Chapter 1

Related Work and Background

1.1 MRFs and Energy Function

1.1.1 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 [1]. 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:

$$p(\boldsymbol{y}) = \frac{1}{Z}\prod_C \Psi_C(\boldsymbol{y}_C) \tag{1.1}$$

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:

$$p(\boldsymbol{y}) = \frac{1}{Z}exp(-\sum_C E_C(\boldsymbol{y}_C)) \tag{1.2}$$

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*:

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\text{argmax}}\, p(\boldsymbol{y}) = \underset{\boldsymbol{y}}{\text{argmin}}(-\sum_C E_C(\boldsymbol{y}_C)) \tag{1.3}$$

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 $\mathcal{C}$ [20] 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:

$$E(\boldsymbol{y}; \boldsymbol{w}) = \sum_{i \in \mathcal{N}} \boldsymbol{w}_i^U \phi^U(\boldsymbol{y}_i) + \sum_{(i,j) \in \mathcal{E}} \boldsymbol{w}_{ij}^P \phi^P(\boldsymbol{y}_i, \boldsymbol{y}_j) + \sum_{\boldsymbol{y}_C \in \mathcal{C}} \boldsymbol{w}_C^H \phi^H(\boldsymbol{y}_C) \tag{1.4}$$

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:

$$E(y_t, w) \leq E(y, w) \, , \forall y \neq y_t, y \in \mathcal{Y} \tag{1.5}$$

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 [20]:

$$\boldsymbol{w}^* = argmax_w(E(y_t, w) - E(y, w)) \, , \forall y \neq y_t, y \in \mathcal{Y} \tag{1.6}$$

So far 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 generalization of k-means clustering known as Gaussian Mixture Models to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians. The grabcut [18] algorithm is used to train GMMs and the final results are used to construct the unary terms. For pairwise terms, we use the Potts model introduced by Kohli et al. [14] to encode pairwise consistency. The *inference* problem is solved by using a graph-cut [3, 4] algorithm and the max margin framework [24] has been addressed to learn parameters of the energy function.

1.1.2 Construction of stocks relationship graph

The stock market is a complicated system with many variables affecting whether or not a particular stock price goes up or down. In order to accurately capture the complex relationships between companies that affect their stock prices, they can be represented as an undirected graph (MRFs). Each node in the graph corresponds to a single company and each edge represents collaboration between a pair of companies. This work uses the companies appearing in the information technology sector of the *HS300* index. Because the relationship of collaborating companies is many-to-many, this would normally be an intractable problem if inference were determined using a message-passing algorithm (the complexity is exponential in the tree-width of the graph). Fortunately, the collaboration relationship is an associative one (i.e. if two companies are collaborators, an increase in the stock price of one of them is likely associated with an increase for the other; likewise, a decrease in the stock price of one of

them is likely associated with a decrease for the other). As mentioned earlier, we are able to use minimum cuts to solve inference queries in polynomial time. Edges in the graph are determined by comparing the estimated result count in the *Sougo* search engine for the two queries:

1. Company A-Company B collaboration
2. Company A-Company B competition

If the first query has more than 1.5 times the results of the second, an edge is added to the graph connecting the nodes for Company A and Company B. This multiplier is necessary to avoid adding edges between companies that are either not strong collaborators or competitors and thus have extremely similar result counts for both queries. Our spider code together with graph data are available on Github `https://github.com/spacegoing/sogou_spider`. The unary terms are measured using GMMs and pairwise terms are measured using Potts model, requiring the SSVM to learn the labelling from the feature vectors at the nodes.

