_i, \mathbf{z}) \right)$$

As mentioned by Yu and Joachims [26] the Concave-Convex Procedure (CCCP) [27] can be used to solve the optimization problem. Our algorithm contains two stages. We first imputes the latent variables $\boldsymbol{z}$ explicitly by equation (3.14). Namely solving the "latent variable completion" problem [26]:

$$z_i^{*} = \underset{\mathbf{z} \in \mathcal{Z}}{\operatorname{argmax}}\, \theta \cdot \phi(\mathbf{y}_i, \mathbf{z}) \tag{3.20}$$

The inference result z_i^{*} for $i = 1, \ldots, n$ is used as completely observed for later stage. With the latent variable z_i^{*} which best explains the ground-truth data y_i in hand, updating the parameter vector $\boldsymbol{\theta}$ is similar to solve the standard max-margin optimization problem described in [9]:

$$\min_{\theta} \left(\frac{1}{2}\|\theta\|^2 + C \sum_{i=1}^{n} \left(\max_{(\hat{\mathbf{y}} \times \hat{\mathbf{z}}) \in \mathcal{Y} \times \mathcal{Z}} [\theta \cdot \phi(\hat{\mathbf{y}}, \hat{\mathbf{z}}) + \Delta(\mathbf{y}_i, \hat{\mathbf{y}}, \hat{\mathbf{z}})] \right) \right) \tag{3.21}$$
$$-C \sum_{i=1}^{n} \left(\theta \cdot \phi(\mathbf{y}_i, \mathbf{z}_i^{*}) \right)$$

The last problem remaining is the initialization method. Because our objective function (3.21) is not convex and the CCCP algorithm is only guaranteed to converge to a local minimum or saddle point[27], initialization of $\boldsymbol{\theta}$ might affect the performance of our algorithm. Since there are no theoretical solution for this problem, we only propose an empirical Algorithm 2:

We assume that the more evenly distributed of $W_c(Y_c)$ where $c \in \mathcal{C}$ on x axis, the more rich representation (number of linear functions) the energy function should

Algorithm 2 Empirical initialization algorithm for θ

$gap = \frac{1}{K}$, $a_1 = \mathcal{U}(0, 1e6)$, $b_1 = 0$, $sp_1 = (0, 0)$, $w_0 = 0$, $counter = 2$
for each clique $c \in \mathcal{C}$ **do**
  Compute weighted clique value $w_c = W_c(y_C)$
  if $w_c - w_{c-1} > gap$ **then**
    $upbound = a_{counter} w_c + b_{counter}$
    $sp_{counter} = (w_c, \mathcal{U}(upbound - 0.5, upbound))$
    Calculate $a_{counter}$ and $b_{counter}$ using $sp_{counter-1}$ and $sp_{counter}$
    $counter = counter + 1$
  end if
end for
If $counter < K$, remaining as and bs are all set to be $a_{counter}$ and $b_{counter}$
Calculate θ using $\{a_k, b_k\}_{k=1}^{K}$

have. In order to initialize θ, we first determine the x-coordinate of sampled points sp. Then we sample its y-coordinate from a uniform distribution $\mathcal{U}(upbound, upbound - 0.5)$ to add some randomness in our initialization as well as maintain concavity. Linear parameters a_k and b_k are later calculated using those sampled points sp_k and sp_{k-1}. At last we encode $\{a_k, b_k\}_{k=1}^{K}$ into θ using equation (3.11).

Our optimization algorithm is summarized in Algorithm 3.

Algorithm 3 Learning lower linear envelope MRFs with latent variables.

Set $MaxIter = 100$
input training set $\{\boldsymbol{y}_i\}_{i=1}^n$, regularization constant $C > 0$, and tolerance $\epsilon \geq 0$
Initialize $\boldsymbol{\theta}$ using Algorithm 2
repeat
  Set $iter = 0$
  for each training example, $i = 1, \ldots, n$ **do**
    compute $\boldsymbol{z}_i^* = \text{argmax}_{\mathbf{z} \in \mathcal{Z}} \, \theta \cdot \phi(\mathbf{y}_i, \mathbf{z})$
  end for
  initialize active constraints set $\mathcal{C}_i = \{\}$ for all i
  repeat
    solve the quadratic programming problem in equation 3.21 with respect to active constraints set $\mathcal{C}_i$ for all i and concavity constraints $A\boldsymbol{\theta} \geq \epsilon$ to get $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\xi}}$
    for each training example, $i = 1, \ldots, n$ **do**
      compute $\hat{\boldsymbol{y}}_i, \hat{\boldsymbol{z}}_i = \text{argmin}_{\boldsymbol{y}} \, E(\boldsymbol{y}, \boldsymbol{z}; \hat{\boldsymbol{\theta}}) - \Delta(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{y}_i)$
      if $\hat{\xi}_i + \epsilon < \Delta(\hat{\boldsymbol{y}}_i, \hat{\boldsymbol{z}}_i, \boldsymbol{y}_i) - E(\hat{\boldsymbol{y}}_i, \hat{\boldsymbol{z}}_i; \hat{\boldsymbol{\theta}}) + E(\boldsymbol{y}_i, \boldsymbol{z}_i^*; \hat{\boldsymbol{\theta}})$ **then**
        $\mathcal{C}_i \leftarrow \mathcal{C}_i \cup \{\boldsymbol{y}_i^\star\}$
      end if
    end for
  until no more violated constraints
  return parameters $\hat{\boldsymbol{\theta}}$
  Set $iter = iter + 1$
until $iter \geq MaxIter$
return parameters $\hat{\boldsymbol{\theta}}$

