Title

Neurosymbolic Learning and Reasoning for Trustworthy AI

Author

Zeng, Zhe

Publication Date

2024

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Neurosymbolic Learning and Reasoning for Trustworthy AI

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Zhe Zeng

2024

ABSTRACT OF THE DISSERTATION

Neurosymbolic Learning and Reasoning for Trustworthy AI

by

Zhe Zeng
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2024
Professor Guy Van den Broeck, Chair

Along with the ubiquitous applications of Artificial Intelligence (AI), the quest for developing trustworthy AI models intensifies. Deep neural networks, while powerful in learning, fall short in reasoning with domain knowledge and offering robustness guarantees. Neurosymbolic AI bridges this gap by melding the learning capabilities of neural networks and reasoning techniques from symbolic AI, thus building models that behave as intended. This dissertation demonstrates my work that addresses the two fundamental challenges in neurosymbolic AI: 1) enabling differentiable learning of deep neural networks under symbolic constraints and 2) performing scalable and reliable probabilistic reasoning over expressive symbolic constraints. It presents how these neurosymbolic approaches achieve trustworthiness through explainability, uncertainty quantification, and domain-knowledge incorporation. These contributions enable broader applications of neurosymbolic AI in various domains including scientific discoveries.

The dissertation of Zhe Zeng is approved.

Kai-Wei Chang

Cho-Jui Hsieh

Todd D. Millstein

Guy Van den Broeck, Committee Chair

University of California, Los Angeles

2024

To my family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my advisor Guy Van den Broeck. I am incredibly fortunate and honored to have him as my advisor. Throughout my research journey, his wisdom and guidance have made the experience not only productive but also genuinely enjoyable. Guy's insightful suggestions and constructive feedback have been critical in shaping me into an independent researcher. His constant encouragement has helped clear my anxieties and given me the strength to carry on. He has instilled in me a passion for pursuing rigorous, elegant, and unexpected research. I will always miss our spontaneous conversations driven by intellectual curiosity and the inspiring journey we've shared. Thank you for everything!

I also want to extend my heartfelt thanks to my other committee members: Todd Millstein, Cho-Jui Hsieh, and Kai-Wei Chang, as well as Mathias Niepert who is in my oral qualification exam committee. Their feedback has played a crucial role in refining my research. Also, their shared experiences and insights have been instrumental in guiding my career planning.

I am extremely lucky to have wonderful collaborators, labmates, and friends: YooJung, Steven, Tal, Yitao, Chenyu, Antonio, Pasha, Kareem, Paolo, Honghua, Meihua, Anji, William, Nikil, Poorva, Daniel, Ellie, Paolo, Arthur, Yujia, Chi, Andy, Haiying, Tong, Chen, Jieyu (and her two adorable cats), Tao, Kuan-Hao, Liunian, Fan, Zijie, Da and so on. I will miss the lovely StarAI Lab.

My heartfelt appreciation goes out to Yuhang, my partner, best friend, and soulmate, by my side through all of life's ups and downs. With you, even the most challenging moments are less scary, and the joyful ones are filled with even more happiness.

Finally, I would like to express my heartfelt gratitude to my family. To my dad, Zhiqiang, and my mom, Yongyi, your unconditional love and unwavering support have been my greatest sources of strength and comfort. Thank you for always believing in me and standing by my side through every step of this journey. I also want to thank my cousin Lisi. She is the first PhD in the family and a true guiding star of my life who has inspired me to follow a similar path. I am constantly motivated by her wisdom, resilience and perseverance.

VITA

2014–2018 B.E. (Mathematics and Applied Mathematics) Zhejiang University.

2018–2021 M.S. (Computer Science) University of California, Los Angeles.

PUBLICATIONS

Chendi Qian, Andrei Manolache, Kareem Ahmed, **Zhe Zeng**, Guy Van den Broeck, Mathias Niepert, and Christopher Morris. Probabilistically rewired message-passing neural networks. In Proceedings of the Eleventh International Conference on Learning Representations (ICLR), 2024

Zhe Zeng and Guy Van den Broeck. Collapsed inference for Bayesian deep learning. In Advances in Neural Information Processing Systems (NeurIPS), 2023

Vinay Shukla, **Zhe Zeng**, Kareem Ahmed, and Guy Van den Broeck. A unified approach to count-based weakly-supervised learning. In Advances in Neural Information Processing Systems (NeurIPS), 2023

Kareem Ahmed, **Zhe Zeng**, Mathias Niepert, and Guy Van den Broeck. SIMPLE: A gradient estimator for k-subset sampling. In Proceedings of the Eleventh International Conference on Learning Representations (ICLR), 2023

Wenzhe Li, **Zhe Zeng**, Antonio Vergari, and Guy Van den Broeck. Tractable computation of expected kernels. In Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence

(UAI), 2021

Zhe Zeng, Paolo Morettin, Fanqi Yan, Antonio Vergari, and Guy Van den Broeck. Probabilistic inference with algebraic constraints: Theoretical limits and practical approximations. In Advances in Neural Information Processing Systems (NeurIPS), 2020.

Zhe Zeng, Paolo Morettin, Fanqi Yan, Antonio Vergari, and Guy Van den Broeck. Scaling up hybrid probabilistic inference with logical and arithmetic constraints via message passing. In Proceedings of the 37th International Conference on Machine Learning (ICML), 2020

Zhe Zeng and Guy Van den Broeck. Efficient search-based weighted model integration. In Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence (UAI), 2019

CHAPTER 1

Introduction

Artificial Intelligence (AI), particularly deep learning, has become pervasive in our daily lives and safety-critical systems. Despite their profound impact, there are increasing concerns about the reliability and trustworthiness of AI models. Therefore, developing trustworthy AI is a pressing need and of great importance for the sustainable advancement of AI technologies.

Even though *neural networks*, the deep learning models, excel in pattern recognition and scalability, they inherently struggle to provide guarantees, explain their decisions, and incorporate domain knowledge – critical elements for ensuring trustworthiness [LQL23]. On the other hand, there is a branch of AI called *symbolic AI* that allows to define models in a transparent way with high explainability and to express the domain knowledge using constraints. Constraints can take various forms from graph structures to logical, arithmetic, and physical ones. Such constraints provide a high-level abstractions of the real-world knowledge, such as safety regulations and scientific phenomenons. Techniques for reasoning over constraints can provide guarantees that the model decisions are consistent with the knowledge.

Neurosymbolic AI emerges as a promising field that combine the good from both worlds for the next-generation AI models *to enable and support decision-making in the presence of probabilistic uncertainty and symbolic knowledge* [GL23, MDM24]. It has been shown to be powerful in solving challenging tasks that are previously out of the reach of pure deep learning models or pure symbolic models [TWL24]. The fusion of neural network-based learning with symbolic reasoning has the potential to revolutionize not only the capabilities of AI models but also their societal impact and roles in scientific discoveries. However, there are some unique challenges in

neurosymbolic AI when combining neural networks with constraints. One challenge is that most symbolic components are *non-differentiable* which prevents their integration into the gradient-based *learning process* of the deep learning models; The other challenge is that reasoning over the symbolic constraints is #P-hard in general, meaning that the computation can take an unfeasibly long time during the *inference process* and hinder reliable and efficient reasoning over the neurosymbolic pipeline.

Trustworthy AI can be achieved by neurosymbolic methods in various ways. For example, when building neurosymbolic models, constraints can be enforced in the output space of a deep generative model which is a neural network model for data generation such that the generated data follows certain domain knowledge. Further, queries for trustworthiness such as "how confident the model is for its decision" can be answered by probabilistic reasoning over the neurosymbolic model. Besides, neurosymbolic methods allow for effective learning by restricting the training to the feasible space defined by the constraints, that is, for any assignments violating the constraints, they are not considered and always assigned low or zero probabilities.

My research aims to address the aforementioned two fundamental challenges in neurosymbolic AI and to build trustworthy AI models using neurosymbolic methods. The main contributions of this thesis are two-fold, which have been shown to achieve trustworthiness including explainability, uncertainty quantification and domain-knowledge incorporation:

i) Our proposed methods enable differentiable learning in the presence of symbolic constraints by building gradient estimators for the symbolic component, and it allows flexible integration of the constraints into any parts of the neural network architecture.
ii) We propose novel reasoning algorithms over complex constraints that can be applied to perform reliable reasoning over neural networks; this is achieved by translating part of the neural networks into symbolic representations, revealing a deep connection between neural and symbolic reasoning.

This thesis is organized as follows. It consists of two parts for neurosymbolic learning and reasoning respectively. Chapter 2, based on [AZN23b] provides how a k-subset constraint can be

enforced into neural network architectures and how the full pipeline can be effectively trained by our proposed gradient estimators such that the constraint can be differentiated. The proposed gradient estimator is further leveraged to derive a probabilistic graph rewiring algorithm for graph neural networks [QMA24].

Chapter 3 generalizes the same design philosophy of gradient estimators from the k-subset constraints over discrete domains to the linear equality defined over continuous domains, which allows introducing inductive bias into the data generation process of deep generative models, based on a work under review [LSB24].

While the previous two chapters consider hard constraints, that is, constraints with guaranteed satisfaction, Chapter 4 focuses on integrating the soft constraints into the training objectives in count-based weakly supervised learning tasks [SZA23].

The second part focuses on reasoning over complex constraints. Note that in this thesis, we use treat reasoning and inference as synonyms. Chapter 5 provides sufficient backgrounds in weighted model integration (WMI), a unified framework for probabilistic reasoning over algebraic constraints, including its formulations and motivations.

Chapter 6 introduce two exact WMI solvers that achieves state-of-art reasoning performance by exploiting problem structures. While the first solver is search-based and is the first exact WMI solver that comes with guarantees on tractability [ZB19], the second solver, built using a message-passing scheme, is able to solver a strictly larger WMI problem class in a tractable way and is the first WMI solver that can perform amortized inference [ZMY20a].

Chapter 7 is based on [ZMY20b] and starts with theoretical analysis of the inherent hardness of WMI problems. We derive the computational complexity analysis on WMI inference using graph structures to characterize its tractability boundary and discover the largest tractable WMI model class so far. Based on the theoretical understanding, we further propose an approximate WMI by the idea of performing exact inference on an approximate model.

Chapter 8, based on [ZB23b], shows that by combining exact WMI solvers with the collapsed

inference scheme from statistical machine learning, they scale to perform uncertain quantification for image classification tasks with large Bayesian neural networks, outperforming strong baselines in Bayesian deep learning including sampling and variational inference.

Chapter 9, the final one, summarizes the thesis and discuss about future research directions.

Part I

Learning

CHAPTER 2

Differentiable Learning under Constraints

This chapter tackles a fundamental challenge in Neurosymbolic AI: *how to enforce constraints within the architecture of neural networks while still allowing for end-to-end training*. Specifically, we consider a k-subset constraint which is ubiquitous in machine learning. The incorporation of constraints is problematic and requires relaxations or approximations in general which can harm the learning efficiency and model performance. We address this issue by minimizing such relaxations such that constraints can be incorporated into any part of the neural network with guaranteed satisfaction and meanwhile the constrained neural networks are effectively trained.

2.1 Background

k-subset sampling, sampling a subset of size k of n variables, is omnipresent in machine learning. It lies at the core of many fundamental problems that rely upon learning sparse features representations of input data, including stochastic high-dimensional data visualization [Maa09], parametric k-nearest neighbors [GWZ18], learning to explain [CSW18], discrete variational auto-encoders [Rol17], and sparse regression, to name a few. All such tasks involve optimizing an expectation of an objective function with respect to a latent *discrete* distribution parameterized by a neural network, which are often *assumed* intractable. Score-function estimators offer a cloyingly simple solution: rewrite the gradient of the expectation as an expectation of the gradient, which can subsequently be estimated using a finite number of samples offering an unbiased estimate of the gradient. Simple as it is, score-function estimators suffer from very high variance which can interfere with training. This provided the impetus for other, low-variance, gradient

estimators, chief among them are those based on the reparameterization trick, which allows for biased, but low-variance gradient estimates. The reparameterization trick, however, does not allow for a direct application to discrete distributions thereby prompting continuous relaxations, e.g. Gumbel-softmax [JGP17b, MMT17b], that allow for reparameterized gradients w.r.t the parameters of a *categorical* distribution. Reparameterizable subset sampling [XE19b] generalizes the Gumbel-softmax trick to k-subsets which while rendering k-subset sampling amenable to backpropagation at the cost of introducing bias in the learning by using relaxed samples.

2.2 Gradient Estimator for k-subset Sampling

We set out with the goal of avoiding all such relaxations. Instead, we fall back to *discrete* sampling on the forward pass. On the backward pass, we reparameterize the gradient of the loss function with respect to the samples as a function of the *exact* marginals of the k-subset distribution. Computing the exact conditional marginals is, in general, intractable [Rot96a]. We give an *efficient* algorithm for computing the k-subset probability, and show that the conditional marginals correspond to partial derivatives, and are therefore tractable for the k-subset distribution. We show that our proposed gradient estimator for the k-subset distribution, coined SIMPLE, is reminiscent of the straight-through (ST) Gumbel estimator when $k = 1$, with the gradients taken with respect to the unperturbed marginals. We empirically demonstrate that SIMPLE exhibits lower bias *and* variance compared to other known gradient estimators, including the ST Gumbel estimator in the case $k = 1$.

2.2.1 Problem Statement and Motivation

We consider models described by the equations

$$\boldsymbol{\theta} = h_{\boldsymbol{v}}(\boldsymbol{x}), \qquad \boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \textstyle\sum_i z_i = k), \qquad \hat{\boldsymbol{y}} = f_{\boldsymbol{u}}(\boldsymbol{z}, \boldsymbol{x}), \tag{2.1}$$

where $\boldsymbol{x} \in \mathcal{X}$ and $\hat{\boldsymbol{y}} \in \mathcal{Y}$ denote feature inputs and target outputs, respectively, $h_{\boldsymbol{v}} : \mathcal{X} \to \Theta$ and $f_{\boldsymbol{u}} : \mathcal{Z} \times \mathcal{X} \to \mathcal{Y}$ are smooth, parameterized maps and $\boldsymbol{\theta}$ are logits inducing a distribution over

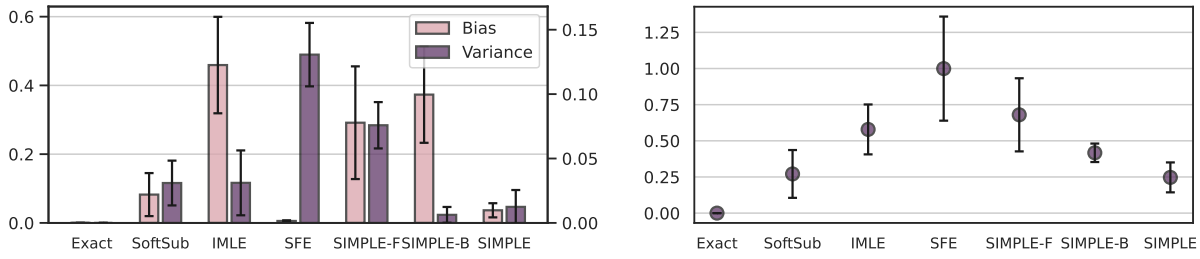


Figure 2.1: A comparison of the bias and variance of the gradient estimators (left) and the average and standard deviation of the cosine distance of a single-sample gradient estimate to the exact gradient. We used the cosine distance, defined as ($1-$ cosine similarity), in place of the euclidean distance as we only care about the direction of the gradient, not magnitude. The bias, variance and error were estimated using a sample of size 10,000. The details of this experiment are provided in Section 2.2.3.1.

the latent binary vector $\boldsymbol{z}$. The induced distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ is defined as

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}) = \prod_{i=1}^{n} p_{\theta_i}(z_i), \quad \text{with} \quad p_{\theta_i}(z_i = 1) = \text{sigmoid}(\theta_i) \quad \text{and} \quad p_{\theta_i}(z_i = 0) = 1 - \text{sigmoid}(\theta_i). \tag{2.2}$$

The goal of our stochastic latent layer is *not* to simply sample from $p_{\boldsymbol{\theta}}(\boldsymbol{z})$, which would yield samples with a Hamming weight between 0 and n (i.e., with an arbitrary number of ones). Instead, we are interested in sampling from the distribution restricted to samples with a Hamming weight of k, for any given k. That is, we are interested in sampling from the conditional distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_i z_i = k)$.

Conditioning the distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ on this k-*subset constraint* introduces intricate dependencies between each of the z_i's. The probability of sampling any given k-subset vector $\boldsymbol{z}$, therefore, becomes

$$p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \textstyle\sum_i z_i = k) = p_{\boldsymbol{\theta}}(\boldsymbol{z}) / p_{\boldsymbol{\theta}}(\textstyle\sum_i z_i = k) \cdot [\![\textstyle\sum_i z_i = k]\!]$$

where $[\![\cdot]\!]$ denotes the indicator function. In other words, the probability of sampling each k-subset is re-normalized by $p_{\boldsymbol{\theta}}\left(\sum_i z_i = k\right)$ – the probability of sampling exactly k items from the *unconstrained* distribution induced by encoder $h_{\boldsymbol{v}}$. The quantity $p_{\boldsymbol{\theta}}(\sum_i z_i = k) = \sum_{\mathbf{z}} p_\theta(\mathbf{z}) \cdot$

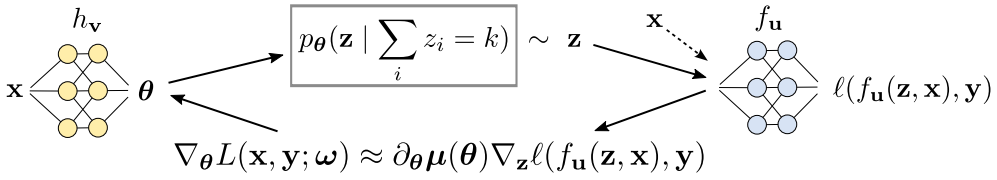


Figure 2.2: The problem setting considered in this section. On the forward pass, a neural network h_v outputs $\boldsymbol{\theta}$ parameterizing a *discrete* distribution over subsets of size k of n items, i.e., the k-subset distribution. We sample exactly, and efficiently, from this distribution, and feed the samples to a downstream neural network. On the backward pass, we approximate the true gradient by the product of the derivative of marginals and the gradient of the sample-wise loss.

Task	**Map** $h_{\boldsymbol{v}}$	**Map** $f_{\boldsymbol{u}}$	**Loss** ℓ
Discrete VAE (Sec. 2.2.3.2)	Encoder	Decoder	ELBO
Learn To Explain (Sec. 2.2.3.3)	Embedding	Regression	RMSE
Sparse Regression (Sec. 2.2.3.4)	Identity	Linear Regression	RMSE

Table 2.1: Architectures of the three experiment settings.

$[\![\sum_i z_i = k]\!]$ appears to be intractable. We show that not to be the case, providing a tractable algorithm for computing it.

Given a set of samples $\mathcal{D}$, we are concerned with learning the parameters $\boldsymbol{\omega} = (\boldsymbol{v}, \boldsymbol{u})$ of the architecture in (3.1) through minimizing the training error L, which is the expected loss:

$$L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}) = \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_i z_i = k)}[\ell(f_{\boldsymbol{u}}(\boldsymbol{z}, \boldsymbol{x}), \boldsymbol{y})] \qquad \textit{with } \boldsymbol{\theta} = h_{\boldsymbol{v}}(\boldsymbol{x}), \tag{2.3}$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is a point-wise loss function. This formulation, illustrated in Figure 2.2, is general and subsumes many settings. Different choices of mappings $h_{\boldsymbol{v}}$ and $f_{\boldsymbol{u}}$, and sample-wise loss ℓ define various tasks. Table 2.1 presents some example settings used in our experimental evaluation.

Learning then requires computing the gradient of L w.r.t. $\boldsymbol{\omega} = (\boldsymbol{v}, \boldsymbol{u})$. The gradient of L w.r.t.

$\boldsymbol{u}$ is

$$\nabla_{\boldsymbol{u}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}) = \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z} | \sum_i z_i = k)} [\partial_{\boldsymbol{u}} f_{\boldsymbol{u}}(\boldsymbol{z}, \boldsymbol{x})^\top \nabla_{\hat{\boldsymbol{y}}} \ell(\hat{\boldsymbol{y}}, \boldsymbol{y})], \tag{2.4}$$

where $\hat{y} = f_{\boldsymbol{u}}(\boldsymbol{z}, \boldsymbol{x})$ is the decoding of a latent sample $\boldsymbol{z}$. Furthermore, the gradient of L w.r.t. $\boldsymbol{v}$ is

$$\nabla_{\boldsymbol{v}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}) = \partial_{\boldsymbol{v}} h_{\boldsymbol{v}}(\boldsymbol{x})^\top \nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}), \tag{2.5}$$

where $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}) := \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z} | \sum_i z_i = k)} [\ell(f_{\boldsymbol{u}}(\boldsymbol{z}, \boldsymbol{x}), \hat{\boldsymbol{y}})]$, the loss' gradient w.r.t. the encoder.

One challenge lies in computing the expectation in (3.2) and (3.3), which has no known closed-form solution. This necessitates a Monte-Carlo estimate via sampling from $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_i z_i = k)$.

A second, and perhaps more substantial hurdle lies in computing $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega})$ in (3.4) due to the non-differentiable nature of discrete sampling. One could rewrite $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega})$ as

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}) = \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{\theta}}\left(\mathbf{z} | \sum_i z_i = k\right)} [\ell(f_{\boldsymbol{u}}(\boldsymbol{z}, \boldsymbol{x}), \boldsymbol{y}) \nabla_{\boldsymbol{\theta}} \log p_\theta \left(\mathbf{z} \mid \sum_i z_i = k\right)]$$

which is known as the REINFORCE estimator [Wil92], or the score function estimator (SFE). It is typically avoided due to its notoriously high variance, despite its apparent simplicity. Instead, typical approaches [XE19b, PR18] reparameterize the samples as a deterministic transformation of the parameters, and some independent standard Gumbel noise, and relaxing the deterministic transformation, the top-k function in this case, to allow for backpropagation.

2.2.2 SIMPLE: Subset Implicit Likelihood Estimation

Our goal is to build a gradient estimator for $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega})$. We start by envisioning a hypothetical sampling-free architecture, where the downstream neural network $f_{\boldsymbol{u}}$ is a function of the marginals, $\boldsymbol{\mu} \coloneqq \mu(\boldsymbol{\theta}) \coloneqq \{p_{\boldsymbol{\theta}}(z_j \mid \sum_i z_i = k)\}_{j=1}^n$, instead of a discrete sample $\boldsymbol{z}$, resulting in a loss L_m s.t.

$$\nabla_{\boldsymbol{\theta}} L_m(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}) = \partial_{\boldsymbol{\theta}} \mu(\boldsymbol{\theta})^\top \nabla_{\boldsymbol{\mu}} \ell_m(f_{\boldsymbol{u}}(\boldsymbol{\mu}, \boldsymbol{x}), \boldsymbol{y}). \tag{2.6}$$

When the marginals $\mu(\boldsymbol{\theta})$ can be efficiently computed and differentiated, such a hypothetical pipeline can be trained end-to-end. Furthermore, [Dom10] observed that, for an arbitrary loss

function ℓ_m defined on the marginals, the Jacobian of the marginals w.r.t. the logits is symmetric:

$$\nabla_{\boldsymbol{\theta}} L_m(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}) = \partial_{\boldsymbol{\theta}} \mu(\boldsymbol{\theta})^\top \nabla_{\boldsymbol{\mu}} \ell_m(f_{\boldsymbol{u}}(\boldsymbol{\mu}, \boldsymbol{x}), \boldsymbol{y}) = \partial_{\boldsymbol{\theta}} \mu(\boldsymbol{\theta}) \nabla_{\boldsymbol{\mu}} \ell_m(f_{\boldsymbol{u}}(\boldsymbol{\mu}, \boldsymbol{x}), \boldsymbol{y}). \tag{2.7}$$

Consequently, computing the gradient of the loss w.r.t. the logits, $\nabla_{\boldsymbol{\theta}} L_m(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega})$, reduces to computing the *directional derivative*, or the Jacobian-vector product, of the marginals w.r.t. the logits in the direction of the gradient of the loss. This offers an alluring opportunity: the conditional marginals characterize the probability of each z_i in the sample, and could be thought of as a differentiable proxy for the samples. Specifically, by reparameterizing $\boldsymbol{z}$ as a function of the conditional marginal $\boldsymbol{\mu}$ under approximation $\partial_{\boldsymbol{\mu}} \boldsymbol{z} \approx \mathbf{I}$ as proposed by [NMF21b], and using the straight-through estimator for the gradient of the sample w.r.t. the marginals on the backward pass, we approximate our true $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega})$ as

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}) \approx \partial_{\boldsymbol{\theta}} \mu(\boldsymbol{\theta}) \nabla_{\boldsymbol{z}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}), \tag{2.8}$$

where the directional derivative of the marginals can be taken along *any downstream gradient*, rendering the whole pipeline end-to-end learnable, even in the presence of non-differentiable sampling.

Now, estimating the gradient of the loss w.r.t. the parameters can be thought of as decomposing into two sub-problems: **(P1)** Computing the derivatives of conditional marginals $\partial_{\boldsymbol{\theta}} \mu(\boldsymbol{\theta})$, which requires the computation of the conditional marginals, and **(P2)** Computing the gradient of the loss w.r.t. the samples $\nabla_{\boldsymbol{z}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega})$ using sample-wise loss, which requires drawing exact samples. These two problems are complicated by conditioning on the k-subset constraint, which introduces intricate dependencies to the distribution, and is infeasible to solve naively, e.g. by enumeration. We will show simple, efficient, and exact solutions to each problem, at the heart of which is the insight that we need not care about the variables' order, only their sum, introducing symmetries that simplify the problem.

2.2.2.1 Derivatives of Conditional Marginals

In many probabilistic models, marginal inference is #P-hard [Rot96a, ZMY20b]. However, we observe that it is not the case for the k-subset distribution. We notice that the conditional marginals correspond to the partial derivatives of the log-probability of the k-subset constraint. To see this, note that the derivative of a multi-linear function with respect to a single variable retains all the terms referencing that variable, and drops all other terms; this corresponds exactly to the unnormalized conditional marginals. By taking the derivative of the log-probability, this introduces the k-subset probability in the denominator, leading to the *conditional* marginals. Intuitively, the rate of change of the k-subset probability w.r.t. a variable only depends on that variable through its length-k subsets.

Theorem 1. *Let $p_{\boldsymbol{\theta}}(\sum_j z_j = k)$ be the probability of exactly-k of the unconstrained distribution parameterized by logits $\boldsymbol{\theta}$. Let $\alpha_i \coloneqq \log p_{\boldsymbol{\theta}}(z_i)$ denote the log marginals. For every variable Z_i, its conditional marginal is*

$$p_{\boldsymbol{\theta}}\Big(z_i \mid \textstyle\sum_j z_j = k\Big) = \dfrac{\partial}{\partial \alpha_i} \log p_{\boldsymbol{\theta}}(\textstyle\sum_j z_j = k). \tag{2.9}$$

To establish the tractability of the above computation of the conditional marginals, we need to show that the probability of the exactly-k constraint $p_{\boldsymbol{\theta}}(\sum_i z_i = k)$ can be obtained tractably, which we demonstrate next.

Proposition 1. *The probability $p_{\boldsymbol{\theta}}\left(\sum_i z_i = k\right)$ of sampling exactly k items from the* unconstrained *distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ over n items as in Equation 2.2 can be computed exactly in time $\mathcal{O}(nk)$.*

Proof. Our proof is constructive. As a base case, consider the probability of sampling $k = -1$ out of $n = 0$ items. We can see that the probability of such an event is 0. As a second base case, consider the probability of sampling $k = 0$ out of $n = 0$ items. We can see that the probably of such an event is 1. Now assume that we are given the probability $p_{\boldsymbol{\theta}}\left(\sum_i^{n-1} z_i = k'\right)$, for $k' = 0, \ldots, k$, and we are interested in computing $p_{\boldsymbol{\theta}}\left(\sum_i^{n} z_i = k\right)$. By the partition theorem, we

Algorithm 1 `PrExactlyk`$(\boldsymbol{\theta}, n, k)$

Input: The logits $\boldsymbol{\theta}$ of the distribution, the number of variables n, and the subset size k

Output: $p_{\boldsymbol{\theta}}(\sum_i z_i = k)$

// $a[i, j] = p_{\boldsymbol{\theta}}(\sum_{m=1}^{i} z_m = j)$ for all i, j

initialize a to be 0 everywhere

$a[0, 0] = 1$ // $p_{\boldsymbol{\theta}}(\sum_{m=1}^{0} z_m = 0) = 1$

for $i = 1$ **to** n **do**

  for $j = 0$ **to** k **do**

    // cf. constructive proof of Prop. 1

    $a[i, j] = a[i-1, j] \cdot p_{\theta_i}(z_i = 0)$

      $+ a[i-1, j-1] \cdot p_{\theta_i}(z_i = 1)$

return $a[n, k]$

Algorithm 2 `Sample`$(\boldsymbol{\theta}, n, k)$

Input: The logits $\boldsymbol{\theta}$ of the distribution, the number of variables n, and the subset size k

Output: $\boldsymbol{z} = (z_1, \ldots, z_n) \sim p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_i z_i = k)$

sample = [ ], $j = k$

for $i = n$ **to** 1 **do**

  // cf. proof of Prop. 2

  $p = a[i-1, j-1]$

  $z_i \sim$ Bernoulli$(p \cdot p_{\theta_i}(z_i = 1)/a[i, j])$

  // Pick next state based on value of sample

  if $z_i = 1$ **then** $j = j - 1$

  sample.append(z_i)

return sample

can see that

$$p_{\boldsymbol{\theta}}\left(\sum_i^n z_i = k\right) = p_{\boldsymbol{\theta}}\left(\sum_i^{n-1} z_i = k\right) \cdot p_{\theta_n}(z_n = 0) + p_{\boldsymbol{\theta}}\left(\sum_i^{n-1} z_i = k-1\right) \cdot p_{\theta_n}(z_n = 1)$$

as events $\sum_i^{n-1} z_i = k$ and $\sum_i^{n-1} z_i = k - 1$ are disjoint and, for any k, partition the sample space. Intuitively, for any k and n, we can sample k out of n items by choosing k of $n-1$ items, and not the n-th item, or choosing $k-1$ of $n-1$ items, and the n-th item. The above process gives rise to Algorithm 1, which returns $p_{\boldsymbol{\theta}}\left(\sum_i z_i = k\right)$ in time $\mathcal{O}(nk)$. □

By the construction described above, we obtain a closed-form $p_{\boldsymbol{\theta}}(\sum_i^n z_i = k)$, which allows us to compute conditional marginals $p_{\boldsymbol{\theta}}(z_i \mid \sum_j z_j = k)$ by Theorem 1 via auto-differentiation. This further allows the computation of the derivatives of conditional marginals

$$\partial_{\boldsymbol{\theta}} \mu(\boldsymbol{\theta})_i = \partial_{\boldsymbol{\theta}}\, p_{\boldsymbol{\theta}}(z_i \mid \textstyle\sum_j z_j = k)$$

to be amenable to auto-differentiation, solving problem **(P1)** exactly and efficiently.

2.2.2.2 Gradients of Loss w.r.t. Samples

As alluded to in Section 2.2.2, we approximate $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega})$ by the directional derivative of the marginals along the gradient of the loss w.r.t. discrete samples $\boldsymbol{z}$, $\nabla_{\boldsymbol{z}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega})$, where $\boldsymbol{z}$ is drawn from the k-subset distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_i z_i = k)$. What remains is to estimate the loss, necessitating faithful sampling from the k-subset distribution, which might appear daunting.

Exact k-subset Sampling Next we show how to sample exactly from the k-subset distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_i z_i = k)$. We start by sampling the variables in reverse order, that is, we sample z_n through z_1. The main intuition being that, having sampled $(z_n, z_{n-1}, \cdots, z_{i+1})$ with a Hamming weight of $k - j$, we sample Z_i with a probability of choosing $k - j$ of $n - 1$ variables *and* the n-th variable *given that* we choose $k - j + 1$ of n variables. We formalize our intuition below.

Proposition 2. *Let* `Sample` *be defined as in Algorithm 2. Given n random variables $Z_1, \cdots, Z_n$, a subset size k, and a k-subset distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_i z_i = k)$ parameterized by log probabilities $\boldsymbol{\theta}$, Algorithm 2 draws exact samples from $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_i z_i = k)$ in time $\mathcal{O}(n)$.*

Proof. Assume that variables $Z_n, \cdots, Z_{i+1}$ are sampled and have their values to be $z_n, \cdots, z_{i+1}$ with $\sum_{m=i+1}^{n} z_m = k - j$. By Algorithm 2 we have that the probability for sampling Z_i is

$$
\begin{aligned}
p_{\texttt{Sample}}(z_i = 1 \mid z_n, \cdots, z_{i+1}) &= \frac{p_{\boldsymbol{\theta}}(\sum_{m=i}^{n} z_m = k - j + 1 \mid \sum_m z_m = k)\, p_{\theta_i}(z_i = 1)}{p_{\boldsymbol{\theta}}(\sum_{m=i+1}^{n} z_m = k - j \mid \sum_m z_m = k)} \\
&= \frac{p_{\boldsymbol{\theta}}(\sum_{m=i+1}^{n} z_m = k - j \mid z_i = 1, \sum_m z_m = k)\, p_{\theta_i}(z_i = 1)}{p_{\boldsymbol{\theta}}(\sum_{m=i+1}^{n} z_m = k - j \mid \sum_m z_m = k)} \\
&= p_{\boldsymbol{\theta}}(z_i = 1 \mid \textstyle\sum_{m=i+1}^{n} z_m = k - j, \sum_m z_m = k) \quad \textit{(by Bayes' theorem)}
\end{aligned}
$$

It follows that samples drawn from Algorithm 2 distribute according to $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_i z_i = k)$. □

2.2.2.3 Connection to Straight-Through Gumbel-Softmax

One might wonder if our gradient estimator reduces to the Straight-Through (ST) Gumbel-Softmax estimator, or relates to it in any way when $k = 1$. On the forward pass, the ST Gumbel Softmax

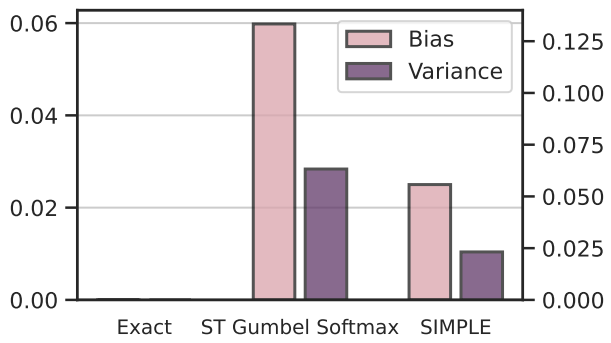


Figure 2.3: Bias and variance of SIMPLE and Gumbel Softmax over 10k samples

estimator makes use of the Gumbel-Max trick [MTM14], which states that we can efficiently sample from a categorical distribution by perturbing each of the logits with standard Gumbel noise, and taking the MAP, or more formally $\boldsymbol{z} = \text{OneHot}(\arg\max_{i \in \{1,\dots,k\}} \boldsymbol{\theta}_i + g_i) \sim p_{\boldsymbol{\theta}}$ where the g_i's are i.i.d Gumbel$(0, 1)$ samples, and OneHot encodes the sample as a binary vector.

Since arg max is non-differentiable, Gumbel-Softmax uses the *perturbed* relaxed samples, $\boldsymbol{y} = \text{Softmax}(\boldsymbol{\theta} + g_i)$ as a proxy for discrete samples $\boldsymbol{z}$ on the backward pass, using differentiable Softmax in place of the non-differentiable arg max, with the entire function returning $(\boldsymbol{z} - \boldsymbol{y}).\,\text{detach}() + \boldsymbol{y}$ where detach ensures that the gradient flows only through the relaxed samples on the backward pass.

That is, just like SIMPLE, ST Gumbel-Softmax returns *exact*, *discrete* samples. However, whereas SIMPLE backpropagates through the exact marginals, ST Gumbel Softmax backpropagates through the *perturbed* marginals that result from applying the Gumbel-max trick. As can be seen in Figure 2.3, such a minor difference means that, empirically, SIMPLE exhibits lower bias and variance compared to ST Gumbel Softmax while being exactly as efficient.

Related Work

There is a large body of work on gradient estimation for categorical random variables. [MMT17b, JGP17b] propose the Gumbel-softmax distribution (named the concrete distribution by the former) to relax categorical random variables. For more complex distributions, such as the k-subset

Algorithm 3 The proposed algorithm for the k-subset distribution

function ForwardPass($\boldsymbol{\theta}$)

// $p_{\boldsymbol{\theta}}(\sum_{m=1}^{i} z_m = j)$ *for all* i, j

$a = \texttt{PrExactlyk}(\boldsymbol{\theta}, n, k)$

// *Sample from* $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_i z_i = k)$

$\boldsymbol{z} = \texttt{Sample}(\boldsymbol{\theta}, n, k)$

save a for the backward pass

return $\boldsymbol{z}$

function BackwardPass($\nabla_{\boldsymbol{z}} \ell(f_{\boldsymbol{u}}(\boldsymbol{z}, \boldsymbol{x}), \boldsymbol{y})$)

load $\boldsymbol{\theta}$ from the forward pass

// *derivatives of* $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_i z_i = k)$

$\boldsymbol{\mu} = \nabla_{\boldsymbol{\theta}} \log a[n, k]$ // by auto-diff

// *Return the directional derivative of the*

// *marginals along the downstream gradients*

return JVP($\boldsymbol{\mu}, \nabla_{\boldsymbol{z}} \ell(f_{\boldsymbol{u}}(\boldsymbol{z}, \boldsymbol{x}))$)

distribution which we are concerned with in this section, existing approaches either use the straight-through and score function estimators or propose tailor-made relaxations (see for instance [KSE16, CSW18, GWZ18]). We directly compare to the score function and straight-through estimator as well as the tailored relaxations of [CSW18, GWZ18] and show that we are competitive and obtain a lower bias and/or variance than these other estimators. [TMM17, GCW18] develop parameterized control variates based on continuous relaxations for the score-function estimator. Lastly, [PCT20] offers a comprehensible work on relaxed gradient estimators, deriving several extensions of the softmax trick. All of the above works, ours included, assume the independence of the selected items, beyond there being k of them. That is with the exception of [PCT20] which make use of a relaxation using pairwise embeddings, but do not make their code available. We leave that to future work.

A related line of work has developed and analyzed sparse variants of the softmax function, motivated by their potential computational and statistical advantages. Representative examples are [BMN20, PNM19, CNM19, MA16]. SparseMAP [NMB18] has been proposed in the context of structured prediction and latent variable models, also replacing the softmax with a sparser distribution. LP-SparseMAP [NM20] is an extension that uses a relaxation of the optimization problem rather than a MAP solver. Sparsity can also be exploited for efficient marginal inference in latent variable models [CNA20]. Contrary to our work, they cannot control the sparsity level

exactly through a k-subset constraint or guarantee a sparse output. Also, we aim at cases where samples in the forward pass are required.

Integrating specialized discrete algorithms into neural networks is growing in popularity. Examples are sorting algorithms [CTV19, BTB20, GWZ18], ranking [RMP20, KVW19], dynamic programming [MB18, CT19], and solvers for combinatorial optimization problems [BBT20, RSZ20, SBK20, NMF21b, MFN23, ZMY21a] or even probabilistic circuits over structured output spaces [ATC22a, Blo19]. There has also been work on making common programming language expression such as conditional statements, loops, and indexing differentiable through relaxations [PBK21]. [XDC20] propose optimal transport to obtain differentiable sorting methods for top-k classification.

2.2.3 Empirical Evaluation

We conduct experiments on four different tasks: 1) A synthetic experiment designed to test the bias and variance, as well as the average deviation of SIMPLE compared to a variety of well-established estimators in the literature. 2) A discrete k-subset Variational Auto-Encoder (DVAE) setting, where the latent space models a probability distribution over k-subsets. We will show that we can compute the evidence lower bound (ELBO) exactly, and that, coupled with exact sampling and our SIMPLE gradient estimator, we attain a much lower loss compared to state of the art in sparse DVAEs. 3) The learning to explain (L2X) setting, where the aim is to select the k-subset of words that best describe the classifier's prediction, where we show an improved mean-squared error, as well as precision, across the board. 4) A novel, yet simple task, sparse linear regression, where, in a vein similar to L2X, we wish to select a k-subset of features that give rise to a linear regression model, avoiding overfitting the spurious features present in the data. Table 2.1 details the architecture with the objective functions. Our code is publicly available at github.com/UCLA-StarAI/SIMPLE.

2.2.3.1 Synthetic Experiments

We carried out a series of experiments with a 5-subset distribution, and a latent space of dimension 10. We set the loss to $L(\boldsymbol{\theta}) = \mathbb{E}_{z \sim p_{\boldsymbol{\theta}}\left(\mathbf{z} | \sum_i z_i = k\right)}[\|\boldsymbol{z} - \mathbf{b}\|^2]$, where $\mathbf{b}$ is the groundtruth logits sampled from $\mathcal{N}(0, \mathbf{I})$. Such a distribution is tractable: we only have $\binom{10}{5} = 252$ k-subsets, which are easily enumerable and therefore, the exact gradient, the golden standard, can be computed in closed form.

In this experiment, we are interested in three metrics: bias, variance, and the average error of each gradient estimator, where the latter is measured by averaging the deviation of each single-sample gradient estimate from the exact gradient. We used the cosine distance, defined as $1-$ cosine similarity as the measure of deviation in our calculation of the metrics above, as we only care about direction.

We compare against four different baselines: *exact*, which denotes the exact gradient; *SoftSub* [XE19b], which uses an extension of the Gumbel-Softmax trick to sample *relaxed* k-subsets on the forward pass; I-MLE, which denotes the IMLE gradient estimator [NMF21b], where *approximate* samples are obtained using perturb-and-map (PAM) on the forward pass, approximating the marginals using PAM samples on the backward pass; and score function estimator, *SFE*.

We tease apart SIMPLE's improvements by comparing three different flavors: *SIMPLE-F*, which only uses *exact* sampling, falling back to estimating the marginals using *exact* samples; *SIMPLE-B*, which uses *exact* marginals on the backward pass with *approximate* PAM samples on the forward pass; and SIMPLE, coupling *exact* samples on the forward pass with *exact* marginals on the backward pass.

Our results are shown in Figure 2.1 As expected, we observe that *SFE* exhibits no bias, but high variance whereas *SoftSub* suffers from both bias and variance, due to the Gumbel noise injection into the samples to make them differentiable. We observe that I-MLE exhibits very high bias, as well as very low variance. This can be attributed to the PAM sampling, which in the case of k-subset distribution does not sample faithfully from the distribution, but is instead

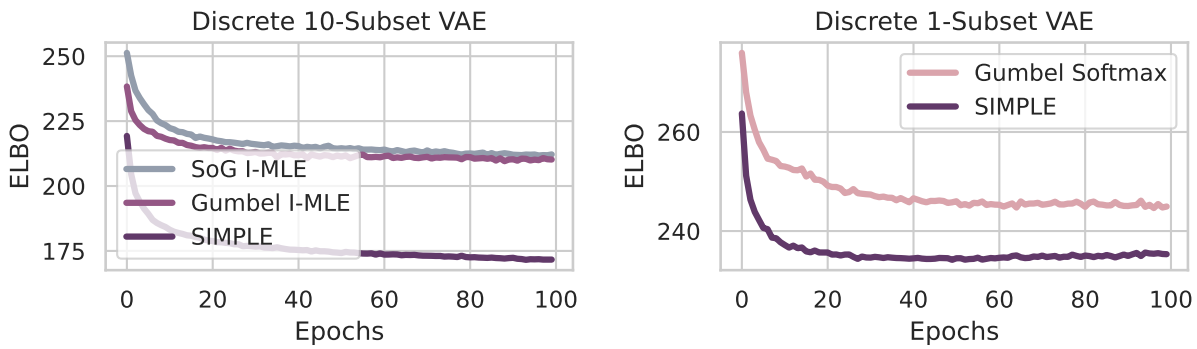


Figure 2.4: ELBO against # of epochs. (Left) Comparison of SIMPLE against different flavors of IMLE on the 10-subset DVAE, and (Right) against ST Gumbel Softmax on the 1-subset DVAE.

biased to sampling only the mode of the distribution. This also means that, by approximating the marginals using PAM samples, there is a lot less variance to our gradients. On to our SIMPLE gradient estimator, we see that it exhibits *less bias as well as less variance* compared to all the other gradient estimators. We also see that each estimated gradient is, on average, much more aligned with the exact gradient. To understand why that is, we compare SIMPLE, SIMPLE-F, and SIMPLE-B. As hypothesized, we observe that *exact sampling*, SIMPLE-F, reduces the bias, but increases the variance compared to I-MLE, this is since, unlike the PAM samples, our exact sample span the entire sample space.We also observe that, even compared to I-MLE, SIMPLE-B, reduces the variance by marginalizing over all possible samples.

2.2.3.2 Discrete Variational Auto-Encoder

Next, we test our SIMPLE gradient estimator in the k-subset discrete variational auto-encoder (DVAE) setting, where the latent variables model a probability distribution over k-subsets, and has a dimensionality of 20. Similar to prior work [JGP17b, NMF21b], the encoding and decoding functions of the VAE consist of three dense layers (encoding: 512-256-20x20; decoding: 256-512-784). The DVAE is trained to minimize the sum of reconstruction loss and KL-divergence of the k-subset distribution and the constrained uniform distribution, known as the ELBO, on MNIST.

In prior work, the KL-divergence was approximated using the unconditional marginals, ob-

Algorithm 4 $\texttt{Entropy}(\boldsymbol{\theta}, n, k)$

Input: The logits $\boldsymbol{\theta}$ of the distribution, the number of variables n, and the subset size k

Output: $H(\boldsymbol{z}) = -\mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z} | \sum_i z_i = k)}[\log p(\boldsymbol{z})]$

$h = \text{zeros}(n, k)$

for $i = k$ **to** n **do**

  for $j = 0$ **to** k **do**

    // $p(z_i \mid \sum_{m=1}^{i} z_m = j)$

    $p = a[i-1, j-1] * p_{\theta_i}(z_i = 1)/a[i, j]$

    $h[i, j] = H_b(p) + p * h[i-1, j] +$
    $(1-p) * h[i-1, j+1]$

return h

tained simply through a Softmax layer. Instead we show that the KL-divergence between the k-subset distribution and the uniform distribution can be computed exactly. First note that, through simple algebraic manipulations, the KL-divergence between the k-subset distribution and the constrained uniform distribution can be rewritten as the sum of negative entropy, $-H(\boldsymbol{z})$, where $\boldsymbol{z} \sim p_{\boldsymbol{\theta}}\left(\boldsymbol{z} \mid \sum_i z_i = k\right)$ and log the number of k-subsets, $\log \binom{n}{k}$, reducing the hardness of computing the KL-divergence, to computing the entropy of a k-subset distribution, for which Algorithm 4 gives a tractable algorithm. Intuitively, the uncertainty in the distribution over a sequence of length n, k of which are true, decomposes as the uncertainty over Z_n, and the average of the uncertainties over the remainder of the sequence.

Proposition 3. *Let* $\texttt{Entropy}$ *be defined as in Algorithm 4. Given variables,* $Z_1, \cdots, Z_n$*, and a* k*-subset distribution* $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_i z_i = k)$*, Algorithm 4 computes entropy of* $p_{\boldsymbol{\theta}}\left(\boldsymbol{z} \mid \sum_i z_i = k\right)$*.*

We plot the loss ELBO against the number of epochs, as seen in Figure 2.4. We compared against I-MLE using sum-of-gamma noise as well as Gumbel noise for PAM sampling, on the 10-subset DVAE, and against ST Gumbel Softmax on the 1-subset DVAE. We observe a *significantly*

lower loss on the test set on the 10-subset DVAE, partly attributable to the exact ELBO computation, but also on the 1-subset DVAE compared to ST Gumbel Softmax, where the sole difference is the backward pass.

2.2.3.3 Learning to Explain

The BeerAdvocate dataset [MLJ12] consists of free-text reviews and ratings for 4 different aspects of beer: appearance, aroma, palate, and taste. The training set has 80k reviews for the aspect Appearance and 70k reviews for all other aspects. In addition to the ratings for all reviews, each sentence in the test set contains annotations of the words that best describe the review score with respect to the various aspects. We address the problem introduced by the L2X paper [CSW18] of learning a k-subset distribution over words that best explain a given rating. We follow the architecture suggested in the L2X paper, consisting of four convolutional and one dense layer.

We compare to relaxation-based baselines L2X [CSW18] and SoftSub [XE19b] as well as to I-MLE which uses perturb-and-MAP to both compute an approximate sample in the forward pass and to estimate the marginals. Prior work has shown that the straight-through estimator (STE) did not work well and we omit it here. We used the standard hyperparameter settings of [CSW18] and choose the temperature parameter $t \in \{0.1, 0.5, 1.0, 2.0\}$ for all methods. We used the standard Adam settings and trained separate models for each aspect using MSE as point-wise loss ℓ. Table 2.3 lists results for $k \in \{5, 10, 15\}$ for the Aroma aspect. The mean-squared error (MSE) of Simple is almost always lower and its subset precision never significantly exceeded by those of the baselines. Table 2.2 shows results on the remaining aspects Appearance, Palate, and Taste for $k = 10$.

2.2.3.4 Sparse Linear Regression

Given a library of feature functions, the task of sparse linear regression aims to learn from data which feature subset best describes the nonlinear partial differential equation (PDE) that the data are sampled from. We propose to tackle this task by learning a k-subset distribution over the feature functions. During learning, we first sample from the k-subset distribution to decide

Method	Appearance		Palate		Taste	
	Test MSE	**Precision**	**Test MSE**	**Precision**	**Test MSE**	**Precision**
SIMPLE (Ours)	**2.35 ± 0.28**	**66.81 ± 7.56**	**2.68 ± 0.06**	**44.78 ± 2.75**	**2.11 ± 0.02**	**42.31 ± 0.61**
L2X (t = 0.1)	10.70 ± 4.82	30.02 ± 15.82	6.70 ± 0.63	**50.39 ± 13.58**	6.92 ± 1.61	32.23 ± 4.92
SoftSub (t = 0.5)	**2.48 ± 0.10**	52.86 ± 7.08	2.94 ± 0.08	39.17 ± 3.17	2.18 ± 0.10	**41.98 ± 1.42**
I-MLE (τ = 30)	**2.51 ± 0.05**	**65.47 ± 4.95**	2.96 ± 0.04	40.73 ± 3.15	2.38 ± 0.04	**41.38 ± 1.55**

Table 2.2: Results for three aspects with $k = 10$: test MSE and subset precision, both $\times 100$

Method	$k = 5$		$k = 10$		$k = 15$	
	Test MSE	**Precision**	**Test MSE**	**Precision**	**Test MSE**	**Precision**
SIMPLE (Ours)	**2.27 ± 0.05**	**57.30 ± 3.04**	**2.23 ± 0.03**	**47.17 ± 2.11**	3.20 ± 0.04	**53.18 ± 1.09**
L2X (t = 0.1)	5.75 ± 0.30	33.63 ± 6.91	6.68 ± 1.08	26.65 ± 9.39	7.71 ± 0.64	23.49 ± 10.93
SoftSub (t = 0.5)	2.57 ± 0.12	**54.06 ± 6.29**	2.67 ± 0.14	44.44 ± 2.27	**2.52 ± 0.07**	37.78 ± 1.71
I-MLE (τ = 30)	2.62 ± 0.05	**54.76 ± 2.50**	2.71 ± 0.10	**47.98 ± 2.26**	2.91 ± 0.18	39.56 ± 2.07

Table 2.3: Results for aspect Aroma: test MSE and subset precision, both $\times 100$, for $k \in \{5, 10, 15\}$.

which feature function subset to choose. With k chosen features, we perform linear regression to learn the coefficients of the features from data, and then update the k-subset distribution logit parameters by minimizing RMSE.

To test our proposed approach, we follow the experimental setting in PySINDy [SCQ20, KSF22] and use the dataset collected by PySINDy where the samples are collected from the Kuramoto–Sivashinsky (KS) equation, a fourth-order nonlinear PDE known for its chaotic behavior. This PDE takes the form $v_t = -v_{xx} - v_{xxxx} - vv_x$, which can be seen as a linear combination of feature functions $\mathcal{V} = \{v_{xx}, v_{xxxx}, vv_x\}$ with the coefficients all set to a value of -1. At test time, we use the MAP estimation of the learned k-subset distribution to choose the k feature

functions. For $k = 3$, our proposed method achieves the same performance as the state-of-the-art solver on this task, PySINDy. It identifies the KS PDE from data by choosing exactly the ground truth feature function subset $\mathcal{V}$, obtaining an RMSE of 0.00622 after applying linear regression on $\mathcal{V}$.