1.1.3 Configuration of Energy Function

In this section we described the configuration of our energy function. We mainly introduce the *GrabCut* algorithm which we used to generate our unary terms. The *Potts model* is used as our pairwise terms. Thus our energy function can be written as:

$$E(\boldsymbol{y}; \boldsymbol{w}) = \theta^U \sum_{i \in \mathcal{N}} \boldsymbol{w}_i^U \boldsymbol{\phi}^U(\boldsymbol{y}_i) + \theta^P \sum_{(i,j) \in \mathcal{E}} \boldsymbol{w}_{ij}^P \boldsymbol{\phi}^P(\boldsymbol{y}_i, \boldsymbol{y}_j) \tag{1.7}$$

where θ^U and θ^P are parameters to be optimized.

The *GrabCut* algorithm was proposed by Rother et al. [18] in order to solve background foreground segmentation problem (see figure 1.1). They first defined MRFs over an labeled image and then use *graph-cuts* [5] method to do the inference. In this section we use 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.

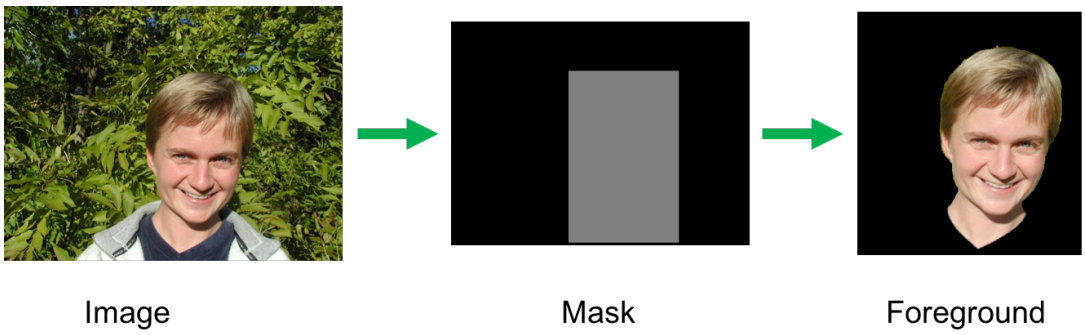

Figure 1.1: Picture on the left is the original picture. Picture on the middle is a user defined mask. The task is to extract foreground pixels within that rectangle. On the right is the ground truth foreground.

Algorithm 1 GrabCut training algorithm

1: **repeat**
2: *Assign GMM components* to stocks:
$\boldsymbol{k}_i^* = \text{argmin}_{\boldsymbol{k}_i} \psi^U(\alpha_i, \boldsymbol{k}_i, \boldsymbol{\theta}, \boldsymbol{z}_i)$
3: *Learn GMM parameters* from data z:
$\boldsymbol{\theta} = \text{argmin}_{\boldsymbol{\theta}} \sum_{i \in \mathcal{N}} \psi^U(\alpha_i, \boldsymbol{k}_i, \boldsymbol{\theta}, \boldsymbol{z}_i)$
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 (1.4) with unary and pairwise terms:

$$G(\alpha, \boldsymbol{k}, \boldsymbol{\theta}, \boldsymbol{z}) = \sum_{i \in \mathcal{N}} \psi^U(\alpha_i, \boldsymbol{k}_i, \boldsymbol{\theta}, \boldsymbol{z}_i) + \sum_{(i,j) \in \mathcal{E}} \psi^P(\alpha_i, \boldsymbol{z}_i) \tag{1.8}$$

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.

In our configuration, we use the graph described in 1.1.2 in replacement of the image. Each node in the graph represents a stock instead of pixel and the edge between stocks represents their pairwise relationship. The label $\alpha \in 0, 1$ equals 1 when the stock price has a positive movement, and vice versa. $\boldsymbol{z}$ denotes the stock's market price vector instead of pixel's RGB value. Other parameters are the same with their configurations.

The *Gaussian Mixture Models* (GMMs) with K components (typically $K = 5$) is used for generating unary terms. Two GMMs, one for positive movement and one for negative movement, are jointly trained by the algorithm. $\boldsymbol{k} = \boldsymbol{k}_1, \ldots, \boldsymbol{k}_i, \ldots, \boldsymbol{k}_N$ with $\boldsymbol{k}_i \in 1, \ldots, K$ assigns each stock (node) i to a unique GMMs component. The component is either belonging to positive movement's GMMs or negative movement'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* [18].