Chapter 4

Experiments

In this chapter, we follow our previous approach in [9] and conduct two experiments with different purposes. We first examine our method's effectiveness by comparing our results with [8, 9] on a synthetic checkerboard. We then extend our work to the real-world "GrabCut" problem introduced in section 2.2.2 and comparing our result with previous researches [21] to investigate our new methods' performance.

4.1 Synthetic Checkerboard

Since the main contribution of our work is extending our previous approximate formulation of lower linear envelope potentials to an exactly formulation, it is necessary to compare the results on synthetic checkerboard to previous work [9].

In this section we will experiment our method on three different problem instances: checkerboard with squares containing monotonous color 4.1.2, checkerboard with squares containing more pixels of one color over another 4.1.3 and checkerboard with uniformly colored squares containing unbalanced color 4.1.4.

4.1.1 Experiment Settings

An image of synthetic checkerboard contains 8×8 pixel squares. Each square (clique) contains 16×16 (256) pixels. The color of each pixel is either black 0 or white 1. Given a ground-truth checkerboard image $\boldsymbol{y}^* = y_1^*, \ldots, y_{16384}^*$, the observed unary terms $\boldsymbol{y} = y_1, \ldots, y_{16384}$ are generated as followings. Let η_0 and η_1 be the signal-to-noise ratios for the black and white squares, the unary terms are generated by destroying groud-truth label to noisy input

$$y_i = \eta_0 [\![y_i^\star = 0]\!] - \eta_1 [\![y_i^\star = 1]\!] + \delta_i \tag{4.1}$$

where $\delta_i \sim \mathcal{U}(-1, 1)$ is additive i.i.d. uniform noise. $[\![x]\!]$ is an indicator function which equals 1 when x is true and 0 otherwise. The task is to recover the ground-truth checkerboard from the noisy input.

Our MRF is constructed on this image by associating each node in the MRF to each pixel in the image. Thus our MRF contains $8 \times 8 \times 256 = 16,384$ variables.

The energy function used in this experiment follows equation (2.4) without pairwise terms.

$$E(\boldsymbol{y};\boldsymbol{\theta}) = \theta^U \sum_{i\in\mathcal{N}} \phi^U(\boldsymbol{y}_i) + \sum_{\boldsymbol{y}_c\in\mathcal{C}} \phi^H(\boldsymbol{y}_c, \boldsymbol{z}_c; \boldsymbol{\theta}^H) \tag{4.2}$$

where $\phi^U(\boldsymbol{y}_i) = \boldsymbol{y}_i$ and θ^U is a scalar weight for unary terms. $\phi^H(\boldsymbol{y}_c, \boldsymbol{z}_c; \boldsymbol{\theta}^H) = \boldsymbol{\theta}^{H\,T}\phi(\boldsymbol{y}_c, \boldsymbol{z}_c)$ is equivalent to equation (3.10) and added for each square (clique c) in the checkerboard. The number of linear equations K in equation (3.11) is set to be 10. The parameters θ^U and $\boldsymbol{\theta}^H$ are learned using Algorithm 3 with $MaxIter = 100$.