2.3 SIMPLE for Graph Rewiring

Besides the applications shown in the previous section, this section demonstrates how SIMPLE excels at a complex tasks, *probabilistically rewiring message-passing graph neural networks (MPNNs)*. MPNNs emerge as powerful tools for processing graph-structured input. However, they operate on a fixed input graph structure, ignoring potential noise and missing information. Furthermore, their local aggregation mechanism can lead to problems such as over-squashing and limited expressive power in capturing relevant graph structures. Existing solutions to these challenges have primarily relied on heuristic methods, often disregarding the underlying data distribution. Hence, devising principled approaches for learning to infer graph structures relevant to the given prediction task remains an open challenge. We devise probabilistically rewired MPNNs (PR-MPNNs), which learn to add relevant edges while omitting less beneficial ones using differentiable k-subset sampling and we show that when combined with SIMPLE, PR-MPNNs show superior performance.

2.3.1 Background

Graph-structured data is prevalent across various application domains, including fields like chemo- and bioinformatics [BO04, JEP21, RNE22], combinatorial optimization [CCK23], and social-network analysis [EK12], highlighting the need for machine learning techniques designed explicitly for graphs. In recent years, *message-passing graph neural networks* (MPNNs) [KW17, GSR17, SGT08a, VCC18] have become the dominant approach, showing promising performance in tasks such as predicting molecular properties [KGG20, JEP21] or enhancing combinatorial solvers [CCK23].

However, MPNNs have a limitation due to their local aggregation mechanism. They focus on encoding local structures, severely limiting their expressive power [MRF19, MLM21, XHL19]. In

addition, MPNNs struggle to capture global or long-range information, possibly leading to phenomena like *under-reaching* [BKM20] or *over-squashing* [AY21]. Over-squashing, as explained by [AY21], refers to excessive information compression from distant nodes due to a source node's extensive receptive field, occurring when too many layers are stacked. Existing strategies to mitigate over-squashing rely on heuristic rewiring methods that may not adapt well to a prediction task or employ computationally intensive global attention mechanisms. Furthermore, the impact of probabilistic rewiring on a model's expressive power remains unclear.

We formalize the necessary notations and concepts as below.

Notations Let $\mathbb{N} \coloneqq \{1, 2, 3, \dots\}$. For $n \geq 1$, let $[n] \coloneqq \{1, \dots, n\} \subset \mathbb{N}$. We use $\{\!\{\dots\}\!\}$ to denote multisets, i.e., the generalization of sets allowing for multiple instances for each of its elements. A *graph* G is a pair $(V(G), E(G))$ with *finite* sets of *vertices* or *nodes* $V(G)$ and *edges* $E(G) \subseteq \{\{u, v\} \subseteq V(G) \mid u \neq v\}$. If not otherwise stated, we set $n \coloneqq |V(G)|$, and the graph is of *order* n. We also call the graph G an n-order graph. For ease of notation, we denote the edge $\{u, v\}$ in $E(G)$ by (u, v) or (v, u). Throughout this section, we use standard notations, e.g., we denote the *neighborhood* of a vertex v by $N(v)$ and $\ell(v)$ denotes its discrete vertex label, and so on.

1-dimensional Weisfeiler–Leman algorithm The 1-WL or color refinement is a well-studied heuristic for the graph isomorphism problem, originally proposed by [WL68]. Formally, let $G = (V(G), E(G), \ell)$ be a labeled graph. In each iteration, $t > 0$, the 1-WL computes a node coloring $C^1_t \colon V(G) \to \mathbb{N}$, depending on the coloring of the neighbors. That is, in iteration $t > 0$, we set

$$C^1_t(v) \coloneqq \mathsf{RELABEL}\Big(\big(C^1_{t-1}(v), \{\!\{C^1_{t-1}(u) \mid u \in N(v)\}\!\}\big)\Big),$$

for all nodes $v \in V(G)$, where RELABEL injectively maps the above pair to a unique natural number, which has not been used in previous iterations. In iteration 0, the coloring $C^1_0 \coloneqq \ell$. To test if two graphs G and H are non-isomorphic, we run the above algorithm in "parallel" on both graphs. If the two graphs have a different number of nodes colored $c \in \mathbb{N}$ at some iteration, the 1-WL *distinguishes* the graphs as non-isomorphic. Moreover, if the number of colors between

two iterations, t and $(t+1)$, does not change, i.e., the cardinalities of the images of C^1_t and C^1_{i+t} are equal, or, equivalently,

$$C^1_t(v) = C^1_t(w) \iff C^1_{t+1}(v) = C^1_{t+1}(w),$$

for all nodes v and w in $V(G)$, the algorithm terminates. For such t, we define the *stable coloring* $C^1_\infty(v) = C^1_t(v)$, for v in $V(G)$. The stable coloring is reached after at most $\max\{|V(G)|, |V(H)|\}$ iterations [Gro17]. It is easy to see that the algorithm cannot distinguish all non-isomorphic graphs [CFI92]. Nonetheless, it is a powerful heuristic that can successfully test isomorphism for a broad class of graphs [BK79]. A function $f\colon V(G) \to \mathbb{R}^d$, for $d > 0$, is 1-*WL-equivalent* if $f \equiv C^1_\infty$.

Message-passing graph neural networks Intuitively, MPNNs learn a vectorial representation, i.e., a d-dimensional real-valued vector, representing each vertex in a graph by aggregating information from neighboring vertices. Let $\mathbf{G} = (G, \mathbf{L})$ be an attributed graph, following, [GSR17] and [SGT08b], in each layer, $t > 0$, we compute vertex features

$$\mathbf{h}_v^{(t)} \coloneqq \mathsf{UPD}^{(t)}\Big(\mathbf{h}_v^{(t-1)}, \mathsf{AGG}^{(t)}\big(\{\!\!\{\mathbf{h}_u^{(t-1)} \mid u \in N(v)\}\!\!\}\big)\Big) \in \mathbb{R}^d,$$

where $\mathsf{UPD}^{(t)}$ and $\mathsf{AGG}^{(t)}$ may be differentiable parameterized functions, e.g., neural networks, and $\mathbf{h}_v^{(t)} = \mathbf{L}_v$. In the case of graph-level tasks, e.g., graph classification, one uses

$$\mathbf{h}_G \coloneqq \mathsf{READOUT}\big(\{\!\!\{\mathbf{h}_v^{(T)} \mid v \in V(G)\}\!\!\}\big) \in \mathbb{R}^d,$$

to compute a single vectorial representation based on learned vertex features after iteration T. Again, READOUT may be a differentiable parameterized function, e.g., a neural network. To adapt the parameters of the above three functions, they are optimized end-to-end, usually through a variant of stochastic gradient descent, e.g., [KB15], together with the parameters of a neural network used for classification or regression.

2.3.2 Probabilistically Rewired MPNNs

By leveraging recent progress in differentiable k-subset sampling [AZN23a], we derive *probabilistically rewired MPNNs* (PR-MPNNs). Concretely, we utilize an *upstream model* to learn

prior weights for candidate edges. We then utilize the weights to parameterize a probability distribution constrained by so-called k-subset constraints. Subsequently, we sample multiple k-edge adjacency matrices from this distribution and process them using a *downstream model*, typically an MPNN, for the final predictions task. To make this pipeline trainable via gradient descent, we adapt recently proposed discrete gradient estimation and tractable sampling techniques [AZN23a, NMF21a, XE19a]; see Figure 2.5 for an overview of our architecture. Our theoretical analysis explores how PR-MPNNs overcome MPNNs' inherent limitations in expressive power and identifies precise conditions under which they outperform purely randomized approaches. Empirically, we demonstrate that our approach effectively mitigates issues like over-squashing and under-reaching. In addition, on established real-world datasets, our method exhibits competitive or superior predictive performance compared to traditional MPNN models and graph transformer architectures. Overall, PR-MPNNs pave the way for the principled design of more flexible MPNNs, making them less vulnerable to potential noise and missing information.

Here, we outline PR-MPNNs based on recent advancements in discrete gradient estimation and tractable sampling techniques [AZN23a]. Let $\mathfrak{A}_n$ denote the set of adjacency matrices of n-order graphs. Further, let $(G, \mathbf{X})$ be a n-order attributed graph with an adjacency matrix $\mathbf{A}(G) \in \mathfrak{A}_n$ and node attribute matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, for $d > 0$. A PR-MPNN maintains a (parameterized) *upstream model* $h_{\boldsymbol{v}} \colon \mathfrak{A}_n \times \mathbb{R}^{n \times d} \to \Theta$, typically a neural network, parameterized by $\boldsymbol{v}$, mapping an adjacency matrix and corresponding node attributes to unnormalized edge priors $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^{n \times n}$.

In the following, we use the *priors* $\boldsymbol{\theta}$ as parameters of a (conditional) probability mass function,

$$p_{\boldsymbol{\theta}}(\mathbf{A}(H)) \coloneqq \prod_{i,j=1}^{n} p_{\theta_{ij}}(\mathbf{A}(H)_{ij}),$$

assigning a probability to each adjacency matrix in $\mathfrak{A}_n$, where $p_{\theta_{ij}}(\mathbf{A}(H)_{ij} = 1) = \text{sigmoid}(\theta_{ij})$ and $p_{\theta_{ij}}(\mathbf{A}(H)_{ij} = 0) = 1 - \text{sigmoid}(\theta_{ij})$. Since the parameters $\boldsymbol{\theta}$ depend on the input graph G, we can view the above probability as a conditional probability mass function conditioned on the graph G.

Unlike previous probabilistic rewiring approaches, e.g., [FNP19], we introduce dependencies

between the graph's edges by conditioning the probability mass function $p_{\theta_{ij}}(\mathbf{A}(H))$ on a k-*subset constraint*. That is, the probability of sampling any given k-edge adjacency matrix $\mathbf{A}(H)$, becomes

$$p_{(\boldsymbol{\theta},k)}(\mathbf{A}(H)) \coloneqq \begin{cases} {}^{p_{\boldsymbol{\theta}}(\mathbf{A}(H))}/_{Z} & \text{if } \|\mathbf{A}(H)\|_1 = k, \\ 0 & \text{otherwise,} \end{cases} \quad \text{with} \quad Z \coloneqq \sum_{\mathbf{B}\in\mathfrak{A}_n\colon \|\mathbf{B}\|_1=k} p_{\boldsymbol{\theta}}(\mathbf{B}). \tag{2.10}$$

The original graph G is now rewired into a new adjacency matrix $\bar{\mathbf{A}}$ by combining N samples $\mathbf{A}^{(i)} \sim p_{(\boldsymbol{\theta},k)}(\mathbf{A}(G))$ for $i \in [N]$ together with the original adjacency matrix $\mathbf{A}(G)$ using a differentiable aggregation function $g\colon \mathfrak{A}_n^{(N+1)} \to \mathfrak{A}_n$, i.e., $\bar{\mathbf{A}} \coloneqq g(\mathbf{A}(G), \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}) \in \mathfrak{A}_n$. Subsequently, we use the resulting adjacency matrix as input to a *downstream model* $f_{\mathbf{d}}$, parameterized by $\mathbf{d}$, typically an MPNN, for the final predictions task.

We have so far assumed that the upstream MPNN computes one set of priors $h_{\boldsymbol{v}}\colon \mathfrak{A}_n \times \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times n}$ which we use to generate a new adjacency matrix $\bar{\mathbf{A}}$ through sampling and then aggregating the adjacency matrices $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}$. In Section 7.3, we show empirically that having multiple sets of priors from which we sample is beneficial. Multiple sets of priors mean that we learn an upstream model $h_{\boldsymbol{v}}\colon \mathfrak{A}_n \times \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times n\times M}$ where M is the number of priors. We can then sample and aggregate the adjacency matrices from these multiple sets of priors.

Learning to sample To learn the parameters of the up- and downstream model $\boldsymbol{\omega} = (\boldsymbol{v}, \boldsymbol{u})$ of the PR-MPNN architecture, we minimize the expected loss

$$L(\mathbf{A}(G), \mathbf{X}, y; \boldsymbol{\omega}) \coloneqq \mathbb{E}_{\mathbf{A}^{(i)}\sim p_{(\boldsymbol{\theta},k)}(\mathbf{A}(G))} \left[\ell\left(f_{\boldsymbol{u}}\left(g\left(\mathbf{A}(G), \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}\right), \mathbf{X}\right), y\right)\right],$$

with $y \in \mathcal{Y}$, the targets, ℓ a point-wise loss such as the cross-entropy or MSE, and $\boldsymbol{\theta} = h_{\boldsymbol{v}}(\mathbf{A}(G), \mathbf{X})$. To minimize the above expectation using gradient descent and backpropagation, we need to efficiently draw Monte-Carlo samples from $p_{(\boldsymbol{\theta},k)}(\mathbf{A}(G))$ and estimate $\nabla_{\boldsymbol{\theta}} L$ the gradients of an expectation regarding the parameters $\boldsymbol{\theta}$ of the distribution $p_{(\boldsymbol{\theta},k)}$.

Sampling To sample an adjacency matrix $\mathbf{A}^{(i)}$ from $p_{(\boldsymbol{\theta},k)}(\mathbf{A}(G))$ conditioned on k-edge constraints, and to allow PR-MPNNs to be trained end-to-end, we use SIMPLE [AZN23a], a recently

proposed gradient estimator. Concretely, we can use SIMPLE to sample *exactly* from the k-edge adjacency matrix distribution $p_{(\boldsymbol{\theta},k)}(\mathbf{A}(G))$ on the forward pass. On the backward pass, we compute the approximate gradients of the loss (which is an expectation over a discrete probability mass function) regarding the prior weights $\boldsymbol{\theta}$ using

$$\nabla_{\boldsymbol{\theta}} L \approx \partial_{\boldsymbol{\theta}} \mu(\boldsymbol{\theta}) \nabla_{\mathbf{A}} \ell \;\text{ with }\; \mu(\boldsymbol{\theta}) \coloneqq \{p_{(\boldsymbol{\theta},k)}(\mathbf{A}(G)_{ij})\}_{i,j=1}^{n} \in \mathbb{R}^{n \times n},$$

with an exact and efficient computation of the marginals $\mu(\boldsymbol{\theta})$ that is differentiable on the backward pass, achieving lower bias and variance. We show empirically that SIMPLE [AZN23a] outperforms other sampling and gradient approximation methods such as GUMBEL SOFTSUB-ST [XE19a] and I-MLE [NMF21a], improving accuracy without incurring a computational overhead.

Computational complexity The vectorized complexity of the exact sampling and marginal inference step is $\mathcal{O}(\log k \log l)$, where k is from our k-subset constraint, and l is the maximum number of edges that we can sample. Assuming a constant number of layers, PR-MPNN's worst-case training complexity is $\mathcal{O}(l)$ for both the upstream and downstream models. Let n be the number of nodes in the initial graph, and $l = \max(\{l_{\text{add}}, l_{\text{rm}}\})$, with l_{add} and l_{rm} being the maximum number of added and deleted edges. If we consider all of the possible edges for l_{add}, the worst-case complexity becomes $\mathcal{O}(n^2)$. Therefore, to reduce the complexity in practice, we select a subset of the possible edges using simple heuristics, such as considering the top l_{add} edges of the most distant nodes. During inference, since we do not need gradients for edges not sampled in the forward pass, the complexity is $\mathcal{O}(l)$ for the upstream model and $\mathcal{O}(L)$ for the downstream model, with L being the number of edges in the rewired graph.

2.3.3 Expressive Power of Probabilistically Rewired MPNNs

We now, for the first time, explore the extent to which probabilistic MPNNs overcome the inherent limitations of MPNNs in expressive power caused by the equivalence to 1-WL in distinguishing non-isomorphic graphs [XLT18, MRF19]. Moreover, we identify formal conditions under which PR-MPNNs outperform popular randomized approaches such as those dropping nodes and edges uniformly at random. We first make precise what we mean by probabilistically separating graphs

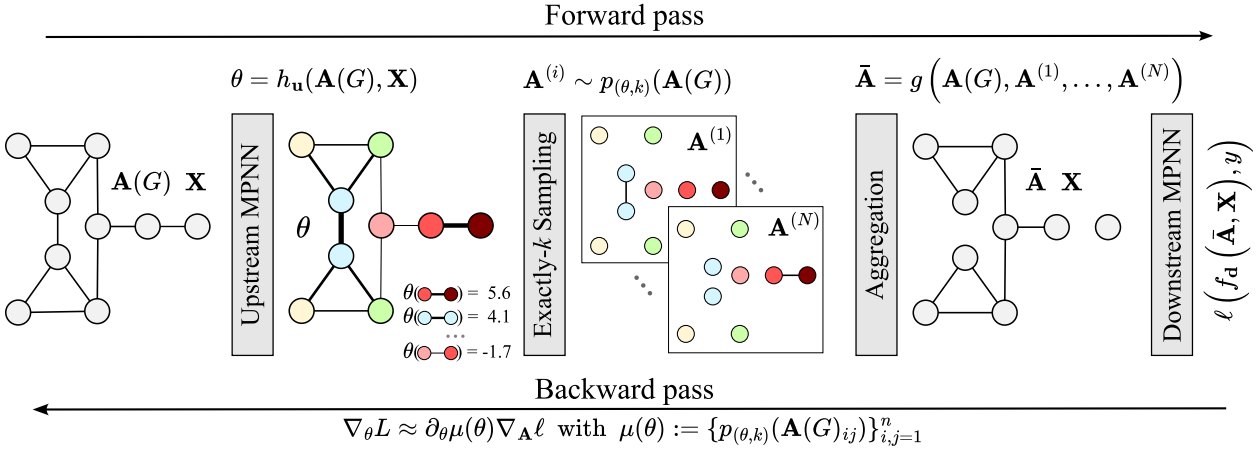


Figure 2.5: Overview of the probabilistically rewired MPNN framework. PR-MPNNs use an *upstream model* to learn priors $\boldsymbol{\theta}$ for candidate edges, parameterizing a probability mass function conditioned on exactly-k constraints. Subsequently, we sample multiple k-edge adjacency matrices (here: $k = 1$) from this distribution, aggregate these matrices (here: subtraction), and use the resulting adjacency matrix as input to a *downstream model*, typically an MPNN, for the final predictions task. On the backward pass, the gradients of the loss ℓ regarding the parameters $\boldsymbol{\theta}$ are approximated through the derivative of the exactly-k marginals in the direction of the gradients of the point-wise loss ℓ regarding the sampled adjacency matrix. We use recent work to make the computation of these marginals exact and differentiable, reducing both bias and variance.

by introducing a probabilistic and generally applicable notion of graph separation.

Let us assume a conditional probability mass function $p\colon \mathfrak{A}_n \to [0, 1]$ conditioned on a given n-order graph, defined over the set of adjacency matrices of n-order graphs. In the context of PR-MPNNs, p is the probability mass function defined in Section 2.3.2 but it could also be any other conditional probability mass function over graphs. Moreover, let $f\colon \mathfrak{A}_n \to \mathbb{R}^d$, for $d > 0$, be a permutation-invariant, parameterized function mapping a sampled graph's adjacency matrix to a vector in $\mathbb{R}^d$. The function f could be the composition of an aggregation function g that removes the sampled edges from the input graph G and of a downstream MPNN. Now, the conditional

probability mass function p *separates* two graphs G and H with probability ρ *with respect to* f if

$$\mathbb{E}_{\bar{G}\sim p(\cdot|G),\bar{H}\sim p(\cdot|H)}\left[f(\mathbf{A}(\bar{G})) \neq f(\mathbf{A}(\bar{H}))\right] = \rho,$$

that is, if in expectation over the conditional probability distribution, the vectors $f(\mathbf{A}(\bar{G}))$ and $f(\mathbf{A}(\bar{H}))$ are distinct with probability ρ.

In what follows, we analyze the case of p being the exactly-k probability distribution defined in Equation 2.10 and f being the aggregation function removing edges and a downstream MPNN. However, our framework readily generalizes to the case of node removal. Following Section 2.3.2, we sample adjacency matrices with exactly k edges and use them to remove edges from the original graph. We aim to understand the separation properties of the probability mass function $p_{(k,\boldsymbol{\theta})}$ in this setting and for various types of graph structures. Most obviously, we do not want to separate isomorphic graphs and, therefore, remain isomorphism invariant, a desirable property of MPNNs.

Theorem 2. *For sufficiently large n, for every $\varepsilon \in (0,1)$ and $k > 0$, we have that for almost all pairs, in the sense of [BES80], of isomorphic n-order graphs G and H and all permutation-invariant, 1-WL-equivalent functions $f\colon \mathfrak{A}_n \to \mathbb{R}^d$, $d > 0$, there exists a probability mass function $p_{(\boldsymbol{\theta},k)}$ that separates the graph G and H with probability at most ε with respect to f.*

Theorem 2 relies on the fact that most graphs have a discrete 1-WL coloring. For graphs where the 1-WL stable coloring consists of a discrete and non-discrete part, the following result shows that there exist distributions $p_{(\boldsymbol{\theta},k)}$ not separating the graphs based on the partial isomorphism corresponding to the discrete coloring.

Proposition 4. *Let $\varepsilon \in (0,1)$, $k > 0$, and let G and H be graphs with identical 1-WL stable colorings. Let V_G and V_H be the subset of nodes of G and H that are in color classes of cardinality 1. Then, for all choices of 1-WL-equivalent functions f, there exists a conditional probability distribution $p_{(\boldsymbol{\theta},k)}$ that separates the graphs $G[V_G]$ and $H[V_H]$ with probability at most ε with respect to f.*

Existing methods such as DropGNN [PMF21] or DropEdge [RHX20] are more likely to separate two (partially) isomorphic graphs by removing different nodes or edges between discrete

color classes, i.e., on their (partially) isomorphic subgraphs. For instance, we prove that pairs of graphs with m edges exist where the probability of non-separation under uniform edge sampling is at most $1/m$. This is undesirable as it breaks the MPNNs' permutation-invariance in these parts.

Now that we have established that distributions with priors from upstream MPNNs are more likely to preserve (partial) isomorphism between graphs, we turn to analyze their behavior in separating the non-discrete parts of the coloring. The following theorem shows that PR-MPNNs are more likely to separate non-isomorphic graphs than probability mass functions that remove edges or nodes uniformly at random.

Theorem 3. *For every $\varepsilon \in (0, 1)$ and every $k > 0$, there exists a pair of non-isomorphic graphs G and H with identical and non-discrete 1-WL stable colorings such that for every 1-WL-equivalent function f,*

(1) there exists a probability mass function $p_{(k,\boldsymbol{\theta})}$ that separates G and H with probability at least $(1 - \varepsilon)$ with respect to f;

(2) removing edges uniformly at random separates G and H with probability at most ε with respect to f.

Finally, we can also show a negative result, namely that there exist classes of graphs for which PR-MPNNs cannot do better than random sampling.

Proposition 5. *For every $k > 0$, there exist non-isomorphic graphs G and H with identical 1-WL colorings such that every probability mass function $p_{(\boldsymbol{\theta},k)}$ separates the two graphs with the same probability as the distribution that samples edges uniformly at random.*

Related Work

MPNNs are inherently biased towards encoding local structures, which limits their expressive power [MRF19, MLM21, XHL19]. Specifically, they are at most as powerful as distinguishing

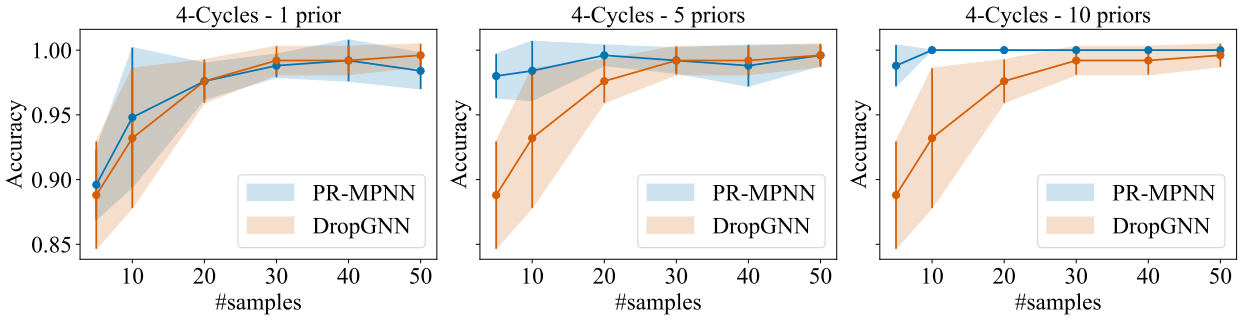


Figure 2.6: Comparison between PR-MPNN and DropGNN on the 4-Cycles dataset. PR-MPNN rewiring is almost always better than randomly dropping nodes, and is always better with 10 priors.

non-isomorphic graphs or nodes with different structural roles as the 1-*dimensional Weisfeiler–Leman algorithm* [WL68], a simple heuristic for the graph isomorphism problem; see Section 2.3.1. Additionally, they cannot capture global or long-range information, often linked to phenomena such as under-reaching [BKM20] or over-squashing [AY21], with the latter being heavily investigated in recent works.

Graph rewiring Several recent works aim to circumvent over-squashing via graph rewiring. Perhaps the most straightforward way of graph rewiring is incorporating multi-hop neighbors. For example, [BYS22] rewires the graphs with k-hop neighbors and virtual nodes and also augments them with positional encodings. MixHop [APK19], SIGN [FRE20], DIGL [GWG19], and SP-MPNN [ADC22] can also be considered as graph rewiring as they can reach further-away neighbors in a single layer. Particularly, [GDB23] rewires the graph similarly to [ADC22] but with a novel delay mechanism, showcasing promising empirical results. Several rewiring methods depend on particular metrics, e.g., Ricci or Forman curvature [BMS22] and balanced Forman curvature [TDC21]. In addition, [DLV22, SVV23] utilize expander graphs to enhance message passing and connectivity, while [KBM22] resort to spectral techniques, and [BKW22] propose a greedy random edge flip approach to overcome over-squashing. Refining [TDC21], [DGB23] analyzed how the architectures' width and graph structure contribute to the over-squashing problem,

showing that over-squashing happens among nodes with high commute time, stressing the importance of rewiring techniques. Contrary to our proposed method, these strategies to mitigate over-squashing either rely on heuristic rewiring methods or use purely randomized approaches that may not adapt well to a given prediction task. Furthermore, the impact of existing rewiring methods on a model's expressive power remains unclear and we close this gap with our work.

Graph transformers Different from the above, graph transformers [DRG22, HHL23, MGM23, RGD22, COB22] and similar global attention mechanisms [LWJ21, WJW21] marked a shift from local to global message passing, aggregating over all nodes. While not understood in a principled way, empirical studies indicate that graph transformers possibly alleviate over-squashing; see, e.g., [MGM23]. However, all transformers suffer from their quadratic space and memory requirements due to computing an attention matrix.

2.3.4 Empirical Evaluation

We explore to what extent our probabilistic graph rewiring leads to improved predictive performance on synthetic and real-world datasets. Concretely, we answer the following questions.

Q1 Can probabilistic graph rewiring mitigate the problems of over-squashing and under-reaching in synthetic datasets?

Q2 Is the expressive power of standard MPNNs enhanced through probabilistic graph rewiring? That is, can we verify empirically that the separating probability mass function of Section 2.3.3 can be learned with PR-MPNNs and that multiple priors help?

Q3 Does the increase in predictive performance due to probabilistic rewiring apply to (a) graph-level molecular prediction tasks and (b) node-level prediction tasks involving heterophilic data?

Datasets To answer **Q1**, we utilized the Trees-NeighborsMatch dataset [AY21]. Additionally, we created the Trees-LeafCount dataset to investigate whether our method could mitigate under-reaching issues. To tackle **Q2**, we performed experiments with the Exp [ACG20] and CSL

datasets [MSR19] to assess how much probabilistic graph rewiring can enhance the models' expressivity. In addition, we utilized the 4-Cycles dataset from [Lou20, PMF21] and set it against a standard DropGNN model [PMF21] for comparison while also ablating the performance difference concerning the number of priors and samples per prior. To answer **Q3** (a), we used the established molecular graph-level regression datasets Alchemy [CCH19], Zinc [JCB17, DJL20], ogbg-molhiv [HFZ20], QM9 [HYL17], LRGB [DRG22] and five datasets from the TUDataset repository [MKB20]. To answer **Q3** (b), we used the Cornell, Wisconsin, Texas node-level classification datasets [PWC20].

Baseline and model configurations For our upstream model $h_{\boldsymbol{v}}$, we use an MPNN, specifically the GIN layer [XHL19]. For an edge $(v, w) \in E(G)$, we compute $\boldsymbol{\theta}_{vw} = \phi([\mathbf{h}_v^T || \mathbf{h}_w^T]) \in \mathbb{R}$, where $[\cdot||\cdot]$ is the concatenation operator and ϕ is an MLP. After obtaining the prior $\boldsymbol{\theta}$, we rewire our graphs by sampling two adjacency matrices for deleting edges and adding new edges, i.e., $g(\mathbf{A}(G), \mathbf{A}^{(1)}, \mathbf{A}^{(2)}) \coloneqq (\mathbf{A}(G) - \mathbf{A}^{(1)}) + \mathbf{A}^{(2)}$ where $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ are two sampled adjacency matrices with a possibly different number of edges, respectively. Finally, the rewired adjacency matrix (or multiple adjacency matrices) is used in a *downstream model* $f_{\boldsymbol{u}} \colon \mathfrak{A}_n \times \mathbb{R}^{n \times d} \to \mathcal{Y}$, typically an MPNN, with parameters $\boldsymbol{u}$ and $\mathcal{Y}$ the prediction target set. For the instance where we have multiple priors, as described in Section 2.3.2, we can either aggregate the sampled adjacency matrices $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}$ into a single adjacency matrix $\bar{\mathbf{A}}$ that we send to a downstream model as described in Figure 2.5, or construct a downstream ensemble with multiple aggregated matrices $\bar{\mathbf{A}}_1, \ldots, \bar{\mathbf{A}}_M$. In practice, we always use a downstream ensemble before the final projection layer when we rewire with more than one adjacency matrix, and we do rewiring by both adding and deleting edges.

All of our downstream models $f_{\mathbf{d}}$ and base models are MPNNs with GIN layers. When we have access to edge features, we use the GINE variant [HLG20] for edge feature processing. For graph-level tasks, we use mean pooling, while for node-level tasks, we take the node embedding $\mathbf{h}_v^T$ for a node v. The final embeddings are then processed and projected to the target space by an MLP.

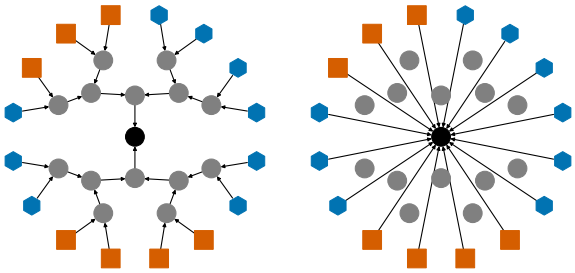


Figure 2.7: Example graph from the TREES-LEAFCOUNT test dataset with radius 4 (left). PR-MPNN rewires the graph, allowing the downstream MPNN to obtain the label information from the leaves in one massage-passing step (right).

For ZINC, ALCHEMY, and OGBG-MOLHIV, we compare our rewiring approaches with the base downstream model, both with and without positional embeddings. Further, we compare to GPS [RGD22] and SAT [COB22], two state-of-the-art graph transformers. For the TUDATASET, we compare with the reported scores from [GRC23] and use the same evaluation strategy as in [XHL19, GRC23], i.e., running 10-fold cross-validation and reporting the maximum average validation accuracy. For different tasks, we search for the best hyperparameters for sampling and our upstream and downstream models. For ZINC, ALCHEMY, and OGBG-MOLHIV, we evaluate multiple gradient estimators in terms of predictive power and computation time. Specifically, we compare GUMBEL SOFTSUB-ST [MMT17a, JGP17a, XE19a], I-MLE [NMF21a], and SIMPLE [AZN23a]. The results in terms of predictive power are detailed in Table 2.4.

Experimental results and discussion Concerning **Q1**, our rewiring method achieves perfect test accuracy up to a problem radius of 6 on both TREES-NEIGHBORSMATCH and TREES-LEAFCOUNT, demonstrating that it can successfully alleviate over-squashing and under-reaching, see Figure 2.8. For TREES-LEAFCOUNT, our model can create connections directly from the leaves to the root, achieving perfect accuracy with a downstream model containing a single MPNN layer. We provide a qualitative result in Figure 2.7. Concerning **Q2**, on the 4-CYCLES dataset, our probabilistic rewiring method matches or outperforms DropGNN. This advantage is most pronounced with 5 and 10 priors, where we achieve 100% task accuracy using 20 samples, as

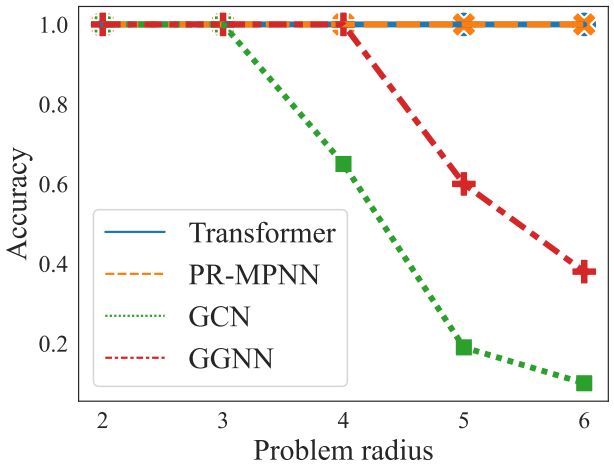


Figure 2.8: Test accuracy of our rewiring method on the TREES-NEIGHBORSMATCH [AY21] dataset, compared to the reported accuracies from [MGM23].

detailed in Figure 2.6. On the EXP dataset, we showcase the expressive power of probabilistic rewiring by achieving perfect accuracy, see Table 2.5. Besides, our rewiring approach can distinguish the regular graphs from the CSL dataset without any positional encodings, whereas the 1-WL-equivalent GIN obtains only random accuracy. Concerning **Q3** (a), the results in Table 2.4 show that our rewiring methods consistently outperform the base models on ZINC, ALCHEMY, and OGBG-MOLHIV and are competitive or better than the state-of-the-art GPS and SAT graph transformer methods. On TUDATASET, see Table 2.7, our probabilistic rewiring method outperforms existing approaches and obtains lower variance on most of the datasets, with the exception being NCI1, where our method ranks second, after the WL kernel. Hence, our results indicate that probabilistic graph rewiring can improve performance for molecular prediction tasks. Concerning **Q3** (b), we obtain performance gains over the base model and other existing MPNNs, indicating that data-driven rewiring has the potential of alleviating the *effects* of over-smoothing by removing undesirable edges and making new ones between nodes with similar features. The graph transformer methods outperform the rewiring approach and the base models, except on the TEXAS dataset, where our method gets the best result. We speculate that GIN's aggregation mechanism for the downstream models is a limiting factor on heterophilic data. We leave the analysis of

combining probabilistic graph rewiring with downstream models that address over-smoothing for future investigations.

2.4 Discussion

The key to the success of SIMPLE is the capability of exact reasoning over the k-subset constraint as the constraint probabilities of the exact samples from the constrained distribution serve as an informative proxy for gradient updates. This paradigm can be potentially generalized to any constraints; that is, the problem of differentiable learning under constraints can be reduced to the problem of tractable computation of the constraint probability. An interesting direction for future work is to come up with efficient approximations of SIMPLE for complex constraints.

Table 2.4: Comparison between PR-MPNN and baselines on three molecular property prediction datasets. We report results for PR-MPNN with different gradient estimators for k-subset sampling: Gumbel SoftSub-ST [MMT17a, JGP17a, XE19a], I-MLE [NMF21a], and Simple [AZN23a] and compare them with the base downstream model, and two graph transformer architectures. The variant using SIMPLE consistently outperforms the base models and is competitive or better than the two graph transformers. We use **green** for the best model, **blue** for the second-best, and **red** for third. We note with + Edge the instances where edge features are provided and with - Edge when they are not.

		ZINC		OGBG-MOLHIV	ALCHEMY
		- EDGE ↓	+ EDGE ↓	+ EDGE ↑	+ EDGE ↓
GIN BACKBONE	K-ST SAT	0.166±0.007	0.115±0.005	0.625±0.039	N/A
	K-SG SAT	0.162±0.013	**0.095**±0.002	0.613±0.010	N/A
	BASE	0.258±0.006	0.207±0.006	**0.775**±0.011	11.12±0.690
	BASE W. PE	0.162±0.001	0.101±0.004	0.764±0.018	7.197±0.094
	PR-MPNN$_{\text{GMB}}$ (OURS)	0.153±0.003	0.103±0.008	0.760±0.025	**6.858**±0.090
	PR-MPNN$_{\text{IMLE}}$ (OURS)	**0.151**±0.001	0.104±0.008	**0.774**±0.015	**6.692**±0.061
	PR-MPNN$_{\text{SIM}}$ (OURS)	**0.139**±0.001	**0.085**±0.002	**0.795**±0.009	**6.447**±0.057
PNA	GPS	N/A	**0.070**±0.004	**0.788**±0.010	N/A
	K-ST SAT	0.164±0.007	0.102±0.005	0.625±0.039	N/A
	K-SG SAT	**0.131**±0.002	**0.094**±0.008	0.613±0.010	N/A

Table 2.5: Comparison between the base GIN model, PR-MPNN, and other more expressive models on the Exp dataset.

Model	Accuracy ↑
GIN	$0.511_{\pm 0.021}$
GIN + ID-GNN	$1.000_{\pm 0.000}$
OSAN	$1.000_{\pm 0.000}$
PR-MPNN (ours)	$1.000_{\pm 0.000}$

Table 2.6: Comparison between the base GIN model and probabilistic rewiring model on CSL dataset, w/o positional encodings.

Model	Accuracy ↑
GIN	$0.100_{\pm 0.000}$
GIN + PosEnc	$1.000_{\pm 0.000}$
PR-MPNN (ours)	$0.998_{\pm 0.008}$
PR-MPNN + PosEnc (ours)	$1.000_{\pm 0.000}$

Table 2.7: Comparison between PR-MPNN and other approaches as reported in [GRC23, KBM22, PMF21]. Our model outperforms existing approaches while keeping a lower variance in most of the cases, except for NCI1, where the WL Kernel is the best. We use **green** for the best model, **blue** for the second-best, and **red** for third.

Model	Mutag	PTC_MR	Proteins	NCI1	NCI109
GK ($k = 3$) [SVP09]	81.4±1.7	55.7±0.5	71.4±0.3	62.5±0.3	62.4±0.3
PK [NGB16]	76.0±2.7	59.5±2.4	73.7±0.7	82.5±0.5	N/A
WL kernel [SSV11]	90.4±5.7	59.9±4.3	75.0±3.1	**86.0**±1.8	N/A
DGCNN [ZCN18]	85.8±1.8	58.6±2.5	75.5±0.9	74.4±0.5	N/A
IGN [MBS19b]	83.9±13.0	58.5±6.9	76.6±5.5	74.3±2.7	72.8±1.5
GIN [XHL19]	89.4±5.6	64.6±7.0	76.2±2.8	82.7±1.7	N/A
PPGNs [MBS19a]	90.6±8.7	66.2±6.6	77.2±4.7	83.2±1.1	82.2±1.4
Natural GN [HCW20]	89.4±1.6	66.8±1.7	71.7±1.0	82.4±1.3	83.0±1.9
GSN [BFZ22]	92.2±7.5	68.2±7.2	76.6±5.0	83.5±2.0	83.5±2.3
CIN [BFW21]	92.7±6.1	68.2±5.6	77.0±4.3	83.6±1.4	**84.0**±1.6
CAN [GBT23]	**94.1**±4.8	**72.8**±8.3	**78.2**±2.0	84.5±1.6	83.6±1.2
CIN++ [GRC23]	**94.4**±3.7	**73.2**±6.4	**80.5**±3.9	**85.3**±1.2	**84.5**±2.4
FoSR [KBM22]	86.2±1.5	58.5±1.7	75.1±0.8	72.9±0.6	71.1±0.6
DropGNN [PMF21]	90.4±7.0	66.3±8.6	76.3±6.1	81.6±1.8	80.8±2.6
PR-MPNN (10-fold CV)	**98.4**±2.4	**74.3**±3.9	**80.7**±3.9	**85.6**±0.8	**84.6**±1.2

CHAPTER 3

Linear-Equality Constrained Deep Generative Models

This chapter focuses on the same challenge of differentiable learning under constraints as the previous chapter but extends from the discrete domains to the continuous ones, from k-subset constraint to linear-equality constraints. Enforcing linear-equality constraints poses unique challenges to the design of gradient estimators. Our aim is to tackle these complexities, demonstrating that our methods, while broadly applicable to any neural networks, enable robust and efficient enforcement of constraints, particularly in deep generative models.

3.1 Background

The linear equality constraint is ubiquitous not only in machine learning such as learning sparse features [CSW18] but also in scientific applications such as modeling charge-neutral molecules in chemistry [RSS20] and count-aware cell type deconvolution in biology [LWZ23]. Even though deep generative models (DGMs) have made great progress in synthesizing realistic data in various domains, they struggle to learn such constraints from data while these constraints can encode necessary domain knowledge of the task at hand. For example, a DGM trained on a charge-neutral molecule dataset often generates molecules where the charges of each atom do not sum to zero. That is, DGMs are excellent at approximating complex distributions but can fail to capture simple constraints [SDC24]. However, such constraints should be integrated into the DGMs to generate realistic data guaranteed to follow the domain knowledge.

While its discrete counterpart, known as k-subset constraints, has been thoroughly studied for being enforced into deep learning models [XE19c, NMF21b, AZN23a], how to integrate linear

equality into DGMs with continuous or discrete domains is underexplored. [AAB19, FNC20] study the problem when the linear equalities are constraints to the optimization problems instead of data distributions. While one can ignore the constraints in the training and use post-processing of the generated data, it harms the generation performance. [SDC24] considers linear inequalities enforced on DGMs by minimally moving the sample into the feasible space defined by the linear inequalities but it does not apply to linear equalities.

3.2 Gradient Estimator for Linear Equalities

This section shows how to integrate linear constraints that encode background knowledge into different DGMs. We start by characterizing the settings when the training objective admits a closed-form expression even though the data distribution is constrained, and thus the DGM allows end-to-end training as in standard settings. We show in the experiment section that such scenarios are common in real-world applications. For the more general settings when no closed-from objectives are available, we propose several gradient estimators by the principle that constrained marginal probabilities can serve as an informative proxy for gradient updates as observed in [NMF21b, AZN23a] and provide a comparison of their bias and variance. The proposed estimators enable linear equality to be enforced not just on the data distribution but also on the distribution of latent variables. With our method, DGMs are transformed into the corresponding constrained DGMs that generate data guaranteed to satisfy the linear constraints.

To evaluate our approach, we modify the MNIST dataset to be constrained by a constant sum of pixel values and conduct an extensive experimental analysis to compare DGMs and their constrained counterparts. Results show that DGMs often fail to satisfy the constraints, that is, they are incapable of learning the domain knowledge. Their constrained counterparts, on the contrary, always generate compliant samples and have better generation quality due to the inductive bias from the constraints. By further comparing the training time, we demonstrate that the injection of the constraints brings little overhead. We also include a real-world experiment on a scientific application where the task is to predict the atomic partial charges on metal-organic frameworks

under a charge-neutral constraint where our method achieves state-of-the-art predictive performance.

Problem Statement We consider a general constrained model described by the equations

$$\boldsymbol{\theta} = h_{\boldsymbol{v}}(\boldsymbol{x}), \qquad \boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \textstyle\sum_i z_i = k), \qquad \hat{\boldsymbol{y}} = f_{\boldsymbol{u}}(\boldsymbol{z}), \tag{3.1}$$

where $\boldsymbol{x} \in \mathcal{X}$ and $\hat{\boldsymbol{y}} \in \mathcal{Y}$ denote feature inputs and target outputs, respectively, $h_{\boldsymbol{v}} : \mathcal{X} \to \Theta$ and $f_{\boldsymbol{u}} : \mathcal{Z} \to \mathcal{Y}$ are smooth, parameterized mappings. $\boldsymbol{\theta}$ are parameters inducing a distribution over the latent variables $\boldsymbol{z}$ conform to a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where parameters $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ consist of the mean vector $\boldsymbol{\mu} \in \mathbb{R}^n$ and the covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$. That is, $\boldsymbol{z}$ has its probability density function (p.d.f.) defined as $p_{\boldsymbol{\theta}}(\boldsymbol{z}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}_{\boldsymbol{\theta}}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{z} - \boldsymbol{\mu}_{\boldsymbol{\theta}})^\top \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{z} - \boldsymbol{\mu}_{\boldsymbol{\theta}})\right)$. A linear equality $\sum_i z_i = k$ with $k \in \mathbb{R}$ is enforced over the distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ inducing a conditional distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_i z_i = k)$. This formulation is general and it subsumes various neural network models that integrate the linear equality constraint in *1) output* (when the mapping $f_{\boldsymbol{u}}$ is the identity function), where the goal of constrained generative modeling is to learn the parameters $\boldsymbol{\theta}$ such that the constrained model distribution $p_{\boldsymbol{\theta}}$ approximates the underlying data distribution; or *2) latent space*, where the constrained generative modeling learns a constrained posterior distribution $p_{\boldsymbol{\theta}}$ over latent variables. The training of such models is performed by optimizing an expected loss as below,

$$L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}) = \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_i z_i = k)}[\ell(f_{\boldsymbol{u}}(\boldsymbol{z}), \boldsymbol{y})] \qquad \textit{with } \boldsymbol{\omega} = (\boldsymbol{v}, \boldsymbol{u}) \textit{ and } \boldsymbol{\theta} = h_{\boldsymbol{v}}(\boldsymbol{x}), \tag{3.2}$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is a point-wise loss function. Figure 2.2 shows a visualization of the model. Note that in this work we formulate the linear equality constraint as an unweighted sum of variables for ease of notation while all of our theoretical results can be extended to a weighted sum of variables of the form $\sum_i \alpha_i z_i = k$ with $\alpha_i \in \mathbb{R}$ for any i. We explore other the distributional assumptions and present results on Poisson distributions for discrete variables with infinite domain in Section 3.2.2.3.

3.2.1 Closed-Form Expected Loss

Before we tackle the challenging task of deriving gradient estimators for training DGMs, it is crucial to first address a fundamental question: when is a gradient estimator necessary, and *when is it not*? A short answer is that when the expected loss in Equation 3.2 admits a closed-form expression, standard training can be applied to the constrained DGMs and thus no gradient estimator is needed. This section presents our theoretical results on characterizing when such closed forms are available.

The first assumption we make is that the mapping $f_{\boldsymbol{u}}$ is an identity function, that is, $\hat{\boldsymbol{y}} = \boldsymbol{z}$, as it can introduce high non-linearity that easily makes a closed-form impossible. Then we show that when the element-wise loss ℓ is $L1$ or $L2$ loss, the expected loss admits a closed-form expression as below.

Proposition 6 (Gaussian Closed-form Expected Loss)**.** *Let $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Let $\boldsymbol{y} = (y_1, \ldots, y_n)^T$ be the ground truth vector subject to the equality constraint $\sum_{j=1}^{n} y_j = k$. Then it holds that*

i) when ℓ is L1 loss, $L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \overline{\sigma}_i \sqrt{\frac{2}{\pi}} \exp\left(\frac{-(\overline{\mu}_i - y_i)^2}{2\overline{\sigma}_i^2}\right) + (\overline{\mu}_i - y_i)\, erf\left(\frac{\overline{\mu}_i - y_i}{\sqrt{2\overline{\sigma}_i^2}}\right)$;

ii) when ℓ is L2 loss, $L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \overline{\mu}_i^2 + \overline{\sigma}_i^2 - 2y_i\overline{\mu}_i + y_i^2$,

with $\overline{\mu}_i := \boldsymbol{\mu}_i + \frac{\boldsymbol{\Sigma}_{i,i} + \sum_{j \neq i}^{n} \boldsymbol{\Sigma}_{i,j}}{\sum_{t=1}^{n} \boldsymbol{\Sigma}_{t,t} + \sum_{j \neq t}^{n} \boldsymbol{\Sigma}_{t,j}} \left(k - \sum_{j=1}^{n} \boldsymbol{\mu}_j\right)$ and $\overline{\sigma}_i^2 := \boldsymbol{\Sigma}_{i,i} - \frac{\left(\boldsymbol{\Sigma}_{i,i} + \sum_{j \neq i}^{n} \boldsymbol{\Sigma}_{i,j}\right)^2}{\sum_{t=1}^{n} \boldsymbol{\Sigma}_{t,t} + \sum_{j \neq t}^{n} \boldsymbol{\Sigma}_{t,j}}$.

We further generalize our results to a system of linear equality constraints.

Corollary 1. *The closed-form expressions for $L(\boldsymbol{\theta})$ defined under L1 loss and L2 loss under the constraint $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{k}$, where $\boldsymbol{A} \in \mathbb{R}^{a \times n}$ and rank$(\boldsymbol{A}) = a < n$, are identical to those presented in Proposition 6, except that the parameters are defined as follows,*

$$\overline{\mu}_i = \mu_i + \boldsymbol{e}_i^T \boldsymbol{\Sigma} \boldsymbol{A} \left(\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^T\right)^{-1} (\boldsymbol{k} - \boldsymbol{A}\boldsymbol{\mu}) \quad \text{and} \quad \overline{\sigma}_i^2 = \boldsymbol{e}_i^T \boldsymbol{\Sigma} \boldsymbol{e}_i - \boldsymbol{e}_i^T \boldsymbol{\Sigma} \boldsymbol{A}^T \left(\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^T\right)^{-1} \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{e}_i$$

where $\boldsymbol{e}_i$ is a standard basis vector in $\mathbb{R}^n$ with only i-th entry being 1 and otherwise 0.

Later we will empirically show that the closed-form expected loss derived above leads to state-of-the-art predictive performance in a scientific application as in Section 3.2.3.4.

3.2.2 Method

As we move on to the general setting where the expected loss in Equation 3.2 does not admit a closed form, standard auto-differentiation can not be directly applied to the expected loss due to two main obstacles. First, for the gradient of L w.r.t. parameters $\boldsymbol{u}$ in the decoder mapping $f_{\boldsymbol{u}}$ defined as

$$\nabla_{\boldsymbol{u}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}) = \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z} | \sum_i z_i = k)} [\partial_{\boldsymbol{u}} f_{\boldsymbol{u}}(\boldsymbol{z}, \boldsymbol{x})^\top \nabla_{\hat{\boldsymbol{y}}} \ell(\hat{\boldsymbol{y}}, \boldsymbol{y})] \tag{3.3}$$

with $\hat{y} = f_{\boldsymbol{u}}(\boldsymbol{z})$ being the decoding of a latent sample $\boldsymbol{z}$, the expectation does not allow closed-form solution in general and requires Monte-Carlo estimations by sampling $\boldsymbol{z}$ from the constrained distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_i z_i = k)$. Another issue arises in the gradient of L w.r.t. parameters $\boldsymbol{v}$ in the encoder mapping which is defined as

$$\nabla_{\boldsymbol{v}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}) = \partial_{\boldsymbol{v}} h_{\boldsymbol{v}}(\boldsymbol{x})^\top \nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}) \tag{3.4}$$

where the obstacle lies in the computation of the gradient of L w.r.t. $\boldsymbol{\theta}$ defined as $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}) := \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z} | \sum_i z_i = k)} [\ell(f_{\boldsymbol{u}}(\boldsymbol{z}, \boldsymbol{x}), \hat{\boldsymbol{y}})]$ that requires gradient estimators. In this section, we tackle the gradient estimation for the linear equality constraint by solving the aforementioned two subproblems: **(P1)** how to *sample exactly* from the constrained distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_i z_i = k)$ and **(P2)** how to *estimate* $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega})$. By combining solutions to these two subproblems, we manage to train the constrained DGMs in an end-to-end manner. We summarize our proposed estimators in Table 3.1.

3.2.2.1 Exact Sampling

We find that the constrained distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_i z_i = k)$ is still a Gaussian distribution and thus it is straightforward to sample exactly from this distribution as long as we obtain the mean vector and the covariance matrix of this Gaussian. We formally state our findings as below.

Proposition 7 (Gaussian Constrained Distribution)**.** *Given* $\boldsymbol{z} = (z_1, \ldots, z_n)^T \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$*, the constrained distribution* $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_{j=1}^n z_j = k)$ *is equivalent to an* $n-1$ *dimensional multivariate*

Table 3.1: Summary of gradient estimators. The first block presents baseline estimators and the second presents our proposed ones. In the forward pass, we sample exactly from the constrained distribution by Proposition 7. In the backward pass, we use $\boldsymbol{m}(\boldsymbol{\theta})$ as a differntiable proxy.

