PARAMETER TYING AND DISSOCIATION

IN GRAPHICAL MODELS

by

Li Kang Chou

APPROVED BY SUPERVISORY COMMITTEE:

Vibhav Gogate, Chair

Latifur Khan

Vincent Ng

Nicholas Ruozzi

To my wife,

Linda,

and

in memory of

our canine extraordinaires,

Eros & Bruce.

PARAMETER TYING AND DISSOCIATION IN GRAPHICAL MODELS

by

LI KANG CHOU, BS, MBA, MS

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

December 2019

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisor, Vibhav Gogate. His class, Statistical Methods in Artificial Intelligence and Machine Learning, inspired and sparked my PhD journey. I am thankful for his unwavering patience, constant motivation, and insightful guidance. Without him, this dissertation and the papers published would not have been possible or intelligible. His high standard for research has motivated me to continue to work on achieving high-quality and impactful research myself. Looking back, I am truly grateful to have had Vibhav as my advisor.

I would like to thank my committee members, Dr. Latifur Khan, Dr. Vincent Ng, and Dr. Nicholas Ruozzi, for their guidance on my dissertation and for their patience and flexibility throughout the approval process. I would also like to thank my collaborators, Dr. Nicholas Ruozzi, Dr. Wolfgang Gatterbauer, Somdeb, and Pracheta. In addition, I would like to recognize Somdeb for his invaluable help and mentorship in the beginning of my research.

I was fortunate to be in a position where I had the opportunity to interact with existing senior graduate students as well as new incoming graduate students. Thank you to those who have graduated; David, Somdeb, Tahrima, Deepak, Luis, and Tatiana, for their guidance, inspiring discussions, and friendship. I enjoyed the friendly competition with David on writing the fastest variable elimination program for computing the partition function of a 20×20 grid Markov network. Thank you to Pracheta, Shasha, Chiradeep, Sara, Yuanzhen, and Hao for the many interesting conversations and their encouragement and involvement in my research presentations. Thank you to Yibo for his technical wizardry.

I would like to thank my family for their support. I am especially thankful to my wife, Linda, for her constant love, support, patience, and humor throughout this journey. This PhD would not have been possible without her encouragement.

Finally, I would like to thank the funding agencies DARPA, AFRL, NSF (under grant prime contract and award numbers: FA8750-14-C-0005, N66001-17-2-4032, IIS-1528037, and IIS-1652835), and The University of Texas at Dallas Computer Science Teaching Assistant scholarship for providing financial support.

August 2019

PARAMETER TYING AND DISSOCIATION

IN GRAPHICAL MODELS

Li Kang Chou, PhD
The University of Texas at Dallas, 2019

Supervising Professor: Vibhav Gogate, Chair

Understanding the implications of today's deluge and high velocity of data is a challenging problem facing researchers across multiple disciplines and domains. Data are typically high-dimensional, unstructured, and noisy; thus, the models produced or learned from applying modern machine learning techniques are often very complex, i.e., large number of parameters.

To this, I present new techniques that impose constraints, in the form of parameter tying, on both the learning and inference task for probabilistic graphical models (PGM). Specifically, in this dissertation, we consider two important problems for PGMs. The first problem is parameter learning given a PGM structure with the objective of improving the generalization of the learned model. Specifically, we present and utilize the concept of parameter tying as a novel alternative regularization (variance reduction) framework for parameter learning in PGMs. In addition to improved generalization, parameter tied PGMs are a new class of models for which inference algorithms can be constructed to achieve efficient inference by exploiting the symmetric parameter structure. The second problem focuses on exploiting parameter tying to develop a bounded inference scheme, which we refer to as dissociation-based bounds. We consider the task of weighted model counting which includes important tasks in PGMs such as computing the partition function and probability of evidence as special cases. Namely, we propose a partition-based bounding algorithm that exploits logical

structure and gives rise to a novel set of inequalities from which lower and upper bounds can be derived efficiently. The bounds come with correctness guarantees and are oblivious in that they can be computed by minimally manipulating the parameters of the model.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Probabilistic graphical models (PGM) have emerged as a powerful tool for modeling and reasoning about uncertainty (Koller and Friedman, 2009; Darwiche, 2009). PGMs provide compact and structured representations of joint probability distributions defined over random variables in high-dimensional space. Under the assumption that the joint distribution represents complete knowledge over the random variables, PGMs can be used to answer any probabilistic query over the random variables (e.g., what is the conditional probability of a random variable given observations, namely a value assignment to a subset of other random variables in the model; what is the most likely value assignment to a subset of variables; etc.).

In PGMs, the distribution is factorized and conceptualized over a graphical structure by exploiting conditional independencies in the distribution. In this dissertation, we focus on two main families of PGMs: *Bayesian* and *Markov* networks, as well as their extensions. Bayesian networks use directed acyclic graphs while Markov networks use undirected graphs to depict conditional independencies, where each node in the graph denotes a random variable. At a high level, absence of an edge between two nodes in the graph denotes that there is no direct dependence between the corresponding variables. Various graph properties and concepts such as directed and undirected separators (Pearl, 1988) are then used to formalize the notion of indirect dependencies. For example, in a Markov network, a variable is conditionally independent of all other variables given its neighbors (in an undirected graph, all nodes having an edge with a node are called its neighbors) since removing the neighbors separates the variable from the rest of the network.

Both Bayesian and Markov networks have been extensively used in a wide variety of application domains such as medical diagnosis, computer vision and natural language processing. For effectively using PGMs in these application areas, the two key tasks an application designer has to solve are (1) learning (or estimating) the parameters and structure from

data; and (2) answering probabilistic queries posed to the learned model. As a result, solving the two aforementioned tasks is of both practical and theoretical interest in the artificial intelligence community.

1.1 Learning

In this dissertation, we focus on the task of parameter learning in PGMs with given structure and from a fully observed dataset, namely a dataset having no missing values. The task is often expressed as the following optimization problem: find an assignment of values to all parameters in the PGM such that the likelihood of the data is maximized. This problem is also known as maximum likelihood estimation (MLE). MLE is the prevailing parameter learning formulation because it has several desirable theoretical properties such as *consistency* (converges in the limit to the correct value) and *asymptotic efficiency* (best possible estimator in the limit). However, in practice, when the dataset is small (relative to the number of parameters) or the number of parameters is large (relative to the dataset size) or both, MLE yields parameter estimates having high variance and the resulting models have poor predictive (generalization) accuracy. To combat this issue, regularization methods such as L_1 and L_2 regularization, and their Bayesian counterparts, Laplace and Gaussian priors, are often employed in practice (Steck and Jaakkola, 2002; Ng, 2004).

An alternative regularization method is *parameter tying* (Chou et al., 2016, 2018), namely partitioning the parameters of a model into groups and forcing all parameters in each group to take the same or similar values. Parameter tying is employed in a large variety of graphical models and their extensions. For example, in convolutional neural networks (CNN) (LeCun et al., 1998) parameters are shared (tied) between various neurons to take advantage of symmetries in images and to control the number of parameters. Similarly, in statistical relational learning (SRL) models (Getoor and Taskar, 2007) such as Markov logic networks (Domingos and Lowd, 2009) and probabilistic soft logic (Bach et al., 2017), weights

(parameters) are tied in order to exploit symmetries in relational domains. However, a key feature of this existing work is that it assumes the tied parameters are specified *a priori*. Our contribution deviates from the existing approach and seeks methods that *automatically* tie the parameters by analyzing the data.

A straightforward approach to solving the automatic parameter tying problem is to express it as a constrained optimization problem. Given m number of parameters and a hyperparameter k, which bounds the maximum number of tied parameters and controls the amount of regularization, find a set of at most k equality constraints that are mutually exclusive (no two equality constraints are specified for same parameter) and exhaustive (all parameters are included) such that the likelihood of the data is maximized. Unfortunately, this problem is computationally difficult as it includes structure learning as a special case.

To overcome this computational difficulty, we propose approaches that tie parameters by *quantizing* the parameter values. Namely, we propose to find a many-to-one function $\mathcal{Q}$ that maps m parameter values to a set having k values, such that $m > k$ and the sum of the Euclidean distance between parameter θ and its quantized value $\mathcal{Q}(\theta)$ is minimized. This subproblem can be solved optimally using dynamic programming (Wang and Song, 2011).

1.2 Inference

As alluded to earlier, one main motivation for learning a PGM is to be able to reason about the various underlying states or configurations of the random variables in the model. That is, we would like to perform inference on or pose probabilistic queries to the learned PGM. To reiterate, for example, we may want to know the most likely configuration of a subset of the variables potentially given observations on a disjoint subset of variables in the model. This query is related to finding the (maximum) mode of the underlying distribution and is often utilized in applications such as computer vision and biological protein design.

For exact inference, elimination-based methods, such as variable or bucket elimination (Zhang and Poole, 1994; Dechter, 1996), are in a family of deterministic approaches that eliminate variables, commonly one at a time, by applying the operations of product and sum (i.e., marginalization) to the factors (or functions) in the model. By using the distributive law, the product and sum operations can be organized according to a variable ordering and factor partitioning to help reduce the overall computation burden. The elimination approach leverages dynamic programming in that the original computation is decomposed into sub-computations such that the intermediate results are stored. While this adds efficiency to the algorithm, we are still faced with the issue that the sub-computations remain intractable in general. In fact, the difficulty in designing inference algorithms is that, for many problems of interest, performing exact inference is computationally intractable. Therefore, in this dissertation, we consider the essential need for *approximate* inference algorithms.

Monte Carlo methods (Metropolis and Ulam, 1949) provide a general framework for stochastic approximation to intractable problems. At a high-level, applying this framework involves two main sub-tasks. The first task is to sample from the model. For PGMs, this means to generate instantiations from the model[1], namely assignments to the variables. The second task is to express the quantity of interest (i.e., probabilistic query) as an expectation. Using the samples, we then compute the expectation as an approximation to the solution.

In addition to regularization, another key benefit from applying parameter tying to the learning algorithm is that the resulting models contain symmetries within the parameterization (i.e., tied parameters). By exploiting the symmetries, we can develop sampling algorithms that explore the sample space in a more efficient and systematic manner, which then lead to faster and more accurate results. We propose a method based on importance sampling.

Relaxation is a term synonymous with mathematical or combinatorial optimization. Depending on the problem at hand, relaxation can have different definitions. However,

[1]This is also referred to as simulating the network.

generally it can be thought of as an alternative modeling strategy applied to the original difficult problem (e.g., NP-hard or #P-complete) with the intention of simplifying the problem. Traditionally, with combinatorial optimization or counting problems (e.g., maximum a posteriori query or partition function computation), this means *relaxing* inherent constraints (e.g., equality). The trade-off is a lose in accuracy, but a gain in information to the solution of the problem.

Referring back to the elimination method, we mentioned that the complication arises from local or sub-computations. We approach this problem by approximating the local subproblems through a structural adjustment. The adjustment we make is to apply dissociation[2] (Gatterbauer and Suciu, 2014; Chou et al., 2018) to the subproblems, which involves two parts. The first part of dissociation is to duplicate the variables. By having multiple copies of the variables, the original equivalence relations or constraints are *relaxed* and the local problems are partitioned or decomposed into further (tractable) subproblems (Dechter and Rish, 2003). The second part is what we refer to as valuation analysis. That is, we analyze the equivalence relations using an algebraic framework to derive inequality constraints in order to recover a closer or tighter approximate solution to the original problem. We view the inequality constraints as applying parameter tying between the original and relaxed problem.

1.3 Dissertation Organization

The dissertation is organized as follows. In Chapter 2, we present a general background related to our contributions. The two sections below outline our contributions, which are detailed in Chapters 3 to 6. In Chapter 7, we summarize the contributions in this dissertation and provide several directions for future work.

[2]Dissociation is related to schemes that involve variable duplication or node splitting.

1.3.1 Contributions to Learning using Parameter Tying

In Chapter 3, we present our first approach to learning parameter tied Bayesian networks. This greedy approach, which we dub learn-tie-relearn (LTR), quantizes the parameterization of the MLE solution as a post processing step. The resulting parameter tied model gives rise to a smoother underlying distribution. The performance of the quantized model can then be further improved by *relearning* the model based on the equality constraints from the quantization. We provide and prove error bounds for our new technique and demonstrate experimentally that it often yields models having higher test set log-likelihood than the ones learned using the MLE.

In Chapter 4, we consider the problem of learning parameter tied Markov networks. We propose a novel approach called automatic parameter tying (APT) that uses *soft* instead of *hard* parameter tying as a regularization method to alleviate overfitting. The key idea behind APT is to set up the learning problem as the task of finding parameters and groupings of parameters such that the likelihood plus a regularization term is maximized. The regularization term penalizes models where parameter values deviate from their group mean parameter value. We propose and use a block coordinate ascent algorithm to solve the optimization task. In addition, we analyze the sample complexity of our new learning algorithm and show that it yields optimal parameters with high probability when the groups are *well separated*. Experimentally, we show that our method improves upon L_2 regularization and suggest several pragmatic techniques for good practical performance.

1.3.2 Contributions to Inference using Parameter Tying

In Chapter 5, we propose a new slice importance sampling algorithm for fast approximate inference in models having several tied parameters. Our experiments show that our new inference algorithm is superior to existing approaches such as Gibbs sampling and MC-SAT on models having tied parameters (learned using our quantization-based approach).

In Chapter 6, we consider the weighted model counting task which includes important tasks in graphical models, such as computing the partition function and probability of evidence as special cases. We propose a novel partition-based bounding algorithm that exploits logical structure and gives rise to a set of inequalities from which upper (or lower) bounds can be derived efficiently. The bounds come with optimality guarantees under certain conditions and are oblivious in that they require only limited observations of the structure and parameters of the problem. We experimentally compare our bounds with the mini-bucket scheme (which is also oblivious) and show that our new bounds are often superior and never worse on a wide variety of benchmark networks.

CHAPTER 2

BACKGROUND

In this chapter, we introduce a general background on representation, inference, and learning for probabilistic graphical models (PGM) with additional relevant topics. For more detailed information on probability theory and statistics, refer to (Wasserman, 2004; Bertsekas and Tsitsiklis, 2008). For more detailed information on PGMs, refer to (Pearl, 1988; Koller and Friedman, 2009; Darwiche, 2009). For more detailed information on Monte Carlo methods, refer to (Liu, 2004).

2.1 Preliminary Notation

Let X, Y, etc. be random variables and $\boldsymbol{X}, \boldsymbol{Y}$, etc. be sets of random variables (e.g., $\boldsymbol{X} = \{X_1, \ldots, X_n\}$). We assume throughout the thesis the variables $X_i \in \boldsymbol{X}$ are discrete, namely taking a finite d number of values. For simplicity, we assume $d = 2$ (i.e., binary), unless otherwise noted. Let x, y, etc. be values the variables can take and denote $\text{Val}(X_i)$ as the domain or set of possible values. Let $\widetilde{x}_i$ be an assignment of a value to a variable X_i, such that $x_i \in \text{Val}(X_i)$. We also use $\text{Val}(\boldsymbol{X}_j)$ to denote the possible values for a subset of variables $\boldsymbol{X}_j \subset \boldsymbol{X}$. We denote $\widetilde{\mathbf{x}} = (\widetilde{x}_1, \ldots, \widetilde{x}_n)$ to be an assignment of values to all variables $\boldsymbol{X}$.

2.2 Representation

We are interested in representing a joint probability distribution P over the set of random variables. Given n binary variables, one approach is to explicitly represent the joint probability distribution using a tabular form with $2^n - 1$ entries such that each entry corresponds to the probability of some joint assignment to all the variables. In practice, this approach is not feasible for at least three reasons. First, it is difficult to elicit such information from domain experts. Second, it requires an enormous amount of data to estimate the probability

distribution statistically. Lastly, it is computationally intractable to manage and store such information.

However, probability distributions defined over high-dimensional data often explicitly or implicitly contain probabilistic structure in the form of independence properties that can be exploited to achieve a more compact *factorized* representation. We next present some relevant concepts from probability theory that form the fundamental building blocks for independence properties.

Definition 2.1. (Conditional Probability). *Given two random variables X and Y such that $P(Y) > 0$. The conditional probability of X given (or conditioned on) Y is defined as*

$$P(X|Y) = \frac{P(X,Y)}{P(Y)}.$$

Note the condition probability can be rewritten as a product, namely $P(X,Y) = P(X|Y)P(Y)$. We can generalize this notion of factoring a joint probability as a product of conditional probabilities through the definition of the chain rule (or product rule).

Definition 2.2. (Chain Rule). *Given a set of random variables $\boldsymbol{X} = \{X_1, \ldots, X_n\}$. By the definition of conditional probability, the joint probability distribution over $\boldsymbol{X}$ can be factored as*

$$P\left(\bigcap_{i=1}^{n} X_i\right) = \prod_{i=1}^{n} P\left(X_i \middle| \bigcap_{j=1}^{n-1} X_j\right).$$

Example 2.3. *Given a joint probability $P(X_1, X_2, X_3, X_4)$, applying the chain rule, one way to factor the joint is $P(X_1, X_2, X_3, X_4) = P(X_1|X_2, X_3, X_4)P(X_2|X_3, X_4)P(X_3|X_4)P(X_4)$.*

Building on the ideas of conditional probability and the chain rule, we next define the concepts of independence and conditional independence.

Definition 2.4. (Independence). *Two random variables X and Y are independent, denoted as $X \perp\!\!\!\perp Y$, iff*

$$P(X, Y) = P(X)P(Y).$$

Equivalently by the definition of conditional probability, we have

$$P(X|Y) = P(X).$$

Definition 2.5. (Conditional Independence). *Given random variables $X, Y,$ and Z. X and Y are conditional independent given Z, denoted as $X \perp\!\!\!\perp Y|Z$, iff*

$$P(X, Y|Z) = P(X|Z)P(Y|Z).$$

2.2.1 Probabilistic Graphical Models

Probabilistic graphical models (PGM) (or graphical models (GM)) are a general class of machine learning models that compactly represent a joint distribution over a set of random variables via a collection of factors (or functions), which can be conceptualized over a graph. The graph can be interpreted as either a skeleton structure for representing the factorized distribution or as a set of independence properties encoded in the distribution. We first present and define two relevant concepts for PGMs.

Definition 2.6. (Factor). *Given a set of random variables $\boldsymbol{X}$, a factor, denoted as $\psi(\boldsymbol{X})$, is a function that maps the possible values of $\boldsymbol{X}$ to a real value. A nonnegative factor is a function that maps $\boldsymbol{X}$ to nonnegative real values.*

Definition 2.7. (Scope). *Given a factor $\psi(\boldsymbol{X})$, the set of random variables $\boldsymbol{X}$ represented by (or appearing in) the factor is the scope of the factor, denoted as Scope$[\psi]$.*

We restrict our focus to nonnegative factors, namely $\psi : \boldsymbol{X} \to \mathbb{R}^+$, where $\mathbb{R}^+ = \{x \in \mathbb{R} : x \geq 0\}$. Two main families of PGMs are Bayesian networks and Markov networks, which we present next.

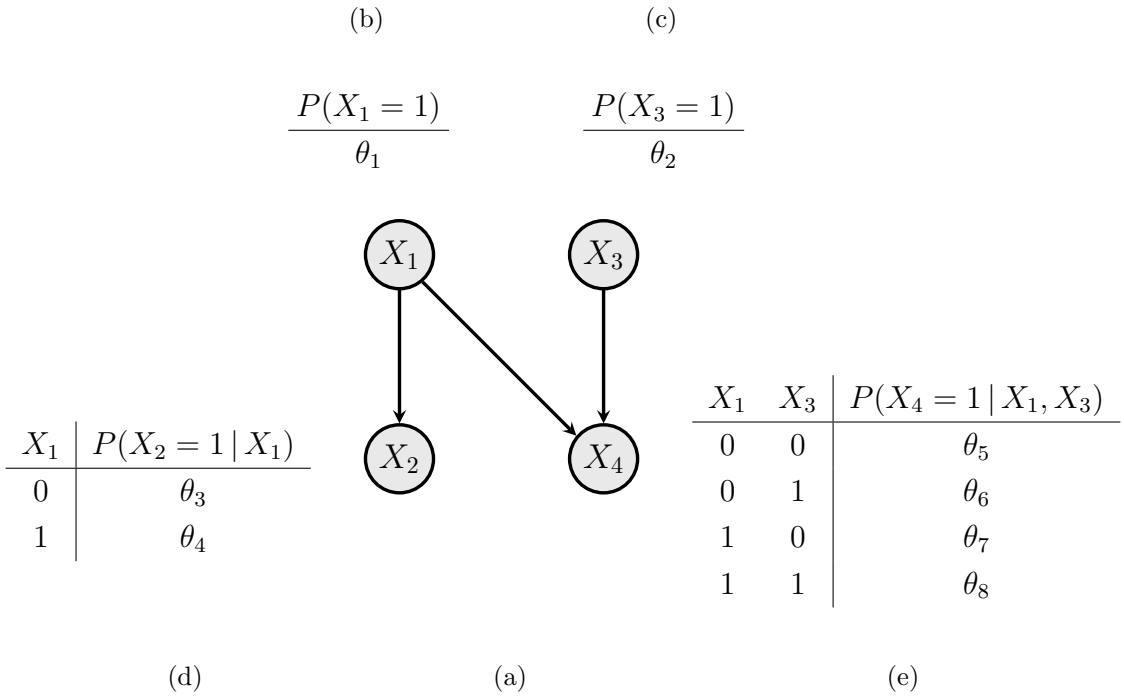


Figure 2.1. An example of a binary Bayesian network.

Bayesian Networks

Bayesian networks (BN) represent joint probability distributions factorized and parameterized over a directed acyclic graph (DAG). A BN is a triple $\mathcal{M}_{BN} \triangleq \langle \boldsymbol{X}, \boldsymbol{\theta}, \mathrm{G}_{BN} \rangle$. $\boldsymbol{X}$ is the set of random variables and $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_m\}$ is the set of parameters. $\mathrm{G}_{BN} = (\mathbf{V}, \mathbf{E})$ is a DAG such that $\mathbf{V}$ is the set of nodes (or vertices) and $\mathbf{E}$ is the set of directed edges. We associate each node $i \in \mathbf{V}$ with one variable $X_{i \in \mathbf{V}}$ (we also use X_i as shorthand). $\mathbf{E}$ is the set of pairs of the form $(\boldsymbol{X}', X_i)$ such that $\boldsymbol{X}' \subset \boldsymbol{X} \setminus X_i$ and $\forall\, Y \in \boldsymbol{X}', Y \rightarrow X_i$. Given a Y, we say Y is the *parent* of X_i and X_i is the *child* of Y. We denote the subset $\boldsymbol{X}'$ (i.e., parents of X_i) as $\mathrm{Pa}[X_i]$. A (local) conditional probability distribution is defined for each variable and corresponding parent set as $P(X_i \mid \mathrm{Pa}[X_i])$. The joint probability distribution is then

factored as

$$P_{\boldsymbol{\theta}}(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \mathrm{Pa}[X_i]).$$

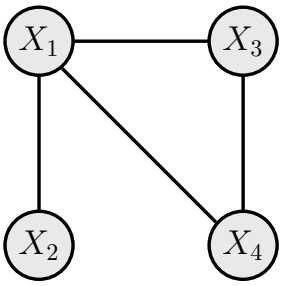
One way to parameterize discrete conditional probability distributions is to utilize a tabular form, which are typically referred to as *conditional probability tables* (CPT). For BNs, the parameters defined in the CPTs have an intuitive probabilistic interpretation, namely $\theta_i \in [0, 1]$. A factor subsumes the notion of the conditional probability distribution defined by a CPT.

Example 2.8. *Figure 2.1 shows a binary Bayesian network. The network (a) is a DAG comprised of four variables:* X_1, X_2, X_3, *and* X_4. *The parent set for each variable are:* $\mathrm{Pa}[X_1] = \{\}$, $\mathrm{Pa}[X_3] = \{\}$, $\mathrm{Pa}[X_2] = \{X_1\}$, *and* $\mathrm{Pa}[X_4] = \{X_1, X_3\}$. *A CPT is specified and parameterized for each variable ((b) to (e)). Notice that for some CPT defined for a variable* X_i, *the probability* $P(X_i = 0 \,|\, \mathrm{Pa}[X_i])$ *is* $1 - \theta_i$. *The BN defines the joint distribution:* $P_{\boldsymbol{\theta}}(X_1, X_2, X_3, X_4) = P(X_1)P(X_3)P(X_2|X_1)P(X_4|X_1, X_3)$.

The BN from Figure 2.1 encodes a set of (conditional) independencies and therefore leads to a compact factorization of the joint distribution stated in Example 2.8. To highlight some independence statements, we have for example $(X_1 \perp\!\!\!\perp X_3)$ and $(X_2 \perp\!\!\!\perp X_3 \,|\, X_1, X_4)$. For a detailed treatment of independencies in graphs for BNs, see (Pearl, 1988; Koller and Friedman, 2009).

Markov Networks

Markov networks (MN) or Markov random fields represent joint probability distributions factorized over an undirected graph. A MN is a triple $\mathcal{M}_{MN} \triangleq \langle \boldsymbol{X}, \boldsymbol{\theta}, \mathrm{G}_{MN} \rangle$. $\boldsymbol{X}$ is the set of random variables and $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_m\}$ is the set of parameters. $\mathrm{G}_{MN} = (\mathbf{V}, \mathbf{E})$ is an undirected graph such that $\mathbf{V}$ is the set of nodes (or vertices) and $\mathbf{E}$ is the set of undirected edges. We associate each node $i \in \mathbf{V}$ with one random variable $X_{i \in \mathbf{V}} \in \boldsymbol{X}$ (we also use X_i



(a)

X_1	X_2	$\psi_{1,2}(X_1, X_2)$
0	0	θ_1
0	1	θ_2
1	0	θ_3
1	1	θ_4

X_1	X_3	$\psi_{1,3}(X_1, X_3)$
0	0	θ_5
0	1	θ_6
1	0	θ_7
1	1	θ_8

X_1	X_4	$\psi_{1,4}(X_1, X_4)$
0	0	θ_9
0	1	θ_{10}
1	0	θ_{11}
1	1	θ_{12}

X_3	X_4	$\psi_{3,4}(X_3, X_4)$
0	0	θ_{13}
0	1	θ_{14}
1	0	θ_{15}
1	1	θ_{16}

(b)

Figure 2.2. An example of a binary pairwise Markov network.

as shorthand). $\mathbf{E}$ is a set of pairs of the form (X_i, X_j) such that $i \neq j$ (we also use (i, j) as shorthand). Given a node X_i, we call the set of nodes connected to X_i (or *neighbors* of X_i) as the *Markov blanket* of X_i, denoted as $\mathrm{Mb}[X_i]$. The size of the Markov blanket for a given variable X_i, namely $|\mathrm{Mb}[X_i]|$, is also know as the *degree* of X_i. Let $\mathcal{I}$ be the set of cliques in G_{MN}. A clique is a subset of nodes in an undirected graph such that every two distinct nodes are adjacent (i.e., connected). We focus on pairwise MNs (i.e., cliques of size 2) since every PGM can be converted to this form (cf. (Koller and Friedman, 2009)). In a pairwise PGM, we associate a factor $\psi_{i,j}$ over each pair $(i, j) \in \mathcal{I}$. The joint probability distribution is then factored as

$$P_{\boldsymbol{\theta}}(X_1, \ldots, X_n) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{(i,j) \in \mathcal{I}} \psi_{i,j}(X_i, X_j),$$

where $Z(\boldsymbol{\theta})$ is the normalizing constant (or partition function) defined as

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{X}} \prod_{(i,j) \in \mathcal{I}} \psi_{i,j}(X_i, X_j).$$

Since the graphs represented by MNs are undirected, and unlike BNs, the parameterization for MNs do not naturally follow a probabilistic interpretation. Therefore, the parameterization of MNs are more effectively described as affinities or the compatibility among the different random variables. However, similar to BNs, the factors can also be represented using a tabular formulation with parameters $\theta_i \in [0, \infty)$. The factors in MNs are often referred to as (*clique*) *potentials*.

Example 2.9. *Figure 2.2 shows a discrete binary Markov network. The network (a) is a undirected graph comprised of four variables, where* $\text{Mb}[X_1] = \{X_2, X_3, X_4\}$, $\text{Mb}[X_2] = \{X_1\}$, $\text{Mb}[X_3] = \{X_1, X_4\}$, *and* $\text{Mb}[X_4] = \{X_1, X_3\}$. *There are four pairwise cliques and a potential is specified for each clique (b), which defines the joint distribution* $P_{\boldsymbol{\theta}}(X_1, X_2, X_3, X_4) = \psi_{1,2}(X_1, X_2)\psi_{1,3}(X_1, X_3)\psi_{1,4}(X_1, X_4)\psi_{3,4}(X_3, X_4)$.

We highlight below some independence properties encoded in MNs. For a detailed treatment, see (Koller and Friedman, 2009). A MN represented by graph G_{MN} satisfies the following Markov properties.

- **Global Markov Property** Given subsets $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{S} \subset \boldsymbol{V}$. Let $\boldsymbol{X_A}$, $\boldsymbol{X_B}$, and $\boldsymbol{X_S}$ be sets of random variables corresponding to the respective vertices. Any two subsets are conditionally independent given a separating set. Namely, $\boldsymbol{X_A} \perp\!\!\!\perp \boldsymbol{X_B} \,|\, \boldsymbol{X_S}$ where every path from a node $i \in \boldsymbol{A}$ to a node $j \in \boldsymbol{B}$ passes through a node $h \in \boldsymbol{S}$.

- **Local Markov Property** Given a random variable X_i. X_i is conditionally independent of all other variables given $\text{Mb}[X_i]$. Namely, $X_i \perp\!\!\!\perp \boldsymbol{X}_{\boldsymbol{V}\setminus \text{Mb}[X_i]} \,|\, \boldsymbol{X}_{\text{Mb}[X_i]}$.

- **Pairwise Markov Property** Two random variables X_i and X_j are conditionally independent given all other variables. Namely, $X_i \perp\!\!\!\perp X_j \,|\, \boldsymbol{X}_{\boldsymbol{V}\setminus\{i,j\}}$.

X_1	X_2	$\psi_{1,2}(X_1, X_2)$
0	0	23
0	1	1
1	0	1
1	1	23

(a)

X_1	X_2	$\log \psi_{1,2}(X_1, X_2)$
0	0	3.135
0	1	0
1	0	0
1	1	3.135

(b)

Figure 2.3. Potentials for Example 2.10.

Log-linear Graphical Models

Although the graph of a PGM provides an explicit structure for a represented distribution, the parameterization within the factors can exhibit patterns that allow for further compact or finer-grained representation. To begin with, by specifying the parameter values in logarithmic space, certain patterns (or structures) become more apparent. In addition, the parameters can now take any real value, namely $\theta_i \in (-\infty, \infty)$. Furthermore, the operations of multiplying potentials now become summations, which offers algebraic convenience and numerical stability.