4.1.2 Monotonous Colored Squares

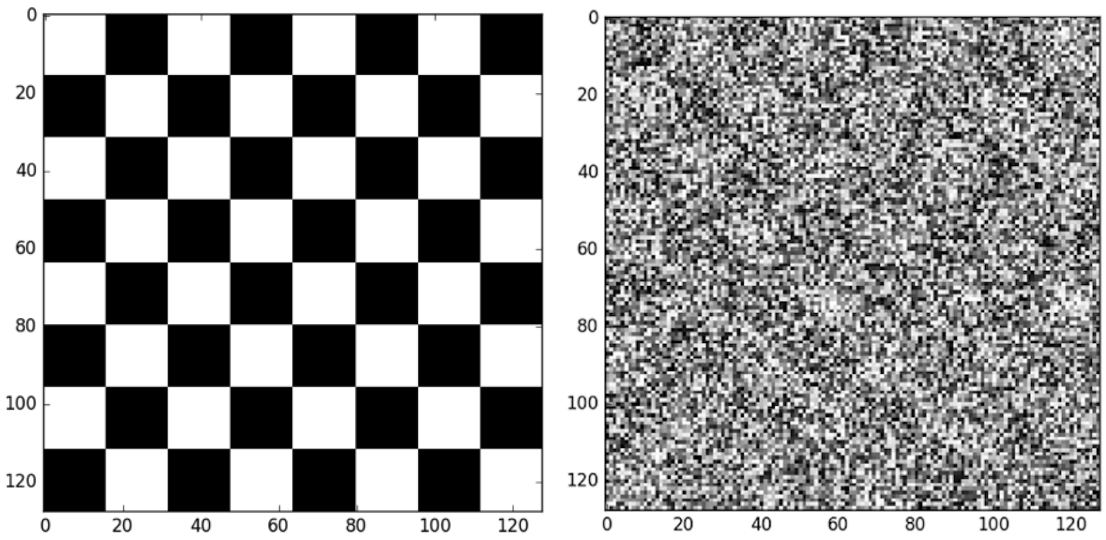


Figure 4.1: Example for monotonous colored squares. figure (a) is the ground-truth checkerboard. Figure (b) is the noisy input (unary terms) destroyed by equation (4.1)

We first repeat our previous black and white checkerboard experiment [8, 9] in order to examine the correctness of our new formulation. Each clique (square) $c \in \mathcal{C}$ in the checkerboard contains either all white pixels $y_i = 1,\ \forall i \in c$ or all black pixels $y_i = 0,\ \forall i \in c$. Figure 4.1 illustrates the ground-truth checkerboard and the noisy input destroyed by equation (4.1) with $\eta_0 = \eta_1 = 0.1$. Figure 4.2 shows the results of our new method (on the bottom) together with our previous method [9] (on the top).

From figure 4.2 we conclude that both formulations can recover checkerboard perfectly so our new formulation's accuracy is as good as previous one. However, there are significant differences between structural SVM formulation (previous method) and latent structural SVM formulation. There are 10 active linear functions in figure 4.2 (b) while there are only 2 active linear functions in figure 4.2 (d). Shapes

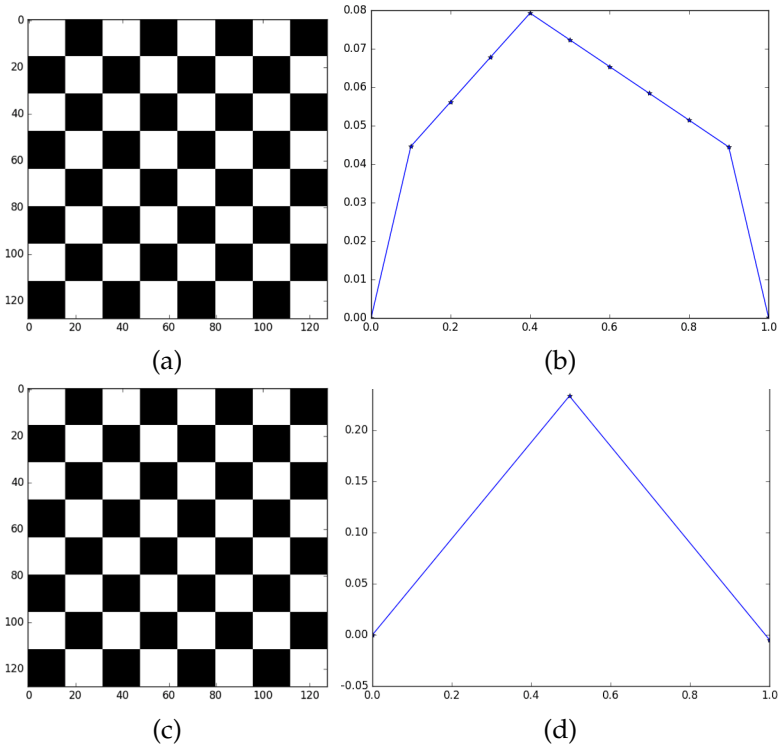


Figure 4.2: Results comparison for monotonous colored squares. Figure (a) and Figure (c) are inferred checkerboard from our previous and current formulation separately. Figure (b) and Figure (d) are lower linear envelopes learned by each formulation.

learned by each formulation are also significantly different.

In general, the second result is more preferable than the first one. The reason is despite the image contains 64 cliques, there are only two kinds of squares in the image: completely black and completely white. Accordingly, our model only see two kinds of cliques: completely 0s (black) and completely 1s (white). In this case, a lower linear envelope contains two linear functions is enough for encoding consistency information. This is reflected in figure 4.2 (d) which gives least penalty (0) when the clique value $W_C(y_c)$ equals either 0 or 1. It gives the highest penalty when $W_C(y_c)$ is in the middle because our model has least probability seen that in training data. The results certificates that our latent structural SVM formulation can learn lower linear envelope exactly. Therefore, we say that our new method learns more preferable lower linear envelope.

In terms of computational performance, because our initial point are generated randomly using Algorithm 2, the performance various between runnings. On average it takes 2 *outer loops* and 47 *inner loops* to converge. Which means the latent structural SVM formulation spends 3.5 times iterations to converge than previous one (27 iterations). Each *inner loop* took under 1s with inference taking about 120ms on a 2.7GHz dual-core Intel CPU, which is the same as our previous method.