Gradient Estimator	**Proxy** $\boldsymbol{m}(\boldsymbol{\theta})$	**Description**
Random	-	Sample a random gradient from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$
Unconstrained Marginal	$p_{\boldsymbol{\theta}}(z_i)$	P.d.f. of unconstrained $\boldsymbol{z}$ as a proxy for $\boldsymbol{z}$
Constrained Marginal	$p_{\boldsymbol{\theta}}\left(z_i \middle\vert \sum_{j=1}^n z_j = k\right)$	P.d.f. of conditional marginals as a proxy for $\boldsymbol{z}$
Marginal Expectation	$\mathbb{E}_{z_i \sim p_{\boldsymbol{\theta}}(z_i \vert \sum_{j=1}^n z_j = k)}[z_i]$	Expectation of conditional marginals as a proxy for $\boldsymbol{z}$

Gaussian distribution with mean $\overline{\boldsymbol{\mu}} \in \mathbb{R}^{n-1}$ *and covariance matrix* $\overline{\boldsymbol{\Sigma}} \in \mathbb{R}^{n-1 \times n-1}$ *defined as below,*

$$\overline{\boldsymbol{\mu}}_i = \boldsymbol{\mu}_i + \frac{\boldsymbol{\Sigma}_{i,i} + \sum_{j \neq i}^n \boldsymbol{\Sigma}_{i,j}}{\sum_{t=1}^n \boldsymbol{\Sigma}_{t,t} + \sum_{j \neq t}^n \boldsymbol{\Sigma}_{t,j}} \left(k - \sum_{j=1}^n \boldsymbol{\mu}_j\right)$$

$$\overline{\boldsymbol{\Sigma}}_{i,j} = \begin{cases} \boldsymbol{\Sigma}_{i,i} - \frac{\left(\boldsymbol{\Sigma}_{i,i} + \sum_{j \neq i}^n \boldsymbol{\Sigma}_{i,j}\right)^2}{\sum_{t=1}^n \boldsymbol{\Sigma}_{t,t} + \sum_{j \neq t}^n \boldsymbol{\Sigma}_{t,j}} & i = j \\ \boldsymbol{\Sigma}_{i,j} - \frac{\left(\boldsymbol{\Sigma}_{i,i} + \sum_{j \neq i}^n \boldsymbol{\Sigma}_{i,j}\right)\left(\boldsymbol{\Sigma}_{j,j} + \sum_{k \neq j}^n \boldsymbol{\Sigma}_{j,k}\right)}{\sum_{t=1}^n \boldsymbol{\Sigma}_{t,t} + \sum_{j \neq t}^n \boldsymbol{\Sigma}_{t,j}} & i \neq j \end{cases}.$$

3.2.2.2 Gradient Estimators

[NMF21b, AZN23a] observe that in the discrete case when $\boldsymbol{z}$ are Bernoulli variables, the problematic term in Equation 3.4 can be effectively approximated as

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}) \approx \partial_{\boldsymbol{\theta}} \boldsymbol{m}(\boldsymbol{\theta}) \nabla_{\boldsymbol{z}} \ell(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\omega}), \tag{3.5}$$

where $\boldsymbol{m}(\boldsymbol{\theta}) \coloneqq \{p_{\boldsymbol{\theta}}(z_i \mid \sum_{j=1}^n z_i = k)\}_{j=1}^n$ denotes the conditional marginal probabilities. The intuition behind this is that the conditional marginals characterize the probability of each z_i in the generated sample and can serve as a differentiable proxy allowing for end-to-end training. We generalize this intuition to the linear-equality constrained distribution in the continuous domains and perform an analysis of its effectiveness. The continuous setting mostly differs from

the discrete one in that there are two choices of $\boldsymbol{m}(\boldsymbol{\theta})$: 1) the conditional marginal probability density $p_{\boldsymbol{\theta}}(z_i|\sum_{j=1}^n z_j = k)$; and 2) the expectation of z_i under the conditional marginal, that is, $\mathbb{E}_{z_i\sim p_{\boldsymbol{\theta}}(z_i|\sum_{j=1}^n z_j=k)}[z_i]$.

While these two quantities are the same in discrete cases, we make an interesting observation in our empirical study that in continuous domains, the use of expectation is consistently more effective than the conditional marginals. We further provide a baseline estimator that chooses $\boldsymbol{m}(\boldsymbol{\theta})$ to be the unconstrained marginals $p_{\boldsymbol{\theta}}(z_i)$, meaning that the constraint is ignored during the training process; empirical results show that such ignorance can harm model performance even though the constraint is enforced at inference time. We refer the readers to the experimental section for the empirical results.

The remaining question is how to obtain the closed-form $\boldsymbol{m}(\boldsymbol{\theta})$ for different estimators. We present below the theoretical results for the constrained marginals and their expectations.

Proposition 8 (Gaussian Conditional Marginal and Expectations)**.** *Given $\boldsymbol{z} = (z_1, \ldots, z_n)^T \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, the conditional marginal $p_{\boldsymbol{\theta}}(z_i \mid \sum_{j=1}^n z_j = k)$ follows a univariate Gaussian distribution with mean $\overline{\mu}_i = \boldsymbol{\mu}_i + \frac{\boldsymbol{\Sigma}_{i,i}+\sum_{j\neq i}^n \boldsymbol{\Sigma}_{i,j}}{\sum_{t=1}^n \boldsymbol{\Sigma}_{t,t}+\sum_{j\neq t}^n \boldsymbol{\Sigma}_{t,j}} \left(k - \sum_{j=1}^n \boldsymbol{\mu}_j\right)$ and variance $\overline{\sigma}_i^2 = \boldsymbol{\Sigma}_{i,i} - \frac{\left(\boldsymbol{\Sigma}_{i,i}+\sum_{j\neq i}^n \boldsymbol{\Sigma}_{i,j}\right)^2}{\sum_{t=1}^n \boldsymbol{\Sigma}_{t,t}+\sum_{j\neq t}^n \boldsymbol{\Sigma}_{t,j}}$. Further, the expectation of the marginal distribution is $\overline{\mu}_i$.*

We also present theoretical results on a system of linear equality constraints.

Proposition 9. *Let $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The constrained distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \boldsymbol{A}\boldsymbol{z} = \boldsymbol{k})$ is equivalent to an $n-a$ dimensional multivariate Gaussian distribution with mean $\overline{\boldsymbol{\mu}} \in \mathbb{R}^{n-a}$ and covariance matrix $\overline{\boldsymbol{\Sigma}} \in \mathbb{R}^{(n-a)\times(n-a)}$ defined as below,*

$$\overline{\mu} = \boldsymbol{E}\boldsymbol{\mu} + \boldsymbol{E}\boldsymbol{\Sigma}\boldsymbol{A}^T\left(\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^T\right)^{-1}(\boldsymbol{k} - \boldsymbol{A}\boldsymbol{\mu}) \quad \textit{and} \quad \overline{\boldsymbol{\Sigma}} = \boldsymbol{E}\boldsymbol{\Sigma}\boldsymbol{E}^T - \boldsymbol{E}\boldsymbol{\Sigma}\boldsymbol{A}^T\left(\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^T\right)^{-1}\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{E}^T$$

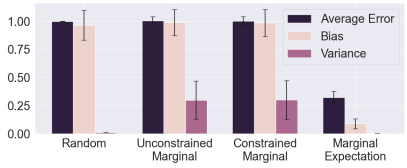
where $\boldsymbol{E} \in \mathbb{R}^{n-a\times n}$ contains the first $n-a$ rows of an identity matrix $\boldsymbol{I} \in \mathbb{R}^{n\times n}$. The marginal distribution subject to the constraint $p_{\boldsymbol{\theta}}(z_i \mid \boldsymbol{A}\boldsymbol{z} = \boldsymbol{k})$ can also be computed in closed form. Specifically, with $\boldsymbol{e}_i$ being a standard basis vector, it holds that $p_{\boldsymbol{\theta}}(z_i \mid \boldsymbol{A}\boldsymbol{z} = \boldsymbol{k}) \sim \mathcal{N}(\overline{\mu}_i, \overline{\sigma}_i^2)$,

$$\textit{with } \overline{\mu}_i = \mu_i + \boldsymbol{e}_i^T\boldsymbol{\Sigma}\boldsymbol{A}\left(\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^T\right)^{-1}(\boldsymbol{k} - \boldsymbol{A}\boldsymbol{\mu}) \text{ and } \overline{\sigma}_i^2 = \boldsymbol{e}_i^T\boldsymbol{\Sigma}\boldsymbol{e}_i - \boldsymbol{e}_i^T\boldsymbol{\Sigma}\boldsymbol{A}^T\left(\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^T\right)^{-1}\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{e}_i.$$

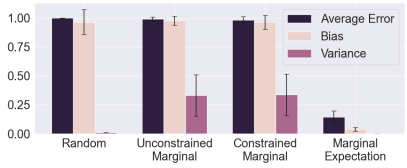
Related Work

A substantial amount of research has been devoted to estimating gradients for categorical random variables. [MMT16] and [JGP16] proposed to refactor the non-differentiable sample from a categorical distribution with a differentiable sample from a novel Gumbel-Softmax distribution, which enables automatic differentiation. Gradient estimation under linear equality constraints for categorical random variables has been widely studied. Existing methods either employ the score function and straight-through estimator or suggest custom relaxation [KSE16, CSW18, GWZ18, XE19b]. [XE19b] extends the Gumbel-softmax technique to k-subsets, enabling backpropagation for k-subset sampling. However, this comes at the trade-off of introducing some bias in the learning process due to the use of relaxed samples. While score function estimators offer a seemingly simple solution, it is widely acknowledged that they are prone to exhibiting exceedingly high variance. A recently introduced gradient estimator known as SIMPLE [AZN23a] surpasses its predecessors but is limited to Bernoulli random variables. Our work deviates from previous research on gradient estimation under linear equality constraints by investigating exact sampling as well as closed-form loss function for Gaussian and Poisson random variables, which are commonly used in scientific applications. We also extended the SIMPLE estimator [AZN23a] to Gaussian and Poisson random variables.

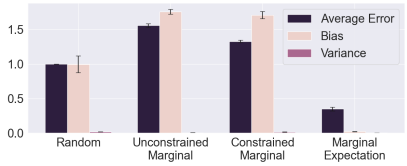
Extensive research has been conducted on numerical sampling from multivariate normal distributions while adhering to various constraints. [AMD14] reviewed classical Gibbs Sampling on a standard simplex (samples are positive and sum to one) and proposed using Hamiltonian Monte Carlo (HMC) methods. Efficient sampling methods for multivariate normal distributions truncated by hyperplanes($\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where $dim(\boldsymbol{x}) = N$ and $rank(\boldsymbol{A}) = n < N$) were investigated by [MBR22] and [CCZ17]. While these studies focus on numerical simulations, our study aims to derive closed-form solutions for not only linear equality constraint but also a system of linear equality constraint. Our theoretical findings, combined with closed-form loss functions, enable researchers to seamlessly incorporate constraints into machine learning models while



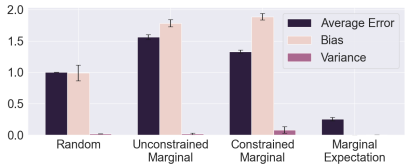
(a) Gaussian L1



(b) Gaussian L2



(c) Poisson L1



(d) Poisson L2

Figure 3.1: Comparisons of different gradient estimators for point-wise loss ℓ being L1 and L2 loss applied to Gaussian and Poisson variables are conducted. To evaluate the direction of the gradient, we utilized the cosine distance, defined as 1 - cosine similarity. The bias, variance, and error of the estimators are assessed using a sample size of 10,000. The bias and variance for Marginal Expectation applied to Poison random variable are extremely close to zero but not identically zero.

maintaining the probabilistic nature of random variables.

Integrating constraints as background knowledge into deep learning models has been widely studied. Traditional methods propose to add penalty terms to the loss function for constraint violations [DGM12, XZF18, FBD19, BGS22, SGL24]. While these methods are relatively straightforward to implement, they do not guarantee constraint satisfaction. Alternative approaches, such as [ATC22b], [GL21], and [SDC24], incorporate background knowledge into the topology of the network itself. In the context of generative tasks, [DAG20] incorporate propositional logic constraints in Generative Adversarial Networks for structured object generation, while [MMS22] demonstrate the integration of [DKT07] with variational autoencoders.

3.2.2.3 Beyond Gaussian

In this section, we present the theoretical results when $\boldsymbol{z}$ are Poisson variables defined over discrete domains. Similar to the Gaussian setting, we find that when the element-wise loss ℓ is $L1$ or $L2$ loss and the mapping $f_{\boldsymbol{u}}$ is an identity function, the expected loss admits closed-form expressions.

Proposition 10 (Poisson Closed-form Expected Loss). *Let $\boldsymbol{z} = (z_1, \ldots, z_n)^T$, where $z_i \sim Poisson(\theta_i)$. Let $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^T$ be the ground truth vector subject to the equality constraint $\sum_{i=1}^n y_i = k$ with $k \in \mathbb{N}^+$. Then it holds that*

i) when ℓ is L1 loss,

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} (k - \lfloor kp_i - d_i \rfloor) p_i Bin(\lfloor kp_i - d_i \rfloor; k, p_i) + \lfloor kp_i - d_i \rfloor (1 - p_i) Bin(\lfloor kp_i - d_i \rfloor; k, p_i) - 2d_i F(\lfloor kp_i - d_i \rfloor; k, p_i) + d_i;$$

ii) when ℓ is L2 loss, $L(\boldsymbol{\theta}) = \sum_{i=1}^n kp_i (1 - p_i) + k^2 p_i^2 - 2y_i kp_i + y_i^2$,

where Bin denotes the probability mass function (p.m.f.) of a binomial distribution and F denotes a regularized incomplete beta function. $d_i = kp_i - y_i$ and $p_i = \frac{\theta_i}{\sum_{j=1}^n \theta_j}$.

In the general setting when there is no closed-form solution for the expected loss, we first show that exact sampling from the constrained distribution can be achieved as it takes its form as a multinomial distribution.

Proposition 11 (Poisson Constrained Distribution). *Given $\boldsymbol{z} = (z_1, \ldots, z_n)^T$ with $z_i \sim Poisson(\theta_i)$, the constrained distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_{j=1}^n z_n = k)$ is equivalent to a multinomial distribution with parameter k and probabilities $\frac{\theta_1}{\sum_{j=1}^n \theta_j}, \ldots, \frac{\theta_n}{\sum_{j=1}^n \theta_j}$.*

In the backward pass, the results below allow us to derive gradient estimators with either the conditional marginals or the expectation of the conditional marginal as a differentiable proxy.

Proposition 12 (Poisson Conditional Marginal and Expectations). *Given* $\boldsymbol{z} = (z_1, \ldots, z_n)^T$ *with* $z_i \sim Poisson(\theta_i)$*, the conditional marginal* $p_{\boldsymbol{\theta}}(z_i \mid \sum_{j=1}^n z_j = k)$ *follows a binomial distribution with parameter* k *and probability* $\frac{\theta_i}{\sum_{j=1}^n \theta_j}$*. Further, its expectation is* $\frac{k\theta_i}{\sum_{j=1}^n \theta_j}$*.*

3.2.3 Empirical Evaluation

We conduct a comprehensive empirical analysis of our proposed method across four different tasks: 1) Synthetic experiments where the ground truth gradients are available and thus we can compare the bias and variance of different estimators. 2) A variational autoencoder (VAE) setting where the posterior of the latent variables is constrained by the linear equality and gradient estimators are necessary for end-to-end training. It allows us to compare the effectiveness of different estimators in terms of generation performance. 3) Another VAE setting where the underlying data distribution is constrained by linear equality. We compare various VAE models and their constrained counterparts, where the success of linear equality integration consistently boosts model performance. 4) A scientific application where a charge-neutral constraint is enforced into the charge predictions of the message-passing neural networks (MPNNs) and the expected loss admits a closed-form solution in this setting.

3.2.3.1 Synthetic Experiments

We consider synthetic settings where the ground truth gradients can be obtained by taking derivatives of the closed-form expected loss as stated in Section 3.2.1 such that we can compare how good the gradient approximations are for each gradient estimator. The distance between the estimated and the ground truth gradient vectors is measured by the cosine distance, defined as 1 - cosine similarity. We evaluate the performance of gradient estimators on three metrics: bias, variance, and average error. We compare the two proposed gradient estimators, *Constrained Marginal* and *Marginal Expectation*, with two baseline estimators, *Random* and *Unconstrained Marginal* as defined in Table 3.1.

Results are shown in Figure 3.1, where *Marginal Expectation* significantly outperforms the

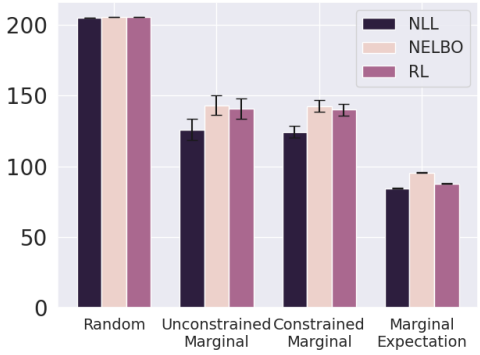


Figure 3.2: Comparison of gradient estimators for VAE with constrained latent space. Negative log-likelihood (NLL), negative ELBO (NELBO), and reconstruction loss (RL) are averaged over 5 trials.

others in all cases. Estimator *Unconstrained Marginal* has similar performances to *Random* which is expected since it discards the constraint information. What is interesting is that estimator *Constrained Marginal*, even though it is informed by the constraint, it also performs as bad as *Random*. Even though in the Bernoulli setting, *Marginal Expectation* and *Constrained Marginal* are the same estimator as shown in [AZN23a], we show that in the Gaussian and Poisson setting, the former is capable of providing decent gradient approximations while the latter is not.

3.2.3.2 Constrained Latent Space

We further test the four gradient estimators in Table 3.1 under the setting where the VAE model has its latent space constrained by linear equality, that is, the latent variables should sum to a constant. The VAE is trained on the MNIST dataset using the evidence lower bound (ELBO) as objective, which consists of a reconstruction loss (RL) and the KL divergence between a constrained approximate posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \sum_i z_i = k, \boldsymbol{x})$ and a prior of the latent space.

Experiment results are presented in Figure 3.2 where the generative performance is evaluated using test negative likelihood, estimated using importance sampling [BGS16], ELBO, and reconstruction loss. Results show that the estimator *Marginal Expectation* outperforms all other estimators, which is consistent with the approximation results that we observe in the synthetic

experiments in Figure 3.1. *Unconstrained Marginal* and *Constrained Margianl* have similar performance, both better than *Random*.

3.2.3.3 Constrained Data Generation

We consider a setting where the underlying data generation distribution is constrained by linear equality as domain knowledge. We modify the MNIST dataset such that for each image, all the pixel values sum to be 100. The first row in Figure 3.3 shows the original MNIST images and our processed ones in the second row. The processed images are then the inputs to various VAE models and their constrained counterparts.

We consider three VAE models: Vanilla VAE [KW22], Ladder VAE [SRM16], and Graph VAE [HGM18]. We compare the performance of these models and their contained counterparts integrated with linear equality using estimator *Marginal Expectation* as it shows the best performance so far. All the VAE models are trained using ELBO as objectives.

The performance of the VAEs is evaluated from three aspects. (1) Generative Ability: We use test log-likelihood (LL) $\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$ estimated using 5,000 importance-weighted samples, ELBO, and Reconstruction Loss (RL) to measure the models' performance, accuracy, and generalization capabilities. (2) Constraint Violation Rate: We measure the proportion of reconstructed testing data that violates the equality constraint. (3) Training Time: We record the average training time of all the models for one epoch. All results are averaged over 5 independent runs. The results are summarized in Table 3.2.

We find out that the vanilla VAE and other VAEs have high constraint violation rates. On the contrary, our method can constrain the model such that their generated data satisfy the constraint while also achieving better generative performance due to the inductive bias. In order to show that the addition of Marginal Expectation has basically no impact on speed, for each model we measure the training time of 1 epoch. We also test the average training time where the results show that the constrained versions are less than 10% slower than the unconstrained version. *Marginal Expectation* adds less than 1 second of additional training time for VAE and Ladder

Table 3.2: Comparison on VAE Generative Ability. The constrained VAE models achieve similar or better performance in terms of their generative ability while strictly satisfying the constraints. The unconstrained counterparts have a high constraint violation rate.

Model	LL ↑	ELBO ↑	RL ↓	Violation ↓
VAE	-22.42 ± 0.29	-23.41 ± 0.22	15.00 ± 0.46	0.30 ± 0.06
Constrained VAE	$\mathbf{-21.48 \pm 0.18}$	$\mathbf{-22.62 \pm 0.07}$	$\mathbf{12.79 \pm 0.11}$	$\mathbf{0 \pm 0}$
Ladder VAE	-24.25 ± 0.07	-30.84 ± 0.51	$\mathbf{23.06 \pm 0.54}$	0.38 ± 0.02
Constrained Ladder VAE	$\mathbf{-23.86 \pm 0.06}$	$\mathbf{-30.78 \pm 0.08}$	$\mathbf{23.40 \pm 0.16}$	$\mathbf{0 \pm 0}$
Graph VAE	-22.74 ± 0.11	-23.54 ± 0.18	15.45 ± 0.41	0.29 ± 0.09
Constrained Graph VAE	$\mathbf{-21.61 \pm 0.20}$	$\mathbf{-22.53 \pm 0.06}$	$\mathbf{12.73 \pm 0.21}$	$\mathbf{0 \pm 0}$

VAE.

3.2.3.4 Partial Charge Predictions for Metal-Organic Frameworks

Metal-organic frameworks (MOFs) represent a class of materials with a wide range of applications in chemistry and materials science. Predicting properties of MOFs, such as partial charges on metal ions, is essential for understanding their reactivity and performance in chemical processes. However, it is challenging due to the complex interactions between metal ions and ligands and the requirement that the predictions need to satisfy the charge neutral constraint, that is, an exactly-zero constraint.

We adopt the same model as [RSS20] where the charges are assumed to be Gaussian and the element loss is L1. Our model architecture extends the Message Passing Neural Network (MPNN) framework and incorporates linear-equality constraint for Gaussian variables, ensuring strict adherence to the critical constraint. The core innovation involves replacing the conventional L1 loss with the closed-form Gaussian loss function 6 as well as the negative log likelihood of the con-

Figure 3.3: The first row displays the original MNIST images, while the second row shows the constrained versions that are input to the VAEs. We compare the reconstructed images produced by various VAE models and their contained counterparts integrated with linear equality.

strained multivariate Gaussian. The L1 loss function penalizes deviations from the linear-equality constraint while considering the probabilistic nature of Gaussian variables, and the negative log likelihood loss models the observed data with higher probability. This comprehensive approach not only enables our model to capture complex structural relationships in MOFs but also ensures accurate predictions of partial charges while respecting the crucial linear-equality constraint, enhancing its applicability in a wide range of graph-based applications, including those pertaining to metal-organic frameworks.

Additionally, we also devise an ensemble methodology to enhance the predictive performance and robustness of our linear-equality constrained MPNN model. To achieve this, we adopt a systematic approach encompassing model variability, aggregation strategies and cross-validation. Two instances of the linear-equality constrained MPNN model are trained with variations in initialization. We apply the averaging aggregation technique to combine the predictions from

these models. Performance assessment is conducted through cross-validation techniques. The ensemble's performance is evaluated on a separate test dataset to ascertain its generalization ability. This ensemble approach not only elevates predictive accuracy but also fortifies the model's resilience, rendering it highly effective for complex tasks, including those pertaining to metal-organic frameworks.

The prediction performance of our four proposed approaches is presented in Table 3.3, compared with baseline approaches reported by [RSS20]. Results show that training using negative log likelihood loss and closed-form expected loss achieves better performance as MPNN (variance) which is considered to be the strongest baseline approach. When further combined with the ensemble method, our approach achieves significantly better predictions.

3.3 Discussion

In this chapter, we present the continuous counterpart of Simple and the subtlety of how to define the constrained marginal probabilities as a proxy for the gradient updates. For discrete samples, both constrained marginal probabilities and the constrained marginal expectation of a sample are the same but for continuous variables, they are different, and interestingly only the latter works well as shown in the empirical evaluations, showing that it is consistently more informative than the previous one.

Table 3.3: The prediction performances of different models for estimating partial charges on metal ions are presented. Comparing to the baseline MPNN (variance), both the closed-form loss function and Likelihood loss function yield superior Mean Absolute Deviation (MAD) results. Moreover, ensemble methods (second block) notably boost the predictive performance of all.

Method	**MAD** $\downarrow$	**NLL** $\downarrow$
(charge neutrality enforcement)	mean $\pm$ std	mean $\pm$ std
Constant Prediction	0.324 ± 0.007	$-$
Element-mean (uniform)	0.154 ± 0.002	$-$
Element-mean (variance)	0.153 ± 0.002	$-$
MPNN (uniform)	0.0260 ± 0.0008	109.8 ± 6.9
MPNN (variance)	0.0251 ± 0.0010	-19.9 ± 71.1
Closed-form (ours)	$\mathbf{0.0245 \pm 0.0009}$	$1.14\text{e+}8 \pm 3.15\text{e+}8$
Likelihood (ours)	0.0248 ± 0.0008	$\mathbf{-252 \pm 24.7}$
MPNN (variance) + Ensemble	0.0238 ± 0.0007	-45.2 ± 55.8
Closed-form + Ensemble (ours)	$\mathbf{0.0230 \pm 0.0008}$	$1.51\text{e+}7 \pm 1.29\text{e+}7$
Likelihood + Ensemble (ours)	$\mathbf{0.0231 \pm 0.0007}$	$\mathbf{-180 \pm 38.3}$

CHAPTER 4

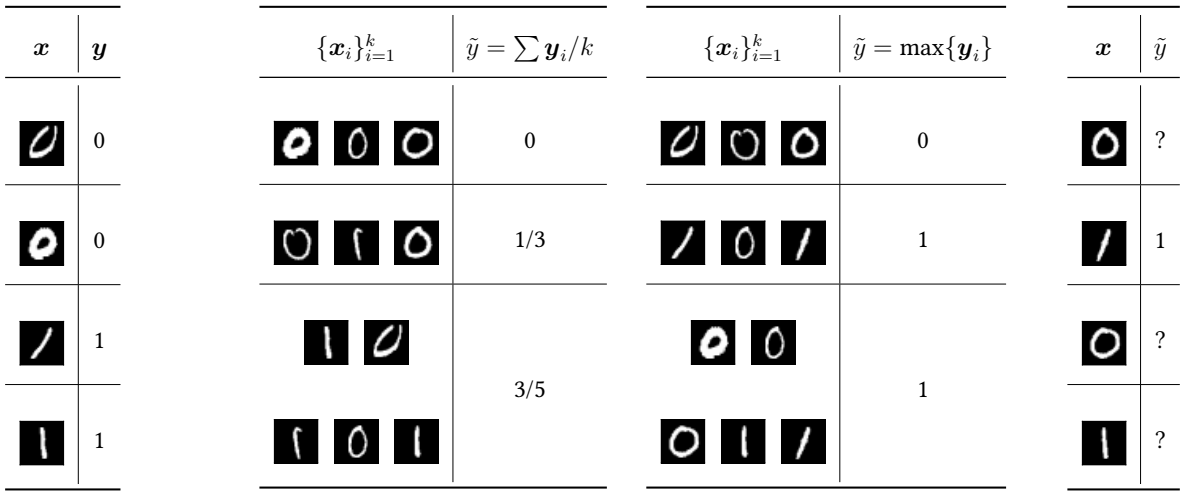
Learning from Weak Supervisions as Constraints

In the previous two chapters, we consider "hard" constraints, that is, such constraints are guaranteed to be satisfied in the deep learning pipeline. In this chapter, we consider "soft" constraints instead, where the models are encouraged to satisfy such constraints in the learning process. Further, instead of enforcing the constraints into the model architectures, this chapter is dedicated to the integration of soft constraints by modifying the loss functions. Specifically, we choose the count-based weakly supervised learning tasks to show that by translating the count supervision into soft constraints and using our methods to incorporate such constraints, model performance can be greatly boosted.

4.1 Count-Based Weakly Supervised Learning

Weakly supervised learning [Zho18] enables a model to learn from data with restricted, partial or inaccurate labels, often known as *weakly-labeled data*. Weakly supervised learning fulfills a need arising in many real-world settings that are subject to privacy or budget constraints, such as privacy sensitive data [WIB11], medical image analysis [BDO18], clinical practice [QCC], personalized advertisement [BD20] and knowledge base completion [GTH15, ZD18], to name a few. In some settings, *instance-level labels* are unavailable. Instead, instances are grouped into *bags* with corresponding *bag-level labels* that are a function of the instance labels, e.g., the proportion of positive labels in a bag. A key insight that we bring forth is that such weak supervision can very often be construed as *enforcing constraints on label counts of data*.

More concretely, we consider three prominent weakly supervised learning paradigms. The



$\boldsymbol{x}$	$\boldsymbol{y}$
	0
	0
	1
	1

(a) Classical

$\{\boldsymbol{x}_i\}_{i=1}^k$	$\tilde{y} = \sum \boldsymbol{y}_i / k$
	0
	1/3
	3/5

(b) LLP

$\{\boldsymbol{x}_i\}_{i=1}^k$	$\tilde{y} = \max\{\boldsymbol{y}_i\}$
	0
	1
	1

(c) MIL

$\boldsymbol{x}$	$\tilde{y}$
	?
	1
	?
	?

(d) PU Learning

Table 4.1: A comparison of the tasks considered in the three weakly supervised settings, LLP (cf. Section 4.1.1), MIL (cf. Section 4.1.2) and PU learning (cf. Section 4.1.3), against the classical fully supervised setting for binary classification, using digits from the MNIST dataset.

first paradigm is known as *learning from label proportions* [QSC08]. Here the weak supervision consists in the *proportion* of positive labels in a given bag, which can be interpreted as *the count of positive instances* in such a bag. The second paradigm, whose supervision is strictly weaker than the former, is *multiple instance learning* [ML97, DLL01]. Here the bag labels only indicate the *existence* of at least one positive instance in a bag, which can be recast as to whether *the count of positive instances* is greater than zero. The third paradigm, *learning from positive and unlabeled data* [DDG99, LDG00], grants access to the ground truth labels for a subset of *only the positive instances*, providing only a class prior for what remains. We can recast the class prior as *a distribution of the count of positive labels*.

We formally introduce the aforementioned weakly supervised learning paradigms below. For notation, let $\mathcal{X} \in \mathbb{R}^d$ be the input feature space over d features and $\mathcal{Y} = \{0, 1\}$ be a binary label space. We write $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$ for the input and output random variables respectively. Recall that in fully-supervised binary classification, it is assumed that each feature and label pair $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$ is sampled independently from a joint distribution $p(\boldsymbol{x}, \boldsymbol{y})$. A classifier

f is learned to minimize the risk $R(f) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim p}[\ell(f(\boldsymbol{x}), \boldsymbol{y})]$ where $\ell : [0,1] \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ is the cross entropy loss function. Typically, the true distribution $p(\boldsymbol{x}, \boldsymbol{y})$ is implicit and cannot be observed. Therefore, a set of n training samples, $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$, is used and the empirical risk, $\hat{R}(f) = \frac{1}{n}\sum_{i=1}^n \ell(f(\boldsymbol{x}_i), \boldsymbol{y}_i)$, is minimized in practice. In the count-based weakly supervised learning settings, the supervision is given at a bag level instead of an instance level. We formally introduce these settings as below.

4.1.1 Learning from Label Proportions

Learning from label proportions (LLP) [QSC08] assumes that each instance in the training set is assigned to bags and only the proportion of positive instances in each bag is known. One example is in light of the coronavirus pandemic, where infection rates were typically reported based on geographical boundaries such as states and counties. Each boundary can be treated as a bag with the infection rate as the proportion annotation.

The goal of LLP is to learn an instance-level classifier $f : \mathcal{X} \to [0,1]$ even though it is trained on bag-level labeled data. Formally, the training dataset consists of m bags, denoted by $\mathcal{D} = \{(B_i, \tilde{y}_i)\}_{i=1}^m$ where each bag $B_i = \{\boldsymbol{x}_j\}_{j=1}^k$ consist of k instances and this k could vary among different bags. The bag proportions are defined as $\tilde{y}_i = \sum_{j=1}^k \boldsymbol{y}_j/k$ with $\boldsymbol{y}_j$ being the instance label that cannot be accessed and only $\tilde{y}_i$ is available during training. An example is shown in Figure 4.1b. We do not assume that the bags are non-overlapping while some existing work suffers from this limitation including [SZ20].

4.1.2 Multiple Instance Learning

Multiple instance learning (MIL) [ML97, DLL01] refers to the scenario where the training dataset consists of bags of instances, and labels are provided at bag level. However, in MIL, the bag label is a single binary label indicating whether there is a positive instance in the bag or not as opposed to a bag proportion defined in LLP. A real-world application of MIL lies in the field of drug activity [DLL01]. We can observe the effects of a group of conformations but not for any specific molecule, motivating a MIL setting. Formally, in MIL, the training dataset consists of m

Table 4.2: A summary of the labels and objective functions for all the settings considered in this chapter.

Task	Label	Label Level	Objective
Classical Fully Supervised	Binary $\boldsymbol{y}$	Instance Level	$-\boldsymbol{y}\log p(\boldsymbol{y}) - (1-\boldsymbol{y})\log(1-p(\boldsymbol{y}))$
Learning from Label Proportion	Continuous $\tilde{y} = \sum_i \boldsymbol{y}_i / k$	Bag Level	$-\log p(\sum \hat{y}_i = k\tilde{y})$
Multiple Instance Learning	Binary $\tilde{y} = \max\{y_i\}$	Bag Level	$-\tilde{y}\log p(\sum \hat{y}_i \geq 1) - (1-\tilde{y})\log p(\sum_i \hat{y}_i = 0)$
Learning from Positive and Unlabeled Data	Binary $\tilde{y}$	Instance Level	1) $\mathbb{D}_{KL}(\texttt{Bin}(k,\beta) \parallel p(\sum_i \hat{y}_i))$ 2) $-\log p(\sum \hat{y}_i = k\beta)$

bags, denoted by $\mathcal{D} = \{(B_i, \tilde{y}_i)\}_{i=1}^m$, with a bag consisting of k instances, i.e., $B_i = \{\boldsymbol{x}_j\}_{j=1}^k$. The size k can vary among different bags. For each instance $\boldsymbol{x}_j$, there exists an instance-level label $\boldsymbol{y}_j$ which is not accessible. The bag-level label is defined as $\tilde{y}_i = \max_j\{\boldsymbol{y}_j\}$. An example is shown in Figure 4.1c.

The main goal of MIL is to learn a model that predicts a bag label while a more challenging goal is to learn an instance-level predictor that is able to discover positive instances in a bag. In this work, we aim to tackle both by training an instance-level classifier whose predictions can be combined into a bag-level prediction as the last step.

4.1.3 Learning from Positive and Unlabeled Data

Learning from positive and unlabeled data or *PU learning* [DDG99, LDG00] refers to the setting where the training dataset consists of only positive instances and unlabeled data, and the unlabeled data can contain both positive and negative instances. A motivation of PU learning is persistence in the case of shifts to the negative-class distribution [PNS15], for example, a spam filter. An attacker may alter the properties of a spam email, making a traditional classifier require a new negative dataset [PNS15]. We note that taking a new unlabeled sample would be more efficient, motivating PU learning. Formally, in PU learning, the training dataset $\mathcal{D} = \mathcal{D}_p \cup \mathcal{D}_u$ where $\mathcal{D}_p = \{(\boldsymbol{x}_i, \tilde{y}_i = 1)\}_{i=1}^{n_p}$ is the set of positive instances with $\boldsymbol{x}_i$ from $p(\boldsymbol{x} \mid \boldsymbol{y} = 1)$ and $\tilde{y}$ denoting

whether the instance is labeled, and $\mathcal{D}_u = \{(\boldsymbol{x}_i, \tilde{y}_i = 0)\}_{i=1}^{n_u}$ the unlabeled set with $\boldsymbol{x}_i$ from

$$p_u(\boldsymbol{x}) = \beta\, p(\boldsymbol{x} \mid \boldsymbol{y} = 1) + (1 - \beta)\, p(\boldsymbol{x} \mid \boldsymbol{y} = 0), \tag{4.1}$$

where the mixture proportion $\beta := p(\boldsymbol{y} = 1 \mid \tilde{y} = 0)$ is the fraction of positive instances among the unlabeled population. Although the instance label $\boldsymbol{y}$ is not accessible, its information can be inferred from the binary selection label $\tilde{y}$: if the selection label $\tilde{y} = 1$, it belongs to the positively labeled set, i.e., $p(\boldsymbol{y} = 1 \mid \tilde{y} = 1) = 1$; otherwise, the instance $\boldsymbol{x}$ can be either positive or negative. An example of such a dataset is shown in Figure 4.1d.

The goal of PU learning is to train an instance-level classifier. However, it is not straightforward to learn from PU data and it is necessary to make assumptions to enable learning with positive and unlabeled data [BD20]. In this work, we make a commonly-used assumption for PU learning, *selected completely at random (SCAR)*, which lies at the basis of many PU learning methods.

Definition 1 (SCAR). *Labeled instances are selected completely at random, independent from input features and the positive distribution* $p(\boldsymbol{x} \mid \boldsymbol{y} = 1)$*, that is,* $p(\tilde{y} = 1 \mid \boldsymbol{x}, \boldsymbol{y} = 1) = p(\tilde{y} = 1 \mid \boldsymbol{y} = 1)$.

4.2 A Unified Approach: Count Loss

In this section, we derive objectives for the three weakly supervised settings, LLP, MIL, and PU learning, from first principles. Our proposed objectives bridge between neural outputs, which can be observed as counts, and arithmetic constraints derived from the weakly supervised labels. The idea is to capture how close the classifier is to satisfying the arithmetic constraints on its outputs. They can be easily integrated with deep learning models, and allow them to be trained end-to-end. For the three objectives, we show that they share the same computational building block: given k instances $\{\boldsymbol{x}_i\}_{i=1}^k$ and an instance-level classifier f that predicts $p(\hat{y}_i \mid \boldsymbol{x}_i)$ with $\hat{y}$ denoting the prediction variable, the problem of inferring the probability of the constraint on counts $\sum_{i=1}^k \hat{y}_i = s$ is to compute the count probability defined below:

$$p(\sum_{i=1}^{k} \hat{y}_i = s \mid \{\boldsymbol{x}_i\}_{i=1}^k) := \sum_{\hat{\boldsymbol{y}} \in \mathcal{Y}^k} [\![\sum_{i=1}^{k} \hat{y}_i = s]\!] \prod_{i=1}^{k} p(\hat{y}_i \mid \boldsymbol{x}_i)$$

Algorithm 5 Count Probability $p(\sum_{i=1}^{k} \hat{y}_i = s)$

Input: A set of k log probabilities $\{t_i\}_{i=1}^{k}$ with $t_i := \log p(\hat{y}_i = 1)$, the number of instances k, and a label sum s

Output: log probabilities $\log p(\sum_{i=1}^{k} \hat{y}_i = s)$ or a set of log probability $\{\log p(\sum_{i=1}^{k} \hat{y}_i = s)\}_{s=0}^{k}$

// $A[i, m] = \log p(\sum_{j=1}^{i} \boldsymbol{y}_j = m)\ \forall i, m$

Initialize an array A to be $-\mathsf{Inf}$ everywhere

$A[0, 0] = 0$ // $p(\sum_{j=1}^{0} \boldsymbol{y}_j = 0) = 1$

Compute $t'_i \leftarrow \texttt{log1mexp}(t_i)$ // $\log p(\boldsymbol{y}_i = 0)$

for $i = 1$ **to** k **do**

  for $m = 0$ **to** s **do**

    $a_+ = A[i-1, m-1] + t_i$

    $a_- = A[i-1, m] + t'_i$

    $A[i, m] = \texttt{logsumexp}(a_+, a_-)$

return $A[k, s]$ or $A[k, :]$

where $[\![\cdot]\!]$ denotes the indicator function and $\hat{\boldsymbol{y}}$ denotes the vector $(\hat{y}_1, \cdots, \hat{y}_k)$. For succinctness, we omit the dependency on the input and simply write the count probability as $p(\sum_{i=1}^{k} \hat{y}_i = s)$. Next, we show how the objectives derived from first principles can be solved by using the count probability as an oracle. We summarize all proposed objectives in Table 4.2. Later, we will show how this seemingly intractable count probability can be efficiently computed by our proposed algorithm.

LLP setting. Given a bag $B = \{\boldsymbol{x}_i\}_{i=1}^{k}$ of size k and its weakly supervised label $\tilde{y}$, by definition, it can be inferred that the number of positive instances (count) in the bag is $k\tilde{y}$. Our objective is to minimize the negative log probability $-\log p(\sum_i \hat{y}_i = k\tilde{y})$. Notice that when each bag consists of only one instance, that is, when the bag-level supervisions are reduced to instance-level ones, this objective is exactly cross-entropy loss. We further show that our method is risk-consistent, that is, the optimal classifier under our proposed loss provides predictions consistent with the underlying risk as in the supervised learning setting.

MIL setting. Given a bag $B = \{\boldsymbol{x}_i\}_{i=1}^{k}$ of size k and a single binary label $\tilde{y}$ as its weakly supervised label, we propose a cross-entropy loss as below

$$\ell(B, \tilde{y}) = -\tilde{y} \log p(\sum \hat{y}_i \geq 1) - (1 - \tilde{y}) \log p(\sum \hat{y}_i = 0).$$

Notice that in the above loss, the probability term $p(\sum \hat{y}_i = 0)$ is accessible to the oracle for computing count probability, and the other probability term $p(\sum \hat{y}_i \geq 1)$ can simply be obtained from $1 - p(\sum \hat{y}_i = 0)$, i.e., the same call to the oracle since all prediction variables $\hat{y}_i$ are binary.

PU Learning setting. Recall that for the unlabeled data $\mathcal{D}_u$ in the training dataset, an unlabeled instance $\boldsymbol{x}_i$ is drawn from a mixture distribution as shown in Equation 4.1 parameterized by a mixture proportion $\beta = p(\boldsymbol{y} = 1 \mid \tilde{y} = 0)$. Under the SCAR assumption, even though only a class prior is given, we show that the mixture proportion can be estimated from the dataset.

Proposition 13. *With SCAR assumption and a class prior* $\alpha := p(\boldsymbol{y} = 1)$*, the mixture proportion* $\beta := p(\boldsymbol{y} = 1 \mid \tilde{y} = 0)$ *can be estimated from dataset* $\mathcal{D}$*.*

Proof. First, the label frequency $p(\tilde{y} = 1 \mid \boldsymbol{y} = 1)$ denoted by c can be obtained by

$$c = \frac{p(\tilde{y} = 1, \boldsymbol{y} = 1)}{p(\boldsymbol{y} = 1)} = \frac{p(\tilde{y} = 1)}{p(\boldsymbol{y} = 1)} \quad \textit{(by the definition of PU learning).}$$

that is, $c = p(\tilde{y} = 1)/\alpha$. Notice that $p(\tilde{y} = 1)$ can be estimated from the dataset $\mathcal{D}$ by counting the proportion of the labeled instances. Thus, we can estimate the mixture proportion as below,

$$\beta = \frac{p(\tilde{y} = 0 \mid \boldsymbol{y} = 1)p(\boldsymbol{y} = 1)}{p(\tilde{y} = 0)} = \frac{(1 - p(\tilde{y} = 1 \mid \boldsymbol{y} = 1))p(\boldsymbol{y} = 1)}{1 - p(\tilde{y} = 1)} = \frac{(1 - c)\alpha}{1 - \alpha c}.$$

□

The probabilistic semantic of the mixture proportion is that if we randomly draw an instance $\boldsymbol{x}_i$ from the unlabeled population, the probability that the true label $\boldsymbol{y}_i$ is positive would be β. Further, if we randomly draw k instances, the distribution of the summation of the true labels $\sum_{i=1}^{k} \boldsymbol{y}_i$ conforms to a binomial distribution $\texttt{Bin}(k, \beta)$ parameterized by the mixture proportion β, i.e.,

$$p(\sum_{i=1}^{k} \boldsymbol{y}_i = s) = \binom{k}{s} \beta^s (1 - \beta)^{k-s}. \tag{4.2}$$

Based on this observation, we propose an objective to minimize the KL divergence between the distribution of predicted label sum and the binomial distribution parameterized by the mixture proportion for a random subset drawn from the unlabeled population, that is,

$$\mathbb{D}_{KL}\left(\texttt{Bin}(k, \beta) \;\|\; p(\sum_{i=1}^{k} \hat{y}_i)\right) = \sum_{s=0}^{k} \texttt{Bin}(s; k, \beta) \log \frac{\texttt{Bin}(s; k, \beta)}{p(\sum_{i=1}^{k} \hat{y}_i = s)}$$

where $\texttt{Bin}(s; k, \beta)$ denotes the probability mass function of the binomial distribution $\texttt{Bin}(k, \beta)$. Again, the KL divergence can be obtained by $k + 1$ calls to the oracle for computing count probability $p(\sum_{i=1}^{k} \hat{y}_i = s)$. The KL divergence is further combined with a cross entropy defined over labeled data $\mathcal{D}_p$ as in the classical binary classification training as the overall objective.

As an alternative, we propose an objective for the unlabeled data that requires fewer calls to the oracle: instead of matching the distribution of the predicted label sum with the binomial distribution, this objective matches only the expectations of the two distributions, that is, to maximize $p(\sum_{i=1}^{k} \hat{y}_i = k\beta)$ where $k\beta$ is the expectation of the binomial distribution $\texttt{Bin}(k, \beta)$. We present empirical evaluations of both proposed objectives in the experimental section.

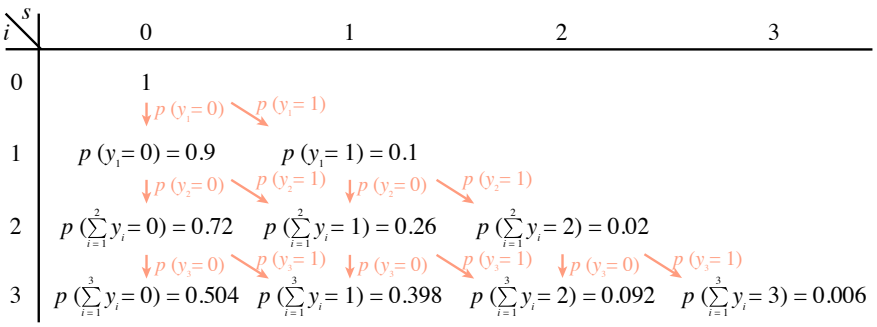


Figure 4.1: An example of how to compute the count probability in a dynamic programming manner. Assume that an instance-level classifier predicts three instances to have $p(\boldsymbol{y}_1 = 1) = 0.1$, $p(\boldsymbol{y}_2 = 1) = 0.2$, and $p(\boldsymbol{y}_3 = 1) = 0.3$ respectively. The algorithm starts from the top-left cell and propagates the results down right. A cell has its probability $p(\sum_{j=0}^{i} \boldsymbol{y}_j = s)$ computed by inputs from $p(\sum_{j=0}^{i-1} \boldsymbol{y}_j = s)$ weighted by $p(\boldsymbol{y}_i = 0)$, and $p(\sum_{j=0}^{i-1} \boldsymbol{y}_j = s - 1)$ weighted by $p(\boldsymbol{y}_i = 1)$ respectively, as indicated by the arrows.

Tractable Computation of Count Probability In the previous section, we show how the count probability $p(\sum_{i=1}^{k} \hat{y}_i = s)$ serves as a computational building block for the objectives derived from first principles for the three weakly supervised learning settings. With a closer look at the count probability, we can see that given a set of instances, the classifier predicts an instance-level probability for each and it requires further manipulation to obtain count information; actually, the number of joint labelings for the set can be exponential in the number of instances. Intractable as it seems, we show that it is indeed possible to derive a tractable computation for the count probability based on a result from [AZN23b].

Proposition 14. *The count probability $p(\sum_{i=1}^{k} \hat{y}_i = s)$ of sampling k prediction variables that sums to s from an unconstrained distribution $p(\boldsymbol{y}) = \prod_{i=1}^{k} p(\hat{y}_i)$ can be computed exactly in time $\mathcal{O}(ks)$. Moreover, the set $\{p(\sum_{i=1}^{k} \hat{y}_i = s)\}_{s=0}^{k}$ can also be computed in time $\mathcal{O}(k^2)$.*

The above proposition can be proved in a constructive way where we show that the count probability $p(\sum_{i=1}^{k} \hat{y}_i = s)$ can be computed in a dynamic programming manner. We provide an illustrative example of this computation in Figure 4.1. In practice, we implement this

computation in log space for numeric stability which we summarized as Algorithm 5, where function $\texttt{log1mexp}$ provides a numerically stable way to compute $\texttt{log1mexp}(x) = \log(1 - \exp(x))$ and function $\texttt{logsumexp}$ a numerically stable way to compute $\texttt{logsumexp}(x, y) = \log(\exp(x) + \exp(y))$. Notice that since we show it is tractable to compute the set $\{p(\sum_{i=1}^{k} \hat{y}_i = s)\}_{s=0}^{k}$, for any two given label sum s_1 and s_2, a count probability $p(s_1 \leq \sum_i \hat{y}_i \leq s_2)$ where the count lies in an interval, can also be exactly and tractably computed. This implies that our tractable computation of count probabilities can potentially be leveraged by other count-based applications besides the three weakly supervised learning settings in the last section.

Related Work

Weakly Supervised Learning. Besides settings explored in our work there are many other weakly-supervised settings. One of which is semi-supervised learning, a close relative to PU Learning with the difference being that labeled samples can be both positive and negative [ZG22, Zhu05]. Another is label noise learning, which occurs when our instances are mislabeled. Two common variations involve whether noise is independent or dependent on the instance [FV13, SKP22]. A third setting is partial label learning, where each instance is provided a set of labels of which exactly one is true [CST11a]. An extension of this is partial multi-label learning, where among a set of labels, a subset is true [XH18].

Unified Approaches. There exists some literature in regards to "general" approaches for weakly supervised learning. One example being the method proposed in [Hul14], which provides a procedure that minimizes the empirical risk on "fuzzy" sets of data. The paper also establishes guarantees for model identification and instance-level recognition. Co-Training and Self-Training are also examples of similar techniques that are applicable to a wide variety of weakly supervised settings [BM98, Yar95]. Self-training involves progressively incorporating more unlabeled data via our model's prediction (with pseudo-label) and then training a model on more data as an iterative algorithm [KMZ21]. Co-Training leverages two models that have different "views" of the data and iteratively augment each other's training set with samples they deem as "well-classified".

They are traditionally applied to semi-supervised learning but can extend to multiple instance learning settings [LZW11, XTX13, LZS23].

LLP. [QSC08] first introduced an exponential family based approach that used an estimation of mean for each class. Others seek to minimize "empirical proportion risk" or EPR as in [YCK14], which is centered around creating an instance-level classifier that is able to reproduce the label proportions of each bag. As mentioned previously, more recent methods use bag posterior approximation and neural-based approaches [AC17, TL20]. One such method is Proportion Loss (PL) [TL20], which we contrast to our approach. This is computed by binary cross entropy between the averaged instance-level probabilities and ground-truth bag proportion.

MIL. MIL finds its earlier approaches with SVMs, which have been used quite prolifically and still remain one of the most common baselines. We start with MI-SVM/mi-SVM [ATH02] which are examples of transductive SVMs [CCG18] that seek a stable instance classification through repeated retraining iterations. MI-SVM is an example of an instance space method [CCG18], which identifies methods that classify instances as a preliminary step in the problem. This is in contrast to bag-space or embedded-space methods that omit the instance classification step. Furthermore, [WYT18] remains one of the hallmarks of the use of neural networks for Multi-Instance Learning. [ITW18], utilize a similar approach but with attention-based mechanisms.

PU learning. [BD20] groups PU Learning paradigms into three main classes: two step, biased, and class prior incorporation. Biased learning techniques train a classifier on the entire dataset with the understanding that negative samples are subject to noise [BD20]. We will focus on a subset of biased learning techniques (Risk Estimators) as they are considered state-of-the-art and relevant to us as baselines. The Unbiased Risk Estimator (uPU) provides an alternative to the inefficiencies in manually biasing unlabeled data [PNS14, PNS15]. Later, Non-negative Risk Estimator (nnPU) [KND17] accounted for weaknesses in the unbiased risk estimator such as overfitting.

Count Loss. To our knowledge, viewing the computation of the "bag posterior" as *probabilistic* is new. However, the prior approaches do this implicitly. Many approaches have tried to approximate the "bag posterior" by averaging the instance-level probabilities in a bag [AC17, TL20]. In MIL settings, among instance-level approaches, the MIL-pooling is an implicit "bag posterior" computation. These include mean, max, and log-sum-exp pooling to approximate the likelihood that a bag has at least one positive instance [WYT18]. But again, these are all approximations of what our computation does *exactly*. In PU Learning, to our best knowledge, the view of unlabeled data as a bag annotated with the mixture proportion is new.

Neuro-Symbolic Losses. In this section, we have dealt with a specific form of distributional constraint. Conversely, there has been a plethora of work exploring the integration of *hard* symbolic constraints into the learning of neural networks. This can take the form of enforcing a hard constraint [ATC22a], whereby the network's predictions are guaranteed to satisfy the prespecified constraints. Or it can take the form of a soft constraint [XZF18, MDK18, AWC21, AWC22, ALT22, ACB23] whereby the network is trained with an additional loss term that penalizes the network for placing any probability mass on predictions that violate the constraint. While in this work we focus on discrete linear inequality constraints defined over binary variables, there is existing work focusing on hybrid linear inequality constraints defined over both discrete and continuous variables and their tractability [BPB15, ZMY21b, ZMY20b]. The development of inference algorithms for such constraints and their applications such as Bayesian deep learning remain an active topic [ZB19, KMZ19, ZMY20a, ZB23a].

