Method of Moments in Approximate Bayesian Inference: From Theory to Practice

by

Haonan Duan

A thesis

presented to the University of Waterloo

in fulfillment of the

thesis requirement for the degree of

Master of Mathematics

in

Computer Science

Waterloo, Ontario, Canada, 2021

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Chapter 2 and Chapter 3 are based on an ICML 2020 paper (link) where I am the first authors with Saeed Nejati. I derived and wrote all the contents in Chapter 2 by myself. George Trimponias helped review the proof. I formulated the core idea and independently wrote all of the materials from Section 3.1 to Section 3.5 in Chapter 3. Saeed Nejati integrated the BMM framework into the SAT solver, and ran some of the experiments in Section 3.6. Chapter 4 is based on an ICML 2021 UDL workshop paper. I wrote all the contents by myself. My supervisor Pascal Poupart edited this thesis and made many valuable suggestions.

Abstract

With recent advances in approximate inference, Bayesian methods have proven successful in larger datasets and more complex models. The central problem in Bayesian inference is how to approximate intractable posteriors accurately and efficiently. Variational inference deals with this problem by projecting the posterior onto a simpler distribution space. The projection step in variational inference is usually done by minimizing Kullback–Leibler divergence, but alternative methods may sometimes yield faster and more accurate solutions. Moments are statistics to describe the shape of a probability distribution, and one can project the distribution by matching a set of moments. The idea of moment matching dates back to the method of moments (MM), a simple approach to estimate unknown parameters by enforcing the moments to match with estimation. While MM has been primarily studied in frequentist statistics, it can lend itself naturally to approximate Bayesian inference.

This thesis aims to better understand how to apply MM in general-purpose Bayesian inference problems and the advantage of MM methods in Bayesian inference. We begin with the simplest model in machine learning and gradually extend to more complex and practical settings. The scope of our work spans from theory, methodology to applications. We first study a specific algorithm that uses MM in mixture posteriors, Bayesian Moment Matching (BMM). We prove consistency of BMM in a naïve Bayes model and then propose an initializer to Boolean SAT solvers based on its extension to Bayesian networks. BMM is quite restrictive and can only be used with conjugate priors. We then propose a new algorithm, Multiple Moment Matching Inference (MMMI), a general-purpose approximate Bayesian inference algorithm based on the idea of MM, and demonstrate its competitive predictive performance on real-world datasets.

Acknowledgements

First and foremost, I would like to thank Pascal Poupart for being the most supportive, inspiring and knowledgeable supervisor. Three years ago, I knocked the door of Pascal's office, asking if he would consider offering me a part-time research assistant position. At that time, I did not know anything about machine learning or have any research experience. After these years, through Pascal's mentorship, I became much better and more mature both as a researcher and as a person.

Also, I would like to thank two readers, Peter van Beek and Yaoliang Yu, for reading this thesis and providing valuable feedback.

I would like to thank my collaborators: Vijay Ganesh, Saeed Nejati, George Trimponias, Abdullah Rashwan, Guiliang Liu, Yudong Luo. I learned a lot from all of you.

I would like to thank my friends for being there for me at my worst and best: Yuhao Wu, Saiyue Lyu, Xuejun Du, Jiayi Zhang, Guiliang Liu, Yudong Luo, Weidi Zhou, Mingyu Zhu, Chutian Chen, Weidi Zhou, Liang Guo, Yao Qian. I would not survive these five years without any of you.

I would like to thank everyone for offering valuable suggestions about my work: Qiang Liu, Guojun Zhang, Fahiem Bacchus, Chris Maddison, Han Zhao and anonymous reviewers.

I would like to thank the professors at Waterloo for having a positive influence on my professional and personal life through courses they taught: Alice Gao, Yaoliang Yu, Kimon Fountoulakis, Peter van Beek, Brian Forrest, Ross Willard, Kathryn Hare, Mark Petrick, Surya Banerjee, Yingli Qin, Sara Humphreys.

I would like to thank the staff members at the school of computer science for supporting my study, especially my coordinator, Marie Kahkejian.

I would like to thank the writing specialists and immigration specialists at Waterloo for helping me succeed in Canada as an international student.

I would like to thank the Vector Institute for providing the computing resources, scholarships and a tight-knit research community.

I would like to thank all heathcare and frontline workers for getting us through this challenging time.

In the end, I would like to thank my parents' unconditional love and support. You are the best parents in the world.

Dedication

This is dedicated to my parents.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we will review background knowledge and summarize the contributions of the thesis. Section 1.1 reviews basic concepts in Bayesian inference. Section 1.2 discusses the importance of approximate inference. Section 1.3 introduces method of moments in both frequentist and Bayesian statistics. Section 1.4 summarizes the contributions of this thesis.

1.1 Exact Bayesian Inference

The philosophical debate between frequentists and Bayesians has never ended in statistics. The core differences between these two paradigms lie in their views of probabilities and unknown parameters [9], as shown in Table 1.1.

Bayesians treat an unknown parameter θ as the random variable Θ. Θ is assumed to follow a distribution $P(\Theta)$ called **prior**, representing our initial belief before seeing any

	Frequentist	Bayesian
Probability	Long-run frequency of a random event [69]	Quantification of a personal belief [20]
Unknown Parameters	Constant	Random variable

Table 1.1: Views of probabilities and unknown parameters in frequentist and Bayesian methods.

evidence. The belief of θ will be updated according to Bayes' theorem as evidence (data) $D := \{D_i\}_{i=1}^n$ comes:

$$P(\Theta|D) = \frac{P(\Theta)P(D|\Theta)}{P(D)} \tag{1.1}$$

The distribution $P(\Theta|D)$ is called **posterior**, which is the updated belief after seeing D. $P(D|\Theta)$ is the **likelihood**, which represents the probability of seeing evidence D given our current belief.

The posterior $P(\Theta|D)$ also allows us to make inference about the unseen observation X by computing the **predictive distribution**:

$$\begin{aligned} P(X|D) &= \int_\theta P(\theta, X|D)d\theta \\ &= \int_\theta P(X|\theta, D)P(\theta|D)d\theta \\ &= \int_\theta P(X|\theta)P(\theta|D)d\theta \end{aligned}$$

Compared with the frequentists, the prediction of Bayesians is a distribution rather than a single value, which provides information on not only what the prediction is but also how confident we are in the prediction.

In sequential Bayesian inference, the evidence $D := \{D_i\}_{i=1}^n$ will come in batches. The posterior $P(\Theta|D_i)$ after seeing the current data point $P(\Theta|D_i)$ will act as the prior for the following data D_{i+1}.

1.2 Approximate Bayesian Inference

The central problem in Bayesian inference lies in computing the posterior distribution in Equation 1.1. The denominator $P(D)$ is calculated by marginalizing over the numerator:

$$\begin{aligned} P(D) &= \int_\theta P(\theta, D)d\theta \\ &= \int_\theta P(\theta)P(D|\theta)d\theta \end{aligned} \tag{1.2}$$

In general, the integral in Equation 1.2 is difficult to compute. However, if the likelihood has **conjugate priors**, the posterior will be in the same distribution family as the prior, and computing the denominator $P(D)$ becomes easy. For the likelihood in the exponential family, the conjugate prior exists, and it is usually also in the exponential family [71]. For example, a normal distribution with known variance is conjugate with itself, and the Bernoulli distribution is conjugate with the Beta distribution.

However, in most cases, the likelihood function is quite complex, and we cannot find a conjugate prior in the common distribution family. Therefore, we need to approximate the posterior, and this problem is called **approximate Bayesian inference**.

Most approximate Bayesian inference methods fall into two categories: **variational inference** and **Markov chain Monte Carlo (MCMC)**.

Variational inference can be formulated as an optimization problem that projects the posterior onto an approximation set. With stochastic gradient descent, variational inference has been shown to scale well to large datasets [34]. The quality of variational inference highly depends on the chosen approximation set. Small sets cannot approximate the true posterior well, while larger sets may bear high computation costs. Therefore, variational inference often requires case-by-case design to strike a good balance between accuracy and computational cost.

MCMC, on the other hand, is a sampling algorithm that constructs a Markov chain whose stationary distribution is the target posterior [64]. Even though samples are guaranteed to converge to the true posterior, it may take many iterations before convergence. Recently, stochastic gradient variants of MCMC, such as stochastic gradient Langevin dynamics [87], stochastic gradient Hamiltonian Monte Carlo [15] and stochastic gradient thermostats [23], have begun to show promising performance in large datasets.

Another line of work focuses on developing new approximate inference algorithms that combine scalability of variational inference and flexibility of MCMC. An outstanding representative is stein variational gradient descent [51], which will be discussed in detail in Chapter 4.

1.3 Method of Moments

Moments can be seen as a quantitative summary for the shape of a probability distribution. More formally, the $n-$th order moment M_n of a distribution $p(\Theta)$ can be defined

	Related Statistics	Interpretation
M_1	Expected value	Where is the distribution centered?
M_2	Variance	How flat is the distribution?
M_3	Skewness	Is the distribution symmetric or skewed to one side?
M_4	Kurtosis	How heavy is the tail of the distribution?

Table 1.2: Interpretation of lower-order moments

as:

$$M_n := \mathbb{E}[\Theta^n] = \int \theta^n p(\theta) d\theta \tag{1.3}$$

The lower-order moments are closely related to statistics describing the shape of a distribution. Their interpretations are summarized in Table 1.2.

While Equation 1.3 is difficult to compute, we can estimate the moments as the average of samples $\{\theta_i\}_{i=1}^k$:

$$M_n :\approx \frac{1}{k} \sum_{i=1}^{k} \theta_i^n \tag{1.4}$$

It can be shown that Equation 1.4 is an unbiased estimator of M_n [35].

Moments can be seen as unique characteristics of a distribution function. In fact, Hausdorff's moment theorem states that moments of all orders (from 0 to ∞) uniquely determine a distribution defined on a bounded region [32].

Method of moments (MM) is one of the simplest approaches for parameter estimation. It enforces the constraint that population moments (function of unknown parameters) are equal to sample moments (numeric values), turning the problem of parameter estimation into solving a system of equations. Despite being simple, the method of moments is a consistent estimator that offers an alternative to maximum likelihood estimation when we do not have access to likelihood functions. In machine learning, the method of moments has proven successful in learning mixture models, latent Dirichlet allocation and hidden Markov models. [3]

As moments describe the shape of a distribution, the difference between moments can be used to represent the distance of distributions. Maximum mean discrepancy (MMD) with the kernel trick measures the distance between all moments of the two distributions

[28]. MMD has been used as a simple and effective metric of distance in several areas of machine learning, such as generative models [46] [81], reinforcement learning [70] and language embeddings [90].

While most studies on MM have been frequentist, MM can lend itself naturally to approximate Bayesian inference. More specifically, we can project the posterior onto a simpler distribution by matching a set of moments. In sequential Bayesian inference, when the posterior is a mixture distribution, the number of components can grow exponentially. **Bayesian moment matching** (BMM) deals with this problem by projecting the mixture distribution to a distribution in the same family as the prior by matching a set of sufficient moments. A detailed description of BMM can be found in Chapter 2. BMM has been successfully applied in topic modelling [36], sum-product networks [74] and hidden Markov models [39].

1.4 Contributions

This thesis studies the method of moments in approximate Bayesian inference. Our contributions can be summarized in 3 areas: theory, application and methodology.

- *Theory:* While the frequentist's method of moments has been proven to be consistent under some mild conditions [31], the theoretical properties of its Bayesian counterpart are largely unstudied. Starting from a simple setting, we prove consistency of Bayesian Moment Matching (BMM) in a naïve Bayes model. This work corresponds to Chapter 2.

- *Application:* The Boolean satisfiability problem (SAT) is one of the most fundamental problems in computer science. Despite its NP-complete nature, SAT solvers are actually well-engineered and can solve large industrial instances efficiently. We propose a BMM-based framework to solve the initialization problem in Boolean SAT solvers. This work corresponds to Chapter 3.

- *Methodology:* BMM requires the posterior moments to have closed-form solutions. However, this is impossible in most practical models, such as Bayesian logistic regression and Bayesian neural networks. To fill in this gap, we propose Multiple Moment Matching Inference (MMMI), a general-purpose and flexible approximate Bayesian inference algorithm based on the idea of moment matching. We further demonstrate its competitive predictive performance on multiple real-world datasets. This work corresponds to Chapter 4.

Chapter 2

On the Naïve Bayes Model: Proof of Consistency Properties

In this chapter, we study parameter estimation for a special naïve Bayes model using BMM, as shown in Figure 2.1. This model serves as the foundation of the algorithm proposed in Chapter 3. Section 2.1 introduces the problem and notation. Section 2.2 states the main theorem. Section 2.3 introduces the main theoretical framework, stochastic approximation. Section 2.4 outlines the proof of the main theorem. Section 2.5 presents the detailed proof.

2.1 Problem Setup

In Figure 2.1, Z is a binary hidden variable and X is a binary observable variable. Furthermore, we assume that the conditional distribution $P(X|Z)$ is fully known:

$$c_1 := P(X = 0|Z = 0)$$
$$c_2 := P(X = 0|Z = 1)$$

Here, c_1 and c_2 are assumed to be given.

Let θ represent the unknown probability associated with the hidden variable, i.e.

$$\theta := P(Z = 0)$$

θ is the quantity we wish to infer from *i.i.d.* observations $\{X_1, X_2, \dots\}$ in an online and Bayesian fashion.

Figure 2.1: X is the observable variable. Z is the hidden variable. Θ is the unknown quantity associated with Z. The goal is to infer the quantity Θ through observations of X.

In Bayesian learning, the unknown quantities will be treated as random variables. In this chapter, we use the big Θ to denote the random variable of θ in Bayesian learning, and $\hat{\theta}$ to denote the true underlying value.

A beta distribution $Beta(\theta; \alpha_0, \beta_0)$ is chosen as the initial prior over Θ:

$$P(\Theta) = Beta(\theta; \alpha_0, \beta_0) = \frac{1}{B(\alpha_0, \beta_0)} \theta^{\alpha_0 - 1}(1-\theta)^{\beta_0 - 1}$$

where $B(\alpha_0, \beta_0)$ represents the beta function of α_0 and β_0, which is defined as:

$$B(\alpha_0, \beta_0) = \int_0^1 t^{\alpha_0 - 1}(1-t)^{\beta_0 - 1} dt$$

We choose a beta distribution because its support is a probability simplex and it is also conjugate with the likelihood [55]. The posterior after observing the first evidence X_1 is:

$$
\begin{aligned}
P(\Theta|X=0) &= \frac{P(\Theta)P(X=0|\Theta)}{P(X=0)} \\
&= \frac{1}{P(X=0)}\frac{\theta^{\alpha_0-1}(1-\theta)^{\beta_0-1}}{B(\alpha_0,\beta_0)}(\theta c_1 + (1-\theta)c_2) \\
&= \frac{1}{P(X=0)}(c_1\frac{\theta^{\alpha_0}(1-\theta)^{\beta_0-1}}{B(\alpha_0,\beta_0)} + c_2\frac{\theta^{\alpha_0-1}(1-\theta)^{\beta_0}}{B(\alpha_0,\beta_0)}) \\
&= a_0\frac{\theta^{\alpha_0}(1-\theta)^{\beta_0-1}}{B(\alpha_0+1,\beta_0)} + b_0\frac{\theta^{\alpha_0-1}(1-\theta)^{\beta_0}}{B(\alpha_0,\beta_0+1)} \\
P(\Theta|X=1) &= a_1\frac{\theta^{\alpha_0}(1-\theta)^{\beta_0-1}}{B(\alpha_0+1,\beta_0)} + b_1\frac{\theta^{\alpha_0-1}(1-\theta)^{\beta_0}}{B(\alpha_0,\beta_0+1)},
\end{aligned}
\tag{2.1}
$$

where

$$
\begin{aligned}
\frac{1}{P(X=0)} &= \frac{1}{\int_0^1 \frac{1}{B(\alpha_0,\beta_0)}\theta^{\alpha_0-1}(1-\theta)^{\beta_0-1}(\theta c_1 + (1-\theta)c_2)d\theta} = \frac{\alpha_0+\beta_0}{c_1\alpha_0 + c_2\beta_0} \\
a_0 &= \frac{c_1}{P(X=0)}\frac{B(\alpha_0+1,\beta_0)}{B(\alpha_0,\beta_0)} = \frac{c_1\alpha_0}{c_1\alpha_0 + c_2\beta_0} \\
b_0 &= \frac{c_2\beta_0}{c_1\alpha_0 + c_2\beta_0} \\
a_1 &= \frac{d_1\alpha_0}{d_1\alpha_0 + d_2\beta_0} \\
b_1 &= \frac{d_2\beta_0}{d_1\alpha_0 + d_2\beta_0}.
\end{aligned}
$$

Note that conjugacy in Equation 2.1 relies on a nice property of the Beta function:

$$
\begin{aligned}
B(\alpha+1,\beta) &= B(\alpha,\beta)\frac{\alpha}{\alpha+\beta} \\
B(\alpha,\beta+1) &= B(\alpha,\beta)\frac{\beta}{\alpha+\beta}
\end{aligned}
\tag{2.2}
$$

Equation 2.1 suggests that the posterior is a mixture of two beta distributions after observing the first point. In sequential inference, this posterior will act as the prior when the next observation arrives. As we have more data points, the number of mixture components in the posterior distributions will grow exponentially since the number of components dou-

bles with each observation, making inference intractable. To solve this problem, BMM uses a single beta distribution $\tilde{P}(\Theta_1) = Beta(\Theta_1; \alpha_1, \beta_1)$ to approximate the mixture posterior by matching the first and second moments after one observation. This way, we will always end up with a posterior of two mixture components. Concretely, α_1, β_1 can be obtained by solving 2.3:

$$\begin{aligned} \mathbb{E}_{\Theta_1 \sim Beta(\Theta_1;\alpha_1,\beta_1)}[\Theta_1] &:= \mathbb{E}_{\Theta_0 \sim P(\Theta|X_1)}[\Theta] \\ \mathbb{E}_{\Theta_1 \sim Beta(\Theta_1;\alpha_1,\beta_1)}[\Theta_1^2] &:= \mathbb{E}_{\Theta_0 \sim P(\Theta|X_1)}[\Theta^2], \end{aligned} \tag{2.3}$$

For Beta distributions, the first and second moments are rational functions:

$$\begin{aligned} \mathbb{E}_{\Theta \sim Beta(\Theta;\alpha,\beta)}[\Theta] &= \frac{\alpha}{\alpha+\beta} \\ \mathbb{E}_{\Theta \sim Beta(\Theta;\alpha,\beta)}[\Theta^2] &= \frac{\alpha(\alpha+1)}{(\alpha+\beta)(\alpha+\beta+1)} \end{aligned} \tag{2.4}$$

Solving Equation 2.3 is equivalent to solving a system of linear equations. Furthermore, the solution can be expressed in closed forms.