4.1.3 Unbalanced Colored Squares

Experiment in section 4.1.2 proves that our latent structural SVM formulation can learn the lower linear envelope exactly. In this section we conduct further experiment to investigate its capability of representing unbalanced input. The desirable result of this experiment should be the shape of the lower linear envelope shifting along with the changing of input data.

We design our checkerboards contain unbalanced colored squares as shown in figure 4.3.

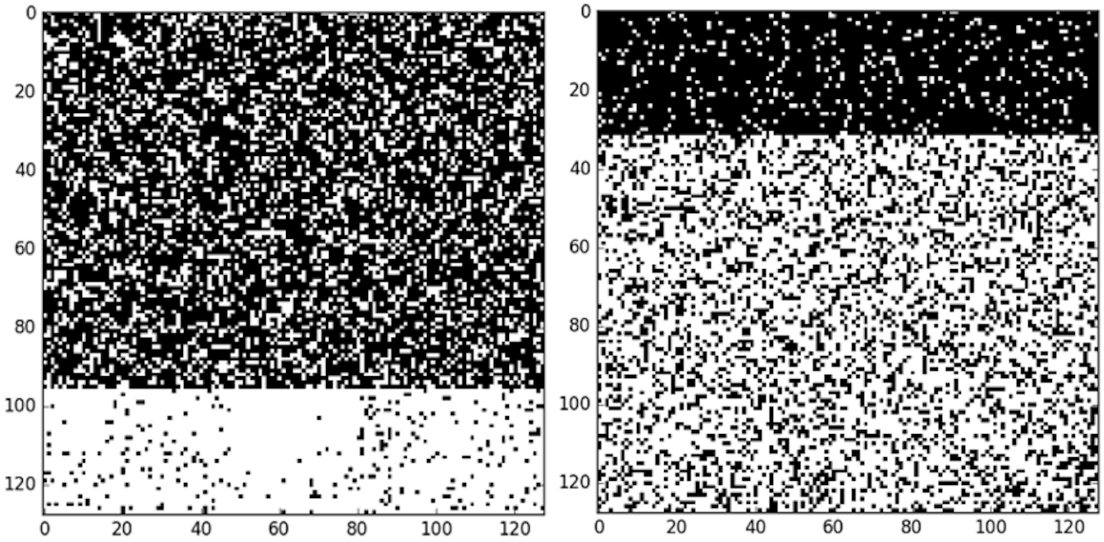


Figure 4.3: Example for unbalanced colored squares. In figure (a) 75% cliques contain more than 85% black pixels while 25% cliques contain more than 85% white pixels. Figure (b) is the opposite of figure (a)

As before, figure 4.4 shows results learned by structural SVM (top row) and latent structural SVM (bottom row). The accuracy performance of both methods are almost the same. Both methods are able to recover $45\% - 50\%$ pixels. The shape of each formulations' results are both preferable and very similar when compared to each other. The most significant difference is the number of linear functions (10 active linear functions v.s. 2). In terms of computational performance, our previous method only takes 10 iterations to converge while the latent structural SVM formulation takes 89 iterations. Our new method is much more computational expensive than our previous method.

