UNIVERSITY OF CALIFORNIA,
IRVINE

# Learning and Inference in Latent Variable Graphical Models

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Wei Ping

Dissertation Committee:
Professor Alexander Ihler, Chair
Professor Charless Fowlkes
Professor Sameer Singh

2016

# DEDICATION

To my grandma,
my parents and my honey bunny forever.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ALGORITHMS

# ACKNOWLEDGMENTS

First and foremost, I want to thank my amazing advisor Prof. Alexander Ihler. He is the best advisor I can ever imagine. He points out important research directions for me when I started to do research with him. He can instantly point out the logical fallacies in my thoughts and give insightful suggestions at the same time. He always encourages me to do the best, and he is always supportive when I need him most. I really appreciate him in these years. For this thesis, Alex have devoted a lot of time to read and edit my draft word by word. Due to my extreme procrastination of finishing the draft, he even gave up his Thanksgiving holidays, appeared at the office in a rainy Saturday and discussed with me about the problems in my draft. I am just so lucky and honoured to be his student.

I would like to thank the two other members of my committee – Prof. Charless Fowlkes and Prof. Sameer Singh. Charless is also on my advancement committee and gives me a lot of constructive suggestions for my research, especially for future directions. I met Sameer and we have lunch together when he visit UCI at the first time. He is very insightful on analysing the new ideas from various perspectives, and always gives inspired comments. In particular, I really appreciate him for saving me at the last minute.

I would also like to thank my peers and collaborators. In particular, I want to thank Qiang Liu who guided me into the area of graphical models, and always gives me constructive suggestions when I encounter difficulties in research. Thanks to Qi Lou who spent huge amount of time with me at DBH 4051. Also, thanks to Nick Gallo for all those refreshing discussions on various topics. I would like to thank all my close friends Mengfan Tang, Yi Li, Yuxiao Wang, Qiguang Wang and Zhengqiu Huang. I will never forget those nights when we are dead drunk. There are so many names I want to mention, so I will simply thank all my friends over the years.

Most importantly, I would like to thank my parents for their unconditional love at any time at any situation. Finally, I want thank my honey bunny Lingjie Weng without whom I could never make it to the end.

My graduate work and this thesis have been supported by NSF grants IIS-1065618 and IIS-1254071.

# CURRICULUM VITAE

## Wei Ping

### EDUCATION

**Doctor of Philosophy in Computer Science** **2016**
University of California, Irvine Irvine, California

**Master of Software Engineering** **2011**
Tsinghua University China

**Bachelor of Computer Science** **2008**
Harbin Institute of Technology China

### RESEARCH EXPERIENCE

**Graduate Research Assistant** **2011–2016**
University of California, Irvine Irvine, California

**Research Intern** **08/2009 – 02/2011**
Microsoft Research Asia Beijing, China

### ACADEMIC REVIEWING

| | |
|---|---|
| IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) | 2015-2016 |
| IEEE Transactions on Signal processing | 2016 |
| Conference on Neural Information Processing Systems (NIPS) | 2016 |
| Conference on Uncertainty in Artificial Intelligence (UAI) | 2016 |
| AAAI Conference on Artificial Intelligence (AAAI) | 2015 |

### PUBLICATIONS

**W. Ping**, A. Ihler. Belief Propagation in Conditional RBMs for Structured Prediction. *In Submission*, 2016.

**W. Ping**, Q. Liu, A. Ihler. Learning Infinite RBMs with Frank-Wolfe. *Neural Information Processing Systems (NIPS)* , 2016.

**W. Ping**, Q. Liu, A. Ihler. Decomposition Bounds for Marginal MAP. *Neural Information Processing Systems (NIPS)* , 2015.

**W. Ping**, Q. Liu, A. Ihler. Marginal structured SVM with hidden variables. *International Conference on Machine Learning (ICML)*, 2014.

Y. Xu, **W. Ping**, A.T. Campbell. Multi-instance Metric Learning. *IEEE International Conference on Data Mining (ICDM)*, 2011.

**W. Ping**, Y. Xu, A.T. Campbell. FAMER: Making Multi-Instance Learning Better and Faster. *SIAM International Conference on Data Mining (SDM)*, 2011.

Y. Xu, Furao Shen, **W. Ping**, Jinxi Zhao. TAKES: a Fast Method to Select Features in the Kernel Space. *ACM International Conference on Information and Knowledge Management (CIKM)*, 2011.

Y. Cai, L. Yang, **W. Ping**, F. Wang, T. Mei, X.S. Hua, S. Li. Million-scale Near-duplicate Video Retrieval System. *ACM International Conference on Multimedia (MM)*, 2011.

**W. Ping**, Y. Xu, K. Ren, C.H. Chi, F. Shen. Non-I.I.D. Multi-Instance Dimensionality Reduction by Learning a Maximum Bag Margin Subspace *AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

# ABSTRACT OF THE DISSERTATION

Learning and Inference in Latent Variable Graphical Models

By

Wei Ping

Doctor of Philosophy in Computer Science

University of California, Irvine, 2016

Professor Alexander Ihler, Chair

Probabilistic graphical models such as Markov random fields provide a powerful framework and tools for machine learning, especially for structured output learning. Latent variables naturally exist in many applications of these models; they may arise from partially labeled data, or be introduced to enrich model flexibility. However, the presence of latent variables presents challenges for learning and inference.

For example, the standard approach of using *maximum a posteriori* (MAP) prediction is complicated by the fact that, in latent variable models (LVMs), we typically want to first marginalize out the latent variables, leading to an inference task called *marginal MAP*. Unfortunately, marginal MAP prediction can be NP-hard even on relatively simple models such as trees, and few methods have been developed in the literature. This thesis presents a class of variational bounds for marginal MAP that generalizes the popular dual-decomposition method for MAP inference, and enables an efficient block coordinate descent algorithm to solve the corresponding optimization. Similarly, when learning LVMs for structured prediction, it is critically important to maintain the effect of uncertainty over latent variables by marginalization. We propose the marginal structured SVM, which uses marginal MAP inference to properly handle that uncertainty inside the framework of max-margin learning.

We then turn our attention to an important subclass of latent variable models, restricted Boltzmann machines (RBMs). RBMs are two-layer latent variable models that are widely used to capture complex distributions of observed data, including as building block for deep probabilistic models. One practical problem in RBMs is model selection: we need to determine the hidden (latent) layer size before performing learning. We propose an infinite RBM model and apply the Frank-Wolfe algorithm to solve the resulting learning problem. The resulting algorithm can be interpreted as inserting a hidden variable into a RBM model at each iteration, to easily and efficiently perform model selection during learning. We also study the role of approximate inference in RBMs and conditional RBMs. In particular, there is a common assumption that belief propagation methods do not work well on RBM-based models, especially for learning. In contrast, we demonstrate that for conditional models and structured prediction, learning RBM-based models with belief propagation and its variants can provide much better results than the state-of-the-art contrastive divergence methods.

Chapter 1

# Introduction

Probabilistic approaches play central roles in modern developments and analyses of machine learning methods [e.g., Murphy, 2012, Friedman et al., 2001]. Among all probabilistic techniques, *probabilistic graphical models*, such as Markov random fields (MRFs) and Bayesian networks, provide a unified framework and powerful computation tools for probabilistic modeling. In real-world modeling applications, a large part of useful graphical models are latent variable models (LVMs). This is mainly because (1) partially labeled data or missing values widely exist in practice, and (2) latent variables have been widely used to capture the complex high-order correlations among observable data. The presence of latent variables presents challenges for learning, inference and model selection. In this chapter, we give an overview of the topics and methodologies that are covered in this thesis.

## 1.1 Inference in Graphical Models

A graphical model defines a probability distribution over a set of random variables, which uses a graph-based representation to encode the relationships among variables (i.e, conditional independences) and organize the required computation. Given a graphical model, *inference* refers to answering probabilistic queries about the model. There are three common types of inference tasks. The first are max-inference or maximum a *posteriori* (MAP) tasks, which

aim to find the most probable state of the joint probability; exact and approximate MAP inference is widely used in structured prediction, as we will discuss later. Sum-inference tasks include calculating marginal probabilities and the normalization constant of the distribution, and play a central role in many learning tasks (e.g., maximum likelihood estimation of MRFs). Finally, marginal MAP tasks naturally arise in latent variable models (LVMs) [e.g., Ping et al., 2014, Naradowsky et al., 2012], in which one need to find the optimal MAP prediction or estimation with latent variables marginalized. They are "mixed" inference problems, which generalize the first two types by marginalizing a subset of variables (i.e., latent variables) before optimizing over the remainder. [1] All three inference types are generally intractable but marginal MAP is more challenging on both theoretical and practical side; the computational complexity of marginal MAP is $\text{NP}^{\text{PP}}$-complete [Park and Darwiche, 2004], which is believed to be harder than MAP inference (NP-hard) and sum-inference (#P-complete), and marginal MAP tasks can be intractable even on tree structured model. As a result, approximate inference, particularly convex relaxations or upper bounding methods, are of great interest.

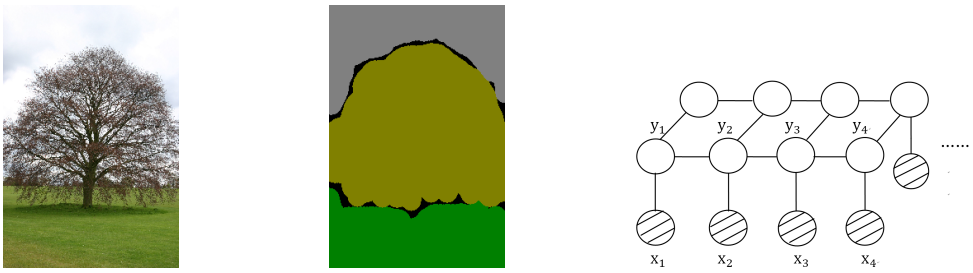
Dual decomposition methods for MAP [e.g., Sontag et al., 2011] give a class of fully decomposed upper bounds which can be directly optimized using coordinate descent algorithms [e.g., Werner, 2007, Globerson and Jaakkola, 2008]. It is easy to ensure both convergence, and that the objective is monotonically decreasing (so that more computation always provides a better bound). Given these desirable properties, it is of great interest to investigate dual decomposition methods for other inference tasks. In particular, marginal MAP is significantly more difficult than MAP and pure marginalization task, and far fewer methods have been developed. In this thesis, we propose a full decomposition bound which is applicable for both marginal inference and marginal MAP inference. We derive block coordinate descent algorithm which ensures convergence and monotonicity.

[1] In some literature [e.g., Park and Darwiche, 2004], marginal MAP is simply called MAP, and the joint MAP task is called MPE.

## 1.2 Structured Output Learning

Structured output learning or structured prediction is one of the most important application domain of graphical models in machine learning, where one need predict a set of correlated variables. In specific, one need to learn a structured predictor $f : \boldsymbol{x} \to \boldsymbol{y}$ using some training data $\{\boldsymbol{x}^n, \boldsymbol{y}^n\}_{n=1}^N$, while both the mapping between input-output pair $(\boldsymbol{x}, \boldsymbol{y})$ and correlations among the output variables are modeled by a probabilistic graphical model. For example, in semantic segmentation task from computer vision, given an input image $\boldsymbol{x}$, we need to classify each pixel as a semantic category. The semantic labels $\boldsymbol{y}$ of these pixels are highly correlated, so we need to make joint predictions. See Figure 1.1(a)-(b) for an example from Microsoft Research Cambridge (MSRC) dataset [Winn et al., 2005]. In part-of-speech (POS) tagging from natural language processing, given a sequence of words, one need to classify each word into a particular POS, and those POS tags are also highly correlated.

Conditional random fields (CRFs) [Lafferty et al., 2001] and structured support vector machines (SSVMs) [Taskar et al., 2003, Tsochantaridis et al., 2005] are standard tools for structured prediction in many important domains, such as computer vision [Nowozin and Lampert, 2011], natural language processing [Getoor and Taskar, 2007] and computational biology [e.g., Li et al., 2007, Sato and Sakakibara, 2005]. See Figure 1.1(c) for an illustration of the popular CRF model for image segmentation, which explicitly encodes the correlations among the pixels through the grid structure. However, many practical cases are not well handled by these tools, due to the presence of latent variables. We will discuss the latent variable problem in next section.




(a) input image $\boldsymbol{x}$ (b) pixel-wise labellings $\boldsymbol{y}$ (c) grid structured CRF

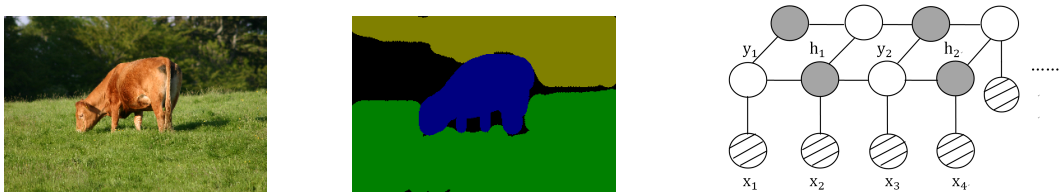
Figure 1.1: (a) An example image from MSRC dataset. (b) Pixel-wise labellings including sky, grass and tree. (c) Grid structured CRF in which the hatching nodes $\boldsymbol{x}$ represent input features, and white nodes $\boldsymbol{y}$ represent output variables.

## 1.3 Latent Variable Models

Latent variables exist in many practical applications of graphical models. They may come from the missing information of partially labeled data; in previous image segmentation example, while it is really expensive to collect labels for every single pixel, especially for boundaries of objects and ambiguous regions (see Figure 1.2(a)-(b) for an example), partially labeled data are relatively easy to obtain [e.g., Verbeek and Triggs, 2007]. On the other hand, latent variables are very important for modeling purpose, because they can either capture the high-order correlations between the observed variables, or they directly represent some important latent/unobserved factors in real world.

**LVMs for Structured Prediction.** Many latent variable models (LVMs), such as hidden conditional random fields [e.g., Quattoni et al., 2007b], have been proposed for structured prediction. See Figure 1.2(c) for an illustration of CRF with hidden variables $\boldsymbol{h}$.

Latent structured SVMs (LSSVMs) [Yu and Joachims, 2009a], which are extended from structured SVMs [Taskar et al., 2003, Tsochantaridis et al., 2005], are among the most popular learning methods for these models. It often outperforms its counterparts in many practical applications, especially when the model assumptions are violated or the number of training data is limited [e.g., Taskar et al., 2003]. However, LSSVM relies on a joint




(a) input image $\boldsymbol{x}$ (b) pixel-wise labellings $\boldsymbol{y}$ (c) CRF with hidden variables $\boldsymbol{h}$

Figure 1.2: (a) An example image from MSRC dataset. (b) Pixel-wise labellings including cow, grass and tree. The black region represents missing labels. (c) CRF with hidden variables (shaded nodes) representing missing labels.

maximum a posteriori (MAP) inference which assigns the latent variables $\boldsymbol{h}$ to deterministic values, and does not take into account their uncertainty. It can produce poor predictions of output variables $\boldsymbol{y}$ even for exact models [Liu and Ihler, 2013] at test phase, and does not maintain the uncertainty of latent variables during learning. A better approach for prediction is marginal MAP inference that averages over possible states of latent variables before optimizing over output variables. In this thesis, we propose a marginal structured SVM framework, which properly accounts for the uncertainty of latent variables during learning by incorporating the marginal MAP inference into the max-margin paradigm.

**Restricted Boltzmann Machines.** Another popular class of latent variable models are restricted Boltzmann machines (RBMs). RBMs are two-layer models that use a layer of hidden variables to model the distribution of observable variables [Smolensky, 1986, Hinton, 2002b]. In the literature on RBMs, hidden and observable variables are referred to as hidden units $\boldsymbol{h}$ and visible units $\boldsymbol{v}$, respectively. RBMs have been widely used to capture complex distributions in numerous application domains, including image modeling [Krizhevsky et al., 2010], human motion capture [Taylor et al., 2006b] and collaborative filtering [Salakhutdinov et al., 2007b], and are also widely used as building blocks for deep generative models, such as deep belief networks [Hinton et al., 2006b] and deep Boltzmann machines [Salakhutdinov and Hinton, 2009]. Due to the intractability of the likelihood function, RBMs are usually learned using the contrastive divergence (CD) algorithm [Hinton, 2002b, Tieleman, 2008],

which approximates the gradient of the likelihood using a Gibbs sampler.

An important model selection problem for RBM is that we need to determine the size of the hidden layer (number of hidden units), and it is challenging to decide what is the optimal size. One heuristic is to search the "best" size of hidden layer using cross validation or testing likelihood within a pre-defined candidate set. Unfortunately, this is extremely time consuming; it involves running a full training algorithm (e.g., CD) for each possible size, and thus we can only search over a relatively small set of sizes using this approach. In this thesis, we propose an infinite RBM model, whose maximum likelihood estimation (MLE) can be solved by an efficient, greedy algorithm by inserting one hidden unit at each iteration. This can be used to easily identify an appropriate number of hidden units during learning.

**Conditional RBM** A conditional restricted Boltzmann machine (CRBM) is the discriminative extension of RBM to include observed features $\boldsymbol{x}$; CRBM is used in deep probabilistic model for supervised learning [Hinton et al., 2006a], and also provides a stand-alone solution to a wide range of problems such as classification [Larochelle and Bengio, 2008], human motion capture [Taylor et al., 2006a], collaborative filtering [Salakhutdinov et al., 2007a], and structured prediction [Mnih et al., 2011, Yang et al., 2014]. For structured prediction, a CRBM need not make any explicit assumptions about the structure of the output variables (visible units $\boldsymbol{v}$). This is especially useful in many applications where the structure of the outputs is challenging to describe (e.g., multi-label learning [Li et al., 2015]). In image denoising or object segmentation, the hidden units can encode higher-order correlations of visible units (e.g. shapes, or parts of object), which play the same role as high-order potentials but can improve the statistical efficiency.

Loopy belief propagation (BP) [Pearl, 1988] is a very popular approximate inference algorithm and usually provides a good approximation of marginals for loopy graphical model. It can be used as inference routine in learning as well as for making predictions after the CRBM

has been learned. However, it was found to be slow on CRBMs for structured prediction and only considered practical on problems with relatively few visible and hidden units [Mnih et al., 2011]. More importantly, there is a pervasive opinion that belief propagation does not work well on RBM-based models, especially for learning [Goodfellow et al., 2016, Chapter 16]. In this thesis, we present a very efficient implementation of BP, and demonstrate that training conditional RBMs with BP as the inference routine can provide significantly better results than current state-of-the-art algorithms.

## 1.4 Outline and Contributions

The general outline and contributions of this thesis are summarized as follows:

**Chapter 3** generalizes dual decomposition to a generic *power sum* inference task, which includes marginal MAP, along with pure marginalization and MAP, as special cases. Specific contributions include:

- We propose a new convex decomposition bound, which is fully decomposed over the individual cliques of graph.
- Based on the full decomposition, we develop a block coordinate descent algorithm which is guaranteed to converge monotonically, and can be parallelized efficiently.
- Our method is faster and more reliable than previous methods on various inference queries defined on real-world problems from the UAI approximate inference challenge.

This Chapter is based the work originally published in Ping et al. [2015].

**Chapter 4** proposes the marginal structured SVM (MSSVM) for structured prediction with hidden variables. MSSVM properly accounts for the uncertainty of hidden variables by incorporating the marginal MAP inference into the max-margin learning framework, and can significantly outperform the previously proposed latent structured SVM (LSSVM; Yu

and Joachims [2009a]) and other state-of-art methods, especially when that uncertainty is large. Specific contributions include:

- Our method results in a smoother objective function, making gradient-based optimization of MSSVMs converge significantly faster than for LSSVMs.
- Our method consistently outperforms hidden conditional random fields (HCRFs; Quattoni et al. [2007a]) on both simulated and real-world datasets.
- We propose a unified framework that includes both our and several other existing methods as special cases, and provides insights into the comparison of different models in practice.

This chapter is based on the work originally published in Ping et al. [2014].

**Chapter 5** investigates the important model selection problem with restricted Boltzmann machine (RBM); that is finding the appropriate size of hidden layer. To that end, we proposes an infinite RBM model, whose maximum likelihood estimation (MLE) corresponds to a constrained convex optimization. We apply the Frank-Wolfe algorithm to solve the program, which provides a solution that can be interpreted as inserting a hidden unit at each iteration, so that the optimization process takes the form of a sequence of finite models of increasing complexity. Specific contributions include:

- Our method can be used to easily and efficiently identify an appropriate number of hidden units during the optimization.
- The resulting model can also be used as an initialization for typical state-of-the-art RBM training algorithms such as contrastive divergence, leading to models with consistently higher test likelihood than random initialization.

This chapter is based on the work that is originally published in Ping et al. [2016].

**Chapter 6** presents a matrix-based implementation of belief propagation algorithms on CRBMs, which is easily scalable to tens of thousands of visible and hidden units. In addition, our algorithms uses standard matrix product and element-wise operations, and is thus highly suitable for modern high performance computing architecture (e.g., GPU). We demonstrate that, in both maximum likelihood and max-margin learning, training conditional RBMs with BP as the inference routine can provide significantly better results than current state-of-the-art contrastive divergence methods on structured prediction problems. We also include practical guidelines on training CRBMs with BP, and some insights into the interaction between learning and inference algorithms for CRBMs. This chapter is based on the work in submission [Ping and Ihler, 2017].

**Chapter 7** concludes this thesis and gives some open directions for future research. The Appendix contains proofs and additional details omitted from the main chapters.

# Chapter 2

# Background

In this chapter we review some background knowledge which is required in this thesis. We organize the chapter as follow. Section 2.1 reviews various types of probabilistic graphical models. Section 2.2 reviews different types of inference tasks and presents a unified inference framework on graphical models. We review variational inference techniques at Section 2.3 and parameter estimation methods at Section 2.4.

## 2.1 Graphical Models

In this section, we review some background material on various types of graphical models used in this thesis.

### 2.1.1 Markov Random Fields

A *Markov random field* (MRF), Markov network or undirected graphical model on discrete random variables $\boldsymbol{x} = [x_1, \ldots, x_n] \in \mathcal{X}$ is a probability distribution,

$$p(\boldsymbol{x}; \theta) = \frac{1}{Z(\theta)} \exp\Big[\sum_{\alpha \in \mathcal{F}} \theta_\alpha(x_\alpha)\Big]; \quad Z(\theta) = \sum_{\boldsymbol{x} \in \mathcal{X}^n} \exp\Big[\sum_{\alpha \in \mathcal{F}} \theta_\alpha(x_\alpha)\Big], \tag{2.1}$$

where $\mathcal{F}$ is a set of subsets of the variables, each associated with a factor $\theta_\alpha(x_\alpha)$, over a subset of the variables $\alpha \subset \{1, \ldots, n\}$, [1] and $Z(\theta)$ is the normalization constant, or partition function. We associate an undirected graph $G = (V, E)$ with $p(\boldsymbol{x})$ by mapping each $x_i$ to a node $i \in V$, and adding an edge $ij \in E$ if and only if there exists $\alpha \in \mathcal{F}$ such that $\{i, j\} \subseteq \alpha$. We say node $i$ and $j$ are neighbours if $ij \in E$. Then, $\mathcal{F}$ is a set of cliques (fully connected subgraphs) of $G$.

We can rewrite the factorized MRF in Eq. (2.1) more compactly as,

$$p(\boldsymbol{x}; \theta) = \exp\big(\theta(\boldsymbol{x}) - \Phi(\theta)\big); \quad \theta(\boldsymbol{x}) = \sum_{\alpha \in \mathcal{F}} \theta_\alpha(x_\alpha) \tag{2.2}$$

where $\theta(\boldsymbol{x})$ are called the natural parameters, and the log partition function $\Phi(\theta) = \log Z(\theta)$.

### Pairwise Model

A special class of models within the family of MRFs that have received considerable attention are pairwise models. Pairwise models only include singleton and pairwise factors on the graph $G = (V, E)$; that is, $\mathcal{F} = V \cup E$, and we can write

$$p(\boldsymbol{x}; \theta) = \exp\Big[\sum_{i \in V} \theta_i(x_i) + \sum_{(ij) \in E} \theta_{ij}(x_i, x_j) - \Phi(\theta)\Big], \tag{2.3}$$

where each singleton factor $\theta_i(x_i)$ is associated with a node $i \in V$, and each pairwise factors $\theta_{ij}(x_i, x_j)$ correspond to an edge $(i, j) \in E$. It is also commonly written as,

$$p(\boldsymbol{x}; \theta) \propto \prod_{i \in V} \psi_i(x_i) \prod_{(ij) \in E} \psi_{ij}(x_i, x_j)$$

where $\psi_i(x_i) = \exp\big[\theta_i(x_i)\big]$ and $\psi_{ij}(x_i, x_j) = \exp\big[\theta_{ij}(x_i, x_j)\big]$.

[1] $\theta_\alpha(x_\alpha)$ is also called log potential function.

#### Log-linear Form

One equivalent representation of an MRF is the log-linear form,

$$p(\boldsymbol{x};\theta) = \frac{1}{Z(\theta)} \exp\Big[\sum_{\alpha\in\mathcal{F}} \theta_\alpha^\top \ \phi(x_\alpha)\Big]$$

where we arrange the values of the log potential function $\theta_\alpha(x_\alpha)$ over each possible configuration into a vector $\theta_\alpha \in \mathbb{R}^{|x_\alpha|}$, [2] whose dimension $|x_\alpha|$ is the number of configurations for $x_\alpha$. The vector $\phi(x_\alpha) \in \mathbb{R}^{|x_\alpha|}$ corresponds to the features of $x_\alpha$, where each element of the vector corresponds to an indicator function $\mathbb{1}(x_\alpha = x_\alpha{}')$ for a configuration $x_\alpha{}'$. One can rewrite it more compactly as,

$$p(\boldsymbol{x};\theta) = \frac{1}{Z(\theta)} \exp\big[\theta^\top \phi(\boldsymbol{x})\big]; \quad Z(\theta) = \sum_{\boldsymbol{x}} \exp\big[\theta^\top \phi(\boldsymbol{x})\big] \tag{2.4}$$

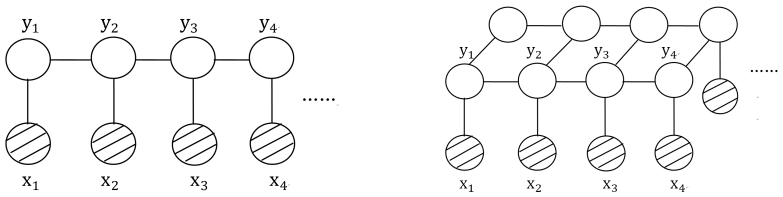
by concatenating all $\{\theta_\alpha\}$ into the model parameter $\theta$ and $\{\phi(x_\alpha)\}$ into the sufficient statistics $\phi(\boldsymbol{x})$ (also called the joint feature map), respectively. This log-linear form is convenient in parameter estimation, because the information of the data is summarized in the sufficient statistics. It also leads to concise notations when extending MRF models into conditional random fields as we discussed in the following part.

### 2.1.2 Conditional Random Fields

Conditional random fields (CRFs) [Lafferty et al., 2001], or undirected conditional models define a conditional distribution on discrete random variables $\boldsymbol{y} = [y_1, \ldots, y_m] \in \mathcal{Y}$ given the observed variables $\boldsymbol{x} = [x_1, \ldots, x_n] \in \mathcal{X}$ (which can be either discrete or continuous),

$$p(\boldsymbol{y} \mid \boldsymbol{x};\theta) = \frac{1}{Z(\boldsymbol{x};\theta)} \exp\big[\theta^\top \phi(\boldsymbol{x},\boldsymbol{y})\big] \tag{2.5}$$

[2] We slightly abuse the notation $\theta_\alpha$ here to represent both the parameter vector and the log-potential function.




(a) chain CRF (b) grid CRF

Figure 2.1: Graphical illustration of (a) chain structured CRF, and (b) grid structured CRF with $\boldsymbol{y}$ as output variables (white nodes), $\boldsymbol{x}$ as observed input features (nodes with hatching).

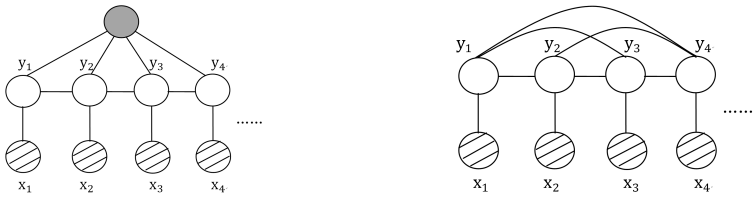
where $\phi(\boldsymbol{x}, \boldsymbol{y}) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^D$ is a set of features which describe the relationships among the $(\boldsymbol{x}, \boldsymbol{y})$, and $\theta \in \mathbb{R}^D$ are the corresponding log-linear weights, or model parameters. The function $Z(\boldsymbol{x}; \theta)$ is the $\boldsymbol{x}$-dependent normalization constant,

$$Z(\boldsymbol{x}; \theta) = \sum_{\boldsymbol{y}} \exp\left[\theta^\top \phi(\boldsymbol{x}, \boldsymbol{y})\right].$$

CRFs are standard tools for structured prediction in many domains. For example, in sequence labeling (e.g., part-of-speech tagging), the chain structured CRF is a standard practice (see Figure 2.1 (a) for a graphical illustration). Also, the grid structured CRF (see Figure 2.1 (b) for an illustration) is widely applied in semantic image segmentation. These discriminative models make them feasible to incorporate rich and overlapping features $\boldsymbol{x}$ without modeling their distribution, which can significantly improve the accuracy in structured prediction [Lafferty et al., 2001].

### 2.1.3 CRF with Hidden Variables

As we discussed in Chapter 1, latent (hidden) variables are useful in representing many structured prediction tasks; they may arise either from missing values in partially labeled datasets, or be introduced to enrich the model's flexibility. One can extend the definition in




(a) CRF with hidden variable (b) CRF after marginalization

Figure 2.2: Graphical illustration of (a) CRF with one hidden variable (shaded nodes), $\boldsymbol{y}$ as output variables (white nodes), $\boldsymbol{x}$ as observed input features (nodes with hatching). (b) After marginalization of hidden variable, the output variables become fully connected.

Eq. (2.5) to include hidden variables $\boldsymbol{h} \in \mathcal{H}$,

$$p(\boldsymbol{y}, \boldsymbol{h} \mid \boldsymbol{x}; \theta) = \frac{1}{Z(\boldsymbol{x}; \theta)} \exp\left[\theta^\top \phi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})\right], \tag{2.6}$$

where $\phi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})$ is the joint feature map which describes the relationships among the $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})$. The marginal distribution over $\boldsymbol{y}$ is,

$$p(\boldsymbol{y} \mid \boldsymbol{x}; \theta) = \frac{1}{Z(\boldsymbol{x}; \theta)} \sum_{\boldsymbol{h}} \exp\left[\theta^\top \phi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})\right]. \tag{2.7}$$

Note, after marginalizing out hidden variables, the log-linear model (2.6) becomes a non-linear model (2.7), which can capture high-order correlations among the output variables. The effect of marginalizing hidden variable is illustrated in Figure 2.2.

### 2.1.4 Restricted Boltzmann Machine

Another popular class of latent variable models are restricted Boltzmann machines (RBMs) [Smolensky, 1986, Hinton, 2002b]. In the following subsections, we review background on RBMs and conditional RBMs. In the literature, people commonly use visible units $\boldsymbol{v}$ to represent the observable variables, and hidden units $\boldsymbol{h}$ to represent the latent variables. We follow this convention across the whole thesis.

An RBM is a special undirected graphical model (see Figure 2.3(a)) that defines a joint

distribution over the vectors of visible units $\boldsymbol{v} \in \{0,1\}^{|\boldsymbol{v}|\times 1}$ and hidden units $\boldsymbol{h} \in \{0,1\}^{|\boldsymbol{h}|\times 1}$,

$$p(\boldsymbol{v}, \boldsymbol{h}|\theta) = \frac{1}{Z(\theta)} \exp\big(-E(\boldsymbol{v}, \boldsymbol{h}; \theta)\big), \tag{2.8}$$

where $|\boldsymbol{v}|$ and $|\boldsymbol{h}|$ are the number of visible units and hidden units respectively; $E(\boldsymbol{v}, \boldsymbol{h}; \theta)$ is the energy function,

$$E(\boldsymbol{v}, \boldsymbol{h}; \theta) = -\boldsymbol{v}^\top W \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{b}^v - \boldsymbol{h}^\top \boldsymbol{b}^h;$$

and $\theta = \{W, \boldsymbol{b}^v, \boldsymbol{b}^h\}$ are the model parameters, including pairwise interaction terms $W \in \mathbb{R}^{|\boldsymbol{v}|\times|\boldsymbol{h}|}$, and bias terms $\boldsymbol{b}^v \in \mathbb{R}^{|\boldsymbol{v}|\times 1}$ for visible units and $\boldsymbol{b}^h \in \mathbb{R}^{|\boldsymbol{h}|\times 1}$ for hidden units. The function $Z(\theta)$ is the normalization constant,

$$Z(\theta) = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp\big(-E(\boldsymbol{v}, \boldsymbol{h}; \theta)\big).$$

**Conditional Distribution** Because RBMs have a bipartite structure, the conditional distributions $p(\boldsymbol{v}|\boldsymbol{h}; \theta)$ and $p(\boldsymbol{h}|\boldsymbol{v}; \theta)$ are fully factorized and can be calculated in closed form,

$$\begin{aligned} p(\boldsymbol{h}|\boldsymbol{v}, \theta) &= \prod_{j=1}^{|\boldsymbol{h}|} p(h_j|\boldsymbol{v}), \quad \text{with} \quad p(h_j = 1|\boldsymbol{v}) = \sigma\big(\boldsymbol{v}^\top W_{\bullet j} + b_j^h\big), \\ p(\boldsymbol{v}|\boldsymbol{h}, \theta) &= \prod_{i=1}^{|\boldsymbol{v}|} p(v_i|\boldsymbol{h}), \quad \text{with} \quad p(v_i = 1|\boldsymbol{h}) = \sigma\big(W_{i\bullet}\boldsymbol{h} + b_i^v\big), \end{aligned} \tag{2.9}$$

where $\sigma(u) = 1/(1+\exp(-u))$ is the logistic function, and $W_{\bullet j}$ and $W_{i\bullet}$ are the $j$-th column and $i$-th row of $W$ respectively. The fully factorized conditional distribution in Eq. (2.9) allows us to derive an efficient blocked Gibbs sampler that iteratively alternates between drawing $\boldsymbol{v}$ and $\boldsymbol{h}$. This Gibbs sampler is the key component of contrastive divergence learning algorithm for RBM [Hinton, 2010].

**Marginal Distribution** Because the energy function of an RBM can be written as the summation of terms associated with each hidden units $h_j$

$$-E(\boldsymbol{v}, \boldsymbol{h}; \theta) = \sum_{j=1}^{|\boldsymbol{h}|} \Big( \boldsymbol{v}^\top W_{\bullet j} h_j + h_j b_j^h \Big) + \boldsymbol{v}^\top \boldsymbol{b}^v,$$

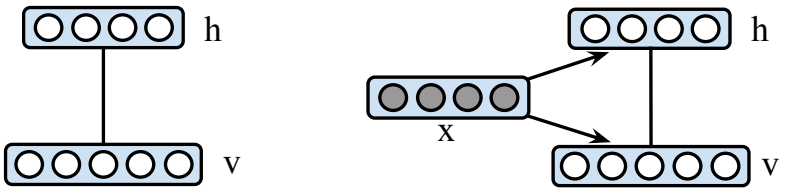
and each $h_j$ takes values in $\{0, 1\}$, the marginal distribution of visible units $\boldsymbol{v}$ has the analytical form,

$$\begin{aligned}
p(\boldsymbol{v} \mid \theta) &= \sum_{\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}; \theta) = \frac{1}{Z(\theta)} \Big\{ \sum_{\boldsymbol{h}} \exp \Big[ \sum_{j=1}^{|\boldsymbol{h}|} \Big( \boldsymbol{v}^\top W_{\bullet j} h_j + h_j b_j^h \Big) \Big] \Big\} \exp \Big[ \boldsymbol{v}^\top \boldsymbol{b}^v \Big] \\
&= \frac{1}{Z(\theta)} \Big\{ \prod_{j=1}^{|\boldsymbol{h}|} \sum_{h_j=0}^{1} \exp \Big( \boldsymbol{v}^\top W_{\bullet j} h_j + h_j b_j^h \Big) \Big\} \exp \Big[ \boldsymbol{v}^\top \boldsymbol{b}^v \Big] \\
&= \frac{1}{Z(\theta)} \Big\{ \prod_{j=1}^{|\boldsymbol{h}|} \big( 1 + \exp(\boldsymbol{v}^\top W_{\bullet j} + b_j^h) \big) \Big\} \exp \Big[ \boldsymbol{v}^\top \boldsymbol{b}^v \Big] \\
&= \frac{1}{Z(\theta)} \exp \Big[ \sum_{j=1}^{|\boldsymbol{h}|} \log \big( 1 + \exp(\boldsymbol{v}^\top W_{\bullet j} + b_j^h) \big) + \boldsymbol{v}^\top \boldsymbol{b}^v \Big], \qquad (2.10)
\end{aligned}$$

where $W_{\bullet j}$ is the $j$-th column of $W$ and corresponds to the weights connected to the $j$-th hidden unit. We take advantage of this nice form in Chapter 5.

RBMs have several desirable properties made it very popular in machine leaning community: (1) The hidden units in an RBM can encode higher-order correlations of visible units, which play the same role as high order potentials but can improve the statistical efficiency. This is because the high order potential function requires a large number of parameters to describe. [3] (2) The bipartite structure enables efficient inference algorithm as indicated by blocked Gibbs sampling through Eq. (2.9). (3) The model is compactly represented by matrix parameters and enables matrix-based implementation of learning and inference algorithms. It should be noted, matrix product and element-wise operations are highly optimized

[3] For example, a high order potential function over 10 binary units requires up to $2^{10} = 1024$ parameters.




(a) RBM (b) conditional RBM

Figure 2.3: Graphical illustration of (a) a RBM with $|\boldsymbol{v}| = 5$ visible units and $|\boldsymbol{h}| = 4$ hidden units, and (b) the extended CRBM with $\boldsymbol{v}$ as output variables, $\boldsymbol{x}$ as observed input features.

in modern high-performance computing architecture. We take advantage of properties (2) and (3) in Chapter 6. (4) The two layer structure is convenient for constructing deep probabilistic models, such as deep Boltzmann machines [Salakhutdinov and Hinton, 2009], which are constructed by a stack of RBMs.

### 2.1.5 Conditional RBM

The conditional RBM (CRBM) extends RBMs to include observed features $\boldsymbol{x}$ (see Figure 2.3(b) for an illustration),[4] and defines a joint conditional distribution over $\boldsymbol{v}$ and $\boldsymbol{h}$ given input features $\boldsymbol{x} \in \mathbb{R}^{|\boldsymbol{x}|\times 1}$,

$$p(\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{x}; \theta) = \frac{1}{Z(\boldsymbol{x}; \theta)} \exp\big(-E(\boldsymbol{v}, \boldsymbol{h}, \boldsymbol{x}; \theta)\big), \tag{2.11}$$

where the energy function $E$ is defined as,

$$E(\boldsymbol{v}, \boldsymbol{h}, \boldsymbol{x}; \theta) = -\boldsymbol{v}^\top W^{vh} \boldsymbol{h} - \boldsymbol{v}^\top W^{vx} \boldsymbol{x} - \boldsymbol{h}^\top W^{hx} \boldsymbol{x} - \boldsymbol{v}^\top \boldsymbol{b}^v - \boldsymbol{h}^\top \boldsymbol{b}^h,$$

[4] One can view an RBM as a special CRBM with $\boldsymbol{x} \equiv 0$.

and $\theta = \{W^{vh}, W^{vx}, W^{hx}, b^v, b^h\}$ are model parameters. $Z(\boldsymbol{x}; \theta)$ is the $\boldsymbol{x}$-dependent partition function,

$$Z(\boldsymbol{x}; \theta) = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp\big( - E(\boldsymbol{v}, \boldsymbol{h}, \boldsymbol{x}; \theta)\big).$$

Conditional RBM can be viewed as a particular type of CRF with hidden variables in Eq (2.6), because the visible units $\boldsymbol{v}$ in CRBMs play the same role as $\boldsymbol{y}$ in in Eq (2.6).

**Conditional Distribution** Because CRBMs still have a bipartite structure given the observed features, the conditional distributions $p(\boldsymbol{v}|\boldsymbol{h}, \boldsymbol{x})$ and $p(\boldsymbol{h}|\boldsymbol{v}, \boldsymbol{x})$ are fully factored and can be written as,

$$\begin{aligned} p(\boldsymbol{v}|\boldsymbol{h}, \boldsymbol{x}) &= \prod_{i=1}^{|\boldsymbol{v}|} p(v_i|\boldsymbol{h}, \boldsymbol{x}), \quad \text{with} \;\; p(v_i = 1|\boldsymbol{h}, \boldsymbol{x}) = \sigma\big(W^{vh}_{i\bullet}\boldsymbol{h} + W^{vx}_{i\bullet}\boldsymbol{x} + b^v_i\big), \\ p(\boldsymbol{h}|\boldsymbol{v}, \boldsymbol{x}) &= \prod_{j=1}^{|\boldsymbol{h}|} p(h_j|\boldsymbol{v}, \boldsymbol{x}), \quad \text{with} \;\; p(h_j = 1|\boldsymbol{v}, \boldsymbol{x}) = \sigma\big(\boldsymbol{v}^T W^{vh}_{\bullet j} + W^{hx}_{j\bullet}\boldsymbol{x} + b^h_j\big), \end{aligned} \tag{2.12}$$

where $\sigma(u) = 1/(1 + \exp(-u))$ is the logistic function, $W^{vh}_{i\bullet}$ and $W^{vh}_{\bullet j}$ are the $i$-th row and $j$-th column of $W^{vh}$ respectively, $W^{vx}_{i\bullet}$ is the $i$-th row of of $W^{vx}$, and $W^{hx}_{j\bullet}$ is the $j$-th row of $W^{hx}$. Eq. (2.12) allows us to derive a blocked Gibbs sampler that iteratively alternates between drawing $\boldsymbol{v}$ and $\boldsymbol{h}$.

**Marginal Distribution** Similar to standard RBMs, the marginal distribution of the visible units $\boldsymbol{v}$ given observed features $\boldsymbol{x}$ is,

$$p(\boldsymbol{v}|\boldsymbol{x}) = \sum_{\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x}; \theta)} \exp\big[ - F(\boldsymbol{v}, \boldsymbol{x}; \theta)\big], \tag{2.13}$$

where the negative energy function has analytic form,

$$-F(\boldsymbol{v}, \boldsymbol{x}; \theta) = \sum_{j=1}^{|\boldsymbol{h}|} \log \left[1 + \exp\left(\boldsymbol{v}^\top W^{vh}_{\bullet j} + W^{hx}_{j\bullet}\boldsymbol{x} + b^h_j\right)\right] + \boldsymbol{v}^\top W^{vx}\boldsymbol{x} + \boldsymbol{v}^\top \boldsymbol{b}^v.$$

Note that, after marginalizing out hidden variables, the log-linear model (2.11) becomes a non-linear model (2.13) (since $F(\boldsymbol{v}, \boldsymbol{x}; \theta)$ is non-linear), which can capture high-order correlations among visible units. This property is essentially important in many applications of CRBMs with structured output [e.g., Salakhutdinov et al., 2007a, Mnih et al., 2011].

## 2.2 Inference Tasks

The use and evaluation of a given graphical model often involves different types of inference tasks. Given the observed features in conditional models, such as generic CRFs or CRBMs, the inference tasks are analogous to their generative counterparts. Without loss of generality, we present the definitions of different inference task in the language of generic MRF (2.1).

### 2.2.1 Marginalization Inference

*Marginalization*, or *sum-inference* tasks perform a sum over the configurations to calculate the log partition function (2.1),

$$\Phi(\theta) = \log Z(\theta) = \log \sum_{\boldsymbol{x} \in \mathcal{X}^n} \exp\Big[\sum_{\alpha \in \mathcal{F}} \theta_\alpha(x_\alpha)\Big],$$

or marginal probabilities of one or a few variables,

$$p(x_i) = \sum_{\boldsymbol{x}_{\setminus i}} p(\boldsymbol{x}) = \frac{1}{Z} \sum_{\boldsymbol{x}_{\setminus i}} \exp\Big[\sum_{\alpha \in \mathcal{F}} \theta_\alpha(x_\alpha)\Big];$$

Marginal inference is useful in many settings; for example, marginal probabilities can be used to make Bayes-optimal prediction under Hamming loss (element-wise). In addition, they are required when calculating the data likelihood and its derivative. It is easy to show

$$\frac{\partial \Phi(\theta)}{\partial \theta_\alpha(x_\alpha)} = p(x_\alpha) = \sum_{\boldsymbol{x}_{\setminus\alpha}} p(\boldsymbol{x}), \tag{2.14}$$

which connects the log partition function to the marginal probability, and plays a central role in maximum likelihood estimation (MLE).

### 2.2.2 MAP Inference

On the other hand, the maximum *a posteriori* (MAP), or *max-inference* tasks performs joint maximization to find the configuration with the highest probability, that is,

$$\Phi_0(\theta) = \max_{\boldsymbol{x}} \sum_{\alpha \in \mathcal{F}} \theta_\alpha(x_\alpha). \tag{2.15}$$

where, since the maximization does not involve the partition function $Z$, it can be dropped. In the literature of Bayesian networks, the MAP inference tasks corresponds to optimizing the configuration of the unobserved variables given some observed evidence, and is often referred to as the most probable explanation (MPE).

