Probabilistic Inference in Piecewise Graphical Models

Hadi Mohasel Afshar

A thesis submitted for the degree of
Doctor of Philosophy at
The Australian National University

2 August 2016

Declaration

I hereby declare that this thesis is my original work which has been done in collaboration with other researchers. This document has not been submitted for any other degree or award in any other university or educational institution. Parts of this thesis have been published in collaboration to other researchers in international conferences as listed below:

- **(Chapter 4)** H. M. Afshar, S. Sanner, and E. Abbasnejad (2015). Linear-time Gibbs Sampling in Piecewise Graphical Models. In Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-15), Austin, U.S.

- **(Chapter 5)** H. M. Afshar, S. Sanner, C. Weber (2016). Closed-form Gibbs Sampling for Graphical Models with Algebraic Constraints, The 30th AAAI Conference on Artificial Intelligence, Phoenix, U.S.

- **(Chapter 6)** H. M. Afshar and J. Domke (2015). Reflection, Refraction and Hamiltonian Monte Carlo, The 29-th Conference on Neural Information Processing Systems (NIPS-15), Montreal, Canada.

Hadi Mohasel Afshar
2 August 2016

To the future anthropomorphic artificially intelligent beings.

Acknowledgments

I would like to express my gratitude towards all who helped me make this thesis possible.

First and foremost, I would like to thank my supervisor **Scott Sanner** for his insightful guidance throughout my PhD, for his caring and patience in introducing me to the field of graphical models, for his intuitive advice about the potentially fruitful lines of research, for all invaluable discussions I had with him and all I learned from him but above all, for his friendship, support and encouragement through the toughest times. I have been quite lucky to have a supervisor that was more concerned about his students and their future than the students would care about themselves and their own future. I am extremely grateful for all these and feel deeply in dept.

I would like to thank the chair of my supervisory panel Prof. **Marcus Hutter** for teaching me how to think in a more disciplined way, how to formalize my intuitions and how to interpret formulas intuitively. I had the opportunity to learn from his subtle vision and enjoyed participating in his reinforcement reading group. I appreciate all the time and energy he spent for me throughout my PhD.

Special thanks to the brilliant researcher **Justin Domke**. I really enjoyed working with him and learned a lot from him.

I would like to thank my adviser Prof. **John Lloyd**, Prof **Bob Williamson** and **Chris Webers** for their advice and the discussions that I had with them.

I acknowledge the financial, academic and technical support provided by the Australian National University (ANU) and appreciate the welcoming and collaborative environment provided by the National ICT of Australia (NICTA) where I had the opportunity to work with talented researchers.

I would like to thank my fellow postgraduate friends. Special thanks to Firouzeh Khoshnoudi, Tom Everitt, Ehsan Abbasnejad, Phuong Nguyen, Mayank Daswani, Tor Lattimore, Jan Leike, Suvash Sedhain, Kar Wai, Paul Scott and many others for great discussions that we had and for making my life so joyful.

Finally I have to thank my parents from the depth of my heart for their endless love and support.

Abstract

In many applications of probabilistic inference the models contain piecewise densities that are differentiable except at partition boundaries. For instance, (1) some models may intrinsically have finite support, being constrained to some regions; (2) arbitrary density functions may be approximated by mixtures of piecewise functions such as piecewise polynomials or piecewise exponentials; (3) distributions derived from other distributions (via random variable transformations) may be highly piecewise; (4) in applications of Bayesian inference such as Bayesian discrete classification and preference learning, the likelihood functions may be piecewise; (5) context-specific conditional probability density functions (tree-CPDs) are intrinsically piecewise; (6) influence diagrams (generalizations of Bayesian networks in which along with probabilistic inference, decision making problems are modeled) are in many applications piecewise; (7) in probabilistic programming, conditional statements lead to piecewise models. As we will show, exact inference on piecewise models is not often scalable (if applicable) and the performance of the existing approximate inference techniques on such models is usually quite poor.

This thesis fills this gap by presenting scalable and accurate algorithms for inference in piecewise probabilistic graphical models. Our first contribution is to present a variation of Gibbs sampling algorithm that achieves an exponential sampling speedup on a large class of models (including Bayesian models with piecewise likelihood functions). As a second contribution, we show that for a large range of models, the time-consuming Gibbs sampling computations that are traditionally carried out per sample, can be computed symbolically, once and prior to the sampling process. Among many potential applications, the resulting *symbolic Gibbs sampler* can be used for fully automated reasoning in the presence of deterministic constraints among random variables. As a third contribution, we are motivated by the behavior of Hamiltonian dynamics in optics —in particular, the reflection and refraction of light on the refractive surfaces— to present a new Hamiltonian Monte Carlo method that demonstrates a significantly improved performance on piecewise models.

Hopefully, the present work represents a step towards scalable and accurate inference in an important class of probabilistic models that has largely been overlooked in the literature.

Contents

Chapter 1

Introduction

1.1 Motivation

Reasoning under uncertainty is an indispensable part of a large range of real-world application domains. In such inference tasks, it is required to predict the future state of the world and/or actions that lead to maximum gains based on some incomplete or uncertain information and partial/noisy observations. For instance, an optical character recognition system may encounter an ambiguous input glyph which can be mapped to more than one alphabet letter or a robot may requrie finding its goal given input signals received from its noisy sensors.

Statistical inference, based on probability theory, provides a natural and robust formalism for such reasoning tasks. The advent of graphical models (GMs) has (1) provided systematic methods to represent complex distributions in succinct and compact forms and (2) facilitated the extraction of structured information such as conditional dependencies of random variables. By exploiting such structures, GMs have (3) led to scalable and effective inference algorithms. Finally, by separating model representation from the inference, (4) GMs have enabled the users (e.g. data scientists) to focus on the design of appropriate models for the systems about which they intend to carry out inference while letting the actual reasoning task be performed by general and automated reasoning mechanisms. As a result, GMs have vastly expanded the application domains in which probabilistic reasoning can be effectively used and consequently, have become a fundamental model of machine learning and artificial intelligence.

Despite a relatively long history of probabilistic reasoning that can be traced back to sixteenth and seventeenth centuries [Hald 2003], for a long time, its use has been almost exclusively restricted to simple models which had closed-form solutions that could be worked out manually. Luckily, the rapid growth in the processing power of computers has changed the situation and led to the emergence of a range of approximate inference algorithms that can handle a large range of models for which manual inference is intractable or impossible. In particular, particle-based algorithms –i.e. methods in which the original distribution is approximated by (a Markov chain of) samples– are appealing because they are asymptotically unbiased in the sense that if the number of sampled particles tends to infinity, the distribution approximation error tends to zero.

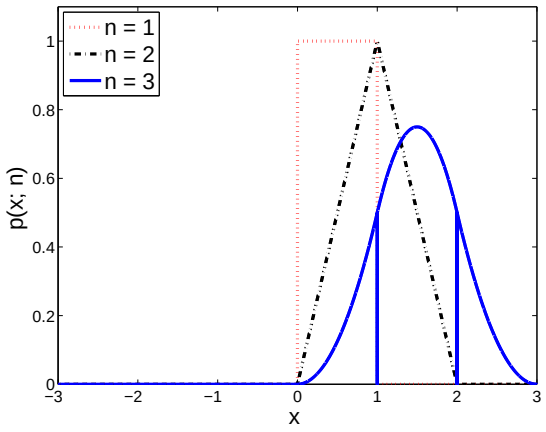


Figure 1.1: Irwin-Hall probability density functions for parameter $n = 1, 2$, and 3.

However, the convergence rate of the particle based algorithms and in a broader view, the performance of all approximate inference methods can significantly depend on the characteristics of the original model. In particular, the performance of the existing approximate inference algorithms on piecewise models is often quite poor. Throughout, by *piecewise models*, we refer to models with probability density functions which are continuous and differentiable except at piecewise partition boundaries. Such models have several applications.

In the remaining part of this subsection, we provide some pervasive application domains associated with models that are intrinsically piecewise.

1.1.1 Bounded support models

In many models, distributions have bounded support. As a result, some parametric distributions[1] that are extensively used in the literature are supported on semi-infinite intervals, such as the following: *Beta prime, Chi, gamma and exponential distributions*.

Some other well-known continuous distributions have bounded supports, such as: *Uniform, U-quadratic, beta, logitnormal, reciprocal, raised cosine, Von Mises, Kent, Wigner semicircle distributions*, etc.

Finally, among bounded support distributions, some like the following are piecewise with more than one segment: *triangular, trapezoidal, Bates and Irwin-Hall distributions*.

To be more concrete, consider an Irwin-Hall distribution with parameter n. This distribution is defined as follows:

$$p(x;n) = \frac{1}{2\,(n-1)!} \sum_{k=0}^{n} (-1)^k \binom{n}{k} (x-k)^{n-1} \operatorname{sgn}(x-k)$$

[1]Throughout, we often do not distinguish between distributions and density functions. The intention should be clear from the context. It should also be pointed out that the main focus of this work is on the continuous models although in many cases, the generalization to continuous/discrete hybrid models is not hard.

Therefore, $p(x;n)$ is an $(n+1)$-piecewise function. For instance, Irwin-Hall distribution with parameters $n = 1$ to 3 are plotted in Figure 1.1. Clearly, $p(x,1)$ is a uniform distribution:

$$p(x;1) = \begin{cases} 1 & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

$p(x,2)$ is a triangular distribution:

$$p(x;2) = \begin{cases} x & \text{if } 0 \leq x \leq 1 \\ 2-x & \text{if } 1 < x \leq 2 \\ 0 & \text{otherwise} \end{cases}$$

and $p(x,3)$ corresponds to a 4-piece distribution:

$$p(x;3) = \begin{cases} \frac{1}{2}x^2 & \text{if } 0 \leq x \leq 1 \\ \frac{1}{2}\left(-2x^2+6x-3\right) & \text{if } 1 < x \leq 2 \\ \frac{1}{2}\left(x^2-6x+9\right) & \text{if } 2 < x \leq 3 \\ 0 & \text{otherwise} \end{cases}$$

1.1.2 Mixture of truncated basis functions

Piecewise exponential or polynomial functions (also known as mixtures of truncated polynomials and exponentials) are expressive enough to approximate arbitrary functions up to arbitrary precisions. For instance, following the approximation method used in [Cobb et al. 2006], Figure 1.2 shows that a log-normal distribution (with parameters $\mu = 0$ and $\sigma^2 = 0.5$) can be approximated by a 5-piece mixture of truncated exponentials (MTE) in a reasonably accurate manner. The corresponding MTE is as follows:

$$\begin{cases} a_{01} + a_{11}\exp\left(b_{11}(x-m)\right) + a_{21}\exp\left(b_{21}(x-m)\right) & \text{if } \exp(\mu-3\sigma) \leq x < d^- \\ a_{02} + a_{12}\exp\left(b_{12}(x-m)\right) + a_{22}\exp\left(b_{22}(x-m)\right) & \text{if } d^- \leq x < m \\ a_{03} + a_{13}\exp\left(b_{13}(x-m)\right) + a_{23}\exp\left(b_{23}(x-m)\right) & \text{if } m \leq x < d^+ \\ a_{04} + a_{14}\exp\left(b_{14}(x-m)\right) + a_{24}\exp\left(b_{24}(x-m)\right) & \text{if } d^+ \leq x < \exp(\mu+3\sigma) \\ 0 & \text{otherwise} \end{cases}$$

where a_{**} and b_{**} are constants, $m = \exp(\mu - \sigma^2)$ and

$$d^{\pm} = \exp\left(\frac{1}{2}(2\mu - 3\sigma^2 \pm \sigma\sqrt{4+\sigma^2})\right)$$

An important feature for models that are entirely made up of piecewise exponential or polynomial functions (with particular forms of partitioning constraints[2]) is that

[2]To date, the partitioning constraints under which closed-form solutions are found is limited to hyper-

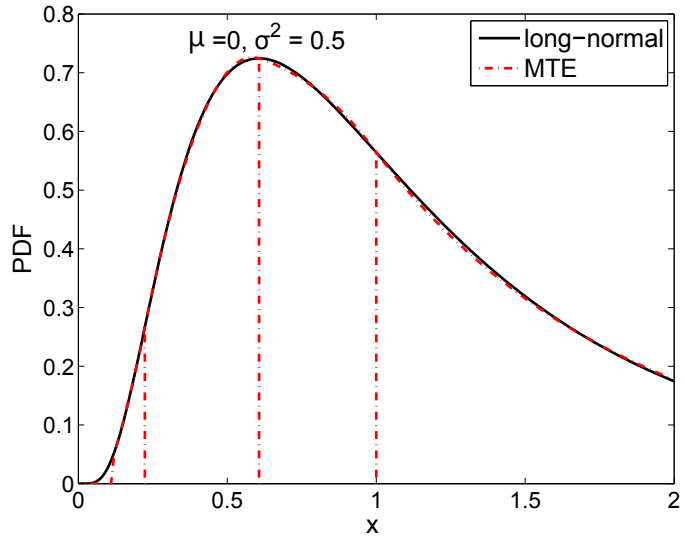


Figure 1.2: Approximation of a log-normal distribution with parameters $(\mu = 0, \sigma^2 = 0.5)$ (back curve) with a 5-piece mixture of truncated exponential (MTE) function (red curve).

marginalization (which is a crucial operation required for probabilistic inference) can be performed in closed-form. This feature guarantees an analytical solution for the model.

The other important feature is that they enable modeling arbitrary density (and more generally speaking, *potential*) functions. This provides a powerful tool for models which are beyond the scope of the families of probabilistic distributions which have known closed-forms (namely, parametric families of functions).

It should be emphasized that the parametric families are too restricted to model distributions that one may encounter in practice. For instance consider a robotic application where the location of a robot is unknown but it is known that it cannot be outside a particular set of corridors, that is, the distribution of the robot location is zero outside some arbitrary shaped area. In such applications, the support is often partitioned into several regions each of which is associated with a truncated (partial or non-normalized) distribution. Subsequently, the total distribution is approximated by the mixture of such truncated distributions.

1.1.3 Random variable transformations

The distribution of a function of some random variables with (semi)bounded supports can be highly piecewise.

As a simple example, it should be pointed out that the Irwin-Hall distribution $p(x; n)$ discussed on Section 1.1.1 is equivalent to a probability distribution of a ran-

rectangular, hyper-rhombus and linear partitioning constraints that have been used in existing work [Shenoy and West 2011; Shenoy 2012; Sanner and Abbasnejad 2012].

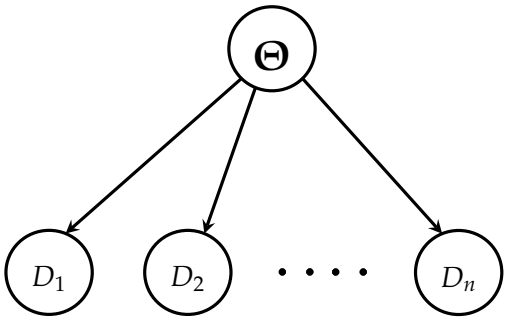


Figure 1.3: A Simple Bayesian network in which random variables D_1 to D_n are conditionally independent given Θ.

dom variable defined as the summation of n independent and uniformly distributed random variables.

As another example, consider a random variable X with probability density function $f(x)$ which is positive on the interval (a, b). Similarly, let Y be a random variable with density $g(x)$ being positive on the interval (c, d). As [Glen et al. 2004] shows, the density function of their product $V = XY$ is:

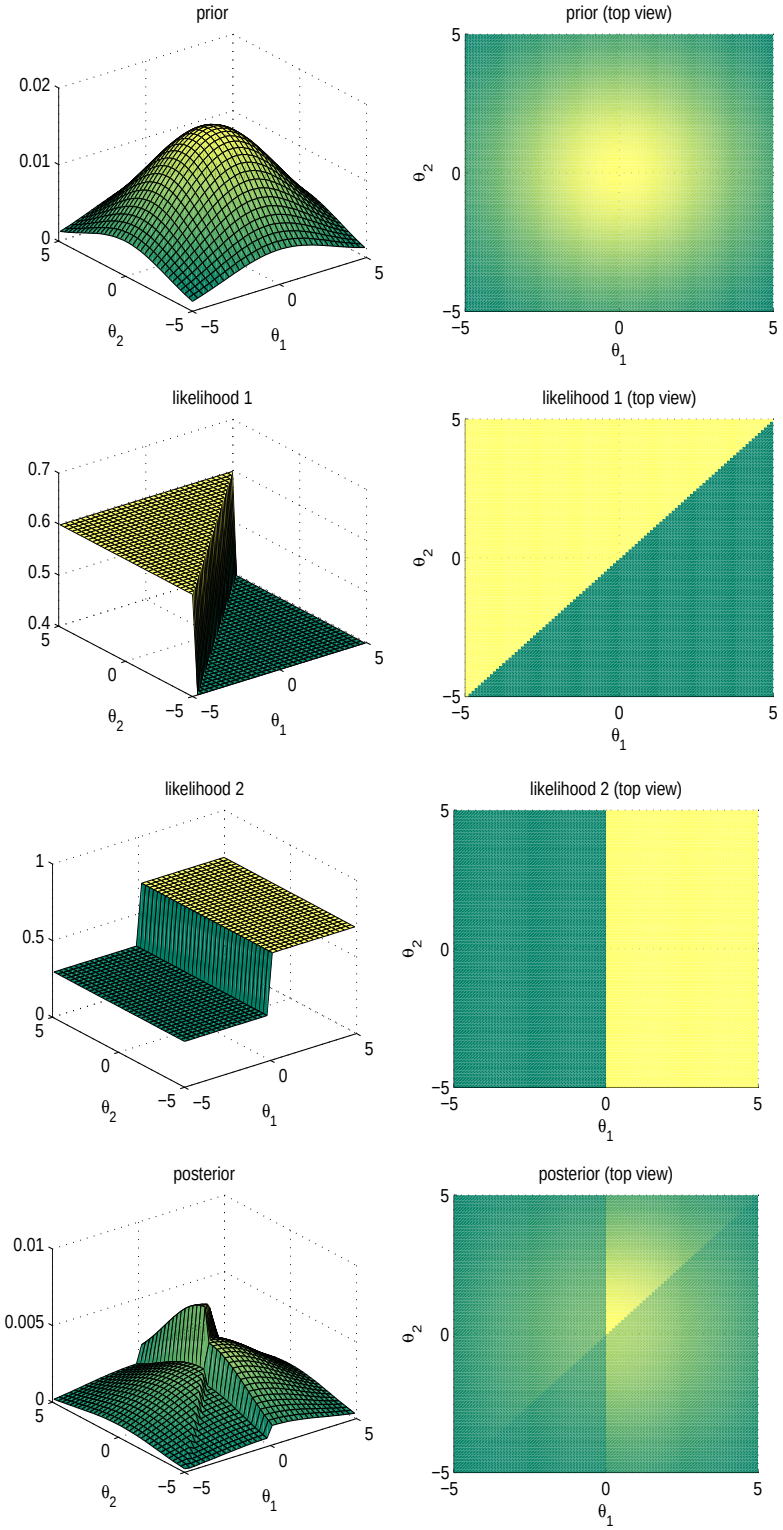
$$h(v) = \begin{cases} \int_a^{v/c} g\left(\frac{v}{x}\right) f(x) \frac{1}{x} \, \mathrm{d}x & \text{if } ac < v < ad \\ \int_{v/d}^{v/c} g\left(\frac{v}{x}\right) f(x) \frac{1}{x} \, \mathrm{d}x & \text{if } ad < v < bc \\ \int_{v/d}^{b} g\left(\frac{v}{x}\right) f(x) \frac{1}{x} \, \mathrm{d}x & \text{if } bc < v < bd \end{cases}$$

1.1.4 Piecewise likelihoods

Figure 1.3 represents a very simple *Bayesian network* (a graphical model that will be introduced in Chapter 2) in which, conditioned on a random variable Θ (called, *parameter*), random variables D_1 to D_n (called, *data*) are independent. With respect to Bayes rule, $p(\Theta \mid D_1, \ldots, D_n)$, the posterior distribution of Θ conditioned on observed variables D_1 to D_n is proportional to the product of the prior distribution $p(\Theta)$ times likelihood functions $p(D_i \mid \Theta)$:

$$p(\Theta \mid D_1, \ldots, D_n) \propto p(\Theta) \prod_{i=1}^{n} p(D_i \mid \Theta)$$

Each likelihood function is a conditional distribution associated with an observation of a data point. If the likelihoods are modeled by piecewise functions, the number of pieces in the posterior can be exponential in the number of observations. This is clarified in Figure 1.4 where two 2-piece likelihood functions lead to a 4-piece posterior. It should be pointed out that practical applications may involve a large number of observations. In such cases, the corresponding piecewise posteriors many be extremely complicated.



posterior $\propto$ prior $\times$ likelihood 1 $\times$ likelihood 2

Figure 1.4: In Bayesian inference with piecewise likelihoods, number of pieces in the posterior can grow exponentially in the number of observations.

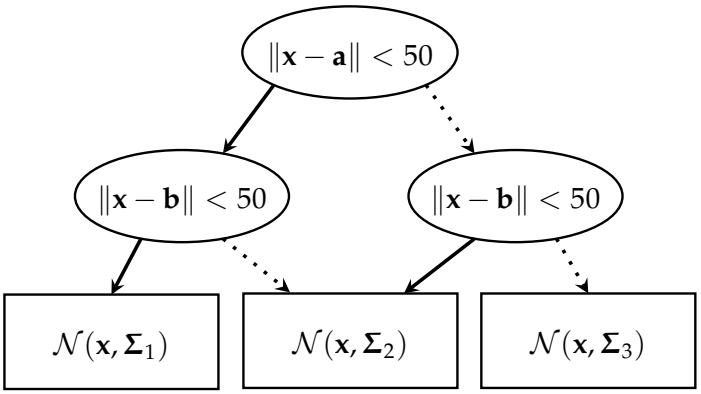


Figure 1.5: An extended algebraic decision diagram (XADD) representing the distribution of the predicted location of a mobile device in the example provided in Section 1.1.5. The Location of the device and two radio antennas are represented by **x**, **a** and **b**, respectively. (Solid and dotted lines represent edges to a low and high children respectively.)

1.1.5 Context-specific conditional densities

Decision diagrams and context-specific conditional probability distributions such as tree-CPDs and rule CPDs are natural structures extensively used to represent discrete models in cases where structural regularities arise in some contexts. In the case of continuous models, context-specific conditional densities correspond to piecewise density functions. For instance, consider the following mobile device localization model:

Example 1. *The location of a mobile device is performed via multilateration of radio signals between radio towers positioned at locations* $\boldsymbol{a} = (a_1, a_2)$ *and* $\boldsymbol{b} = (b_1, b_2)$ *as well as GPS signals. The coverage range of each antenna tower is 50 meters. If the device position (say,* $\boldsymbol{x} = (x_1, x_2)$*) is within the coverage range of both antennas, the distribution of its predicted location* $\boldsymbol{y}$ *is a bivariate normal distribution* $\mathcal{N}(\boldsymbol{x}, \mathbf{\Sigma}_1)$*. If the device location is within the range of only one antenna, the distribution of its predicted location is* $\mathcal{N}(\boldsymbol{x}, \mathbf{\Sigma}_2)$ *(less precise) and if it is not in the range of any antenna, the distribution follows by* $\mathcal{N}(\boldsymbol{x}, \mathbf{\Sigma}_3)$ *(least precise localization):*

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \begin{cases} \mathcal{N}(\boldsymbol{x}, \mathbf{\Sigma}_1) & \text{if } \|\boldsymbol{x} - \boldsymbol{a}\| < 50,\ \|\boldsymbol{x} - \boldsymbol{b}\| < 50 \\ \mathcal{N}(\boldsymbol{x}, \mathbf{\Sigma}_2) & \text{if } \|\boldsymbol{x} - \boldsymbol{a}\| \geq 50,\ \|\boldsymbol{x} - \boldsymbol{b}\| < 50 \\ \mathcal{N}(\boldsymbol{x}, \mathbf{\Sigma}_2) & \text{if } \|\boldsymbol{x} - \boldsymbol{a}\| < 50,\ \|\boldsymbol{x} - \boldsymbol{b}\| \geq 50 \\ \mathcal{N}(\boldsymbol{x}, \mathbf{\Sigma}_3) & \text{if } \|\boldsymbol{x} - \boldsymbol{a}\| \geq 50,\ \|\boldsymbol{x} - \boldsymbol{b}\| \geq 50 \end{cases}$$

In this example, the device distance from antennas i.e. $\|\mathbf{x} - \mathbf{a}\|$ and $\|\mathbf{x} - \mathbf{b}\|$ act as the *context*. Figure 1.5 illustrates a *extended algebraic decision diagram* (XADD) representing the context specific density function associated with the prediction mobile device location. An XADD is a generalization of a decision diagram to continuous (and discrete/continuous hybrid) models [Sanner et al. 2011].

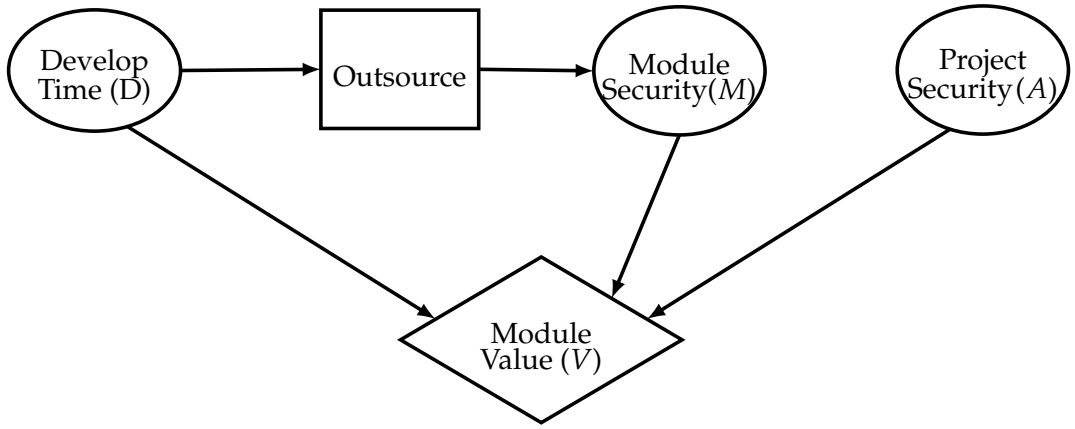


Figure 1.6: An influence diagram representing a situation where a company plans whether to outsource the development of a new software module or does not do so. The uncertainty nodes are *Develop. speed* (representing the time it take to develop the module internally), *Module security* (representing the quantification of the security level of the module to be developed) and *Project security* (representing the security level of the existing project modules). The decision node, *Outsource*, indicates whether to outsource the module or not. The value node, *Module value*, is a function of development speed and the security of the new project augmented with the module.

1.1.6 Influence diagrams

An influence diagram (ID) is a generalization of a graphical model in which along with probabilistic inference problems, decision making problems are modeled (based on maximizing expected utility criteria) [Howard and Matheson 2005].

An ID is a directed acyclic graph with three different kinds of nodes:

- *random nodes*, a.k.a. *uncertainty nodes* which are represented as ovals and are associated with probability distribution or density functions. An ID exclusively made of random nodes is a graphical model.
- *decision nodes* which are represented as rectangles and indicate the options available for a decision-maker.
- An often singular *value node* that is represented by a diamond and is associated with a utility function upon which the decision-maker acts.

To solve for an ID is to find an optimal action policy that maximizes the expected utility associated with the value node.

ID Value nodes/utility functions can be sophisticated and in many applications piecewise. An example of an ID for which the natural modeling requires a piecewise value function is presented in Figure 1.6. This ID corresponds a situation where a software company requires adding a new module to an existing project.

The solution to this ID is to decide optimally whether to outsource the new module or develop it internally w.r.t. a utility function $u_V(\cdot)$ which is a function of three different uncertain factors:

- *Development time (D)*: The amount of time it takes to develop the module in the company compared to outsourcing.
- *Module security (M)*: Security factor associated with developing the module inside the company as opposed to relying on the third parties.
- *Project security (A)*: Security factor associated with the existing project code.

Let utility element associated with the above factors be $f_D(\cdot)$, $f_M(\cdot)$ and $f_A(\cdot)$, respectively. The total security of a system is the minimum of the security of its components. Therefore, the utility element associated with the security of the new module added to the existing code is the minimum of the two security utility elements. As a result, it is natural to model the total utility function in a piecewise manner as below:

$$u_V(D=d, M=m, A=a) = f_D(d) + \min\left(f_S(m) + f_S(a)\right)$$
$$= \begin{cases} f_D(d) + f_S(m) & \text{if } f_S(m) \leq f_S(a) \\ f_D(d) + f_S(a) & \text{if } f_S(m) > f_S(a) \end{cases}$$

An efficient way to solve an ID is to reduce it to an ordinary graphical model [Zhang 1998]. This involves the replacement of the value node with a newly introduced random variable probability of which is typically proportional to the utility of the original value node plus some bias [Cooper 1988].

Therefore, an ID with piecewise utility function corresponds to a piecewise graphical model. Piecewise functions provide an appropriate and natural framework for ID modeling and some existing ID modeling/inference formalisms such as [Cobb and Shenoy 2004] are already based on piecewise functions.

1.1.7 Probabilistic programming

Emerging probabilistic programming languages (PPLs) try to unify general purpose programming with probabilistic modeling [Vajda 2014]. While the input values of a function in a general purpose programming languages typically consists of deterministic values, the inputs of a PPL function may be distributions over random variables. Therefore, in a PPL function, computations take place over uncertain data and the function output is an implicitly defined distribution over the possible output values.

In PPL, conditional statements (i.e. if-statements) are intrinsically piecewise, leading to piecewise output distributions.