4.3 Empirical Evaluation

In this section, we present a thorough empirical evaluation of our proposed count loss on the three weakly supervised learning problems, *LLP*, *MIL*, and *PU learning*.[1]

[1]Code and experiments are available at `https://github.com/UCLA-StarAI/CountLoss`

4.3.1 Learning from Label Proportions

We experiment on two datasets: 1) *Adult* with 8192 training samples where the task is to predict whether a person makes over $50k$ a year or not given personal information as input; 2) *Magic Gamma Ray Telescope* with 6144 training samples where the task is to predict whether the electromagnetic shower is caused by primary gammas or not given information from the atmospheric Cherenkov gamma telescope [DG17].[2]

We follow [SZ20] where two settings are considered: one with label proportions uniformly on $[0, \frac{1}{2}]$ and the other uniformly on $[\frac{1}{2}, 1]$. Additionally, we experiment on a third setting with label proportions distributing uniformly on $[0, 1]$ which is not considered in [SZ20] but is the most natural setting since the label proportion is not biased toward either 0 or 1. We experiment on four bag sizes $n \in \{8, 32, 128, 512\}$.

Count loss (CL) denotes our proposed approach using the loss objective defined in Table 4.2 for LLP. We compare our approach with a mutual contamination framework for LLP (LMMCM) [SZ20] and against Proportion Loss (PL) [TL20].

Results and Discussions We show our results in Table 4.3. Our method showcases superior results against the baselines on both datasets and variations in bag sizes. Especially in cases with lower bag sizes, i.e. $8, 32$, CL greatly outperforms all other methodologies. Among our baselines are methods that approximate the bag posterior (PL), which we show to be less effective than optimizing the exact bag posterior with CL.

4.3.2 Multiple Instance Learning

We first experiment on the MNIST dataset [LeC98] and follow the MIL experimental setting in [ITW18]: the training and test set bags are randomly sampled from the MNIST training and test set respectively; each bag can have images of digits from 0 to 9, and bags with the digit 9 are labeled positive. Moreover, the dataset is constructed in a balanced way such that there is an equal

[2]Publicly available at `archive.ics.uci.edu/ml`

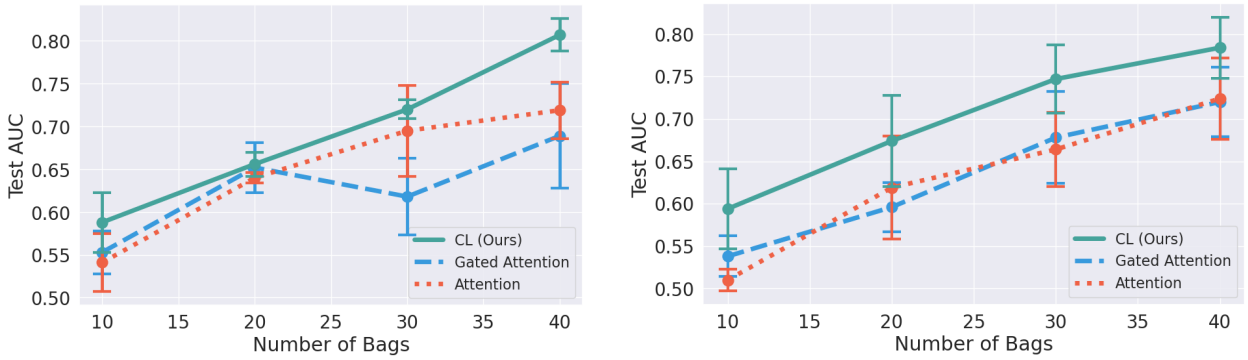


Figure 4.2: MIL MNIST dataset experiments with decreased numbers of training bags and lower bag size. Left: bag sizes sampled from $\mathcal{N}(10, 2)$; Right: bag sizes sampled from $\mathcal{N}(5, 1)$. We plot the mean test AUC (aggregated over 3 trials) with standard errors for 4 bag sizes. Best viewed in color.

amount of positively and negatively labeled bags as in [ITW18]. The task is to train a classifier that is able to predict bag labels; the more challenging task is to *discover key instances*, that is, to train a classifier that identifies images of digit 9. Following [ITW18], we consider three settings that vary in the bag generation process: in each setting, bags have their sizes generated from a normal distribution being $\mathcal{N}(10, 2), \mathcal{N}(50, 10), \mathcal{N}(100, 20)$ respectively. The number of bags in training set n is in $\{50, 100, 150, 200, 300, 400, 500\}$. Thus, we have $3 \times 7 = 21$ settings in total. Additionally, we introduce experimental analysis on *how the performance of the learning methods would degrade as the number of bags and total samples in training set decreases*, by modulating the number of training bags n to be $\{10, 20, 30, 40\}$ and selecting bag sizes from $\mathcal{N}(5, 1)$ and $\mathcal{N}(10, 2)$.

We also experiment on the Colon Cancer dataset [SRT16] to simulate a setting where bag instances are not independent. The dataset consists of 100 total hematoxylin-eosin (H&E) stained images, each of which contains images of cell nuclei that are classified as one of: epithelial, inflammatory, fibroblast, and miscellaneous. Each image represents a bag and instances are 27×27 patches extracted from the original image. A positively labeled bag or image is one that contains the epithelial nuclei. For both datasets, we include the Attention and Gated Attention mecha-

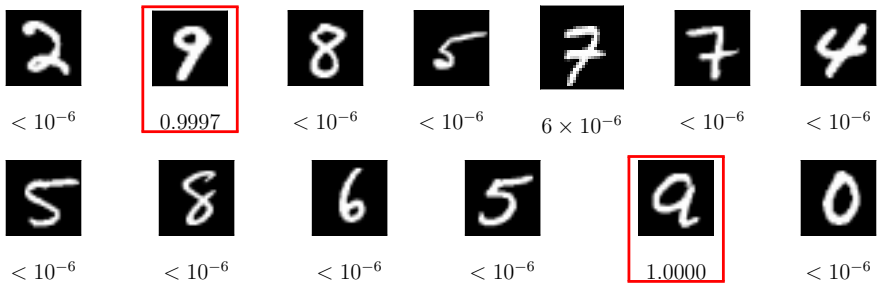


Figure 4.3: A test bag from our MIL experiments, where we set only the digit 9 as a positive instance. Highlighted in red are digits identified to be positive with corresponding probability beneath.

nism [ITW18] as baselines. We also use the MIL objective defined in Table 4.2.

Results and Discussions For the MNIST experiments, CL is able to outperform all other baselines or exhibit highly comparable performance for bag-level predictions as shown in Table 4.4. A more interesting setting is to compare how robust the learning methods are if the number of training bags decreases. [WYT18] claim that instance-level classifiers tend to lose against embedding-based methods. However, we show in our experiment that this is not true in all cases as seen in Figure 4.2. While Attention and Gated Attention are based on embedding, they suffer from a more severe drop in predictive performance than CL when the number of training bags drops from 40 to 10; our method shows great robustness and consistently outperforms all baselines. The rationale we provide is that with a lower number of training instances, we need more supervision over the limited samples we have. Our constraint provides this additional supervision, which accounts for the difference in performance.

We provide an additional investigation in Figure 4.3 to show that our approach learns effectively and delivers accurate instance-level predictions under bag-level supervision. In Figure 4.3, we can see that even though the classifier is trained on feedback about whether a bag contains the digit 9 or not, it accurately discovers all images of digit 9.

Our experimental results on the Colon Cancer dataset are shown in Table 4.5. We show that both our proposed objectives are able to consistently outperform baseline methods on all met-

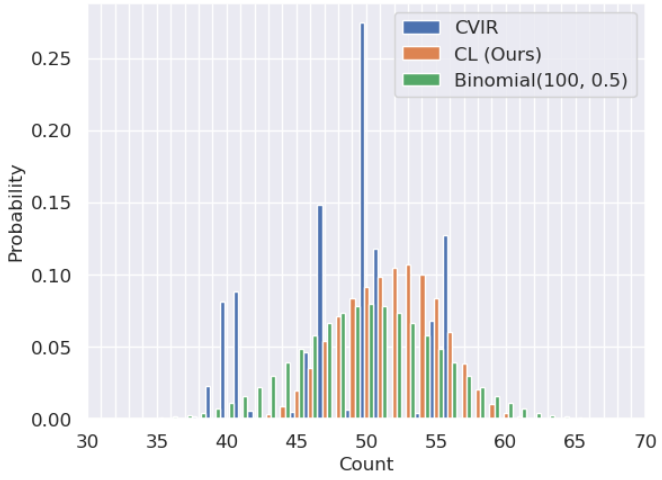


Figure 4.4: MNIST17 setting for PU Learning: We compute the average discrete distribution for CL and CVIR, over 5 test bags, each of which contain 100 instances. A ground truth binomial distribution of counts is also shown.

rics. Interestingly, we do not expect CL to perform well when instances in a bag are dependent; however, the results indicate that our count loss is robust to these settings.

4.3.3 Learning from Positive and Unlabeled Data

We experiment on dataset MNIST and CIFAR-10 [KH09a], following the four simulated settings from [GWS21]: 1) Binarized MNIST: the training set consist of images of digits $0-9$ and images with digits in range $[0, 4]$ are positive instances while others as negative; 2) MNIST17: the training set consist of images of digits 1 and 7 and images with digit 1 are defined as positive while 7 as negative; 3) Binarized CIFAR: the training set consists of images from ten classes and images from the first five classes is defined as positive instances while others as negative; 4) CIFAR Cat vs. Dog: the training set consist of images of cats and dogs and images of cats are defined as positive while dogs as negative. The mixture proportion is 0.5 in all experiments. The performance is evaluated using the accuracy on a test set of unlabeled data.

As shown in Table 4.2, we propose two objectives for PU learning. Our first objective is denoted by CL whereas the second approach is denoted by CL-expect. We compare against the

Conditional Value Ignoring Risk approach (CVIR) [GWS21], nnPU [KND17], and uPU [PNS15].

Results and Discussions Accuracy results are presented in Table 4.6 where we can see that our proposed methods perform better than baselines on 3 out of the 4 simulated PU learning settings. CL-expect builds off a similar "exactly-k" count approach, which we have shown to work well in the label proportion setting. The more interesting results are from CL where we fully leverage the information from a distribution as supervision instead of simply using the expectation. We think of this as applying a loss on each count weighted by their probabilities from the binomial distribution. We provide further evidence that our proposed count loss effectively guides the classifier towards predicting a binomial distribution as shown in Figure 4.4: we plot the count distributions predicted by CL and CVIR as well as the ground-truth binomial distribution. We can see that CL is able to generate the expected distribution, proving the efficacy of our approach.

4.4 Discussion

In this chapter, we present a unified approach to several weakly-supervised tasks, i.e., LLP, MIL, PU. We construct our approach based on the idea of using weak labels to constrain count-based probabilities computed from model outputs. A future direction for our work can be to extend to multi-class classification as well as explore the applicability to other weakly-supervised settings, e.g. label noise learning, semi-supervised learning, and partial label learning [CST11b, NDR13, Zhu05].

Table 4.3: LLP results across different bag sizes. We report the mean and standard deviation of the test AUC over 5 seeds for each setting. The highest metric for each setting is shown in **boldface**.

Dataset	**Dist**	**Method**	8	32	128	512
Adult	$[0, \frac{1}{2}]$	PL	0.8889 ± 0.0024	0.8782 ± 0.0036	$\mathbf{0.8743 \pm 0.0039}$	0.8678 ± 0.0085
Adult	$[0, \frac{1}{2}]$	LMMCM	0.8728 ± 0.0019	0.8693 ± 0.0047	0.8669 ± 0.0041	0.8674 ± 0.0040
Adult	$[0, \frac{1}{2}]$	CL (Ours)	$\mathbf{0.8984 \pm 0.0013}$	$\mathbf{0.8848 \pm 0.0041}$	$\mathbf{0.8743 \pm 0.0052}$	$\mathbf{0.8703 \pm 0.0070}$
Adult	$[\frac{1}{2}, 1]$	PL	0.8781 ± 0.0038	0.8731 ± 0.0035	$\mathbf{0.8699 \pm 0.0057}$	0.8556 ± 0.0180
Adult	$[\frac{1}{2}, 1]$	LMMCM	0.8584 ± 0.0164	0.8644 ± 0.0052	0.8601 ± 0.0045	0.8500 ± 0.0186
Adult	$[\frac{1}{2}, 1]$	CL (Ours)	$\mathbf{0.8854 \pm 0.0022}$	$\mathbf{0.8738 \pm 0.0039}$	0.8675 ± 0.0043	$\mathbf{0.8607 \pm 0.0056}$
Adult	$[0, 1]$	PL	0.8884 ± 0.0030	0.8884 ± 0.0008	$\mathbf{0.8879 \pm 0.0025}$	$\mathbf{0.8828 \pm 0.0051}$
Adult	$[0, 1]$	LMMCM	0.8831 ± 0.0026	0.8819 ± 0.0006	0.8821 ± 0.0017	0.8786 ± 0.0052
Adult	$[0, 1]$	CL (Ours)	$\mathbf{0.8985 \pm 0.0010}$	$\mathbf{0.8891 \pm 0.0013}$	0.8871 ± 0.0021	0.8790 ± 0.0056
Magic	$[0, \frac{1}{2}]$	PL	0.8900 ± 0.0095	0.8510 ± 0.0032	0.8405 ± 0.0110	0.8332 ± 0.0149
Magic	$[0, \frac{1}{2}]$	LMMCM	0.8918 ± 0.0077	0.8799 ± 0.0113	0.8753 ± 0.0157	0.8734 ± 0.0092
Magic	$[0, \frac{1}{2}]$	CL (Ours)	$\mathbf{0.9088 \pm 0.0056}$	$\mathbf{0.8830 \pm 0.0097}$	$\mathbf{0.8926 \pm 0.0049}$	$\mathbf{0.8864 \pm 0.0107}$
Magic	$[\frac{1}{2}, 1]$	PL	0.9066 ± 0.0016	0.8818 ± 0.0108	0.8769 ± 0.0101	0.8429 ± 0.0443
Magic	$[\frac{1}{2}, 1]$	LMMCM	0.8911 ± 0.0083	0.8790 ± 0.0091	0.8684 ± 0.0046	0.8567 ± 0.0292
Magic	$[\frac{1}{2}, 1]$	CL (Ours)	$\mathbf{0.9105 \pm 0.0020}$	$\mathbf{0.8980 \pm 0.0059}$	$\mathbf{0.8851 \pm 0.0255}$	$\mathbf{0.8816 \pm 0.0083}$
Magic	$[0, 1]$	PL	0.9039 ± 0.0029	0.8870 ± 0.0037	0.9002 ± 0.0092	0.8807 ± 0.0200
Magic	$[0, 1]$	LMMCM	0.9070 ± 0.0026	0.9048 ± 0.0058	0.9113 ± 0.0058	0.8934 ± 0.0097
Magic	$[0, 1]$	CL (Ours)	$\mathbf{0.9173 \pm 0.0018}$	$\mathbf{0.9102 \pm 0.0057}$	$\mathbf{0.9146 \pm 0.0051}$	$\mathbf{0.9088 \pm 0.0039}$

Table 4.4: MIL experiment on the MNIST dataset. Each block represents a different distribution from which we draw bag sizes—First Block: $\mathcal{N}(10, 2)$, Second Block: $\mathcal{N}(50, 10)$, Third Block: $\mathcal{N}(100, 20)$. We run each experiment for 3 runs and report mean test AUC with standard error. The highest metric for each setting is shown in **boldface**.

Training Bags	50	100	150	200	300	400	500
Gated Attention	0.775 ± 0.034	0.894 ± 0.012	0.935 ± 0.005	0.939 ± 0.006	$\mathbf{0.963 \pm 0.002}$	0.959 ± 0.002	$\mathbf{0.966 \pm 0.003}$
Attention	0.807 ± 0.026	$\mathbf{0.913 \pm 0.006}$	$\mathbf{0.940 \pm 0.004}$	0.942 ± 0.007	0.957 ± 0.002	0.961 ± 0.005	0.965 ± 0.004
CL (Ours)	$\mathbf{0.818 \pm 0.024}$	0.906 ± 0.009	0.929 ± 0.005	$\mathbf{0.946 \pm 0.001}$	0.952 ± 0.004	$\mathbf{0.962 \pm 0.002}$	0.963 ± 0.002
Gated Attention	$\mathbf{0.943 \pm 0.005}$	0.949 ± 0.009	$\mathbf{0.970 \pm 0.005}$	$\mathbf{0.977 \pm 0.001}$	0.983 ± 0.002	0.986 ± 0.004	$\mathbf{0.987 \pm 0.002}$
Attention	0.936 ± 0.010	$\mathbf{0.962 \pm 0.006}$	$\mathbf{0.970 \pm 0.001}$	$\mathbf{0.977 \pm 0.002}$	0.981 ± 0.002	$\mathbf{0.987 \pm 0.001}$	$\mathbf{0.987 \pm 0.002}$
CL (Ours)	0.939 ± 0.010	0.960 ± 0.002	0.964 ± 0.007	0.972 ± 0.002	$\mathbf{0.982 \pm 0.003}$	0.982 ± 0.001	$\mathbf{0.987 \pm 0.002}$
Gated Attention	0.975 ± 0.003	0.981 ± 0.004	0.992 ± 0.002	0.987 ± 0.004	$\mathbf{0.996 \pm 0.001}$	$\mathbf{0.998 \pm 0.001}$	0.990 ± 0.004
Attention	$\mathbf{0.984 \pm 0.001}$	0.982 ± 0.001	$\mathbf{0.996 \pm 0.000}$	0.987 ± 0.007	0.992 ± 0.004	0.994 ± 0.002	0.998 ± 0.000
CL (Ours)	0.981 ± 0.007	$\mathbf{0.989 \pm 0.000}$	$\mathbf{0.996 \pm 0.002}$	$\mathbf{0.995 \pm 0.001}$	$\mathbf{0.996 \pm 0.002}$	0.993 ± 0.003	$\mathbf{0.999 \pm 0.001}$

Table 4.5: MIL: We report mean test accuracy, AUC, F1, precision, and recall averaged over 5 runs with std. error on the Colon Cancer dataset. The highest value for each metric is shown in **boldface**.

Method	**Accuracy**	**AUC**	**F**1	**Precision**	**Recall**
Gated Attention	0.909 ± 0.014	0.908 ± 0.013	0.886 ± 0.021	0.916 ± 0.020	0.879 ± 0.020
Attention	0.893 ± 0.015	0.890 ± 0.008	0.876 ± 0.017	0.908 ± 0.016	0.879 ± 0.018
CL (Ours)	$\mathbf{0.915 \pm 0.008}$	$\mathbf{0.912 \pm 0.010}$	$\mathbf{0.903 \pm 0.010}$	$\mathbf{0.936 \pm 0.014}$	$\mathbf{0.898 \pm 0.007}$

Table 4.6: PU Learning: We report accuracy and standard deviation on a test set of unlabeled data, which is aggregated over 3 runs. The results from CVIR, nnPU, and uPU are aggregated over 10 epochs, as in [GWS21], while we choose the single best epoch based on validation for our approaches. The highest metric for each setting is shown in **boldface**.

Dataset	**Network**	**CL-expect (Ours)**	**CL (Ours)**	**CVIR**	**nnPU**	**nPU**
Binarized MNIST	MLP	95.9 ± 0.15	$\mathbf{96.4 \pm 0.01}$	96.3 ± 0.07	96.1 ± 0.14	95.2 ± 0.19
MNIST17	MLP	98.7 ± 0.17	$\mathbf{99.0 \pm 0.19}$	98.7 ± 0.09	98.4 ± 0.20	98.4 ± 0.09
Binarized CIFAR	ResNet	79.2 ± 0.27	80.1 ± 0.34	$\mathbf{82.3 \pm 0.18}$	77.2 ± 1.03	76.7 ± 0.74
CIFAR Cat vs. Dog	ResNet	$\mathbf{76.5 \pm 1.86}$	74.8 ± 1.64	73.3 ± 0.94	71.8 ± 0.33	68.8 ± 0.53

Part II

Reasoning

CHAPTER 5

Foundations: Weighted Model Integration

In the previous chapters, we demonstrate that the learning performance under constraints highly depends on the capability of probabilistic reasoning over constraints. This chapter addresses another fundamental challenge in Neurosymbolic AI: *how to perform scalable and reliable probabilistic reasoning over expressive symbolic constraints*. Instead of considering specific constraints, we aim for a general framework that handles complex constraints and performs probabilistic reasoning in a principled way.

Weighted model integration (WMI) is a such a framework for performing advanced probabilistic inference in hybrid domains, i.e., on distributions over mixed continuous-discrete random variables and in the presence of complex logical and arithmetic constraints. This chapter is dedicated to provide the background on WMI.

5.1 Overview

In many real-world scenarios, performing probabilistic inference requires reasoning over domains with complex logical and arithmetic constraints while dealing with variables that are heterogeneous in nature, i.e., both continuous and discrete.

Consider for example the task of matching players in a game by their skills. Performing probabilistic inference for this task has been popularized by [MCZ18] and is at the core of several online gaming services. A probabilistic model for this task has to deal with continuous variables, such as the player and team performance, and reason over discrete attributes such as membership in a squad and the achieved scores. Moreover, such a model would need to take into account

constraints such as the team performance being bounded by that of the players in it, and that forming a squad boosts performance. Ultimately, this translates into performing probabilistic inference in the presence of logical and arithmetic constraints and dependencies.

These hybrid scenarios are beyond the reach of probabilistic models including variational autoencoders [KW13] and generative adversarial networks [GPM14], whose inference capabilities, despite their recent success, are limited. Classical probabilistic graphical models [KF09], while providing more flexible inference routines, are generally incapacitated when dealing with continuous and discrete variables at once [SW11], or they tend to make simplistic [HG95, LW89] or overly strong assumptions about their parametric forms [YBR14]. Even recent efforts in modeling these hybrid scenarios while delivering tractable inference [MVD18, VMP19] can not perform inference in the presence of complex constraints.

Weighted Model Integration (WMI) [BPB15, MPS17] is a recent framework for probabilistic inference that offers all the aforementioned "ingredients" needed for hybrid probabilistic reasoning with logical constraints, *by design*. WMI leverages the expressive language of Satisfiability Modulo Theories (SMT) [BMR10] for describing problems over continuous and discrete variables. Moreover, WMI provides a principled way to perform hybrid probabilistic inference: asking for the probability of a complex query with logical and arithmetic constraints can be done by integrating weight functions over the regions that satisfy the constraints and query at hand.

5.2 Formalization

Notation. We use uppercase letters for random variables (e.g., X, B) and lowercase letters for their assignments (e.g., x, b). Bold uppercase letters denote sets of variables (e.g., $\mathcal{X}, \mathbf{B}$) and their lowercase denote their assignments (e.g., $\boldsymbol{x}, \boldsymbol{b}$). We represent logical formulas by capital Greek letters, (e.g., Λ, Φ, Δ), and literals (i.e., atomic formulas or their negation) by lowercase ones (e.g., ϕ, δ) or ℓ. We denote satisfaction of a formula Φ by one assignment $\boldsymbol{x}$ by $\boldsymbol{x} \models \Phi$ and we denote its corresponding indicator function as $[\![\boldsymbol{x} \models \Phi]\!]$. For undirected graphs, neigh denotes the set of neighboring nodes; for directed ones, pa and ch denote the parent node and the set of

child nodes respectively.

Satisfiability Modulo Theories (SMT). SMT [BT18] generalizes the well-known SAT problem [BHM09] to determining the satisfiability of a logical formula w.r.t. a decidable theory. Rich mixed logical/arithmetic constraints can be expressed in SMT for hybrid domains. In particular, we consider quantifier-free SMT formulas in the theory of linear arithmetic over the reals, or SMT($\mathcal{LRA}$). Here, formulas are propositional combinations of atomic Boolean literals and of atomic $\mathcal{LRA}$ literals over real variables, for which satisfaction is defined in a natural way. W.l.o.g. we assume SMT formulas to be in conjunctive normal form (CNF). In the following, we will use the shorthand SMT to denote SMT($\mathcal{LRA}$).

Example 1 (SMT representation of a skill matching system)**.** *In a skill rating system for online games, the team performance $X_{\mathcal{T}}$ of each team $\mathcal{T}$ is defined by the performance X_i of each player i in team $\mathcal{T}$, both of which are real variables. The team performance $X_{\mathcal{T}}$ is also related to a Boolean variable B indicating whether players in the team form a squad, i.e., a group of friends, which offsets (boosts) the team performance. We can build an SMT formula Γ of the relationship among these variables as follows. For brevity, we omit the domains for real variables in the formula.*

$$\Gamma := \bigwedge_{i \in \mathcal{T}} \ | \ X_{\mathcal{T}} - X_i \ | < 1 \ \bigwedge \ (B \Rightarrow X_{\mathcal{T}} > 2)$$

We show in Figure 5.1 the feasible regions of formula Γ i.e., the volumes for which the constraints are satisfied.

Example 2 (SMT representation of a house price model)**.** *For a house i, let price_i be its price and sqft_i its square footage. We can build a simple SMT($\mathcal{LRA}$) formula of the relationship between these real variables, with the corresponding solution space depicted in Figure 5.2. That is, SMT($\mathcal{LRA}$) formula Γ_i is*

$$\begin{aligned} \Gamma_i := \ & (\mathit{price}_i < 10 \cdot \mathit{sqft}_i + 1000) \vee \ (\mathit{price}_i < 20 \cdot \mathit{sqft}_i + 100) \\ & \wedge (0 < \mathit{price}_i < 3000) \wedge (0 < \mathit{sqft}_i < 200). \end{aligned}$$

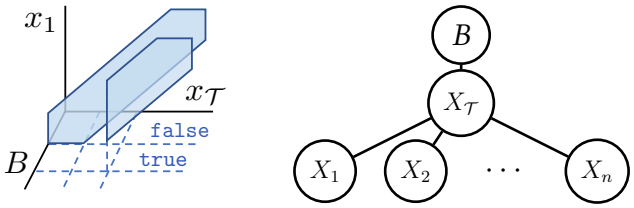


Figure 5.1: Feasible region (left) of formula Γ with one player and primal graph (right) of formula Γ with n players from Example 1.

Weighted Model Integration (WMI). Weighted Model Integration (WMI) [BPB15, MPS17] provides a framework for probabilistic inference with models defined over the logical constraints given by SMT formulas.

Definition 2. *(WMI) Let $\mathcal{X}$ be a set of continuous random variables defined over $\mathbb{R}$, and $\mathbf{B}$ a set of Boolean random variables defined over $\mathbb{B} = \{\texttt{true}, \texttt{false}\}$. Given an SMT formula Δ over $\mathcal{X}$ and $\mathbf{B}$, and a weight function $w : (\boldsymbol{x}, \boldsymbol{b}) \mapsto \mathbb{R}^+$ belonging to some parametric weight function family Ω, the weighted model integration (*WMI*) task computes*

$$\mathsf{WMI}(\Delta, w; \mathcal{X}, \mathbf{B}) \triangleq \sum_{\boldsymbol{b} \in \mathbb{B}^{|\mathbf{B}|}} \int_{(\boldsymbol{x}, \boldsymbol{b}) \models \Delta} w(\boldsymbol{x}, \boldsymbol{b})\, d\boldsymbol{x}. \tag{5.1}$$

That is, summing over all possible Boolean assignments $\boldsymbol{b} \in \mathbb{B}^{|\mathbf{B}|}$ while integrating over the weighted assignments of $\mathcal{X}$ making the evaluation of the formula SAT: $(\boldsymbol{x}, \boldsymbol{b}) \models \Delta$.

Weight functions w are usually defined as products of literal weights [BPB15, CD08, ZB19]. That is, for a set of literals $\mathcal{L}$, a set of per-literal weight functions $\mathcal{W} = \{w_\ell(\boldsymbol{x})\}_{\ell \in \mathcal{L}}$ is given, with weight functions w_ℓ defined over variables in literal ℓ. Then, the weight of assignment $(\boldsymbol{x}, \boldsymbol{b})$ is:

$$w(\boldsymbol{x}, \boldsymbol{b}) = \prod\nolimits_{\ell \in \mathcal{L}} w_\ell(\boldsymbol{x})^{[\![\boldsymbol{x}, \boldsymbol{b} \models \ell]\!]}.$$

When all variables are Boolean (i.e., $\mathcal{X} = \emptyset$), the per-literal weights $w_\ell(\boldsymbol{x})$ are constants and we retrieve the original definition of the well-known weighted model counting (WMC) task [CD08] as a special case of WMI. In this section, we assume that all per-literal weights are from some certain weight function family, and for literals not in the set $\mathcal{L}$, their weights are the constant

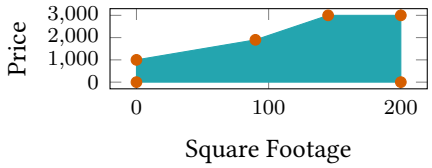


Figure 5.2: Feasible region of SMT theory Γ_i from Example 2

function one. This setting is expressive enough to approximate many continuous distributions [BPB15].

Example 3 (WMI formulation of a skill matching system)**.** *Consider the team performance SMT model* Γ *in Example 1. Assume that a set of per-literal weights* $w_{\ell_i}(X_\mathcal{T}, X_i) = 0.1 \cdot (X_\mathcal{T} + X_i - 6)^2$ *is associated to literals* $\ell_i = X_\mathcal{T} - X_i < 1$*, quantifying how likely the team performance is upper bounded by player performances. Then the* WMI *of the formula* Γ *with two players is* $\mathsf{WMI}(\Gamma, w; \mathcal{X}, \mathbf{B}) \approx 170.69$.

Example 4 (WMI formulation of a house price model)**.** *Consider a formula* $(b \vee \neg b) \wedge \Gamma_i$ *where* b *is a Boolean variable and* Γ_i *is as defined in Example 2. Consider the set of literals* $\mathcal{L} = \{b, (0 < \mathit{price}_i < 3000)\}$ *and per-literal weight functions* $\mathcal{P} = \{p_b, p_{(0<\mathit{price}_i<3000)}\}$*, with* $p_b(\boldsymbol{x}) = 1.5$ *and* $p_{(0<\mathit{price}_i<3000)}(\boldsymbol{x}) = \mathit{price}_i^2$*. Then, in worlds with both literals in* $\mathcal{L}$ *satisfied, our weight function is*

$$p_b(\mathit{price}_i, \mathit{sqft}_i) \cdot p_{(0<\mathit{price}_i<3000)}(\mathit{price}_i, \mathit{sqft}_i) = 1.5 \cdot \mathit{price}_i^2.$$

In worlds where b *is false and only* $(0 < \mathit{price}_i < 3000)$ *is satisfied, the weight function is* price_i^2.

Intuitively, $\mathsf{WMI}(\Delta, w; \mathcal{X}, \mathbf{B})$ equals the partition function of the unnormalized probability distribution induced by weights w on formula Δ. In the following, we will adopt the shorthand $\mathsf{WMI}(\Delta, w)$ for computing the WMI with all the variables in Δ in scope. The set of weight functions w together act as an unnormalized probability density while the formula Δ represents logical constraints defining its structure. Therefore, it is possible to compute the (now normalized) probability of any logical *query* Φ expressible as an SMT formula involving complex constraints:

$$\Pr_\Delta(\Phi) = \mathsf{WMI}(\Delta \wedge \Phi, w) \;/\; \mathsf{WMI}(\Delta, w).$$

Example 5 (WMI inference for skill rating)**.** *Suppose we want to quantify the squad effect in a 2v2 game. Specifically, given two teams $\mathcal{T}_1$ and $\mathcal{T}_2$ whose players have the same performance, but team $\mathcal{T}_1$ is a squad while $\mathcal{T}_2$ is not, that is, $\Phi_c = (B_1 = \texttt{true} \wedge B_2 = \texttt{false})$. We wonder what is the probability of query $\Phi = X_{\mathcal{T}_1} > X_{\mathcal{T}_2}$, that is team $\mathcal{T}_1$ wins and $\mathcal{T}_2$ loses. The probability of query Φ can be computed by two* WMI *tasks as follows.*

$$\mathsf{Pr}_{\Delta}(\Phi \,|\, \Phi_c) = \frac{\mathsf{WMI}(\Delta \wedge \Phi_c \wedge \Phi, w)}{\mathsf{WMI}(\Delta \wedge \Phi_c, w)} = \frac{4,206}{7,225} \approx 58.22\%$$

with the SMT formula $\Delta := \Gamma_1 \wedge \Gamma_2$ where the two sub-formulas Γ_1 and Γ_2 model the two teams as in Example 1.

W.l.o.g, from here on we will focus on WMI problems on continuous variables only. We can safely do this since a WMI problem defined on continuous and Boolean variables of the form $\mathsf{WMI}(\Delta, w; \mathcal{X}, \mathbf{B})$ can always be reduced in polytime to a new WMI problem $\mathsf{WMI}(\Delta', w'; \mathcal{X}')$ on continuous variables only, by properly introducing auxiliary variables in $\mathcal{X}'$ to account for Boolean variables $\mathbf{B}$ without increasing the problem size [ZB19].

5.3 Related Work

WMI generalizes weighted model counting (WMC) [SBK05] to hybrid domains [BPB15]. WMC is one of the state-of-the-art approaches for inference in many discrete probabilistic models. Existing exact WMI solvers for arbitrarily structured problems include DPLL-based search with numerical [BPB15, MPS17, MPS19] or symbolic integration [SOG16] and compilation-based algorithms [KMS18, ZDD19].

Motivated by their success in WMC, [BBP16] present a caching scheme for WMI that allows reusing computations at the cost of not supporting algebraic constraints between variables. Different from usual, [MAD17] adopt Gaussian distributions, while [ZDD19] fixed univariate parametric assumptions for weight functions. Many recent efforts in WMI converged in the *pywmi* [KMZ19] python framework.

Among the exact WMI solvers, the majority ignores the problem structure to be as general-purpose as possible [BPB15, MPS17, MPS19, KMS18]. However, by doing so they are unable to scale beyond tens of variables in practice. Conversely, our recent efficient alternatives such as SMI [ZB19] and MP-WMI [ZMY20a] can greatly scale but only on WMI problems amenable to tractable inference. We further leverage the strengths of the latter to efficiently solve iterative integration problems in building approximate WMI solvers.

So far, most approximate WMI solvers rely on sampling, and as such inherit all the classical issues of Monte Carlo approaches like poor scalability and convergence [CC96]. Among these, SAMPO [ZDD19] employs Gibbs sampling but does not support generic polynomial weights. A very recent alternative is a fully polynomial randomized approximation scheme [ACD20]. However, it can only operate on DNF SMT formulas, and it is not applicable to our CNF representation as a conversion into DNF can blow up the problem size. Other MCMC variants [ASA15, AD15, ASW16] operating with algebraic constraints, while more effective, cannot be readily used for WMI inference problems. The only alternative to sampling schemes is the hashing-based WMI algorithm [BBP15a] which is known to perform poorly on non-trivial problems due the hardness of calibrating the *tilt* [CFM14].

Research on learning WMI distributions from data is at its early stages. Parameter learning for piecewise constant densities has been addressed in [BPB15]. Recently, an approach for jointly learning the structure and parameters of a WMI problem has been proposed in [MKT20]. Developing faster inference algorithms is thus beneficial in learning scenarios as, typically, learning a full model requires numerous calls to an inference procedure. WMI inference is closely related to probabilistic program inference, where complex arithmetic and logical constraints are induced by the program structure or its abstraction [HMB17, HBM18].

CHAPTER 6

Exact Inference Over Constraints

Despite the appealing features of WMI, current state-of-the-art WMI solvers are far from being applicable to high-dimensional real-world scenarios. This is due to the fact that most solvers ignore the dependency structure of the problem, here expressible through the notion of a primal or factor graph of an SMT formula [DM07]. Thus, their practical utility is limited by their inability to scale up the WMI inference. This chapter presents two proposed exact WMI solvers that are capable of leveraging the dependency structure and thus achieve superior inference performance than the existing solvers.

6.1 Search-based WMI Solver

As a first approach to leverage structure, we propose a search-based inference procedure for exact model integration that leverages decomposition to speed up inference. We demonstrate how local structure encoded in SMT theories gives rise to context-specific decomposition during search, reducing the number of models to be generated and integrated over. The integration problem is decomposed into sub-problems by instantiating shared variables and recursing independently on the resulting simplified SMT theories. We show how to choose finitely many values to instantiate continuous variables with, and subsequently do polynomial interpolation to recover exact answers to WMI problems. Our complexity analysis proves the first tractability result for a non-trivial class of WMI problems. Moreover, our experimental evaluation shows that our solver is drastically faster than existing solvers on WMI problems with sparse, tree-shaped primal graphs.

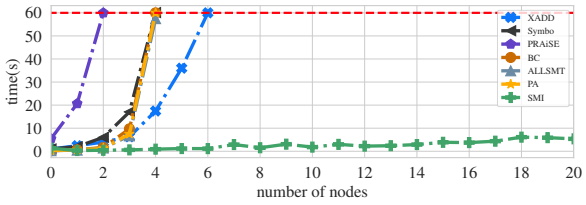


Figure 6.1: WMI runtime on independent model in Example 6.

6.1.1 Structure in WMI Problems

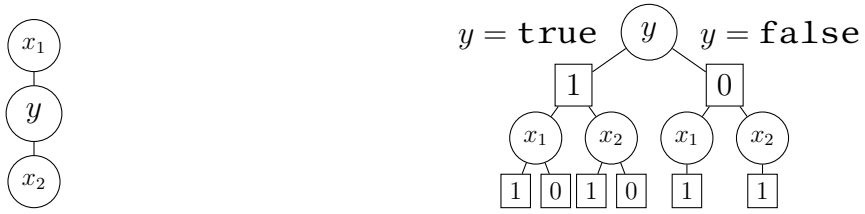
This section shows how to reduce WMI to model integration (MI) problems whose structural independence properties can be captured by graph abstractions.

6.1.1.1 Independence

We begin by motivating why we want to exploit independence structure during probabilistic reasoning.

Example 6. *Consider n houses, and conjoin the theory Γ_i from Example 2 n times, once for each house, into a larger SMT theory $\Gamma = \wedge_{i=1}^{n}\Gamma_i$. The n houses are independent since no formula in Γ connects properties of different houses. Thus, the WMI of Γ can be computed by multiplying the WMI of each individual theory Γ_i.*
Figure 6.1 takes a trivial weight function and compares existing WMI solvers on this simple problem. None is able to exploit the extreme independence structure in Γ. Our proposed method SMI, however, runs in linear time, as expected by the trivial factorization.

This explosion in runtime is due to the fact that existing solvers ignore independence between variables in the SMT($\mathcal{LRA}$) theory. However, in discrete graphical models and WMC, leveraging independence to decompose problems is at the core of all exact inference methods, and search-based algorithms in particular [Dar09, DM07]. Specifically, exact discrete inference methods *create* independence even when it is not immediately present, by performing a case analysis on selected discrete variables, instantiating them to all values, and simplifying the model.



(a) Primal graph

(b) Discrete And/Or Search Tree

Figure 6.2: Primal graph and search tree for $(y \vee x_1) \wedge (y \vee x_2)$.

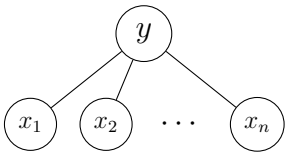
Through this process, search-based inference algorithms induce and exploit context-specific independence [BFG96]. The decompositions afforded by (conditional and context-specific) independence vastly reduce the computational cost of inference. Example 6 illustrated that this intuition carries over to WMI problems.

In what follows, we first describe the graph abstraction of SMT theories that characterizes dependencies between variables. These form the basis of our algorithm. Second, we show how WMI in hybrid domains can be reduced to unweighted MI in real domains. Hence, the solver we develop in this section will target MI problems.

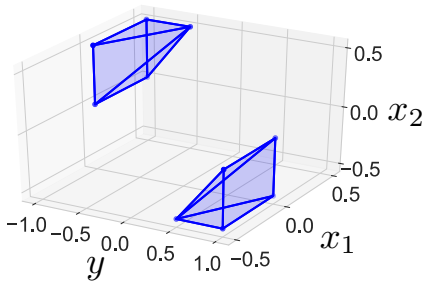
6.1.1.2 Graph Abstarction of SMT

Primal graphs are often used to characterize variable dependencies. For the example Boolean CNF formula $\theta_B = (y \vee x_1) \wedge (y \vee x_2)$ the primal graph is shown in Figure 6.2a. Its edges encode that variable pairs (y, x_1) and (y, x_2) appear in the same clause, while (x_1, x_2) never appear together, and are thus independent given y. Similarly, we will use primal graphs for SMT theories to capture variable dependency information as follows.

Definition 3. ***(Primal graph of SMT)*** *The primal graph of an SMT($\mathcal{LRA}$) CNF is an undirected graph whose vertices are all variables and whose edges connect any two variables that appear in the same clause.*



(a) Primal graph of theory θ_n



(b) Feasible region for θ_2

Figure 6.3: Primal graph and feasible region from Example 7.

Example 7. *Consider the following theory* θ_n*.*

$$\theta_n = \begin{cases} (-1 \leq y \leq 1) \wedge (-0.5 \leq x_1, \cdots, x_n \leq 0.5) \\ (x_i + 1 \leq y) \vee (y \leq x_i - 1), \textit{ for all } i \in [n] \end{cases}$$

Figure 6.3 shows its primal graph and solution space.

While there are many flavors of *search-based* exact inference, including recursive conditioning [Dar01], DPLL model counting [SBK05], knowledge compilation [CD08], and SumProd algorithms [BDP09], we use the And/Or-search framework to illustrate the required concepts [Nil82, DM07].

The And/Or search algorithm for WMC problems recursively simplifies a discrete counting problem by alternating between two steps. The first (OR) step selects a Boolean variable and tries to instantiate it to both true and false (we will later see how to choose the variable). The second (AND) step finds ways of partitioning the WMC problem into independent sub-problems that can be solved separately. Such sub-problems are introduced by instantiating variables in the OR step in a way that creates independence. The OR step is also called the Shannon expansion. The AND step is also referred to as component caching [SBK05] or detecting decomposability [CD08].

This process is illustrated in Figure 6.2b for the earlier Boolean CNF θ_B. Circles denote OR-step variables whose square-node children are its instantiations. After instatiating y, the search tree creates independent problems for x_1 and x_2. This independence can be read off directly

from the primal graph in Figure 6.2a. Search-based algorithms (with caching) are known to run efficiently on WMC problems with a tree or tree-like primal graph [Dar09, BDP09].

6.1.1.3 WMI to Model Integration Reduction

This section casts hybrid WMI problems into MI problems over only real variables. We consider the case where per-literal weight functions are monomials – functions of the form $\beta x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ over real variables x_i where $\beta \in \mathbb{R}$ and $\alpha_i \in \mathbb{N}$. We further assume that literals in $\mathcal{L}$ also appear in the theory, and that literals and their weights range over the same real variables.

We first show that any WMI problem with Boolean variables can be reduced to a WMI problem without Booleans. Then we show that WMI problems with per-literal weights can be reduced to an unweighted MI problem where the weight function is 1.

Proposition 15. *For each problem* $\mathsf{WMI}(\theta, w \mid \boldsymbol{x}, \boldsymbol{b})$ *there exists an equivalent problem* $\mathsf{WMI}(\theta', w' \mid \boldsymbol{x}')$ *without Boolean variables* $\boldsymbol{b}$ *such that*

$$\mathsf{WMI}(\theta, w \mid \boldsymbol{x}, \boldsymbol{b}) = \mathsf{WMI}(\theta', w' \mid \boldsymbol{x}')$$

and the primal graphs of θ *and* θ' *are isomorphic.*

This reduction encodes Boolean variables using fresh real variables and replaces each Boolean atom and its negation by two exclusive $\mathcal{LRA}$ atoms over those real variables. Proposition 15 allows us to focus on WMI problems without Boolean variables involved. Certain weight functions can also be reduced, as we show next.

Proposition 16. *For each problem* $\mathsf{WMI}(\theta, w \mid \boldsymbol{x})$ *with per-literal weights* w *as defined in this section, there exists an equivalent unweighted problem* $\mathsf{MI}(\theta' \mid \boldsymbol{x}')$ *s.t.*

$$\mathsf{WMI}(\theta, w \mid \boldsymbol{x}) = \mathsf{MI}(\theta' \mid \boldsymbol{x}').$$

Moreover, when weights w *are defined over univariate literals, theories* θ *and* θ' *have identical primal graph treewidth [RS86].*

This reduction encodes weights using auxiliary parameter variables. For each literal over which a weight is defined, two set of clauses will be appended such that if the literal holds, the MI over the auxiliary variables equals the monomial weight function; otherwise, it equals 1.

Crucially, both reductions can be constructed in polynomial time. Similar efficient reductions exist for arbitrary polynomial weight functions, but can slightly increase treewidth.

Example 8. *Consider SMT($\mathcal{LRA}$) theory $(b \vee \neg b) \wedge \Gamma_i$ with literal set $\mathcal{L}$ and per-literal weight functions $\mathcal{P}$ as defined in Example 4. There exists an equivalent MI problem $\mathsf{MI}(\Delta \mid \boldsymbol{x} \cup \{\lambda_b, z_b, z_i^{(1)}, z_i^{(2)}\})$ with a weight function of 1 and without Boolean variables. Its SMT($\mathcal{LRA}$) theory Δ is shown below. Note that its primal graph remains a tree.*

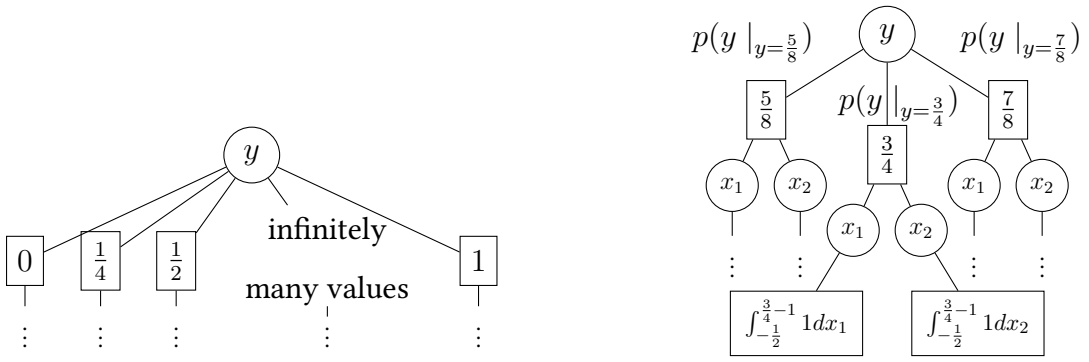
$$\Delta = \begin{cases} \Gamma_i \wedge (-1 < \lambda_b < 1) \wedge_{j=1,2} \ (0 < z_i^{(j)} < \textit{price}_i) \\ \quad \lambda_b > 0 \ \Rightarrow (0 < z_b < 1.5) \\ \neg(\lambda_b > 0) \Rightarrow (0 < z_b < 1). \end{cases}$$

6.1.2 Method

The goal of our work is to take advantage of the independence structure in SMT($\mathcal{LRA}$) theories to reduce the computational cost of model integration. Our solution is to exploit context-specific independence by search.

One obstacle is that to introduce independence in discrete search, we instantiate a variable with all values in its domain. Unfortunately, when the variable has a real domain (e.g., $y \in [0, 1]$), we cannot instantiate it with every value in its domain, since there are uncountably many (see Figure 6.4a). This basic limitation has precluded the use of search-based inference in continuous graphical models.

We overcome this problem by observing that MI is an integration over a piecewise polynomial, which can be fully recovered from a finite number of points. Specifically, for real variable y in theory θ, if we instantiate the variable y with a value α, then the MI of theory $\theta \wedge (y = \alpha)$ is the density of $\mathsf{WMI}(\theta, w)$ at $y = \alpha$. Recall that a polynomial function $p(y)$ with degree d defined over



(a) Infinite Search Tree

(b) Our Finite Search Tree

Figure 6.4: Continuous search trees for θ_2 from Example 7.

an interval I is uniquely defined by its values at $d+1$ distinct points in I, and that a closed-form expression for $p(y)$ can be recovered exactly and efficiently.

Consider again the theory Γ_i from Example 2. As shown in Figure 5.2, function $f(\alpha) = \mathsf{MI}(\Gamma_i \wedge (\mathit{sqft}_i = \alpha))$ is a piecewise polynomial with three intervals. We can recover all three polynomials from a finite number of points, and thus obtain the integration of $f(\alpha)$, that is, the model integration $\mathsf{MI}(\Gamma_i)$. This motivates the search-based model integration algorithm we develop next.

6.1.2.1 Variable Instantiation

We first show that when per-literal weight functions $\mathcal{P}$ are polynomials, WMI of theory θ can be obtained by doing search with finite instantiations on real variables.

Proposition 17. *Let y be a real variable in SMT($\mathcal{LRA}$) theory θ with a tree primal graph. If per-literal weight functions $\mathcal{P}$ are polynomials, the WMI is an integration over a univariate piecewise polynomial $p(y)$, that is,*

$$\mathsf{WMI}(\theta, w \mid \boldsymbol{x}, \boldsymbol{b}) = \int_I p(y)dy \tag{6.1}$$

where piecewise polynomial $p(y)$ is integrated over set $I = \{y^* \mid \exists \hat{\boldsymbol{x}}^*, \exists \boldsymbol{b}^* \; s.t. \; \theta(y^*, \hat{\boldsymbol{x}}^*, \boldsymbol{b}^*)$ is SAT$\}$ with $\hat{\boldsymbol{x}}$ being the remaining real variables.

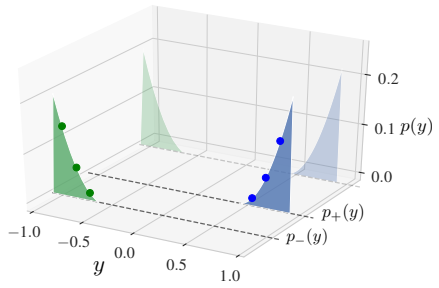


Figure 6.5: Piecewise polynomial $p(y)$ as defined in Proposition 17 for theory θ_2 from Example 7, whose integration is $\mathsf{MI}(\theta_2)$. The two polynomials $p_-(y)$ and $p_+(y)$ are unknown, but we can recover them from a finite number of points.

The set I is a union of disjoint supports for piecewise polynomial $p(y)$. We refer to these intervals as "pieces". To describe our MI algorithm, we first assume in this section that these intervals and their polynomial degrees are given. Our method to explicitly find these intervals and degrees will be given in Section 6.1.2.2.

Although Proposition 17 holds for WMI problems with polynomial per-literal weight functions in general, we use the insights from Section 6.1.1.3 to only focus on MI problems. For interval set I defined in Proposition 17, suppose we are given the interval pieces $[l, u] \in I$ and degrees d of their associated polynomials. If we instantiate variable y with $d+1$ distinct values in each piece $[l, u]$ of degree d, and solve any sub-problems recursively, we can recover polynomial $p_{l,u}(y)$ defined on interval $[l, u]$ by performing interpolation on $d + 1$ points. Finally, MI of the full theory θ can be computed as follows.

$$\mathsf{MI}(\theta, w \mid \boldsymbol{x}, \boldsymbol{b}) = \sum\nolimits_{[l,u]\in I} \int_l^u p_{l,u}(y)dy. \tag{6.2}$$

For example, consider theory θ_2 from Example 7. We can interpret $\mathsf{MI}(\theta_2)$ as an integration over piecewise polynomial $p(y)$ whose intervals $[-1, -0.5]$ and $[0.5, 1]$ both have associated degree two. After instantiating y to three values in each interval, we get two independent sub-MI problems that contain variable x_1 and variable x_2 respectively. By solving these sub-problems, we obtain three points fitted by each polynomial $p_-(y)$ and $p_+(y)$ as shown in Figure 6.5. There-

Algorithm 6 *SMI*: Search-Based Model Integration

Input: T: pseudo tree, θ: SMT($\mathcal{LRA}$) theory

Output: p: MI of theory θ

if T is a forest of trees T' **then**
  $\theta' \leftarrow$ sub-theories containing variables in T'
  return $\prod_{T'}$ *SMI*(T', θ')
$p = 0, y = root(T)$, $ST_y =$ set of subtrees below y
I = PE_NODE(θ, y)
for all polynomial piece $\{[l, u], d\} \in I$ **do**
  select $d + 1$ distinct values α_i's in $[l, u]$
  $p_i \leftarrow$ *SMI*$(ST_r, \theta \mid_{(y=\alpha_i)})$
  $p_{l,u}(y) \leftarrow$ polynomial interpolation on (α_i, p_i)'s
  $p \leftarrow p + \int_l^u p_{l,u}(y)dy$
return p

fore, we can recover both by polynomial interpolation and can obtain $\mathsf{MI}(\theta_2)$ by Equation 6.2. Figure 6.4b depicts the search space of our algorithm on interval $[0.5, 1]$.

The above discussion has shown that for MI problems, we can instantiate a real variable to finitely many values, decompose the problem into independent parts, and then solve the sub-problems recursively. Algorithm 6 follows exactly this strategy for search-based model integration. The role of pseudo trees will be explained in Section 6.2.2.4. The remaining problem is how to exactly obtain pieces $[l, u]$ and their associated degrees d in function PE_NODE. We address this problem next.

6.1.2.2 Finding Pieces via Critical Points

Recall that by Proposition 17, WMI of SMT($\mathcal{LRA}$) theory θ can be rewritten as $\mathsf{WMI}(\theta, w \mid \boldsymbol{x}, \boldsymbol{b}) = \int_I p(y)dy$ where $p(y)$ is a piecewise polynomial, set I is a union of disjoint support of

polynomials in $p(y)$, and each piece $[l, u] \in I$ is associated with a polynomial degree d. We hope that when a real variable y in theory θ is chosen to be instantiated, we can exactly find all pieces and their associated degrees for piecewise polynomial $p(y)$.