The MAP is widely used in structured prediction applications, e.g., image denoising and semantic segmentation in computer vision [Nowozin and Lampert, 2011]. In these application, one has a conditional random field (CRF) defined in (2.5), and makes a prediction by

$$\hat{\boldsymbol{y}}(\theta) = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} \theta^T \phi(\boldsymbol{x}, \boldsymbol{y}), \tag{2.16}$$

where $\boldsymbol{x}$ are the input features and $\boldsymbol{y}$ are the output variables.

### 2.2.3 Marginal MAP

A generalization of max- and sum- inference is *marginal MAP*, or *mixed-inference*, in which we are interested in first marginalizing a subset $A$ of variables (e.g., hidden variables), and then maximizing the remaining variables $B$ (whose values are of direct interest), that is,

$$\Phi_{AB}(\theta) = \max_{\boldsymbol{x}_B} Q(\boldsymbol{x}_B) = \max_{\boldsymbol{x}_B} \log \sum_{\boldsymbol{x}_A} \exp\Big[\sum_{\alpha \in \mathcal{F}} \theta_\alpha(x_\alpha)\Big], \tag{2.17}$$

where $A \cup B = V$ (all the variables) and $A \cap B = \emptyset$. The objective function

$$Q(\boldsymbol{x}_B) = \log \sum_{\boldsymbol{x}_A} \exp\big[\theta(\boldsymbol{x}_A, \boldsymbol{x}_B)\big] = \log \sum_{\boldsymbol{x}_A} \exp\big[\theta(\boldsymbol{x})\big]$$

can be used to measure the quality of a particular decoding $\boldsymbol{x}_B$. Obviously, both sum- and max- inference are special cases of marginal MAP when $A = V$ and $B = V$, respectively. Marginal MAP inference is particularly useful in systems defined with latent variables or missing information. In fact, it provides Bayes optimal prediction of $\boldsymbol{x}_B$ measured by zero-one loss.

### 2.2.4 A Unified Inference Framework

It will be useful to introduce an more general inference task, based on a power sum operator:

$$\sum_{x_i}^{\tau_i} f(x_i) = \big[\sum_{x_i} f(x_i)^{1/\tau_i}\big]^{\tau_i},$$

where $f(x_i)$ is any non-negative function and $\tau_i$ is a *temperature* or *weight* parameter. The power sum reduces to a standard sum when $\tau_i = 1$, and approaches $\max_x f(x)$ when $\tau_i \to 0^+$, so that we define the power sum with $\tau_i = 0$ to equal the max operator.

The power sum is helpful for unifying max- and sum- inference [e.g., Weiss et al., 2007],

as well as marginal MAP [Liu, 2014]. Specifically, we can apply power sums with different weights $\tau_i$ to each variable $x_i$ along a predefined elimination order (e.g., $[x_1, \ldots, x_n]$), to define the *weighted log partition function*:

$$\Phi_{\boldsymbol{\tau}}(\theta) = \log \sum_{\boldsymbol{x}}^{\boldsymbol{\tau}} \exp\Big[\sum_{\alpha \in \mathcal{F}} \theta_\alpha(x_\alpha)\Big] = \log \sum_{x_n}^{\tau_n} \ldots \sum_{x_1}^{\tau_1} \exp\Big[\sum_{\alpha \in \mathcal{F}} \theta_\alpha(x_\alpha)\Big], \tag{2.18}$$

where we note that the value of (2.18) depends on the elimination order unless all the weights are equal. Obviously, (2.18) includes marginal MAP (2.17) as a special case by setting weights $\boldsymbol{\tau}_A = 1$ and $\boldsymbol{\tau}_B = 0$. We will demonstrate in Chapter 3 that this representation provides a useful tool for understanding and deriving new algorithms for general inference tasks, especially marginal MAP, for which relatively few efficient algorithms exist.

## 2.3 Approximate Inference and Variational Methods

The computational complexity of three common inference tasks are in order of increasing difficulty: MAP is NP-complete, sum-inference is #P-complete, and marginal MAP is $\text{NP}^{\text{PP}}$-complete [Park and Darwiche, 2004]. Thus, they are intractable on loopy graphs, and marginal MAP can be intractable even for trees [Koller and Friedman, 2009a]. As a result, approximate inference algorithms, particularly efficient variational methods, are of great interest. This section reviews the variational inference framework and several important approximate inference algorithms.

### 2.3.1 Exponential Family and Exact Variational Form

Given this exponential family representation in Eq. (2.2), we have the following important properties about the log-partition function $\Phi(\theta)$, which are central to both inference and learning [Wainwright and Jordan, 2008]:

**Proposition 2.3.1.** *(1) $\Phi(\theta)$ can be represented using the variational form,*

$$\Phi(\theta) = \log \sum_{\boldsymbol{x}} \exp(\theta(\boldsymbol{x})) = \max_{b \in \mathbb{M}(G)} \big\{ \langle \theta, b \rangle + H(\boldsymbol{x}; b) \big\} \tag{2.19}$$

*where the marginal polytope $\mathbb{M}(G)$ is the set of all possible marginal distributions $b = \{b_\alpha(x_\alpha) : \alpha \in \mathcal{F}\}$ that are consistent with a valid joint distribution $q(\boldsymbol{x})$, that is,*

$$\mathbb{M}(G) = \{b \mid b_\alpha(x_\alpha) = \sum_{\boldsymbol{x}_{\setminus \alpha}} q(\boldsymbol{x}) \quad for \quad \forall \alpha \in \mathcal{F}, \forall x_\alpha \in \mathcal{X}_\alpha\}, \tag{2.20}$$

*and $\langle \theta, b \rangle$ is the inner product,*

$$\langle \theta, b \rangle = \sum_{\alpha \in \mathcal{F}} \sum_{x_\alpha} b_\alpha(x_\alpha) \theta_\alpha(x_\alpha).$$

*$H(\boldsymbol{x}; b)$ is the entropy of $b(\boldsymbol{x})$,*

$$H(\boldsymbol{x}; b) = -\mathbb{E}_{b(\boldsymbol{x})} \log b(\boldsymbol{x}) = -\sum_{\boldsymbol{x}} b(\boldsymbol{x}) \log b(\boldsymbol{x}),$$

*We slightly abuse the notation $b(\boldsymbol{x})$ to represent an unique joint distribution that achieves the maximum entropy among all the joint distributions whose marginals are equal to $b = \{b_\alpha(x_\alpha) : \alpha \in \mathcal{F}\}$.* [5]

*In addition, the maximum of Eq. (2.19) is attained when $b$ is equal to the marginals of $p(\boldsymbol{x}) = \exp(\theta(\boldsymbol{x}) - \Phi(\theta))$, that is, $b_\alpha(x_\alpha) = p(x_\alpha)$ for $\forall \alpha \in \mathcal{F}, \forall x_\alpha \in \mathcal{X}_\alpha$.*

*(2) $\Phi(\theta)$ is a convex function with $\theta$.*

*Proof.* (1) See [Wainwright and Jordan, 2008, Chapter 3].

[5] There are many joint distributions $\{q(\boldsymbol{x})\}$ whose marginals equal to $\{b_\alpha\}$. According to the maximum entropy principle [Jaynes, 1957], there is an unique joint distribution $q^*(\boldsymbol{x})$ which achieves maximum entropy. In addition, $q^*(\boldsymbol{x})$ is in the exponential family.

(2) From Eq. (2.19), $\Phi(\theta)$ is the supremum of a set of linear functions with $\theta$, thus it is convex. □

The form (2.19) plays the central role for the development of variational inference methods. It transform the marginalization inference task, i.e., calculating the log-partition function $\Phi(\theta)$ into a continuous optimization problem. However, it should be noted that the form (2.19) does not decrease the computational complexity of sum inference, because (1) the entropy $H(\boldsymbol{x}; b)$ is generally intractable to calculate exactly from the marginals $b$, and (2) the marginal polytope $\mathbb{M}$ may require an exponential number of linear constraints to characterized exactly. Although exact calculation remains intractable, the variational from (2.19) provides a powerful framework for deriving a spectrum of variational inference algorithms by approximating both the marginal polytope $\mathbb{M}(G)$ and the entropy $H(\boldsymbol{x}; b)$ using various techniques. In the subsequent sections, we will introduce the local consistency polytope for approximating the marginal polytope $\mathbb{M}$, Bethe and various convex approximation for the the entropy $H(\boldsymbol{x}; b)$.

Before that, we first generalize the variational form of marginalization inference to other different inference task. For any scalar $\varepsilon > 0$ (including $\varepsilon \to 0^+$), we have

$$\begin{aligned}\Phi_\varepsilon(\theta) &= \varepsilon \log \sum_{\boldsymbol{x}} \exp(\frac{\theta(\boldsymbol{x})}{\varepsilon}) = \varepsilon \max_{b \in \mathbb{M}} \big\{ \langle \frac{\theta}{\varepsilon}, b \rangle + H(\boldsymbol{x}; b) \big\} \quad \text{(according to (2.19))} \\ &= \max_{b \in \mathbb{M}} \big\{ \langle \theta, b \rangle + \varepsilon H(\boldsymbol{x}; b) \big\}.\end{aligned}$$

When $\varepsilon \to 0^+$, we have the variational form of MAP inference

$$\Phi_0(\theta) = \max_{b \in \mathbb{M}} \big\{ \langle \theta, b \rangle \big\}. \tag{2.21}$$

As demonstrated in [Liu, 2014, Liu and Ihler, 2011], one can further generalize the variational

form of the above scalar-weighted log partition function to the vector-weighted log partition function in Eq. (2.18).

**Proposition 2.3.2.** *The weighted log partition function can be represented using the variational form,*

$$\Phi_{\boldsymbol{\tau}}(\theta) = \log \sum_{x_n}^{\tau_n} \cdots \sum_{x_1}^{\tau_1} \exp(\theta(x)) = \max_{b \in \mathbb{M}(G)} \big\{ \langle \theta, b \rangle + \sum_i \tau_i H(x_i | x_{i+1:n}; b) \big\}, \tag{2.22}$$

*where* $H(x_i|x_{i+1:n}; b)$ *is the conditional entropy on* $b(x)$*, and is defined as* $H(x_i|x_{i+1:n}; b) = -\sum_x b(x) \log(b(x_i|x_{i+1:n}))$*. The maximum is attained when*

$$b(x) = \prod_{i=1}^{n} b(x_i|x_{i+1:n}); \quad b(x_i|x_{i+1:n}) = (Z_{i-1}(x_{i:n})/Z_i(x_{i+1:n}))^{1/\tau_i},$$

*where* $Z_i$ *is the partial powered-sum up to* $x_{1:i}$*,*

$$Z_i(x_{i+1:n}) = \sum_{x_i}^{\tau_i} \cdots \sum_{x_1}^{\tau_1} \exp(\theta(x)).$$

*Proof.* See Theorem 4.1 of [Liu, 2014]. □

A notable special case of (2.22) is the variational form of marginal MAP [Liu and Ihler, 2013],

$$\Phi_{AB}(\theta) = \max_{\boldsymbol{x}_B} \log \sum_{\boldsymbol{x}_A} \exp\big(\theta(\boldsymbol{x})\big) = \max_{b \in \mathbb{M}(G)} \big\{ \langle \theta, b \rangle + H(\boldsymbol{x}_A|\boldsymbol{x}_B; b) \big\}, \tag{2.23}$$

where $H(\boldsymbol{x}_A|\boldsymbol{x}_B; b)$ is a conditional entropy,

$$H(\boldsymbol{x}_A|\boldsymbol{x}_B; b) = -\sum_{\boldsymbol{x}} b(\boldsymbol{x}) \log b(\boldsymbol{x}_A|\boldsymbol{x}_B) = H(\boldsymbol{x}; b) - H(\boldsymbol{x}_B; b)$$

It can be directly obtained from (2.22) by setting weights $\boldsymbol{\tau}_A = 1$ and $\boldsymbol{\tau}_B = 0$.

### 2.3.2 Loopy Belief Propagation and Bethe Approximation

An complete approximating treatment of the exact variational form Eq. (2.19) includes three components: (1) an approximation to the marginal polytope; (2) an approximation to the joint entropy $H(\boldsymbol{x}; b)$; and (3) an efficient message-passing algorithm solving the continuous optimization with approximations (1)–(2). In this subsection, we review the local consistency polytope, Bethe entropy approximation and loopy belief propagation (BP) algorithm, which fulfill the goals of (1), (2) and (3), respectively.

#### Local Consistency Polytope

As we mentioned before, it is intractable to specify the marginal polytope $\mathbb{M}(G)$ on general graphs, that is, it requires an exponential number of linear constraints to characterize it exactly. The local consistency polytope provides a convenient approximation for $\mathbb{M}(G)$, and it is a shared component of most belief-propagation-style algorithms. For notational simplicity, we restrict our discussion to pairwise graphical models defined on graph $G = (V, E)$ as in Eq. (2.3),

$$p(\boldsymbol{x}; \theta) = \exp\Big[\sum_{i \in V} \theta_i(x_i) + \sum_{(ij) \in E} \theta_{ij}(x_i, x_j) - \Phi(\theta)\Big].$$

The marginal polytope of pairwise model $G$ is the set of all possible marginal distributions (including only singleton marginals and pairwise marginals),

$$b = \big\{b_i(x_i), b_{ij}(x_i, x_j) \ : \ \forall i \in V, \ \forall (i,j) \in E\big\},$$

that are consistent with a valid joint distribution $q(\boldsymbol{x})$ (i.e., $q(\boldsymbol{x}) \geq 0$ and $\sum_{\boldsymbol{x}} q(\boldsymbol{x}) = 1$), that is,

$$\mathbb{M}(G) = \Big\{ b \;\mid\; b_i(x_i) = \sum_{\boldsymbol{x}_{\setminus i}} q(\boldsymbol{x}),\; b_{ij}(x_i, x_j) = \sum_{\boldsymbol{x}_{\setminus ij}} q(\boldsymbol{x}), \quad \text{for } \forall i \in V,\; \forall (i,j) \in E \Big\}.$$

Note that for any $b \in \mathbb{M}$, the singleton marginals and pairwise marginals should be consistent with each other, that is, the marginalization constraints $b_i(x_i) = \sum_{x_j} b_{ij}(x_i, x_j)$ are naturally hold, because they are both the marginals of a valid joint distribution $q(\boldsymbol{x})$. This motivates the definition of the local consistency polytope,

$$\mathbb{L}(G) = \Big\{ b \;\mid\; b_i(x_i) = \sum_{x_j} b_{ij}(x_i, x_j),\; \sum_{x_i} b_i(x_i) = 1,\; b_{ij}(x_i, x_j) \geq 0, \quad \text{for } \forall i \in V,\; \forall (i,j) \in E \Big\}. \tag{2.24}$$

For any $b \in \mathbb{M}$, we have that $b \in \mathbb{L}$, and thus $\mathbb{M}(G) \subseteq \mathbb{L}(G)$.

In addition, $\mathbb{M} = \mathbb{L}$ when G is a tree [Wainwright and Jordan, 2008]. In fact, on tree structured model, an valid joint distribution $q(\boldsymbol{x})$ can be explicitly constructed from the local consistent marginals $\{b_i, b_{ij}\}$ by the junction tree theorem [Wainwright and Jordan, 2008],

$$q(\boldsymbol{x}) = \prod_{i \in V} b_i(x_i) \prod_{(ij) \in E} \frac{b_{ij}(x_i, x_j)}{b_i(x_i) b_j(x_j)},$$

One can easily verify that $b_i(x_i) = \sum_{\boldsymbol{x}_{\setminus i}} q(\boldsymbol{x})$ by treating $i \in V$ as the root of tree, and always eliminating leaf/child nodes at first when performing the summation: $\sum_{\boldsymbol{x}_{\setminus i}} q(\boldsymbol{x})$. See Proposition 4.1 in Wainwright and Jordan [2008] for more details.

### Bethe Entropy Approximation

The exact joint entropy $H(\boldsymbol{x}; b)$ is intractable on general graph $G$. However, when $G$ is a tree, the joint entropy can be represented exactly using a linear combination of local entropies defined on singleton marginals and pairwise marginals. This representation motivates the Bethe entropy approximation on general graph.

**Proposition 2.3.3.** *On a tree structured model $G$, the joint entropy $H(\boldsymbol{x}; b)$ can be calculated exactly using the local entropies $H(x_i; b_i)$ and $H(x_i, x_j; b_{ij})$ defined on the local marginals $b_i(x_i)$ and $b_{ij}(x_i, x_j)$, respectively, that is,*

$$H(\boldsymbol{x}; b) = \sum_{i \in V} (1 - |\partial(i)|) H(x_i \ ; \ b_i) + \sum_{(i,j) \in E} H(x_i, x_j \ ; \ b_{ij}),$$

*where $\partial(i) = \{j \mid (i,j) \in E\}$, so that $|\partial(i)|$ is the degree or number of neighbors of node $i$, and*

$$H(x_i \ ; \ b_i) = -\sum_{x_i} b_i(x_i) \log b_i(x_i), \qquad H(x_i, x_j \ ; \ b_{ij}) = -\sum_{x_i} \sum_{x_j} b_{ij}(x_i, x_j) \log b_{ij}(x_i, x_j).$$

*It can be equivalently written as*

$$H(\boldsymbol{x}; b) = \sum_{i \in V} H(x_i; b_i) - \sum_{(i,j) \in E} I_{ij}(b_{ij}), \tag{2.25}$$

*because the mutual information $I_{ij}(b_{ij})$ is*

$$I_{ij}(b_{ij}) = \sum_{x_i, x_j} b_{ij}(x_i, x_j) \log \frac{b_{ij}(x_i, x_j)}{b_i(x_i) b_j(x_j)} = H(x_i; b_i) + H(x_j; b_j) - H(x_i, x_j; b_{ij})$$

*Proof.* According to the junction tree theorem [Wainwright and Jordan, 2008], when $G$ is a

tree, the joint distribution $b(\boldsymbol{x})$ can be explicitly written as [6]

$$b(\boldsymbol{x}) = \prod_{i \in V} b_i(x_i) \prod_{(ij) \in E} \frac{b_{ij}(x_i, x_j)}{b_i(x_i) b_j(x_j)}.$$

Thus, joint entropy $H(\boldsymbol{x}; b)$

$$\begin{aligned} H(\boldsymbol{x}; b) &= -\mathbb{E}_{b(\boldsymbol{x})}\big[\log b(\boldsymbol{x})\big] = -\mathbb{E}_{b(\boldsymbol{x})}\Big[\sum_{i \in V} \log b_i(x_i) + \sum_{(ij) \in E} \log \frac{b_{ij}(x_i, x_j)}{b_i(x_i) b_j(x_j)}\Big] \\ &= \sum_{i \in V} H(x_i; b_i) - \sum_{(i,j) \in E} I_{ij}(b_{ij}), \end{aligned}$$

This completes the proof. □

The entropy decomposition in Eq. (2.25) does not hold in loopy graphs, but it provides a well-defined approximation on general graphs,

$$H(\boldsymbol{x}; b) \approx H^{\text{bethe}}(\boldsymbol{x}; b) \stackrel{\text{def}}{=\!=} \sum_{i \in V} H(x_i; b_i) - \sum_{(i,j) \in E} I_{ij}(b_{ij}), \tag{2.26}$$

which is referred to as the *Bethe entropy approximation*. It should be noted that, although the exact joint entropy $H(\boldsymbol{x}; b)$ is concave w.r.t. $b$, the Bethe entropy $H^{\text{bethe}}$ is not guaranteed to be concave, except on tree structured models. We will discuss convex entropy approximations in Section 2.3.3.

[6] We slightly abuse the notation $b$ to represent both the joint distribution and marginals.

**Algorithm 2.1** Loopy sum-product BP on pairwise models

**Input**: Pairwise model $p(\boldsymbol{x}) \propto \prod_{i \in V} \exp\left[\theta_i(x_i)\right] \prod_{(ij) \in E} \exp\left[\theta_{ij}(x_i, x_j)\right]$ on undirected graph $G = (V, E)$. Let $\partial(i)$ be the neighborhood of $i \in V$.

**Output**: Approximate marginals $\{b_i(x_i), b_{ij}(x_i, x_j) \colon i \in V, (ij) \in E\}$.

1. Pass messages between the nodes until convergence:

$$m_{i \to j}(x_j) \propto \sum_{x_i} \exp\left[\theta_{ij}(x_i, x_j)\right] \exp\left[\theta_i(x_i)\right] \prod_{i' \in \partial(i) \setminus \{j\}} m_{i' \to i}(x_i). \tag{2.27}$$

2. Calculate the approximate marginals,

$$b_i(x_i) \propto \exp\left[\theta_i(x_i)\right] \prod_{i' \in \partial(i)} m_{i' \to i}(x_i),$$

$$b_{ij}(x_i, x_j) \propto \exp\left[\theta_{ij}(x_i, x_j) + \theta_i(x_i) + \theta_j(x_j)\right] \prod_{i' \in \partial(i) \setminus \{j\}} m_{i' \to i}(x_i) \prod_{j' \in \partial(j) \setminus \{i\}} m_{j' \to j}(x_j),$$

and the approximated log-partition function,

$$\Phi(\theta) \approx \langle \theta, b \rangle + \sum_{i \in V} H_i(b_i) - \sum_{(ij) \in E} I_{ij}(b_{ij}).$$

### Loopy Belief Propagation

Applying the local consistency polytope (2.24) and the Bethe entropy approximation (2.26), we obtain the following Bethe variational optimization,

$$\Phi^{\text{bethe}}(\theta) \stackrel{\text{def}}{=\!=} \max_{b \in \mathbb{L}(G)} \left\{ \langle \theta, b \rangle + \sum_{i \in V} H(x_i; b_i) - \sum_{(i,j) \in E} I_{ij}(b_{ij}). \right\} \tag{2.28}$$

This continuous optimization is non-convex in general because the Bethe entropy can be non-concave. An elegant and well known result by Yedidia et al. [2000] is that the stationary points (including local optima) of Eq. (2.28) are fixed points of the popular loopy sum-product BP algorithm [Pearl, 1988], which is summarized in Algorithm 2.1. In fact, the fixed-point updates in Eq. (2.27) can be derived using a Lagrangian method, in which case the messages correspond to the Lagrange multipliers that enforce the local consistency condition.

For more details of the derivation, we refer the reader to Theorem 4.2 in Wainwright and Jordan [2008].

### 2.3.3 Tree-reweighted Variational Method

In the following sections, we review convex variational methods, especially the tree-reweighted (TRW) variational bound [Wainwright et al., 2005].

#### Convex Variational Methods

The variational optimization Eq. (2.28) is generally non-convex, because the Bethe entropy approximation is non-concave with $b$. A large spectrum of concave entropy approximations have been proposed by using more general counting numbers, or coefficients in the linear combination of the local entropies. In general, they can be constructed by taking the non-negatively weighted sum of joint/conditional entropies, ensuring that they are provably concave because both the joint and conditional entropy are concave functions.

#### TRW Variational Bound

Wainwright et al. [2005] propose an important class of provably concave entropy approximations, referred to as TRW, which is also guaranteed to give an upper bound of the exact joint entropy. The basic idea is reparameterize the loopy graph as an combination of spanning trees. Let $\mathcal{T}$ is the set of spanning trees of G, and $\boldsymbol{w}^{\mathcal{T}} = \{w^T > 0 : T \in \mathcal{T}\}$ is a set of positive weights associated with $\mathcal{T}$, such that $\sum_{T\in\mathcal{T}} w^T = 1$, that is, $\boldsymbol{w}^{\mathcal{T}}$ defines a distribution over the set of spanning trees. One can define $\rho_{ij}$ to be the edge appearance probabilities under this distribution,

$$\rho_{ij} = \sum_{T:(ij)\in T} w^T, \tag{2.29}$$

which is the sum of weights of spanning trees that include $(ij)$ as an edge.

The TRW entropy approximation $H^{\text{trw}}(\boldsymbol{x}; b)$ is defined as,

$$H^{\text{trw}}(\boldsymbol{x}; b) \stackrel{\text{def}}{=\!=} \sum_{T \in \mathcal{T}} w^T H^{\text{bethe}}(\boldsymbol{x}; b, T); \tag{2.30}$$

where $H^{\text{bethe}}(\boldsymbol{x}; b, T)$ is the Bethe entropy defined on the spanning tree T, that is,

$$H^{\text{bethe}}(\boldsymbol{x}; b, T) = \sum_{i \in V} H(x_i; b_i) - \sum_{(i,j) \in T} I_{ij}(b_{ij}). \tag{2.31}$$

We have the following properties for $H^{\text{trw}}(\boldsymbol{x}; b)$ : (1) According to Proposition 2.3.3, the Bethe entropy $H^{\text{bethe}}(\boldsymbol{x}; b, T)$ is exact on the tree model defined by corresponding marginals $\{b_i(x_i), b_{ij}(x_i, x_j) \mid i \in V,\ (i,j) \in T)\}$, thus it is concave w.r.t. $b$. As a result, $H^{\text{trw}}(\boldsymbol{x}; b)$ in Eq. (2.30) is also concave ; (2) If $b \in \mathbb{M}(G)$, then $H^{\text{bethe}}(\boldsymbol{x}; b, T) \geq H(\boldsymbol{x}; b)$ [Wainwright and Jordan, 2008]. Therefore,

$$H^{\text{trw}}(\boldsymbol{x}; b) = \sum_{T \in \mathcal{T}} w^T H^{\text{bethe}}(\boldsymbol{x}; b, T) \geq \sum_{T \in \mathcal{T}} w^T H(\boldsymbol{x}; b) = H(\boldsymbol{x}; b).$$

As a result, $H^{\text{trw}}(\boldsymbol{x}; b)$ is both a concave approximation and an upper bound of the exact joint entropy.

One can substitute Eq. (2.31) into Eq. (2.30), and use $\{\rho_{ij}\}$ from (2.29) to rewrite $H^{\text{trw}}(\boldsymbol{x}, b)$ as,

$$H^{\text{trw}}(\boldsymbol{x}, b) = \sum_{i \in V} H(x_i; b_i) - \sum_{(i,j) \in E} \rho_{ij} I_{ij}(b_{ij}). \tag{2.32}$$

Combined with the local consistency polytope, the TRW variational optimization is

$$\Phi^{\text{trw}}(\{\rho_{ij}\};\theta) \stackrel{\text{def}}{=\!=} \max_{b\in\mathbb{L}(G)} \Big\{\langle\theta, b\rangle + \sum_{i\in V} H(x_i; b_i) - \sum_{(i,j)\in E} \rho_{ij} I_{ij}(b_{ij})\Big\}, \tag{2.33}$$

which is analogous the the Bethe variational optimization Eq. (2.28). After maximizing the right-hand side, it will provide an upper bound on the log-partition function, that is, $\Phi^{\text{trw}}(\{\rho_{ij}\};\theta) \geq \Phi(\theta)$, because it maximizes an upper bound approximation on a lager set $\mathbb{L}(G) \supseteq \mathbb{M}(G)$, in contrast to the exact variational optimization Eq. (2.19). This convex optimization (2.33) can be efficiently solved by the tree-reweighted belief propagation (TRBP) algorithm [Wainwright et al., 2005], which iteratively passes the messages and is analogous to loopy BP in Algorithm 2.1, but uses power sum in the message updates.

One can also optimize the edge appearance probabilities $\{\rho_{ij}\}$ to minimize $\Phi^{\text{trw}}(\{\rho_{ij}\};\theta)$ and obtain the tightest upper bound. Wainwright et al. [2005] presents a double-loop conditional gradient descent algorithm for this minimization, in which the inner-loop optimizes $b$ with fixed $\{\rho_{ij}\}$ via TRBP, and the outer-loop takes conditional gradient descent updates on $\{\rho_{ij}\}$. Unfortunately, this double-loop procedure is very slow due to the expensive inner-loops; because it is a $\min_{\{\rho_{ij}\}} \max_b$ saddle point problem, and the gradient of weights $\{\rho_{ij}\}$ is only valid when then inner maximization is completely solved. Thus, most TRW implementations [e.g., Mooij, 2010] simply adopt fixed $\{\rho_{ij}\}$. In Chapter 3, we present a fully decomposed upper bound on which we can simultaneously perform message-passing iterations and update the weights.

### TRW Primal Bound

In the above, we interpret the tree-reweighted method as a convex entropy approximation in variational (dual) form. Another equivalent derivation is directly bounding the log-partition function $\Phi(\theta)$ via Jensen's Inequality in primal form.

We denote a set of *reparameterization* parameters $\{\theta^T : T \in \mathcal{T}\}$ on the set of spanning trees $\mathcal{T}$, such that (1) $\theta(\boldsymbol{x}) = \sum_{T\in\mathcal{T}} \theta^T(\boldsymbol{x})$ for $\forall \boldsymbol{x}$, and (2) the Markov random field of $p(\boldsymbol{x}|\theta^T) = \exp\big(\theta^T(\boldsymbol{x}) - \Phi(\theta^T)\big)$ is a tree structured model defined on the spanning tree $T$.

Because $\Phi(\theta)$ is a convex function with $\theta$ (see Proposition 2.3.1) and $\sum_{T\in\mathcal{T}} w^T = 1$, we have

$$\Phi(\theta) = \Phi\Big(\sum_{T\in\mathcal{T}} w^T \frac{\theta^T}{w^T}\Big) \leq \sum_{T\in\mathcal{T}} w^T \Phi(\frac{\theta^T}{w^T}) \stackrel{\text{def}}{=\!=} L^{\text{trw}}(\{\theta^T\}, \{w^T\}), \tag{2.34}$$

by Jensen's inequality. It is easy to show that the upper bound $L^{\text{trw}}(\{\theta^T\}, \{w^T\})$ is jointly convex with $\{\theta^T\}$ and $\{w^T\}$, because one can rewrite it into

$$\begin{aligned}\sum_{T\in\mathcal{T}} w^T \Phi(\frac{\theta^T}{w^T}) &= \sum_{T\in\mathcal{T}} w^T \max_{b^T\in\mathbb{M}(T)} \Big\{\langle \frac{\theta^T}{w^T}, b^T\rangle + H(\boldsymbol{x}, b^T)\Big\} \quad \text{(by applying Eq. (2.19))} \\ &= \sum_{T\in\mathcal{T}} \max_{b^T\in\mathbb{M}(T)} \Big\{\langle \theta^T, b^T\rangle + w^T H(\boldsymbol{x}, b^T)\Big\},\end{aligned}$$

and notice that it is a supremum of a set of linear function over $\{\theta^T\}$ and $\{w^T\}$.

The tightness of the upper bound $L^{\text{trw}}$ depend on the choice of $\{\theta^T\}$ and $\{w^T\}$, and one can optimize them to get the tightest bound. Wainwright et al. [2005] showed the following important proposition:

**Proposition 2.3.4.** *The TRW variational optimization in Eq.* (2.33) *is the dual problem of minimizing primal TRW bound in Eq.* (2.34) *with* $\{\theta^T\}$ *given fixed weights* $\{w^T\}$*, that is,*

$$\min_{\{\theta^T\}} \Big\{\sum_{T\in\mathcal{T}} w^T \Phi(\frac{\theta^T}{w^T})\Big\} \;=\; \max_{b\in\mathbb{L}} \Big\{\langle \theta, b\rangle + \sum_{i\in V} H(x_i; b_i) - \sum_{(i,j)\in E} \rho_{ij} I_{ij}(b_{ij})\Big\}. \tag{2.35}$$

*where* $\sum_{T\in\mathcal{T}} \theta^T = \theta$, $\sum_{T\in\mathcal{T}} w^T = 1$ *on the left-hand side, and* $\{\rho_{ij}\}$ *on the right hand side are induced from* $\{w^T\}$ *by definition Eq.*(2.29).

*As a result, the tightest upper bound obtainable by TRW is,*

$$\min_{\{w^T\}} \min_{\{\theta^T\}} L^{trw}(\{\theta^T\}, \{w^T\}) \;=\; \min_{\{\rho_{ij}\}} \max_{b \in \mathbb{L}} \Big\{ \langle \theta, b \rangle + \sum_{i \in V} H(x_i; b_i) - \sum_{(i,j) \in E} \rho_{ij} I_{ij}(b_{ij}) \Big\}. \quad (2.36)$$

The two forms of TRW bound have their own pros and cons:

- The primal bound $L^{\text{trw}}(\{\theta^T\}, \{w^T\})$ on the left-hand side (LHS) provides an "any-time" bound for log-partition function with any values of $\{\theta^T\}, \{w^T\}$, while the variational form at right-hand side (RHS) is only guaranteed to be an upper bound when $b$ is fully optimized.
- The joint minimization on the LHS enables simultaneous optimization over the reparameterization $\{\theta^T\}$ and weights $\{w^T\}$, while the saddle point problem on the RHS needs a double loop algorithm with an expensive inner-loop maximization.
- On the other hand, it is computationally intractable to fully optimize the primal bound $L^{\text{trw}}(\{\theta^T\}, \{w^T\})$ directly, because the number of spanning trees $\mathcal{T}$ on the graph G can be extremely large. For efficiency reasons, some methods [e.g., Jancsary and Matz, 2011] heuristically select a small subset of trees, but if too few trees are included, the bound will be loose.

In Chapter 3, we introduce a full decomposition bound, which has (1) much more compact representation compared to the primal TRW, and (2) an any-time property in the sense that it provide a bound at any point of the optimization, also (3) joint minimization over all parameters compared to variational form of TRW.

### 2.3.4 Dual Decomposition for MAP

This subsection reviews the dual decomposition method for MAP inference [e.g., Komodakis et al., 2007, Sontag et al., 2011]. In MAP, linear programming methods and their related dual decomposition methods have become a dominant approach in the last decade, with

numerous optimization methods such as coordinate descent [Werner, 2007, Globerson and Jaakkola, 2008], subgradient descent [Komodakis et al., 2011], and alternating direction methods [Meshi and Globerson, 2011, Forouzan and Ihler, 2013].

One can directly obtain the following upper bound of MAP inference (2.15), [7]

$$\Phi_0(\theta) = \max_{\boldsymbol{x}} \Big[ \sum_{i \in V} \theta_i(x_i) + \sum_{\alpha \in \mathcal{F}} \theta_\alpha(x_\alpha) \Big] \leq \sum_{i \in V} \max_{x_i} \theta_i(x_i) + \sum_{\alpha \in \mathcal{F}} \max_{x_\alpha} \theta_\alpha(x_\alpha), \tag{2.37}$$

which is a application of the inequality: $\max_x \big[ \sum_k f_k(x) \big] \leq \sum_k \max_x f_k(x)$.

To increase the flexibility of the upper bound, one can introduce a set of *cost-shifting* or *reparameterization* variables $\delta = \{\delta_i^\alpha(x_i) \mid \forall (i,\alpha), i \in \alpha\}$ on each variable-factor pair $(i,\alpha)$ [e.g., Ihler et al., 2012], and rewrite $\Phi_0(\theta)$ as,

$$\Phi_0(\theta) = \max_{\boldsymbol{x}} \Big[ \sum_{i \in V} \big( \theta_i(x_i) + \sum_{\alpha \in N_i} \delta_i^\alpha(x_i) \big) + \sum_{\alpha \in \mathcal{F}} \big( \theta_\alpha(x_\alpha) - \sum_{i \in \alpha} \delta_i^\alpha(x_i) \big) \Big] \tag{2.38}$$

$$\text{by noticing that:} \quad \sum_{i \in V} \sum_{\alpha \in N_i} \delta_i^\alpha(x_i) = \sum_{\alpha \in \mathcal{F}} \sum_{i \in \alpha} \delta_i^\alpha(x_i)$$

where $N_i = \{\alpha \mid \alpha \ni i\}$ is the set of cliques incident to $i$. Then, one can simply apply the inequality (2.37) on Eq.(2.38) and obtain the fully decomposed upper bound,

$$L(\delta) \overset{\text{def}}{=\!=} \sum_{i \in V} \max_{x_i} \Big[ \theta_i(x_i) + \sum_{\alpha \in N_i} \delta_i^\alpha(x_i) \Big] + \sum_{\alpha \in \mathcal{F}} \max_{x_\alpha} \Big[ \theta_\alpha(x_\alpha) - \sum_{i \in \alpha} \delta_i^\alpha(x_i) \Big] \geq \Phi_0(\theta). \tag{2.39}$$

Note, $L(\delta)$ gives a class of easy-to-evaluate upper bounds, because it decomposes the joint max on $\boldsymbol{x}$ into a sum of independent max over nodes $x_i$ and smaller cliques $x_\alpha$, which significantly reduces computational complexity. In addition, one can optimize the cost-shifting variables $\delta$ to get a much tighter upper bound $\min_\delta L(\delta)$, where the objective is convex w.r.t. $\delta$ because it is the supremum over sets of linear functions.

[7] Following Sontag et al. [2011], we add singleton factors $\{\theta_i(x_i)\}$ on each node $i \in V$ for the purpose of illustration. Obviously, these node factors can be absorbed into clique factors $\{\theta_\alpha(x_\alpha)\}$.

In the above, we directly derive the dual-decomposition bound (2.39) from an "primal" bound (2.37) of MAP value $\Phi_0(\theta)$. Another equivalent derivation is starting from the linear programming (LP) relaxation of MAP inference [e.g., Globerson and Jaakkola, 2008],

$$\Phi_0(\theta) = \max_{b \in \mathbb{M}(G)} \langle \theta, b \rangle \leq \max_{b \in \mathbb{L}(G)} \langle \theta, b \rangle,$$

where $\mathbb{M}(G) = \{b \mid \exists \text{ valid joint distribution } q(\boldsymbol{x}),\ b_\alpha(x_\alpha) = \sum_{\boldsymbol{x}_{\setminus\alpha}} q(\boldsymbol{x})\}$ is the marginal polytope, and $\mathbb{L}(G) = \{b \mid b_i(x_i) = \sum_{x_{\alpha\setminus i}} b_\alpha(x_\alpha),\ \sum_{x_i} b_i(x_i) = 1\}$ is the local consistency polytope. The first equality is from Eq. (2.21), and the second inequality naturally holds because of $\mathbb{M}(G) \subseteq \mathbb{L}(G)$. Note,

$$\max_{b \in \mathbb{L}(G)} \langle \theta, b \rangle = \max_{b \in \mathbb{L}(G)} \Big\{ \sum_{i \in V} \sum_{x_i} \theta_i(x_i) b_i(x_i) + \sum_{\alpha \in \mathcal{F}} \theta_\alpha(x_\alpha) b_\alpha(x_\alpha) \Big\}. \tag{2.40}$$

We derive the Lagrangian dual problem of this LP relaxation. [8] We first introduce Lagrange multipliers $\delta = \{\delta_i^\alpha(x_i) \mid \forall (i,\alpha), i \in \alpha\}$ associated with each marginalization constraint of $\mathbb{L}(G)$, and frame the Lagrangian function of Eq.(2.40) as

$$\begin{aligned}
\mathcal{LG}(b,\delta) &= \sum_{i \in V} \sum_{x_i} \theta_i(x_i) b_i(x_i) + \sum_{\alpha \in \mathcal{F}} \sum_{x_\alpha} \theta_\alpha(x_\alpha) b_\alpha(x_\alpha) + \sum_{\alpha \in \mathcal{F}} \sum_{i \in \alpha} \sum_{x_i} \delta_i^\alpha(x_i) \Big( b_i(x_i) - \sum_{x_{\alpha\setminus i}} b_\alpha(x_\alpha) \Big) \\
&= \sum_{i \in V} \sum_{x_i} \Big[ \Big( \theta_i(x_i) + \sum_{\alpha \in N_i} \delta_i^\alpha(x_i) \Big) \cdot b_i(x_i) \Big] + \sum_{\alpha \in \mathcal{F}} \sum_{x_\alpha} \Big[ \Big( \theta_\alpha(x_\alpha) - \sum_{i \in \alpha} \delta_i^\alpha(x_i) \Big) \cdot b_\alpha(x_\alpha) \Big].
\end{aligned}$$

It should be noted that, since we do not introduce Lagrange multipliers for the normalization constraint, $b$ still needs to satisfy $\{b \mid \sum_{x_i} b_i(x_i) = 1,\ \sum_{x_\alpha} b_\alpha(x_\alpha) = 1\}$. As a result, it is straightforward to see that

$$\max_b \mathcal{LG}(b,\delta) = \sum_{i \in V} \max_{x_i} \Big[ \theta_i(x_i) + \sum_{\alpha \in N_i} \delta_i^\alpha(x_i) \Big] + \sum_{\alpha \in \mathcal{F}} \max_{x_\alpha} \Big[ \theta_\alpha(x_\alpha) - \sum_{i \in \alpha} \delta_i^\alpha(x_i) \Big] = L(\delta),$$

[8] The term "dual" in dual-decomposition indeed refers to the dual of LP relaxation.

because the maximum can be attained by setting

$$b_i^*(x_i) = \mathbb{1}(x_i = x_i^*), \quad \text{where } x_i^* = \operatorname*{argmax}_{x_i} \left[\theta_i(x_i) + \sum_{\alpha \in N_i} \delta_i^\alpha(x_i)\right]$$
$$b_\alpha^*(x_\alpha) = \mathbb{1}(x_\alpha = x_\alpha^*), \text{ where } x_\alpha^* = \operatorname*{argmax}_{x_\alpha} \left[\theta_i(x_\alpha) - \sum_{i \in \alpha} \delta_i^\alpha(x_i)\right]$$

where $\mathbb{1}$ is the indicator function. According to the strong duality of linear programming [Boyd and Vandenberghe, 2004],

$$\max_{b \in \mathbb{L}(G)} \langle \theta, b \rangle = \max_b \min_\delta \mathcal{LG}(b, \delta) = \min_\delta \max_b \mathcal{LG}(b, \delta) = \min_\delta L(\delta),$$

we obtain the same dual-decomposition bound as Eq. (2.39).

One can apply the subgradient algorithm to solve the optimization $\min_\delta L(\delta)$. Suppose the decoded value on node $i$ and the decoded values on clique $\alpha$ are,

$$x_i^* = \operatorname*{argmax}_{x_i} \left[\theta_i(x_i) + \sum_{\alpha \in N_i} \delta_i^\alpha(x_i)\right] \text{ and } x_\alpha^* = \max_{x_\alpha} \left[\theta_\alpha(x_\alpha) - \sum_{i \in \alpha} \delta_i^\alpha(x_i)\right], \tag{2.41}$$

respectively. We also denote $x_{\alpha[i]}^*$ as the value of $x_i$ in $x_\alpha^*$ where $i \in \alpha$. Then, one can show that the subgradient of $L(\delta)$ (2.39) w.r.t. $\delta_i^\alpha(x_i)$ is

$$\frac{\partial L(\delta)}{\delta_i^\alpha(x_i)} = \begin{cases} 0, & x_i \neq x_i^* \text{ and } x_i \neq x_{\alpha[i]}^* \\ -1, & x_i = x_i^* \text{ and } x_i \neq x_{\alpha[i]}^* \\ 1, & x_i \neq x_i^* \text{ and } x_i = x_{\alpha[i]}^* \\ 0, & x_i = x_i^* \text{ and } x_i = x_{\alpha[i]}^* \end{cases}$$

thus the subgradient descent algorithm is directly applicable. A well known result about the subgradient descent method is that it is guaranteed to converge to optimality whenever the step sizes are chosen such that $\lim_{t \to \infty} \eta_t = 0$ and $\sum_{t=0}^\infty \eta_t = \infty$ [e.g., Anstreicher and

Wolsey, 2009].

A more popular approach for minimizing $L(\delta)$ (2.39) is the coordinate descent algorithm (e.g., max-product linear programming (MPLP) [Globerson and Jaakkola, 2008]). In Chapter 3, we propose the generalized dual-decomposition algorithm, which extends previous block coordinate descent algorithms to summation and marginal MAP inference.

## 2.4 Learning Methods

This section reviews different parameter estimation methods for graphical models.

### 2.4.1 MLE for Graphical Models

Maximum likelihood estimation (MLE) is the single most common parameter estimation method for probabilistic graphical models. We review MLE for generic Markov random fields (MRFs) and conditional random fields (CRFs).

#### MLE for Markov Random Fields

Assume we have a training set $S = \{\boldsymbol{x}^n\}_{n=1}^N \in \mathcal{X}^N$, and we use the following undirected model defined in (2.4),

$$p(\boldsymbol{x};\theta) = \frac{1}{Z(\theta)} \exp\left[\theta^\top \phi(\boldsymbol{x})\right]; \quad Z(\theta) = \sum_{\boldsymbol{x}} \exp\left[\theta^\top \phi(\boldsymbol{x})\right]$$

The maximum of (averaged) log-likelihood is, [9]

$$\max_{\theta} \ \mathcal{L}(\theta) = \frac{1}{N}\sum_{n=1}^{N} \log p(\boldsymbol{x}^n;\theta) = \frac{1}{N}\sum_{n=1}^{N} \theta^\top \phi(\boldsymbol{x}^n) - \log Z(\theta). \tag{2.42}$$

[9] One can also introduce the regularization term, e.g., a 2-norm of $\theta$, to avoid overfitting.

One can do gradient descent or stochastic gradient descent for this optimization, and one can easily verify the gradient of this log-likelihood function is

$$\nabla_\theta \mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}^n) - \frac{1}{Z(\theta)}\sum_{\boldsymbol{x}}\Big\{\exp\big[\theta^\top\phi(\boldsymbol{x})\big]\cdot\phi(\boldsymbol{x})\Big\} = \frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}^n) - \mathbb{E}_{p(\boldsymbol{x};\theta)}\big[\phi(\boldsymbol{x})\big], \tag{2.43}$$

where $\mathbb{E}_{p(\boldsymbol{x};\theta)}\big[\phi(\boldsymbol{x})\big] = \sum_{\boldsymbol{x}} p(\boldsymbol{x};\theta)\phi(\boldsymbol{x})$ is the model expectation of the sufficient statistics. Note, the stationary point (zero-gradient) enforces an intuitive matching between the empirical moment and the model moment, which is commonly referred to as moment matching. The model expectation is generally intractable, and for $\phi(\boldsymbol{x})$ taken to be the set of indicator functions, it reduce to marginal probabilities (i.e., over-complete representation [Wainwright and Jordan, 2008], and can be approximated using the "pseudo" marginals obtained from belief propagation algorithms, e.g., Algorithm 2.1.