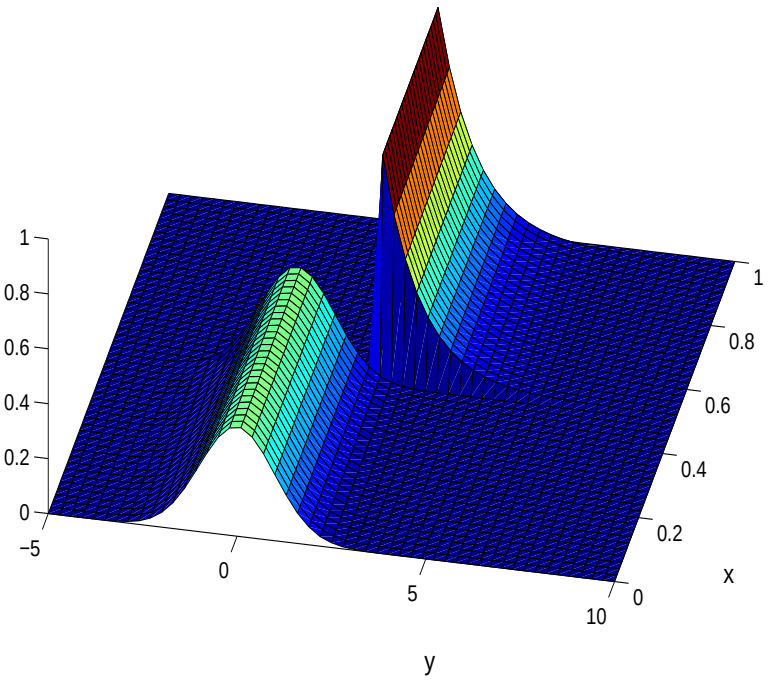
As an instance, a simple probabilistic program that includes a conditional statement is presented in Figure 1.7-a.[3] As it can be seen in Figure 1.7-b, its corresponding joint distribution is piecewise.

The application domain examples presented in the above subsections is sufficient to illustrate the need for a general probabilistic inference algorithm that can robustly

[3]This example is taken from `http://people.seas.harvard.edu/~dduvenaud/talks/probabilistic-programming-introduction.pdf`.

```
x = rand;                      % Draw from Uniform(0,1)
if x < .5;
     y = randn;              % Draw from standard Normal
else
     y = randg + 2.0;          % Draw from Gamma(1,1)
end
```

(a)



(b)

Figure 1.7: (a) Pseudo-code of a simple probabilistic program and (b) corresponding (non-normalized) joint distribution of x and y which is piecewise due to the if-statement in the program.

handle piecewise graphical models. The aim of the current work is to move a step in the direction of designing such an inference tool.

1.2 Contributions

The focus of this thesis is on the design of probabilistic inference tools that perform well in the presence of piecewise and typically discontinuous models. As such, we propose new samplers which perform much better on piecewise models compared to their baseline counterparts. For the reasons that will be mentioned shortly, our contributions are based on Gibbs sampling [Casella and George 1992] and Hamiltonian MCMC [Neal 2011] methods.

1.2.1 Gibbs sampling on piecewise models.

To start with, we concentrate on the Gibbs sampling mechanism: Gibbs sampling is a powerful particle-based tool that in the context of inference on piecewise models, is arguably the first choice. The reason is that in contrast to most samplers, on such models Gibbs sampling performs well in terms of convergence to the true distribution.

Nonetheless, Gibbs sampling has major drawbacks that in practice limit the domain of its usage:

1. Gibbs sampling is typically slow since it requires several (univariate) integrations per sample. More precisely, per sample and for each random variable in the space of the model, Gibbs instantiates the remaining variables and computes a univariate PDF integral. This integration process can be computationally quite costly.

2. Univariate integrations required for Gibbs sampling are not always computable in closed-form.

1.2.1.1 Linear time Gibbs sampling on piecewise models

In our first contribution, we focus on piecewise polynomial density functions. Since in these models the integration can be carried out in closed-form, only the first drawback needs to be addressed. This drawback, nonetheless, can be quite challenging on piecewise models which consist of a large number of partitions. An example of such a scenario has already been provided in Section 1.1.4 where it was shown that in a Bayesian model with piecewise likelihoods, the number of pieces in the posterior density may grow exponentially in the number of observations. As a matter of fact, this example is an instance of the following more general problem: *In a graphical model with piecewise factors, the number of pieces in the joint distribution may grow exponentially in the number of factors.*

Our first contribution is to propose a variation of Gibbs sampling that effectively handles the aforementioned problem and leads to an exponential-to-linear reduction in the complexity over a naive implementation. Our main insight is that a piecewise

model can be treated as a mixture of truncated densities. Drawing a Gibbs sample from the mixture model is equivalent to Gibbs sampling from a fixed number of partitions in the original model. This significantly decreases the amount of computation required for Gibbs sampling. The trade-off is a slower Monte Carlo *mixing rate*. That is, compared to the baseline Gibbs sampling, in order to converge to the true distribution, more samples are required. Nonetheless, as it will be shown, in high dimensional models, the latter effect is not significant and compared to the naive Gibbs sampler implementation, in order to reach an arbitrary precision, our proposed algorithm illustrates an exponential to linear decrease in the total sampling time.

1.2.1.2 Fully-automated symbolic Gibbs sampling on an expressive family of piecewise models

As an orthogonal contribution we deal with the second drawback of Gibbs sampling. We study a highly expressive family of models which are not studied in the literature so far: *Piecewise fractions (of polynomials) with polynomial partitioning constraints*.

Not only for a large range of such models, univariate Gibbs integrals can be computed in closed-form but we can also compute such integrals without instantiating the remaining variables. This immediately leads to our second contribution that is *Fully-automated symbolic Gibbs sampling*.

We create a mapping from random variables to their corresponding analytical integrals which are computed only once and prior to the actual sampling process rather than per sample. Consequently, multiple integrations per sample are reduced to multiple function evaluations per sample which is significantly faster.

It should be pointed out that the target family of models has interesting characteristics such as being closed under a large range of (non)linear random variable transformations. This opens new doors to application domains such as handling deterministic constraints among random variables.

1.2.2 Hamiltonian Monte Carlo on piecewise models

To apply the proposed Gibbs samplers on arbitrary models, they should be approximated by piecewise polynomials or fractional functions. This is a major drawback since such approximations may not be easy to find or the number of required partitions may grow exponentially with dimensionality. On the other hand, like any other class of functions, piecewise polynomial/fractional models have their own shortcomings. Most notably, in graphical models with huge number of factors, computation of the joint posterior requires several polynomial/fractional multiplications which can be sensitive to numerical errors. For such graphical models, the family of (piecewise) exponential distributions is more suitable since in order to multiply distributions/factors of the latter class, it is sufficient to add their logarithms.

Unfortunately, in general Gibbs sampling cannot be directly used on the exponential family. The reason is that Gibbs sampling relies on integrals that in the case of exponential functions, often cannot be solvable in closed-form. In our third contribu-

tion, we focus on sampling from piecewise distributions without any assumption on the form of the modeling functions in any partition. Instead of Gibbs sampling we turn to Hamiltonian Monte Carlo (HMC), a particle-based inference method that has become quite popular in the recent years and has been utilized in the state-of-the-art probabilistic programming language STAN [Stan Development Team 2014].

Unlike Gibbs sampling, HMC works with the logarithm of densities and therefore is less sensitive to numerical errors.[4] HMC does not require any assumption on the class of distributions; therefore, in theory it can be used on any continuous space model. Nonetheless, on the piecewise models, its performance can be disastrously poor. A brief explanation is as follows: HMC sampling is motivated by Hamiltonian dynamics in physical systems. More precisely speaking, in HMC, the negated logarithm of a density is assumed to be a potential energy function and to generate a sample, the trajectory of a particle is followed using the Hamiltonian dynamics. In HMC, Hamiltonian equations are approximated by methods such as *leapfrog integration* or similar algorithms. Such methods can produce very poor simulations if the energy function is discontinuous which is typically the case with piecewise models.

Our contribution is to design a variation of HMC sampling method, called *reflective HMC* (RHMC), the performance of which does not deteriorate in piecewise models. Being motivated by the behavior of Hamiltonian dynamics of physical systems such as optics, we generalize HMC and its leapfrog integration mechanism to become appropriate for piecewise/discontinuous models. For instance, a moving object that hits an obstacle either passes over it or bounces back. Similarly, light may reflect or refract over a refractive surface. In any case, the momentum vector in the collision point (which is a point where the potential energy is discontinuous) is modified in such a way that the system's Hamiltonian remains unchanged.

A key point for the correctness of HMC is the volume preservation property of Hamiltonian dynamics. Due to this property, the Hamiltonian trajectories can be used to define complex mappings without requiring to perform any adjustment via a Jacobian factor [Neal 2011]. It is well known that this property holds even if the dynamics are approximated via the leap frog algorithm or a similar mechanism. We prove that the volume preservation property also holds under reflection and refraction mappings. This guarantees the correctness of the proposed method and as our empirical results show, it can significantly outperform the baseline HMC and be much less sensitive to the parameter tuning.

1.3 Thesis outline

In this chapter we introduced and motivated the research problem that we are interested in: efficient particle-based inference in piecewise graphical models. The remainder of the thesis is structured as follows: Chapter 2 provides the background material required to understand exact and approximate probabilistic reasoning in graphical

[4] HMC has its own problems: unlike Gibbs, it requires input parameters, and its performance can drastically deteriorate if they are not tuned well.

models. The main focus of this chapter would be on particle-based inference methods (sampling). Chapter 3 formalizes piecewise models and their associated data structures. In Chapter 4 we discuss our first proposed method: fast Gibbs sampling via exponential speedup in the number of (piecewise) factors. The performance of the proposed algorithm is evaluated on a Bayesian pairwise preference learning (BPPL) and a Bayesian market making model. In Chapter 5 our second contribution, symbolic Gibbs, is presented. As an application, the focus is mainly on handling deterministic constraints among random variables. Chapter 6 deals with the third contribution, reflective HMC, and compares it to baseline HMC and another sampling tool. Finally, Chapter 7 concludes this thesis and provides a summary and potential direction for future research.

Chapter 2

Probabilistic Graphical Models

This chapter contains background material regarding representation and inference on probabilistic graphical models. Probability theory is a formalism for the analysis of random phenomena. Our primary focus is on multidimensional continuous probability functions p defined in the most natural way. A formal set of definitions and assumptions representing the basic probabilistic model that we are interested in, is presented in Section 2.1. The representation and inference on probabilistic graphical models are dealt with in Sections 2.2 and 2.3 respectively.

2.1 Basic Definitions and Assumptions

Outcome. The result of a single execution of a probabilistic model is called an *outcome*.

Sample space. The set of all possible outcomes is referred to as *sample space* (Ω).

Event. An *event* is a set of zero or more outcomes.

Probability space & measure. A *probability space* $\langle \Omega, \mathcal{F}, p \rangle$ is a tuple of a *sample space* Ω, a collection $\mathcal{F}$ of events (to be considered) that forms a σ-algebra and a *probability measure* p i.e. a function that assigns values (*probabilities*) to events in $\mathcal{F}$. The measure p has to satisfy the following properties:

1. $p(\emptyset) = 0$
2. $p(\Omega) = 1$
3. if (a countable collection of) events $E_1, E_2, \ldots$ are pairwise disjoint sets then the probability value assigned to the union of them is equal to the summation of their probabilities (*countable additivity* property).

As it is commonly done, throughout, in case Ω is countable we assume $\mathcal{F}$ is the power set of Ω and if it is uncountable we let $\mathcal{F}$ be the *Borel algebra* of Ω.[1]

(Univariate) random variable. A (single) *random variable* is a measurable mapping

[1] Borel algebra is the smallest σ-algebra that makes all open sets measurable.

from Ω to another measurable space (*state space*). Throughout we only consider random variables where the set of real numbers is their associated state space. That is,

$$X : \Omega \to \mathbb{R}$$

$p(X = x)$ is a short-form for the probability of the event:

$$\{\omega \in \Omega \,:\, X(\omega) = x\}$$

Probability mass function (PMF) & probability density function (PDF). In case a random variable is discrete (i.e. takes discrete realizations), probability measure is called a *probability mass function* (PMF) and if it is continuous, the measure is called *probability density function* (PDF) or simply *density*. Throughout, when there is no ambiguity about whether the model is discrete or continuous, or if both cases are intended, we may refer to the probability measure simply as the *distribution*.

Multivariate random variable & joint density function. Throughout, we use bold letters to represent vectors of elements. A *multivariate random variable* $\mathbf{X} = (X_1, \ldots, X_n)$ is a vector of random variables X_1 to X_n with the following *joint density function*:

$$p(X_1 = x_1, \ldots, X_n = x_n) := p\{\omega \in \Omega \,:\, X_1(\omega) = x_1, \ldots, X_n(\omega) = x_n\}$$

We denote random variables by capital letters (e.g. X_i) and their realizations by small letters (e.g. x_i). Nevertheless, in case there is no ambiguity, we may omit the random variables (i.e. writing $p(x_1, x_2)$ instead of $p(X_1 = x_1, X_2 = x_2)$). We may also omit the realizations (i.e. writing $p(X)$ instead of $p(X = x)$) in case the statement holds for all realizations.

Marginal distribution. For continuous random variables the following equality always holds:

$$p(x_i) = \int p(x_1, \ldots, x_n)\, \mathrm{d}x_1 \ldots\, \mathrm{d}x_{i-1}\, \mathrm{d}x_{i+1} \ldots\, \mathrm{d}x_n$$

The above integration is called *marginalization* (of $\mathbf{X}_{-i}$ i.e. all random variables except X_i). In this context, $p(x_i)$ is referred to as the *marginal distribution*.

Similarly, if a marginalized random variable is discrete, its associated integration is replaced by summation over its possible values. For instance, in case all $\mathbf{X}_{-i}$ are discrete,

$$p(x_i) = \sum_{x_1 \in \mathrm{VAL}(X_1)} \cdots \sum_{x_{i-1} \in \mathrm{VAL}(X_{i-1})} \sum_{x_{i+1} \in \mathrm{VAL}(X_{i+1})} \cdots \sum_{x_n \in \mathrm{VAL}(X_n)} p(x_1, \ldots, x_n)$$

Cumulative distribution function F (CDF). *(Joint) cumulative distribution function*, denoted as $F(x_1, \ldots, x_n)$ or $\textsc{Cdf}(x_1, \ldots, x_n)$, represents the probability that each random variable X_i takes on a value not greater than x_i:

$$F(x_1, \ldots, x_n) = p(X_1 \leq x_1, \ldots, X_n \leq x_n)$$

Each CDF has the following properties:

1. Being monotonically non-decreasing for each of its variables
2. Being right-continuous for each of its variables
3. $0 \leq F(x_1, \cdots, x_n) \leq 1$
4. $\lim_{x_1 \to +\infty, \cdots, x_n \to +\infty} F(x_1, \ldots, x_n) = 1$
5. $\lim_{x_i \to -\infty} F(x_1, ..., x_n) = 0, \quad \text{for all } i = 1, \ldots, n$

Clearly, the density function can be derived from the CDF as follows:

$$p(x_1, \ldots, x_n) = \frac{\partial^n F(x_1, \ldots, x_n)}{\partial x_1 \cdots \partial x_n}$$

Conditional probability. By definition, the conditional probability of a random variable or a vector of random variables (or more generally, an event) $\mathbf{X}$ given *observed* random variable(s) (resp. event) $\mathbf{Y}$ is,

$$p(\mathbf{X} \,|\, \mathbf{Y}) := \frac{p(\mathbf{X}, \mathbf{Y})}{p(\mathbf{Y})} \tag{2.1}$$

Independence. Let $\mathbf{X}$ and $\mathbf{Y}$ be a pair of random variable(s) (or more generally, events). We denote $\mathbf{X} \perp \mathbf{Y}$ and say $\mathbf{X}$ and $\mathbf{Y}$ are independent if and only if,

$$p(\mathbf{X} \,|\, \mathbf{Y}) = p(\mathbf{X}) \tag{2.2}$$

Clearly, if $\mathbf{X} \perp \mathbf{Y}$ then $\mathbf{Y} \perp \mathbf{X}$ since relations (2.1) and (2.2) imply,

$$\mathbf{X} \perp \mathbf{Y} \quad \Rightarrow \quad p(\mathbf{X}, \mathbf{Y}) = p(\mathbf{X}) \cdot p(\mathbf{Y})$$

Conditional independence. A pair of random variables $\mathbf{X}$ and $\mathbf{Y}$ (or more generally two events) are *conditionally independent* given a third random variable $\mathbf{Z}$ (resp. event) (denoted as: $\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z}$) if and only if,

$$p(\mathbf{X}, \mathbf{Y} \,|\, \mathbf{Z}) = p(\mathbf{X} \,|\, \mathbf{Z}) \cdot p(\mathbf{Y} \,|\, \mathbf{Z})$$

Expected value. The *expected value* (or *expectation*) of a (continuous) random variable X (w.r.t. a reference distribution p) is,

$$\mathbb{E}[X] := \int x \cdot p(x) \, \mathrm{d}x$$

A similar definition exists for discrete random variables where instead of integration, summation over all possible values of X is computed:

$$\mathbb{E}[X] := \sum_{x \in \mathrm{VAL}(X)} x \cdot p(x)$$

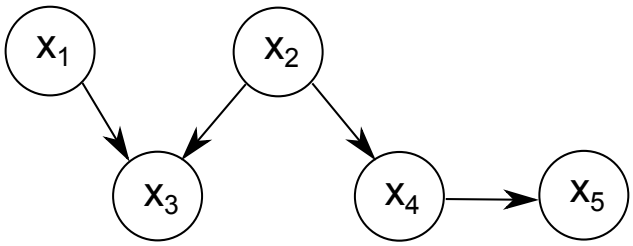


Figure 2.1: A Bayesian network representing the joint probability distribution over random variables X_1 to X_5 for the example provided in subsection 2.2.1.

2.2 Representation of Graphical models

The framework of probabilistic graphical models (GMs) provides graph structured specifications of probabilistic models. Using GMs, joint distributions are represented in compact and factorized manners by means of exploiting conditional dependencies between random variables.[2] Such dependency structures are used to design automated, scalable and effective inference algorithms for the associated probabilistic models.

In this section we focus on introducing common graphical model representations while the associated inference algorithms are postponed to Section 2.3. Two main families of graphical models are:

1. directed acyclic models, known as *Bayesian networks* (BNs), and
2. undirected models referred to as *Markov random fields* (MRFs).

After introducing these two families of GMs, we present a more general framework, namely *factor graphs*, that encompasses both aforementioned families.

2.2.1 Bayesian networks (BNs)

A *Bayesian network* also known as a *probabilistic directed graphical model* is a directed acyclic graph (DAG) suitable for representation of conditional dependencies of random variables suitable. In such networks, each vertex represents a distinct random variable (therefore, the terms *vertex* and *random variable* may be used interchangeably throughout). Whenever an arrow between two vertices is missing, it means that their corresponding random variables are (conditionally) independent. For example, consider a probability distribution over five variables X_1 to X_5. The general form of such a distribution can be factorized as follows:

$$p(x_1, x_2, x_3, x_4, x_5) = \\ p(x_1) \cdot p(x_2 \mid x_1) \cdot p(x_3 \mid x_1, x_2) \cdot p(x_4 \mid x_1, x_2, x_3) \cdot p(x_5 \mid x_1, x_2, x_3, x_4)$$

[2]The main references of this section are [Bishop 2006; Koller and Friedman 2009; Russell and Norvig 2003].

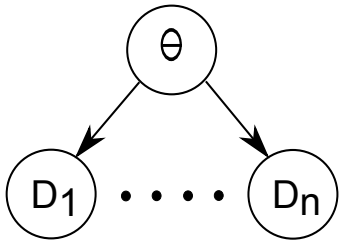


Figure 2.2: A Bayesian network representing the Bayesian paradigm.

Nevertheless, The Bayesian network depicted in Figure 2.1 visualizes the case where X_1 and X_2 are independent (i.e. $X_1 \perp X_2$) (since no arrow links them) and given X_2, X_3 and X_4 are independent (i.e. $X_3 \perp X_4 \mid X_2$) , etc. It can be seen that in such a case, the aforementioned factorization can be simplified to,

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1) \cdot p(x_2) \cdot p(x_3 \mid x_1, x_2) \cdot p(x_4 \mid x_2) \cdot p(x_5 \mid x_4)$$

More generally, if we let $\mathbf{X} = \{X_1, \ldots, X_n\}$ be the set of all relevant random variables (i.e. the set of all vertex labels in the model) and the set of parents of a vertex $X_i \in \mathbf{X}$ in a Bayesian network be represented by $\text{PA}(X_i)$, that network represents a joint distribution that factorizes into,

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i \mid \text{PA}(x_i)) \tag{2.3}$$

Bayesian networks and conditional independence. Each Bayesian network has the following properties:

- Local Markov property: Any random variable is conditionally independent of its non-descendant vertices given it's parent variables,

  $$X_i \perp \mathbf{X} \backslash \text{DE}(X_i) \mid \text{PA}(X_i)$$

  where $\text{DE}(X_i)$ denotes the set of descendants of X_i (including itself).

- Each random variable is conditionally independent of all other variables in the network given its *Markov blanket* i.e. its parents, its children and all other parents of its children.

2.2.1.1 Bayesian inference / Bayesian paradigm

Bayesian paradigm boils down the uncertainty about the model of the world into maintaining a belief distribution over a univariate or multivariate *parameter* Θ that is updated via observation of *data* random variables D_1 to D_n. Given Θ, it is assumed

that D_1 to D_n are identically distributed and independent from each other:

$$D_i \perp D_j \mid \Theta \qquad \forall i, j \neq i \in \{1, \dots n\}$$

Figure 2.2 depicts a Bayesian network representing this paradigm where Θ is regarded as a hidden random variable and D_1 to D_n as observed vertices. According to the topology of this network,

$$p(\Theta, D_1, \dots, D_n) = p(\Theta) \prod_{i=1}^{n} p(D_i \mid \Theta)$$

In addition, by the definition of conditional probability,

$$p(\Theta \mid D_1, \dots, D_n) = \frac{p(\Theta, D_1, \dots, D_n)}{p(D1, \dots, D_n)} \propto p(\Theta, D1, \dots, D_n)$$

Combining these two relations, it is clear that the *posterior* distribution $p(\Theta \mid D_1, \dots, D_n)$ is proportional to the product of the *prior* distribution $p(\Theta)$ and *likelihood* functions $p(D_1 \mid \Theta)$ to $p(D_n \mid \Theta)$,

$$p(\Theta \mid D_1, \dots, D_n) \propto p(\Theta) \cdot \prod_{i=1}^{n} p(D_i \mid \Theta) \tag{2.4}$$

2.2.2 Markov random fields (MRFs)

Markov random fields (MRF) also known as *undirected graphical models* form the second major class of graphical models. In contrast to BNs, an MRF network is a graph representation where edges do not carry directional information. MRFs provide a different perspective of the independence structure among random variables and are more suitable for modeling phenomena where a directionality cannot be naturally ascribed to the interaction between random variables.

Let G be an undirected graph where the vertices represent a set of random variables $\mathbf{X} = \{X_1, \dots, X_n\}$ and $\text{CL}(G)$ represent the set of *maximal cliques*[3] of G. Also let for any $C \in \text{CL}(G)$, $\text{VA}(C)$ is the set of variables in C. With respect to G, $\mathbf{X}$ forms a Markov random field if the joint distribution can be factorized over CL(G) as,

$$p(\mathbf{X}) = \frac{1}{Z} \prod_{C \in \text{CL}(G)} \psi_C(\text{VA}(C)) \tag{2.5}$$

In the above equation, Z is a normalization constant and ψ_C is a strictly positive function called a *potential function* of clique C. If potential functions are represented by exponentials[4],

$$\psi_C(\text{VA}(C)) = \exp(-E(\text{VA}(C)))$$

[3] A *maximal clique* of an undirected graph, G, is a subgraph of it that is a complete graph and is not a proper subset of any subgraph of G which is a complete graph.

[4] Such a representation is called *Boltzmann distribution*.

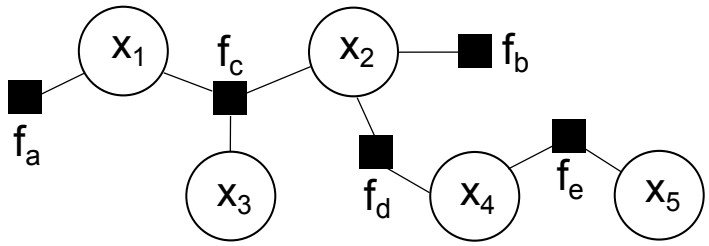


Figure 2.3: A *factor graph* representing the same distribution as the Bayesian network in Figure 2.1

E is called an *energy function*. Obviously, in case a relatively low (or high) energy is assigned to a set of variable values, a higher (resp. lower) probability is assigned to those values.

Markov random fields and conditional independence. Each MRF associated with a model with random variables $\mathbf{X} = \{X_1, \dots X_n\}$ has the following properties:

- Pairwise Markov property: Any two distinct random variables X_i and X_j that are not connected by a link are conditionally independent given the rest of variables:
$$p(x_i, x_j \,|\, \mathbf{x}\backslash\{x_i, x_j\}) = p(x_i \,|\, \mathbf{x}\backslash\{x_i, x_j\}) \cdot p(x_j \,|\, \mathbf{x}\backslash\{x_i, x_j\})$$
or more concisely:
$$X_i \perp X_j \mid \mathbf{X}\backslash\{X_i, X_j\}$$

- Local Markov property: Any variable X_i is conditionally independent of all other variables given the set of its neighbors $\text{NE}(X_i)$:
$$X_i \perp \mathbf{X}\backslash(\text{NE}(X_i) \cup \{X_i\}) \mid \text{NE}(X_i))$$

- Global Markov property: Let A and B be two distinct subsets of $\mathbf{X}$ (equivalently, two subgraphs of a MRF G) and C be a *separating subset* i.e. a subset of G such that any graph path containing a member of A and a member of B passes through (i.e. contains a member of) C:
$$A \perp B \mid C$$

2.2.3 Factor graphs

Bayesian networks and Markov random fields are two subsets of a more general framework: *Factor graphs*.

In the latter framework, the joint distribution (over a set of variables $\mathbf{X}$) is factor-

ized as:[5]

$$p(\mathbf{X}) = \prod_{s=1}^{m} f_s(\mathbf{X}_s) \tag{2.6}$$

where as before, $\mathbf{X} = \{X_1, \ldots, X_n\}$, for all $1 \leq s \leq m$, each $\mathbf{X}_s \subset \mathbf{X}$ and f_s is a function of them. Clearly, both factorization mechanisms of relations (2.3) and (2.5) can be seen as special cases of relation (2.6).

Factor graph structure. The factor graph of an *objective function* $p(\cdot)$ is a data structure visualizing how variables $\mathbf{X}$ and local functions $\{f_i\}_{i=1}^{m}$ factorize $p(\mathbf{X})$. It is a bipartite graph with *n variable-typed* nodes and *m factor-typed* nodes. Each variable node represents an (observable/ hidden) variable in $\mathbf{X}$ and each factor node represents a local function such that the node corresponding f_k is linked to the variable nodes corresponding $\mathbf{X}_k$.

As an instance, Figure 2.3 is a factor graph representation of the distribution which is visualized by Figure 2.1. Here, factor nodes correspond to the following functions:

$$f_a = p(x_1) \qquad f_b = p(x_2) \qquad f_c = p(x_3 \mid x_1, x_2)$$

$$f_d = p(x_4 \mid x_2) \qquad f_e = p(x_5 \mid x_4)$$

2.3 Inference in Graphical Models

The task of inference on a graphical model is to compute the posterior probability distribution of some *query variables* (corresponding some *hidden* nodes), given a set of *evidence variables* (*observed* nodes) [Russell and Norvig 2003].

Let $\mathbf{Q} = \{Q_1, \ldots, Q_m\}$ be the set of query variables, $\mathbf{E} = \{E_1, \ldots, E_n\}$ be the set of evidence variables and $\mathbf{U} = \{U_1, \ldots, U_k\}$ be the set of variables neither mentioned in the query nor in the evidence. Also let $(\mathbf{Q} = \mathbf{q})$, be the short form for,

$$(Q_1 = q_1, \ldots, Q_m = q_m)$$

$(\mathbf{E} = \mathbf{e})$ be the short form for,

$$(E_1 = e_1, \ldots, E_n = e_n)$$

and $(\mathbf{U} = \mathbf{u})$ be a short form for $(U_1 = u_1, \ldots, U_k = u_k)$, etc.

By Bayes rule, the posterior probability of a query $\mathbf{Q}$ given an evidence $\mathbf{E}$ is proportional to the joint distribution of $\mathbf{Q}$ and $\mathbf{E}$. Therefore, the random variables of the Bayesian network neither in $\mathbf{Q}$ nor in $\mathbf{E}$ should be marginalized out.

[5] Instead of the product, in some contexts, factorization is performed by summation (see e.g. [Yedidia 2011]).

$$p(\mathbf{Q}=\mathbf{q} \mid \mathbf{E}=\mathbf{e}) = \frac{p((\mathbf{Q}=\mathbf{q}),(\mathbf{E}=\mathbf{e}))}{p(\mathbf{E}=\mathbf{e})} = \frac{\int_{u_1=-\infty}^{+\infty} \dots \int_{u_k=-\infty}^{+\infty} p(\mathbf{U}=\mathbf{u},\mathbf{Q}=\mathbf{q},\mathbf{E}=\mathbf{e})\, \mathrm{d}u_1 \dots\, \mathrm{d}u_k}{\int_{u_1=-\infty}^{+\infty} \dots \int_{u_k=-\infty}^{+\infty} \int_{r_1=-\infty}^{+\infty} \dots \int_{r_m=-\infty}^{+\infty} p(\mathbf{U}=\mathbf{u},\mathbf{Q}=\mathbf{r},\mathbf{E}=\mathbf{e})\, \mathrm{d}u_1 \dots\, \mathrm{d}u_k\, \mathrm{d}r_1 \dots\, \mathrm{d}r_m}$$

Clearly, if a marginalized variable is discrete, instead of integration, the summation over its possible values should be computed. For instance, in a network where $\mathbf{U}$ and $\mathbf{Q}$ are discrete:

$$p(\mathbf{Q}=\mathbf{q} \mid \mathbf{E}=\mathbf{e}) = \frac{p((\mathbf{Q}=\mathbf{q}),(\mathbf{E}=\mathbf{e}))}{p(\mathbf{E}=\mathbf{e})} = \frac{\sum_{u_1 \in \mathrm{VAL}(U_1)} \dots \sum_{u_k \in \mathrm{VAL}(U_k)} p(\mathbf{U}=\langle u_1,\dots,u_k\rangle,\mathbf{Q}=\mathbf{q},\mathbf{E}=\mathbf{e})}{\sum_{u_1 \in \mathrm{VAL}(U_1)} \dots \sum_{u_k \in \mathrm{VAL}(U_k)} \sum_{r_1 \in \mathrm{VAL}(Q_1)} \dots \sum_{r_m \in \mathrm{VAL}(Q_m)} p(\mathbf{U}=\langle u_1,\dots,u_k\rangle,\mathbf{Q}=\langle r_1,\dots,r_m\rangle,\mathbf{E}=\mathbf{e})}$$

where $\mathrm{VAL}(V_i)$ is the set of possible values of random variable V_i.