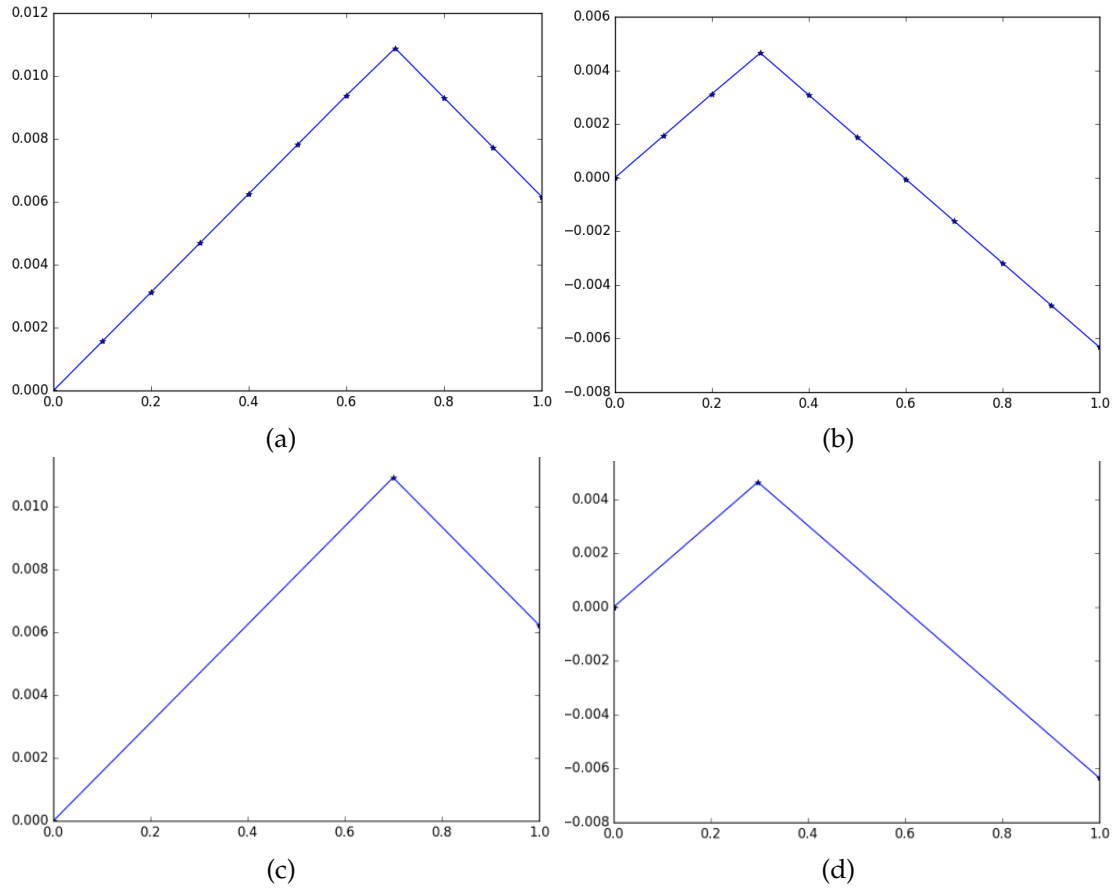


Figure 4.4: Results comparison for unbalanced colored squares. Figure (a) and Figure (b) are lower linear (more black and more white) envelopes learned by structural SVM. Figure (c) and Figure (d) are learned by latent structural SVM.

4.1.4 Uniformly Colored Squares

All of the above experiments show that our new method can significantly simplify the shape of the lower linear envelope function while maintaining the inference performance at the same level. However, one significant cost is the computational performance. It still remains obscure if there exists any other advantages. In this section we design a much harder problem. $W_c(y_c)$ is uniformly distributed from 0 to 1. Figure 4.5 shows the result. The preferable shape of the lower linear envelope should contain a line which is parallel to the x-axis.

Results are shown in figure 4.6. As we can see that shapes are very different between two formulations. Our latent structural formulation (figure 4.6 (b)) learned a very flat representation of the lower linear envelope function, which is much preferable, while the structural SVM formulation preserves much concavity in the shape. This might because in previous work [8, 9] we imposed strict concave constraints on parameter vector $\boldsymbol{\theta}$.

The performance of accuracy also various significantly. Under this formulation

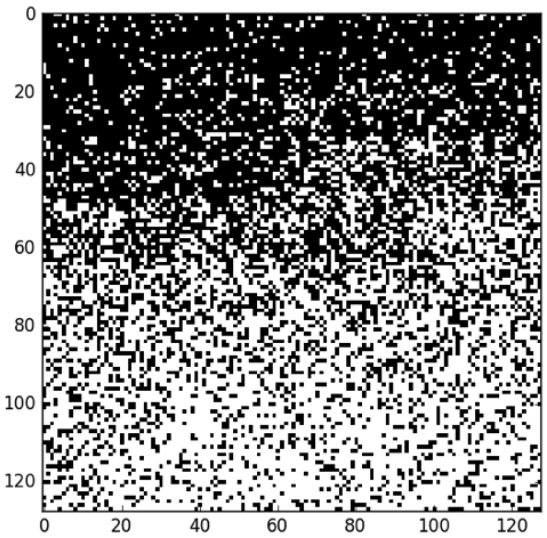
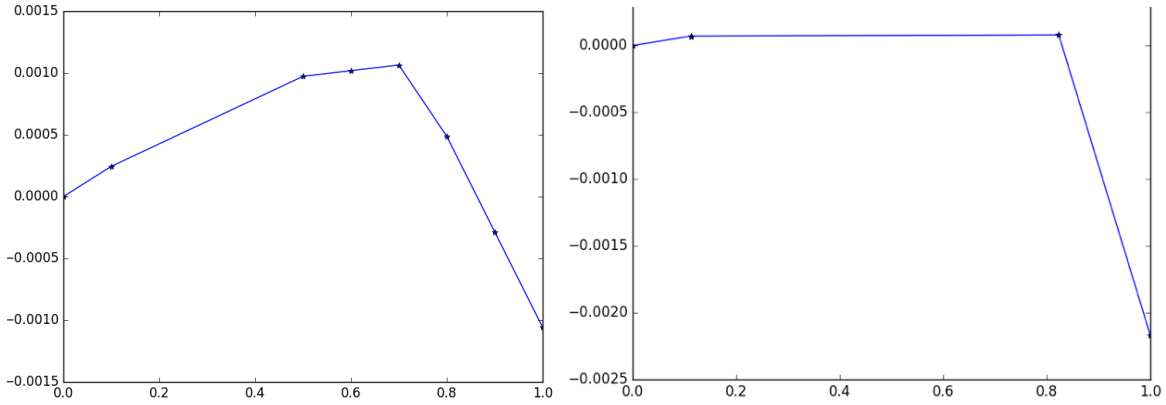


Figure 4.5: Uniformly colored squares example. $W_c(\boldsymbol{y}_c) = \sum_{i \in c} w_i^c y_i$ is uniformly distributed from 0 to 1.

our new method is still able to recover $45\% - 50\%$ pixels while our previous can only recover $25\% - 30\%$ pixels on average. Therefore, our new formulation finally outperforms previous one. In terms of computational performance, the new formulation takes 129 *inner loops* in total (2 *outer loops*) while our previous formulation takes 75 iterations to converge. Although the new formulation is still more computational expensive than previous one, the gap decreases significantly.