### MLE for Conditional Random Fields

Suppose we have a training set $S = \{(\boldsymbol{x}^n, \boldsymbol{y}^n)\}_{n=1}^N \in (\mathcal{X}\times\mathcal{Y})^N$, and we use the CRF model in (2.5) for structured prediction on $\boldsymbol{y}$. The maximum conditional log-likelihood $\mathcal{CL}(\theta)$ is

$$\max_\theta \ \mathcal{CL}(\theta) = \frac{1}{N}\sum_{n=1}^{N}\log p(\boldsymbol{y}^n|\boldsymbol{x}^n;\theta) = \frac{1}{N}\sum_{n=1}^{N}\big\{\theta^\top\phi(\boldsymbol{x}^n,\boldsymbol{y}^n) - \log Z(\boldsymbol{x}^n,\theta)\big\}. \tag{2.44}$$

One can solve this optimization by gradient descent or stochastic gradient descent, and the gradient is

$$\nabla_\theta \mathcal{CL} = \frac{1}{N}\sum_{n=1}^{N}\Big\{\phi(\boldsymbol{x}^n,\boldsymbol{y}^n) - \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x}^n;\theta)}\big[\phi(\boldsymbol{x}^n,\boldsymbol{y})\big]\Big\}, \tag{2.45}$$

where $\mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x}^n;\theta)}\big[\phi(\boldsymbol{x}^n,\boldsymbol{y})\big] = \sum_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}^n;\theta)\phi(\boldsymbol{x}^n,\boldsymbol{y})$ is the conditional expectation, which is generally intractable and can be approximated using belief propagation algorithms. The

moment matching condition (zero-gradient) is analogous to the MLE of MRF (2.43).

### 2.4.2 Structured SVM

In this subsection we review the structured SVM (SSVM), or max-margin Markov network method [Taskar et al., 2003, Tsochantaridis et al., 2005], which is a popular learning method in structured prediction. Suppose we use the undirected conditional model defined in (2.5), and we have a training set $S = \{(\boldsymbol{x}^n, \boldsymbol{y}^n)\}_{n=1}^N \in (\mathcal{X} \times \mathcal{Y})^N$. The structured SVM minimizes an upper bound of the empirical risk function. The risk is measured by an user-specified empirical loss function $\Delta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n)$, which quantifies the difference between an predictor $\hat{\boldsymbol{y}}^n$ and the ground truth output $\boldsymbol{y}^n$. It is usually difficult to directly minimize this loss function because it is typically non-convex and discontinuous with the model parameter $\theta$ (e.g., Hamming loss). Instead, one adopts surrogate upper bounds to overcome this difficulty.

Assume $\hat{\boldsymbol{y}}^n(\theta)$ is the MAP prediction on instance $\boldsymbol{x}^n$ as defined in (2.16). One can upper bound the empirical loss function $\Delta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n(\theta))$ as follows,

$$\begin{aligned}\Delta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n(\theta)) &\leq \Delta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n(\theta)) + \theta^\top \phi(\boldsymbol{x}^n, \hat{\boldsymbol{y}}^n(\theta))) - \theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}^n) \\ &\leq \max_{\boldsymbol{y}} \left\{\Delta(\boldsymbol{y}^n, \boldsymbol{y}) + \theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y})\right\} - \theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}^n),\end{aligned}$$

where the first inequality holds because $\hat{\boldsymbol{y}}^n(\theta)$ is the MAP prediction (2.16), and the second because it jointly maximizes two terms. Minimizing this upper bound over the training set with a $L_2$ regularization, one obtain the following objective function $\mathcal{SL}$ for structured SVM,

$$\min_{\theta} \ \frac{1}{2}\|\theta\|^2 + C \sum_{n=1}^{N} \Big\{ \max_{\boldsymbol{y}} \left\{\Delta(\boldsymbol{y}^n, \boldsymbol{y}) + \theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y})\right\} - \theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}^n) \Big\}, \tag{2.46}$$

which is often referred to as structured hinge-loss, because it generalize the hinge-loss function in standard SVMs. Note that this objective is convex w.r.t. $\theta$ because the supremum

over sets of linear functions is always convex. Because SSVM explicitly minimizes a surrogate of the desired loss function, it often outperforms the CRF model learned by MLE in many practical applications, especially when the model assumptions are violated or one have limited training data [e.g., Taskar et al., 2003].

One can also use the sub-gradient descent to solve this optimization [Ratliff et al., 2007], and the sub-gradient of the SSVM objective (2.46) is:

$$\nabla_\theta \mathcal{SL} = \theta + C\sum_{n=1}^{N}\phi(\boldsymbol{x}^n, \hat{\boldsymbol{y}}^n) - C\sum_{n=1}^{N}\phi(\boldsymbol{x}^n, \boldsymbol{y}^n), \tag{2.47}$$

$$\text{where} \quad \hat{\boldsymbol{y}}^n = \underset{\boldsymbol{y}\in\mathcal{Y}}{\operatorname{argmax}} \Big\{\Delta(\boldsymbol{y}^n, \boldsymbol{y}) + \theta^\top\phi(\boldsymbol{x}^n, \boldsymbol{y})\Big\} \tag{2.48}$$

where $\hat{\boldsymbol{y}}^n$ is the loss-augmented MAP prediction, which can be approximate via max-product belief propagation or dual-decomposition if the loss function $\Delta(\boldsymbol{y}^n, \boldsymbol{y})$ is decomposed over the undirected model $G = (V, E)$; for example, Hamming loss $\Delta^{\text{hamm}}(\boldsymbol{y}^n, \boldsymbol{y}) := \sum_{i\in V}\mathbb{1}(y_i^n = y_i)$ is decomposed over the nodes.

SVMs are often formulated in terms of a quadratic program (QP) called the constraint form. The constraint form of structured SVM (2.46) is,

$$\begin{aligned} &\min_{\theta,\{\xi_n\geq 0\}} \quad \frac{1}{2}\|\theta\|^2 + C\sum_{n=1}^{N}\xi_n, \\ &\text{s.t.} \ \ \forall\, n,\ \forall\, \boldsymbol{y}\in\mathcal{Y},\ \theta^\top\phi(\boldsymbol{x}^n, \boldsymbol{y}^n) - \theta^\top\phi(\boldsymbol{x}^n, \boldsymbol{y}) \geq \Delta(\boldsymbol{y}^n, \boldsymbol{y}) - \xi_n, \end{aligned} \tag{2.49}$$

where the slack variable $\xi_n$ enforces a soft constraint that the "score" of training instance $(\boldsymbol{x}^n, \boldsymbol{y}^n)$, [10] must be larger than the score of $(\boldsymbol{x}^n, \boldsymbol{y})$ with arbitrary $\boldsymbol{y}\in\mathcal{Y}$ by a large margin $\Delta(\boldsymbol{y}^n, \boldsymbol{y})$. Intuitively, the larger difference between $\boldsymbol{y}$ and the ground truth $\boldsymbol{y}^n$, the larger margin it will enforce. It is also referred to as the *margin rescaling* form of the SSVM because

[10] In the literature of structured SVM, $s(\boldsymbol{x}, \boldsymbol{y}) \overset{\text{def}}{=\!=} \theta^\top\phi(\boldsymbol{x}, \boldsymbol{y})$ is also referred to as score function.

it rescales the margin of the regular support vector machine to $\Delta(\boldsymbol{y}^n, \boldsymbol{y})$ [Tsochantaridis et al., 2005]. It is straightforward show that the optimal $\{\xi_n^*\}_{n=1}^N$ satisfy,

$$\xi_n^* = \max_{\boldsymbol{y}} \Big\{ \Delta(\boldsymbol{y}^n, \boldsymbol{y}) + \theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}) \Big\} - \theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}^n),$$

which obtains the same objective value as its unconstrained form (2.46). One can use the cutting plane training algorithm [Joachims et al., 2009] for this constraint optimization. Note, Eq. (2.49) is a quadratic program (QP) with an exponential number ($N \cdot |\mathcal{Y}|$) of constraints. The cutting plane algorithm actively maintains a working set of constraints by adding the most violated constraint at each iteration, then it updates the current parameter $\theta$ by solving the QP with the working set constraints.

Another interesting formulation of structured SVM is *slack rescaling*,

$$\begin{aligned} \min_{\theta, \{\xi_n \geq 0\}} \quad & \frac{1}{2}\|\theta\|^2 + C \sum_{n=1}^{N} \xi_n, \\ \text{s.t.} \quad & \forall\, n,\ \forall\, \boldsymbol{y} \in \mathcal{Y},\ \theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}^n) - \theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}) \geq 1 - \frac{\xi_n}{\Delta(\boldsymbol{y}^n, \boldsymbol{y})}, \end{aligned}$$

where the slack variable $\xi_n$ itself is rescaled by the loss function $\Delta(\boldsymbol{y}^n, \boldsymbol{y})$. Although it is believed to be more accurate and better-behaved than margin rescaling, it results in a more challenging inference problem which needs to be carefully approximated [e.g., Sarawagi and Gupta, 2008]. In this thesis, we focus on the margin rescaling framework.

Chapter 3

# Dual-decomposition Bounds for Marginal MAP

Marginal MAP inference involves making MAP predictions in systems defined with latent variables or missing information. It is significantly more difficult than pure marginalization and MAP tasks, for which a large class of efficient and convergent variational algorithms, such as dual decomposition, exist. In this chapter, we generalize dual decomposition to a generic *power sum* inference task, which includes marginal MAP, along with pure marginalization and MAP, as special cases. Our method is based on a block coordinate descent algorithm on a new convex decomposition bound, that is guaranteed to converge monotonically, and can be parallelized efficiently. We demonstrate our approach on marginal MAP queries defined on real-world problems from the UAI approximate inference challenge, showing that our framework is faster and more reliable than previous methods. This chapter is based on our work [Ping et al., 2015].

## 3.1 Introduction

Decomposition methods provide a useful and computationally efficient class of bounds on inference problems. As we reviewed in Section 2.3.4 of Chapter 2, dual decomposition meth-

ods for MAP [e.g., Sontag et al., 2011] give a class of easy-to-evaluate upper bounds, which decompose the graphical model into independent components (see Eq. (2.39)). In addition, these full decomposition bounds can be directly optimized using coordinate descent [Werner, 2007, Globerson and Jaakkola, 2008], subgradient updates [Komodakis et al., 2011], or other methods [e.g., Meshi and Globerson, 2011]. It is easy to ensure both convergence, and that the objective is monotonically decreasing (so that more computation always provides a better bound). The resulting bounds can be used either as stand-alone approximation methods [Globerson and Jaakkola, 2008, Komodakis et al., 2011], or as a component of search [Ihler et al., 2012].

In summation problems, a notable decomposition bound is tree-reweighted BP (TRW), which bounds the partition function with a combination of trees [e.g., Wainwright et al., 2005, Meltzer et al., 2009, Jancsary and Matz, 2011, Domke, 2011]. See our detailed review of TRW in Section 2.3.3 of Chapter 2. These bounds are useful in joint inference and learning (or "inferning") frameworks, allowing learning with approximate inference to be framed as a joint optimization over the model parameters and decomposition bound, often leading to more efficient learning [e.g., Meshi et al., 2010]. However, far fewer methods have been developed for marginal MAP problems.

In this chapter, we present a decomposition bound that has a number of desirable properties:

- (1) *Generality*: our bound is sufficiently general to be applied easily to marginal MAP.
- (2) *Any-time*: it yields a bound at any point during the optimization (not just at convergence), so it can be used in an any-time way.
- (3) *Monotonic and convergent*: more computational effort gives strictly tighter bounds; note that (2) and (3) are particularly important for high-width approximations, which are expensive to represent and update.
- (4) Allows *optimization over all parameters*, including the "weights", or fractional

counting numbers, of the approximation; these parameters often have a significant effect on the tightness of the resulting bound.

- (5) *Compact representation*: within a given class of bounds, using fewer parameters to express the bound reduces memory and typically speeds up optimization.

We organize the rest of the chapter as follows. Section 3.2 reviews connections to related work. We derive our decomposed bound in Section 3.3, and present a (block) coordinate descent algorithm for monotonically tightening it in Section 3.4. We report experimental results in Section 3.6 and conclude this chapter in Section 3.7.

## 3.2 State of the Art

Variational upper bounds on MAP and the partition function, along with algorithms for providing fast, convergent optimization, have been widely studied in the last decade. In MAP, dual decomposition and linear programming (LP) methods have become a dominant approach, with numerous optimization methods [Werner, 2007, Globerson and Jaakkola, 2008, Sontag and Jaakkola, 2009, Komodakis et al., 2011, Yarkony et al., 2010, Ruozzi and Tatikonda, 2013, Meshi and Globerson, 2011]. In particular, the dual of LP enables monotonic techniques to tighten the loose approximations by adding cycles into the relaxation [Sontag et al., 2008, Komodakis and Paragios, 2008].

For summation problems, most upper bounds are derived from the tree-reweighted (TRW) family of convex bounds [Wainwright et al., 2005], or more generally conditional entropy decompositions [Globerson and Jaakkola, 2007]. TRW bounds can be framed as optimizing over a convex combination of tree-structured models, or in a dual representation as a message-passing, TRW belief propagation algorithm. This illustrates a basic tension in the resulting bounds: in its primal form Eq. (2.34) [1] (a combination of trees), TRW is inefficient: it

[1]Despite the term "dual decomposition" used in MAP tasks, in this work we refer to decomposition

maintains a weight and $O(|V|)$ parameters for each tree, [2] and a large number of trees may be required to obtain a tight bound; this uses memory and makes optimization slow. On the other hand, the dual, or free energy, form Eq. (2.33) uses only $O(|E|)$ parameters (the TRW messages) to optimize over the set of all possible spanning trees – but, the resulting optimization is only guaranteed to be a upper bound at convergence, making it difficult to use in an anytime fashion (we will demonstrate this point in Experiment Section 3.6.1). Similarly, the gradient of the weights is only correct at convergence, making it difficult to optimize over these parameters; most implementations [e.g., Mooij, 2010] simply adopt fixed weights.

Thus, most algorithms do not satisfy all the desirable properties listed in the introduction. For example, many works have developed convergent message-passing algorithms for convex free energies [e.g., Hazan and Shashua, 2008, 2010]. However, by optimizing the dual they do not provide a bound until convergence, and the representation and constraints on the counting numbers do not facilitate optimizing the bound over these parameters. To optimize counting numbers, Hazan et al. [2012] adopt a more restrictive free energy form requiring positive counting numbers on the local entropies; but this cannot represent marginal MAP, whose free energy involves conditional entropies (equivalent to the difference between two entropy terms; see Eq. (2.23)).

On the other hand, working in the primal domain ensures a bound, but usually at the cost of enumerating a large number of trees. Jancsary and Matz [2011] heuristically select a small number of trees to avoid being too inefficient, while Meltzer et al. [2009] focus on trying to speed up the updates on a given collection of trees. Another primal bound is weighted mini-bucket (WMB, [Liu and Ihler, 2011]), which can represent a large collection of trees

bounds as "primal" bounds, since they can be viewed as directly bounding the result of variable elimination. This is in contrast to, for example, the linear programming relaxation of MAP, which bounds the result only after optimization.

[2] In graphical model $G = (V, E)$, $|V|$ is the number of nodes and $|E|$ is the number of edges.

compactly and is easily applied to marginal MAP using the weighted log partition function viewpoint [Liu, 2014, Marinescu et al., 2014]; however, existing optimization algorithms for WMB are non-monotonic, and often fail to converge, especially on marginal MAP tasks.

While our focus is on variational bounds [Liu and Ihler, 2011, 2013], there are many non-variational approaches for marginal MAP as well. Park and Darwiche [2003] and Yuan and Hansen [2009] provide upper bounds on marginal MAP by reordering the order in which variables are eliminated, and using exact inference in the reordered join-tree; however, this is exponential in the size of the (unconstrained) treewidth, and can easily become intractable. Meek and Wexler [2011] give an approximation closely related to mini-bucket [Dechter and Rish, 2003] to bound the marginal MAP value; however, unlike (weighted) mini-bucket, these bounds cannot be improved iteratively. The same is true for the algorithm of Maua and de Campos [2012], which also has a strong dependence on treewidth. Other examples of marginal MAP algorithms include local search [e.g., Park and Darwiche, 2004] and Markov chain Monte Carlo methods [e.g., Doucet et al., 2002, Yuan et al., 2004]. Most recently, Xue et al. [2016] solve the marginal MAP by transforming the intractable inner sum into a series of optimization problems with parity constraints, which in turn are folded into the joint optimization task.

Another popular class of methods for solving marginal MAP as well as sum-inference, are search based. For example, Marinescu et al. [2014], Lou et al. [2017], Lee et al. [2016], Marinescu et al. [2017] apply heuristic search on AND/OR search trees (or graphs) to provide deterministic bounds. In particular, they use some variational bounds (e.g., WMB) as heuristics to guide the search algorithm (e.g., best-first search), thus a monotonic variational algorithm providing high-quality bound is highly desirable.

## 3.3 Fully Decomposed Upper Bound

In this section, we develop a new general form of upper bound and provide an efficient, monotonically convergent optimization algorithm. Our new bound is based on fully decomposing the graph into disconnected cliques, allowing very efficient local computation, but can still be as tight as WMB or the TRW bound with a large collection of spanning trees once the weights and shifting variables are chosen or optimized properly. Our bound reduces to dual decomposition for MAP inference, but is applicable to more general mixed-inference settings.

Our main result is based on the following generalization of the classical Hölder's inequality [Hardy et al., 1952]:

**Theorem 3.3.1.** *For a given graphical model $p(\boldsymbol{x};\theta)$ as defined in Eq.* (2.1)*,*

$$p(\boldsymbol{x};\theta) = \exp\Big[\sum_{\alpha\in\mathcal{F}}\theta_\alpha(x_\alpha) - \Phi(\theta)\Big],$$

*with cliques $\mathcal{F}=\{\alpha\}$, and the generic power sum inference as defined in Eq.* (2.18)*,*

$$\Phi_{\boldsymbol{\tau}}(\theta) = \log\sum_{\boldsymbol{x}}^{\boldsymbol{\tau}}\exp\Big[\sum_{\alpha\in\mathcal{F}}\theta_\alpha(x_\alpha)\Big] = \log\sum_{x_n}^{\tau_n}\cdots\sum_{x_1}^{\tau_1}\prod_{\alpha\in\mathcal{F}}\exp\Big[\theta_\alpha(x_\alpha)\Big],$$

*with a elimination order $[x_1,\cdots,x_n]$ and a set of non-negative weights $\boldsymbol{\tau}=\{\tau_i\geq 0, i\in V\}$, we define a set of "split weights" $\mathbf{w}^\alpha=\{w_i^\alpha\geq 0, i\in\alpha\}$ on each variable-clique pair $(i,\alpha)$, that satisfies $\sum_{\alpha|\alpha\ni i} w_i^\alpha=\tau_i$. Then we have*

$$\sum_{\boldsymbol{x}}^{\boldsymbol{\tau}}\prod_{\alpha\in\mathcal{F}}\exp\big[\theta_\alpha(x_\alpha)\big] \leq \prod_{\alpha\in\mathcal{F}}\sum_{x_\alpha}^{\mathbf{w}^\alpha}\exp\big[\theta_\alpha(x_\alpha)\big], \tag{3.1}$$

*where the left-hand side is the powered-sum along order $[x_1,\ldots,x_n]$ as defined in* (2.18)*, and the right-hand side is the product of the powered-sums on subvector $x_\alpha$ with weights $\mathbf{w}^\alpha$ along*

*the same elimination order; that is,*

$$\sum_{x_\alpha}^{\mathbf{w}^\alpha} \exp\left[\theta_\alpha(x_\alpha)\right] = \sum_{x_{k_c}}^{w^\alpha_{k_c}} \cdots \sum_{x_{k_1}}^{w^\alpha_{k_1}} \exp\left[\theta_\alpha(x_\alpha)\right],$$

*where* $x_\alpha = [x_{k_1}, \ldots, x_{k_c}]$*, and* $k_1 < \cdots < k_c$ *are ranked to be consistent with the elimination order* $[x_1, \ldots, x_n]$ *as used in the left-hand side.*

*Proof.* Note that Hölder's inequality is

$$\left[\sum_x \prod_j f_j(x)^{1/\xi_0}\right]^{\xi_0} \leq \prod_j \left[\sum_x f_j(x)^{1/\xi_j}\right]^{\xi_j},$$

where $\{f_j(x)\}$ are arbitrary positive functions, and $\{\xi_j\}$ are non-negative numbers that satisfy $\sum_j \xi_j = \xi_0$. Note we extend the inequality by defining power sum with $\xi_j = 0$ to equal the max operator. Our result follows by applying Hölder's inequality on each $x_i$ sequentially along the elimination order $[x_1, x_2, \cdots, x_n]$. □

A key advantage of the bound (3.1) is that it decomposes the joint power sum on $\boldsymbol{x}$ into a product of independent power sums over smaller cliques $x_\alpha$, which significantly reduces computational complexity and enables parallel computation.

### 3.3.1 Including Cost-shifting Variables

In order to increase the flexibility of the upper bound, we introduce a set of *cost-shifting* or *reparameterization* variables $\delta = \{\delta_i^\alpha(x_i) \mid \forall(i,\alpha), i \in \alpha\}$ on each variable-factor pair $(i,\alpha)$, which can be optimized to provide a much tighter upper bound. Note that $\Phi_{\boldsymbol{\tau}}(\theta)$ can be rewritten as,

$$\Phi_{\boldsymbol{\tau}}(\theta) = \log \sum_{\boldsymbol{x}}^{\boldsymbol{\tau}} \exp\Big[\sum_{i\in V}\sum_{\alpha\in N_i} \delta_i^\alpha(x_i) + \sum_{\alpha\in\mathcal{F}} \big(\theta_\alpha(x_\alpha) - \sum_{i\in\alpha} \delta_i^\alpha(x_i)\big)\Big],$$

where $N_i = \{\alpha \mid \alpha \ni i\}$ is the set of cliques incident to $i$. Applying the inequality (3.1), we have that,

$$\Phi_{\boldsymbol{\tau}}(\theta) \leq \sum_{i \in V} \log \sum_{x_i}^{w_i} \exp\Big[\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)\Big] + \sum_{\alpha \in \mathcal{F}} \log \sum_{x_\alpha}^{\mathbf{w}^\alpha} \exp\Big[\theta_\alpha(x_\alpha) - \sum_{i \in \alpha} \delta_i^\alpha(x_i)\Big] \stackrel{\text{def}}{=\!=} L(\delta, \mathbf{w}), \quad (3.2)$$

where the nodes $i \in V$ are also treated as cliques within inequality (3.1), and a new weight $w_i$ is introduced on each variable $i$; the new weights $\mathbf{w} = \{w_i, w_i^\alpha \mid \forall (i,\alpha), i \in \alpha\}$ should satisfy

$$w_i + \sum_{\alpha \in N_i} w_i^\alpha = \tau_i, \quad w_i \geq 0,\ w_i^\alpha \geq 0, \quad \forall (i,\alpha). \quad (3.3)$$

The bound $L(\delta, \mathbf{w})$ is convex w.r.t. the cost-shifting variables $\delta$ and weights $\mathbf{w}$, enabling an efficient optimization algorithm that we present in Section 3.4. As we will discuss in Section 3.4.1, these shifting variables correspond to Lagrange multipliers that enforce a moment matching condition.

### 3.3.2 Variational Form and Connection With Existing Bounds

It is straightforward to see that our bound Eq.(3.2) reduces to dual decomposition bound Eq. (2.39) when applied on MAP inference with all $\tau_i = 0$, and hence $w_i = w_i^\alpha = 0$, that is, [3]

$$L(\delta) = \sum_{i \in V} \max_{x_i} \Big[\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)\Big] + \sum_{\alpha \in \mathcal{F}} \max_{x_\alpha} \Big[\theta_\alpha(x_\alpha) - \sum_{i \in \alpha} \delta_i^\alpha(x_i)\Big].$$

On the other hand, its connection with sum-inference bounds such as WMB and TRW is seen more clearly via a dual representation of Eq. (3.2):

**Theorem 3.3.2.** *The tightest upper bound obtainable by Eq.*(3.2)*, that is,*

[3] In compare to Eq. (2.39), we omit the singleton potentials $\theta_i(x_i)$ here because they can be absorbed into $\theta_\alpha(x_\alpha)$.

$$\min_{\mathbf{w}} \min_{\delta} L(\delta, \mathbf{w}) = \min_{\mathbf{w}} \max_{b \in \mathbb{L}(G)} \big\{ \langle \theta, b \rangle + \sum_{i \in V} w_i H(x_i; b_i) + \sum_{\alpha \in \mathcal{F}} \sum_{i \in \alpha} w_i^{\alpha} H(x_i | x_{\mathrm{pa}_i^{\alpha}} \; ; \; b_{\alpha}) \big\}, \quad (3.4)$$

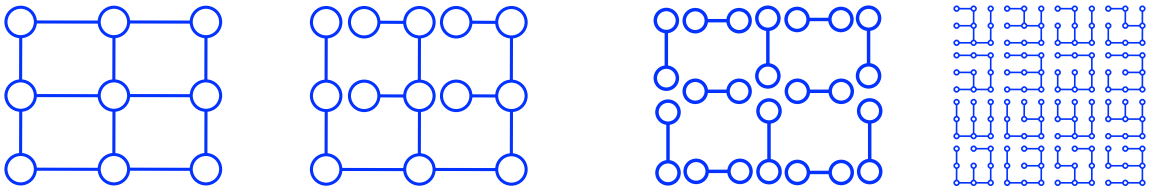
*where* $b = \{b_i(x_i), b_\alpha(x_\alpha) \mid \forall (i, \alpha), i \in \alpha\}$ *is a set of pseudo-marginals (or beliefs) defined on the singleton variables and the cliques, and* $\mathbb{L}$ *is the corresponding local consistency polytope defined by* $\mathbb{L}(G) = \{b \mid b_i(x_i) = \sum_{x_{\alpha \setminus i}} b_\alpha(x_\alpha), \; \sum_{x_i} b_i(x_i) = 1\}$. *Here,* $H(\cdot)$ *are their corresponding marginal or conditional entropies, and* $\mathrm{pa}_i^{\alpha}$ *is the set of variables in* $\alpha$ *that rank later than* $i$*, that is, for the global elimination order* $[x_1, \ldots, x_n]$, $\mathrm{pa}_i^{\alpha} = \{j \in \alpha \mid j \succ i\}$.

The proof details can be found in Appendix A.1. It is useful to compare Theorem 3.3.2 with other dual representations. As the sum of non-negatively weighted conditional entropies, the bound is clearly convex and within the general class of conditional entropy decompositions (CED) [Globerson and Jaakkola, 2007], but unlike generic CED it has a simple and efficient primal form Eq. (3.2). In contrast, the primal form derived in Globerson and Jaakkola [2007] (a geometric program) is computationally infeasible. Comparing to the dual form of WMB in Theorem 4.2 of Liu and Ihler [2011], our bound is as tight as WMB, and hence the class of TRW / CED bounds attainable by WMB [Liu and Ihler, 2011]. In Appendix A section A.1.2, we describe a simple weight setting method which matches our bound to WMB with uniform weights on each mini-bucket.

Most duality-based (free energy) forms are expressed in the following linear combination of local joint entropies rather than conditional entropies [e.g., Yedidia et al., 2005, Hazan and Shashua, 2010],

$$\langle \theta, b \rangle + \sum_{\beta} c_\beta H(x_\beta; b_\beta), \quad (3.5)$$

where $\beta$ refers the region, $c_\beta$ refers the general counting number, and $b_\beta(x_\beta)$ is the local belief. Although our dual representation Eq. (3.4) can be converted into the joint entropy representation (3.5), the converted one has some undesirable properties. To see this, we first



(a) $3 \times 3$ grid (b) WMB: covering tree (c) Full decomposition (d) TRW

Figure 3.1: Illustrating WMB, TRW and our bound on (a) $3 \times 3$ grid. (b) WMB uses a covering tree with a minimal number of splits and cost-shifting. (c) Our decomposition Eq.(3.2) further splits the graph into small cliques (here, edges), introducing additional cost-shifting variables but allowing for easier, monotonic optimization. (d) Primal TRW splits the graph into many spanning trees, requiring even more cost-shifting variables. Note that all three bounds attain the same tightness after optimization.

rewrite our dual representations (3.4) as,

$$\langle \theta, b \rangle + \sum_{i \in V} w_i H(x_i; b_i) + \sum_{\alpha \in \mathcal{F}} \sum_{i \in \alpha} w_i^\alpha \big( H(x_i, x_{\mathrm{pa}_i^\alpha} \ ; \ b_\alpha) - H(x_{\mathrm{pa}_i^\alpha} \ ; \ b_\alpha) \big),$$

where $\mathrm{pa}_i^\alpha$ is the set of variables in $\alpha$ that rank later than $i$. Without loss of generality, assuming $x_\alpha = [x_1, \cdots, x_i, x_j, \cdots x_c]$, and $[i, j]$ are the pairs that are adjacent in the order, we can obtain

$$\langle \theta, b \rangle + \sum_{i \in V} w_i H(x_i; b_i) + \sum_{\alpha \in \mathcal{F}} \Big\{ w_1^\alpha H(x_\alpha; b_\alpha) + \sum_{[i,j] \sqsubseteq \alpha} (w_j^\alpha - w_i^\alpha) H(x_{\mathrm{pa}_i^\alpha} \ ; \ b_{\mathrm{pa}_i^\alpha}) \Big\} \tag{3.6}$$

where belief $b_{\mathrm{pa}_i^\alpha}$ is defined by $b_{\mathrm{pa}_i^\alpha}(x_{\mathrm{pa}_i^\alpha}) = \sum_{x_{\alpha \setminus \mathrm{pa}_i^\alpha}} b_\alpha(x_\alpha)$. One can view Eq. (3.6) in terms of Eq. (3.5), by selecting the region $\beta \in \{i \in V\} \cup \{\alpha \in \mathcal{F}\} \cup \{\mathrm{pa}_i^\alpha \mid \forall (i, \alpha)\}$. However, the resulting counting numbers $c_\beta$ will be the differences of weights $w_j^\alpha - w_i^\alpha$, which obfuscates its convexity, makes it harder to maintain the relative constraints on the counting numbers during optimization, and makes some counting numbers negative (rendering some methods inapplicable [e.g., Hazan et al., 2012], which requires positive counting numbers).

Finally, like most variational bounds in dual form (e.g., the dual form of TRW on the right-hand side of Eq. (2.36)), the right-hand side of Eq.(3.4) has an inner maximization and hence guaranteed to bound $\Phi_{\boldsymbol{\tau}}(\theta)$ only at its optimum. In contrast, our Eq. (3.2) is a primal

bound (hence, a bound for any $\delta$). It is similar to the primal form of TRW on the left-hand side of Eq. (2.36), except that (1) the individual regions are single cliques, rather than spanning trees of the graph, [4] and (2) the fraction weights $\mathbf{w}^\alpha$ associated with each region are vectors, rather than a single scalar. The representation's efficiency can be seen with an example in Figure 3.1, which shows a $3 \times 3$ grid model and three relaxations that achieve the same bound. Assuming $d$ states per variable and ignoring the equality constraints, our decomposition in Figure 3.1(c) uses $24d$ cost-shifting parameters ($\delta$), and 24 weights. WMB (Figure 3.1(b)) is slightly more efficient, with only $8d$ parameters for $\delta$ and and 8 weights, but its lack of decomposition makes parallel and monotonic updates difficult. On the other hand, the equivalent primal TRW uses 16 spanning trees, shown in Figure 3.1(d), for $16 \cdot 8 \cdot d^2 = 128d^2$ parameters, and 16 weights. The increased dimensionality of the optimization slows convergence, and updates are non-local, requiring full message-passing sweeps on the involved trees (although this cost can be amortized in some cases [Meltzer et al., 2009]).

## 3.4 Monotonically Tightening the Bound

In this section, we propose a block coordinate descent algorithm (Algorithm 3.1) to minimize the upper bound $L(\delta, \mathbf{w})$ in Eq.(3.2) w.r.t. the shifting variables $\delta$ and weights $\mathbf{w}$. Our algorithm has a monotonic convergence property, and allows efficient, distributable local computation due to the full decomposition of our bound. Our framework allows generic powered-sum inference, including max-, sum-, or mixed-inference as special cases by setting different weights.

[4] While non-spanning subgraphs can be used in the primal TRW form, doing so only leads to loose bounds; in contrast, our decomposition's terms consist of individual cliques.

### 3.4.1 Moment Matching and Entropy Matching

We start with deriving the gradient of $L(\delta, \mathbf{w})$ w.r.t. $\delta$ and $\mathbf{w}$. We show that the zero-gradient equation w.r.t. $\delta$ has a simple form of moment matching that enforces a consistency between the *singleton beliefs* with their related *clique beliefs*, and that of weights $\mathbf{w}$ enforces a consistency of *marginal* and *conditional entropies*.

**Theorem 3.4.1.** *(1) For $L(\delta, \mathbf{w})$ in (3.2), its zero-gradient w.r.t. $\delta_i^\alpha(x_i)$ is*

$$\frac{\partial L}{\partial \delta_i^\alpha(x_i)} = \mu_i(x_i) - \sum_{x_{\alpha\setminus i}} \mu_\alpha(x_\alpha) = 0, \tag{3.7}$$

*where $\mu_i(x_i) \propto \exp\left[\frac{1}{w_i}\sum_{\alpha\in N_i}\delta_i^\alpha(x_i)\right]$ can be interpreted as a singleton belief on $x_i$, and $\mu_\alpha(x_\alpha)$ can be viewed as clique belief on $x_\alpha$, defined with a chain rule (assuming $x_\alpha = [x_1, \ldots, x_c]$), $\mu_\alpha(x_\alpha) = \prod_{i=1}^c \mu_\alpha(x_i|x_{i+1:c})$; $\mu_\alpha(x_i|x_{i+1:c}) = (Z_{i-1}(x_{i:c})/Z_i(x_{i+1:c}))^{1/w_i^\alpha}$, where $Z_i$ is the partial powered-sum up to $x_i$ on the clique, that is,*

$$Z_i(x_{i+1:c}) = \sum_{x_i}^{w_i^\alpha} \cdots \sum_{x_1}^{w_1^\alpha} \exp\Big[\theta_\alpha(x_\alpha) - \sum_{i\in\alpha}\delta_i^\alpha(x_i)\Big], \quad Z_0(x_\alpha) = \exp\Big[\theta_\alpha(x_\alpha) - \sum_{i\in\alpha}\delta_i^\alpha(x_i)\Big],$$

*where the summation order should be consistent with the global elimination order $\mathbf{o} = [x_1, \ldots, x_n]$.*

*(2) The gradients of $L(\delta, \mathbf{w})$ w.r.t. the weights $\{w_i, w_i^\alpha\}$ are marginal and conditional entropies defined on the beliefs $\{\mu_i, \mu_\alpha\}$, respectively,*

$$\frac{\partial L}{\partial w_i} = H(x_i; \mu_i), \qquad \frac{\partial L}{\partial w_i^\alpha} = H(x_i|x_{i+1:c}; \mu_\alpha) = -\sum_{x_\alpha} \mu_\alpha(x_\alpha)\log\mu_\alpha(x_i|x_{i+1:c}). \tag{3.8}$$

*Therefore, the optimal weights should satisfy the following KKT condition*

$$w_i\big(H(x_i;\mu_i) - \bar{H}_i\big) = 0, \quad w_i^\alpha\big(H(x_i|x_{i+1:c};\mu_\alpha) - \bar{H}_i\big) = 0, \quad \forall(i,\alpha) \tag{3.9}$$

**Algorithm 3.1** Generalized Dual-decomposition (GDD)

**Input:** weights $\{\tau_i \mid i \in V\}$, elimination order $\mathbf{o}$.
**Output:** the optimal $\delta^*, \mathbf{w}^*$ giving tightest upper bound $\mathrm{L}(\delta^*, \mathbf{w}^*)$ for $\Phi_\tau(\theta)$ in (3.2).
initialize $\delta = 0$ and weights $\mathbf{w} = \{w_i, w_i^\alpha\}$.
**repeat**
  **for** node $i$ (in parallel with node j, $(i,j) \notin E$) **do**
    **if** $\tau_i = 0$ **then**
      update $\boldsymbol{\delta}_{N_i} = \{\delta_i^\alpha | \forall \alpha \in N_i\}$ with the closed-form update (3.10);
    **else if** $\tau_i \neq 0$ **then**
      optimize $\boldsymbol{\delta}_{N_i}$ and $\mathbf{w}_{N_i}$ with gradient-based methods using (3.7) and (3.11), combined with a backtracking line search;
    **end if**
  **end for**
**until** convergence
$\delta^* \leftarrow \delta$, $\mathbf{w}^* \leftarrow \mathbf{w}$, and evaluate $\mathrm{L}(\delta^*, \mathbf{w}^*)$ by (3.2);

*Remark.* GDD solves MAP, marginalization and marginal MAP inference by setting different values of weights $\{\tau_i\}$.

*where* $\bar{H}_i = \big(w_i H(x_i; \mu_i) + \sum_\alpha w_i^\alpha H(x_i | x_{i+1:c}; \mu_\alpha)\big)/\tau_i$ *is the average entropy on node* $i$.

The proof details can be found in Section A.2 of the Appendix A. The matching condition (3.7) enforces that $\mu = \{\mu_i, \mu_\alpha \mid \forall (i,\alpha)\}$ belong to the local consistency polytope $\mathbb{L}$ as defined in Theorem 3.3.2; similar moment matching results appear commonly in variational inference algorithms [e.g., Wainwright et al., 2005]. As we reviewed in Section 2.3.3, Wainwright et al. [2005] also derive a gradient of the weights, but it is based on the free energy form Eq. (2.33), and is correct only after full message-passing optimization; our form holds at any point, enabling efficient joint optimization of $\delta$ and $\mathbf{w}$.

### 3.4.2 Block Coordinate Descent

We derive a block coordinate descent method in Algorithm 3.1 to minimize our bound, in which we sweep through all the nodes $i$ and update each block $\boldsymbol{\delta}_{N_i} = \{\delta_i^\alpha(x_i) \mid \forall \alpha \in N_i\}$ and $\mathbf{w}_{N_i} = \{w_i, w_i^\alpha \mid \forall \alpha \in N_i\}$ with the neighborhood parameters fixed. Our algorithm applies two update types, depending on whether the variables have zero weight: (1) For nodes $i$

with $\tau_i = 0$ and $i$ ranks later than $\{j \mid \tau_j \neq 0 \text{ and } (i,j) \in E\}$ in elimination order $\mathbf{o}$ (e.g., max nodes $i \in B$ in marginal MAP), we derive a closed-form coordinate descent rule for the associated shifting variables $\boldsymbol{\delta}_{N_i}$; these nodes do not require to optimize $\mathbf{w}_{N_i}$ since it is fixed to be zero. (2) For nodes with $\tau_i \neq 0$ (e.g., sum nodes $i \in A$ in marginal MAP), we lack a closed form update for $\boldsymbol{\delta}_{N_i}$ and $\mathbf{w}_{N_i}$, and optimize by local gradient descent combined with line search.

The lack of a closed form coordinate update for nodes $\tau_i \neq 0$ is mainly because the order of power sums with different weights cannot be exchanged. However, the gradient descent inner loop is still efficient, because each gradient evaluation only involves the local variables in clique $\alpha$.

**Closed-form Update.** For any node $i$ with $\tau_i = 0$ and $i$ ranks later than $\{j \mid \tau_j \neq 0 \text{ and } (i,j) \in E\}$ in elimination order $\mathbf{o}$ (e.g., max nodes $i \in B$ in marginal MAP), and its associated $\boldsymbol{\delta}_{N_i} = \{\delta_i^\alpha(x_i) \mid \forall \alpha \in N_i\}$, the following update gives a closed form solution for the zero (sub-)gradient equation in (3.7) (keeping the other $\{\delta_j^\alpha \mid j \neq i, \forall \alpha \in N_i\}$ fixed):

$$\delta_i^\alpha(x_i) \leftarrow \frac{|N_i|}{|N_i|+1}\gamma_i^\alpha(x_i) - \frac{1}{|N_i|+1}\sum_{\beta \in N_i \setminus \alpha} \gamma_i^\beta(x_i), \tag{3.10}$$

where $|N_i|$ is the number of neighborhood cliques, and

$$\gamma_i^\alpha(x_i) = \log \sum_{x_{\alpha \setminus i}}^{\mathbf{w}_{\setminus i}^\alpha} \exp\left[\theta_\alpha(x_\alpha) - \sum_{j \in \alpha \setminus i} \delta_j^\alpha(x_j)\right],$$

where $x_{\alpha \setminus i} = \{x_j : j \in \alpha, j \neq i\}$, $\mathbf{w}_{\setminus i}^\alpha = \{w_j^\alpha : j \in \alpha, j \neq i\}$. Note that the update in (3.10) works regardless of the weights of nodes $\{\tau_j \mid \forall j \in \alpha,\ \forall \alpha \in N_i\}$ in the neighborhood cliques; when all the neighboring nodes also have zero weight ($\tau_j = 0$ for $\forall j \in \alpha,\ \forall \alpha \in N_i$), it is analogous to the "star" update of dual decomposition for MAP [Sontag et al., 2011]. The detailed derivation is shown in Proposition A.5.1 and A.5.2 in Appendix A.

The update in (3.10) for whole $\boldsymbol{\delta}_{N_i}$ can be calculated with a cost of only $O(|N_i| \cdot d^{|\alpha|})$, where $d$ is the number of states of $x_i$, and $|\alpha|$ is the clique size, by computing and saving all the shared $\{\gamma_i^\alpha(x_i) \mid \forall \alpha \in N_i\}$ before updating $\boldsymbol{\delta}_{N_i}$. Furthermore, the updates of $\boldsymbol{\delta}_{N_i}$ for different nodes $i$ are independent if they are not directly connected by some clique $\alpha$; this makes it easy to parallelize the coordinate descent process by partitioning the graph into independent sets, and parallelizing the updates within each set. More elaborate update scheduling, such as the priority queue based schemes in residual BP [Elidan et al., 2006] or residual splash [Gonzalez et al., 2009], can be applied to improve empirical convergence and parallelism. We leave them for future work.

**Local Gradient Descent.** For nodes with $\tau_i \neq 0$ (or $i \in A$ in marginal MAP), there is no closed-form solution for $\{\delta_i^\alpha(x_i)\}$ and $\{w_i, w_i^\alpha\}$ to minimize the upper bound. However, because of the fully decomposed form, the gradient w.r.t. $\boldsymbol{\delta}_{N_i}$ and $\mathbf{w}_{N_i}$, (3.7)–(3.8), can be evaluated efficiently via local computation with $O(|N_i| \cdot d^{|\alpha|})$, and again can be parallelized between nonadjacent nodes. To handle the normalization constraint (3.3) on $\mathbf{w}_{N_i}$, we use an exponential gradient descent: let $w_i = \exp(v_i)/\big[\exp(v_i) + \sum_\alpha \exp(v_i^\alpha)\big]$ and $w_i^\alpha = \exp(v_i^\alpha)/\big[\exp(v_i) + \sum_\alpha \exp(v_i^\alpha)\big]$; taking the gradient w.r.t. $v_i$ and $v_i^\alpha$ and transforming back gives the following update

$$
\begin{aligned}
& w_i \leftarrow w_i \exp\big[-\eta w_i\big(H(x_i;\mu_i) - \bar{H}_i\big)\big], \quad w_i^\alpha \leftarrow w_i^\alpha \exp\big[-\eta w_i^\alpha H(x_i|x_{\mathrm{pa}_i^\alpha};\mu_\alpha)\big] \\
& \text{then,} \quad w_i \leftarrow w_i/\big(w_i + \sum_{\alpha\in N_i} w_i^\alpha\big), \qquad w_i^\alpha \leftarrow w_i^\alpha/\big(w_i + \sum_{\alpha\in N_i} w_i^\alpha\big)
\end{aligned} \tag{3.11}
$$

where $\eta$ is the step size, $\mathrm{pa}_i^\alpha = \{j \in \alpha \mid j \succ i\}$ and the entropy terms are defined at Eq. (3.9). In our implementation, to avoid numerical issues with very small positive weights $w_i$ and $w_i^\alpha$, we always make them larger than a small constant $\epsilon = 0.002$. We use the minFunc package [Schmidt, 2005] and try different options. We find that (1) a few local gradient steps (e.g., 5) with the backtracking line search can work well in practice. and (2) a few local quasi-Newton steps (e.g., 5) of L-BFGS will give the best result. The Newton's method

are also applicable, but one need carefully regularize the near-singular Hessian cases. See our derivation of Hessian matrix in Appendix A.4.

## 3.5 Extensions to Junction Graph

Our bound (3.2) uses a standard "factor graph" representation in which the cost-shifts $\{\delta_i^\alpha\}$ are defined for each variable-factor pair $(i, \alpha)$, and are functions of single variables $x_i$. We can extend our bound to use more general shifting parameters using a junction graph representation; this allows us to exploit higher order clique structures, leading to better performance.