Example 2.10. *Given a potential ψ defined over two random variables X_1 and X_2 shown in Figure 2.3 (a). The corresponding parameterization in log-space is shown in Figure 2.3 (b). We observe that a more compact representation only needs to capture the fact that if X_1 and X_2 are equal, then the parameter for the potential is* 3.135. *Otherwise, the parameter value is* 0.

One general framework, as an alternative parameterization, is a *log-linear model*. A log-linear PGM is a triple $\mathcal{M}_\ell \triangleq \langle \boldsymbol{X}, \boldsymbol{F}, \boldsymbol{\theta} \rangle$, where $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ is a set of variables, $\boldsymbol{F} = \{f_1(\boldsymbol{X}_1), \ldots, f_m(\boldsymbol{X}_m)\}$ is a set of features such that $\boldsymbol{X}_i$ is a subset of variables (i.e., $\boldsymbol{X}_i \subset \boldsymbol{X}$), and $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_m\}$ is the set of weights (parameters) such that θ_i is the weight of feature $f_i(\boldsymbol{X}_i)$ (we also use f_i as shorthand when it is apparent from the context). A

feature is a factor without the non-negativity constraint on the parameter (i.e., log-space). One type of feature is known as an *indicator feature*. That is, given an assignment $\widetilde{\mathbf{x}}$, $f_i(\widetilde{\mathbf{x}})$ evaluates to 1 if some values $\widetilde{\mathbf{x}} \in \text{Val}(\boldsymbol{X}_i)$ and 0 otherwise. $\mathcal{M}_\ell$ represents the probability distribution

$$P_{\boldsymbol{\theta}}(\widetilde{\mathbf{x}}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left(\sum_{i=1}^{m} \theta_i f_i(\widetilde{\mathbf{x}})\right),$$

where

$$Z(\boldsymbol{\theta}) = \sum_{\widetilde{\mathbf{x}} \in \boldsymbol{X}} \exp\left(\sum_{i=1}^{m} \theta_i f_i(\widetilde{\mathbf{x}})\right)$$

is the normalization constant (or partition function).

Features allow for more compact representations that account for particular patterns or relationships among the variables in a factor. Building on the observation from Example 2.10, we can reparameterize the potential $\psi(X_1, X_2)$ compactly as two features: $f_1(X_1, X_2) \triangleq \mathbb{1}(X_1 = X_2)$ associated with the weight $\theta_1 = 3.135$ and $f_2(X_1, X_2) \triangleq \mathbb{1}(X_1 \neq X_2)$ associated with the weight $\theta_2 = 0$. We can convert and represent both BNs and MNs as log-linear model forms.

Intuitively, we can view these indicator features as a set of constraints and therefore naturally lead to a logical interpretation. While PGMs provide the ability to reason under uncertainty, we next present logic as a formalism to complement this reasoning ability.

2.2.2 Propositional Logic

Propositional logic provides the formalism to reason about the truth or falsehood of logical assertions. Propositional logic consists of propositional sentences or statements[1], which are represented by variables and logical connectives (or operators). Logical connectives allow for new and complex propositions, namely, formulas, to be constructed using operations such as $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), $\Rightarrow$ (implication), and $\Leftrightarrow$ (equivalence). An

[1]Statements can be considered as certain types of sentences, but we do not make the distinction.

atomic proposition or atom is a proposition that does not include any logical connectives. Given some variable X_i, for compactness, we use the notation $\overline{X_i}$ to denote negation (cf. $\neg X_i$). A non-negated or negated atom is also referred to as a positive (+) literal or negative (−) literal respectively.

Definition 2.11. (Monotonicity) *A formula F is "monotone in variable X_i" iff X_i appears in F as either positive or negative (but not both). A formula F is "monotone" iff it is monotone in all variables. Otherwise, F is non-monotone.*

Atoms and formulas take values (i.e., truth assignments) from the set $\{false, true\}$ (or $\{0, 1\}$). Given a set of variables $\boldsymbol{X} = \{X_1, \ldots, X_n\}$, let Ω be the set of the 2^n truth assignments to $\boldsymbol{X}$. Let $\widetilde{\mathbf{x}} = (x_1, \ldots, x_n) \in \Omega$ be truth assignments to all variables in $\boldsymbol{X}$ such that $X_i = x_i$. We denote with the symbol '$*$' for the case when X_i can take either $false$ or $true$ value, namely $(0 \vee 1)$ or otherwise known as the *don't care* condition. We say a formula is *satisfied* if given an assignment to the propositions evaluates the formula to true. The assignment that satisfies the formula is also known as a *model* of the formula.

Example 2.12. *Let X_1, X_2, and X_3 be atoms. Let $F = \overline{X_1} \vee X_2 \vee \overline{X_3}$ be a formula. The literals appearing in F are $\overline{X_1}$, X_2, and $\overline{X_3}$. F is monotone with 2^3 possible assignments. The assignment, $(X_1 = 1$, $X_2 = 1$, and $X_3 = 1) \in \Omega$, evaluates F to 1 (or true). Given the assignment $X_2 = 1$, X_1 and X_3 become don't care conditions (i.e., $(X_1 = *$, $X_2 = 1$, and $X_3 = *)$), that is, F is satisfied once X_2 is 1.*

The *propositional satisfiability problem* (or *Boolean satisfiability problem*), abbreviated as SAT, is the problem of deciding if there exists an assignment that satisfies a given formula. SAT was the first problem to be proven as NP-complete (Cook, 1971; Levin, 1973). The *propositional model counting problem*, abbreviated as #SAT, is the problem of computing the number of distinct satisfying assignments for a given formula. #SAT generalizes SAT and it is known to be #P-complete (Valiant, 1979).

X_1	X_2	$(\overline{X}_1 \vee X_2) \wedge (X_1 \vee \overline{X}_2)$
0	0	1
0	1	0
1	0	0
1	1	1

Figure 2.4. Truth table for Example 2.15

One canonical form used within the SAT and #SAT problem setting is the so-called *conjunctive normal form*, which we define below.

Definition 2.13. (Conjunctive normal form) *A propositional formula is in conjunctive normal form (CNF) if it is a conjunction of clauses, where a clause is a disjunction of literals.*

Example 2.14. *Let A and B be atoms. The formula $F = (\overline{X_1} \vee X_2) \wedge (X_1 \vee \overline{X_2})$ is in conjunctive normal form. F is non-monotone since both X_1 and X_2 are not monotone in F (i.e., X_1 and X_2 appear both as negative and positive).*

Example 2.15. *Consider the formula F in Example 2.14. We use the truth-table shown in Figure 2.4 to illustrate the solution to the model counting problem for F. The #SAT solution for F is 2 because there are two distinct assignments that satisfies F, namely $X_1 = X_2 = 0$ and $X_1 = X_2 = 1$.*

In Chapter 6, we focus on the *weighted model counting* (WMC) problem, a further extension of the model counting problem. (Weighted) model counting falls under the general notion of *inference*, which we present next.

2.3 Inference

With a PGM, one task is to reason about or pose probabilistic queries to the model. We refer to this broad task as *probabilistic inference* or simply *inference*. Specifically, we would like

to compute quantities or configurations of interest using the PGM. These queries typically consists of two disjoint subsets of variables: (1) query (or non-evidence) variables, denoted as $\boldsymbol{Y}$, and (2) evidence (or observed) variables and an instantiation of or assignment to the evidence variables, denoted as $\boldsymbol{E} = \boldsymbol{e}$. Formally, $\boldsymbol{Y} \subset \boldsymbol{X}$, $\boldsymbol{E} \subset \boldsymbol{X}$, and $\boldsymbol{Y} \cap \boldsymbol{E} = \emptyset$. The common inference tasks for PGMs include the following:

- Compute the normalizing constant (or partition function):

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{X}} \prod_{\xi \in \mathcal{I}} \psi_{\xi}(\boldsymbol{X}_{\xi}),$$

  where ξ indexes the variables of the PGM. Within the BN setting, this task is also called computing the probability of evidence (i.e., $P(\boldsymbol{E} = \boldsymbol{e})$).

- Compute the posterior probability distribution over $\boldsymbol{Y}$, conditioned on evidence:

$$P(\boldsymbol{Y}|\boldsymbol{E} = \boldsymbol{e}) = \frac{P(\boldsymbol{Y}, \boldsymbol{E} = \boldsymbol{e})}{P(\boldsymbol{E} = \boldsymbol{e})}.$$

  This quantity is also known as the posterior marginal probability.

- Determine the most probable (i.e., mode) assignment to the non-evidence variables of the distribution:

$$\underset{\boldsymbol{y}}{\text{argmax}} \; P(\boldsymbol{Y}, \boldsymbol{E} = \boldsymbol{e}).$$

  This problem is also know as the maximum a posteriori (MAP) query. $\boldsymbol{Y}$ is typically referred to as MAX variables. Within the BN setting, this task is also called determining the most probable explanation (MPE).

- Marginal MAP:

$$\underset{\boldsymbol{y}}{\text{argmax}} \; \sum_{\boldsymbol{U}} P(\boldsymbol{Y}, \boldsymbol{U}, \boldsymbol{E} = \boldsymbol{e}).$$

  This problem is considered to be the more general setting of the MAP query. $\boldsymbol{Y}$ is the set of MAX variables and $\boldsymbol{U}$ is the set of SUM variables, where $\boldsymbol{U} = \boldsymbol{X} \setminus \boldsymbol{Y} \setminus \boldsymbol{E}$.

Algorithm 2.1: Bucket Elimination (BE) for computing the partition function

Input: Variable ordering $o = [X_1, X_2, \ldots, X_{|\boldsymbol{X}|}]$,
factors (or functions) of the PGM $\Psi = \{\psi_\xi(\boldsymbol{X}_\xi)\}$

Output: Partition function

begin

  for $i \leftarrow 1$ *to* $|\boldsymbol{X}|$ **do**

    1. Find the set of factors (bucket) that involve variable X_i and update the factors.

$$\mathbf{B}_{X_i} \leftarrow \{\psi : \psi \in \Psi \wedge X_i \in \text{Scope}[\psi]\},$$

$$\Psi \leftarrow \Psi \setminus \mathbf{B}_{X_i}.$$

    2. Eliminate variable X_i. Take the product of all factors in the bucket and sum-out X_i. The result is a new factor ψ^*_ξ, which is then added to the factors.

$$\psi^*_\xi \leftarrow \sum_{X_i} \prod_{\psi_\xi \in \mathbf{B}_{X_i}} \psi_\xi,$$

$$\Psi \leftarrow \Psi \cup \psi^*_\xi.$$

  The remaining factors ψ are constants.

  return *Partition function* $Z \leftarrow \prod_{\psi \in \Psi} \psi$

end

Although there exists an array of exact inference algorithms for solving the aforementioned tasks, in general, the quantities of interest are computationally intractable. Thus, in this dissertation, we address the important need for approximate inference schemes. Two popular families for approximate inference schemes can be broadly categorized as elimination-based and Monte Carlo methods. We next describe these schemes.

2.3.1 Elimination-based Methods

Using a variable ordering, elimination-based methods, such as variable or bucket elimination (Dechter, 1996; Zhang and Poole, 1994), leverage the idea of dynamic programming and work

Algorithm 2.2: Mini-bucket elimination (MB) for computing a bound on the partition function.

Input: Variable ordering $o = [X_1, X_2, \ldots, X_{|\boldsymbol{X}|}]$,
factors (or functions) of the PGM $\Psi = \{\psi_\xi(\boldsymbol{X}_\xi)\}$, Integer $i > 0$ (*i-bound*)

Output: Upper bound on the partition function
(replace max with min in step 3 (line 7) for lower bound)

begin

  for $i \leftarrow 1$ *to* $|\boldsymbol{X}|$ **do**

    1. Find the set of factors (bucket) that involve variable X_i and update the factors.

$$\mathbf{B}_{X_i} \leftarrow \{\psi : \psi \in \Psi \wedge X_i \in \text{Scope}[\psi]\},$$

$$\Psi \leftarrow \Psi \setminus \mathbf{B}_{X_i}.$$

    2. Partition the bucket $\mathbf{B}_{X_i}$ into B disjoint groups such that the cardinality of the union of scopes for the factors in each group $\mathbf{B}^{(b)}_{X_i}$ (mini-bucket) is less than or equal to *i-bound* plus 1.

$$\mathbf{B}^{(1)}_{X_i} \cup \mathbf{B}^{(b)}_{X_i} \cup, \ldots, \cup \mathbf{B}^{(B)}_{X_i} = \mathbf{B}_{X_i},$$

$$\forall b \in 1, \ldots, B \quad \left|\{x : x \in \text{Scope}[\psi] \wedge \psi \in \mathbf{B}^{(b)}_{X_i}\}\right| \leq i + 1.$$

    3. Eliminate variable X_i. For each mini-bucket, take the product of all factors within it. Sum-out X_i on one of the mini-bucket and max-out (or min-out) X_i on the remaining mini-buckets. The results are new factors ψ^*_ξ, which are then added to the factors.

    for $b \leftarrow 1, \ldots, B$ **do**

$$\psi^*_\xi \leftarrow \begin{cases} \sum\limits_{X_i} \prod\limits_{\psi_\xi \in \mathbf{B}^{(b)}_{X_i}} \psi_\xi & \text{if } b = 1 \\ \max\limits_{X_i} \prod\limits_{\psi_\xi \in \mathbf{B}^{(b)}_{X_i}} \psi_\xi & \text{otherwise} \end{cases}, \qquad \Psi \leftarrow \Psi \cup \psi^*_\xi.$$

  The remaining factors ψ are constants.

  return *Bound on the partition function* $\hat{Z} \leftarrow \prod\limits_{\psi \in \Psi} \psi$.

end

on directly eliminating each variable, by performing the operations of product and sum (i.e., marginalization). We first present the bucket elimination algorithm for exact inference.

Bucket Elimination

Bucket elimination (BE) (Dechter, 1996) is a simple and effective algorithm for performing exact inference. The high-level idea is to first organize the factors of a PGM into so-called *buckets* using a variable ordering. Then, using the same variable ordering, eliminate each variable accordingly from each bucket (i.e., by applying product and marginalization operations). Each step of the elimination process creates new factors, which are then assigned to an existing bucket based on the variable ordering and the scope of the factor. The details of bucket elimination are presented in Algorithm 2.1. We focus our discussion on the task of computing the partition function, but the algorithm can be derived for other inference tasks. From Algorithm 2.1, we can see that the computational complexity of bucket elimination is exponential in the size of the largest bucket (step 2). This is also referred to as the max cluster size or *induced width*. To alleviate the complexity burden, we next present mini-bucket elimination, a general approximate inference scheme to bucket elimination.

Mini-bucket Elimination

Mini-bucket elimination (MB) (Dechter and Rish, 2003) is an approximation of bucket or variable elimination (Dechter, 1996; Zhang and Poole, 1994). MB returns an upper or lower bound on the partition function. MB is shown in Algorithm 2.2. We mentioned previously the complication for BE is the intractable induced width, which is exponential in general. The basic idea is that MB overcomes this issue by partitioning and splitting the bucket into so-called *mini-buckets* shown in step 2 of Algorithm 2.2. The maximum size of the mini-bucket is controlled by an input parameter called the *i-bound*. This relaxation allows the complexity to be exponential in the *i-bound* as a trade-off to accuracy. Subsequently

in step 3, marginalization (sum-out) is applied to one of the mini-buckets and depending on if an upper or lower bound on the partition is selected, maximization or minimization is applied to all remaining mini-buckets. Processing the newly created functions then follows that of BE. MB is a simple and straightforward approximation scheme and achieves tighter bounds with higher *i-bound* values.

2.3.2 Monte Carlo Methods

The main idea behind Monte Carlo[2] methods (Metropolis and Ulam, 1949) is to leverage randomness (i.e., random numbers) and invoke the (strong and weak) law of large numbers (LLN). LLN states that the empirical or sample average for a sequence of independent and identically distributed (i.i.d.) random variables, $X_i^{(1)}, \ldots, X_i^{(D)}$ (i.e., $1/D \sum_{d=1}^{D} X_i^{(d)}$), converges in probability to the expectation of X_i (i.e., $\mathbb{E}[X_i]$) with high probability (Wasserman, 2004). The random variables are i.i.d. if all the random variable have the same probability distribution and are mutually independent. Monte Carlo methods encompass a general family of stochastic approximation techniques and can be used for PGMs to solve either or both of the following problems:

1. Generate instantiations, $\{\widetilde{\mathbf{x}}^{(1)}, \ldots, \widetilde{\mathbf{x}}^{(D)}\}$, to all or some of the variables from a given PGM using random sampling.

2. Estimate expectations of functions under the distribution, $P_{\boldsymbol{\theta}}(\boldsymbol{X})$, represented by a given PGM.

In general, sampling from high-dimensional distributions is hard. However, under the framework of PGMs, we can devise and implement amenable and efficient algorithms, which we describe next.

[2]Also referred to as *Ordinary* Monte Carlo or i.i.d. Monte Carlo as compared to Markov Chain Monte Carlo (Brooks et al., 2011)

Algorithm 2.3: Forward Sampling for BNs

Input: Topological ordering $o = [X_1, X_2, \ldots, X_{|\boldsymbol{X}|}]$,
Bayesian network $\mathcal{M}_{BN} \triangleq \langle \boldsymbol{X}, \boldsymbol{\theta}, \mathrm{G}_{BN} \rangle$
Output: An instantiation to all variables (i.e., sample)
begin
  for $i \leftarrow 1$ *to* $|\boldsymbol{X}|$ **do**
    1. Get the assignment $(\widetilde{x}_1, \ldots, \widetilde{x}_{n-1})$ to the parents of X_i: $\widetilde{\mathbf{x}}_{\mathrm{Pa}[X_i]}$
    2. Sample $x_i \sim P(X_i \,|\, \widetilde{\mathbf{x}}_{\mathrm{Pa}[X_i]})$
  return $\widetilde{\mathbf{x}}$
end

Forward Sampling

Generating samples from a BN is a straightforward process. The sampling procedure is known as *forward sampling* or *probabilistic logic sampling* (Henrion, 1986), which is described by Algorithm 2.3. At a high-level, using a topological ordering of the variables, the sampling process samples one variable at a time according to the conditional distribution corresponding to the variable. The main consideration is step 2 of the algorithm. Here, we can use a (pseudo) random number generator to sample a value x from the condition distribution (i.e., CPT). Specifically, consider the domain, $\mathrm{Val}(X_i) = \{x^1, \ldots, x^j\}$, for some variable X_i, corresponding to parameters $(\theta_1, \ldots, \theta_j)$. First, d number of subintervals are created as follows: $[0, \theta_1), [\theta_1, \theta_1 + \theta_2), \ldots, [\theta_{j-1}, \sum_{k=1}^{j-1} \theta_k), [\theta_j, 1]$. Next, a value c is randomly generated (i.e., sampled) from an uniform distribution defined over the interval $[0, 1]$. If c is in the d-th subinterval, then the value x^d is selected as the sample. The forward sampling process is also referred to as simulating the BN (Darwiche, 2009).

Algorithm 2.4: Forward Sampling with Likelihood Weighting for BNs

Input: Topological ordering $o = [X_1, X_2, \ldots, X_{|\boldsymbol{X}|}]$,
Bayesian network $\mathcal{M}_{BN} \triangleq \langle \boldsymbol{X}, \boldsymbol{\theta}, \mathrm{G}_{BN} \rangle$, evidence set $\boldsymbol{E}$
Output: An instantiation to all variables (i.e., sample) and a corresponding weight w
begin
  Initialize: $w \leftarrow 1$
  for $i \leftarrow 1$ *to* $|\boldsymbol{X}|$ **do**
    1. Get the assignment $(\widetilde{x}_1, \ldots, \widetilde{x}_{n-1})$ to the parents of X_i: $\widetilde{\mathbf{x}}_{\mathrm{Pa}[X_i]}$
    if $X_i \in \boldsymbol{E}$ **then**
      2. Assign evidence value $x_i \leftarrow e_i$
      3. Compute likelihood weight $w \leftarrow w \times P(\widetilde{x}_i \,|\, \widetilde{\mathbf{x}}_{\mathrm{Pa}[X_i]})$
    else
      4. Sample $x_i \sim P(X_i \,|\, \widetilde{\mathbf{x}}_{\mathrm{Pa}[X_i]})$
  return $(\widetilde{\mathbf{x}}, w)$
end

Likelihood Weighting

To handle the harder task of computing conditional probability queries (i.e., $P(\boldsymbol{Y} \,|\, \boldsymbol{E} = \boldsymbol{e})$), a priori, we can utilize a naïve approach known as *rejection sampling*[3]. Given an evidence set, $\boldsymbol{E} = \boldsymbol{e}$, we follow the aforementioned forward sampling process, but *reject* samples that are not consistent with $\boldsymbol{e}$. However, we can immediately notice an issue with this approach. That is, in the case when $P(\boldsymbol{E} = \boldsymbol{e})$ has a low probability. For example, if $P(\boldsymbol{E} = \boldsymbol{e}) = .001$, then for every $1,000$ samples collected, we can expect to reject 999 of the samples. Therefore, rejection sampling under this scenario is not practical. On the other hand, if we use forward sampling and *clamp* the evidence, meaning when the sampling procedure reaches an evidence variable that variable is assigned the value of evidence, then this procedure can generate

[3]Also known as *hit-or-miss* in older literature on Monte Carlo methods (Hammersley and Handscomb, 1964).

incorrect results in general because it is not sampling according to the conditional probability distribution (Koller and Friedman, 2009).

One way to overcome the rejection issue, and to sample more efficiently, is to use the so-called *likelihood weighting* (LW) approach (Fung and Chang, 1989; Shachter and Peot, 1989). In LW, we use forward sampling with clamping, but each sample is weighted according to the likelihood of the evidence. Namely, the likelihood is the product of the probabilities for the evidence variables in each sample. The procedure is showing in Algorithm 2.4. The result of this process is that we now have *weighted* samples. LW is a special case of a general weighted sampling framework called *importance sampling*, which we present next.

Importance Sampling

We previously mentioned that Monte Carlo methods are also used for computing expectations. Importance sampling (IS) (Marshall, 1956) is one such technique commonly used to evaluate the expectation of a real function H according to a probability distribution P (shorthand for $P_{\boldsymbol{\theta}}$), namely $\mathbb{E}_P[H(\boldsymbol{X})] = \sum_{\widetilde{\mathbf{x}} \in \boldsymbol{X}} H(\widetilde{\mathbf{x}}) P(\widetilde{\mathbf{x}})$. The general idea is to generate samples $\widetilde{\mathbf{x}}^{(1)}, ..., \widetilde{\mathbf{x}}^{(N)}$ from a proposal distribution Q and estimate $\mathbb{E}_P[H(\boldsymbol{X})]$ as

$$\mathbb{E}_P[f(\boldsymbol{X})] \approx \frac{\sum_{i=1}^{N} H(\widetilde{\mathbf{x}}^{(i)}) w(\widetilde{\mathbf{x}}^{(i)})}{\sum_{i=1}^{N} w(\widetilde{\mathbf{x}}^{(i)})},$$

where $w(\widetilde{\mathbf{x}}^{(i)}) = P(\widetilde{\mathbf{x}}^{(i)}) / Q(\widetilde{\mathbf{x}}^{(i)})$ is the *importance weight* of sample $\widetilde{\mathbf{x}}^{(i)}$. Note the importance weight needs to be known only up to a multiplicative constant. Q does not have to be positive everywhere. It is sufficient for $Q > 0$ when $H(\widetilde{\mathbf{x}}) P(\widetilde{\mathbf{x}}) \neq 0$. Importance sampling can be used to estimate the partition function (normalizing constant) and single variable marginal probabilities. In this dissertation, we focus on computing the single variable marginal

Algorithm 2.5: Gibbs Sampling for BNs

Input: Evidence set $\boldsymbol{E}$, Variables to sample $\boldsymbol{Y} = \boldsymbol{X} \setminus \boldsymbol{E}$, Number of samples N, Bayesian network $\mathcal{M}_{BN} \triangleq \langle \boldsymbol{X}, \boldsymbol{\theta}, \mathrm{G}_{BN} \rangle$
Output: Samples
begin
  Initialize $\widetilde{\mathbf{x}}^{(0)} \cup \{x_i \leftarrow e_i : e_i \in \boldsymbol{E}\}$
  for $s \leftarrow 1$ *to* N **do**
    $\widetilde{\mathbf{x}}^{(s)} \leftarrow \widetilde{\mathbf{x}}^{(s-1)}$
    for $Y_i \in \boldsymbol{Y}$ **do**
      Sample $y_i \sim P_{\boldsymbol{\theta}}(Y_i \mid \boldsymbol{X} \setminus Y_i)$
  return $(\widetilde{\mathbf{x}}^{(s)})$ for $s = 1$ to N
end

probability, which can be estimated as

$$\widehat{P}_N(\widetilde{x}) = \frac{\sum_{i=1}^{N} \mathbb{1}(\widetilde{\mathbf{x}}^{(i)}) w(\widetilde{\mathbf{x}}^{(i)})}{\sum_{i=1}^{N} w(\widetilde{\mathbf{x}}^{(i)})}, \tag{2.1}$$

where $\mathbb{1}(\widetilde{\mathbf{x}}) = 1$ iff $\widetilde{\mathbf{x}}$ contains the assignment $\widetilde{x}$, and 0 otherwise, and $w(\widetilde{\mathbf{x}}) = \exp(\sum_i \theta_i f_i(\widetilde{\mathbf{x}}))$ / $Q(\widetilde{\mathbf{x}})$. We will make the standard assumption that Q is a BN (Fung and Chang, 1989; Ortiz and Kaelbling, 2000; Cheng and Druzdzel, 2001; Gogate, 2009), since it is computationally efficient to generate independent samples from a BN using forward sampling. The quality of estimation, namely the accuracy of $\widehat{P}_N(\widetilde{x})$ is highly dependent on how far Q is from P, and as a result most of the research on importance sampling is about designing a good Q. In Chapter 5, we design a Q for parameter tied graphical models.

Gibbs Sampling

While LW improves on rejection sampling, the performance of LW is effected by the location of the evidence nodes. Specifically, if the set of evidence variables are at the leaves of the

network (nodes with no child nodes), then the samples will be generated from the prior distribution, which can be, in general, divergent from the posterior distribution. One way to overcome this issue is to utilize *Gibbs* sampling (Geman and Geman, 1984). The procedure is shown in Algorithm 2.5. In Gibbs sampling, starting with an initialized sample, each of variables in $\boldsymbol{Y}$ are then iteratively sampled based on conditioning the current sampled values for all other variables (line 8 of the algorithm). Under the assumption that the conditional distributions are easy to sample, the process is straightforward. For example, given two conditional distributions $P_{\boldsymbol{\theta}}(Y|X)$ and $P_{\boldsymbol{\theta}}(X|Y)$. We set $Y = y^{(0)}$ and $X = x^{(0)}$ to some initial value. Then, starting with Y, we next repeat the following sampling process: $y^{(1)} \sim P_{\boldsymbol{\theta}}(Y|X = x^{(0)})$, $x^{(1)} \sim P_{\boldsymbol{\theta}}(X|Y = y^{(1)})$, $y^{(2)} \sim P_{\boldsymbol{\theta}}(Y|X = x^{(1)})$, and so on. Note, we can also start with X, in which the sampling starts with $x^{(1)} \sim P_{\boldsymbol{\theta}}(X|Y = y^{(0)})$. The intuition is, by following this process for the PGM case, the various states of the variable or information moves through the network. In particular, the downstream information from the evidence nodes are collected and thereby samples will be generated closer to the posterior distribution. Note that, although Algorithm 2.5 is specified for BNs, Gibbs sampling is a general technique that can also be used for MNs. Referring back to the simple example, we have the resulting sequence of samples: $(y^{(0)}, x^{(0)}), (y^{(1)}, x^{(1)},), (y^{(2)}, x^{(2)},), (y^{(3)}, x^{(3)},), \ldots,$ etc. This sequence satisfies the Markov chain property because the current sample $(y^{(i)}, x^{(i)})$ depends only on the previous sample $(y^{(i-1)}, x^{(i-1)})$. In fact, Gibbs sampling is in the general family of sampling techniques called Markov chain Monte Carlo (MCMC). In Chapter 5, we develop a slice importance sampling approach belonging to the MCMC family.

2.4 Learning

In Section 2.2, we mentioned one way to acquire a PGM is to construct the model manually, often with the aid of domain experts. However, this is clearly infeasible and not practical due to the large amount of knowledge and time required to develop a useful model. On the

other hand, it is frequently the case that we have access to examples (i.e., data) generated from models for which we are interested in. Thus, if we assume the domain we are interested in modeling is governed by some distribution P^*, which is induced by some PGM, we can develop *learning* algorithms that use data (or samples), $\mathcal{D} = \{\widetilde{\mathbf{x}}^{(1)}, \ldots, \widetilde{\mathbf{x}}^{(D)}\}$ generated from P^*, to construct PGMs.

The two main learning tasks for PGMs are: (1) learn (or estimate) the parameters of a given fixed PGM structure; and (2) learn both the parameter and structure of a PGM. In this dissertation, we focus on learning the parameters generatively from a given complete or fully observed dataset and PGM structure.

2.4.1 Maximum Likelihood Estimation

One motivation for learning a PGM is to use it for reasoning (e.g., perform inference on the model). The goal of this setting is often framed as *density estimation*, that is to construct a PGM representing some distribution P which is close to P^*. Intuitively, we would like to learn models that assign high probability to the given data. The statistical connection between data and models is commonly expressed as the *likelihood* (i.e., the of likelihood the data, given a model), which we formally define below.

Definition 2.16. (Likelihood and log-likelihood function). *Let $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ be a set of random variables with joint probability distribution P where the set of parameters $\boldsymbol{\theta}$ can take values from some parameter space Θ. Given the dataset or observed values over the random variables, $\mathcal{D} = \{\widetilde{\mathbf{x}}^{(1)}, \ldots, \widetilde{\mathbf{x}}^{(D)}\}$, the likelihood function $\mathcal{L} : \Theta \to [0, \infty)$ is defined as*

$$\mathcal{L}(\boldsymbol{\theta} : \mathcal{D}) = P(\mathcal{D} : \boldsymbol{\theta}), \quad \boldsymbol{\theta} \in \Theta.$$

It is often algebraically convenient and numerically stable to work with the natural logarithm of the likelihood function, which we refer to as the log-likelihood function and is defined as

$$\ell(\boldsymbol{\theta} : \mathcal{D}) = \log \mathcal{L}(\boldsymbol{\theta} : \mathcal{D}).$$

The log-likelihood can be viewed as a function of the parameters $\boldsymbol{\theta}$ for a given fixed dataset $\mathcal{D}$. If $\mathcal{D}$ is i.i.d., then the log-likelihood function is

$$\ell(\boldsymbol{\theta} : \mathcal{D}) = \log \prod_{d=1}^{D} P_{\boldsymbol{\theta}}(\widetilde{\mathbf{x}}^{(d)}) = \sum_{d=1}^{D} \log P_{\boldsymbol{\theta}}(\widetilde{\mathbf{x}}^{(d)}).$$

Note, since $\ell : \Theta \to (-\infty, 0]$, the log-likelihood is not necessarily a normalized probability distribution.