We consider all of those improvements are due to our new method is able to learn the lower linear envelope exactly.

One subtle thing is that the linear function on the right side in figure 4.6 (b) decreases sharply which seems abnormally at first glance. The reason is that we assume $b_1 = 0$ in section 3.1.1 which fixes the y-intercept of the first linear function to be zero. Therefore the last linear function can be arbitrarily deep while the first linear function is fixed at the original point.



(a) (b)

Figure 4.6: Results of uniformly colored squares experiment. Figure (a) is the result learned by structural SVM formulation. Figure (b) is the result learned by latent structural SVM formulation.

4.1.5 Conclusions

From above experiments we conclude our findings as followings:

- All of those experiments verified that our latent structural formulation is able learn the lower linear envelope exactly.
- In general (see section 4.1.2 and section 4.1.3), our new method have equivalent accuracy performance to our old method (structural SVM formulation[8, 9]).
- In terms of computational performance, the new formulation is much more computational expensive than the previous one during training. However, it is more efficient during testing due to it simplicity for the lower linear envelope potentials.
- For harder problem (see section 4.1.4), the new method outperforms the previous one significantly. The gap of computational performance also decreases a significant amount.

4.2 Foreground Extraction

To evaluate our method on real-world applications, we repeat the "Interactive Figure-Ground Segmentation" experiment in our previous researches [8, 9]. As described in section 2.2.2, the goal of this experiment is to extract foreground pixels from a bounding box of the object. We follow settings in previous work and replace the max margin algorithm with latent structural SVM Algorithm 3.

4.2.1 Experiment Settings

The data set we use is collected by Lempitsky et al. [19] which contains 50 images (contains 630×480 pixels on average) with ground-truth labels and user-annotated bounding boxes. Following previous work we perform Algorithm 3 by leave-one-out cross-validation on this data set. In order to be comparable with previous results, the performance is measured by *accuracy*. Our energy function for this problem is defined as:

$$E(\boldsymbol{y};\boldsymbol{\theta}) = \theta^U \sum_{i\in\mathcal{N}} \phi^U(\boldsymbol{y}_i) + \theta^P \sum_{(i,j)\in\mathcal{E}} \phi^P(\boldsymbol{y}_i,\boldsymbol{y}_j) + \sum_{\boldsymbol{y}_c\in\mathcal{C}} \phi^H(\boldsymbol{y}_c,\boldsymbol{z}_c;\boldsymbol{\theta}^H) \tag{4.3}$$

As we described in section 2.2.2, unary terms are generated by GMMs (both foreground and background) trained by *GrabCut* [21] algorithm. The unary terms are assigned as following:

UNARY	0	1
$\phi^U(y_i)$	$\phi^U(0,\boldsymbol{k}_i,\boldsymbol{\theta},\boldsymbol{z}_i)$	$\phi^U(1,\boldsymbol{k}_i,\boldsymbol{\theta},\boldsymbol{z}_i)$

Table 4.1: Table of GrabCut unary terms. ϕ^U is taken from equation (2.13). 1 is the label for foreground and 0 for background

The pairwise terms are defined as:

$$\phi^P(\boldsymbol{y}_i,\boldsymbol{y}_j) = \frac{\lambda}{d_{ij}}[[y_i \neq y_j]] exp\left\{ -\frac{||x_i - x_j||^2}{2\beta} \right\}, \text{ for } \forall(i,j)\in\mathcal{E} \tag{4.4}$$

where $\mathcal{E}$ denotes the set of pairs of neighboring pixels which is defined as 8-way adjacent pixels (horizontally, vertically and diagonally). d_{ij} is the Euclidean distance of neighboring pixels. x_i is the RGB value vector. β denotes the expectation over $(x_i - x_j)^2$. The only free parameter λ is learned by Algorithm 3 during cross-validation and determines the strength of the pairwise smoothness term.

The *SLIC* algorithm introduced in section 2.2.1 is used to generate cliques $c \in \mathcal{C}$ for higher order terms. $\phi^H(\boldsymbol{y}_c,\boldsymbol{z}_c;\boldsymbol{\theta}^H) = \boldsymbol{\theta}^{H\,T}\boldsymbol{\phi}(\boldsymbol{y}_c,\boldsymbol{z}_c)$ is equivalent to equation (3.10) and added for each clique c in the image. The number of linear equations K in equation (3.11) is set to be 10.