Let $(\mathcal{C}, \mathcal{S})$ be a junction graph of $p(\boldsymbol{x}; \theta)$ where $\mathcal{C} = \{c \mid c \subset V\}$ is the set of clusters, and $\mathcal{S} = \{s = c_k \cap c_l \mid \ c_k, c_l \in \mathcal{C}\}$ is the set of separators. Assume $p(\boldsymbol{x}; \theta)$ can be reparameterized into the form,

$$p(\boldsymbol{x}; \theta) = \exp\Big[\sum_{c \in \mathcal{C}} \theta_c(x_c) - \Phi(\theta)\Big], \tag{3.12}$$

and the weighted log partition function is rewritten as $\Phi_{\boldsymbol{\tau}}(\theta) = \log \sum_{\boldsymbol{x}}^{\boldsymbol{\tau}} \exp\Big[\sum_{c \in \mathcal{C}} \theta_c(x_c)\Big]$. Similar to the derivation of bound (3.2) in the main text, we can apply Theorem 3.3.1, but with a set of more general cost-shifting variables $\delta_s^c$, defined on each adjacent separator-cluster pair $(s, c)$; this gives the more general upper bound,

$$\Phi_{\boldsymbol{\tau}}(\theta) \leq \sum_{s \in \mathcal{S}} \log \sum_{x_s}^{\mathbf{w}^s} \exp\Big[\sum_{c \supseteq s} \delta_s^c(x_s)\Big] + \sum_{c \in \mathcal{C}} \log \sum_{x_c}^{\mathbf{w}^c} \exp\Big[\theta_c(x_c) - \sum_{s \subseteq c} \delta_s^c(x_s)\Big], \tag{3.13}$$

where we introduce the set of non-negative weights $\mathbf{w}^s = \{w_i^s \mid i \in s\}$ on each separator and $\mathbf{w}^c = \{w_i^c \mid i \in c\}$ on each cluster, which should satisfy $\sum_{s \in N_i^{se}} w_i^s + \sum_{c \in N_i^c} w_i^c = \tau_i$, where $N_i^{se} = \{s \mid i \in s\}$ are all the separators that include node $i$, and $N_i^c = \{c \mid i \in c\}$ are all

the clusters that include node $i$. Obviously, our earlier bound (3.2) in the main text can be viewed as a special case of (3.13) with a special junction graph whose separators consist of only single variables, that is, $\mathcal{S} = V$.

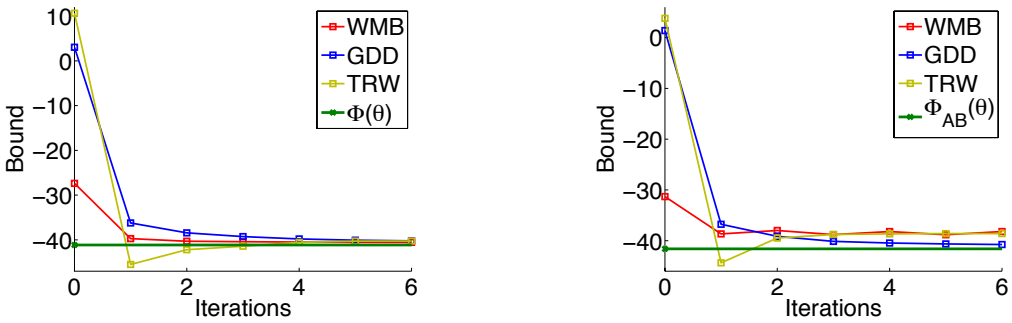
A block coordinate descent algorithm similar to Algorithm 3.1 can be derived to optimize the junction graph bound. In this case, we sweep through all the separators $s$ and perform block coordinate update on all $\{\delta_s^c | \forall c \supseteq s\}$ at each iteration. Similarly to Algorithm 1, we can derive a closed-form update for separators with all-zero weights (that is, $\tau_i = 0$, $\forall i \in s$, corresponding to $s \subseteq B$ in marginal MAP), and perform local gradient descent otherwise.

## 3.6 Experiments

In this section, we compare our algorithm with other state-of-the-art inference algorithms on both toy Ising model and real-world graphical models from recent UAI inference challenges,

### 3.6.1 Ising Model

Our GDD directly optimizes a primal bound, and is thus guaranteed to be an upper bound of the partition function even before the algorithm converges, enabling a desirable "any-time" property. In contrast, typical implementations of tree reweighted (TRW) belief propagation optimize the dual free energy function [Wainwright et al., 2005], and are not guaranteed to be a bound before convergence. We illustrate this point using an experiment on a toy $5 \times 5$ Ising grid, with parameters generated by normal ditribution $N(0, 2)$ and half nodes selected as max-nodes for marginal MAP. Figure 3.2(a)-(b) shows the TRW free energy objective and GDD, WMB upper bounds across iterations; each iteration of the different algorithms corresponds to a full sweep over the graph. Note that the dual formulation (TRW) is not a bound until convergence; for example, at iteration 1, its objective function is below the true $\Phi$. We can see that GDD and WMB always give valid upper bounds.




(a) sum-inference (b) marginal MAP

Figure 3.2: Sum-inference and marginal MAP results on a toy Ising model ($5 \times 5$ grid). Both (a) and (b) show the TRW free energy and GDD, WMB bounds across iterations. The TRW free energy does not provide a bound until convergence. Instead, GDD and WMB provide valid upper bound at any time. (best viewed in color)

### 3.6.2 UAI Inference Challenges

In this subsection, we demonstrate our algorithm on a set of real-world graphical models from recent UAI inference challenges, including two diagnostic Bayesian networks with 203 and 359 variables and max domain sizes 7 and 6, respectively, and several MRFs for pedigree analysis with up to 1289 variables, max domain size of 7 and clique size 5.[5] We construct marginal MAP problems on these models by randomly selecting various percentages of the variables to be max nodes, and the rest as sum nodes.

We implement several algorithms that optimize the same primal marginal MAP bound, including our GDD (Algorithm 3.1), the WMB algorithm in [Liu and Ihler, 2011] with $ibound = 1$, which uses the same cliques and a fixed point heuristic for optimization, and an off-the-shelf L-BFGS implementation [Schmidt, 2005] that directly optimizes our decomposed bound. For comparison, we also computed several related primal bounds, including standard mini-bucket [Dechter and Rish, 2003] and elimination reordering [Park and Darwiche, 2003, Yuan and Hansen, 2009], limited to the same computational limits ($ibound = 1$). We also tried MAS [Meek and Wexler, 2011] but found its bounds extremely loose.[6]

[5] See http://graphmod.ics.uci.edu/uai08/Evaluation/Report/Benchmarks.

[6] The instances tested have many zero probabilities, which make finding lower bounds difficult; since MAS' bounds are symmetrized, this may contribute to its upper bounds being loose.

Decoding (finding a configuration $\hat{\boldsymbol{x}}_B$) is more difficult in marginal MAP than in MAP. We decode each node $i \in B$ locally on its reparametrization

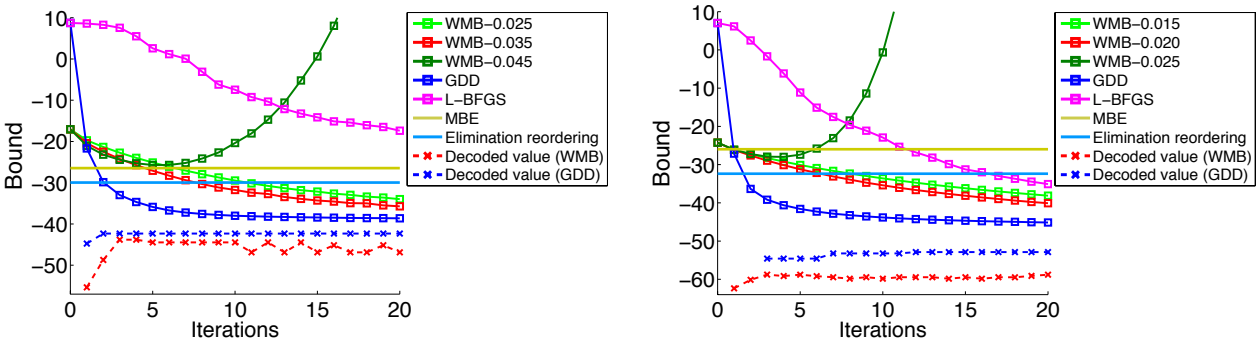
$$\hat{x}_i = \operatorname*{argmax}_{x_i} \big[ \sum_{\alpha \in N_i} \delta_i^{\alpha}(x_i) \big],$$

which is standard in dual decomposition [Sontag et al., 2011]. However, evaluating the objective, $Q(\hat{\boldsymbol{x}}_B) = \log \sum_{\boldsymbol{x}_A} \exp \big[\theta(\boldsymbol{x}_A, \hat{\boldsymbol{x}}_B)\big]$ in Eq. (2.17), involves a potentially difficult sum over $x_A$, making it hard to score each decoding. For this reason, we evaluate the score of each decoding, but show the most recent decoding rather than the best (as is standard in MAP) to simulate behaviour in practice.

Figure 3.3 and Figure 3.5 compare the convergence of the different algorithms, where we define the iteration of each algorithm to correspond to a full sweep over the graph, with the same order of time complexity: one iteration for GDD is defined in Algorithm 3.1; for WMB represents a full forward and backward message pass, as in Algorithm 2 of Liu and Ihler [2011]; and for L-BFGS is a joint quasi-Newton step on all variables. The elimination order that we use is obtained by a weighted-min-fill heuristic [Dechter, 2013] constrained to eliminate the sum nodes first.

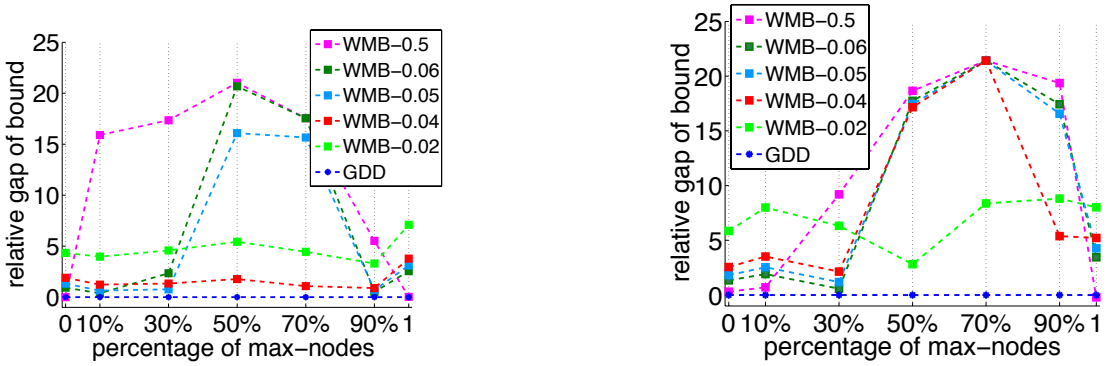
#### Diagnostic Bayesian Networks.

Figure 3.3(a)-(b) shows that our GDD converges quickly and monotonically on both the networks, while WMB does not converge without proper damping; we experimented different damping ratios for WMB, and found that it is slower than GDD even with the best damping ratio found (e.g., in Figure 3.3(a), WMB works best with damping ratio 0.035 (WMB-0.035), but is still significantly slower than GDD). Our GDD also gives better decoded marginal MAP solution $x_B$ (obtained by rounding the singleton beliefs). Both WMB and our GDD provide a much tighter bound than the non-iterative mini-bucket elimination (MBE) [Dechter and




(a) BN-1 (203 nodes) (b) BN-2 (359 nodes)

Figure 3.3: Marginal MAP results on BN-1 and BN-2 with 50% randomly selected max-nodes. We plot the upper bounds of different algorithms across iterations; the objective function $Q(x_B)$ (2.17) of the decoded solutions $x_B$ are also shown (dashed lines). At the beginning, $Q(x_B)$ may equal to $-\infty$ because of zero entries in BN-1 and BN-2 model. (best viewed in color)

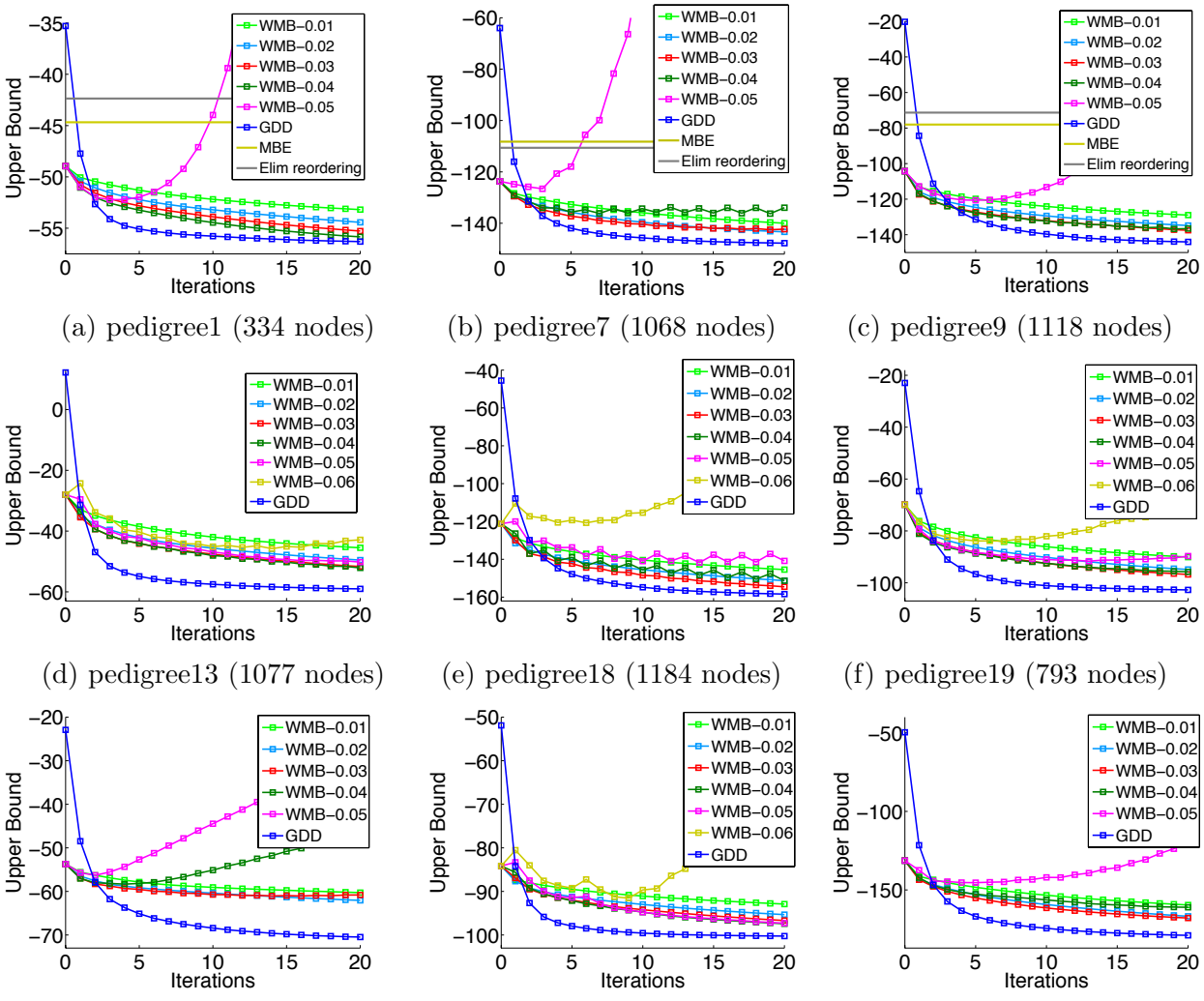



(a) BN-1 (b) BN-2

Figure 3.4: More marginal MAP results with various percentages of randomly selected max-nodes on two diagnostic Bayesian networks. Note that 0% of max-nodes corresponds to sum-inference, and 100% max-nodes corresponds to MAP. We report the best results obtained by GDD and WMB with 20 iterations. (Best viewed in color)

Rish, 2003] or reordered elimination [Park and Darwiche, 2003, Yuan and Hansen, 2009] methods.

In addition to the marginal MAP results with 50% max-nodes on BN-1 and BN-2, we vary the percentage of max-nodes when generating the marginal MAP problems; the reported results in Figure 3.4(a)-(b) are the best bound obtained by the different algorithms within the first 20 iterations. In all cases, GDD's results are as good or better than WMB. WMB-




(a) pedigree1 (334 nodes) (b) pedigree7 (1068 nodes) (c) pedigree9 (1118 nodes)

(d) pedigree13 (1077 nodes) (e) pedigree18 (1184 nodes) (f) pedigree19 (793 nodes)

(g) pedigree20 (437 nodes) (h) pedigree23 (402 nodes) (i) pedigree30 (1289 nodes)

Figure 3.5: Marginal MAP inference on nine pedigree linkage analysis models. We randomly select half the nodes as max-nodes in these models. We tune the damping rate of WMB from 0.01 to 0.06, but we omit WMB-0.06 in the plot if WMB-0.05 is already diverged. (Best viewed in color)

0.5 (WMB with damping ratio 0.5) appears to work well on sum-only and max-only (MAP) problems, i.e., when the percentage of max-nodes equals 0% and 100% respectively, but performs very poorly on intermediate settings. The far more heavily damped WMB-0.04 or WMB-0.02 work better on average, but have much slower convergence.

**Genetic Pedigree Instances.**

Figure 3.5 shows similar results on a set of pedigree instances from the UAI08 inference challenge. We construct marginal MAP problems by randomly selected 50% of nodes to be max-nodes. Again, GDD outperforms WMB even with the best possible damping, and out-performs the non-iterative bounds after only one iteration (a single pass through the graph).

## 3.7 Conclusion

In this chapter, we propose a new class of decomposition bounds for general powered-sum inference, which is capable of representing a large class of primal variational bounds but is much more computationally efficient. Unlike previous primal sum bounds, our bound decomposes into computations on small, local cliques, increasing efficiency and enabling parallel and monotonic optimization. We derive a block coordinate descent algorithm for optimizing our bound over both the cost-shifting parameters (reparameterization) and weights (fractional counting numbers), which generalizes dual decomposition and enjoy similar monotonic convergence property. Taking the advantage of its monotonic convergence, our new algorithm can be widely applied as a building block for improved heuristic construction in search, or more efficient learning algorithms.

Chapter 4

# Marginal Structured SVM

In this chapter, we propose a marginal structured SVM (MSSVM) model for structured prediction with hidden variables. MSSVM properly handles the uncertainty of hidden variables, and can significantly outperform the previously proposed latent structured SVM method (LSSVM; Yu and Joachims [2009a]) and other state-of-art methods, especially when the uncertainty of hidden variables is large. Our method also results in a smoother objective function, making gradient-based optimization of MSSVMs converge significantly faster than for LSSVMs. We also show that our method consistently outperforms hidden conditional random fields (HCRFs; Quattoni et al. [2007a]) on both simulated and real-world datasets. Furthermore, we propose a unified framework that includes both our and several other existing methods as special cases, and provides insights into the comparison of different models in practice. This Chapter is based on our work [Ping et al., 2014].

## 4.1 Introduction

Conditional random fields (CRFs) [Lafferty et al., 2001] and structured SVMs (SSVMs) [Taskar et al., 2003, Tsochantaridis et al., 2005] are standard tools for structured prediction. However, many practical cases are not well handled by these tools, due to the presence of latent variables or partially labeled datasets. For example, one approach to image segmen-

tation classifies each pixel into a predefined semantic category. While it is expensive to collect labels for every single pixel (perhaps even impossible for ambiguous regions), partially labeled data are relatively easy to obtain [e.g., Verbeek and Triggs, 2007]. Examples also arise in natural language processing, such as semantic role labeling, where the semantic predictions are inherently coupled with latent syntactic relations [Naradowsky et al., 2012]. However, accurate syntactic annotations are unavailable in many language resources.

In past few years, several solutions have been proposed to address hidden variable problems in structured prediction. Perhaps the most notable of these are hidden conditional random fields (HCRFs) [Quattoni et al., 2007a] and latent structured SVMs (LSSVMs) [Yu and Joachims, 2009a], which are derived from conditional random fields and structured SVMs, respectively. However, both approaches have several shortcomings. CRF-based models often perform worse than SSVM-based methods in practical datasets, especially when the number of training instances is small or the model assumptions are heavily violated [e.g., Taskar et al., 2003]. On the other hand, LSSVM relies on a joint maximum *a posteriori* (MAP) procedure that assigns the hidden variables to deterministic values, and does not take into account their uncertainty. Unfortunately, this can produce poor predictions of the output variables even for exact models [Liu and Ihler, 2013]. A better approach is to average over possible states, corresponding to a *marginal MAP* inference task [Koller and Friedman, 2009a, Liu and Ihler, 2013] that marginalizes the hidden variables before optimizing over the output variables.

**Contributions.** We propose a novel structured SVM algorithm that takes into account the uncertainty of the hidden variables, by incorporating marginal MAP inference that "averages" over the possible hidden states. We show that our method performs significantly better than LSSVM and other state of art methods, especially when the uncertainty of the hidden variables is high. Our method also inherits the general advantages of structured SVMs and consistently outperforms HCRFs, especially when the training sample size is small. We also

study the effect of different training algorithms under various models. In particular we show that gradient-based algorithms for our framework are much more efficient than for LSSVM, because our objective function is smoother than that of LSSVM as it marginalizes, instead of maximizes, over the hidden variables. Finally, we propose a unified framework that includes both our and existing methods as special cases, and provide general insights on the choice of models and optimization algorithms for practitioners.

We organize the rest of the chapter as follows. In Section 4.2, we introduce related work. We present background and notation in Section 4.3, and derive our marginal structured SVM in Section 4.4. The unified framework is proposed in Section 4.5. Learning and inference algorithms for the model are presented in Section 4.6. We report experimental results in Section 4.7 and conclude the chapter in Section 4.8.

## 4.2 Related Work

HCRFs naturally extend CRFs to include hidden variables, and have found numerous applications in areas such as object recognition [Quattoni et al., 2004] and gesture recognition [Wang et al., 2006]. HCRFs have the same pros and cons as general CRFs; in particular, they perform well when the model assumptions hold and when there are enough training instances, but may otherwise perform badly. Alternatively, the LSSVM [Yu and Joachims, 2009a] is an extension of structured SVM that handles hidden variables, with wide application in areas like object detection [Zhu et al., 2010], human action recognition [Wang and Mori, 2009], document-level sentiment classification [Yessenalina et al., 2010] and link prediction [Xu et al., 2013]. However, LSSVM relies on a joint MAP procedure, and may not perform well when a non-trivial uncertainty exists in the hidden variables. Recently, Schwing et al. [2012] proposed an $\epsilon$-extension framework for discriminative graphical models with hidden variables that includes both HCRFs and LSSVM as special cases.

A few recent works also incorporate uncertainty over hidden variables explicitly into their optimization frameworks. For example, Miller et al. [2012] proposed a max margin min-entropy (M3E) model that minimizes an uncertainty measure on hidden variables while performing max-margin learning. They assume that minimizing hidden uncertainty will improve the output accuracy. This is valid in some applications, such as object detection, where reducing the uncertainty of object location can improve the category prediction. However, in cases like image segmentation, the missing labels may come from ambiguous regions, and maintaining that ambiguity can be important. In another work, Kumar et al. [2012] proposes a learning procedure that encourages agreement between two separate models – one for predicting outputs and another for representing the uncertainty over the hidden variables. They model the uncertainty of hidden variable during training, and rely on a joint MAP procedure during prediction.

Our proposed method builds on recent work for marginal MAP inference [Koller and Friedman, 2009a, Liu and Ihler, 2013], which averages over the hidden variables (or variables that are not of direct interest), and then optimizes over the output variables (or variables of direct interest). In many domains, marginal MAP can provide significant improvement over joint MAP estimation, which jointly optimizes hidden and output variables; recent examples include blind deconvolution in computer vision [Fergus et al., 2006, Levin et al., 2011] and relation extraction and semantic role labeling in natural language processing [Naradowsky et al., 2012]. Unfortunately, marginal MAP tasks on graphical models are notoriously difficult; marginal MAP can be NP-hard even when the underlying graphical model is tree-structured [Koller and Friedman, 2009a]. Recently, Liu and Ihler [2013] proposed efficient variational algorithms that approximately solve marginal MAP. In our work, we use their mixed-product belief propagation algorithm as our inference component.

Sub-gradient decent (SGD) [Ratliff et al., 2007] and the concave-convex procedure (CCCP) [Yuille and Rangarajan, 2003] are two popular training algorithms for structured prediction

problems. Generally, SGD is straightforward to implement and effective in practice, but may be slow to converge, especially on non-convex and non-smooth objective functions as arise in LSSVMs. CCCP is a general framework for minimizing non-convex functions by transforming the non-convex optimization into a sequence of convex optimizations by iteratively linearizing the non-convex component of the objective. It has been applied widely in many areas of machine learning, particularly when hidden variables or missing data are involved. We explore both these training methods and compare them across the various models we consider.

## 4.3 Structured Prediction with Hidden Variables

In this section we review the background on structured prediction with hidden variables. Assume we have structured input-output pairs $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$, $\mathcal{Y}$ are the spaces of the input and output variables. In many applications, this input-output relationship is not only characterized by $(\boldsymbol{x}, \boldsymbol{y})$, but also depends on some unobserved hidden or latent variables $\boldsymbol{h} \in \mathcal{H}$. Suppose $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})$ follows a conditional model, [1]

$$p(\boldsymbol{y}, \boldsymbol{h}|\boldsymbol{x}; \theta) = \frac{1}{Z(x; \theta)} \exp\, [\theta^\top \phi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})], \tag{4.1}$$

where $\phi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h}) : \mathcal{X} \times \mathcal{Y} \times \mathcal{H} \to \mathbb{R}^D$ is a set of features which describe the relationships among the $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})$, and $\theta \in \mathbb{R}^D$ are the corresponding log-linear weights, or model parameters. The function $Z(\boldsymbol{x}; \theta)$ is the normalization constant, or *partition function*,

$$Z(\boldsymbol{x}; \theta) = \sum_{\boldsymbol{y}} \sum_{\boldsymbol{h}} \exp\, [\theta^\top \phi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})].$$

[1] See more illustration in Section 2.1.3 of Chapter 2.

Assuming the weights $\theta$ are known, the LSSVM of Yu and Joachims [2009a] decodes the output variables $\boldsymbol{y}$ given input variables $\boldsymbol{x}$ by performing a joint maximum *a posteriori* (MAP) inference,

$$[\tilde{\boldsymbol{y}}(\theta), \tilde{\boldsymbol{h}}(\theta)] = \underset{(\boldsymbol{y},\boldsymbol{h})\in\mathcal{Y}\times\mathcal{H}}{\operatorname{argmax}} \; p(\boldsymbol{y}, \boldsymbol{h}|\boldsymbol{x}) = \underset{(\boldsymbol{y},\boldsymbol{h})\in\mathcal{Y}\times\mathcal{H}}{\operatorname{argmax}} \; \theta^\top \phi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h}).$$

This gives the optimal prediction of the $(\boldsymbol{y}, \boldsymbol{h})$-pair, and one obtains a prediction on $\boldsymbol{y}$ by simply discarding the $\boldsymbol{h}$ component. Unfortunately, the optimal prediction for $(y, h)$ jointly does not necessarily give an optimal prediction on $y$; instead, it may introduce strong biases even for simple cases (e.g., see Example 1 in Liu and Ihler [2013]). Intuitively, the joint MAP prediction is "overly optimistic", since it deterministically assigns the hidden variables to their most likely states; this approach is not robust to the inherent uncertainty in $h$, which may cause problems if that uncertainty is significant.

To address this issue, we use a marginal MAP predictor,

$$\hat{\boldsymbol{y}}(\theta) = \underset{\boldsymbol{y}\in\mathcal{Y}}{\operatorname{argmax}} \sum_{\boldsymbol{h}} p(\boldsymbol{y}, \boldsymbol{h}|\boldsymbol{x}; \theta) = \underset{\boldsymbol{y}\in\mathcal{Y}}{\operatorname{argmax}} \log \sum_{\boldsymbol{h}} \exp\, [\theta^\top \phi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h})], \tag{4.2}$$

which explicitly takes into account the uncertainty of the hidden variables. It should be noted that $\hat{\boldsymbol{y}}(\theta)$ is in fact the Bayes optimal prediction of $\boldsymbol{y}$, measured by zero-one loss. The main contribution of this work is to introduce a novel structured SVM-based method for learning the models with marginal MAP predictor, which significantly improves over previous methods.

## 4.4 Marginal Structured SVM

In this section we derive our main method, the marginal structured SVM (MSSVM), which minimizes an upper bound of the empirical risk function. Assume we have a set of training

instances $S = \{(\boldsymbol{x}^n, \boldsymbol{y}^n)\}_{n=1}^N \in (\mathcal{X} \times \mathcal{Y})^N$. The risk is measured by an user-specified empirical loss function $\Delta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n)$, which quantifies the difference between an estimator $\hat{\boldsymbol{y}}^n$ and the correct output $\boldsymbol{y}^n$. It is usually difficult to exactly minimize the loss function because it is typically non-convex and discontinuous with $\theta$ (e.g., Hamming loss). Instead, one adopts surrogate upper bounds to overcome this difficulty.

Assume $\hat{\boldsymbol{y}}^n(\theta)$ is the marginal MAP prediction on instance $\boldsymbol{x}^n$ as defined in (4.2). We upper bound the empirical loss function $\Delta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n(\theta))$ as follows,

$$\begin{aligned}\Delta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n(\theta)) &\leq \Delta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n(\theta)) + \log \sum_{\boldsymbol{h}} \exp\left[\theta^\top \phi(\boldsymbol{x}^n, \hat{\boldsymbol{y}}^n(\theta)), \boldsymbol{h})\right] - \log \sum_{\boldsymbol{h}} \exp[\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{h})] \\ &\leq \max_{\boldsymbol{y}} \left\{\Delta(\boldsymbol{y}^n, \boldsymbol{y}) + \log \sum_{\boldsymbol{h}} \exp\ [\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})]\right\} - \log \sum_{\boldsymbol{h}} \exp\ \left[\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{h})\right],\end{aligned}$$

where the first inequality holds because $\hat{\boldsymbol{y}}^n(\theta)$ is the marginal MAP prediction (4.2), and the second because it jointly maximizes the two terms.

Minimizing this upper bound over the training set with a $L_2$ regularization, we obtain the following objective function for our marginal structured SVM,

$$\frac{1}{2}\|\theta\|^2 + C \sum_{n=1}^N \left\{ \max_{\boldsymbol{y}} \left\{\Delta(\boldsymbol{y}^n, \boldsymbol{y}) + \log \sum_{\boldsymbol{h}} \exp\left[\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})\right]\right\} - \log \sum_{\boldsymbol{h}} \exp\left[\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{h})\right] \right\}. \tag{4.3}$$

Note that the first part of the objective (except the regularization term) requires a loss-augmented *marginal MAP* inference, which marginalizes the hidden variables $\boldsymbol{h}$ and then optimizes over the output variables $\boldsymbol{y}$, while the second part only requires a marginalization over the hidden variables. Both these terms and their gradients are intractable to compute on loopy graphical models, but can be efficiently approximated by mixed-product belief propagation [Liu and Ihler, 2013] and sum-product belief propagation [Wainwright and Jordan, 2008], respectively. We will discuss training algorithms for optimizing this objective in Sec-

tion 4.6. Similar to the constraint form of the structured SVM in Eq. (2.49), the constraint form of our MSSVM (4.3) is,

$$\min_{\theta,\{\xi_n\geq 0\}} \quad \frac{1}{2}\|\theta\|^2 + C\sum_{n=1}^{N}\xi_n, \tag{4.4}$$
$$\text{s.t.} \quad \forall n,\ \forall \boldsymbol{y}\in\mathcal{Y},\ \log\sum_{\boldsymbol{h}}\exp\left[\theta^\top\phi(\boldsymbol{x}^n,\boldsymbol{y}^n,\boldsymbol{h})\right] - \log\sum_{\boldsymbol{h}}\exp\left[\theta^\top\phi(\boldsymbol{x}^n,\boldsymbol{y},\boldsymbol{h})\right] \geq \Delta(\boldsymbol{y}^n,\boldsymbol{y}) - \xi_n,$$

where $\{\xi_n\}_{n=1}^N$ are the slack variables. One can show that the optimal $\{\xi_n^*\}_{n=1}^N$ satisfy,

$$\xi_n^* = \max_{\boldsymbol{y}}\left\{\Delta(\boldsymbol{y}^n,\boldsymbol{y}) + \log\sum_{\boldsymbol{h}}\exp\left[\theta^\top\phi(\boldsymbol{x}^n,\boldsymbol{y},\boldsymbol{h})\right]\right\} - \log\sum_{\boldsymbol{h}}\exp\left[\theta^\top\phi(\boldsymbol{x}^n,\boldsymbol{y}^n,\boldsymbol{h})\right],$$

which gives the same objective value as the the unconstrained form (4.3). The cutting plane training algorithm [Joachims et al., 2009] is applicable for this constraint optimization. It actively maintains a working set of constraints by adding the most violated constraint in (4.4) at each iteration. However, one need to minimize a quadratic objective with nonlinear constraints, which is more challenging than the standard SSVM situation. In this work, we focus on the unconstrained form (4.3) and its related training algorithms.

## 4.5 A Unified Framework

In this section, we compare our framework with a spectrum of existing methods, and introduce a more general framework that includes all these methods as special cases. To start, note that the objective function of the LSSVM [Yu and Joachims, 2009a] is

$$\frac{1}{2}\|\theta\|^2 + C\sum_{n=1}^{N}\left\{\max_{\boldsymbol{y}}\max_{\boldsymbol{h}}\left\{\Delta(\boldsymbol{y}^n,\boldsymbol{y}) + \theta^\top\phi(\boldsymbol{x}^n,\boldsymbol{y},\boldsymbol{h})\right\} - \max_{\boldsymbol{h}}\left[\theta^\top\phi(\boldsymbol{x}^n,\boldsymbol{y}^n,\boldsymbol{h})\right]\right\}. \tag{4.5}$$

Our objective in (4.3) is similar to (4.5), except replacing the *max* operator of $h$ with the log-sum-exp function, the so called *soft-max* operator. One may introduce a "temperature"

Table 4.1: Model comparisons within our unified framework.

| Model | $\epsilon_h \to 0^+ (\max_{\boldsymbol{h}})$ | $\epsilon_h = 1$ $(\sum_{\boldsymbol{h}})$ |
|---|---|---|
| $\epsilon_y \to 0^+$ $(\max_{\boldsymbol{y}})$ | LSSVM | MSSVM |
| $\epsilon_y = 1$ $(\sum_{\boldsymbol{y}})$ | MLLR | HCRF |
| $\epsilon_y = \epsilon_h \in (0, 1)$ | $\epsilon$-extension model | |

parameter that smooths between *max* and *soft-max*, which motivates a more general objective function that includes MSSVM, LSSVM and other previous methods as special cases,

$$\frac{1}{2}\|\theta\|^2 + C\sum_{n=1}^{N}\left\{\epsilon_y \log \sum_{\boldsymbol{y}} \exp\left[\frac{1}{\epsilon_y}\left(\Delta(\boldsymbol{y}^n, \boldsymbol{y}) + \epsilon_h \log \sum_{\boldsymbol{h}} \exp\left(\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\right)\right)\right]\right.$$
$$\left. - \epsilon_h \log \sum_{\boldsymbol{h}} \exp\left(\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{h})}{\epsilon_h}\right)\right\}, \tag{4.6}$$

where $\epsilon_y$ and $\epsilon_h$ are temperature parameters that control how much uncertainty we want account for in $\boldsymbol{y}$ and $\boldsymbol{h}$, respectively. Similar temperature-based approaches have been used both in structured prediction [Hazan and Urtasun, 2010, Schwing et al., 2012] and in other problems, such as semi-supervised learning [Samdani et al., 2012, Dhillon et al., 2012].

One can show (Lemma B.1.1 in Appendix B) that objective (4.6) is also an upper bound of the empirical loss function $\Delta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n_{\epsilon_h}(\theta))$ over the training set, where the prediction $\hat{\boldsymbol{y}}^n_{\epsilon_h}(\theta)$ is decoded by "annealed" marginal MAP,

$$\hat{\boldsymbol{y}}^n_{\epsilon_h}(\theta) = \operatorname*{argmax}_{\boldsymbol{y}} \ \epsilon_h \log \sum_{\boldsymbol{h}} \exp\left[\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\right].$$

This framework includes a number of existing methods as special cases. It reduces to our MSSVM in (4.3) if $\epsilon_y \to 0^+$ and $\epsilon_h = 1$, and LSSVM in (4.5) if $\epsilon_y \to 0^+$ and $\epsilon_h \to 0^+$. If we set $\epsilon_y = \epsilon_h = 1$, we obtain the loss-augmented likelihood objective in Volkovs et al. [2011], and it further reduces to the standard likelihood objective of HCRFs if we assume $\Delta(\boldsymbol{y}^n, \boldsymbol{y}) \equiv 0$. Our framework also generalizes the $\epsilon$-extension model by Schwing et al. [2012], which corresponds to the restriction that $\epsilon_y = \epsilon_h$. Most recently, Xu et al. [2016]

proposes the multinomial latent logistic regression (MLLR) method, which corresponds to $\epsilon_y = 1$ and $\epsilon_h \to 0^+$. See Table 4.1 for a summarization of these model comparisons. In the sequel, we provide some general insights on selecting among these different models through our empirical evaluations.

It should be noted, for all above models, the inference routine for predictions should be matched with the inference routine used in learning, which means we use mixed-product BP, max-product BP and sum-product BP for predictions with MSSVM, LSSVMs and HCRFs, respectively.

## 4.6 Training Algorithms

In this section, we introduce two optimization algorithms for minimizing the objective function in (4.3): a sub-gradient descent (SGD) algorithm, and a concave-convex procedure (CCCP). An empirical comparison of these two algorithms is given in the experiments of Section 4.7.

### 4.6.1 Sub-gradient Descent

According to Danskin's theorem [See Proposition B.25 in Bertsekas, 1999], the sub-gradient of the MSSVM objective (4.3) is:

$$\nabla_\theta M = \theta + C\sum_{n=1}^{N} \mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{x}^n,\hat{\boldsymbol{y}}^n)}[\phi(\boldsymbol{x}^n,\hat{\boldsymbol{y}}^n,\boldsymbol{h})] - C\sum_{n=1}^{N} \mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{x}^n,\boldsymbol{y}^n)}[\phi(\boldsymbol{x}^n,\boldsymbol{y}^n,\boldsymbol{h})], \tag{4.7}$$

$$\text{where } \hat{\boldsymbol{y}}^n = \operatorname*{argmax}_{\boldsymbol{y}\in\mathcal{Y}} \left\{ \Delta(\boldsymbol{y}^n,\boldsymbol{y}) + \log \sum_{\boldsymbol{h}} \exp\left[\theta^\top \phi(\boldsymbol{x}^n,\boldsymbol{y},\boldsymbol{h})\right] \right\} \tag{4.8}$$

is the loss-augmented marginal MAP prediction, which can be approximated via mixed-product belief propagation as described in Liu and Ihler [2013]. The $\mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{x}^n,\hat{\boldsymbol{y}}^n)}$ and $\mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{x}^n,\boldsymbol{y}^n)}$

**Algorithm 4.1** Sub-gradient Descent for MSSVM

**Input:** number of iterations $T$, learning rate $\eta$
**Output:** the learned weight vector $\theta^*$
$\theta = 0$;
**for** $t = 1$ **to** $T$ **do**
  $\nabla_\theta = 0$;
  **for** $n = 1$ **to** $N$ **do**
    1. Calculate $\phi_m = \mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{x}^n, \hat{\boldsymbol{y}}^n)}[\phi(\boldsymbol{x}^n, \hat{\boldsymbol{y}}^n, \boldsymbol{h})]$ by mixed-product BP;
    2. Calculate $\phi_s = \mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{x}^n, \boldsymbol{y}^n)}[\phi(\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{h})]$ by sum-product BP;
    3. $\nabla_\theta \leftarrow \nabla_\theta + C(\phi_m - \phi_s)$;
  **end for**
  $\theta \leftarrow (1 - \eta)\theta - \eta\nabla_\theta$;
**end for**
$\theta^* \leftarrow \theta$;

denote the expectation over the distributions $p(\boldsymbol{h}|\boldsymbol{x}^n, \hat{\boldsymbol{y}}^n)$ and $p(\boldsymbol{h}|\boldsymbol{x}^n, \boldsymbol{y}^n)$, respectively. Both expectations can similarly be approximated using the marginal probabilities obtained from belief propagation. See Algorithm 4.1 for details of the sub-gradient descent (SGD) algorithm for MSSVM. Furthermore, one can show (Lemma B.1.2 in Appendix B) that the gradient of the unified framework (4.6) is

$$\nabla_\theta U = \theta + C\sum_{n=1}^{N} \mathbb{E}_{p^{(\epsilon_y,\epsilon_h)}(\boldsymbol{y},\boldsymbol{h}|\boldsymbol{x}^n)}[\phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})] - C\sum_{n=1}^{N} \mathbb{E}_{p^{\epsilon_h}(\boldsymbol{h}|\boldsymbol{x}^n, \boldsymbol{y}^n)}[\phi(\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{h})]. \tag{4.9}$$

where the corresponding temperature controlled distributions are defined as,

$$p^{\epsilon_h}(\boldsymbol{h}|\boldsymbol{x}^n, \boldsymbol{y}) \propto \exp\Big[\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\Big],$$
$$p^{(\epsilon_y,\epsilon_h)}(\boldsymbol{y}|\boldsymbol{x}^n) \propto \exp\Big\{\frac{1}{\epsilon_y}\big[\Delta(\boldsymbol{y}, \boldsymbol{y}^n) + \epsilon_h \log\sum_{\boldsymbol{h}} \exp\big(\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\big)\big]\Big\},$$
$$p^{(\epsilon_y,\epsilon_h)}(\boldsymbol{y}, \boldsymbol{h}|\boldsymbol{x}^n) = p^{\epsilon_h}(\boldsymbol{h}|\boldsymbol{x}^n, \boldsymbol{y}) \cdot p^{(\epsilon_y,\epsilon_h)}(\boldsymbol{y}|\boldsymbol{x}^n).$$

Exactly as in Table 4.1, this reduces to the sub-gradient of MSSVM (4.7) if $\epsilon_y \to 0^+$ and $\epsilon_h = 1$, the sub-gradient of LSSVM if $\epsilon_y \to 0^+$ and $\epsilon_h \to 0^+$, and the gradient of HCRF if $\epsilon_y = 1$, $\epsilon_h = 1$ and $\Delta(\boldsymbol{y}, \boldsymbol{y}^n) \equiv 0$. One can simply substitute these (sub-)gradients into

Algorithm 4.1 to obtain the corresponding training algorithms for LSSVM and HCRF. In those cases, max-product BP and sum-product BP can be used to approximate the inference operations instead.

### 4.6.2 CCCP Training Algorithm

The concave-convex procedure (CCCP) [Yuille and Rangarajan, 2003] is a general non-convex optimization algorithm with wide application in machine learning. It is based on the idea of rewriting the non-convex objective function into a sum of a convex function and a concave function (or equivalently a difference of two convex functions), and transforming the non-convex optimization problem into a sequence of convex sub-problems by linearizing the concave part. In learning with latent variable models, the log-likelihood functions are usually the difference of two convex functions. As a result, CCCP generalizes the expectation–maximization (EM) algorithm [Dempster et al., 1977], in which the E-step is analogous to linearization step in CCCP, and the M-step corresponds to solving the convex sub-problem in CCCP.

CCCP provides a straightforward solution for our problem, since the objective functions of all the methods we have discussed – in (4.3), (4.5) and (4.6) – are naturally differences of two convex functions. For example, the MSSVM objective in (4.3) can be written as,

$$
\begin{aligned}
f(\theta) &= f^{+}(\theta) - f^{-}(\theta), \text{ where} \\
f^{+}(\theta) &= \frac{1}{2}\|\theta\|^2 + C\sum_{n=1}^{N} \max_{\boldsymbol{y}} \Big\{ \Delta(\boldsymbol{y}^n, \boldsymbol{y}) + \log \sum_{\boldsymbol{h}} \exp\left[\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})\right] \Big\}, \\
f^{-}(\theta) &= C\sum_{n=1}^{N} \log \sum_{\boldsymbol{h}} \exp\left[\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{h})\right].
\end{aligned}
$$

Denoting the parameter vector at iteration $t$ by $\theta^t$, the CCCP algorithm updates to new

**Algorithm 4.2** CCCP Training of MSSVM

**Input:** number of outer iterations $T$, learning rate $\eta$, tolerance $\epsilon$ for inner loops
**Output:** the learned weight vector $\theta^*$
$\theta = 0$;
**for** $t = 1$ **to** $T$ **do**
  $u = 0$;
  \\ linearization step:
  **for** $n = 1$ **to** $N$ **do**
    1. Calculate $\phi_s = \mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{x}^n, \boldsymbol{y}^n)}[\phi(\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{h})]$ by sum-product BP;
    2. $u = u + \phi_s$;
  **end for**
  \\ minimizing convex surrogate by gradient descent:
  **repeat**
    $\nabla_\theta = 0$;
    **for** $n = 1$ **to** $N$ **do**
      1. Calculate $\phi_m = \mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{x}^n, \hat{\boldsymbol{y}}^n)}[\phi(\boldsymbol{x}^n, \hat{\boldsymbol{y}}^n, \boldsymbol{h})]$ by mixed-product BP;
      2. $\nabla_\theta \leftarrow \nabla_\theta + C\phi_m$;
    **end for**
    $\nabla_\theta = \nabla_\theta - Cu$;
    $\theta \leftarrow (1 - \eta)\theta - \eta\nabla_\theta$;
  **until** $||\nabla_\theta|| \leq \epsilon$
**end for**
$\theta^* \leftarrow \theta$;

parameters $\theta^{t+1}$ by minimizing a convex surrogate function where $f^-(\theta)$ is linearized:

$$\theta^{t+1} \leftarrow \operatorname*{argmin}_{\theta} \left\{ f^+(\theta) - \theta^\top \nabla f^-(\theta^t) \right\}, \quad \text{where } \nabla f^-(\theta^t) = C \sum_{n=1}^{N} \mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{x}^n, \boldsymbol{y}^n)}[\phi(\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{h})]$$

is the gradient of $f^-(\theta)$ at $\theta_t$ and its expectation can be evaluated (approximately) by belief propagation. The convex optimization can be solved by gradient descent. See Algorithm 4.2 for more details of CCCP for the MSSVM.