As mentioned to earlier, we are interested in estimating parameter values with higher likelihood since it is more likely to have generated the data. To formulate the learning task, we define a hypothesis space, Θ (i.e., possible parameterization) and an objective function to evaluate different parameter settings relative to the dataset $\mathcal{D}$. We use the likelihood function as the objective function and the value that maximizes the likelihood is referred to as the maximum likelihood estimator. Formally, we defined the learning problem as

$$\max_{\boldsymbol{\theta} \in \Theta} \ \ell(\boldsymbol{\theta} : \mathcal{D}).$$

This method of parameter learning or estimation is referred to as *maximum likelihood estimation* (MLE). Note that since the log-likelihood function is monotonically related to the likelihood function, maximizing $\ell(\boldsymbol{\theta} : \mathcal{D})$ is equivalent to maximizing $\mathcal{L}(\boldsymbol{\theta} : \mathcal{D})$. MLE is a general technique for learning the parameters of a PGM using data (or observations) which we describe next.

MLE for Bayesian Networks

In this section, we describe the MLE setting for learning the parameters of a given Bayesian network structure. Given a set of random variables $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ with joint probability distribution $P_{\boldsymbol{\theta}}(\boldsymbol{X})$ defined over a Bayesian network, $\mathcal{M}_{BN}$, and a fully observed dataset, $\mathcal{D} = \{\widetilde{\mathbf{x}}^{(1)}, \ldots, \widetilde{\mathbf{x}}^{(D)}\}$. The likelihood function is defined as

$$\mathcal{L}(\boldsymbol{\theta} : \mathcal{D}) = \prod_{d=1}^{D} \prod_{i=1}^{n} P_{\boldsymbol{\theta}}(\widetilde{x}_i^{(d)} \mid \widetilde{\mathbf{x}}_{\mathrm{Pa}[X_i]}^{(d)}),$$

where $\widetilde{x}_i^{(d)}$ is the value assigned to X_i in $\widetilde{\mathbf{x}}^{(d)}$ and $\widetilde{\mathbf{x}}^{(d)}_{\mathrm{Pa}[X_i]}$ are the values assigned to the parents of X_i in $\widetilde{\mathbf{x}}^{(d)}$. Since the distribution represented by a Bayesian network is a product of (local) condition probability distributions, the likelihood decomposes as a product of independent terms. The parameters to be estimated are defined as

$$\theta_i = P(\widetilde{x}_i \,|\, \widetilde{\mathbf{x}}_{\mathrm{Pa}[X_i]}).$$

Following the MLE solution for a discrete multinomial distribution (Darwiche, 2009), the estimates for the parameters admit an analytical solution. We first specify the following two indicator functions for collecting sufficient statistics from the dataset.

$$\mathbb{1}(\widetilde{x}_i,\ \widetilde{\mathbf{x}}_{\mathrm{Pa}[X_i]} : \widetilde{\mathbf{x}}^{(d)}) = \begin{cases} 1 & (\widetilde{x}_i \wedge \widetilde{\mathbf{x}}_{\mathrm{Pa}[X_i]} \in \widetilde{\mathbf{x}}^{(d)}) \vee (\mathrm{Pa}[X_i] = \emptyset \wedge \widetilde{\mathbf{x}}_{\mathrm{Pa}[X_i]} \in \widetilde{\mathbf{x}}^{(d)}) \\ 0 & \text{otherwise} \end{cases}.$$

$$\mathbb{1}(\widetilde{\mathbf{x}}_{\mathrm{Pa}[X_i]} : \widetilde{\mathbf{x}}^{(d)}) = \begin{cases} 1 & (\widetilde{\mathbf{x}}_{\mathrm{Pa}[X_i]} \in \widetilde{\mathbf{x}}^{(d)}) \\ 0 & \text{otherwise} \end{cases}.$$

The sufficient statistics contains all the information required to compute the estimate of a parameter. Therefore, the MLE for the parameters is computed as

$$\theta_i = \frac{\sum_{d=1}^{D} \mathbb{1}(x_i, \widetilde{\mathbf{x}}_{\mathrm{Pa}[X_i]} : \widetilde{\mathbf{x}}^{(d)})}{\sum_{d=1}^{D} \mathbb{1}(\widetilde{\mathbf{x}}_{\mathrm{Pa}[X_i]} : \widetilde{\mathbf{x}}_i^{(d)})}.$$

The MLE solution is sensitive to the size of the dataset. In general, the variance of the solution decreases as the size of the dataset increases.

MLE for Markov Networks

In this section, we describe the MLE setting for learning the parameters of a given Markov network structure. Given a collection of random variables $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ with joint

probability distribution $P_{\boldsymbol{\theta}}(\boldsymbol{X})$ defined over a Markov network and a fully observed dataset $\mathcal{D} = \{\widetilde{\mathbf{x}}^{(1)}, \ldots, \widetilde{\mathbf{x}}^{(D)}\}$. We use a log-linear model representation of the Markov network and the log-likelihood function, which is defined as

$$\begin{aligned}
\log P(\mathcal{D} : \boldsymbol{\theta}) &= \sum_{d=1}^{D} \log P_{\boldsymbol{\theta}}(\widetilde{\mathbf{x}}^{(d)}) \\
&= \sum_{d=1}^{D} \log \left\{ \frac{1}{Z(\boldsymbol{\theta})} \exp \left\{ \sum_{i=1}^{m} \theta_i f_i(\widetilde{\mathbf{x}}_i^{(d)}) \right\} \right\} \\
&= \sum_{d=1}^{D} \sum_{i=1}^{m} \theta_i f_i(\widetilde{\mathbf{x}}_i^{(d)}) - D \log Z(\boldsymbol{\theta}) \\
&= \sum_{i=1}^{m} \theta_i \sum_{d=1}^{D} f_i(\widetilde{\mathbf{x}}_i^{(d)}) - D \log Z(\boldsymbol{\theta}) \\
&= D \sum_{i=1}^{m} \theta_i \mathbb{E}_{\mathcal{D}}[f_i(\widetilde{\mathbf{x}}_i)] - D \log Z(\boldsymbol{\theta}) \\
\frac{1}{D}\ell(\mathcal{D} : \boldsymbol{\theta}) &= \sum_{i=1}^{m} \theta_i \mathbb{E}_{\mathcal{D}}[f_i(\widetilde{\mathbf{x}}_i)] - \log Z(\boldsymbol{\theta}),
\end{aligned}$$

where $\log Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{X}} \exp\left\{ \sum_{i=1}^{m} \theta_i f_i(\mathbf{x}_i) \right\}$. Since the MLE does not admit a closed-form solution, we can instead use any standard gradient-based method (i.e., gradient ascent) to solve the optimization problem. The gradient w.r.t. θ_i is computed as

$$\begin{aligned}
\frac{1}{D}\frac{\partial \ell(\mathcal{D} : \boldsymbol{\theta})}{\partial \theta_i} &= \mathbb{E}_{\mathcal{D}}[f_i(\widetilde{\mathbf{x}}_i)] - \frac{1}{Z(\boldsymbol{\theta})} f_i(\mathbf{x}_i) \exp\left\{\theta_i f_i(\mathbf{x}_i)\right\} \\
&= \mathbb{E}_{\mathcal{D}}[f_i(\widetilde{\mathbf{x}}_i)] - \frac{\exp\left\{\theta_i f_i(\mathbf{x}_i)\right\}}{Z(\boldsymbol{\theta})} f_i(\mathbf{x}_i) \\
&= \mathbb{E}_{\mathcal{D}}[f_i(\widetilde{\mathbf{x}}_i)] - \mathbb{E}_{P_{\boldsymbol{\theta}}(\boldsymbol{X})}[f_i(\mathbf{x}_i)].
\end{aligned}$$

2.4.2 Maximum Pseudolikelihood Estimation

Unlike Bayesian networks, the likelihood function does not decompose for Markov networks. Therefore, evaluating the likelihood, as well as the gradient, requires computing the partition function. Since computing the latter is #P-hard in general, alternative computationally

tractable approximations to likelihood, such as the pseudolikelihood (Besag, 1975), are often used in practice. Given an instance $\widetilde{\mathbf{x}}$ and a Markov network G_{MN} representing some positive distribution $P_{\boldsymbol{\theta}}$, recalling the chain rule from Definition 2.2, we can write the probability of the instance as

$$P_{\boldsymbol{\theta}}(\widetilde{\mathbf{x}}) = \prod_{i=1}^{n} P(\widetilde{x}_i \,|\, \widetilde{x}_1, \ldots, \widetilde{x}_{i-1}).$$

The likelihood can be approximated as

$$P_{\boldsymbol{\theta}}(\widetilde{\mathbf{x}}) \approx \prod_{i=1}^{n} P(\widetilde{x}_i \,|\, \widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n),$$

which is the conditional probability of x_i given all other variables. Applying the local Markov property, we can simplify the approximation to

$$P_{\boldsymbol{\theta}}(\widetilde{\mathbf{x}}) \approx \prod_{i=1}^{n} P(\widetilde{x}_i \,|\, \widetilde{\mathbf{x}}_{\mathrm{Mb}[X_i]}).$$

Furthermore, the complexity of the likelihood function has been reduced from a global exponential summation (i.e., partition function) to a *local* partition function requiring only a summation over the values of X_i. The more tractable formulation for the conditional probabilities becomes

$$P(\widetilde{x}_i \,|\, \widetilde{\mathbf{x}}_{\mathrm{Mb}[X_i]}) = \frac{P(\widetilde{x}_i, \widetilde{\mathbf{x}}_{\mathrm{Mb}[X_i]})}{\sum\limits_{x_i \in \mathrm{Val}(X_i)} P(x_i, \widetilde{\mathbf{x}}_{\mathrm{Mb}[X_i]})}.$$

MPLE for Log-linear Models

In this section, we describe the maximum pseudo-loglikelihood (MPLE) setting for learning the parameters of a given log-linear PGM structure. Given a collection of random variables $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ with joint probability distribution $P_{\boldsymbol{\theta}}(\boldsymbol{X})$ defined over a log-linear PGM

and a fully observed dataset $\mathcal{D} = \{\widetilde{\mathbf{x}}^{(1)}, \ldots, \widetilde{\mathbf{x}}^{(D)}\}$. The log-likelihood function is defined as

$$\begin{aligned}
\log P(\mathcal{D} : \boldsymbol{\theta}) &= \sum_{d=1}^{D} \log P_{\boldsymbol{\theta}}(\widetilde{\mathbf{x}}^{(d)}) \\
&= \sum_{d=1}^{D} \sum_{i=1}^{n} \log \tilde{P}_{\boldsymbol{\theta}}(\widetilde{x}_i^{(d)} \mid \widetilde{\mathbf{x}}_{\{-x_i\}}^{(d)}) \\
&= \sum_{d=1}^{D} \sum_{i=1}^{n} \log \frac{P(\widetilde{x}_i^{(d)},\ \widetilde{\mathbf{x}}_{\{-x_i\}}^{(d)})}{P(\widetilde{\mathbf{x}}_{\{-x_i\}}^{(d)})} \\
&= \sum_{d=1}^{D} \sum_{i=1}^{n} \log P(\widetilde{x}_i^{(d)},\ \widetilde{\mathbf{x}}_{\{-x_i\}}^{(d)}) - \log \sum_{x \in \mathrm{Val}(X_i)} P(x,\ \widetilde{\mathbf{x}}_{\{-x_i\}}^{(d)}).
\end{aligned}$$

where $\{-x_i\} = x_1, \ldots, x_{i-1}, x_{i+1}, x_n$. By using the Markov local property, the other variables $\widetilde{\mathbf{x}}_{\{-x_i\}}$ simplify to the Markov blanket $\mathrm{Mb}[X_i]$. The pseudo-loglikelihood is defined as

$$\begin{aligned}
p\ell(\boldsymbol{\theta} : \mathcal{D}) &= \sum_{d=1}^{D} \sum_{i=1}^{n} \log \left\{ \exp\left\{\theta_i f_i(\widetilde{x}_i^{(d)},\ \widetilde{\mathbf{x}}_{\mathrm{Mb}[X_i]}^{(d)})\right\}\right\} - \log \sum_{x \in \mathrm{val}(X_i)} \exp\left\{\theta_i f_i(x,\ \widetilde{\mathbf{x}}_{\mathrm{Mb}[X_i]}^{(d)})\right\} \\
&= \sum_{d=1}^{D} \sum_{i=1}^{n} \theta_i f_i(\widetilde{x}_i^{(d)},\ \widetilde{\mathbf{x}}_{\mathrm{Mb}[X_i]}^{(d)}) - \log \sum_{x \in \mathrm{val}(X_i)} \exp\left\{\theta_i f_i(x,\ \widetilde{\mathbf{x}}_{\mathrm{Mb}[X_i]}^{(d)})\right\}.
\end{aligned}$$

Since the MPLE does not admit a closed-form solution, we can instead use any standard gradient-based method (i.e., gradient ascent) to solve the optimization problem. The gradient w.r.t. θ_i is computed as

$$\frac{\partial p\ell(\boldsymbol{\theta} : \mathcal{D})}{\partial \theta_i} = \frac{1}{D} \sum_{d=1}^{D} \left(f_i(\widetilde{x}_i^{(d)},\ \widetilde{\mathbf{x}}_{\mathrm{Mb}[X_i]}^{(d)}) - \frac{\sum_{x \in \mathrm{val}(X_i)} f_i(x,\ \widetilde{\mathbf{x}}_{\mathrm{Mb}[X_i]}^{(d)}) \exp\left\{\theta_i f_i(x,\ \widetilde{\mathbf{x}}_{\mathrm{Mb}[X_i]}^{(d)})\right\}}{\sum_{x \in \mathrm{val}(X_i)} \exp\left\{\theta_i f_i(x,\ \widetilde{\mathbf{x}}_{\mathrm{Mb}[X_i])}^{(d)})\right\}} \right).$$

2.5 L_2 Regularization

MLE is prone to overfitting when the size of the training dataset is small compared to the number of parameters. One way to combat overfitting, using the Bayesian approach, is to introduce a prior distribution over the parameters. A zero-mean Gaussian (uniformed) prior

distribution is the standard choice. After taking logs, the regularizer function is of the form

$$-\frac{\lambda}{2}||\boldsymbol{\theta}||_2^2 = -\frac{\lambda}{2}\sum_{i=1}^{m}(\theta_i)^2.$$

This penalty term, when added to the learning objective, is generally referred to as L_2 regularization. L_2 penalizes parameter values with large magnitude, which aides in smoothing out the variations in the data. The penalty grows quadratically and thus larger parameter values are penalized more than smaller parameter values. Since the penalty term is concave in the parameters, adding it with a concave learning objective preserves concavity and the resulting objective can be efficiently optimized using gradient-based methods. The hyperparameter $\lambda \propto 1/\sigma^2$ controls the variance of the Gaussian distribution. Due to the inverse relation, high values of λ result in low variance and vice versa.

2.6 Quantization and k-means

Quantization is the process of mapping a set of real numbers to a smaller set. Formally, a quantization function $\mathcal{Q}$, is a many-to-one mapping from a set of real numbers $\boldsymbol{A}$ to a set of real numbers $\boldsymbol{B}$, such that $|\boldsymbol{A}| \geq |\boldsymbol{B}|$. Our aim is to find a quantization that minimizes the average quantization error, namely to optimize the objective

$$\min \ \frac{1}{|\boldsymbol{A}|}\sum_{a\in\boldsymbol{A}}|a - \mathcal{Q}(a)|.$$

A *k-level* quantizer fixes the size of $\boldsymbol{B}$, namely $|\boldsymbol{B}| = k$.

Closely related to the optimal *k-level* quantization problem is the k-means clustering problem (Pollard, 1982; Bottou and Bengio, 1994). Note that in our work, we only need to solve the 1-dimensional k-means clustering problem, which admits a polynomial time algorithm, $O(m^2k)$, via dynamic programming (Wang and Song, 2011). Given a set $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_m\}$ of real numbers, the 1-dimensional k-means algorithm seeks to partition $\boldsymbol{\theta}$ into k clusters such

that the following objective function is minimized

$$\sum_{j=1}^{m}(\theta_j - \mu_{a_j})^2,$$

where $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_k\}$ are the cluster centers or means and $a_j \in \{1, \ldots, k\}$ denotes the cluster assignments, namely the cluster to which θ_j is assigned to. The cluster mean, μ_i, is given by

$$\mu_i = \frac{1}{\sum_{j=1}^{m} \mathbb{1}(a_j, i)} \sum_{j=1}^{m} \theta_j \cdot \mathbb{1}(a_j, i),$$

where $\mathbb{1}(a_j, i)$ is an indicator function which equals 1 if $a_j = i$ and 0 otherwise.

From any k-clustering $\langle \boldsymbol{a}, \boldsymbol{\mu} \rangle$ where $\boldsymbol{a} = \{a_1, \ldots, a_m\}$ and $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_k\}$ of $\boldsymbol{\theta}$, we can define an equivalent quantizer such that $\mathcal{Q}(\theta_j) = \mu_{a_j}$.

CHAPTER 3

LEARNING PARAMETER TIED BAYESIAN NETWORKS

3.1 Introduction

In this chapter, we describe a parameter tying algorithm as a alternative regularization technique (variance reduction) for parameter learning in Bayesian networks. To simplify the presentation, we refer to and notate the Bayesian network as log-linear form. The idea of parameter tying is to force several parameters of the graphical model to take the same value. We propose a greedy approach that ties parameters by *quantizing* the learned parameters. Empirically, our approach improves the generalization of a learned model (i.e., combats the issue of overfitting) with fewer parameters.

A number of (automatic) parameter tying schemes have been investigated. (Nowlan and Hinton, 1991) proposed utilizing a Gaussian mixture model (prior) for parameter sharing to simplify neural networks. While Gaussian mixtures are a general model, the trade-offs with this approach are significant, e.g., added complexity resulting from an increase in the number of parameters that need to be selected with validation data, the creation of a large number of local minima, and slow convergence. (Han et al., 2016) investigated compression and pruning in neural networks, utilizing a form of k-means quantization and parameter tying. They incorporate these elements into their pipelined algorithm as a post-processing step and empirically validate the performance. (Liu and Page, 2013) utilized k-means to initialize a nonparametric Bayesian (hard) tying approach. Recently, there has been growing interest in locality-sensitive hashing (Gionis et al., 1999) where, at a high-level, the objective is to develop hash functions such that the probability of collision for similar items is maximized to solve the approximate nearest neighbor (Indyk and Motwani, 1998) search problem. Quantization is generally used as a subroutine to partition a lower-dimensional feature space (Wang et al., 2016). Deep learning neural networks have been proposed to learn such hash functions (Zhu et al., 2016).

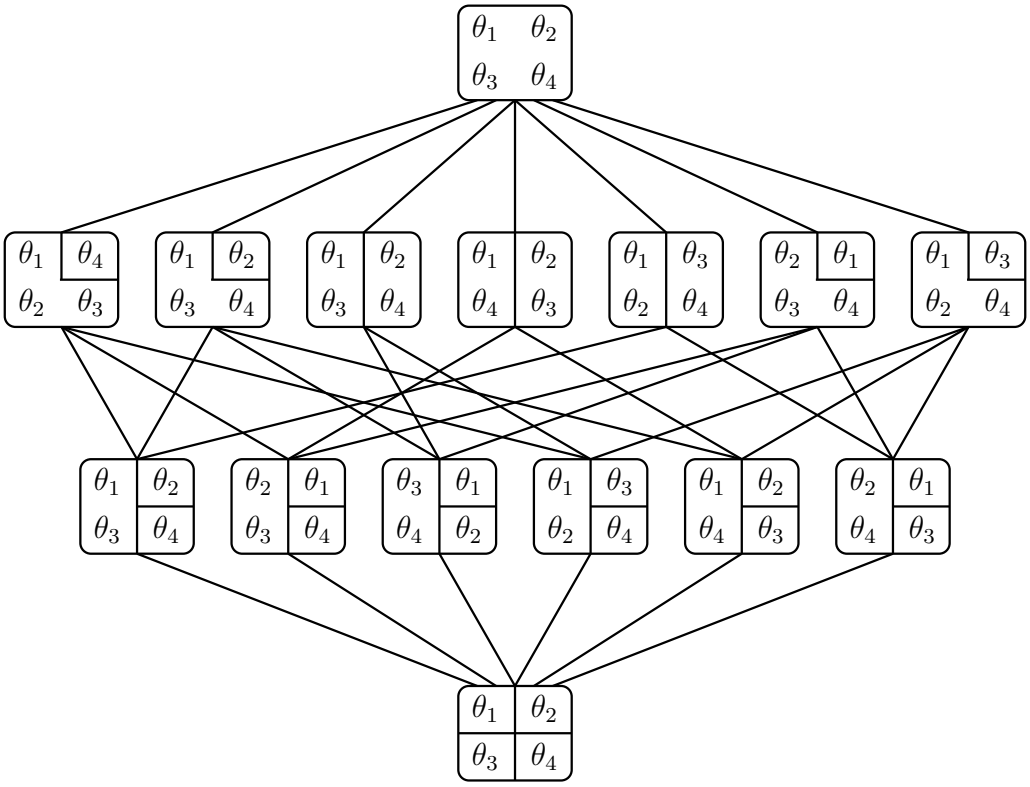


Figure 3.1. Hasse diagram showing the partitions for 4 parameters.

3.2 Problem Definition and Approach

We are interested in learning a parameter tied graphical model which we define as follows.

Definition 3.1. (Parameter tied graphical model) *A parameter tied graphical model (PTGM) is a triple* $\mathcal{M}_t \triangleq \langle \boldsymbol{X}, \boldsymbol{\theta}, \mathcal{C} \rangle$*, where* $\boldsymbol{X}$ *is the set of variables,* $\boldsymbol{\theta}$ *is the set of parameters, and* $\mathcal{C}$ *is the set of equality constraints of the form* $\theta_i = \theta_j$ *for some* $\theta_i, \theta_j \in \boldsymbol{\theta}$ *such that* $i \neq j$*.*

We are interested in learning the optimal PTGM $\mathcal{M}_t^*$. Formally, we defined the problem as: given training data $\mathcal{D}$ on variable set $\boldsymbol{X}$, find the constraint set $\mathcal{C}$ and parameters $\boldsymbol{\theta}$ such that the parameters respect the constraints and the (log) likelihood of data is maximized. For a fixed set of constraints, the learning problem can be set up as maximizing a concave objective (i.e., likelihood) over a convex set of constraints.

In general, solving the aforementioned optimization problem is hard because it requires searching over all possible constraint sets, which is clearly impractical. In particular, the number of possible constraint sets of size k for m parameters equals the number of partitions of size k for a set of size m (denoted by $\left\{ {m \atop k} \right\}$). This number is given by the so-called *S*tirling numbers of the second kind computed as

$$\left\{ {m \atop k} \right\} = \frac{1}{k!} \sum_{j=0}^{k} (-1)^j (k-j)^m,$$

which grows exponentially with m. The total number of partitions of a set is given by the *Bell number*, $B_m = \sum_{k=1}^{m} \left\{ {m \atop k} \right\}$.[1] Figure 3.1 shows the number of ways to partition 4 parameters. The partitions are: 1 partition, 7 ways to create 2 partition, 6 ways to create 3 partition, and 4 partition.

Our approach. To remedy this computational difficulty, we propose the following greedy approach. First, we learn the parameters of the log-linear model using MLE. Then, we quantize these parameters to k levels using the one-dimensional k-means clustering algorithm.

3.3 Theoretical Analysis of Quantization

Although our approach is simple and straightforward, we show next that it will yield models that have high log-likelihood score yet fewer parameters under the assumption that the quantization error is small. Let $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$ denote the parameters of a log-linear model learned from a dataset $\mathcal{D}$ having D examples. Without loss of generality, we assume all parameter values are positive. Let $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_k)$, where $k \leq m$, be a quantization of $\boldsymbol{\theta}$ with respect to the quantizer $\mathcal{Q}$ between $\boldsymbol{\theta}$ and $\boldsymbol{\mu}$ such that

$$|\theta_i - \mathcal{Q}(\theta_i)| \leq \epsilon \quad , \forall \theta_i \in \boldsymbol{\theta}$$

[1] The fact that the optimization problem is computationally difficult also follows from the observation that it includes structure learning as a special case where the number of true features is $k-1$.

hold, where $\epsilon \geq 0$ is a small constant. Let $\ell(\boldsymbol{\theta} : \mathcal{D})$ and $\ell(\boldsymbol{\mu} : \mathcal{D})$ denote the log-likelihood scores of $\mathcal{D}$ with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\mu}$. Then, we can prove that the difference between the average log-likelihood scores of the quantized model and the original model is bounded by $2m\epsilon$.

Theorem 3.2. *(Average Likelihood Error for Quantization). The average error of likelihood due to quantization is bounded by*

$$\frac{1}{D}\left(\ell(\boldsymbol{\theta} : \mathcal{D}) - \ell(\boldsymbol{\mu} : \mathcal{D})\right) \leq 2m\epsilon.$$

Proof.

$$\begin{aligned}
\ell(\boldsymbol{\theta} : \mathcal{D}) &= \sum_{d=1}^{D} \log\left\{\frac{1}{Z}\exp\left\{\sum_{i=1}^{m}\theta_i f_i(\widetilde{\mathbf{x}}^{(d)})\right\}\right\} \\
&= \sum_{d=1}^{D}\left(\sum_{i=1}^{m}\theta_i f_i(\widetilde{\mathbf{x}}^{(d)}) - \log\left\{\sum_{x'}\exp\left\{\sum_{i=1}^{m}\theta_i f_i(x')\right\}\right\}\right) \\
&\leq \sum_{d=1}^{D}\left(\sum_{i=1}^{m}(\mathcal{Q}(\theta_i)+\epsilon) f_i(\widetilde{\mathbf{x}}^{(d)}) - \log\left\{\sum_{x'}\exp\left(\sum_{i=1}^{m}(\mathcal{Q}(\theta_i)-\epsilon) f_i(x')\right\}\right\}\right) \\
&= \sum_{d=1}^{D}\left(\sum_{i=1}^{m}\mathcal{Q}(\theta_i) f_i(\widetilde{\mathbf{x}}^{(d)}) + \epsilon\sum_{i=1}^{m} f_i(\widetilde{\mathbf{x}}^{(d)}) - \log\left\{\sum_{x'}\exp\left\{\sum_{i=1}^{m}\mathcal{Q}(\theta_i) f_i(x')\right\}\cdot\right.\right. \\
&\quad \left.\left.\exp\left\{-\epsilon\sum_{i=1}^{m} f_i(x')\right\}\right\}\right) \\
&\leq \sum_{d=1}^{D}\left(\sum_{i=1}^{m}\mathcal{Q}(\theta_i) f_i(\widetilde{\mathbf{x}}^{(d)}) + m\epsilon - \log\left\{\sum_{x'}\exp\left\{\sum_{i=1}^{m}\mathcal{Q}(\theta_i) f_i(x')\right\}\cdot\exp\{-m\epsilon\}\right\}\right) \\
&= \sum_{d=1}^{D}\left(\sum_{i=1}^{m}\mathcal{Q}(\theta_i) f_i(\widetilde{\mathbf{x}}^{(d)}) - \log\left\{\sum_{x'}\exp\left\{\sum_{i=1}^{m}\mathcal{Q}(\theta_i) f_i(x')\right\}\right\} + 2m\epsilon\right) \\
&= \sum_{d=1}^{D}\left(\sum_{i=1}^{m}\mathcal{Q}(\theta_i) f_i(\widetilde{\mathbf{x}}^{(d)}) - \log\left\{\sum_{x'}\exp\left\{\sum_{i=1}^{m}\mathcal{Q}(\theta_i) f_i(x')\right\}\right\}\right) + 2mD\epsilon \\
&= \ell(\boldsymbol{\mu} : \mathcal{D}) + 2mD\epsilon,
\end{aligned}$$

where the second inequality is due to the fact that all the features (f_i) are binary features and the sum over all of them would be bounded by m. Similarly, we can show that $\ell(\boldsymbol{\theta} : \mathcal{D}) \geq \ell(\boldsymbol{\mu} : \mathcal{D}) - 2mD\epsilon$. □

As the number of quantization levels, k, increases, the quantization error, ϵ, decreases, and vanishes when $k = m$. As a result, the bound specified in Theorem 3.2 becomes tighter, and our greedy learning approach yields more accurate results while using smaller number of model parameters. Using the quantization error bound, we derive the sample complexity bounds as follows.

Theorem 3.3. *(MLE Sample Complexity for Quantization). Let any $\epsilon, \delta > 0$, $\lambda \in (0, 1)$ be given. Let $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_m\}$ denote the parameters learned from a dataset $\mathcal{D}$ having n examples such that $\lambda \geq \theta_i \leq 1 - \lambda$ holds. Let $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_k\}$, $k \leq m$ be a quantization of $\boldsymbol{\theta}$ with respect to quantizer $\mathcal{Q}$ between $\boldsymbol{\theta}$ and $\boldsymbol{\mu}$. Then, to recover the given optimal cluster assignments $\boldsymbol{c}$ and centers $\boldsymbol{\mu}$ with quantization error less than $|\theta_i - \mathcal{Q}(\theta_i)|$ with probability at least $(1 - \delta)$, it suffices to have training set size*

$$n \geq \frac{\left(1 + \frac{\epsilon}{2m}\right)^2}{2\lambda^2 \left(\frac{\epsilon}{2m}\right)^2} \log \frac{2}{\delta}.$$

Proof. . From Theorem 3.2, $1/n(\ell(\boldsymbol{\theta} : \mathcal{D}) - \ell(\boldsymbol{\mu} : \mathcal{D})) \leq 2m\epsilon'$ where $\epsilon' \geq 0$ is a small constant. For $2m\epsilon' \leq \epsilon$ to hold, it suffices that $\epsilon' \leq \epsilon/2m$. Applying Lemma 17 from (Abbeel et al., 2006) completes the proof. □

3.4 Relearning

The log-likelihood score of the quantized log-linear model can be further improved by relearning the model, treating all parameters that are quantized to the same value as tied. Since the

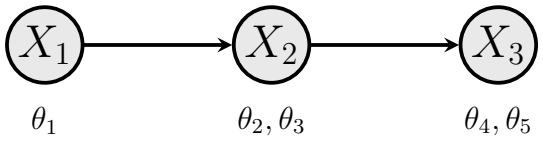


Figure 3.2. Bayesian network for Example 3.5.

quantizer $\mathcal{Q}$ induces a k-partition on the original parameter set $\boldsymbol{\theta}$, we set up the following learning problem.

$$\begin{aligned} &\max_{\boldsymbol{\theta}} \ \ell(\boldsymbol{\theta} : \mathcal{D}) \\ &\text{s.t. } \{\theta_i = \theta_j : \theta_i, \theta_j \in \boldsymbol{\theta}, \mathcal{Q}(\theta_i) = \mathcal{Q}(\theta_j)\}. \end{aligned} \tag{3.1}$$

The optimization problem given above is concave and we can use the same approaches (e.g., gradient ascent) that are used to solve the MLE problem. As mentioned earlier, the key benefit of the formulation in equation (3.1) is that it has fewer (unique) parameters and therefore the data statistics are estimated using a larger sample size than the ones used in the unconstrained (MLE) version of this problem. From standard sampling theory (Liu, 2004), this reduces the variance of the estimates.