Then we run the Algorithm 3 with $MaxIter = 100$. Results are shown in section 4.2.2.

4.2.2 Experiment Result

In this section we compare our new latent structural SVM formulation's results with our previous work [9]. On average it takes 5.3 hours to train a cross-validation fold while our previous method only takes 3 hours. For some folds our method can take more than 8 hours and still not converge. Table 4.2 shows comparison between different models:

MODEL	ACCURACY
Baseline	90.9
Structural SVM Formulation	91.6
Latent SSVM Formulation	92.27

Table 4.2: GrabCut experiment results comparison. Data of Baseline model and Structural SVM model is from previous work [9].

It turns out that our new formulation slightly outperforms our previous method by 0.6%. The following table 4.3 shows a more detailed accuracy distribution among 50 images:

ACCURACY INTERVAL	NUMBER OF IMAGES
Over 99.5%	35
$90\% - 99.5\%$	9
$70\% - 90\%$	4
$50\% - 70\%$	1
Below 50%	1

Table 4.3: GrabCut results' accuracy distribution.

From this table we can conclude that our new formulation performs very well on 75% of images in the data set. From figure 4.7 to figure 4.10 show the comparison of results which we illustrated in previous work [9]. Figure in the middle column is the ground-truth foreground image. Figure on the left is the foreground inferred by our previous method. Figure on the right is inferred by our new method.

In figure 4.7, our new method has a very strong results. It almost perfectly extracted the foreground image. The accuracy is over 99.5%. In figure 4.8 our new method still performs better. However, it left the cheetah's tail out compared to our previous method. One important thing to notice is that there are many holes in the image on the left (inferred by our previous method) while the image inferred by our new method is very smooth without holes. This certificates that by learning the lower linear envelope exactly, our new method has a better performance on preserving higher order consistency. In figure 4.9 two methods have almost the same performance. Previous method left the woman's head but was able to capture her full legs while our

Figure 4.7: GrabCut Results.

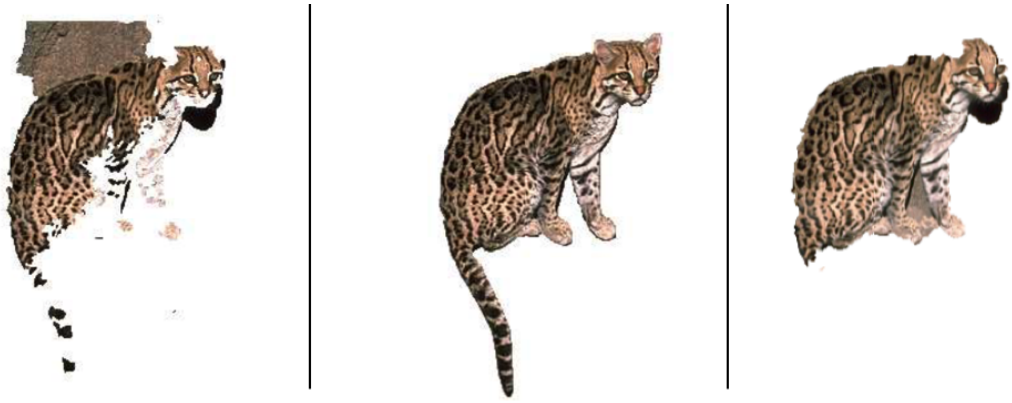


Figure 4.8: GrabCut Results.

new method only able to capture half of her legs but with full head.

However, the performance various significantly between images. In figure 4.10, even though the sculpture inferred by our new method is still very smooth without any hole on it, our method failed to segment the background out of the foreground (there are large amount of pixels belong to stairs and glass). Our model completely fails to learn the image "189080.png" which has the worst performance 42.5% (the Algorithm 3 stopped at the maximum number of iterations which is 10 *outer loops* and 100 *inner loops* for each *outer loop*). The inferred foreground image is completely blank as shown in figure 4.11.

Figure 4.9: GrabCut Results.

Figure 4.10: GrabCut Results.

Figure 4.11: Worst accuracy image. The accuracy is only 42.5%. The foreground inferred is completely blank which means information learned by our model is failed to be generalize to this image.

4.2.3 Conclusions

From above experiments we conclude our findings as followings:

- In general, our new method outperforms previous one. The accuracy is increased by 0.6%. This might because our new method learns the lower linear envelope exactly thus has a richer representation than previous one.
- In general, our new method has a better higher-order consistency. This can be seen from figure 4.7 to figure 4.10. This is also because the lower linear envelope function has better quality than our previous method.
- The new method's performance various differently among images. The performance ranges from 42.5% to 99.5%.
- The new method is more computationally expensive than our previous method during training. It takes 5.3 hours to train one cross-validation fold while previous method only takes 3 hours. However, it is more efficient during testing due to it simplicity for the lower linear envelope potentials.
- There seems like to exist some generalization issues in our Algorithm 3. For example, the new method has bad performance of filtering background in figure 4.10 and it completely fails to infer the figure 4.11.