It turns out that this can be achieved. While integrating over satisfying assignments with respect to a certain variable given an SMT($\mathcal{LRA}$) theory, integration upper bounds and lower bonds are defined by its literals. Changes in integration bounds give rise to different pieces of integration and therefore result in the piecewise nature of the polynomial in Proposition 17. In our method we determine these pieces by collecting points where certain bounds meet. Further, by propagating polynomial piece and degree information in a bottom-up manner along the primal graph, we can obtain the pieces and degree for the chosen piecewise polynomial.

We will first describe our method in a basic case where there are only two real variables in the theory. Then we extend this approach to theories with tree primal graphs.

6.1.2.3 Base Case: Pieces of Two Real Variables

First we investigate a simple case where there are only two real variables x and y in SMT($\mathcal{LRA}$) theory θ. Recall that we are solving an unweighted MI problem. We would like to find pieces and associated degrees for variable y such that we can instantiate y as in Section 6.1.2.1:

$$p(y) = \int_{\theta(x,y)} 1 \, dx = \sum_{[l(y),u(y)] \in I(y)} \int_{l(y)}^{u(y)} 1 \, dx = \sum_{[l(y),u(y)] \in I(y)} u(y) - l(y)$$

where set $I(y)$ is defined as

$$\{[l(y), u(y)] \mid \forall x \in [l(y), u(y)], \theta(x, y) \text{ is SAT}\}. \tag{6.3}$$

That is, for any fixed value y^*, the set $I(y^*)$ consists of intervals of consistent values for variable x. For any $[l(y), u(y)] \in I(y)$, it gives a pair of integration bounds for variable x. Further by integrating over x we can obtain a polynomial with respect to variable y.

Each piece $[l, u]$ corresponds to a certain class of values that gives the same symbolic integration bounds to variable x. The two values $y = l$ and $y = u$ are endpoints of the piece only if

integration bound set $I(y)$ changes at these points, since the piecewise polynomial $p(y)$ is defined by these bounds. That is, for arbitrarily small ϵ, we have $I(l-\epsilon) \neq I(l+\epsilon)$, and it also holds at point $y = u$. We formally define critical points below.

Definition 4. ***(Critical Point)*** *Let θ be an SMT($\mathcal{LRA}$) theory with two real variables, and denote one of the real variables by y. Let $I(y)$ be an integration bound set as defined in Equation 6.3. Then $y = \alpha$ is a critical point if for arbitrarily small ϵ, it holds that $I(\alpha-\epsilon) \neq I(\alpha+\epsilon)$.*

Remark. The comparison of set $I(y)$ is done symbolically. That is, for two distinct values α, β, we say $I(\alpha) = I(\beta)$ if they have the same set of symbolic integration bounds. For example, if at $y = \alpha$, $I(y) = \{[1, y]\}$ and at $y = \beta \neq \alpha$, $I(x) = \{[1, y]\}$, it holds that $I(\alpha) = I(\beta)$. However, if at $y = \alpha$, $I(y) = \{[1, y]\}$ and at $y = \beta$, $I(y) = \{[y, 2]\}$, then we say $I(\alpha) \neq I(\beta)$.

Our idea is that, if we can find all critical points $y = \alpha$ where the set $I(y)$ changes, then we can partition real domains of y into disjoint intervals, such that any support of piecewise polynomial $p(y)$ is either one of these intervals or a union of some intervals. For the resulting interval $[l, u]$, we can apply an SMT($\mathcal{LRA}$) solver to $\theta' = \theta \wedge (l < y < u)$ to check whether it is a satisfiable piece of function $p(y)$; if this is true, we can obtain the polynomial degree of $p_{l,u}(y)$ defined over this piece by simply traversing theory θ'.

6.1.2.4 General Case: Pieces of Tree Structures

Given an SMT($\mathcal{LRA}$) theory θ with a tree-shape primal graph G, our goal is to enumerate pieces and their associated degrees for the root variable y, building on the algorithm we developed in the base case above. This can be done in a bottom-up manner with tree primal graphs.

Specifically, we first partition theory θ into sub-theories $\theta_{r,c}$ and θ_{G_c} for each c, such that $\theta = \bigwedge_c (\theta_{r,c} \wedge \theta_{G_c})$, where variables c are the child variables of root r, and graph G_c is the sub-tree rooted at variable c. Each theory $\theta_{r,c}$ contains only variables r and c, on which we can apply the enumeration for the base case above, and each theory θ_{G_c} contains only variables in sub-tree G_c. This is possible provided that the primal graph of theory θ has a tree structure, which is why our algorithm is restricted to SMT($\mathcal{LRA}$) theories with tree-shaped primal graphs.

For each child variable c, we first obtain its pieces with respect to theory θ_{G_c} in a recursive way. Then we can apply our enumeration algorithm for two-variable theory PE_EDGE to theory $\theta_{r,c}$ with the given pieces of variable c. What we would get are sets of pieces for each child variable c. To be consistent with theory θ, we need to take intersections of these sets which we refer to as the shattering operation. Finally, the resulting intersections are pieces and polynomial degrees for root variable r.

As described above, our piece enumeration algorithm is applicable to MI problems for theories with tree primal graphs. Moreover, it is also applicable to WMI problems whose SMT theory has a tree primal graph and whose per-literal weights are monomials over univariate literals as described in Section 6.1.1.3, since our reduction process can preserve the tree structure of the primal graph.

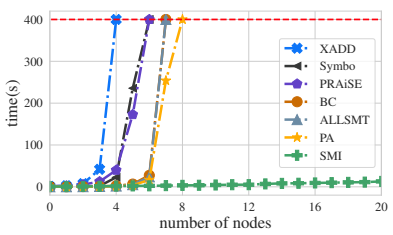
6.1.2.5 Complexity Analysis

Inference over networks involving real variables raises considerable challenges for inference, and network structures that are tractable in the discrete case, such as polytrees, give rise to NP-hard inference problems in the hybrid case [KF09]. We show that the complexity of our algorithm is mainly exponentially bounded by the tree height of the primal graph.

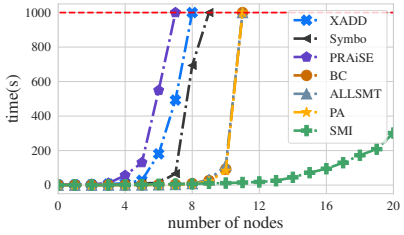
Our search algorithm for MI needs to choose which variables to instantiate first. This choice can be based on a tree data structure that orders the variables. Such a tree characterizes the computational complexity as it does for discrete And/Or search algorithms. We first formally defined the tree that helps guide our search.

Definition 5. ***(Pseudo Tree)*** *Given an undirected graph G with vertices and edges (V, E_G), a pseudo tree for G is a directed rooted tree T with vertices and edges (V, E_T), such that any edge e that is in G but not in T must connect a vertex in T to one of its ancestors.*

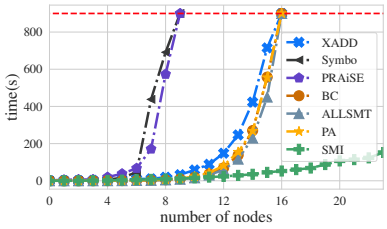
That is, edge $e = (v_1, v_2)$ such that $e \in E_G$ and $e \notin E_T$ implies that either vertex v_1 is an ancestor of vertex v_2 in T or vertex v_2 is an ancestor of vertex v_1 in T. Note that the pseudo tree



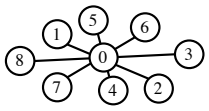
(a) Runtime on star graphs.



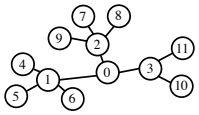
(b) Runtime on 3-ary tree graphs.



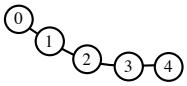
(c) Runtime on path graphs.



(d) Star graph with nodes $n = 8$.



(e) 3-ary tree with nodes $n = 12$.



(f) Path graph with nodes $n = 5$.

Figure 6.6: (a)-(c) MI execution time on SMT($\mathcal{LRA}$) with tree primal graphs. (d)-(f) Example tree primal graphs.

has the same set of vertices as G. Such a pseudo tree guides SMI (Algorithm 6) in deciding which variable to instantiate, and when to decompose.

Next, we analyze the complexity of SMI. Since our algorithm performs search, its time and space complexity is characterized by the size of its search space. Our analysis does not take caching improvements into consideration.

Theorem 4. ***(Size of Search Space)*** *Consider an SMT($\mathcal{LRA}$) theory θ with a tree-shaped primal graph with height h_p, and a pseudo tree T with l leaves and height h_t. Let m be the number of $\mathcal{LRA}$ literals in θ, and n be the number of real variables. Then the size of the SMI search space is $\mathcal{O}(l \cdot (n^3 \cdot m^{h_p})^{h_t})$.*

Hence, we can conclude that the complexity of our algorithm is bounded exponentially by tree heights of both the primal graph and pseudo tree. In fact, for any tree-shaped primal graph, we can always choose a pseudo tree whose height h_t is $\mathcal{O}(\log n)$ to guide the search [DM07]. Moreover, the number of leaves l in pseudo tree T is no larger than the number of nodes n. Thus, we have the following corollary.

Corollary 2. *Following the notation in Theorem 4, with properly chosen pseudo tree T whose tree height h_t is $\mathcal{O}(\log n)$, the size of the search space generated by SMI is $\mathcal{O}\left(n^{1+3\log n+h_p \log m}\right)$.*

Therefore, the complexity of our algorithm is mainly decided by tree heights of primal graphs h_p. In the worst case when tree primal graphs have height $\mathcal{O}(n)$ – for instance path graphs, whose tree height is n when rooted at the start node – then the worst-case complexity of our algorithm is $\mathcal{O}(n^{n \log m})$ by Corollary 2. That is, the time complexity is worst-case super-exponential.

In cases when the tree primal graph has tree height of size $\mathcal{O}(\log n)$, the complexity of our algorithm is $\mathcal{O}(n^{1+(3+\log m)\log n})$ which is of quasi-polynomial complexity, and considered to be efficient. Trees with tree height in $\mathcal{O}(\log n)$ are a general class of trees used in various models. Balancing trees like AVL trees and full k-ary trees are of tree height $\mathcal{O}(\log n)$. Another example is a star graph, which has one internal node and all other nodes as leaves. This graph corresponds to the well-known naive Bayes structure for directed graphical models. It is the primal graph of a theory modeling independent variables predicting one and the same dependent (class) variable. The tree height of star graphs is constant 1 when choosing the internal node as root. Hence, our algorithm runs efficiently on such WMI problems.

6.1.3 Empirical Evaluation

We analyze the performance of our search-based MI algorithm on SMT($\mathcal{LRA}$) theories with tree primal graphs. First, we show that our algorithm is efficient for theories whose primal graphs have constant tree heights, or tree heights of log scale w.r.t. the number of real variables n. For theories whose primal graph has tree height in $\mathcal{O}(n)$ – the cases where our algorithm has super-exponential worst-case complexity in theory – empirical results show that our algorithm still runs efficiently. We also consider a more complex house price model where house sizes are dependent, as opposed to those in Example 6. Moreover, the house price model has non-trivial weight functions that our algorithm first reduces to a MI problem as outlined in Section 6.1.1.3. We compare our algorithm to alternative WMI solvers and conclude that it significantly outperforms existing solvers on these benchmarks.

Benchmarks We compare our algorithm (SMI) with other WMI solvers. The block-clause-strategy-based solver (BC) [BPB15] iteratively generates new models by adding the negation of the latest model to the formula for the following iteration. The all-satisfying-assignments-based solver (ALLSMT) [BBP16] first generates the set of all $\mathcal{LRA}$-satisfiable total truth assignments on atoms that propositionally satisfy the theory. The implementation of [SOG16] (PRAiSE) is a variable-elimination-based solver. The predicate-abstraction-based solver (PA) [MPS17] exploits the power of SMT-based predicate abstraction to reduce the number of models to be integrated over. Both the extended algebraic-decision-diagram-based solver (XADD) [KMS18] and sentential-decision-diagram-based solver (Symbo) [ZDD19] use circuit-based compilation languages and exploits the circuit structures.

6.1.3.1 Tree Primal Graphs

We investigate the performance of our algorithm on SMT($\mathcal{LRA}$) theories with three types of tree primal graphs: 1) star graphs, consisting of one center node connected to all other nodes, and no other connections; 2) full 3-ary trees, whose non-leaf vertices have exactly three children and all levels are full except for some rightmost position of the bottom level; 3) path graphs, consisting of linearly connected nodes. These structural constraints arise naturally in data and many probabilistic graphical modeling problems.

For each graph type, given a number of nodes n, we introduce n real variables which we denote by $\boldsymbol{x} = \{x_0, x_1, \cdots, x_{n-1}\}$ with bounded domains $\forall i, (-1 \leq x_i \leq 1)$. Denote the graph by $G = (V, E)$ where $V = \{0, 1, \cdots, n-1\}$ is the vertex set and $E = \{(i, j), i, j \in V\}$ the edge set. We perform MI for the following theories and increasing n.

$$\theta(\boldsymbol{x}) = \begin{cases} \bigwedge_{i \in V} \ (-1 \leq x_i \leq 1) \\ \bigwedge_{(i,j) \in E} \ ((x_i + 1 \leq x_j) \vee (x_j \leq x_i - 1)) \end{cases}$$

Figure 6.6 shows example primal graphs and the execution time of experiments comparing SMI with baselines.

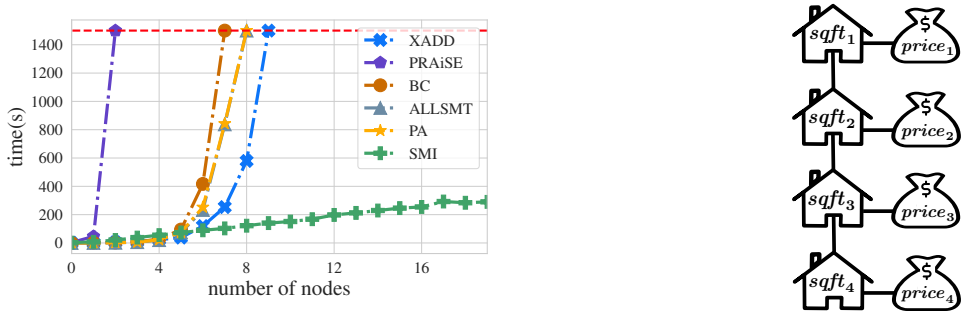


Figure 6.7: Runtime and primal graph for house price model.

For MI over theories with all three types of tree primal graphs, our algorithm significantly outperform other WMI solvers in terms of execution time. The runtime curves of other solvers grow seemingly exponentially while our curve grows slowly with the number of real variables. For theories with star graphs and full 3-ary trees as primal graphs, the time curves of SMI are consistent with the complexity analysis in Section 6.2.2.4 stating that our algorithm has quasi-polynomial complexity. For theories with path graphs as primal graphs, which are still sparse graphs, we perform caching and the runtime curve grows slowly, even though our worst-case analysis allows for a super-exponential time complexity.

6.1.3.2 House Price SMT($\mathcal{LRA}$) Model

In Example 6 we performed MI for multiple houses based on extreme independence assumptions. Now we consider a more complicated case where houses are not independent and there are Boolean variables in the SMT($\mathcal{LRA}$) model. Moreover, we choose non-trivial per-literal weight functions in order to evaluate our algorithm for reducing WMI to unweighted MI problems.

Specifically, we consider n houses that are located along a street. Each house i has its price and square footage model as in Example 2. Also, we enforce the constraint that square footage between two neighboring houses should not vary too much and we use a Boolean variable b to indicate whether or not these houses are located in an urban area. This gives the following SMT

theory.

$$\Gamma_{\textit{street}} = \begin{cases} (b \vee \neg b) \wedge \bigwedge_{i=1}^{n} \Gamma_i \\ \bigwedge_{i=1}^{n-1} (\textit{sqft}_i \leq \textit{sqft}_{i+1} + \textit{offset}) \end{cases}$$

with *offset* a constant characterizing maximum difference in square footage between two neighboring houses. For weights w, consider the set of literals $\mathcal{L} = \{b\} \cup \{0 < \textit{price}_i < 3000, i = 1, \cdots, n\}$ and per-literal weight functions $\mathcal{P} = \{p_b\} \cup \{p_{(0<\textit{price}_i<3000)}, i = 1, \cdots, n\}$, with $p_b(\boldsymbol{x}) = 1.5$ and $p_{(0<\textit{price}_i<3000)}(\boldsymbol{x}) = \textit{price}_i^2$ for all i. Then, in worlds where all literals in $\mathcal{L}$ are satisfied, our weight function is $1.5 \prod_{i=1}^{n} \textit{price}_i^2$. In worlds where b is false but other literals are satisfied, the weight function is $\prod_{i=1}^{n} \textit{price}_i^2$. Figure 6.7 shows an example primal graph and WMI runtime for this house price model.

6.2 Message-Passing WMI Solver

While SMI directly exploits the problem structure encoded in primal graphs, in order to perform a tractable reduction, SMI is limited to a restricted set of weights, and hence a very narrow set of WMI problems. To build more powerful exact WMI solvers, we make the following contributions. First, we theoretically trace the boundaries for the classes of tractable WMI problems known in the literature. Second, we expand these boundaries by devising a polytime algorithm for exact WMI inference on a class that is strictly larger than the class previously known to be tractable. Our proposed WMI solver, called MP-WMI, adopts a novel message-passing scheme for WMI problems. It is able to exactly compute all the variable marginal densities at once. By doing so, we are able to scale inference beyond the capabilities of all current exact WMI solvers. Moreover, we can amortize inference *inter-queries* for rich SMT queries that conform to the problem structure.

6.2.1 Tractability Analysis

The major efforts in advancing WMI inference have been so far concentrated on devising sophisticated WMI solvers to deliver exact inference routines for general scenarios without investigating the effect of the structure of a WMI problems on its complexity. Little to no attention has gone

to formally understand which classes of WMI problems can be guaranteed to be solved exactly and in polynomial time, that is, *tractably*.

One notable exception can be found in [ZB19] where the search-based MI (SMI) solver is introduced. WMI problems for which SMI guarantees polytime exact inference constitute the first class of tractable WMI. Intuitively, SMI solves MI problems by using search to leverage the conditional independence among variables.

As in [ZB19] we characterize the structure of an SMT formula via its *primal graph*.

Definition 6. ***(Primal graph of SMT)*** *The primal graph of an SMT formula Δ is an undirected graph $\mathcal{G}_\Delta$ whose vertices are variables in formula Δ and whose edges connect any two variables that appear in a same clause in the formula Δ.*

An example primal graph of the SMT formula in Example 1 is shown in Figure 5.1. The SMI solver guarantees polynomial time execution for the class of MI problems with certain tree-shaped primal graphs, which we denote as treeMI.

Definition 7. *(*treeMI *Problem Class) Let* treeMI *be the set of all MI problems over real variables whose SMT formula Δ induces a primal graph $\mathcal{G}_\Delta$ with treewidth one and with bounded diameter d. Problems in* treeMI *can be solved in polytime via SMI [ZB19].*

Note that in Definition 7 the primal graph diameter here plays the role of a constant since, otherwise, SMI complexity can be worst-case exponential in diameter d. In the following we will try to answer if larger classes than treeMI are still amenable to tractable inference. We start by demonstrating a novel result that states the hardness of a larger class of MI problems, still focusing on dependencies between two variables, but allowing for non-tree-shaped primal graphs.

Definition 8. *(*2MI *Problem Class) Let* 2MI *be the set of all MI problems over real variables whose SMT formula Δ is a conjunction of clauses comprising at most two variables.*

Note that a clause comprising at most two variables can be a conjunction of arbitrarily many literals. Moreover, when there are more than two variables in a clause, in the primal graph there

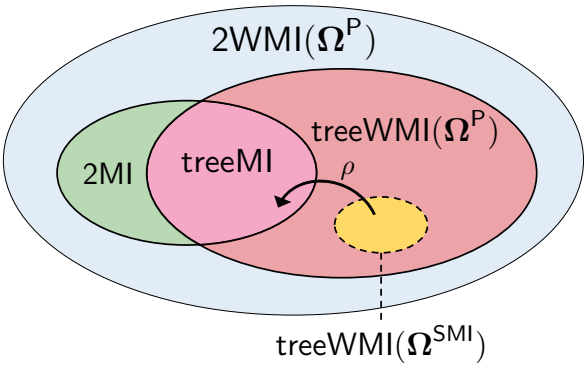


Figure 6.8: **The current landscape of classes of WMI problems.** We enlarge the boundaries of tractable WMI inference from treeMI to treeWMI and prove the hardness of 2MI and 2WMI.

must be a loop and thus the treewidth of the primal graph is larger than one. Hence all MI problems with tree-shaped primal graph must be in the class 2MI.

Theorem 5. *(Hardness of* 2MI*) Given an MI problem in* 2MI *with an SMT formula* Δ*, computing* $\mathsf{MI}(\Delta)$ *is #P-hard.*

Sketch of Proof. The proof is done by reducing the #P-complete problem #2SAT to an MI problem in 2MI with an SMT formula Δ such that counting the number of satisfying assignments to the #2SAT problem equates to the MI of formula Δ. □

From Theorem 5 it follows that the problem class $\mathsf{2WMI}(\mathbf{\Omega})$, i.e., the WMI problems with SMT formulas being a conjunction of clauses comprising at most two variables, and with per-literal weights in weight function family $\mathbf{\Omega}$, is also hard since class 2MI is a sub-class of $\mathsf{2WMI}(\mathbf{\Omega})$. We revert our attention to WMI problems exhibiting a dependency tree structure. Notice that for WMI problems, the tractability not only depends on the logical structure defined by the SMT formulas, but also the statistical structure defined by weight functions. Next in our analysis, we take into consideration the weight function families. Analogously to what Definition 7 states, we introduce the notion of $\mathsf{treeWMI}(\mathbf{\Omega})$ with the associated weight function family specified as follows.

Definition 9. *(*treeWMI($\mathbf{\Omega}$) *Problem Class) Let* treeWMI($\mathbf{\Omega}$) *be the set of all WMI problems over real variables whose SMT formula Δ induces a primal graph $\mathcal{G}_\Delta$ with treewidth one and with bounded diameter d, and whose per-literal weights are in a function family $\mathbf{\Omega}$.*

[ZB19] propose a WMI-to-MI reduction such that some treeWMI($\mathbf{\Omega}$) problems with polynomial weights are reduced in polynomial time to treeMI problems amenable to tractable inference by the SMI solver. Intuitively, the reduction process introduces auxiliary continuous variables and SMT formulas over these variables to encode the polynomial weight functions. We refer the readers to [ZB19] for a detailed description of the reduction. However, as shown next, the set of treeWMI problems that can be reduced to treeMI is rather limited.

Definition 10. *($\mathbf{\Omega}^{\text{SMI}}$ Weight Function Family) Let $\mathbf{\Omega}^{\text{SMI}}$ be the family of per-literal weight functions that are monomials associated with either (i) univariate literals or (ii) a literal that appears exclusively in a unit clause, i.e., a clause consisting of a single literal.*

Theorem 6. *Let ρ be the polytime WMI-to-MI reduction for* treeWMI($\mathbf{\Omega}$) *problems as defined in [ZB19]. Then the image $\{\rho(\nu) \mid \nu \in \text{treeWMI}(\mathbf{\Omega})\} \subset$ treeMI if-and-only-if $\mathbf{\Omega} \subset \mathbf{\Omega}^{\text{SMI}}$.*

Sketch of Proof. The necessary condition can be proved by the reduction process and the sufficient one can be proved by contradiction. □

Therefore, the SMI solver is limited to a rather restricted subset of treeWMI($\mathbf{\Omega}$) since from the definition of $\mathbf{\Omega}^{\text{SMI}}$ we can tell that it is a strict subset of monomial per-literal weights. In order to enlarge the tractable class of WMI problems, next we will define a rich family of weight functions.

Definition 11. *(Tractable Weight Conditions) Let $\mathbf{\Omega}$ be a family of per-literal weight functions. We say that the tractable weight conditions (TWC) hold for $\mathbf{\Omega}$ if we have:*

(i) closedness under product: $\forall f, g \in \mathbf{\Omega}$, $f \cdot g \in \mathbf{\Omega}$;

(ii) tractable symbolic integration: $\forall f \in \mathbf{\Omega}$, the symbolic antiderivative of function f can be tractably computed by symbolic integration;

(iii) closedness under definite integration: $\forall f \in \mathbf{\Omega}$ *with its antiderivative denoted by* F*, given integration bounds* $l(x), u(x)$ *in* $\mathcal{LRA}$ *with* $x \in \mathcal{X}$*,* $F(u(x)) - F(l(x)) \in \mathbf{\Omega}$*.*

Some example weight function families that satisfy TWC include the polynomial family, exponentiated linear function family and the function family resulting from their product. Moreover note that piecewise function families, when pieces belong to the above families, also satisfy TWC. It turns out that the weight function families that satisfy TWC subsume and extend all the parametric weight functions adopted in the WMI literature so far. The following proposition is a direct result from the fact that the piecewise polynomial weight family $\mathbf{\Omega}^{\mathsf{P}}$ is a strict superset of the family $\mathbf{\Omega}^{\mathsf{SMI}}$.

Proposition 18. *Let* $\mathbf{\Omega}^{\mathsf{P}}$ *be the piecewise polynomial weight function family. The WMI problem class* $\mathsf{treeWMI}(\mathbf{\Omega}^{\mathsf{P}})$ *is a strict superset of problem class* $\mathsf{treeWMI}(\mathbf{\Omega}^{\mathsf{SMI}})$*.*

Theorem 7. *If a weight function family* $\mathbf{\Omega}$ *satisfies TWC as in Definition 11, WMI problems in class* $\mathsf{treeWMI}(\mathbf{\Omega})$ *are tractable, i.e., they can be solved in polynomial time.*

The proof to the above theorem is provided in the next two sections by construction where in Section 6.2.2 we proposed our WMI solver, called MP-WMI, operating on WMI problems in $\mathsf{treeWMI}(\mathbf{\Omega})$ with its complexity analysis in Section 6.2.2.4. A summary of the WMI problem classes is shown in Figure 6.8.

6.2.2 Method

Message passing on tree-structured graphs has achieved remarkable attention in the PGM literature [Pea88, KFL01]. Its classical formulation and efficiency relies on compact factor representations allowing easy computations. However, adapting existing message-passing algorithms to WMI inference is non-trivial. This is due to the fact that inference is computed in a hybrid structured space with logical and arithmetic constraints. We present our message-passing scheme by first deriving a factorized representation of WMI problems.

6.2.2.1 Factor Graph Representation of WMI

In the literature of WMC, *incidence graphs* are proposed to characterize the structure of problems defined by Boolean CNF formulas [SS10]. Incidence graphs are bipartite graphs with clause nodes and variable nodes, where a clause and a variable node are joined by an edge if the variable occurs in the clause. We derive the analogous representation for the more general SMT formulas, which we then turn into a factor graph of WMI problems.

Recall that for the joint distribution represented by a WMI problem, the support is defined by the logical constraints and the unnormalized density is defined by weight functions. In the following, we first factorize the SMT formula Δ of a WMI problem $\mathsf{WMI}(\Delta, w)$ in the class $\mathsf{treeWMI}$:

$$\Delta = \bigwedge_{i \in \mathcal{V}} \Delta_i \wedge \bigwedge_{i,j \in \mathbb{E}} \Delta_{ij} \tag{6.4}$$

where the set $\mathcal{V}$ is the index set of variables and the set $\mathbb{E}$ is the index pairs of variables in the same clause. Then a WMI problem can be conveniently represented as a bipartite graph, known as factor graph, that includes two sets of nodes: variable nodes X_i, and factor nodes $f_{\mathcal{S}}$, where $\mathcal{S}$ denotes a factor scope, i.e., the set of indices of the variables appearing in it. A variable node X_i is connected to a factor node $f_{\mathcal{S}}$ if and only if $i \in \mathcal{S}$. Specifically, the factors are defined as follows:

$$f_{\mathcal{S}}(\boldsymbol{x}_{\mathcal{S}}) = \prod_{\Gamma \in \mathit{CLS}(\Delta_{\mathcal{S}})} [\![\boldsymbol{x}_{\mathcal{S}} \models \Gamma]\!] \prod_{\ell \in \mathit{LITS}(\Gamma)} w_\ell(\boldsymbol{x}_{\mathcal{S}})^{[\![\boldsymbol{x}_{\mathcal{S}} \models \ell]\!]} \tag{6.5}$$

where $\boldsymbol{x}_{\mathcal{S}}$ denotes the restriction of $\boldsymbol{x}$ to the variables in factor $f_{\mathcal{S}}$ and analogously $\Delta_{\mathcal{S}}$ is the restriction of formula Δ to the clauses over the variables in $\mathcal{S}$. Here, the set of clauses in the SMT formula Δ is denoted by $\mathit{CLS}(\Delta)$, and the set of literals in a clause Γ is denoted by $\mathit{LITS}(\Gamma)$. Intuitively, the factors include the parameterized densities as in the classic PGM literature, here represented by the per-literal weights, but also the structure enforced by the logical constraints in the SMT formula, via the indicator functions. Figure 6.9 shown an example of a factor graph.

As in every tree-shaped factor graphs, we define an unnormalized joint distribution corre-

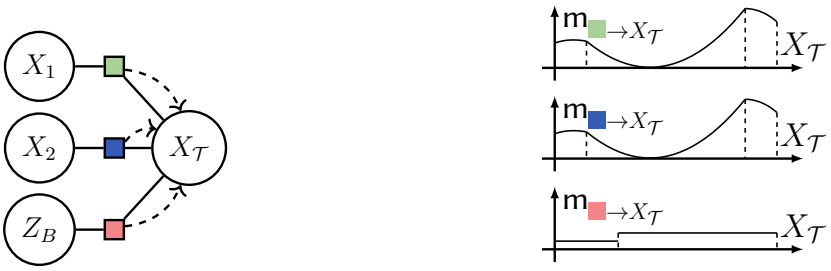


Figure 6.9: **Factor graph** (left) of formula Γ with two players and **piecewise polynomial messages** (right) sent from the three factor nodes to variable node $X_\mathcal{T}$ when solving the WMI in Example 3 by MP-WMI.

sponding to the WMI problem in the form of a product of factors as follows.

$$p(\boldsymbol{x}) = \prod_{\mathcal{S}} f_\mathcal{S}(\boldsymbol{x}_\mathcal{S}) = \prod_{i \in \mathcal{V}} f_i(X_i) \prod_{i,j \in \mathbb{E}} f_{ij}(X_i, X_j) \tag{6.6}$$

By construction, it is easy to see that the normalization constant of such a distribution equals computing the corresponding weighted model integral.

Proposition 19. *Given a problem* $\mathsf{WMI}(\Delta, w)$ *in* treeWMI*, let* $p(\boldsymbol{x})$ *being the unnormalized joint distribution as defined in Equation 6.6. Then the partition function of distribution* $p(\boldsymbol{x})$ *is equal to* $\mathsf{WMI}(\Delta, w)$.

6.2.2.2 Message-Passing Scheme

Deriving a message-passing scheme for WMI poses unique and considerable challenges. First, different from discrete domains, on continuous or hybrid domains one generally does not have universal and compact representations for messages, and logical constraints in WMI make it even harder to derive such representations. Moreover, marginalization over real variables requires integration over polytopes, which is already #P-hard [DF88]. The integration poses the problem of whether the messages defined are integrable and how hard it is to perform the integration. In the following part, we will present our solutions to these challenges by first describing a general message-passing scheme for WMI and then investigating of which form the messages are, given certain weight families.

Given the factorized representation of WMI in Section 6.2.2.1, our message-passing scheme, called MP-WMI and summarized in Algorithm 7, comprises exchanging messages between nodes in the factor graph. Before the message passing starts, we choose an arbitrary node in the factor graph as root and orient all edges away from the root to define the message sending order. MP-WMI operates in two phases: an upward pass and a downward one. First, we send messages up from the leaves to the root (upward pass) such that each node has all information from its children and then we incorporate messages from the root down to the leaves (downward pass) such that each node also has information from its parent. The messages are formulated as follows.

Proposition 20. *Both messages* $\mathsf{m}_{f_{ij}\to X_i}$ *from factor node to variable node and messages* $\mathsf{m}_{X_i\to f_{ij}}$ *from variable node to factor node have iterative formulations as follows.*

(i) $\mathsf{m}_{f_{ij}\to X_i}(x_i) = \int f_{ij}(x_i, x_j)\cdot \mathsf{m}_{X_j\to f_{ij}}(x_j)\, dx_j$*;*

(ii) $\mathsf{m}_{X_i\to f_{\mathcal{S}}}(x_i) = \prod_{f_{\mathcal{S}'}\in \mathsf{neigh}(X_i)\setminus f_{\mathcal{S}}} \mathsf{m}_{f_{\mathcal{S}'}\to X_i}(x_i)$.

For the start of sending messages, when a leaf node is a variable node X_i, the message that it sends along its one and only edge to a factor $f_{\mathcal{S}}$ is $\mathsf{m}_{X_i\to f_{\mathcal{S}}}(c_i) = 1$; in the case when a leaf node is a factor node f_i, the message from the factor node f_i to a variable node X_i is $\mathsf{m}_{f_i\to X_i}(x_i) = f_i(x_i)$. Even though the weight function family is not specified here, it can be shown that when the integration in Proposition 20 is well-defined, i.e., the integrands are integrable, then the messages are univariate piecewise functions, which is a striking difference with classical message-passing schemes.

Proposition 21. *For any problem in* treeWMI*, the messages as in Proposition 20 are univariate piecewise functions.*

The specific form of messages also depends on the chosen weight function family as mentioned in Section 6.2.1. For example, when the weight functions are chosen to be polynomials, the messages are piecewise polynomials, as in the example in Figure 6.9. We show how to compute the piecewise polynomial messages in Algorithm 8 with functions *critical-points* and *get-msg-pieces* as subroutines to compute the numeric and symbolic integration bounds for the message

Algorithm 7 MP-WMI(Δ)

$\mathbf{V}_{\mathsf{up}} \leftarrow$ sort variable nodes in factor graph
for [**do**upward pass]**each** $X_i \in \mathbf{V}_{\mathsf{up}}$
    send-message(X_i,$f_{i,\mathsf{pa}(i)}$)
    send-message($f_{i,\mathsf{pa}(i)}$, $X_{\mathsf{pa}(i)}$)
$\mathbf{V}_{\mathsf{down}} \leftarrow$ sort nodes in set $\mathbf{V}_{\mathsf{up}}$ in reverse order
for [**do**downward pass]**each** $X_i \in \mathbf{V}_{\mathsf{down}}$
    for each $X_c \in \mathsf{ch}(X_i)$ **do**
        send-message(X_i, f_{ic})
        send-message(f_{ic}, X_c)
return $\{\mathsf{m}_{X_i \to f_s}, \mathsf{m}_{f_s \to X_i}\}_{(x_i, f_s) \in \mathbb{E}}$

pieces. Both of them can be efficiently implemented, see [ZB19] for details. The actual integration of the polynomial pieces can be efficiently performed symbolically, as supported by many scientific computing packages.

When MP-WMI terminates, the information stored in the obtained messages is sufficient to compute the unnormalized marginals for each variable and it is independent of the choice of root. Moreover, the integration of unnormalized marginals equals to $\mathsf{WMI}(\Delta, w)$.

Proposition 22. *Let Δ be an SMT formula with a tree factor graph and with per-literal weights w. For any variable X_i, the unnormalized marginal $p(x_i)$ is*

$$p(x_i) = \prod_{f_{\mathcal{S}} \in \mathsf{neigh}(X_i)} \mathsf{m}_{f_{\mathcal{S}} \to X_i}(x_i).$$

Moreover, the partition function for any x_i is the WMI of SMT formula Δ, i.e., $\mathsf{WMI}(\Delta, w) = \int_{x_i} p(x_i) dx_i$.

6.2.2.3 Amortization

We will show that by leveraging the messages pre-computed in MP-WMI, we are able to speed up (amortize) inference time over multiple queries on formula Δ. More specifically, when answering

Algorithm 8 send-message(s, t)

if $s = X_i$ and $t = f_{ij}$ **then**
  Return $\prod_{f_{s'} \in \mathsf{neigh}(X_i) \setminus f_{ij}} \mathsf{m}_{f_{s'} \to X_i}$
else if $s = f_{ij}$ and $t = X_i$ **then**
  $\mathcal{P} \leftarrow$ critical-points($\mathsf{m}_{X_j \to f_{ij}}, \Delta_{ij}$)
  $\mathcal{I} \leftarrow$ intervals-from-points($\mathcal{P}$)
  for interval $I \in \mathcal{I}$ consistent with formula Δ_{ij} **do**
    $\langle l_s, u_s, p \rangle \leftarrow$ get-msg-pieces($\mathsf{m}_{X_j \to f_{ij}}, I, w$)
    $p'(x_i) \leftarrow \int_{l_s}^{u_s} p(x_i, x_j)\, dx_j$
    $\mathsf{m}_{f_{ij} \to X_i} \leftarrow \mathsf{m}_{f_{ij} \to X_i} + [\![x_i \in I]\!] \cdot p'(x_i)$
return $\mathsf{m}_{s \to t}$

queries that do not change the tree structure in the factor graph of formula Δ, we only need to update messages that are related to the queries while other messages are pre-computed. Some examples are SMT queries on a node variable or queries over a pair of variables that are connected by an edge in the factor graph, since these queries either add leaf nodes or do not change existing nodes. Thus we can reuse the local information encoded in messages.

Proposition 23. *Let* $\mathsf{WMI}(\Delta, w)$ *be a problem in* treeWMI*, and* Φ *be an SMT query over a factor* f_s^* *involving a variable* $X_i \in \mathcal{X}$*. Given pre-computed messages* $\{\mathsf{m}_{f_{\mathcal{S}} \to X_i}\}_{f_{\mathcal{S}} \in \mathsf{neigh}(X_i)}$*,*

$$\mathsf{WMI}(\Delta \wedge \Phi) = \int_{\mathbb{R}} \mathsf{m}^*_{f_s^* \to X_i}(x_i) \cdot \prod_{f_s \in \mathsf{neigh}(X_i) \setminus f_s^*} \mathsf{m}_{f_s \to X_i}(x_i) dx_i$$

with message $\mathsf{m}^*_{f_s^* \to X_i}$ *computed over factor* $f_s^*(\boldsymbol{x}_s) := f_s(\boldsymbol{x}_s) \cdot [\![\boldsymbol{x}_s \models \Phi]\!]$ *as in Proposition 20.*

Pre-computing messages can dramatically speed up inference by amortization, as we will show in our experiments, especially when traversing the factor graph is expensive or the number of queries is large.

6.2.2.4 Complexity Analysis

This section provides a complexity analysis of our proposed WMI solver MP-WMI. Given the SMT formula Δ with a tree factor graph with a chosen root node, each factor node would be traversed exactly once in each phase of the message-passing scheme. We denote the set of directed factor nodes by $\mathcal{F} := \{\overrightarrow{f_s}\} = \{f_s^+, f_s^- \mid f_s \in \mathcal{V}\}$ where f_s^+ denotes the factor node f_s visited in the upward pass and f_s^- denotes the one visited in downward pass respectively.

To characterize the message-passing scheme, we define a nilpotent matrix A as follows. The matrix $A \in \mathbb{N}^{|\mathcal{F}|\times|\mathcal{F}|}$ has both its columns and rows denoted by the factor nodes in set $\mathcal{F}$. At each column denoted by $\overrightarrow{f_s}$, only entries at rows denoted by factor nodes visited right after $\overrightarrow{f_s}$ are non-zero.

Proposition 24. *The nilpotent matrix A as described above has its order at most the diameter of the factor graph.*

Next we show how to define the non-zero entries in matrix A with parameters about the SMT formulas in WMI problems.

Proposition 25. *Suppose that the two variables X_i and X_j are connected in the factor graph by a factor f_{ij} associated with a sub-formula Δ_{ij} of size c, then in MP-WMI:*

(i) the number of pieces in message $\mathsf{m}_{X_i\to f_{ij}}$ is bounded by $\sum m_s$, where m_s is the number of pieces in message $\mathsf{m}_{f_s\to X_i}$ with $f_s \in \mathsf{neigh}(X_i)\backslash f_{ij}$;

(ii) the number of pieces in message $\mathsf{m}_{f_{ij}\to X_j}$ is bounded by $2mc + c^2$ with m being the number of pieces in message $\mathsf{m}_{X_i\to f_{ij}}$.

Now we show how to use the matrix A to bound the number of pieces in messages. We define the non-zero entries in the nilpotent matrix A to be $2c$ with c being a constant that bounds the size of sub-formulas associated to factors. Define a vector $v^{(t)} \in \mathbb{N}^{|\overrightarrow{\mathbb{E}}|}$ for the state of the message-passing scheme at step t – by state it means that each entry in vector $v^{(t)}$ is denoted by a factor node in set $\mathcal{F}$ and the entry denoted by $\overrightarrow{f_s}$ bounds the number of pieces in the message

sent to f_s in the MP-WMI. For the initial state vector $v^{(0)}$, it has all non-zero entries to be c, the constant bounding the sub-formula size, and these entries are those denoted by $\overrightarrow{f_s} = f_s^+$ with factor node f_s connected with a leaf.

Proposition 26. *Let A be the nilpotent matrix and v the initial state vector as described above. Also let $v^{(t)} := Av^{(t-1)} + c^2 \cdot \text{sgn}(Av^{(t-1)})$ with* sgn *being the sign function. Then each entry in vector $v^{(t)}$ denoted by $\overrightarrow{f_s}$ bounds the number of pieces in the message $\mathsf{m}_{X_i \to f_s}$ received by factor f_s from some variable node X_i at step t in MP-WMI.*

Proposition 27. *Let A be the nilpotent matrix and $v^{(t)}$ the state vectors as described above. The total number of pieces in the all the messages is bounded by $\| \sum_{t=0}^{d} v^{(t)} \|_1$ with d being the diameter of the factor graph. Moreover, it holds that $\| \sum_{t=0}^{d} v^{(t)} \|_1$ is of $\mathcal{O}((4nc)^{2d+2})$.*

This gives the worst-case total number of message pieces in MP-WMI. From Proposition 27, it holds that the problems in class treeWMI($\mathbf{\Omega}$) with the weight function family $\mathbf{\Omega}$ satisfying TWC are tractable to MP-WMI, since the complexity of MP-WMI is the total number of message pieces multiplied by the symbolic integration cost of each piece, which is tractable for functions in family $\mathbf{\Omega}$ by definition. This finishes the constructive proof for Theorem 7 in Section 6.2.1. Notice the complexity of WMI problems depends on the graph structures. In our experiments, we will compare solvers on WMI problems with three representative problem classes with different factor graph diameters.

6.2.3 Empirical Evaluation

In this Section, we aim to answer the following research questions:[1] **Q1)** Can we effectively scale WMI inference with MP-WMI? **Q2)** How beneficial is inter-query amortization with MP-WMI?

To answer **Q1**, we generated a benchmark of WMI problems with tree-shaped primal graphs of different diameters: star-shaped graphs (STAR), complete ternary trees (SNOW) and linear

[1] Our implementation of MP-WMI and the code for reproducing our empirical evaluation can be found at `https://github.com/UCLA-StarAI/mpwmi`.

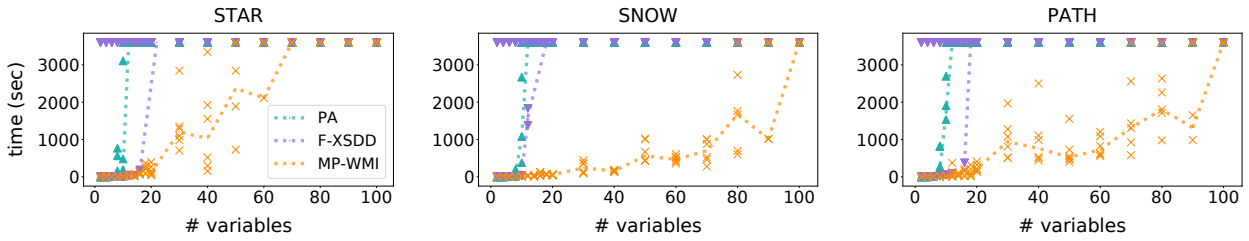


Figure 6.10: Results of the comparison between MP-WMI, WMI-PA and F-XSDD on WMI problems with tree dependencies. In this setting, MP-WMI remarkably scales to problems having up to 60 variables on STAR, while solving SNOW and PATH problems having up to 90 variables, considerably "raising the bar" for the size of tractable WMI inference problems.

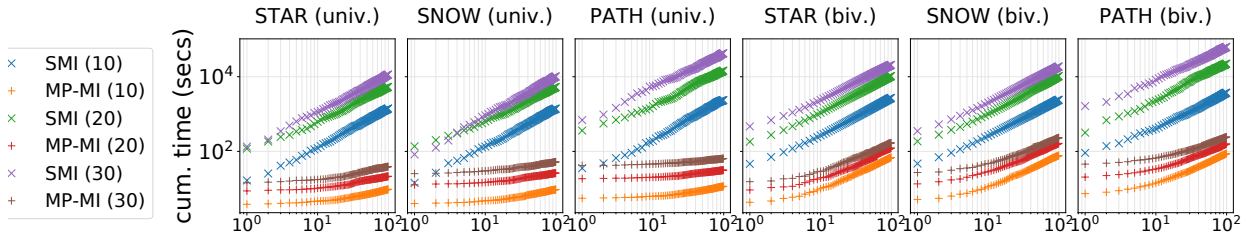


Figure 6.11: Log-log plot of cumulative time (seconds, y-axis) for MP-WMI (orange, red, brown) and SMI (blue, green, purple) over STAR, SNOW and PATH primal graphs (see text) with 10, 20 and 30 variables for increasing numbers of univariate and bivariate queries (x-axis). For every class, MP-WMI takes up to two order of magnitude less time when amortizing 100 queries, while being faster than SMI on a single query.

chains (PATH). These structures were originally investigated by the authors of SMI and are prototypical of tree shapes that can be encountered in real-world scenarios such as phylogenetic trees [NK00], hierarchies in file and networks systems [VGR81], and natural language grammars [PBT06].

We sampled random SMT formulas with N variables with the tree structures described above and polynomial weights mapping a subset of literals to a random non-negative polynomials. We generated problems with N ranging from 2 to 20 with step size 2, and from 20 to 100 with

step size 10. We compared our MP-WMI python implementation against the following baselines: WMI-PA [MPS19], a solid general-purpose WMI solver exploiting SMT-based predicate abstraction techniques that is less sensitive to the problem structure; and F-XSDD(BR) [ZKD19], a compilation-based solver achieving state-of-the-art results in several WMI benchmarks.

Fig. 6.10 shows that, with timeout being an hour, our proposed solver MP-WMI is able to scale up to 60 variables for STAR problems and up to 90 variables for SNOW and PATH problems, while the other two solvers stop at problem size 20 for all three classes. Note that the results are in line with those reported in [ZKD19]. This answers **Q1** affirmatively, raising the bar of the size of WMI problems that can be solved exactly up to 100 variables.

We tackle **Q2** by comparing MP-WMI with SMI [ZB19] on tree-structured MI problems. SMI is a search-based MI solver that has been shown to be efficient for such problems. WMI-PA, F-XSDD and the SGDPLL(T) [SOG16] solver are not included in the comparison since they were already shown in [ZB19] to not be competitive on such problems. The synthetic SMT formulas range over $n \in \{10, 20, 30\}$ variables with tree factor graphs being STAR, SNOW and PATH. We generate 100 univariate or bivariate random queries for each MI problem.

Figure 6.11 shows the cumulative runtime of answering random queries by both solvers. As expected, MP-WMI takes a fraction of the time of SMI (up to two order of magnitudes) to answer 100 univariate or bivariate queries in all experimental scenarios, since it is able to amortize inference *inter-query*. More surprisingly, by looking at the first point of each curve, we can tell that MP-WMI is even faster than SMI to compute a single query. This is because SMI solves polynomial integration numerically, by reconstructing the univariate polynomials before the numeric integration via interpolation, e.g., Lagrange interpolation; while in MP-WMI we adopt symbolic integration. Hence the complexity of the former is always quadratic in the degree of the polynomial, while for the latter the average case is linear in the number of monomials in the polynomial to integrate, which in practice might be much less than the degree of the polynomial.

6.3 Discussion

In this chapter, we first introduce a search-based WMI algorithm that exploits structural independence properties to improve efficiency. For WMI on SMT($\mathcal{LRA}$) theories with tree primal graphs and piecewise polynomial weight functions, it decomposes WMI problems during search. A complexity analysis showed that for balanced tree primal graphs, it yields quasi-polynomial complexity. By further theoretically tracing the boundaries of tractable WMI inferece, we introduce another exact WMI solver based on message-passing, MP-WMI, which is efficient on a rich class of tractable WMI problems with tree-shaped factor graphs, the largest known so far. Furthermore, MP-WMI dramatically reduces the answering time of multiple queries by amortizing local computations and allows to compute all marginals and moments simultaneously.

CHAPTER 7

Approximate Inference Over Constraints

Beyond our effort on exact WMI solvers, we advance the WMI framework on two fronts. First, we deepen the theoretical understanding of the complexity of WMI inference on real-world problems by proving hardness results. Second, we deliver an efficient and accurate approximate WMI solver as a practical algorithmic solution to deploy WMI inference at a larger scale.

7.1 On the hardness of WMI

While the general formulation of WMI we have provided in the previous section is elegant and appealing for advanced probabilistic reasoning, it is, however, not practical in general. In fact, it requires solving an arbitrarily complex integral, which is a #P-hard problem [BBD11].

To fill this gap, recent works have started looking for *classes of tractable WMI problems*, i.e., problems for which a solution can be computed exactly in polytime [ZB19, ZMY20a]. These classes of problems can be characterized by two parameters: the *treewidth* and the *diameter* of the primal graph of the SMT formulas considered, where the latter is generally expressed as a function of the number of variables in the problem. Note that this is strikingly different from classical discrete probabilistic graphical models, where most of the complexity results are stated in terms of the treewidth alone [KF09, Rot96b].

Definition 12. *($\mathcal{WMI}(\boldsymbol{\Omega}, \delta, t)$ Problem Class) Let $\mathcal{WMI}(\boldsymbol{\Omega}, \delta, t)$ be the class of WMI problems over models of the form $(\Delta, \mathcal{W})$ on real domains, having primal graph $\mathcal{G}_\Delta$ with diameter of $\mathcal{O}(\delta(n))$ and treewidth t, where n is the number of variables in the formula Δ; and having per-literal weights $\mathcal{W}$ in a function family $\boldsymbol{\Omega}$.*

The largest tractable WMI class known so far has been introduced in our previous work [ZMY20a] as $\mathcal{WMI}(\mathbf{\Omega}, \log(n), 1)$, i.e., the class of problems over n real variables whose primal graph is tree-shaped (treewidth 1) and has diameter of length logarithmic in n, and whose weight functions belong to a function family $\mathbf{\Omega}$ satisfying some conditions called *tractable weight conditions* (TWCs).

Definition 13. *(TWCs) Given a parametric weight function family $\mathbf{\Omega}$, it satisfies the TWCs iff*

i) it is closed under product, i.e., $\forall f, g \in \mathbf{\Omega}$, $f \cdot g \in \mathbf{\Omega}$;

ii) it is closed under definite integration, i.e., $\forall f \in \mathbf{\Omega}, F(u(x)) - F(l(x)) \in \mathbf{\Omega}$ where F is the antiderivative of f, and $l(x), u(x)$ are SMT($\mathcal{LRA}$) integration bounds for any $x \in \mathcal{X}$;

iii) the symbolic antiderivative of any $f \in \mathbf{\Omega}$ can be tractably computed by symbolic integration.

Examples of weight functions in family $\mathbf{\Omega}$ include the largely adopted family of (piecewise) polynomials [BPB15], the family of exponentiated linear functions and the family of their products. In the following analysis, we will restrict our attention to weight function families satisfying the TWCs.

In [ZMY20a] the tractability of problem class $\mathcal{WMI}(\mathbf{\Omega}, \log(n), 1)$ is demonstrated by construction, where they introduce a message passing scheme, named MP-WMI, that runs in polytime on tree-shaped and diameter-bounded primal graphs. That is, some *sufficient* conditions for tractable WMI classes are provided. Here we provide a finer charting of the "tractable islands" of WMI problems by questioning the necessity of the above conditions while looking for larger tractable classes. We prove that unless $P = NP$, larger classes are not tractable. We begin by proving that increasing the diameter of a tree-shaped problem structure makes it hard.

Theorem 8. *Let $\mathcal{WMI}(\mathbf{\Omega}, n, 1)$ be the class of WMI problems whose weight function family Ω satisfies the TWCs. Then inference in $\mathcal{WMI}(\mathbf{\Omega}, n, 1)$ is #P-hard.*

Proof. We prove our complexity result by reducing a #P-complete variant of the subset sum problem [GJ02] to an MI problem over an SMT($\mathcal{LRA}$) formula Δ with tree primal graph whose diameter is n. This problem is a counting version of subset sum problem saying that given a set of

positive integers $S = \{s_1, s_2, \cdots, s_n\}$, and a positive integer L, the goal is to count the number of subsets $S' \subseteq S$ such that the sum of all the integers in the subset S' equals to L. Notice that our proof can be applied to rational numbers as well and we assume binary representations for numbers.