Without loss of generality, let's assume the observation $X_1 = 0$. Since the posterior is a mixture of Beta distributions, the RHS of Equation 2.4 can be easily computed, denoted by M_1, M_2:

$$\begin{aligned} M_1 &:= a_0 \frac{\alpha_0 + 1}{\alpha_0 + \beta_0 + 1} + (1 - a_0) \frac{\alpha_0}{\alpha_0 + \beta_0 + 1} \\ M_2 &:= a_0 \frac{(\alpha_0 + 1)((\alpha_0 + 2))}{(\alpha_0 + \beta_0 + 1)(\alpha_0 + \beta_0 + 2)} + (1 - a_0) \frac{\alpha_0(\alpha_0 + 1)}{(\alpha_0 + \beta_0 + 1)(\alpha_0 + \beta_0 + 2)} \end{aligned}$$

One can show that the solution α_1, β_1 to Equation 2.4 is:

$$\begin{aligned} \alpha_1 &= \frac{(M_2 - M_1)M_1}{M_1^2 - M_2} \\ \beta_1 &= \frac{(M_2 - M_1)(1 - M_1)}{M_1^2 - M_2} \end{aligned}$$

The above process is repeated for each new observation. The approximate Beta posterior from the current step serves as the prior over Θ for the next step. Even though some information is lost when we use a single beta distribution to approximate a mixture

of betas, we can prove the consistency of the first moment in this setting by formulating it as a stochastic approximation (SA) problem in this setting.

Please note that the purpose of this chapter is not to propose the simplest approach to solve this problem. Instead, our goal is to use this setting to introduce the BMM algorithm and study its consistency. One simple way to solve θ is using maximum likelihood estimation (MLE), which gives us a consistent and unbiased estimator:

$$\tilde{\theta} = \frac{\frac{\sum_{i=1}^{n} X_i}{N} - c_2}{c_1 - c_2} \tag{2.5}$$

2.2 Main Theorem

If we apply BMM n consecutive times with observations $\{X_1, \ldots, X_n\}$, our n^{th} estimate Θ_n will be by induction distributed according to $Beta(\Theta; \alpha_n, \beta_n)$, for suitable α_n, β_n.

Let μ_n denote the mean of Θ_n :

$$\mu_n := \frac{\alpha_n}{\alpha_n + \beta_n}$$

τ_n denotes the precision of Θ_n:

$$\tau_n := \alpha_0 + \beta_0$$

and X_{n+1} is the binary random variable for the next new instance. Note that due to randomness of the observations X, μ_n and τ_n are also considered to be random variables.

Even though some information is lost when the posterior is projected onto a unimodal distribution, we prove that μ_n is actually a consistent estimator of $\hat{\theta}$:

$$P(\lim_{n\to\infty} \mu_n = \hat{\theta}) = 1$$

.

Theorem 1 *When performing BMM with the first and second moment in the naïve Bayes model, the update in Eq. (2.7) for the first moment converges almost surely to the true underlying $\hat{\theta}$ under three mild conditions:*

1. $\hat{\theta} \in (0, 1)$

2. $c_1, c_2 \in (0, 1)$

3. $c_1 \neq c_2$

We provide a sketch for the proof of Theorem 1 in the next two sections.

2.3 Stochastic Approximation

Stochastic approximation (SA) algorithms use recursive updates to find roots for the noisy function. If the objective function f is the gradient of another function, SA can also be viewed as optimization algorithms. The field of SA begins with the celebrated Robbins–Monro algorithm, which proposes an iterative scheme to find roots for some function families through noisy observations with a guarantee of consistency [75].

In general, SA aims to find the solution θ^* to:

$$\hat{f}(\theta^*) := \mathbb{E}_W(f(\theta, W)) = 0$$

where W represents the noise.

SA is a mature field of numerical analysis, offering a theoretical framework to analyze many modern machine learning algorithms, such as stochastic variational inference [34], stochastic gradient langevin dynamics [87] and Q-learning [83].

Chen and Ryzhov propose a general form of SA updates: [16]:

$$x_{n+1} = x_n - \psi_n(Q_n(W_{n+1}, x_n) + \zeta_n(W_{n+1}, x_n, \psi_n)) \tag{2.6}$$

where $(x_n)_{n=0}^{\infty} \in \Re^m$, $(\psi_n)_{n=0}^{\infty}$ is the step size (deterministic or stochastic), $(W_n)_{n=0}^{\infty}$ is a sequence of random variables, $(Q_n)_{n=0}^{\infty}$ and $(\zeta_n)_{n=0}^{\infty}$ are two sequences of real measurable functions. The function ζ_n corresponds to the bias.

Based on the above notation, the following 2 terms are defined:

$$\begin{aligned} \mathcal{F}_n &= \mathcal{B}(W_1, ..., W_n, x_1, ..., x_n, \psi_1, ..., \psi_n) \\ R_n(x) &= \mathbb{E}[Q_n(W_{n+1}, x)|\mathcal{F}_n] \end{aligned}$$

where $\mathcal{B}$ denotes the Borel sigma-algebra.

Theorem 2 proves the sufficient conditions for SA algorithms to converge almost surely [16].

Theorem 2 *Let x_n be defined by Equation 2.6. Then $x_n \to \theta$ almost surely if the following four conditions are met*

1. *For any n, the system of equations $R_n(x)$ has a unique root θ, which doesn't depend on n.*
2. *For $n = 1, 2, \ldots$ and any $\epsilon > 0$,*
$$\inf_{\|x-\theta\|_2^2 > \epsilon, n \in \mathbb{N}} (x-\theta)^T R_n(x) > 0.$$
3. *There exist positive constants C_1 and C_2 such that:*
   - $\sup_{n\in\mathbb{N}} \mathbb{E}[\|Q_n(W_{n+1}, x)\|_2^2 | \mathcal{F}_n] \le C_1(1 + \|x-\theta\|_2^2)$
   - $\sup_{n\in\mathbb{N}} \mathbb{E}[\|\zeta_n(W_{n+1}, x, \psi_n)\|_2^2 | \mathcal{F}_n]/\psi_n^2 \le C_2(1 + \|x-\theta\|_2^2)$

   for all x.
4. *$\sum_{n=0}^{\infty} \psi_n = \infty, \sum_{n=0}^{\infty} \psi_n^2 < \infty$.*

2.4 Proof Sketch

It has been observed that an approximate Bayesian update can be viewed as SA with the addition of a "bias" term representing the difference between the frequentist and Bayesian versions of the stochastic gradient [16]. We interpret the setting in Section 2.1 as an SA problem and use Theorem 2 [16] to derive the consistency result.

For the problem described in Section 2.1, by matching the first two moments of the posterior $P(\Theta_n | X_{n+1})$ at step $n+1$ with the moments of a Beta distribution $Beta(\theta; \alpha_{n+1}, \beta_{n+1})$, we get two recursive update equations, one for α_{n+1} and another for β_{n+1}.

Equivalently, we can write the recursive update equations in terms of μ_{n+1} and τ_{n+1} using $\alpha_n, \beta_n, \mu_n, \tau_n$ (μ_n, τ_n are functions of α_n, β_n). For instance, the update equation for μ_{n+1} is:

$$\begin{aligned} \mu_{n+1} = &\mu_n + \frac{1}{\tau_n + 1}\Big[\Big(\frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n} - \mu_n\Big)(1 - X_{n+1}) \\ &+ \Big(\frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n} - \mu_n\Big)X_{n+1}\Big]. \end{aligned} \tag{2.7}$$

Equation 2.7 can be seen as an SA update, with stochasticity coming from the observations X.

Actually, one can show that Equation 2.7 can be written in the form of Equation 2.6. We prove consistency of Θ_n by showing that Equation 2.7 satisfies the four sufficient conditions of Theorem 2. The detailed proof of each assumption is in Section 2.5.

2.5 Proof

2.5.1 SA Formulation

We define:

$$d_1 := P(X = 1|Z = 0) = 1 - c_1$$
$$d_2 := P(X = 1|Z = 1) = 1 - c_2$$

To avoid degenerate edge cases, we assume that $\hat{\theta} \in (0, 1)$ and $c_1 \neq c_2$. Furthermore, we consider that $c_1, c_2 \in (0, 1)$, which trivially implies that $d_1, d_2 \in (0, 1)$. We choose a beta distribution $Beta(\theta_0; \alpha_0, \beta_0)$ as the initial prior over Θ.

After observing n binary *i.i.d.* observations $\{X_1, \ldots, X_n\}$ and performing BMM, we will have an estimate θ_n for Θ which is distributed as a Beta distribution $Beta(\theta_n; \alpha_n, \beta_n)$. The posterior after observing the $(n + 1)^{th}$ point X_{n+1} is:

$$\begin{aligned}
P(\theta_{n+1}|X_{n+1} = 0) &= \frac{P(\theta_n)P(X_{n+1} = 0|\theta_n)}{P(X_{n+1} = 0)} \\
&= \frac{1}{P(X_{n+1} = 0)} \frac{\theta_n^{\alpha_n - 1}(1 - \theta_n)^{\beta_n - 1}}{B(\alpha_n, \beta_n)} (\theta_n c_1 + (1 - \theta_n)c_2) \\
&= \frac{1}{P(X_{n+1} = 0)} (c_1 \frac{\theta_n^{\alpha_n}(1 - \theta_n)^{\beta_n - 1}}{B(\alpha_n, \beta_n)} + c_2 \frac{\theta_n^{\alpha_n - 1}(1 - \theta_n)^{\beta_n}}{B(\alpha_n, \beta_n)}) \\
&= a_{n,0} \frac{\theta_n^{\alpha_n}(1 - \theta_n)^{\beta_n - 1}}{B(\alpha_n + 1, \beta_n)} + b_{n,0} \frac{\theta_n^{\alpha_n - 1}(1 - \theta_n)^{\beta_n}}{B(\alpha_n, \beta_n + 1)} \\
P(\theta_{n+1}|X_{n+1} = 1) &= a_{n,1} \frac{\theta_n^{\alpha_n}(1 - \theta_n)^{\beta_n - 1}}{B(\alpha_n + 1, \beta_n)} + b_{n,1} \frac{\theta_n^{\alpha_n - 1}(1 - \theta_n)^{\beta_n}}{B(\alpha_n, \beta_n + 1)},
\end{aligned}$$

where

$$\frac{1}{P(X_{n+1}=0)} = \frac{1}{\int_0^1 \frac{1}{B(\alpha_n,\beta_n)}\theta_n^{\alpha_n-1}(1-\theta_n)^{\beta_n-1}(\theta_n c_1+(1-\theta_n)c_2)d\theta_n} = \frac{\alpha_n+\beta_n}{c_1\alpha_n+c_2\beta_n}$$

$$\begin{aligned}
a_{n,0} &= \frac{c_1}{P(X_{n+1}=0)}\frac{B(\alpha_n+1,\beta_n)}{B(\alpha_n,\beta_n)} = \frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n} \in (0,1), \text{ since } c_1,c_2 \in (0,1) \wedge \alpha_n,\beta_n > 0 \\
b_{n,0} &= \frac{c_2\beta_n}{c_1\alpha_n+c_2\beta_n} \in (0,1) \\
a_{n,1} &= \frac{d_1\alpha_n}{d_1\alpha_n+d_2\beta_n} \in (0,1) \\
b_{n,1} &= \frac{d_2\beta_n}{d_1\alpha_n+d_2\beta_n} \in (0,1).
\end{aligned} \tag{2.8}$$

Let θ_n be a r.v. distributed according to the Beta distribution $Beta(\theta_n;\alpha_n,\beta_n)$. We use μ_n to denote the mean of θ_n, σ_n^2 to denote the variance of $\theta_n = \alpha_n+\beta_n$, τ_n to denote the precision of θ_n, and λ_n to denote $\frac{1}{\tau_n^2\sigma_n^2}$. In particular, the following equations hold (note the first two are standard identities for the Beta distribution):

$$\begin{aligned}
\mu_n &= \frac{\alpha_n}{\alpha_n+\beta_n} \in (0,1) \text{ (since } \alpha_n,\beta_n > 0) \\
\sigma_n &= \frac{\alpha_n\beta_n}{(\alpha_n+\beta_n)^2(\alpha_n+\beta_n+1)} > 0 \\
\tau_n &= \alpha_n+\beta_n > 0 \\
\lambda_n &= \frac{1}{\tau_n^2\sigma_n^2} > 0.
\end{aligned}$$

BMM approximates the mixture posterior $P(\theta_{n+1}|X_{n+1})$ by a simpler Beta distribution $Beta(\theta_{n+1};\alpha_{n+1},\beta_{n+1})$ by matching the first and second moments. The first moment of this beta distribution is $\frac{\alpha_{n+1}}{\alpha_{n+1}+\beta_{n+1}} = \mu_{n+1}$ while its second moment is $\frac{\alpha_{n+1}(\alpha_{n+1}+1)}{(\alpha_{n+1}+\beta_{n+1})(\alpha_{n+1}+\beta_{n+1}+1)}$. By matching the first moments, we get the update equation of μ_{n+1} as follows. The equation incorporates the two cases that the observed instance I_{n+1} is 0 or 1.

$$
\begin{aligned}
\mu_{n+1} &= (\frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n}\frac{\alpha_n+1}{\alpha_n+\beta_n+1} + \frac{c_2\beta_n}{c_1\alpha_n + c_2\beta_n}\frac{\alpha_n}{\alpha_n+\beta_n+1})(1-X_{n+1}) \\
&+ (\frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n}\frac{\alpha_n+1}{\alpha_n+\beta_n+1} + \frac{(1-c_1)\beta_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n}\frac{\alpha_n}{\alpha_n+\beta_n+1})X_{n+1} \\
&= \frac{1}{\tau_n+1}[(\frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n}(\alpha_n+1) + \frac{c_2\beta_n}{c_1\alpha_n + c_2\beta_n}\alpha_n)(1-X_{n+1}) \\
&+ (\frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n}(\alpha_n+1) + \frac{(1-c_1)\beta_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n}\alpha_n)X_{n+1}] \\
&= \frac{1}{\tau_n+1}[(\alpha_n + \frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n})(1-X_{n+1}) + (\frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n} + \alpha_n)X_{n+1}] \\
&= \mu_n + \frac{1}{\tau_n+1}[(\alpha_n + \frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n})(1-X_{n+1}) + (\frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n} + \alpha_n)X_{n+1}] - \mu_n \\
&= \mu_n + \frac{1}{\tau_n+1}[(\alpha_n + \frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n})(1-X_{n+1}) + (\frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n} + \alpha_n)X_{n+1} - \mu_n(\tau_n+1)] \\
&= \mu_n + \frac{1}{\tau_n+1}[\frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n}(1-X_{n+1}) + \frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n}X_{n+1} + \alpha_n\mu_n(\tau_n+1)] \\
&= \mu_n + \frac{1}{\tau_n+1}[(\frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n} - \mu_n)(1-X_{n+1}) + (\frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n} - \mu_n)X_{n+1}] \\
&= \mu_n + \lambda_n[\frac{\sigma_n^2\tau_n^2}{\tau_n+1}(\frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n} - \mu_n)(1-X_{n+1}) + \frac{\sigma_n^2\tau_n^2}{\tau_n+1}(\frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n} - \mu_n)X_{n+1}].
\end{aligned}
\tag{2.9}
$$

It is straightforward to show that $0 < \mu_{n+1} < 1$, given $\alpha_n, \beta_n > 0, \forall n$.

By matching the second moments we similarly get the update equation for the second moment:

$$
M_{n+1,2} = \frac{(\mu_n\tau_n+1)(\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n})}{(\tau_n+1)(\tau_n+2)}(1-X_{n+1}) + \frac{(\mu_n\tau_n+1)(\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n})}{(\tau_n+1)(\tau_n+2)}(X_{n+1}) \tag{2.10}
$$

But in this setting it is more convenient to use instead the first moment and the precision ($\tau_{n+1} := \alpha_{n+1} + \beta_{n+1}$) (see also [16]). The update equation for precision is obtained by first solving Eq. (2.9) and Eq. (2.10) in terms of α_{n+1} and β_{n+1} using the fact that

$\mu_{n+1} = \frac{\alpha_{n+1}}{\alpha_{n+1}+\beta_{n+1}}$ and $M_{n+1,2} = \frac{\alpha_{n+1}(\alpha_{n+1}+1)}{(\alpha_{n+1}+\beta_{n+1})(\alpha_{n+1}+\beta_{n+1}+1)}$:

$$\tau_{n+1} = \tau_n + \frac{A_{n,0}(\tau_n+1) - \tau_n(\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n})B_{n,0}}{B_{n,0}(\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n})}(1-X_{n+1}) \tag{2.11}$$

$$+ \frac{A_{n,1}(\tau_n+1) - \tau_n(\mu_n\tau_n + \frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n+(1-c_2)\beta_n})B_{n,1}}{B_{n,1}(\mu_n\tau_n + \frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n+(1-c_2)\beta_n})}(X_{n+1}), \tag{2.12}$$

where we have defined:

$$\begin{aligned}
A_{n,0} &= (\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n})^2(\tau_n+2) - (\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n})(\mu_n\tau_n+1)(\mu_n\tau_n + 2\frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n}) \\
B_{n,0} &= (\mu_n\tau_n+1)(\mu_n\tau_n + 2\frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n})(\tau_n+1) - (\tau_n+2)(\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n})^2 \\
A_{n,1} &= (\mu_n\tau_n + \frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n+(1-c_2)\beta_n})^2(\tau_n+2) \\
&\quad - (\mu_n\tau_n + \frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n+(1-c_2)\beta_n})(\mu_n\tau_n+1)(\mu_n\tau_n + 2\frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n+(1-c_2)\beta_n}) \\
B_{n,1} &= (\mu_n\tau_n+1)(\mu_n\tau_n + 2\frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n+(1-c_2)\beta_n})(\tau_n+1) \\
&\quad - (\tau_n+2)(\mu_n\tau_n + \frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n+(1-c_2)\beta_n})^2.
\end{aligned} \tag{2.13}$$

We finally derive the update equation for the variance below, using the standard formula $\sigma^2_{n+1} = M_{n+1,2} - \mu^2_{n+1}$ together with Eq. (2.9) and Eq. (2.10). We will need this in Section 2.5.5.

$$\begin{aligned}
\sigma^2_{n+1} =& \sigma^2_n + \frac{1}{(\tau_n+1)(\tau_n+2)}[(\tau_n+1)(\mu_n\tau_n+1)(\mu_n\tau_n + 2\frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n}) \\
&- (\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n})^2(\tau_n+2) - \mu_n(1-\mu_n)](\tau_n+1)^2(\tau_n+2)(1-X_{n+1})] \\
&+ \frac{1}{(\tau_n+1)^2(\tau_n+2)}[(\tau_n+1)(\mu_n\tau_n+1)(\mu_n\tau_n + 2\frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n+(1-c_2)\beta_n}) \\
&- (\mu_n\tau_n + \frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n+(1-c_2)\beta_n})^2(\tau_n+2) - \mu_n(1-\mu_n)(\tau_n+1)(\tau_n+2)](X_{n+1}).
\end{aligned} \tag{2.14}$$

We next define:

$$E_n := \frac{\sigma_n^2 \tau_n^2}{\tau_n + 1} > 0,$$

$$Q_n(X_{n+1}, E_n, \frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n}, \frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n}, \mu_n)$$

$$:= -[E_n(\frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n} - \mu_n)(1 - X_{n+1}) + E_n(\frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n} - \mu_n)X_{n+1}].$$

Then, the update rule in Equation (2.9) can be rewritten as:

$$\mu_{n+1} = \mu_n - \lambda_n \cdot Q_n(X_{n+1}, E_n, \frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n}, \frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n}, \mu_n),$$

which has the general SA structure (2.6) with no bias term ($\zeta_n := 0$) and λ_n as the step size ($\psi_n := \lambda_n$).