## 4.7 Experiments

In this section, we compare our MSSVM with other state-of-the-art methods on both simulated and real-world datasets. We demonstrate that the MSSVM significantly outperforms

the LSSVM, max-margin min-entropy (M3E) model [Miller et al., 2012], and loss-based learning by modeling latent variable(ModLat) [Kumar et al., 2012], especially when the uncertainty over hidden variables is high. Our method also consistently outperforms HCRFs in all experiments, especially with a small training sample size.

### 4.7.1 Simulated Data

We simulate both training and testing data from a pairwise Markov random field (MRF) over graph $G = (V, E)$ with discrete random variables taking values in $\{0, 1, 2, 3\}^n$, given by,

$$p(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{h} \mid \theta) \propto \exp\Big[ \sum_{i \in V_x} \theta_i^\top \phi(x_i) + \sum_{j \in V_y} \theta_j^\top \phi(y_j) + \sum_{k \in V_h} \theta_k^\top \phi(h_k) + \sum_{(i,j) \in E_{xy}} \theta_{(i,j)}^\top \phi(x_i, y_j) + \sum_{(i,k) \in E_{xh}} \theta_{(i,k)}^\top \phi(x_i, h_k) + \sum_{(j,k) \in E_{yh}} \theta_{(j,k)}^\top \phi(y_j, h_k) \Big],$$

where the graph structure G is either a "hidden chain" (40 nodes) or a 2D grid (size $6 \times 6 \times 2 = 72$ nodes), as illustrated in Figure 4.1. The shaded nodes denote hidden variables $h$, while the unshaded nodes are the output variables $y$ and nodes with hatching are the inputs $x$. The log-linear weights $\theta$ are randomly generated from normal distributions. The singleton parameters $\theta_i$, $\theta_j$ and $\theta_k$ are drawn from $\mathcal{N}(0,\ \sigma_x^2 \cdot I)$, $\mathcal{N}(0,\ \sigma_y^2 \cdot I)$ and $\mathcal{N}(0, \sigma_h^2 \cdot I)$, respectively, corresponding to indicator vectors $\phi(x_i)$, $\phi(y_j)$ and $\phi(h_k)$. The pairwise parameters $\theta_{(j,k)}[y_j = s, h_k = t]$, $\theta_{(i,j)}[x_i = r, y_j = s]$ and $\theta_{(i,k)}[x_i = r, h_k = t]$ are drawn from $\mathcal{N}(0, \sigma_{yh}^2)$, $\mathcal{N}(0, \sigma_{xy}^2)$ and $\mathcal{N}(0, \sigma_{xh}^2)$, respectively, corresponding to indicators $\phi(y_j = s,\ h_k = t)$, $\phi(x_i = r,\ y_j = s)$ and $\phi(x_i = r,\ h_k = t)$. Note that the variance parameters $\sigma_h$ and $\sigma_{yh}$ control the degree of uncertainty in the hidden variables and their importance for estimating the output variables $y$: the uncertainty of $h$ is high for small values of $\sigma_h$, and the correlation between $h$ and $y$ is high when $\sigma_{yh}$ is large.

We sample 20 training instances and 100 test instances from both the 40-node hidden chain

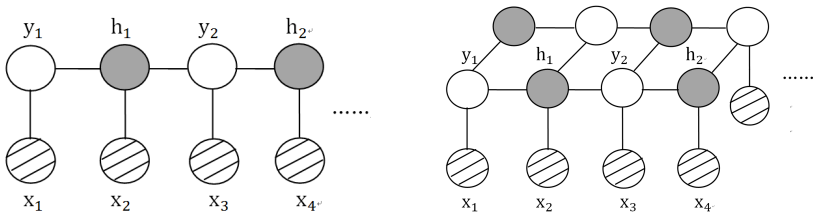



Figure 4.1: (a) The hidden chain and (b) 2D grid model used in our simulation experiments. The shaded nodes denote hidden variables $h$, while the unshaded nodes are the output variables $y$ and nodes with hatching are the inputs $x$.

Table 4.2: Average accuracy (%) of MSSVM, LSSVM, HCRFs using SGD and CCCP when the data are simulated from 40-node hidden chain and $6 \times 6$ 2D-grid graph as shown in Figure 4.1. The results are averaged over 20 random trials.

| Hidden Chain | MSSVM | LSSVM | HCRFs |
|---|---|---|---|
| SGD | **69.20** | 66.87 | 68.75 |
| CCCP | **69.63** | 67.91 | 69.03 |

| 2D-grid graph | MSSVM | LSSVM | HCRFs |
|---|---|---|---|
| SGD | **74.12** | 71.96 | 73.51 |
| CCCP | **74.08** | 73.38 | 73.62 |

MRF and $6 \times 6$ 2D grid MRF as as shown in Figure 4.1. We set $\sigma_x = \sigma_y = \sigma_h = 0.1$, $\sigma_{yh} = \sigma_{yx} = \sigma_{hx} = 2$. Then, we train our MSSVM, LSSVM and HCRF models using both SGD and CCCP. Hamming loss is used in both training and evaluation. In our experiments, we set the regularization weight $C = 1$, because we find that it will give good enough results on the simulated data. See Table 4.2 for the results across different algorithms. The results are averaged over 20 random trials. We can see that our MSSVM always achieves the highest accuracy when using either training algorithm. It is worth noting that LSSVM obtains a significantly better result using CCCP than SGD; this is mainly due to SGD's difficulty converging on the piecewise linear objective of LSSVM.

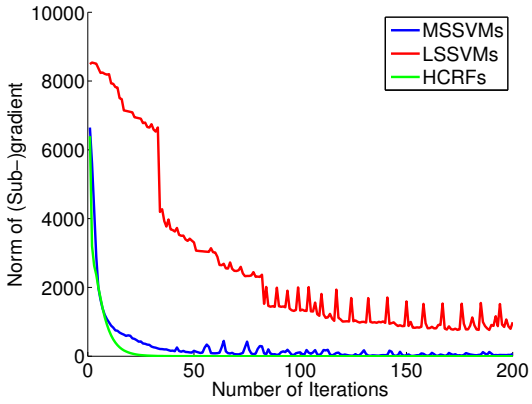



Figure 4.2: Convergence behaviours of (sub-)gradient descent on MSSVMs, LSSVMs and HCRFs. Our MSSVM has smoother objective function and faster convergence than LSSVM.

**Empirical Convergence of SGD and CCCP.** Using sub-gradient descent with learning rate $\eta_M = 0.02$, we found that for our MSSVM, training error converged quickly (within 50 iterations). However, sub-gradient descent on the LSSVM would only converge using a much smaller learning rate ($\eta_L = 0.001$), and converged more slowly (usually after 250 iterations). This effect is mainly because the LSSVM hard-max makes the objective function nonsmooth, causing sub-gradient descent to be slow to converge. On the other hand, gradient descent on HCRFs converges more easily and quickly than either MSSVM or LSSVM, because its objective function is smoother. Figure 4.2 shows the oscillation during the iteration of (sub-)gradient descent for each model, and empirically illustrates the convergence process.

We also observe CCCP converging faster than SGD (using smaller number of inference steps), especially for LSSVM, since CCCP transforms the complex piecewise linear objective into a sequence of easier convex sub-problems. In our empirical study, CCCP always converged well even using approximate inference and non-convex objectives.[2] To provide a fair comparison, all methods are trained using the CCCP algorithm in the sequel.

[2]However, it is challenging to provide rigorous convergence guarantees for the non-convex & intractable setting, and not really the focus of this work.

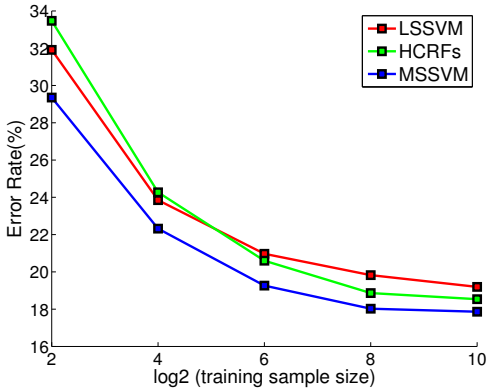



Figure 4.3: The error rate of MSSVM, LSSVM and HCRFs as the training sample size increases. Results are averaged over 5 random trials.

**Training Sample Size.** We compared the influence of sample size on each method by varying the training size from $2^2$ to $2^{10}$ (with a testing size of 500). The data are all simulated from a MRF on the 20-node hidden chain shown in Figure 4.1(a). We set $\sigma_x = \sigma_y = \sigma_h = 0.1$ and $\sigma_{yh} = \sigma_{yx} = \sigma_{hx} = 2$ as before. Results are averaged over 5 random trials and are shown in Figure 4.3. We found that our MSSVM always considerably outperforms LSSVM, and largely outperforms HCRFs when the training sample sizes are small. HCRFs perform worse than LSSVM for few training data, but outperform LSSVM as the training sample increases.

Our experiment shows that MSSVM consistently outperforms HCRFs even with reasonably large training sets on a relatively simple toy model. Although the maximum likelihood estimator (as used in HCRFs) is generally considered asymptotically optimal if the model assumptions are correct, this assumes a sufficiently large training size, which may be difficult to achieve in practice. Given enough data (and the correct model), the HCRF should thus eventually improve, but this seems unrealistic in practice since most applications are likely to exhibit high dimensional parameters and relatively few training instances.

**Likelihood vs. Prediction Accuracy** However, it is worth noting that the HCRF model always achieves higher test likelihood than the MSSVM and LSSVM on our simulated data set. As an example, Figure 4.4 shows the test log-likelihood across the different methods

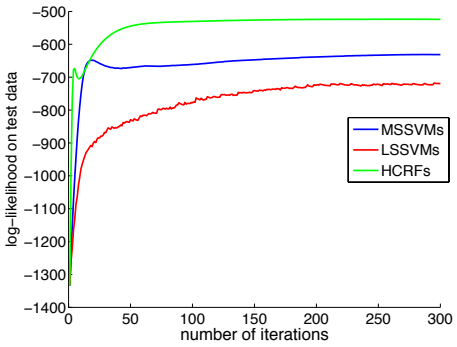



Figure 4.4: The test log-likelihood of MSSVM, LSSVM and HCRF using SGD across iterations.

when 20 training and 100 test instances are sampled from 40-node hidden chain MRF. This should not be surprising, since the HCRF model directly optimizes the likelihood objective, and (in this case) the model class being optimized is correct (i.e., the data were drawn from a true model with the same structure). However, as we showed in Table 4.2, higher likelihood does not necessarily imply that the HCRF will have better predictions on the target variables. As was illustrated in previous part, explicitly minimizing the empirical loss can lead to better predictions in situations with high dimensional model parameters and relatively few training instances.

**Uncertainty of Hidden Variables.** We investigate the influence of uncertainty in the hidden variables for each method by adjusting the noise level $\sigma_h$, which controls the uncertainty of the hidden variables. Small values of $\sigma_h$ correspond to high uncertainty in hidden variables. We draw 20 training samples and 100 test samples from a MRF on a 40-node hidden chain shown in Figure 4.1(a), with fixed $\sigma_x = \sigma_y = 0.1$ and $\sigma_{yh} = \sigma_{yx} = \sigma_{hx} = 2$. For comparison, we also evaluate the performance of M3E [Miller et al., 2012] and ModLat [Kumar et al., 2012]. In accordance with our default setting $C = 1$, we use the default hyper-parameters in their package. It should be noted, MSSVM, LSSVM and HCRF have fewer hyper-parameters than M3E and ModLat, and may tend to be less sensitive in practice.

Results are averaged over 20 random trials and are shown in Table 4.3. We find that

Table 4.3: The accuracy (%) of MSSVM, LSSVM, HCRFs, M3E and ModLat under different $\sigma_h$, which governs the level of uncertainty in the hidden variables. Small values of $\sigma_h$ correspond to high uncertainty in hidden variables. Results are averaged over 20 random trials.

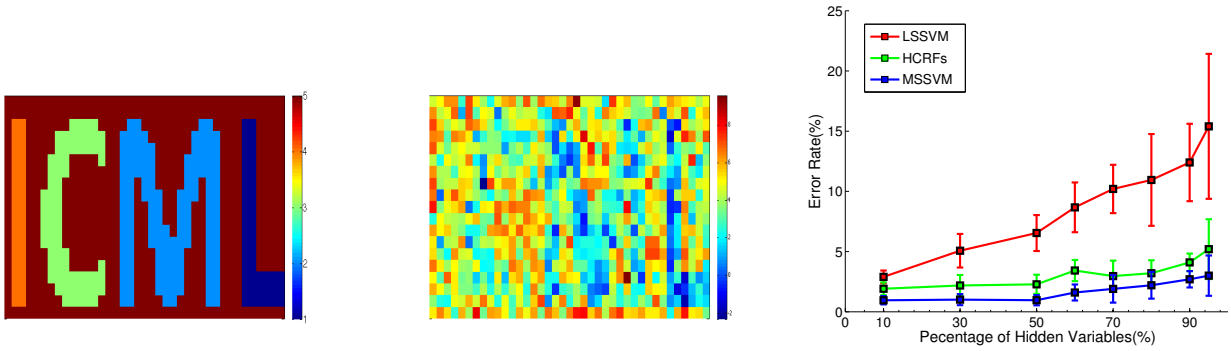
| $\sigma_h$ | MSSVM | LSSVM | HCRFs | M3E | ModLat |
|---|---|---|---|---|---|
| 10 | 79.30 | **79.46** | 78.68 | 79.04 | 77.16 |
| 1 | 70.00 | **70.07** | 69.88 | 68.53 | 67.91 |
| 0.5 | **67.24** | 65.98 | 66.66 | 66.05 | 65.15 |
| 0.1 | **69.63** | 67.91 | 69.03 | 65.19 | 67.96 |
| 0.01 | **73.88** | 71.38 | 72.58 | 67.21 | 71.52 |
| 1e-3 | **72.08** | 69.24 | 70.88 | 65.48 | 66.54 |
| Avg. | **72.02** | 70.67 | 71.28 | 68.58 | 69.37 |

our MSSVM is competitive with LSSVM and M3E when the uncertainty in the hidden variables is low, and becomes significantly better than them as the uncertainty increases. Because LSSVM uses the joint MAP, it does not take into account this uncertainty. On the other hand, M3E explicitly tries to minimize this uncertainty, which can also mislead the prediction. Our MSSVM consistently outperforms HCRFs for moderate training sample sizes. Because current implementations of M3E and ModLat do not provide approximate inference algorithms on general MRFs, we only provide their results on chain models.

### 4.7.2 Image Segmentation

In this section, we evaluate our MSSVM method on the task of segmenting weakly labeled images. Our settings are modeled the experiments in Schwing et al. [2012]. We assume a ground truth image of 20×40 pixels as shown in Figure 4.5 (a), where each pixel $i$ has a label $y_i$ taking values in $\{1, \cdots, 5\}$. The observed image $x$ is obtained by adding Gaussian noise, $\mathcal{N}(0, 5)$, on the ground truth image as Figure 4.5 (b).

We use the 2D-grid model as in Figure 4.1 (b), with local features $\phi(y_i, x_i) = e_{y_i} \otimes x_i$ and pairwise features $\phi(y_i, y_j) = e_{y_i} \otimes e_{y_j} \in \mathbb{R}^{5\times 5}$ as defined in Nowozin and Lampert [2011], where




(a) (b) (c)

Figure 4.5: (a) The ground truth image. (b) An example of an observed noisy image. (c) The performance of each algorithm as the percentage of missing labels varies from 10% to 95%. Results are averaged over 5 random trials, each using 10 training instances and 10 test instances.

$e_{y_i}$ is the unit normal vector with entry one on dimension $y_i$ and $\otimes$ is the outer product. The set of missing labels (hidden variables) are determined at random, in proportions ranging from 10% to 95%. The performance of MSSVM, LSSVM, and HCRFs are evaluated using the CCCP algorithm. Figure 4.5 (c) lists the performance of each method as the percentage of missing labels is increased. Results are averaged over 5 random trials, each using 10 training instances and 10 test instances. We can see that the performance of LSSVM degrades significantly as the number of hidden variables grows. Most notably, MSSVM is consistently the best method across all settings. This can be explained by the fact that the MSSVM combines both the max-margin property and the improved robustness given by properly accounting for uncertainty in the hidden labels.

### 4.7.3 Object Categorization

Finally, we evaluate our MSSVM method on the task of object categorization using partially labeled images. We use the Microsoft Research Cambridge data set [Winn et al., 2005], consisting of 240 images with 213×320 pixels and their partial pixel-level labelings. The missing labels may correspond to ambiguous regions, undefined categories or object boundaries, etc.

Figure 4.6: The example images and Pixel-wise labellings including building, grass, sky, tree and car from MSRC dataset. The black regions represent missing labels.

Table 4.4: Average patch level accuracy (%) of MSSVM, LSSVM, HCRFs for MSRC data by 2-fold cross validation.

| MSRC Data | MSSVM | LSSVM | HCRFs |
|---|---|---|---|
| Building | **72.4** | 70.7 | 71.7 |
| Grass | **89.7** | 88.9 | 88.3 |
| Sky | **88.3** | 85.6 | 88.2 |
| Tree | **71.9** | 71.0 | 70.1 |
| Car | **70.8** | 69.4 | 70.2 |

See Figure 4.6 for illustration. Modeled on the approach outlined in Verbeek and Triggs [2007], we use $20 \times 20$ pixel patches with centers at 10 pixel intervals and treat each patch as a node in our model. This results in a $20 \times 31$ grid model as in Figure 4.1 (b). The local features of each patch are encoded using texture and color descriptors. For texture, we compute the 128-dimensional SIFT descriptor of the patch and vector quantize it into a 500-word codebook, learned by k-means clustering of all patches in the entire dataset. For color, we take 48-dimensional RGB color histogram for each patch. In our experiment, we select the 5 most frequent categories in the dataset and use 2-fold cross validation for testing.

Table 4.4 shows the accuracies of each method across the various categories. Again, we find that MSSVM consistently outperforms other methods across all categories, which can be explained by both the superiority of SSVM-based methods for moderate sample size and the improved robustness by maintaining the uncertainty over the missing labels in the learning procedure.

## 4.8 Conclusion

We proposed a novel structured SVM method for structured prediction with hidden variables. We demonstrate that our MSSVM consistently outperforms state-of-the-art methods in both simulated and real-world datasets, especially when the uncertainty of hidden variables is large. Compared to the popular LSSVM, the objective function of our MSSVM is easier to optimize due to the smoothness of its objective function. We also provide a unified framework which includes our method as well as a spectrum of previous methods as special cases.

Chapter 5

# Learning Infinite RBMs with Frank-Wolfe

Latent variable models (LVMs) can model the highly complex distribution of observable variables, and provide useful hidden representations for other end tasks, such as classification. Restricted Boltzmann machines (RBMs) are popular two-layer LVMs that use a layer of hidden units $\boldsymbol{h}$ to model the distribution of observable units $\boldsymbol{v}$ [Smolensky, 1986, Hinton, 2002b]. In practice, it is challenging to determine the size of hidden layer (i.e., the number of hidden units) before performing learning. In this chapter, we propose an infinite restricted Boltzmann machine by defining a distribution over the hidden layer, whose maximum likelihood estimation (MLE) corresponds to a constrained convex optimization. We apply the Frank-Wolfe algorithm [Frank and Wolfe, 1956], also known as conditional gradient, to solve the resulting optimization, which provides a solution that can be interpreted as inserting a hidden unit at each iteration, so that the optimization process takes the form of a sequence of finite models of increasing complexity. As a side benefit, this can be used to easily and efficiently identify an appropriate number of hidden units during the optimization. The resulting model can also be used as an initialization for typical state-of-the-art RBM training algorithms such as contrastive divergence, leading to models with consistently higher test likelihood than random initialization.

## 5.1 Introduction

Restricted Boltzmann machines (RBMs) have been widely applied to capture the complex distributions of observable data in numerous application domains, including image modeling [Krizhevsky et al., 2010], human motion capture [Taylor et al., 2006b] and collaborative filtering [Salakhutdinov et al., 2007b]. In addition, RBMs are also widely used as building blocks for state-of-the-art deep generative models, such as deep belief networks [Hinton et al., 2006b] and deep Boltzmann machines [Salakhutdinov and Hinton, 2009]. Due to the existence of partition function, the log-likelihood function of RBMs are generally intractable to calculate. In the literature, RBMs are usually learned using the contrastive divergence (CD) algorithm [Hinton, 2002b, Tieleman, 2008], which approximates the gradient of the log-partition function using a Gibbs sampler.

One important model selection problem when using a RBM is that we need to decide the size of the hidden layer (number of hidden units) before performing learning, and it can be challenging to decide what is the optimal size. One simple heuristic is to search the 'best" number of hidden units using cross validation or testing likelihood within a pre-defined candidate set. Unfortunately, this is extremely time consuming, which involves running a full training algorithm (e.g., contrastive divergence (CD) [Hinton, 2002b]) for each possible size, and thus we can only search over a relatively small set of sizes using this approach. In addition, because the log-likelihood of the RBM is highly non-convex, its performance is sensitive to the initialization of the learning algorithm. Although random initializations (to relatively small values) are routinely used in practice with algorithms like CD, it would be valuable to explore more robust algorithms that are less sensitive to the initialization, as well as smarter initialization strategies to obtain better results.

In this chapter, we propose a fast, greedy algorithm for training RBMs by inserting one hidden unit at each iteration. Our algorithm provides an efficient way to determine the size

of the hidden layer in an adaptive fashion, and can also be used as an initialization for a full CD-like learning algorithm. Our method is based on constructing a convex relaxation of the RBM that is parameterized by a distribution over the weights of the hidden units, for which the training problem can be framed as a convex functional optimization and solved using an efficient Frank-Wolfe algorithm [Frank and Wolfe, 1956, Jaggi, 2013] that effectively adds one hidden unit at each iteration by solving a relatively fast inner loop optimization.

## 5.2 Related Work

Our contributions connect to a number of different themes of existing work within machine learning and optimization. Here we give a brief discussion of prior related work.

There have been a number of works on convex relaxations of latent variable models in functional space, which are related to the classical gradient boosting method [Friedman, 2001]. In supervised learning, Bengio et al. [2005] propose a convex neural network in which the number of hidden units is unbounded and can be learned, and Bach [2014] analyzes the appealing theoretical properties of such a model. For clustering problems, several works on convex functional relaxation have also been proposed [e.g., Nowozin and Bakir, 2008, Bradley and Bagnell, 2009]. Other forms of convex relaxation have also been developed for two layer latent variable models [e.g., Aslan et al., 2013].

There has also been considerable work on extending directed/hierarchical models into "infinite" models such that the dimensionality of the latent space can be automatically inferred during learning. Most of these methods are Bayesian nonparametric models, and a brief overview can be found in Orbanz and Teh [2011]. A few directions have been explored for undirected models, particularly RBMs. Welling et al. [2002] propose a boosting algorithm in the feature space of the model; a new feature is added into the RBM at each boosting

iteration, instead of a new hidden unit. Nair and Hinton [2010] conceptually tie the weights of an infinite number of binary hidden units, and connect these sigmoid units with noisy rectified linear units (ReLUs). Recently, Côté and Larochelle [2015] extend an ordered RBM model with infinite number of hidden units, and Nalisnick and Ravi [2015] use the same technique for word embedding. The ordered RBM is sensitive to the ordering of its hidden units and can be viewed as an mixture of RBMs. In contrast, our model incorporates regular RBM, as a special case, and enables model selection for standard RBMs.

The Frank-Wolfe method [Frank and Wolfe, 1956], also known as conditional gradient, is a classical algorithm to solve constrained convex optimization. It has recently received much attention because it unifies a large number of sparse greedy methods [Jaggi, 2013], including boosting algorithms [e.g., Beygelzimer et al., 2015], learning with dual structured SVM [Lacoste-Julien et al., 2013] and marginal inference using MAP in graphical models [e.g., Belanger et al., 2013, Krishnan et al., 2015].

Verbeek et al. [2003] proposed a greedy learning algorithm for Gaussian mixture models, which inserts a new component at each step and resembles our algorithm in its procedure. As one benefit, it provides a better initialization for EM than random initialization. Likas et al. [2003] investigate greedy initialization in k-means clustering.

## 5.3 Background and Notations

As we reviewed in Section 2.1.4 of Chapter 2, a restricted Boltzmann machine (RBM) is an undirected graphical model that defines a joint distribution over the vectors of visible units $\boldsymbol{v} \in \{0,1\}^{|\boldsymbol{v}|\times 1}$ and hidden units $\boldsymbol{h} \in \{0,1\}^{|\boldsymbol{h}|\times 1}$,

$$p(\boldsymbol{v}, \boldsymbol{h} \mid \theta) = \frac{1}{Z(\theta)} \exp\left(\boldsymbol{v}^\top W \boldsymbol{h} + \boldsymbol{b}^\top \boldsymbol{v}\right); \quad Z(\theta) = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp\left(\boldsymbol{v}^\top W \boldsymbol{h} + \boldsymbol{b}^\top \boldsymbol{v}\right), \tag{5.1}$$

where $|\boldsymbol{v}|$ and $|\boldsymbol{h}|$ are the dimensions of $\boldsymbol{v}$ and $\boldsymbol{h}$ respectively, and $\theta \coloneqq \{W, \boldsymbol{b}\}$ are the model parameters including the pairwise interaction term $W \in \mathbb{R}^{|\boldsymbol{v}|\times|\boldsymbol{h}|}$ and the bias term $\boldsymbol{b} \in \mathbb{R}^{|\boldsymbol{v}|\times 1}$ for the visible units. In this chapter, we drop the bias term for the hidden units $\boldsymbol{h}$, since it simplifies our derivation and can be achieved by introducing a dummy visible unit whose value is always one. The partition function $Z(\theta)$ serves to normalize the probability to sum to one, and is typically intractable to calculate exactly.

According to the marginal distribution of RBM in Eq. (2.10), the marginal log-likelihood of the RBM is:

$$\log p(\boldsymbol{v} \mid \theta) = \sum_{i=1}^{|\boldsymbol{h}|} \log\left(1 + \exp(\boldsymbol{w}_i^\top \boldsymbol{v})\right) + \boldsymbol{b}^\top \boldsymbol{v} - \log Z(\theta), \tag{5.2}$$

where $\boldsymbol{w}_i \coloneqq W_{\bullet i}$ is the $i$-th column of $W$ and corresponds to the weights connected to the $i$-th hidden unit. Because each hidden unit $h_i$ takes values in $\{0, 1\}$, we get the *softplus* function $\log(1 + \exp(\boldsymbol{w}_i^\top \boldsymbol{v}))$ when we marginalize $h_i$. This form shows that the (marginal) free energy of the RBM is a sum of a linear term $\boldsymbol{b}^\top \boldsymbol{v}$ and a set of *softplus* functions with different weights $\boldsymbol{w}_i$; this provides a foundation for our development.

Given a dataset $\{\boldsymbol{v}^n\}_{n=1}^N$, the gradient of the log-likelihood for each data point $\boldsymbol{v}^n$ is

$$\frac{\partial \log p(\boldsymbol{v}^n|\theta)}{\partial W} = \mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{v}^n;\theta)}\left[\boldsymbol{v}^n \boldsymbol{h}^\top\right] - \mathbb{E}_{p(\boldsymbol{v},\boldsymbol{h}|\theta)}\left[\boldsymbol{v}\boldsymbol{h}^\top\right] = \boldsymbol{v}^n (\boldsymbol{\mu}^n)^\top - \mathbb{E}_{p(\boldsymbol{v},\boldsymbol{h}|\theta)}\left[\boldsymbol{v}\boldsymbol{h}^\top\right], \tag{5.3}$$

where $\boldsymbol{\mu}^n = \sigma(W^\top \boldsymbol{v}^n)$ and the logistic function $\sigma(u) = 1/(1 + \exp(-u))$ is applied in an element-wise manner. The positive part of the gradient can be calculated exactly, since the conditional distribution $p(\boldsymbol{h}|\boldsymbol{v}^n)$ is fully factorized. The negative part arises from the derivatives of the log-partition function and is intractable. Stochastic optimization algorithms, such as CD [Hinton, 2002b] and persistent CD [Tieleman, 2008], are popular methods to approximate the intractable expectation using Gibbs sampling.

## 5.4 RBM with Infinite Hidden Units

In this section, we first generalize the RBM model defined in Eq. (5.2) to a model with an infinite number of hidden units, which can also be viewed as a convex relaxation of the RBM in functional space. Then, we describe the learning algorithm.

### 5.4.1 Model Definition

Our general model is motivated by Eq. (5.2), in which the first term can be treated as an empirical average of the *softplus* function $\log(1+\exp(\boldsymbol{w}^\top \boldsymbol{v}))$ under an empirical distribution over the weights $\{\boldsymbol{w}_i\}$. To extend this, we define a general distribution $q(\boldsymbol{w})$ over the weight $\boldsymbol{w}$, and replace the empirical averaging with the expectation under $q(\boldsymbol{w})$; this gives the following generalization of an RBM with an infinite (possibly uncountable) number of hidden units,

$$\log p(\boldsymbol{v} \mid q, \vartheta) = \alpha \mathbb{E}_{q(\boldsymbol{w})}\big[\log(1+\exp(\boldsymbol{w}^\top \boldsymbol{v}))\big] + \boldsymbol{b}^\top \boldsymbol{v} - \log Z(q,\vartheta), \tag{5.4}$$
$$Z(q,\vartheta) = \sum_{\boldsymbol{v}} \exp\Big(\alpha \mathbb{E}_{q(\boldsymbol{w})}\big[\log(1+\exp(\boldsymbol{w}^\top \boldsymbol{v}))\big] + \boldsymbol{b}^\top \boldsymbol{v}\Big),$$

where $\vartheta \coloneqq \{\boldsymbol{b}, \alpha\}$ and $\alpha > 0$ is a temperature parameter which controls the "effective number" of hidden units in the model, and $\mathbb{E}_{q(\boldsymbol{w})}[f(\boldsymbol{w})] \coloneqq \int_{\boldsymbol{w}} q(\boldsymbol{w}) f(\boldsymbol{w}) d\boldsymbol{w}$. Note that $q(\boldsymbol{w})$ is assumed to be properly normalized, i.e., $\int_{\boldsymbol{w}} q(\boldsymbol{w}) d\boldsymbol{w} = 1$. Intuitively, (5.4) defines a semi-parametric model whose log probability is a sum of a linear bias term parameterized by $\boldsymbol{b}$, and a nonlinear term parameterized by the weight distribution $\boldsymbol{w}$ and $\alpha$ that controls the magnitude of the nonlinear term. This model can be regarded as a convex relaxation of the regular RBM, as shown in the following result.

**Proposition 5.4.1.** *The model in Eq.* (5.4) *includes the standard RBM* (5.2) *as special case by constraining* $q(\boldsymbol{w}) = \frac{1}{|\boldsymbol{h}|}\sum_{i=1}^{|\boldsymbol{h}|} \mathbb{1}(\boldsymbol{w} = \boldsymbol{w}_i)$ *and* $\alpha = |\boldsymbol{h}|$. *Moreover, the log-likelihood of the*

*model is concave w.r.t. the function $q(\boldsymbol{w})$, $\alpha$ and $\boldsymbol{b}$ respectively, and is jointly concave with $q(\boldsymbol{w})$ and $\boldsymbol{b}$.*

We should point out that the parameter $\alpha$ plays a special role in this model: we reduce to the standard RBM only when $\alpha$ equals the number $|\boldsymbol{h}|$ of particles in $q(\boldsymbol{w}) = \frac{1}{|\boldsymbol{h}|}\sum_{i=1}^{|\boldsymbol{h}|} \mathbb{1}(\boldsymbol{w} = \boldsymbol{w}_i)$, and would otherwise get a *fractional* RBM. The fractional RBM leads to a more challenging inference problem than a standard RBM, since the standard Gibbs sampler is no longer directly applicable. We discuss this point further in Section 5.4.3.

Given a dataset $\{\boldsymbol{v}^n\}_{n=1}^N$, we learn the parameters $q$ and $\vartheta$ using a penalized maximum likelihood estimator (MLE) that involves a convex functional optimization:

$$\underset{q\in\mathbb{M},\ \vartheta}{\operatorname{argmax}} \left\{ L(q,\vartheta) \equiv \frac{1}{N}\sum_{n=1}^{N} \log p(\boldsymbol{v}^n \mid q,\vartheta) - \frac{\lambda}{2}\mathbb{E}_{q(\boldsymbol{w})}[||\boldsymbol{w}||^2] \right\}, \tag{5.5}$$

where $\mathbb{M}$ is the set of valid distributions and we introduce a functional L2 norm regularization $\mathbb{E}_{q(\boldsymbol{w})}[||\boldsymbol{w}||^2]$ to penalize the likelihood for large values of $\boldsymbol{w}$. Alternatively, we could equivalently optimize the likelihood on $\mathbb{M}_C = \{q \mid q(\boldsymbol{w}) \geq 0 \text{ and } \int_{||\boldsymbol{w}||^2\leq C} q(\boldsymbol{w}) = 1\}$, which restricts the probability mass to a 2-norm ball $||\boldsymbol{w}||^2 \leq C$.

### 5.4.2 Learning Infinite RBMs with Frank-Wolfe

It is challenging to directly solve the optimization in Eq. (5.5) by standard gradient descent methods, because it involves optimizing the density function $q(\boldsymbol{w})$ in function space with infinite dimensions. Instead, we propose to solve it using the Frank-Wolfe algorithm [Jaggi, 2013], which is projection-free and provides a sparse solution that is a convex combination of only few "atoms".

In Frank-Wolfe algorithm, assume we have the the density function $q_t(\boldsymbol{w})$ at the iteration $t$; then the algorithm finds $q_{t+1}$ by maximizing the linearization of the objective function, and

taking a step in that direction :

$$q_{t+1} \leftarrow (1-\beta_{t+1})q_t + \beta_{t+1} r_{t+1}, \quad \text{where} \;\; r_{t+1} \leftarrow \operatorname*{argmax}_{q \in \mathbb{M}} \langle q, \nabla_q L(q_t, \vartheta_t) \rangle, \tag{5.6}$$

therein $\beta_{t+1} \in [0,1]$ is a step size parameter, and the convex combination step guarantees the new $q_{t+1}$ remains a distribution after the update. A typical step size is $\beta_t = 1/t$, in which case we have $q_t(\boldsymbol{w}) = \frac{1}{t}\sum_{s=1}^{t} r_s(\boldsymbol{w})$, that is, $q_t$ equals the average of all the earlier solutions obtained by the linear program.

To apply Frank-Wolfe to solve our problem, we need to solve the linear programming defined in E.q. (5.6), thus we first calculate the functional gradient of $L(q_t, \vartheta_t)$ w.r.t. the density function $q(\boldsymbol{w})$ as,

$$\begin{aligned}
\nabla_q L(q, \vartheta) = -\frac{\lambda}{2}||\boldsymbol{w}||^2 + \alpha \Bigg[ & \frac{1}{N}\sum_{n=1}^{N} \log(1+\exp(\boldsymbol{w}^\top \boldsymbol{v}^n)) \\
& - \frac{\sum_{\boldsymbol{v}} \exp\left(\alpha \mathbb{E}_{q(\boldsymbol{w})}[\log(1+\exp(\boldsymbol{w}^\top \boldsymbol{v}))] + \boldsymbol{b}^\top \boldsymbol{v}\right) \cdot \log(1+\exp(\boldsymbol{w}^\top \boldsymbol{v}))}{Z(q, \boldsymbol{b}, \alpha)} \Bigg] \\
= -\frac{\lambda}{2}||\boldsymbol{w}||^2 + \alpha & \left[ \frac{1}{N}\sum_{n=1}^{N} \log(1+\exp(\boldsymbol{w}^\top \boldsymbol{v}^n)) - \sum_{\boldsymbol{v}} p(\boldsymbol{v} \mid q, \vartheta) \log(1+\exp(\boldsymbol{w}^\top \boldsymbol{v})) \right],
\end{aligned}$$

where $p(\boldsymbol{v} \mid q_t, \vartheta_t)$ is the distribution parametrized by the weight density $q_t(\boldsymbol{w})$ and parameter $\vartheta_t$ at $t$-th iteration,

$$p(\boldsymbol{v} \mid q_t, \vartheta_t) = \frac{\exp\left(\alpha_t \mathbb{E}_{q_t(\boldsymbol{w})}[\log(1+\exp(\boldsymbol{w}^\top \boldsymbol{v}))] + \boldsymbol{b}_t^\top \boldsymbol{v}\right)}{Z(q_t, \vartheta_t)}. \tag{5.7}$$

It turns out that the (functional) linear program in Eq. (5.6) is equivalent to an optimization

over weight vector $\boldsymbol{w}$ :

$$\begin{aligned}
&\max_{q\in\mathbb{M}}\langle q, \nabla_q L(q_t, \vartheta_t)\rangle = \max_{q\in\mathbb{M}} \mathbb{E}_{q(\boldsymbol{w})}[\nabla_q L(q_t, \vartheta_t)] \\
=&\max_{\boldsymbol{w}} \left\{ -\frac{\lambda}{2}||\boldsymbol{w}||^2 + \frac{1}{N}\sum_{n=1}^{N}\log(1+\exp(\boldsymbol{w}^\top \boldsymbol{v}^n)) - \sum_{\boldsymbol{v}} p(\boldsymbol{v} \mid q_t, \vartheta_t)\log(1+\exp(\boldsymbol{w}^\top \boldsymbol{v}))\right\},
\end{aligned} \tag{5.8}$$

because the optimal $q(\boldsymbol{w})$ is a point mass at some $\boldsymbol{w}$ that can optimize the linear program.

The gradient of the objective Eq. (5.8) is,

$$\nabla_{\boldsymbol{w}}\delta(\boldsymbol{w}) = -\lambda\boldsymbol{w} + \frac{1}{N}\sum_{n=1}^{N}\sigma(\boldsymbol{w}^\top \boldsymbol{v}^n)\cdot \boldsymbol{v}^n - \mathbb{E}_{p(\boldsymbol{v}|q_t,\vartheta_t)}\big[\sigma(\boldsymbol{w}^\top \boldsymbol{v})\cdot \boldsymbol{v}\big],$$

where the expectation over $p(\boldsymbol{v} \mid q_t, \vartheta_t)$ can be intractable to calculate, and one may use stochastic optimization and draw samples using MCMC. Note that the second two terms in the gradient enforce an intuitive moment matching condition: the optimal $\boldsymbol{w}$ introduces a set of "importance weights" $\sigma(\boldsymbol{w}^\top \boldsymbol{v})$ that adjust the empirical data and the previous model, such that their moments match with each other.

Now, suppose $\boldsymbol{w}_t^*$ is the optimum of Eq. (5.8) at iteration $t$, the atom $r_t(\boldsymbol{w})$ we added in Eq. (5.6) can be shown to be the indicator function over $\boldsymbol{w}_t^*$, that is, $r_t(\boldsymbol{w}) = \mathbb{1}(\boldsymbol{w} = \boldsymbol{w}_t^*)$; in addition, we have $q_t(\boldsymbol{w}) = \frac{1}{t}\sum_{s=1}^{t}\mathbb{1}(\boldsymbol{w} = \boldsymbol{w}_s^*)$ when the step size is taken to be $\beta_t = \frac{1}{t}$. Therefore, this Frank-Wolfe update can be naturally interpreted as greedily inserting a hidden unit into the current model $p(\boldsymbol{v} \mid q_t, \vartheta_t)$. In particular, if we update the temperature parameter as $\alpha_t \leftarrow t$, according to Proposition 5.4.1, we can directly transform our model $p(\boldsymbol{v} \mid q_t, \vartheta_t)$ to a regular RBM after each Frank-Wolfe step, which enables the convenient blocked Gibbs sampling for inference.

Compared with the (regularized) MLE of the standard RBM (e.g. in Eq. (5.3)), the opti-

mization in Eq. (5.8) has the following nice properties: (1) The current model $p(\boldsymbol{v} \mid q_t, \vartheta_t)$ does not depend on $\boldsymbol{w}$, which means we can draw enough samples from $p(\boldsymbol{v} \mid q_t, \vartheta_t)$ at each iteration $t$, and reuse them during the optimization of $\boldsymbol{w}$. (2) The objective function in Eq. (5.8) can be evaluated explicitly given a set of samples, and hence efficient off-the-shelf optimization tools such as L-BFGS can be used to solve the optimization very efficiently. (3) Each iteration of our method involves many fewer parameters (only the weights for a single hidden unit, which is $|\boldsymbol{v}| \times 1$ instead of the full $|\boldsymbol{v}| \times |\boldsymbol{h}|$ weight matrix are updated), and hence defines a series of easier problems that can be less sensitive to initialization. We note that a similar greedy learning strategy has been successfully applied for learning mixture models [Verbeek et al., 2003], in which one greedily inserts a component at each step, and that this approach can provide better initialization for EM optimization than using multiple random initializations.

Once we obtain $q_{t+1}$, we update the bias parameter $\boldsymbol{b}_t$ by gradient descent,

$$\nabla_{\boldsymbol{b}} L(q_{t+1}, \vartheta_t) = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{v}^n - \sum_{\boldsymbol{v}} p(\boldsymbol{v} | q_{t+1}, \vartheta_t) \boldsymbol{v}. \tag{5.9}$$

We can further optimize $\alpha_t$ by gradient descent, and the gradient of $L(q, \vartheta)$ w.r.t. $\alpha$ is

$$\nabla_{\alpha} L(q, \vartheta) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{w})} \big[ \log(1 + \exp(\boldsymbol{w}^\top \boldsymbol{v}^n)) \big] - \sum_{\boldsymbol{v}} p(\boldsymbol{v} \mid q, \vartheta) \; \mathbb{E}_{q(\boldsymbol{w})} \big[ \log(1 + \exp(\boldsymbol{w}^\top \boldsymbol{v})) \big].$$

However, we find simply updating $\alpha_t \leftarrow t$ is more efficient and works well in practice, and has the additional advantage to be a valid RBM. We summarize our Frank-Wolfe learning algorithm in Algorithm 5.1.

*Adding hidden units on RBM.* Besides initializing $q(\boldsymbol{w})$ to be a delta function at some random $\boldsymbol{w}'$ and learning the model from scratch, one can also adapt Algorithm 5.1 to incrementally add hidden units into an existing RBM in Eq. (5.2) (e.g. have been learned by CD). Ac-

**Algorithm 5.1** Frank-Wolfe Learning Algorithm

**Input:** training data $\{\boldsymbol{v}^n\}_{n=1}^N$; step size $\eta$; regularization $\lambda$.
**Output:** sparse solution $q^*(\boldsymbol{w})$, and $\vartheta^*$

Initialize $q_0(\boldsymbol{w}) = \mathbb{1}(\boldsymbol{w} = \boldsymbol{w}')$ at random $\boldsymbol{w}'$; $\boldsymbol{b}_0 = 0$; $\alpha_0 = 1$;

**for** $t = 1 : T$ [or, stopping criterion] **do**

Draw sample $\{\boldsymbol{v}^s\}_{s=1}^S$ from $p(\boldsymbol{v} \mid q_{t-1}, \vartheta_{t-1})$;

$\boldsymbol{w}_t^* = \text{argmax}_{\boldsymbol{w}} \left\{ -\frac{\lambda}{2}||\boldsymbol{w}||^2 + \frac{1}{N}\sum_{n=1}^N \log(1+\exp(\boldsymbol{w}^\top \boldsymbol{v}^n)) - \frac{1}{S}\sum_{s=1}^S \log(1+\exp(\boldsymbol{w}^\top \boldsymbol{v}^s)) \right\}$;

Update $q_t(\boldsymbol{w}) \leftarrow (1 - \frac{1}{t}) \cdot q_{t-1}(\boldsymbol{w}) + \frac{1}{t} \cdot \mathbb{1}(\boldsymbol{w} = \boldsymbol{w}_t^*)$;

Update $\alpha_t \leftarrow t$ (optional: gradient descent);

Set $\boldsymbol{b}_t = \boldsymbol{b}_{t-1}$;

**repeat**

Draw a mini-batch samples $\{\boldsymbol{v}^m\}_{m=1}^M$ from $p(\boldsymbol{v} \mid q_t, \vartheta_t)$

Update $\boldsymbol{b}_t \leftarrow \boldsymbol{b}_t + \eta \cdot (\frac{1}{N}\sum_{n=1}^N \boldsymbol{v}^n - \frac{1}{M}\sum_{m=1}^M \boldsymbol{v}^m)$

**until**

**end for**

Return $q^*(\boldsymbol{w}) = q_t(\boldsymbol{w})$; $\vartheta^* = \{\boldsymbol{b}_t, \alpha_t\}$;

cording to Proposition 5.4.1, one can simply initialize $q_t(\boldsymbol{w}) = \frac{1}{|\boldsymbol{h}|}\sum_{i=1}^{|\boldsymbol{h}|} \mathbb{1}(\boldsymbol{w} = \boldsymbol{w}_i)$, $\alpha_t = |\boldsymbol{h}|$, and continue the Frank-Wolfe iterations at $t = |\boldsymbol{h}| + 1$.