In either case, calculations required for the computation of marginal distributions are quite expensive. As a matter of fact, the most important challenge facing probabilistic inference is to carry out marginalization efficiently. As such, the inference techniques that will be presented shortly, try to perform this task in an effective way.

2.3.1 Exact inference

Exact inference is possible only if the marginal distributions have closed form solutions. For continuous random variables, this is often not the case. As such, in this section our focus is on discrete models.

As mentioned in the previous chapter, both Bayesian networks and Markov random fields are subsets of factor graph. As a result, to cover both cases, we present the inference algorithms on the factor graphs only.

The main task of inference in factor graphs is to compute an expression of a form that is known as *sum-product* when random variables are discrete:

$$\sum_{Z_k} \cdots \sum_{Z_1} \prod_{\phi \in \Phi} \phi$$

(or more concisely $\sum_{\mathbf{Z}} \prod_{\phi \in \Phi} \phi$) where Φ is a set of factors[6] and $\mathbf{Z} = \{Z_1, \dots, Z_k\}$ is the set of variables that should be marginalized out. The following exact inference techniques find ways to perform such a computation with minimum repeated operations.

[6]Each factor ϕ is a function from $\mathrm{VAL}(X_1) \times \dots \mathrm{VAL}(X_n)$ to $\mathbb{R}$ where $\mathbf{x} = \{X_i\}_{i=1}^{n}$ is the set of all (relevant) random variables and $\mathrm{VAL}(X_i)$ is the set of possible values of variable X_i.

2.3.1.1 Variable elimination (VE)

Algorithm 1: Variable elimination algorithm for factor graphs (following [Koller and Friedman 2009])

procedure sum-product-VE (
  Φ /*set of factors*/,
  $\mathbf{Z}$ /*k-tuple of variables $Z_1, \ldots, Z_k$ to be eliminated sequentially*/)

for $i = 1$ **to** k **do**
  $\Phi \leftarrow$ sum-product-eliminate-a-var(Φ, Z_i)
end for
return $\prod_{\phi \in \Phi} \phi$

procedure sum-product-eliminate-a-var (
  Φ /*set of factors*/,
  Z /*a variable to be eliminated*/)

/*create Φ', the set of all factors that involve Z: */
$\Phi' \leftarrow \{\phi \in \Phi \mid Z \in \text{SCOPE}(\phi)\}$
/*create Φ'', the set of all factors that do not involve Z: */
$\Phi'' \leftarrow \Phi \backslash \Phi'$
/*create ψ, the product of the factors that involve Z: */
$\psi \leftarrow \prod_{\phi \in \Phi'} \phi$
/* make a new factor τ by summing out Z on ψ: */
$\tau \leftarrow \sum_Z \psi$
/*return the set of factors that do not involve Z plus a factor that is made by summing out Z in the factors that involve it: /*
return $\Phi'' \cup \{\tau\}$

The idea behind the variable elimination algorithm is the observation that the scope of each factor (i.e. the set of variables involved in that factor) is often a small subset of the total model random variables and this allows us to utilize the reverse distributive law and *push in* each $\sum_{Z_i}$ to the right till it operates only on the factors that have the variable Z_i in their scope.

In order to calculate the sum-product $\sum_{\mathbf{Z}} \prod_{\phi \in \Phi} \phi$, the VE algorithm eliminates (i.e. marginalizes) one variable Z_i at a time. It finds the set Φ' of factors that have Z_i in their scope, multiplies them to make a new factor ψ and finally, sums out Z_i on ψ to make a new factor that should replace Φ' in Φ (see Algorithm 1) [Koller and Friedman 2009].

As an example consider the a set of (discrete) reference random variables $\mathbf{X}$ and a set of factors Φ, defined as follows:

$$\mathbf{X} := \{A, B, C, D\} \qquad \Phi := \{\phi_A, \phi_{AB}, \phi_{BC}, \phi_{CD}\}$$

where the subtitle of each factor indicates the variables mentioned in it. For instance,

$$\text{SCOPE}(\phi_{BC}) = \{B, C\}$$

If the ordered list of variables that should be eliminated is $\mathbf{Z} = \langle A, B, C \rangle$ (so that variables are eliminated in the alphabetic order), the final factor produced by VE is:

$$f_1(D) := \sum_C \phi_{CD} \sum_B \phi_{BC} \sum_A (\phi_{AB} \phi_A)$$

One the other hand, if the variables are eliminated in the reverse order i.e. $\mathbf{Z}' = \langle C, B, A \rangle$, the result of VE will be:

$$f_2(D) := \sum_A \phi_A \sum_B \phi_{AB} \sum_C (\phi_{CD} \phi_{BC})$$

Both $f_1(\cdot)$ and $f_2(\cdot)$ are clearly equivalent to the following (non-factorized) function:

$$f(D) := \sum_C \sum_B \sum_A (\phi_A \phi_{AB} \phi_{BC} \phi_{CD})$$

Nevertheless, either $f_1(\cdot)$ or $f_2(\cdot)$ can be computed much more efficiently than $f(\cdot)$ because intermediate factors that are created by marginalization are simpler than the original factors.

2.3.1.2 Clustering (join tree) algorithms

Clustering algorithms provide an effective way to compute posterior probabilities for each variable in the network. In this approach an algorithm called *belief propagation* is executed over a data structure called *clique tree*. These concepts are defined below:

Cluster tree. A cluster tree U for a set of factors Φ over a set of random variables $\{X_1, \ldots, X_n\}$ is an undirected graph such that its nodes are (associated with) *clusters* (a.k.a. *cliques*) C_i which correspond to subsets of random variables in the original network, that is:

$$C_i \subseteq \{X_1, \ldots, X_n\},$$

and any edge linking clusters C_i and C_j is associated with a set of random variables called *sepset* $S_{i,j}$:

$$S_{i,j} = C_i \cap C_j$$

A cluster tree must preserve the following two properties:

1. *Family preservation property*: For each factor ϕ_k (in Φ), there should exist (at least one) cluster $C_{\alpha(k)}$ in U such that $\text{Scope}[\phi_k] \subseteq C_{\alpha(k)}$. Otherwise stated, cluster $C_{\alpha(k)}$ should be able to accommodate factor ϕ_k. This property guarantees that U allows Φ to be encoded.

2. *Running intersection property*: For any variable X_i, the set of nodes and edges of U that contain X_i form a tree. This property guarantees that U connects all

information about all variables without any feedback loops.

Clique tree. A clique tree (also know as *join tree* or *junction tree*) is a directed tree simply made by orienting a cluster tree. To do so, an arbitrary node in the cluster tree is designated as the root and the edges are directed toward this node.

Belief propagation (BP). BP is a *message passing*[7] [Yedidia 2011] algorithm applied to a clique tree to carry out inference. The idea is to catch intermediate factors computed during variable elimination and pass them via messages. This algorithm consists of the following steps:

1. *Assigning factors to clusters*: Given a set of factors Φ, the model is initialized with assigning each $\phi_k \in \Phi$ to a cluster $C_{\alpha(k)}$ such that $\text{Scope}[\phi_k] \subseteq C_{\alpha(k)}$ (this can be done with respect to the family preservation property).

2. *Constructing initial potentials*: To each cluster (node) C_i, an initial potential $\psi_i(C_i)$ is assigned as follows:
$$\psi_i(C_i) = \prod_{k:\alpha(k)=i} \phi_k$$

3. *Message initialization*: All messages $\delta_{i\to j}$ (passing from cluster nodes C_i to C_j) are initialized to 1.

4. *Messaging cycle*: Edges are sorted/visited from the leaves to the root. Whenever an edge (i, j) is visited, firstly the potential assigned to cluster C_i is updated:
$$\psi_i(C_i) \;\leftarrow\; \psi_i(C_i) \times \prod_{k\in(N_i\setminus\{j\})} \delta_{k\to i},$$
where N_i is the set of neighboring nodes of C_i. Then, the $\delta_{i\to j}$ (the message from node C_i to node C_j) is updated:
$$\delta_{i\to j} = \sum_{C_i\setminus S_{i,j}} \psi_i(C_i)$$
Informally speaking, $\delta_{i\to j}(S_{i,j})$ is generated as follows: Firstly, the product of all incoming messages to C_i (except the one received from C_j) times the initial potential assigned to C_i is computed and then the variables in C_i that are not in $S_{i,j}$ are summed out from it.

5. *Updating the root potential:* In the end, the potential associated with the root, say C_r, is updated:
$$\psi_r(C_r) \;\leftarrow\; \psi_r(C_r) \times \prod_{k\in(N_r)} \delta_{k\to r}$$

The resulting potentials ψ_i are proportional to the joint marginal distribution of the set of variables in cliques C_i.

[7] *Message passing* algorithms are a group of solutions to the optimization problem when the problem is represented as a factor graph. They get their name from the fact that in each step, messages are sent between nodes in the factor graph.

2.3.2 Approximate inference by sampling (Monte Carlo methods)

When the networks are large and the nodes are connected via a huge number of links, exact inference becomes intractable and computationally prohibitive. In such situations, the joint distribution has to be computed by means of approximate inference techniques. On the other hand, regardless of the size and the topology of the network, if a model is continuous (or discrete/continuous hybrid) the integrations required for marginalization often do not have closed-form solution which also necessitates approximate solutions.

The most general and therefore, arguably, the most important class of approximate inference techniques is known as Monte Carlo sampling methods (a.k.a. *particle-based or sampling-based methods*).

The idea is to take samples from a probability distribution by means of constructing a Markov chain that has the original distribution as its stationary distribution.

An important characteristic of Monte Carlo-based approximate inference is that these methods provide *asymptotically unbiased solutions* in the sense that if the number of samples drawn from a density function is sufficiently large, the approximation error will be arbitrarily small and in the limit tends to zero.

Prior to describing sampling mechanisms,we introduce some terminology:

Particle (sample). A *particle* ξ, drawn from a distribution $p(\mathbf{X} = \mathbf{x})$ assigns appropriate values,

$$\xi(X_i) \in \textsc{val}(X_i)$$

to each random variable $X_i \in \mathbf{X}$. If

$$\mathbf{Y} = \{Y_1, \ldots, Y_k\} \subseteq \mathbf{X}$$

by $\xi\langle\mathbf{Y}\rangle$ we denote the event $\{Y_i = \xi(Y_i)\}_{i=1}^{k}$.

Sampling. Sampling is the practice of simulating a distribution by means of a sequence of (pseudo)randomly generated particles (or samples) that are distributed according to that distribution. From a set of such particles,

$$\Xi = \{\xi_1, \ldots, \xi_N\}$$

the expectation of each function f can be approximated by

$$\hat{\mathbb{E}}_{\Xi}(f) = \frac{1}{N}\sum_{i=1}^{N} f(\xi_i)$$

More specifically, by substituting $f(\xi_i)$ by $\mathbb{1}[\xi_i\langle\mathbf{Y}\rangle = \mathbf{y}]$ [8]:

$$\hat{p}_{\Xi}(\mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[\xi_i\langle\mathbf{Y}\rangle = \mathbf{y}]$$

[8] If $c \equiv \top$, $\mathbf{1}[c]$ returns 1 otherwise 0.

which is the number of particles in which $\mathbf{Y}$ is valuated by $\mathbf{y}$ divided by the total number of particles.

2.3.2.1 Inverse transform sampling

A central prerequisite for sampling from arbitrary distributions is the ability to take samples from a uniform distribution. This can be done by various *pseudorandom number generation* (PRNG) algorithms or by using *hardware random number generators*. Since modern programming languages are equipped with this ability, PRNG algorithms are not covered by this thesis.

Being equipped with a uniform sampler, by a simple mechanism, one can generate samples for an arbitrary continuous distribution if the (inverse of) cumulative distribution function (CDF) of the target distribution is provided via a mechanism known as *inverse transform sampling*. We only consider the univariate case. Generalization to the multivariate case is also feasible and based on the idea of mapping *elementary events* to the interval $[0, 1]$ by associating intervals to them proportional to their probabilities and then uniformly taking sample from the interval.

In order to draw a sample from a continuous distribution $p(X)$ associated with CDF $F(X)$, it suffices to uniformly sample a number u between 0 and 1, i.e.

$$u \sim \mathcal{U}(0, 1)$$

and return $x = F^{-1}(u)$ i.e. a value $x \in \text{VAL}(X)$ such that $F(x) = u$.

The proof of correctness (which is a slightly modified version of the proof presented in [Devroye 1986, Chap. 2]) is as follows:

Proposition 1. *Let F be a continuous CDF on $\mathbb{R}$ with inverse function F^{-1}. If U is random variable uniformly distributed over $[0, 1]$, then the CDF of $F^{-1}(U)$ is F.*

Proof. We should show that $p(F^{-1}(U) \leq x) = p(X \leq x)$.

$$\begin{aligned} p\big(F^{-1}(U) \leq x\big) &= p\big(F(F^{-1}(U)) \leq F(x)\big) && \text{, since } F \text{ is monotonic} \\ &= p\big(U \leq F(x)\big) \\ &= F(x) && \text{, since } U \text{ is uniform hence } pr(U \leq m) = m. \end{aligned}$$

□

Algorithm 2 illustrates a variation of the inverse transform sampling that

1. Does not require the inverse of CDF being known. (F^{-1} is approximated.)
2. Does not require the CDF to be normalized i.e. works with any function being proportional to the target CDF.

The first property is due to the fact that cumulative distribution functions are monotonically increasing function and therefore can be approximated via *binary search*.

Algorithm 2: INVERSETRANSFORMSAMPLING($F(\cdot)$)

Input: $F(x)$, (non-normalized) cumulative distribution function of a (univariate) random variable X. (That is, a function proportional to $p(X \leq x)$.)
Output: a samples drawn for X via *inverse transform sampling*.

/* Uniformly sample a number u in $[0, F(\infty)]$: */
$u \sim \mathcal{U}(0, F(\infty))$
/* return the approximation of $F^{-1}(u)$: */
Return INVERSE($F, u, 0, F(\infty)$)

// Approximates $F^{-1}(u) = \inf\{x : F(x) = u, a \leq u \leq b\}$ via binary search.
procedure INVERSE(
$F(\cdot)$ /*a univariate monotone function*/,
u /*a value for which the inverse function has to be computed*/,
a, b /*associated lower and upper bounds*/)

$l_1 \leftarrow a; l_2 \leftarrow b$
while true **do**
$\Delta = F(\frac{l_1+l_2}{2}) - u$
/* ϵ is a constant indicating the maximum acceptable approximation error. */
if $|\Delta| < \epsilon$ **then return** $\frac{l_1+l_2}{2}$
if $\Delta > 0$ **then** $l_2 \leftarrow \frac{l_1+l_2}{2}$ **else** $l_1 \leftarrow \frac{l_1+l_2}{2}$
end while

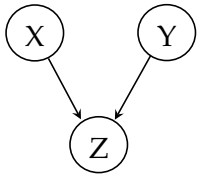
The second property is due to the fact that if a CDF is not normalized it converges to the normalization constant and therefore, in practice the latter constant can be approximated by evaluating the (non-normalized) CDF function for a large value.

Clearly, in case the support of the target distribution is finite, that is, the distribution is surrounded by a 0-probability space, the aforementioned normalization constant can be computed precisely: it is sufficient to evaluate the CDF for a value more than the upper bound of the support.

2.3.2.2 Forward sampling

If the task of sampling from a Bayesian network is to estimate marginal probabilities without being conditioned on particular observations, that is, if we intend to draw a sample from a prior distribution, the task can be accomplished in a fairly easy manner: It is sufficient to sample parent nodes prior to their children.

For instance, consider a simple BN over random variables X, Y and Z where no vertex is observed and Z is conditionally dependent on the other variables (as depicted in the right hand side). In this example, variable sampling order should either be $\langle X, Y, Z \rangle$ or $\langle Y, X, Z \rangle$. This process is called *forward* or *ancestral* sampling. More formally, in order to assign a sam-



ple value to a vertex, forward sampling is performed on its associated distribution conditioned on the sample values already assigned to its parent nodes (See Algorithm 3).

Algorithm 3: A simple sampling on the prior distributions in a Bayesian network

procedure FORWARDSAMPLING (
$\mathcal{B}$ /*a Bayesian network over $\mathbf{X} = \{X_1, \ldots, X_n\}$ where indices indicate a topological ordering*/)

Let ξ be a particle vector that should be returned.
for $i = 1$ **to** n **do**
/* Sample X_i conditioned on the samples already taken from its parents:*/
$\xi(X_i) \sim p(X_i \mid \xi\langle \text{PA}(X_i)\rangle)$
end for
return ξ

The rest of this subsection deals with sampling from conditional distributions (e.g. $p(\mathbf{Y} = \mathbf{y} \mid \mathbf{E} = \mathbf{e})$). To be more specific, consider the problem of sampling from a Bayesian network with observed nodes $\mathbf{E} = \{E_1, \ldots, E_k\}$. In this case, the generated samples should comply with the observation. Forward sampling cannot be directly used to sample these nodes and their parents/ancestors (although the particle values of the remaining nodes can still be generated by forward sampling as before). As a specific form of rejection sampling (that will be outlined in Section 2.3.2.3), a simple solution is to sample the prior distribution and discard the particles that do not comply with the observation. This technique can only be applied to discrete distributions and even in that case, if the probability of the evidence is low, the rate of discarded samples can be arbitrary large.

Therefore, sampling from posterior distributions (and in particular, continuous space posteriors) is a major challenge and in the remaining part of this subsection we deal with the relevant solutions proposed in the literature.

Before presenting such techniques, it should be pointed out that in contrast with priors distributions that often have simple forms (e.g. are uniform of normal) the posterior distributions can in particular be complicated functions e.g. may be multimodal, disconnected and piecewise.

2.3.2.3 Rejection sampling

Rejection sampling (a.k.a. *acceptance-rejection* algorithm) is a basic Monte Carlo based sampling method.

Algorithm 4 illustrates a simple version of the rejection sampling algorithm where in order to approximate a posterior distribution in a Bayesian network, samples are taken from the corresponding prior distribution; however, the samples that are inconsistent with the evidence are rejected and then based on the non-rejected (i.e. accepted) samples, the posterior distribution is approximated.

Algorithm 4: Rejection sampling from a posterior distribution associated with a Bayesian network.

procedure REJECTIONSAMPLING (
$\mathcal{B}$ /*a Bayesian network over $\mathbf{x} = \{X_1, \ldots, X_n\}$ where indices indicate a topological ordering*/,
E /*set of random variables that their values are determined by evidence s.t. the value of $E_i \in \mathbf{E}$ is accessible by $\text{VAL}(E_i)$*/,)

Let ξ be a particle that should be returned.
repeat
  particle-rejected $\leftarrow \bot$
  for $i = 1$ **to** n **do**
    $\xi(X_i) \leftarrow$ take a sample from $p(X_i \mid \xi\langle \text{PA}(X_i)\rangle)$
    if $X_i \in \mathbf{E}$ & $\xi(X_i) \neq \text{VAL}(X_i)$ **then**
      particle-rejected $\leftarrow \top$
      break
    end if
  end for
until ¬particle-rejected
return ξ

A more general version of rejection sampling facilitates generating particles from any distribution $f(\mathbf{x})$ from which direct sampling is difficult, provided with an auxiliary distribution $g(\mathbf{x})$ from which sampling is easier and it is known that $\frac{f(\mathbf{x})}{g(\mathbf{x})}$ is bounded from above by a constant $c > 1$. That is, $c \cdot g(\mathbf{x})$ is an *envelope* for $f(\mathbf{x})$.

As illustrated in algorithm 5, to take a sample from f using the latter version of rejection sampling, a sample $\mathbf{x}$ is taken from g and accepted with probability,

$$f(\mathbf{x}) / \big(c \cdot g(\mathbf{x})\big)$$

otherwise it is rejected and the process is repeated. Note that for the correctness of convergence to the true target distribution, neither f nor g are required to be normalized.

Advantages and disadvantages. The main advantage of rejection sampling is the fact that (unlike the *Markov chain Monte Carlo* methods that will be introduced shortly) the sampled particles are independent from each other. This, in its turn, leads to a rapid rate of convergence to the target distribution as the number of generated samples grows.

On the other hand, the major disadvantage of this method is that the sampling time increases exponentially in the variable space dimensionality. Clearly, if the envelope constant c is too large, the speed of this algorithm is slow since often lots of samples are required until one is accepted. However, even if c is small (i.e. envelope is tight) the curse of dimensionality is often inevitable.

As a very simple example consider the problem of rejection sampling from an n-dimensional target distribution that is positive in a hyper-cubic region with side length 0.9. Suppose that the envelope is a uniform function which also has a hyper-cubic support but its side length is 1.0. Even if the envelope is quite tight, at least

$$\frac{1.0^n - 0.9^n}{0.9^n} \times 100$$

percent of the samples are expected to be taken from the gap between the two hyper-cubes and therefore rejected. Clearly, since the proportional volume of this gap grows exponentially in the dimensionality of the random variable space, beyond some number of variables, rejection sampling becomes prohibitively slow and in the limit 100% of the proposed samples are rejected.

2.3.2.4 Sampling by likelihood weighting (LW)

Rejection sampling is an inefficient method since many proposed particles are simply dropped. A better approach is to restrict the taken samples to the regions of the space which are consistent with the evidence. In order to restrict the sampling to such events, in LW sampling method, the observed values are fixed, and the hidden values are sampled and weighted by the likelihood that they are consistent with the evidence. What is returned by one iteration of LW sampling algorithm is a pair $\langle \xi, w \rangle$ containing a particle and its weight (See Algorithm 6). Given the weight of each sampled particle, the posterior probability can be approximated by,

$$\hat{p}_\Xi(\mathbf{y} \mid \mathbf{E} = \mathbf{e}) = \frac{\sum_{i=1}^N w_i \mathbb{1}[\xi_i \langle \mathbf{Y} \rangle = \mathbf{y}]}{\sum_{i=1}^N w_i},$$

where $\Xi = \{\langle \xi_i, w_i \rangle\}_{i=1}^N$.

Algorithm 5: General rejection sampling.

procedure REJECTIONSAMPLING (
$f(\mathbf{x})$ /*a (non-normalized) distribution from which direct sampling is hard*/,
$g(\mathbf{x})$ /*a (non-normalized) distribution from which direct sampling is easy*/,
c /*a constant s.t. for all $\mathbf{x}$, $\frac{f(\mathbf{x})}{c \cdot g(\mathbf{x})} \leq 1$/*)
output a particle generated for $f(\mathbf{x})$.

repeat
  Let $\mathbf{x'} \sim g(\mathbf{x})$
  Let $u \sim \mathcal{U}(0, 1)$
until $\frac{f(\mathbf{x'})}{c \cdot g(\mathbf{x'})} > u$
return ξ

Algorithm 6: LW particle generation (on a posterior distrib.) in a Bayesian network

procedure LW-generate-particle (
$\mathcal{B}$ /*a Bayesian network over $\mathbf{x} = \{X_1, \ldots, X_n\}$ where indices indicate a topological ordering*/,
E /*set of random variables that their values are determined by evidence s.t. the value of $E_i \in \mathbf{E}$ is known to be $\text{VAL}(E_i)$*/)

Let ξ be a particle that should be returned.
Let $w \leftarrow 1$ be the weight of the returning particle.
for $i = 1$ **to** n **do**
  if $X_i \notin \mathbf{E}$ **then**
    $\xi(X_i) \leftarrow$ take a sample from $p(X_i \mid \xi\langle \text{PA}(X_i)\rangle)$
  else
    $\xi(X_i) \leftarrow \text{VAL}(X_i)$
    $w \leftarrow w \cdot p(X_i = x_i \mid \xi\langle \text{PA}(X_i)\rangle)$ /* particle's weight is multiplied by the likelihood of the known value.*/
  end if
end for
return $\langle \xi, w \rangle$

2.3.2.5 Markov Chain Monte Carlo (MCMC)

The sampling methods that are used in practice are often based on the Markov chain Monte Carlo mechanism (MCMC) [Gilks 2005]. In these methods, each sample is obtained via a random walk that starts from the previous sample (and is therefore, conditioned on the previous sample). Since,

$$p(\mathbf{x}^{(t)} \mid \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t-1)}) = p(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)})$$

this mechanism clearly forms a Markov chain:

MCMC algorithms are constructed in such a way that the target distribution $p(\mathbf{x})$ is the (unique) *equilibrium state* (a.k.a *steady state* or *stationary distribution*) of the chain. For this purpose,

1. The designed Markov chain should be *irreducible* (i.e. have a state space with a single *communicating class*). This means that it should be possible to get to any state from any state (via a finite number of transitions).

2. The designed Markov chain should be *reversible* (otherwise stated, it should satisfy *detailed balance*), that is, for any pair of particles (i.e. states) $\mathbf{x}'$ and $\mathbf{x}''$ the following equality should hold:

$$p(\mathbf{x}')T(\mathbf{x}' \to \mathbf{x}'') = p(\mathbf{x}'')T(\mathbf{x}'' \to \mathbf{x}') \tag{2.7}$$

where $T(\mathbf{x} \to \mathbf{y})$ is the probability of transition from $\mathbf{x}$ to $\mathbf{y}$ via an MCMC algo-

rithm. It can be shown that if p satisfies detailed balance then p is a stationary distribution [Wasserman 2013].

An intuition behind the success of a typical MCMC method is as follows: In high-dimensional applications of probabilistic graphical models, the probability mass is often concentrated on some connected regions which proportionally occupy a very small volume of the (random) variable space. If an MCMC sampler is provided with an initial sample (i.e. a point in the space where the distribution is non-zero), by means of random walk, it follows the pattern of the distribution and is not "lost" in the surrounding empty space. The price paid for this advantage property are the following inherent shortcomings:

1. Dealing with particular classes of models, some MCMC methods may never converge to the target distribution. This often happens when the distribution is multi-modal with no random walk paths between non-zero probability states. Luckily, in many models, such pathological examples often do not happen in practice however, the following shortcoming is more serious.

2. Since consecutive samples are dependent, a large number of samples should be taken to guarantee that the convergence to the target distribution takes place with an acceptable approximation error. In other words, the *mixing rate* of the Markov chain can be quite slow. Furthermore, in many models, detecting the sufficient number of samples that are required to converge to the steady state (within a particular approximation bound) may be a hard task, if not impossible.

The dependency between the consecutive samples entails another negative effect as well: the dependence of the Markov chain to its initial state. Nonetheless, this problem is often easy to solve. The reason is that as the number of drawn particles in the chain grows, the dependence of the samples to the initial point reduces till it becomes negligible. Therefore it is sufficient to throw away an initial number of samples (e.g. the first 1000 samples). These discarded first particles are referred to as *burn-in* samples (and the associated time, the *burn-in* period).

2.3.2.6 Metropolis-Hastings sampling (MH)

Metropolis-Hastings algorithm [Hastings 1970] is a popular and easy to implement MCMC sampler. Let the $(t-1)$-th sample drawn from a distribution $p(\mathbf{x})$ via a Metropolis-Hastings Markov chain be $\mathbf{x}^{(t-1)}$. In order to generate $\mathbf{x}^{(t)}$ (i.e the t-th sample), the following two steps are executed:

1. A sample $\mathbf{x}'$ is taken from a *proposal distribution* $q(\mathbf{x}|\,\mathbf{x}^{(t-1)})$ from which samples can be taken efficiently.

2. With probability,

$$a(\mathbf{x}'|\mathbf{x}^{(t-1)}) = \min\left[1, \frac{p(\mathbf{x}') \cdot q(\mathbf{x}^{(t-1)} \mid \mathbf{x}')}{p(\mathbf{x}^{(t-1)}) \cdot q(\mathbf{x}' \mid \mathbf{x}^{(t-1)})}\right] \tag{2.8}$$

$\mathbf{x}'$ is accepted i.e. returned as the next sample,

$$\mathbf{x}^{(t)} \leftarrow \mathbf{x}'$$

otherwise it is rejected and consequently the last taken sample is returned as the next sample,

$$\mathbf{x}^{(t)} \leftarrow \mathbf{x}^{(t-1)}$$

It should be noted that the existence of an *acceptance ratio* in equation 2.8 indicates an important property that neither p nor q need to be normalized.

No restriction is imposed on the choice of the proposal distribution. Nonetheless, if the chosen function is symmetric in the sense that,

$$q(\mathbf{x}|\,\mathbf{x}') = q(\mathbf{x}'|\,\mathbf{x})$$

Equation 2.8 will be simplified as below,

$$a(\mathbf{x}'|\mathbf{x}^{(t-1)}) = \min\left[1, \frac{p(\mathbf{x}')}{p(\mathbf{x}^{(t-1)})}\right] \tag{2.9}$$

In practice the chosen proposal, $q(\mathbf{x}|\,\mathbf{x}^{(t-1)})$, is often an isotropic (multivariate) *Gaussian* centered at $\mathbf{x}^{(t-1)}$.

The detailed mechanism of Metropolis-Hastings method is presented in Algorithm 7.