The pairwise function ψ^P in (1.8) is defined as a smoothness indicator which measures both feature vector (stock price vector) and spatial distances (graph 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, a two-step learning algorithm is used. 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 1 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 use GMMs trained by GrabCut algorithm for our unary terms ϕ^U in equation (1.7).

The pairwise function ϕ^P in our energy function (1.7) is defined as a Potts model:

$$\phi^P(\boldsymbol{y}_i, \boldsymbol{y}_j) = \begin{cases} 0 & \text{if} x_i = x_j \\ \psi^P_{ij}(y_i, y_j) = \frac{\lambda}{d_{ij}} [\![y_i \neq y_j]\!] \exp\left\{ -\frac{\|x_i - x_j\|^2}{2\beta} \right\} & \text{otherwise} \end{cases} \tag{1.9}$$

where d_{ij} is the graph distance between stocks i and j. x_i and x_j are stock market price vectors.

1.1.4 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, 19] are able to be minimized exactly using *graph-cuts* like algorithms [7, 10] when they satisfy submodularity condition [2]. Kohli et al. [15] 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 for energy function containing unary and pairwise potentials.

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

$$E^c(\boldsymbol{y}_c, \boldsymbol{z}) = a_1 W_c(\boldsymbol{y}_c) + b_1 + \sum_{k=1}^{K-1} z_k \left((a_{k+1} - a_k) W_c(\boldsymbol{y}_c) + b_{k+1} - b_k \right) \tag{1.10}$$

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. Another important property of optimized $\boldsymbol{z}$ under this formulation is that it automatically satisfies the constraint [9]:

$$z_{k+1} \leq z_k \tag{1.11}$$

In order to construct the *st-min-cut* graph, Gould [9] rewrote equation (1.10) into *posiform* [2]:

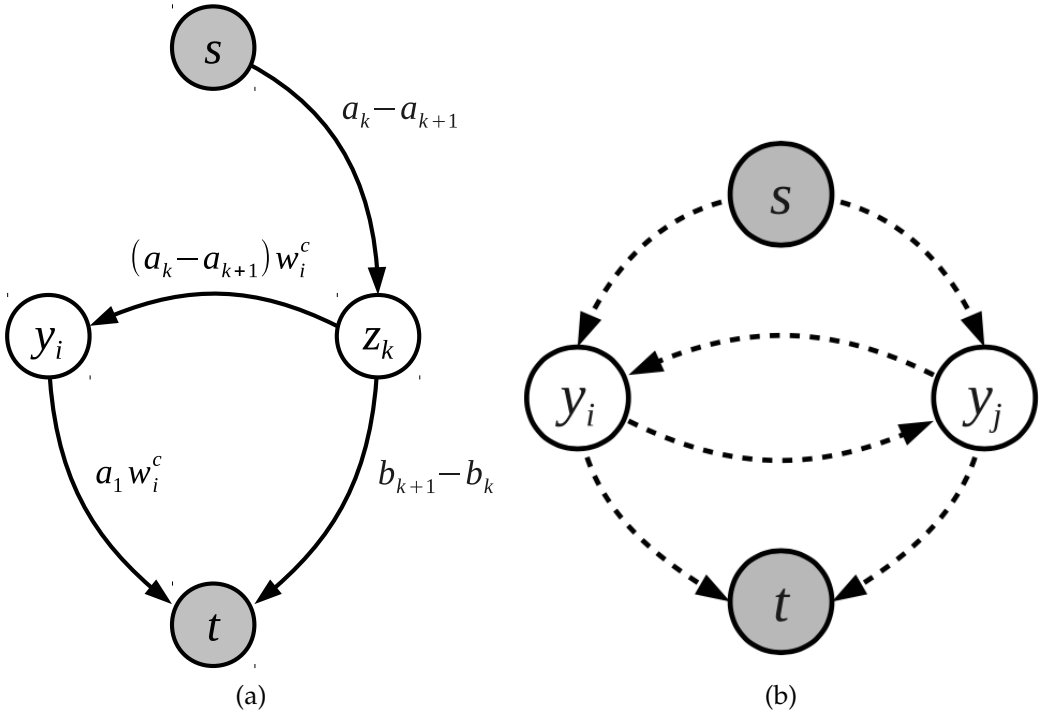

Figure 1.2: *st*-graph construction [9] for equation (1.12), unary and pairwise terms. Every cut corresponds to an assignment to the random variables, where variables associated with nodes in the $\mathcal{S}$ set take the value one, and those associated with nodes in the $\mathcal{T}$ set take the value zero. With slight abuse of notation, we use the variables to denote nodes in our graph.

$$E^c(\boldsymbol{y}_c, \boldsymbol{z}) = b_1 - (a_1 - a_K) + \sum_{i \in c} a_1 w_i^c y_i + \sum_{k=1}^{K-1} (b_{k+1} - b_k) z_k + \sum_{k=1}^{K-1} (a_k - a_{k+1}) \bar{z}_k + \sum_{k=1}^{K-1} \sum_{i \in c} (a_k - a_{k+1}) w_i^c \bar{y}_i z_k \tag{1.12}$$

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. After proving *submodularity* of the energy function (1.12), Gould [9] constructed the *st-min-cut* graph based on equation (1.12).

The construction is explained in Figure 1.2. Figure (a) denotes construction for equation (1.12). Figure (b) denotes construction for unary and pairwise terms (see [16]). 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.

Chapter 2

Methodology

2.1 Learning the MRFs

2.1.1 Structural SVMs

Traditional Support Vector Machines (SVMs) is used to solve binary classification problem. Crammer and Singer [6] extended SVM into multiclass classifier by generalizing the concept of *margin* to measure multiclass distances and a quadratic objective function was constructed. To approach structural prediction Tsochantaridis et al. [24] extends their approach by specifying discriminant functions that exploit the structure and dependencies within label space $\mathcal{Y}$. In this section we briefly summarize their work.

Suppose we are given a training set of input-output structure pairs $S = \{(x_1, y_1), ..., (x_n, y_n)\} \in (\mathcal{X} \times \mathcal{Y})^n$. We want to learn a linear prediction rule of the form

$$f_w(x) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} [\boldsymbol{w} \cdot \boldsymbol{\phi}(x, y)] \tag{2.1}$$

where $\boldsymbol{\phi}$ is a joint feature vector that describes the relationship between input x and structured output y, with $\boldsymbol{w}$ being the parameter vector. The optimization problem of computing this argmax is typically referred to as the *inference* problem.

When training Structural SVMs, the parameter vector $\boldsymbol{w}$ is determined by minimizing the (regularized) risk on the training set $(x_1, y_1), ..., (x_n, y_n)$. Risk is measured through a user-supplied loss function $\Delta(y, \hat{y})$ that quantifies how much the prediction $\hat{y}$ differs from the correct output y. Note that Δ is typically nonconvex and discontinuous and there are usually exponentially many possible structures $\hat{y}$ in the output space $\mathcal{Y}$. The Structural SVM formulation[24] overcomes these difficulties by replacing the loss function Δ with a piecewise linear convex upper bound (margin rescaling)

$$\Delta(y_i, \hat{y}_i(\boldsymbol{w})) \leq max_{\hat{y} \in \mathcal{Y}} \left[\Delta(y_i, \hat{y}) + \boldsymbol{w} \cdot \boldsymbol{\phi}(x_i, \hat{y}) \right] - \boldsymbol{w} \cdot \boldsymbol{\phi}(x_i, y_i) \tag{2.2}$$

where $\hat{y}_i(\boldsymbol{w}) = \operatorname{argmax}_{y \in \mathcal{Y}} \boldsymbol{w} \cdot \boldsymbol{\phi}(x_i, y)$.