*Removing hidden units.* Since the hidden units are added in a greedy manner, one may want to remove an old hidden unit during the Frank-Wolfe learning, provided it is bad with respect to our objective Eq. (5.8) after more hidden units have been added. A variant of Frank-Wolfe with *away-steps* [Guélat and Marcotte, 1986] fits this requirement and can be directly applied. As shown by [Clarkson, 2010], it can improve the sparsity of the final solution (i.e., fewer hidden units in the learned model).

### 5.4.3 MCMC Inference for Fractional RBMs

As we point out in Section 5.4.1, we need to take $\alpha$ equal to the number of particles in $q(\boldsymbol{w})$ (that is, $\alpha_t \leftarrow t$ in Algorithm 5.1) in order to have our model reduce to the standard RBM. If $\alpha$ takes a more general real number, we obtain a more general *fractional* RBM model, for

which inference is more challenging because the standard block Gibbs sampler is not directly applicable. In practice, we find that setting $\alpha_t \leftarrow t$ to correspond to a regular RBM seems to give the best performance, but for completeness, we discuss the fractional RBM in more detail in this section, and propose a Metropolis-Hastings algorithm to draw samples from the fractional RBM. We believe that this fractional RBM framework provides an avenue for further improvements in future work.

To frame the problem, let us assume $\alpha q(\boldsymbol{w}) = \sum_i c_i \cdot \mathbb{1}(\boldsymbol{w} = \boldsymbol{w}_i)$, where $c_i$ is a general real number; the corresponding model is

$$\log p(\boldsymbol{v} \mid q, \vartheta) = \sum_i c_i \log(1 + \exp(\boldsymbol{w}_i^\top \boldsymbol{v})) + \boldsymbol{b}^\top \boldsymbol{v} - \log Z(q, \vartheta), \tag{5.10}$$

which differs from the standard RBM in (5.2) because each *softplus* function is multiplied by $c_i$. Nevertheless, one may push the $c_i$ into the softplus function, and obtain a standard RBM that forms an approximation of (5.10):

$$\log \widetilde{p}(\boldsymbol{v} \mid q, \vartheta) = \sum_i \log(1 + \exp(c_i \cdot \boldsymbol{w}_i^\top \boldsymbol{v})) + \boldsymbol{b}^\top \boldsymbol{v} - \log \widetilde{Z}(q, \vartheta). \tag{5.11}$$

This approximation can be justified by considering the special case when the magnitude of the weights $\boldsymbol{w}$ is very large, so that the softplus function essentially reduces to a rectified linear unit (ReLU) function, that is, $\log(1 + \exp(\boldsymbol{w}_i^\top \boldsymbol{v})) \approx \max(0, \boldsymbol{w}_i^\top \boldsymbol{v})$. In this case, (5.10) and (5.11) become equivalent because $c_i \max(0, x) = \max(0, c_i x)$. More concretely, we can guarantee the following bound:

**Proposition 5.4.2.** *For any $0 < c_i \leq 1$, we have*

$$\frac{1}{2^{1-c_i}}(1 + \exp(c_i \cdot \boldsymbol{w}_i^\top \boldsymbol{v})) \leq (1 + \exp(\boldsymbol{w}_i^\top \boldsymbol{v}))^{c_i} \leq 1 + \exp(c_i \cdot \boldsymbol{w}_i^\top \boldsymbol{v}).$$

*Proof.* For any $0 < c \leq 1$, we have following classical inequality,

$$\sum_k x_k \leq (\sum_k x_k^c)^{1/c}, \quad \text{and} \quad \frac{1}{2}\sum_k x_k \leq (\frac{1}{2}\sum_k x_k^c)^{1/c}$$

Let $x_1 = 1$ and $x_2 = \exp(\boldsymbol{w}_i^\top \boldsymbol{v})$, and the proposition is a direct result of above two inequalities. □

Note that we apply the bound when $c_i > 1$ by splitting $c_i$ into the sum of its integer part and fractional remainder, and apply the bound to the fractional part.

Therefore, the fractional RBM (5.10) can be well approximated by the standard RBM (5.11), and this can be leveraged to design an inference algorithm for (5.10). As one example, we can use the Gibbs update of (5.11) as a proposal for a Metropolis-Hastings update for (5.10). To be specific, given a configuration $\boldsymbol{v}$, we perform Gibbs update in RBM $\widetilde{p}(\boldsymbol{v} \mid q, \vartheta)$ to get $\boldsymbol{v}'$, and accept it with probability $\min(1,\ A(\boldsymbol{v} \to \boldsymbol{v}'))$,

$$A(\boldsymbol{v} \to \boldsymbol{v}') = \frac{p(\boldsymbol{v}')\widetilde{T}(\boldsymbol{v}' \to \boldsymbol{v})}{p(\boldsymbol{v})\widetilde{T}(\boldsymbol{v} \to \boldsymbol{v}')},$$

where $\widetilde{T}(\boldsymbol{v} \to \boldsymbol{v}')$ is the Gibbs transition of RBM $\widetilde{p}(\boldsymbol{v} \mid q, \vartheta)$. Because the acceptance probability of a Gibbs sampler equals one, we have $\frac{\widetilde{p}(\boldsymbol{v})\widetilde{T}(\boldsymbol{v}\to\boldsymbol{v}')}{\widetilde{p}(\boldsymbol{v}')\widetilde{T}(\boldsymbol{v}'\to\boldsymbol{v})} = 1$ . This gives

$$A(\boldsymbol{v} \to \boldsymbol{v}') = \frac{p(\boldsymbol{v}')\widetilde{p}(\boldsymbol{v})}{p(\boldsymbol{v})\widetilde{p}(\boldsymbol{v}')} = \frac{\prod_i (1 + \exp(\boldsymbol{w}_i^\top \boldsymbol{v}'))^{c_i} \cdot \prod_i (1 + \exp(c_i \cdot \boldsymbol{w}_i^\top \boldsymbol{v}))}{\prod_i (1 + \exp(\boldsymbol{w}_i^\top \boldsymbol{v}))^{c_i} \cdot \prod_i (1 + \exp(c_i \cdot \boldsymbol{w}_i^\top \boldsymbol{v}'))} \ .$$

## 5.5 Experiments

In this section, we test the performance of our Frank-Wolfe (FW) learning algorithm on two datasets: MNIST [LeCun et al., 1998] and Caltech101 Silhouettes [Marlin et al., 2010]. The MNIST handwritten digits database contains 60,000 images in the training set and 10,000

test set images, where each image $\boldsymbol{v}^n$ includes $28 \times 28$ pixels and is associated with a digit label $y^n$. We binarize the grayscale images by thresholding the pixels at 127, and randomly select 10,000 images from training as the validation set. The Caltech101 Silhouettes dataset [Marlin et al., 2010] has 8,671 images with $28 \times 28$ binary pixels, where each image represents objects silhouette and has a class label (overall 101 classes). The dataset is divided into three subsets: 4,100 examples for training, 2,264 for validation and 2,307 for testing.

**Training algorithms** We train RBMs with CD-10 algorithm. [1] A fixed learning rate is selected from the set $\{0.05, 0.02, 0.01, 0.005\}$ using the validation set, and the mini-batch size is selected from the set $\{10, 20, 50, 100, 200\}$. We use 200 epochs for training on MINIST and 400 epochs on Caltech101. Early stopping is applied by monitoring the difference of average log-likelihood between training and validation data, so that the intractable log-partition function is cancelled [Hinton, 2010]. We train RBMs with $\{20, 50, 100, 200, 300, 400, 500, 600, 700\}$ hidden units. We incrementally train a RBM model using the Frank-Wolfe (FW) algorithm 5.1. A fixed step size $\eta$ is selected from the set $\{0.05, 0.02, 0.01, 0.005\}$ using the validation data, and a regularization strength $\lambda$ is selected from the set $\{1, 0.5, 0.1, 0.05, 0.01\}$. We set $T = 700$ in Algorithm 5.1, and use the same early stopping criterion as CD. We randomly initialize the CD algorithm 5 times and select the best one on the validation set; meanwhile, we also initialize CD by the model learned from Frank-Wolfe.

**Test likelihood** To evaluate the test likelihood of the learned models, we estimate the partition function using annealed importance sampling (AIS) [Salakhutdinov and Murray, 2008]. The temperature parameter is selected following the standard guidance: first 500 temperatures spaced uniformly from 0 to 0.5, and 4,000 spaced uniformly from 0.5 to 0.9, and 10,000 spaced uniformly from 0.9 to 1.0; this gives a total of 14,500 intermediate distributions. We summarize the averaged test log-likelihood of MNIST and Caltech101 Silhouettes in

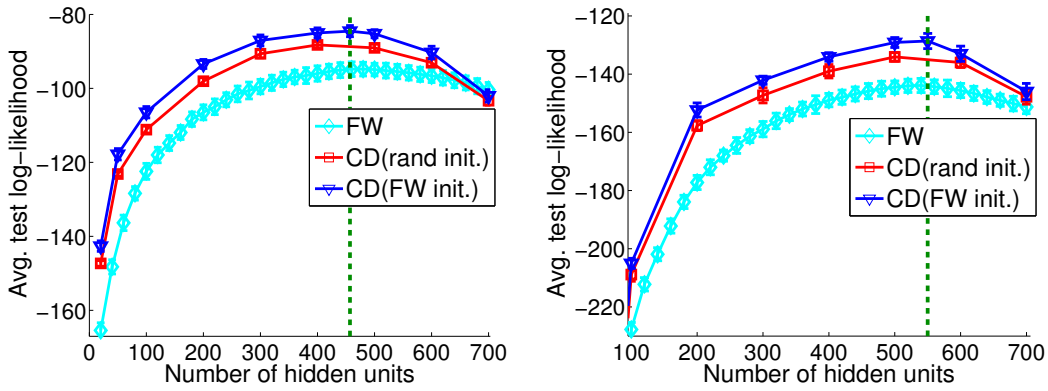
[1] CD-k refers to using k-step Gibbs sampler to approximate the gradient of the log-partition function.

Figure 5.1, where we report the result averaged over 500 AIS runs in all experiments, with the error bars indicating the 3 standard deviations of the estimations.

We evaluate the test likelihood of the model learned by FW after adding every 20 hidden units. We perform early stopping when the gap of average log-likelihood between training and validation data largely increases. As shown in Figure 5.1, this procedure selects 460 hidden units on MNIST (as indicated by the green dashed lines), and 550 hidden units on Caltech101; purely for illustration purposes, we continue FW in the experiment until reaching $T = 700$ hidden units. We can see that the identified number of hidden units roughly corresponds to the maximum of the test log-likelihood of all the three algorithms, suggesting that FW can identify the appropriate number of hidden units during the optimization.

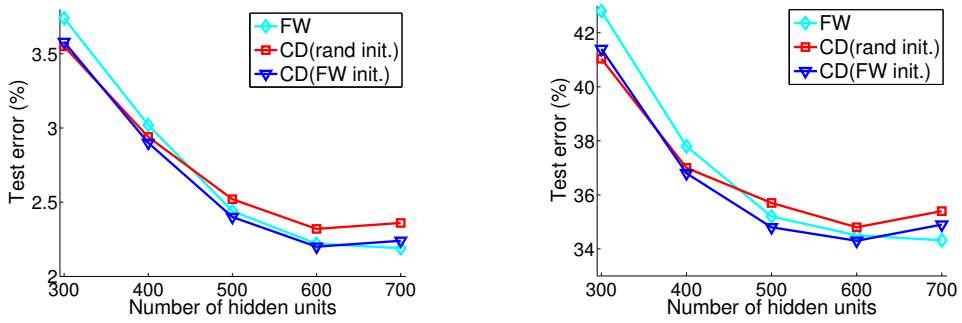
We also use the model learned by FW as an initialization for CD (the blue lines in Figure 5.2), and find it consistently performs better than the best result of CD with 5 random initializations. In our implementation, the running time of the FW procedure is at most twice as CD for the same number of hidden units. Therefore, FW initialized CD provides a practical strategy for learning RBMs: it requires approximately three times of computation time as a single run of CD, while simultaneously identifying the proper number of hidden units and obtaining better test likelihood.

**Classification** The performance of our method is further evaluated using discriminant image classification tasks. We take the hidden units' activation vectors $\mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{v}^n)}[\boldsymbol{h}]$ generated by the three algorithms in Figure 5.1 and use it as the feature in a multi-class logistic regression on the class labels $y^n$ in MNIST and Caltech101. From Figure 5.2, we find that our basic FW tends to be worse than the fully trained CD (best in 5 random initializations) when only small numbers of hidden units are added, but outperforms CD when more hidden units (about 450 in both cases) are added. Meanwhile, the CD initialized by FW outperforms CD using the best of 5 random initializations.




(a) MNIST (b) Caltech101 Silhouettes

Figure 5.1: Average test log-likelihood on the two datasets as we increase the number of hidden units. We can see that FW can correctly identify an appropriate hidden layer size with high test log-likelihood (marked by the green dashed line). In addition, CD initialized by FW gives higher test likelihood than random initialization for the same number of hidden units. Best viewed in color.




(a) MNIST (b) Caltech101 Silhouettes

Figure 5.2: Classification error when using the learned hidden representations as features. (Best viewed in color).

## 5.6 Conclusion

In this chapter, we propose a convex relaxation of the RBM with an infinite number of hidden units, whose MLE corresponds to a constrained convex program in a function space. We solve the program using Frank-Wolfe, which provides a sparse greedy solution that can be interpreted as inserting a single hidden unit at each iteration. Our new method allows us to easily identify the appropriate number of hidden units during the progress of learning, and can provide an advanced initialization strategy for other state-of-the-art training methods such as CD to achieve higher test likelihood than random initialization.

Chapter 6

# Belief Propagation in Conditional RBMs for Structured Prediction

Conditional RBMs (CRBMs) are popular latent variable models for many supervised learning tasks, such as human motion capture [Taylor et al., 2006a], collaborative filtering [Salakhutdinov et al., 2007a], and general structured prediction [Mnih et al., 2011, Yang et al., 2014]. Typically, learning on such models is dominated by contrastive divergence (CD) and its variants. Although belief propagation (BP) algorithms can be used as inference routines in training as well as for making predictions in test, they are believed to be slow on RBM-based models (e.g., Mnih et al. [2011]), and not as good as CD when applied in learning (e.g., Larochelle et al. [2012]). In this chapter, we present a matrix-based implementation of BP algorithms, which is easily scalable to tens of thousands of visible and hidden units. In addition, our algorithms uses standard matrix product and element-wise operations, and is thus highly suitable for modern high-performance computing architecture. We demonstrate that, in both maximum likelihood and max-margin learning, training CRBMs with BP as the inference routine can provide significantly better results than current state-of-the-art CD methods on structured prediction problems. We also include practical guidelines on training CRBMs with BP, and some insights on the interaction of learning and inference algorithms for CRBMs. This Chapter is based on our paper in submission [Ping and Ihler, 2017].

## 6.1 Introduction

As we reviewed in Section 2.1.4 of Chapter 2, a restricted Boltzmann machine (RBM) uses a layer of hidden units $\boldsymbol{h}$ to model the distribution of visible units $\boldsymbol{v}$. Due to the intractability of the partition function in maximum likelihood estimation (MLE), RBMs are usually learned using the contrastive divergence (CD) algorithm [Hinton, 2002a], which approximates the gradient of the log-partition function using a k-step Gibbs sampler (referred to as CD-k). To speed up the convergence of the Markov chain, a critical trick in CD-k is to initialize the state of the Markov chain with each training instance. Although it has been shown that CD-k does not follow the gradient of any objective function [Sutskever and Tieleman, 2010], it works well in many practical applications [Hinton, 2010]. An important variant of CD-k is persistent CD (PCD) [Tieleman, 2008]. PCD uses a persistent Markov chain during learning, where the Markov Chain is not reset between parameter updates. Because the learning rate is usually small and the model changes only slightly between parameter updates, the long-run persistent chain in PCD usually provides a better approximation to the target distribution than the limited step chain in CD-k.

A conditional restricted Boltzmann machine (CRBM) is the discriminative extension of RBM to include observed features $\boldsymbol{x}$ (see Section 2.1.5 in Chapter 2 for detailed review). In contrast to the success of CD methods for RBMs, it has been noted that both CD-k and PCD may not be well suited to learning conditional RBMs [Mnih et al., 2011]. In particular, PCD is not appropriate for learning such conditional models, because the observed features $\boldsymbol{x}$ greatly affect the model potentials. This means we need to run a separate persistent chain for every training instance, which is costly for large datasets. To make things worse, as we revisit a training instance in stochastic gradient descent (SGD) (which is standard practice for large datasets), the model parameters will have changed substantially, making the persistent chain for this instance far from the target distribution. Also, given the observed features,

CRBMs tend to be more peaked than RBMs in a purely generative setting. CD methods may make slow progress because it is difficult for the sampling procedure to explore these peaked but multi-modal distributions. It was also observed that the important trick in CD-k, which initializes the Markov chain using the training data, does not work well for CRBMs in structured prediction [Mnih et al., 2011]. In contrast, starting the Gibbs chain with a random state (which resembles the original learning algorithm for Boltzmann machines [Ackley et al., 1985]) provides better results.

Approximate inference methods, such as mean field (MF) and belief propagation (BP), can be employed as inference routines in learning as well as for making predictions after the CRBM has been learned [Welling and Teh, 2003, Yasuda and Tanaka, 2009]. Although loopy BP usually provides a better approximation of marginals than MF [Murphy et al., 1999], it was found to be slow on CRBMs for structured prediction and only considered practical on problems with few visible and hidden nodes [Mnih et al., 2011, Mandel et al., 2011]. This inefficiency prevents it from being widely applied to conditional RBMs for structured prediction, in which the CRBMs may have thousands of visible and hidden units. More importantly, there is a pervasive opinion that belief propagation does not work well on RBM-based models, especially for learning [Goodfellow et al., 2016, Chapter 16].

In this chapter, we present an efficient implementation of belief propagation algorithms for conditional RBMs. It takes advantage of the bipartite graph structure and is scalable to tens of thousands of visible units and hidden units.[1] Our algorithm uses a compact representation and only depends on matrix product and element-wise operations, which are typically optimized in modern high-performance computing (HPC) architecture. We demonstrate that, in the conditional setting, learning RBM-based models with belief propagation and its variants can provide consistently better results than the state-of-the-art CD methods. We also show

[1] For random RBMs with $10,000$ visible units and $2,000$ hidden units, our Matlab implementation converges within a few seconds on a desktop with Intel Core i7 (3.6 GHz).

that the marginal structured SVM (MSSVM; in Chapter 4) can provide improvements for max-margin learning of CRBMs [Yang et al., 2014]. We include practical guidelines on training CRBMs, and some insights on the interaction of learning and message-passing algorithms for CRBMs.

We organize the rest of the chapter as follows. Section 6.2 discusses some connections to related work. We review the RBM model and conditional RBMs in Section 6.3 and discuss the learning algorithms in Section 6.4. In Section 6.5, we provide our efficient inference procedure. We report experimental results in Section 6.6 and conclude the paper in Section 6.7.

## 6.2 Related Work

Mnih et al. [2011] proposed the CD-PercLoss algorithm for conditional RBMs, which uses a CD-like stochastic search procedure to minimize the perceptron loss on training data. Given the observed features of the training instance, CD-PercLoss starts the Gibbs chain using the logistic regression component of the CRBM. Yang et al. [2014] trained CRBMs using a latent structured SVM (LSSVM) objective [Yu and Joachims, 2009b], and used a greedy search (i.e., iterated conditional modes) for joint maximum a posteriori (MAP) inference over both hidden and visible units.

It is also feasible to apply the mean-field (MF) approximation for the partition function in MLE learning of RBMs and CRBMs [Peterson and Anderson, 1987]. Although efficient, this is conceptually problematic in the sense that it effectively maximizes an upper bound of the log-likelihood during learning. In addition, MF uses a unimodal proposal to approximate the multi-modal distribution, which may lead to unsatisfactory results.

Although belief propagation (BP) and its variants have long been used to learn conditional random fields (CRFs) with hidden variables [Quattoni et al., 2007b, Ping et al., 2014], they

are mainly applied on sparsely connected graphs (e.g., chains and grids) and were believed to be ineffective and slow on very dense graphs like CRBMs [Mnih et al., 2011, Goodfellow et al., 2016]. A few recent works [Krähenbühl and Koltun, 2012, Zhang and Chen, 2012] impose particular assumptions on the type of edge potentials and provide efficient inference algorithms for fully connected CRFs. For example, the edge potentials in [Krähenbühl and Koltun, 2012] are defined by a linear combination of Gaussian kernels, which enables efficient message-passing using Gaussian filtering . In this chapter, however, we propose to speed up general belief propagation on conditional RBMs without any potential function restrictions.

## 6.3 Background and Notations

As we reviewed in Section 2.1.5 in Chapter 2, the conditional RBM (CRBM) extends RBMs to include observed features $\boldsymbol{x}$, and defines a joint conditional distribution over $\boldsymbol{v}$ and $\boldsymbol{h}$ given input features $\boldsymbol{x} \in \mathbb{R}^{|\boldsymbol{x}|\times 1}$,

$$p(\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{x}; \theta) = \frac{1}{Z(\boldsymbol{x}; \theta)} \exp\big(-E(\boldsymbol{v}, \boldsymbol{h}, \boldsymbol{x}; \theta)\big), \tag{6.1}$$

where the energy function $E$ is defined as,

$$E(\boldsymbol{v}, \boldsymbol{h}, \boldsymbol{x}; \theta) = -\boldsymbol{v}^\top W^{vh} \boldsymbol{h} - \boldsymbol{v}^\top W^{vx} \boldsymbol{x} - \boldsymbol{h}^\top W^{hx} \boldsymbol{x} - \boldsymbol{v}^\top \boldsymbol{b}^v - \boldsymbol{h}^\top \boldsymbol{b}^h,$$

and $\theta = \{W^{vh}, W^{vx}, W^{hx}, b^v, b^h\}$ are model parameters. $Z(\boldsymbol{x}; \theta)$ is the $\boldsymbol{x}$-dependent partition function,

$$Z(\boldsymbol{x}; \theta) = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp\big(-E(\boldsymbol{v}, \boldsymbol{h}, \boldsymbol{x}; \theta)\big).$$

### 6.3.1 Structured Prediction with CRBMs

The conditional RBMs are widely applied in supervised learning. In this Chapter, we consider the general structured prediction framework, which incorporates many popular applications, such as collaborative filtering [Salakhutdinov et al., 2007b], multi-label learning [Li et al., 2015], image denoising [Mnih et al., 2011] and semantic segmentation [Yang et al., 2014].

In structured prediction, the visible units $\boldsymbol{v}$ typically represent output variables, while the observed $\boldsymbol{x}$ represent input features, and the hidden units $\boldsymbol{h}$ facilitate the modeling of output variables given observed features. To make predictions, one choice is to infer the modes of the singleton marginals, $p(v_i|\boldsymbol{x}) = \sum_{\boldsymbol{v}_{\setminus i}} \sum_{\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{x})$. This marginalization inference is intractable and is closely related to calculating the partition function. One can also decode the output $\boldsymbol{v}$ by performing joint maximum *a posteriori* (MAP) inference [e.g., Yu and Joachims, 2009b, Yang et al., 2014],

$$(\hat{\boldsymbol{v}}, \hat{\boldsymbol{h}}) = \operatorname*{argmax}_{\boldsymbol{v},\boldsymbol{h}} \; p(\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{x}),$$

which gives a prediction for the pair $(\boldsymbol{v}, \boldsymbol{h})$; one obtains a prediction of $\boldsymbol{v}$ by simply discarding the $\boldsymbol{h}$ component. Intuitively, the joint MAP prediction is "over-confident", since it deterministically assigns the hidden units to their most likely states, and is not robust when the uncertainty of the hidden units is high. As we discussed in Chapter 4, one promising alternative for CRBMs to use is marginal MAP prediction:

$$\tilde{\boldsymbol{v}} = \operatorname*{argmax}_{\boldsymbol{v}} p(\boldsymbol{v}|\boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp\big(-E(\boldsymbol{v}, \boldsymbol{h}, \boldsymbol{x}; \theta)\big),$$

which explicitly takes into account the uncertainty of the hidden units by marginalizing them out. In general, these predictions are intractable in CRBMs, and one must use approximate inference methods, such as mean field or belief propagation.

## 6.4 Learning with CRBMs

In this section, we discuss different learning methods for conditional RBMs.

### 6.4.1 MLE and Related Algorithms

Assume we have a training set $\{\boldsymbol{v}^n, \boldsymbol{x}^n\}_{n=1}^N$; then, the log-likelihood can be written as,[2]

$$\sum_{n=1}^{N} \Big\{ \log \sum_{\boldsymbol{h}} \exp\big( - E(\boldsymbol{v}^n, \boldsymbol{h}, \boldsymbol{x}^n; \theta)\big) - \log Z(\boldsymbol{x}^n; \theta) \Big\}.$$

To efficiently maximize the objective function, stochastic gradient descent (SGD) is usually applied. Given a randomly chosen instance $\{\boldsymbol{v}^n, \boldsymbol{x}^n\}$, one can show that the gradient of log-likelihood w.r.t. $W^{vh}$ is,

$$\frac{\partial \log p(\boldsymbol{v}^n|\boldsymbol{x}^n)}{\partial W^{vh}} = \boldsymbol{v}^n (\boldsymbol{\mu}^n)^\top - \mathbb{E}_{p(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{x}^n)}\big[\boldsymbol{v}\boldsymbol{h}^\top\big], \tag{6.2}$$

where $\boldsymbol{\mu}^n = \mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{v}^n,\boldsymbol{x}^n)}\big[\boldsymbol{h}\big] = \sigma(W^{vh\top}\boldsymbol{v}^n + W^{hx}\boldsymbol{x}^n + \boldsymbol{b}^h)$ and the logistic function $\sigma$ is applied in an element-wise manner. The positive part of the gradient can be calculated exactly. The negative part arises from the derivatives of the log-partition function and is intractable to calculate. The gradients of log-likelihood w.r.t. other pairwise and bias parameters are analogous to Eq. (6.2), and listed as follows,

$$\begin{aligned}
\frac{\partial \log p(\boldsymbol{v}^n|\boldsymbol{x}^n)}{\partial W^{vx}} &= \boldsymbol{v}^n \boldsymbol{x}^{n\top} - \mathbb{E}_{p(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{x}^n)}\big[\boldsymbol{v}\boldsymbol{x}^{n\top}\big], \\
\frac{\partial \log p(\boldsymbol{v}^n|\boldsymbol{x}^n)}{\partial W^{hx}} &= \boldsymbol{\mu}^n \boldsymbol{x}^{n\top} - \mathbb{E}_{p(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{x}^n)}\big[\boldsymbol{h}\boldsymbol{x}^{n\top}\big], \\
\frac{\partial \log p(\boldsymbol{v}^n|\boldsymbol{x}^n)}{\partial b^v} &= \boldsymbol{v}^n - \mathbb{E}_{p(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{x}^n)}\big[\boldsymbol{v}\big], \\
\frac{\partial \log p(\boldsymbol{v}^n|\boldsymbol{x}^n)}{\partial b^h} &= \boldsymbol{\mu}^n - \mathbb{E}_{p(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{x}^n)}\big[\boldsymbol{h}\big],
\end{aligned}$$

[2] One can also introduce some regularization terms, e.g. a Frobenius norm of $W^{vh}$, $W^{vx}$, $W^{hx}$, to avoid overfitting.

All the negative parts of these gradients are intractable to calculate, and must be approximated during learning.

CD-k initializes the Gibbs chain by instance $\boldsymbol{v}^n$, and performs $k$-step Gibbs sampling by Eq. (2.12). Then, the empirical moment is used as a substitute for the intractable expectation $\mathbb{E}_{p(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{x}^n)}\left[\boldsymbol{v}\boldsymbol{h}^\top\right]$. Although this works well on RBMs, it gives unsatisfactory results on CRBMs. In practice, the conditional distributions $p(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{x}^n)$ are strongly influenced by the observed features $\boldsymbol{x}^n$ , and usually more peaked than generative RBMs. It is usually difficult for a Markov chain with few steps (e.g., 10) to explore these peaked and multi-modal distributions. PCD uses a long-run persistent Markov chain to improve convergence, but is not suitable for CRBMs as discussed in Section 6.1.

Sum-product BP and mean field methods provide pseudo-marginals as substitutes for the intractable expectations in Eq. (6.2). These deterministic gradient estimates have the advantage that a larger learning rate can be used. BP tends to give a more accurate estimate of $\log Z$ and marginals, but is reported to be slow on CRBMs and is impractical on problems with large output dimension and hidden layer sizes [Mnih et al., 2011] in structured prediction.

More importantly, it was observed that belief propagation usually gives unsatisfactory results when learning vanilla RBMs. This is mainly because the parameters' magnitude gradually increases during learning; the RBM model eventually undergoes a "phase transition" after which BP has difficulty converging [Ihler et al., 2005, Mooij and Kappen, 2005]. If BP does not converge, it can not provide a meaningful gradient direction to update the model, and the leaning becomes stuck. However, CRBMs appear to behave quite differently, due to operating in the "high signal" regime provided by an informative observation $\boldsymbol{x}$. This improves the convergence behaviour of BP, which may not be surprising since loopy BP is widely accepted as useful in learning other conditional models (e.g., grid CRFs for image

segmentation). In addition, given $N$ training instances for learning the CRBM, BP is actually performed on $N$ different RBMs corresponding to different features $\boldsymbol{x}^n$. During any particular phase of learning, BP may have trouble converging on some training instances, but we can still make progress as long as BP converges on the majority of instances. We demonstrate this behaviour in our experiments.

### 6.4.2 Max-Margin Learning

Another by-product of using BP is that it enables us to apply our marginal structured SVM (MSSVM) framework for max-margin learning of CRBMs,

$$\min_{\theta} \sum_{n=1}^{N} \Big\{ \max_{\boldsymbol{v}} \log \sum_{\boldsymbol{h}} \exp\Big(\Delta(\boldsymbol{v}, \boldsymbol{v}^n) - E(\boldsymbol{v}, \boldsymbol{h}, \boldsymbol{x}; \theta)\Big) - \log \sum_{\boldsymbol{h}} \exp\Big( - E(\boldsymbol{v}^n, \boldsymbol{h}, \boldsymbol{x}^n; \theta)\Big)\Big\}, \tag{6.3}$$

where the loss function $\Delta(\boldsymbol{v}, \boldsymbol{v}^n) = \sum_i \Delta(v_i, v_i^n)$ is decomposable (e.g., Hamming loss). In contrast to LSSVM [Yu and Joachims, 2009b, Yang et al., 2014], MSSVM marginalizes over the uncertainty of hidden variables, and can significantly outperform LSSVM when that uncertainty is large [Ping et al., 2014]. Experimentally, we find that MSSVM improves performance of max-margin CRBMs, likely because there is usually non-trivial uncertainty in the hidden units. Given an instance $\{\boldsymbol{v}^n, \boldsymbol{x}^n\}$, the stochastic gradient of Eq. (6.3) w.r.t. $W^{vh}$ is,

$$\frac{\partial l(\boldsymbol{v}^n, \boldsymbol{x}^n)}{\partial W^{vh}} = \mathbb{E}_{p(\boldsymbol{h}|\hat{\boldsymbol{v}}, \boldsymbol{x}^n)}\big[\hat{\boldsymbol{v}}\boldsymbol{h}^T\big] - \boldsymbol{v}^n(\boldsymbol{\mu}^n)^\top, \tag{6.4}$$

where $\boldsymbol{\mu}^n$ is defined as in Eq. (6.2); $\hat{\boldsymbol{v}}$ is the loss-augmented marginal MAP prediction,

$$\hat{\boldsymbol{v}} = \operatorname*{argmax}_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp\Big(\Delta(\boldsymbol{v}, \boldsymbol{v}^{(n)}) - E(\boldsymbol{v}, \boldsymbol{h}, \boldsymbol{x}^n; \theta)\Big);$$

and "mixed-product" belief propagation for marginal MAP [Liu and Ihler, 2013] can provide pseudo-marginals to estimate the intractable expectation. (The gradients for other parameters are analogous.)

## 6.5 Approximate Inference in RBM

In this section, we present a matrix-based implementation of sum-product and mixed-product BP algorithms for RBMs. Given a particular $\boldsymbol{x}^n$ in CRBM (6.1), we obtain a $\boldsymbol{x}^n$-dependent RBM model,

$$p(\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{x}^n) = \frac{1}{Z(\theta(\boldsymbol{x}^n))} \exp\left(\boldsymbol{v}^\top W^{vh}\boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{b}^1 + \boldsymbol{h}^\top \boldsymbol{b}^2\right),$$

where the bias terms $\boldsymbol{b}^1 = \boldsymbol{b}^v + W^{vx}\boldsymbol{x}^n$, $\boldsymbol{b}^2 = \boldsymbol{b}^h + W^{hx}\boldsymbol{x}^n$, and thus we can directly apply the algorithm to CRBMs.

### 6.5.1 Message-passing in RBMs

We first review the standard message-passing form in RBMs. Because RBM is a special pairwise model, one can directly apply the loopy BP algorithm defined in Algorithm 2.1. However, on a dense graphical models like RBMs, to reduce the amount of calculation, one should always pre-compute the product of incoming messages (or the beliefs) on the nodes, and reuse them to perform updates of all outgoing messages. In sum-product BP, we write the fixed-point update rule for the message sent from hidden unit $h_j$ to visible unit $v_i$ as,

$$m_{j\to i}(v_i) \propto \sum_{h_j} \exp\left(v_i W_{ij}^{vh} h_j\right) \cdot \frac{\tau(h_j)}{m_{i\to j}(h_j)}, \tag{6.5}$$

where the belief on $h_j$ is

$$\tau(h_j) \propto \exp\left(h_j b_j^2\right) \cdot \prod_{k=1}^{|\boldsymbol{v}|} m_{k \to j}(h_j). \tag{6.6}$$

The update rule for the message sent from $v_i$ to $h_j$ is,

$$m_{i \to j}(h_j) \propto \sum_{v_i} \exp\left(v_i W_{ij}^{vh} h_j\right) \cdot \frac{\tau(v_i)}{m_{j \to i}(v_i)}, \tag{6.7}$$

where the belief on $v_i$ is,

$$\tau(v_i) \propto \exp\left(v_i b_i^1\right) \cdot \prod_{k=1}^{|\boldsymbol{h}|} m_{k \to i}(v_i). \tag{6.8}$$

In mixed-product BP, the message sent from hidden unit to visible unit is the same as Eq. (6.5). The message sent from visible unit $v_i$ to hidden unit $h_j$ is

$$\tilde{m}_{i \to j}(h_j) \propto \exp\left(\tilde{v_i} W_{ij}^{vh} h_j\right) \cdot \frac{\tau(\tilde{v_i})}{m_{j \to i}(\tilde{v_i})}. \tag{6.9}$$

where $\tilde{v_i} = \text{argmax}_{v_i} \tau(v_i)$, and $\tau(v_i)$ is defined in Eq. (6.8). These update equations are repeatedly applied until the values converge (hopefully), or a stopping criterion is satisfied. Then, the pairwise belief on $(v_i, h_j)$ is calculated as,

$$\tau(v_i, h_j) \propto \exp\left(v_i W_{ij}^{vh} h_j\right) \cdot \frac{\tau(v_i)}{m_{j \to i}(v_i)} \cdot \frac{\tau(h_j)}{m_{i \to j}(h_j)}.$$

In the literature, the above algorithm is sometimes referred to as message-passing with division [Koller and Friedman, 2009b]. It is well known that BP on loopy graphs is not guaranteed to converge, although in practice it usually does [Murphy et al., 1999].

### 6.5.2 Matrix-based BP Implementations

Our algorithms use a compact matrix representation. In addition, because the visible unit $v_i$ and hidden unit $h_j$ are all binary, we only need a single scalar to characterize the probabilities of two states. To this end, we denote the "free" belief vectors and matrices as,

$$\begin{aligned}
&\boldsymbol{\tau}^v \in \mathbb{R}^{|\boldsymbol{v}|\times 1}, \text{ where } \tau_i^v = \tau(v_i = 1),\\
&\boldsymbol{\tau}^h \in \mathbb{R}^{|\boldsymbol{h}|\times 1}, \text{ where } \tau_j^h = \tau(h_j = 1),\\
&\Gamma \in \mathbb{R}^{|\boldsymbol{v}|\times|\boldsymbol{h}|}, \text{ where } \Gamma_{ij} = \tau(v_i = 1, h_j = 1).
\end{aligned}$$

Other beliefs can be represented by these "free" beliefs:

$$\begin{aligned}
\tau(v_i = 0) &= 1 - \tau_i^v,\\
\tau(h_j = 0) &= 1 - \tau_j^h,\\
\tau(v_i = 1, h_j = 0) &= \tau_i^v - \Gamma_{ij},\\
\tau(v_i = 0, h_j = 1) &= \tau_j^h - \Gamma_{ij},\\
\tau(v_i = 0, h_j = 0) &= 1 + \Gamma_{ij} - \tau_i^v - \tau_j^h.
\end{aligned}$$

We similarly define the normalized message matrices,

$$\begin{aligned}
&M^{vh} \in \mathbb{R}^{|\boldsymbol{v}|\times|\boldsymbol{h}|}, \quad M_{ij}^{vh} = m_{j\to i}(v_i = 1),\\
&M^{hv} \in \mathbb{R}^{|\boldsymbol{h}|\times|\boldsymbol{v}|}, \quad M_{ji}^{hv} = m_{i\to j}(h_j = 1).
\end{aligned}$$

Thus, $M^{vh}$ represents all the messages sent from $\boldsymbol{h}$ to $\boldsymbol{v}$, and $M^{hv}$ represents all the messages from $\boldsymbol{v}$ to $\boldsymbol{h}$. In the following, we derive the message-passing equations purely based on this compact matrix representation.

**Proposition 6.5.1.** *In both sum-product and mixed-product BP, the update equation for*

*message matrix $M^{vh}$ is*

$$M^{vh} = \sigma\Big(\log\Big(\frac{\exp(W^{vh}) \circ \Lambda_1^{vh} + \Lambda_2^{vh}}{\Lambda_1^{vh} + \Lambda_2^{vh}}\Big)\Big), \tag{6.10}$$
$$\textit{where} \quad \Lambda_1^{vh} = (\mathbf{1}^{hv} - M^{hv})^\top \cdot \mathrm{diag}(\boldsymbol{\tau}^h), \qquad \Lambda_2^{vh} = {M^{hv}}^\top \cdot \mathrm{diag}(\mathbf{1}^h - \boldsymbol{\tau}^h),$$

*where $\mathbf{1}^{hv}$ is a $|\boldsymbol{h}| \times |\boldsymbol{v}|$ matrix of ones, $\mathbf{1}^h$ is a $|\boldsymbol{h}| \times 1$ vector of ones, $\circ$ is the element-wise Hadamard product, and* $\mathrm{diag}(\cdot)$ *extracts the elements in a vector to form a diagonal matrix. The logarithm, fraction and logistic function are all applied in an element-wise manner.*

*Proof.* We first look into the $(i, j)$-th element of matrix $M^{vh}$,

$$\begin{aligned} M_{ij}^{vh} &= \frac{m_{j\to i}(v_i = 1)}{m_{j\to i}(v_i = 1) + m_{j\to i}(v_i = 0)} = \sigma\Big(\log \frac{m_{j\to i}(v_i = 1)}{m_{j\to i}(v_i = 0)}\Big) \\ &= \sigma\Big(\log \frac{\exp(W_{ij}^{vh}) \cdot \frac{\tau_j^h}{M_{ji}^{hv}} + \frac{1-\tau_j^h}{1-M_{ji}^{hv}}}{\frac{\tau_j^h}{M_{ji}^{hv}} + \frac{1-\tau_j^h}{1-M_{ji}^{hv}}}\Big) \text{ (by Eq. (6.5))} \\ &= \sigma\Big(\log \frac{\exp(W_{ij}^{vh}) \cdot (1 - M_{ji}^{hv})\tau_j^h + M_{ji}^{hv}(1 - \tau_j^h)}{(1 - M_{ji}^{hv})\tau_j^h + M_{ji}^{hv}(1 - \tau_j^h)}\Big). \end{aligned}$$

Then, it is easy to verify the update equation Eq. (6.10) for $M^{vh}$. □

Analogously, the update equation for message matrix $M^{hv}$ in sum-product BP is

$$M^{hv} = \sigma\Big(\log\Big(\frac{\exp({W^{vh}}^\top) \circ \Lambda_1^{hv} + \Lambda_2^{hv}}{\Lambda_1^{hv} + \Lambda_2^{hv}}\Big)\Big), \tag{6.11}$$
$$\text{where} \quad \Lambda_1^{hv} = (\mathbf{1}^{vh} - M^{vh})^\top \cdot \mathrm{diag}(\boldsymbol{\tau}^v), \qquad \Lambda_2^{hv} = {M^{vh}}^\top \cdot \mathrm{diag}(\mathbf{1}^v - \boldsymbol{\tau}^v),$$

with $\mathbf{1}^{vh}$ a $|\boldsymbol{v}| \times |\boldsymbol{h}|$ matrix of ones, and $\mathbf{1}^v$ a $|\boldsymbol{v}| \times 1$ vector of ones.

In mixed-product BP, the update equation for message matrix $M^{vh}$ is

$$\tilde{M}^{hv} = \sigma\Big(W^{vh^\top} \cdot \mathrm{diag}(\tilde{\boldsymbol{v}})\Big), \tag{6.12}$$

where $\tilde{v_i} = \mathrm{argmax}_{v_i}\, \tau^v(v_i)$ for all $v_i$.