Convergence to the target distribution . In order to show that Metropolis-Hastings sampling mechanism has the target distribution $p(\mathbf{x})$ as its steady stead, the detailed balance should hold. Here the probability of transition from state $\mathbf{x}^{(t-1)}$ to state $\mathbf{x}'$ (namely, $T\big(\mathbf{x}^{(t-1)} \rightarrow \mathbf{x}'\,\big)$ is the probability that $\mathbf{x}'$ is sampled from the proposition distribution conditioned on the information that the current state is $\mathbf{x}^{(t-1)}$ (i.e. $q(\mathbf{x}' \mid \mathbf{x}^{(t-1)})$) times the probability that the proposed new state is accepted (namely, $a(\mathbf{x}' \mid \mathbf{x}^{(t-1)})$):

$$T\big(\mathbf{x}^{(t-1)} \rightarrow \mathbf{x}'\big) = q\big(\mathbf{x}' \mid \mathbf{x}^{(t-1)}\big) \cdot a\big(\mathbf{x}' \mid \mathbf{x}^{(t-1)}\big)$$

Similarly,

$$T\big(\mathbf{x}' \rightarrow \mathbf{x}^{(t-1)}\big) = q\big(\mathbf{x}^{(t-1)} \mid \mathbf{x}'\big) \cdot a\big(\mathbf{x}^{(t-1)} \mid \mathbf{x}'\big)$$

Therefore, the detailed balance (that is, equation 2.7) will be as below:

$$p(\mathbf{x}') \cdot q\big(\mathbf{x}^{(t-1)} \mid \mathbf{x}'\big) \cdot a\big(\mathbf{x}^{(t-1)} \mid \mathbf{x}'\big) = p(\mathbf{x}^{(t-1)}) \cdot q\big(\mathbf{x}' \mid \mathbf{x}^{(t-1)}\big) \cdot a(\mathbf{x}' \mid \mathbf{x}^{(t-1)})$$

or equivalently,

$$\frac{a(\mathbf{x}' \mid \mathbf{x}^{(t-1)})}{a(\mathbf{x}^{(t-1)} \mid \mathbf{x}')} = \frac{p(\mathbf{x}') \cdot q\big(\mathbf{x}^{(t-1)} \mid \mathbf{x}'\big)}{p(\mathbf{x}^{(t-1)}) \cdot q\big(\mathbf{x}' \mid \mathbf{x}^{(t-1)}\big)}$$

Clearly, this condition is satisfied by equation 2.8 which completes the proof of correct convergence to the target distribution.

Algorithm 7: Metropolis-Hastings algorithm

procedure METROPOLISHASTINGS (
$p(\mathbf{x})$ /* target distribution*/,
$q(\mathbf{x} \mid \mathbf{x}^*)$ /*proposal distribution*/,
$(\mathbf{x}^{(0)})$, /*initial sample*/,
T /*number of samples to be taken*/)
output a sequence of T samples taken from $p(\mathbf{x})$.

for $t = 1$ **to** T **do**
  $\mathbf{x}' \sim q(\mathbf{x} \mid \mathbf{x}^{(t-1)})$
  $u \sim \mathcal{U}(0, 1)$
  if $u \leq \dfrac{p(\mathbf{x}') \cdot q\left(\mathbf{x}^{(t-1)} \mid \mathbf{x}'\right)}{p\left(\mathbf{x}^{(t-1)}\right) \cdot q\left(\mathbf{x}' \mid \mathbf{x}^{(t-1)}\right)}$ **then**
    /*Accept: */
    $\mathbf{x}^{(t)} \leftarrow \mathbf{x}'$
  else
    /*Reject: */
    $\mathbf{x}^{(t)} \leftarrow \mathbf{x}^{(t-1)}$
  end if
end for
return $\langle \mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(T)} \rangle$

Advantages and disadvantages. Metropolis-Hastings is easy to implement. In case it is tuned well, it shows good performance on a large range of target distributions. Moreover, this method does not require any assumption or knowledge about the characteristics of the target distribution. More specifically, Metropolis-Hastings treats the target distribution as a black box that returns the function value associated with proposed sample assignments. This property, arguably, makes Metropolis-Hastings the most popular MCMC sampling method. Nonetheless, this method suffers from the following two weaknesses:

1. The performance of the Metropolis-Hastings method is sensitive to the way the proposal distribution is chosen and tuned. Choosing a good *proposal* is problem-dependent. Intuitively, the best sampling performance occurs when the proposal is as close (e.g. in terms of KL-divergence) to the target distribution as possible. However, the problem is that finding such a distribution from which samples can be taken easily is often not trivial (if even possible). As mentioned, in practice, proposals are often chosen to be Gaussian. However, the variance of the proposal is still a parameter that can significantly affect the performance of the algorithm and requires being tuned. There are algorithms for automated tuning of the aforementioned variance parameter. A famous approach [Roberts et al. 1997] suggests testing several proposal variances and choosing the one that leads to an acceptance rate closest to 0.24. However, finding an optimal param-

eter is still an art that in many applications may require multiple manual trial and error.

2. The performance of Metropolis-Hastings algorithm on models studied in the present work, i.e. piecewise continuous distributions, can in particular be unsatisfactory. The reason is that near the border of partitions, the difference between the smooth proposal distributions and the disconnected target distributions is often significant.

3. Due to significant number of rejected samples and consequently, repeated particles, the correlation between the particles generated by Metropolis-Hastings method is notoriously high. This means that in order to approximate the target distribution, a significant number of samples may be required. For instance, millions of Metropolis-Hastings particles may be needed to produce a same approximation that a few thousand or hundred rejection sampling particles do.

2.3.2.7 Gibbs sampling

Gibbs sampling [Casella and George 1992] is a mechanism by which, the problem of sampling from a multidimensional distribution is reduced to sampling from univariate distributions which is essentially simpler. In this MCMC method, the random variables are sampled in turn. That is, in order to generate a particle from an n-dimensional target distribution, n steps are required. In each step, a random variable is sampled conditioned on the current instantiation of the other variables.

A more formal description is follows: Let the target distribution be $p(X_1, \ldots, X_n)$ and the initial (or current) sample assignment be $(x_1, \ldots, x_n)$. In the i-th step of Gibbs sampling, X_i is sampled from $p(X_i \mid \mathbf{x}_{\setminus i})$ or in the expanded notation,

$$x_i \sim p(X_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}, X_{i+1} = x_{i+1}, \ldots X_n = x_n)$$

Note that since the conditional and joint distributions are proportional, it is sufficient to sample from the target (joint) distribution where all variables except one are instantiated with their current values:

$$x_i \sim p(x_1, x_{i-1}, X_i, x_{i+1}, \ldots x_n)$$

The detailed mechanism is clarified by Algorithm 8.

Clearly, compared to a multivariate distribution, sampling from a univariate distribution is simpler. For the latter task, the inverse-transform sampling method explained in Section 2.3.2.1 and elaborated in Algorithm 2 can be utilized.

Advantages and disadvantages. The main advantages of Gibbs sampling are the following two characteristics:

1. Despite being an MCMC method, (compared to methods such as MH), in practice the correlation of the nearby Gibbs samples are often low.

Algorithm 8: Gibbs sampling algorithm

procedure GIBBSSAMPLER (
$\mathbf{X} = \{X_1, \ldots, X_n\}$, /* set of variables where indices indicate the ordering*/,
$p(\cdot)$, /*An n-dimensional target distribution*/,
$\mathbf{x}^{(0)}$ /*initial state (particle) $\langle x_1^{(0)}, \ldots, x_n^{(0)} \rangle$*/,
T /*number of required samples*/)

for $t = 1$ **to** T **do**
//initializing variables of the new time step due to their previous values:
$\langle x_1^{(t)}, \ldots, x_n^{(t)} \rangle \leftarrow \langle x_1^{(t-1)}, \ldots, x_n^{(t-1)} \rangle$
//taking new samples for variables due to their order:
for $i = 1$ **to** n **do**
$x_i^{(t)} \leftarrow$ sample $X_i^{(t)}$ from $p(X_i \mid x_1^{(t)}, \ldots, x_{i-1}^{(t)}, x_{i+1}^{(t)}, \ldots, x_n^{(t)})$ /*note that despite their indices, $x_{i+1}^{(t)}$ to $x_n^{(t)}$ are not updated from time step $(t-1)$.*/
end for
end for
return $\langle \langle x_1^{(0)}, \ldots, x_n^{(0)} \rangle, \ldots, \langle x_1^{(T)}, \ldots, x_n^{(T)} \rangle \rangle$

2. Gibbs sampler (in its basic version, at least) required no parameter setting or tuning. This leads to robust and fully automated Gibbs samplers.

To justify the first advantageous characteristic, it should be pointed out that Gibbs sampler can be regarded as a special case of Metropolis-Hastings algorithms in which the proposal distribution is a partially instantiated version of the target distribution itself. As a result, the acceptance rate of a Gibbs proposed samples (as in equation 2.8) is always 1.0 and therefore proposed samples are always accepted i.e. the consecutive samples are not repeated.

Nonetheless, Gibbs sampling has disadvantageous properties as well:

1. In the basic or *exact Gibbs* sampling, the univariate sampling that happens in each step relies on inverse-transform sampling which in its turn, requires the computation of a cumulative distribution function. In most cases, the integration, required for the latter task cannot be computed analytically. This shortcoming, severely limits the applications of exact Gibbs sampling.

2. The aforementioned problem can be avoided by approximating the univariate sampling operations via arbitrary univariate sampling methods (such as rejection sampling, Metropolis-Hastings, etc.) Nonetheless, since the inner samplers typically require parameter tuning, such a solution may invalidate the desired characteristic of exact Gibbs sampling that does not involve any tuning.

3. Since the number of steps in Gibbs sampling is equal to the dimensionality of the variable space, its complexity grows linearly in the number of random variables. Keeping in mind that (either exact or approximate) computation of CDFs

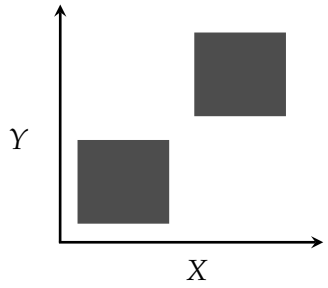


Figure 2.4: Joint distribution of X and Y. The distribution is only positive in dark regions. This is a pathological example where Gibbs sampling cannot move between modes.

as well as their inverse functions (which is also required for inverse transform sampling) per sample per random variable is computationally costly, Gibbs samplers are typically slow and hard to implement.

4. The transitions that Gibbs sampling MCMC mechanism relies upon are random walks that are in parallel to the basis vectors of the variable space. Such movements do not necessarily end in an irreducible Markov chain. A simple pathological example is depicted in Figure 2.4 where the target distribution forms detached islands of positive probability that are not accessible by the aforementioned random walk pattern. Clearly, in such scenarios Gibbs is trapped in one region and never converges to the target. This problem may be avoided by tricks such as rotation of basis vectors.

We seek to address the first shortcoming in Chapter 5 for a large family of models. As a result of this contribution, the second shortcoming is avoided and the gravity of the third shortcoming is lessened.

2.3.2.8 Hamiltonian Monte Carlo (HMC)

In Section 2.3.2.6 is was mentioned that a major shortcoming of Metropolis-Hastings MCMC algorithm is the strong correlation of the nearby samples. This problem is a direct consequence of random walk behavior. More specifically, the MH proposed transitions in the Markov chain are often close to the current state since far apart proposed transitions are often rejected. This is due to the fact that the proposal density required by Metropolis-Hastings method often does not follow the changes in the curvature of the target distribution.

Hamiltonian Monte Carlo (HMC) (which was originally called *hybrid Monte Carlo* [Duane et al. 1987]) addresses this issue by introducing a *gradient-aware* mechanism for proposing transitions. The gradient of the target distribution indicate its curvature and the directions that lead to higher probability areas. As a result, the proposed HMC transitions often lie quite distant in the variable space while in contrast to Metropolis-

Hastings method, a reasonably high acceptance ratio is often maintained for these transitions.

Following the gradient should be such that the detailed balance condition (which is required for correct sampling) holds [Neal 2011]. To satisfy this requirement, the *Hamiltonian dynamics* is utilized.

Hamiltonian dynamics. Hamiltonian is a concept originated from the field of mechanics. Hamiltonian mechanics is an alternative reformulation of the classic mechanics in which the state of a system in an n-dimensional space is determined by a *position vector* (a.k.a. *canonical coordinates*)

$$\mathbf{q} = (q_1 \ldots, q_n)$$

and a *momentum vector* (a.k.a *conjugate momenta*)

$$\mathbf{p} = (p_1 \ldots, p_n)$$

Although we are interested in non-physical MCMC applications of Hamiltonian dynamics, to motivate the discussion, consider the dynamics of a physical system that consists of a mass particle sliding over a frictionless manifold (e.g. a surface in a 3D space). The potential energy of the particle, $U(\cdot)$, is a function of the position vector (e.g. in a 2D space, it is often defined proportional to the height of the surface). The *kinetic energy*, $K(\cdot)$, is often defined as,

$$\|\mathbf{p}\|^2/2m$$

where m is the mass of the particle. For further simplicity, it is often assumed that the particle is a unit mass, leading to,

$$K(\mathbf{p}) = \frac{\|\mathbf{p}\|^2}{2} \tag{2.10}$$

Clearly, in this case, the momentum and velocity are the same.

The Hamiltonian $H(\cdot)$ of the system, is the summation of the potential and kinetic energies (i.e. system's total energy),

$$H(\mathbf{q}, \mathbf{p}) = U(\mathbf{q}) + K(\mathbf{p})$$

The evolution of the system over time, t, is determined by the following partial derivatives of the Hamiltonian which are known as *Hamiltonian equations of motion* (or simply, *Hamiltonian equations*):

$$\frac{\mathrm{d}q_i}{\mathrm{d}t} = \frac{\partial H}{\partial p_i} \tag{2.11}$$

$$\frac{\mathrm{d}p_i}{\mathrm{d}t} = -\frac{\partial H}{\partial q_i} \tag{2.12}$$

where i is in $1, \ldots, n$.

Hamiltonian conservation. According to the above Hamiltonian equations, the Hamiltonian of a system remains invariant over time. This can easily be verified as follows:

$$\begin{aligned}\frac{\mathrm{d}H}{\mathrm{d}t} = \sum_{i=1}^{n}\left[\frac{\mathrm{d}q_i}{\mathrm{d}t}\frac{\partial H}{\partial q_i} + \frac{\mathrm{d}p_i}{\mathrm{d}t}\frac{\partial H}{\partial p_i}\right] &= \sum_{i=1}^{n}\left[\frac{\partial H}{\partial p_i}\frac{\partial H}{\partial q_i} + \frac{\mathrm{d}p_i}{\mathrm{d}t}\frac{\partial H}{\partial p_i}\right] && \text{, by (2.11)} \\ &= \sum_{i=1}^{n}\left[\frac{\partial H}{\partial p_i}\frac{\partial H}{\partial q_i} - \frac{\partial H}{\partial q_i}\frac{\partial H}{\partial p_i}\right] && \text{, by (2.12)} \\ &= 0.\end{aligned}$$

Volume preservation (volume conservation). Another important property of Hamiltonian dynamics is that it conserves the volume V of any region R in the space $(\mathbf{q}, \mathbf{p})$ (namely, *phase space*) in time. That is, according to Hamiltonian dynamics, a region may move and change its shape but its total volume remains constant.

In the context of Hamiltonian Monte Carlo (that will be introduced shortly), this property means that we do not need to take into account any change of volume in the acceptance probability of Metropolis-Hastings updates [Neal 2011].

Let us take a closer look at the volume preservation. Suppose that time evolution of a system is determined by the Hamiltonian equations of motion. According to our notation, if at time $t = 0$ a phase space point is at $(\mathbf{q}(0), \mathbf{p}(0))$, then at time $t = \tau$, it is evolved to $(\mathbf{q}(\tau), \mathbf{p}(\tau))$. Time evolution is equivalent to the following coordinate transformation T,

$$\big(\mathbf{q}(0), \mathbf{p}(0)\big) \xrightarrow{\mathrm{T}} \big(\mathbf{q}(\tau), \mathbf{p}(\tau)\big)$$

If under T, a region volume V is mapped to V', then according to the Hamiltonian volume preservation property,

$$\int_{V'} \mathrm{d}\mathbf{q}(\tau)\ \mathrm{d}\mathbf{p}(\tau) = \int_{V} \mathrm{d}\mathbf{q}(0)\ \mathrm{d}\mathbf{p}(0) \tag{2.13}$$

On the other hand, by the definition of Jacobians,

$$\int_{V'} \mathrm{d}\mathbf{q}(\tau)\ \mathrm{d}\mathbf{p}(\tau) = \int_{V} |J|\ \mathrm{d}\mathbf{q}(0)\ \mathrm{d}\mathbf{p}(0) \tag{2.14}$$

where,

$$|J| := \left|\det \frac{\partial \mathbf{q}(\tau)\partial \mathbf{p}(\tau)}{\partial \mathbf{q}(0)\partial \mathbf{p}(0)}\right|$$

By substituting (2.14) in (2.14) we get $|J| = 1$. Therefore, volume preservation holds if and only if the absolute value of the determinant of the Jacobian matrix of transformation T is equal to 1. In chapter 6, we will leverage this property to prove that for our proposed generalisation of the Hamiltonian Monte Carlo algorithm volume preservation still holds.

Hamiltonian Monte Carlo algorithm. Provided with sufficient background on the properties of Hamiltonian equations of motion, we are ready to present the sampling

Algorithm 9: Hamiltonian Monte Carlo algorithm

procedure HAMILTONIANMONTECARLO (
$\mathbf{q}_0$ /*current sample*/,
U /*potential function*/,
L /*number of leapfrog steps*/,
ϵ /*leapfrog step size*/)
output next sample

$\mathbf{q} \leftarrow \mathbf{q}_0$
$\mathbf{p} \sim \mathcal{N}(0, 1)$
$H_0 \leftarrow \|\mathbf{p}\|^2/2 + U(\mathbf{q})$
for $l = 1$ **to** L **do**
  /* *Half-step evolution of momentum*: */
  $\mathbf{p} \leftarrow \mathbf{p} - \epsilon \nabla U(\mathbf{q})/2$
  /* *Full-step evolution of position*: */
  $\mathbf{q} \leftarrow \mathbf{q} + \epsilon \mathbf{p}$
  /* *Half-step evolution of momentum*: */
  $\mathbf{p} \leftarrow \mathbf{p} - \epsilon \nabla U(\mathbf{q})/2$
end for
$H \leftarrow \|\mathbf{p}\|^2/2 + U(\mathbf{q})$
$\Delta H \leftarrow H - H_0$
$u \sim \mathcal{U}(0, 1)$
if $u < e^{-\Delta H}$ **then**
  return q
else
  return $\mathbf{q}_0$
end if

mechanism. Suppose sampling from an n-dimensional continuous probability distribution $\Pr(\mathbf{q})$ is intended.

To start with, the variable space is augmented by an n-dimensional *momentum variable* vector $\mathbf{p}$.

A simple (but not unique) way to define a Hamiltonian is to choose the kinetic energy according to 2.10 and let the potential energy be the negation of the logarithm of the target distribution:

$$U(\mathbf{q}) = -\log\big(\Pr(\mathbf{q})\big) \tag{2.15}$$

where the probability measure is denoted by Pr rather than p to avoid conflation and ambiguity.

In order to generate a sample using HMC, firstly an assignment is proposed for the momentum variables using a newly introduced momentum distribution $\Pr(\mathbf{p})$. This distribution is chosen in such a way that it can be sampled easily. In practice, a (multivariate) Gaussian is often the choice.

Subsequently, using the Hamiltonian equations of motion, for both $\mathbf{p}$ and $\mathbf{q}$ new proposals are generated. In order to do so, we should let the system develop for a fixed time interval I. Starting from the current states of $\mathbf{p}$ and $\mathbf{q}$, by definitions 2.11 and 2.12 and the way the kinetic and potential energies are chosen (equations 2.10 and 2.15), the trajectory of movement is governed by the following differential equations,

$$\frac{\mathrm{d}q_i}{\mathrm{d}t} = p_i \tag{2.16}$$

$$\frac{\mathrm{d}p_i}{\mathrm{d}t} = -\frac{\partial U}{\partial q_i} = \frac{\partial \log(\Pr(q_i))}{\partial q_i} \tag{2.17}$$

The above constraints are often not solvable in closed form. However, they can be effectively approximated by means of discretization methods. One of the most famous ways to approximate differential equations is *Euler's method* [Butcher 2000]. However, it is shown that *leapfrog method* leads to much better results [Neal 2011]. In the latter method, the total transition time I is divided into L steps each of which takes an step-size ϵ. That is,

$$I = L \times \epsilon$$

where L and ϵ are fixed parameters. In other words, the system trajectory is approximated with L pieces and in each piece, the location and momentum vectors are interchangeably evolved for a time period ϵ. The leapfrog mechanism in each time period (step-size) ϵ is as follows:

1. *Half-step evolution of momentum.* The value of $\mathbf{q}$ is kept fixed and the (proposed) value of $\mathbf{p}$ is modified as follows ($\mathbf{p}$ is evolved for a period of $\epsilon/2$):

$$\mathbf{p} \leftarrow \mathbf{p} - \frac{\epsilon}{2} \cdot \nabla U(\mathbf{q})$$

2. *Full-step evolution of position.* $\mathbf{p}$ is left unchanged and $\mathbf{q}$ is evolved for a time period of ϵ.

$$\mathbf{q} \leftarrow \mathbf{q} + \epsilon \cdot \mathbf{p}$$

3. *Half-step evolution of momentum.* Step 1 is repeated.

$$\mathbf{p} \leftarrow \mathbf{p} - \epsilon \cdot \nabla U(\mathbf{q})/2$$

The generated proposal is fed to a Metropolis-Hastings algorithm and will be the next sample in case accepted. The formal HMC mechanism is illustrated by Algorithm 9.

Advantages and disadvantages. The main advantage of Hamiltonian Monte Carlo (HMC) is that (in case it is tuned well) the correlation between the nearby samples is small, its corresponding Markov chain mixing rate is high and consequently, compared to Metropolis-Hastings algorithm, convergence to the target distribution takes place with a smaller number of sampled particles.

Nonetheless, this method suffers from the following shortcomings:

1. Unlike other sampling mechanisms that have versions for continuous and discrete state space, the Hamiltonian Monte Carlo can exclusively be applied to continuous distributions. As a potential solution to this restriction, it is suggested to approximate discrete distributions with continuous state piecewise distributions [Pakman and Paninski 2013].

2. Unlike Metropolis-Hastings, HMC requires to be provided with the gradient of the target distribution. In many applications, this extra information may not be available.

3. The performance of HMC algorithm depends on the choice of L and ϵ. In practice, these parameters are often not easy to adjust. As a matter of fact, compared to Metropolis-Hastings, HMC is often much more sensitive to parameter tuning [Homan and Gelman 2014].

Roughly speaking, parameter ϵ controls the simulation accuracy of the Hamiltonian equations of motion while the product of L and ϵ affects the distance between the current and the proposed state.

If L and ϵ are both chosen too small, then the random walk behavior is not resolved and the decorrelation of the neighboring particles that is the raison d'être of this method does not taken place.

If either parameter (and in particular, ϵ) is chosen too large then the simulation of Hamiltonian dynamics will not be sufficiently accurate which leads to many rejection samples that in its turn negatively affects the mixing rate due to the abundance of consecutive repeated samples.

If L is very large but ϵ is sufficiently small, then the simulation accuracy and the decorrelation of samples will be both satisfactory; nonetheless, the computation costs will be considerable and the time required to generate a single sample can be long. As a result, despite low correlation between nearby samples, in this case the convergence to the target distribution can be slow.

The optimal choice of parameters depends on the characteristics of the target distribution. If the rate of changes in the target distribution (and consequently, the variations in its corresponding potential energy function) is too high (e.g. the function contains several anomalies rather than following a smooth trend), the leapfrog mechanism should be fine-tuned (i.e. requires a smaller step-size, ϵ). In the limit, if the function is not smooth, regardless of the choice of ϵ, the simulation of the Hamiltonian dynamics would be inaccurate and can lead to significant a number of rejections. The piecewise distributions that are in the focus of this thesis lie in the latter group.

2.3.3 Other approximate inference methods

Besides sampling methods, there are other categories of approximate inference that (since we do not require them in the next chapters) are not covered in detail. Nonetheless, for the sake of completeness we briefly mention the most important ones:

2.3.3.1 Direct approximation of query

In these methods the query i.e. the posterior distribution $p(Z|E)$ of the latent variable Z is directly approximated in a closed form.

- **Variational Bayesian methods.** Variational Bayes approximates the posterior $p(Z \mid E)$ by a simpler distribution $Q(Z)$ that belong to a parametric class of functions with an exact analytical solution [Jordan et al. 1999], [Wainwright and Jordan 2008].

  The main task is to find a *variational distribution* $Q(Z)$ that is as similar as possible to the target posterior therefore, the dissimilarity between Q and P (say, $D(Q \,||\, P)$) has to be minimized. The most commonly used dissimilarity (or distance) function is the *cross-entropy* (Kullback-Leibler (KL) divergence):

$$D_{\mathrm{KL}}(Q \,||\, P) = \sum_{Z} Q(Z) \log \frac{Q(Z)}{p(Z|E)}$$

- **Expectation propagation** (EP), is an iterative method for proving approximations to probability distributions [Minka 2001]. Similar to variational methods, EP techniques are based on minimizing the KL-divergence. However in the latter methods, the divergence is calculated in a reverse form. That is, while in variational methods $D_{\mathrm{KL}}(Q \,||\, P)$ is calculated and minimized, EP methods are based on $D_{\mathrm{KL}}(P \,||\, Q)$ where P is the target distribution and Q is its desired approximation.

2.3.3.2 Message passing based approximate inference

In Section 2.3.1, an exact inference technique based on message passing on polytrees was introduced. The following approximate message passing approaches are based on this technique.

- **Propagation-based approximation.** In these techniques the messages are generated in an exact manner and approximation takes place in their propagation.
  - **Loopy belief propagation.** In this approach, the assumption that the underlying graph is a polytree is removed. Therefore, although messages are exact, they are propagated through graphs containing loops (undirected cycles) in an iterative manner, resulting in *loopy belief propagation*. The algorithm is motivated by the success of turbo codes [McEliece et al. 1998] which is shown to be expressible by this algorithm [Murphy 2001].
  - **Generalized loopy belief propagation (GBP).** Based on Kikuchi algorithm [Kikuchi 1951] and via joining a few variables, the generalized loopy BP methods try to reduce the number of graph loops and generate a graph that is more similar to a polytree. The complexity of GBP algorithms is adjustable and they can be significantly more accurate than loopy BP [Yedidia et al. 2000].

- **Propagation with approximate messages.** In the second approach, the messages themselves and/or inner structure of subnetworks within clusters are simplified to a more manageable form. In Bayesian inference, two commonly used approaches are *variational message passing* (VMP) and *moment matching*. Both approaches factorize a message and use KL-divergence to find the best approximation.

2.4 Summary

This chapter covered basic definitions, notation and background material concerning probabilistic inference that is required to present our contributions for inference on piecewise graphical models. It dealt with the representation of probabilistic graphical models as well as a spectrum of exact and approximate inference techniques that can be used on such models. In particular, MCMC based inference techniques where studied in more detail. The reason is that the inference methods that will be proposed by this thesis are based on Gibbs sampling (see Chapters 4 and 5) and Hamiltonian Monte Carlo (see Chapter 6).

Nonetheless, before making our contributions, we still require to formally define piecewise functions and discuss their characteristics and corresponding operations. We should also introduce the data structures that can be utilized to represent piecewise functions as well as the algorithms that can be used to simplify such data structures. These remaining background materials are covered by the next chapter.

Chapter 3

Piecewise Data Structures and Symbolic Operations

The previous chapter provided sufficient background material for probabilistic inference in graphical models. In this chapter we offer background material for the symbolic presentation, manipulation and simplification of piecewise continuous functions. The formal definition of piecewise functions is presented in Section 3.1. Section 3.2 is concerned with symbolic operations where their output or at least one of their operands is a piecewise expression. Theses operations will be frequently used in the next chapters where our actual contributions regarding the probabilistic inference in piecewise graphical models are presented. Section 3.3 introduces the data structures that are used (or may be used) to model piecewise function. Finally, in Section 3.4 we briefly introduce the existing tools for manipulation of symbolic formulas, as well as checking the satisfiability of first order logic predicates. These symbolic tools can in particular be used to simplify piecewise data structures by means of eliminating unsatisfiable case-statements.

3.1 Piecewise functions

A function is piecewise if it is in the following general form,

$$f = \sum_{i=1}^{m} \mathbb{I}[\phi_i(\mathbf{x})] \cdot f_i(\mathbf{x}) \tag{3.1}$$

where $\mathbb{I}[\cdot]$ denotes the *indicator function*[1] and the constraints ϕ_1 to ϕ_m are assumed to be pairwise exclusive (w.r.t. all variable valuations), that is,

$$\big(\phi_i(\mathbf{x}) \wedge \phi_j(\mathbf{x})\big) \equiv \bot \qquad \forall i \neq j, \forall \mathbf{x}$$

Since the space of variables, $\mathbf{x}$, is partitioned by the constraints, each ϕ_i is alternatively referred to as a *partition* (also known as *region*). In case the constraints are affine functions, a partition may also be called a *polytope*.

[1] That is, if a boolean expression ϕ is true then $\mathbb{I}[\phi]$ returns 1 otherwise 0.

Clearly, if for a variable assignment $\mathbf{x}$, a constraint $\phi_i(\mathbf{x})$ is satisfied (where $i \in \{1, \ldots, m\}$) then the function f returns the value of its i-th sub-function,

$$\phi_i(\mathbf{x}) \equiv \top \iff f(\mathbf{x}) = f_i(\mathbf{x})$$

Alternatively, equation 3.1 can be stated in the following expanded form:

$$f = \begin{cases} f_1(\mathbf{x}) & \text{if } \phi_1(\mathbf{x}) \\ \vdots & \vdots \\ f_m(\mathbf{x}) & \text{if } \phi_m(\mathbf{x}) \end{cases} \tag{3.2}$$

In the above relations, boolean expressions ϕ_1 to ϕ_m are called *constraints* (or *sub-domains*) and f_1 to f_m are referred to as *sub-functions*.