Proposition 3.4. *Let $\boldsymbol{\mu}$ denote the set of quantized parameters, $\mathcal{Q}$ be the corresponding quantizer, and $\boldsymbol{\rho}$ be the relearned parameters (optimal solution of the optimization problem given by* (3.1)*), then we have*

$$\ell(\boldsymbol{\rho} : \mathcal{D}) \geq \ell(\boldsymbol{\mu} : \mathcal{D}).$$

Proof. Since $\boldsymbol{\mu}$ is a feasible solution of the problem given in (3.1), namely it satisfies all the constraints), and $\boldsymbol{\rho}$ is the optimal solution, it follows that $\ell(\boldsymbol{\rho} : \mathcal{D}) \geq \ell(\boldsymbol{\mu} : \mathcal{D})$. □

We illustrate the ideas of quantization and relearning with the following example.

Example 3.5. *Given the binary Bayesian network shown in Figure 3.2 with five parameters $(\theta_1, \ldots, \theta_5)$ corresponding to the three random variables (X_1, X_2, X_3). For brevity, we do*

Table 3.1. MLE computations for learning parameter θ_1 and θ_4 under no tying and tying (quantize and relearn).

Parameter	No tie	Tie (quantize:avg)	Relearn
θ_1	$\frac{\alpha_1}{\alpha_1+\alpha_2}$	$\frac{1}{2}\left(\frac{\alpha_1}{\alpha_1+\alpha_2}+\frac{\beta_1}{\beta_1+\beta_2}\right)$	$\frac{\alpha_1+\beta_1}{\alpha_1+\alpha_2+\beta_1+\beta_2}$
θ_4	$\frac{\beta_1}{\beta_1+\beta_2}$		

not include the $1-\theta_i$ *parameters. Given a dataset* $\mathcal{D}$ *over the three variables, the relevant sufficient statistics are as follows: let* $\alpha_1 = \#\mathbb{1}(X_1 = 1 : \mathcal{D})$ *be the number of times* X_1 *appears as* 1 *in the data set. Similarly, let* $\alpha_2 = \#\mathbb{1}(X_1 = 0 : \mathcal{D})$. *Let* $\beta_1 = \#\mathbb{1}(X_3 = 1|X_2 = 1 : \mathcal{D})$ *be the number of times* X_3 *and* X_2 *both appear as* 1 *in the data set. Similarly, let* $\beta_2 = \#\mathbb{1}(X_3 = 0|X_2 = 1 : \mathcal{D})$. *Assume* $\alpha_1, \alpha_2, \beta_1,$ *and* $\beta_2 > 0$. *Table 3.1 shows the MLE computations required for learning parameters* θ_1 *and* θ_4 *under no tying and tying scenarios.*

3.5 Experiments

We evaluated the performance of our quantized approach on learning tasks using several publicly available benchmark datasets from the UAI 2008 probabilistic inference competition repository (http://graphmod.ics.uci.edu/uai08). All experiments were performed on quad-core Intel i7 based machines with 16GB of RAM running Ubuntu.

First, we compared our quantized tied weight learning algorithms to the MLE with a Laplacian prior on a collection of Bayesian network learning problems. For each selected Bayesian network, we used forward sampling to generate 100 sets of 6,000 training, 2,000 validation and 2,000 test data points. Using the training data, we learned three models corresponding to the MLE, the quantized MLE, and a MLE obtained by relearning after

Table 3.2. Network information and error analysis for MLE, (Q) Quantized, and (RL) Relearned.

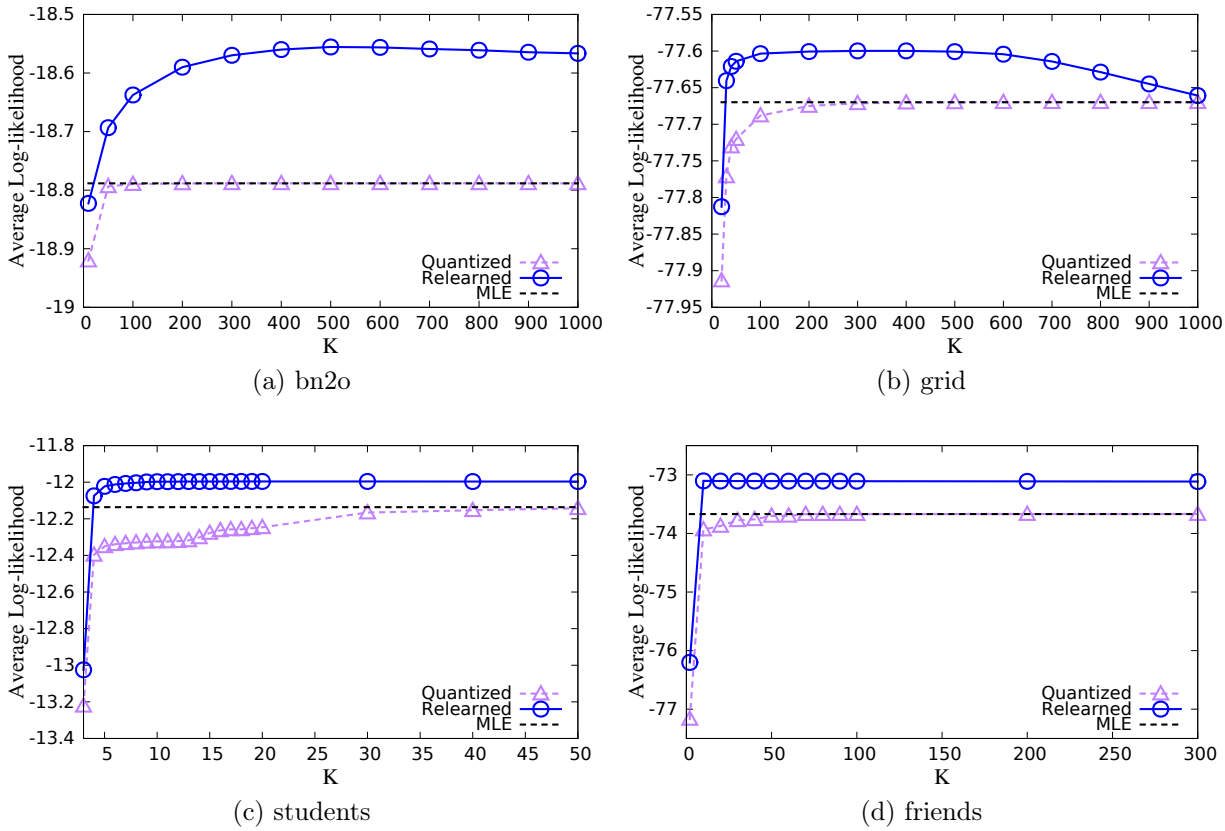
Network	Total # Param.	True k	Est. k	Max Error		
				MLE	Q	RL
bn2o	20,510	20,491	500	0.339	0.339	0.299
students	1,308	13	20	0.141	0.141	0.140
grid	1,089	596	400	0.0186	0.0186	0.0181
friends	3,899	6	10	0.008	0.008	0.007

quantization for different values of k. Performance of each learning technique was evaluated using the average log-likelihood over the test set.

We consider three kinds of Bayesian networks: two-layered noisy-or Bayesian networks (Savicky and Vomlel, 2009), relational Bayesian networks constructed from the Primula tool (UCLA), and grid networks (Sang et al., 2005). The various networks and the respective number of parameters in the networks are shown in Table 3.2.

We experimented on various noisy-or networks. The results for these networks were consistently similar. Figure 3.3 (a) shows the result for one of the networks. For these models, the MLE has a higher average log-likelihood than the quantized MLE at low values for k, but we obtained a significant performance improvement over the MLE by relearning after quantization. As k increases, relearning greatly outperforms the MLE and reaches a steady level. The performance difference tends to zero as k converges to the actual number of parameter in the network. Even for small k, the relatively small difference in log-likelihood appears to be a reasonable trade-off for the drastic reduction in the number of model parameters.

We also selected two relational networks and one grid network. The results for these models appear in Figure 3.3 (b)–(d). The performance for these networks are similar to the noisy-or networks. For small k, the performance improvement by relearning is realized fairly immediately, while the quantization also converges to the MLE quickly.



(a) bn2o (b) grid

(c) students (d) friends

Figure 3.3. Average log-likelihood on test data plotted for each parameter learned graphical model (MLE, Quantized and Relearned) varying the value for k level of quantization.

We conducted error analysis on parameter estimation between the three different learned models and the true model by using the validation set to obtain an optimally estimated k. With the exception of bn2o, our estimated k is reasonably close to the true k in the various models as shown in third and fourth column of Table 3.2. The bn2o networks contains many very similar parameter values and thus resulted in a much lower estimated k. For each model set, we calculated the average absolute point-wise error between the true and learned parameters and selected the maximum among the experimental set. The results are also shown in Table 3.2. The relearning has a consistent lower error rate compared to MLE and quantization, while the quantization has similar error rate as MLE with fewer parameters.

This reduction in the number of parameters often translates to better inference performance at prediction time as we show in the next subsection.

3.6 Discussion

We proposed a greedy method to learn tied parameter models that quantizes the parameters learned via maximum likelihood estimation using k-means clustering. Despite its simplicity, we demonstrated empirically that our approach can be used both as a regularizer and as a technique to reduce model complexity, which comports with the theoretical bounds on the error resulting from quantization that we provided.

CHAPTER 4

LEARNING PARAMETER TIED MARKOV NETWORKS

4.1 Introduction

In this chapter, we describe a parameter tying algorithm as a regularization technique for parameter learning in Markov networks (MN). In Chapter 3 we proposed an automatic approach for tying (parameters) in Bayesian networks (BNs). The parameter learning algorithm has three basic steps: (1) learn the parameters of the given BN using the maximum likelihood estimation (MLE) objective (this can be done in closed-form in BNs); (2) given a positive integer k, use the 1-dimensional k-means algorithm to group the conditional probabilities into k clusters; and (3) relearn the probabilities by forcing all parameters in each cluster to take the same value (again this can be done in closed form). Through experimental evaluations on a few benchmark datasets, we showed that the hard parameter tying approach often yields models that not only have higher test set log-likelihood scores but also admit faster and more accurate inference compared to models trained using the MLE objective.

It is not clear on how to apply the method presented in Chapter 3 to solve the harder parameter learning task for MNs. Unlike BNs, the choices of (clique) potential functions for MNs are not unique (i.e., MNs are not identifiable). Specifically, there is an infinite continuum of parameter value settings that all represent the same probability distribution. For instance, multiplying all parameters in a potential function by a real constant $c > 0$ does not change the underlying distribution. Another example is a MN with two pairwise potentials that share one common variable. The information on the common variable can be split (shifted) in arbitrary ways that result in the same distribution. A consequence of this invariance is that applying the aforementioned k-means clustering technique to achieve parameter tying may not produce useful results.

We address this limitation by proposing a *soft*, instead of hard, parameter tying scheme dubbed APT. Given that a MN has high degree of freedom in terms of parameter settings,

soft tying allows for greater flexibility for parameters to shift among cluster assignments. This type of soft tying can be viewed as a generalization of L_2 regularization that allows for k Gaussian priors (instead of one) and k different cluster center means (instead of only zero means). To automate parameter tying, we set up the learning problem as jointly selecting the parameters, group membership, and means such that either the MLE or maximum pseudolikelihood (MPLE) plus the penalty term, informally described above, is maximized. We then propose a block coordinate ascent algorithm for this optimization problem and show that it converges to a local maximum.

The second contribution is a detailed theoretical analysis of our proposed algorithm. In particular, building on the work of (Bradley and Guestrin, 2012) and (Ravikumar et al., 2010), we prove sample complexity bounds for our algorithm within the probably approximately correct (PAC) (Valiant, 1984) framework. We show that our algorithm can learn the optimal cluster memberships of the parameters with high probability when the clusters are well separated (i.e., sufficiently far from each other). Moreover, when the data is generated from a model having tied parameters, we show that the sample complexity of our algorithm can be significantly smaller than the MLE/MPLE learning task with L_2 regularization. These results provide the first rigorous theoretical justification of quantization that we are aware of.

We end the chapter with a detailed empirical evaluation of our proposed algorithm. We compare with L_2 regularization on binary pairwise Markov network structures generated using the L_1 regularized logistic regression algorithm of (Ravikumar et al., 2010). Our results show APT outperforms L_2 regularization according to pseudo-loglikelihood (PLL) score, especially on dense networks. We also evaluated the impact of changing the number of clusters on the PLL score and found that small to medium values for k often achieve the best score. These results demonstrate that APT is a promising, practical approach for controlling model complexity and improving generalization performance.

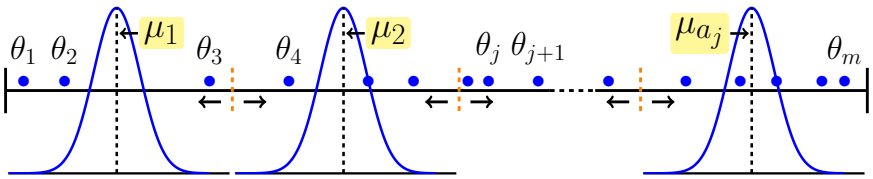


Figure 4.1. Quantization intervals denoting tied parameters. Each k-partition (interval) contains a set of quantized parameters θ_i (dots) and is associated with a local Gaussian distribution parameterized by $(\mu = \mu_{a_j}, \sigma)$. Short dashed lines on the intervals denote the quantization boundaries which shift according to an optimum (local) penalized parameter setting.

4.2 Problem Definition

We are interested in learning a parameter tied graphical model (Chapter 3, Definition 3.1) for Markov networks. For a fixed set of constraints, the learning problem can be formulated as maximizing a concave objective (the log-likelihood) over a convex set (set of equality constraints). This can be accomplished via projected gradient ascent. Since the total numbers of partitions is given by the *Bell number*, searching over all possible constraint sets is not feasible. However, the problem can be simplified by relaxing the equality constraints. Our approach is to approximate the equality constraints utilizing a penalty function which enforces a *soft tying* of parameters. We reformulate the learning of a parameter tied graphical model by adding a penalty for poor clusterings to the log-likelihood objective.

$$\arg\max_{\boldsymbol{\theta},\boldsymbol{a},\boldsymbol{\mu}} \quad \ell(\boldsymbol{\theta}) - \frac{\lambda}{2}\sum_{j=1}^{m}(\theta_j - \mu_{a_j})^2. \tag{4.1}$$

The objective in equation (4.1) represents a regularized log-likelihood similar to L_2 regularization. Here, the penalty function is the k-means objective with an additional tuning hyperparameter λ that controls the magnitude of the penalty. This corresponds to a collection of k Gaussian priors such that the i-th prior has mean μ_i and variance proportional to $1/\lambda$ (see Figure 4.1). However, while equation (4.1) is concave in $\boldsymbol{\theta}$ for a fixed clustering, the objective function is no longer a concave optimization problem when $\boldsymbol{a}$ is not given (note that,

in general, the k-means objective is not convex). It is easy to show that L_2 regularization is a special case of the objective in equation (4.1); all we have to do is assume that there is only one cluster and $\mu_1 = 0$.

4.3 Block Coordinate Ascent Learning Algorithm

In this section, we derive a block coordinate ascent technique for equation (4.1), thus achieving our aim of automating parameter tying for MNs. We will refer to this general algorithm simply as APT going forward. Giving fully observed training dataset $\mathcal{D}$, the MN structure $\mathcal{M}_{MN} \triangleq \langle \boldsymbol{X}, \boldsymbol{\theta}, \mathrm{G}_{MN} \rangle$, k clusters, and penalty term λ, the APT algorithm performs coordinate ascent on the objective in equation by alternating between finding the optimal parameters for a fixed clustering and finding the optimal clustering for a fixed vector of parameters. Both of these optimizations are straightforward: a regularized maximum likelihood optimization problem and a 1-dimensional k-means clustering problem respectively. The former can be solved using standard gradient-based methods while the latter can be solved in polynomial time using dynamic programming. Next, we make remarks about Algorithm 4.1, that illustrate the flexibility and utility of our proposed method.

First, Algorithm 4.1 returns a soft clustering of the parameters. However, we can turn the soft clustering into hard clustering and relearn the parameters using the MLE objective while enforcing equality constraints $\mathcal{C}$ on all parameters assigned to the same cluster. This can be done via projected gradient ascent. That is, after each gradient step, the parameter vector may step outside of the set of constraints. If this happens, we simply project the parameters back into the constraint set. As the cluster constraints insist that all parameters in cluster i must have the same value, the projection operation simply replaces all parameters in cluster i with the average of all parameters in cluster i.

Second, since the objective function in equation (4.1) is bounded from above, the coordinate ascent procedure is guaranteed to converge to a local maxima. The algorithm increases the

Algorithm 4.1: Automatic Parameter Tying (APT)

Input: A Markov network structure $\mathcal{M}_{MN} \triangleq \langle \boldsymbol{X}, \boldsymbol{\theta}, \mathrm{G}_{MN} \rangle$, Integer $k > 0$, Integer $T > 0$, penalty λ

Output: Feature vector $\boldsymbol{\theta}$, clustering $\langle \boldsymbol{a}, \boldsymbol{\mu} \rangle$

begin

Initialize $\boldsymbol{\theta}^{(0)}$ and $\langle \boldsymbol{a}, \boldsymbol{\mu} \rangle$ to random values.

for $t \leftarrow 1$ *to* T *or until convergence* **do**

1. Update $\boldsymbol{\theta}$ given $\langle \boldsymbol{a}, \boldsymbol{\mu} \rangle$ using gradient ascent.

$$\boldsymbol{\theta}^{(t)} \leftarrow \arg\max_{\boldsymbol{\theta}} \ \ell(\boldsymbol{\theta}) - \frac{\lambda}{2} \sum_{j=1}^{m} (\theta_j - \mu_{a_j^{(t-1)}}^{(t-1)})^2$$

2. Update $\langle \boldsymbol{a}, \boldsymbol{\mu} \rangle$ using 1D k-means given $\boldsymbol{\theta}$.

$$\langle \boldsymbol{a}^{(t)}, \boldsymbol{\mu}^{(t)} \rangle \leftarrow \arg\min_{\boldsymbol{a}, \boldsymbol{\mu}} \sum_{j=1}^{m} (\theta_j^{(t)} - \mu_{a_j})^2$$

where $\mu_i \frac{1}{\sum_{j=1}^{m} I(a_j, i)} \sum_{j=1}^{m} \theta_j \cdot \mathbb{1}(a_j, i)$ and $\mathbb{1}(a_j, i)$ is an indicator function which equals 1 if $a_j = i$ and 0 otherwise.

return $\langle \boldsymbol{\theta}^{(T)}, \boldsymbol{a}^{(T)}, \boldsymbol{\mu}^{(T)} \rangle$

end

objective function each iteration since each of the two sub-optimization problems, finding the optimal parameters for a fixed clustering (increasing log-likelihood) and finding the optimal clustering for a fixed vector of parameters (reducing penalty), contributes to increasing the objective function. In practice, the rate of convergence can be improved by initializing the parameters and cluster means to small values or after running a few iterations of gradient ascent that optimizes MLE plus an L_2 regularization term.

Third, note that Algorithm 4.1 optimizes MLE plus a penalty term which requires inference over the MN. The latter is often infeasible in practice. Its practical performance and convergence can be improved by using two strategies: (a) stochastic or mini-batch gradient ascent and (b) using the MPLE objective instead of MLE. Stochastic and mini batch gradient

ascent can be used in two ways. First, we can use it to optimize $\boldsymbol{\theta}$ in Step 1 of the for loop of Algorithm 4.1. Second, we can use it in the outer loop by not running Step 1 until convergence, namely we run gradient ascent only for a few iterations in Step 1. In our experiments, we employ both strategies to speed up our algorithm.

4.4 Theoretical Analysis

In this section, we analyze the sample complexity of the proposed method and provide conditions under which it provably recovers the correct clustering assignments $\boldsymbol{a}$ and approximate cluster means $\boldsymbol{\mu}$ with small L_1 error. We demonstrate polynomial sample complexity when the clusters are well-separated (defined formally below). We consider two cases: (1) *hard tying*: when the true MN from which the data is generated has exactly $k < m$ unique parameters; and (2) *soft tying*: when the MN has m parameters and k is the number of clusters.

We begin by establishing conditions for well-separation of clusters in Lemmas 4.3 and Corollary 4.4.

Definition 4.1. (Maximum Cluster Width). *Let ω denote the maximum width of a cluster i such that the width of a cluster is the Euclidean distance between two farthest parameter points, θ_r, θ_s, in the cluster.*

$$\omega = \max_i \max_{(r,s)|a_r=i,a_s=i} |\theta_r - \theta_s|.$$

Definition 4.2. (Minimum Inter-cluster Distance). *Let α denote the Euclidean distance between two closest parameter points, θ_r, θ_s, in different clusters i, j.*

$$\alpha = \min_{(r,s)|a_r=i,a_s=i,i\neq j} |\theta_r - \theta_s|.$$

We will refer to α as the minimum inter-cluster distance.

Lemma 4.3. *Let $\hat{\theta}_i$ and $\hat{\mu}_j$ denote the estimates of the true parameters θ_i and μ_j respectively based on N samples drawn independently and identically from the true MN. Let $\max_i |\theta_i - \hat{\theta}_i| \leq \epsilon$ and ω, α be true maximum cluster and minimum inter-cluster distances. If $\epsilon < \frac{\alpha-\omega}{4}$, then the 1-dimensional k-means ($k > 0$) clustering algorithm is guaranteed to return optimal cluster assignments.*

Proof. Without loss of generality, since $\max_i |\theta_i - \hat{\theta}_i| \leq \epsilon$, the estimated maximum cluster width is bounded above by, $\omega_{UB} \leq \omega + 2\epsilon$, and the estimated minimum inter-cluster width is bounded below by $\alpha_{LB} \geq \alpha - 2\epsilon$. To ensure the parameter estimates $\hat{\theta}_i$ and $\hat{\theta}_j$, with cluster assignments, $\hat{a}_i$ and $\hat{a}_j$, of any two parameters, θ_i and θ_j such that $\hat{a}_i = a_i$ and $\hat{a}_j = a_j$, requires the condition $\alpha_{LB} > \omega_{UB}$. Thus, by satisfying $\alpha - 2\epsilon > \omega + 2\epsilon$, rearranged as $\epsilon < \frac{\alpha-\omega}{4}$, the 1-dimensional k-means clustering algorithm is guaranteed to return optimal cluster assignments. □

Corollary 4.4. *For the hard tying case (namely there are $k > 0$ unique parameters) the 1-dimensional k-means clustering algorithm is guaranteed to return optimal cluster assignments if $\epsilon < \frac{\alpha}{4}$.*

Proof. When there are $k > 0$ unique parameters that also correspond to the cluster centers, $\omega = 0$ and the proof follows from Lemma 4.3. □

We use Lemma 4.3 and Corollary 4.4 to derive the following definition for well-separation.

Definition 4.5. *(Well-separation) We say that the triple $\langle \boldsymbol{\theta}, \boldsymbol{a}, \boldsymbol{\mu} \rangle$ denoting the parameters as well as cluster assignments and centers is well-separated for a given error bound ϵ iff $\epsilon < \frac{\alpha-\omega}{4}$ for the soft tying case and $\epsilon < \frac{\alpha}{4}$ for the hard tying case.*

Next, we use Lemma 4.3 and Corollary 4.4 in conjunction with the PAC and sample complexity bounds for MLE derived in (Bradley and Guestrin, 2012) and (Ravikumar et al., 2010) to yield our desired sample complexity bounds. Formally,

Theorem 4.6 (**MLE Sample Complexity for Soft and Hard Tying**). *Let $C_{min} > 0$ be a lower bound on the minimum eigenvalue of the Hessian of the negative log likelihood. Let the regularization hyperparameter λ be chosen such that $\lambda = C_{min}^2 n^{-\xi/2}/(2^6 m^2)$, where n is the number of training samples, m is the number of feature weights and $\xi \in (0, 1)$. Then, to recover the optimal cluster assignments $\boldsymbol{a}$ and centers $\boldsymbol{\mu}$ with L_1 error smaller than $(\alpha - \omega)/4$ with probability at least $(1 - \delta)$, it suffices to have training set size*

$$n \geq \frac{2^9}{C_{min}^2} \frac{16m^2}{(\alpha - \omega)^2} \log \frac{2m(m+1)}{\delta}.$$

For hard tying, in which we have k unique parameters, the sample complexity for finding optimal cluster assignments $\boldsymbol{a}$ and centers $\boldsymbol{\mu}$ with L_1 error smaller than $\alpha/4$ is given by

$$n \geq \frac{2^9}{C_{min}^2} \frac{16k^2}{\alpha^2} \log \frac{2m(m+1)}{\delta}.$$

Proof. Our proof follows that of (Bradley and Guestrin, 2012). Given n training samples, m number of feature weights, k number of cluster centers, and constant $\delta > 0$. Let ϕ be indicator function features with range $[0, 1]$ s.t. the maximum magnitude of any feature $\phi_{max} = 1$. Let $C_{min} > 0$ be a lower bound on the minimum eigenvalue of the Hessian of the negative log-likelihood. Using Lemma 9.3 from (Bradley and Guestrin, 2012), we have the probability of failure on the lower bound of the parameter estimation error as

$$2m \exp\left(-\frac{\delta^2 n}{2}\right) - 2m^2 \exp\left(\frac{nC_{min}^2}{2^5 m^2}\right).$$

Choose

$$\lambda = \delta = \frac{C_{min}^2}{2^6 m^2} n^{-\xi/2},$$

where $\xi \in (0, 1)$. Substituting δ, we have the probability of failure as

$$2m \exp\left(-\frac{C_{min}^2}{2^6 m^2} n^{-\xi/2} \frac{n}{2}\right) - 2m^2 \exp\left(\frac{nC_{min}^2}{2^5 m^2}\right) \;=\; 2m(m+1) \exp\left(-\frac{C_{min}^4}{2^{13} m^4} n^{1-\xi}\right).$$

To have probability of failure of at most δ for n samples, choose ξ as

$$\begin{aligned} 2m(m+1)\exp\left(-\frac{C_{min}^4}{2^{13}m^4}n^{1-\xi}\right) &\leq \delta \\ \xi &\leq 1-\frac{1}{\log n}\left(\log\frac{2^{13}m^4}{C_{min}^4}+\log\log\frac{2m(m+1)}{\delta}\right). \end{aligned}$$

From Lemma 4.3, for soft tying, to have the 1-dimensional k-means clustering algorithm guarantee to return optimal cluster assignments, it suffices to bound the L_1 parameter estimation error at most $\epsilon < (\alpha-\omega)/4$. Therefore, we have

$$\begin{aligned} \frac{C_{min}^2}{2^6m^2}n^{-\xi/2} &\leq \epsilon < \frac{\alpha-\omega}{4} \\ \log\frac{C_{min}^2}{2^6m^2}-\frac{\xi}{2}\log n &\leq \log\frac{\alpha-\omega}{4} \\ \frac{\xi}{2}\log n &\geq \log\frac{C_{min}^2}{2^6m^2}-\log\frac{\alpha-\omega}{4} \\ \frac{1}{2}\left(1-\frac{1}{\log n}\left(\log\frac{2^{13}m^4}{C_{min}^4}+\log\log\frac{2m(m+1)}{\delta}\right)\right)\log n &\geq \log\frac{C_{min}^2}{2^6m^2}-\log\frac{\alpha-\omega}{4} \end{aligned}$$

$$n \geq \frac{2^9}{C_{min}^2}\frac{16m^2}{(\alpha-\omega)^2}\log\frac{2m(m+1)}{\delta}.$$

For the hard-tying case, the parameters are set to cluster centers. This sets $\omega = 0$ and $\lambda = \delta = (C_{min}^2/2^6k^2)n^{-\xi/2}$. Then, from Corollary 4.4, to have the 1-dimensional k-means clustering algorithm guarantee to return optimal cluster assignments, it suffices to bound the L_1 parameter estimation error at most $\epsilon < \alpha/4$. Therefore, we have

$$n \geq \frac{2^9}{C_{min}^2}\frac{16k^2}{\alpha^2}\log\frac{2m(m+1)}{\delta}.$$

□

The sample complexity bound implies that when the minimum eigenvalue is large and/or when the difference between the minimum inter-cluster distance and the maximum cluster width is large (namely, the clusters are well separated), our algorithm is statistically efficient. As expected, the hard tying case is statistically more efficient than the soft tying case since the former does not depend on the cluster width.

Table 4.1. Dataset characteristics.

Dataset	#vars	#train	#valid	#test
nltcs	16	16181	2157	3236
msnbc	17	291326	38843	58265
kdd	64	180092	19907	34955
plants	69	17412	2321	3482
audio	100	15000	2000	3000
jester	100	9000	1000	4116
netflix	100	15000	2000	3000
accidents	111	12758	1700	2551
retail	135	22041	2938	4408
pumsb∗	163	12262	1635	2452
dna	180	1600	400	1186
kosarek	190	33375	4450	6675
msweb	294	29441	3270	5000
book	500	8700	1159	1739
tmovie	500	4524	1002	591
webkb	839	2803	558	838
reuters	889	6532	1028	1540
20ng	910	11293	3764	3764
bbc	1058	1670	225	330
ad	1556	2461	327	491

4.5 Experiments

4.5.1 Experimental Setup

We evaluated APT on 20 real-world datasets which have been widely used in recent years to evaluate learning algorithms for probabilistic graphical models (Rahman and Gogate, 2016; Rooshenas and Lowd, 2014; Davis and Domingos, 2010) (see Table 4.1 for details on the binary datasets). We implemented APT in C++ and all experiments were conducted on Intel i7 Ubuntu machines with 16GB of RAM.