First, we reduce the counting subset sum problem in polynomial time to a model integration problem by constructing the following SMT($\mathcal{LRA}$) formula Δ on real variables $\mathcal{X}$ whose primal graph is shown in Figure 7.1:

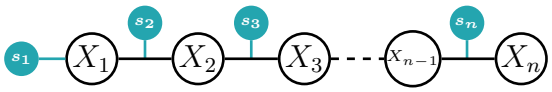


Figure 7.1: Primal graph $\mathcal{G}_\Delta$ used for the #P-hardness reduction in Theorem 8. We construct the corresponding formula Δ such that $\mathcal{G}_\Delta$ has maximum diameter (it is a chain). We graphically augment graph $\mathcal{G}_\Delta$ by introducing blue nodes to indicate that integers s_i in set S are contained in clauses between two variables.

$$\Delta = \begin{cases} \underbrace{s_1 - \frac{1}{2n} < x_1 < s_1 + \frac{1}{2n}}_{\ell(1,0)} \vee \underbrace{-\frac{1}{2n} < x_1 < \frac{1}{2n}}_{\ell(1,1)} \\ \underbrace{x_{i-1} + s_i - \frac{1}{2n} < x_i < x_{i-1} + s_i + \frac{1}{2n}}_{\ell(i,0)} \vee \underbrace{x_{i-1} - \frac{1}{2n} < x_i < x_{i-1} + \frac{1}{2n}}_{\ell(i,1)}, \quad i = 2, \cdots n \end{cases}$$

For brevity, we denote the first and the second literal in the i-th clause by $\ell(i,0)$ and $\ell(i,1)$ respectively as shown above. Also We choose two constants $l = L - \frac{1}{2}$ and $u = L + \frac{1}{2}$.

In the following, we prove that $n^n \mathsf{MI}(\Delta \wedge (l < X_n < u))$ equals to the number of subset $S' \subseteq S$ whose element sum equals to L, which indicates that WMI problem whose tree primal graph has diameter $\mathcal{O}(n)$ is #P-hard.

Let $\boldsymbol{a}^k = (a_1, a_2, \cdots, a_k)$ be some assignment to Boolean variables $(A_1, A_2, \cdots, A_k)$ with $a_i \in \{0, 1\}$, $i \in [k]$. Given an assignment $\boldsymbol{a}^k$, we define subset sums to be $S(\boldsymbol{a}^k) \triangleq \sum_{i=1}^k a_i s_i$, and formulas $\Delta_{\boldsymbol{a}^k} \triangleq \bigwedge_{i=1}^k \ell(i, a_i)$.

Claim 9. *The model integration for formula $\Delta_{\boldsymbol{a}^k}$ with an given assignment $\boldsymbol{a}^k \in \{0,1\}^k$ is $\mathsf{MI}(\Delta_{\boldsymbol{a}^k}) = (\frac{1}{n})^k$. Moreover, for each variable X_i in $\Delta_{\boldsymbol{a}^k}$, its satisfying assignments consist of the interval $[\sum_{j=1}^{i} a_j s_j - \frac{i}{2n}, \sum_{j=1}^{i} a_j s_j + \frac{i}{2n}]$. Specifically, the satisfying assignments for variable X_n in formula $\Delta_{\boldsymbol{a}^n}$ can be denoted by the interval $[S(\boldsymbol{a}^n) - \frac{1}{2}, S(\boldsymbol{a}^n) + \frac{1}{2}]$.*

Proof. (Claim 9) First we prove that $\mathsf{MI}(\Delta_{\boldsymbol{a}^k}) = (\frac{1}{n})^k$. For brevity, denote $a_i s_i$ by $\hat{s_i}$. By definition of model integration and the fact that the integral is absolutely convergent (since we are integrating a constant function, i.e., one, over finite volume regions), we have the following equation.

$$\mathsf{MI}(\Delta_{\boldsymbol{a}^k}) = \int_{(x_1,\cdots,x_k)\models\Delta_{\boldsymbol{a}^k}} 1\, dx_1 \cdots dx_k = \int_{\hat{s}_1-\frac{1}{2n}}^{\hat{s}_1+\frac{1}{2n}} dx_1 \cdots \int_{x_{k-2}+\hat{s}_{k-1}-\frac{1}{2n}}^{x_{k-2}+\hat{s}_{k-1}+\frac{1}{2n}} dx_{k-1} \int_{x_{k-1}+\hat{s}_k-\frac{1}{2n}}^{x_{k-1}+\hat{s}_k+\frac{1}{2n}} 1\, dx_k$$

Observe that for the most inner integration over variable x_k, the integration result is $\frac{1}{n}$. By doing this iteratively, we have that $\mathsf{MI}(\Delta_{\boldsymbol{a}^k}) = (\frac{1}{n})^k$.

Next we prove that satisfying assignments for variable X_i in formula $\Delta_{\boldsymbol{a}^k}$ is the interval $[\sum_{j=1}^{i} a_j s_j - \frac{i}{2n}, \sum_{j=1}^{i} a_j s_j + \frac{i}{2n}]$ by mathematical induction. For $i = 1$, since X_1 is in interval $[a_1 s_1 - \frac{1}{2n}, a_1 s_1 + \frac{1}{2n}]$, the statement holds in this case. Suppose that the statement holds for $i = m$, i.e. variable X_m has its satisfying assignments in interval $[\sum_{j=1}^{m} a_j s_j - \frac{m}{2n}, \sum_{j=1}^{m} a_j s_j + \frac{m}{2n}]$. Since variable X_{m+1} has its satisfying assignments in interval $[X_m + a_{m+1} s_{m+1} - \frac{1}{2n}, X_m + a_{m+1} s_{m+1} + \frac{1}{2n}]$, then its satisfying assignments consist interval $[\sum_{j=1}^{m+1} a_j s_j - \frac{m+1}{2n}, \sum_{j=1}^{m+1} a_j s_j + \frac{m+1}{2n}]$, that is, the statement also holds for $i = m + 1$. Thus the claim holds. □

The above claim shows how to compute the model integration of formula $\Delta_{\boldsymbol{a}^k}$. We will show in the next claim how to compute the model integration of formula $\Delta_{\boldsymbol{a}^n}$ conjoined with a query $l < X_n < u$.

Claim 10. *For each assignment $\boldsymbol{a}^n \in \{0,1\}^n$, the model integration of formula $\Delta_{\boldsymbol{a}^n} \wedge (l < X_n < u)$ falls into one of the following cases:*

i) If $S(\boldsymbol{a}^n) < L$ or $S(\boldsymbol{a}^n) > L$, it holds that $\mathsf{MI}(\Delta_{\boldsymbol{a}^n} \wedge (l < X_n < u)) = 0$.

ii) If $S(\boldsymbol{a}^n) = L$, it holds that $\mathsf{MI}(\Delta_{\boldsymbol{a}^n} \wedge (l < X_n < u)) = (\frac{1}{n})^n$.

Proof. (Claim 10) From the previous Claim 9, it is shown that variable X_n has its satisfying assignments in interval $[S(\boldsymbol{a}^n)-\frac{1}{2}, S(\boldsymbol{a}^n)+\frac{1}{2}]$ in formula $\Delta_{\boldsymbol{a}^n}$ for each $\boldsymbol{a}^n \in \{0,1\}^n$. If $S(\boldsymbol{a}^n) < L$, given that $S(\boldsymbol{a}^n)$ is a sum of positive integers, then it holds that $S(\boldsymbol{a}^n) + \frac{1}{2} \leq (L-1) + \frac{1}{2} = L - \frac{1}{2} = l$ and therefore, $\mathsf{MI}(\Delta_{\boldsymbol{a}^n} \wedge (l < X_n < u)) = 0$; similarly, if $S(\boldsymbol{a}^n) > L$, then it holds that $S(\boldsymbol{a}^n) - \frac{1}{2} \geq u$ and therefore, $\mathsf{MI}(\Delta_{\boldsymbol{a}^n} \wedge (l < X_n < u)) = 0$. If $S(\boldsymbol{a}^n) = L$, by Claim 9 we have that the satisfying assignment interval is inside the interval $[l, u]$ and thus it holds that $\mathsf{MI}(\Delta_{\boldsymbol{a}^n} \wedge (l < X_n < u)) = \mathsf{MI}(\Delta_{\boldsymbol{a}^n}) = (\frac{1}{n})^n$. □

In the next claim, we show how to compute the model integration of formula Δ as well as for formula Δ conjoined with query $l < X_n < u$ based on the already proven Claim 9 and Claim 10.

Claim 11. *The following two equations hold:*

i) $\mathsf{MI}(\Delta) = \sum_{\boldsymbol{a}^n} \mathsf{MI}(\Delta_{\boldsymbol{a}^n})$.

ii) $\mathsf{MI}(\Delta \wedge (l < X_n < u)) = \sum_{\boldsymbol{a}^n} \mathsf{MI}(\Delta_{\boldsymbol{a}^n} \wedge (l < X_n < u))$.

Proof. (Claim 11) Observe that for each clause in Δ, literals are mutually exclusive since each s_i is a positive integer. Then we have that formulas $\Delta_{\boldsymbol{a}^n}$ are mutually exclusive and meanwhile $\Delta = \bigvee_{\boldsymbol{a}^n} \Delta_{\boldsymbol{a}^n}$. Thus it holds that $\mathsf{MI}(\Delta) = \sum_{\boldsymbol{a}^n} \mathsf{MI}(\Delta_{\boldsymbol{a}^n})$. Similarly, we have formulas $(\Delta_{\boldsymbol{a}^n} \wedge (l < X_n < u))$'s are mutually exclusive and meanwhile $\Delta \wedge (l < X_n < u) = \bigvee_{\boldsymbol{a}^n} \Delta_{\boldsymbol{a}^n} \wedge (l < X_n < u)$. Thus the second equation holds. □

From the above claims, we can conclude that $\mathsf{MI}(\Delta \wedge (l < X_n < u)) = t(\frac{1}{n})^n$ where t is the number of assignments $\boldsymbol{a}^n$ s.t. $S(\boldsymbol{a}^n) = L$. Notice that for each $\boldsymbol{a}^n \in \{0,1\}^n$, there is a one-to-one correspondance to a subset $S' \subseteq S$ by defining $\boldsymbol{a}^n$ as $a_i = 1$ if and only if $s_i \in S'$; and $S(\boldsymbol{a}^n)$ equals to L if and only if the sum of elements in $S'$is L. Therefore $n^n \mathsf{MI}(\Delta \wedge (l < X_n < u))$ equals to the number of subset $S' \subseteq S$ whose element sum equals to L. This finishes the proof for the statement that inference in $\mathcal{WMI}(\mathbf{\Omega}, n, 1)$ is #P-hard. □

Next, we turn our attention to another class of WMI problems, the class $\mathcal{WMI}(\mathbf{\Omega}, \log(n), 2)$, having logarithmic diameter but treewidth 2. This class is also supposed to be "easy" in the sense

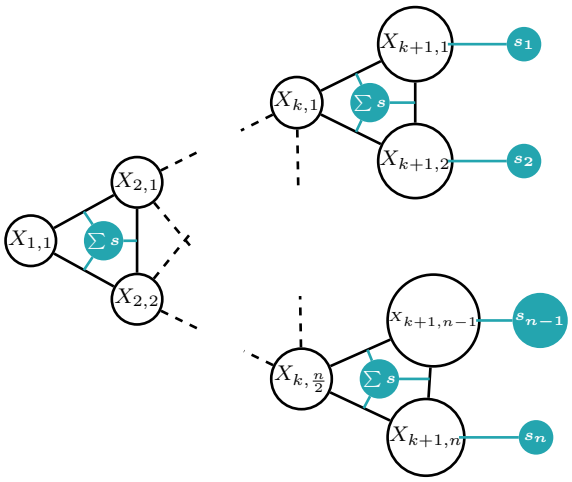


Figure 7.2: Primal graph used for #P-hardness reduction in Theorem 7. We also put blue nodes to indicate that integer s_i's in set S are contained in some clauses and that model integration over some cliques is the sum of some s_i's.

that it extends the tractable class $\mathcal{WMI}(\mathbf{\Omega}, \log(n), 1)$ by slightly increasing the treewidth by one. Unfortunately, inference in $\mathcal{WMI}(\mathbf{\Omega}, \log(n), 2)$ is also hard.

Theorem 12. *Let $\mathcal{WMI}(\mathbf{\Omega}, \log(n), 2)$ be the class of WMI problems whose parametric weight function family Ω satisfies the TWCs. Then inference in $\mathcal{WMI}(\mathbf{\Omega}, \log(n), 2)$ is #P-hard.*

Proof. Again we prove our complexity result by reducing the #P-complete variant of the subset sum problem [GJ02] to an MI problem over an SMT($\mathcal{LRA}$) formula Δ with primal graph whose diameter is $\mathcal{O}(\log n)$ and treewidth two. In the #P-complete subset sum problem, we are given a set of positive integers $S = \{s_1, s_2, \cdots, s_n\}$, and a positive integer L. Notice that our proof can be applied to rational numbers as well and we assume binary representations for numbers. The goal is to count the number of subsets $S' \subseteq S$ such that the sum of all the integers in S' equals L.

First, we reduce this problem in polynomial time to a model integration problem with the following SMT($\mathcal{LRA}$) formula Δ where variables are real and u and l are two constants. Its primal graph is shown in Figure 7.2. Consider $n = 2^k$, $n, k \in \mathbb{N}$.

$$\Delta = \bigwedge_{i\in[n]} (-\frac{1}{4n} < X_{k+1,i} < \frac{1}{4n} \vee -\frac{1}{4n} + s_i < X_{k+1,i} < \frac{1}{4n} + s_i) \bigwedge \Delta_t$$

$$\text{where } \Delta_t = \bigwedge_{j\in[k],i\in[2^j]} -\frac{1}{4n} + X_{j+1,2i-1} + X_{j+1,2i} < X_{j,i} < \frac{1}{4n} + X_{j+1,2i-1} + X_{j+1,2i}$$

For brevity, we denote all the variables by $\mathcal{X}$ and denote the literal $-\frac{1}{4n} < X_{k+1,i} < \frac{1}{4n}$ by $\ell(i,0)$ and literal $-\frac{1}{4n} + s_i < X_{k+1,i} < \frac{1}{4n} + s_i$ by $\ell(i,1)$ respectively. Also We choose two constants $l = L - \frac{1}{2}$ and $u = L + \frac{1}{2}$. In the following, we prove that $(2n)^{2n-1}\mathsf{MI}(\Delta \wedge (l < X_{1,1} < u))$ equals to the number of subset $S' \subseteq S$ whose element sum equals to L, which indicates that model integration problem with primal graph whose diameter is $\mathcal{O}(\log n)$ and treewidth two is #P-hard.

Let $\boldsymbol{a}^n = (a_1, a_2, \cdots, a_n) \in \{0,1\}^n$ be some assignment to Boolean variables $(A_1, A_2, \cdots, A_n)$. Given assignment $\boldsymbol{a}^n$, define sum as $S(\boldsymbol{a}^n) \triangleq \sum_{i=1}^n a_i s_i$, and formula as $\Delta_{\boldsymbol{a}^n} \triangleq \bigwedge_{i=1}^n \ell(i, a_i) \wedge \Delta_t$.

Claim 13. *The model integration for formula $\Delta_{\boldsymbol{a}^n}$ with given $\boldsymbol{a}^n \in \{0,1\}^n$ is $\mathsf{MI}(\Delta_{\boldsymbol{a}^n}) = (\frac{1}{2n})^{2n-1}$. Moreover, for each variable $X_{j,i}$ in formula $\Delta_{\boldsymbol{a}^n}$, its satisfying assignments consist of the interval $[\sum_l a_l s_l - \frac{2^{k-j+2}-1}{4n}, \sum_l a_l s_l + \frac{2^{k-j+2}-1}{4n}]$ where $l \in \{l \mid X_{k+1,l}$ is a descendant of $X_{j,i}\}$. Specifically, the satisfying assignments for the root variable $X_{1,1}$ can be denoted the interval $[S(\boldsymbol{a}^n) - \frac{2n-1}{4n}, S(\boldsymbol{a}^n) + \frac{2n-1}{4n}] \subset [S(\boldsymbol{a}^n) - \frac{1}{2}, S(\boldsymbol{a}^n) + \frac{1}{2}]$.*

Proof. (Claim 13) First we prove that $\mathsf{MI}(\Delta_{\boldsymbol{a}^n}) = (\frac{1}{2n})^{2n-1}$. For brevity, denote $a_i s_i$ by $\hat{s_i}$. By definition of model integration and the fact that the integral is absolutely convergent (since we are integrating a constant function, i.e., one, over finite volume regions), we have the following

$$\mathsf{MI}(\Delta_{\boldsymbol{a}^n}) = \int_{\boldsymbol{x}\models\Delta_{\boldsymbol{a}^n}} 1 \, d\mathcal{X}$$
$$= \int_{-\frac{1}{4n}+\hat{s}_n}^{\frac{1}{4n}+\hat{s}_n} dx_{k+1,n} \cdots \int_{-\frac{1}{4n}+\hat{s}_1}^{\frac{1}{4n}+\hat{s}_1} dx_{k+1,1} \int_{-\frac{1}{4n}+x_{k+1,n-1}+x_{k+1,n}}^{\frac{1}{4n}+x_{k+1,n-1}+x_{k+1,n}} dx_{k,2^{k-1}} \cdots \int_{-\frac{1}{4n}+x_{2,1}+x_{2,2}}^{\frac{1}{4n}+x_{2,1}+x_{2,2}} 1 \, dx_{1,1} \, .$$

Observe that for the most inner integration over variable $x_{1,1}$, the integration result is $\frac{1}{2n}$. By doing this iteratively, we have that $\mathsf{MI}(\Delta_{\boldsymbol{a}^k}) = (\frac{1}{2n})^{2n-1}$ where the $2n-1$ comes from the

number of variables.

Then we prove that satisfying assignments for variable $X_{j,i}$ in formula $\Delta_{\boldsymbol{a}^n}$ lie in the interval $[\sum_l a_l s_l - \frac{2^{k-j+2}-1}{4n}, \sum_l a_l s_l + \frac{2^{k-j+2}-1}{4n}]$ where $l \in \{l \mid X_{k+1,l} \text{ is a descendant of } X_{j,i}\}$ by performing mathematical induction in a bottom-up way.

For $j = 1$, any variable $X_{k+2-j,i}$ with $i \in [2^{k+2-j}]$ has satisfying assignments consisting of the interval $[a_i s_i - \frac{1}{4n}, a_i s_i + \frac{1}{4n}]$. Thus the statement holds for this case.

Suppose that the statement holds for $j = m$, that is, for any $i \in [2^{k+2-m}]$, any variable $X_{k+2-m,i}$ has satisfying assignments consisting interval $[\sum_l a_l s_l - \frac{2^m-1}{4n}, \sum_l a_l s_l + \frac{2^m-1}{4n}]$ where $l \in \{l \mid X_{k+1,l} \text{ is a descendant of } X_{k+2-m,i}\}$.

Then for $j = m+1$ and any $i \in [2^{k+1-m}]$, the variable $X_{k+1-m,i}$ has two descendants, variable $X_{k+2-m,2i-1}$ and variable $X_{k+2-m,2i}$. Moreover, we have that $-\frac{1}{4n} + X_{k+2-m,2i-1} + X_{k+2-m,2i} < X_{k+1-m,i} < \frac{1}{4n} + X_{k+2-m,2i-1} + X_{k+2-m,2i}$. Then the lower bound of the interval for variable $X_{k+1-m,i}$ is $-\frac{1}{4n} + \sum_l a_l s_l - 2\frac{2^m-1}{4n} = \sum_l a_l s_l - \frac{2^{m+1}-1}{4n}$; similarly the upper bound of the interval is $\sum_l a_l s_l + \frac{2^{m+1}-1}{4n}$, where $l \in \{l \mid X_{k+1,l} \text{ is a descendant of } X_{k+1-m,i}\}$. That is, the statement also holds for $j = m + 1$ which finishes our proof. □

The above claim shows what the model integration of formula $\Delta_{\boldsymbol{a}^k}$ is like. We'll show in the next claim what the model integration of formula $\Delta_{\boldsymbol{a}^n}$ conjoined with a query $l < X_{1,1} < u$ is like.

Claim 14. *For each assignments $\boldsymbol{a}^n \in \{0,1\}^n$, the model integration of $\Delta_{\boldsymbol{a}^n} \wedge (l < X_{1,1} < u)$ falls into one of the following cases:*

i) If $S(\boldsymbol{a}^n) < L$ or $S(\boldsymbol{a}^n) > L$, then $\mathsf{MI}(\Delta_{\boldsymbol{a}^n} \wedge (l < X_{1,1} < u)) = 0$.

ii) If $S(\boldsymbol{a}^n) = L$, then $\mathsf{MI}(\Delta_{\boldsymbol{a}^n} \wedge (l < X_{1,1} < u)) = (\frac{1}{2n})^{2n-1}$.

Proof. (Claim 14) From previous Claim 13, it is shown that variable $X_{1,1}$ has its satisfying assignments in the interval $[S(\boldsymbol{a}^n) - \frac{2n-1}{4n}, S(\boldsymbol{a}^n) + \frac{2n-1}{4n}]$ in formula $\Delta_{\boldsymbol{a}^n}$ for each $\boldsymbol{a}^n \in \{0,1\}^n$.

If $S(\boldsymbol{a}^n) < L$, given that $S(\boldsymbol{a}^n)$ is a sum of positive integers, then it holds that $S(\boldsymbol{a}^n) + \frac{1}{2} \leq$

$(L-1)+\frac{2n-1}{4n} < L-\frac{1}{2} = l$ and therefore, $\mathsf{MI}(\Delta_{\boldsymbol{a}^n} \wedge (l < X_{1,1} < u)) = 0$; similarly, if $S(\boldsymbol{a}^n) > L$, then it holds that $S(\boldsymbol{a}^n) - \frac{1}{2} > u$ and therefore, $\mathsf{MI}(\Delta_{\boldsymbol{a}^n} \wedge (l < X_{1,1} < u)) = 0$. If $S(\boldsymbol{a}^n) = L$, then by Claim 13 we have that the satisfying assignment interval is inside the interval $[l, u]$ and thus it holds that $\mathsf{MI}(\Delta_{\boldsymbol{a}^n} \wedge (l < X_{1,1} < u)) = \mathsf{MI}(\Delta_{\boldsymbol{a}^n}) = (\frac{1}{2n})^{2n-1}$. □

Claim 15. *The following two equations hold:*

i) $\mathsf{MI}(\Delta) = \sum_{\boldsymbol{a}^n} \mathsf{MI}(\Delta_{\boldsymbol{a}^n})$.

ii) $\mathsf{MI}(\Delta \wedge (l < X_{1,1} < u)) = \sum_{\boldsymbol{a}^n} MI(\Delta_{\boldsymbol{a}^n} \wedge (l < X_{1,1} < u))$.

Proof. (Claim 15) Observe that for each pair of literals $\ell(i, 0)$ and $\ell(i, 1), i \in [n]$, literals are mutually exclusive since each s_i is a positive integer. Then we have that formulas $\Delta_{\boldsymbol{a}^n}$ are mutually exclusive and meanwhile formula $\Delta = \bigvee_{\boldsymbol{a}^n} \Delta_{\boldsymbol{a}^n}$. Thus it holds that $\mathsf{MI}(\Delta) = \sum_{\boldsymbol{a}^n} \mathsf{MI}(\Delta_{\boldsymbol{a}^n})$. Similarly, we have formulas $(\Delta_{\boldsymbol{a}^n} \wedge (l < X_{1,1} < u))$'s are mutually exclusive and meanwhile $\Delta \wedge (l < X_{1,1} < u) = \bigvee_{\boldsymbol{a}^n} \Delta_{\boldsymbol{a}^n} \wedge (l < X_{1,1} < u)$. Thus the second equation holds. □

From the above claims, we can conclude that $\mathsf{MI}(\Delta \wedge (l < X_{1,1} < u)) = t(\frac{1}{2n})^{2n-1}$ where t is the number of assignments $\boldsymbol{a}^n$ s.t. $S(\boldsymbol{a}^n) = L$. Notice that for each $\boldsymbol{a}^n \in \{0, 1\}^n$, there is a one-to-one correspondence to a subset $S' \subseteq S$ by defining $\boldsymbol{a}^n$ as $a_i = 1$ if and only if $s_i \in S'$; and $S(\boldsymbol{a}^n)$ equals to L if and only if the sum of elements in S' is L. Therefore $(2n)^{2n-1}\mathsf{MI}(\Delta \wedge (l < X_{1,1} < u))$ equals to the number of subset $S' \subseteq S$ whose element sum equals to L. This finishes the proof for the statement that inference in $\mathcal{WMI}(\boldsymbol{\Omega}, \log(n), 2)$ is #P-hard. □

Note that our result differs from the one presented in [ZMY20a] for the hardness of the class $2\mathsf{WMI}(\boldsymbol{\Omega})$, containing WMI problems with SMT formulas being conjunctions of clauses comprising at most two variables. In fact, $\mathcal{WMI}(\boldsymbol{\Omega}, \log(n), 2)$ is contained in $2\mathsf{WMI}(\boldsymbol{\Omega})$. As such, we trace the tractablity boundaries of WMI inference with higher precision, as the next corollary states. Its proof follows from Theorems 8 and 12 and from the sufficiency as demonstrated in [ZMY20a].

Corollary 3. *Let $\mathcal{WMI}(\mathbf{\Omega}, \log(n), t)$ be the class of WMI problems whose parametric weight function family Ω satisfies the TWCs. Then $\mathcal{WMI}(\mathbf{\Omega}, \log(n), t)$ is a tractable WMI class for inference if-and-only-if treewidth $t = 1$.*

These complexity results set the standard for the solver complexity: every exact WMI solver that aims to be efficient, needs to operate in the regime of Corollary 3. However, real-world problems do not always conform to the structural desiderata for primal graphs stated in it. This implies that efficient approximations might not only be useful in these scenarios, but *needed*. In the next section we fill this gap, by introducing our approximate WMI solver that navigates the tractable islands in WMI problems by performing efficient inference on a relaxed version of intractable WMI problems.

7.2 ReCoin: Approximate WMI Solver

Our algorithm to approximate WMI inference comprises three phases: i) *RElaxing* an intractable WMI model into a simpler one amenable to exact inference by removing dependencies from it; then ii) introduce certain literals and weights to *COmpensate* for the dependency structure lost in this way and iii) optimize them by solving a series of exact *INtegration* problems. We name it ReCoIn. With ReCoIn we can navigate *a spectrum of approximations* — with the original primal graph $\mathcal{G}_\Delta$ on one end, and a fully disconnected version on the other — by removing more and more edges. As such, ReCoIn can be viewed as extending the *relax-compensate-recover* (RCR) framework [CD06, CD10, CD12] for approximate inference on discrete probabilistic models to continuous representations and in presence of algebraic constraints.

7.2.1 Relaxation: introducing and then "breaking" equivalence constraints

The aim of the relaxation step is to obtain a new SMT formula Δ^{rel} such that its associated primal graph $\mathcal{G}_{\Delta^{\text{rel}}}$, serves as the simplification of the original $\mathcal{G}_\Delta$ by removing a given set of edges. We will show that the removal of any edge can be formulated as the removal of an equivalence edge [CD12]. This process consists of two steps. First, we create an *augmented formula* Δ^{aug} by

introducing new variables to Δ and enforcing them to act as *copies* of certain original variables by explicitly adding *equivalence constraints*. Second, we deliver the relaxed $\mathcal{G}_{\Delta^{\text{rel}}}$ by removing these equivalence constraints.

7.2.1.1 Augmentation.

The detailed process of distilling a new augmented model $(\Delta^{\text{aug}}, \mathcal{W}^{\text{aug}})$ from $(\Delta, \mathcal{W})$, given a subset of edges $\mathbb{E}_d \subseteq \mathbb{E}$ in $\mathcal{G}_\Delta$ to remove, is listed in Algorithm 9. At its core, there are routines for copying one variable and adding the corresponding equivalence constraints and compensating literals. For each edge $X_i - X_j \in \mathbb{E}_d$ to be removed, one of its variables is arbitrarily selected, say X_i. Then a variable X_i^c, as a copy of the chosen X_i, is introduced in Δ^{aug} as well as one equivalence constraint between the two as the literal $\hat{\ell} : (X_i^c = X_i)$ with associated weight function $\delta(X_i, X_i^c)$ where δ is the Dirac delta function. Then we properly rename all occurrences of X_i by X_i^c in the literals appearing in the clauses of Δ^{aug} that also contain X_j and introduce copied literals for the univariate clauses over X_i only. These steps cause the primal graph $\mathcal{G}_{\Delta^{\text{aug}}}$ to now contain the dependency $X_i - X_i^c - X_j$ but not $X_i - X_j$.

Note that the augmented WMI model $(\Delta^{\text{aug}}, \mathcal{W}^{\text{aug}})$ now contains more variables than the original one. Specifically, for each variable $X_i \in \mathcal{G}_\Delta$ we might have introduced C_i different copies in $\mathcal{G}_{\Delta^{\text{aug}}}$, denoted as $X_i^1, \ldots, X_i^{C_i}$, if we removed C_i edges over X_i. We will denote the original X_i as X_i^0 for notation consistency. Even if the dimensionality of the augmented WMI problem is increased by augmentation, the next propositions are guaranteeing that we are not altering the partition function and the marginal distributions of Pr_Δ, and that introducing equivalence constraints does not alter the induced distribution.

Proposition 28. *Let Δ be an SMT formula with primal graph $\mathcal{G}_\Delta$ and per-literal weight functions $\mathcal{W}$, and let Δ^{aug} and $\mathcal{W}^{\text{aug}}$ be the output of Algorithm 9 when applied to Δ and $\mathcal{G}_\Delta$ given a certain subset of edges in $\mathcal{G}_\Delta$. Then it holds that $\mathsf{WMI}(\Delta, \mathcal{W}) = \mathsf{WMI}(\Delta^{\text{aug}}, \mathcal{W}^{\text{aug}})$. Moreover, for any X_i in $\mathcal{G}_\Delta$ and univariate literal ℓ over X_i, it holds that $\mathsf{Pr}_\Delta(\ell) = \mathsf{Pr}_{\Delta^{\text{aug}}}(\ell)$.*

Algorithm 9 augmentModel($\Delta, \mathcal{W}, \mathbb{E}_d$)

Input: a WMI model with SMT formula Δ and per-literal weights $\mathcal{W}$ and a set $\mathbb{E}_d$ of edges to be deleted

Output: augmented WMI model $(\Delta^{\text{aug}}, \mathcal{W}^{\text{aug}})$ and equivalence constraint set $\mathcal{L}$

$\Delta^{\text{aug}} \leftarrow \text{copy}(\Delta)$
$\mathcal{W}^{\text{aug}} \leftarrow \text{copy}(\mathcal{W})$
$\mathcal{L} \leftarrow \{\}$
for edge $X_i - X_j \in \mathbb{E}_d$ **do**
    $X_i^c \leftarrow \text{copy}(X_i)$ *// Assume to copy X_i*
    $\hat{\ell} \leftarrow (X_i = X_i^c)$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{\hat{\ell}\}$
    $\Delta' \leftarrow \Delta^{\text{aug}} \wedge \hat{\ell}$,
    $w_{\hat{\ell}} := \delta(X_i, X_i^c)$
    $\mathcal{W}^{\text{aug}} \leftarrow \mathcal{W}^{\text{aug}} \cup \{w_{\hat{\ell}}\}$
    for clause $\Gamma \in \Delta_{i,j}$ **do** *// Rename edges*
        $\Gamma' \leftarrow \Gamma\{X_i : X_i^c\}$
        $\Delta' \leftarrow \Delta'\{\Gamma : \Gamma'\}$
        for each literal $\ell \in \Gamma$ **do**
            $\ell' \leftarrow \ell\{X_i : X_i^c\}$
            $w_{\ell'} \leftarrow \text{copy}(w_\ell)$
            $\mathcal{W}^{\text{aug}} \leftarrow \mathcal{W}^{\text{aug}} \cup \{w_{\ell'}\} \setminus \{w_\ell\}$
    for clause $\Gamma \in \Delta_i$ **do** *// Copy and rename bounding-box literals*
        $\Gamma' \leftarrow \text{copy}(\Gamma)$
        $\Delta' \leftarrow \Delta' \wedge \Gamma'\{X_i : X_i^c\}$
    $\Delta^{\text{aug}} \leftarrow \Delta'$
return $\Delta^{\text{aug}}, \mathcal{W}^{\text{aug}}, \mathcal{L}$

Algorithm 10 relaxModel($\Delta^{\text{aug}}, \mathcal{W}^{\text{aug}}, \mathcal{L}$)

Input: an augmented WMI model $(\Delta^{\text{aug}}, \mathcal{W}^{\text{aug}})$, $\mathcal{L}$: equivalence constraints to be relaxed

Output: a relaxed WMI model $(\Delta^{\text{rel}}, \mathcal{W}^{\text{rel}})$, and its "remaining-part" model $(\Delta^{\text{rem}}, \mathcal{W}^{\text{rem}})$.

$\Delta^{\text{rem}} \leftarrow \top$
$\mathcal{W}^{\text{rem}} \leftarrow \{\}$
$\Delta^{\text{rel}} \leftarrow \mathsf{copy}(\Delta^{\text{aug}})$
$\mathcal{W}^{\text{rel}} \leftarrow \mathsf{copy}(\mathcal{W}^{\text{aug}})$
for each $\ell^* : (X_i = X_i^c) \in \mathcal{L}$ **do**
  for clause $\Gamma \in \Delta_i$ **do**
    $\Delta^{\text{rem}} \leftarrow \Delta^{\text{rem}} \wedge \Gamma \wedge \Gamma\{X_i : X_i^c\}$
    for each literal $\ell \in \Gamma$ **do**
      $\ell' \leftarrow \ell\{X_i : X_i^c\}$
      $w_{\ell'} \leftarrow \mathsf{copy}(w_\ell)$
      $\mathcal{W}^{\text{rel}} \leftarrow \mathcal{W}^{\text{rel}} \cup \{w_{\ell'}\}$
      $\mathcal{W}^{\text{rem}} \leftarrow \mathcal{W}^{\text{rem}} \cup \{w_\ell, w_{\ell'}\}$
  $\Delta^{\text{rel}} \leftarrow \Delta^{\text{rel}}\{\ell^* : \top\}$ *// disconnect X_i and copy X_i^c*
  $\mathcal{W}^{\text{rel}} \leftarrow \mathcal{W}^{\text{rel}} \setminus \{w_{\ell^*}\}$
  $\Delta^{\text{rem}} \leftarrow \Delta^{\text{rem}} \wedge \ell^*$
  $\mathcal{W}^{\text{rem}} \leftarrow \mathcal{W}^{\text{rem}} \cup \{w_{\ell^*}\}$
return $(\Delta^{\text{rel}}, \mathcal{W}^{\text{rel}})$, $(\Delta^{\text{rem}}, \mathcal{W}^{\text{rem}})$

7.2.1.2 Removing equivalence constraints.

Given an augmented model $(\Delta^{\text{aug}}, \mathcal{W}^{\text{aug}})$, we remove equivalence constraints introduced at the augmentation step to obtain the relaxed model $(\Delta^{\text{rel}}, \mathcal{W}^{\text{rel}})$. As a result, each original variable in $\mathcal{G}_{\Delta^{\text{rel}}}$ will be detached from its copies, thus ignoring the dependencies encoded by the edges $\mathbb{E}_d$ that were marked to be removed. Algorithm 10 details this procedure. Note that relaxation "breaks" the augmented formula Δ^{aug} into a relaxed part Δ^{rel} and a "remaining part" Δ^{rem}, which

contains the equivalence constraints just removed.

7.2.1.3 Which edges to relax?

After relaxing enough constraints, we can obtain a WMI problem amenable to exact inference, for example, one whose primal graph $\mathcal{G}_{\Delta^{\text{rel}}}$ has treewidth one and logarithmic diameter. Running an exact WMI solver on such a problem would already deliver a cheap way to perform approximate inference. However, the quality of such an approximation can be greatly improved if we compensate for the relaxed constraints. We will discuss this in the next section.

A question remains: *how to select the set of edges $\mathbb{E}_d$ to relax?* Note that the more edges we remove from Δ, the easier it is to perform inference on Δ^{rel} given fewer dependencies, but the lower the approximation quality, and the harder to compensate for them all, since it would differ from the augmented model more, and meanwhile from the original model as Proposition 28 indicates. For example, removing all edges in $\mathcal{G}_{\Delta}$ will yield a fully disconnected $\mathcal{G}_{\Delta^{\text{rel}}}$ where performing exact inference on each component is going to be embarrassingly parallelizable. This would correpond to perform a loopy version of the MP-WMI algorithm. Analogous to its discrete counterpart, loopy belief propagation, it would be susceptible to poor converge rates [KF09, CD06]. Therefore we propose a simple strategy for selecting the edges to be removed, which is to retrieve a spanning tree of the original primal graph. In Section 7.3 we demonstrate its practical effectiveness on a range of inference problems of increasing complexity. Devising and evaluating alternative relaxing strategies is an interesting topic for future work.

7.2.1.4 Compensation

The aim of the compensation phase is to recover the relaxed equivalence constraints and hence, make the distribution $\Pr_{\Delta^{\text{rel}}}$ better approximate $\Pr_{\Delta^{\text{aug}}}$ and thus better approximate $\Pr_{\Delta}$ as Proposition 28 suggests. In order to do so, we introduce new literals, named *compensating literals*, to the variables and their copies in the relaxed formula Δ^{rel} and equip them with parameterized weights, named *compensating weights*, and further we optimize them in order to synchronize the variable marginals among a copied variable and its copies.

For each variable $X_i = X_i^0$ and its C_i copies $X_i^1, \ldots, X_i^{C_i}$ in formula Δ^{rel}, we generate K different univariate literals of the form $\ell_{i,k}^c : (X_i^{(c)} \leq \sigma_{i,k} \cdot \tau_{i,k})$ for $k = 1, \ldots, K$ and $c = 0, 1, \ldots, C_i$ where each $\sigma_{i,k}$ and $\tau_{i,k}$ are respectively drawn at uniform from $\{+1, -1\}$ and the support of X_i as encoded in Δ_i^{rel}. Note that the $\sigma_{i,k}, \tau_{i,k}$ are *shared* across all the copies. Each compensating literal $\ell_{i,k}^c$ is therefore responsible for a portion of the support of the marginal distribution of X_i^c, and also for the (unnormalized) marginal density of X_i^c by equipping it with a parameterized weight $w_{\ell_{i,k}^c}$.

To retain tractable inference, the parametric function family chosen for each $w_{\ell_{i,k}^c}$ should satisfy the TWCs as discussed in section 7.1. Striving for simplicity, we employ constant weights of the form $w_{\ell_{i,k}^c} := \exp(\theta_{i,k}^c)$. Therefore, our induced marginal density takes the form of a piecewise constant approximation. As such, by increasing the number of compensating literals K one could obtain a finer approximation, however at the price of introducing more parameters to optimize for. We empirically investigate the effect of increasing K in our experiments in section 7.3.

7.2.1.5 Iterative integration

Instead of matching marginal density functions we settle for the weaker condition of *matching the marginal probabilities of the newly introduced compensating literals*. This in turn can be stated by the following set of equivalence constraints for each variable X_i:

$$\mathsf{Pr}_{\Delta^{\text{rem}}}\Big(\bigwedge\nolimits_{c=0}^{C_i} \ell_{k,i}^c\Big) = \mathsf{Pr}_{\Delta^{\text{rel}}}\big(\ell_{k,i}^0\big) = \mathsf{Pr}_{\Delta^{\text{rel}}}\big(\ell_{k,i}^1\big) = \cdots = \mathsf{Pr}_{\Delta^{\text{rel}}}\big(\ell_{k,i}^{C_i}\big), \;\; \textit{for } k = 1, \cdots, K. \quad (7.1)$$

where the first term $\mathsf{Pr}_{\Delta^{\text{rem}}}\big(\bigwedge_{j=0}^{C_i} \ell_{k,i}^c\big)$ is the probability of the compensating literals in the remaining WMI model $(\Delta^{\text{rem}}, \mathcal{W}^{\text{rem}})$ and $\mathsf{Pr}_{\Delta^{\text{rel}}}\big(\ell_{k,i}^c\big)$ are the probabilities of compensating literals in the relaxed formula Δ^{rel}. Intuitively, for a single equivalence constraint that has been relaxed, there exists a set of parameters θ for the compensating weights that exactly match the probabilities in Equation 7.1 and hence guarantee exact marginal recovery [CD06]. The next theorem better formalizes it.

Algorithm 11 RECOIN $(\Delta, \mathcal{W}, K)$

Input: a WMI model $(\Delta, \mathcal{W})$, K number of compensating literals

Output: $(\Delta^{\text{rel}}, \mathcal{W}^{\text{rel}})$: a relaxed and compensated WMI model

$\mathbb{E}_d \leftarrow \text{initStrategy}(\Delta, \mathcal{W})$ *// Select edges to remove*

$\Delta^{\text{aug}}, \mathcal{W}^{\text{aug}}, \mathcal{L} \leftarrow \text{augmentModel}(\Delta, \mathcal{W}, \mathbb{E}_d)$

$(\Delta^{\text{rel}}, \mathcal{W}^{\text{rel}}), (\Delta^{\text{rem}}, \mathcal{W}^{\text{rem}}) \leftarrow \text{relaxModel}(\Delta^{\text{aug}}, \mathcal{W}^{\text{aug}}, \mathcal{L})$

$\Delta^{\text{rel}}, \mathcal{W}^{\text{rel}} \leftarrow \text{addingCompensations}(\Delta^{\text{rel}}, \mathcal{W}^{\text{rel}}, \mathcal{L}, K)$

while *not* converged **do**

  for $X_i \in \text{copiedNodes}(\Delta^{\text{rel}})$ **do**

    for $k = 1, \ldots, K$ **do**

      $r^k \leftarrow \text{WMI}(\Delta^{\text{rem}}, \mathcal{W}^{\text{rem}}) \;/\; \text{WMI}(\Delta^{\text{rem}} \wedge \bigwedge_{c=0}^{C_i} \ell_{k,i}^c, \mathcal{W}^{\text{rem}}) - 1$

      for $c = 0, 1, \ldots, C_i$ **do**

        $\theta_{k,i}^{c,(t+1)} \leftarrow \log(r^k \alpha_{k,\sigma(c)}) - \log(1 - \alpha_{k,\sigma(c)}) - \sum_{c' \neq c} \theta_{k,i}^{c,(t)}$

Return $(\Delta^{\text{rel}}, \mathcal{W}^{\text{rel}})$

Theorem 16. *Suppose that a relaxed model* $(\Delta^{\mathsf{rel}}, \mathcal{W}^{\mathsf{rel}})$ *and a remaining model* $(\Delta^{\mathsf{rem}}, \mathcal{W}^{\mathsf{rem}})$ *are obtained by relaxing a single equivalence constraint* $(X_i = X_i^c)$ *from an augmented model* Δ^{aug}*, and that the primal graph of* Δ^{rel} *is split into two disconnected components by the relaxation. Let* $(\ell_{i,k}, \ell_{i,k}^c)$ *for* $k = 1, \ldots, K$ *be the* K *pairs of compensating literals introduced, and* $\theta_{k,i}, \theta_{k,i}^c$*, for* $k = 1, \ldots, K$*, be the parameters attached to the compensating weights. Then Equation 7.1 holds when the compensating weight parameters satisfy the following equalities.*

$$\theta_{k,i} = \log \frac{r^k \alpha_{k,c}}{1 - \alpha_{k,c}} - \theta_{k,i}^c, \qquad \theta_{k,i}^c = \log \frac{r^k \alpha_k}{1 - \alpha_k} - \theta_{k,i} \quad \textit{for } k = 1, \ldots, K$$

where

$$r^k = \frac{\mathsf{WMI}(\Delta^{\mathsf{rem}} \bigwedge \neg \ell_{k,i} \bigwedge \neg \ell_{k,i}^c, \mathcal{W}^{\mathsf{rem}})}{\mathsf{WMI}(\Delta^{\mathsf{rem}} \bigwedge \ell_{k,i} \bigwedge \ell_{k,i}^c, \mathcal{W}^{\mathsf{rem}})}, \;\; \alpha_k = \mathsf{Pr}_{\Delta^{\mathsf{rel}}}(\ell_{i,k}), \; \alpha_{k,c} = \mathsf{Pr}_{\Delta^{\mathsf{rel}}}(\ell_{i,k}^c), \;\; \textit{for } k = 1, \ldots, K.$$

Theorem 16 suggests an iterative optimization scheme to find the fixed point solutions for all the compensating parameters introduced to compensate multiple relaxed equivalence constraints. Specifically, starting from a random initialization of the parameters of the compensating weights,[1] at each iteration $t + 1$, we can update each parameter $\theta_{k,i}^{c,(t+1)}$ as

$$\theta_{k,i}^{c,(t+1)} \leftarrow \log(r^k \alpha_{k,\pi(c)}) - \log(1 - \alpha_{k,\pi(c)}) - \sum\nolimits_{c' \neq c} \theta_{k,i}^{c',(t)}, \tag{7.2}$$

where π is a permutation over the copies and each $\alpha_{k,\pi(c)}$ is computed as the probability of $\ell_{k,i}^{\pi(c)}$ according to the relaxed model.

Therefore, at each iteration t, we need to solve $2K$ integration problems for computing the r^k terms and $C_i \cdot K$ integrations for $\mathsf{Pr}_{\Delta^{\mathsf{rel}}}(\ell_{k,i}^{\pi(c)})$ for each pair of variable and its copies. While in principle we could use any exact WMI solver to solve these problems, we adopt MP-WMI [ZMY20a] because it is the fastest solver yet for tree-shaped and bounded diameter problems, and even more importantly, it allows to *amortize inference across queries*. That is, we can compute all the $C_i \cdot K$ literal probabilities in a single message-passing step with it.

[1] Following [CD10], we initialize all parameters to 1.

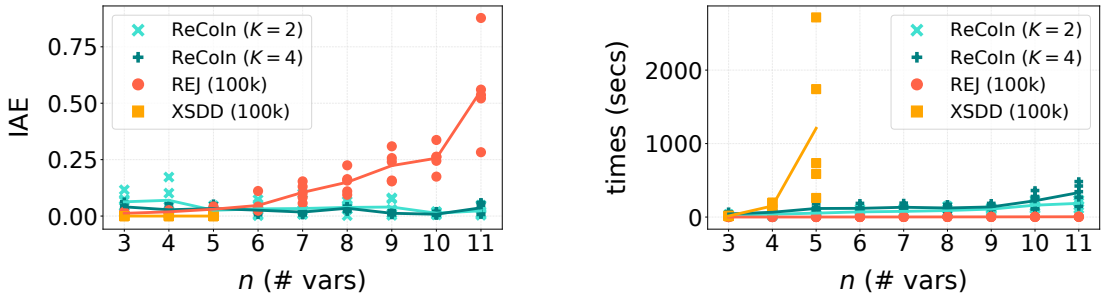


Figure 7.3: Average integrated absolute errors (left) and times in seconds (right) for 5 problems of increasing size (n, x-axis) for ReCoIn and competitors. Number of compensating literals (2-4) or samples used are in parentheses. Mean values per problem size are connected by a line.

From this perspective, our algorithm ReCoIn generates a sequence of induced distributions $\Pr^{(1)}_{\Delta^{\text{rel}}}, \ldots, \Pr^{(2)}_{\Delta^{\text{rel}}}, \Pr^{(t)}_{\Delta^{\text{rel}}}$, that should converge to a fixed-point distribution. In practice to check for convergence, one can monitor the quality of the literal probability approximations and stop when a threshold ϵ is met before a certain number of iterations are done. We choose the threshold to be the maximum L-∞ norm of compensation literal probability differences. To ease convergence, we apply *dampening*, that is, we smooth each parameter update at iteration $t+1$ by a factor $\lambda > 0$: $\theta^{c,(t+1)}_{k,i} \leftarrow (1-\lambda) \cdot \theta^{c,(t+1)}_{k,i} + \lambda \cdot \theta^{c,(t+1)}_{k,i}$. This completes the steps in our ReCoIn solver. Algorithm 11 recaps them.

7.3 Empirical Evaluation

We aim to answer the following questions: **(Q1)** how fast and scalable is ReCoIn?, **(Q2)** how accurate are its approximations?, **(Q3)** what is the effect of increasing the number of compensating literals K?

We generate WMI problems whose primal graphs are random Watts-Strogatz graphs [WS98] with increasing size $n = 1, \ldots, 11$, with two additional neighbor connections and probability of rewiring 0.5, to which we attach randomly generated clauses of length 2 and piecewise constant densities. For each setting we generate 5 independent problems.

We run ReCoIn for up to 20 iterations, employing a dampening coefficient $\lambda = 0.5$ in two settings that differ by the number of compensating literals $K = 2, 4$. We compare it against the fastest sampling scheme available, the rejection sampler (REJ) implemented in [KMZ19] and the hybrid solver XSDD(Sampling) [ZKD19] employing sophisticated knowledge-compilation [DM02] techniques [KMS18] to guide sampling. For both REJ and XSDD we employ 100 thousand samples per query.

To compare the quality of approximations for a problem, we compute for a model $\mathcal{M}$ the mean integral absolute error (IAE) as $\frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|}\sum_{j=1}^{B}|\mathsf{Pr}_{\mathsf{G}}(X_i \in b_j) - \mathsf{Pr}_{\mathsf{M}}(X_i \in b_j)|$ where we partition the support for each marginal $i = 1, \ldots, n$ into B equal-widths bins b_j for $j = 1, \ldots, B$ and compare the probability Pr_{M} according to model M against the ground truth Pr_{G}, which we compute using PA [MPS17]. We employ PA as it is so far the most reliable general-purpose exact WMI solver [ZMY20a]. Note that as such REJ and XSDD are bounded to solve $|\mathcal{X}| \cdot B$ independent WMI problems, while ReCoIn can naturally amortize $|\mathcal{X}| \cdot B$ queries after a single run of optimization (cf. Section 7.2.1.5). We impose a timeout of 1 hour.

Figure 7.3 reports the IAEs and running times (in seconds) for all problems, settings and competitors. Concerning **Q1** and **Q2**, ReCoIn is the best performer overall. The naive sampling strategy in REJ, while being the fastest as expected, cannot exploit the structure in the problem and clearly suffers from the curse of dimensionality. Conversely, XSDD can deliver accurate approximations thanks to compiling the problem structure, but on highly loopy graphs compilation cannot scale beyond $n = 5$. On the other hand, ReCoIn gracefully scales to larger problem sizes and multiple queries, and delivers very low IAE scores that are close to the best by XSDD on small problem sizes. Note that while ReCoIn can solve much larger problems within our timeout, we could not retrieve a ground truth for them with PA in reasonable time (more than 24 hours per problem).

Concerning **Q3**, more compensating literals ($K = 4$) are achieving marginally lower IAEs at the expense of linearly increasing running times. Exploring the time-accuracy trade-off by increasing K or employing different relaxation strategies is an interesting avenue to investigate

in the future. All in all, this empirical evidence candidates ReCoIn as one of the best general-purpose approximate WMI solvers in the current landscape of WMI solvers.

7.4 Discussion

In this chapter, we advance the WMI framework by tracing the theoretical requirements for tractable WMI inference with the highest precision so far. We introduced ReCoIn as the first solver that by exploiting our tractability insights can reliably scale approximate inference on general WMI problems. We believe these two contributions can help strengthen our theoretical understanding on the challenges and guarantees around approximate hybrid probabilistic inference and at the same time propel the construction of more efficient and scalable WMI solvers.

CHAPTER 8

WMI for Uncertainty Quantification

This chapter shows an application of WMI in quantifying uncertainty. Uncertainty estimation is crucial for decision making. Deep learning models, including those in safety-critical domains, tend to estimate uncertainty poorly. To overcome this issue, Bayesian deep learning obtains a posterior distribution over the model parameters hoping to improve predictions and provide reliable uncertainty estimates. Among Bayesian inference procedures with neural networks, Bayesian model averaging (BMA) is particularly compelling [Was00, FBL18, MIG19]. However, computing BMAs is distinctly challenging since it involves marginalizing over posterior parameters, which possess some unusual topological properties such as mode-connectivity [IVH21]. We show that even with simple low-dimensional approximate parameter posteriors as uniform distributions, doing BMA requires integrating over highly *non-convex* and *multi-modal* distributions with discontinuities arising from non-linear activations (cf. Figure 8.1a). Accurately approximating the BMA can achieve significant performance gains [IVH21]. Existing methods mainly focus on general-purpose MCMC, which can fail to converge, or provides inaccurate few-sample predictions [KEH22], because running longer sampling chains is computationally expensive, and variational approaches that typically use a mean-field approximation that ignores correlations induced by activations [JLB22].