For instance in Figure 1.4 (Chapter 1) the (non-normalized) posterior is the following 4-piece function:

$$\begin{cases} 0.12 \cdot \mathcal{N}_{x,y}(\mathbf{0}, \mathbf{\Sigma}) & \text{if } (y > x), (y < 0) \\ 0.28 \cdot \mathcal{N}_{x,y}(\mathbf{0}, \mathbf{\Sigma}) & \text{if } (y > x), (y \geq 0) \\ 0.18 \cdot \mathcal{N}_{x,y}(\mathbf{0}, \mathbf{\Sigma}) & \text{if } (y \leq x), (y < 0) \\ 0.42 \cdot \mathcal{N}_{x,y}(\mathbf{0}, \mathbf{\Sigma}) & \text{if } (y \leq x), (y \geq 0) \end{cases} \tag{3.3}$$

where $\mathcal{N}_{x,y}(\mathbf{0}, \mathbf{\Sigma})$ is the bivariate normal distribution of x and y centered at $\mathbf{0} = [0, 0]^T$ and with covariance matrix $\mathbf{\Sigma} = 10\mathbf{I}_2$.

The constraints ϕ_1 to ϕ_m are not necessarily exhaustive collectively. In case they are, that is, if for any variable valuation $\mathbf{x}$,

$$\bigvee_{i=1}^{m} \phi_i(\mathbf{x}) \equiv \top$$

then f is a total function otherwise it is partial. Throughout, by default we assume that the piecewise functions are total functions except when otherwise stated.

By the i-th *case statement* of a piecewise function in the form 3.1 we refer to $\mathbb{I}[\phi_i] \cdot f_i$. A piecewise function with m case-statements is called an *m-piece function*.

In general the variable space over which piecewise functions are defined can contain continuous, discrete and boolean variables. However, since the focus of this thesis is on piecewise continuous probability functions, unless otherwise stated, we assume that the aforementioned variable space exclusively contains real variables. Generalization of most discussions and presented algorithms to the discrete/continuous hybrid case is straightforward.

3.1.1 $\mathfrak{C}$-piecewise $\mathfrak{D}$ function

Here we present a more restricted but important class of piecewise functions. Let $\mathfrak{C}$ and $\mathfrak{D}$ be two families of functions. We say that a function f is a *$\mathfrak{C}$-piecewise $\mathfrak{D}$ function*

(e.g. linear-piecewise polynomial, etc.) if the sub-functions (f_1 to f_m) are members of the family $\mathfrak{D}$ and each constraint ϕ_i is a boolean expression in the form of an inequality or a conjunction of inequalities,

$$\phi_i = (\varphi_{i,1} \lessgtr 0) \wedge (\varphi_{i,2} \lessgtr 0) \wedge \cdots$$

where $\lessgtr$ stands for $>$, $<$, $\leq$ or $\geq$ and each $\phi_{i,j}$ is an expression of type $\mathfrak{C}$. For instance the function provided in the previous example (equation 3.3) is *linear-piecewise exponential* (or more specifically stated, *linear-piecewise Gaussian*).

The predicates $(\phi_{i,j} \lessgtr 0)$ are called *atomic constraints* (a.k.a. *atomic inequalities*). It should be pointed out that throughout we use piecewise structure to represent piecewise continuous probability density functions. The value of such functions on the border of partitions does not matter (since we assume that the total measure on the partitioning borders is zero). As a result, we often do not distinguish between strict and non-strict inequalities.

3.2 Piecewise Calculus

Following [Sanner et al. 2011; Sanner and Abbasnejad 2012], in this subsection some useful piecewise operations are presented and discussed. These operations are of particular importance in the context of probabilistic inference over piecewise continuous models.

3.2.1 Piecewise/non-piecewise operations

Let f be a piecewise function defined by equation 3.1 (or 3.2). and c be a non-piecewise function[2]. As the following formulas illustrate, the elementary operations that involve c and f are piecewise with the same partitions that are associated with the piecewise operand f.

$$c \cdot f = \begin{cases} c \cdot f_1 & \text{if } \phi_1 \\ \vdots & \vdots \\ c \cdot f_m & \text{if } \phi_m \end{cases} \qquad c + f = \begin{cases} c + f_1 & \text{if } \phi_1 \\ \vdots & \vdots \\ c + f_m & \text{if } \phi_m \end{cases}$$

3.2.2 Piecewise/piecewise elementary operations

Throughout this thesis, whenever instead of the elementary operations $+$, $-$, $*$ and $/$, "o" symbols $\oplus$, $\ominus$, $\otimes$ and $\oslash$ are used respectively, that is to emphasize that both sides of the binary operation are piecewise functions. Application of an elementary binary operation over an N-piece and an M-piece operands results in an $N \times M$-piece

[2]Technically speaking, any function is a piecewise function however, when a function is associated with a single partition (i.e. its only sub-domain is its domain), we consider it a *non-piecewise* function. Therefore, what we refer to as *non-piecewise functions* are in fact *1-piece functions*.

structure. This is illustrated in equations 3.4 to 3.7 were elementary operations over a couple of 2-piece functions lead to 4-piece outputs.

$$\begin{cases} f_1 & \text{if } \phi_1 \\ f_2 & \text{if } \phi_2 \end{cases} \oplus \begin{cases} g_1 & \text{if } \psi_1 \\ g_2 & \text{if } \psi_2 \end{cases} = \begin{cases} f_1 + g_1 & \text{if } \phi_1 \wedge \psi_1 \\ f_1 + g_2 & \text{if } \phi_1 \wedge \psi_2 \\ f_2 + g_1 & \text{if } \phi_2 \wedge \psi_1 \\ f_2 + g_2 & \text{if } \phi_2 \wedge \psi_2 \end{cases} \tag{3.4}$$

$$\begin{cases} f_1 & \text{if } \phi_1 \\ f_2 & \text{if } \phi_2 \end{cases} \ominus \begin{cases} g_1 & \text{if } \psi_1 \\ g_2 & \text{if } \psi_2 \end{cases} = \begin{cases} f_1 - g_1 & \text{if } \phi_1 \wedge \psi_1 \\ f_1 - g_2 & \text{if } \phi_1 \wedge \psi_2 \\ f_2 - g_1 & \text{if } \phi_2 \wedge \psi_1 \\ f_2 - g_2 & \text{if } \phi_2 \wedge \psi_2 \end{cases} \tag{3.5}$$

$$\begin{cases} f_1 & \text{if } \phi_1 \\ f_2 & \text{if } \phi_2 \end{cases} \otimes \begin{cases} g_1 & \text{if } \psi_1 \\ g_2 & \text{if } \psi_2 \end{cases} = \begin{cases} f_1 \times g_1 & \text{if } \phi_1 \wedge \psi_1 \\ f_1 \times g_2 & \text{if } \phi_1 \wedge \psi_2 \\ f_2 \times g_1 & \text{if } \phi_2 \wedge \psi_1 \\ f_2 \times g_2 & \text{if } \phi_2 \wedge \psi_2 \end{cases} \tag{3.6}$$

$$\begin{cases} f_1 & \text{if } \phi_1 \\ f_2 & \text{if } \phi_2 \end{cases} \oslash \begin{cases} g_1 & \text{if } \psi_1 \\ g_2 & \text{if } \psi_2 \end{cases} = \begin{cases} f_1/g_1 & \text{if } \phi_1 \wedge \psi_1 \\ f_1/g_2 & \text{if } \phi_1 \wedge \psi_2 \\ f_2/g_1 & \text{if } \phi_2 \wedge \psi_1 \\ f_2/g_2 & \text{if } \phi_2 \wedge \psi_2 \end{cases} \tag{3.7}$$

Indeed (the constraints of) many produced case-statements may be unsatisfiable (corresponding to empty partitions) and therefore discarded. As a result, $N \times M$ is only an upper bound on the number of output pieces.

From relations 3.4 to 3.7 it can also be noticed that a family of $\mathfrak{C}$-piecewise $\mathfrak{D}$ functions is closed under an elementary operation if and only if the family of functions $\mathfrak{D}$ is closed under that operation (no limitation is imposed on $\mathfrak{C}$).

3.2.3 Maximum and Minimum

As it is evident from formula 3.8, maximum (or minimum) of n non-piecewise expressions is a piecewise expression with (at most) n cases-statements each of which is associated with a constraint that is the combination of (at most) $n - 1$ atomic inequal-

ities.

$$\max(h_1, \ldots, h_n) = \begin{cases} h_1 & \text{if } (h_1 > h_2) \wedge (h_1 > h_3) \wedge \ldots \wedge (h_1 > h_n) \\ \vdots & \\ h_n & \text{if } (h_n > h_1) \wedge (h_n > h_2) \wedge \ldots \wedge (h_n > h_{n-1}) \end{cases} \tag{3.8}$$

These operations can be executed over piecewise functions as well. Formula 3.9 illustrates that maximum (or minimum) of a pair of N and M-piece functions can contain (up to) $2M \cdot N$ case-statements.

$$\max\left(\begin{cases} f_1 & \text{if } \phi_1 \\ f_2 & \text{if } \phi_2 \end{cases}, \begin{cases} g_1 & \text{if } \psi_1 \\ g_2 & \text{if } \psi_2 \end{cases} \right) = \begin{cases} f_1 & \text{if } \phi_1 \wedge \psi_1 \wedge (f_1 > g_1) \\ g_1 & \text{if } \phi_1 \wedge \psi_1 \wedge (f_1 \leq g_1) \\ f_1 & \text{if } \phi_1 \wedge \psi_2 \wedge (f_1 > g_2) \\ g_2 & \text{if } \phi_1 \wedge \psi_2 \wedge (f_1 \leq g_2) \\ f_2 & \text{if } \phi_2 \wedge \psi_1 \wedge (f_2 > g_1) \\ g_1 & \text{if } \phi_2 \wedge \psi_1 \wedge (f_2 \leq g_1) \\ f_2 & \text{if } \phi_2 \wedge \psi_2 \wedge (f_2 > g_2) \\ g_2 & \text{if } \phi_2 \wedge \psi_2 \wedge (f_2 \leq g_2) \end{cases} \tag{3.9}$$

Formula 3.9 also shows that a family of $\mathfrak{C}$-piecewise $\mathfrak{D}$ functions is closed under min/max operation if $\mathfrak{D} \subset \mathfrak{C}$ (hence $(f_i \lesseqgtr g_i) \in \mathfrak{C}$). For instance, linear-piecewise polynomials are not closed under min and max operations but linear-piecewise linear functions and polynomial-piecewise polynomials are closed.

3.2.4 Comparisons

Comparison based operations are closely related to min/max operations. For instance, formula 3.10 represents the indicator of *greater* ($>$) comparison where the operands are 2-piece functions.

$$\mathbb{I}\left[\begin{cases} f_1 & \text{if } \phi_1 \\ f_2 & \text{if } \phi_2 \end{cases} > \begin{cases} g_1 & \text{if } \psi_1 \\ g_2 & \text{if } \psi_2 \end{cases} \right] = \begin{cases} 1 & \text{if } \phi_1 \wedge \psi_1 \wedge (f_1 > g_1) \\ 0 & \text{if } \phi_1 \wedge \psi_1 \wedge (f_1 \leq g_1) \\ 1 & \text{if } \phi_1 \wedge \psi_2 \wedge (f_1 > g_2) \\ 0 & \text{if } \phi_1 \wedge \psi_2 \wedge (f_1 \leq g_2) \\ 1 & \text{if } \phi_2 \wedge \psi_1 \wedge (f_2 > g_1) \\ 0 & \text{if } \phi_2 \wedge \psi_1 \wedge (f_2 \leq g_1) \\ 1 & \text{if } \phi_2 \wedge \psi_2 \wedge (f_2 > g_2) \\ 0 & \text{if } \phi_2 \wedge \psi_2 \wedge (f_2 \leq g_2) \end{cases} \tag{3.10}$$

3.2.5 Substitution

Substituting a variable x of a non-piecewise (potentially multivariate) function h by an m-piece function f,

$$f = \begin{cases} f_1 & \text{if } \phi_1 \\ \vdots & \vdots \\ f_m & \text{if } \phi_m \end{cases},$$

leads to a piecewise function with (at most) m pieces.

$$h|_{x \leftarrow f} = \begin{cases} h|_{x \leftarrow f_1} & \text{if } \phi_1 \\ \vdots & \vdots \\ h|_{x \leftarrow f_m} & \text{if } \phi_m \end{cases} \tag{3.11}$$

Similarly, substituting a variable in an n-piecewise function g by an m-piece function f leads to a piecewise function with (at most) $m \times n$ pieces since with respect to 3.11, each of the n partition of g can be divided to m pieces.

3.2.6 Integration

We only consider the definite integration of linear-piecewise functions over interval $(-\infty, +\infty)$ which is required for the marginalization of random variables while generalization to other cases is straight-forward. Built on [Sanner and Abbasnejad 2012], in order to integrate a linear-piecewise function, it is sufficient to sum up the integrals of its case statements,

$$\int_{-\infty}^{\infty} \sum_{i=1}^{m} \mathbb{I}[\phi_i] \cdot f_i \, \mathrm{d}x = \sum_{i=1}^{m} \int_{-\infty}^{\infty} \mathbb{I}[\phi_i] \cdot f_i \, \mathrm{d}x$$

Therefore, we only focus on the integration of a single case-statement,

$$\int_{-\infty}^{\infty} \mathbb{I}[\phi_i] \cdot f_i \, \mathrm{d}x$$

Without loss of generality, let ϕ_i be a conjunction of simpler constraints:

$$\phi_i := \psi_i \wedge \psi_i'$$

where ψ_i does not involve variable x and ψ_i' is the conjunction of atomic constraints,

$$\psi_i' := (x > L_{i,1}) \wedge \ldots \wedge (x > L_{i,n_i}) \wedge (x < U_{i,1}) \wedge \ldots \wedge (x < U_{i,m_i})$$

which are either in the form $x > L_{i,j}$ (or $x \geq L_{i,j}$) or in the form $x < U_{i,j}$ (or $x \leq U_{i,j}$) where $L_{i,j}$ and $U_{i,j}$ are expression that do not involve x. We do not differentiate between strict and non-strict inequalities since they do not affect the result of integra-

tion.[3]

$$\int_{-\infty}^{\infty} \mathbb{I}[\phi_i] \cdot f_i \, \mathrm{d}x = \int_{-\infty}^{\infty} \mathbb{I}\left[\bigwedge_{j=1}^{n_i} (x > L_{i,j}) \bigwedge_{k=1}^{m_i} (x < U_{i,k}) \bigwedge \psi_i\right] \cdot f_i \, \mathrm{d}x$$

Since ψ_i is not a function of x, it can be pushed outside the integral. On the other hand the lower and upper integration bounds are determined by the maximum of $L_{i,j}$ and minimum of $U_{i,j}$ respectively,

$$LB = \max(L_{i,1}, \ldots, L_{i,n_i}) \qquad UB = \min(U_{i,1}, \ldots, U_{i,m_i})$$

However, since the lower and upper bounds are symbolic, the final result should be multiplied by $\mathbb{I}[LB \leq UB]$ to guarantee that the former is not higher than the latter.

$$\begin{aligned}\int_{-\infty}^{\infty} \mathbb{I}[\phi_i] \cdot f_i \, \mathrm{d}x \;&=\; \mathbb{I}[\psi_i] \cdot \int \mathbb{I}\left[\bigwedge_{j=1}^{n_i} (x > L_{i,j}) \bigwedge_{k=1}^{m_i} (x < U_{i,k})\right] \cdot f_i \, \mathrm{d}x \\ &= \mathbb{I}[\psi_i] \cdot \int_{\max(L_{i,1},\ldots,L_{i,n_i})}^{\min(U_{i,1},\ldots,U_{i,m_i})} f_i \, \mathrm{d}x \otimes \mathbb{I}[LB \leq UB]\end{aligned}$$

Note that in the above formula, the latter multiplication is represented by $\otimes$ to indicate that its both sides are piecewise functions. Therefore, the final result is as follows:

$$\begin{aligned}\int_{-\infty}^{\infty} \mathbb{I}[\phi_i] \cdot f_i \, \mathrm{d}x \;=\; \mathbb{I}[\psi_i] \cdot \Bigg(&\int f_i \, \mathrm{d}x \Big|_{\min(U_{i,1},\ldots,U_{i,m_i})} \ominus \int f_i \, \mathrm{d}x \Big|_{\max(L_{i,1},\ldots,L_{i,n_i})} \Bigg) \\ &\otimes \mathbb{I}[\min(U_{i,1}, \ldots, U_{i,m_i}) > \max(L_{i,1}, \ldots, L_{i,n_i})] \quad (3.12)\end{aligned}$$

which is a piecewise function (by equations 3.8, 3.10 and 3.11). To clarify the process we present an example:

Example 2. *Suppose that symbolic computation of the following case-statement integration is desired.*

$$\int_{-\infty}^{\infty} \Big[x^3 + xy \quad \textit{if } (x > \mathbf{3}), (x > \mathbf{y+1}), (x < \mathbf{2y-7z}), (\mathbf{-yz} > \mathbf{1}), (\mathbf{y} > \mathbf{0})\Big] \, \mathrm{d}x$$

For readability, expressions that determine the lower/upper bound are colored blue and green while constraints that do not involve the integration variable x are colored red. By equation 3.12, this integral is equal to,

$$\mathbb{I}[(\mathbf{-yz} > \mathbf{1}) \wedge (\mathbf{y} > \mathbf{0})] \cdot \left[\int_{\max\{\mathbf{3}, \mathbf{y+1}\}}^{\mathbf{2y-7z}} x^3 + xy \, \mathrm{d}x\right] \otimes \mathbb{I}[\mathbf{2y-7z} > \max\{\mathbf{3}, \mathbf{y+1}\}]$$

The number of pieces in the integral of a piecewise function f can grow exponen-

[3] We assume the integrated function does not involve *Dirac delta*s.

tial in the number of partitions in f. The following proposition shows that this growth is also affected by the atomic constraints in each partition.

Proposition 2 (Complexity of piecewise integration)**.** *Let f be an N-piece function where the number of atomic inequalities in each of its case-statements is bound by k. The upper bound on the number of pieces in the integral of f (w.r.t. a variable x, as in equation 3.12) is $(k^2/4)^N$.*

Proof. It suffices to show that the upper bound on the number of pieces in the integral of a single partition (say, i-th partition) is $L = K^2/4$. Because then by mathematical induction it can easily be shown that the number of pieces in the summation of N, L-piece functions is L^N which will complete the proof.

Consider the following function,

$$\mathbb{I}[\psi_i] \cdot \left(\int f_i \, \mathrm{d}x \Big|_{UB} - \int f_i \, \mathrm{d}x \Big|_{LB} \right) \cdot \mathbb{I}[UB > LB] \tag{3.13}$$

where LB and UB are uninstantiated variables.

According to the discussion presented in Section 3.2.5, the following couple of substitutions,

$$LB \leftarrow \max(L_{i,1}, \ldots, L_{i,n_i}) \qquad UB \leftarrow \min(U_{i,1}, \ldots, U_{i,m_i})$$

in (3.13) leads to a piecewise expression with $n_i \times m_i$ partitions. This function equals the equation (3.12). Therefore the upper bound on the number of pieces in the integral of the i-th partition is $n_i \cdot m_i$.

According to the assumption, the number of atomic constraints in the i-th partition is k. Therefore, the number of atomic constraints that involve x cannot exceed k.

$$n_i + m_i \leq k$$

The equality happens when ψ_i does not contain any constraints. In this situation, if $n_i = m_i = k/2$ then $n_i \cdot m_i$ (i.e. the number of partitions in 3.13) is maximized to $k^2/4$ which is the intended result. □

In order to conduct exact inference on piecewise smooth graphical models, one requires to conduct multiple integrals in closed-form since such integrations are required for analytical multivariate marginalization. Nonetheless, Proposition 2 shows that such computations can rapidly become intractable, which severely restricts the application of exact inference on a high-dimensional piecewise models.

3.3 Representation of piecewise functions

As already mentioned, binary operations over piecewise functions may lead to an exponential growth in the number of output case-statements. In practice, maintaining a large list of case-statements might be prohibitively expensive. A more severe problem is the time complexity of the evaluation of such data structures which also grows

exponentially (in the number of binary operations). The reason is that in the naive representation, with respect to each variable assignment vector, the constraints associated with case-statements should be evaluated one by one until the activated partition is found. This method clearly requires redundant computations. For instance, the product of an M-piece and an N-piece functions leads to $M \cdot N$ case statements. As a result in the naive representation, finding the activated sub-function requires $M \cdot N$ evaluations. However, more efficient data structures do not require more than $M + N$ evaluations since to find the activated sub-function of the product it is sufficient to find the activated sub-functions of the operand functions.

Decision diagrams (DDs) are data structures with an evaluation time complexity that remains linear in the number of binary operations. In particular, in the context of this thesis, we should highlight a particular DD known an *extended algebraic decision diagram* (XADD) which is suitable for representation of piecewise continuous functions. The XADD data structures (introduced by [Sanner et al. 2011]) are compact representations motivated by *algebraic decision diagrams* (ADDs) [Bahar et al. 1997] that in their own turn, generalize the concept of *Binary decision diagrams* (BDDs) [Lee 1959].

Since many properties are in common among these data structures, in this subsection we firstly introduce BDDs and ADDs (in Sections 3.3.1 and 3.3.2 respectively) and then deal with XADDs in Section 3.3.3. Finally, Section 3.3.4 deals with other decision diagrams that although not used in this thesis, may find potential usage in the context of probabilistic inference.

3.3.1 Binary decision diagram (BDD)

Binary decision diagram (BDD), a data structure introduced by [Lee 1959], is a compact representation of Boolean functions, $\mathfrak{B}^n \to \mathfrak{B}$. It is a rooted, directed, acyclic graph (DAG), that consists of 0 or more binary *nonterminal vertices* (i.e. vertices with *out-degree* equal to 2) and some *terminal vertices*. Each nonterminal vertex v is associated with a boolean variable (or w.r.t. some definitions, an index of a variable) and has two child nodes: $\text{LOW}(v)$ (a *low child*) and $\text{HIGH}(v)$ (a *high child*). In the graph, the relation $v' = \text{LOW}(v)$ (similarly $v' = \text{HIGH}(v)$) is represented by a dotted (resp. solid) arrow from v to v'. Each terminal vertex, v, is associated with a binary value. A path from the root vertex to a 1-terminal (or a 0-terminal) represents a variable assignment for which the represented Boolean function returns 1 (resp. 0).[4] In a variable assignment that corresponds to a path linking a vertex v to its high (or low) child, the value assigned to the variable (associated with) v is 1 (resp. 0).

A BDD is in fact a graphical representation of *Shannon's decomposition rule* indicating that Boolean functions can be reduced by means of the following identity:

$$f = x_i \cdot f|_{x_i=1} + (1 - x_i) \cdot f|_{x_i=0} \tag{3.14}$$

where f is a function, $f|_{x_i=b}(x_1, \ldots, x_n) := f(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_n)$ and x_k, $b \in \mathfrak{B}$. Note that $\mathfrak{B} = \{0, 1\}$ (representing $\{\text{false}, \text{true}\}$), $\cdot$ and $+$ denote *conjunction* and

[4] If a path does not contain all variables occurred in the function, the variable assignment is partial.

disjunction operations respectively and $(1 - x_i)$ represents the *negation* of x_i. Figure 3.1 illustrates a BDD that represents $f = x_1.x_2 + \bar{x_1}.x_2$. As it can be seen, neither f nor its corresponding BDD is simplified.

If different variables appear in the same order on all paths from the root (i.e. if the order of a variable associated with a vertex is strictly less than the order of each of its children) it is called an *Ordered BDD* (OBDD). Clearly the BDD of Figure 3.1 is ordered.

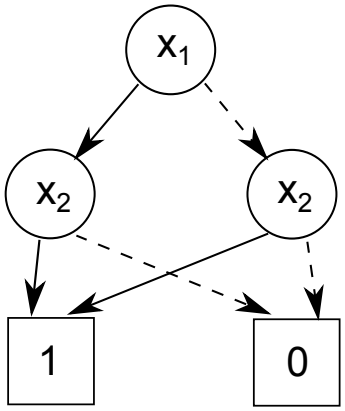


Figure 3.1: An (ordered) BDD representing $f = x_1.x_2 + \bar{x_1}.x_2$

3.3.1.1 Reduced ordered binary decision diagram

An OBDD is *reduced* (called ROBDD) if the following *reduction rules* are applied to it:

1. *S-deletion rule*[5] : If both outgoing edges of a nonterminal vertex v lead to the same vertex w, v is deleted from the graph and the edges that lead to it are redirected to w.

2. *Merging rule*: If there are vertices v and w with the same label (i.e. associated variable) or the same value (in the case of terminal vertices) and the same 0-successor and the same 1-successor (if any), one of those vertices (i.e. v) is deleted from the graph and the edges that lead to it are redirected to the other vertex (i.e. w).

Figure 3.2 represents an ROBDD that corresponds to the graph of Figure 3.1. In order to end in such a graph, the following actions are performed: In the first step, the terminal vertices with same values are merged (by Merging rule). In the second step, vertices with label x_2 are merged (by merging rule). The resulting vertex is deleted in the third step by S-deletion rule.

3.3.1.2 The canonicality of OBDDs

By applying reduction rules (repeatedly until neither of the rules are applicable), all OBDDs that represent the same Boolean function produce the same reduced OBDD

[5]Abbreviation for *Shannon deletion rule*.

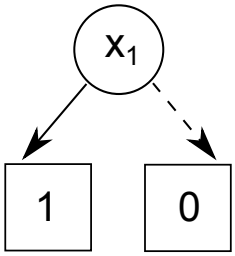


Figure 3.2: A reduced OBDD corresponding OBDD of Figure 3.1

.

[Bryant 1986]. In this sense, reduced OBDDs provide *canonical* representations of Boolean functions. This is an important property of OBDDs; however, it should be noted that the ROBDD size of a function depends on the chosen variable ordering (see [Bryant 1986] for more discussion). The process of finding the optimal variable ordering which leads to an ROBDD with minimum size is referred to as *MinOBDD*. It is known that MinOBDD is NP-hard [Bollig and Wegener 1996].

3.3.2 Algebraic decision diagram (ADD)

ADDs generalize BDDs by allowing non-Boolean terminal vertices [Bahar et al. 1997]. For example, Figure 3.3 depicts an ADD with terminal values $\{0, 1, 2, 3\}$. Note that nonterminal vertices are still Boolean, therefore *Shannon decomposition rule* still holds for them. Since in contrast to ROBDDs, reduced ordered ADDs consist of more than two terminal vertices, they are also known as *multi-terminal BDDs* (MTBDDs) [Clarke et al. 1993]. A variation of ADDs that allow vertices to have more than two children (corresponding multi-valued variables) is known as MDD [Srinivasan et al. 1990].

A major drawback of ADD data structure is that by increasing the number of terminal nodes, the number of internal vertices rapidly increases (exponential in the worst case). To address this issue, various algorithms have been proposed that try to minimize the number of necessary terminal values by introducing edge-values. In the rest of this subsection some of them are introduced.

3.3.3 Extended algebraic decision diagram (XADD)

In the context of probabilistic inference on piecewise smooth models over an n dimensional variable space, we require decision diagrams which have the following properties:

- Nonterminal nodes should be $\mathbb{R}^n \to \mathfrak{B}$ functions. Therefore, partition borders would be representable by hyperplanes.
- Terminal nodes should be $\mathbb{R}^n \to \mathbb{R}$ functions (to represent the sub-functions that are not necessarily constant).

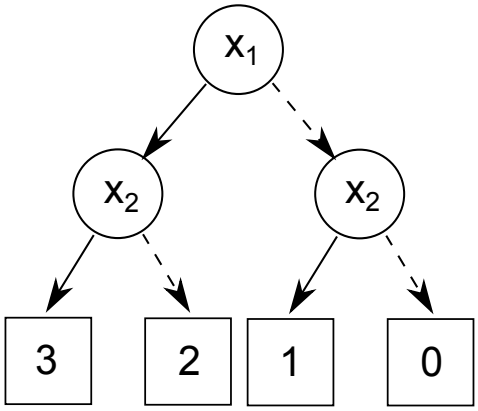


Figure 3.3: An ADD for coding integers 0 to 3

.

These properties are possessed by the extended algebraic decision diagrams (XADDs) (introduced by [Sanner et al. 2011]) since XADD nonterminal vertices are associated with arbitrary first-order logic formulas rather than boolean variables. Moreover, the values of the XADD terminal nodes are arbitrary first order logic terms. As an example, a simple XADD is illustrated in Figure 3.4. It should be pointed out that unlike most DDs, no algorithm is known by which XADDs can be converted to canonical forms.

In the works offered by this thesis that are dealt with in the remaining chapters, the only utilized data structures are XADDs with simplifying assumptions (i.e. working with partial piecewise functions and assuming they are 0 whenever not defined, etc.). Nonetheless, for the sake of completeness in Section 3.3.4 other decision diagrams (namely, edge-valued decision diagrams) are briefly introduced. Although, up to our knowledge, in the context of probabilistic reasoning, these structures are not utilized so far, the possibility of their successful use can be a subject of future investigation.

3.3.4 Edge-valued decision diagrams

ADD (and BDD) diagrams provide efficient representations for functions whose structures are conjunctive/disjunctive in nature. However, the ADD-based (and BDD-based) representation of some functions with additive/multiplicative structures is quite large [Sanner and McAllester 2005]. Edge-valued (or affine) decision diagrams provide compact representations for functions with arithmetic structure. Some such functions (e.g. $f = x_0 + 2x_1 + 4x_2 + ...2^n x_n$) can be represented exponentially more compactly with Edge-valued ADDs (resp. Edge-valued BDDs).

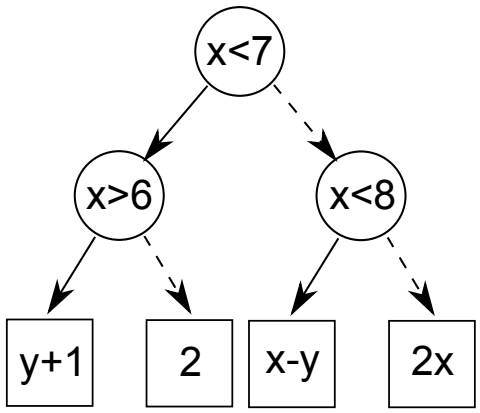


Figure 3.4: A typical XADD diagram.