Table 4.2. Test set negative PLL scores (k and λ selected using the validation set) on 20 benchmark datasets for L_2 regularization, LTR and APT algorithm under various values of d.

	$d = 5$			$d = 15$			$d = 50$		
Dataset	L_2	LTR	APT	L_2	LTR	APT	L_2	LTR	APT
nltcs	5.05	**5.02**	5.02	5.10	4.99	**4.98**	–	–	–
msnbc	6.17	6.12	**6.11**	6.22	6.10	**6.08**	–	–	–
kdd	2.12	2.11	**2.11**	2.09	**2.08**	2.09	2.14	2.08	**2.07**
plants	10.68	10.63	**10.59**	10.46	10.23	**10.21**	11.11	10.26	**10.24**
audio	38.88	38.46	**38.44**	38.34	37.31	**37.22**	40.73	37.47	**37.03**
jester	51.94	51.41	**51.28**	51.21	50.00	**49.75**	54.97	50.53	**50.04**
netflix	54.91	54.41	**54.40**	54.22	52.84	**52.68**	57.52	53.32	**52.67**
accidents	14.71	14.50	**14.47**	13.21	12.78	**12.70**	13.85	12.90	**12.69**
retail	10.53	10.46	**10.45**	10.57	10.41	**10.40**	10.93	10.40	**10.39**
pumsb∗	11.58	11.47	**11.46**	10.13	9.80	**9.79**	11.17	9.94	**9.79**
dna	59.16	58.54	**58.46**	61.92	59.73	**59.54**	69.26	63.13	**62.84**
kosarek	10.41	10.35	**10.34**	10.32	10.17	**10.17**	10.59	10.27	**10.25**
msweb	16.98	16.80	**16.79**	17.04	**16.60**	16.60	14.80	13.74	**13.71**
book	35.90	36.68	**35.82**	35.49	37.70	**35.20**	36.48	42.14	**35.88**
tmovie	71.91	72.87	**71.49**	63.16	66.43	**62.94**	61.22	66.22	**58.50**
webkb	158.31	163.21	**158.08**	157.30	169.17	**155.51**	180.85	203.54	**158.71**
reuters	91.33	92.29	**91.26**	**88.55**	91.98	88.65	91.19	99.66	**88.83**
20ng	163.90	164.36	**163.30**	**160.82**	162.38	162.29	170.94	167.38	**166.71**
bbc	259.96	275.06	**259.18**	267.44	292.66	**256.60**	331.67	343.90	**260.95**
ad	6.79	6.58	**6.55**	6.37	**6.11**	6.16	6.22	**6.01**	6.06

For each dataset, we learned a pairwise binary MN structure (not the parameters) using the L_1 regularization based structure learning algorithm of (Ravikumar et al., 2010). This algorithm constructs the MN structure as follows. It learns a L_1 regularized logistic regression classifier $L(X_i)$ for predicting the value of each variable X_i given all other variables. Then, it adds an edge between two variables X_i and X_j if the features corresponding to X_i or X_j have non-zero weights in $L(X_j)$ and $L(X_i)$ respectively. Unfortunately, on many datasets, this method yields dense models. Therefore, in order to achieve sparsity we constrained the regularization hyperparameter (λ) so that the degree (size of the Markov blanket) of

each variable is bounded by d. In our experiments, we used the following values for $d = \{5, 10, 15, 30, 50\}$ where $d = 10, 30$ are shown in supplemental material due to space constraint. We learned the L_1 regularized logistic regression classifier using the Orthant-Wise Limited-memory Quasi Newton (OWL-QN) method (Andrew and Gao, 2007).

4.5.2 APT versus L_2 regularization

Table 4.2 shows the test set negative PLL scores (using various values of d) for APT and L_2 regularization on the 20 datasets. For each dataset, we select the k and λ values using the validation set. Lower negative PLL values are better and bold signifies the higher value achieved by the respective regularization method. From the results, we can clearly see APT outperforms L_2 across the majority (with the exception of reuters and 20ng) of datasets and complexity of model structure. Moreover, as the structure becomes increasingly dense (more neighboring nodes), the performance gap widens. One key takeaway here is when the underlying model has a large number of parameters, it is prudent to utilize APT for better generalization performance. Parameter learning algorithms for complex models such as CNNs or SRL models can leverage our method since the model will contain parameters that take on similar values, which is evident from the results.

4.5.3 APT versus LTR

We also compared APT with the approach presented in Chapter 3 to MNs, which we refer to as the learn-tie-relearn algorithm (LTR). The previous algorithm focuses on BNs and a straightforward extension of the previous method to parameter learning in MNs is the following: (1) learn the parameters using the MLE objective; (2) cluster the resulting parameters into k clusters; and (3) relearn the parameters by adding equality constraints over the parameters in each cluster. However, we found that this approach has high variance. This is likely due to the scale invariance property of MNs. To combat this, in step (1) of

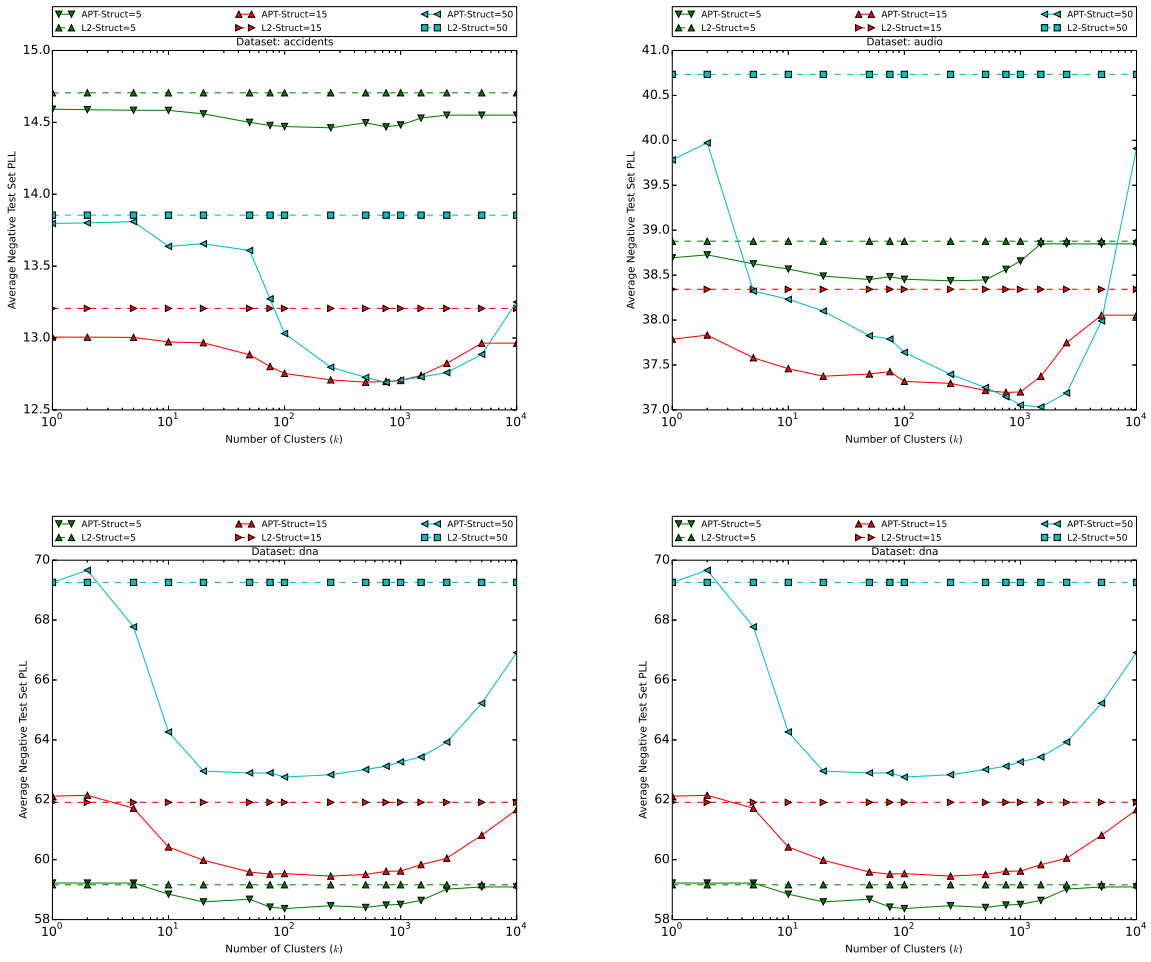


Figure 4.2. Average test set negative PLL scores for L_2 (dotted) and APT (solid) for varying number of clusters (k) and model complexity (d). To minimize clutter, we show graphs for only three values of d and do not include results for LTR (whose variance is quite high).

the algorithm, we learned the parameters using L_2 regularization, which greatly reduces the variance and thus improves the performance of LTR.

Table 4.2 also shows the test set negative PLL scores (using various values of d) for APT and LTR. For each dataset we select the k and λ values using the validation set. Lower negative PLL values are better and bold signifies the best value achieved by the respective regularization method. From the results, we clearly observe that APT outperforms LTR across the majority of datasets and complexity of model structure (measured by d). However, the

noticeable deviation from the previous results is that wider differences occur in datasets with higher complexity (more variables and neighboring nodes) as in the case with bbc ($d = 50$). Comparatively, we see that LTR mostly outperforms L_2. Thus, hard tying the parameters output by L_2 is highly beneficial.

4.5.4 Impact of varying k

Figure 4.2 shows average negative test set PLL scores for APT and L_2 regularization as a function of the number of clusters k on four randomly chosen datasets. To better organize the results and to avoid clutter, the comparison was made by fixing the maximum number of d neighboring nodes to 5, 15 and 50 for the learned MN structures and across varying k clusters. Consistent with the previous results, more complex structures ($d = 50$) create a wider performance gap between APT and L_2. Conversely, the performance gap is closer for simpler models ($d = 5$). The plots also show that by having the ability to control the parameter k (number of clusters), there is an optimal setting where the lowest test average negative PLL score can be achieved. For example, in dna, the best test average negative PLL score requires approximately 20 clusters. Overall, for each of the datasets, there is a setting of k where APT outperforms L_2. This demonstrates the utility of our approach.

We found that our algorithm converges rapidly and requires roughly the same number of iterations as L_2 to converge in practice. Moreover, its variance is also low (and therefore not plotted in the graphs and tables), namely most local maxima reached achieve similar solution quality.

We end this section by mentioning several pragmatic techniques to achieve good practical performance. First, it is often beneficial to increase the regularization hyperparameter λ with the number of iterations. As the cluster centers and assignments converge, increasing λ increases the penalty and yields rapid convergence to good solutions. Second, hard tying is beneficial only for moderate to large values of k and in fact for small values of k, it may hurt

the performance significantly. For small values of k, soft tying works much better. Finally, the rate of convergence can be improved in practice by updating the cluster centers but not the cluster assignments at each iteration of gradient ascent in Step 1 of Algorithm 1. Periodically updating the cluster assignments appears to yield much faster convergence than updating them at each iteration.

4.6 Discussion

We investigated parameter tying, an alternative regularization method for MNs. Unlike other machine learning frameworks where parameter tying is specified *a priori*, we introduced an automatic approach to tying parameters (APT). Specifically, we incorporated a more informative and general penalty term that leverages clustering into the objective function for parameter learning in MNs. We showed that our approach generalizes L_2 regularization. Since our formulation of the penalized learning problem is no longer concave, we proposed a block coordinate ascent algorithm to solve the optimization problem efficiently. We provided sample complexity bounds for our proposed algorithm which show significant improvement over standard L_2 regularization with high probability when the clusters are well-separated. Empirically, we showed that our approach outperforms L_2 regularization on a variety of real-world datasets.

CHAPTER 5

SLICE IMPORTANCE SAMPLING FOR PARAMETER TIED GRAPHICAL MODELS

5.1 Introduction

As mentioned earlier, a key benefit of parameter tying is that it introduces symmetries in the learned model, and exploiting the symmetric structure makes inference easier and more efficient. In this chapter, we develop a novel sampling-based approximate inference algorithm for fast and accurate sampling in parameter tied models. The sampling algorithm is based on slice importance sampling (Neal, 2003; Gogate and Domingos, 2010). Slice sampling belongs to the family of Markov Chain Monte Carlo (MCMC) algorithms and is related to other MCMC algorithms such as Gibbs and Metropolis-Hastings. Our experiments conclusively show that our new sampling algorithm outperforms MC-SAT (Poon and Domingos, 2006), a popular sampling algorithm that combines MCMC sampling and satisfiability solution sampling, and Gibbs sampling (baseline) in the presence of tied parameters.

5.2 Slice Importance Sampling

In slice importance sampling, the proposal distribution is defined over the features of a log-linear model rather than over the variables. For instance, we can define a proposal distribution

$$Q(\boldsymbol{F}) = Q(f_1) \prod_{i=2}^{m} Q(f_i | f_1, \ldots, f_{i-1}),$$

over the set of features $\boldsymbol{F}$. Sampling each feature in order from Q yields a 0/1 assignment to the features, where 0 indicates that the negation of the feature is true while 1 indicates that the feature is true. We can consider this 0/1 assignment as a slice over the possible assignments to the variables in the following sense. All (variable) assignments that satisfy all

features assigned to 1 and all negations of features assigned to 0 will have the same probability. Uniformly sampling over this subset of variable assignments (the slice) gives us the required sample. The benefit of slice sampling is that all variable assignments having the same probability in the distribution represented by the log-linear model have the same probability (assuming uniform sampling over the slice can be done) in the proposal distribution. This reduces the variance (Gogate and Domingos, 2010).

In our new algorithm, the main idea is to define the proposal distribution over the tied features rather than over the individual features, further reducing the variance. We first create a set of *super-features* $\boldsymbol{\mathcal{G}}$, such that each of them contains features with tied parameters.

$$\boldsymbol{G_i} = \{f_j | f_j \text{ has parameter } \theta_i\}.$$

Using the chain rule, define a proposal over super-features as

$$Q(\boldsymbol{\mathcal{G}}) = Q(\boldsymbol{G_1}) \prod_{i=2}^{k} Q(\boldsymbol{G_i} | \boldsymbol{G_1}, \ldots, \boldsymbol{G_{i-1}}).$$

However, one problem with defining a proposal over the super-features is that the number of values each super-feature $\boldsymbol{G_i}$ can take is exponential in the number of features it contains, namely exponential in $|\boldsymbol{G_i}|$. To compactly represent these exponentially many assignments, we use the so-called *counting assignments* (Milch et al., 2008; Jha et al., 2010) as follows. We partition the assignments to the features in the set $\boldsymbol{G_i}$ into $|\boldsymbol{G_i}| + 1$ subsets where the j-th subset contains all assignments in which exactly j features in $\boldsymbol{G_i}$ are assigned to true and the remaining are assigned to false. Thus, each super-feature $\boldsymbol{G_i}$ can take $|\boldsymbol{G_i}| + 1$ values, yielding exponential reduction in complexity.

Another benefit of using counting assignments is that they yield better proposal distributions. In particular, the set of valid counting assignments to all super-features partitions the set of assignments to the variables into equiprobable subsets, where each subset is composed of assignments to variables that satisfy a counting assignment to all super-features. In an

Algorithm 5.1: Tied Weight Importance Sampling

Input: A log-linear model $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{F}, \boldsymbol{\mu} \rangle$ with k unique weights, number of samples N
Output: Importance weighted samples
begin
  Create one super-feature $\boldsymbol{G_i}$ for each parameter μ_i
  Construct a proposal distribution $Q(\boldsymbol{G})$ over the super-features
  for $s \leftarrow 1$ *to* N **do**
    $S \leftarrow \emptyset$
    $w^{(s)} \leftarrow 1$
    for $i \leftarrow 1$ *to* k **do**
      Sample $j_i \sim Q(\boldsymbol{G_i} | \boldsymbol{G_1}, \ldots, \boldsymbol{G_{i-1}})$
      Add j_i randomly selected features from $\boldsymbol{G_i}$ to S
      Add the negation of the features from $\boldsymbol{G_i}$ not selected in the previous step to S

$$w^{(s)} \leftarrow w^{(s)} \times \binom{|\boldsymbol{G_i}|}{j_i} \frac{\exp(j_i \mu_i)}{Q(\boldsymbol{G_i} | \boldsymbol{G_1}, \ldots, \boldsymbol{G_{i-1}})}$$

    Sample $\widetilde{\mathbf{x}}^{(s)} \sim \mathcal{U}_{SAT}(S)$
    $w^{(s)} \leftarrow w^{(s)} \times \#S$
  return $(\widetilde{\mathbf{x}}^{(N)}, w^{(N)})$ for $s \leftarrow 1$ *to* N
end

ideal proposal distribution all such assignments must have the same probability, and defining the proposal over the counting assignments preserves this property.

Example 5.1. *Consider a log-linear PGM having three features* $f_1 = X_1 \vee X_2$, $f_2 = X_2 \vee X_3$, *and* $f_3 = X_1 \vee X_3$ *and three binary variables* X_1, X_2, *and* X_3. *Let the features* f_1 *and* f_2 *have the same weight* θ_1 *and* θ_2 *be the weight associated with* f_3. *We define two super-features:* $\boldsymbol{G_1} = \{f_1, f_2\}$ *and* $\boldsymbol{G_2} = \{f_3\}$. $\boldsymbol{G_1}$ *has four possible assignments:* $\{(f_1 = 0, f_2 = 0), (f_1 = 0, f_2 = 1), (f_1 = 1, f_2 = 0), (f_1 = 1, f_2 = 1)\}$ *but only three possible counting assignments* $\{0, 1, 2\}$, *where* 0, 1 *and* 2 *correspond to the subset of assignments* $\{(f_1 = 0, f_2 = 0)\}$, $\{(f_1 = 0, f_2 = 1), (f_1 = 1, f_2 = 0)\}$ *and* $\{(f_1 = 1, f_2 = 1)\}$ *respectively. The reader can verify*

that the assignments to the the variables that satisfy either $(f_1 = 0, f_2 = 1)$ *or* $(f_1 = 1, f_2 = 0)$ *have the same probability.*

However, counting assignments introduce the following problem. To generate a sample, we need to generate an assignment to variables uniformly at random from the subset of variable assignments that satisfy the counting assignment. Unfortunately, this problem is extremely challenging and to our knowledge no general-purpose algorithms exist for it. To alleviate this computational difficulty, we propose to use the following proposal distribution.

$$Q(f_1, \ldots, f_{|\boldsymbol{G_i}|} \,|\, \forall j \; f_j \in \boldsymbol{G_i}, \boldsymbol{G_i} = t) = \frac{1}{\binom{|\boldsymbol{G_i}|}{t}}.$$

Sampling from this proposal distribution yields a 0/1 assignment to the features and the only problem that remains to be solved is generating an assignment to variables uniformly at random from the subset of variable assignments that satisfy the given assignment to the features. This problem can be reduced to the uniform solution sampling problem, a well-researched problem for which a number of general-purpose solvers and techniques exist (Gogate and Dechter, 2011; Wei et al., 2004; Gogate, 2009).

Algorithm 5.1 formally describes our proposed slice importance sampling algorithm. The sampling process begins by constructing a proposal distribution Q over the super-features $\boldsymbol{G_i}$ associated with the same parameter μ_i. Then we sample assignments (j_i) for each super-feature $\boldsymbol{G_i}$ from the proposal Q in a selected order (step 10). Then we select j_i random features from $\boldsymbol{G_i}$ and set their assignment as 1. The rest of the features in $\boldsymbol{G_i}$ are assigned 0. This sampled 0/1 assignment to all features defines a satisfiability problem S. Solutions of this satisfiability problem correspond to the subset of variable assignments that have the same probability. Thus to generate a sample, all we have to do is uniformly sample the solutions of S (step 13). For this (procedure $\mathcal{U}_{SAT}$), we can use uniform solution samplers such as SampleSAT (Wei et al., 2004) and SampleSearch (Gogate and Dechter, 2011). In our experiments, we used the latter. The weight $w^{(s)}$ of the generated sample is proportional to

the ratio between the probability of generating the sample from $\mathcal{M}$ and the probability of generating it from the proposal distribution and is computed iteratively in steps 12 and 14.

In our experiments, we have used a simple proposal which factorizes independently over the super-features.

$$Q(\boldsymbol{\mathcal{G}}) = \prod_{i=1}^{k} Q(\boldsymbol{G_i}).$$

Let $\mu_1, \ldots, \mu_k$ be the parameters and let $\boldsymbol{G_i}$ denote the super-feature that is associated with μ_i. We define $Q(\boldsymbol{G_i})$ as the binomial distribution

$$Q(\boldsymbol{G_i} = t) \triangleq \binom{|\boldsymbol{G_i}|}{t} \frac{\exp(t\mu_i)}{(1+\exp(\mu_i))^{|\boldsymbol{G_i}|}},$$

where $t \in \{0, \ldots, |\boldsymbol{G_i}|\}$. Notice that the binomial is defined over $|\boldsymbol{G_i}| + 1$ points as opposed to the conventional proposal which would have been defined over $2^{|\boldsymbol{G_i}|}$ points.

5.3 Experiments

We evaluated the performance of our quantized approach on inference tasks using several publicly available benchmark datasets from the UAI 2008 probabilistic inference competition repository (http://graphmod.ics.uci.edu/uai08). All experiments were performed on quad-core Intel i7 based machines with 16GB of RAM running Ubuntu.

We compared our proposed approximate inference method based on slice importance sampling (denoted TW) to MC-SAT (Poon and Domingos, 2006) in the Alchemy system (Kok et al., 2006) and Gibbs sampling for inference in Bayesian networks. Each algorithm was run for 500 seconds and then evaluated by computing the average Hellinger distance (Kokolakis and Nanopoulos, 2001) between the single-variable marginals obtained by each algorithm and marginals obtained by an exact solver. The Hellinger distance between two probability distributions $P = (p_1, p_2, \ldots, p_m)$ and $Q = (q_1, q_2, \ldots, q_m)$ is defined as $h(P,Q) = (1/\sqrt{2})\sqrt{\sum_{i=1}^{m} \sqrt{p_i} - \sqrt{q_i}}$. The results were averaged over 10 runs of each algorithm

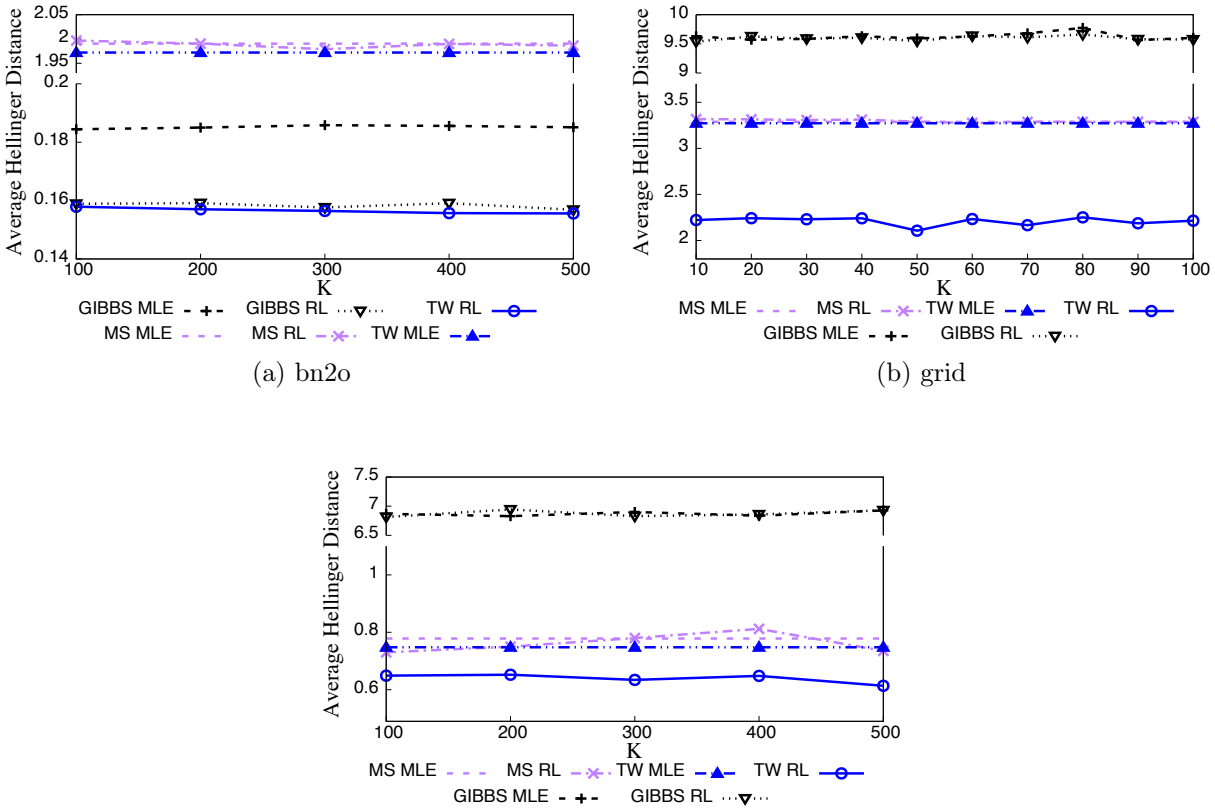


Figure 5.1. Average Hellinger distance between the exact and the approximate one-variable marginals plotted as a function of k level of quantization for MS MLE (MC-SAT MLE), MS RL (MC-SAT Relearned), TW MLE (Tied Weight MLE), TW RL (Tied Weight Relearned), GIBBS MLE (Gibbs MLE), and GIBBS RL (Gibbs Relearned). Result for each of the network types (noisy-or, relational, and grid) are shown.

using the MLE as well as parameters that were relearned (RL) after quantization. Figure 5.1 (a) to (c) shows our experimental results on each the three types of networks (noisy-or, relational, and grid).

The average Hellinger distances are consistently similar between MC-SAT and our method across the three network types when no parameters are tied (the MLE case). This is expected as MC-SAT is a special case of our algorithm. However, with the relearned parameters, our algorithm significantly outperforms both MC-SAT and Gibbs sampling on the students

and grid networks. This shows that even though the test-set log likelihood of the MLE solution and the parameter tied models are roughly the same, at prediction time (estimating marginals), models having tied parameters outperform untied models provided methods such as TW that explicitly exploit the tied parameters are used.

One explanation for the poor performance of MC-SAT on parameter tied models is that MC-SAT is based on local search with a strong bias towards features that have high weights. Thus, without the ability to make large moves, MC-SAT is not able to efficiently traverse through the state space. Analogously, MCMC techniques such as Gibbs sampling, can also become trapped within a local region and may require a large number of samples to escape. Since our algorithm systematically partitions the overall state-space through the tied weight structure, it is able to move across the various regions more easily.

5.4 Discussion

We introduced a new slice importance sampling technique that exploits the symmetry resulting from the parameter tied model. Specifically, better proposal distributions can be constructed over the groups of tied parameters (features). In addition, with slice sampling, the algorithm explores the sample space more effectively. Experimentally, our sampling algorithm outperforms comparable algorithms, such as MC-SAT and Gibbs sampling, when there are tied parameters in the model.

CHAPTER 6

DISSOCIATION-BASED BOUNDING ALGORITHMS

6.1 Introduction

It is well known that probabilistic inference in GM can be reduced to weighted model counting (WMC) (Chavira and Darwiche, 2008). The reduction has two main components. The first component is to encode the GM as a propositional knowledge base. The second component is to leverage state-of-the-art propositional model counters to develop a WMC-based algorithm for solving the desired inference task. However, a major drawback of the aforementioned methods is that they are computationally intractable for most real-world problems. Therefore, developing fast, scalable approximate schemes is a subject of fundamental interest.

In this chapter, we focus on developing a novel bounding algorithm for the WMC problem. Specifically, we extend the work of Gatterbauer and Suciu (Gatterbauer and Suciu, 2014, 2017), which is applicable to only monotone SAT formulas, to the task of WMC for arbitrary (*non-monotone*) formulas. Our method is related to the class of bounded complexity inference schemes such as mini-buckets (MB) and their extensions (Choi et al., 2007; Liu, 2014). MB relaxes the original problem by decomposing it into local subproblems (by splitting/dissociating nodes) that are then solved exactly.

We make the following contributions. First, we analyze the idea of dissociation based oblivious bounds (Gatterbauer and Suciu, 2014) using the framework of WMC and extend it to the general non-monotone case. Second, we take advantage of logical structure and derive a novel set of inequalities for bounding methods that dissociate until the formula has a tree structure (namely the *i-bound* in MB is equal to 1). Third, we theoretically compare the idea of dissociation with MB and show that MB bounds are a special case of our bounds and can be quite inferior. Lastly, we empirically demonstrate that dissociation based bounds are more accurate than MB on several synthetic and real-world datasets.

6.1.1 Weighted Model Counting Problem

Given a propositional formula F, a satisfying assignment or model of F is a truth assignment to all variables in F such that F evaluates to *true* ($\widetilde{\mathbf{x}} \models F$). As mentioned in Chapter 2, the problem of determining if there exists a satisfying assignment $\widetilde{\mathbf{x}}$ for F is called the *Boolean satisfiability problem* or SAT. *Propositional model counting* or #SAT is the task of computing the number of models of F.

The weighted model counting (WMC) problem (Chavira and Darwiche, 2008; Sang et al., 2005) extends model counting by associating the following probability distribution (weight function) ϕ_i to each propositional variable X_i.

$$\phi_i(X_i) = \begin{cases} p_i & \text{if } X_i \text{ evaluates to 1} \\ \overline{p_i} & \text{otherwise} \end{cases},$$

where $p_i \in [0,1]$ and $\overline{p_i} \triangleq 1 - p_i$.[1] The functions ϕ_i yield a *weighted representation* $\mathcal{F}$ of the CNF F and is called WCNF. Formally, $\mathcal{F}$ is a triple $\langle \boldsymbol{X}, \Phi, \mathbf{C} \rangle$, where $\boldsymbol{X}$ is a set of n Boolean variables in F, Φ is a set of weight functions ϕ_i associated with each Boolean variable $X_i \in \boldsymbol{X}$ and $\mathbf{C}$ is a set of clauses of F. $\mathcal{F}$ represents the following probability distribution

$$P_{\mathcal{F}}(\widetilde{\mathbf{x}}) = \begin{cases} \dfrac{1}{Z_{\mathcal{F}}} \displaystyle\prod_{i=1}^{n} \phi_i(X_i = x_i) & \text{if } \widetilde{\mathbf{x}} \models F \\ 0 & \text{otherwise} \end{cases},$$

where $Z_{\mathcal{F}}$ is the *partition function*, also referred to as the *weighted model count* (WMC) of $\mathcal{F}$, and is given by

$$Z_{\mathcal{F}} = \sum_{(\widetilde{\mathbf{x}} \in \Omega \wedge \widetilde{\mathbf{x}} \models F)} \prod_{i=1}^{n} \phi_i(X_i = x_i).$$

When $p_i = 1/2$ is assigned for all variables, then the product $2^n Z_{\mathcal{F}}$ equals the special case of (unweighted) model count of F.