In this work, we are interested in developing *collapsed samplers*, also known as *cutset* or *Rao-Blackwellised* samplers for BMA. A collapsed sampler improves over classical particle-based methods by limiting sampling to a subset of variables and further pairing each sample with a closed-form representation of a conditional distribution over the rest whose marginalization is

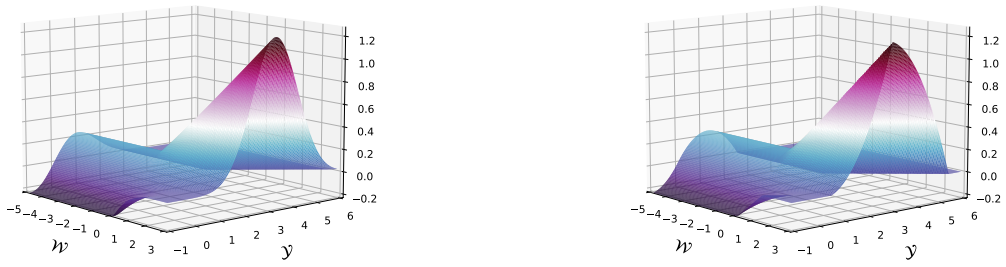
often tractable. Collapsed samplers are effective at variance reduction in graphical models [KF09], however no collapsed samplers are known for Bayesian deep learning. We believe that this is due to the lack of a closed-form marginalization technique congruous with the non-linearity in deep neural networks. Our aim is to overcome this issue and improve BMA estimation by incorporating exact marginalization over (close approximate) conditional distributions into the inference scheme. Nevertheless, scalability and efficiency are guaranteed by the sampling part of our proposed algorithm.

Marginalization is made possible by our observation that BMA reduces to weighted volume computation. Certain classes of such problems can be solved exactly by so-called weighted model integration (WMI) solvers [BPB15]. By closely approximating BMA with WMI, these solvers can provide accurate approximations to marginalization in BMA (cf. Figure 8.1b). With this observation, we propose CIBER, a collapsed sampler that uses WMI for computing conditional distributions. In the few-sample setting, CIBER delivers more accurate uncertainty estimates than the gold-standard Hamiltonian Monte Carlo (HMC) method (cf. Figure 8.2). We further evaluate the effectiveness of CIBER on regression and classification benchmarks and show significant improvements over other Bayesian deep learning approaches in terms of both uncertainty estimation and accuracy.

8.1 Bayesian Model Averaging as Weighted Volume Computation

In *Bayesian Neural Networks (BNN)*, given a neural network $f_{\boldsymbol{w}}$ parameterized by weights $\boldsymbol{w}$, instead of doing inference with deterministic $\boldsymbol{w}$ that optimize objectives such as cross-entropy or mean squared error, Bayesian learning infers a posterior $p(\boldsymbol{w} \mid \mathcal{D})$ over parameters $\boldsymbol{w}$ after observing data $\mathcal{D}$. During inference, this posterior distribution is then marginalized to produce final predictions. This process is called *Bayesian Model Averaging (BMA)*. It can be seen as learning an ensemble of an infinite number of neural nets and aggregating their results. Formally, given input $\boldsymbol{x}$, the posterior predictive distribution and the expected prediction for a regression task are

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \int p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w})\, p(\boldsymbol{w} \mid \mathcal{D})\, d\boldsymbol{w}, \qquad \text{and} \qquad \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}[\boldsymbol{y}] = \int \boldsymbol{y}\, p(\boldsymbol{y} \mid \boldsymbol{x})\, d\boldsymbol{y}. \tag{8.1}$$



(a) $p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w})$ being Gaussian. See Example 10.

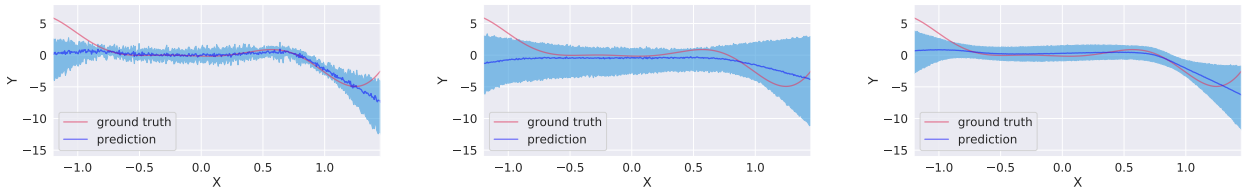
(b) $p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w})$ being triangular. See Section 8.2.

Figure 8.1: The integral surface of (a) the expected prediction in BMA, and (b) our proposed approximation. Both are highly non-convex and multi-modal. The z-axis is the weighted prediction $\boldsymbol{y}\, p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w})\, p(\boldsymbol{w} \mid \mathcal{D})$. Integration of (a) does not admit a closed-form solution, yet integration of (b) is a close approximation that can be solved exactly and efficiently by WMI solvers.

For classification, the (most likely) prediction is the class $\arg\max_{\boldsymbol{y}} p(\boldsymbol{y} \mid \boldsymbol{x})$. BMA is intuitively attractive because it can be risky to base inference on a single neural network model. The marginalization in BMA gets around this issue by averaging over models according to a Bayesian posterior.

BMA requires approximations to compute posterior predictive distributions and expected predictions, as the integrals in Equation 8.1 are intractable in general. Deriving efficient and accurate approximations remains an active research topic [IVH21]. We approach this problem by observing that the marginalization in BMA with ReLU neural networks can be cast as weighted volume computation (WVC). Later we show that it can be generalized to any neural network when combined with sampling. In WVC, various tools exist for solving certain WVC problem classes [BBD14, KMZ19, ZMY20b]. This section reveals the connection between BMA and WVC. It opens up a new perspective for developing BMA approximations by leveraging WVC tools.

Definition 14 (WVC)**.** *A weighted volume computation (WVC) problem is defined by a pair* $(\square, \phi)$ *where a region* $\square$ *is a conjunction of arithmetic constraints and weight* $\phi : \square \to \mathbb{R}$ *is an integrable function assigning weights to elements in* $\square$*. The task of WVC is to compute the integral* $\int_{\square} \phi(\boldsymbol{x})\, d\boldsymbol{x}$*.*



(a) HMC with 10 samples (b) CIBER with 10 collapsed samples (c) HMC with $2k$ samples

Figure 8.2: Uncertainty estimates for regression. The red line is the ground truth. The dark blue line shows the predictive mean. The shaded region is the 90% confidence interval of the predictive distribution. For the same number of samples, (b) CIBER is closer than (a) small-sample HMC to (c) a highly accurate but slow HMC with a large number of samples.

8.1.1 A Warm-Up Example

Consider a simple yet relevant setting where the predictive distribution $p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w})$ is a Dirac delta distribution with zero mass everywhere except at $f_{\boldsymbol{w}}(\boldsymbol{x})$, such that $\int \boldsymbol{y}\, p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w})\, d\boldsymbol{y} = f_{\boldsymbol{w}}(\boldsymbol{x})$.

Example 9. *Assume a model $f_{\boldsymbol{w}}(x) = \mathsf{ReLU}(w \cdot x)$ with a uniform posterior over the parameter: $p(w \mid \mathcal{D}) = \frac{1}{6}$ with $w \in [-3, 3]$. Let the input be $x = 1$. For parameter $w \in [-3, 0]$, the model $f_{\boldsymbol{w}}$ always predicts 0, and otherwise (i.e., $w \in (0, 3]$), it predicts w. Thus, the expected prediction (Equation 8.1) is $\mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}[\boldsymbol{y}] = \int_{\square_\perp} 0 \cdot \frac{1}{6}\, d\boldsymbol{w} + \int_{\square_\top} w \cdot \frac{1}{6}\, d\boldsymbol{w}$. That is, a summation of two WVC problems $(\square_\perp, 0)$ and $(\square_\top, w/6)$ with $\square_\perp = (-3 \leq w \leq 0)$ and $\square_\top = (0 \leq w \leq 3)$. The BMA integral decomposes into WVC problems with different weights due to the* ReLU *activation.*

These WVC problems have easy closed-form solutions. This is no longer the case in the following.

Example 10. *Assume a model $f_{\boldsymbol{w}}(x)$ and posterior distribution $p(w \mid \mathcal{D})$ as in Example 9. Let the predictive distribution $p(y \mid x, w)$ be a Gaussian distribution $p_{\mathcal{N}}(y; f_{\boldsymbol{w}}(x), 1)$ with mean $f_{\boldsymbol{w}}(x)$ and variance 1. Given input $x = 1$, the expected prediction (Equation 8.1) is*

$$\mathbb{E}_{p(y|x=1)}[y] = \int_{\square_\perp} y \cdot p_{\mathcal{N}}(y \mid 0, 1) \cdot \frac{1}{6}\, dy\, dw + \int_{\square_\top} y \cdot p_{\mathcal{N}}(y \mid w, 1) \cdot \frac{1}{6}\, dy\, dw.$$

It is a summation of two WVC problems with $\square_\bot = (-3 \leq w \leq 0) \wedge (y \in \mathbb{R})$ *and* $\square_\top = (0 \leq w \leq 3) \wedge (y \in \mathbb{R})$*, whose joint integral surface is shown in Figure 8.1a.*

These WVC problems do not admit closed-form solutions since they involve truncated Gaussian distributions. Moreover, Figure 8.1a shows that computing BMA, even in such a low-dimensional parameter space, requires integration over non-convex and multi-modal functions.

8.1.2 General Reduction of BMA to WVC

Let model $f_{\boldsymbol{w}}$ be a ReLU neural net. Denote the set of inputs to its ReLU activations by $\mathcal{R} = \{r_i\}_{i=1}^{R}$, where each r_i is a linear combination of weights. For a given input $\boldsymbol{x}$, the parameter space is partitioned by whether each ReLU activation outputs zero or not. This gives the WVC reduction

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \sum_{\boldsymbol{B} \in \{0,1\}^R} \int_{\square_{\boldsymbol{B}}} p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w})\, p(\boldsymbol{w} \mid \mathcal{D})\, d\boldsymbol{w},$$

where $\boldsymbol{B}$ is a binary vector. The region $\square_{\boldsymbol{B}}$ is defined as $\wedge_{i=1}^{R} \ell_i$ where arithmetic constraint ℓ_i is $r_i \geq 0$ if $\boldsymbol{B}_i = 1$ and $r_i \leq 0$ otherwise. The expected prediction $\mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}[\boldsymbol{y}]$ is analogous but includes an additional factor and variable of integration y in each WVC problem.

This general reduction, however, is undesirable since it amounts to a brute-force enumeration that implies a complexity exponential in the number of ReLU activations. Moreover, not all WVC problems resulting from this reduction are amenable to existing solvers. We will therefore appeal to a framework called weighted model integration (WMI) that allows for a compact representation of these WVC problems, and a characterization of their tractability for WMI solvers [KMZ19]. This inspires us to approximate BMA by first reducing it to WVC problems and further closely approximating those with tractable WMI problems.

8.2 Approximating BMA by WMI

WMI is a modeling and inference framework that supports integration in the presence of logical and arithmetic constraints [BPB15, BBP15b]. Various WMI solvers have been proposed in recent

years [KMZ19], ranging from general-purpose ones to others that assume some problem structures to gain scalability. However, even with the reduction from BMA to WVC from the previous section, WMI solvers are not directly applicable. Existing solvers have two main limitations: (i) feasible regions need to be defined by Boolean combinations of linear arithmetic constraints, and (ii) weight functions need to be polynomials. In this section, we show that these issues can be bypassed using a motivating example of how to form a close approximation to BMA using WMI.

In WMI, the feasible region is defined by *satisfiability modulo theories* (SMT) constraints [BMR10]: an SMT formula is a (typically quantifier-free) expression containing both propositional and theory literals connected with logical connectives; the theory literals are often restricted to *linear real arithmetic*, where literals are of the form $(\mathbf{c}^T\mathbf{X} \leq b)$ with variable $\mathbf{X}$ and constants $\mathbf{c}^T$ and b.

Example 11. *The* ReLU *model $f_{\boldsymbol{w}}(x)$ of Example 9 can be encoded as an SMT formula (see box).*

$$\Delta_{\mathsf{ReLU}} = \begin{cases} W \cdot x > 0 \Rightarrow Z = W \cdot x \\ W \cdot x \leq 0 \Rightarrow Z = 0 \end{cases}$$

The curly bracket denotes logical conjunction, the symbol $\Rightarrow$ is a logical implication, variable W is the weight, and variable Z denotes the model output.

The encoding of ReLU neural networks into SMT formulas is explored in existing work to enable verification of the behavior of neural networks and provide formal guarantees [KBD17, HKW17, SFM20]. We propose to use this encoding to define the feasible region of WMI problems. Let $\boldsymbol{x} \models \Delta$ denote the satisfaction of an SMT formula Δ by an assignment $\boldsymbol{x}$, and $[\![\boldsymbol{x} \models \Delta]\!]$ be its corresponding indicator function. We formally introduce WMI next.

Definition 15. *(WMI) Let $\mathcal{X}$ be a set of continuous random variables. A weighted model integration problem is a pair $\mathcal{M} = (\Delta, \Phi)$, where Δ is an SMT formula over $\mathcal{X}$ and Φ is a set of per-literal*

weights defined as $\Phi = \{\phi_\ell\}_{\ell \in \mathcal{L}}$*, where* $\mathcal{L}$ *is a set of SMT literals and each* ϕ_ℓ *is a function defined over variables in literal* ℓ*. The task of weighted model integration is to compute*

$$\mathsf{WMI}(\Delta, \Phi) = \int_{\boldsymbol{x} \models \Delta} \prod_{\ell \in \mathcal{L}} \phi_\ell(\boldsymbol{x})^{[\![\boldsymbol{x} \models \ell]\!]} \, d\boldsymbol{x}.$$

That is, the task is to integrate over the weighted assignments of $\mathcal{X}$ that satisfy the SMT formula Δ.[1]

An approximation to the BMA of Example 10 can be achieved with WMI using the following four steps:

Step 1. Encoding model $f_{\boldsymbol{w}}(x)$**.** This has been shown as the SMT formula Δ_{ReLU} in Example 11.

Step 2. Encoding posterior distribution $p(w \mid \mathcal{D})$**.** The uniform distribution $p(w \mid \mathcal{D}) = \frac{1}{6}$ with $w \in [-3, 3]$ can be encoded as a WMI problem pair $(\Delta_{pos}, \Phi_{pos})$ as follows:

$$\Delta_{pos} = -3 \leq W \leq 3 \qquad \Phi_{pos} = \left\{ \phi_\ell(W) = \frac{1}{6} \;\; \textit{with} \;\; \ell = \texttt{true} \right\}$$

Step 3. Approximate encoding of predictive distribution $p(y \mid w, x)$**.** Recall that $p(y \mid w, x) = p_{\mathcal{N}}(y; f_{\boldsymbol{w}}(x), 1)$ is Gaussian, which cannot be handled by existing WMI solvers. To approximate it with polynomial densities, we simply use a triangular distribution encoded as a WMI problem pair:

$$\Delta_{pred} = \begin{cases} Y \leq Z + \alpha \\ Y \geq Z - \alpha \end{cases} \qquad \Phi_{pred} = \left\{ \begin{array}{l} \phi_{\ell_1}(Y, Z) = \frac{1 - Y + Z}{\alpha} \;\; \textit{with} \;\; \ell_1 = Y \geq Z \\ \phi_{\ell_2}(Y, Z) = \frac{1 + Y - Z}{\alpha} \;\; \textit{with} \;\; \ell_2 = Y < Z \end{array} \right\}$$

In this encoding, α is a constant that defines the shape of the triangular distribution. It is obtained by minimizing the $L2$ distance between a standard normal distribution and the symmetric triangular distribution. We visualize this approximation in Figure 8.3.

[1] In this chapter, since we only work with WMI problems over continuous variables, we ignore the discrete ones in the definition for succinctness.

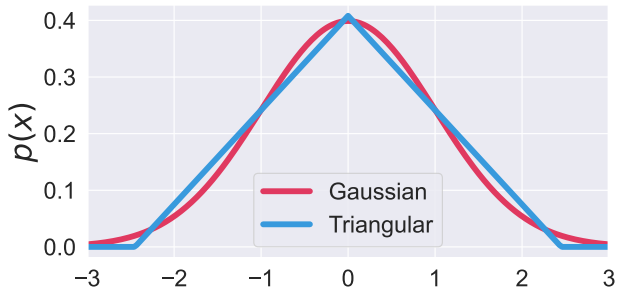


Figure 8.3: Approximating the Gaussian distribution with a triangluar distribution.

Step 4. Approximating BMA by calling WMI solvers. With the above encodings, the predictive posterior $p(y \mid x)$ (Equation 8.1) can be computed using two calls to a WMI solver. For example, the uncertainty of a prediction $y = 1$ for input $x = 1$ is

$$p(y = 1 \mid x = 1) \;=\; \mathsf{WMI}(\Delta \wedge (Y = 1), \Phi) \;/\; \mathsf{WMI}(\Delta, \Phi) \;=\; 0.164 \;/\; 1,$$

where $\Delta = \Delta_{\mathsf{ReLU}} \wedge \Delta_{pos} \wedge \Delta_{pred}$ and $\Phi = \Phi_{pos} \cup \Phi_{pred}$. Similarly, the expected prediction $\mathbb{E}_{p(y|x=1)}[y]$ (Equation 8.1) can be computed using two calls to a WMI solver:

$$\mathbb{E}_{p(y|x=1)}[y] \;=\; \mathsf{WMI}(\Delta, \Phi^*) \;/\; \mathsf{WMI}(\Delta, \Phi) \;=\; 0.752 \;/\; 1,$$

where $\Phi^* = \Phi \cup \{\phi_\ell(Y) = Y \;\; \textit{with} \;\; \ell = \texttt{true}\}$. The above formulations also work for unnormalized distributions since the WMI in the denominator serves to compute the partition function.

We visualize the integral surface of the resulting approximate BMA problem in Figure 8.1b. It is very close to the integral surface of the original BMA problem in Figure 8.1a. However, it can be exactly integrated using existing WMI solvers while the original one does not admit such solutions. Next, we show how this process can be generalized to a scalable and accurate approximation of BMA.

8.3 CIBER: Collapsed Inference for Bayesian Deep Learning via WMI

Given a BNN with a large number of weights, naively approximating it by WMI problems can lead to computational issues, since it involves doing integration over polytopes in arbitrarily high

dimensions and this is known to be #P-hard [Val79, DDK12, ZMY20b]. Further, weights involved with non-ReLU activation might not be amenable to the WMI encoding. To tackle these issues, we propose to use collapsed samples to combine the strengths from two worlds: the scalability and flexibility from sampling and the accuracy from WMI solvers.

Definition 16. *(Collapsed BMA) Let $(\mathcal{W}_s, \mathcal{W}_c)$ be a partition of parameters $\mathcal{W}$. A collapsed sample is a tuple $(\boldsymbol{w}_s, q)$, where $\boldsymbol{w}_s$ is an assignment to the sampled parameters $\mathcal{W}_s$ and q is a representation of the conditional posterior $p(\mathcal{W}_c \mid \boldsymbol{w}_s, \mathcal{D})$ over the collapsed parameter set $\mathcal{W}_c$. Given collapsed samples $\mathcal{S}$,* collapsed BMA *estimates the predictive posterior and expected prediction as*

$$
\begin{aligned}
p(\boldsymbol{y} \mid \boldsymbol{x}) &\approx \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{w}_s, q) \in \mathcal{S}} \left[\int p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w})\; q(\boldsymbol{w}_c)\, d\boldsymbol{w}_c \right], \text{ and} \\
\mathbb{E}_{p(\boldsymbol{y}\mid\boldsymbol{x})}[\boldsymbol{y}] &\approx \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{w}_s, q) \in \mathcal{S}} \left[\int \boldsymbol{y}\; p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w})\; q(\boldsymbol{w}_c)\, d\boldsymbol{w}_c\, d\boldsymbol{y} \right].
\end{aligned}
\tag{8.2}
$$

The size of the collapsed set $\mathcal{W}_c$ determines the trade-off between scalability and accuracy. The more parameters in the collapsed set, the more accurate the approximation to BMA is. The fewer parameters in $\mathcal{W}_c$, the more efficient the computations of the integrals are since the integration is performed in a lower-dimensional space. Later in our experiments, we choose a subset of weights at the last or second-to-last hidden layer of the neural networks to be the collapsed set. This choice is known to be effective in capturing uncertainty as shown in [KHH20, SRS15].

To develop an algorithm to compute collapsed BMA, we are faced with two main design choice questions: **(Q1)** how to sample $\boldsymbol{w}_s$ from the posterior? **(Q2)** what should be the representation of the conditional posterior q such that the integrals in Equation 8.2 can be computed exactly? Next, we provide our answers to these two questions that together give our proposed algorithm CIBER.

8.3.1 Approximation to Posteriors

For **(Q1)**, we follow [MIG19] and sample from the stochastic gradient descent (SGD) trajectory after convergence and use the information contained in SGD trajectories to efficiently approxi-

mate the posterior distribution over the parameters of the neural network, leveraging the interpretation of SGD as approximate Bayesian inference [MHB17, CLT20]. Given a set of parameter samples $\mathcal{W}$ from the SGD trajectory, the sample set is defined as $\mathcal{W}_s = \{\boldsymbol{w}_s \mid \boldsymbol{w} \in \mathcal{W}\}$. For each assignment $\boldsymbol{w}_s$, an approximation $q(\mathcal{W}_c)$ to the conditional posterior $p(\mathcal{W}_c \mid \boldsymbol{w}_s, \mathcal{D})$ is necessary since the posteriors induced by SGD trajectories are implicit. Next, we discuss the choice of approximation to the conditional posterior that is amenable to WMI.

8.3.2 Encoding into WMI Problems

As shown in Section 8.2, if a BNN can be encoded as a WMI problem, the posterior predictive distribution and the expected prediction, which involve marginalization over the parameter space, can be computed exactly using WMI solvers. This inspires us to use the WMI framework as the closed-form representation for the conditional posteriors of parameters. The main challenge is how to approximate the integrand in Equation 8.2 using an SMT formula and a polynomial weight function in order to obtain a WMI problem amenable to existing solvers.

For the conditional posterior approximation $q(\mathcal{W}_c)$, we choose it to be a uniform distribution that can be encoded into a WMI problem as $\mathcal{M}_{pos} = (\Delta_{pos}, \Phi_{pos})$ with the SMT formula being $\Delta_{pos} = \wedge_{i \in c} \, (l_i \leq W_i \leq u_i)$ and weights being $\Phi_{pos} = \{\phi_\ell(\mathcal{W}_c) = 1 \mid \ell = \texttt{true}\}$, where l_i and u_i are domain lower and upper bounds for the uniform distribution respectively. While seemingly over-simplistic, this choice of approximation to the conditional posterior is sufficient to robustly deliver surprisingly strong empirical performance as shown in Section 8.4. The intuition is that uniform distributions are better than a few samples. We further illustrate this point by comparing the predictive distributions of CIBER and HMC in a few-sample setting. Figure 8.2 shows that even with the same 10 samples drawn from the posterior distribution, since CIBER further approximates the 10 samples with a uniform distribution, it yields a predictive distribution closer to the ground truth than HMC, indicating that using a uniform distribution instead of a few samples forms a better approximation.

For the choice of predictive distribution $p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w})$, we propose to use piecewise polynomial

densities. Common predictive distributions can be approximated by polynomials up to arbitrary precision in theory by the Stone–Weierstrass theorem [De 59]. For regression, the de facto choice is Gaussian and we propose to use triangular distribution as the approximation, i.e., $p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w}) = \frac{1}{r} - \frac{1}{r^2}|\boldsymbol{y} - f_{\boldsymbol{w}}(\boldsymbol{x})|$, with domain $|\boldsymbol{y} - f_{\boldsymbol{w}}(\boldsymbol{x})| \leq r$, and $r := \alpha\sqrt{\sigma^2(\boldsymbol{x})}$ where the constant α parameterizes the triangular distribution as described in Section 8.2. Here, $\sigma^2(\boldsymbol{x})$ is the variance estimate, which can be a function of input $\boldsymbol{x}$ depending on whether the BNN is homoscedastic or heteroscedastic. Then $p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w})$ can be encoded into WMI as:

$$\Delta_{pred} = \begin{cases} Y - f_{\boldsymbol{w}}(\boldsymbol{x}) \leq r \\ Y - f_{\boldsymbol{w}}(\boldsymbol{x}) \geq -r \end{cases} \quad \Phi_{pred} = \begin{cases} \phi_{\ell_1}(Y, \mathcal{W}_c) = \frac{1}{r} - \frac{Y - f_{\boldsymbol{w}}(\boldsymbol{x})}{r^2} \;\; \textit{with} \;\; \ell_1 = (Y > f_{\boldsymbol{w}}(\boldsymbol{x})) \\ \phi_{\ell_2}(Y, \mathcal{W}_c) = \frac{1}{r} - \frac{f_{\boldsymbol{w}}(\boldsymbol{x}) - Y}{r^2} \;\; \textit{with} \;\; \ell_2 = (f_{\boldsymbol{w}}(\boldsymbol{x}) > Y) \end{cases}$$

Similar piecewise polynomial approximations are adopted for classification tasks when the predictive distributions are induced by softmax functions.

8.3.3 Exact Integration in Collapsed BMA

By encoding the collapsed BMA into WMI problems, we are ready to answer **(Q2)**, i.e., how to perform exact computation of the integrals shown in Equation 8.2.

Proposition 29. *Let the SMT formula $\Delta = \Delta_{\mathsf{ReLU}} \wedge \Delta_{pos} \wedge \Delta_{pred}$, and the set of weights $\Phi = \Phi_{pos} \cup \Phi_{pred}$ as defined in Section 8.3.2. Let the set of weights $\Phi^* = \Phi \cup \{\phi_\ell(Y) = Y \;\; \textit{with} \;\; \ell = \texttt{true}\}$. The integrals in collapsed BMA (Equation 8.2) can be computed by WMI solvers as*

$$\int p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w})\; q(\boldsymbol{w}_c)\, d\boldsymbol{w}_c = \mathsf{WMI}(\Delta \wedge (\boldsymbol{Y} = \boldsymbol{y}), \Phi) \;/\; \mathsf{WMI}(\Delta, \Phi), \;\; \textit{and}$$

$$\int \boldsymbol{y}\; p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w}) \;\; q(\boldsymbol{w}_c)\, d\boldsymbol{w}_c\, d\boldsymbol{y} = \mathsf{WMI}(\Delta, \Phi^*) \;/\; \mathsf{WMI}(\Delta, \Phi).$$

With both questions **(Q1)** and **(Q2)** answered, we summarize our proposed algorithm CIBER in Algorithm 12. To quantitatively analyze how close the approximation delivered by CIBER is to the ground-truth BMA, we consider the following experiments with closed-form BMA.

Regression. We consider a Bayesian linear regression setting where exact sampling from the posterior distribution is available. Both the likelihood and the weight posterior are Gaussian

Algorithm 12 CIBER

Input: input $\boldsymbol{x}$, sampled weights $\mathcal{W}$, neural network model $f_{\boldsymbol{w}}$, prediction ground truth y^*

Ouput: predictions and likelihoods

1: Choose a partition $(\mathcal{W}_s, \mathcal{W}_c)$ for network parameters
2: Derive approximate posterior $q(\boldsymbol{w}_c)$ from sampled weights $\{\boldsymbol{w}_c \mid \boldsymbol{w} \in \mathcal{W}\}$ *// cf. Section 8.3.2*
3: Encode posterior $q(\boldsymbol{w}_c)$ into WMI problem $\mathcal{M}_{pos} = (\Delta_{pos}, \Phi_{pos})$ *// cf. Section 8.3.2*
4: $\mathcal{Y} \leftarrow \emptyset, \mathcal{P} \leftarrow \emptyset$ *// Initialization*
5: **for** sample $\boldsymbol{w}_s$ in $\{\boldsymbol{w}_s \mid \boldsymbol{w} \in \mathcal{W}\}$ **do**
6: Encode neural network model f_{ReLU} parameterized by $(\boldsymbol{w}_s, \mathcal{W}_c)$ into an SMT formula $\Delta_{f_{\boldsymbol{w}}}$
7: Encode predictive $p(Y \mid \boldsymbol{x}, \boldsymbol{w}_s, \mathcal{W}_c)$ into a WMI problem $\mathcal{M}_{pred} = (\Delta_{pred}, \Phi_{pred})$
8: SMT formula $\Delta \leftarrow \Delta_{\mathsf{ReLU}} \wedge \Delta_{pos} \wedge \Delta_{pred}$
9: Weights $\Phi \leftarrow \Phi_{pos} \cup \Phi_{pred}$
10: Weights $\Phi^* \leftarrow \Phi \cup \{\phi_\ell(Y) = Y \ \textit{with} \ \ell = \texttt{true}\}$
11: Add prediction $y = \mathsf{WMI}(\Delta, \Phi^*)/\mathsf{WMI}(\Delta, \Phi)$ to prediction set $\mathcal{Y}$ *// cf. Section 8.3.3*
12: Add likelihood $p = \mathsf{WMI}(\Delta \wedge (Y = y^*), \Phi)/\mathsf{WMI}(\Delta, \Phi)$ to set $\mathcal{P}$ *// cf. Section 8.3.3*
13: **return** $y = \textsc{Mean}(\mathcal{Y}), p(y^* \mid \boldsymbol{x}) = \textsc{Mean}(\mathcal{P})$

such that the ground-truth posterior predictive distribution is Gaussian as well. With samples drawn from the weight posterior, CIBER approximates the samples with a uniform distribution as posterior $p(\boldsymbol{w}|\mathcal{D})$ and further approximates the likelihood with a triangular distribution such that the integral $p(\boldsymbol{y}|\boldsymbol{x}, \mathcal{D}) = \int p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) p(\boldsymbol{w}|\mathcal{D})\, d\boldsymbol{w}$ can be computed exactly by WMI.

We first evaluate the posterior predictive distribution estimated by CIBER and Monte Carlo (MC) method, using the same five samples drawn from the weight posterior. Results averaged over 10 trials are shown in Figure 8.4 where the estimations by CIBER are much closer to the ground truth posterior predictive distribution than those by the MC method. Further, the averaged KL divergence between the ground truth and CIBER is 0.069 while the one for MC estimations is 0.130,

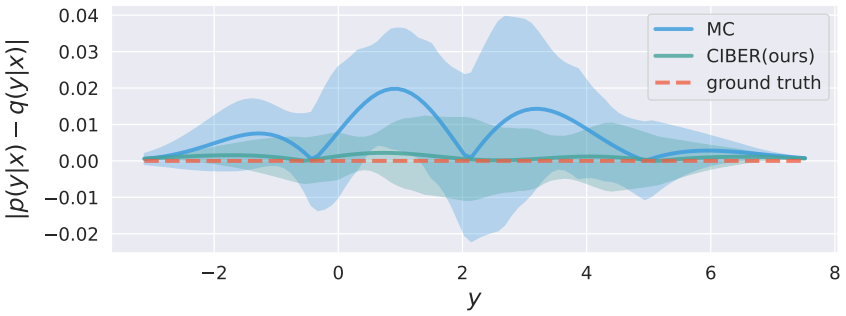


Figure 8.4: Posterior predictive distributions in Bayesian linear regression. The y-axis shows the absolute difference between an estimated predictive distribution $p(y \mid \mathbf{x})$ and the ground-truth predictive distribution $q(y \mid \mathbf{x})$. Shaded regions are the 95% confidence interval.

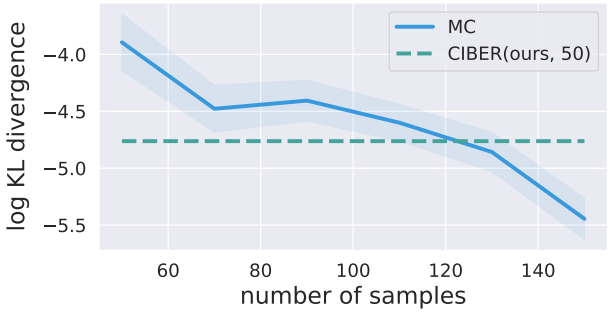


Figure 8.5: KL divergence in Bayesian linear regression. The x-axis shows the number of samples the MC method uses for estimations, ranging from 50 to 150. The blue curve shows the MC method, and the green dashed curve shows CIBER using 50 samples.

again indicating that CIBER yields a better BMA approximation in the few-sample setting.

We further explore the question of how many samples the MC method needs to match the performance of CIBER. The performances of both approaches are evaluated using KL divergence between the ground-truth posterior distribution and the estimated one, averaged over 10 trials. The result is shown in Figure 8.5 where the dashed green line shows the performance of CIBER with 50 samples and the blue curve shows the performance of MC with an increasing number of samples. As expected, the MC method yields lower KL divergence as the number of samples increases; however, it takes more than 100 samples to match CIBER, indicating its low sample

efficiency and that developing efficient and effective inference algorithms such as CIBER for estimating BMA is a meaningful question.

Classification. For analyzing classification performance, [KEH22] propose to compute the integral $I = \int \sigma(f_*)p_{\mathcal{N}}(f_*)\,df_*$ with σ being the sigmoid function and $f_* = f(\boldsymbol{x}^*; \boldsymbol{w})$ that amounts to the posterior predictive distribution. We consider a simple case with $f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{x}$ such that the ground-truth integral can be obtained. With a randomly chosen $\boldsymbol{x}$, the ground-truth integral is $I = 0.823$. The integral estimated by CIBER is $I_C = 0.826$ while the MC estimate is $I_{MC} = 0.732$. That is, CIBER gives an estimate with a much lower error than the MC estimation error, indicating that CIBER is able to deliver high-quality approximations in classification tasks.

Related Work

Bayesian Deep Learning. Bayesian inference over deep neural networks [Mac92] is proposed to fix the issue that deep learning models give poor uncertainty estimations and suffer from overconfidence [NYC15, HAB19, MTP23, MTS21]. Some methods use samples from SGD trajectories to approximate the implicit true posteriors similar to us: [IMK20] (SI) proposes to perform Bayesian inference in a subspace of the parameter space spanned by a few vectors derived from principal component analysis (PCA+ESS(SI)) or variational inference (PCA+VI(SI)); SWAG [MIG19] proposes to approximate the full parameter space using an approximate Gaussian posterior whose mean and covariance are from a partial SGD trajectory with a modified learning rate scheduler.

Some other approaches using approximate posteriors include MC Dropout (MCD) [GG15, GG16] which is one of the Bayesian dropout methods and recently, one of its modifications called Variational Structured Dropout (VSD) [NNN21] using variational inference is proposed. Other state-of-the-art approximate BNN inference methods including deterministic variational inference (DVI) [WNM19], deep Gaussian processes (DGP) [BHH16] with Gaussian process layers and variational inference (VI) [KW13]. Closely related to DGP is the deep kernel process [AYO21] that writes DGPs as deep Wishart processes.

WMI Solvers. WMI generalizes weighted model counting (WMC) [SBK05], a state-of-the-art inference approach in many discrete probabilistic models, from discrete to mixed discrete-continuous domains [BPB15, BBP15b]. Recent research on WMI includes its tractability [ZMY20b, ZMY21a, ACD20] and the advancements in WMI solver development. Existing exact WMI solvers for arbitrarily structured problems include DPLL-based search with numerical [BPB15, MPS17, MPS19] or symbolic integration [SOG16] and compilation-based algorithms [KMS18, ZDD19, DHD20] that use extended algebraic decision diagrams (XADDs) [SDD12] as a compilation target which is a powerful tool for inference on mixed domains [SA12, ZSF12]. Some exact WMI solvers aiming to improve efficiency for a certain class of models are proposed such as SMI [ZB19] and MP-WMI [ZMY20a] which are greatly scalable for WMI problems that satisfy certain structural constraints. Approximate solvers are also proposed including sampling-based ones [Zui20] and relaxation-based ones [ZMY20b, ZMY20c]. Recent WMI efforts converge in the `pywmi` library [KMZ19]. The SMT formulas considered in this work can be seen as distributional constraints on continuous domains. There is also plenty of work in neuro-symbolic AI exploring the integration of discrete constraints into neural networks models including the architectures [ATC22a, AZN23c] and the loss [XZF18, AWC22, ALT22].

8.4 Empirical Evaluation

We conduct experimental evaluations of our proposed approach CIBER [1] on regression and classification benchmarks and compare its performance on uncertainty estimation as well as prediction accuracy with a wide range of baseline methods.

8.4.1 Regression on Small and Large UCI Datasets

We experiment on 5 small UCI datasets: *boston*, *concrete*, *yacht*, *naval* and *energy*. We follow the setup of [IMK20] and use a fully connected network with a single hidden layer and 50 units with ReLU activations. We further experiment on 6 large UCI datasets: *elevators*, *keggdirected*,

[1] Code and experiments are available at `https://github.com/UCLA-StarAI/CIBER`.

Table 8.1: Average test log likelihood for the small UCI regression task.

	Boston	Concrete	Yacht	Naval	Energy
Ciber (second)	**-2.471 ± 0.140**	-2.975 ± 0.102	-0.678 ± 0.301	7.276 ± 0.532	**-0.716 ± 0.211**
Ciber (last)	**-2.471 ± 0.140**	**-2.959 ± 0.109**	-0.687 ± 0.301	**7.482 ± 0.188**	**-0.716 ± 0.211**
SWAG	-2.761 ± 0.132	-3.013 ± 0.086	-0.404 ± 0.418	6.708 ± 0.105	-1.679 ± 1.488
PCA+ESS (SI)	-2.719 ± 0.132	-3.007 ± 0.086	**-0.225 ± 0.400**	6.541 ± 0.095	-1.563 ± 1.243
PCA+VI (SI)	-2.716 ± 0.133	-2.994 ± 0.095	-0.396 ± 0.419	6.708 ± 0.105	-1.715 ± 1.588
SGD	-2.752 ± 0.132	-3.178 ± 0.198	-0.418 ± 0.426	6.567 ± 0.185	-1.736 ± 1.613
DVI	-2.410 ± 0.020	-3.060 ± 0.010	-0.470 ± 0.030	6.290 ± 0.040	-1.010 ± 0.060
DGP	-2.330 ± 0.060	-3.130 ± 0.030	-1.390 ± 0.140	3.600 ± 0.330	-1.320 ± 0.030
VI	-2.430 ± 0.030	-3.040 ± 0.020	-1.680 ± 0.040	5.870 ± 0.290	-2.380 ± 0.020
MCD	-2.400 ± 0.040	-2.970 ± 0.020	-1.380 ± 0.010	4.760 ± 0.010	-1.720 ± 0.010
VSD	-2.350 ± 0.050	-2.970 ± 0.020	-1.140 ± 0.020	4.830 ± 0.010	-1.060 ± 0.010

Table 8.2: Average test RMSE for the small UCI regression task.

	Boston	Concrete	Yacht	Naval	Energy
Ciber (second)	3.488 ± 1.123	4.880 ± 0.506	0.828 ± 0.241	**0.000 ± 0.000**	**0.447 ± 0.081**
Ciber (last)	3.478 ± 1.128	**4.854 ± 0.503**	**0.752 ± 0.294**	**0.000 ± 0.000**	**0.447 ± 0.081**
SWAG	3.517 ± 0.981	5.233 ± 0.417	0.973 ± 0.375	0.001 ± 0.000	1.594 ± 0.273
PCA+ESS (SI)	3.453 ± 0.953	5.194 ± 0.448	0.972 ± 0.375	0.001 ± 0.000	1.598 ± 0.274
PCA+VI (SI)	**3.457 ± 0.951**	5.142 ± 0.418	0.973 ± 0.375	0.001 ± 0.000	1.587 ± 0.272
SGD	3.504 ± 0.975	5.194 ± 0.446	0.973 ± 0.374	0.001 ± 0.000	1.602 ± 0.275
MCD	2.830 ± 0.170	4.930 ± 0.140	0.720 ± 0.050	0.000 ± 0.000	1.080 ± 0.030
VSD	2.640 ± 0.170	4.720 ± 0.110	0.690 ± 0.060	0.000 ± 0.000	0.470 ± 0.010

keggundirected, *pol*, *protein* and *skillcraft*. We use a feedforward network with five hidden layers of sizes $[1000, 1000, 500, 50, 2]$ and ReLU activations on all datasets except *skillcraft*. For *skillcraft*, a smaller architecture is adopted with four hidden layers of size $[1000, 500, 50, 2]$. All models have two outputs for the prediction and the heteroscedastic variance respectively.

We run Ciber with two different ways of choosing the collapsed parameter set: *Ciber (last)* chooses all the weights at the last layer to be the collapsed set; *Ciber (second)* chooses three out of all the weights at the second-to-last layer to be the collapsed set. The heuristic we use for choosing the weights is to look into the sampled weights from SGD trajectories to see which ones have the greatest variance. The intuition is that a greater variance indicates that the weight is prone to have greater uncertainty and thus one might want to perform a more accurate inference over it.

Baselines. We compare Ciber to the state-of-the-art approximate BNN inference methods. We separate these methods into two categories: those sampling from SGD trajectories as approximate posteriors, which includes SWAG [MIG19], PCA+ESS (SI) and PCA+VI (SI) [IMK20], vs.

Table 8.3: Average test log likelihood for the large UCI regression task.

	Elevators	KeggD	KeggU	Protein	Skillcraft	Pol
Ciber (second)	-0.378 ± 0.026	**1.245 ± 0.090**	**1.125 ± 0.269**	-0.720 ± 0.036	-1.003 ± 0.035	**2.555 ± 0.115**
Ciber (last)	-0.371 ± 0.023	1.178 ± 0.088	0.964 ± 0.231	-0.720 ± 0.036	**-1.001 ± 0.032**	2.506 ± 0.150
SWAG	-0.374 ± 0.021	1.080 ± 0.035	0.749 ± 0.029	**-0.700 ± 0.051**	-1.180 ± 0.033	1.533 ± 1.084
PCA+ESS (SI)	-0.351 ± 0.030	1.074 ± 0.034	0.752 ± 0.025	-0.734 ± 0.063	-1.181 ± 0.033	-0.185 ± 2.779
PCA+VI (SI)	**-0.325 ± 0.019**	1.085 ± 0.031	0.757 ± 0.028	-0.712 ± 0.057	-1.179 ± 0.033	1.764 ± 0.271
SGD	-0.538 ± 0.108	1.012 ± 0.154	0.602 ± 0.224	-0.854 ± 0.085	-1.162 ± 0.032	1.073 ± 0.858
OrthVGP	-0.448	1.022	0.701	-0.914	—	0.159
NL	-0.698 ± 0.039	0.935 ± 0.265	0.670 ± 0.038	-0.884 ± 0.025	-1.002 ± 0.050	-2.840 ± 0.226

those who do not, which includes the SGD baseline, deterministic variational inference baseline (DVI) [WNM19], Deep Gaussian Processes (DGP) [BHH16], variational inference (VI) [KW13], MC Dropout (MCD) [GG15, GG16], and variational structured dropout (VSD) [NNN21]. These methods achieved state-of-the-art performance on the small UCI datasets. We also compare to baselines Bayesian final layers (NL) [RTS18], deep kernel learning (DKL) [WHS16], orthogonally decoupled variational GPs (OrthVGP) [SCB18] and Fastfood approximate kernels (FF) [YWS15], which have achieved state-of-the-art performance on the large UCI datasets.

Results. We present the test log likelihoods for small UCI datasets in Table 8.1 and those for large UCI datasets in Table 8.3. In both tables, the first block summarizes SGD-trajectory sampling-based approaches and the second summarizes the rest. Underlined results are the best among all and bold results are the best among SGD-trajectory sampling-based approaches. From the results, our Ciber has substantially better performance than all others on three out of the five small UCI datasets four out of six large UCI datasets, with comparable performance on the rest, demonstrating that Ciber provides accurate uncertainty estimation. We also present the test rooted-mean-squared error results, where Ciber outperforms all other SGD-trajectory sampling-based baselines on four out of five small UCI datasets and four out of six large UCI datasets; it

Table 8.4: Average test RMSE for the large UCI regression task.

	Elevators	KeggD	KeggU	Protein	Skillcraft	Pol
Ciber (second)	**0.088 ± 0.002**	0.142 ± 0.074	**0.115 ± 0.007**	0.438 ± 0.009	**0.251 ± 0.010**	2.212 ± 0.230
Ciber (last)	**0.088 ± 0.002**	0.142 ± 0.072	0.118 ± 0.012	0.438 ± 0.009	**0.251 ± 0.010**	**2.199 ± 0.182**
SWAG	**0.088 ± 0.001**	0.129 ± 0.029	0.160 ± 0.043	**0.415 ± 0.018**	0.293 ± 0.015	3.110 ± 0.070
PCA+ESS (SI)	0.089 ± 0.002	0.129 ± 0.028	0.160 ± 0.043	0.425 ± 0.017	0.293 ± 0.015	3.755 ± 6.107
PCA+VI (SI)	**0.088 ± 0.001**	**0.128 ± 0.028**	0.160 ± 0.043	0.418 ± 0.021	0.293 ± 0.015	2.499 ± 0.684
SGD	0.103 ± 0.035	0.132 ± 0.017	0.186 ± 0.034	0.436 ± 0.011	0.288 ± 0.014	3.900 ± 6.003
NL	0.101 ± 0.002	0.134 ± 0.036	0.120 ± 0.003	0.447 ± 0.012	0.253 ± 0.011	4.380 ± 0.853
DKL	0.084 ± 0.020	0.100 ± 0.010	0.110 ± 0.000	0.460 ± 0.010	0.250 ± 0.000	6.617
OrthVGP	0.095	0.120	0.117	0.461	–	4.300 ± 0.200
FF	0.089 ± 0.002	0.120 ± 0.000	0.120 ± 0.000	0.470 ± 0.010	0.250 ± 0.020	–

outperforms all baselines on two small UCI datasets and one large UCI datasets and has comparable performance on the rest. This further illustrates that exact marginalization over conditional approximate posteriors enabled by WMI solvers achieves accurate estimation of the true BMA and boosts predictive performance.

8.4.2 Image Classification

CIFAR datasets. We experiment with two image datasets: CIFAR-10 and CIFAR-100 [KH09b] and evaluate the test performance using three metrics: 1) negative log likelihood (NLL) that reflects the quality of both uncertainty estimation and prediction accuracy, 2) classification accuracy (ACC), and 3) expected calibration errors (ECE) [NCH15] that show the difference between predictive confidence and accuracy and should be close to zero for a well-calibrated approach.

We run Ciber by choosing the collapsed parameter set to be 10 weights and 100 weights at the last layer of the neural network models for CIFAR-10 and CIFAR-100 respectively. The weights are chosen using the same heuristic as the one for regression tasks, i.e., to choose the weights

Table 8.5: Average test performance for image classification tasks on CIFAR-10 and CIFAR-100.

Metric	NLL		ACC		ECE	
Dataset	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
CIBER	**0.1927 ± 0.0029**	**0.9193 ± 0.0027**	**93.64 ± 0.09**	**74.71 ± 0.18**	0.0130 ± 0.0011	**0.0168 ± 0.0025**
SWAG	0.2503 ± 0.0081	1.2785 ± 0.0031	93.59 ± 0.14	73.85 ± 0.25	0.0391 ± 0.0020	0.1535 ± 0.0015
SGD	0.3285 ± 0.0139	1.7308 ± 0.0137	93.17 ± 0.14	73.15 ± 0.11	0.0483 ± 0.0022	0.1870 ± 0.0014
SWA	0.2621 ± 0.0104	1.2780 ± 0.0051	93.61 ± 0.11	74.30 ± 0.22	0.0408 ± 0.0019	0.1514 ± 0.0032
SGLD	0.2001 ± 0.0059	0.9699 ± 0.0057	93.55 ± 0.15	74.02 ± 0.30	**0.0082 ± 0.0012**	0.0424 ± 0.0029
KFAC	0.2252 ± 0.0032	1.1915 ± 0.0199	92.65 ± 0.20	72.38 ± 0.23	0.0094 ± 0.0005	0.0778 ± 0.0054

Table 8.6: Average test performance for image transfer learning tasks.

Metric	NLL		ACC		ECE	
Model	VGG-16	PreResNet-164	VGG-16	PreResNet-164	VGG-16	PreResNet-164
CIBER	**0.9869 ± 0.0102**	**0.9684 ± 0.0075**	**72.56 ± 0.23**	75.70 ± 0.17	**0.0925 ± 0.0028**	**0.0704 ± 0.0031**
SWAG	1.3425 ± 0.0015	1.3842 ± 0.0122	72.30 ± 0.11	**76.30 ± 0.06**	0.1988 ± 0.0028	0.1668 ± 0.0006
SGD	1.6528 ± 0.0390	1.4790 ± 0.0000	72.42 ± 0.07	75.56 ± 0.00	0.2149 ± 0.0027	0.1758 ± 0.0000
SWA	1.3993 ± 0.0502	1.3552 ± 0.0000	71.92 ± 0.01	76.02 ± 0.00	0.2082 ± 0.0056	0.1739 ± 0.0000

whose samples from the SGD trajectories have large variances. We compare Ciber with strong baselines including SWAG [MIG19] reproduced by their open-source implementation, standard SGD, SWA [IPG18], SGLD [WT11] and KFAC [RBB18].

Transfer from CIFAR-10 to STL-10. We further consider a transfer learning task using the model trained on CIFAR-10 to be evaluated on dataset STL-10 [CNL11]. STL-10 shares nine out of ten classes with the CIFAR-10 dataset but has a different image distribution. It is a common benchmark in transfer learning to adapt models trained on CIFAR-10 to STL-10.

Results. We present the test classification performance on dataset CIFAR-10 and CIFAR-100

in Table 8.5 and that of transfer learning in Table 8.6. The neural network models used in the classification task are VGG-16 networks and the models used in the transfer learning task are VGG-16 and PreResNet-164. With the same number of samples as SWAG, CIBER outperforms SWAG and other baselines in most evaluations and delivers comparable performance otherwise, demonstrating the effectiveness of using collapsed samples in improving uncertainty estimation as well as classification performance.

8.5 Discussion

We reveal the connection between BMA, a way to perform Bayesian deep learning and WVC, which inspires us to approximate BMA using the framework of WMI. To further make this approximation scalable and flexible, we combine it with collapsed samples which gives our algorithm CIBER. CIBER compares favorably to Bayesian deep learning baselines on regression and classification tasks. A future direction would be to explore what other layers can be expressed as SMT formulas and thus amenable to SMT encoding. Also, the current WMI solvers are limited to polynomial weights, and thus the reduction to WMI problems is applicable to piecewise polynomial weights. This limitation might be alleviated in the future by the development of new WMI solvers that allow various weight function families.

CHAPTER 9

Conclusion

Nowadays people have realized that there is a limit to what AI models can do without symbolic representations. Neurosymbolic AI is considered as the third wave of AI development, following the earlier waves dominated by symbolic AI and deep learning, and is a promising way to build next-generation AI models. This thesis summarize my research on tackling the fundamental challenges in neurosymbolic learning and reasoning to advance the field. These approaches are applied to build AI models that behave as intended to achieve trustworthiness.

For neurosymbolic learning, a key insight in our work is that *constraint probability lies at the center of the differentiable learning under constraints*–not only enabling the development of gradient estimators achieving lower bias and variances compared to state-of-the-art estimators but also aiding in deriving probabilistically sound training objectives. Our estimator is effective in latent space regularization for deep generative models and explainability for learning to explain tasks. It leads to direct follow-up work extending it to learning graph structure in a task-adaptive way for graph neural networks, the de facto standard models for geometry-heavy applications. Further, these techniques are extended to derive a unified framework for count-based weakly-supervised learning tasks.

At the core of scaling neurosymbolic models lies the development of efficient reasoning algorithms for highly expressive models. For the reasoning part, we focus on probabilistic reasoning over arithmetic constraints that arise in scenarios such as stochastic neural networks with piecewise linear activation functions. We use the the expressive WMI framework that allows such reasoning, enabling the modeling of structured, complex, and noisy data. Still, its expressiveness

comes at a cost: building reasoning algorithms for WMI problems is highly challenging. I develop a series of techniques to perform both exact and approximate inference, and further scale to large problems. The development of WMI solvers remains an open challenge and it will benefit tons of applications given its expressiveness in representing domain knowledge.

In summary, neurosymbolic AI plays a pivotal role in advancing the trustworthiness of AI by ensuring transparency, consistency, and reliability in decision-making processes, making it a critical research field for the sustainable development of AI technologies.

References

[AAB19] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. "Differentiable convex optimization layers." *Advances in neural information processing systems*, **32**, 2019.

[AC17] Ehsan Mohammady Ardehaly and Aron Culotta. "Co-training for demographic classification using deep learning from label proportions." In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 1017–1024. IEEE, 2017.

[ACB23] Kareem Ahmed, Kai-Wei Chang, and Guy Van den Broeck. "Semantic Strengthening of Neuro-Symbolic Learning." In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS)*, apr 2023.

[ACD20] Ralph Abboud, Ismail Ilkan Ceylan, and Radoslav Dimitrov. "On the Approximability of Weighted Model Integration on DNF Structures." *arXiv preprint arXiv:2002.06726*, 2020.

[ACG20] Ralph Abboud, Ismail Ilkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. "The surprising power of graph neural networks with random node initialization." *arXiv preprint arXiv:2010.01179*, 2020.

[AD15] Hadi Mohasel Afshar and Justin Domke. "Reflection, refraction, and hamiltonian monte carlo." In *Advances in neural information processing systems*, pp. 3007–3015, 2015.

[ADC22] Ralph Abboud, Radoslav Dimitrov, and Ismail Ilkan Ceylan. "Shortest path networks for graph property prediction." In *Learning on Graphs Conference*, pp. 5–1. PMLR, 2022.