3.3.4.1 Edge-valued BDD (EVBDD)

EVBDD [Lai and Sastry 1992] provides some advantages over OBDDs. For example, it can be derived from a higher level specification (e.g., 'x + y' as a specification of an adder) instead of requiring a vector of boolean expressions. EVBDD preserves both canonical and compactness properties from OBDDS. It is usually much more compact than a OBDD when it is used to describe an arithmetic function. In this data structure, each nonterminal vertex v is labeled by a variable x and a parameter v called a (multiplicative) weight. It computes a function that is represented by the 2-tuple $\langle \mathrm{v}, x \rangle$. As before, x is a boolean variable associated with a vertex. As will be shown below, the weight v is in fact associated with the high outgoing edge of the vertex (i.e. the outgoing edge that links v to its high child) and is added to a function that is calculated by this node. The only terminal edge of a (reduced) EVBDD has value 0 and is not associated with an additive weight. However, there is a link to the root of the graph associated with a parameter c. The function that is represented by an EVBDD is $f := c + f_{\mathrm{root}}$, where f_{root} is the function calculated in the root of the graph. The set of all variables that label internal vertices of the graph is the same as the set of all free variables occurring in f. The function that corresponds to a vertex v associated with variable x_i and additive weight v_i, is calculated by the following relation:[6]

$$f_v = \langle \mathrm{v}_i, x_i \rangle = x_i(\mathrm{v}_i + f_{\mathrm{HIGH}(v)}) + (1 - x_i) f_{\mathrm{LOW}(v)} \tag{3.15}$$

where $f_{\mathrm{HIGH}(v)}$ and $f_{\mathrm{LOW}(v)}$ are functions calculated in the high child and low child of v respectively.

Let us clarify these definitions and operations by an example. Figure 3.5 represents the EVBDD that corresponds to function $f = -2 + 5x_2 + x_2x_3 + 3x_1x_2 + 4x_1x_2x_3 -$

[6] In some works, the multiplicative weights of EVBDD vertices affect both edges [Drechsler et al. 1996], [Drechsler and Sieling 2001].That is: $f_v := \langle \mathrm{v}_i, x_i \rangle := \mathrm{v}_i(x_i \cdot f_{\mathrm{HIGH}(v)} + (1 - x_i) f_{\mathrm{LOW}(v)})$.

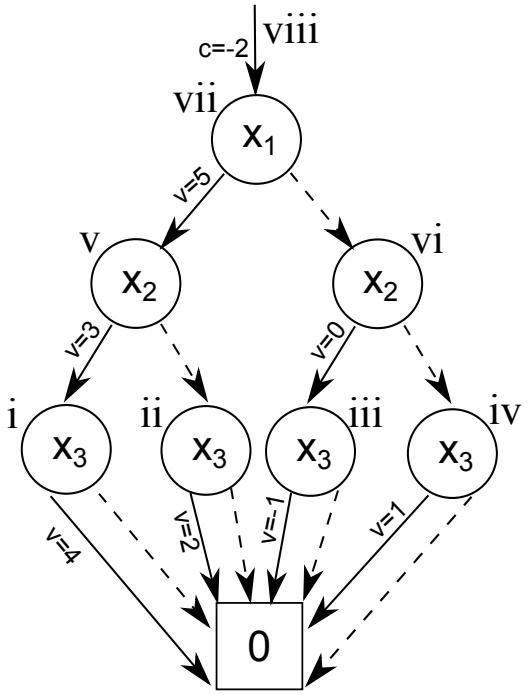


Figure 3.5: EVBDD representing $f = -2 + 5x_2 + x_2x_3 + 3x_1x_2 + 4x_1x_2x_3 - 2x_1x_3 + x_3$.

$2x_1x_3 + x_3$ where $x_1, x_2, x_3 \in \{0, 1\}$. Variables associated with vertices are shown as labels inside the nodes and additive weights associated with each vertex is depicted on the vertex's high outgoing edge to emphasize that such weights are only added to the functions calculated in the high child vertices (However, other variations exist as well). Latin numbers are the indices of internal vertices (except (viii) which is the index of the function of the graph i.e. f). This process is shown in Table 3.1.

Table 3.1: Calculating the function represented by EVBDD of Figure 3.5.

Index	Vertex Function	Function formula
	T	$T = 0 =$ value of the terminal node
i	$\langle 4, x_3 \rangle$	$x_3(4 + T) + (1 - x_3)T = 4x_3$
ii	$\langle 2, x_3 \rangle$	$x_3(2 + T) + (1 - x_3)T = 2x_3$
iii	$\langle -1, x_3 \rangle$	$x_3(-1 + T) + (1 - x_3)T = -x_3$
iv	$\langle 1, x_3 \rangle$	$x_3(1 + T) + (1 - x_3)T = x_3$
v	$\langle 3, x_2 \rangle$	$x_2(3 + \langle 4, x_3 \rangle) + (1 - x_2)\langle 2, x_3 \rangle$
vi	$\langle 0, x_2 \rangle$	$x_2(0 + \langle -1, x_3 \rangle) + (1 - x_2)\langle 1, x_3 \rangle$
vii	$\langle 5, x_1 \rangle$	$x_1(5 + \langle 3, x_2 \rangle + (1 - x_1)\langle 0, x_2 \rangle$
viii	f	$-2 + \langle 5, x_1 \rangle$

3.3.4.2 Factored Edge-Valued BDD (FEVBDD)

In this data structure which is an extension to EVBDD, in addition to EVBDD's additive weight (associated with each high edge), a multiplicative weight is associated with each edge (either high or low) [P. Tafertshofer 1997]. FEVBDDs can be used to represent a wider range of functions concisely. As a result, the computational complexity of certain operations can be reduced (compared to EVBDDs). Let v be a vertex, x be its associated boolean variable, v_{add} (which corresponds to what was named v in the case of EVBDDs) be the additive weight added to the result of the high child of v, and w_h and w_l be the multiplicative weights associated with the high and low outgoing edges of v respectively. The function calculated in v is:

$$f_v = \langle \mathrm{v}, x, w_h, w_l \rangle = x(\mathrm{v} + w_h \cdot f_{\mathrm{HIGH}(v)}) + (1 - x) \cdot w_l \cdot f_{\mathrm{LOW}(v)}$$

3.3.4.3 FDD, OFDD, BMD and *BMD

Decomposition of Boolean functions by (positive) *Davio decomposition rule* i.e.

$$f = f|_{x_i=0} \oplus x_i(f|_{x_i=0} \oplus f|_{x_i=1}) \tag{3.16}$$

rather than the Shannon's decomposition rule (represented in equation (3.14)) results in *functional decision diagram* (FDD) (and *ordered functional decision diagram* OFDD) [Kebschull and Rosenstiel 1993; Drechsler and Sieling 2001] rather than BDD (resp. OBDD) where $\oplus$ stands for modulo-2 sum (correspondent to Exclusive Or in propositional logic). For the decision diagrams in which each path contains all variables, there is a close relation between BDD and FDD representations [Becker et al. 1995] which is given by a boolean transformation called *Muller Transform*. In general, FDDs and BDDs somehow compliment of each other in the sense that in the case of many functions, FDD representation might be appropriate while BDD is not efficient and vice versa. More specifically, for ordered FDDs, AND and OR operations may lead to an exponential blow-up while for ordered BDDs, exponential explosion may happen for XOR operations. Multi-terminal generalization of ordered FDD is known as *binary moment diagram* (similar to the way ADD generalizes BDD) and multiplicative generalization of BMD is known as *BMD (similar to the way EVBDD generalizes ADD) [Bryant and Chen 1995] with a relation comparable to (3.15):

$$f_v = \langle \mathrm{v}_i, x_i \rangle = \mathrm{v}_i(f_{\mathrm{LOW}(v)} + x_i(f_{\mathrm{HIGH}(v)} - f_{\mathrm{LOW}(v)}) \tag{3.17}$$

3.3.4.4 Hybrid decision diagrams: OKFDD, KBMD, K*BMD, *PHDD

Ordered Kronecker functional decision diagrams (OKFDDs) use more than one decomposition rule in a single graph [Becker et al. 1997]. That is, among the three decomposition rules, Shannon, positive Davio and negative Davio rule, dynamically, a rule that leads to better compression of a subgraph is chosen. [Drechsler and Becker 1995] *Kronecker Multiplicative BMD* data structures (KBMDs) generalize BMDs by the same idea i.e. by allowing different decomposition types per variable. K*BMD is a

variation of KBMD that like FEVBDD uses a multiplicative weight as well [Drechsler et al. 1996]. *Multiplicative power hybrid decision diagram* (*PHDD) is a version of K*BMD where weights are interpreted as powers (of a given basis) [Chen and Bryant 1997].

3.3.4.5 Affine Algebraic Decision Diagrams

The affine ADD (AADD) is a data structure where each vertex function is represented by a 5-tuple $\langle x, c_h, b_h, c_l, b_l \rangle$ where as usual, x is the variable associated with the vertex and c_h and b_h (similarly c_l and b_l) are additive and multiplicative weights of the high (resp. low) outgoing edge of the vertex.

$$f_v = \langle x, c_h, b_h, c_l, b_l \rangle = x(c_h + b_h \cdot f_{\text{HIGH}(v)}) + (1 - x)(c_l + b_l \cdot f_{\text{LOW}(v)})$$

Here, c_h and c_l are real values in the range $[0, 1]$, b_h and b_l are real values in $(0, 1]$ and there are some restrictions on them such as: $\min(c_h, c_l) = 0$, $\max(c_h + b_h, c_l + b_l) = 1$, etc. AADDs can represent additive and multiplicative structures in a compact form and therefore, are suitable for probabilistic inference [Sanner and McAllester 2005].

3.4 Pruning piecewise functions and processing symbolic expressions

In Section 3.2 it was shown that a binary operation over an N-piece and an M-piece functions results in an $M \times N$-piece function. Nonetheless, in practice, the constraints associated with many such pieces may be infeasible (that is, correspond to empty partitions) and therefore, can be pruned i.e. eliminated from the output structure. In our implementation of piecewise case-statements, we have introduced a set of simple patterns and rules that can detect some unsatisfiable combination of constraints. Over time, we can add more rules to the code to be able to identify more such constraints. The ultimate goal is to implement a robust automated system that addresses the constraint satisfiability problem of first order logic sentences. Such tasks are typically performed by *automated theorem proving* (ATP) systems as well as frameworks referred as *satisfiability modulo theories* (SMTs). These systems can be regarded as generalizations of Boolean SAT problem (in the propositional calculus) to (decidable subsets of) first order logic that (particularly in the case of SMTs) are interpreted in the context of specific theories (in our case, the theory of real arithmetic). Otherwise stated, to detect infeasible partitions, we are interested in solving symbolic inequalities i.e. inequalities involving uninterpreted terms. These inequalities over real variables are evaluated with respect to the theory of real arithmetic.

Apart from solving inequalities, in Chapter 6 (and to a lesser extent in Chapter 5) we are interested in factorization of (multivariate) polynomials and symbolic integration of fractional functions. We will impose some restriction on the models to be able to carry out these tasks symbolically by relying on a small set of solution formulas. In the future we may enrich this set of formulas. This will enable us to handle larger sets of probabilistic models. Automated factorization and integration are tasks which are

typically accomplished by computer algebra systems (CASs). As a matter of fact, we can claim that in order to carry out the inference tasks that will be introduced in the next chapters, we have already implemented a simple CAS/ATP/SMT software.

In case the readers find the contributions offered in this thesis applicable to the domains that they are interested in but they require to handle a larger set of models, they may need to consider the abilities of the existing full-fledged CAS and ATP/SMT systems with an eye to the possibility of combining them with our algorithms. To address such a potential requirement, in this subsection we will introduce the basic principles and functionalities of the existing such systems.

Since both Computer algebra systems (such as MAPLE) and computer theorem provers (like Vampire) are computer systems with the formal ability of symbolic manipulation, they may superficially look similar, however, in practice there is surprisingly little common ground between them, either by the way they internally work or by the communities of their users and creators [Harrison 1996]. Computer algebra systems are used mostly by applied mathematicians, scientists and engineers. These systems work quickly, (due to imposing some intuitive assumptions that in rare cases do not hold but in general lead to efficient solutions) occasionally make mistakes and mainly perform algebraic and continuous mathematics such as differentiation and integrals. In contrast, theorem provers are mainly used by computer scientists interested in system verification or by logicians experimenting with new logics. These latter systems, perform logical reasoning by means of proof procedures for particular domains such as natural numbers. They are typically biased towards discrete mathematics, are meant to be reliable and are comparatively more difficult to use [Harrison 1996].

3.4.1 Computer Algebra Systems (CASs)

CASs such as Axiom, Derive, Macsyma, Maple, Mathematica, MuPAD, Reduce, SymbolicC++, FORM, Fermat, GAP, Magma, Mathomatic, Maxima, PARI/GP, OpenAxiom, Symbolism, SymPy, Xcas and Sage are software programs that perform manipulation of mathematical expressions in symbolic form.

Expressions. Typical expressions which are manipulated by CASs include: polynomials, logarithmic, exponential and trigonometric functions, arbitrary function symbols and constants, derivatives, integrals, sums, products, matrices and numerical values.

Manipulations. Typical manipulations include: simplification of expressions to smaller (or standard forms); expansion of expressions; symbolic/numeric substitution of symbols; factorization; closed form differentiation/(indefinite) integration; solutions for (non)linear equations; taking limits, matrix operations, etc.

Some underlying mathematical/symbolic techniques of CASs.

- *Rule-based operations*: Closed-form (in)definite integrals, derivatives, limits, operations of mathematical series, etc. of many types of expressions (such as some (inverse) trigonometric, (inverse) hyperbolic, exponential, logarithmic and Gaussian and other special functions) are known, hence maintained/applied by means of a look-up table [Brychkov 2008].

- *Heuristics algorithms*: In the absence of a general symbolic solution for definite integrals of some functions related to Laplace/Fourier transforms, CAS developers have implemented heuristics based on pattern matching, differentiation, variable transformation, and occasionally, the use of limits by starting from some special functions (including beta and (incomplete) gamma functions) [Geddes et al. 1990]. Although this approach is heuristic rather than algorithmic, it is an effective method for solving many definite integrals encountered by practical engineering applications. This method was pioneered by developers of *Maple* system and later on, emulated by *Mathematica*, *Axiom* and other systems.
- *Risch algorithm*: This algorithm determines if an indefinite integral of an elementary function (i.e a function built from a finite number of exponentials, logarithms, constants, and roots through composition and combinations using the four elementary operations) is elementary and in case it is, integrates it [Davenport 1981]. The generalized Risch algorithm is implemented by Manuel Bronstein in *Axiom* CAS [Bronstein 1990].
- *Cantor–Zassenhaus algorithm*: is arguably the dominant algorithm for solving the problem of factorising polynomials over finite fields. The algorithm consists mainly of exponentiation and polynomial greatest common divisor (GCD) computations. [Cantor and Zassenhaus 1981]
- *Gaussian elimination*: is an algorithm for solving systems of linear equations by means of a sequence of operations performed on their associated matrix of coefficients. This method is also used to find the rank of a matrix, to calculate the determinant of a matrix, and to calculate the inverse of an invertible square matrix. [Grcar 2011]

Algebraic geometry. Algorithmic geometry [Cox et al. 1998] and in particular, algorithms providing solutions for systems of multivariate polynomial equations as well as finding *polynomial greatest common divisor* (or more generally, polynomial factorization) are quite relevant to the scope of the proposed thesis and in particular can be used for the simplification of expressions required in an inference algorithm called *symbolic Gibbs sampling* that will be introduced in Chapter 5.

3.4.2 Automated theorem-proving

Automated theorem proving (ATP) (or automatic deduction) is a subfield of automated reasoning (along with interactive theorem proving, automated proof checking, analogy induction and abduction) dealing with proving mathematical theorems by computer programs. First-order theorem proving is one of the most mature subfields of automated theorem proving. Some famous first-order ATP systems are: E [Schulz 2002], SETHEO [Letz et al. 1992], Vampire [Riazanov and Voronkov 2002] and SPASS [Weidenbach et al. 2009].

Proof theories. There exist many calculi of formal proofs including Hilbert-style calculi, Natural deduction systems such as Nm, Ni, Nc and Gentzen systems such as

G1, G2, G3, one sided Gentzen systems GS1, GS2, GS3 and their variations [Troelstra and Schwichtenberg 2000]. Each branch of proof calculi differ in some axioms. Many proof procedures of Genzen systems are based on a rule called *cut*. It is defined as

$$\frac{\Gamma \Rightarrow \Delta, A \qquad A, \Gamma' \Rightarrow \Delta'}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'},$$

where a typical expression $A, B \Rightarrow C, D$ is a *sequent* and should be read as: "The conjunction of (formulas in) A and B proves the disjunction of (formulas in) C and D". *Cut* is not an axiom for any Gentzen system, since *Gentzens cut elimination* theorem states, "if a sequent is provable using cut rule, it has a cut-free proof as well". However, cut-free proofs are typically very long and in this sense, *cut* is a powerful tool.

Resolution. An instance of *cut* rule, called *(linear) resolution rule* [Robinson 1965; Troelstra and Schwichtenberg 2000] is as follows:

$$\frac{\Gamma \Rightarrow A \qquad A, \Gamma' \Rightarrow}{\Gamma, \Gamma' \Rightarrow}$$

If $\Gamma = \{\neg a_1, \ldots \neg a_n\}$, $\Gamma' = \{\neg b_1, \ldots \neg b_m\}$ and $A = \{c\}$, the resolution rule (for propositional logic) leads to:

$$\frac{a_1 \vee \ldots \vee a_n \vee c \qquad b_1 \vee \ldots \vee b_m \vee \neg c}{a_1 \vee \ldots \vee a_n \vee b_1 \vee \ldots \vee b_m}$$

Resolution method for propositional logic. Using the resolution rule, a sound and complete algorithm for deciding the satisfiability of a propositional formula can be constructed: If there exists a directed acyclic graph (DAG) with vertices labeled by propositions in *conjunctive normal form* (CNF) where each sibling pair of vertices are labeled by clauses in form $A \vee X$ and $B \vee \neg X$, and a corresponding parent vertex labeled $A \vee B$ such that the root of the DAG is labeled with the empty clause, the *knowledge base* (KB) corresponding to the leaves of the tree is unsatisfiable, otherwise it is satisfiable.

Resolution method for first-order logic. In order to perform resolution on a first-order knowledge base, the sentences should be converted to Prenex normal form (i.e. in each sentence, all quantifiers should be pushed to the left side, leaving the right part quantifier free) removing existential quantifiers by Skolemization [Boolos et al. 2002], removing universal quantifiers (since free variables are understood as being universally quantified) and performing resolution on a pair of clauses containing the same predicate (negated in only one clause) after performing an appropriate process known as *unification* [Gallier 1985]. The goal of unification is to find a substitution demonstrating that two seemingly different terms are in fact either identical or just equal. Three important types of FOL *unification* processes are discussed next:

- **(Syntactic) unification.** The simplest and most widely used kind of unification does not need any assumption on the interpretation of symbolic expressions.

For example the formulas $f(x, a)$ and $f(b, y)$ are unified under the substitution $\{x \mapsto b, y \mapsto a\}$.

- **Associative-commutative unification (AC-Unification)** Since functions which are associative and commutative (such as the arithmetic addition and multiplication functions) are often the subject of automated theorem proving, [Stickel 1981] presents an algorithm which unifies terms whose corresponding functions have these properties. For example by AC-Unification, expressions $z \times (x \times x) + y$ and $y + (w \times x) \times x$ can be unified by the substitution $\{w \mapsto z\}$.

- **Equational-unification (E-Unification)** E-unification is concerned with solving term equations modulo an equational theory E. Syntactic unification can be seen as a special kind of E-Unification where $E = \emptyset$. Similary, AC-Unification is a special kind of E-Unification where $E = \{f(x, y) = f(y, x),\ f(x, f(y, z)) = f(f(x, y), z)\}$ [Baader 1992]. As another example, if besides associativity and commutativity of $\times$ and $+$ functions, E includes distributivity of $\times$ over $+$, by E-Unification, expressions $x \times (y + z)$ and $x \times z + w \times x$ can be unified by substitution $\{y \mapsto w\}$.

3.4.3 Satisfiability Modulo Theories (SMT)

SMT addresses the problem of constraint satisfaction for decidable subsets of first order logic (just as SAT is the problem of constraint satisfaction in propositional calculus). The aim of Satisfiability Modulo Theories (SMT) is to check satisfiability of a (set of) quantifier-free formula(s) F with respect to one or more theories.[7] SMT is based on a practical viewpoint towards the satisfiability of first-order logic. That is, *SMT solvers* concentrate on specific classes of background theories and try to address the problem of satisfiability w.r.t. those theories (using domain specific decision procedures [Manna and Zarba 2003]), rather than dealing with the problem in a general and theoretic sense.

A (quantifier-free) first order formula is a logical combination of atomic sentences analogous to the combination of propositional variables in a proposition. The difference is that while propositional variables take value independently from each other, in general, atomic sentences do not; for example both atomic sentences $x > 0$ and $x < 0$ cannot be true in one interpretation. The task of determining whether a conjunction of atomic sentences (also known as constraints or theory literals) is satisfiable or not is performed by a process known as *theory solving*. Therefore, a typical SMT solver is a combination of a SAT technique and some theory solvers (T-solvers) [de Moura and Bjrner 2009].

Let us clarify these concepts by an example: Consider the satisfiability problem of

[7] In the literature of SMT, theory is often defined as a set of sentences (e.g. see [de Moura and Bjrner 2009]). However, it should be mentioned that by a more formal definition, theory is a set of sentences that includes all consequences of itself [Boolos et al. 2002].

the following sentence ϕ:

$$\phi \equiv (y - x \geq 0) \wedge (z - y \geq 0) \wedge ((x - z \geq 2) \vee (x - z \geq 4))$$

From left to right, let associate atomic sentences with propositions a to d. The following propositions R_1 to R_4 are due to the transitivity property of $\geq$:

$$R_1 \equiv \neg(a \wedge b \wedge c) \quad R_2 \equiv \neg(a \wedge b \wedge d) \quad R_3 \equiv \neg(\neg a \wedge \neg b \wedge \neg c) \quad R_4 \equiv \neg(\neg a \wedge \neg b \wedge \neg d)$$

In this example, satisfiability of R_1 to R_4 has to be determined by a *theory solver* (T-solver). On the other hand, the structure of ϕ is encoded by the following proposition P:

$$P \equiv a \wedge b \wedge (c \vee d)$$

and the task of determining the satisfiability of conjunction of R_1 to R_4 and P is delegated to a SAT-solver.

With respect to the way SAT and theory solvers are combined, [Sheini and Sakallah 2006] divides SMT solvers into three main categories:

1. *Eager approaches:* In these approaches, in a single step, the original first-order formulas are translated to pure propositional formulas which have solutions isomorphic to the solutions of the original problem (see e.g. UCLID [Lahiri and Seshia 2004]). Therefore, these approaches directly convert SMT problems to SAT problems. In the case of the above example, this means directly feeding $\neg(a \wedge b \wedge c) \wedge \neg(a \wedge b \wedge d) \wedge \neg(\neg a \wedge \neg b \wedge \neg c) \wedge \neg(\neg a \wedge \neg b \wedge \neg d) \wedge (a \wedge b \wedge (c \vee d))$ to a SAT solver. In general, finding such a translation might be hard and may lead to an explosion of boolean variables.

2. *Off-line lazy approaches:* The proposition that represents the structure of the original sentence (e.g. P in the former example) is fed to a SAT solver reporting the set of (proposition) truth assignments under which it is satisfiable (e.g. $\{a \wedge b \wedge c \wedge \neg d, a \wedge b \wedge c \wedge d, a \wedge b \wedge \neg c \wedge d\}$ in the former example). Obviously, if such a set is empty, regardless of the semantics of the atomic sentences, the original formula is unsatisfiable. If not, the conjunction of atomic sentences that corresponds to each truth assignment (i.e. the conjunction of true theory literals) are fed to a T-solver. If at least one such conjunction is satisfiable then the original formula is satisfiable, otherwise it is not.

3. *Online solving approaches:* These methods are very similar to off-line approaches with the difference that the SAT and theory solvers are tightly combined such that the consistency of theory literals are checked incrementally as soon as they are returned by the SAT solver For instance in the previous example, as soon as SAT solver generates $\neg(a \wedge b \wedge c \wedge d)$, its validity is checked by the T-solver and in case it is valid, SAT solver does not need to generate the other possible truth assignments.

SAT solvers used in SMT solvers are typically based on the DPLL backtrack search algorithm [Davis et al. 1962]. The variety of theory solvers is much more and a typical

SMT solver combines more than one theory solver. Some classes of theories integrated in modern SMT solvers are as follows:

Difference arithmetic theories (DA). Expressions in which the only allowed form of predicates is: $x - y \leq c$, where x and y are variables (or constants) and c is a numeral (constant).[8]

Such a problem can easily be solved by Bellman-Ford graph negative-weight cycle detection algorithms, a survey of which is given in [Cherkassky and Goldberg 1996].

Linear (additive) arithmetic theories (LA). Expressions (i.e. theories) involving relations $<$, $\leq$, $+$ and function symbols $+$, $-$ and $\times$ where only multiplication with numerals is allowed. Solving these kinds of expressions is also relatively easy (see e.g. [Dutertre and Moura 2006]).

Equality (free functions). If a theory only consists of (un)equality of some terms, its satisfiability can be decided by a simple process called *congruence closure* [Bachmair et al. 2003]. This algorithm basically constitutes congruence classes for equal terms and then iteratively, adds more complicated structures (which are made of congruent pairwise elements) to new congruent classes.

3.5 Summary

This chapter contained the prerequisite materials related to piecewise continuous algebra that are required in the next Chapters or may be useful in their potential future extensions.

The formal definition of piecewise functions as well as the relevant notations that are used throughout were presented in Section 3.1. Section 3.2 dealt with the piecewise operations as well as their complexity and worst-case increment in the number of resulting partitions. These operations include addition, multiplication and integration as well as min/max that are typical operations required for probabilistic inference.

In Section 3.3, various decision diagrams, i.e. data structures suitable for compact representation of piecewise functions, were introduced. Among them, *extended algebraic decision diagram* (XADD) is suitable for representation of piecewise continuous (or piecewise discrete/continuous hybrid) functions (see Section 3.3.3) and therefore, is the only data structure used throughout. Nonetheless, *edge-valued decision diagrams* (see Section 3.3.4) are capable of representing additive and multiplicative structures in a compact manner and can potentially be utilized in future works.

We showed that binary operations on piecewise structures can lead to exponential blow up in the number of partitions. To prevent this behavior, a key insight is that in many cases the resulting partitions are empty (i.e. correspond to infeasible constraints). To detect and omit such partitions, *automated theorem proving* (ATP) and *satisfiability modulo theory* (SMT) tools can be used. These techniques and tools were

[8] Evidently, equality constraints $x = y$ can be captured by $x - y \leq 0 \wedge y - x \leq 0$. For $x, y, z \in \mathbb{Z}$, strict inequalities (such as $x - y < c$) are expressible as: $x - y \leq c - 1$, and the negation of $x - y \leq c$ is $y - x \leq -c - 1$.

briefly introduced in Section 3.4. While our current implementation uses simple SMT rules, in order to increase its performance, in future we may combine it with full-fledged ATP and SMT systems.

Familiarized with probabilistic inference in Chapter 2 and reinforced with the piecewise algebra in the current chapter, we are now ready to proceed to the next chapters where the actual contributions are presented.

Chapter 4

Gibbs Sampling from augmented piecewise models

In Section 3.2, it was stated that in general, exact closed-form inference on highly piecewise graphical models is intractable. Due to being asymptotically unbiased, Markov Chain Monte Carlo methods provide the second best option. Nonetheless, as argued in section 2.3.2, in terms of the rate of convergence to the target distribution, the performance of most MCMC methods on piecewise models is typically low. For instance, respectively in Sections 2.3.2.6 and 2.3.2.8 we stated that Metropolis-Hastings (MH) and Hamiltonian Monte Carlo (HMC) algorithms are designed based on the assumption of smoothness of the target distribution — an assumption that in the case of piecewise functions, does not hold near the partitioning borders.

Luckily, Gibbs sampling does not suffer from this problem (see Section 2.3.2.7) and can be successfully applied to piecewise densities. Nonetheless, if the number of model partitions in relatively large, this sampler becomes slow since in this case, in each step of Gibbs sampling many (univariate) integrations are required.

The later problem typically occurs in case a Graphical model consists of several piecewise factors since the number of partitions in the joint distribution can grow exponentially in the number of such factors (see Section 3.2.2). This is exactly the problem tackled in this chapter:

We propose a Gibbs sampling mechanism that takes samples from a Graphical model with piecewise factors in linear (rather than exponential) time.

More specifically, we propose a mechanism to transform piecewise models into equivalent mixture models and then apply blocked Gibbs sampling to the transformed models and achieve an *exponential-to-linear* reduction in space and time compared to a conventional Gibbs sampler. This enables fast, asymptotically unbiased Bayesian inference in an expressive class of piecewise graphical models. As our empirical results will show, particularly in models where the number of partitions in the joint distribution grows rapidly in the number of piecewise factors, our proposed sampler requires orders of magnitude less time than Metropolis-Hastings and conventional Gibbs sampling methods to achieve the same level of accuracy.

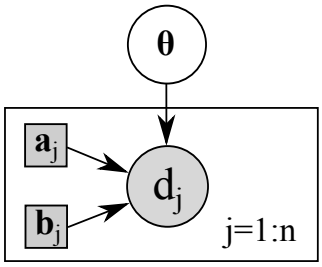


Figure 4.1: Graphical model for BPPL problem described in Section 4.1.

Without loss of generality and for notational convenience, throughout this chapter our focus is on Bayesian inference problems (introduced in Section 2.2.1.1) where the likelihood functions are piecewise. An instance of this problem is presented in Section 1.1.4 where Figure 1.4 clearly shows that in such problems, the number of posterior partitions often grows exponentially in the number of observed data points.