**Proposition 6.5.2.** *In both sum-product and mixed-product BP, the belief vectors $\boldsymbol{\tau}^v$ and $\boldsymbol{\tau}^h$ can be calculated as,*

$$\boldsymbol{\tau}^v = \sigma\Big(\boldsymbol{b}^1 + \log\Big(\frac{M^{vh}}{\mathbf{1}^{vh} - M^{vh}}\Big) \cdot \mathbf{1}^h\Big), \tag{6.13}$$

$$\boldsymbol{\tau}^h = \sigma\Big(\boldsymbol{b}^2 + \log\Big(\frac{M^{hv}}{\mathbf{1}^{hv} - M^{hv}}\Big) \cdot \mathbf{1}^v\Big), \tag{6.14}$$

*where $\cdot$ is the matrix product.*

*Proof.* We first look into the $i$-th element of $\boldsymbol{\tau}^v$,

$$\begin{aligned}
\tau_i^v &= \frac{\tau(v_i = 1)}{\tau(v_i = 1) + \tau(v_i = 0)} = \frac{1}{1 + \frac{\exp\left(0 + \sum_{j=1}^{|\boldsymbol{h}|} \log m_{j\to i}(v_i = 0)\right)}{\exp\left(b_i + \sum_{j=1}^{|\boldsymbol{h}|} \log m_{j\to i}(v_i = 1)\right)}} \qquad \text{(by Eq. (6.8))} \\
&= \frac{1}{1 + \exp\big\{-b_i^1 - \sum_{j=1}^{|\boldsymbol{h}|}\big(\log M_{ij}^{vh} - \log(1 - M_{ij}^{vh})\big)\big\}} \\
&= \sigma\Big(b_i^1 + \big(\log M_{i\bullet}^{vh} - \log(\mathbf{1}^{\boldsymbol{h}^\top} - M_{i\bullet}^{vh})\big) \cdot \mathbf{1}^h\big\}\Big).
\end{aligned}$$

Then, it is easy to verify the update equation Eq. (6.13) for $\boldsymbol{\tau}^v$. The update of $\boldsymbol{\tau}^h$ in Eq. (6.14) is derived analogously. □

These update equations are repeatedly applied until the stopping criterion is satisfied. After

**Algorithm 6.1** Sum-product BP on RBM

**Input:** $\{W^{vh}, \boldsymbol{b}^1, \boldsymbol{b}^2\}$, number of iterations $T$
**Output:** beliefs $\{\boldsymbol{\tau}^v, \boldsymbol{\tau}^h, \Gamma\}$
initialize message matrices:
$M^{vh} = 0.5 \times \mathbf{1}^{vh}, \quad M^{hv} = 0.5 \times \mathbf{1}^{hv}$;
initialize beliefs:
$\boldsymbol{\tau}^v = \sigma(\boldsymbol{b}^1), \quad \boldsymbol{\tau}^h = \sigma(\boldsymbol{b}^2)$;
**for** $t = 1$ **to** $T$ **do**
send messages from $\boldsymbol{h}$ to $\boldsymbol{v}$:
$\Lambda_1^{vh} = (\mathbf{1}^{hv} - M^{hv})^\top \cdot \mathrm{diag}(\boldsymbol{\tau}^h)$;
$\Lambda_2^{vh} = {M^{hv}}^\top \cdot \mathrm{diag}(\mathbf{1}^h - \boldsymbol{\tau}^h)$;
$M^{vh} = \sigma\Big(\log\Big(\frac{\exp(W^{vh}) \circ \Lambda_1^{vh} + \Lambda_2^{vh}}{\Lambda_1^{vh} + \Lambda_2^{vh}}\Big)\Big)$; (6.10)
$\boldsymbol{\tau}^v = \sigma\Big(\boldsymbol{b}^1 + \log\Big(\frac{M^{vh}}{\mathbf{1}^{vh} - M^{vh}}\Big) \cdot \mathbf{1}^h\Big)$; (6.13)
send messages from $\boldsymbol{v}$ to $\boldsymbol{h}$:
$\Lambda_1^{hv} = (\mathbf{1}^{vh} - M^{vh})^\top \cdot \mathrm{diag}(\boldsymbol{\tau}^v)$;
$\Lambda_2^{hv} = {M^{vh}}^\top \cdot \mathrm{diag}(\mathbf{1}^v - \boldsymbol{\tau}^v)$;
$M^{hv} = \sigma\Big(\log\Big(\frac{\exp({W^{vh}}^\top) \circ \Lambda_1^{hv} + \Lambda_2^{hv}}{\Lambda_1^{hv} + \Lambda_2^{hv}}\Big)\Big)$; (6.11)
$\boldsymbol{\tau}^h = \sigma\Big(\boldsymbol{b}^2 + \log\Big(\frac{M^{hv}}{\mathbf{1}^{hv} - M^{hv}}\Big) \cdot \mathbf{1}^v\Big)$; (6.14);
**end for**
$\Gamma = \frac{\Gamma^{11}}{\Gamma^{11} + \Gamma^{01} + \Gamma^{10} + \Gamma^{00}}$ as defined in Eq. (6.15);

that, the pairwise belief matrix $\Gamma$ can be calculated as (see Appendix C.1 for derivation),

$$
\begin{aligned}
\Gamma &= \frac{\Gamma^{11}}{\Gamma^{11} + \Gamma^{01} + \Gamma^{10} + \Gamma^{00}}, \qquad (6.15)\\
\text{where } \Gamma^{11} &= \exp(W^{vh}) \circ (\boldsymbol{\tau}^v \cdot {\boldsymbol{\tau}^h}^\top) \circ (\mathbf{1}^{vh} - M^{vh}) \circ (\mathbf{1}^{hv} - M^{hv})^\top,\\
\Gamma^{01} &= \big((\mathbf{1}^v - \boldsymbol{\tau}^v) \cdot {\boldsymbol{\tau}^h}^\top\big) \circ M^{vh} \circ (\mathbf{1}^{hv} - M^{hv})^\top,\\
\Gamma^{10} &= \big(\boldsymbol{\tau}^v \cdot (\mathbf{1}^h - \boldsymbol{\tau}^h)^\top\big) \circ (\mathbf{1}^{vh} - M^{vh}) \circ {M^{hv}}^\top,\\
\Gamma^{00} &= \big((\mathbf{1}^v - \boldsymbol{\tau}^v) \cdot (\mathbf{1}^h - \boldsymbol{\tau}^h)\big) \circ M^{vh} \circ {M^{hv}}^\top.
\end{aligned}
$$

We summarize the matrix-based sum-product BP and mixed-product BP in Algorithm 6.1 and 6.2 respectively. It is well known that asynchronous (sequential) BP message updates usually converge much faster than synchronous updates [e.g., Wainwright et al., 2003, Gonzalez et al., 2009]; in Algorithm 6.1 and 6.2, although messages are sent in parallel from all

**Algorithm 6.2** Mixed-product BP on RBM

**Input:** $\{W^{vh}, \boldsymbol{b}^1, \boldsymbol{b}^2\}$, number of iterations $T$
**Output:** beliefs $\{\boldsymbol{\tau}^v, \boldsymbol{\tau}^h, \Gamma\}$
initialize message matrices:
$M^{vh} = 0.5 \times \mathbf{1}^{vh}, \quad M^{hv} = 0.5 \times \mathbf{1}^{hv}$;
initialize beliefs:
$\boldsymbol{\tau}^v = \sigma(\boldsymbol{b}^1), \quad \boldsymbol{\tau}^h = \sigma(\boldsymbol{b}^2)$;
**for** $t = 1$ **to** $T$ **do**
send messages from $\boldsymbol{h}$ to $\boldsymbol{v}$:
$\Lambda_1^{vh} = (\mathbf{1}^{hv} - M^{hv})^\top \cdot \mathrm{diag}(\boldsymbol{\tau}^h)$;
$\Lambda_2^{vh} = {M^{hv}}^\top \cdot \mathrm{diag}(\mathbf{1}^h - \boldsymbol{\tau}^h)$;
$M^{vh} = \sigma\Big(\log\Big(\frac{\exp(W^{vh}) \circ \Lambda_1^{vh} + \Lambda_2^{vh}}{\Lambda_1^{vh} + \Lambda_2^{vh}}\Big)\Big)$; (6.10)
$\boldsymbol{\tau}^v = \sigma\Big(\boldsymbol{b}^1 + \log\Big(\frac{M^{vh}}{\mathbf{1}^{vh} - M^{vh}}\Big) \cdot \mathbf{1}^h\Big)$; (6.13)
send messages from $\boldsymbol{v}$ to $\boldsymbol{h}$:
$\tilde{M}^{hv} = \sigma\Big({W^{vh}}^\top \cdot \mathrm{diag}(\tilde{\boldsymbol{v}})\Big)$; (6.12)
$\boldsymbol{\tau}^h = \sigma\Big(\boldsymbol{b}^2 + \log\Big(\frac{M^{hv}}{\mathbf{1}^{hv} - M^{hv}}\Big) \cdot \mathbf{1}^v\Big)$; (6.14);
**end for**
$\Gamma = \frac{\Gamma^{11}}{\Gamma^{11} + \Gamma^{01} + \Gamma^{10} + \Gamma^{00}}$ as defined in Eq. (6.15);

hidden units to visible units, the bipartite graph structure ensures that these are actually *asynchronous* updates, which helps convergence in practice. Our method is also related to message-passing algorithms designed for other binary networks, such as binary LDPC codes [Kschischang et al., 2001], which parametrize each message by a single real number using a hyperbolic tangent transform. Our algorithm is specially designed for RBM-based models, and significantly speeds up BP by taking advantage of the RBM structure and using only matrix operations.

## 6.6 Experiments

In this section, we compare our methods with state-of-the-art algorithms for learning CRBMs on two datasets: MNIST [LeCun et al., 1998] and Caltech101 Silhouettes [Marlin et al., 2010].

**MNIST:** The MNIST handwritten digits database contains $60,000$ images in the training set and $10,000$ test set images. We randomly select $10,000$ images from training as the validation set. Each image is $28 \times 28$ pixels, thus $|\boldsymbol{v}| = 784$. We binarize the grayscale images by thresholding the pixels at 127, to obtain the clean image $\boldsymbol{v}$. We test two types of structured prediction tasks in our experiment. The first task is image denoising and denoted "*noisy* MNIST", where the noisy image $\boldsymbol{x}$ is obtained by flipping either 10% or 20% of the entries in $\boldsymbol{v}$. The second task is image completion, denoted *occluded* MNIST, where the occluded image $\boldsymbol{x}$ is obtained by setting a random patch within the image $\boldsymbol{v}$ to 0. The patch size is either $8 \times 8$ or $12 \times 12$ pixels. See Figure 6.1 for an illustration.

**Caltech101 Silhouettes:** The Caltech101 Silhouettes dataset has $8,671$ images with $28 \times 28$ binary pixels, where each image represents object silhouette. The dataset is divided into three subsets: $4,100$ examples for training, $2,264$ for validation and $2,307$ for testing. We test both image denoising and image completion tasks. The noisy image $\boldsymbol{x}$ in *noisy* Caltech101 is obtained by flipping 20% of the pixels from the clean $\boldsymbol{v}$, and the occluded image in *occluded* Caltech101 is obtained by setting a random $12 \times 12$ patch to 1.

**Model:** Following [Mnih et al., 2011], we structured the CRBM model with 256 hidden units, giving 1 million parameters in the model. All the learning algorithms are applied to learn this CRBM model. The logistic regression method can be viewed as learning this CRBM with only $W^{vx}$ and $\boldsymbol{b}^v$ non-zero.

**Algorithms:** We train several CRBMs using the state-of-the-art CD methods, including CD-1, CD-10 and CD-PercLoss. We also train models to optimize likelihood (MLE) using mean field (MLE-MF) and sum-product BP (MLE-BP).[3] Finally, we train MSSVM CRBMs using mixed-product BP, and LSSVM CRBMs using max-product BP. A fixed learning rate is selected from the set $\{0.05, 0.02, 0.01, 0.005\}$ using the validation set, and the mini-batch

[3] In previous work [Mnih et al., 2011], MLE-BP was considered impractical on this task due to the efficiency issue.

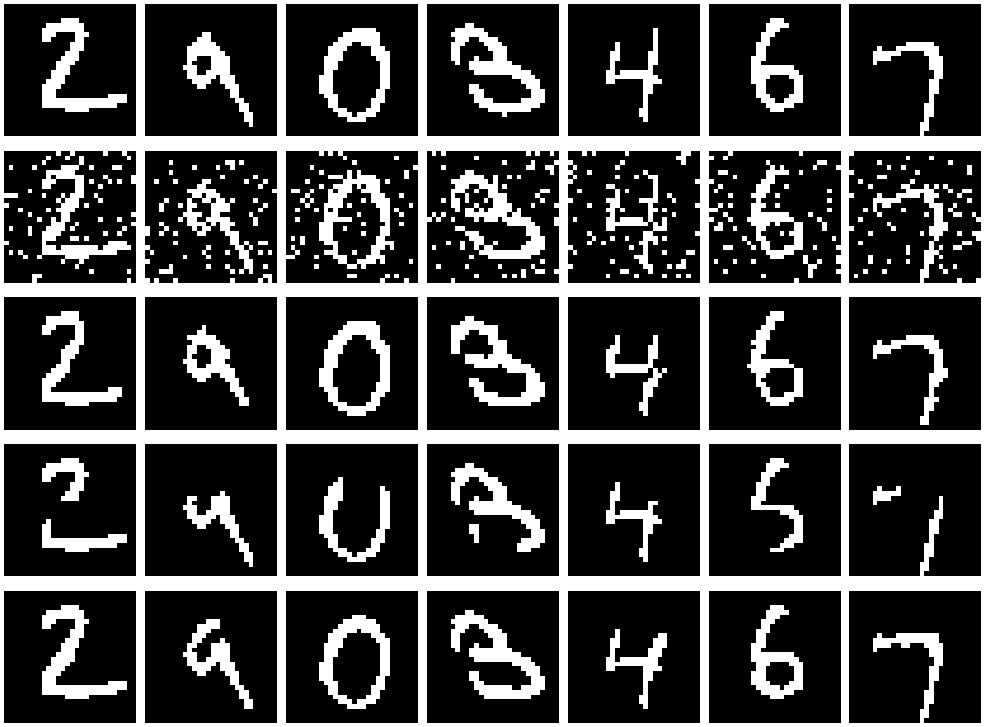


Figure 6.1: (Row 1) 7 original images from the test set. (Row 2) The noisy (10%) images. (Row 3) The images predicted from noisy images. (Row 4) The occluded ($8 \times 8$) images. (Row 5) The images predicted from the occluded images. Rows 3 and 5 use our MLE-BP for learning.

size is selected from the set $\{10, 20, 40, 80, 160\}$. The CD-PercLoss algorithm uses 10-step Gibbs sampling in the stochastic search process. All the CD methods use 200 epochs in training. In contrast, MLE-MF, MLE-BP, MSSVM and LSSVM use 50 epochs, because BP and MF provide a deterministic gradient estimate and larger learning rates can be applied. Early stopping based on the validation error is also used for all methods.[4] We test the learned models of the CD methods and MLE-MF with mean-field predictions; the learned model of MLE-BP with sum-product BP predictions; MSSVM with mixed-product BP; and LSSVM with max-product BP.

**Results:** Table 6.1 shows the percentage of incorrectly labeled pixels on the *noisy* MNIST for different methods. "All" denotes the errors among all pixels and is the main measure-

[4]In experiments, we found that early stopping always worked better than the Frobenius norm regularization.

Table 6.1: Average test error (%) for image denoising on *noisy* MINIST. All denotes the percentage incorrectly labeled pixels among all pixels. Changed denotes the percentage of errors among pixels that were changed by the noise process.

| Dataset / Method | Noisy (10%) All | Noisy (10%) Changed | Noisy (20%) All | Noisy (20%) Changed |
|---|---|---|---|---|
| LR | 1.960 | 12.531 | 4.088 | 12.609 |
| CD-1 | 1.925 | 12.229 | 4.012 | 12.597 |
| CD-10 | 1.816 | 11.103 | 3.995 | 11.271 |
| CD-PercLoss | 1.760 | 11.121 | 3.970 | 10.876 |
| MLE-MF | 1.862 | 11.319 | 3.917 | 10.939 |
| MLE-BP | **1.688** | **10.718** | **3.691** | **10.409** |
| LSSVM | 1.807 | 11.565 | 3.910 | 11.175 |
| MSSVM | 1.751 | 11.023 | 3.804 | 10.627 |

Table 6.2: Average test error (%) for image completion on *occluded* MINIST.

| Dataset / Method | Occluded ($8 \times 8$) All | Occluded ($8 \times 8$) Changed | Occluded ($12 \times 12$) All | Occluded ($12 \times 12$) Changed |
|---|---|---|---|---|
| LR | 1.468 | 61.304 | 3.498 | 53.971 |
| CD-1 | 1.814 | 63.130 | 3.983 | 58.376 |
| CD-10 | 1.707 | 67.925 | 3.921 | 63.237 |
| CD-PercLoss | 1.394 | 45.684 | 3.483 | 35.755 |
| MLE-MF | 1.492 | 49.553 | 3.477 | 40.703 |
| MLE-BP | **1.329** | **39.785** | **3.117** | 36.233 |
| LSSVM | 1.496 | 44.037 | 3.468 | 39.140 |
| MSSVM | 1.391 | 41.829 | 3.273 | **35.712** |

ment. We also report the "Changed" errors among the pixels that were changed by the noise/occlusion process. MLE-BP works best and provides 4% and 7% relative improvement over CD-PercLoss on two datasets with different noise levels. Table 6.2 shows the results on *occluded* MNIST. Here MLE-BP provides 4% and 10% relative improvement over CD-PercLoss on the two datasets, respectively. CD-k is not appropriate for training conditional RBMs and gives unsatisfactory results in both cases. Here MSSVM performs worse than MLE-BP, but better than the other methods in Table 6.1 and 6.2. The image completion task is viewed as more difficult on Changed pixels. However, again training the CRBM with MLE-BP gives very good results; see the last two rows of images in Figure 6.1. Table 6.3

Table 6.3: Average test error (%) for image denoising & completion on Caltech101 Silhouettes dataset.

| Dataset | Noisy (20%) | | Occluded ($12 \times 12$) | |
|---|---|---|---|---|
| Method | All | Changed | All | Changed |
| LR | 5.653 | 11.460 | 4.771 | 16.587 |
| CD-1 | 5.876 | 12.423 | 5.033 | 20.300 |
| CD-10 | 5.736 | 12.013 | 5.149 | 21.087 |
| CD-PercLoss | 5.622 | 10.808 | 5.081 | 15.102 |
| MLE-MF | 5.617 | 11.083 | 4.692 | 15.995 |
| MLE-BP | **5.445** | **10.731** | 4.548 | 16.541 |
| LSSVM | 5.628 | 11.468 | 4.703 | 16.014 |
| MSSVM | 5.549 | 11.389 | **4.534** | **14.918** |

demonstrate the results on Caltech101 Silhouettes; in this setting, MLE-BP and MSSVM perform the best for image denoising and image completion, respectively.

**Discussion:** We include several observations on the interaction of learning and inference algorithms for CRBMs:

- Early on in learning, message passing is fast to converge, typically within $\approx 7$ iterations. As learning continues, the magnitudes of the parameters gradually increase, and it becomes harder for BP to converge quickly. One simple but effective strategy is to set the number of iterations to $T = 7 + \text{epoch}$ (e.g., at epoch 10, $T = 17$). See Figure 6.2 for an illustration of the convergence behavior of BP using this strategy during training. We set the convergence tolerance $\epsilon = 0.001$. The model undergoes a change of convergence behaviour around epoch 3, but we can still make progress as BP converges on the majority of training instances.
- No damping is better. Although message damping can improve the convergence of BP, it always requires more iterations of message-passing and effectively slows down the progress of learning CRBMs.
- The approximate inference algorithms used in learning and test should be matched, which means the inference method (BP or mean-field), number of iterations etc., should

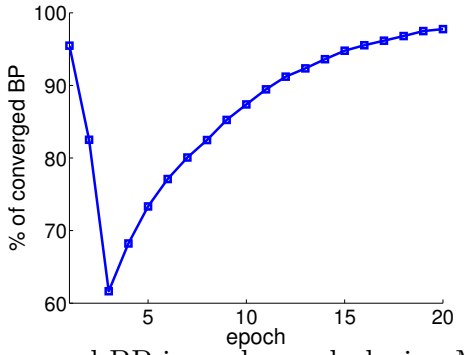


Figure 6.2: Percentage of converged BP in each epoch during MLE-BP training on *occluded* $(8 \times 8)$ MNIST.

be the same. For example, we train our model with MLE-BP using 30 iterations of message-passing ($T = 30$). Then, we test the learned model by mean-field predictions with 30 iterations (abbr. $\text{MF}^{30}$), BP predictions with 30 iterations (abbr. $\text{BP}^{30}$), and BP predictions with 50 iterations and tuned damping (abbr. $\text{BP}^{50}_{\text{damp}}$). We found $\text{BP}^{30}$ are significant better than $\text{MF}^{30}$, and measurable better than $\text{BP}^{50}_{\text{damp}}$. The later one is particularly interesting, because one may figure that more accurate inference is always preferred. Indeed, the learning and approximate inference routine may be deeply coupled. A related theoretical investigation can be found in Wainwright [2006].

- As we discussed in Section 6.4.1, learning CRBMs and vanilla RBMs are quite different in practice. As the literature suggests, in vanilla RBMs we also find that CD methods work better than MLE-BP, and that the latter also requires using and carefully tuning the damping rate.

## 6.7 Conclusions and Future Work

In contrast to past work, we argue that belief propagation can be an excellent choice for learning and inference with RBM-based models in the conditional setting. We present a matrix-based expression of the BP updates for CRBMs, which is scalable to tens of thousands of visible and hidden units. Our implementation takes advantage of the bipartite

graphical structure and uses a compact representation of messages and beliefs. Since it uses only matrix product and element-wise operations, it is highly suited for modern computational architecture (e.g., GPU). We demonstrate that learning CRBMs with sum-product BP (MLE) and mixed-product BP (MSSVM) can provide significantly better results than the state-of-the-art CD methods on structured prediction problems. Future directions include a GPU-based implementation and applying the method to deep probabilistic models, such as deep Boltzmann machines.

Chapter 7

# Conclusions and Future Directions

In this thesis, we research on various challenges for learning and inference in latent variable graphical models.

- In latent variable models (LVMs), the standard MAP prediction is complicated by the fact that, we need to first marginalize out latent variables, leading to the challenging marginal MAP inference. In Chapter 3, we generalize the popular dual-decomposition method for MAP inference, and provide an efficient block coordinate descent algorithm to solve the resulting optimization. Our dual-decomposition method for marginal MAP assumes that the maximization are taken over discrete variables. In many latent variable models, the standard parameter estimation method is marginal MAP estimation, where one need to marginalize out discrete latent variables at first, then take MAP estimation over the continuous parameters. In the future, it is of great interest to develop efficient variational inference algorithm for continuous system, especially for marginal MAP problem.
- When learning LVMs for structured prediction, the inference is used to "fill in" the latent variables. The popular latent structured SVM method impute the latent variables with most probable assignment, which is analogous to hard EM and does not maintain the uncertainty of latent variables. In Chapter 4, we propose the marginal structured SVM which marginalizes latent variables to properly handle their uncertainty in max-

margin learning. Our formulation of marginal structured SVM Eq. (4.3) is based on the margin rescaling framework. As we discussed in Section 2.4.2, another formulation of structured SVM is slack rescaling, which is believed to be more accurate and better-behaved. Thus, it is interesting to derive the slack rescaling version of marginal structured SVM; In particular, one need design an efficient variational approximation to the resulting inference bottleneck [Sarawagi and Gupta, 2008].

- The latent variables are introduced to increase the model's flexibility. This result in the challenges for model selection, that is, how to determine the complexity of latent representation. In restricted Boltzmann machines (RBMs), this correspond to identify the approximate number of hidden units. In Chapter 5, we propose an one-shot Frank-Wolfe learning algorithm for both parameter estimation and model selection in RBM model, which greedily inserts a hidden unit at each iteration and can efficiently identify an appropriate number of hidden units. Our method is directly applicable for conditional RBM model, which will train a sequence of structured classifiers with increasing complexity. It is also very interesting to investigate the Frank-Wolfe learning method for deep generative models, such as deep Boltzmann machine [Salakhutdinov and Hinton, 2009].
- Learning and inference are coupled in latent variable models. In Chapter 6, we study the role of approximate inference in learning with RBMs and conditional RBMs. We provide efficient implementation of belief propagation (BP) algorithms on these models, and we demonstrate that for conditional models and structured prediction, learning RBM-based models with BP can significantly outperforms other popular contrastive divergence methods. It is very interesting to investigate the role of approximate inference in deep Boltzmann machine, and provide a GPU-based implementation in the future.

# Bibliography


D. Ackley, G. Hinton, and T. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive science*, 1985.

K. M. Anstreicher and L. A. Wolsey. Two well-known properties of subgradient optimization. *Mathematical Programming*, 120(1):213–220, 2009.

Ö. Aslan, H. Cheng, X. Zhang, and D. Schuurmans. Convex two-layer modeling. In *NIPS*, 2013.

F. Bach. Breaking the curse of dimensionality with convex neural networks. *arXiv:1412.8690*, 2014.

D. Belanger, D. Sheldon, and A. McCallum. Marginal inference in MRFs using Frank-Wolfe. In *NIPS Workshop on Greedy Optimization, Frank-Wolfe and Friends*, 2013.

Y. Bengio, N. L. Roux, P. Vincent, O. Delalleau, and P. Marcotte. Convex neural networks. In *NIPS*, 2005.

D. P. Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

A. Beygelzimer, E. Hazan, S. Kale, and H. Luo. Online gradient boosting. In *NIPS*, 2015.

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

D. M. Bradley and J. A. Bagnell. Convex coding. In *UAI*, 2009.

K. L. Clarkson. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Transactions on Algorithms*, 2010.

M.-A. Côté and H. Larochelle. An infinite restricted Boltzmann machine. *Neural Computation*, 2015.

R. Dechter. Reasoning with probabilistic and deterministic graphical models: Exact algorithms. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2013.

R. Dechter and I. Rish. Mini-buckets: A general scheme for bounded inference. *JACM*, 2003.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

P. Dhillon, S. S. Keerthi, K. Bellare, O. Chapelle, and S. S. and. Deterministic annealing for semi-supervised structured output learning. In *Proceedings of AISTAT*, pages 299–307, 2012.

J. Domke. Dual decomposition for marginal inference. In *AAAI*, 2011.

A. Doucet, S. Godsill, and C. Robert. Marginal maximum a posteriori estimation using Markov chain Monte Carlo. *Statistics and Computing*, 2002.

G. Elidan, I. McGraw, and D. Koller. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Proceedings of UAI*, 2006.

R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. In *Proceeding of ACM SIGGRAPH*, pages 787–794, 2006.

S. Forouzan and A. T. Ihler. Linear approximation to admm for map inference. In *Proceedings of ACML*, pages 48–61, 2013.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 1956.
J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.
J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 2001.
L. Getoor and B. Taskar. *Introduction to statistical relational learning*. The MIT press, 2007.
A. Globerson and T. Jaakkola. Approximate inference using conditional entropy decompositions. In *AISTATS*, 2007.
A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *NIPS*, 2008.
J. Gonzalez, Y. Low, and C. Guestrin. Residual splash for optimally parallelizing belief propagation. In *Proceedings of AISTATS*, 2009.
I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. Book in preparation for MIT Press, 2016. URL http://www.deeplearningbook.org.
J. Guélat and P. Marcotte. Some comments on Wolfe's away step. *Mathematical Programming*, 1986.
G. H. Hardy, J. E. Littlewood, and G. Polya. *Inequalities*. Cambridge University Press, 1952.
T. Hazan and A. Shashua. Convergent message-passing algorithms for inference over general graphs with convex free energies. In *UAI*, 2008.
T. Hazan and A. Shashua. Norm-product belief propagation: Primal-dual message-passing for approximate inference. *IEEE Transactions on Information Theory*, 2010.
T. Hazan and R. Urtasun. A primal-dual message-passing algorithm for approximated large scale structured prediction. In *Proceedings of NIPS*, 2010.
T. Hazan, J. Peng, and A. Shashua. Tightening fractional covering upper bounds on the partition function for high-order region graphs. In *UAI*, 2012.
G. Hinton. A practical guide to training restricted Boltzmann machines. *UTML TR*, 2010.
G. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 2006a.
G. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 2006b.
G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 2002a.
G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 2002b.
A. Ihler, N. Flerova, R. Dechter, and L. Otten. Join-graph based cost-shifting schemes. In *UAI*, 2012.
A. T. Ihler, W. F. John III, and A. S. Willsky. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 2005.
M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *ICML*, 2013.
J. Jancsary and G. Matz. Convergent decomposition solvers for TRW free energies. In *AISTATS*, 2011.

E. T. Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. The MIT press, 2009a.

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009b.

N. Komodakis and N. Paragios. Beyond loose lp-relaxations: Optimizing MRFs by repairing cycles. In *European conference on computer vision*, 2008.

N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *ICCV*, 2007.

N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *TPAMI*, 2011.

P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with Gaussian edge potentials. In *Proceedings of NIPS*, 2012.

R. G. Krishnan, S. Lacoste-Julien, and D. Sontag. Barrier Frank-Wolfe for marginal inference. In *NIPS*, 2015.

A. Krizhevsky, G. E. Hinton, et al. Factored 3-way restricted Boltzmann machines for modeling natural images. In *AISTATS*, 2010.

F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 2001.

P. Kumar, B. Packer, and D. Koller. Modeling latent variable uncertainty for loss-based learning. In *Proceedings of ICML*, pages 465–472, 2012.

S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-coordinate Frank-Wolfe optimization for structural SVMs. In *ICML*, 2013.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289, 2001.

H. Larochelle and Y. Bengio. Classification using discriminative restricted Boltzmann machines. In *Proceedings of ICML*, 2008.

H. Larochelle, M. Mandel, R. Pascanu, and Y. Bengio. Learning algorithms for the classification restricted Boltzmann machine. *JMLR*, 2012.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

J. Lee, R. Marinescu, R. Dechter, and A. Ihler. From exact to anytime solutions for marginal map. In *AAAI Conference on Artificial Intelligence*, 2016.

A. Levin, Y. Weiss, F. Durand, and W. Freeman. Efficient marginal likelihood optimization in blind deconvolution. In *Proceedings of CVPR*, pages 2657–2664, 2011.

M. H. Li, L. Lin, X. Wang, and T. Liu. Protein–protein interaction site prediction based on conditional random field. *Bioinformatics*, 23, 2007.

X. Li, F. Zhao, and Y. Guo. Conditional restricted Boltzmann machines for multi-label learning with incomplete labels. In *Proceedings of AISTATS*, 2015.

A. Likas, N. Vlassis, and J. J. Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 2003.

Q. Liu. *Reasoning and Decisions in Probabilistic Graphical Models–A Unified Framework*. PhD thesis, University of California, Irvine, 2014.

Q. Liu and A. Ihler. Bounding the partition function using Hölder's inequality. In *ICML*, 2011.

Q. Liu and A. Ihler. Variational algorithms for marginal map. *JMLR*, 14:3165–3200, 2013.

Q. Lou, R. Dechter, and A. Ihler. Anytime anyspace and/or search for bounding the partition function. In *AAAI Conference on Artificial Intelligence*, 2017.

M. Mandel, R. Pascanu, H. Larochelle, and Y. Bengio. Autotagging music with conditional restricted Boltzmann machines. *arXiv:1103.2832*, 2011.

R. Marinescu, R. Dechter, and A. Ihler. AND/OR search for marginal MAP. In *Proceedings of UAI*, 2014.

R. Marinescu, R. Dechter, and A. Ihler. Anytime best+depth-first search for bounding marginal MAP. In *AAAI Conference on Artificial Intelligence*, 2017.

B. M. Marlin, K. Swersky, B. Chen, and N. D. Freitas. Inductive principles for restricted Boltzmann machine learning. In *AISTATS*, 2010.

D. Maua and C. de Campos. Anytime marginal maximum a posteriori inference. In *ICML*, 2012.

C. Meek and Y. Wexler. Approximating max-sum-product problems using multiplicative error bounds. *Bayesian Statistics*, 2011.

T. Meltzer, A. Globerson, and Y. Weiss. Convergent message passing algorithms: a unifying view. In *UAI*, 2009.

O. Meshi and A. Globerson. An alternating direction method for dual MAP LP relaxation. In *ECML/PKDD*, 2011.

O. Meshi, D. Sontag, T. Jaakkola, and A. Globerson. Learning efficiently with approximate inference via dual losses. In *ICML*, 2010.

K. Miller, P. Kumar, B. Packer, D. Goodman, and D. Koller. Max-margin min-entropy models. In *Proceedings of AISTATS*, pages 779–787, 2012.

V. Mnih, H. Larochelle, and G. E. Hinton. Conditional restricted Boltzmann machines for structured output predition. In *Proceedings of UAI*, 2011.

J. Mooij. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *JMLR*, 2010.

J. Mooij and H. Kappen. On the properties of the bethe approximation and loopy belief propagation on binary networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2005.

K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of UAI*, 1999.

K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *ICML*, 2010.

E. Nalisnick and S. Ravi. Infinite dimensional word embeddings. *arXiv:1511.05392*, 2015.

J. Naradowsky, S. Riedel, and D. Smith. Improving NLP through marginalization of hidden syntactic structure. In *Proceeding of EMNLP*, pages 810–820, 2012.

S. Nowozin and G. Bakir. A decoupled approach to exemplar-based unsupervised learning. In *ICML*, 2008.

S. Nowozin and C. Lampert. Structured prediction and learning in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6, 2011.
P. Orbanz and Y. W. Teh. Bayesian nonparametric models. In *Encyclopedia of Machine Learning*. Springer, 2011.
J. Park and A. Darwiche. Solving MAP exactly using systematic search. In *UAI*, 2003.
J. Park and A. Darwiche. Complexity results and approximation strategies for MAP explanations. *JAIR*, 2004.
J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.
C. Peterson and J. R. Anderson. A mean field theory learning algorithm for neural networks. *Complex Systems*, 1987.
W. Ping and A. Ihler. Belief propagation in conditional RBMs for structured prediction. In *In submission*, 2017.
W. Ping, Q. Liu, and A. Ihler. Marginal structured SVM with hidden variables. In *Proceedings of ICML*, 2014.
W. Ping, Q. Liu, and A. Ihler. Decomposition bounds for marginal MAP. In *Advances in Neural Information Processing Systems*, 2015.
W. Ping, Q. Liu, and A. Ihler. Learning infinite RBMs with Frank-Wolfe. In *Advances in Neural Information Processing Systems*, 2016.
A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *Proceedings of NIPS*, pages 1097–1104, 2004.
A. Quattoni, S. Wang, L. Morency, M. Collins, and T. Darrell. Hidden conditional random fields. *IEEE Transactions on PAMI*, 29:1848–1852, 2007a.
A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell. Hidden conditional random fields. *IEEE Transactions on PAMI*, 2007b.
N. Ratliff, J. A. Bagnell, and M. Zinkevich. (Online) Subgradient methods for structured prediction. In *Proceedings of AISTATS*, pages 380–387, 2007.
N. Ruozzi and S. Tatikonda. Message-passing algorithms: Reparameterizations and splittings. *IEEE Transactions on Information Theory*, 2013.
R. Salakhutdinov and G. E. Hinton. Deep Boltzmann machines. In *AISTATS*, 2009.
R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. In *ICML*, 2008.
R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of ICML*, pages 791–798, 2007a.
R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *ICML*, 2007b.
R. Samdani, M. Chang, and D. Roth. Unified expectation maximization. In *Proceedings of NAACL*, pages 688–698, 2012.
S. Sarawagi and R. Gupta. Accurate max-margin training for structured output spaces. In *Proceedings of ICML*, pages 888–895, 2008.
K. Sato and Y. Sakakibara. RNA secondary structural alignment with conditional random fields. *Bioinformatics*, 21, 2005.
M. Schmidt. minFunc: unconstrained differentiable multivariate optimization in Matlab. 2005. URL http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html.

A. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Efficient structured prediction with latent variables for general graphical models. In *Proceedings of ICML*, pages 959–966, 2012.

P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document, 1986.

D. Sontag and T. Jaakkola. Tree block coordinate descent for MAP in graphical models. *AISTATS*, 2009.

D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening LP relaxations for MAP using message passing. In *UAI*, 2008.

D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. *Optimization for Machine Learning*, 2011.

I. Sutskever and T. Tieleman. On the convergence properties of contrastive divergence. In *Proceedings of AISTATS*, 2010.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Proceedings of NIPS*, 2003.

G. W. Taylor, G. E. Hinton, and S. Roweis. Modeling human motion using binary latent variables. In *Proceedings of NIPS*, 2006a.

G. W. Taylor, G. E. Hinton, and S. Roweis. Modeling human motion using binary latent variables. In *NIPS*, 2006b.

T. Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *ICML*, 2008.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484, 2005.

J. Verbeek and B. Triggs. Scene segmentation with CRFs learned from partially labeled images. In *Proceedings of NIPS*, pages 1553–1560, 2007.

J. J. Verbeek, N. Vlassis, and B. Kröse. Efficient greedy learning of Gaussian mixture models. *Neural Computation*, 2003.

M. Volkovs, H. Larochelle, and R. S. Zemel. Loss-sensitive training of probabilistic conditional random fields. *Technical report*, 2011.

M. Wainwright. Estimating the wrong graphical model: Benefits in the computation-limited setting. *JMLR*, 2006.

M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 2008.

M. Wainwright, T. Jaakkola, and A. Willsky. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, 2005.

M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on information theory*, 49(5):1120–1146, 2003.

S. Wang, A. Quattoni, L. Morency, and D. Demirdjian. Hidden conditional random fields for gesture recognition. In *Proceedings of CVPR*, pages 1521–1527, 2006.

Y. Wang and G. Mori. Max-margin hidden conditional random fields for human action recognition. In *Proceedings of CVPR*, pages 872–879, 2009.

Y. Weiss, C. Yanover, and T. Meltzer. MAP estimation, linear programming and belief propagation with convex free energies. In *UAI*, 2007.

M. Welling and Y. W. Teh. Approximate inference in Boltzmann machines. *Artificial Intelligence*, 2003.

M. Welling, R. S. Zemel, and G. E. Hinton. Self supervised boosting. In *NIPS*, 2002.

T. Werner. A linear programming approach to max-sum problem: A review. *TPAMI*, 2007.

J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *Proceedings of ICCV*, pages 1800–1807, 2005.

Y. Xu, D. Rockmore, and A. Kleinbaum. Hyperlink prediction in hypernetworks using latent social features. In *Discovery Science*, pages 324–339, 2013.

Z. Xu, Z. Hong, Y. Zhang, J. Wu, A. C. Tsoi, and D. Tao. Multinomial latent logistic regression for image understanding. *IEEE Transactions on Image Processing*, 25(2):973–987, 2016.

Y. Xue, S. Ermon, C. P. Gomes, B. Selman, et al. Solving marginal map problems with np oracles and parity constraints. In *Advances In Neural Information Processing Systems*, 2016.

J. Yang, S. Safar, and M.-H. Yang. Max-margin Boltzmann machines for object segmentation. In *Proceedings of CVPR*, 2014.

J. Yarkony, C. Fowlkes, and A. Ihler. Covering trees and lower-bounds on quadratic assignment. In *CVPR*, 2010.

M. Yasuda and K. Tanaka. Approximate learning algorithm in Boltzmann machines. *Neural computation*, 2009.

J. S. Yedidia, W. T. Freeman, Y. Weiss, et al. Generalized belief propagation. In *NIPS*, volume 13, pages 689–695, 2000.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 2005.

A. Yessenalina, Y. Yue, and C. Cardie. Multi-level structured models for document-level sentiment classification. In *Proceedings of EMNLP*, pages 1046–1056, 2010.

C. Yu and T. Joachims. Learning structural SVMs with latent variables. In *Proceedings of ICML*, pages 1169–1176, 2009a.

C.-N. J. Yu and T. Joachims. Learning structural svms with latent variables. In *Proceedings of ICML*, 2009b.

C. Yuan and E. Hansen. Efficient computation of jointree bounds for systematic map search. *IJCAI*, 2009.

C. Yuan, T. Lu, and M. Druzdzel. Annealed MAP. In *UAI*, 2004.

A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15: 915–936, 2003.

Y. Zhang and T. Chen. Efficient inference for fully-connected CRFs with stationarity. In *Proceedings of CVPR*, pages 582–589, 2012.

L. Zhu, Y. Chen, A. Yuille, and W. Freeman. Latent hierarchical structural learning for object detection. In *Proceedings of CVPR*, pages 1062–1069, 2010.

# Appendix A

# Derivations and Proofs for Dual-decomposition Bounds

## A.1 Dual Representations

### A.1.1 Proof of Thereom 4.2

We now prove the following dual representation of our bound,

$$\min_{\delta} L(\delta, \mathbf{w}) = \max_{b \in \mathbb{L}(G)} \big\{ \langle \theta, b \rangle + \sum_{i \in V} w_i H(x_i; b_i) + \sum_{\alpha \in \mathcal{F}} \sum_{i \in \alpha} w_i^{\alpha} H(x_i | x_{\mathrm{pa}_i^{\alpha}}; b_{\alpha}) \big\}, \tag{A.1}$$

where $\mathbb{L}(G) = \{b \mid b_i(x_i) = \sum_{x_{\alpha \setminus i}} b_{\alpha}(x_{\alpha}),\ \sum_{x_i} b_i(x_i) = 1\}$ is the local consistency polytope, and $\mathrm{pa}_i^{\alpha} = \{j \in \alpha | j \succ i\}$. Thereom 3.3.2 follows directly from (A.1).

*Proof.* In our primal bound $L(\delta, \mathbf{w})$ (3.2) in the main text, we let $\widetilde{\theta}_i(x_i) = \theta_i(x_i) + \sum_{\alpha \in N_i} \delta_i^{\alpha}(x_i)$ (we add dummy singleton $\theta_i(x_i) \equiv 0$), and $\widetilde{\theta}_{\alpha}(x_{\alpha}) = \theta_{\alpha}(x_{\alpha}) - \sum_{i \in \alpha} \delta_i^{\alpha}(x_i)$, then the bound can be rewritten as,

$$L(\widetilde{\theta}, \mathbf{w}) = \sum_{i \in V} \log \sum_{x_i}^{w_i} \exp\big[\widetilde{\theta}_i(x_i)\big] + \sum_{\alpha \in \mathcal{F}} \log \sum_{x_{\alpha}}^{\mathbf{w}^{\alpha}} \exp\big[\widetilde{\theta}_{\alpha}(x_{\alpha})\big].$$

Note, for any assignment $x$, we have $\sum_i \widetilde{\theta}_i(x_i) + \sum_\alpha \widetilde{\theta}_\alpha(x_\alpha) = \sum_\alpha \theta_\alpha(x_\alpha)$.

By applying the dual form of the powered sum (2.22) on each node and clique respectively, we have

$$L(\widetilde{\theta}, \mathbf{w}) = \sum_{i \in V} \max_{b_i \in \mathbb{M}(G_i)} \left\{ \langle \widetilde{\theta}_i, b_i \rangle + w_i H(x_i; b_i) \right\} + \sum_{\alpha \in \mathcal{F}} \max_{b_\alpha \in \mathbb{M}(G_\alpha)} \left\{ \langle \widetilde{\theta}_\alpha, b_\alpha \rangle + \sum_{i \in \alpha} w_i^\alpha H(x_i | x_{\mathrm{pa}_i^\alpha}; b_\alpha) \right\},$$

where $\mathrm{pa}_i^\alpha$ is the set of variables in $\alpha$ that are summed out later than $i$, $\mathbb{M}(G_i)$ and $\mathbb{M}(G_\alpha)$ are the marginal polytopes on singleton node $i$ and clique $\alpha$ respectively, which enforce $\{b_i, b_\alpha\}$ to be properly normalized.

The above equation can be more compactly rewritten as,

$$L(\widetilde{\theta}, \mathbf{w}) = \max_{b \in \widetilde{\mathbb{M}}} \left\{ \langle \widetilde{\theta}, b \rangle + \sum_{i \in V} w_i H(x_i; b_i) + \sum_{\alpha \in \mathcal{F}} \sum_{i \in \alpha} w_i^\alpha H(x_i | x_{\mathrm{pa}_i^\alpha}; b_\alpha) \right\},$$

where $\widetilde{\mathbb{M}} = \{\mathbb{M}(G_i), \mathbb{M}(G_\alpha) \mid \forall\, i \in V, \alpha \in \mathcal{F}\}$, and the elements $\{b_i, b_\alpha\}$ of $b$ are independently optimized.

Then, by tightening *reparameterization* $\widetilde{\theta} = \{\widetilde{\theta}_i, \widetilde{\theta}_\alpha\}$, we have

$$\min_{\widetilde{\theta}} L(\widetilde{\theta}, \mathbf{w}) = \max_{b \in \widetilde{\mathbb{M}}} \min_{\widetilde{\theta}} \left\{ \langle \widetilde{\theta}, b \rangle + \sum_{i \in V} w_i H(x_i; b_i) + \sum_{\alpha \in \mathcal{F}} \sum_{i \in \alpha} w_i^\alpha H(x_i | x_{\mathrm{pa}_i^\alpha}; b_\alpha) \right\}$$

where the order of min and max are commuted according to the strong duality (it's convex with $\widetilde{\theta}$, and concave with $b$).

The inner minimization $\min_{\widetilde{\theta}} \langle \widetilde{\theta}, b \rangle$ is a linear program, and it turns out can be solved analytically. To see this, given the relationship between $\widetilde{\theta}$ and $\delta$, we rewrite the linear program

as

$$\min_{\widetilde{\theta}} \langle \widetilde{\theta}, b \rangle = \min_{\delta} \Big\{ \langle \theta, b \rangle + \sum_{i \in V} \sum_{x_i} \sum_{\alpha \in N_i} \delta_i^{\alpha}(x_i) b_i(x_i) - \sum_{\alpha \in \mathcal{F}} \sum_{x_\alpha} \sum_{i \in \alpha} \delta_i^{\alpha}(x_i) b_\alpha(x_\alpha) \Big\},$$
$$= \min_{\delta} \Big\{ \langle \theta, b \rangle + \sum_{(i,\alpha)} \sum_{x_i} \delta_i^{\alpha}(x_i) \Big( b_i(x_i) - \sum_{x_{\alpha \backslash i}} b_\alpha(x_\alpha) \Big) \Big\}.$$

Then, it is easy to observe that the linear program is either equal to $\langle \theta, b \rangle$ only if $b$ satisfy the marginalization constraint $\sum_{x_{\alpha \backslash i}} b_\alpha(x_\alpha) = b_i(x_i)$ for $\forall (i, \alpha)$, or it will become negative infinity. Considering the outer maximization, we have

$$\min_{\widetilde{\theta}} L(\widetilde{\theta}, \mathbf{w}) = \max_{b \in \mathbb{L}(G)} \big\{ \langle \theta, b \rangle + \sum_{i \in V} w_i H(x_i; b_i) + \sum_{\alpha \in \mathcal{F}} \sum_{i \in \alpha} w_i^{\alpha} H(x_i | x_{\mathrm{pa}_i^{\alpha}}; b_\alpha) \big\},$$

where $\mathbb{L}(G)$ is the local consistency polytope that is obtained by enforcing both $\widetilde{\mathbb{M}}$ and the marginalization constraint. □

### A.1.2 Matching Our Bound to WMB

After the weights are optimized, our GDD bound matches to WMB bound with optimum weights. A simple weight initialization method matches our bound to WMB with uniform weights on each mini-bucket, which often gives satisfactory result; a similar procedure can be used to match the bound with more general weights as in Section 3.5. We first set $w_i = 0$ for all nodes $i$. We then visit the nodes $x_i$ along the elimination order $\mathbf{o} = [x_1, x_2, \cdots, x_n]$, and divide $x_i$'s neighborhood cliques $N_i = \{\alpha | \alpha \ni i\}$ into two groups: (1) the *children cliques* in which all $x_{\alpha \backslash i}$ have already been eliminated, that is, $N_i^{ch} = \{\alpha \mid \forall j \in \alpha \backslash i,\ j \prec i \text{ in } \mathbf{o}\}$; (2) the other, *parent cliques* $N_i^{pa} = \{\alpha \mid \exists j \in \alpha \backslash i,\ j \succ i \text{ in } \mathbf{o}\}$. We set $w_i^{\alpha} = 0$ for all the children cliques ($\alpha \in N_i^{ch}$), and uniformly split the weights, that is, $w_i^{\alpha} = \tau_i / |N_i^{pa}|$, across the parent cliques.