We further define:

$$\mathcal{F}_n := \mathcal{B}(X_1, ..., X_n, \mu_1, ..., \mu_n, \lambda_1, ..., \lambda_n)$$

$$R_n(x) := \mathbb{E}[Q_n(X_{n+1}, E_n, \frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n}, \frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n}, x)|\mathcal{F}_n]$$

Also, we assume that the sequences $(\alpha_n)_{n=0}^{\infty}$ and $(\beta_n)_{n=0}^{\infty}$ have positive lower bounds, which is standard in many SA convergence proofs and also consistent with empirical evidence. Then we can show that $\mu_n \to \theta$ almost surely by verifying the 4 assumptions proposed by [16]. The proof for our setting shares elements with Proposition EC.1 for the setting described in Section 4.5.1 of [16].

2.5.2 Proof of Assumption 1

In our formulation

$$\begin{aligned} R_n(x) =& \mathbb{E}[Q_n(X_{n+1}, E_n, \frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n}, \frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n}, x)|\mathcal{F}_n] \\ =& - E_n \left(\frac{c_1 x}{c_1 x + c_2(1-x)} - x \right) (1 - (\theta d_1 + (1-\theta)d_2)) \\ & - E_n \left(\frac{d_1 x}{d_1 x + d_2(1-x)} - x \right) (\theta d_1 + (1-\theta)d_2)) \end{aligned}$$

it is easy to see that $R_n(\theta) = 0$ for any n. We now show that θ is additionally the unique root for $x \in (0, 1)$. Given that $E_n > 0$, this is equivalent to showing:

$$\left(\frac{c_1}{c_1 x + c_2(1-x)} - 1 \right) (1 - (\theta d_1 + (1-\theta)d_2)) + \left(\frac{d_1}{d_1 x + d_2(1-x)} - 1 \right) (\theta d_1 + (1-\theta)d_2) = 0$$

Indeed, we have:

$$\begin{aligned} & \left(\frac{c_1}{c_1 x + c_2(1-x)} - 1 \right) (1 - (\theta d_1 + (1-\theta)d_2)) + \left(\frac{d_1}{d_1 x + d_2(1-x)} - 1 \right) (\theta d_1 + (1-\theta)d_2) = 0 \\ \Rightarrow & \frac{c_1 - c_1 x - c_2(1-x)}{c_1 x + c_2(1-x)} (1 - (\theta d_1 + (1-\theta)d_2)) + \frac{d_1 - d_1 x - d_2(1-x)}{d_1 x + d_2(1-x)} (\theta d_1 + (1-\theta)d_2) = 0 \\ \Rightarrow & \frac{(c_1 - c_2)(1-x)}{c_1 x + c_2(1-x)} (1 - (\theta d_1 + (1-\theta)d_2)) + \frac{(d_1 - d_2)(1-x)}{d_1 x + d_2(1-x)} (\theta d_1 + (1-\theta)d_2) = 0 \end{aligned}$$

Given $c_1 - c_2 = -(d_1 - d_2)$ and $x \in (0, 1)$, the last equation can be written as:

$$\frac{1}{c_1 x + c_2(1-x)}(1 - (\theta d_1 + (1-\theta)d_2)) - \frac{1}{d_1 x + d_2(1-x)}(\theta d_1 + (1-\theta)d_2) = 0$$
$$\Rightarrow \frac{1 - (\theta d_1 + (1-\theta)d_2)}{c_1 x + c_2(1-x)} = \frac{\theta d_1 + (1-\theta)d_2}{d_1 x + d_2(1-x)}$$
$$\Rightarrow (1 - (\theta d_1 + (1-\theta)d_2))(d_1 x + d_2(1-x)) = (\theta d_1 + (1-\theta)d_2)(c_1 x + c_2(1-x))$$
$$\Rightarrow (1 - (\theta d_1 + (1-\theta)d_2))(d_1 - d_2)x + (1 - (\theta d_1 + (1-\theta)d_2))d_2$$
$$= (\theta d_1 + (1-\theta)d_2)(c_1 - c_2)x + (\theta d_1 + (1-\theta)d_2)c_2$$
$$\Rightarrow (1 - (\theta d_1 + (1-\theta)d_2))d_2 - (\theta d_1 + (1-\theta)d_2)c_2 = (c_1 - c_2)x$$
$$\Rightarrow d_2 - (\theta d_1 + (1-\theta)d_2)d_2 - (\theta d_1 + (1-\theta)d_2)c_2 = -(d_1 - d_2)x$$
$$\Rightarrow d_2 - (\theta d_1 + (1-\theta)d_2) = (d_2 - d_1)x$$
$$\Rightarrow (d_2 - d_1)\theta = (d_2 - d_1)x$$
$$\Rightarrow x = \theta.$$

Note that the last step in the above derivation is valid since we have assumed $c_1 \neq c_2$, or equivalently, $d_1 \neq d_2$.

Technically, note that there is also a root at $x = 0$, which is not in $(0, 1)$. However, it is not hard to show that $x \cdot R_n(x) < 0$ in the neighborhood of $x = 0$ when $\theta \in (0, 1)$, which implies that Assumption 2 is violated at $x = 0$, and the SA algorithm is repelled from $x = 0$, provided that the initial $x_0 \in (0, 1)$ [11]. Similar arguments hold for $x = 1$. Alternatively, we can use the previously mentioned update form (2.6) with a projection operator Π_H that projects x_{n+1} into a suitable closed interval $[M_{low}, M_{high}] \subset (0, 1)$, where $0 < M_{low}, M_{high} < 1$, so that $x_0, \theta \in H$ [44]. In that case, the equation $R_n(x) = 0$ has a sole root in the interval H. Finally, given that $\mu_n \in (0, 1) \forall n$, we safely assume everywhere in the proof that $x \in (0, 1)$.

To simplify the proof, from now on we can assume a projection operator Π_H that projects x_{n+1} into a suitable closed interval $[M_{low}, M_{high}] \subset (0, 1)$, as explained above.

2.5.3 Proof of Assumption 2

To show that Assumption 2 holds, it is sufficient to show that when $x > \theta$, $R_n(x) > 0$ and when $x < \theta$, $R_n(x) < 0$ (for $x \in (0, 1)$).

Given that $E_n > 0$, it suffices to show that the following expression satisfies the property

above:

$$
\begin{aligned}
&-\frac{1}{1-x}[(\frac{c_1}{(c_1x+c_2(1-x))}-1)(1-(\theta d_1+(1-\theta)d_2))+(\frac{d_1}{(d_1x+d_2(1-x))}-1)(\theta d_1+(1-\theta)d_2)]\\
&=-\frac{1}{1-x}[\frac{(c_1-c_2)(1-x)}{(c_1x+c_2(1-x))})(1-(\theta d_1+(1-\theta)d_2))+(\frac{(d_1-d_2)(1-x)}{(d_1x+d_2(1-x))})(\theta d_1+(1-\theta)d_2)]\\
&=-[\frac{(c_1-c_2)}{(c_1x+c_2(1-x))})(1-(\theta d_1+(1-\theta)d_2))+(\frac{(d_1-d_2)}{(d_1x+d_2(1-x))})(\theta d_1+(1-\theta)d_2)]\\
&=-(c_1-c_2)[\frac{(1-(\theta d_1+(1-\theta)d_2))}{(c_1x+c_2(1-x))})-(\frac{(\theta d_1+(1-\theta)d_2)}{(d_1x+d_2(1-x))})]\\
&=-(c_1-c_2)[\frac{(1-(\theta d_1+(1-\theta)d_2))}{1-(d_1x+d_2(1-x))})-(\frac{(\theta d_1+(1-\theta)d_2)}{(d_1x+d_2(1-x))})]\\
&=-(c_1-c_2)\frac{(x-\theta)(d_1-d_2)}{(1-(d_1x+d_2(1-x)))((d_1x+d_2(1-x)))}\\
&=\frac{(x-\theta)(d_1-d_2)^2}{(1-(d_1x+d_2(1-x)))((d_1x+d_2(1-x)))}
\end{aligned}
$$

Given our assumption that $d_1 \neq d_2$, the last expression suggests that for $x \in (0,1)$ when $x > \theta$, then $R_n(x) > 0$, and when $x < \theta$, then $R_n(x) < 0$.

2.5.4 Proof of Assumption 3

Since the bias term is identically 0, it is trivial to show the second inequality of Assumption 3. For the first one, we have:

$$
\begin{aligned}
&Q_n(X_{n+1}, E_n, \frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n}, \frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n+(1-c_2)\beta_n}, x)\\
&=-[E_n(\frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n}-x)(1-X_{n+1})+E_n(\frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n+(1-c_2)\beta_n}-x)X_{n+1}].
\end{aligned}
$$

We can show that E_n is upper bounded by $\frac{1}{2}$ for any n as follows:

$$
E_n = \frac{\sigma_n^2\tau_n^2}{\tau_n+1} = \frac{\alpha_n\beta_n}{(\alpha_n+\beta_n+1)^2} \leq \frac{\alpha_n\beta_n}{2\alpha_n\beta_n} = \frac{1}{2}.
$$

Furthermore, given $x, a_{n,0}, a_{n,1} \in (0,1) \forall n$, it is easy to see that the terms

$$\sup_{n\in\mathbb{N}} |Q_n(X_{n+1}, E_n, \frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n}, \frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n}, x)|$$

and

$$\sup_{n\in\mathbb{N}} \{Q_n^2(X_{n+1}, E_n, \frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n}, \frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n}, x)\}$$

are upper bounded.

Consequently, there exists a positive constant C_1 such that:

$$\sup_{n\in\mathbb{N}} \mathbb{E}[Q_n^2(X_{n+1}, E_n, \frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n}, \frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n}, x)|\mathcal{F}_n] \leq C_1.$$

2.5.5 Proof of Assumption 4

We only consider the case where the observed term is $X_{n+1} = 0$. The expression when $X_{n+1} = 1$ can be tackled in a similar manner.

By using Eq. (2.11) and (2.14) to replace τ_{n+1} and σ_{n+1}^2, and doing the tedious calculations, we get the following:

$$\begin{aligned} \frac{1}{\lambda_{n+1}} - \frac{1}{\lambda_n} =& \sigma_{n+1}^2\tau_{n+1}^2 - \sigma_n^2\tau_n^2 \\ =& S_{n,1} + S_{n,2} + S_{n,3} \end{aligned} \tag{2.15}$$

where

$$\begin{aligned} S_{n,1} &= \frac{\mu_n(1-\mu_n)}{\tau_n+1}\left(\frac{A_{n,0}(\tau_n+1) - \tau_n(\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n})B_{n,0}}{B_{n,0}(\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n})}\right)^2 \\ S_{n,2} &= 2\frac{\mu_n(1-\mu_n)}{\tau_n+1}\tau_n\frac{A_{n,0}(\tau_n+1) - \tau_n(\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n})B_{n,0}}{B_{n,0}(\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n})} \\ S_{n,3} &= \left[\frac{1}{(\tau_n+2)}[(\tau_n+1)(\mu_n\tau_n+1)(\mu_n\tau_n + 2\frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n})\right. \\ &\quad \left. - (\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n})^2(\tau_n+2) - \mu_n(1-\mu_n)(\tau_n+1)(\tau_n+2)\right] \\ &\quad \cdot \left(\frac{A_{n,0}}{B_{n,0}(\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n+c_2\beta_n})}\right)^2. \end{aligned}$$

To prove Assumption 4, it suffices to show that Equation 2.15 has a positive upper bound and positive lower bound. Indeed, if there exist positive constants $\gamma_*, \gamma^* > 0$ such that for all n

$$\gamma_* \leq \frac{1}{\lambda_{n+1}} - \frac{1}{\lambda_n} \leq \gamma^*,$$

then we must have by [16] that

$$\frac{1}{\lambda_0} + n\gamma_* \leq \frac{1}{\lambda_n} \leq \frac{1}{\lambda_0} + n\gamma^*.$$

The last inequality, in turn, implies that [16]

$$\sum_{n=0}^{\infty} \lambda_n = \infty, \sum_{n=0}^{\infty} \lambda_n^2 < \infty,$$

which is what we want to show.

Positive Upper and Lower Bound

If we add the three terms $S_{n,1}, S_{n,2}, S_{n,3}$ in Eq. (2.15), it is simple to see that the denominator is positive since $\tau_n \in (0, \infty), \mu_n \in (0,1) \wedge a_{n,0} \in (0,1) \forall n$. Furthermore, if τ_n approaches 0, it is simple to show that the denominator is lower bounded by a positive constant assuming μ_n is projected to a closed interval with a projection operator Π_H, as explained in Assumption 1.

Next, we show that the following expression corresponding to the numerator of the sum is always positive:

$$\begin{aligned}
&\mu_n(1-\mu_n)(A_{n,0}(\tau_n+1) \\
&- \tau_n(\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n})B_{n,0})^2(\tau_n+2) + 2\mu_n(1-\mu_n)\tau_n(A_{n,0}(\tau_n+2) \\
&- \tau_n(\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n})B_{n,0})(\tau_n+2)(\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n})B_{n,0}) \\
&+ [(3\mu_n^2 - (2 + 2\frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n})\mu_n + 2\frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n} - \frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n}^2)\tau_n \\
&+ (2\mu_n^2 - 2\mu_n - 2\frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n}^2 + 2\frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n})]A_{n,0}^2(\tau_n+1).
\end{aligned} \tag{2.16}$$

Equation (2.16) can be viewed as a function of 3 free variables: $\mu_n, \frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n}, \tau_n$. For

simplicity, we let $x := \mu_n, y := \frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n}, z := \tau_n$. Then it suffices to show that (2.16) is positive when $x \in (0,1), y \in (0,1), z \in (0,\infty)$. Note also that $x \neq y$, because we have assumed that $c_1 \neq c_2$.

We can factorize (2.16) as follows:

$$\begin{aligned}
&z^2 * (x * z + 1) * (x * z - z - 1) \\
&\quad * (x - y)^2 \\
&* (x^2 * z^2 + 2 * x * y * z - x * z^2 + y^2 * z - x * z + 2 * y^2 - 2 * y * z - 2 * y) * (x * z + y)^2.
\end{aligned}$$

We can immediately see that the terms $z^2, (x * z + 1), (x - y)^2, (x * z + y)^2$ are positive. Hence it remains to show that the term $(x * z - z - 1)(x^2 * z^2 + 2 * x * y * z - x * z^2 + y^2 * z - x * z + 2 * y^2 - 2 * y * z - 2 * y)$ is also positive.

Given $x \in (0,1), z > 0$, we have that

$$xz - z - 1 < z - z - 1 < 0.$$

It thus remains to show that $(x^2 * z^2 + 2 * x * y * z - x * z^2 + y^2 * z - x * z + 2 * y^2 - 2 * y * z - 2 * y)$ is negative. Let's rewrite this expression in terms of z:

$$\begin{aligned}
&x^2 * z^2 + 2 * x * y * z - x * z^2 + y^2 * z - x * z + 2 * y^2 - 2 * y * z - 2 * y \\
&= (x^2 - x) * z^2 + (y^2 + (2 * x - 2) * y - x) * z + 2 * y^2 - 2 * y.
\end{aligned}$$

This is a quadratic function of z, where $z \in (0,\infty)$. Because $x, y \in (0,1)$, it is easy to verify that the coefficients of degree 2 and degree 0 are negative. We can additionally show that the coefficient of degree 1 is negative as follows:

$$\begin{aligned}
&y^2 + (2x - 2)y - x \\
&< y + (2x - 2)y - x \\
&= y + 2xy - 2y - x \\
&= 2xy - (x + y) \\
&< 2xy - (x^2 + y^2) \\
&= -(x - y)^2 \\
&< 0.
\end{aligned}$$

We have thus established that the sum $S_{n,1} + S_{n,2} + S_{n,3}$ is always positive. Next, we discuss why it is also bounded above and below by positive constants.

In this direction, we first notice that both the numerator and the denominator can be viewed as polynomials of τ_n (or, z), with coefficients that are functions of μ_n (i.e., x) and $a_{n,0}$ (i.e., y). Furthermore, assuming we do BMM with a projection operator Π_H that projects μ_n into a suitable closed interval $[M_{low}, M_{high}] \subset [0,1]$, where $0 < M_{low}, M_{high} < 1$ and $\theta \in H$, we can see that the numerator can only be 0 if τ_n is 0. Since we showed that Eq. (2.15) is positive for $\tau \in (0, \infty)$, in order to show a lower and upper bound it suffices to investigate the cases where $\tau_n \to 0$ or $\tau_n \to \infty$.