To motivate the discussion, in Sections 4.1 and 4.2 we formalize two case studies where the likelihood functions are naturally piecewise. These models are revisited in the latter part of this chapter where the performance of MCMC samplers in empirically compared.

The complexity of Bayesian inference in piecewise models is studied formally in Section 4.3. The novel inference algorithm (referred to as *augmented Gibbs* sampling throughout) as well as the proof of its correctness are presented in Section 4.4. The experimental results are dealt with in Section 4.5. These outcomes indicate the in many models, the proposed algorithm remarkably outperforms the baseline Gibbs and Metropolis-Hastings sampling methods.

4.1 Case study I: Bayesian Pairwise Preference Learning (BPPL)

In order to provide a concrete example for a Bayesian inference model where the likelihood functions are piecewise, we begin with a case study, namely, *Bayesian pairwise preference learning* (BPPL).

Preference learning deals with the problem of Bayesian utility learning given an agent's preferences. In this setting, the objective is to infer a posterior distribution over an agent's utility function based on previously observed preferences. These posterior utility functions can then be used in a variety of applications such as predicting the future agent's decisions, etc.

As it will be formalized shortly, BPPL is a Bayesian approach to preference learning where the objective is to learn a user's weighting of attributes (such as color, size, price, etc.) for classes of items (e.g., cars, apartment rentals, movies) given their responses to pairwise comparison queries over those items.

Bayesian pairwise preference learning (BPPL). Let each *item* $\mathbf{a}$ be modeled by a D-dimensional real-valued *attribute choice vector*:

$$\mathbf{a} := (\alpha_1, \ldots, \alpha_D)$$

The goal is to learn an *attribute weight vector* $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_D) \in \mathbb{R}^D$ that describes the utility of each attribute choice from user responses to preference queries. As commonly done in *multi-attribute utility theory* [Keeney and Raiffa 1993], the overall item utility $u(\mathbf{a}|\,\boldsymbol{\theta})$ is decomposed additively over the attribute choices of $\mathbf{a}$:

$$u(\mathbf{a}|\,\boldsymbol{\theta}) = \sum_{i=1}^{D} \theta_i \cdot \alpha_i \tag{4.1}$$

User responses are in the form of n queries (i.e. observed data points) d_1 to d_n where for each $1 \leq j \leq n$, d_j is a pairwise comparison of some items $\mathbf{a}_j$ and $\mathbf{b}_j$ with the following possible responses:

- $\mathbf{a}_j \succ \mathbf{b}_j$: In the j-th query, the user prefers item $\mathbf{a}_j$ over $\mathbf{b}_j$.
- $\mathbf{a}_j \preceq \mathbf{b}_j$: In the j-th query, the user does not prefer item $\mathbf{a}_j$ over $\mathbf{b}_j$.

It is assumed that with an *elicitation noise* $0 \leq \eta < 0.5$, the item with a greater utility is preferred:

$$p(\mathbf{a}_j \succ \mathbf{b}_j \,|\, \boldsymbol{\theta}) = \begin{cases} \eta & \text{if } u(\mathbf{a}_j|\boldsymbol{\theta}) < u(\mathbf{b}_j|\boldsymbol{\theta}) \\ 0.5 & \text{if } u(\mathbf{a}_j|\boldsymbol{\theta}) = u(\mathbf{b}_j|\boldsymbol{\theta}) \\ 1-\eta & \text{if } u(\mathbf{a}_j|\boldsymbol{\theta}) > u(\mathbf{b}_j|\boldsymbol{\theta}) \end{cases} \tag{4.2}$$

Clearly, since $\mathbf{a}_j$ is either preferred to $\mathbf{b}_j$ or not:

$$p(\mathbf{a}_j \preceq \mathbf{b}_j \,|\, \boldsymbol{\theta}) = 1 - p(\mathbf{a}_j \succ \mathbf{b}_j \,|\, \boldsymbol{\theta}) \tag{4.3}$$

The corresponding graphical model is illustrated in Figure 4.1 and indicates that our posterior belief over the user's attribute weights is provided by the standard Bayesian inference expression:

$$p(\boldsymbol{\theta}|\, d_1, \ldots, d_n) \propto pr(\boldsymbol{\theta}, d_1, \ldots, d_n) = pr(\boldsymbol{\theta}) \cdot \prod_{j=1}^{n} p(d_j|\,\boldsymbol{\theta}) \tag{4.4}$$

A simple instance of the provided BPPL setting is provided in Figure 4.2. It is assumed that items are associated with a couple of real-valued attributes (e.g. item weight and price),

$$\mathbf{a} = (\alpha_1, \alpha_2)$$

The utility weight θ_i of each attribute α_i is unknown in advance therefore as Figure 4.2-

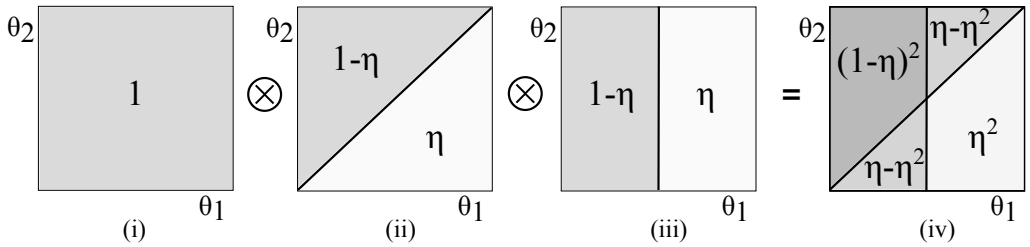


Figure 4.2: A two dimensional instance of BPPL model: (i) An (unnormalized) prior uniform in a rectangle with center (0,0). (ii) Likelihood model $p(\mathbf{a}_1 \succ \mathbf{b}_1 | \boldsymbol{\theta})$ and (iii) $p(\mathbf{a}_2 \succ \mathbf{b}_2 | \boldsymbol{\theta})$ (as in Eq. 4.2) where $\mathbf{a}_1 = (5,3)$, $\mathbf{b}_1 = (6,2)$, $\mathbf{a}_2 = \mathbf{a}_1$ and $\mathbf{b}_2 = (6,3)$. (iv) A piecewise function proportional to the posterior distribution.

i shows, a uniform distribution centered at 0 is assigned to each,

$$\theta_i \sim U(-c, c) \qquad \text{for } i = 1, 2$$

The value of parameter c is not important since the only thing that matters is the relative weight of the attributes.

Suppose it is observed that the agent prefers item $\mathbf{a}_1 = (5,3)$ over $\mathbf{b}_1 = (6,2)$. By definitions 4.1 and 4.2, the likelihood of such an event is,

$$p(\mathbf{a}_1 \succ \mathbf{b}_1 \,|\, \boldsymbol{\theta}) = \begin{cases} \eta & \text{if } 5.0 \cdot \theta_1 + 3.0 \cdot \theta_2 < 6.0 \cdot \theta_1 + 2.0 \cdot \theta_2 \\ 0.5 & \text{if } 5.0 \cdot \theta_1 + 3.0 \cdot \theta_2 = 6.0 \cdot \theta_1 + 2.0 \cdot \theta_2 \\ 1 - \eta & \text{if } 5.0 \cdot \theta_1 + 3.0 \cdot \theta_2 > 6.0 \cdot \theta_1 + 2.0 \cdot \theta_2 \end{cases}$$

which is depicted in Figure 4.2-ii. Note that the second case-statement can be discarded since it is associated with the constraint,

$$\theta_1 = \theta_2$$

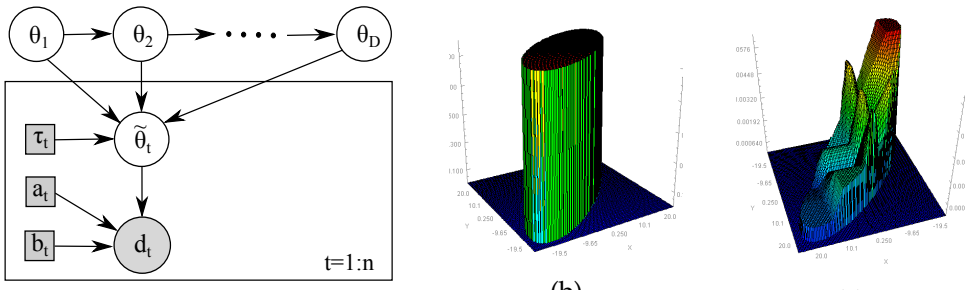
which corresponds to a 0-volume partition value of which does not affect the cumulative density function. Therefore for simplicity, we drop it and represent the aforementioned likelihood by a partial function that after simplification is as follows,

$$p(\mathbf{a}_1 \succ \mathbf{b}_1 \,|\, \boldsymbol{\theta}) = \begin{cases} \eta & \text{if } \theta_2 < \theta_1 \\ 1 - \eta & \text{if } \theta_2 > \theta_1 \end{cases} \tag{4.5}$$

Also suppose that we observe that the agent also prefers item $\mathbf{a}_2 = (5,3)$ over $\mathbf{b}_2 = (6,3)$. Note that different indices can refer to a same item as here,

$$\mathbf{a}_1 = \mathbf{a}_2$$

Again, by definitions 4.1 and 4.2 and discarding the 0-probable partition $\theta_1 = 0$, the



(a) (b) (c)

Figure 4.3: Bayesian Market Making problem (MM) presented in the case study II (Section 4.2). (a) Prior distribution of two instrument types (therefore, a 2D space). (b) Corresponding posterior given 4 observed data points (trader responses).

following likelihood is derived (see Figure 4.2-iii),

$$p(\mathbf{a}_2 \succ \mathbf{b}_2 \,|\, \boldsymbol{\theta}) = \begin{cases} \eta & \text{if } \theta_1 > 0 \\ 1-\eta & \text{if } \theta_1 < 0 \end{cases} \tag{4.6}$$

With respect to the Bayesian network topology of this model defined by formula 4.4 and by relations 4.5 and 4.6, the posterior distribution of the weight vector (a.k.a. parameter vector) is as follows,

$$\begin{aligned} p\big(\boldsymbol{\theta} \,|\, (\mathbf{a}_1 \succ \mathbf{b}_1), (\mathbf{a}_2 \succ \mathbf{b}_2)\big) &\propto p(\boldsymbol{\theta}) \cdot p(\mathbf{a}_1 \succ \mathbf{b}_1 \,|\, \boldsymbol{\theta}) \cdot p(\mathbf{a}_2 \succ \mathbf{b}_2 \,|\, \boldsymbol{\theta}) \\ &= \begin{cases} 1 & \text{if } -c < \theta_1 < c, -c < \theta_2 < c \\ 0 & \text{otherwise} \end{cases} \\ &\otimes \begin{cases} \eta & \text{if } \theta_2 < \theta_1 \\ 1-\eta & \text{if } \theta_1 < \theta_2 \end{cases} \otimes \begin{cases} \eta & \text{if } 0 < \theta_1 \\ 1-\eta & \text{if } \theta_1 < 0 \end{cases} \\ &= \begin{cases} \eta^2 & \text{if } -c < \theta_2 < \theta_1 < c,\ 0 < \theta_1 \\ \eta(1-\eta) & \text{if } -c < \theta_2 < \theta_1 < c,\ \theta_1 < 0 \\ \eta(1-\eta) & \text{if } -c < \theta_1 < \theta_2 < c,\ 0 < \theta_1 \\ (1-\eta)^2 & \text{if } -c < \theta_1 < \theta_2 < c,\ \theta_1 < 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

which is depicted in Figure 4.2-iv.

4.2 Case study II: Bayesian Market Making (MM)

The second case study deals with the problem of Bayesian trade valuation modeling and *Market making* (MM) which is motivated by [Das and Magdon-Ismail 2008]:

Market making (MM). Suppose there are D different types of *instruments* with respective valuations $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_D)$. There is a *market maker* who at each time step t deals

an instrument of type $\tau_t \in \{1, \ldots, D\}$ by setting *bid* and *ask* b_t and a_t denoting prices at which she is willing to buy and sell each unit respectively (where $b_t \leq a_t$). The "true" valuation of different types are unknown (to her) but any a priori knowledge over their dependencies that can be expressed via a DAG structure over their associated random variables is permitted. Nonetheless, without loss of generality we only consider the following simple dependency: Assume the types indicate different versions of the same product and each new version is more expensive than the older ones ($\theta_i \leq \theta_{i+1}$). The valuation of the oldest version is within some given price range $[L, H]$ and the price difference of any consecutive versions is bound by a known parameter δ:

$$p(\theta_1) = \mathcal{U}(L, H)$$

$$p(\theta_{i+1}) = \mathcal{U}(\theta_i, \theta_i + \delta) \quad \forall i \in \{1, \ldots, D-1\}$$

where $\mathcal{U}(\cdot, \cdot)$ denotes uniform distributions. At each time-step t, a trader arrives. He has a noisy estimation $\widetilde{\theta}_t$ of the actual value of the presented instrument τ_t. We assume $p(\widetilde{\theta}_t | \boldsymbol{\theta}, \tau_t = i) = \mathcal{U}(\theta_i - \epsilon, \theta_i + \epsilon)$. The trader response to bid and ask prices a_t and b_t is d_t in $\{\text{BUY}, \text{SELL}, \text{HOLD}\}$. If he thinks the instrument is undervalued by the ask price (or overvalued by the bid price), with probability 0.8, he buys it (resp. sells it), otherwise holds.

$$p(\text{BUY} \mid \widetilde{\theta}_t, a_t, b_t) = \begin{cases} 0 & \text{if } \widetilde{\theta}_t < a_t \\ 0.8 & \text{if } \widetilde{\theta}_t \geq a_t \end{cases}$$

$$p(\text{SELL} \mid \widetilde{\theta}_t, a_t, b_t) = \begin{cases} 0.8 & \text{if } \widetilde{\theta}_t \leq b_t \\ 0 & \text{if } \widetilde{\theta}_t > b_t \end{cases}$$

$$p(\text{HOLD} \mid \widetilde{\theta}_t, a_t, b_t) = \begin{cases} 1 & \text{if } b_t < \widetilde{\theta}_t < a_t \\ 0.2 & \text{if } \widetilde{\theta}_t \leq b_t \vee \widetilde{\theta}_t \geq a_t \end{cases}$$

Based on traders' responses, the market maker intends to compute the posterior distribution of the valuations of all instrument types. To transform this problem (with corresponding model shown in Figure 4.3a) to the model represented by Equation 4.4, the variables $\widetilde{\theta}_t$ should be marginalized. For instance:

$$\begin{aligned} p(\text{BUY} \mid \boldsymbol{\theta}, a_t, b_t, \tau_t) &= \int_{-\infty}^{\infty} p(\text{BUY} \mid \widetilde{\theta}_t, a_t, b_t) \cdot p(\widetilde{\theta}_t \mid \boldsymbol{\theta}, \tau_t) \, \mathrm{d}\widetilde{\theta}_t \\ &= \int_{-\infty}^{\infty} \begin{cases} 0 & \text{if } \widetilde{\theta}_t < a_t \\ 0.8 & \text{if } \widetilde{\theta}_t \geq a_t \end{cases} \otimes \begin{cases} \frac{1}{2\epsilon} & \text{if } \theta_{\tau_t} - \epsilon \leq \widetilde{\theta}_t \leq \theta_{\tau_t} + \epsilon \\ 0 & \text{if } \widetilde{\theta}_t < \theta_{\tau_t} - \epsilon \vee \widetilde{\theta}_t > \theta_{\tau_t} + \epsilon \end{cases} \mathrm{d}\widetilde{\theta}_t \\ &= \begin{cases} 0 & \text{if } \theta_{\tau_t} \leq a_t - \epsilon \\ 0.4(1 + \frac{\theta_{\tau_t} - a_t}{\epsilon}) & \text{if } a_t - \epsilon < \theta_{\tau_t} \leq a_t + \epsilon \\ 0.8 & \text{if } \theta_{\tau_t} > a_t + \epsilon \end{cases} \end{aligned}$$

The presented case studies illustrated that due to applications of piecewise likelihood functions in the context of Bayesian inference, scalable inference on piecewise models is in demand. This problem is addressed in the subsequent sections.

4.3 Complexity of Inference on Piecewise Models.

The cases studies presented in Sections 4.1 and 4.2 are examples of a more general framework where Bayesian inference is carried out on a model with piecewise likelihood functions. We have already mentioned that this can lead to an exponential blow up in the number of pieces in the posterior. In this short section, the complexity of inference on such piecewise models is studied more formally.

Once again it should be pointed out that the inference method that will be presented in Section 4.4 can be generalized to arbitrary graphical models with piecewise factor in a straight forward manner. However, our focus in this work is on Bayesian networks factorized as in equation 4.4, that is the following standard form:

$$p(\boldsymbol{\theta}|\, d_1, \ldots, d_n) \propto p(\boldsymbol{\theta}, d_1, \ldots, d_n) = p(\boldsymbol{\theta}) \cdot \prod_{j=1}^{n} p(d_j|\boldsymbol{\theta}) \tag{4.7}$$

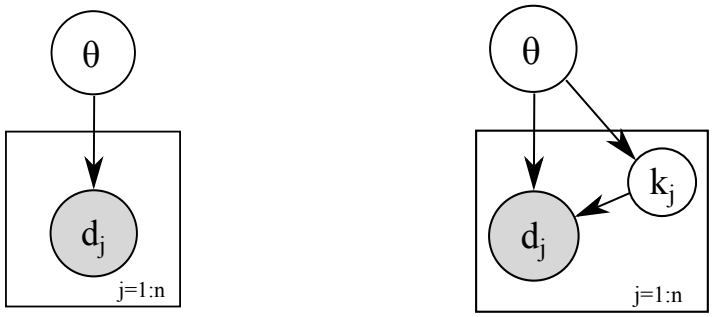
where $\boldsymbol{\theta} := (\theta_1, \ldots, \theta_D)$ is a parameter vector and d_j are observed data points that are independent conditioned on the parameter vector (see Section 2.2.1.1).

If in the model described by Equation 4.7, the prior $p(\boldsymbol{\theta})$ is an L-piece distribution and each of the n likelihoods is a piecewise function with number of partitions bounded by M, then the joint distribution is a piecewise function with number of partitions bounded by LM^n (therefore, $O(M^n)$) since as discussed in the previous chapter, the number of partitions in the product of two piecewise functions is bounded by the product of the number of their partitions. Figure 4.2 represents an example where the number of pieces in the posterior actually grows exponentially in the number of observations.

Clearly, Gibbs sampling in such complicated models is not scalable in the amount of data (i.e. observations) and may become rapidly intractable. Before explaining the proposed solution, it should be pointed out that in the implementation of the code which is utilized in the present chapter it is assumed that the piecewise structures are *linear/quadratic piecewise polynomials*. That is, their associated constraints are restricted to linear or quadratic inequalities while sub-functions are polynomials with real exponents. However, in theory, the algorithm can be applied to *any* family of piecewise models in which the roots of univariate constraint expressions can be found and sub-functions (and their products) are integrable.

4.4 Piecewise Models as Mixture Models

In this section we detail how to overcome the exponential complexity of standard Gibbs sampling by transforming piecewise models to (augmented) mixture models



(a) (b)

Figure 4.4: (a) Plate notation corresponding a Bayesian inference model with parameter (vector) θ and data points d_1 to d_n where the likelihood model is piecewise and defined by equation 4.9. (b) An equivalent mixture model with augmented variable k where $p(k \mid \theta)$ and $p(d \mid k, \theta)$ are defined by equation 4.10.

and performing linear time Gibbs sampling in the latter models. While augmented models to facilitate Gibbs sampling have been proposed previously, e.g., Swendsen-Wang (SW) sampling [Swendsen and Wang 1987] and more recently FlyMC sampling [Maclaurin and Adams 2014], methods like SW are specific to restricted Ising models and FlyMC requires careful problem-specific proposal design and tuning. In this section, we present a generic augmented model for an expressive class of piecewise models and an analytical Gibbs sampler that does not require problem-specific proposal design. Furthermore, our *augmented Gibbs* proposal achieves a novel *exponential-to-linear reduction* in the complexity of sampling from a Bayesian posterior with an exponential number of pieces.

We motivate the algorithm by first introducing the augmented posterior for piecewise likelihoods:

$$p(\theta \mid d_1, \ldots, d_n) \propto p(\theta) \otimes \begin{cases} \boxed{k_1 = 1.} \; f_1^1(\theta) & \text{if } \phi_1^1(\theta) \\ \vdots & \\ \boxed{k_1 = M.} \; f_M^1(\theta) & \text{if } \phi_M^1(\theta) \end{cases} \otimes \cdots \otimes \begin{cases} \boxed{k_n = 1.} \; f_1^n(\theta) & \text{if } \phi_1^n(\theta) \\ \vdots & \\ \boxed{k_n = M.} \; f_M^n(\theta) & \text{if } \phi_M^n(\theta) \end{cases}$$

In the above, k_j is the partition-counter of the j-th likelihood function. ϕ_v^j is its v-th constraint and f_v^j is its associated sub-function. Also for readability, case statements are numbered and without loss of generality, we assume the number of partitions in each likelihood function is M.

We observe that each k_j can be seen as a random variable. It deterministically takes the value of the partition whose associated constraint holds (given θ) and its possible outcomes are in

$$\textsc{Val}(k_j) = \{1, \ldots, M\}$$

Note that for any given θ, exactly one constraint holds for a piecewise function. There-

fore,

$$\sum_{k_j=1}^{M} p(k_j \,|\, \boldsymbol{\theta}) = 1$$

Intuitively, it can be assumed that k_j is the underlying variable that determines which partition of each likelihood function is 'chosen'. As we have:

$$p(d_j|\, \boldsymbol{\theta}) = \sum_{k_j=1}^{M} p(k_j|\, \boldsymbol{\theta}) p(d_j|\, k_j, \boldsymbol{\theta}) \tag{4.8}$$

we can claim that a piecewise likelihood function is a mixture model in which sub-functions f_v^j are the *mixture components* and k_j provide binary *mixture weights*. Hence:

$$\begin{aligned} p(\boldsymbol{\theta}|\, d_1, \ldots, d_n) &\propto p(\boldsymbol{\theta}) \otimes \sum_{k_1} p(k_1|\, \boldsymbol{\theta}) p(d_1|\, k_1, \boldsymbol{\theta}) \\ &\qquad \otimes \cdots \otimes \sum_{k_n} p(k_n|\, \boldsymbol{\theta}) p(d_n|\, k_n, \boldsymbol{\theta}) \\ &\propto \sum_{k_1} \ldots \sum_{k_n} p(\boldsymbol{\theta}, d_1, \ldots, d_n, k_1, \ldots, k_n) \end{aligned}$$

This means that the Bayesian networks in Figures 4.4a and 4.4b are equivalent. Therefore, instead of taking samples from 4.4a, they can be taken from the *augmented model* 4.4b. A key observation, however, is that unlike the conditional distributions $p(\theta_i|\, \boldsymbol{\theta}_{-i})$, in $p(\theta_i|\boldsymbol{\theta}_{-i}, k_1, \ldots, k_n)$ the number of partitions is constant rather than growing as M^n.

The reason is that if $\mathbf{k} = (k_1, \ldots, k_n)$ are given, for the j-th likelihood a single sub-function $f_{k_j}^j$ is 'chosen' and

$$p(\theta_i|\, \boldsymbol{\theta}_{-i}, k_1, \ldots, k_n) \propto p(\theta_i|\, \boldsymbol{\theta}_{-i}) \prod_{j=1}^{n} f_{k_j}^j(\boldsymbol{\theta})$$

Since the sub-functions are not piecewise themselves, the number of partitions in $p(\theta_i|\, \boldsymbol{\theta}_{-i}, k_1, \ldots, k_n)$ is bounded by the number of partitions in the prior.

In the following proposition, we prove Equation 4.8 is valid:

Proposition 3. *The following likelihood function:*

$$p(d|\, \boldsymbol{\theta}) := \begin{cases} f_1(\boldsymbol{\theta}) & \text{if } \phi_1(\boldsymbol{\theta}) \\ \vdots & \\ f_M(\boldsymbol{\theta}) & \text{if } \phi_M(\boldsymbol{\theta}) \end{cases} \tag{4.9}$$

is equivalent to

$$\sum_{k=1}^{M} p(k|\, \boldsymbol{\theta}) \cdot p(d|\, k, \boldsymbol{\theta})$$

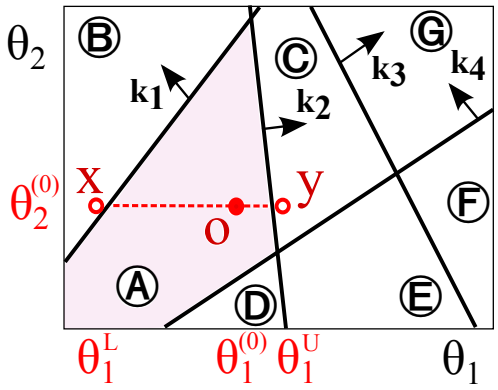


Figure 4.5: A piecewise joint distribution of (θ_1, θ_2) partitioned by bi-valued linear constraints. In the side, specified by each arrow, its associated auxiliary variable k_j is 1 otherwise 2. A Gibbs sampler started from an initial point $O = (\theta_1^{(0)}, \theta_2^{(0)})$, is trapped in an initial partition (A) where $k_1 = k_2 = k_3 = 2$ and $k_4 = 1$.

where:

$$p(k \mid \boldsymbol{\theta}) := \begin{cases} 1 & \text{if } \phi_k(\boldsymbol{\theta}) \\ 0 & \text{if } \neg\phi_k(\boldsymbol{\theta}) \end{cases} \qquad p(d \mid k, \boldsymbol{\theta}) := f_k(\boldsymbol{\theta}) \tag{4.10}$$

Proof. Since constraints ϕ_k are mutually exclusive and jointly exhaustive:[1]

$$\sum_{k=1}^{M} p(k \mid \boldsymbol{\theta}) = \sum_{k=1}^{M} \begin{cases} 1 & \text{if } \phi_k(\boldsymbol{\theta}) \\ 0 & \text{if } \neg\phi_k(\boldsymbol{\theta}) \end{cases} = 1$$

Therefore $p(k \mid \boldsymbol{\theta})$ is a proper probability function. On the other hand, by marginalizing k, (4.10) trivially lead to (4.9):

$$\begin{aligned} \sum_k p(k \mid \boldsymbol{\theta}) \cdot p(d \mid k, \boldsymbol{\theta}) &= \sum_{k=1}^{M} \begin{cases} 1 & \text{if } \phi_k(\boldsymbol{\theta}) \\ 0 & \text{if } \neg\phi_k(\boldsymbol{\theta}) \end{cases} \cdot f_k(\boldsymbol{\theta}), && \text{by (4.10)} \\ &= \sum_{k=1}^{M} \begin{cases} f_k(\boldsymbol{\theta}) & \text{if } \phi_k(\boldsymbol{\theta}) \\ 0 & \text{if } \neg\phi_k(\boldsymbol{\theta}) \end{cases} \\ &= \begin{cases} f_1(\boldsymbol{\theta}) & \text{if } \phi_1(\boldsymbol{\theta}) \\ \vdots & \\ f_M(\boldsymbol{\theta}) & \text{if } \phi_M(\boldsymbol{\theta}) \end{cases} = p(d \mid \boldsymbol{\theta}) && \text{by (4.9)} \end{aligned}$$

in which the third equality holds since constraints ϕ_k are mutually exclusive. □

4.4.1 Deterministic Dependencies and Blocked Sampling

It is known that in the presence of determinism, Gibbs sampling gives poor results [Poon and Domingos 2006]. In our setting, deterministic dependencies arise from the definition of $p(k|\,\boldsymbol{\theta})$ in (4.10), were the value of k is decided by $\boldsymbol{\theta}$. This problem is illustrated in Figure 4.5 by a simple example: A Gibbs sampler started from an initial point:

$$O = (\theta_1^{(0)}, \theta_2^{(0)})$$

is trapped in the initial partition (A). The reason is that conditioned on the initial value of the auxiliary variables, the partition is deterministically decided as being (A), and conditioned on any point in (A), the auxiliary variables keep their initial values.

4.4.1.1 Blocked Gibbs

We avoid deterministic dependencies by Blocked sampling as follows:

At each step of Gibbs sampling, a parameter variable θ_i is jointly sampled with (at least) one auxiliary variable k_j conditioned on the remaining variables:

$$(\theta_i, k_j) \sim p(\theta_i, k_j|\,\boldsymbol{\theta}_{-i}, \mathbf{k}_{-j})$$

This is done in 2 steps:

1. k_j is marginalized out and θ_i is sampled (*collapsed Gibbs sampling*):

$$\theta_i \;\sim\; \sum_{k_j} p(k_j|\,\boldsymbol{\theta}_{-i}, \mathbf{k}_{-j}) \cdot p(\theta_i|\,k_j, \boldsymbol{\theta}_{-i}, \mathbf{k}_{-j})$$

2. The value of k_j is deterministically found given $\boldsymbol{\theta}$:

$$k_j \leftarrow v \in \textsc{Val}(k_j) \text{ s.t. } \phi_v^j(\boldsymbol{\theta}) = \mathit{true}$$

where ϕ_v^j is the v-th constraint of the j-th likelihood function.

For instance in Figure 4.5, if for sampling θ_2, k_1 (or k_2) is collapsed, then the next sample will be in the union of partition (A) and (B) (resp. the union of (A) and (C)). Note that in this example, collapsing k_3 or k_4 would not help because conditioned on $\theta_2 = \theta_2^0$ the borders of partition (A) are determined by k_1 and k_2. More formally, under such a condition the values of auxiliary variables k_3 and k_4 are determined by k_1 and k_2.

4.4.1.2 Targeted Selection of Collapsed Auxiliary Variables

We provide a mechanism for finding auxiliary variables k_j that are not determined by the other auxiliary variables, $\mathbf{k}_{-j}$ when jointly sampled with a parameter variable θ_i.

[1] In case the original piecewise function is partial (i.e. not defined on a subset of its domain), we can assume that its value is 0 over that subset.