## A.2 Proof of Therom 5.1

*Proof.* For each $\delta_i^\alpha(x_i)$, the involved terms in $L(\delta, \mathbf{w})$ are $L_i^\alpha(\delta) = \Phi_{w_i}(\delta) + \Phi_{\mathbf{w}^\alpha}(\delta)$, where

$$\Phi_{w_i}(\delta) = \log \overline{\sum_{x_i}^{w_i}} \exp\Big[\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)\Big], \quad \Phi_{\mathbf{w}^\alpha}(\delta) = \log \overline{\sum_{x_\alpha}^{\mathbf{w}^\alpha}} \exp\Big[\theta_\alpha(x_\alpha) - \sum_{i \in \alpha} \delta_i^\alpha(x_i)\Big].$$

Our result follows by showing that

$$\begin{aligned} &\frac{\partial \Phi_{w_i}(\delta)}{\partial \delta_i^\alpha(x_i)} = \mu_i(x_i) && \text{and} \quad \frac{\partial \Phi_{w_i}(\delta)}{\partial w_i} = H(x_i; \mu_i), \\ &\frac{\partial \Phi_{\mathbf{w}^\alpha}(\delta)}{\partial \delta_i^\alpha(x_i)} = -\sum_{x_{\alpha \setminus i}} \mu_\alpha(x_\alpha) && \text{and} \quad \frac{\partial \Phi_{\mathbf{w}^\alpha}(\delta)}{\partial w_i^\alpha} = H(x_i | x_{i+1:c}; \mu_\alpha). \end{aligned}$$

The gradient of $\Phi_{w_i}(\delta)$ is straightforward to calculate,

$$\frac{\partial \Phi_{w_i}}{\partial \delta_i^\alpha(x_i)} = \frac{\partial}{\partial \delta_i^\alpha(x_i)} \Big( w_i \log \sum_{x_i} \exp\Big[\frac{\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)}{w_i}\Big]\Big) = \frac{\exp\big[\frac{\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)}{w_i}\big]}{Z_{w_i}} = \mu_i(x_i), \quad \text{(A.2)}$$

where $Z_{w_i} = \sum_{x_i} \exp\big[\frac{\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)}{w_i}\big]$, and

$$\begin{aligned} \frac{\partial \Phi_{w_i}}{\partial w_i} &= \log Z_{w_i} + w_i \cdot \frac{1}{Z_{w_i}} \cdot \sum_{x_i} \Big\{ \exp\Big[\frac{\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)}{w_i}\Big] \cdot \frac{\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)}{-w_i^2} \Big\} \\ &= \log Z_{w_i} - \sum_{x_i} \Big\{ \mu_i(x_i) \cdot \frac{\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)}{w_i} \Big\} \\ &= -\sum_{x_i} \Big\{ \mu_i(x_i) \cdot \Big[ \frac{\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)}{w_i} - \log Z_{w_i} \Big] \Big\} = H(x_i; \mu_i). \end{aligned}$$

The gradient of $\Phi_{\mathbf{w}^\alpha}(\delta)$ is more involved; see Proposition A.3.1 for a detailed derivation.

Given the gradients, the moment matching condition (3.7) in Therom 3.4.1 obviously holds.

We now prove the entropy matching condition in (3.9). The constraint optimization is

$$\min_{\mathbf{w}} L(\mathbf{w}), \quad \text{s.t. } w_i \geq 0,\ w_i^\alpha \geq 0,\ w_i + \sum_\alpha w_i^\alpha = \tau_i.$$

Note, when $\tau_i = 0$, the optimization is trival, so we simply assume $\tau_i > 0$. We frame the Lagrangian as

$$\mathcal{LG}(\mathbf{w}, \boldsymbol{\lambda}, \boldsymbol{g}) \stackrel{\text{def}}{=\!=} L(\mathbf{w}) + \sum_i \lambda_i \big(w_i + + \sum_\alpha w_i^\alpha - \tau_i\big) + \sum_i g_i w_i + \sum_{(i,\alpha)} g_i^\alpha w_i^\alpha.$$

Note $\boldsymbol{g} \leq 0$ (dual feasibility), otherwise $\max_{\boldsymbol{g},\boldsymbol{\lambda}} \mathcal{LG}(\mathbf{w}, \boldsymbol{\lambda}, \boldsymbol{g})$ will approach infinity. The KKT conditions are

$$\text{stationarity:} \quad \frac{\partial \mathcal{LG}}{\partial w_i} = H(x_i; \mu_i) + \lambda_i + g_i = 0, \tag{A.3}$$

$$\frac{\partial \mathcal{LG}}{\partial w_i^\alpha} = H(x_i | x_{i+1:c}; \mu_\alpha) + \lambda_i + g_i^\alpha = 0, \tag{A.4}$$

$$\text{complementary slackness:} \quad g_i w_i = 0, \quad g_i^\alpha w_i^\alpha = 0. \tag{A.5}$$

We multiply $w_i$ and $w_i^\alpha$ to (A.3) and (A.4) respectively, then we can eliminate the KKT multipliers $g_i$ and $g_i^\alpha$ by applying the complementary slackness (A.5),

$$w_i H(x_i; \mu_i) + w_i \lambda_i = 0, \tag{A.6}$$

$$w_i^\alpha H(x_i | x_{i+1:c}; \mu_\alpha) + w_i^\alpha \lambda_i = 0. \tag{A.7}$$

By summing (A.7) over all $\alpha \in N_i$ and adding (A.6), we can solve the multiplier $\lambda_i$ as

$$\lambda_i = -\frac{w_i H(x_i; \mu_i) + \sum_\alpha w_i^\alpha H(x_i | x_{i+1:c}; \mu_\alpha)}{\tau_i} = -\bar{H}_i.$$

We plug it into (A.6) and (A.7), and obtain the entropy matching condition (3.9) in the Therom 3.4.1. □

## A.3 Derivations of Gradient

**Proposition A.3.1.** *Given a weight vector* $\mathbf{w}^\alpha = [w_1^\alpha, \cdots, w_i^\alpha, \cdots, w_c^\alpha]$ *associated with variables* $x_\alpha = \{x_1, \cdots, x_i, \cdots, x_c\}$ *on clique* $\alpha$*, where* $c = |\alpha|$ *the power sum over clique* $\alpha$ *is,*

$$\Phi_{\mathbf{w}^\alpha}(\delta) = \log \sum_{x_\alpha}^{\mathbf{w}^\alpha} \exp\Big[\theta_\alpha(x_\alpha) - \sum_{i\in\alpha} \delta_i^\alpha(x_i)\Big] = \log \sum_{x_c}^{w_c^\alpha} \cdots \sum_{x_i}^{w_i^\alpha} \cdots \sum_{x_1}^{w_1^\alpha} \exp\Big[\theta_\alpha(x_\alpha) - \sum_{i\in\alpha} \delta_i^\alpha(x_i)\Big].$$

*We recursively denote* $Z_i$ *as the partial power sum up to* $x_{1:i}$*,*

$$Z_0(x_\alpha) = \exp\Big[\theta_\alpha(x_\alpha) - \sum_{i\in\alpha} \delta_i^\alpha(x_i)\Big] \quad \textit{and} \quad Z_i(x_{i+1:c}) = \sum_{x_i}^{w_i^\alpha} Z_{i-1}(x_{i:c}), \tag{A.8}$$

*thus* $\log Z_c = \Phi_{\mathbf{w}^\alpha}$*. We also denote the "pseudo marginal" (or, belief) on* $x_\alpha$*,*

$$\mu_\alpha(x_\alpha) = \prod_{i=1}^{c} \mu_\alpha(x_i|x_{i+1:c}); \quad \mu_\alpha(x_i|x_{i+1:c}) = \Big(\frac{Z_{i-1}(x_{i:c})}{Z_i(x_{i+1:c})}\Big)^{1/w_i^\alpha},$$

*and it is easy to verify that* $\mu_\alpha(x_i|x_{i+1:c})$ *and* $\mu_\alpha(x_\alpha)$ *are normalized.*

*Then, the derivative of* $\Phi_{\mathbf{w}^\alpha}$ *w.r.t.* $\delta_i^\alpha(x_i)$ *can be written by beliefs,*

$$\frac{\partial \Phi_{\mathbf{w}^\alpha}}{\partial \delta_i^\alpha(x_i)} = -\mu_\alpha(x_i) = -\sum_{x_{\alpha\setminus i}} \mu_\alpha(x_\alpha) = -\sum_{x_c} \cdots \sum_{x_{i+1}} \prod_{j=i}^{c} \mu_\alpha(x_j|x_{j+1:c}) \tag{A.9}$$

*In addition, the derivative of* $\Phi_{\mathbf{w}^\alpha}$ *w.r.t.* $w_i^\alpha$ *is the conditional entropy,*

$$\frac{\partial \Phi_{\mathbf{w}^\alpha}}{\partial w_i^\alpha} = H(x_i|x_{i+1:c}; \mu_\alpha(x_\alpha)) = -\sum_{x_\alpha} \mu_\alpha(x_\alpha) \log \mu_\alpha(x_i|x_{i+1:c}) \tag{A.10}$$

*Proof.* Denote the reparameterization on clique $\alpha$ as $\widetilde{\theta}_\alpha(x_\alpha) = \theta_\alpha(x_\alpha) - \sum_{i\in\alpha} \delta_i^\alpha(x_i)$.

From the recursive definition of $Z_i(x_{i+1:c})$ (A.8), we have the following recursive rule for

gradient,

$$\begin{aligned}\frac{\partial \log Z_i(x_{i+1:c})}{\partial \widetilde{\theta}_\alpha(x_\alpha)} &= \frac{\partial}{\partial \widetilde{\theta}_\alpha(x_\alpha)}\Big(w_i^\alpha \log \sum_{x_i}\big[Z_{i-1}(x_{i:c})\big]^{1/w_i^\alpha}\Big) \\ &= w_i^\alpha \cdot \frac{\frac{1}{w_i^\alpha}\cdot Z_{i-1}(x_{i:c})^{\frac{1}{w_i^\alpha}}}{\sum_{x_i}\big[Z_{i-1}(x_{i:c})\big]^{\frac{1}{w_c^\alpha}}} \cdot Z_{i-1}(x_{i:c})^{-1} \cdot \frac{\partial Z_{i-1}(x_{i:c})}{\partial \widetilde{\theta}_\alpha(x_\alpha)} \\ &= \mu_\alpha(x_i|x_{i+1:c}) \cdot \frac{\partial \log Z_{i-1}(x_{i:c})}{\partial \widetilde{\theta}_\alpha(x_\alpha)}. \end{aligned} \tag{A.11}$$

It should be noted, implicitly, $x_{i+1:c}$ within $\widetilde{\theta}_\alpha(x_\alpha)$ should take the same value as $x_{i+1:c}$ in $\log Z_i(x_{i+1:c})$, otherwise, the derivative will equal 0.

As a result, we can calculate the derivatives of $\Phi_{\mathbf{w}^\alpha}(\widetilde{\theta}_\alpha)$ w.r.t. $\widetilde{\theta}_\alpha(x_\alpha)$ recursively as,

$$\frac{\partial \Phi_{\mathbf{w}^\alpha}}{\partial \widetilde{\theta}_\alpha(x_\alpha)} = \frac{\partial \log Z_c}{\partial \widetilde{\theta}_\alpha(x_\alpha)} = \mu_\alpha(x_c) \cdot \frac{\partial \log Z_{c-1}(x_c)}{\partial \widetilde{\theta}_\alpha(x_\alpha)} = \cdots = \prod_{i=1}^{c} \mu_\alpha(x_i|x_{i+1:c}) = \mu_\alpha(x_\alpha). \tag{A.12}$$

By the chain rule,

$$\frac{\partial \Phi_{\mathbf{w}^\alpha}}{\partial \delta_i^\alpha(x_i)} = \sum_{x_{\alpha\setminus i}} \frac{\partial \Phi_{\mathbf{w}^\alpha}}{\partial \widetilde{\theta}_\alpha(x_i, x_{\alpha\setminus i})} \cdot \frac{\partial \widetilde{\theta}_\alpha(x_i, x_{\alpha\setminus i})}{\partial \delta_i^\alpha(x_i)} = -\sum_{x_{\alpha\setminus i}} \mu_\alpha(x_\alpha),$$

then (A.9) has been proved.

Applying the variational form of powered-sum (2.22) to $\Phi_{\mathbf{w}^\alpha}$, we have

$$\Phi_{\mathbf{w}^\alpha}(\widetilde{\theta}_\alpha) = \max_{b_\alpha \in \mathbb{M}_\alpha(G)} \big\{\langle \widetilde{\theta}_\alpha, b_\alpha\rangle + \sum_i w_i^\alpha H(x_i|x_{i+1:n}; b_\alpha)\big\}.$$

According to Danskin's theorem, the derivative $\frac{\partial \Phi_{\mathbf{w}^\alpha}}{\partial \widetilde{\theta}_\alpha(x_\alpha)} = b_\alpha^*(x_\alpha)$, which is the optimum of RHS. Combined with (A.12), we have $b_\alpha^* = \mu_\alpha$ immediately, and the derivative w.r.t. $w_i^\alpha$ is,

$$\frac{\partial \Phi_{\mathbf{w}^\alpha}}{\partial w_i^\alpha} = H(x_i|x_{i+1:c}; \mu_\alpha(x_\alpha)),$$

then (A.10) has been proved. □

## A.4 Derivation of Hessian

**Proposition A.4.1.** *Given* $\Phi_{w_i}(\delta) = \log \sum_{x_i}^{w_i} \exp\Big[\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)\Big]$ *in section A.2, the Hessian matrix is,*

$$\forall \alpha, \beta \in N_i, \quad \frac{\partial^2 \Phi_{w_i}(\delta)}{\partial \delta_i^\alpha(x_i) \partial \delta_i^\beta(x_i')} = \frac{1}{w_i}\Big[\mathbb{I}(x_i = x_i') \cdot \mu_i(x_i) - \mu_i(x_i) \cdot \mu_i(x_i')\Big], \tag{A.13}$$

$$\textit{i.e. in matrix form} \quad \frac{\partial^2 \Phi_{w_i}(\delta)}{\partial \delta_i^\alpha \partial \delta_i^\beta} = \frac{1}{w_i}\Big[\operatorname{diag}(\mu_i) - \mu_i \mu_i^T\Big].$$

*where* $\mu_i(x_i) = \frac{\exp\left[\frac{\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)}{w_i}\right]}{Z_i}$ *and* $Z_i = \sum_{x_i} \exp\left[\frac{\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)}{w_i}\right]$.

*Proof.* Starting from the first-order derivative (A.2), when $x_i = x_i'$,

$$\begin{aligned}
\frac{\partial^2 \Phi_{w_i}(\delta)}{\partial \delta_i^\alpha(x_i) \partial \delta_i^\beta(x_i)} &= \frac{\exp\Big[\frac{1}{w_i}\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)\Big] \cdot \frac{1}{w_i} \cdot Z_i - \exp\Big[\frac{1}{w_i}\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)\Big] \cdot \frac{\partial Z_i}{\partial \delta_i^\beta(x_i)}}{Z_i^2} \\
&= \frac{1}{w_i}\mu_i(x_i) - \frac{\exp\Big[\frac{1}{w_i}\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)\Big] \cdot \exp\Big[\frac{1}{w_i}\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)\Big]}{Z_i^2} \\
&= \frac{1}{w_i}\Big[\mu_i(x_i) - \mu_i(x_i)^2\Big],
\end{aligned}$$

when $x_i \neq x_i'$,

$$\begin{aligned}
\frac{\partial^2 \Phi_{w_i}(\delta)}{\partial \delta_i^\alpha(x_i) \partial \delta_i^\beta(x_i')} &= \frac{0 - \exp\Big[\frac{1}{w_i}\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)\Big] \cdot \frac{\partial Z_i}{\partial \delta_i^\beta(x_i')}}{Z_i^2} \\
&= \frac{0 - \exp\Big[\frac{1}{w_i}\sum_{\alpha \in N_i} \delta_i^\alpha(x_i)\Big] \cdot \exp\Big[\frac{1}{w_i}\sum_{\alpha \in N_i} \delta_i^\alpha(x_i')\Big]}{Z_i^2} \\
&= \frac{1}{w_i}\Big[0 - \mu_i(x_i)\mu_i(x_i')\Big].
\end{aligned}$$

Thus, we obtain the combined result (A.13). □

**Proposition A.4.2.** *Given* $\Phi_{\mathbf{w}^\alpha}(\delta) = \log \sum_{x_\alpha}^{\mathbf{w}^\alpha} \exp\Big[\theta_\alpha(x_\alpha) - \sum_{i\in\alpha} \delta_i^\alpha(x_i)\Big]$ *as in Proposition A.3.1. The Hessian matrix of* $\Phi_{\mathbf{w}^\alpha}(\delta)$ *w.r.t.* $\delta_i^\alpha$ *is,*

$$\frac{\partial^2 \Phi_{\mathbf{w}^\alpha}(\delta)}{\partial \delta_i^\alpha(x_i)\partial \delta_i^\alpha(x_i')} = \frac{1}{w_i^\alpha}\mathbb{I}(x_i = x_i')\mu_\alpha(x_i) + \sum_{j=i}^{c-1}\Big[\big(\frac{1}{w_{j+1}^\alpha} - \frac{1}{w_j^\alpha}\big)\sum_{x_{\alpha\setminus i}}\Big(\mu_\alpha(x_\alpha)\mu_\alpha(x_i'|x_{j+1:c})\Big)\Big] - \frac{1}{w_c^\alpha}\mu_\alpha(x_i)\mu_\alpha(x_i'), \tag{A.14}$$

*where* $\mu_\alpha(x_i) = \sum_{x_{\alpha\setminus i}} \mu_\alpha(x_\alpha)$ *is defined in Proposistion A.3.1 Eq.* (A.9)*, and*

$$\mu_\alpha(x_i'|x_{j+1:c}) = \frac{\mu_\alpha(x_i', x_{j+1:c})}{\mu_\alpha(x_{j+1:c})} = \frac{\sum_{x_{[1:j]\setminus i}} \mu_\alpha(x_{\alpha\setminus i}, x_i')}{\sum_{x_{1:j}} \mu_\alpha(x_\alpha)}$$

*As a special case, when the weights are equal,* $w_1^\alpha = \cdots w_i^\alpha = \cdots w_c^\alpha$*,* (A.14) *becames*

$$\forall \alpha \in N_i, \quad \frac{\partial^2 \Phi_{\mathbf{w}^\alpha}(\delta)}{\partial \delta_i^\alpha(x_i)\partial \delta_i^\alpha(x_i')} = \frac{1}{w_i^\alpha}\Big[\mathbb{I}(x_i = x_i')\mu_\alpha(x_i) - \mu_\alpha(x_i)\mu_\alpha(x_i')\Big], \tag{A.15}$$
$$i.e. \quad \frac{\partial^2 \Phi_{\mathbf{w}^\alpha}(\delta)}{\partial \delta_i^\alpha \partial \delta_i^\alpha} = \frac{1}{w_i^\alpha}\Big[\operatorname{diag}(\boldsymbol{\mu}_i^\alpha) - \boldsymbol{\mu}_i^\alpha \boldsymbol{\mu}_i^{\alpha T}\Big],$$

*where the column vector* $\boldsymbol{\mu}_i^\alpha = [\mu_\alpha(x_i = 1), ..., \mu_\alpha(x_i = d)]^T$*.*

*Proof.*

When $w_1^\alpha = \cdots w_i^\alpha = \cdots w_c^\alpha$, we can also verify the special case (A.15) directly, which is similar to the proof provided in Proposition A.4.1.

To obtain the general result (A.14), we notice that

$$\begin{aligned}\frac{\partial^2 \Phi_{\mathbf{w}^\alpha}(\delta)}{\partial \delta_i^\alpha(x_i)\partial \delta_i^\alpha(x_i')} &= \frac{\partial}{\partial \delta_i^\alpha(x_i')}\Big(\sum_{x_{\alpha\setminus i}} -\mu_\alpha(x_\alpha)\Big) \quad \text{(by Eq. (A.9) )} \\ &= -\sum_{x_{\alpha\setminus i}} \frac{\partial \mu_\alpha(x_\alpha)}{\partial \delta_i^\alpha(x_i')} = -\sum_{x_{\alpha\setminus i}} \mu_\alpha(x_\alpha)\frac{\partial \log \mu_\alpha(x_\alpha)}{\delta_i^\alpha(x_i')}. \end{aligned} \tag{A.16}$$

We transform the intractable $\frac{\partial \mu_\alpha(x_\alpha)}{\partial \delta_i^\alpha(x_i')}$ to $\mu_\alpha(x_\alpha)\frac{\partial \log \mu_\alpha(x_\alpha)}{\delta_i^\alpha(x_i')}$ by noticing the definition of $\mu_\alpha(x_\alpha)$ in Proposition A.3.1,

$$\mu_\alpha(x_\alpha) = \prod_{j=1}^{c}\Big(\frac{Z_{j-1}(x_{j:c})}{Z_j(x_{j+1:c})}\Big)^{\frac{1}{w_j^\alpha}},$$

and thus,

$$\begin{aligned}\log \mu_\alpha(x_\alpha) &= \sum_{j=1}^{c}\frac{1}{w_j^\alpha}\Big(\log Z_{j-1}(x_{j:c}) - \log Z_j(x_{j+1:c})\Big) \\ &= \frac{1}{w_1^\alpha}\log Z_0(x_{1:c}) + \sum_{j=1}^{c-1}\big(\frac{1}{w_{j+1}^\alpha} - \frac{1}{w_j^\alpha}\big)\log Z_j(x_{j+1:c}) - \frac{1}{w_c^\alpha}\log Z_c.\end{aligned} \tag{A.17}$$

According to Equation (A.11),

$$\begin{aligned}\forall j \in [1:c], \quad \frac{\partial \log Z_j(x_{j+1:c})}{\partial \widetilde{\theta}_\alpha(x_\alpha)} &= \mu_\alpha(x_j|x_{j+1:c}) \cdot \frac{\partial \log Z_{j-1}(x_{j:c})}{\partial \widetilde{\theta}_\alpha(x_\alpha)} \\ &= \mu_\alpha(x_j|x_{j+1:c}) \cdot \mu_\alpha(x_{j-1}|x_{j:c}) \cdots \mu_\alpha(x_1|x_{2:c}) \\ &= \mu_\alpha(x_{1:j}|x_{j+1:c}).\end{aligned}$$

Given $i \in [1:c]$, there are three cases of $j \in [1:c]$

$$(a)\ j = i, \quad \frac{\partial \log Z_j(x_{j+1:c})}{\delta_i^\alpha(x_i')} = \sum_{x_{1:j-1}} \frac{\partial \log Z_j(x_{j+1:c})}{\partial \widetilde{\theta}_\alpha(x_{1:j-1}, x_i', x_{j+1:c})} \cdot \frac{\partial \widetilde{\theta}_\alpha(x_{1:j-1}, x_i', x_{j+1:c})}{\delta_i^\alpha(x_i')}$$

$$= \sum_{x_{1:j-1}} \mu_\alpha(x_{1:j-1}, x_i' | x_{j+1:c}) \cdot (-1)$$

$$= -\mu_\alpha(x_i' | x_{j+1:c}),$$

$(b)\ \forall j \in [i+1:c]$,

$$\frac{\partial \log Z_j(x_{j+1:c})}{\delta_i^\alpha(x_i')} = \sum_{x_{1:i-1}} \sum_{x_{i+1:j}} \frac{\partial \log Z_j(x_{j+1:c})}{\partial \widetilde{\theta}_\alpha(x_{1:i-1}, x_i', x_{i+1:j}, x_{j+1:c})} \cdot \frac{\partial \widetilde{\theta}_\alpha(x_{1:i-1}, x_i', x_{i+1:j}, x_{j+1:c})}{\delta_i^\alpha(x_i')}$$

$$= \sum_{x_{1:i-1}} \sum_{x_{i+1:j}} \mu_\alpha(x_{1:i-1}, x_i', x_{i+1:j} | x_{j+1:c}) \cdot (-1)$$

$$= -\mu_\alpha(x_i' | x_{j+1:c}),$$

$(c)\ \forall j \in [0:i-1]$,

$$\frac{\partial \log Z_j(x_{j+1:c})}{\delta_i^\alpha(x_i')} = \sum_{x_{1:j}} \frac{\partial \log Z_j(x_{j+1:i-1}, x_i, x_{i+1:c})}{\partial \widetilde{\theta}_\alpha(x_{1:j}, x_{j+1:i-1}, x_i', x_{i+1:c})} \cdot \frac{\partial \widetilde{\theta}_\alpha(x_{1:j}, x_{j+1:i-1}, x_i', x_{i+1:c})}{\delta_i^\alpha(x_i')}$$

$$= \sum_{x_{1:j}} \mathbb{I}(x_i = x_i') \cdot \mu_\alpha(x_{1:j} | x_{j+1:c}) \cdot (-1)$$

$$= -\mathbb{I}(x_i = x_i').$$

It should be noted that during the enumeration of $\widetilde{\theta}_\alpha(x_\alpha)$ in above chain rule applications, $x_{j+1:c}$ within $\widetilde{\theta}_\alpha$ should be fixed and take the same values as $x_{j+1:c}$ in $\log Z_j(x_{j+1:c})$, else the derivative is equal to 0.

Plugging the above results for $\frac{\partial \log Z_j(x_{j+1:c})}{\delta_i^\alpha(x_i')}$ into (A.17), we get

$$\begin{aligned}
\frac{\partial \log \mu_\alpha(x_\alpha)}{\partial \delta_i^\alpha(x_i')} &= \sum_{j=1}^{i-1} \frac{1}{w_j^\alpha}\Big( -\mathbb{I}(x_i = x_i') + \mathbb{I}(x_i = x_i')\Big) + \frac{1}{w_i^\alpha}\Big( -\mathbb{I}(x_i = x_i') + \mu_\alpha(x_i'|x_{i+1:c})\Big) \\
&\qquad + \sum_{j=i+1}^{c} \frac{1}{w_j^\alpha}\Big( -\mu_\alpha(x_i'|x_{j:c}) + \mu_\alpha(x_i'|x_{j+1:c})\Big) \\
&= -\frac{1}{w_i^\alpha}\mathbb{I}(x_i = x_i') + \frac{1}{w_i^\alpha}\mu_\alpha(x_i'|x_{i+1:c}) + \sum_{j=i+1}^{c} \frac{1}{w_j^\alpha}\Big( -\mu_\alpha(x_i'|x_{j:c}) + \mu_\alpha(x_i'|x_{j+1:c})\Big) \\
&= -\frac{1}{w_i^\alpha}\mathbb{I}(x_i = x_i') + \sum_{j=i}^{c-1}\Big(\frac{1}{w_j^\alpha} - \frac{1}{w_{j+1}^\alpha}\Big)\mu_\alpha(x_i'|x_{j+1:c}) + \frac{1}{w_c^\alpha}\mu_\alpha(x_i'). \qquad \text{(A.18)}
\end{aligned}$$

Plugging (A.18) into (A.16),

$$\begin{aligned}
&\frac{\partial^2 \Phi_{\mathbf{w}^\alpha}(\delta)}{\partial \delta_i^\alpha(x_i)\partial \delta_i^\alpha(x_i')} = -\sum_{x_{\alpha\setminus i}} \mu_\alpha(x_\alpha)\Big[ -\frac{1}{w_i^\alpha}\mathbb{I}(x_i = x_i') + \sum_{j=i}^{c-1}\Big(\frac{1}{w_j^\alpha} - \frac{1}{w_{j+1}^\alpha}\Big)\mu_\alpha(x_i'|x_{j+1:c}) + \frac{1}{w_c^\alpha}\mu_\alpha(x_i')\Big] \\
&= \frac{1}{w_i^\alpha}\mathbb{I}(x_i = x_i')\cdot \mu_\alpha(x_i) + \sum_{j=i}^{c-1}\Big[(\frac{1}{w_{j+1}^\alpha} - \frac{1}{w_j^\alpha})\sum_{x_{\alpha\setminus i}}\Big(\mu_\alpha(x_\alpha)\mu_\alpha(x_i'|x_{j+1:c})\Big)\Big] - \frac{1}{w_c^\alpha}\mu_\alpha(x_i)\mu_\alpha(x_i').
\end{aligned}$$

□

## A.5 Derivations of Closed-form Update

We first derive the closed-form update rule for $\delta_i^\alpha(x_i)$ in Proposition A.5.1. We derive the closed-form update rule for the block $\boldsymbol{\delta}_{N_i} = \{\delta_i^\alpha(x_i) \mid \forall \alpha \in N_i\}$ in Proposition A.5.2.

**Proposition A.5.1.** *Given max node $i$ in marginal MAP (i.e., $\tau_i = 0$ ) and one clique $\alpha \ni i$ (i.e. $i \in \alpha$), keeping all $\delta$ fixed except $\delta_i^\alpha(x_i)$, there is a closed-form update rule,*

$$\delta_i^\alpha(x_i) \leftarrow \frac{1}{2}\log \sum_{x_{\alpha\setminus i}}^{w_{\setminus i}^\alpha} \exp\left[\theta_\alpha(x_\alpha) - \sum_{j\in\alpha\setminus i}\delta_j^\alpha(x_j)\right] - \frac{1}{2}\sum_{\beta\in N_i\setminus\alpha}\delta_i^\beta(x_i), \qquad \text{(A.19)}$$

*where* $x_{\alpha\backslash i} = \{x_j : j \in \alpha, j \neq i\}$, $w^{\alpha}_{\backslash i} = \{w^{\alpha}_j : j \in \alpha, j \neq i\}$, *and* $N_i = \{\alpha | \alpha \ni i\}$ *is the set of all clique factors in the neighborhood of node* $i$. *Futhermore, this update will monotonically decrease the upper bound.*

*Proof.* The terms within the bound $L(\delta, \mathbf{w})$ that depend on $\delta^{\alpha}_i(x_i)$ are,

$$\max_{x_i} \big[ \sum_{\alpha \in N_i} \delta^{\alpha}_i(x_i) \big] + \max_{x_i} \log \sum_{x_{\alpha\backslash i}}^{w^{\alpha}_{\backslash i}} \exp \big[ \theta_{\alpha}(x_{\alpha}) - \sum_{i \in \alpha} \delta^{\alpha}_i(x_i) \big]. \tag{A.20}$$

The sub-gradient of (A.20) w.r.t. $\delta^{\alpha}_i(x_i)$ equal to zero if and only if,

$$x^*_i = \operatorname*{argmax}_{x_i} \big[ \sum_{\alpha \in N_i} \delta^{\alpha}_i(x_i) \big] = \operatorname*{argmax}_{x_i} \log \sum_{x_{\alpha\backslash i}}^{w^{\alpha}_{\backslash i}} \exp \big[ \theta_{\alpha}(x_{\alpha}) - \sum_{i \in \alpha} \delta^{\alpha}_i(x_i) \big],$$

which is "argmax" matching. One sufficient condition of this matching is,

$$\sum_{\alpha \in N_i} \delta^{\alpha}_i(x_i) = \log \sum_{x_{\alpha\backslash i}}^{w^{\alpha}_{\backslash i}} \exp \big[ \theta_{\alpha}(x_{\alpha}) - \sum_{i \in \alpha} \delta^{\alpha}_i(x_i) \big]$$

which impllies matching of "pseudo marginals". Then, one can pull $\delta^{\alpha}_i(x_i)$ outside from the operator $\log \sum_{x_{\alpha\backslash i}}^{w^{\alpha}_{\backslash i}} \exp$, and get the closed-form equation

$$\delta^{\alpha}_i(x_i) = \frac{1}{2} \log \sum_{x_{\alpha\backslash i}}^{w^{\alpha}_{\backslash i}} \exp \big[ \theta_{\alpha}(x_{\alpha}) - \sum_{j \in \alpha\backslash i} \delta^{\alpha}_j(x_j) \big] - \frac{1}{2} \sum_{\beta \in N_i \backslash \alpha} \delta^{\beta}_i(x_i).$$

To prove monotonicity, we substitute above update equation of $\delta^{\alpha}_i(x_i)$ into (A.20); then we get,

$$\max_{x_i} \Big\{ \sum_{\beta \in N_i \backslash \alpha} \delta^{\beta}_i(x_i) + \log \sum_{x_{\alpha\backslash i}}^{w^{\alpha}_{\backslash i}} \exp \big[ \theta_{\alpha}(x_{\alpha}) - \sum_{j \in \alpha\backslash i} \delta^{\alpha}_j(x_j) \big] \Big\}. \tag{A.21}$$

Clearly, (A.21) $\leq$ (A.20) by using the fact that $\max_x[f(x)+g(x)] \leq \max_x f(x) + \max_x g(x)$.

□

**Proposition A.5.2.** *Given node $i \in B$ (i.e., a max node) and all neighborhood cliques $N_i = \{\alpha | \alpha \ni i\}$, we can jointly optimize $\boldsymbol{\delta}_{N_i} = \{\delta_i^\alpha(x_i) \mid \forall \alpha \in N_i\}$ in closed-form by keeping the other $\{\delta_j^\alpha \mid j \neq i, \forall \alpha \in N_i\}$ fixed. The update rule is,*

$$\delta_i^\alpha(x_i) \leftarrow \frac{|N_i|}{|N_i|+1}\gamma_i^\alpha(x_i) - \frac{1}{|N_i|+1}\sum_{\beta \in N_i \backslash \alpha} \gamma_i^\beta(x_i), \tag{A.22}$$

*where $|N_i|$ is the number of neighborhood cliques, and $\{\gamma_i^\alpha(x_i) \mid \forall \alpha \in N_i\}$ are defined by*

$$\gamma_i^\alpha(x_i) = \log \sum_{x_{\alpha \backslash i}}^{\mathbf{w}_{\backslash i}^\alpha} \exp\big[\theta_\alpha(x_\alpha) - \sum_{j \in \alpha \backslash i} \delta_j^\alpha(x_j)\big]. \tag{A.23}$$

*Futhermore, this upate will monotonically decrease the upper bound.*

*Proof.* For $\forall \alpha \in N_i$, we have closed-form solutions for $\delta_i^\alpha(x_i)$ as Proposition A.5.1. We rewrite it as,

$$\forall \alpha \in N_i, \quad 2\delta_i^\alpha(x_i) + \sum_{\beta \in N_i \backslash \alpha} \delta_i^\beta(x_i) = \log \sum_{x_{\alpha \backslash i}}^{w_{\backslash i}^\alpha} \exp\big[\theta_\alpha(x_\alpha) - \sum_{j \in \alpha \backslash i} \delta_j^\alpha(x_j)\big]. \tag{A.24}$$

Note, for $\forall \alpha, \beta \in N_i$, there is a linear relationship between $\delta_i^\alpha(x_i)$ and $\delta_i^\beta(x_i)$.

We denote column vector $\boldsymbol{\gamma}_i(x_i)$ filled $\alpha$-th element with

$$\gamma_i^\alpha(x_i) = \log \sum_{x_{\alpha \backslash i}}^{w_{\backslash i}^\alpha} \exp\big[\theta_\alpha(x_\alpha) - \sum_{j \in \alpha \backslash i} \delta_j^\alpha(x_j)\big].$$

We also frame all $\{\delta_i^\alpha(x_i) \mid \alpha \in N_i\}$ into a column vector $\boldsymbol{\delta}_{N_i}(x_i)$, and denote $|N_i| \times |N_i|$

matrix A

$$A = \begin{pmatrix} 2 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 2 \end{pmatrix}, \text{ and note } A^{-1} = \begin{pmatrix} \frac{|N_i|}{|N_i|+1} & -\frac{1}{|N_i|+1} & \cdots & -\frac{1}{|N_i|+1} \\ -\frac{1}{|N_i|+1} & \frac{|N_i|}{|N_i|+1} & \cdots & -\frac{1}{|N_i|+1} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{1}{|N_i|+1} & -\frac{1}{|N_i|+1} & \cdots & \frac{|N_i|}{|N_i|+1}. \end{pmatrix}$$

It is easy to verify $A\boldsymbol{\delta}_{N_i}(x_i) = \boldsymbol{\gamma}_i(x_i)$. from (A.24). Since $A$ is invertable, one can solve

$$\boldsymbol{\delta}_{N_i}(x_i) = A^{-1}\boldsymbol{\gamma}_i(x_i).$$

Then, one can read out the closed-form update rule (A.22). The monotonicity holds directly by noticing that the update rule (A.22) are solutions which jointly satisfy equation (A.19).

□

# Appendix B

# Derivations and Proofs for Marginal Structured SVM

In this Appendix, we give proofs for two lemmas with respect to the unified framework, which is referenced but omitted from the Chapter 4.

## B.1 Properties of Unified Framework

**Lemma B.1.1.** *The objective of the unified framework (Eq.* (4.6) *in Chapter 4) is an upper bound of the empirical loss function* $\Delta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n_{\epsilon_h}(\theta))$ *over the training set, where the prediction* $\hat{\boldsymbol{y}}^n_{\epsilon_h}(\theta)$ *is decoded by "annealed" marginal MAP:* $\hat{\boldsymbol{y}}^n_{\epsilon_h}(\theta) = \operatorname{argmax}_{\boldsymbol{y}} \log \sum_{\boldsymbol{h}} \exp\left[\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\right]$.

*Proof.*

$$\begin{aligned}
\Delta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n_{\epsilon_h}(\theta)) &\leq \Delta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n_{\epsilon_h}(\theta)) + \epsilon_h \log \sum_{\boldsymbol{h}} \exp\big[\frac{\theta^\top \phi(\boldsymbol{x}^n, \hat{\boldsymbol{y}}^n_{\epsilon_h}(\theta), \boldsymbol{h})}{\epsilon_h}\big] - \epsilon_h \log \sum_{\boldsymbol{h}} \exp\big[\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{h})}{\epsilon_h}\big] \\
&\leq \epsilon_y \log \sum_{\boldsymbol{y}} \exp\Big\{\frac{1}{\epsilon_y}\Big[\Delta(\boldsymbol{y}^n, \boldsymbol{y}) + \epsilon_h \log \sum_{\boldsymbol{h}} \exp\Big(\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\Big)\Big]\Big\} \\
&\quad - \epsilon_h \log \sum_{\boldsymbol{h}} \exp\big[\frac{\theta^T \phi(\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{h})}{\epsilon_h}\big],
\end{aligned}$$

where the first inequality holds by the definition of $\hat{\boldsymbol{y}}^n_{\epsilon_h}(\theta)$, and the second holds for $\forall \epsilon_y > 0$, because the summation over $\boldsymbol{y}$ contains $\hat{\boldsymbol{y}}^n_{\epsilon_h}(\theta)$. □

For convenience, we denote this upper bound as

$$U_n(\theta; \epsilon_y, \epsilon_h) = U_n^+(\theta; \epsilon_y, \epsilon_h) - U_n^-(\theta; \epsilon_h) \tag{B.1}$$

where

$$\begin{aligned} U_n^+(\theta; \epsilon_y, \epsilon_h) &= \epsilon_y \log \sum_{\boldsymbol{y}} \exp\Big\{\frac{1}{\epsilon_y}\Big[\Delta(\boldsymbol{y}^n, \boldsymbol{y}) + \epsilon_h \log \sum_{\boldsymbol{h}} \exp\Big(\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\Big)\Big]\Big\} \\ U_n^-(\theta; \epsilon_h) &= \epsilon_h \log \sum_{\boldsymbol{h}} \exp\big[\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{h})}{\epsilon_h}\big]. \end{aligned}$$

**Lemma B.1.2.** *The (sub-)gradient of $U_n(\theta; \epsilon_y, \epsilon_h)$ in (B.1) is,*

$$\nabla_\theta U_n(\theta; \epsilon_y, \epsilon_h) = \mathbb{E}_{p^{(\epsilon_y, \epsilon_h)}(\boldsymbol{y}, \boldsymbol{h}|\boldsymbol{x}^n)}[\phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})] - \mathbb{E}_{p^{\epsilon_h}(\boldsymbol{h}|\boldsymbol{x}^n, \boldsymbol{y}^n)}[\phi(\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{h})],$$

*where the corresponding temperature controlled distribution is defined as,*

$$\begin{aligned} p^{\epsilon_h}(\boldsymbol{h}|\boldsymbol{x}^n, \boldsymbol{y}) &\propto \exp\Big[\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\Big], \\ p^{(\epsilon_y, \epsilon_h)}(\boldsymbol{y}|\boldsymbol{x}^n) &\propto \exp\Big\{\frac{1}{\epsilon_y}\big[\Delta(\boldsymbol{y}, \boldsymbol{y}^n) + \epsilon_h \log \sum_{\boldsymbol{h}} \exp\big(\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\big)\big]\Big\}, \\ p^{(\epsilon_y, \epsilon_h)}(\boldsymbol{y}, \boldsymbol{h}|\boldsymbol{x}^n) &= p^{\epsilon_h}(\boldsymbol{h}|\boldsymbol{x}^n, \boldsymbol{y}) \cdot p^{(\epsilon_y, \epsilon_h)}(\boldsymbol{y}|\boldsymbol{x}^n). \end{aligned}$$

*Proof.*

$$\nabla_\theta\Big(\epsilon_h \log \sum_{\boldsymbol{h}} \exp\big[\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\big]\Big) = \epsilon_h \frac{\sum_{\boldsymbol{h}} \Big\{ \exp\Big[\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\Big] \cdot \Big[\frac{\phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\Big]\Big\}}{\sum_{\boldsymbol{h}} \exp\Big[\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\Big]}$$

$$= \sum_{\boldsymbol{h}} \Big\{ \frac{\exp\Big[\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\Big]}{\sum_{\boldsymbol{h}} \exp\Big[\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\Big]} \cdot \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h}) \Big\} = \mathbb{E}_{p^{\epsilon_h}(\boldsymbol{h}|\boldsymbol{x}^n, \boldsymbol{y})}[\phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})] \tag{B.2}$$

As a result, $\nabla_\theta U_n^-(\theta; \epsilon_h) = \mathbb{E}_{p^{\epsilon_h}(\boldsymbol{h}|\boldsymbol{x}^n, \boldsymbol{y}^n)}[\phi(\boldsymbol{x}^n, \boldsymbol{y}^n, \boldsymbol{h})]$, and

$\nabla_\theta U_n^+(\theta; \epsilon_y, \epsilon_h)$

$$= \epsilon_y \frac{\sum_{\boldsymbol{y}} \Big\{ \exp\Big\{ \frac{1}{\epsilon_y}\Big[\Delta(\boldsymbol{y}^n, \boldsymbol{y}) + \epsilon_h \log \sum_{\boldsymbol{h}} \exp\Big(\frac{\theta^T \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\Big)\Big]\Big\} \cdot \frac{1}{\epsilon_y} \cdot \nabla_\theta\Big(\epsilon_h \log \sum_{\boldsymbol{h}} \exp\big[\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\big]\Big)\Big\}}{\sum_{\boldsymbol{y}} \exp\Big\{ \frac{1}{\epsilon_y}\Big[\Delta(\boldsymbol{y}^n, \boldsymbol{y}) + \epsilon_h \log \sum_{\boldsymbol{h}} \exp\Big(\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\Big)\Big]\Big\}}$$

Subinstitute the gradient result (B.2),

$$= \frac{\sum_{\boldsymbol{y}} \Big\{ \exp\Big\{ \frac{1}{\epsilon_y}\Big[\Delta(\boldsymbol{y}^n, \boldsymbol{y}) + \epsilon_h \log \sum_{\boldsymbol{h}} \exp\Big(\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\Big)\Big]\Big\} \cdot \mathbb{E}_{p^{\epsilon_h}(\boldsymbol{h}|\boldsymbol{x}^n, \boldsymbol{y})}[\phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})]\Big\}}{\sum_{\boldsymbol{y}} \exp\Big\{ \frac{1}{\epsilon_y}\Big[\Delta(\boldsymbol{y}^n, \boldsymbol{y}) + \epsilon_h \log \sum_{\boldsymbol{h}} \exp\Big(\frac{\theta^\top \phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})}{\epsilon_h}\Big)\Big]\Big\}}$$

$$= \mathbb{E}_{p^{(\epsilon_y, \epsilon_h)}(\boldsymbol{y}|\boldsymbol{x}^n)} \mathbb{E}_{p^{\epsilon_h}(\boldsymbol{h}|\boldsymbol{x}^n, \boldsymbol{y})}[\phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})]$$

$$= \mathbb{E}_{p^{(\epsilon_y, \epsilon_h)}(\boldsymbol{y}, \boldsymbol{h}|\boldsymbol{x}^n)}[\phi(\boldsymbol{x}^n, \boldsymbol{y}, \boldsymbol{h})]$$

which completes the proof. □

# Appendix C

# Derivations and Proofs for Conditional RBMs

## C.1 Derivation of Matrix-based BP

The $(i, j)$ element of pairwise belief matrix:

$$\Gamma_{ij} = \frac{\tau(v_i = 1, h_j = 1)}{\sum_{v_i, h_j} \tau(v_i, h_j)} = \frac{\frac{\exp(w_{ij}^{vh})\tau_i^v \tau_j^h}{M_{ij}^{vh} M_{ji}^{hv}}}{\frac{\exp(w_{ij}^{vh})\tau_i^v \tau_j^h}{M_{ij}^{vh} M_{ji}^{hv}} + \frac{(1-\tau_i^v)\tau_j^h}{(1-M_{ij}^{vh}) M_{ji}^{hv}} + \frac{\tau_i^v (1-\tau_j^h)}{M_{ij}^{vh}(1-M_{ji}^{hv})} + \frac{(1-\tau_i^v)(1-\tau_j^h)}{(1-M_{ij}^{vh})(1-M_{ji}^{hv})}}$$

We can denote the intermediate terms

$$\begin{aligned}
\Gamma^{11} &= \exp(W^{vh}) \circ (\boldsymbol{\tau}^v \cdot \boldsymbol{\tau}^{h\top}) \circ (\mathbf{1}^{vh} - M^{vh}) \circ (\mathbf{1}^{hv} - M^{hv})^\top, \\
\Gamma^{01} &= \left((\mathbf{1}^v - \boldsymbol{\tau}^v) \cdot \boldsymbol{\tau}^{h\top}\right) \circ M^{vh} \circ (\mathbf{1}^{hv} - M^{hv})^\top, \\
\Gamma^{10} &= \left(\boldsymbol{\tau}^v \cdot (\mathbf{1}^h - \boldsymbol{\tau}^h)^\top\right) \circ (\mathbf{1}^{vh} - M^{vh}) \circ M^{hv\top}, \\
\Gamma^{00} &= \left((\mathbf{1}^v - \boldsymbol{\tau}^v) \cdot (\mathbf{1}^h - \boldsymbol{\tau}^h)\right) \circ M^{vh} \circ M^{hv\top}.
\end{aligned}$$

Then, the pairwise belief matrix $\Gamma = \frac{\Gamma^{11}}{\Gamma^{11}+\Gamma^{01}+\Gamma^{10}+\Gamma^{00}}$.