We first show that, with probability 1, we cannot have that that $\tau_n \to 0$. From Eq. (2.11) we get:

$$\tau_{n+1} = \frac{A_{n,0}(\tau_n + 1)}{B_{n,0}(\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n})}(1 - X_{n+1}) + \frac{A_{n,1}(\tau_n + 1)}{B_{n,1}(\mu_n\tau_n + \frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n})}(X_{n+1}). \quad (2.17)$$

Assuming that $\tau_n \approx 0$, we get the following approximation in the limit after we substitute $A_{n,0}, A_{n,1}, B_{n,0}, B_{n,1}$ by (2.13):

$$\frac{A_{n,0}(\tau_n + 1)}{B_{n,0}(\mu_n\tau_n + \frac{c_1\alpha_n}{c_1\alpha_n + c_2\beta_n})} \to \frac{1}{2}(\frac{\mu_n}{a_{n,0}} + \frac{1 - \mu_n}{1 - a_{n,0}}) \cdot \tau_n, \quad (2.18)$$

$$\frac{A_{n,1}(\tau_n + 1)}{B_{n,1}(\mu_n\tau_n + \frac{(1-c_1)\alpha_n}{(1-c_1)\alpha_n + (1-c_2)\beta_n})} \to \frac{1}{2}(\frac{\mu_n}{a_{n,1}} + \frac{1 - \mu_n}{1 - a_{n,1}}) \cdot \tau_n. \quad (2.19)$$

Based on (2.17), (2.18) and (2.19), we then get for $\tau_n \approx 0$:

$$\mathbb{E}[\tau_{n+1}|\mathcal{F}_n] \approx \frac{1}{2} \cdot \Big((c_1\theta + c_2(1-\theta))(\frac{\mu_n}{a_{n,0}} + \frac{1 - \mu_n}{1 - a_{n,0}}) + (d_1\theta + d_2(1-\theta))(\frac{\mu_n}{a_{n,1}} + \frac{1 - \mu_n}{1 - a_{n,1}})\Big) \cdot \tau_n. \quad (2.20)$$

It holds that $d_1\theta + d_2(1-\theta) = (1-c_1)\theta + (1-c_2)(1-\theta) = 1 - (c_1\theta + c_2(1-\theta))$. Without loss of generality, let's assume that $c_1 < c_2$. Given $\theta \in (0,1)$, we then trivially get for the terms $r = c_1\theta + c_2(1-\theta)$ and $1 - r = d_1\theta + d_2(1-\theta)$:

$$c_1 < r < c_2 \wedge 1 - c_2 < 1 - r < 1 - c_1. \quad (2.21)$$

We then have for the term on the right hand side of Eq. (2.20):

$$\frac{1}{2}\cdot\Big(r(\frac{\mu_n}{a_{n,0}}+\frac{1-\mu_n}{1-a_{n,0}})+(1-r)(\frac{\mu_n}{a_{n,1}}+\frac{1-\mu_n}{1-a_{n,1}})\Big) \tag{2.22}$$

$$=\frac{1}{2}\cdot\Big(r\Big(\frac{\frac{1}{1+\nu}}{\frac{1}{1+\frac{c_2}{c_1}\nu}}+\frac{\frac{\nu}{1+\nu}}{\frac{\nu}{\nu+\frac{c_1}{c_2}}}\Big)+(1-r)\Big(\frac{\frac{1}{1+\nu}}{\frac{1}{1+\frac{1-c_2}{1-c_1}\nu}}+\frac{\frac{\nu}{1+\nu}}{\frac{\nu}{\nu+\frac{1-c_1}{1-c_2}}}\Big)\Big)$$

$$=\frac{1}{2}\cdot\Big(r\Big(\frac{1+\frac{c_2}{c_1}\nu}{1+\nu}+\frac{\nu+\frac{c_1}{c_2}}{1+\nu}\Big)+(1-r)\Big(\frac{1+\frac{1-c_2}{1-c_1}\nu}{1+\nu}+\frac{\nu+\frac{1-c_1}{1-c_2}}{1+\nu}\Big)\Big),\ \text{where } \nu=\frac{\beta_n}{\alpha_n}\in(0,\infty). \tag{2.23}$$

We can now show that the term in (2.22) is greater than 1:

$$\frac{1}{2}\cdot\Big(r\Big(\frac{1+\frac{c_2}{c_1}\nu}{1+\nu}+\frac{\nu+\frac{c_1}{c_2}}{1+\nu}\Big)+(1-r)\Big(\frac{1+\frac{1-c_2}{1-c_1}\nu}{1+\nu}+\frac{\nu+\frac{1-c_1}{1-c_2}}{1+\nu}\Big)\Big)>1\Leftrightarrow$$

$$r\Big(1+\frac{c_2}{c_1}\nu+\nu+\frac{c_1}{c_2}\Big)+(1-r)\Big(1+\frac{1-c_2}{1-c_1}\nu+\nu+\frac{1-c_1}{1-c_2}\Big)>2\cdot(1+\nu)\Leftrightarrow$$

$$r\Big(\frac{c_2}{c_1}\nu+\frac{c_1}{c_2}\Big)+(1-r)\Big(\frac{1-c_2}{1-c_1}\nu+\frac{1-c_1}{1-c_2}\Big)>1+\nu\Leftrightarrow$$

$$r\frac{c_1}{c_2}+(1-r)\frac{1-c_1}{1-c_2}+\Big(r\frac{c_2}{c_1}+(1-r)\frac{1-c_2}{1-c_1}\Big)\cdot\nu>1+\nu.$$

But the last inequality is true, since $r\frac{c_1}{c_2}+(1-r)\frac{1-c_1}{1-c_2}>1$ and $r\frac{c_2}{c_1}+(1-r)\frac{1-c_2}{1-c_1}>1$; this is easy to show given $c_1<r<c_2$ from our original assumption. In fact, both terms can be bounded away from 1, given c_1,c_2,r are distinct (and fixed). As a result of this, Eq. (2.20) gives for $\tau_n\approx 0$:

$$\mathbb{E}[\tau_{n+1}|\mathcal{F}_n]>\tau_n. \tag{2.24}$$

The variance will also be finite, because if we assume that $\tau_n\to 0$, then τ_n must have an upper bound. Furthermore, $\tau_n\to 0$ implies that there exists a positive constant K such that $|\tau_{n+1}-\tau_n|\leq K,\forall n$. However, with these assumptions standard martingale theory suggests that, with probability 1, τ_n does not converge to a zero limit [30], which is a contradiction. Indeed, by Doob's decomposition theorem, due to Eq. (2.24) τ can be decomposed into a martingale M and an integrable predictable process A with $A_0=0$ that is almost surely increasing [30]. Since M converges to 0 almost surely by the martingale central limit theorem and A is strictly increasing almost surely, it is straightforward to see that, with probability 1, τ_n does not converge to a 0 limit.

Finally, we examine the case where $\tau_n\to\infty$. In this direction, we observe that in Eq.

(2.15) if we expand the sum $S_{n,1} + S_{n,2} + S_{n,3}$, both the numerator and the denominator have the same degree. Given the leading coefficients are positive as well as lower and upper bounded since we use a projection operator Π_H, we conclude that the sum must also be positive as well as upper and lower bounded as $\tau_n \to \infty$.

This concludes our proof that $\mu_n \to \hat{\theta}$ almost surely in the naïve Bayes setting.

Chapter 3

On the Bayesian Network: Heuristics based on Bayesian learning to Initialize SAT Solvers

In this chapter, we model solving the Boolean satisfiability (SAT) instances as learning Bayesian networks using Bayesian Moment Matching (BMM) and show how this perspective can help initialize SAT solvers. Section 3.1 reviews basic concepts of SAT and SAT solvers, in addition to their connections to machine learning. Section 3.2 introduces the initialization problem of conflict-driven-clause-learning (CDCL) SAT solvers. Section 3.3 illustrates our BMM framework on a small SAT instance. Section 3.4 derives the algorithm for a general case. Section 3.2 discusses how to integrate our BMM initializer into CDCL solvers. Section 3.6 presents results of experiments.

3.1 SAT, SAT solvers and Machine Learning

3.1.1 Boolean Satisfiability Problems

The Boolean satisfiability problem (SAT) is the decision problem where we ask if there exists an assignment to each variable such that the Boolean formula evaluates to True (T). Below we review some important terminologies of SAT.

Definition 1 ***Literal:*** *A literal is either a (Boolean) variable (called positive literals) or the negation of a variable (called negative literals).*

Definition 2 ***Clause:*** *A clause is a disjunction of literals, say* $C : a_1 \vee \cdots \vee a_n$*, where* $a_1 \vee \cdots \vee a_n$ *are all literals.*

In this chapter, we only study SAT formulas in conjunctive normal forms (CNF). A SAT formula is in CNF if it is a conjunction of clauses or a single clause [18]. 3.1 is an example of SAT instances in CNF forms, where C_1, C_2, C_3, C_4 are clauses and x, y, z are variables.

$$\begin{aligned} & C_1 : x \vee y \vee \neg z \\ \wedge\ & C_2 : x \vee y \vee z \\ \wedge\ & C_3 : x \vee \neg y \vee z \\ \wedge\ & C_4 : \neg x \vee \neg y \vee \neg z \end{aligned} \tag{3.1}$$

SAT is the first problem proven to be NP-complete by Cook and Levin [17] [45]. This implies that all problems falling in the NP class, such as the travelling salesman problem and the graph coloring problem, are at most as hard as SAT [42]. There are no known polynomial-time algorithms for general SAT instances, and whether such algorithms exist belongs to the debate of P vs NP, one of the most famous open problems in computer science.

Connections to Machine Learning

SAT occurs naturally in many areas of computer science, such as cryptography [61], formal verification [7] and bioinformatics [53]. However, it may not seem immediately obvious what is the connection between SAT and modern machine learning. In fact, there are two-way connections between SAT and ML: some ML tasks can be encoded into the form of SAT and solved efficiently by SAT solvers, while some components of traditional SAT solvers can also be improved with the help of ML.

Later sections will focus on how ML can be used to improve SAT solvers. Below, we present some examples of reframing ML problems in the form of SAT:

- *SAT and supervised learning.* Supervised learning fits a function on training data. There is a clear connection between function fitting and constraint satisfaction: each labelled data can be seen as one constraint on the function's parameter. Mezard and Mora [59] show that learning binary neural networks can be encoded as a SAT

problem. Narodytska et al. [65] developed a SAT-based algorithm to construct the smallest decision tree on a given dataset.

- *SAT and causality.* Ibrahim et al. [38] proposed a novel approach to check the assumption of causality in acyclic binary models using SAT. This method is very efficient and can scale to large models due to the recent progress of SAT solvers. Another line of work is to use SAT in causal discovery by encoding all the available information in a causal graph as constraints in propositional logic [37] [82] [24].

- *SAT and fairness, robustness.* Narodytska et al. [66] developed a rigorous way to verify properties of robustness of neural networks by encoding them into Boolean representations. Ghosh et al. [26] proposed a stochastic SAT framework that formally verifies multiple fairness notions of machine learning algorithms.

3.1.2 SAT Solvers

Despite the NP-completeness nature of SAT, researchers have developed scalable SAT solvers for instances involving tens of millions of variables and clauses by utilizing the special logic structure in the problem. Most state-of-the-art SAT solvers are based on the same paradigm, conflict-driven-clause-learning (CDCL) [58] [62].

The CDCL algorithm is based on the celebrated Davis–Putnam–Logemann–Loveland (DPLL) algorithm [19]. DPLL is a backtracking algorithm making use of unit propagation. At each branching step, a variable will be assigned to either T or F. Whenever a conflict is found, backtracking is executed to undo branching steps until an unflipped branch is reached [7].

CDCL solvers improve upon the DPLL algorithm by introducing clause learning. When a conflict is found, CDCL solvers look at the guess made and its implications in an implication graph. Using this graph, CDCL is able to learn a new clause that helps the solver avoid the same mistakes in the future. This newly learned clause will then be added to the instance for future search [7].

Connections to Machine Learning

Recently, there have been many attempts to introduce ML algorithms to SAT solvers, which can be roughly divided into two categories:

- *ML to assist existing SAT solvers:* This line of work focuses on improving one or more components of CDCL SAT solvers using ML algorithms. Haim and Walsh [29] trained a machine-learning-based satisfiability classifier and runtime prediction model to help choose the best restart strategies for CDCL solvers. Liang et al. [47] modeled variable selection in branching heuristics as reinforcement learning problems. Xu et al. [89] proposed an ML approach to construct algorithm portfolios for each SAT instance to help select the best solver.

- *ML as standalone SAT solvers:* With recent advances in deep learning, some researchers proposed end-to-end ML systems to solve SAT problems. NeuroSAT [78] trains a graph neural network on the graph representation of a SAT instance to predict its satisfiability and variable assignment. SATNet [85] integrates the semidefinite program of MAXSAT into end-to-end differentiable learning systems. Amizadeh et al. [2] developed a graph embedding architecture to extract representations for SAT instances and then trained the model using policy gradient methods. Even though the empirical performance of these ML-based SAT solvers is not on par with CDCL solvers on large instances, they provide fresh perspectives to look at SAT solvers.

3.2 Motivation

There has been lots of progress in the SAT community to improve different components of CDCL solvers, such as branching heuristics [47], restarting [49] and clause learning [4]. However, few studies have focused on how to initialize the search of CDCL solvers, even though solver developers have known for a long time that the initialization can have a significant impact on the performance of CDCL SAT solvers.

The initialization problem for SAT solvers can be defined as follows: given a SAT formula ϕ, compute an **initial order** over the variables and **initial value** for each of them. More specifically, *initial order* means a total order over variables chosen by the CDCL solver S, and by *initial value* we mean a mapping from variables to assignments at the beginning of its search.

In the following sections, we will introduce an initialization algorithm to CDCL solvers based on BMM.

3.3 Bayesian Formulation

We introduce a novel Bayesian perspective to look at SAT, with the goal of finding an assignment that satisfies as many clauses as possible.

As shown in Figure 3.1, in our Bayesian formulation, each variable in the SAT formula is seen as a Bernoulli random variable with an unknown probability θ_i being assigned to T and each clause is treated as evidence for the variables. The main idea is to update our beliefs about θ_i after "observing" each clause, hoping to improve the likelihood of the clause being satisfied.

We shall illustrate our framework using the toy instance 3.1. We use $\theta_x, \theta_y, \theta_z$ to denote $P(x = T), P(y = T), P(z = T)$ respectively:

$$\begin{aligned}\theta_x &:= P(x = T)\\ \theta_y &:= p(y = T)\\ \theta_z &:= P(z = T)\end{aligned}$$

The goal is to infer the values of $\theta_x, \theta_y, \theta_z$ using the information provided by C_1, C_2, C_3, C_4 sequentially.

To learn $\theta_x, \theta_y, \theta_z$ by Bayesian inference, we assume that each of them is a random variable initially distributed according to a beta distribution and that they are mutually independent. Concretely, the prior for the joint distribution is:

$$P(\Theta_x, \Theta_y, \Theta_z) = \prod_{i=x,y,z} Beta(\theta_i; \alpha_i, \beta_i).$$

Suppose our first observation is C_1. We want to update our beliefs about $\theta_x, \theta_y, \theta_z$ by computing the posterior $P(\Theta_x, \Theta_y, \Theta_z | C_1)$

To satisfy a clause, at least one of the literals needs to be satisfied, which can be done in many different ways if we have many literals in a clause. However, there is only one way to falsify the clause. Therefore, we compute the likelihood function as the complement probability of falsifying the observed clause. For example, the only possible assignment to falsify C_1 is:

$$x := F, y := F, z := T$$

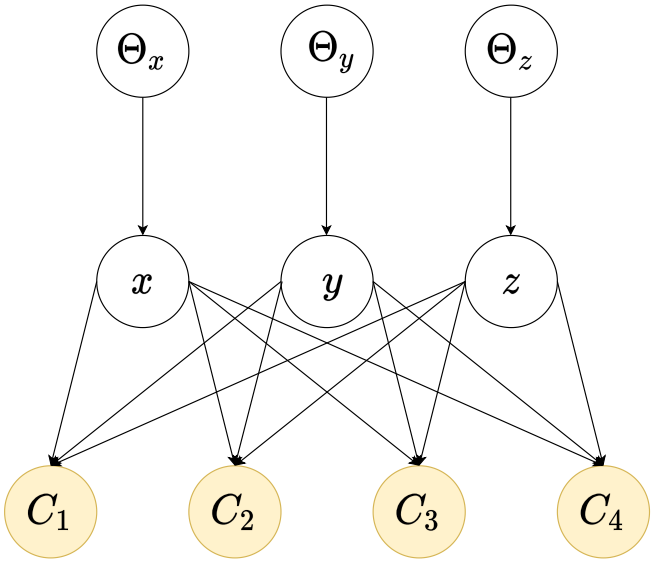

Figure 3.1: We formulate SAT as a Bayesian network, where each variable is hidden and each clause is observable. The conditional distributions associated with edges are fully known based on the logic structure of a clause. The goal is to infer $\Theta_x, \Theta_y, \Theta_z$ from C_1, C_2, C_3, C_4. This model is an extension of the naïve Bayesian model we studied in Figure 2.1.