Chapter 5

Conclusion and Future Work

We summarize our work in this chapter. We will also conclude its advantages and disadvantages basing on our synthetic (section 4.1) and real-world (section 4.2) experiments' results. Our work also provide some insights to our future work which we will briefly discuss in this chapter.

5.1 Conclusion

Lower linear envelope binary MRFs are raising interests due to its capability for encoding higher-order consistency constraints over large sets of random variables. Gould [9] has shown how to perform exact inference and learning of this problem under the max margin framework. In order to transform the lower linear envelope function to a linear combination formulation, they interpolates it with a set of fixed space sample points. Thus their algorithm is only able to learn the shape of the lower linear envelope function approximately.

The main goal of our research is to learn the lower linear envelope function exactly. Based on their work, we explore a variant of their formulation by introducing auxiliary variables back to the energy function to formulate an exact representation. We find that the lower linear envelope function under the quadratic pseudo-Boolean formulation (3.9) itself is an inner product of parameters and features thus can be written into a linear combination directly. Under this formulation the inference algorithm (*st min cut* construction) developed by Gould [9] still adapts to our problem. Therefore, we are still able to conduct exact inference on our problem. We developed the learning Algorithm 3 using an extension of the max margin framework which is known as latent structural SVM. However, this algorithm is only guaranteed to decrease the objective function to a local minimum thus the initial point will affect the overall performance. In order to overcome this issue we also proposed an empirical initialization Algorithm 2.

In order to examine the effectiveness of our new algorithm, we repeat two experiments Gould [9] conducted in their research and compare both results. In the first synthetic checkerboard experiment, we found that in general the new algorithm's accuracy is at least as well as the previous one. But on harder problem 4.1.4 the new method outperforms previous one significantly. The new method is much more com-

putationally expensive during the training period. However it is more efficient during the testing stage because of the simplicity of the shape of the lower linear envelope. We also found that the shape learned by the new method can shift along with the changes of the input data which proves that we can learn the lower linear envelope exactly.

We then take our algorithm to a harder real-world experiment 4.2. It turns out that our new method has a slightly increasing in overall accuracy (0.6%) compared to the previous method. There are much less holes in images inferred by our new method which certificates the lower linear envelope function learned by our formulation can better enforce higher order consistency in large cliques. However, the performance various significantly between cross-validation folds which indicates there are some generalization issues existing in our new method.

At last we summarize the advantages and disadvantages as following.

Advantages of the new method (compared to the previous method [8, 9]):

- Able to learn the lower linear envelope exactly.
- Performs better (higher accuracy) on harder problems.
- Efficient to compute during testing due to the simplicity of the shape of the lower linear envelope function.

Disadvantages of the new method:

- Only guaranteed to decrease to the local minimum.
- Computationally expensive during training.
- Generalization various significantly.

5.2 Future Work

As we suggested in the conclusion, our new method seems to have some generalization issues (figure 4.11 for example). It will be our primal goal to keep investigating into this problem. We also proposed an empirical initialization method in section 3.2.2. For future work we would compare this method to others.

Our research also provide insights to further directions. Extending our approach to multi-label MRFs seems to be very promising. Other straightforward extensions include the introduction of features for modulating the higher-order terms and the use of dynamic graph cuts [14] for accelerating loss-augmented inference within our learning framework. Other optimization algorithms for solving our learning problem may also be considered, e.g., the subgradient method [2, 20].

Bibliography

1. R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.
2. D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2004.
3. C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
4. E. Boros and P. L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123:155–225, 2002.
5. Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2001.
6. P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.
7. 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.
8. 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.
9. S. Gould. Learning weighted lower linear envelope potentials in binary markov random fields. *IEEE transactions on pattern analysis and machine intelligence*, 37(7): 1336–1346, 2015.
10. P. L. Hammer. Some network flow problems solved with psuedo-boolean programming. *Operations Research*, 13:388–399, 1965.
11. H. Ishikawa. Exact optimization for Markov random fields with convex priors. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 25:1333–1336, 2003.
12. H. Ishikawa. Higher-order clique reduction in binary graph cut. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.
13. 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.
14. 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.
15. 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.
16. P. Kohli, L. Ladicky, and P. H. S. Torr. Graph cuts for minimizing higher order

potentials. Technical report, Microsoft Research, 2008.
17. P. Kohli, P. H. Torr, et al. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.
18. 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.
19. 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.
20. 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.
21. 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.
22. 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.
23. M. Szummer, P. Kohli, and D. Hoiem. Learning CRFs using graph-cuts. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2008.
24. 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.
25. I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research*, pages 1453–1484, 2005.
26. C.-N. J. Yu and T. Joachims. Learning structural svms with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1169–1176. ACM, 2009.
27. A. L. Yuille, A. Rangarajan, and A. Yuille. The concave-convex procedure (cccp). *Advances in neural information processing systems*, 2:1033–1040, 2002.