[ALT22] Kareem Ahmed, Tao Li, Thy Ton, Quan Guo, Kai-Wei Chang, Parisa Kordjamshidi, Vivek Srikumar, Guy Van den Broeck, and Sameer Singh. "PYLON: A PyTorch Frame-

work for Learning with Constraints." In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (Demo Track)*, feb 2022.

[AMD14] Yoann Altmann, Steve McLaughlin, and Nicolas Dobigeon. "Sampling from a multivariate Gaussian distribution truncated on a simplex: a review." In *2014 IEEE Workshop on Statistical Signal Processing (SSP)*, pp. 113–116. IEEE, 2014.

[APK19] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing." In *International Conference on Machine Learning*, pp. 21–29. PMLR, 2019.

[ASA15] Hadi Mohasel Afshar, Scott Sanner, and Ehsan Abbasnejad. "Linear-time gibbs sampling in piecewise graphical models." In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[ASW16] Hadi Mohasel Afshar, Scott Sanner, and Christfried Webers. "Closed-Form Gibbs Sampling for Graphical Models with Algebraic Constraints." In *AAAI*, 2016.

[ATC22a] Kareem Ahmed, Stefano Teso, Kai-Wei Chang, Guy Van den Broeck, and Antonio Vergari. "Semantic Probabilistic Layers for Neuro-Symbolic Learning." In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, dec 2022.

[ATC22b] Kareem Ahmed, Stefano Teso, Kai-Wei Chang, Guy Van den Broeck, and Antonio Vergari. "Semantic probabilistic layers for neuro-symbolic learning." *Advances in Neural Information Processing Systems*, **35**:29944–29959, 2022.

[ATH02] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. "Support Vector Machines for Multiple-Instance Learning." In *NIPS*, 2002.

[AWC21] Kareem Ahmed, Eric Wang, Kai-Wei Chang, and Guy Van den Broeck. "Leverag-

ing Unlabeled Data for Entity-Relation Extraction through Probabilistic Constraint Satisfaction.", mar 2021.

[AWC22] Kareem Ahmed, Eric Wang, Kai-Wei Chang, and Guy Van den Broeck. "Neuro-Symbolic Entropy Regularization." In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022.

[AY21] Uri Alon and Eran Yahav. "On the Bottleneck of Graph Neural Networks and its Practical Implications." In *International Conference on Learning Representations*, 2021.

[AYO21] Laurence Aitchison, Adam Yang, and Sebastian W Ober. "Deep kernel processes." In *International Conference on Machine Learning*, pp. 130–140. PMLR, 2021.

[AZN23a] Kareem Ahmed, Zhe Zeng, Mathias Niepert, and Guy Van den Broeck. "SIMPLE: A Gradient Estimator for k-subset sampling." In *International Conference on Learning Representations*, 2023.

[AZN23b] Kareem Ahmed, Zhe Zeng, Mathias Niepert, and Guy Van den Broeck. "SIMPLE: A Gradient Estimator for k-subset sampling." In *Proceedings of the International Conference on Learning Representations (ICLR)*, may 2023.

[AZN23c] Kareem Ahmed, Zhe Zeng, Mathias Niepert, and Guy Van den Broeck. "SIMPLE: A Gradient Estimator for k-subset sampling." In *Proceedings of the International Conference on Learning Representations (ICLR)*, may 2023.

[BBD11] Velleda Baldoni, Nicole Berline, Jesus De Loera, Matthias Köppe, and Michèle Vergne. "How to integrate a polynomial over a simplex." *Mathematics of Computation*, **80**(273):297–325, 2011.

[BBD14] Velleda Baldoni, Nicole Berline, Jesús A De Loera, Brandon Dutra, Matthias Köppe, Stanislav Moreinis, Gregory Pinto, Michele Vergne, and Jianqiu Wu. "A user's guide for LattE integrale v1. 7.2." *Optimization*, **22**(2), 2014.

[BBP15a] Vaishak Belle, Guy Van den Broeck, and Andrea Passerini. "Hashing-Based Approximate Probabilistic Inference in Hybrid Domains." In *UAI*, pp. 141–150, 2015.

[BBP15b] Vaishak Belle, Guy Van den Broeck, and Andrea Passerini. "Hashing-Based Approximate Probabilistic Inference in Hybrid Domains." In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (UAI)*, 2015.

[BBP16] Vaishak Belle, Guy Van den Broeck, and Andrea Passerini. "Component Caching in Hybrid Domains with Piecewise Polynomial Densities." In *AAAI*, pp. 3369–3375, 2016.

[BBT20] Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis R. Bach. "Learning with Differentiable Pertubed Optimizers." In *NeurIPS*, 2020.

[BD20] Jessa Bekker and Jesse Davis. "Learning from positive and unlabeled data: A survey." *Machine Learning*, **109**:719–760, 2020.

[BDO18] Gerda Bortsova, Florian Dubost, Silas Ørting, Ioannis Katramados, Laurens Hogeweg, Laura Thomsen, Mathilde Wille, and Marleen de Bruijne. "Deep learning from label proportions for emphysema quantification." In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II 11*, pp. 768–776. Springer, 2018.

[BDP09] Fahiem Bacchus, Shannon Dalmao, and Toniann Pitassi. "Solving #SAT and Bayesian inference with backtracking search." *Journal of Artificial Intelligence Research*, **34**:391–442, 2009.

[BES80] László Babai, Paul Erdos, and Stanley M Selkow. "Random graph isomorphism." *SIaM Journal on computing*, **9**(3):628–635, 1980.

[BFG96] Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. "Context-specific independence in Bayesian networks." In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, pp. 115–123. Morgan Kaufmann Publishers Inc., 1996.

[BFW21] Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F Montufar, Pietro Lio, and Michael Bronstein. "Weisfeiler and lehman go topological: Message passing simplicial networks." In *International Conference on Machine Learning*, pp. 1026–1037. PMLR, 2021.

[BFZ22] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M Bronstein. "Improving graph neural network expressivity via subgraph isomorphism counting." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **45**(1):657–668, 2022.

[BGS16] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. "Importance Weighted Autoencoders.", 2016.

[BGS22] Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. "Logic tensor networks." *Artificial Intelligence*, **303**:103649, 2022.

[BHH16] Thang Bui, Daniel Hernández-Lobato, Jose Hernandez-Lobato, Yingzhen Li, and Richard Turner. "Deep Gaussian processes for regression using approximate expectation propagation." In *International conference on machine learning*, pp. 1472–1481. PMLR, 2016.

[BHM09] Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, volume 185. IOS press, 2009.

[BK79] Laszlo Babai and Ludik Kucera. "Canonical labelling of graphs in linear average time." In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pp. 39–46. IEEE, 1979.

[BKM20] Pablo Barceló, Egor V. Kostylev, Mikael Monet, Jorge Pérez, Juan Reutter, and Juan Pablo Silva. "The Logical Expressiveness of Graph Neural Networks." In *International Conference on Learning Representations*, 2020.

[BKW22] Pradeep Kr Banerjee, Kedar Karhadkar, Yu Guang Wang, Uri Alon, and Guido Montúfar. "Oversquashing in GNNs through the lens of information contraction and graph expansion." In *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1–8. IEEE, 2022.

[Blo19] Mathieu Blondel. "Structured prediction with projection oracles." *Advances in neural information processing systems*, **32**, 2019.

[BM98] Avrim Blum and Tom Mitchell. "Combining Labeled and Unlabeled Data with Co-Training." In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, p. 92–100, New York, NY, USA, 1998. Association for Computing Machinery.

[BMN20] Mathieu Blondel, André FT Martins, and Vlad Niculae. "Learning with Fenchel-Young losses." *J. Mach. Learn. Res.*, **21**(35):1–69, 2020.

[BMR10] Clark Barrett, Leonardo de Moura, Silvio Ranise, Aaron Stump, and Cesare Tinelli. "The SMT-LIB initiative and the rise of SMT (HVC 2010 award talk)." In *Proceedings of the 6th international conference on Hardware and software: verification and testing*, pp. 3–3. Springer-Verlag, 2010.

[BMS22] Jakub Bober, Anthea Monod, Emil Saucan, and Kevin N Webster. "Rewiring Networks for Graph Neural Network Training Using Discrete Geometry." *arXiv preprint arXiv:2207.08026*, 2022.

[BO04] Albert-Laszlo Barabasi and Zoltan N Oltvai. "Network biology: Understanding the cell's functional organization." *Nature reviews genetics*, **5**(2):101–113, 2004.

[BPB15] Vaishak Belle, Andrea Passerini, and Guy Van den Broeck. "Probabilistic inference in hybrid domains by weighted model integration." In *Proceedings of IJCAI*, pp. 2770–2776, 2015.

[BT18] Clark Barrett and Cesare Tinelli. "Satisfiability modulo theories." In *Handbook of Model Checking*, pp. 305–343. Springer, 2018.

[BTB20] Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. "Fast differentiable sorting and ranking." In *International Conference on Machine Learning*, pp. 950–959. PMLR, 2020.

[BYS22] Rickard Brüel-Gabrielsson, Mikhail Yurochkin, and Justin Solomon. "Rewiring with positional encodings for graph neural networks." *arXiv preprint arXiv:2201.12674*, 2022.

[CC96] Mary Kathryn Cowles and Bradley P Carlin. "Markov chain Monte Carlo convergence diagnostics: a comparative review." *Journal of the American Statistical Association*, **91**(434):883–904, 1996.

[CCG18] Marc-André Carbonneau, Veronika Cheplygina, Eric Granger, and Ghyslain Gagnon. "Multiple instance learning: A survey of problem characteristics and applications." *Pattern Recognition*, 2018.

[CCH19] Guangyong Chen, Pengfei Chen, Chang-Yu Hsieh, Chee-Kong Lee, Benben Liao, Renjie Liao, Weiwen Liu, Jiezhong Qiu, Qiming Sun, Jie Tang, et al. "Alchemy: A quantum chemistry dataset for benchmarking ai models." *arXiv preprint arXiv:1906.09427*, 2019.

[CCK23] Quentin Cappart, Didier Chételat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. "Combinatorial Optimization and Reasoning with Graph Neural Networks." *Journal of Machine Learning Research*, **24**(130):1–61, 2023.

[CCZ17] Yulai Cong, Bo Chen, and Mingyuan Zhou. "Fast simulation of hyperplane-truncated multivariate normal distributions." 2017.

[CD06] Arthur Choi and Adnan Darwiche. "An edge deletion semantics for belief propagation and its practical impact on approximation quality." In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, p. 1107. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[CD08] Mark Chavira and Adnan Darwiche. "On probabilistic inference by weighted model counting." 2008.

[CD10] Arthur Choi and Adnan Darwiche. "Relax, compensate and then recover." In *JSAI International Symposium on Artificial Intelligence*, pp. 167–180. Springer, 2010.

[CD12] Arthur Choi and Adnan Darwiche. "Approximating the partition function by deleting and then correcting for model edges." *arXiv preprint arXiv:1206.3241*, 2012.

[CFI92] Jin-Yi Cai, Martin Fürer, and Neil Immerman. "An optimal lower bound on the number of variables for graph identification." *Combinatorica*, **12**(4):389–410, 1992.

[CFM14] Supratik Chakraborty, Daniel J Fremont, Kuldeep S Meel, Sanjit A Seshia, and Moshe Y Vardi. "Distribution-aware sampling and weighted model counting for SAT." In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[CLT20] Xi Chen, Jason D Lee, Xin T Tong, and Yichen Zhang. "Statistical inference for model parameters in stochastic gradient descent." *The Annals of Statistics*, **48**(1):251–273, 2020.

[CNA20] Gonçalo M Correia, Vlad Niculae, Wilker Aziz, and André FT Martins. "Efficient Marginalization of Discrete and Structured Latent Variables via Sparsity." *Advances in Neural Information Processing Systems*, 2020.

[CNL11] Adam Coates, Andrew Ng, and Honglak Lee. "An analysis of single-layer networks in unsupervised feature learning." In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.

[CNM19] Gonçalo M Correia, Vlad Niculae, and André FT Martins. "Adaptively Sparse Transformers." In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2174–2184, 2019.

[COB22] Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. "Structure-aware transformer for graph representation learning." In *International Conference on Machine Learning*, pp. 3469–3489. PMLR, 2022.

[CST11a] Timothee Cour, Ben Sapp, and Ben Taskar. "Learning from partial labels." *The Journal of Machine Learning Research*, **12**:1501–1536, 2011.

[CST11b] Timothee Cour, Ben Sapp, and Ben Taskar. "Learning from Partial Labels." *J. Mach. Learn. Res.*, **12**(null):1501–1536, jul 2011.

[CSW18] Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. "Learning to Explain: An Information-Theoretic Perspective on Model Interpretation." In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 883–892. PMLR, 10–15 Jul 2018.

[CT19] Caio Corro and Ivan Titov. "Differentiable Perturb-and-Parse: Semi-Supervised Parsing with a Structured Variational Auto-Encoder." In *International Conference on Learning Representations*, 2019.

[CTV19] Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. "Differentiable ranking and

sorting using optimal transport." *Advances in neural information processing systems*, **32**, 2019.

[DAG20] Luca Di Liello, Pierfrancesco Ardino, Jacopo Gobbi, Paolo Morettin, Stefano Teso, and Andrea Passerini. "Efficient generation of structured objects with constrained adversarial networks." *Advances in neural information processing systems*, **33**:14663–14674, 2020.

[Dar01] Adnan Darwiche. "Recursive conditioning." *Artificial Intelligence*, **126**(1-2):5–41, 2001.

[Dar09] Adnan Darwiche. *Modeling and reasoning with Bayesian networks*. Cambridge university press, 2009.

[DDG99] Francesco De Comité, François Denis, Rémi Gilleron, and Fabien Letouzey. "Positive and Unlabeled Examples Help Learning." pp. 219–230, 12 1999.

[DDK12] Jesús A De Loera, Brandon Dutra, Matthias Koeppe, Stanislav Moreinis, Gregory Pinto, and Jianqiu Wu. "Software for exact integration of polynomials over polyhedra." *ACM Communications in Computer Algebra*, **45**(3/4):169–172, 2012.

[De 59] Louis De Branges. "The stone-weierstrass theorem." *Proceedings of the American Mathematical Society*, **10**(5):822–824, 1959.

[DF88] Martin E. Dyer and Alan M. Frieze. "On the complexity of computing the volume of a polyhedron." *SIAM Journal on Computing*, **17**(5):967–974, 1988.

[DG17] Dheeru Dua and Casey Graff. "UCI Machine Learning Repository.", 2017.

[DGB23] Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael M Bronstein. "On over-squashing in message passing neural networks: The impact of width, depth, and topology." In *International Conference on Machine Learning*, pp. 7865–7885. PMLR, 2023.

[DGM12] Michelangelo Diligenti, Marco Gori, Marco Maggini, and Leonardo Rigutini. "Bridging logic and kernel machines." *Machine learning*, **86**:57–88, 2012.

[DHD20] Vincent Derkinderen, Evert Heylen, Pedro Zuidberg Dos Martires, Samuel Kolb, and Luc Raedt. "Ordering variables for weighted model integration." In *Conference on Uncertainty in Artificial Intelligence*, pp. 879–888. PMLR, 2020.

[DJL20] Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. "Benchmarking graph neural networks." *arXiv preprint arXiv:2003.00982*, 2020.

[DKT07] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. "ProbLog: A probabilistic Prolog and its application in link discovery." In *IJCAI 2007, Proceedings of the 20th international joint conference on artificial intelligence*, pp. 2462–2467. IJCAI-INT JOINT CONF ARTIF INTELL, 2007.

[DLL01] Thomas Dietterich, Richard Lathrop, and Tomás Lozano-Pérez. "Solving the Multiple Instance Problem with Axis-Parallel Rectangles." *Artificial Intelligence*, **89**:31–71, 03 2001.

[DLV22] Andreea Deac, Marc Lackenby, and Petar Veličković. "Expander graph propagation." In *Learning on Graphs Conference*, pp. 38–1. PMLR, 2022.

[DM02] Adnan Darwiche and Pierre Marquis. "A knowledge compilation map." *Journal of Artificial Intelligence Research*, **17**:229–264, 2002.

[DM07] Rina Dechter and Robert Mateescu. "AND/OR search spaces for graphical models." *Artificial intelligence*, **171**(2-3):73–106, 2007.

[Dom10] Justin Domke. "Implicit Differentiation by Perturbation." In *Advances in Neural Information Processing Systems 23*, pp. 523–531. 2010.

[DRG22] Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. "Long range graph benchmark." *Advances in Neural Information Processing Systems*, **35**:22326–22340, 2022.

[EK12] David Easley, Jon Kleinberg, et al. "Networks, crowds, and markets." *Cambridge Books*, 2012.

[FBD19] Marc Fischer, Mislav Balunovic, Dana Drachsler-Cohen, Timon Gehr, Ce Zhang, and Martin Vechev. "DL2: training and querying neural networks with logic." In *International Conference on Machine Learning*, pp. 1931–1941. PMLR, 2019.

[FBL18] Tiago M Fragoso, Wesley Bertoli, and Francisco Louzada. "Bayesian model averaging: A systematic review and conceptual classification." *International Statistical Review*, **86**(1):1–28, 2018.

[FNC20] Thomas Frerix, Matthias Nießner, and Daniel Cremers. "Homogeneous linear inequality constraints for neural network activations." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 748–749, 2020.

[FNP19] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. "Learning discrete structures for graph neural networks." In *International Conference on Machine Learning*, pp. 1972–1982. PMLR, 2019.

[FRE20] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. "Sign: Scalable inception graph neural networks." *arXiv preprint arXiv:2004.11198*, 2020.

[FV13] Benoît Frénay and Michel Verleysen. "Classification in the presence of label noise: a survey." *IEEE transactions on neural networks and learning systems*, **25**(5):845–869, 2013.

[GBT23] Lorenzo Giusti, Claudio Battiloro, Lucia Testa, Paolo Di Lorenzo, Stefania Sardellitti, and Sergio Barbarossa. "Cell attention networks." In *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2023.

[GCW18] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. "Backpropagation through the void: Optimizing control variates for black-box gradient estimation." *ICLR*, 2018.

[GDB23] Benjamin Gutteridge, Xiaowen Dong, Michael M Bronstein, and Francesco Di Giovanni. "DRew: dynamically rewired message passing with delay." In *International Conference on Machine Learning*, pp. 12252–12267. PMLR, 2023.

[GG15] Yarin Gal and Zoubin Ghahramani. "Bayesian convolutional neural networks with Bernoulli approximate variational inference." *arXiv preprint arXiv:1506.02158*, 2015.

[GG16] Yarin Gal and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

[GJ02] Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.

[GL21] Eleonora Giunchiglia and Thomas Lukasiewicz. "Multi-label classification neural networks with hard logical constraints." *Journal of Artificial Intelligence Research*, **72**:759–818, 2021.

[GL23] Artur d'Avila Garcez and Luis C Lamb. "Neurosymbolic AI: The 3 rd wave." *Artificial Intelligence Review*, **56**(11):12387–12406, 2023.

[GPM14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[GRC23] Lorenzo Giusti, Teodora Reu, Francesco Ceccarelli, Cristian Bodnar, and Pietro Liò. "CIN++: Enhancing Topological Message Passing." *arXiv preprint arXiv:2306.03561*, 2023.

[Gro17] Martin Grohe. *Descriptive complexity, canonisation, and definable graph structure theory*, volume 47. Cambridge University Press, 2017.

[GSR17] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. "Neural message passing for quantum chemistry." In *International Conference on Machine Learning*, pp. 1263–1272. PMLR, 2017.

[GTH15] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M Suchanek. "Fast rule mining in ontological knowledge bases with AMIE++." *The VLDB Journal*, **24**(6):707–730, 2015.

[GWG19] Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. "Diffusion improves graph learning." *Advances in Neural Information Processing Systems*, **32**, 2019.

[GWS21] Saurabh Garg, Yifan Wu, Alexander J Smola, Sivaraman Balakrishnan, and Zachary Lipton. "Mixture proportion estimation and pu learning: a modern approach." *Advances in Neural Information Processing Systems*, **34**:8532–8544, 2021.

[GWZ18] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. "Stochastic Optimization of Sorting Networks via Continuous Relaxations." In *International Conference on Learning Representations*, 2018.

[HAB19] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. "Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 41–50, 2019.

[HBM18] Steven Holtzen, Guy Van den Broeck, and Todd Millstein. "Sound Abstraction and Decomposition of Probabilistic Programs." In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

[HCW20] Pim de Haan, Taco S. Cohen, and Max Welling. "Natural graph networks." *Advances in Neural Information Processing Systems*, **33**:3636–3646, 2020.

[HFZ20] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. "Open graph benchmark: Datasets for machine learning on graphs." *Advances in Neural Information Processing Systems*, **33**:22118–22133, 2020.

[HG95] David Heckerman and Dan Geiger. "Learning Bayesian networks: a unification for discrete and Gaussian domains." In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp. 274–284. Morgan Kaufmann Publishers Inc., 1995.

[HGM18] Jiawei He, Yu Gong, Joseph Marino, Greg Mori, and Andreas Lehrmann. "Variational autoencoders with jointly optimized latent dependency structure." In *International conference on learning representations*, 2018.

[HHL23] Xiaoxin He, Bryan Hooi, Thomas Laurent, Adam Perold, Yann LeCun, and Xavier Bresson. "A generalization of vit/mlp-mixer to graphs." In *International Conference on Machine Learning*, pp. 12724–12745. PMLR, 2023.

[HKW17] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. "Safety verification of deep neural networks." In *International conference on computer aided verification*, pp. 3–29. Springer, 2017.

[HLG20] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. "Strategies for Pre-training Graph Neural Networks." In *International Conference on Learning Representations*, 2020.

[HMB17] Steven Holtzen, Todd Millstein, and Guy Van den Broeck. "Probabilistic Program Abstractions." In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.

[Hul14] Eyke Hüllermeier. "Learning from imprecise and fuzzy observations: Data disambiguation through generalized loss minimization." *International Journal of Approximate Reasoning*, **55**(7):1519–1534, 2014.

[HYL17] William L. Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs." In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.

[IMK20] Pavel Izmailov, Wesley J Maddox, Polina Kirichenko, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. "Subspace inference for Bayesian deep learning." In *Uncertainty in Artificial Intelligence*, pp. 1169–1179. PMLR, 2020.

[IPG18] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. "Averaging weights leads to wider optima and better generalization." *arXiv preprint arXiv:1803.05407*, 2018.

[ITW18] Maximilian Ilse, Jakub Tomczak, and Max Welling. "Attention-based deep multiple instance learning." In *International conference on machine learning*, pp. 2127–2136. PMLR, 2018.

[IVH21] Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Gordon Wilson. "What are Bayesian neural network posteriors really like?" In *International Conference on Machine Learning*, pp. 4629–4640. PMLR, 2021.

[JCB17] Wengong Jin, Connor Coley, Regina Barzilay, and Tommi Jaakkola. "Predicting organic reaction outcomes with weisfeiler-lehman network." *Advances in Neural Information Processing Systems*, **30**, 2017.

[JEP21] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. "Highly accurate protein structure prediction with AlphaFold." *Nature*, **596**(7873):583–589, 2021.

[JGP16] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical reparameterization with gumbel-softmax." *arXiv preprint arXiv:1611.01144*, 2016.

[JGP17a] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical Reparameterization with Gumbel-Softmax." In *International Conference on Learning Representations*, 2017.

[JGP17b] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical Reparameterization with Gumbel-Softmax." In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[JLB22] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. "Hands-on Bayesian neural networks—A tutorial for deep learning users." *IEEE Computational Intelligence Magazine*, **17**(2):29–48, 2022.

[KB15] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In *International Conference on Learning Representations*, 2015.

[KBD17] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. "Reluplex: An efficient SMT solver for verifying deep neural networks." In *International conference on computer aided verification*, pp. 97–117. Springer, 2017.

[KBM22] Kedar Karhadkar, Pradeep Kr Banerjee, and Guido Montúfar. "FoSR: First-order spectral rewiring for addressing oversquashing in GNNs." *arXiv preprint arXiv:2210.11790*, 2022.

[KEH22] Agustinus Kristiadi, Runa Eschenhagen, and Philipp Hennig. "Posterior Refinement Improves Sample Efficiency in Bayesian Neural Networks." In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pp. 30333–30346. Curran Associates, Inc., 2022.

[KF09] Daphne Koller and Nir Friedman. "Probabilistic Graphical Models." 2009.

[KFL01] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. "Factor graphs and the sum-product algorithm." *IEEE Transactions on information theory*, **47**(2):498–519, 2001.

[KGG20] Johannes Klicpera, Janek Groß, and Stephan Günnemann. "Directional Message Passing for Molecular Graphs." In *International Conference on Learning Representations*, 2020.

[KH09a] Alex Krizhevsky and Geoffrey Hinton. "Learning multiple layers of features from tiny images." Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

[KH09b] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images." 2009.

[KHH20] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. "Being bayesian, even just a bit, fixes overconfidence in relu networks." In *International conference on machine learning*, pp. 5436–5446. PMLR, 2020.

[KMS18] Samuel Kolb, Martin Mladenov, Scott Sanner, Vaishak Belle, and Kristian Kersting. "Efficient Symbolic Integration for Probabilistic Inference." In *IJCAI*, pp. 5031–5037, 2018.

[KMZ19] Samuel Kolb, Paolo Morettin, Pedro Zuidberg Dos Martires, Francesco Sommavilla, Andrea Passerini, Roberto Sebastiani, and Luc De Raedt. "The pywmi Framework and Toolbox for Probabilistic Inference using Weighted Model Integration." In *Proceedings*

of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI, pp. 6530–6532, 7 2019.

[KMZ21] Giannis Karamanolakis, Subhabrata Mukherjee, Guoqing Zheng, and Ahmed Hassan Awadallah. "Self-training with weak supervision." *arXiv preprint arXiv:2104.05514*, 2021.

[KND17] Ryuichi Kiryo, Gang Niu, Marthinus C Du Plessis, and Masashi Sugiyama. "Positive-unlabeled learning with non-negative risk estimator." *Advances in neural information processing systems*, **30**, 2017.

[KSE16] Carolyn Kim, Ashish Sabharwal, and Stefano Ermon. "Exact sampling with integer linear programs and random perturbations." In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[KSF22] Alan A. Kaptanoglu, Brian M. de Silva, Urban Fasel, Kadierdan Kaheman, Andy J. Goldschmidt, Jared Callaham, Charles B. Delahunt, Zachary G. Nicolaou, Kathleen Champion, Jean-Christophe Loiseau, J. Nathan Kutz, and Steven L. Brunton. "PySINDy: A comprehensive Python package for robust sparse system identification." *Journal of Open Source Software*, **7**(69):3994, 2022.

[KVW19] Wouter Kool, Herke Van Hoof, and Max Welling. "Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement." In *International Conference on Machine Learning*, pp. 3499–3508. PMLR, 2019.

[KW13] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes." *arXiv preprint arXiv:1312.6114*, 2013.

[KW17] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks." In *International Conference on Learning Representations*, 2017.

[KW22] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes.", 2022.

[LDG00] Fabien Letouzey, François Denis, and Rémi Gilleron. "Learning From Positive and Unlabeled examples." In *Proceedings of the 11th International Conference on Algorithmic Learning Theory, ALT'00*, pp. 71–85, Sydney, Australia, 2000. Springer Verlag.

[LeC98] Yann LeCun. "The MNIST database of handwritten digits." 1998.

[Lou20] Andreas Loukas. "What graph neural networks cannot learn: depth vs width." In *International Conference on Learning Representations*, 2020.

[LQL23] Bo Li, Peng Qi, Bo Liu, Shuai Di, Jingen Liu, Jiquan Pei, Jinfeng Yi, and Bowen Zhou. "Trustworthy ai: From principles to practices." *ACM Computing Surveys*, **55**(9):1–46, 2023.

[LSB24] Ruoyan Li, Dipti Ranjan Sahu, Guy Van den Broeck, and Zhe Zeng. "Linear-Equality Constrained Deep Generative Models." *(submitted)*, 2024.

[LW89] Steffen Lilholt Lauritzen and Nanny Wermuth. "Graphical models for associations between variables, some of which are qualitative and some quantitative." *The annals of Statistics*, pp. 31–57, 1989.

[LWJ21] Meng Liu, Zhengyang Wang, and Shuiwang Ji. "Non-local graph neural networks." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **44**(12):10270–10276, 2021.

[LWZ23] Zhiyuan Liu, Dafei Wu, Weiwei Zhai, and Liang Ma. "SONAR enables cell type deconvolution with spatially weighted Poisson-Gamma model for spatial transcriptomics." *Nature Communications*, **14**(1):4727, 2023.

[LZS23] Kangning Liu, Weicheng Zhu, Yiqiu Shen, Sheng Liu, Narges Razavian, Krzysztof J Geras, and Carlos Fernandez-Granda. "Multiple instance learning via iterative self-paced supervised contrastive learning." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3355–3365, 2023.

[LZW11] Huchuan Lu, Qiuhong Zhou, Dong Wang, and Ruan Xiang. "A co-training framework for visual tracking with multiple instance learning." In *2011 IEEE International Conference on Automatic Face & Gesture Recognition (FG)*, pp. 539–544, 2011.

[MA16] Andre Martins and Ramon Astudillo. "From softmax to sparsemax: A sparse model of attention and multi-label classification." In *International conference on machine learning*, pp. 1614–1623. PMLR, 2016.

[Maa09] Laurens van der Maaten. "Learning a Parametric Embedding by Preserving Local Structure." In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pp. 384–391, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

[Mac92] David JC MacKay. "The evidence framework applied to classification networks." *Neural computation*, **4**(5):720–736, 1992.

[MAD17] David Merrell, Aws Albarghouthi, and Loris D'Antoni. "Weighted Model Integration with Orthogonal Transformations." *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017.

[MB18] Arthur Mensch and Mathieu Blondel. "Differentiable dynamic programming for structured prediction and attention." In *International Conference on Machine Learning*, pp. 3462–3471. PMLR, 2018.

[MBR22] Hassan Maatouk, Xavier Bay, and Didier Rullière. "A note on simulating hyperplane-truncated multivariate normal distributions." *Statistics & Probability Letters*, **191**:109650, 2022.

[MBS19a] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. "Provably Powerful Graph Networks." In *Advances in Neural Information Processing Systems*, pp. 2153–2164, 2019.

[MBS19b] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. "Invariant and Equivariant Graph Networks." In *International Conference on Learning Representations*, 2019.

[MCZ18] Tom Minka, Ryan Cleven, and Yordan Zaykov. "Trueskill 2: An improved bayesian skill rating system." 2018.

[MDK18] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. "DeepProbLog: Neural Probabilistic Logic Programming." In *NeurIPS*, 2018.

[MDM24] Giuseppe Marra, Sebastijan Dumančić, Robin Manhaeve, and Luc De Raedt. "From statistical relational to neurosymbolic artificial intelligence: A survey." *Artificial Intelligence*, p. 104062, 2024.

[MFN23] Pasquale Minervini, Luca Franceschi, and Mathias Niepert. "Adaptive Perturbation-Based Gradient Estimation for Discrete Latent Variable Models." In *AAAI*, 2023.

[MGM23] Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampášek. "Attending to graph transformers." *arXiv preprint arXiv:2302.04181*, 2023.

[MHB17] Stephan Mandt, Matthew D Hoffman, and David M Blei. "Stochastic gradient descent as approximate bayesian inference." *arXiv preprint arXiv:1704.04289*, 2017.

[MIG19] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. "A simple baseline for bayesian uncertainty in deep learning." *Advances in Neural Information Processing Systems*, **32**:13153–13164, 2019.

[MKB20] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. "Tudataset: A collection of benchmark datasets for learning with graphs." *arXiv preprint arXiv:2007.08663*, 2020.

[MKT20] Paolo Morettin, Samuel Kolb, Stefano Teso, and Andrea Passerini. "Learning Weighted Model Integration Distributions." In *AAAI*, pp. 5224–5231, 2020.

[ML97] Oded Maron and Tomás Lozano-Pérez. "A Framework for Multiple-Instance Learning." In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1997.

[MLJ12] Julian McAuley, J. Leskovec, and Dan Jurafsky. "Learning Attitudes and Attributes from Multi-aspect Reviews." *2012 IEEE 12th International Conference on Data Mining*, pp. 1020–1025, 2012.

[MLM21] Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M Kriege, Martin Grohe, Matthias Fey, and Karsten Borgwardt. "Weisfeiler and leman go machine learning: The story so far." *arXiv preprint arXiv:2112.09992*, 2021.

[MMS22] Eleonora Misino, Giuseppe Marra, and Emanuele Sansone. "Vael: Bridging variational autoencoders and probabilistic logic programming." *Advances in Neural Information Processing Systems*, **35**:4667–4679, 2022.

[MMT16] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. "The concrete distribution: A continuous relaxation of discrete random variables." *arXiv preprint arXiv:1611.00712*, 2016.

[MMT17a] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables." In *International Conference on Learning Representations*, 2017.

[MMT17b] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables." In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[MPS17] Paolo Morettin, Andrea Passerini, and Roberto Sebastiani. "Efficient weighted model integration via SMT-based predicate abstraction." In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 720–728. AAAI Press, 2017.

[MPS19] Paolo Morettin, Andrea Passerini, and Roberto Sebastiani. "Advanced SMT techniques for weighted model integration." *Artificial Intelligence*, **275**:1–27, 2019.

[MRF19] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. "Weisfeiler and leman go neural: Higher-order graph neural networks." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019.

[MSR19] Ryan Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. "Relational pooling for graph representations." In *International Conference on Machine Learning*, pp. 4663–4673. PMLR, 2019.

[MTM14] Chris J Maddison, Daniel Tarlow, and Tom Minka. "A∗ Sampling." In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[MTP23] Lassi Meronen, Martin Trapp, Andrea Pilzer, Le Yang, and Arno Solin. "Fixing Overconfidence in Dynamic Neural Networks." *arXiv preprint arXiv:2302.06359*, 2023.

[MTS21] Lassi Meronen, Martin Trapp, and Arno Solin. "Periodic activation functions induce stationarity." *Advances in Neural Information Processing Systems*, **34**:1673–1685, 2021.

[MVD18] Alejandro Molina, Antonio Vergari, Nicola Di Mauro, Sriraam Natarajan, Floriana Esposito, and Kristian Kersting. "Mixed sum-product networks: A deep architecture for hybrid domains." In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[NCH15] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. "Obtaining well

calibrated probabilities using bayesian binning." In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[NDR13] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. "Learning with Noisy Labels." In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[NGB16] Marion Neumann, Roman Garnett, Christian Bauckhage, and Kristian Kersting. "Propagation kernels: efficient graph kernels from propagated information." *Machine learning*, **102**:209–245, 2016.

[Nil82] Nils J Nilsson. "Principles of Artificial Intelligence." *Symbolic Computation, Berlin: Springer, 1982*, 1982.

[NK00] Masatoshi Nei and Sudhir Kumar. *Molecular evolution and phylogenetics*. Oxford university press, 2000.

[NM20] Vlad Niculae and André F. T. Martins. "LP-SparseMAP: Differentiable Relaxed Optimization for Sparse Structured Prediction." In *ICML*, 2020.

[NMB18] Vlad Niculae, André F. T. Martins, Mathieu Blondel, and Claire Cardie. "SparseMAP: Differentiable Sparse Structured Inference." In *ICML*, 2018.

[NMF21a] Mathias Niepert, Pasquale Minervini, and Luca Franceschi. "Implicit MLE: backpropagating through discrete exponential family distributions." *Advances in Neural Information Processing Systems*, **34**:14567–14579, 2021.

[NMF21b] Mathias Niepert, Pasquale Minervini, and Luca Franceschi. "Implicit mle: Backpropagating through discrete exponential family distributions." *Advances in Neural Information Processing Systems*, **34**, 2021.

[NNN21] Son Nguyen, Duong Nguyen, Khai Nguyen, Khoat Than, Hung Bui, and Nhat Ho. "Structured Dropout Variational Inference for Bayesian Neural Networks." *Advances in Neural Information Processing Systems*, **34**, 2021.

[NYC15] Anh Nguyen, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436, 2015.

[PBK21] Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. "Learning with Algorithmic Supervision via Continuous Relaxations." *Advances in Neural Information Processing Systems*, **34**, 2021.

[PBT06] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. "Learning accurate, compact, and interpretable tree annotation." In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp. 433–440, 2006.

[PCT20] Max B Paulus, Dami Choi, Daniel Tarlow, Andreas Krause, and Chris J Maddison. "Gradient Estimation with Stochastic Softmax Tricks." *arXiv preprint arXiv:2006.08063*, 2020.

[Pea88] Judea Pearl. "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.", 1988.

[PMF21] Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. "DropGNN: Random dropouts increase the expressiveness of graph neural networks." *Advances in Neural Information Processing Systems*, **34**:21997–22009, 2021.

[PNM19] Ben Peters, Vlad Niculae, and André FT Martins. "Sparse Sequence-to-Sequence Models." In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1504–1519, 2019.

[PNS14] Marthinus C du Plessis, Gang Niu, and Masashi Sugiyama. "Analysis of Learning from Positive and Unlabeled Data." In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[PNS15] Marthinus Du Plessis, Gang Niu, and Masashi Sugiyama. "Convex Formulation for Learning from Positive and Unlabeled Data." In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, 2015.

[PR18] Tobias Plötz and Stefan Roth. "Neural Nearest Neighbors Networks." In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 1095–1106, 2018.

[PWC20] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. "Geom-GCN: Geometric Graph Convolutional Networks." In *International Conference on Learning Representations*, 2020.

[QCC] Gwenolé Quellec, Guy Cazuguel, Béatrice Cochener, and Mathieu Lamard. "Multiple-instance learning for medical image and video analysis." *IEEE reviews in biomedical engineering*.

[QMA24] Chendi Qian, Andrei Manolache, Kareem Ahmed, Zhe Zeng, Guy Van den Broeck, Mathias Niepert, and Christopher Morris. "Probabilistically Rewired Message-Passing Neural Networks." In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*, 2024.

[QSC08] Novi Quadrianto, Alex Smola, Tibério Caetano, and Quoc Le. "Estimating labels from label proportions." 2008.

[RBB18] Hippolyt Ritter, Aleksandar Botev, and David Barber. "A scalable laplace approximation for neural networks." In *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*, volume 6. International Conference on Representation Learning, 2018.

[RGD22] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. "Recipe for a general, powerful, scalable graph transformer." *Advances in Neural Information Processing Systems*, **35**:14501–14515, 2022.

[RHX20] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. "DropEdge: Towards Deep Graph Convolutional Networks on Node Classification." In *International Conference on Learning Representations*, 2020.

[RMP20] Michal Rolinek, Vít Musil, Anselm Paulus, Marin Vlastelica, Claudio Michaelis, and Georg Martius. "Optimizing rank-based metrics with blackbox differentiation." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7620–7630, 2020.

[RNE22] Patrick Reiser, Marlen Neubert, André Eberhard, Luca Torresi, Chen Zhou, Chen Shao, Houssam Metni, Clint van Hoesel, Henrik Schopmans, Timo Sommer, et al. "Graph neural networks for materials science and chemistry." *Communications Materials*, **3**(1):93, 2022.

[Rol17] Jason Tyler Rolfe. "Discrete Variational Autoencoders." In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[Rot96a] Dan Roth. "On the Hardness of Approximate Reasoning." *Artif. Intell.*, **82**(1-2):273–302, 1996.

[Rot96b] Dan Roth. "On the hardness of approximate reasoning." *Artificial Intelligence*, **82**(1–2):273–302, 1996.

[RS86] Neil Robertson and P.D Seymour. "Graph minors. II. Algorithmic aspects of tree-width." *Journal of Algorithms*, **7**(3):309 – 322, 1986.

[RSS20] Ali Raza, Arni Sturluson, Cory M Simon, and Xiaoli Fern. "Message passing neural networks for partial charge assignment to metal–organic frameworks." *The Journal of Physical Chemistry C*, **124**(35):19070–19082, 2020.

[RSZ20] Michal Rolínek, Paul Swoboda, Dominik Zietlow, Anselm Paulus, Vít Musil, and Georg Martius. "Deep Graph Matching via Blackbox Differentiation of Combinatorial Solvers." In *ECCV*, 2020.

[RTS18] Carlos Riquelme, George Tucker, and Jasper Snoek. "Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling." *arXiv preprint arXiv:1802.09127*, 2018.

[SA12] Scott Sanner and Ehsan Abbasnejad. "Symbolic variable elimination for discrete and continuous graphical models." In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[SBK05] Tian Sang, Paul Beame, and Henry A Kautz. "Performing Bayesian inference by weighted model counting." In *AAAI*, volume 5, pp. 475–481, 2005.

[SBK20] Sergey Shirobokov, Vladislav Belavin, Michael Kagan, Andrei Ustyuzhanin, and Atilim Gunes Baydin. "Black-box optimization with local generative surrogates." *Advances in Neural Information Processing Systems*, **33**:14650–14662, 2020.

[SCB18] Hugh Salimbeni, Ching-An Cheng, Byron Boots, and Marc Deisenroth. "Orthogonally decoupled variational Gaussian processes." *Advances in neural information processing systems*, **31**, 2018.

[SCQ20] Brian de Silva, Kathleen Champion, Markus Quade, Jean-Christophe Loiseau, J. Kutz, and Steven Brunton. "PySINDy: A Python package for the sparse identification of

nonlinear dynamical systems from data." *Journal of Open Source Software*, **5**(49):2104, 2020.

[SDC24] Mihaela Cătălina Stoian, Salijona Dyrmishi, Maxime Cordy, Thomas Lukasiewicz, and Eleonora Giunchiglia. "How Realistic Is Your Synthetic Data? Constraining Deep Generative Models for Tabular Data." *arXiv preprint arXiv:2402.04823*, 2024.

[SDD12] Scott Sanner, Karina Valdivia Delgado, and Leliane Nunes De Barros. "Symbolic dynamic programming for discrete and continuous state MDPs." *arXiv preprint arXiv:1202.3762*, 2012.

[SFM20] Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein, and Guy Van den Broeck. "Counterexample-Guided Learning of Monotonic Neural Networks." In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, dec 2020.

[SGL24] Mihaela Cătălina Stoian, Eleonora Giunchiglia, and Thomas Lukasiewicz. "Exploiting t-norms for deep learning in autonomous driving." *arXiv preprint arXiv:2402.11362*, 2024.

[SGT08a] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. "Computational capabilities of graph neural networks." *IEEE Transactions on Neural Networks*, **20**(1):81–102, 2008.

[SGT08b] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. "The graph neural network model." *IEEE Transactions on Neural Networks*, **20**(1):61–80, 2008.

[SKP22] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. "Learning from noisy labels with deep neural networks: A survey." *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[SOG16] Rodrigo de Salvo Braz, Ciaran O'Reilly, Vibhav Gogate, and Rina Dechter. "Probabilistic inference modulo theories." In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pp. 3591–3599. AAAI Press, 2016.

[SRM16] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. "Ladder Variational Autoencoders.", 2016.

[SRS15] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. "Scalable bayesian optimization using deep neural networks." In *International conference on machine learning*, pp. 2171–2180. PMLR, 2015.

[SRT16] Korsuk Sirinukunwattana, Shan e Ahmed Raza, Yee Tsang, David Snead, Ian Cree, and Nasir Rajpoot. "Locality Sensitive Deep Learning for Detection and Classification of Nuclei in Routine Colon Cancer Histology Images." *IEEE Transactions on Medical Imaging*, **35**:1–1, 02 2016.

[SS10] Marko Samer and Stefan Szeider. "Algorithms for propositional model counting." *Journal of Discrete Algorithms*, **8**(1):50–64, 2010.

[SSV11] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, **12**(9), 2011.

[SVP09] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. "Efficient graphlet kernels for large graph comparison." In *Artificial Intelligence and Statistics*, pp. 488–495. PMLR, 2009.

[SVV23] Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J Sutherland, and Ali Kemal Sinop. "Exphormer: Sparse transformers for graphs." *arXiv preprint arXiv:2303.06147*, 2023.

[SW11] Prakash P Shenoy and James C West. "Inference in hybrid Bayesian networks using mixtures of polynomials." *International Journal of Approximate Reasoning*, **52**(5):641–657, 2011.

[SZ20] Clayton Scott and Jianxin Zhang. "Learning from Label Proportions: A Mutual Contamination Framework." In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pp. 22256–22267. Curran Associates, Inc., 2020.

[SZA23] Vinay Shukla, Zhe Zeng, Kareem Ahmed, and Guy Van den Broeck. "A Unified Approach to Count-Based Weakly-Supervised Learning." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[TDC21] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. "Understanding over-squashing and bottlenecks on graphs via curvature." *arXiv preprint arXiv:2111.14522*, 2021.

[TL20] Kuen-Han Tsai and Hsuan-Tien Lin. "Learning from label proportions with consistency regularization." In *Asian Conference on Machine Learning*, pp. 513–528. PMLR, 2020.

[TMM17] George Tucker, Andriy Mnih, Chris J Maddison, Dieterich Lawson, and Jascha Sohl-Dickstein. "Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models." *Advances in Neural Information Processing Systems*, 2017.

[TWL24] Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. "Solving olympiad geometry without human demonstrations." *Nature*, **625**(7995):476–482, 2024.

[Val79] Leslie G Valiant. "The complexity of enumeration and reliability problems." *SIAM Journal on Computing*, **8**(3):410–421, 1979.

[VCC18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. "Graph Attention Networks." In *International Conference on Learning Representations*, 2018.

[VGR81] William E Vesely, Francine F Goldberg, Norman H Roberts, and David F Haasl. "Fault tree handbook." Technical report, Nuclear Regulatory Commission Washington DC, 1981.

[VMP19] Antonio Vergari, Alejandro Molina, Robert Peharz, Zoubin Ghahramani, Kristian Kersting, and Isabel Valera. "Automatic Bayesian density analysis." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5207–5215, 2019.

[Was00] Larry Wasserman. "Bayesian model selection and model averaging." *Journal of mathematical psychology*, **44**(1):92–107, 2000.

[WHS16] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. "Deep kernel learning." In *Artificial intelligence and statistics*, pp. 370–378. PMLR, 2016.

[WIB11] Janusz Wojtusiak, Katherine Irvin, Aybike Birerdinc, and Ancha V Baranova. "Using published medical results and non-homogenous data in rule learning." In *2011 10th International Conference on Machine Learning and Applications and Workshops*, volume 2, pp. 84–89. IEEE, 2011.

[Wil92] Ronald J. Williams. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning." *Mach. Learn.*, **8**:229–256, 1992.

[WJW21] Zhanghao Wu, Paras Jain, Matthew Wright, Azalia Mirhoseini, Joseph E Gonzalez, and Ion Stoica. "Representing long-range context for graph neural networks with global attention." *Advances in Neural Information Processing Systems*, **34**:13266–13279, 2021.

[WL68] Boris Weisfeiler and Andrei Leman. "The reduction of a graph to canonical form and the algebra which appears therein." *nti, Series*, **2**(9):12–16, 1968.

[WNM19] Anqi Wu, Sebastian Nowozin, Edward Meeds, Richard E. Turner, Jose Miguel Hernandez-Lobato, and Alexander L. Gaunt. "Deterministic Variational Inference for Robust Bayesian Neural Networks." In *International Conference on Learning Representations*, 2019.

[WS98] Duncan J Watts and Steven H Strogatz. "Collective dynamics of 'small-world'networks." *nature*, **393**(6684):440, 1998.

[WT11] Max Welling and Yee W Teh. "Bayesian learning via stochastic gradient Langevin dynamics." In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

[WYT18] Xinggang Wang, Yongluan Yan, Peng Tang, Xiang Bai, and Wenyu Liu. "Revisiting multiple instance neural networks." *Pattern Recognition*, **74**:15–24, 2018.

[XDC20] Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. "Differentiable top-k with optimal transport." *Advances in Neural Information Processing Systems*, **33**:20520–20531, 2020.

[XE19a] Sang Michael Xie and Stefano Ermon. "Reparameterizable Subset Sampling via Continuous Relaxations." *International Joint Conference on Artificial Intelligence*, 2019.

[XE19b] Sang Michael Xie and Stefano Ermon. "Reparameterizable Subset Sampling via Continuous Relaxations." In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 3919–3925. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[XE19c] Sang Michael Xie and Stefano Ermon. "Reparameterizable subset sampling via continuous relaxations." *arXiv preprint arXiv:1901.10517*, 2019.

[XH18] Ming-Kun Xie and Sheng-Jun Huang. "Partial multi-label learning." In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[XHL19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How Powerful are Graph Neural Networks?" In *International Conference on Learning Representations*, 2019.

[XLT18] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. "Representation learning on graphs with jumping knowledge networks." In *International Conference on Machine Learning*, pp. 5453–5462. PMLR, 2018.

[XTX13] Chang Xu, Dacheng Tao, and Chao Xu. "A survey on multi-view learning." *arXiv preprint arXiv:1304.5634*, 2013.

[XZF18] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. "A semantic loss function for deep learning with symbolic knowledge." In *International conference on machine learning*, pp. 5502–5511. PMLR, 2018.

[Yar95] David Yarowsky. "Unsupervised Word Sense Disambiguation Rivaling Supervised Methods." In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, ACL '95, p. 189–196, USA, 1995. Association for Computational Linguistics.

[YBR14] Eunho Yang, Yulia Baker, Pradeep Ravikumar, Genevera Allen, and Zhandong Liu. "Mixed graphical models via exponential families." In *Artificial Intelligence and Statistics*, pp. 1042–1050, 2014.

[YCK14] Felix X Yu, Krzysztof Choromanski, Sanjiv Kumar, Tony Jebara, and Shih-Fu Chang. "On learning from label proportions." *arXiv preprint arXiv:1402.5902*, 2014.

[YWS15] Zichao Yang, Andrew Wilson, Alex Smola, and Le Song. "A la carte–learning fast kernels." In *Artificial Intelligence and Statistics*, pp. 1098–1106. PMLR, 2015.

[ZB19] Zhe Zeng and Guy Van den Broeck. "Efficient Search-Based Weighted Model Integration." *Proceedings of UAI*, 2019.

[ZB23a] Zhe Zeng and Guy Van den Broeck. "Collapsed Inference for Bayesian Deep Learning." *arXiv preprint arXiv:2306.09686*, 2023.

[ZB23b] Zhe Zeng and Guy Van den Broeck. "Collapsed Inference for Bayesian Deep Learning." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[ZCN18] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. "An end-to-end deep learning architecture for graph classification." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[ZD18] Kaja Zupanc and Jesse Davis. "Estimating rule quality for knowledge base completion with the relationship between coverage assumption." In *Proceedings of the 2018 World Wide Web Conference*, pp. 1073–1081, 2018.

[ZDD19] Pedro Miguel Zuidberg Dos Martires, Anton Dries, and Luc De Raedt. "Exact and Approximate Weighted Model Integration withProbability Density Functions Using Knowledge Compilation." In *Proceedings of the 30th Conference on Artificial Intelligence*. AAAI Press, 2019.

[ZG22] Xiaojin Zhu and Andrew B Goldberg. *Introduction to semi-supervised learning*. Springer Nature, 2022.

[Zho18] Zhi-Hua Zhou. "A brief introduction to weakly supervised learning." *National science review*, **5**(1):44–53, 2018.

[Zhu05] Xiaojin Jerry Zhu. "Semi-supervised learning literature survey." 2005.

[ZKD19] Pedro Miguel Zuidberg Dos Martires, Samuel Kolb, and Luc De Raedt. "How to Exploit Structure while Solving Weighted Model Integration Problems." *UAI 2019 Proceedings*, 2019.

[ZMY20a] Zhe Zeng, Paolo Morettin, Fanqi Yan, Antonio Vergari, and Guy Van den Broeck. "Scaling up Hybrid Probabilistic Inference with Logical and Arithmetic Constraints via Message Passing." In *Proceedings of the International Conference of Machine Learning (ICML)*, 2020.

[ZMY20b] Zhe Zeng, Paolo Morettin, Fanqi Yan, Antonio Vergari, and Guy Van den Broeck. "Probabilistic inference with algebraic constraints: Theoretical limits and practical approximations." *Advances in Neural Information Processing Systems*, **33**:11564–11575, 2020.

[ZMY20c] Zhe Zeng, Paolo Morettin, Fanqi Yan, Antonio Vergari, and Guy Van den Broeck. "Relax, compensate and then integrate." In *Proceedings of the ECML-PKDD Workshop on Deep Continuous-Discrete Machine Learning (DeCoDeML)*, 2020.

[ZMY21a] Zhe Zeng, Paolo Morettin, Fanqi Yan, Andrea Passerini, Guy Van den Broeck, et al. "Is Parameter Learning via Weighted Model Integration Tractable?" In *The 4th Workshop on Tractable Probabilistic Modeling*, 2021.

[ZMY21b] Zhe Zeng, Paolo Morettin, Fanqi Yan, Antonio Vergari, and Guy Van den Broeck. "Is Parameter Learning via Weighted Model Integration Tractable?" In *Proceedings of the UAI Workshop on Tractable Probabilistic Modeling (TPM)*, jul 2021.

[ZSF12] Zahra Zamani, Scott Sanner, and Cheng Fang. "Symbolic dynamic programming for continuous state and action mdps." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pp. 1839–1845, 2012.

[Zui20] Pedro Miguel Zuidberg Dos Martires et al. "Monte Carlo Anti-Differentiation for Approximate Weighted Model Integration." In *Proceedings of the Ninth International*

Workshop on Statistical Relational AI. International Workshop on Statistical Relational AI, 2020.