[1] WMC is typically defined by attaching weights to literals, and the corresponding potential function over each variable is constructed by exponentiating the weights. We consider an equivalent representation in which the potential function is normalized to yield a probability distribution.

Table 6.1. Clauses for W2CNF Encoding of a GM.

$(\overline{X_i} \vee Y_{i,j,1})$	$(\overline{X_j} \vee Y_{i,j,1})$
$(\overline{X_i} \vee Y_{i,j,2})$	$(X_j \vee Y_{i,j,2})$
$(X_i \vee Y_{i,j,3})$	$(\overline{X_j} \vee Y_{i,j,3})$
$(X_i \vee Y_{i,j,4})$	$(X_j \vee Y_{i,j,4})$

6.1.2 WCNF Encoding of a GM

We describe here a possible translation from GM to WCNF. For more details see (Chavira and Darwiche, 2008; Gogate and Domingos, 2010, 2016). Since we focus on pairwise binary GMs, we can convert them to WCNFs in which each clause has at most two literals. We will refer to such WCNFs as W2CNF.

Encoding 1. Given a GM, we can construct an equivalent W2CNF as follows. We start with a W2CNF $\mathcal{F}$ defined over the variables $\boldsymbol{X}$ of the GM such that the set of clauses $\mathbf{C}$ of $\mathcal{F}$ is empty and $p_{X_i} = 0.5$ for each variable $X_i \in \boldsymbol{X}$. Then, for each pairwise binary potential $\psi_{i,j}$ in the GM such that $\psi_{ij} : X_i = 0, X_j = 0 \rightarrow w_{i,j,1}$, $\psi_{ij} : X_i = 0, X_j = 1 \rightarrow w_{i,j,2}$, $\psi_{ij} : X_i = 1, X_j = 0 \rightarrow w_{i,j,3}$, $\psi_{ij} : X_i = 1, X_j = 1 \rightarrow w_{i,j,4}$, we add a variable for each weight to $\mathcal{F}$. We will denote the variables associated with $w_{i,j,1}$, $w_{i,j,2}$, $w_{i,j,3}$ and $w_{i,j,4}$ by $Y_{i,j,1}$, $Y_{i,j,2}$, $Y_{i,j,3}$ and $Y_{i,j,4}$ respectively. Utilizing these weight variables, we add the the clauses given in Table 6.1 to $\mathbf{C}$ for $k = 1, \ldots, 4$. We also add the following probability distribution for each variable $Y_{i,j,k}$

$$\phi(y_{i,j,k}) = \begin{cases} \dfrac{w_{i,j,k} - 1}{w_{i,j,k}} & \text{if } y_{i,j,k} \text{ is } false \text{ or } 0 \\ \dfrac{1}{w_{i,j,k}} & \text{otherwise} \end{cases} .$$

Note that when $w_{i,j,k} < 1$, $\phi(y_{i,j,k})$ will be negative. To avoid this condition, we can easily rescale the potentials of the GM by multiplying them with an appropriate constant. Also,

X_1	X_2	$\psi_{1,2}(X_1, X_2)$
0	0	0
0	1	$\overline{p_1}p_2$
1	0	$p_1\overline{p_2}$
1	1	p_1p_2

X_1	X_3	$\psi_{1,3}(X_1, X_3)$
0	0	0
0	1	p_3
1	0	$\overline{p_3}$
1	1	p_3

Figure 6.1. Potentials for Example 6.2

zero weights can be handled by adding the corresponding negated assignment as a clause to $\mathbf{C}$. For example, if $w_{i,j,1} = 0$, we add the clause $X_i \vee X_j$ to $\mathbf{C}$. Using previous work (Chavira and Darwiche, 2008; Gogate and Domingos, 2010), it is straight-forward to show that:

Proposition 6.1. W2CNF *output by Encoding 1 represents the same probability distribution over* $\boldsymbol{X}$ *as the input GM.*

6.1.3 Mini-bucket Elimination for W2CNF

We can utilize inference algorithms such as bucket or variable elimination to compute the weighted model count of a W2CNF. However, since the complexity of using such algorithms is in general exponential in the treewidth (or induced width), a more practical approach is to approximate the task by introducing relaxations techniques that control model complexity (i.e., the induced width given a fixed elimination order). Mini-bucket (MB) is one such approximate scheme that builds on bucket elimination (BE) for generating upper (or lower) bounds on the partition function or weighted model count. We will use the following running example to illustrate BE and MB for WMC.

Example 6.2. *Consider the* W2CNF $\mathcal{F}$ *such that* $\boldsymbol{X} = \{X_1, Y_2, Y_3\}$, $\mathbf{C} = \{(X_1 \vee Y_2), (X_1 \vee Y_3)\}$ *and* $\Phi = \{\phi_1, \phi_2, \phi_3\}$. *For simplicity we denote* ϕ_1 *for* $\phi_1(X_1)$, *etc. We can convert the clauses and potentials of* $\mathcal{F}$ *to the two potentials shown in Figure 6.1 yielding a more convenient form for BE.*

Without loss of generality, we assume the elimination ordering as $[X_1, Y_2, Y_3]$ *(although it is clearly not optimal, it will help us illustrate the main ideas). BE begins by creating* $|\boldsymbol{X}|$ *number of buckets and groups the functions by placing each function involving some variable* X_i *(or* Y_i *in our example) in a bucket* $\mathbf{B}_{X_i}$ *according to the position of* X_i *in the ordering. The resulting computation is* $Z_{\mathcal{F}}^{\mathrm{BE}} = \sum_{Y_3} \sum_{Y_2} \sum_{X_1} \psi_{12}\psi_{13}$ *where* $\mathbf{B}_{X_1} = \{\psi_{12}, \psi_{13}\}$ *is first processed by taking the product of the two potentials and summing out variable* X_1*. The resulting new potential* ψ'_{23} *is placed in bucket* $\mathbf{B}_{Y_2}$ *in which variable* Y_2 *is summed out. Summing out* Y_3 *from the subsequent function* ψ'_3 *yields* $Z_{\mathcal{F}}^{\mathrm{BE}}$*.*

MB follows similarly. However, MB partitions each bucket into two or more so called mini-buckets according to an input parameter called the i-bound, which defines the maximum number (i-bound + 1) of variables in each mini-bucket. The mini-buckets are then processed independently. To obtain an upper bound, the sum-product operation is performed on one of the mini-buckets and the max-product for the remaining (min-product for lower bound). Using i-bound = 1, $\mathbf{B}_{X_1}$ *is split into two mini-buckets* $\mathbf{B}'_{X_1} = \{\psi_{12}\}$ *and* $\mathbf{B}''_{X_1} = \{\psi_{13}\}$*. One possible resulting computation is*

$$\underbrace{\sum_{Y_3Y_2}\left(\sum_{X_1}\psi_{12}\right)\left(\min_{X_1}\psi_{13}\right)}_{Z_{\mathcal{F}}^{\mathrm{MB}(L)}} \;\leq\; \sum_{Y_3}\sum_{Y_2}\sum_{X_1}\psi_{12}\psi_{13} \;\leq\; \underbrace{\sum_{Y_3Y_2}\left(\sum_{X_1}\psi_{12}\right)\left(\max_{X_1}\psi_{13}\right)}_{Z_{\mathcal{F}}^{\mathrm{MB}(U)}},$$

where the MB upper bound on the partition function, $Z_{\mathcal{F}}^{\mathrm{MB}(U)}$*, is computed by maxing out* X_1 *from* ψ_{13} *independently from summing out* X_1 *from* ψ_{12}*. Summing out* Y_2 *and* Y_3 *from the resulting two new potentials,* ψ'_2, ψ'_3*, and taking their product gives the upper bound. The lower bound,* $Z_{\mathcal{F}}^{\mathrm{MB}(L)}$*, is computed similarly using min instead of max.*

MB is a fast and simple algorithm for computing upper (or lower) bounds. The resulting complexity of inference is exponential in the *i-bound*. Lower *i-bound* values translates to simpler models and provides the trade-off between complexity and accuracy.

Next, we present the idea of dissociation based oblivious bounds for the case of monotone W2CNF and extend it to the non-monotone case by exploiting logical structure in Section 6.3. As mentioned earlier, in this dissertation, we focus on the case where variables are dissociated until the resulting formula is a tree. In other words, our scheme is comparable to the case when the *i-bound* in MB equals 1.

6.2 Dissociation

Our task is to compute the WMC $Z_{\mathcal{F}}$ of a given WCNF $\mathcal{F}$. Since the problem is computationally intractable in general (e.g., high treewidth), approximate methods are required. In this dissertation we use a *bounded inference* approach, where we approximate the original $\mathcal{F}$ with $\mathcal{F}'$ from which the upper (or lower) bounds on $Z_{\mathcal{F}}$ can be computed efficiently. We build upon (Gatterbauer and Suciu, 2014, 2017) which presents a bounding scheme called *dissociation* that can be applied to WMC. The derived bounds are *oblivious* to the set of weight functions ϕ_i, i.e. they can be calculated by only observing a limited subset of clauses. However, these bounds only apply to monotone formulas, whereas we are interested in extending the underlying ideas to more general non-monotone formulas (Section 6.3). Here, we first give a general intuition of prior results followed by the formal definition and then present optimal oblivious bounds for monotone formulas.

At a high level, dissociation is the process of replacing an existing variable X_i in $\mathcal{F}$ with new variables $X_{i;1}, \ldots, X_{i;d}$ and assigning them new probability distributions. The technique is closely related to *variable or node splitting* (Choi et al., 2007) in which the new variables are referred to as clones. The partitioning of mini-buckets can also be classified under the general notion of variable splitting.

By creating new variables, we are implicitly ignoring (or relaxing) a set of equality constraints (Choi and Darwiche, 2009). However, we can recover the set by defining and incorporating the function $\varphi(X_{i;1} = x_{i;1}, \ldots, X_{i;n} = x_{i;d})$ which evaluates to 1 iff $x_{i;1} = \ \ldots = x_{i;d}$,

and 0 otherwise, for the d copies $X_{i;j}, j \in [d]$ of variable X_i, and $x_{i;j}$ being the corresponding truth assignment. We can also incorporate equivalence clauses for each new pair of variables into a formula with the new clauses. For example, consider the formula $F = (X_1 \vee Y_1)(X_1 \vee Y_2)$. We can create the equivalent formula $F' = (X_{1;1} \vee Y_1)(X_{1;2} \vee Y_2)(X_{1;1} \Leftrightarrow X_{1;2})$ using copies of X_1 for the unweighted model counting case. We see two issues arising. First, for general 2-CNF formulas, we will require $d-1$ equality constraints (equivalence ($\Leftrightarrow$) formulas). Second, it is not immediately clear on how to integrate the weight functions so that weighted model counts can be computed using this scheme.

Dissociation expands on the notion of variable duplication and provides an *algebraic framework* to analyze and approximate the aforementioned set of equality constraints. The result is a novel class of inequalities to construct upper (or lower) bounds on the WMC. We first give the formal definition of dissociation for W2CNF.

Definition 6.3. (Dissociation). *Let $\mathcal{F} = \langle \boldsymbol{X}, \Phi, \mathbf{C} \rangle$. Select a variable $X_i \in \boldsymbol{X}$ and let $C(X_i) \subseteq \mathbf{C}$ be the subset of all clauses that involve variable X_i. We say $\mathcal{F}' = \langle \boldsymbol{X}', \Phi', \mathbf{C}' \rangle$ is a dissociation of $\mathcal{F}$ on X_i iff*

- $\boldsymbol{X}' = \boldsymbol{X} \setminus X_i \cup X_{i;1} \cup \cdots \cup X_{i;d}$ *with* $d \leq |C(X_i)|$,
- $\Phi' = \Phi \setminus \phi_i \cup \phi_{i;1} \cup \cdots \cup \phi_{i;d}$, *and*
- $\mathbf{C}'[\theta_{X_i}(\boldsymbol{X}')] = \mathbf{C}[\boldsymbol{X}]$ *with* θ_{X_i} *being the substitution* $\theta_{X_i}[\{(X_{i;j}/X_i), j \in [d]\}]$.

We say a dissociation is full if $d = |C(X_i)|$.

Example 6.4. (Dissociation). *Consider $\mathcal{F}$ from Example 6.2. Dissociating X_1 results in adding two new variables, $\boldsymbol{X}' = \boldsymbol{X} \setminus X_1 \cup X_{1;1} \cup X_{1;2}$, and two new associated weight functions $\Phi' = \Phi \setminus \phi_1 \cup \phi_{1;1} \cup \phi_{1;2}$. Applying the substitution $\theta_{X_1}[(X_{1;1}/X_1), (X_{1;2}/X_1)]$ on $C(X_i)$ results in $\mathbf{C}' = \mathbf{C} \setminus C(X_i) \cup (X_{1;1} \vee Y_2) \cup (X_{1;2} \vee Y_3)$.*

Table 6.2. Dissociation valuation analysis for Example 6.5. * denotes the don't care condition.

Y_2	Y_3	X_1	$X_{1;1}$	$X_{1;2}$	ϕ_1	$\phi_{1;1}, \phi_{1;2}$
0	0	1	1	1	p_1	$p_{1;1}p_{1;2}$
0	1	1	1	$*$	p_1	$p_{1;1}$
1	0	1	$*$	1	p_1	$p_{1;2}$
1	1	$*$	$*$	$*$	1	1

Once we have defined the new weight functions (for dissociated variables), the question we are interested in is how to parameterize the new functions in order to obtain guaranteed upper (or lower) bounds. In particular, we are interested in *oblivious bounds*, i.e. when the *new probabilities are chosen independently of the probabilities of all other variables*. We achieve that by considering all possible valuations (or truth assignments) of the non-dissociated variables, The assignments give rise to a set of inequalities which are then evaluated to develop necessary and sufficient conditions for upper (or lower) bounds. We next illustrate with an example.

Example 6.5. (Oblivious bounds). *Consider the two sets of clauses,* $\{(X_1 \vee Y_2), (X_1 \vee Y_3)\}$ *and* $\{(X_{1;1} \vee Y_2), (X_{1;2} \vee Y_3)\}$ *from Example 6.2 and Example 6.4. We analyze the* $2^2 = 4$ *possible truth assignments to the non-dissociated variables* Y_2 *and* Y_3*. Table 6.2 shows each possible valuation of* Y_2 *and* Y_3 *and the corresponding assignments to* $X_1, X_{1;1}$ *and* $X_{1;2}$ *required to satisfy the clauses. We also show the weights (probabilities) of the original (column* ϕ_1*) and dissociated formulas (column* $\phi_{1;1}\phi_{1;2}$*).*

For example, consider the assignment, $Y_2 = 0 \wedge Y_3 = 1$*: The assignment* $X_1 = 1$ *is required to satisfy* $\mathcal{F}$*, resulting in the term* $p_1\overline{p}_2p_3$*. The assignments* $(X_{1;1} = 1 \wedge X_{1;2} = 0)$ *or* $(X_{1;1} = 1 \wedge X_{1;2} = 1)$ *are required to satisfy* $\mathcal{F}'$*, i.e.* $X_{1;2}$ *can take any assignment* $(*)$*, resulting in the term* $p_{1;1}\overline{p}_2p_3$*. Utilizing the two terms, simplifying by removing the common terms* $(\overline{p}_2p_3)$ *and assuming that we are interested in computing lower bounds, we create the inequality* $p_1 \geq p_{1;1}$*. Repeating the same analysis for the three remaining cases results in the inequalities*

$p_1 \geq p_{1;1}p_{1;2}$ *and* $p_1 \geq p_{1;2}$*. The last case* $1 \geq 1$ *is trivially satisfied. Combining the resulting inequalities, and doing a similar analysis for computing upper bounds (where we replace* $\geq$ *by* $\leq$*) gives rise to the following conditions for oblivious (U)pper and (L)ower bounds:*

- *U*: $(p_1 \leq p_{1;1}p_{1;2}) \wedge (p_1 \leq p_{1;1}) \wedge (p_1 \leq p_{1;2})$.
- *L*: $(p_1 \geq p_{1;1}p_{1;2}) \wedge (p_1 \geq p_{1;1}) \wedge (p_1 \geq p_{1;2})$.

Notice the valuation process creates $2^{|C(X_i)|}$ inequalities, one for each truth assignment. However, we can simplify the conditions by removing subsumed inequalities.

Definition 6.6. (Subsumed inequality). *We say an inequality* I_i *subsumes inequality* I_j $(i \neq j)$ *iff* $I_i \Rightarrow I_j$*, i.e. satisfying* I_i *also satisfies* I_j*.*

Example 6.7. *Consider the upper and lower bound conditions in Example 6.5. For the upper bound, clearly* $p_1 \leq p_{1;1}p_{1;2}$ *subsumes the remaining inequalities since* $\forall p_1, p_{1;1}, p_{1;2} \in [0,1]$: $(p_1 \leq p_{1;1}p_{1;2}) \Rightarrow (p_1 \leq p_{1;1}) \wedge (p_1 \leq p_{1;2})$*. For the lower bound, clearly* $(p_1 \geq p_{1;1}) \wedge (p_1 \geq p_{1;2})$ *subsumes the remaining inequality since* $\forall p_1, p_{1;1}, p_{1;2} \in [0,1]$: $((p_1 \geq p_{1;1}) \wedge (p_1 \geq p_{1;2})) \Rightarrow (p_1 \geq p_{1;1}p_{1;2})$*. Therefore, we can reduce the required conditions for the oblivious bounds to:*

- *U*: $p_1 \leq p_{1;1}p_{1;2}$.
- *L*: $(p_1 \geq p_{1;1}) \wedge (p_1 \geq p_{1;2})$.

Following the preceding analysis, we can now state the conditions for oblivious bounds for monotone W2CNF.

Theorem 6.8. (Gatterbauer and Suciu, 2014) (Oblivious bounds for monotone W2CNF). *Let* $\mathcal{F}$ *be a monotone* W2CNF*. Let* $\mathcal{F}'$ *be the result of applying a series of dissociation steps on* $\mathcal{F}$*. For every set of weight functions defined for a dissociate variable, namely* $X_{i;1}, \ldots, X_{i;d}$ *and* $\{\phi_{i;1}, \ldots, \phi_{i;d}\}$ *with* $d > 1$*, we have the following* oblivious bounds*:*

- U: $\prod_{j=1}^{d} p_{i;j} \geq p_i$.
- L: $\forall j : p_{i;j} \leq p_i$.

Optimal oblivious bounds are defined as those that are not dominated, i.e. they cannot be improved without knowledge of the probabilities of all other variables. They are obtained by replacing inequality with equality. Notice that optimal oblivious lower bounds are uniquely defined, $\forall j : p_{i;j} = p_i$, whereas there are infinitely many optimal oblivious upper bounds, e.g. symmetric ones: $\forall j : p_{i;j} = \sqrt[d]{p_i}$, and finding the best one requires access to all other probabilities (den Heuvel et al., 2018).

Note that optimal oblivious bounds are different from augmented mini-buckets (AMB) (Liu, 2014). For example, in AMB for computing upper bounds, the potential over each dissociated variable is initialized to $\phi_{i;j}(X_{i;j} = 1) = \phi_i(X_i = 1)^{1/d}$ and $\phi_{i;j}(X_{i;j} = 0) = \phi_i(X_i = 0)^{1/d}$ where we have d dissociations. A better initialization would be $\phi_{i;j}(X_{i;j} = 1) = \phi_i(X_i = 1)^{1/d}$, and $\phi_{i;j}(X_{i;j} = 0) = 1 - \phi_i(X_{i;j} = 1)^{1/d}$.

6.2.1 Comparison with Mini-bucket

We use Example 6.2 to analyze the base case bounds for monotone dissociation (X_1 to $X_{1;1}$ and $X_{1;2}$) and compare it with MB (*i-bound* $= 1$).

Lower bound. Dissociation results in the partition function

$$Z_{\mathcal{F}'}^{\mathrm{DIS}(U)} = p_2 p_3 + p_{1;1}\overline{p_2} p_3 + p_{1;2} p_2 \overline{p_3} + p_{1;1} p_{1;2} \overline{p_2 p_3}.$$

The two possible partition functions according to MB are:

1. $\sum_{X_1} \psi_1 \min_{X_1} \psi_2 \Rightarrow Z_{\mathcal{F}}^{\mathrm{MB}(L1)} = p_2 p_3 + p_1 \overline{p_2} p_3$;
2. $\min_{X_1} \psi_1 \sum_{X_1} \psi_2 \Rightarrow Z_{\mathcal{F}}^{\mathrm{MB}(L2)} = p_2 \min(\overline{p_1}, p_1)(1 + p_3)$.

Clearly, $Z_{\mathcal{F}'}^{\mathrm{DIS}(L)} \geq Z_{\mathcal{F}}^{\mathrm{MB}(L)} \quad \forall p_1, p_{1;1}, p_{1;2}, p_2, p_3 \in [0, 1]$.

Upper bound. Notice there exist an infinite number of settings to $p_{1;1}$ and $p_{1;2}$ that satisfy $p_{1;1}p_{1;2} = p_1$ under dissociation. We analyze two possible cases.

1. $(p_{1;1} = p_1) \wedge (p_{1;2} = 1) \Rightarrow Z_{\mathcal{F}'}^{\mathrm{DIS}(U1)} = p_{1;1} + \overline{p_{1;1}}p_2$;

2. $(p_{1;2} = p_1) \wedge (p_{1;1} = 1) \Rightarrow Z_{\mathcal{F}'}^{\mathrm{DIS}(U2)} = p_{1;2} + \overline{p_{1;2}}p_3$.

The two possible partition functions according to MB are:

1. $\sum_{X_1} \psi_1 \max_{X_1} \psi_2 \Rightarrow Z_{\mathcal{F}}^{\mathrm{MB}(U1)} = p_1 + \overline{p_1}p_1$;

2. $\max_{X_1} \psi_1 \sum_{X_1} \psi_2 \Rightarrow Z_{\mathcal{F}}^{\mathrm{MB}(U2)} = (1 + p_3)(p_2 \max(\overline{p_1}, p_1) + p_1\overline{p_1})$.

We first observe the bounds are equivalent between dissociation and MB in setting (1) and also for (2) if the functions are unweighted (e.g., $\forall i\ p_i = 1/2$). However, note there exist more degrees of freedom (solutions) for dissociation, and this example simply demonstrates one such setting for which we observe equivalency under certain conditions.

6.3 Dissociation for Non-monotone Formulas

In this section, we extend dissociation bounds from the monotone case to arbitrary non-monotone W2CNFs. Unlike monotone W2CNFs, we can apply logical inference techniques such as resolution and unit propagation to reduce non-monotone W2CNFs which in turn may improve our dissociation-based bounds. Moreover, logical propagation can be applied as a preprocessing step before dissociating a variable X_i.

6.3.1 Preprocessing

We say that a W2CNF $\mathcal{F}$ is *minimal* if the following steps are applied to its set of clauses $\mathbf{C}$ until convergence.

1. **(Binary) Resolution**: If $\mathbf{C}$ contains two clauses of the form $L_i \vee L_j$ and $\overline{L}_i \vee L_k$, where L_i, L_j and L_k are literals of variables X_i, X_j and X_k respectively, we add the clause $L_j \vee L_k$ to $\mathbf{C}$.

2. **Unit Resolution**: If $\mathbf{C}$ contains two clauses of the form $L_i \vee L_j$ and $\overline{L}_i \vee L_j$, where L_i and L_j are literals of variables X_i and X_j respectively, we add the unit clause L_j to $\mathbf{C}$.

3. **Clause Deletion and Reduction**: If $\mathbf{C}$ contains a unit clause L_i where L_i is a literal of X_i then we delete all clauses of the form $L_i \vee L_j$ and remove $\overline{L}_i$ from all clauses that mention $\overline{L}_i$. If $\mathbf{C}$ contains both unit clauses L_i and $\overline{L_i}$, $\mathbf{C}$ is inconsistent and we return a lower/upper bound of 0.[2]

Example 6.9 (Minimal formula)**.** *Consider* $\mathbf{C} = \{(X_1 \vee \overline{X_2}), (X_1 \vee X_2), (\overline{X_2} \vee Y_4), (X_1 \vee X_3), (X_3 \vee X_5)\}$. $\mathbf{C}$ *is not minimal and we can make it minimal using the aforementioned steps. After applying Unit Resolution on the first two clauses, we get* $\mathbf{C} = \{(X_1), (X_1 \vee \overline{X_2}), (X_1 \vee X_2), (\overline{X_2} \vee Y_4), (X_1 \vee X_3), (X_3 \vee X_5)\}$. *After applying Clause deletion and Reduction, we get* $\mathbf{C} = \{(X_1), (\overline{X_2} \vee Y_4), (X_3 \vee X_5)\}$*, which is minimal.*

6.3.2 Types of Non-Monotone Formulas

In the sequel, we assume that the input W2CNF $\mathcal{F}$ to our algorithm is minimal. To formulate oblivious bounds for non-monotone W2CNF, we first establish a *canonical representation* that helps us take advantage of symmetry and reduces the number of cases (inequalities) we need to consider for our proposed oblivious bounds. Specifically, given a candidate *dissociation variable* X_i, we convert the set of clauses $\mathbf{C}$ into a canonical representation:

Definition 6.10 (Canonical representation)**.** *We say that* $\mathcal{F}$ *is canonical w.r.t. a variable* X_i *if* $\mathcal{F}$ *is minimal and all clauses in* $C(X_i)$ *satisfy the following two properties:*

[2]Note that our scheme will return an upper bound of 0 only when $\mathbf{C}$ is inconsistent.

1. *If a variable Y_j appears only once in $C(X_i)$ then it only appears positively, i.e. it appears in clauses of the form $X_i \vee Y_j$ or $\overline{X}_i \vee Y_j$ (but not of the form $X_i \vee \overline{Y}_j$ or $\overline{X}_i \vee \overline{Y}_j$).*

2. *If a variable Y_j appears twice in $C(X_i)$, then it appears in the following two clauses $X_i \vee Y_j$ and $\overline{X}_i \vee \overline{Y}_j$ (but not in the clauses $\overline{X}_i \vee Y_j$ and $X_i \vee \overline{Y}_j$).*

Note that since $\mathcal{F}$ is minimal, Y_j cannot appear more than twice in $C(X_i)$, nor twice with the same sign. If $C(X_i)$ is not in canonical form, we can easily make it canonical by using the following procedure:

- If Y_j violates either condition (1) or (2) in definition 6.10, then replace Y_j by a new variable Y_k in all clauses of $\mathcal{F}$ (where Y_j appears) such that $Y_k = \overline{Y}_j$, and set $\phi(Y_k) = \phi(\overline{Y_j})$ and $\phi(\overline{Y_k}) = \phi(Y_j)$.

Example 6.11 (Canonical representation)**.** *Consider* $\boldsymbol{C} = \{(X_1 \vee \overline{Y_2}), (\overline{X_1} \vee Y_2), (X_1 \vee \overline{Y_3})\}$. $\mathbf{C}$ *is not in canonical form w.r.t.* X_1 *because* Y_2 *and* Y_3 *violate the second and first property respectively in definition 6.10. To convert it to canonical form, set* $Y_4 = \overline{Y_2}$, $Y_5 = \overline{Y_3}$, $\phi(Y_4) = \phi(\overline{Y_2})$, $\phi(\overline{Y_4}) = \phi(Y_2)$, $\phi(Y_5) = \phi(\overline{Y_3})$ *and* $\phi(\overline{Y_5}) = \phi(Y_3)$. *Thus, the canonical representation of* $\mathbf{C}$ *is the set* $\{(X_1 \vee Y_4), (\overline{X_1} \vee \overline{Y_4}), (X_1 \vee Y_5)\}$.

We call variables Y_j which appear only once in $C(X_i)$ *single-occurrence neighbors* of X_i and those which appear twice *two-occurrence neighbors.*

6.3.3 Characterizing Oblivious Bounds

We now derive oblivious bounds based on whether $C(X_i)$ has two-occurrence neighbors or not. In the following, let $\mathcal{F}$ denote a W2CNF that is canonical w.r.t. X_i and let $\mathcal{F}'$ be the result of applying a series of dissociation steps on $\mathcal{F}$. Let Y_j be a single-occurrence neighbor of X_i. Let S^+ and S^- denote the set of indices of the dissociated variables that appear in clauses $(X_i \vee Y_j)$ and $(\overline{X_i} \vee Y_j)$ respectively in $C(X_i)$. Let Y_k be a two-occurrence neighbor

of X_i. Let T^+ and T^- denote the set of indices of the dissociated variables in clauses $X_i \vee Y_k$ and $\overline{X}_i \vee \overline{Y}_k$ respectively in $C(X_i)$. (We use S and T to refer to "single-occurrence" and "two-occurrence" variables, respectively.)

Example 6.12 (Indices)**.** *Consider* $\mathbf{C} = \{(X_1 \vee Y_5), (X_1 \vee Y_8), (\overline{X_1} \vee Y_6), (X_1 \vee Y_7), (\overline{X_1} \vee \overline{Y_7}), (X_1 \vee Y_9), (\overline{X_1} \vee \overline{Y_9})\}$. *After applying dissociation on* X_1*, we get* $C(X_1') = \{(X_{1;1} \vee Y_5), (X_{1;2} \vee Y_8), (\overline{X_{1;3}} \vee Y_6), (X_{1;4} \vee Y_7), (\overline{X_{1;5}} \vee \overline{Y_7}), (X_{1;6} \vee Y_9), (\overline{X_{1;7}} \vee \overline{Y_9})\}$. *Then* $S^+ = \{1, 2\}, S^-=\{3\}, T^+ = \{4, 6\}$, *and* $T^- = \{5, 7\}$.

We next analyze the two possible non-monotone cases in Theorems 6.13 and 6.15. The first case is when $C(X_i)$ has only single-occurrence neighbors (but no two-occurrence neighbors). This generalizes the monotone case, in which only one type of single-occurrence variables are present. In particular, in the monotone case either clauses of the form $(X_i \vee Y_j)$ or $(\overline{X_i} \vee Y_j)$ are present but not both while in the non-monotone case both clauses can be present in $C(X_i)$. Note that bounds given in Theorem 6.8 are a special case of the bounds in Theorem 6.13 presented next.