Therefore, the likelihood of C_1 being satisfied can be computed as:

$$\begin{aligned} P(C_1|\Theta_x, \Theta_y, \Theta_z) &= 1 - P(\neg C_1|\Theta_x, \Theta_y, \Theta_z) \\ &= 1 - P(x = F, y = F, z = T|\Theta_x, \Theta_y, \Theta_z) \\ &= 1 - (1 - \Theta_x) \cdot (1 - \Theta_y) \cdot \Theta_z \end{aligned} \tag{3.2}$$

The posterior after seeing the first clause C_1 is:

$$\begin{aligned} P(\Theta_x, \Theta_y, \Theta_z|C_1) &\propto P(\Theta_x, \Theta_y, \Theta_z)P(C_1|\Theta_x, \Theta_y, \Theta_z) \\ &\propto P(\Theta_x, \Theta_y, \Theta_z)[1 - (1 - \Theta_x)(1 - \Theta_y)\Theta_z] \\ &\propto P(\Theta_x, \Theta_y, \Theta_z) - (1 - \Theta_x)(1 - \Theta_y)\Theta_z P(\Theta_x, \Theta_y, \Theta_z) \end{aligned} \tag{3.3}$$

Because of the conjugate properties (shown in Equation 2.2), Equation 3.3 can be further written as:

$$\begin{aligned} P(\Theta_x, \Theta_y, \Theta_z|C_1) \propto\ & Beta(\theta_x; \alpha_x, \beta_x) \cdot Beta(\theta_y; \alpha_y, \beta_y) \cdot Beta(\theta_z; \alpha_z, \beta_z) \\ & - \frac{\beta_x}{\alpha_x + \beta_x} \frac{\beta_y}{\alpha_y + \beta_y} \frac{\alpha_z}{\alpha_z + \beta_z} Beta(\theta_x; \alpha_x, \beta_x + 1) \\ & \cdot Beta(\theta_y; \alpha_y, \beta_y + 1) \cdot Beta(\theta_z; \alpha_z + 1, \beta_z) \end{aligned} \tag{3.4}$$

We can also rewrite the likelihood $P(C_1|\Theta_x, \Theta_y, \Theta_z)$ in Equation 3.4 as the sum of joint probabilities of all the possible assignments to satisfy C_1:

$$\begin{aligned} P(C_1|\Theta_x, \Theta_y, \Theta_z) = {} & \Theta_x\Theta_y\Theta_z + \Theta_x\Theta_y(1 - \Theta_z) + \Theta_x(1 - \Theta_y)\Theta_z + \Theta_x(1 - \Theta_y)(1 - \Theta_z) \\ & + (1 - \Theta_x)\Theta_y\Theta_z + (1 - \Theta_x)\Theta_y(1 - \Theta_z) + (1 - \Theta_x)(1 - \Theta_y)(1 - \Theta_z) \end{aligned} \tag{3.5}$$

Equation 3.5 suggests that the posterior is a mixture of products of Beta distributions. Therefore, an issue similar to the one in Chapter 2 will arise as more clauses are encountered: exponential growth of the number of mixture components in the posterior. To solve this intractability issue, we approximate the true mixture $P(\Theta_x, \Theta_y, \Theta_z|C_1)$ using a product of Beta distributions $\tilde{P}(\tilde{\Theta}_x, \tilde{\Theta}_y, \tilde{\Theta}_z)$:

$$\tilde{P}(\tilde{\Theta}_x, \tilde{\Theta}_y, \tilde{\Theta}_z) = \prod_{i=x,y,z} Beta(\tilde{\theta}_i; \tilde{\alpha}_i, \tilde{\beta}_i)$$

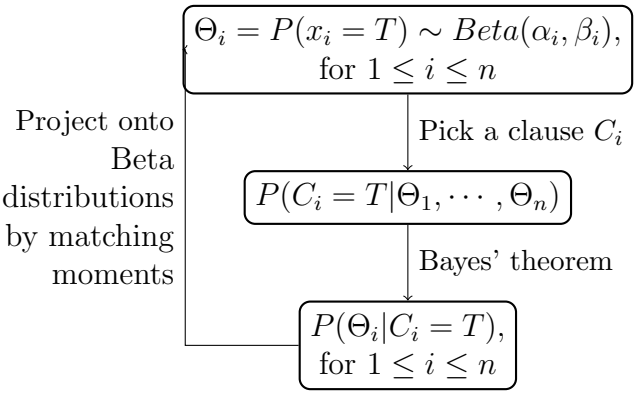

Figure 3.2: A Beta prior is assigned to each variable in the beginning. The posteriors are then calculated each time when encountering a new clause. We project the posteriors back to Beta distributions using BMM, which serves as priors for the next clause.

The parameters $\tilde{\alpha}_x, \tilde{\beta}_x, \tilde{\alpha}_y, \tilde{\beta}_y, \tilde{\alpha}_z, \tilde{\beta}_z$ are then computed by matching the first and second moments of the marginal distribution of the posterior:

$$\begin{aligned}
\mathbb{E}_{\tilde{\Theta}_x \sim Beta(\tilde{\theta}_x;\tilde{\alpha}_x,\tilde{\beta}_x)}[\tilde{\Theta}_x] &:= \mathbb{E}_{\Theta_x \sim P(\Theta_x|C_1)}[\Theta_x] \\
\mathbb{E}_{\tilde{\Theta}_x \sim Beta(\tilde{\theta}_x;\tilde{\alpha}_x,\tilde{\beta}_x)}[\tilde{\Theta}_x^2] &:= \mathbb{E}_{\Theta_x \sim P(\Theta_x|C_1)}[\Theta_x^2] \\
\mathbb{E}_{\tilde{\Theta}_y \sim Beta(\tilde{\theta}_y;\tilde{\alpha}_y,\tilde{\beta}_y)}[\tilde{\Theta}_y] &:= \mathbb{E}_{\Theta_y \sim P(\Theta_y|C_1)}[\Theta_y] \\
\mathbb{E}_{\tilde{\Theta}_y \sim Beta(\tilde{\theta}_y;\tilde{\alpha}_y,\tilde{\beta}_y)}[\tilde{\Theta}_y^2] &:= \mathbb{E}_{\Theta_y \sim P(\Theta_y|C_1)}[\Theta_y^2] \\
\mathbb{E}_{\tilde{\Theta}_z \sim Beta(\tilde{\theta}_z;\tilde{\alpha}_z,\tilde{\beta}_z)}[\tilde{\Theta}_z] &:= \mathbb{E}_{\Theta_z \sim P(\Theta_z|C_1)}[\Theta_z] \\
\mathbb{E}_{\tilde{\Theta}_z \sim Beta(\tilde{\theta}_z;\tilde{\alpha}_z,\tilde{\beta}_z)}[\tilde{\Theta}_z^2] &:= \mathbb{E}_{\Theta_z \sim P(\Theta_x|C_1)}[\Theta_z^2]
\end{aligned} \tag{3.6}$$

It can be easily shown that each marginal posterior follows a mixture of Beta distributions. Therefore, similar to Equation 2.3, Equation 3.6 is a system of linear equations with a closed-form solution.

Subsequently, $\tilde{P}(\tilde{\Theta}_x, \tilde{\Theta}_y, \tilde{\Theta}_z)$ is used as the prior when C_2 is observed. During one epoch, the above update is repeated once for each clause. Figure 3.2 presents a road map of our Bayesian framework.

I did a numerical experiment to show the effectiveness of our framework in the toy example 3.1. Fig. 3.3 presents how the distributions for $\Theta_x, \Theta_y, \Theta_z$ are updated with more epochs. We can see that eventually the posterior of each variable converges to a solution that satisfies all of the clauses.

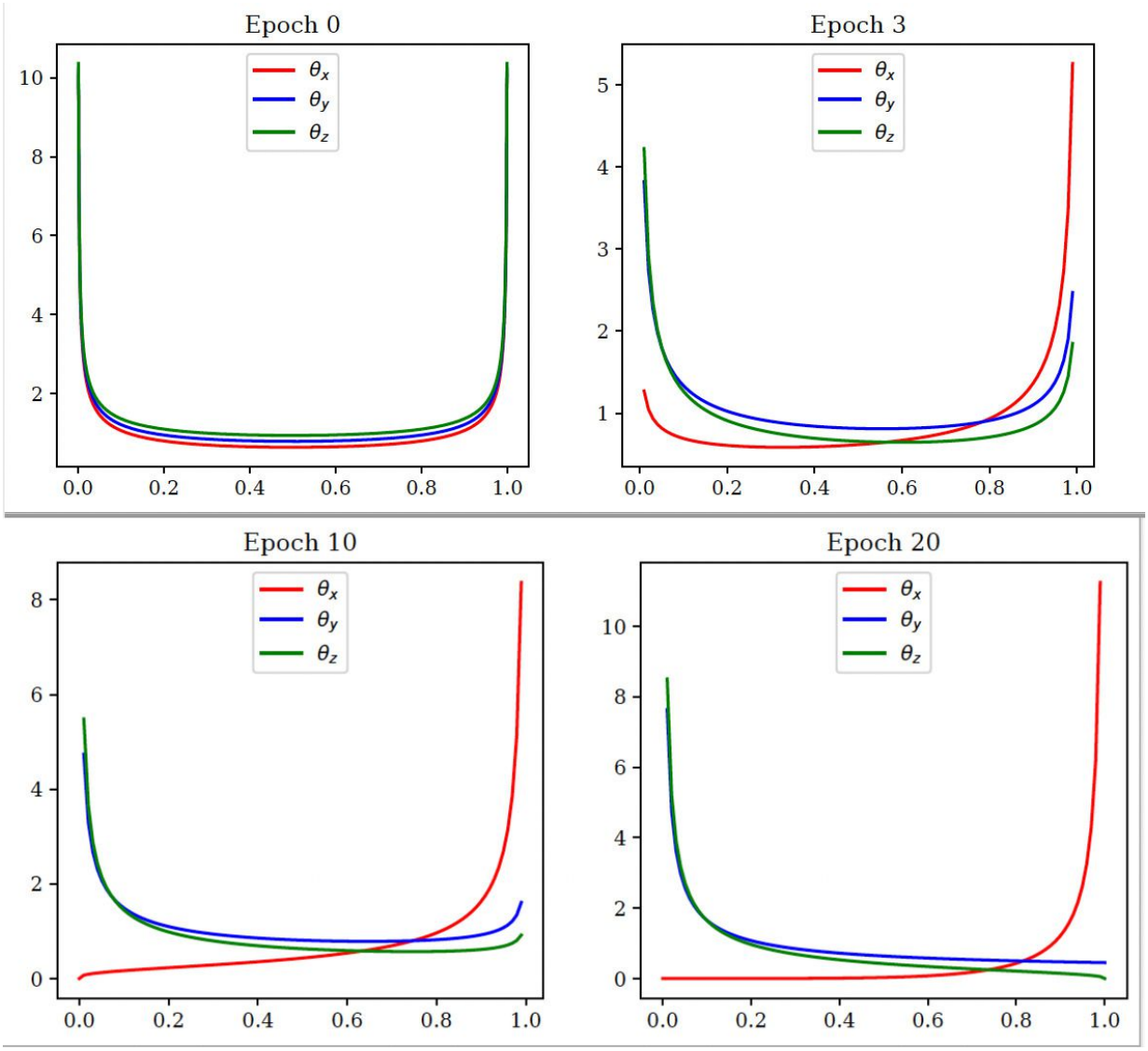

Figure 3.3: The probability density functions of θ_x, θ_y and θ_z for the example in section 3 using BMM update. As the number of epochs increases, the densities of θ_y and θ_z are shifting towards 0 while θ_x is shifting towards 1. This suggests an assignment $x := T, y := F, z := F$, which satisfies all clauses

3.4 Algorithms

In this section, we derive the posterior distribution for a general SAT clause and then present the pseudocode of our algorithm.

We consider a clause C, which is a disjunction of m literals. Without loss of generality, we assume that all positive literals appear before negative literals in C, and there are h $(0 \leq h \leq m)$ positive literals:

$$C = (\bigvee_{0 \leq i < h} l_i) \vee (\bigvee_{h \leq j < m} \neg l_j).$$

Our goal is to infer the value of the random vector $\mathbf{\Theta}$, whose element represents the probabilities of each literal being true:

$$\mathbf{\Theta} = \{\theta_k : 0 \leq k < m, \theta_k = P(l_k = T)\}.$$

We assign a factorized beta distributions as the prior for $\mathbf{\Theta}$:

$$P(\mathbf{\Theta}) = \prod_{0 \leq k < m} Beta(\theta_k; \alpha_k, \beta_k)$$

The posterior after observing clause C can be computed as:

$$\begin{aligned} P(\mathbf{\Theta}|C) &= \frac{1}{P(C)}(P(\mathbf{\Theta})P(C|\mathbf{\Theta})) \\ &= \frac{1}{P(C)}[\prod_{0 \leq k < m} Beta(\theta_k; \alpha_k, \beta_k)(1 - \prod_{0 \leq i < h}(1 - \theta_i) \prod_{h \leq j < m} \theta_j)] \\ &= \frac{1}{P(C)}[\prod_{0 \leq k < m} Beta(\theta_k; \alpha_k, \beta_k) - \prod_{0 \leq i < h} Beta(\theta_i; \alpha_i, \beta_i)(1 - \theta_i) \prod_{h \leq j < m} Beta(\theta_j; \alpha_j, \beta_j)\theta_j] \end{aligned} \tag{3.7}$$

Due to conjugate properties, Equation 3.7 can be further expanded as:

$$P(\mathbf{\Theta}|C) = \frac{1}{P(C)}[\prod_{0\leq k<m} Beta(\theta_k;\alpha_k,\beta_k) - \prod_{0\leq i<h}\frac{\beta_i}{\alpha_i+\beta_i}Beta(\theta_i;\alpha_i,\beta_i+1) \cdot \prod_{h\leq j<m}\frac{\alpha_j}{\alpha_j+\beta_j}Beta(\theta_j;\alpha_j+1,\beta_j)] \tag{3.8}$$

Furthermore, we can define the value p as:

$$p := \prod_{0\leq i<h}\frac{\beta_i}{\alpha_i+\beta_i}\prod_{h\leq j<m}\frac{\alpha_j}{\alpha_j+\beta_j} \tag{3.9}$$

Then Equation 3.4 can be simplified as:

$$P(\mathbf{\Theta}|C) = \frac{1}{P(C)}[\prod_{0\leq k<m} Beta(\theta_k;\alpha_k,\beta_k) - p\prod_{0\leq i<h} Beta(\theta_i;\alpha_i,\beta_i+1)\prod_{h\leq j<m} Beta(\theta_j;\alpha_j+1,\beta_j)] \tag{3.10}$$

The denominator can also be expressed in clean forms after plugging in Equation :

$$\begin{aligned} P(C) &= \int_{(0,1)^m} P(\mathbf{\Theta})P(C|\mathbf{\Theta})d\mathbf{\Theta} \\ &= \int_{(0,1)^m} \prod_{0\leq k<m} Beta(\theta_k;\alpha_k,\beta_k) - p\prod_{0\leq i<h} Beta(\theta_i;\alpha_i,\beta_i+1)\prod_{h\leq j<m} Beta(\theta_j;\alpha_j+1,\beta_j)d\mathbf{\Theta} \\ &= 1 - \prod_{0\leq i<h}\frac{\beta_i}{\alpha_i+\beta_i}\prod_{h\leq j<m}\frac{\alpha_j}{\alpha_j+\beta_j}. \end{aligned}$$

Following the notation defined in Equation 3.9, the denominator $P(C)$ can be rewritten as:

$$P(C) = 1 - p$$

We observe that the posterior is a mixture $P(\mathbf{\Theta}|C)$ of products of Beta distributions. The number of mixtures grows exponentially as more clauses are encountered. To address this, we use BMM to approximate the true mixture $P(\mathbf{\Theta}|C)$ by a single product of Beta distributions:

$$\tilde{P}(\tilde{\mathbf{\Theta}}) = \prod_{0\leq k<m} Beta(\tilde{\theta}_k;\tilde{\alpha}_k,\tilde{\beta}_k).$$

The parameters $\tilde{\alpha}_k, \tilde{\beta}_k$ for literal l_k are then computed by matching the first and second moments of the marginal distribution of θ_k (we proceed similarly for other literals):

$$\begin{cases} \mathbb{E}_{\tilde{\theta}_k \sim Beta(\tilde{\theta}_k;\tilde{\alpha}_k,\tilde{\beta}_k)}[\tilde{\theta}_k] = \mathbb{E}_{\theta_k \sim P_{\theta_k}(\theta_k|C)}[\theta_k] \\ \mathbb{E}_{\tilde{\theta}_k \sim Beta(\tilde{\theta}_k;\tilde{\alpha}_k,\tilde{\beta}_k)}[\tilde{\theta}_k^2] = \mathbb{E}_{\theta_k \sim P_{\theta_k}(\theta_k|C)}[\theta_k^2] \end{cases} \iff \begin{cases} \frac{\tilde{\alpha}_k}{\tilde{\alpha}_k+\tilde{\beta}_k} = \mathbb{E}_{\theta_k \sim P_{\theta_k}(\theta_k|C)}[\theta_k] \\ \frac{\tilde{\alpha}_k(\tilde{\alpha}_k+1)}{(\tilde{\alpha}_k+\tilde{\beta}_k)(\tilde{\alpha}_k+\tilde{\beta}_k+1)} = \mathbb{E}_{\theta_k \sim P_{\theta_k}(\theta_k|C)}[\theta_k^2] \end{cases}.$$

In the above expression, we have used the fact that the first moment (mean) of the beta distribution $Beta(\theta;\alpha,\beta)$ is $\frac{\alpha}{\alpha+\beta}$, while its second moment is $\frac{\alpha(\alpha+1)}{(\alpha+\beta)(\alpha+\beta+1)}$ [41]. Thus, this projection procedure is equivalent to solving a system of linear equations. We will discuss how to calculate the right side briefly below.

If the literal l_k is positive in C, then:

$$\begin{aligned} P_{\theta_k}(\theta_k|C) &= \int_{(0,1)^{m-1}} P(\mathbf{\Theta}|C) d\theta_0 ... d\theta_{k-1} d\theta_{k+1} ... d\theta_{m-1} \\ &= \frac{1}{1-p}[Beta(\theta_k;\alpha_k,\beta_k) - p \cdot Beta(\theta_k;\alpha_k,\beta_k+1)]. \end{aligned}$$

If the literal l_k is negative in C, then:

$$\begin{aligned} P_{\theta_k}(\theta_k|C) &= \int_{(0,1)^{m-1}} P(\mathbf{\Theta}|C) d\theta_0 ... d\theta_{k-1} d\theta_{k+1} ... d\theta_{m-1} \\ &= \frac{1}{1-p}[Beta(\theta_k;\alpha_k,\beta_k) - p \cdot Beta(\theta_k;\alpha_k+1,\beta_k)]. \end{aligned}$$

We thus get:

$$\mathbb{E}_{\theta_k \sim P_{\theta_k}(\theta_k|C)}[\theta_k] = \begin{cases} \frac{1}{1-p}(\frac{\alpha_k}{\alpha_k+\beta_k} - p \cdot \frac{\alpha_k}{\alpha_k+\beta_k+1}), & \text{if } l_k \text{ is positive in } C \\ \frac{1}{1-p}(\frac{\alpha_k}{\alpha_k+\beta_k} - p \cdot \frac{\alpha_k+1}{\alpha_k+\beta_k+1}), & \text{if } l_k \text{ is negative in } C. \end{cases}$$

Similarly, we get:

$$\mathbb{E}_{\theta_k \sim P_{\theta_k}(\theta_k|C)}[\theta_k^2] = \begin{cases} \frac{1}{1-p}(\frac{\alpha_k(\alpha_k+1)}{(\alpha_k+\beta_k)(\alpha_k+\beta_k+1)} - p \cdot \frac{\alpha_k(\alpha_k+1)}{(\alpha_k+\beta_k+1)(\alpha_k+\beta_k+2)}), & \text{if } l_k \text{ is positive in } C \\ \frac{1}{1-p}(\frac{\alpha_k(\alpha_k+1)}{(\alpha_k+\beta_k)(\alpha_k+\beta_k+1)} - p \cdot \frac{(\alpha_k+1)(\alpha_k+2)}{(\alpha_k+\beta_k+1)(\alpha_k+\beta_k+2)}), & \text{if } l_k \text{ is negative in } C. \end{cases}$$

Based on the above discussion, the pseudocode of BMM for SAT is given in Algorithm 3.4. The variable $MaxEpochs$ denotes the number of epochs. During one epoch, we visit each clause exactly once. As explained in the main document, we find empirically that as

few as 10 epochs suffice for a good initialization.