To train Structural SVMs we then solve the following convex optimization prob-

lem

$$min_{\boldsymbol{w}} \frac{1}{2}||\boldsymbol{w}||^2 + \mathcal{C} \sum_{i=1}^{n} \left[max_{\hat{y} \in \mathcal{Y}} \left[\Delta(y_i, \hat{y}) + \boldsymbol{w} \cdot \boldsymbol{\phi}(x_i, \hat{y}) \right] - \boldsymbol{w} \cdot \boldsymbol{\phi}(x_i, y_i) \right] \tag{2.3}$$

Despite the typically exponential size of $\mathcal{Y}$, this optimization problem can be solved efficiently using cutting-plane or stochastic gradient methods. Structural SVMs give excellent performance on many structured prediction tasks, especially when the model $\boldsymbol{\phi}$ is high-dimensional and it is necessary to optimize to non-standard loss functions Δ.

2.1.2 Learning under the large margin framework

We now show how the max-margin framework can be used to learn parameters of our MRFs. We begin by reviewing a variant of the max-margin framework introduced by Tsochantaridis et al. [22] and Taskar et al. [21].

Given an energy function $E(\boldsymbol{y}; \boldsymbol{\theta}) = \boldsymbol{\theta}^T \boldsymbol{\phi}(\boldsymbol{y})$ parameterized as a linear combination of features $\boldsymbol{\phi}(\boldsymbol{y}) \in \mathbb{R}^m$ 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, MAXMARGINQP$\left(\{\boldsymbol{y}_t, \mathcal{Y}_t\}_{t=1}^T, \boldsymbol{G}, \boldsymbol{h}\right)$:

$$\begin{aligned} &\text{minimize} \quad \tfrac{1}{2}\|\boldsymbol{\theta}\|^2 + \tfrac{C}{T} \textstyle\sum_{t=1}^T \xi_t \\ &\text{subject to} \\ &\quad \boldsymbol{\theta}^T \delta\boldsymbol{\phi}_t(\boldsymbol{y}) + \xi_t \geq \Delta(\boldsymbol{y}, \boldsymbol{y}_t), \quad \forall t, \boldsymbol{y} \in \mathcal{Y}_t, \\ &\quad \xi_t \geq 0, \qquad\qquad\qquad\qquad \forall t, \\ &\quad \boldsymbol{G\theta} \geq \boldsymbol{h} \end{aligned} \tag{2.4}$$

where $\delta\boldsymbol{\phi}_t(\boldsymbol{y}) \triangleq (\boldsymbol{\phi}_t(\boldsymbol{y}) - \boldsymbol{\phi}_t(\boldsymbol{y}_t))$ 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 in-

Algorithm 2 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 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 $\hat{\boldsymbol{\theta}}$

crementally. 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 when no more violated constraints are found (see Algorithm 2).

Note that while we use the Hamming loss in this work, the loss function $\Delta(\boldsymbol{y}, \boldsymbol{y}_t)$ in Equation 2.4 can be more general. For example, Pletscher and Kohli [17] 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.

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. [23] 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.

Chapter 3

Experiments

Bibliography

1. C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
2. E. Boros and P. L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123:155–225, 2002.
3. 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.
4. 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.
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. K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002.
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, 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.
15. P. Kohli, L. Ladicky, and P. H. S. Torr. Graph cuts for minimizing higher order potentials. Technical report, Microsoft Research, 2008.
16. 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.

17. 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.
18. 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.
19. 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.
20. M. Szummer, P. Kohli, and D. Hoiem. Learning CRFs using graph-cuts. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2008.
21. 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.
22. 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.
23. 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.
24. 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.