Theorem 6.13. (Oblivious bounds for W2CNFs having only single-occurrence neighbors w.r.t. X_i). *For a given variable* X_i*, if* $\mathcal{F}$ *contains only single-occurrence neighbors but no two-occurrence neighbors then we have the following oblivious bounds for* X_i*:*

- *U*: $\Big(\prod_{j \in S^+} p_{i;j} \geq p_i\Big) \wedge \Big(\prod_{j \in S^-} \overline{p_{i;j}} \geq \overline{p_i}\Big)$
- *L*: *Either of following two conditions hold:*
  1. $\Big(\forall j \in S^+ \colon p_{i;j} \leq p_i\Big) \wedge \Big(\forall j \in S^- \colon \overline{p_{i;j}} = 0\Big)$
  2. $\Big(\forall j \in S^- \colon \overline{p_{i;j}} \leq \overline{p_i}\Big) \wedge \Big(\forall j \in S^+ \colon p_{i;j} = 0\Big)$

Optimal oblivious bounds are obtained by replacing inequality with equality in the bound conditions.

Proof. Let $\mathcal{F}$ be a W2CNF that is minimal and in canonical form. Moreover, for a given X_i, let $\mathcal{F}$ contain at least one two-occurrence neighbor.

- **Case**: $X_i = *$

  Non-dissociate: Let Y be the set of single-occurrence neighbors of the non-dissociated variable X_i such that $|Y| > 1$. If $\forall Y_k \in Y, Y_k = 1$, then $C(X_i)$ is a tautology. The result is that $C(X_i)$ is satisfied regardless of the assignment to X_i. Therefore, we have the term

  $$p_i \prod_{Y_k \in Y} p_k + \overline{p_i} \prod_{Y_k \in Y} p_k = (p_i + \overline{p_i}) \prod_{Y_k \in Y} p_k = \prod_{Y_k \in Y} p_k.$$

  Dissociate: Let Y be the set of single-occurrence neighbors of the dissociated variable X_i'. If $\forall Y_k \in Y, Y_k = 1$, then $C(X_i')$ is a tautology. The result is that $C(X_i')$ is satisfied regardless of the assignment to $X_{i;j}$, $\forall j$. Therefore, we have the term

  $$\prod_{Y_k \in Y} p_k \prod_{j=1}^{|C(X_i)|} (p_{i;j} + \overline{p_{i;j}}) = \prod_{Y_k \in Y} p_k.$$

  The inequality condition for this case is trivially satisfied since

  $$\prod_{Y_k \in Y} p_k \leqslant \prod_{Y_k \in Y} p_k \Rightarrow 1 = 1.$$

Without loss of generality (WLOG), since the parameters $(\forall Y_k \in Y : \prod p_k)$ for the single-occurrence neighbors appears on both sides of the inequality, and therefore cancels out, we do not include it in the subsequent case analysis. Let $P = \{p_{i;j} : j \in (S^+ \cup S^-)\}$ be the set of parameters corresponding to the dissociated variables $X_{i;j}$. Then, for the remaining $2^{|C(X_i)|} - 1$ assignments to the single-occurrence neighbors to consider, the number of possible combinations of P is given as

$$\sum_{a=1}^{|C(X_i)|} \binom{|C(X_i)|}{a}. \tag{6.1}$$

Let Y_+ be single-occurrence neighbors of the positive appearing non-dissociated variable X_i and let Y_- be single-occurrence neighbors of the negative appearing non-dissociated variable $\overline{X_i}$.

- **Case**: $X_i = \perp$

  Non-dissociate: If for some $Y_j \in Y_+$ and $Y_k \in Y_-$ such that $j \neq k$ and $Y_j = Y_k = 0$, then $C(X_i)$ is unsatisfiable (i.e., contradiction). Therefore, we have the term 0.

  Dissociate: From (6.1), let P' be the set of possible combinations of P by setting $a = 2$. Let P'' be the set of possible combinations of P for $a = 3$ to $a = |C(X_i)|$. We know $\forall b \in P''$, the corresponding term from non-dissociation is 0 because the number of don't care assignments to $X_{i;j}, \forall j$ is less than 2. Then we only need to consider the conditions when $a = 2$ because $\forall b \in P'$, $b \subseteq P''$ and therefore satisfies the subsumed inequality definition. For $a = 2$, let $P''' = (\forall j \in S^+ : p_{i;j}) \times (\forall j \in S^- : \overline{p_{i;j}})$ s.t. $P''' \subseteq P'$. Setting either $p_{i;j} \in P'''$ or $\overline{p_{i;j}} \in P'''$ to 0 satisfies the case. Therefore, the lower bound condition for this case is

$$(\forall j \in S^- : \overline{p_{i;j}} = 0) \vee (\forall j \in S^+ : p_{i;j} = 0).$$

Let $P'''' = (P' \setminus P''') \cup P$. Let Y'_+ be single-occurrence neighbors of dissociated variable $X_{i;j}$ s.t. $j \in S^+$ and let Y'_- be single-occurrence neighbors of dissociated variable $\overline{X_{i;j}}$ s.t. $j \in S^-$. If $\forall\, Y_j \in Y^+$ and $\forall\, Y_k \in Y^-$, set $Y_j = 1, Y_k = 1$. Then the associated clauses of the form $(\overline{X_i} \vee Y_j)(X_i \vee Y_k)$ are trivially satisfied (i.e., case: $X_i = *$). Similarly, if $\forall\, Y_k \in Y'_+$ and $\forall\, Y_l \in Y'_-$, set $Y_k = 1, Y_l = 1$. Then the associated clauses of the form $(\overline{X_{i;j}} \vee Y_k)(X_{i;j} \vee Y_l)$ are trivially satisfied (i.e., case: $X_i = *$).

- **Case**: $X_i = 1$

  Non-dissociate: If $\forall\, Y_j \in Y_+$ and $\forall\, Y_k \in Y_-$, set $Y_j = 1, Y_k = 0$, then the associated clauses of the form $(X_i \vee Y_k)$ are satisfied by setting $X_i = 1$. Therefore, we have the term p_i.

  Dissociate: If $\forall\, Y_k \in Y'_+$, and $\forall\, Y_l \in Y'_-$, set $Y_k = 1, Y_l = 0$, then the associated clauses

of the form $(X_{i;j} \vee Y_l)$ are satisfied by setting $X_{i;j} = 1$. The resulting term is $p_{i;j}$. We know $\forall b \in P''''$ s.t. $p_{i;j} \in b$, the corresponding term from non-dissociation is p_i. Then by the definition of subsumed inequality, the lower bound condition for this case is

$$\forall j \in S^+ : p_{i;j} \leq p_i,$$

and the upper bound condition is

$$\prod_{j \in S^+} p_{i;j} \geq p_i.$$

- **Case**: $X_i = 0$

  Non-dissociate: If $\forall\, Y_j \in Y_+$ and $\forall\, Y_k \in Y_-$, set $Y_j = 0, Y_k = 1$, then the associated clauses of the form $(\overline{X_i} \vee Y_j)$ are satisfied by setting $X_i = 0$. Therefore, we have the term $\overline{p_i}$.

  Dissociate: If $\forall\, Y_k \in Y'_+$, and $\forall\, Y_l \in Y'_-$, set $Y_k = 0, Y_l = 1$, then the associated clauses of the form $(\overline{X_{i;j}} \vee Y_k)$ are satisfied by setting $X_{i;j} = 0$. The resulting term is $\overline{p_{i;j}}$. We know $\forall b \in P''''$ s.t. $\overline{p_{i;j}} \in b$, the corresponding term from non-dissociation is $\overline{p_i}$. Then by the definition of subsumed inequality, the lower bound condition for this case is

  $$\forall j \in S^- : \overline{p_{i;j}} \leq \overline{p_i},$$

  and the upper bound condition is

  $$\prod_{j \in S^-} \overline{p_{i;j}} \geq \overline{p_i}.$$

Combining the four cases gives the conditions for the oblivious bounds for X_i.

□

Example 6.14. *Consider* $C(X'_1) = \{(X_{1;1} \vee Y_2), (\overline{X_{1;2}} \vee Y_3), (X_{1;3} \vee Y_4), (\overline{X_{1;4}} \vee Y_5)\}$. *Theorem 6.13 gives the conditions for upper and lower oblivious bounds as:*

Table 6.3. Dissociation for Example 6.14. $\perp$ denotes contradiction (i.e., formula cannot be satisfied). $*$ denotes the don't care condition.

Y_2	Y_3	Y_4	Y_5	X_1	$X_{1;1}$	$X_{1;2}$	$X_{1;3}$	$X_{1;4}$	ϕ_1	$\phi_{1;1}, \phi_{1;2}, \phi_{1;3}, \phi_{1;4}$
0	0	0	0	$\perp$	1	0	1	0	0	$p_{1;1}\overline{p_{1;2}}p_{1;3}\overline{p_{1;4}}$
0	0	0	1	$\perp$	1	0	1	$*$	0	$p_{1;1}\overline{p_{1;2}}p_{1;3}$
0	0	1	0	$\perp$	1	0	$*$	0	0	$p_{1;1}\overline{p_{1;2}}\overline{p_{1;4}}$
0	0	1	1	$\perp$	1	0	$*$	$*$	0	$p_{1;1}\overline{p_{1;2}}$
0	1	0	0	$\perp$	1	$*$	1	0	0	$p_{1;1}p_{1;3}\overline{p_{1;4}}$
0	1	0	1	1	1	$*$	1	$*$	p_1	$p_{1;1}p_{1;3}$
0	1	1	0	$\perp$	1	$*$	$*$	0	0	$p_{1;1}\overline{p_{1;4}}$
0	1	1	1	1	1	$*$	$*$	$*$	p_1	$p_{1;1}$
1	0	0	0	$\perp$	$*$	0	1	0	0	$\overline{p_{1;2}}p_{1;3}\overline{p_{1;4}}$
1	0	0	1	$\perp$	$*$	0	1	$*$	0	$\overline{p_{1;2}}p_{1;3}$
1	0	1	0	0	$*$	0	$*$	0	$\overline{p_1}$	$\overline{p_{1;2}}\overline{p_{1;4}}$
1	0	1	1	0	$*$	0	$*$	$*$	$\overline{p_1}$	$\overline{p_{1;2}}$
1	1	0	0	$\perp$	$*$	$*$	1	0	0	$p_{1;3}\overline{p_{1;4}}$
1	1	0	1	1	$*$	$*$	1	$*$	p_1	$p_{1;3}$
1	1	1	0	0	$*$	$*$	$*$	0	$\overline{p_1}$	$\overline{p_{1;4}}$
1	1	1	1	$*$	$*$	$*$	$*$	$*$	1	1

- U: $\big(p_{1;1}p_{1;3} \geq p_1\big) \wedge \big(\overline{p_{1;2}}\overline{p_{1;4}} \geq \overline{p_1}\big)$.
- L: *Either of following two conditions hold:*
  1. $\big(p_{1;1} \leq p_1\big) \wedge \big(p_{1;3} \leq p_1\big) \wedge \big(\overline{p_{1;2}} = \overline{p_{1;4}} = 0\big)$
  2. $\big(\overline{p_{1;2}} \leq \overline{p_1}\big) \wedge \big(\overline{p_{1;4}} \leq \overline{p_1}\big) \wedge \big(p_{1;1} = p_{1;3} = 0\big)$

The valuation analysis is shown in Table 6.3.

Our second non-monotone case is when $\mathcal{F}$ has at least one two-occurrence neighbor. Intuitively, dissociated variables which form clauses with two-occurrence neighbors are more constrained than those that appear with single-occurrence neighbors. Thus, there are more constraints on probabilities associated with two-occurrence neighbors (indexed by T^+ and

T^-) than those associated with single-occurrence neighbors (indexed by S^+ and S^-); see conditions 1. and 2. in Theorem 6.15.

Theorem 6.15. (Oblivious bounds for W2CNFs having two-occurrence neighbors w.r.t. X_i). *For a given variable X_i, if $\mathcal{F}$ contains at least one two-occurrence neighbor then we have the following oblivious bounds for X_i:*

- *U*: $\Big(\prod_{j\in(S^+\cup T^+)} p_{i;j} \geq p_i\Big) \wedge \Big(\prod_{j\in(S^-\cup T^-)} \overline{p_{i;j}} \geq \overline{p_i}\Big)$
- *L*: *Either of following three conditions hold:*
  1. $\Big(\prod_{j\in T^+} p_{i;j} \leq p_i\Big) \wedge \Big(\forall j \in (S^- \cup T^-) \colon \overline{p_{i;j}} = 0\Big)$
  2. $\Big(\prod_{j\in T^-} \overline{p_{i;j}} \leq \overline{p_i}\Big) \wedge \Big(\forall j \in (S^+ \cup T^+) \colon p_{i;j} = 0\Big)$
  3. *If* $|T^+| = |T^-| = 1$ *and* $T^+ = \{a\} \wedge T^- = \{b\}$*:*
     $\Big(p_{i;a} \leq p_i\Big) \wedge \Big(\forall j \in S^- \colon \overline{p_{i;j}} = 0\Big) \wedge \Big(\overline{p_{i;b}} \leq \overline{p_i}\Big) \wedge \Big(\forall j \in S^+ \colon p_{i;j} = 0\Big)$

Optimal oblivious bounds are obtained by replacing inequality with equality in the bound conditions.

Proof. Let $\mathcal{F}$ be a W2CNF that is minimal and in canonical form. Moreover, for a given X_i, let $\mathcal{F}$ contain at least one two-occurrence neighbor. Similar to the proof of Theorem 6.13, WLOG, the parameters for the non-dissociated variables cancels out and thus are not included in the case analysis.

- **Case**: $X_i = \bot$

  Let $|T^-| \geq 1$ and $|T^+| \geq 1$.

  Let $|S^-| > 0$.

Non-dissociate:

Let Y_l be a single-occurrence neighbor appearing with negative non-dissociated variable $\overline{X_i}$ and Y_j be a two-occurrence neighbor appearing with positive non-dissociated variable X_i in clauses of the form $(X_i \vee Y_j)(\overline{X_i} \vee Y_l)$ s.t. $j \neq l$. If $Y_j = Y_l = 0$, then the $C(X_i)$ cannot be satisfied (i.e., contradiction). Therefore, we have the term 0.

Dissociate:

Let Y_l be a single-occurrence neighbor appearing with negative dissociated variable $\overline{X_{i;k}}$ and Y_h be a two-occurrence neighbor appearing with positive dissociated variable $X_{i;j}$ in clauses of the form $(X_{i;j} \vee Y_h)(\overline{X_{i;k}} \vee Y_l)$ s.t. $j \neq k$ and $h \neq l$. Assign $Y_h = Y_l = 0$. Then we have the term

$$\prod_{j \in T^+} p_{i;j} \prod_{k \in S^-} \overline{p_{i;k}}.$$

Let $|S^+| > 0$.

Non-dissociate:

Let Y_l be a single-occurrence neighbor appearing with positive non-dissociated variable X_i and Y_j be a two-occurrence neighbor appearing with positive non-dissociated variable X_i in clauses of the form $(X_i \vee Y_j)(\overline{X_i} \vee \overline{Y_j})(X_i \vee Y_l)$ s.t. $j \neq l$. If $Y_j = 1$ and $Y_l = 0$, then $C(X_i)$ cannot be satisfied (i.e., contradiction). Therefore, we have the term 0.

Dissociate:

Let Y_h be a two-occurrence neighbor appearing with negative dissociated variable $\overline{X_{i;j}}$ and Y_l be a single-occurrence neighbor appearing with dissociated variable $X_{i;k}$ in clauses of the form $(\overline{X_{i;j}} \vee \overline{Y_h})(X_{i;k} \vee Y_l)$ s.t. $j \neq k$ and $h \neq l$. Assign $Y_h = 1$ and $Y_l = 0$. Then we have the term

$$\prod_{j \in T^-} \overline{p_{i;j}} \prod_{k \in S^+} p_{i;k}.$$

Let $|T^-| > 1$ and $|T^+| > 1$. Additionally, we have the following.

Non-dissociate:

Let Y_j and Y_k be two-occurrence neighbors appearing with positive non-dissociate variable X_i in clauses of the form $(X_i \vee Y_j)(\overline{X_i} \vee \overline{Y_j})(X_i \vee Y_k)(\overline{X_i} \vee \overline{Y_k})$ s.t. $j \neq k$. Let a_1 and a_2 be 0/1 assignments to Y_j and Y_k respectively. If $a_1 \neq a_2$, then $C(X_i)$ cannot be satisfied (i.e., contradiction). Therefore, we have the term 0.

Dissociate:

Let Y be the set of two-occurrence neighbors appearing with positive non-dissociate variable $X_{i;j}$ in clauses of the form $(X_{i;1} \vee Y_k)(\overline{X_{i;2}} \vee \overline{Y_k})(X_{i;3} \vee Y_l)(\overline{X_{i;4}} \vee \overline{Y_l}) \ldots (X_{i;j} \vee Y_k)(\overline{X_{i;j+1}} \vee \overline{Y_k})$ s.t. $Y_k, Y_l \in Y$. Given some $Y_k, Y_l \in Y$ s.t. $k \neq l$. Let a_1 and a_2 be 0/1 assignments to Y_k and Y_l respectively s.t. $a_1 \neq a_2$. If $a_1 = 0$, then we have the term

$$\prod_{j \in T^+} p_{i;j} \prod_{j \in T^-} \overline{p_{i;j}}.$$

By the definition of subsumed inequality, the lower bound condition are either

1. $\forall j \in (S^+ \cup T^+) : p_{i;j} = 0$
2. $\forall j \in (S^- \cup T^-) : \overline{p_{i;j}} = 0$

- **Case**: $X_i = 1$

  Let $|T^-| = |T^+| = 1$.

  Let $|S^+| = |S^+| = 0$.

  Non-dissociate: Clearly, from valuation analysis, we have the term p_i.

  Dissociate: Clearly, from valuation analysis, we have the term $p_{i;j}$.

  Let $|S^+| > 0$ and $|S^-| > 0$.

Non-dissociate:

Let Y_k be a positive two-occurrence neighbor of non-dissociate variable X_i. Let $Y_k = 0$. Setting $X_i = 1$ satisfies $C(X_i)$. Then we have the term p_i.

Dissociate:

Let Y' be the set of single-occurrence neighbors for dissociated variable $X_{i;j}$ s.t. $j \in S^+$. Let Y'' be the set of single-occurrence neighbors for dissociated variable $X_{i;j}$ s.t. $j \in S^-$. Given the positive two-occurrence neighbor Y_k of dissociate variable $X_{i;j}$, let $Y_k = 0$. If $\forall\, Y_l \in Y', Y_l = 0$, then setting $X_{i;j} = 1, \forall j \in S^+$ satisfies $C(X_i)$ (let P be the set of jth indices for these parameters). $\forall j \in S^-, \overline{X_{i;j}}$ are don't cares because $\forall\, Y_l \in Y'', Y_l = 1$, otherwise it would be contradiction. The possible combinations for P is given by

$$\binom{|P|}{b} \quad \text{, for } b = 0 \ldots |P|. \tag{6.2}$$

Let P' be the set of the possible combination of P given by (6.2). Then we have the terms

$$\{p_{i;a} \ : \ T^+ = \{a\}\} \cup \left\{ \prod_{j \in P''} p_{i;j} \ : \ \forall\, P'' \in P' \right\}.$$

By the definition of subsumed inequality, combined with the case $X_i = \bot$, the lower bound condition is

$$(p_{i;a} \leq p_i) \wedge (\forall j \in S^- : \overline{p_{i;j}} = 0).$$

Let $|T^-| > 1$ and $|T^+| > 1$.

Let $|S^+| \geq 0$ and $|S^-| \geq 0$.

Non-dissociate:

Let Y' be the set of positive two-occurrence and Y'' be the set of single-occurrence neighbors of non-dissociated variable X_i. Let $\forall\, Y_k \in Y', Y_k = 0$ and $\forall\, Y_l \in Y'', Y_l = 1$. Setting $X_i = 1$ satisfies $C(X_i)$. Then we have the term p_i.

Dissociate:

Let Y' be the set of two-occurrence neighbors for dissociated variable $X_{i;j}$ s.t. $j \in T^+$. Let Y'' be the set of two-occurrence neighbors for dissociated variable $X_{i;j}$ s.t. $j \in T^-$. Let Y''' be the set of single-occurrence neighbors for dissociated variable $X_{i;j}$ s.t. $j \in S^+$. Let Y'''' be the set of single-occurrence neighbors for dissociated variable $X_{i;j}$ s.t. $j \in S^-$. Let $\forall\, Y_k \in Y', Y_k = 0$ and $\forall\, Y_l \in Y'''', Y_l = 1$. Note that for some $Y_k \in Y'$ and $Y_l \in Y''$, setting $Y_k = a_1$ and $Y_l = a_2$ s.t. $a_1 \neq a_2$ results in a contradiction. To satisfy $C(X_i)$, first set $X_{i;j} = 1, \forall\, j \in T^+$. This results in $\forall j \in S^-, \overline{X_{i;j}}$ and $\forall j \in T^-, \overline{X_{i;j}}$ are don't cares. If $\forall\, Y_k \in Y''', Y_k = 0$, then setting $X_{i;j} = 1, \forall j \in S^+$ satisfies $C(X_i)$ (let P be the set of jth indices for these parameters). The possible combinations for P is given by

$$\binom{|P|}{b} \quad \text{, for } b = 0 \ldots |P|. \tag{6.3}$$

Let P' be the set of the possible combination of P given by (6.3). Then we have the terms

$$\left\{ \prod_{j \in T^+} p_{i;j} \right\} \cup \left\{ \prod_{k \in P''} p_{i;k} \;:\; \forall\, P'' \in P' \right\}.$$

By the definition of subsumed inequality, combined with the case $X_i = \bot$, the lower bound condition is

$$\left(\prod_{j \in T^+} p_{i;j} \leq p_i \right) \wedge \left(\forall j \in (S^- \cup T^-) : \overline{p_{i;j}} = 0 \right).$$

The upper bound condition for this case is

$$\prod_{j \in (S^+ \cup T^+)} p_{i;j} \geq p_i.$$

- **Case**: $X_i = 0$

  Let $|T^-| = |T^+| = 1$.

Let $|S^+| = |S^+| = 0$.

Non-dissociate: Clearly, from valuation analysis, we have the term $\overline{p_i}$.

Dissociate: Clearly, from valuation analysis, we have the term $\overline{p_{i;j}}$.

Let $|S^+| > 0$ and $|S^-| > 0$.

Non-dissociate:

Let Y_k be a negative two-occurrence neighbor of non-dissociated variable $\overline{X_i}$ and Y' be the set of single-occurrence neighbors of positive appearing non-dissociated variable X_i. Let $Y_k = 1$. Setting $X_i = 0$ and $\forall\, Y_l \in Y', Y = 1$ satisfies $C(X_i)$. Then we have the term $\overline{p_i}$.

Dissociate:

Let Y' be the set of single-occurrence neighbors for dissociated variable $X_{i;j}$ s.t. $j \in S^+$. Let Y'' be the set of single-occurrence neighbors for dissociated variable $X_{i;j}$ s.t. $j \in S^-$. Let Y_k be the negative two-occurrence neighbor of dissociate variable $\overline{X_{i;j}}$, s.t. $Y_k = 1$. If $\forall\, Y_l \in Y', Y_l = 1$, then $\forall j \in S^+$, $X_{i;j}$ are don't cares. If $\forall\, Y_l \in Y', Y_l = 1$, then $\forall j \in S^-$, $\overline{X_{i;j}}$ are don't cares. The possible combination of setting $Y_l = 0$ s.t. $Y_l \in P''$ is given by

$$\binom{|S^-|}{b} \quad \text{, for } b = 0 \ldots |S^-|. \tag{6.4}$$

Let P be the set of the possible combination of S^- given by (6.4). Then we have the terms

$$\{\overline{p_{i;a}} \ : \ T^- = \{a\}\} \cup \left\{ \prod_{j \in P'} \overline{p_{i;j}} \ : \ \forall\, P' \in P \right\}.$$

By the definition of subsumed inequality, combined with the case $X_i = \bot$, the lower bound condition is

$$(\overline{p_{i;a}} \leq \overline{p_i}) \wedge (\forall j \in S^+ : p_{i;j} = 0).$$

Let $|T^-| > 1$ and $|T^+| > 1$.

Let $|S^+| \geq 0$ and $|S^-| \geq 0$.

Non-dissociate:

Let Y' be the set of negative two-occurrence neighbor of non-dissociated variable $\overline{X_i}$ and Y'' be the set of single-occurrence neighbors of positive appearing non-dissociated variable X_i. Let $\forall\, Y_k \in Y', Y_k = 0$ and $\forall\, Y_l \in Y'', Y_l = 1$. Setting $X_i = 0$ satisfies $C(X_i)$. Then we have the term $\overline{p_i}$.

Dissociate:

Let Y' be the set of single-occurrence neighbors for dissociated variable $X_{i;j}$ s.t. $j \in S^+$. Let Y'' be the set of single-occurrence neighbors for dissociated variable $X_{i;j}$ s.t. $j \in S^-$. Let Y''' be the set of two-occurrence negative appearing neighbors for dissociated variable $X_{i;j}$ s.t. $j \in T^-$. Let $\forall\, Y_l \in P''', Y_l = 1$. If $\forall\, Y_l \in Y', Y_l = 1$, then $\forall j \in S^+$, $X_{i;j}$ are don't cares. If $\forall\, Y_l \in Y', Y_l = 1$, then $\forall j \in S^-$, $\overline{X_{i;j}}$ are don't cares. The possible combination of setting $Y_l = 0$ s.t. $Y_l \in P''$ is given by

$$\binom{|S^-|}{b} \quad \text{, for } b = 0 \ldots |S^-|. \tag{6.5}$$

Let P be the set of the possible combination of S^- given by (6.5). Then we have the terms

$$\left\{ \prod_{j \in T^-} \overline{p_{i;j}} \right\} \cup \left\{ \prod_{k \in P'} \overline{p_{i;k}} \;:\; \forall\, P' \in P \right\}.$$

By the definition of subsumed inequality, combined with the case $X_i = \perp$, the lower bound condition is

$$\left(\prod_{j \in T^-} \overline{p_{i;j}} \leq \overline{p_i} \right) \wedge \left(\forall j \in (S^+ \cup T^+) : p_{i;j} = 0 \right).$$

The upper bound condition for this case is

$$\prod_{j \in (S^- \cup T^-)} \overline{p_{i;j}} \geq \overline{p_i}.$$

Table 6.4. Dissociation for Example 6.16. $\perp$ denotes contradiction (i.e., formula cannot be satisfied). $*$ denotes the don't care condition.

Y_3	Y_4	Y_5	Y_6	X_1	$X_{1;1}$	$X_{1;2}$	$X_{1;3}$	$X_{1;4}$	$X_{1;5}$	ϕ_1	$\phi_{1;1}, \phi_{1;2}, \phi_{1;3}, \phi_{1;4}, \phi_{1;5}$
0	0	0	0	$\perp$	1	1	$*$	1	1	0	$p_{1;1}p_{1;2}p_{1;4}p_{1;5}$
0	0	0	1	1	1	1	$*$	1	$*$	p_i	$p_{1;1}p_{1;2}p_{1;4}$
0	0	1	0	$\perp$	1	1	$*$	$*$	1	0	$p_{1;1}p_{1;2}p_{1;5}$
0	0	1	1	1	1	1	$*$	$*$	$*$	p_i	$p_{1;1}p_{1;2}$
0	1	0	0	$\perp$	1	$*$	0	1	1	0	$p_{1;1}\overline{p_{1;3}}p_{1;4}p_{1;5}$
0	1	0	1	$\perp$	1	$*$	0	1	$*$	0	$p_{1;1}\overline{p_{1;3}}p_{1;4}$
0	1	1	0	$\perp$	1	$*$	0	$*$	1	0	$p_{1;1}\overline{p_{1;3}}p_{1;5}$
0	1	1	1	$\perp$	1	$*$	0	$*$	$*$	0	$p_{1;1}\overline{p_{1;3}}$
1	0	0	0	$\perp$	$*$	1	$*$	1	1	0	$p_{1;2}p_{1;4}p_{1;5}$
1	0	0	1	1	$*$	1	$*$	1	$*$	p_i	$p_{1;2}p_{1;4}$
1	0	1	0	$\perp$	$*$	1	$*$	$*$	1	0	$p_{1;2}p_{1;5}$
1	0	1	1	1	$*$	1	$*$	$*$	$*$	p_i	$p_{1;2}$
1	1	0	0	$\perp$	$*$	$*$	0	1	1	0	$\overline{p_{1;3}}p_{1;4}p_{1;5}$
1	1	0	1	$\perp$	$*$	$*$	0	1	$*$	0	$\overline{p_{1;3}}p_{1;4}$
1	1	1	0	0	$*$	$*$	0	$*$	0	$\overline{p_i}$	$\overline{p_{1;3}}\overline{p_{1;5}}$
1	1	1	1	0	$*$	$*$	0	$*$	$*$	$\overline{p_i}$	$\overline{p_{1;3}}$

Combining the three cases gives the conditions for the oblivious bounds for X_i. □

Example 6.16. *Consider* $C(X_1') = \{(X_{1;1} \vee Y_3), (X_{1;2} \vee Y_4), (\overline{X_{1;3}} \vee \overline{Y_4}), (X_{1;4} \vee Y_5)(\overline{X_{1;5}} \vee Y_6)\}$. *Theorem 6.15 gives the following conditions for upper and lower oblivious bounds:*

- *U:* $\Big(p_{1;1}p_{1;2}p_{1;4} \geq p_1\Big) \wedge \Big(\overline{p_{1;3}}\overline{p_{1;5}} \geq \overline{p_1}\Big)$
- *L: Either of the following three conditions hold:*
  1. $(p_{1;2} \leq p_1) \wedge (\overline{p_{1;3}} = \overline{p_{1;5}} = 0)$
  2. $(\overline{p_{1;3}} \leq \overline{p_1}) \wedge (p_{1;1} = p_{1;2} = p_{1;4} = 0)$
  3. $(p_{1;2} \leq p_1) \wedge (\overline{p_{1;3}} \leq \overline{p_1}) \wedge (p_{1;1} = p_{1;4} = \overline{p_{1;5}} = 0)$

Table 6.5. Summary of oblivious bound conditions. T: whether $C(X_i)$ has two-occurrence neighbors, S^+ and S^-: whether $C(X_i)$ has single-occurrence neighbors which appear in clauses $(X_i \vee Y_j)$ and $(\overline{X_i} \vee Y_j)$ respectively. An entry in a cell means that neighbors of the respective types are either present ($\surd$), absent ($\times$), or either present or absent ($*$). Bold text in Case and Solution columns denote novel contributions of this dissertation while normal font text indicates previous work.

S^+	S^-	T	Case	Solution
$\surd$	$\times$	$\times$	Monotone	Theorems 6.8 & **6.13**
$\times$	$\surd$	$\times$		
$\surd$	$\surd$	$\times$	**Single-occurrence**	**Theorem 6.13**
$*$	$*$	$\surd$	**Two-occurrence**	**Theorem 6.15**

The valuation analysis is shown in Table 6.4.