3.5 Integration into CDCL Solvers

Look-back branching heuristics [6] are widely used in the modern CDCL SAT solvers. Each variable maintains some scores to help the algorithm to answer two questions: which unassigned variable to pick (Variable order) and what value to assign to that variable (Polarity/Value selection). As described in Section 3.2, an important question is how to set the initial score for variable order and value selection.

Our BMM framework, Algorithm 3.4, finds an assignment satisfying most clauses in a short period of time. We can run this algorithm for several epochs before starting the SAT solvers. The learned BMM posterior distribution for each variable can provide the CDCL SAT solvers with useful information about the *initial order* and *initial value*.

Initial Value: The initial value of a variable is determined by the first moment of the BMM posterior:

$$x = \begin{cases} T & \mathbb{E}(\theta_x) > 0.5 \\ F & \mathbb{E}(\theta_x) \leq 0.5 \end{cases}$$

Initial Order: In CDCL solvers, a score $s(x)$, called activity, is maintained for each variable x. The variable with the highest activity will be picked as the decision variable. Our algorithm assigns higher activities to the variables closer to 0 or 1, which implies we are more confident about the initial assignment. For each variable x, the activity score $s(x)$ is defined to be a number in the range [0, 0.5] as follows:

$$s(x) = \begin{cases} \mathbb{E}(\theta_x), & \mathbb{E}(\theta_x) < 0.5 \\ 1 - \mathbb{E}(\theta_x), & \mathbb{E}(\theta_x) \geq 0.5 \end{cases}$$

3.6 Results of Experiments

In this Section, we present the performance of our BMM initializer with other initialization methods for different types of CDCL solvers over cryptographic benchmarks and application benchmarks. We would like to thank Saeed Nejati for integrating BMM into multiple SAT solvers [68].

3.6.1 Baselines for Initialization Methods

- **Default:** Set the initial polarity to F and the initial activity score to 0 for each variable.
- **Random:** Initial polarities are randomly assigned to be T or F with probabilities 0.5. Initial activity scores are randomly sampled from the uniform distribution $U(0.5, 1)$.
- **Jeroslow-Wang [40]:** This initialization method assigns a higher score to variables appearing in shorter clauses. The intuition is that these variables, when assigned by the solver, create unit clauses sooner than others.
- **Survey Propagation [12]:** A SAT instance can be encoded into a factor graph. Then survey propagation finds the right assignment to each variable iteratively through message passing.

3.6.2 Cryptographic Benchmarks

We used cryptographic instances encoding preimage of round reduced SHA-1 hash function. We encoded 22 rounds of SHA-1 and used 50 randomly generated hash values to be inverted [68]. All jobs were run on Intel Xeon E5-2667 CPUs at 3.20GHz and 8GB of RAM, with a time limit of 4 hours.

We choose 3 top-performing solvers in cryptographic benchmarks as the base: MapleSAT [47], Glucose-4 [5] and CryptoMiniSAT-5 [80]. Table 3.1 shows the number of total solved instances and average running time of 3 base solvers with 4 initialization methods. We can see that for all three solvers, the BMM-initialization method solves more cases in 4 hours in a shorter time compared with other initializers. CryptoMiniSAT with BMM solves all 50 instances with the shortest running time among the 15 settings, which is also nearly 50% shorter compared with the second-fastest one, MapleSAT with random initialization.

3.6.3 Application Benchmark

This benchmark is from the main track of the SAT competition 2018. The 400 instances were collected from various application domains, such as scheduling, verification and planning [33]. We use the same time limit and memory limit as required in the SAT competition, 5000 seconds and 8GB. We choose the winners of SAT competitions 2017 and 2018, MapleCOMSPS [48] and MapleLCMDistChronoBT [76], as base solvers.

Table 3.1: The number of solved instances in 4 hours and average runtime of MapleSAT, Glucose and CryptoMiniSAT with different initialization methods.

	Initialization Method	Total instances (out of 50)	Avg. time (s)
MapleSAT	Default	48	3645.08
	Random	48	3180.42
	Jeroslow-Wang	47	3389.27
	Survey Propagation	40	3405.20
	BMM	**50**	**2238.85**
Glucose	Default	33	4817.69
	Random	32	5741.74
	Jeroslow-Wang	32	6334.71
	Survey Propagation	30	5386.74
	BMM	**38**	**4563.08**
CryptoMiniSAT	Default	**50**	3475.06
	Random	**50**	3223.48
	Jeroslow-Wang	49	5387.20
	Survey Propagation	41	3501.00
	BMM	**50**	**1706.63**

Table 3.2 presents the number of solved instances and the average runtime of MapleCOMSPS and MapleLCMDistChronoBT with different initialization methods. For both solvers, BMM-initialization gives the most solved instances and the shortest running time.

Table 3.2: Number of solved instances in 5000s and average runtime of MapleCOMSPS and MapleLCMDistChronoBT with different initialization methods on SAT competition 2018 benchmark.

	Initialization	Total (out of 400)	Avg. time (in seconds)
MapleCOMSPS	Default	218	674.43
	Random	214	678.09
	Jeroslow-Wang	222	654.05
	Survey Propagation	157	862.30
	BMM	**230**	**646.18**
MapleLCMDist	Default	**240**	769.85
	Random	232	673.02
	Jeroslow-Wang	235	655.98
	Survey Propagation	173	885.50
	BMM	**240**	**652.80**

Figure 3.4 BMM for SAT

Output: An assignment to all literals
initialize prior $Beta(\theta_k; \alpha_k, \beta_k)$ for each literal l_k; (we typically initialize α_k and β_k to 0.1)
for $n = 1$ **to** $MaxEpochs$ **do**
  for each clause C **do**
    $p := 1$;
    for each literal l_k in C **do**
      if l_k is positive in C **then**
        $p := p \cdot \frac{\beta_k}{\alpha_k+\beta_k}$;
      else
        $p := p \cdot \frac{\alpha_k}{\alpha_k+\beta_k}$;
      end if
    end for
    for each literal l_k in C **do**
      if l_k is positive in C **then**
        $NewFirstMoment := \frac{1}{1-p}(\frac{\alpha_k}{\alpha_k+\beta_k} - p \cdot \frac{\alpha_k}{\alpha_k+\beta_k+1})$;
        $NewSecondMoment := \frac{1}{1-p}(\frac{\alpha_k(\alpha_k+1)}{(\alpha_k+\beta_k)(\alpha_k+\beta_k+1)} - p \cdot \frac{\alpha_k(\alpha_k+1)}{(\alpha_k+\beta_k+1)(\alpha_k+\beta_k+2)})$;
      else
        $NewFirstMoment := \frac{1}{1-p}(\frac{\alpha_k}{\alpha_k+\beta_k} - p \cdot \frac{\alpha_k+1}{\alpha_k+\beta_k+1})$;
        $NewSecondMoment := \frac{1}{1-p}(\frac{\alpha_k(\alpha_k+1)}{(\alpha_k+\beta_k)(\alpha_k+\beta_k+1)} - p \cdot \frac{(\alpha_k+1)(\alpha_k+2)}{(\alpha_k+\beta_k+1)(\alpha_k+\beta_k+2)})$;
      end if
      Solve the following system of equations to compute the new α_k, β_k :

$$\frac{\alpha_k}{\alpha_k + \beta_k} = NewFirstMoment;$$
$$\frac{\alpha_k(\alpha_k + 1)}{(\alpha_k + \beta_k)(\alpha_k + \beta_k + 1)} = NewSecondMoment;$$

    end for
  end for
end for
for each literal l_k **do**
  if $\alpha_k > \beta_k$ **then**
    $l_k := T$;
  else
    $l_k := F$;
  end if
end for

Chapter 4

On the Neural Network: Multiple Moment Matching Inference

In this chapter, we will introduce Multiple Moment Matching Inference (MMMI), a general-purpose sequential Bayesian inference algorithm using the idea of moment matching. MMMI can be seen as an extension of BMM to general settings where likelihoods are not conjugate with priors. In Section 4.1, we will discuss the limitations of the BMM algorithm. Section 4.2 will present our new algorithm, Multiple Moment Matching Inference. Section 4.3 reviews the related work and discusses connections with MMMI. Section 4.4 will show the results of MMMI for Bayesian neural networks on multiple real-world datasets.

4.1 Motivation

To apply BMM algorithms, we need to compute the posterior moments first, each of which is defined by an integral over the posterior distribution:

$$\mathbb{E}_{\Theta\sim P(\Theta|X)}[f(\Theta)] = \int f(\Theta)P(\Theta|X)d\Theta \tag{4.1}$$

For a general posterior distribution, the integral in Equation 4.1 does not have a closed-form solution. What is worse, for most Bayesian inference tasks, the posterior itself cannot be expressed in closed form because of difficulties to marginalize for the denominator. While we can estimate the moments as sample averages, generating posterior samples is a

difficult task on its own. This severely prevents BMM from being applied in some of the most common models, such as Bayesian logistic regression and Bayesian neural networks.

The likelihood for Bayesian logistic regression is:

$$P(y|x,\theta) = y\sigma(x^T\theta) + (1-y)(1-\sigma(x^T\theta)) \tag{4.2}$$

where y is the binary label, x is the feature vector, θ is the weight vector and $\sigma()$ is the sigmoid function.

The likelihood for Bayesian neural network for regression is :

$$P(y|x,\theta) = \mathcal{N}(y; f(x;\theta), \lambda^{-1}) \tag{4.3}$$

where $\mathcal{N}()$ represents the normal distribution, $f(x;\theta)$ is the neural network function parametrized by θ, and λ denotes the precision.

BMM cannot be applied to the likelihood function in Equations 4.2 and 4.3 because they do not have conjugate priors. To solve this problem, we propose Multiple Moment Matching Inference (MMMI), a general-purpose and flexible approximate Bayesian inference algorithm, without the need for conjugate priors.

4.2 Algorithms

We consider sequential Bayesian inference, where observations are streaming. Training in batches is common in modern deep learning, and Bayesian methods lend themselves naturally to online inference [13].

Our goal is to infer the unknown parameter Θ. For illustration purposes, we assume that Θ is one-dimensional. The algorithm can be easily extended to multi-dimensions. Given a set of particles (samples) $\{\theta_i\}_{i=1}^n$ for the prior distribution $P(\Theta)$, we want to transform them to match the posterior distribution $P(\Theta|X)$ without computing its analytical form. A key observation is that the posterior moments can be estimated by reweighting the samples of the prior distribution, as shown in Subsection 4.2.1. Then the particles will be transformed in a direction that minimizes the discrepancy between the prior and posterior moments with respect to a given function set, as shown in Subsection 4.2.2. Afterwards, the transformed particles will represent the prior for the next observation.

4.2.1 Estimation of Posterior Moments

Given particles $\{\theta_i\}_{i=1}^n$ for the prior distribution $P(\Theta)$, we can estimate a prior moment for the function f_j [1] as a sample average:

$$\mathbb{E}_{\Theta \sim P(\Theta)}[f_j(\Theta)] \approx \sum_{i=1}^{n} f_j(\theta_i) \tag{4.4}$$

Directly applying Equation 4.4 to estimate posterior moments requires sampling from the posterior distributions, which is a difficult task on its own. However, with some simple derivations, we show that posterior moments of f_j can be estimated using only prior samples $\{\theta_i\}_{i=1}^n$.

$$\begin{aligned}
\mathbb{E}_{\Theta \sim P(\Theta|X)}[f_j(\Theta)] &= \int_{\Theta} f_j(\Theta) P(\Theta|X) d\Theta \\
&= \int_{\Theta} f_j(\Theta) \frac{P(X|\Theta)P(\Theta)}{P(X)} d\theta \\
&= \frac{\int_{\Theta} f_j(\Theta) P(X|\Theta) P(\Theta) d\Theta}{\int_{\Theta} P(X|\Theta) P(\Theta) d\Theta} \\
&= \frac{\mathbb{E}_{\Theta}[f_j(\Theta) P(X|\Theta)]}{\mathbb{E}_{\Theta}[P(X|\Theta)]} \\
&\approx \frac{\sum_{i=1}^{n} f_j(\theta_i) P(X|\theta_i)}{\sum_{i=1}^{n} P(X|\theta_i)}
\end{aligned} \tag{4.5}$$

As shown in Figure 4.1, our estimation scheme for posterior moments can be interpreted as a weighted average of prior samples, with weights being linearly proportional to the likelihoods. The particles with higher likelihoods on the observation will contribute more to the estimation of posterior moments.

4.2.2 Multi-objective Optimization

We use $\hat{\mu}_{f_j}$ to denote the estimation of posterior moments μ_{f_j} from Equation 4.5, i.e., $\hat{\mu}_{f_j} \approx \mathbb{E}_{\Theta \sim P(\Theta|X)}[f_j(\Theta)]$

[1] In this section, we use a generalized notion of moments. f_j is not restricted to be power functions. We only require f_j to be differentiable.

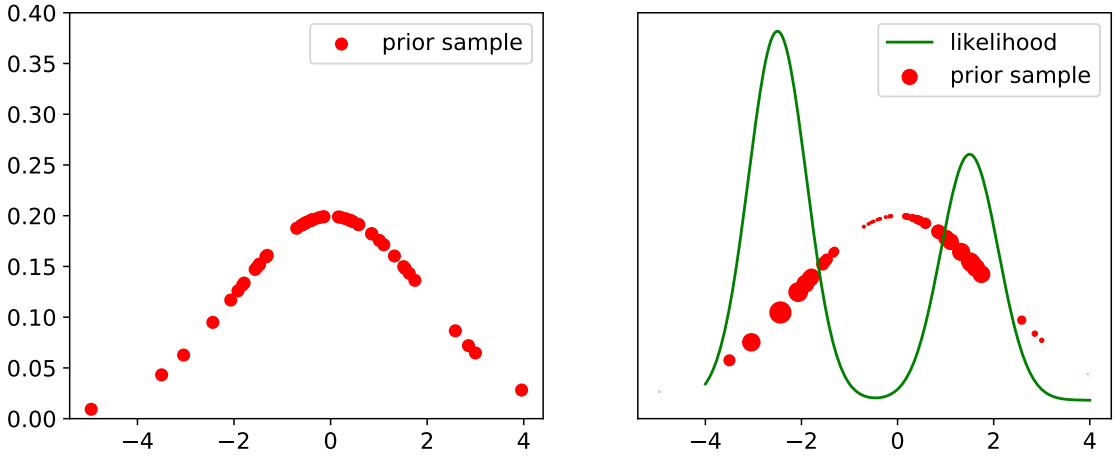

Figure 4.1: Left figure: When estimating prior moments, we use the unweighted average of prior samples. Right figure: When estimating posterior moments, we use the weighted average of prior samples, where weights are proportional to the likelihood. Note that the $1d$-samples are drawn along their density curve for better visualization.

As a reminder, our goal is to transform the prior particles $\{\theta_i\}_{i=1}^n$ to the posterior particles $\{\tilde{\theta}_i\}_{i=1}^n$ through moment matching. In other words, we want the moments of transformed particles $\frac{1}{n}\sum_{i=1}^n f_j(\tilde{\theta}_i)$ to match the corresponding target value $\hat{\mu}_{f_j}$. Our objective can be formulated as finding $\{\tilde{\theta}_i\}_{i=1}^n$ to minimize the discrepancy with $\hat{\mu}_{f_j}$:

$$\min_{\tilde{\boldsymbol{\theta}}:=(\tilde{\theta}_1\cdots\tilde{\theta}_n)} \left|\left| \frac{1}{n}\sum_{i=1}^n f_j(\tilde{\theta}_i) - \hat{\mu}_{f_j} \right|\right|_2^2 \tag{4.6}$$

The gradient of the objective in Equation 4.6 is:

$$\frac{2}{n}\left(\frac{1}{n}\sum_{i=1}^n f_j(\tilde{\theta}_i) - \hat{\mu}_{f_j}\right)\nabla f_j(\tilde{\boldsymbol{\theta}}) \tag{4.7}$$

Note that Equation 4.7 does not require computing the gradients of the likelihood. Computing $\nabla f_j(\tilde{\boldsymbol{\theta}})$ is usually much easier than the gradient of the likelihood. The likelihood function is usually parameterized by complex models, such as neural networks, while f is chosen by the user and usually in a simple form.

On the contrary, most popular approximate inference methods, such as, stochastic

variational inference [34], Laplace approximation [56], Hamiltonian Monte Carlo [67] and SVGD [51], all require gradients of the likelihood. Computing gradients can be costly for large and complex models. Furthermore, there are some cases where gradients are not naturally defined (such as discrete objectives) or are not accessible to the user (such as black-box models). Our method, MMMI, could have unique advantages on the above occasions.

In most situations, we want to match more than one moment for different f_j's. Actually, only matching the first moment could lead to particle collapsing. The Hausdorff moment theorem states that the moments of all orders (from 0 to ∞) uniquely determine a distribution defined on a bounded region [32]. Therefore, intuitively, the more moments are matched, the better is the approximation. Matching multiple moments is a multi-objective optimization problem, where each objective S_j corresponds to minimizing discrepancy for one moment of function f_j.

Multiple Gradient Descent Algorithm (MGDA)

Let $\{S_i(\theta)\}_{i=1}^m, \theta \in \mathbb{R}^N$ be the m smooth objective functions. Let $u_i(\theta)$ denote the gradient of each objective:

$$u_i(\theta) := \nabla S_i(\theta),\ i = 1...m$$

The goal of multi-objective optimization is to find **Pareto-optimality** [60] of all objectives, which is defined in Definition 3.

Definition 3 ***Pareto-optimality****: A point $\tilde{\theta}$ is said to reach Pareto-optimality for $\{S_i(\theta)\}_{i=1}^m$ if we cannot find another point $\theta \in \mathbb{R}^N$ such that $\forall i = 1...m, S_i(\theta) \leq S_i(\tilde{\theta})$ and there exists an objective $S_i(\theta)$ such that $S_i(\theta) < S_i(\tilde{\theta})$.*

In other words, *Pareto-optimality* implies that none of the objectives can be further decreased without increasing some of the other objectives. We can think of the concept of *Pareto-optimality* as a generalization of global optimality in single-objective optimization. Likewise, we can generalize the concept of local optimality to multi-objective optimization, which is *Pareto-stationarity* [60].

Definition 4 ***Pareto-stationarity****: The point $\bar{\theta}$ is said to be at Pareto-stationarity if there exists a convex combination of the gradient vectors $\{u_i(\bar{\theta})\}$ that is equal to zero:*

$$\exists \alpha > 0 \wedge \sum_i^n \alpha_i = 1, \sum_i^n \alpha_i u_i(\bar{\theta}) = 0$$

Theorem 3 *[60] If a point $\tilde{\theta}$ is at Pareto-optimality, then it is at Pareto-stationarity. However, vice versa is not true.*

Multiple Gradient Descent Algorithm (MGDA) is an iterative gradient-based multi-objective optimization algorithm that updates the parameter in a direction that decreases all objectives simultaneously [63] [25] [21] [79] with a guarantee of convergence to Pareto-stationarity.

In each iteration, MGDA finds a vector w such that:

$$(u_i(\theta), w) \geq 0, \quad i = 1...m \tag{4.8}$$

Namely, the angles between w and each gradient u_i are smaller than $90°$. Therefore, following the direction of $-w$, the value of all objective functions $\{S_i(\theta)\}_{i=1}^m$ will decrease simultaneously. By Theorem 4, MGDA reduces the problem of finding the direction w to finding the min-norm element of the convex hull formed by the gradients u_i [21] [63] [25], which has been studied extensively in computational geometry [57] [88] [77]. Figure 4.2 visualizes the intuition in $2d$ space.

Theorem 4 *[63] [25][21] Let U denote the convex hull of the gradient vectors $u_i(\theta_0)$ at point θ_0:*

$$U = \{u \in \mathbb{R}^n | u = \sum_{i=1}^{n} \alpha_i u_i(\theta), \alpha_i \geq 0, \forall i = 1...n, \sum_{i=1}^{n} \alpha_i = 1\}$$

Let w be the minimum-norm element in U:

$$w = argmin_{u \in U} ||u||$$

Since U is compact and convex, we can always find the unique minimum-element w. Then two cases are possible:

- *If $w = 0$, θ_0 is already at Pareto-stationary.*
- *If $w \neq 0$, then for all $u \in U$, $(u, w) \geq ||w||^2$. Thus w defines a descent direction common to all objective functions.*

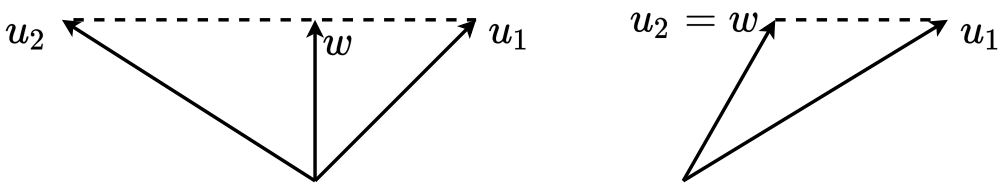

Figure 4.2: Visualization of the minimum-norm element w in the convex hull U of u_1 and u_2. By inspection, the angles between w and all elements in U are smaller than 90°.

Finding the minimum-norm element w in a convex hull is quadratic programming:

$$\begin{aligned} &\min_{\alpha=(\alpha_1,\ldots,\alpha_n)} || \sum_{i=1}^{n} \alpha_i u_i(\theta) ||_2^2 \\ &s.t. \alpha_i > 0, \quad \forall i = 1...n \\ &\sum_{i=1}^{n} \alpha_i = 1 \end{aligned} \tag{4.9}$$

Sener and Koltun [79] proposed a numerical method based on the Frank-Wolfe algorithm to solve Equation 4.9 efficiently in high-dimension settings.

If the update is following the direction of Equation 4.9 iteratively, Désidéri proved that the points would eventually converge to Pareto-stationarity under some constraints on the step size.

4.2.3 Algorithm

The pseudocode for MMMI is in Algorithm 1. For each observation, we first estimate posterior moments using Equation 4.5. Then we compute the gradient for each moment according to 4.7. In the end, we apply MGDA to find a direction that minimizes the discrepancy for each moment, which is then be applied to update the particles.

Particle-based algorithms are prone to have particle-collapse problems, i.e., most particles collapsing to one point. Traditional particle filters deal with this problem by resampling after some iterations. The objective function of SVGD has a term that drives the particles away from each other. Our algorithm, MMMI, prevent particle-collapse problems by imposing more than one optimization objectives and only doing partial optimization at each

Algorithm 1 MMMI

Input: Particles $\boldsymbol{\theta} := (\theta_1, \cdots \theta_i \cdots, \theta_n)$. Function set $\{f_j\}_{j=1}^m$. Data $\{X_k\}_{k=1}^d$. Learning rate λ. The likelihood function $P(X|\Theta)$.
for each data point X_k **do**
  for each particle θ_i **do**
    $L_i \leftarrow P(X_k|\Theta = \theta_i)$
  end for
  for each function f_j **do**
    $\hat{\mu}_{f_j} \leftarrow \frac{\sum_i^n L_i f_j(\theta_i)}{\sum_i^n f_j(\theta_i)}$
    $\nabla S_j \leftarrow \frac{2}{n}\left(\frac{1}{n}\sum_i^n f_j(\theta_i) - \hat{\mu}_{f_j}\right)\nabla f_j(\boldsymbol{\theta})$
  end for
  $\boldsymbol{w} \leftarrow MGDA(\nabla S_1, \cdots, \nabla S_m)$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \boldsymbol{w}$
end for

step. We found our technique to be quite effective at solving particle-collapse problems empirically.

4.3 Related Work

4.3.1 Particle-based Variational Inference

Our algorithm, MMMI belongs to the family of Particle-based Variational Inference (PVI) algorithms, which perform deterministic updates on a set of particles to transform them towards the posterior distribution. The motivation of particle transformation comes from bridging the gap between variational inference and MCMC. Like variational inference, those algorithms perform iterative deterministic updates to decrease the distance with the target distribution, while having the advantage of being non-parametric and generic like MCMC.

The representative of PVI algorithms is SVGD [51], which performs functional gradient descent in kernel Hilbert space to minimize KL divergence. The update equation of SVGD for one particle θ_j is:

$$\theta_j \leftarrow \theta_j - \epsilon[\sum_{i=1}^{n} k(\theta, \theta_i)\nabla_\theta \log q(\theta_i|D) + \nabla_\theta k(\theta_i, \theta)]$$

, where $k(\theta, \theta_i)$ is the chosen kernel function. The first term of the update equation can be interpreted as driving the particles into high-likelihood areas, while the second term is pushing the particles away from each other to encourage multi-modality. We will talk about SVGD in detail in Subsection 4.3.2.

There is then lots of follow-up work on improving SVGD from different perspectives. Stein Variational Newton method (SVN) [22] turns SVGD into a Newton-like iteration in function space by incorporating second-order information. To solve the problem of mode collapse in SVGD, Wang et al. proposed to use particles to represent functions directly instead of samples [86], while message passing SVGD aims to reduce the high-dimensional space into a set of local ones over the Markov blanket with lower dimensions [92].

Compared with other PVI methods, MMMI allows for more flexible priors and likelihoods, which further widens the approximate set for posteriors and also extends its use to black-box models.

- *More flexible priors*: MMMI does not require the prior to be specified in a parametric form. Other methods, such as SVGD, need a parametric prior distribution to compute the unnormalized posterior. Our prior in Bayesian neural networks can be generated directly from well-studied initialization algorithms [27], which extend the potential distribution family of priors and posteriors.

- *More flexible likelihoods*: Unlike most algorithms like SVGD, our method MMMI does not require gradients of the likelihood function. This can speed up inference where gradient computation is slow and expensive. Furthermore, our method MMMI can be applied to the setting where gradients are undefined (such as discrete objectives) or unknown (such as black-box models).

4.3.2 Stein Variational Gradient Descent

Stein Variational Gradient Descent (SVGD) [51] is the outstanding representative of PVI family. Briefly, SVGD moves the particles $\{\theta_i\}_{i=1}^n$ in the steepest direction that maximally decrease the KL divergence with the posterior distribution in a unit ball of the reproducing kernel Hilbert space (RKHS).

Let Θ be a continuous random variable. Given a set of particles $\{\theta_i\}_{i=1}^n$ for the distribution $p(\Theta)$, we want to transform them to match the target distribution $q(\Theta)$. In Bayesian inference, p is the prior and q is the posterior.

Let $k(\theta, \theta') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite kernel, and $\mathcal{H}$ be its corresponding reproducing kernel Hilbert space (RKHS):

$$\mathcal{H} := \{f : f(\theta) = \sum_{i=1}^{m} a_i k(\theta, \theta_i), a_i \in \mathbb{R}, m \in \mathbb{N}, \theta_i \in \mathcal{X}\}$$

We use $\mathcal{F}$ to represent the unit ball of the vector-valued RKHS:

$$\mathcal{F} := \{\phi \in \mathcal{H} | ||\phi|| \leq 1\}$$

Let $\phi(\theta)$ be a perturbation direction, and we define

$$p_{[\epsilon\phi]}(\theta) := \theta + \epsilon\phi(\theta), \theta \sim p(\Theta)$$

.

SVGD finds a perturbation direction ϕ^* in $\mathcal{F}$ that maximally decreases the KL divergence with the target distribution q in $\mathcal{F}$:

$$\phi^* := \text{argmax}_{\phi \in \mathcal{F}} \{-\frac{\partial}{\partial \epsilon} KL(p_{[\epsilon\phi]} || q) | \epsilon = 0\} \tag{4.10}$$

Liu et al. [50] showed that Equation 4.10 has a closed-form solution:

$$\phi^*(\theta') = \mathbb{E}_{\theta \sim p}[k(\theta, \theta') \nabla_\theta \log q(\theta) + \nabla_\theta k(\theta, \theta')] \tag{4.11}$$

In Bayesian inference, we want to transform the particles $\{\theta_i\}_{i=1}^n$ representing the prior distribution $p(\Theta)$ to the posterior distribution $q(\Theta|D)$, where Θ is the parameter of interest and D is the observation. The expectation in Equation 4.11 can thus be estimated as:

$$\phi^*(\theta) \approx \sum_{i=1}^{n} k(\theta, \theta_i) \nabla_\theta \log q(\theta_i | D) + \nabla_\theta k(\theta_i, \theta) \tag{4.12}$$

Thus, during one iteration, SVGD transforms that particle θ_j follows:

$$\theta_j \leftarrow \theta_j - \epsilon [\sum_{i=1}^{n} k(\theta, \theta_i) \nabla_\theta \log q(\theta_i | D) + \nabla_\theta k(\theta_i, \theta)] \tag{4.13}$$

The update direction in Equation 4.12 has a nice interpretation: the first term $k(\theta, \theta_i)\nabla_\theta \log q(\theta_i|D)$ is driving the particles into high-likelihood areas, while the second term $\nabla_\theta k(\theta_i, \theta)$ is pushing the particles away from each other to encourage multi-modality.

SVGD as moment matching

Liu and Wang showed that SVGD matches the moments of the posterior distribution implicitly [52]. More formally, the fixed-point conditions of the SVGD updates guarantee that the particles $\{\theta_i\}_{i=1}^n$ are transformed to match the expectations of all the functions in a stein matching set F^*. However, unlike our algorithm MMMI, users cannot choose which moment they want to match in SVGD. Choosing a specific moment f in SVGD is equivalent to solving a differential equation with no guarantee of closed-form solutions. The success of SVGD confirms the effectiveness of moment matching in general-purpose approximate inference problems. Our algorithm MMMI further offers a more explicit, more flexible and simpler approach to do moment matching in Bayesian learning.

4.4 Experiments

We compare our algorithm MMMI with backpropagation (BP) and SVGD on neural networks. We use 8 datasets from the paper of SVGD [51] plus 4 Kaggle datasets: Australia Weather [91], IMDB [54], Tripadvisor [1] and covertype [10].

Preprocessing

The inputs of all datasets are standardized to mean 0 and variance 1. All datasets are randomly split according to the ratio 90 : 10. The missing values are imputed by the mean of the same column. All 3 algorithms perform 10 epochs for the 8 smaller datasets and 1 epoch for the Kaggle datasets. For Australia Weather, we split the *date* feature into 3 features: *year*, *month* and *day*. For IMDB and Tripadvisor, we transform the reviews to $100-$dimension vectors using Sent2vec [72].

Implementation Details

We use a neural network of one hidden layer with 50 units and RELU as the activation function. For Backpropagation, we use the PyTorch [73] implementation with Adam optimizer. For MMMI and SVGD, we use 200 particles for Banana, Diabetis, German plus

	N	d	BP	SVGD	MMMI
Banana	5300	2	0.847 ± 0.021	0.867 ± 0.015	$\mathbf{0.874 \pm 0.012}$
Diabetis	768	8	0.779 ± 0.044	0.785 ± 0.040	$\mathbf{0.801 \pm 0.042}$
German	1000	20	0.771 ± 0.046	0.778 ± 0.051	$\mathbf{0.793 \pm 0.043}$
Image	2086	18	0.898 ± 0.010	$\mathbf{0.902 \pm 0.013}$	0.899 ± 0.017
Ringnorm	7400	20	0.980 ± 0.003	0.982 ± 0.002	$\mathbf{0.985 \pm 0.003}$
Splice	2991	60	0.916 ± 0.025	$\mathbf{0.926 \pm 0.020}$	0.916 ± 0.026
Two norm	7400	20	0.975 ± 0.009	0.983 ± 0.006	$\mathbf{0.987 \pm 0.005}$
Waveform	5000	21	0.924 ± 0.009	0.929 ± 0.014	$\mathbf{0.931 \pm 0.017}$
Australia Weather	142193	24	0.846 ± 0.004	$\mathbf{0.856 \pm 0.007}$	0.847 ± 0.010
IMDB	50000	100	0.836 ± 0.005	0.839 ± 0.006	$\mathbf{0.841 \pm 0.006}$
Tripadvisor	18307	100	$\mathbf{0.940 \pm 0.008}$	0.935 ± 0.007	$\mathbf{0.940 \pm 0.005}$
Covertype	581012	54	0.812 ± 0.003	0.814 ± 0.007	$\mathbf{0.828 \pm 0.023}$

Table 4.1: Comparison of frequentist BP, SVGD, MMMI on the 8 classification datasets for neural networks. We report the average and standard deviation of test accuracy over 10 random seeds.

Image, and 1000 for the other datasets. The implementation and the parameters for SVGD are chosen to be the same as the original paper [51]. For MMMI, the prior particles are generated from the PyTorch default weight initialization with Gaussian perturbations. We match the first and second marginal moments. We find that running MGDA for only 1 iteration is sufficient to obtain good results. We also use Adam [43] to update the step size of the gradients.

The performance averaged over 10 random trials is reported in Table 4.1. We find that our algorithm MMMI achieves the best results for most datasets. The results of each run are in Figure 4.5.

What distribution should we use to generate prior samples?

We did experiments to study the impact of prior distributions on the performance of MMMI. More specifically, we compare the test accuracy of samples generated from uniform distributions, normal distributions, PyTorch's default weight initialization, and default initialization plus Gaussian perturbations, while keeping other parameters the same. The results over 10 random trials on 4 datasets are summarized in Figure 4.3. We find that PyTorch's default weight initialization plus Gaussian perturbations give the best results

across all datasets. We think that it might be because the PyTorch's initialization helps the particles lie in the "good" region while Gaussian perturbations improve the "diversity" of the samples.

What moments to match?

We also did experiments on the relationship between the type of moments matched and predictive performance of MMMI. We compared the test accuracy of matching different combinations of the first moment, the second moment and the third moment. The results averaged over 10 random trials on 4 datasets are summarized in Figure 4.4. We find that all combinations except only matching the second moment give similarly good performance on these datasets.

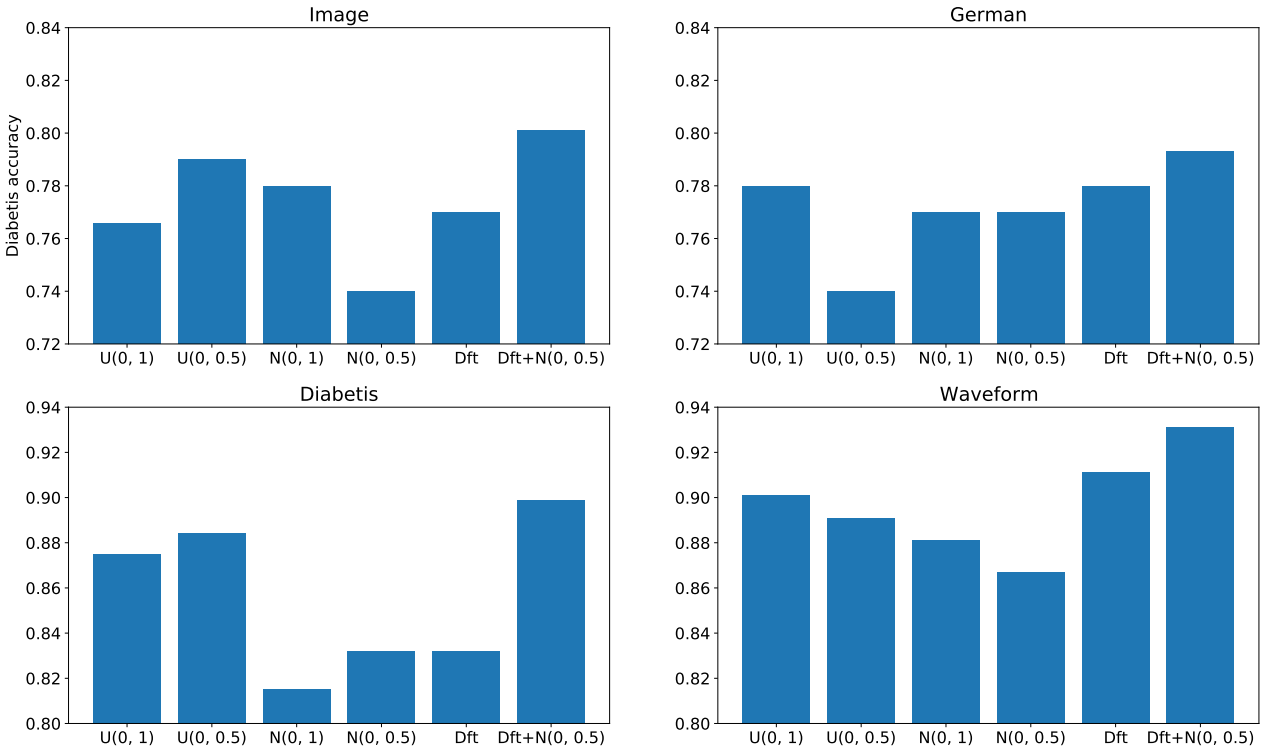

Figure 4.3: This figure shows the impact of distributions generating prior samples on test accuracy. $U(a, b)$ represents the uniform distribution on the interval $[a, b]$. $N(\mu, \sigma)$ represents the normal distribution with mean μ and standard deviation σ. Dft is the default weight initialization in PyTorch, which is essentially customized uniform distributions for each layer. $Dft + N(0, 1)$ uses the default PyTorch initialization added with $N(0, 1)$ Gaussian noise. We find that $Dft + N(0, 1)$ gives the best result across 4 datasets.