Table 6.5 summarizes the oblivious bound conditions. Theorems 6.13 and 6.15 yield the algorithm given in Algorithm 6.1 for bounding the partition function of a given W2CNF.

6.4 Experiments

We evaluated the performance of DIS (see Algorithm 6.1) and compared it with MB on generated synthetic datasets and benchmark datasets from the UAI 2008 probabilistic inference competition repository (http://graphmod.ics.uci.edu/uai08) for the task of computing upper and lower bounds on the weighted model count (or partition function). All experiment were conducted on quad-core Intel i7 based machines with 24GB RAM running Ubuntu.

6.4.1 Synthetic Datasets

We generated non-monotone W2CNF formulas encoded as $m \times m$ grid structure graphical models parameterized by univariate and pairwise binary potentials. We then compared error bound performance of DIS and MB (*i-bound* $= 1$) from the aspects of (1) varying grid sizes

Algorithm 6.1: (DIS) Dissociation Bounds for WMC

Input: W2CNF $\mathcal{F} = \langle \mathbf{X}, \Phi, \mathbf{C} \rangle$,
Variable ordering $o = [X_1, X_2, \ldots, X_{|\mathbf{X}|}]$
Output: Lower (or upper) bound on the WMC
begin
  Initialize: $Z_B \leftarrow 1$ (Bound on the partition function)
  for $i \leftarrow 1$ *to* $|\mathbf{X}|$ **do**
    Convert $\mathcal{F}$ to a minimal $\mathcal{F}$
    Convert $C(X_i)$ to canonical form
    if $\mathbf{C}$ *is inconsistent* **then**
      return 0
    else if $C(X_i) = \{X_i\}$ **then**
      $Z_B \leftarrow Z_B \times p_i$
    else if $C(X_i) = \{\overline{X_i}\}$ **then**
      $Z_B \leftarrow Z_B \times \overline{p_i}$
    else if $C(X_i)$ *has two-occurrence neighbors* **then**
      Update Z_B using Theorem 6.15
    else if $C(X_i)$ *has single-occurrence neighbors* **then**
      Update Z_B using Theorem 6.13
  return Z_B
end

under random weight function settings; and (2) varying weight function settings according to determinism strength under fixed grid sizes. For each model, we computed the true weighted model count Z^*. We then compared each algorithm's approximated bound Z^{algo} and calculated the error bound as $\log(Z^*/Z^{\text{algo}})$ for the lower bound and the same negated for the upper bound. A lower error bound value is better. For each setting, we generated 50 random problem instances and ran DIS and MB 100 times for each instance. From the 100 solutions, we selected the best, namely either the lowest upper bound or the highest lower bound. We then computed the average error bound across the 50 problem instances.

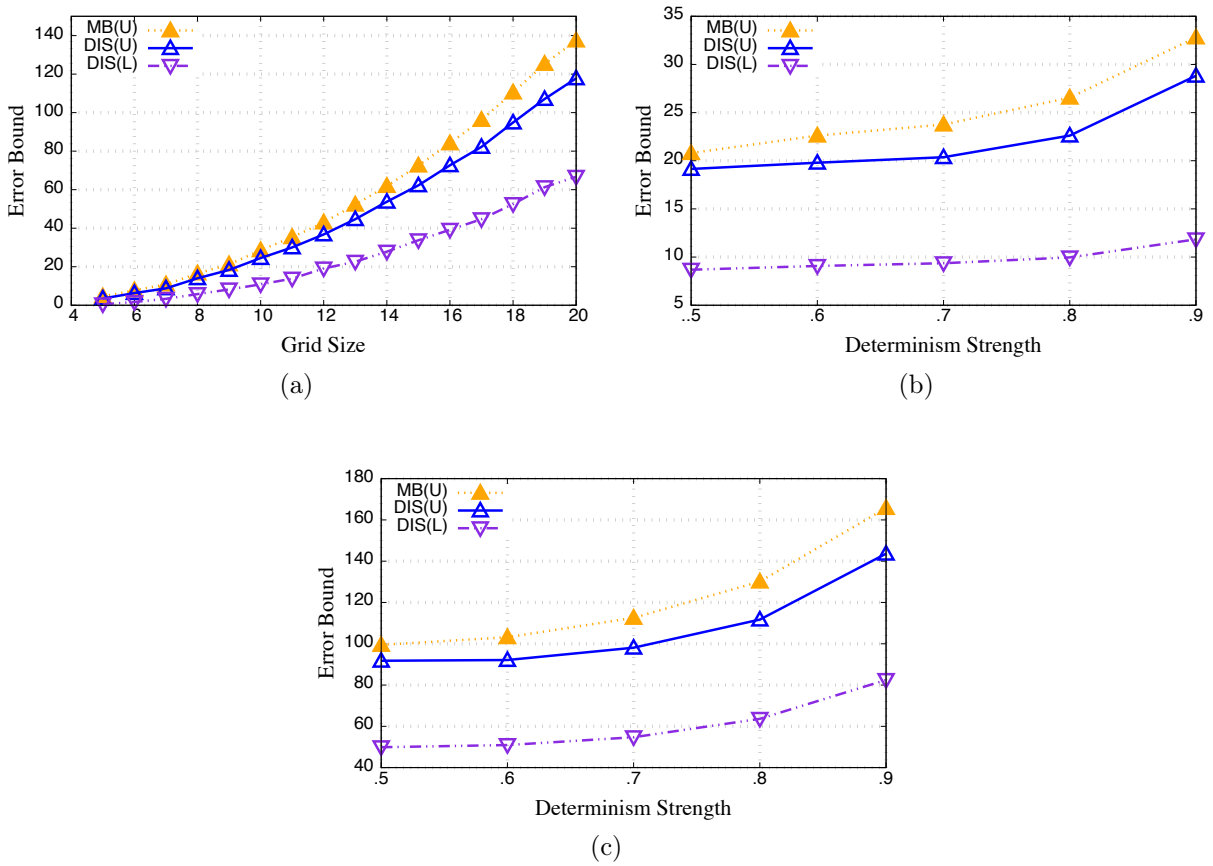


Figure 6.2. Upper bound estimates for dissociation DIS(U) and mini-bucket MB(U), and lower bound estimates for dissociation DIS(L). Error bound by varying (a) grid size (b) level of determinism for 10×10 grid (c) 20×20 grid. Lower value is better.

Grid size. We generated $m \times m$ grids using values of $m = \{5, 6, 7, \ldots, 20\}$. For the weight function values, we sampled from an uniform $U(0, 1)$ distribution. We also uniformly generated the clauses. The results are shown in Figure 6.2 (a). For the upper bound, DIS noticeably begins to outperform MB starting at around grid size 10×10 and the performance gap widens as the grid size increases. Since MB utilizes the max function, it has a higher tendency to overestimate the upper bound. This was accomplished only by setting the weight function values to the k-th root (e.g., $p_{X_{1;1}} = p_{X_{1;2}} = \sqrt{p_{X_1}}$ for $|C(X_1)| = 2$). We would expect the performance gap to be wider, favoring DIS, by optimizing the inequalities. For

Table 6.6. The log relative upper bound between dissociation DIS(U) and mini-bucket MB(U) on UAI 2008 repository problem instances. Lower value is better for DIS.

Instance	$\log \frac{Z^{\mathrm{DIS}(U)}}{Z^{\mathrm{MB}(U)}}$	Instance	$\log \frac{Z^{\mathrm{DIS}(U)}}{Z^{\mathrm{MB}(U)}}$
sg2-17	−277.8	orc111	−87.6
sg7-11	−293.4	orc175	−96.3
sg8-18	−281.9	orc180	−124.4
sg9-24	−292.8	orc203	−111.0
sg17-4	−303.3	orc218	−4.4
smk10	−50.9	orc62	−393.4
smk20	−165.9	orc154	−97.0
orc42	−119.6	orc225	−137.3
orc45	−261.1	orc139	−155.0

the lower bound, MB produced 0 for all problems and thus was not plotted. MB has a high tendency to converge to the so called degenerate solution (i.e., 0) due to the min function. The lower bound for DIS is tighter, as compared to the upper bounds, since the settings to the lower bound inequalities do not need to be optimized.

Determinism strength. We analyzed the performance of DIS and MB according to various levels of determinism, namely the distance from uniform .5 (unweighted) towards 0 and 1. To accomplish this, we set all weight functions to the same value $p_X \in \{.5, .6, .7, .8, .9\}$. The results are shown in Figure 6.2 (b) and (c). For the lower bound, MB produced 0 for all problems and thus was not plotted. The overall relative performance comparison is similar to that of varying grid size. Again, the lower bound performance for DIS is tighter and all bounds had higher bound error as the determinism strength increased. Intuitively, as the gap between p_{X_i} and $\overline{p_{X_i}}$ widens, the tendency to overestimate (underestimate) the upper (lower) bound increases.

Table 6.7. The log relative lower bound between ground truth and dissociation DIS(L) on UAI 2008 repository problem instances. Lower value is better for DIS.

Instance	$\log \frac{Z^*}{Z^{\mathrm{DIS}(L)}}$	Instance	$\log \frac{Z^*}{Z^{\mathrm{DIS}(L)}}$
sg2-17	732.4	orc111	209.8
sg7-11	759.4	orc175	342.6
sg8-18	727.3	orc180	375.0
sg9-24	774.5	orc203	346.8
sg17-4	752.1	orc218	18.2
smk10	191.3	orc62	–
smk20	799.8	orc154	354.7
orc42	407.9	orc225	499.7
orc45	747.8	orc139	576.6

6.4.2 UAI Inference Datasets

We also compared DIS to MB on the segmentation (sg), promedas (orc) and smokers (smk) dataset from the UAI 2008 repository. The variables in the models are binary and the number of variables range from ∼100 to 1000. We converted the non-pairwise models to pairwise models and then encoded them as W2CNF. We used *i-bound* = 1 for MB. We ran DIS and MB 100 times and similarly, we selected the best. For the upper bound, we evaluated using the log relative upper bound, namely $\log(Z^{DIS(U)}/Z^{MB(U)})$. Lower value is better for DIS. The results are shown in Table 6.6. DIS outperforms MB by a wide margin on the majority of the datasets. The solution quality of DIS for sg was quite consistent while for orc it had higher variance. For the lower bound, we evaluated dissociation's lower bound against the ground truth, namely $\log(Z^*/Z^{DIS(L)})$. MB produced 0 for all problems and thus was not shown. The results are shown in Table 6.7 (orc62 was not tractable).

In summary, DIS performs consistently better than MB on harder WMC problems. In particular, the lower bounds output by DIS are always better than MB.

6.5 Discussion

We proposed an approximate, oblivious bounding scheme for WMC, extending the idea of dissociation to non-monotone formulas and exploiting logical structure. Dissociation yields a novel set of inequalities for which upper and lower bounds can be derived efficiently. Empirically, we showed that our method outperforms mini-buckets—a popular oblivious bounding scheme—on various datasets. The lower bounds are robust since they do not require optimization (in the monotone case). For upper bounds, we utilized naïve settings, namely the k-th root applied to the parameter of a dissociated variable.

CHAPTER 7

CONCLUSION

In this chapter, we conclude by providing an overview of our contributions and listing future directions for research.

7.1 Contributions

The goals of this dissertation have been to utilize the idea of parameter tying to improve generalization of learning and efficiency of inference for graphical models. At a high-level, we leverage parameter tying to:

- **Learning**. Develop novel alternative regularization schemes that are automatic (cf. a priori) and can be integrated with standard learning objectives for graphical models to achieve higher generalization accuracy (i.e., likelihood) compared to existing regularization techniques. For Bayesian networks, the regularization for the learning procedure follows a greedy post-processing approach that is fast and efficient. On the other hand, with Markov networks, the regularization for the learning procedure is iterative and penalty-based that facilitates both soft and hard parameter tying.

- **Inference**. Develop inference algorithms that can exploit the parameter structure, in the form of symmetries, that are the result of learning a parameter tied graphical model, to achieve efficient inference. To this we present a sampling-based algorithm which explores the sample space in a more efficient manner, yielding more accurate results for probabilistic queries. In addition, we develop bounded inference schemes for graphical models, which we refer to as dissociation-based bounds. We propose a deterministic bounding scheme for weighted model counting, which include computing the partition function and probability of evidence in graphical models as special cases.

Our approach is partition-based (Dechter and Rish, 2003) and gives rise to a novel class of inequalities for which upper (or lower) bounds can be efficiently derived. The bounds are oblivious, that is they require only limited observation of parameterization to the problem, which yields faster methods.

7.2 Directions for Future Research

Our research offers ample room for additional improvements for continued investigation in the future.

7.2.1 Towards Canonical Parameter Tied GMs

In Chapter 4 we stated Markov networks are not identifiable in general. For parameter learning, there exists a continuum of parameterizations for a given structure and probability distribution. We can use finer-grained representations such as feature-based (log-linear) models, however, Markov networks remain generally overparameterized. Although the soft-tying of APT addresses the overparameterization issue to a certain extent, the parameter tying procedure (i.e., quantization function) can be further improved.

The Hammersley-Clifford theorem (Besag, 1974), which states the equivalence between the Markov properties and the Gibbs distribution, gives rise to the notion of a canonical parameterization. The canonical parameterization is defined by a set of energy functions over all non-empty cliques and the parameters of the functions are defined relative to a arbitrary fixed assignment. While the canonical parameterization eliminates the issue of identifiability (i.e., ambiguity), it gives rise to two main issues. First, the representation is generally computationally intractable due to computing the probability of the fixed assignment (i.e., partition function). Second, the canonical parameterization generally results in extremely sparse feature representation, which renders the model uninterpretable.

For future work, we aim to improve the parameter tying procedure to achieve a canonical (or pseudo-canonical) parameter tied GM that also overcomes the issues of tractability and interpretability. Since the parameters (weights) of a Markov network describe the *affinity* between random variables and thus do not have a probabilistic interpretation, we will focus on parameter tying that more aligns with the (conditional) probabilities encoded in the model. To achieve this, we consider and investigate alternative GMs such as dependency networks (DNs) (Heckerman et al., 2000). DNs are constructed by learning the conditional probabilities for each variable in the model. The advantage of this approach is all the conditional probabilities can be learned independently (i.e., Markov blanket) and thus computationally tractable. DNs are similar to Bayesian networks, which are more interpretable. However, an issue with DNs is that the conditional probabilities will not be consistent with the joint distribution. Recently, (Lowd, 2012) proposed heuristics based on model averaging to seek consistent distributions. We will also consider the issue of inconsistency and seek more principle methods which incorporate parameter tying.

7.2.2 Beyond k-means for Parameter Tying

For this dissertation, we have focused on using k-means as the quantization function for parameter tying. By framing the parameters as an one dimensional vector, we were able to perform the quantization efficiently by leveraging a k-means algorithm with polynomial complexity (Wang and Song, 2011). There exists other mapping functions to tie parameters such as using coding theory or hashing. Autoencoders are neural network architectures that allow for coding of data in an unsupervised manner. For future work, we are interested in leveraging an autoencoder's coding ability as the quantization mechanism for parameter tying. Although the parameter tying will be done in a *latent* space, this approach allows for more efficient tying of larger number of parameters. We will first apply this method on learning the parameters of a dependency network.

7.2.3 Improving Dissociation-Based Bounds

For future work, we are interested in obtaining better (tighter) upper and lower bounds. To do so, we can leverage four powerful complementary techniques described in literature (cf. (Liu and Ihler, 2014; Lam et al., 2014; Ihler et al., 2012; Ping et al., 2015; Gogate and Domingos, 2011, 2013)): cost-shifting (or re-parameterization), higher *i-bound*, quantization and Hölder's inequality. For instance, applying Hölder's inequality to our running example (see Example 6.2) gives the optimization problem $\min_{\omega}(p_{X_1}^{\omega} + (\overline{p_{X_1}}p_{Y_2})^{\omega})^{1/\omega}(1 + p_{Y_3}^{(1-\omega)})^{(1/1-\omega)}$ such that $0 \leq \omega \leq 1$. We can also apply Hölder's inequality to dissociation which alternatively gives us the optimization problem $\min_{p_{X_{1;1}}, p_{X_{1;2}}, \omega}(p_{X_{1;1}}^{\omega} + (\overline{p_{X_{1;1}}}p_{Y_2})^{\omega})^{1/\omega}(p_{X_{1;2}}^{(1-\omega)} + (\overline{p_{X_{1;2}}}p_{Y_3})^{(1-\omega)})^{(1/1-\omega)}$ such that $p_{X_{1;1}}p_{X_{1;2}} = p_{X_1}$ and $0 \leq \omega \leq 1$. We are particularly interested in developing algorithms to optimize the latter problem and to determine which formulation will consistently yield tighter upper and lower bounds. Another line of future work is investigating the utility of our approach when applied to other inference tasks such as maximum a posteriori (MAP) estimation and marginal maximum a posteriori (MMAP) estimation.

REFERENCES

Abbeel, P., D. Koller, and A. Y. Ng (2006). Learning factor graphs in polynomial time and sample complexity. *Journal of Machine Learning Research 7*, 1743–1788.

Andrew, G. and J. Gao (2007). Scalable training of L_1-regularized log-linear models. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, pp. 33–40.

Bach, S. H., M. Broecheler, B. Huang, and L. Getoor (2017). Hinge-loss Markov random fields and probabilistic soft logic. *Journal of Machine Learning Research 18*, 3846–3912.

Bertsekas, D. P. and J. N. Tsitsiklis (2008). *Introduction to Probability*. Athena Scientific.

Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B. 36*(2), 192–236.

Besag, J. (1975). Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society. Series D. 24*(3), 179–195.

Bottou, L. and Y. Bengio (1994). Convergence properties of the k-means algorithms. In *Proceedings of the Seventh International Conference on Neural Information Processing Systems*, pp. 585–592.

Bradley, J. K. and C. Guestrin (2012). Sample complexity of composite likelihood. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, pp. 136–160.

Brooks, S., A. Gelman, G. L. Jones, and X. Meng (2011). *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC.

Chavira, M. and A. Darwiche (2008). On probabilistic inference by weighted model counting. *Artificial Intelligence 172*, 772–799.

Cheng, J. and M. J. Druzdzel (2001). Confidence inference in Bayesian networks. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pp. 75–82.

Choi, A., M. Chavira, and A. Darwiche (2007). Node splitting: A scheme for generating upper bounds in Bayesian networks. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, pp. 57–66.

Choi, A. and A. Darwiche (2009). Relax then compensate: On max-product belief propagation and more. In *Proceedings of the Twenty-Second International Conference on Neural Information Processing Systems*, pp. 351–359.

Chou, L., W. Gatterbauer, and V. Gogate (2018). Dissociation-based oblivious bounds for weighted model counting. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, pp. 866–875.

Chou, L., P. Sahoo, S. Sarkhel, N. Ruozzi, and V. Gogate (2018). Automatic parameter tying: A new approach for regularized parameter learning in Markov networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 2860–2867.

Chou, L., S. Sarkhel, N. Ruozzi, and V. Gogate (2016). On parameter tying by quantization. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 3241–3247.

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pp. 151–158.

Darwiche, A. (2009). *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

Davis, J. and P. Domingos (2010). Bottom-up learning of Markov network structure. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, pp. 271–278.

Dechter, R. (1996). Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pp. 211–219.

Dechter, R. and I. Rish (2003). Mini-buckets: A general scheme for bounded inference. *Journal of the ACM 50*, 107–153.

den Heuvel, M. V., W. Gatterbauer, M. Theobald, and F. Geerts (2018). A general framework for anytime approximation in probabilistic databases. In *Eighth International Workshop on Statistical Relational AI*.

Domingos, P. and D. Lowd (2009). *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan and Claypool.

Fung, R. M. and K. Chang (1989). Weighing and integrating evidence for stochastic simulation in Bayesian networks. In *Proceedings of the Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 209–220.

Gatterbauer, W. and D. Suciu (2014). Oblivious bounds on the probability of Boolean functions. *ACM Transactions on Database Systems 39*(5), 1–34.

Gatterbauer, W. and D. Suciu (2017). Dissociation and propagation for approximate lifted inference with standard relational database management systems. *The International Journal on Very Large Data Bases 26*, 5–30.

Geman, S. and D. Geman (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 6*, 721–741.

Getoor, L. and B. Taskar (2007). *Introduction to Statistical Relational Learning*. MIT Press.

Gionis, A., P. Indyk, and R. Motwani (1999). Similarity search in high dimensions via hashing. In *Proceedings of the Twenty-Fifth International Conference on Very Large Data Bases*, pp. 518–529.

Gogate, V. (2009). *Sampling Algorithms for Probabilistic Graphical Models with Determinism*. Ph. D. thesis, University of California, Irvine.

Gogate, V. and R. Dechter (2011). SampleSearch: Importance sampling in presence of determinism. *Artificial Intelligence 175*, 694–729.

Gogate, V. and P. Domingos (2010). Formula-based probabilistic inference. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pp. 210–219.

Gogate, V. and P. Domingos (2011). Approximation by quantization. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pp. 247–255.

Gogate, V. and P. Domingos (2013). Structured message passing. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pp. 252–261.

Gogate, V. and P. Domingos (2016). Probabilistic theorem proving. *Communications of the ACM 59*, 107–115.

Hammersley, J. M. and D. C. Handscomb (1964). *Monte Carlo Methods*. Methuen & Co.

Han, S., H. Mao, and W. J. Dally (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *Fourth International Conference on Learning Representation*.

Heckerman, D., D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie (2000). Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research 1*, 49–75.

Henrion, M. (1986). Propagation of uncertainty in Bayesian networks by probabilistic logic sampling. In *Proceedings of the Second Conference on Uncertainty in Artificial Intelligence*, pp. 149–163.

Ihler, A., N. Flerova, R. Dechter, and L. Otten (2012). Join-graph based cost-shifting schemes. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pp. 397–406.

Indyk, P. and R. Motwani (1998). Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pp. 604–613.

Jha, A., V. Gogate, A. Meliou, and D. Suciu (2010). Lifted inference from the other side: The tractable features. In *Proceedings of the Twenty-Fourth International Conference on Neural Information Processing Systems*, pp. 973–981.

Kok, S., M. Sumner, M. Richardson, P. Singla, H. Poon, and P. Domingos (2006). The Alchemy system for statistical relational AI. Technical report, Department of Computer Science and Engineering, University of Washington. http://alchemy.cs.washington.edu.

Kokolakis, G. E. and P. Nanopoulos (2001). Bayesian multivariate micro-aggregation under the Hellinger's distance criterion. *Research in Official Statistics 4*(1), 117–126.

Koller, D. and N. Friedman (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Lam, W., K. Kask, R. Dechter, and A. Ihler (2014). Beyond static mini-bucket: Towards integrating with iterative cost-shifting based dynamic heuristics. In *Proceedings of the Seventh Annual Symposium of Combinatorial Search*, pp. 105–113.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 2278–2324.

Levin, L. A. (1973). Universal sequential search problems (in Russian). *Problemy Peredachi Informatsii 9*, 115–116.

Liu, J. and D. Page (2013). Bayesian estimation of latently-grouped parameters in undirected graphical models. In *Proceedings of the Twenty-Sixth International Conference on Neural Information Processing Systems*, pp. 1232–1240.

Liu, J. S. (2004). *Monte Carlo Strategies in Scientific Computing*. Springer.

Liu, Q. (2014). *Reasoning and Decisions in Probabilistic Graphical Models – A Unified Framework*. Ph. D. thesis, University of California, Irvine.

Liu, Q. and A. Ihler (2014). Bounding the partition function using Hölder's inquality. In *Proceedings of the Twenty-Eighth International Conference on Machine Learning*, pp. 849–856.

Lowd, D. (2012). Closed-form learning of Markov networks from dependency networks. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pp. 533–542.

Marshall, A. W. (1956). The use of multistage sampling schemes in Monte Carlo computations. In M. Meyer (Ed.), *Symposium on Monte Carlo Methods*, pp. 123–140. Wiley, New York.

Metropolis, N. and S. Ulam (1949). The Monte Carlo method. *Journal of the American Statistical Association 44* (247), 335–341.

Milch, B., L. S. Zettlemoyer, K. Kersting, M. Haime, and L. P. Kaelbling (2008). Lifted probabilistic inference with counting formulas. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pp. 1062–1068.

Neal, R. M. (2003). Slice sampling. *The Annals of Statistics 31* (3), 705–767.

Ng, A. Y. (2004). Feature selection, L_1 vs. L_2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pp. 78–85.

Nowlan, S. J. and G. E. Hinton (1991). Adaptive soft weight tying using Gaussian mixtures. In *Proceedings of the Fourth International Conference on Neural Information Processing Systems*, pp. 993–1000.

Ortiz, L. E. and L. P. Kaelbling (2000). Adaptive importance sampling for estimation in structured domains. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pp. 446–454.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Ping, W., Q. Liu, and A. Ihler (2015). Decomposition bounds for marginal MAP. In *Proceedings of the Twenty-Eighth International Conference on Neural Information Processing Systems*, pp. 3267–3275.

Pollard, D. (1982). Quantization and the method of k-means. *IEEE Transactions on Information Theory IT-28* (2), 199–205.

Poon, H. and P. Domingos (2006). Sound and efficient inference with probabilistic and deterministic dependencies. In *Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence*, pp. 458–463.

Rahman, T. and V. Gogate (2016). Merging strategies for sum-product networks: From trees to graphs. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pp. 617–626.

Ravikumar, P., M. J. Wainwright, and J. D. Lafferty (2010). High-dimensional Ising model selection using ℓ_1-regularized logistic regression. *The Annals of Statistics 38* (3), 1287–1319.

Rooshenas, A. and D. Lowd (2014). Learning sum-product networks with direct and indirect variable interactions. In *Proceedings of the Thirty-First International Conference on Machine Learning*, pp. 710–718.

Sang, T., P. Beame, and H. Kautz (2005). Performing Bayesian inference by weighted model counting. In *Proceedings of the Twentieth AAAI Conference on Artificial Intelligence*, pp. 475–481.

Savicky, P. and J. Vomlel (2009). Triangulation heuristics for BN2O networks. In *Proceedings of the Tenth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pp. 566–577.

Shachter, R. D. and M. A. Peot (1989). Simulation approaches to general probabilistic inference on belief networks. In *Proceedings of the Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 221–230.

Steck, H. and T. S. Jaakkola (2002). On the Dirichlet prior and Bayesian regularization. In *Proceedings of the Fifteenth International Conference on Neural Information Processing Systems*, pp. 713–720.

Valiant, L. G. (1979). The complexity of computing the permanent. *Theoretical Computer Science 8*, 189–201.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM 27*(11), 1134–1142.

Wang, H. and M. Song (2011). Ckmeans.1d.dp: Optimal k-means clustering in one dimension by dynamic programming. *The R Journal 3*, 29–33.

Wang, Z., L. Duan, T. Huang, and W. Gao (2016). Affinity preserving quantization for hashing: A vector quantization approach to learning compact binary codes. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 1102–1108.

Wasserman, L. (2004). *All of Statistics: A Concise Course in Statistical Inference*. Springer.

Wei, W., J. Erenrich, and B. Selman (2004). Towards efficient sampling: Exploiting random walk strategies. In *Proceedings of the Nineteenth AAAI Conference on Artificial Intelligence*, pp. 670–676.

Zhang, N. L. and D. Poole (1994). A simple approach to Bayesian network computations. In *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*, pp. 171–178.

Zhu, H., M. Long, J. Wang, and Y. Cao (2016). Deep hashing network for efficient similarity retrieval. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 2415–2421.

BIOGRAPHICAL SKETCH

Li K. Chou earned his Bachelor of Science in Computer Science and Master of Business Administration from Southern Methodist University. Prior to starting the Computer Science PhD program at The University of Texas at Dallas, Li worked extensively in the financial industry. His research interests broadly encompass the areas of artificial intelligence, machine learning, and data science.

CURRICULUM VITAE

Li K. Chou

August 2019

Contact Information:

Department of Computer Science
The University of Texas at Dallas
800 W. Campbell Rd.
Richardson, TX 75080
Email: li.chou@utdallas.edu
Citizenship: United States

Education:

PhD, Computer Science, The University of Texas at Dallas, 2019
MS, Computer Science, The University of Texas at Dallas, 2018
MBA, Finance and Accounting, Southern Methodist University, 2003
BS, Computer Science and Minor in Mathematics, Southern Methodist University, 1997

Selected Experience:

The University of Texas at Dallas, Richardson, TX
Department of Computer Science
Graduate Research Assistant 5/2014−8/2015, 5/2016−8/2019
Graduate Teaching Assistant 8/2015−5/2016

Adobe Research, San Jose, CA
BigData Experience Lab
Data Science Research Intern Summer 2018

Refereed Conference Publication:

Chou, L., W. Gatterbauer, and V. Gogate (2018). Dissociation-Based Oblivious Bounds for Weighted Model Counting. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, pp. 866–875.

Chou, L., P. Sahoo, S. Sarkhel, N. Ruozzi, and V. Gogate (2018). Automatic Parameter Tying: A New Approach for Regularized Parameter Learning in Markov Networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 2860–2867.

Chou, L., S. Sarkhel, N. Ruozzi, and V. Gogate (2016). On Parameter Tying by Quantization. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 3241–3247.

Teaching Experience:

Guest Lecture:

1. Artificial Intelligence (CS 6364) Fall 2018, *Logical Agents*
2. Machine Learning (CS 6375) Spring 2018, *Naïve Bayes*
3. Advanced Machine Learning (CS 7301) Spring 2017, *Point Estimation*
4. Big Data Management and Analytics (CS 6350) Spring 2016, *Classification*, *Decision Tree*, *Naïve Bayes*
5. Big Data Management and Analytics (CS 6350) Spring 2016, *k-Nearest Neighbors*, *Ensemble Methods*
6. Convex Optimization (CS 8V02) Fall 2015, *Frank-Wolfe Algorithm*
7. Big Data Management and Analytics (CS 6350) Fall 2015, *Classification*, *Decision Tree*, *Naïve Bayes*
8. Big Data Management and Analytics (CS 6350) Fall 2015, *k-Nearest Neighbors*, *Ensemble Methods*

Teaching Assistant:

- Advanced Machine Learning (CS 7301) Spring 2016
- Big Data Management and Analytics (CS 6350) Spring 2016
- Artificial Intelligence (CS 6364) Fall 2015
- Big Data Management and Analytics (CS 6350) Fall 2015

Professional Service:

Organizer: UAI 2019 Workflow Chair

Program Committee: AAAI 2020, IJCAI 2018

Reviewer: JMLR, ICML, UAI, AAAI, IJCAI

Co-organizer: UAI 2014, 2016 Probabilistic Inference Competition/Evaluation