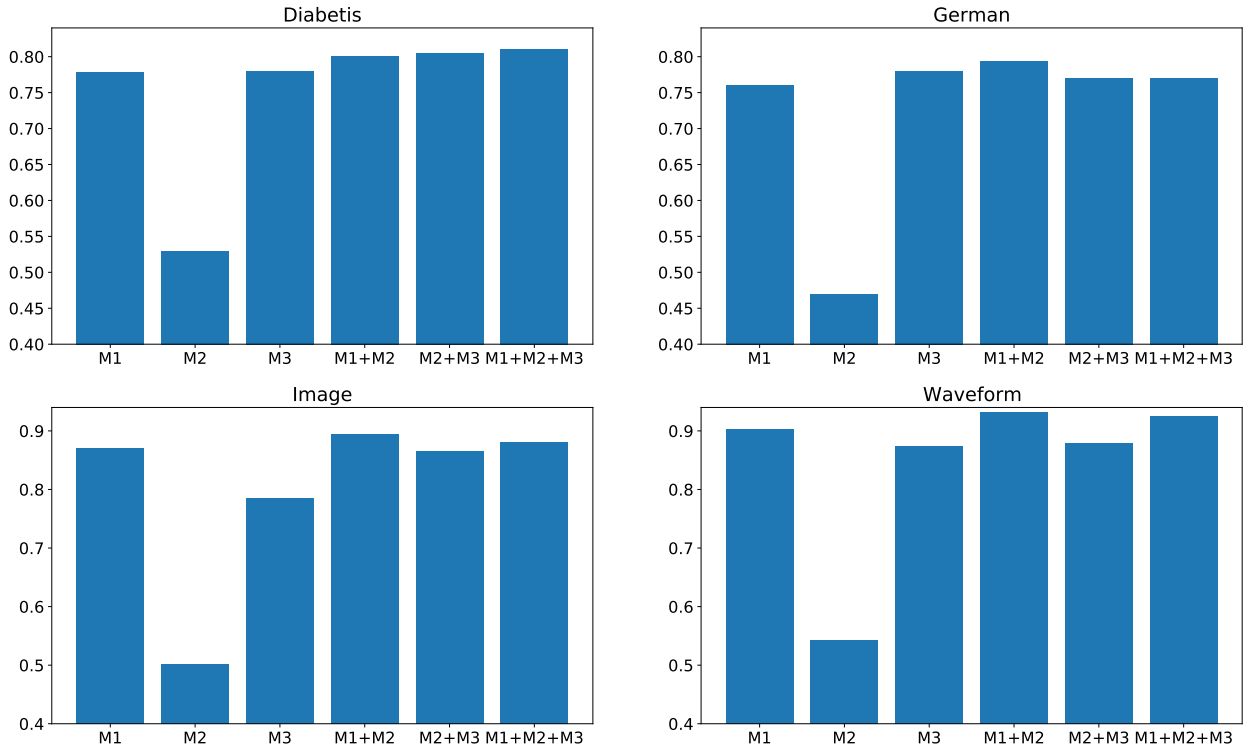

Figure 4.4: This figure shows the impact of matched moments on test accuracy. The first 3 columns $M1, M2, M3$ indicate that we only match the first, second, third moment respectively. $M1 + M2$ means that we match the first and second moments. $M2 + M3$ means that we match the second and third moments. $M1 + M2 + M3$ means that we match the first, second and third moments.

Figure 4.5: Test accuracy of BP, SVGD, MMMI for neural networks on 12 datasets over 10 random seeds

Chapter 5

Conclusions and Future Work

In the previous chapters, we study the method of moments(MM) in approximate Bayesian inference from different perspectives. We hope this thesis will contribute to a better understanding of the following two questions:

- *How to apply MM in approximate Bayesian inference?* In Chapter 2 and 3, we show how BMM can be applied to approximate mixture posteriors in both univariate and multivariate settings. In these cases, BMM turns approximate inference into solving a system of linear equations with closed-form solutions. In Chapter 4, we propose MMMI to extend MM to general inference tasks. MMMI turns approximate inference into multi-objective optimization problems, minimizing discrepancy for different moments.

- *When to apply MM in approximate Bayesian inference?* BMM is used when the mixture components grow exponentially in sequential Bayesian inference. In Chapter 2 and 3, moments can be matched exactly and efficiently. However, other approximate inference methods, such as variational inference, may suffer from local optimality and need multiple iterations before convergence. MMMI in Chapter 4 is a flexible particle-based sampling algorithm. It allows for more flexible priors and likelihoods than other algorithms, which further improves its representation power and extends potential application domains.

In the end, we will point out some directions for future work based on this thesis.

- In Chapter 2, We prove that BMM is consistent in the naïve Bayes model. However, BMM is not consistent in the model of Chapter 3 because it does not always find a

satisfiable solution when there exists one. Therefore, we want to develop a general framework to analyze consistency of BMM, summarizing the sufficient and necessary conditions for BMM to be consistent.

- It's worthwhile exploring if BMM can be used to improve other components of CDCL solvers besides initialization. What is more, is it possible to develop a standalone SAT solver based on Bayesian inference?

- For MMMI, we hope to perform a theoretical analysis of MMMI, such as deriving convergence properties and asymptotic behaviors. Also, we plan to apply MMMI on larger models and more interesting applications.

References

[1] Md Hijbul Alam, Woo-Jong Ryu, and SangKeun Lee. Joint multi-grain topic sentiment: modeling semantic aspects for online reviews. *Information Sciences*, 339:206–223, 2016.

[2] Saeed Amizadeh, Sergiy Matusevych, and Markus Weimer. Learning to solve circuit-sat: An unsupervised differentiable approach. In *International Conference on Learning Representations*, 2018.

[3] Animashree Anandkumar, Daniel Hsu, and Sham M Kakade. A method of moments for mixture models and hidden markov models. In *Conference on Learning Theory*, pages 33–1. JMLR Workshop and Conference Proceedings, 2012.

[4] Gilles Audemard, George Katsirelos, and Laurent Simon. A restriction of extended resolution for clause learning sat solvers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010.

[5] Gilles Audemard and Laurent Simon. Glucose and syrup: Nine years in the sat competitions. *Proc. of SAT Competition*, pages 24–25, 2018.

[6] Roberto J Bayardo Jr and Robert Schrag. Using csp look-back techniques to solve real-world sat instances. In *Aaai/iaai*, pages 203–208. Providence, RI, 1997.

[7] Armin Biere, Alessandro Cimatti, Edmund Clarke, and Yunshan Zhu. Symbolic model checking without bdds. In *International conference on tools and algorithms for the construction and analysis of systems*, pages 193–207. Springer, 1999.

[8] Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, volume 185. IOS press, 2009.

[9] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[10] Jock A Blackard. *Comparison of neural networks and discriminant analysis in predicting forest cover types*. Colorado State University, 1998.

[11] V.S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge University Press, 2008.

[12] Alfredo Braunstein, Marc Mézard, and Riccardo Zecchina. Survey propagation: An algorithm for satisfiability. *Random Structures & Algorithms*, 27(2):201–226, 2005.

[13] Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C Wilson, and Michael I Jordan. Streaming variational bayes. In *NIPS*, 2013.

[14] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[15] Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691. PMLR, 2014.

[16] Ye Chen and Ilya O. Ryzhov. Technical note–consistency analysis of sequential learning under approximate bayesian inference. *Operations Research*, 68(1):295–307, 2020.

[17] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[18] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[19] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[20] Bruno De Finetti. *Theory of probability: A critical introductory treatment*, volume 6. John Wiley & Sons, 2017.

[21] Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318, 2012.

[22] Gianluca Detommaso, Tiangang Cui, Alessio Spantini, Youssef Marzouk, and Robert Scheichl. A stein variational newton method. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 9187–9197, 2018.

[23] Nan Ding, Youhan Fang, Ryan Babbush, Changyou Chen, Robert D Skeel, and Hartmut Neven. Bayesian sampling using stochastic gradient thermostats. In *NIPS*, 2014.

[24] Frederick Eberhardt. Introduction to the foundations of causal discovery. *International Journal of Data Science and Analytics*, 3(2):81–91, 2017.

[25] Jörg Fliege and Benar Fux Svaiter. Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research*, 51(3):479–494, 2000.

[26] Bishwamittra Ghosh, Debabrota Basu, and Kuldeep S Meel. Justicia: A stochastic sat approach to formally verify fairness. *arXiv preprint arXiv:2009.06516*, 2020.

[27] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[28] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel method for the two-sample problem. *Journal of Machine Learning Research*, 1:1–10, 2008.

[29] Shai Haim and Toby Walsh. Restart strategy selection using machine learning techniques. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 312–325. Springer, 2009.

[30] P. Hall and C.C. Heyde. *Martingale Limit Theory and its Application*. Academic Press, 1980.

[31] Lars Peter Hansen. Large sample properties of generalized method of moments estimators. *Econometrica: Journal of the Econometric Society*, pages 1029–1054, 1982.

[32] Felix Hausdorff. Summationsmethoden und momentfolgen. i. *Mathematische Zeitschrift*, 9(1):74–109, 1921.

[33] Marijn JH Heule, Matti Järvisalo, and Martin Suda. Sat competition 2018. *Journal on Satisfiability, Boolean Modeling and Computation*, 11(1):133–154, 2019.

[34] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(5), 2013.

[35] Robert V Hogg, Joseph McKean, and Allen T Craig. *Introduction to mathematical statistics*. Pearson Education, 2005.

[36] Wei-Shou Hsu and Pascal Poupart. Online bayesian moment matching for topic modeling with unknown number of topics. In *NIPS*, pages 4529–4537, 2016.

[37] Antti Hyttinen, Patrik O Hoyer, Frederick Eberhardt, and Matti Järvisalo. Discovering cyclic causal models with latent variables: A general sat-based procedure. In *Uncertainty in Artificial Intelligence*, page 301. Citeseer, 2013.

[38] Amjad Ibrahim, Simon Rehwald, and Alexander Pretschner. Efficient checking of actual causality with sat solving. *Eng. Secur. Dependable Softw. Syst*, 53:241, 2019.

[39] Priyank Jaini, Zhitang Chen, Pablo Carbajal, Edith Law, Laura Middleton, Kayla Regan, Mike Schaekermann, George Trimponias, James Tung, and Pascal Poupart. Online bayesian transfer learning for sequential data modeling. *ICLR*, 2016.

[40] Robert G Jeroslow and Jinchang Wang. Solving propositional satisfiability problems. *Annals of mathematics and Artificial Intelligence*, 1(1-4):167–187, 1990.

[41] N.L. Johnson, S. Kotz, and N. Balakrishnan. *Continuous univariate distributions*, volume 2. Wiley & Sons, 2nd edition, 1995.

[42] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[43] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[44] H. Kushner and G.G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer New York, 2013.

[45] Leonid Anatolevich Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973.

[46] Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727. PMLR, 2015.

[47] Jia Hui Liang, Vijay Ganesh, Pascal Poupart, and Krzysztof Czarnecki. Learning rate based branching heuristic for sat solvers. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 123–140. Springer, 2016.

[48] Jia Hui Liang, Chanseok Oh, Vijay Ganesh, Krzysztof Czarnecki, and Pascal Poupart. Maple-comsps lrb vsids and maplecomsps chb vsids. *Proc. of SAT Competition*, pages 20–21, 2017.

[49] Jia Hui Liang, Chanseok Oh, Minu Mathew, Ciza Thomas, Chunxiao Li, and Vijay Ganesh. Machine learning-based restart policy for cdcl sat solvers. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 94–110. Springer, 2018.

[50] Qiang Liu, Jason Lee, and Michael Jordan. A kernelized stein discrepancy for goodness-of-fit tests. In *International conference on machine learning*, pages 276–284. PMLR, 2016.

[51] Qiang Liu and Dilin Wang. Stein variational gradient descent: a general purpose bayesian inference algorithm. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2378–2386, 2016.

[52] Qiang Liu and Dilin Wang. Stein variational gradient descent as moment matching. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8868–8877, 2018.

[53] Inês Lynce and Joao Marques-Silva. Efficient haplotype inference with boolean satisfiability. In *National Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2006.

[54] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

[55] David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, 2002.

[56] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[57] Naoki Makimoto, Ikuo Nakagawa, and Akihisa Tamura. An efficient algorithm for finding the minimum norm point in the convex hull of a finite point set in the plane. *Operations research letters*, 16(1):33–40, 1994.

[58] João P Marques-Silva and Karem A Sakallah. GRASP: A Search Algorithm for Propositional Satisfiability. *Computers, IEEE Transactions on*, 48(5):506–521, 1999.

[59] Marc Mézard and Thierry Mora. Constraint satisfaction problems and neural networks: A statistical physics perspective. *Journal of Physiology-Paris*, 103(1-2):107–113, 2009.

[60] Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 2012.

[61] Ilya Mironov and Lintao Zhang. Applications of sat solvers to cryptanalysis of hash functions. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 102–115. Springer, 2006.

[62] Matthew W Moskewicz, Conor F Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th annual Design Automation Conference*, pages 530–535. ACM, 2001.

[63] Hiroaki Mukai. Algorithms for multicriterion optimization. *IEEE Transactions on Automatic Control*, 25(2):177–186, 1980.

[64] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[65] Nina Narodytska, Alexey Ignatiev, Filipe Pereira, Joao Marques-Silva, and IS RAS. Learning optimal decision trees with sat. In *IJCAI*, pages 1362–1368, 2018.

[66] Nina Narodytska, Shiva Kasiviswanathan, Leonid Ryzhyk, Mooly Sagiv, and Toby Walsh. Verifying properties of binarized deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[67] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[68] Saeed Nejati. Cdcl (crypto) and machine learning based sat solvers for cryptanalysis. 2020.

[69] Jerzy Neyman. Outline of a theory of statistical estimation based on the classical theory of probability. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 236(767):333–380, 1937.

[70] Thanh Tang Nguyen, Sunil Gupta, and Svetha Venkatesh. Distributional reinforcement learning via moment matching. *AAAI*, 2021.

[71] Frank Nielsen and Vincent Garcia. Statistical exponential families: A digest with flash cards. *arXiv preprint arXiv:0911.4863*, 2009.

[72] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In *NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics*, 2018.

[73] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[74] Abdullah Rashwan, Han Zhao, and Pascal Poupart. Online and distributed bayesian moment matching for parameter learning in sum-product networks. In *Artificial Intelligence and Statistics*, pages 1469–1477. PMLR, 2016.

[75] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[76] Vadim Ryvchin and Alexander Nadel. Maple_lcm_dist_chronobt: Featuring chronological backtracking. *Proc. of SAT Competition*, pages 29–29, 2018.

[77] Kazuyuki Sekitani and Yoshitsugu Yamamoto. A recursive algorithm for finding the minimum norm point in a polytope and a pair of closest points in two polytopes. *Mathematical Programming*, 61(1):233–249, 1993.

[78] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L Dill. Learning a sat solver from single-bit supervision. *International Conference on Learning Representations*, 2018.

[79] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 525–536, 2018.

[80] Mate Soos. The cryptominisat 5.5 set of solvers at the sat competition 2018. *Proc. of SAT Competition*, pages 17–18, 2018.

[81] Dougal J Sutherland, Hsiao-Yu Tung, Heiko Strathmann, Soumyajit De, Aaditya Ramdas, Alexander J Smola, and Arthur Gretton. Generative models and model criticism via optimized maximum mean discrepancy. In *ICLR (Poster)*, 2017.

[82] Sofia Triantafillou and Ioannis Tsamardinos. Constraint-based causal discovery from multiple interventions over overlapping variable sets. *The Journal of Machine Learning Research*, 16(1):2147–2205, 2015.

[83] John N Tsitsiklis. Asynchronous stochastic approximation and q-learning. *Machine learning*, 16(3):185–202, 1994.

[84] Dilin Wang, Zhe Zeng, and Qiang Liu. Stein variational message passing for continuous graphical models. In *International Conference on Machine Learning*, pages 5219–5227. PMLR, 2018.

[85] Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, pages 6545–6554. PMLR, 2019.

[86] Ziyu Wang, Tongzheng Ren, Jun Zhu, and Bo Zhang. Function space particle optimization for bayesian neural networks. In *International Conference on Learning Representations*, 2018.

[87] 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)*, pages 681–688. Citeseer, 2011.

[88] Philip Wolfe. Finding the nearest point in a polytope. *Mathematical Programming*, 11(1):128–149, 1976.

[89] Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *Journal of artificial intelligence research*, 32:565–606, 2008.

[90] Pengcheng Yang, Fuli Luo, Shuangzhi Wu, Jingjing Xu, Dongdong Zhang, and Xu Sun. Learning unsupervised word mapping by maximizing mean discrepancy. *arXiv preprint arXiv:1811.00275*, 2018.

[91] Joe Young. Rain in australia, 2020.

[92] Jingwei Zhuo, Chang Liu, Jiaxin Shi, Jun Zhu, Ning Chen, and Bo Zhang. Message passing stein variational gradient descent. In *International Conference on Machine Learning*, pages 6018–6027. PMLR, 2018.