We observe that the set of partitions satisfying the current valuation of $\mathbf{k}$ often differs with its adjacent partitions in a single auxiliary variable. Since such a variable is not determined by other variables, it can be used in the blocked sampling.

However, in case some likelihood functions share the same constraint, some adjacent partitions would differ in multiple auxiliary variables. In such cases, more than one auxiliary variable should be used in blocked sampling.

Finding such auxiliary variables has a simple geometric interpretation. As such, we explain it by a simple example depicted in Figure 4.5. Consider finding a proper k_j for the following blocked sampling:

$$p(\theta_1, k_j | \, \theta_2^{(0)}, \mathbf{k}_{-j})$$

It suffices to pass the end points of the (red dotted) line segment where,

$$p(\theta_1 | \, \theta_2^{(0)}, \mathbf{k}) > 0$$

by an extremely small value ε to the following end in points x and y,

$$x := (\theta_1^L, \theta_2^{(0)}) \qquad\qquad y := (\theta_1^U, \theta_2^{(0)})$$

which are located in partitions (B) and (C) respectively.

As it will be shown in Section 4.4.1.3, to guarantee the detailed balance of the algorithm, with probability 0.5 the auxiliary variable that is not shared in (A) and (B) (i.e. k_1) should be collapsed otherwise k_2 i.e. the auxiliary variable not in common between (A) and (C) should be collapsed.

Let the process of finding auxiliary variables not shared with neighboring partitions be discussed in a more formal and general way: In the i-th step of Gibbs sampling i.e. sampling from the i-th parameter θ_i conditioned on the rest $\boldsymbol{\theta}_{-i}$, the values θ_i^L and θ_i^U are defined as follows:

$$\theta_i^L := \inf\{\theta_i \mid \, p(\theta_i | \, \boldsymbol{\theta}_{-i}, \mathbf{k}) > 0\} - \varepsilon$$
$$\theta_i^U := \sup\{\theta_i \mid \, p(\theta_i | \, \boldsymbol{\theta}_{-i}, \mathbf{k}) > 0\} + \varepsilon$$

where $0 < \varepsilon \ll 1$.

In this way, the neighboring partitions (e.g. (B) and (C) in Figure 4.5) can be found. Nonetheless, in order to find the auxiliary variables that differ between the current partition (A) with its neighboring partitions (B) (and (C)) it is sufficient to test the validity of the constraints that define the current partition for θ_1^L (resp. θ_1^U) and pick the one (or ones) that are not satisfied (k_1 (resp. k_2) in Figure 4.5).

4.4.1.3 Convergence to the target distribution

In Section 2.3.2.5 it was mentioned that an MCMC method converges to the desired target distribution if (a) the Markov chain is associated with a single closed communicating class that is, any state is *accessible* from any other state and (b) the detailed

balance holds.

It is well-known that in pathological examples, the first condition does not hold for Gibbs sampling. In the case of the proposed method, such situations can occur more frequently. For instance, consider a piecewise posterior which consists of some positive probability islands. For the simplicity of discussion, also assume that the auxiliary variables are binary and at each Gibbs step, a single auxiliary variable is collapsed (as in Figure 4.5). While the Gibbs sampler can often bridge the 0-probability segments, in our proposed method, states located in different islands are not accessible from each other. The reason is that in the latter method, transitions between partitions take place via direct neighboring partitions. To overcome this problem, in such cases, more than one auxiliary variable should be collapsed at each step of Gibbs. Clearly, in the extreme case where all the auxiliary variables are always collapsed, this method is reduced to the plain Gibbs.

It seems reasonable to believe that detailed balance automatically holds for the proposed method. Nonetheless, since there is a dependency between the collapsed auxiliary variables and the current state of the sampler some concerns may be raised. To remove such doubts, here we formally show that the detailed balance indeed holds.

Again, for the sake of simplicity we assume that the auxiliary variables are bivariate and that at each step of Gibbs, a single auxiliary variable is collapsed since the generalization of the argument is straightforward.

It is sufficient to show that in each i-th step of Gibbs sampling is reversible. That is,

$$p(\theta_i \mid \boldsymbol{\theta}_{-i}) \cdot T(\theta_i \to \theta_i' \mid \boldsymbol{\theta}_{-i}) = p(\theta_i' \mid \boldsymbol{\theta}_{-i}) \cdot T(\theta_i' \to \theta_i \mid \boldsymbol{\theta}_{-i})$$

or equivalently,

$$\begin{aligned} T(\theta_i \to \theta_i' \mid \boldsymbol{\theta}_{-i}) \cdot p(\theta_1, \ldots, \theta_{i-1}, \theta_i, \theta_{i+1}, \ldots, \theta_D) \\ = T(\theta_i' \to \theta_i \mid \boldsymbol{\theta}_{-i}) \cdot p(\theta_1, \ldots, \theta_{i-1}, \theta_i', \theta_{i+1}, \ldots, \theta_D) \end{aligned} \quad (4.11)$$

where D is the dimensionality of the space of parameters and $T(\theta_i \to \theta_i' \mid \boldsymbol{\theta}_{-i})$ is the probability that the i-th parameter is changed from θ_i to θ_i' if the rest of the parameters are fixed. For notational ease let points a and b be defined as follows,

$$a = (\theta_1, \ldots, \theta_{i-1}, \theta_i, \theta_{i+1}, \ldots, \theta_D)$$

$$b = (\theta_1, \ldots, \theta_{i-1}, \theta_i', \theta_{i+1}, \ldots, \theta_D)$$

Using this notation, the detailed balance equation 4.11 can be restated in the standard form:

$$p(a) \cdot T(a \to b) = p(b) \cdot T(b \to a) \quad (4.12)$$

If the points a and b belong to non-neighboring partitions then the probability of transition from one to the other via the proposed method is 0. Therefore, detailed balance holds for this case. We study the case where a and b are in neighboring partitions (say, R and R') which are respectively associated with auxiliary variable vectors $\mathbf{k}$ and $\mathbf{k}'$ which differ in a single variable k_j. The probability of transition from a to b is the

probability that k_j is block sampled (which is 0.5 since with equal probability one end point of the line segment $(\boldsymbol{\theta}_{-i}, \mathbf{k})$ is continued) times the probability that b is sampled from the line segment $(\boldsymbol{\theta}_{-i}, \mathbf{k}_{-j})$:[2]

$$T(a \to b) = 0.5 \cdot p(b \mid \boldsymbol{\theta}_{-i}, \mathbf{k}_{-j}) = 0.5 \cdot \frac{p(b)}{p(\boldsymbol{\theta}_{-i}, \mathbf{k}_{-j})}$$

With a similar reasoning,

$$T(b \to a) = 0.5 \cdot p(a \mid \boldsymbol{\theta}_{-i}, \mathbf{k}_{-j}) = 0.5 \cdot \frac{p(a)}{p(\boldsymbol{\theta}_{-i}, \mathbf{k}_{-j})}$$

These two equations immediately lead to Equation 4.12 which completes the proof of detailed balance for the proposed variation of Gibbs sampling.

4.5 Experimental Results

In this section we show that the mixing time of the proposed method, *augmented Gibbs* sampling, is faster than Rejection sampling, baseline Gibbs and MH. The algorithms are tested against the presented case studies that motivated our discussion. Namely, *Bayesian preference learning* (BPPL) model of Section 4.1 and *Market maker* (MM) model formalized in Section 4.2.

Models are configured as follows: In BPPL, we let $\eta = 0.4$ and the prior parameter distribution is assumed to be uniform in a hypercube of side length 1 centered at $\mathbf{0}$. In MM, we let $L = 0$, $H = 20$, $\epsilon = 2.5$ and $\delta = 10$.

For each combination of the parameter space dimensionality D and the number of observed data n, we generate data points from each model and simulate the associated expected value of ground truth posterior distribution by running rejection sampling on a 4 core, 3.40GHz PC for 15 minutes. Subsequently, using each algorithm, particles $\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)} \ldots$ are generated and the mean of the samples taken till each time point t is computed:

$$\bar{\boldsymbol{\theta}}(t) := \sum_{i=1}^{N(t)} \boldsymbol{\theta}^{(i)}$$

where $N(t)$ is the number of samples taken till time point t.

Based on the generated samples, The *mean absolute error* (MAE) that is the average absolute error between the ground truth mean vector $\boldsymbol{\theta}^*$ and mean of samples taken till each time point, i.e. $\bar{\boldsymbol{\theta}}(t)$, is computed:

$$\text{MAE}(t) = ||\bar{\boldsymbol{\theta}}(t) - \boldsymbol{\theta}^*||_1 = \frac{1}{D} \sum_{i=1}^{D} |\bar{\theta}_i(t) - \theta_i^*|$$

[2]Note that the line segment $(\boldsymbol{\theta}_{-i}, \mathbf{k}_{-j})$ is equivalent to,

$$(\boldsymbol{\theta}_{-i}, \mathbf{k}_{-j}, k_j = 1) \vee (\boldsymbol{\theta}_{-i}, \mathbf{k}_{-j}, k_j = 2)$$

and therefore is spanned on both regions R and R'.

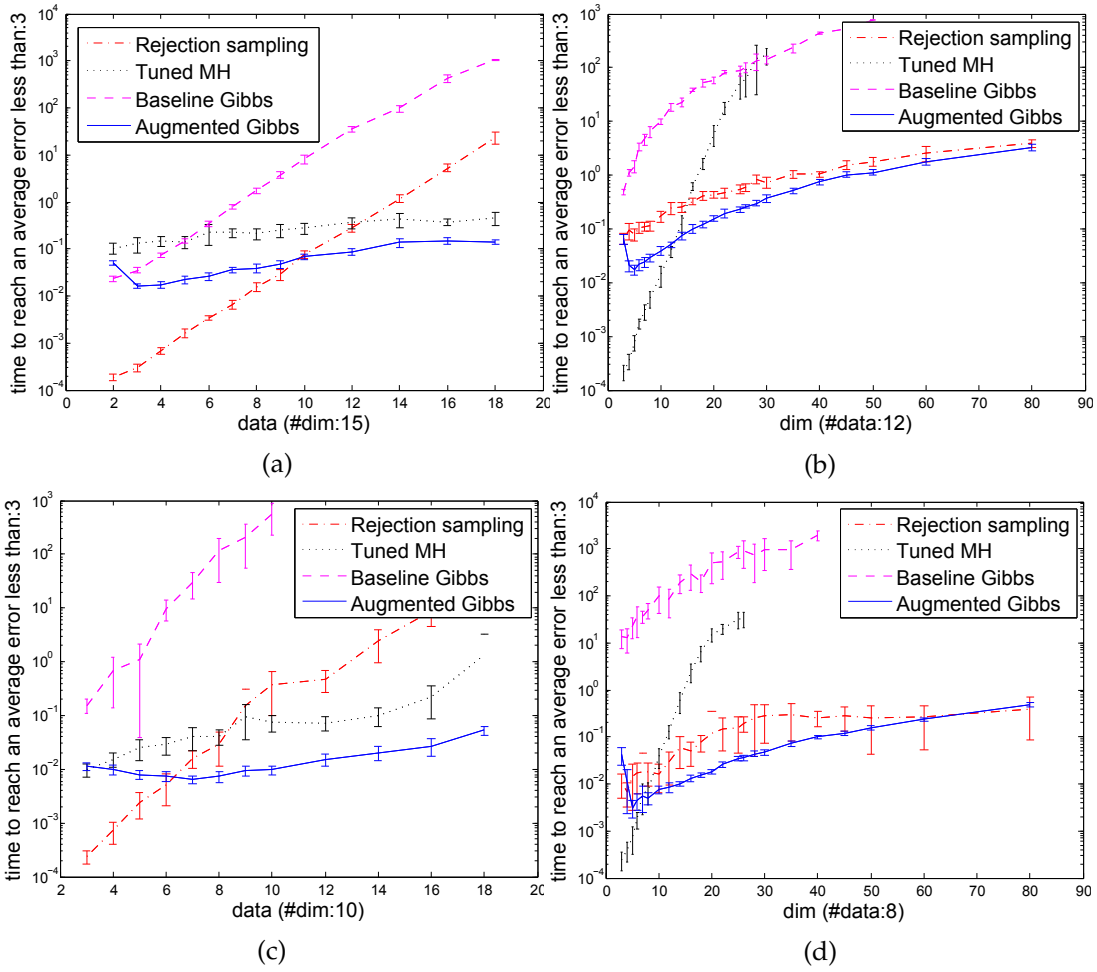


Figure 4.6: Performance of Rejection, MH, baseline and augmented Gibbs (in terms of the time (s) the samplers require to reach a MSE less than 3) on (a) & (b) BPPL and (c) & (d) MM models against different configurations of the number of observed data points and the dimensionality of the parameter space. In (a) & (c), the number of data points varies from 2 to 20 while the dimensionality is fixed (resp. to 15 and 10). In (b) & (d), the dimensionality varies from 2 to 80 while the number of observed data points is fixed (resp. to 12 and 8).

The time till the absolute error reaches the threshold error 3.0 is recorded. For each algorithm, three independent Markov chains are executed and the results are averaged. The whole process is repeated 15 times and the results are averaged and standard errors are computed.

We observe that in both models, the behavior of each algorithm has a particular pattern (Figure 4.6). The speed of rejection sampling and consequently its mixing time deteriorates rapidly as the number of observations increases.The reason is that by observing new data, the posterior density tends to concentrate in smaller areas, leaving most of the space sparse and therefore hard to sample from by rejection sampling.

It is known that the efficiency of MH depends crucially on the tuning of the *proposal*. We carefully tuned MH to reach the optimal acceptance rate of 0.234 [Roberts et al. 1997]. The experimental results show that MH is scalable in observations but its mixing time increases rapidly as the dimensionality increases. These results are rather surprising since we expected that as an MCMC, MH does not suffer from the curse of dimensionality as rejection sampling does. A reason may be that piecewise distributions can be non-smooth or broken (see Figure 4.3) which is far from the characteristics of the Gaussian *proposal density* used in MH.

The efficiency of the baseline Gibbs sampling in particular decreases as data points increase since this leads to an exponential blow-up in the number of partitions in the posterior density. On the other hand, Augmented Gibbs is scalable in both data and dimension for both models. Interestingly, its efficiency even increases as dimensionality increases from 2 to 5 — the reason may be that proportional to the total number of posterior partitions, in lower dimensions the neighbors of each partition are not as numerous. For instance, regardless of the number of observed data in the 2 dimensional BPPL, each partition is neighbored by only two partitions (see Figure 4.5) leading to a slow transfer between partitions. In higher dimensions however, this is often not the case.

4.6 Conclusion

In Bayesian inference with piecewise likelihood models, the number of pieces in the posterior can grow exponentially in the amount of observed data. In this chapter, we tackled this problem by showing that such piecewise models can be transformed into equivalent mixture models via introducing auxiliary discrete random variables each of which is associated with one data point and its value is deterministically decided by a piece in the likelihood function that corresponds to that observation.

By proposing a specific blocked Gibbs sampling mechanism for the generated augmented model, we showed that on such models we can achieve an *exponential-to-linear* reduction in space and time compared to a conventional Gibbs sampler. Unlike rejection sampling and baseline Gibbs sampling, the time complexity of the proposed *augmented Gibbs* method does not grow exponentially with the amount of observed data and yields faster mixing times in high dimensions than Metropolis-Hastings.

In a more general framework, this technique can be directly applied to any Graphical model where the posterior is the product of a large number of piecewise potential functions (regardless of their interpretations) since the augmentation trick prevents the exponential blow up in the number of pieces caused by consecutive piecewise multiplications. As such, a future extensions of this work can examine new application of this work to non-Bayesian inference models. For example, some clustering models can be formalized as piecewise models with latent cluster assignments for each datum – the method proposed here allows linear-time Gibbs sampling in such models. To this end, this work opens up a variety of future possibilities for efficient asymptotically unbiased (Bayesian) inference in expressive piecewise graphical mod-

els that to date have proved intractable or inaccurate for existing (Markov Chain) Monte Carlo inference approaches.

Nonetheless, the limitations of the presented approach should also be recognized and taken into account: An issue is that not all piecewise models can be expressed in a significantly factorized form. On such models, augmented Gibbs sampling may not perform much better than the baseline. The second issue is related to the convergence of the augmented Gibbs: At each step of Gibbs sampling on the augmented model, only a subset of space is sampled and this, as stated in Section 4.4.1.3, may lead to more pathological examples where transitions between non-zero probability regions is impossible and as a result, the Markov chain does not converge to the true target distribution.

As such, the problem of sampling from highly piecewise distributions should not be considered as being completely solved. In the next chapter we study highly piecewise distributions that are not necessarily factorized. On such models we will tackle the problem of efficient Gibbs sampling by another strategy. That is, we will propose increasing the efficiency of Gibbs sampling not by sampling from subsets of the variable space but by increasing the sampling efficiency via reducing the computational redundancies. As we will see, on an expressive class of models, in Gibbs sampling a significant amount of per-sample computations can be avoided.

Chapter 5

Symbolic Gibbs Sampling in Algebraic Graphical Models

In theory, *augmented Gibbs sampler* presented in the previous chapter can be applied to any graphical model. Nonetheless, it reveals its strength in models that are factorized into several piecewise potential functions (via associating an augmented variable to each factor).

However, in models that cannot effectively be factorized into a large number of piecewise potentials, augmented Gibbs does not significantly outperform the baseline. For instance, in the worse case where the model consists of a single piecewise potential function that cannot be factorized, augmented Gibbs diminishes into the baseline.

In this chapter we present another powerful tool based on Gibbs sampling. The design of this approximate inference mechanism is not based on any assumption on the factorization of the joint density functions. Otherwise stated, in this chapter we do not try to increase the performance of Gibbs sampling via sampling from subsets of the variable space. Instead, we try to avoid most costly computations that are traditionally carried out per sample (namely, the computation of the univariate CDF functions that Gibbs sampling rely on) by computing them in closed form prior to the sampling process. In order to be able to do so, we restrict the class of functions that can be used as potential functions. More specifically, we consider *polynomial-piecewise fractional* (PPF) models, that is, models where the borders of partitions are polynomials and the function value in each partition is either a polynomial or a fraction with a polynomial numerator and denominator.

A key insight is that on a large subset of PPFs, most costly computations required for Gibbs sampling can be performed in closed-form and prior to the sampling process rather than per sample. This saving leads to a very fast variation of Gibbs sampling that we refer to as *Symbolic Gibbs* sampling.

5.1 Introduction

Recently, piecewise polynomial representations of distributions in graphical models have attracted attention for their closed-form analytical properties. However, to date, such forms have been quite restricted as evidenced by the piecewise polynomials with

hyperrectangular, hyper-rhombus and linear partitioning constraints that have been used in existing work [Shenoy and West 2011; Shenoy 2012; Sanner and Abbasnejad 2012]. In this chapter we focus on, *polynomial-piecewise fractional* (PPF) functions — a highly expressive family of piecewise algebraic functions that is a superset of the aforementioned families and is significantly larger.

An important characteristic of the aforementioned PPF family is that it remains closed under polynomial and fractional random variable transformations. It is not very hard to show that the later property holds in case the transformation is invertible with respect to at least one random variable (using a Jacobian-based change of variables). In this chapter, we will leverage the properties of Dirac delta and show that the above predicate *often* holds even in the case the transformation is not invertible with respect to any random variable. Being closed under variable transformation is an important property because such transformations are tools that enable inference in the presence of observed constraints among random variables. Probabilistic inference in many real-world problems requires graphical models with deterministic algebraic constraints between random variables. These constraints typically emerge in probabilistic inference applications where instead of direct observation of random variables, functions of them are observed. For instance in a Newtonian mechanical system the prior distribution of mass M and velocity V of a particle might be given. However, these quantities may not be directly observable. Instead, the system's momentum $P = MV$, that is, the product of the random variables might be observed. Endless number of such models can be found in the realm of physical measurements as well as other domains.

Such constraints lead to posterior density functions that are only positive on submanifolds of the space of random variables. Exact inference on the corresponding models is in general impossible and approximate inference is often extremely hard and often ends in unsatisfactory approximation error bounds. A solution is to *collapse* random variables, that is, to reduce the dimensionality of the variable space via random variable transformations. Nonetheless, as we will see shortly, the collapsing process can easily lead to highly piecewise and sophisticated density functions that pose challenges for existing probabilistic inference tools.

The inference mechanism that will be presented in this chapter is an efficient variation of Gibbs sampling that handles approximate inference on a large subset of the aforementioned problematic models. A key insight is that many piecewise algebraic functions permit one analytical integral. This means that the conditional CDFs required for Gibbs sampling can often be computed automatically and in closed form prior to the sampling process rather than per sample. This property immediately leads to a *symbolic Gibbs sampler* that is the main contribution of the present chapter.

In the next section we present a formal representation of the aforementioned momentum model. This model is used as running example throughout the chapter. In Section 5.3 we present our collapsing algorithm for inference in models with deterministic dependencies among random variables. In Section 5.4 we introduce the family or piecewise algebraic fractional (PPF) models and show that they are closed under operations required the collapsing algorithm. In Section 5.5 the *symbolic Gibbs* algo-

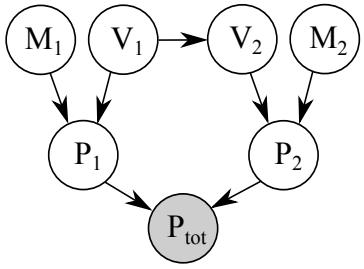


Figure 5.1: Bayesian network for the momentum model. P_1, P_2 and consequently $P_{tot} = P_1 + P_2$ are deterministic functions of M_1, V_1, M_2 and V_2 and the shaded circle indicate that P_{tot} is observed.

rithm is introduced. Experimental results are presented in Section 5.6. These results show that the proposed sampler converges at least an order of magnitude faster than existing Monte Carlo samplers.

5.2 Momentum Model

To motivate the discussion consider the following model which will be used as the running example throughout.

Masses M_1 and M_2 with velocities V_1 and V_2 (and consequently momenta $P_1 = M_1V_1$ and $P_2 = M_2V_2$) collide to form a single mass $(M_1 + M_2)$ with momentum $P_{tot} = P_1 + P_2$ (assuming that there is no dissipation). The masses and velocities are unknown but their prior distributions are given as follows:[1]

$$\begin{aligned} p(M_1) &= \mathcal{U}(0.1,\, 2.1) & p(M_2) &= \mathcal{U}(0.1,\, 2.1) \\ p(V_1) &= \mathcal{U}(-2,\, 2) & p(V_2 \mid V_1) &= \mathcal{U}(-2,\, V_1) \end{aligned} \tag{5.1}$$

The conditional dependencies of the random variables of this model are illustrated in the Bayesian network of Figure 5.1.

Assume that the total momentum of the produced mass M_3 is measured to be 3.0*. Conditioned on this value, we wish to infer the posterior probability of the collided masses and their initial velocities. That is,*

$$p(M_1, M_2, V_1, V_2 \mid P_{tot} = 3)$$

◇

Despite its humble appearance, the probabilistic inference on this model is challenging. Let's take a closer look on the nature of the problem by manually carrying out the computations required for probabilistic reasoning. It can easily be seen that in

[1] $\mathcal{U}(a,\, b)$ denotes a uniform distribution on interval $[a, b]$.

the *momentum model*, the joint density of masses and velocities is

$$p(M_1, M_2, V_1, V_2) = \begin{cases} \frac{1}{16V_1+32} & \text{if } 0.1 < M_1 < 2.1,\ 0.1 < M_2 < 2.1, \\ & \quad -2 < V_1 < 2,\ -2 < V_2 < V_1 \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

which is a 4 dimensional function from which samples can be taken fairly easily.

On the other hand, the total momentum P_{tot} is determined by random variables M_1, M_2, V_1 and V_2:

$$P_{\text{tot}} = M_1 V_1 + M_2 V_2$$

the following *likelihood*:

$$p(P_{\text{tot}} = 3 \mid M_1, V_1, M_2, V_2) = \delta(M_1 V_1 + M_2 V_2 - 3) \tag{5.3}$$

where $\delta(\cdot)$ denotes the *Dirac delta* – a normal distribution with a variance tending to 0. That is,

$$\delta(x) := \lim_{a \to 0} \delta_a(x)$$

where

$$\delta_a(x) := \frac{1}{a\sqrt{\pi}} e^{-x^2/a^2}$$

By *Bayes* rule, the *posterior*,

$$p(M_1, M_2, V_1, V_2 \mid P_{\text{tot}} = 3)$$

is proportional to the product of equations (5.2) and (5.3). Note that despite being 4-dimensional, the mass of this function is only non-zero on a region that tends to the 3 dimensional *hypersurface*:

$$M_1 V_1 + M_2 V_2 = 3$$

In general, observation of an algebraic relationship between random variables of an N-dimensional prior space leads to a posterior mass concentrated on an $(N-1)$-dimensional hyperspace. Due to this dimensionality mismatch, direct MCMC sampling from such models is almost impossible (unless the proposal can be chosen in such a way that the MCMC trajectory follows the manifold). As a tangible example, note that taking samples from a 2D curve via random walk in a 3D continuous space is impossible.

5.3 Stochastic-Deterministic graphical models

To deal with graphical models where some variables are deterministic functions of others, we introduce some new notation and specification, as follows:

We assume that the set of reference random variables $\mathbf{X}$ consists of disjoint sets of *stochastic* and *deterministic (random) variables* which are denoted as $\mathbf{Y}$ and $\mathbf{Z}$ respec-

tively.

$$\mathbf{X} := \mathbf{Y} \cup \mathbf{Z}$$

We also assume that for each deterministic (random) variable $Z_j \in \mathbf{Z}$ there exists exactly one potential Ψ (called a *deterministic potential*) in the form $\delta(G_j(\cdot) - Z_j)$ where δ denotes the Dirac delta and G_j (which we call the *logical value* associated with Z_j) is an expression that does not involve Z_j.[2,3] We further assume that the sets of variables involved in different deterministic potential functions are disjoint.[4]

For instance, in the *momentum model*, P_1 is a deterministic variable associated with deterministic potential $\delta(M_1V_1 - P_1)$ and therefore, the logical value of P_1 is M_1V_1.

We denote the set of all potential functions (i.e. factors that make the prior joint density function) by $\mathbf{\Psi}$.

$$p(\mathbf{X}) \propto \prod_{\Psi \in \mathbf{\Psi}} \Psi_i(\cdot)$$

The set of all deterministic potentials is denoted by $\mathbf{\Psi}^D$. The remaining potentials $\mathbf{\Psi}^S := \mathbf{\Psi} \backslash \mathbf{\Psi}^D$ are called stochastic and do not involve Dirac deltas. Therefore:

$$p(\mathbf{Y}, \mathbf{Z}) \propto \prod_{\Psi \in \mathbf{\Psi}^S} \Psi_i(\cdot) \prod_j \delta\big(G_j(\cdot) - Z_j\big) \tag{5.4}$$

5.3.1 Collapsing determinism (δ-collapsing)

In this section we present an algorithm (called δ-collapsing, throughout) to transform Stochastic-Deterministic graphical models to pure stochastic graphical models with fewer variables. This process is formalized in Algorithm 10.

Using MCMC methods, the variables of the latter graphical model can then be sampled. Provided with these samples, (the particle values associated with) the former graphical model can be reconstructed.

To clarify Algorithm 10, it is run on the momentum model: *In the momentum model, the evidence is the singleton set*

$$\mathcal{E} := \{\langle P_{tot}, 3\rangle\}$$

and the potentials can be indexed as follows:

$$\Psi_1 \text{ to } \Psi_4 \text{ as in Eq. (5.1)} \qquad \Psi_5 := \delta(M_1V_1 - P_1)$$
$$\Psi_6 := \delta(M_2V_2 - P_2) \qquad \Psi_7 := \delta(P_1 + P_2 - P_{tot})$$

The algorithm procedure is as follows:

[2] Circular/recursive definition of deterministic variables are not allowed either.

[3] $\delta\big(G_j(\cdot) - Z_j\big)$ should be thought of as a limit of a normal distribution centered at $G_j(\cdot)$ and a variance that tends to zero. We leave it to the graphical model designer to specify a deterministic potential according to the limiting process that produces the intended distribution. See the literature on the Borel-Kolmogorov paradox [Kolmogorov 1950] for more details.

[4] The reason is that the product of Dirac delta functions can only be defined under narrow circumstances [**?**].

Algorithm 10: δ-COLLAPSING $(\mathcal{E}; \boldsymbol{\Psi})$

Input: $\mathcal{E} := \{\langle E_i, e_i\rangle\}_i$: evidence variables and valuations. $\boldsymbol{\Psi} := \boldsymbol{\Psi}^{\boldsymbol{S}} \cup \boldsymbol{\Psi}^{\boldsymbol{D}}$: potentials where $\boldsymbol{\Psi}^{\boldsymbol{S}}$ are stochastic and $\boldsymbol{\Psi}^{\boldsymbol{D}} := \left\{\Psi_{Z_j} := \delta\big(G_j(\cdot) - Z_j\big) | Z_j \in \mathbf{Z}\right\}$ are deterministic.
Output unnormalized posterior joint of a subset of variables from which, all other variables can be reconstructed.

//STEP 1. *Observed variable instantiation:*
for all $\langle E, e\rangle \in \mathcal{E}$ **do**
  for all $\Psi \in \boldsymbol{\Psi}$ **do** $\Psi \leftarrow \Psi|_{E \leftarrow e}$
end for

//STEP 2. *Marginalizing determinism:*
for all $\Psi_{Z_j} = \delta\big(G_j(\cdot) - Z_j\big) \in \boldsymbol{\Psi}^{\boldsymbol{D}}$ **do**
  forall $\Psi \in \boldsymbol{\Psi}$ **do** $\Psi \leftarrow \Psi|_{Z \leftarrow G_j}$
end for

//STEP 3. *Dimension reduction:*
$t := 0$ // *counter*
$\Phi_t := \prod_{\Psi \in \boldsymbol{\Psi}^S} \Psi$ // Φ_0*: product of stochastic potentials*
for all $\langle Z_j \in \mathbf{Z}, c\rangle \in \mathcal{E}$ **do**
  let Y be a variable involved in G_j such that $\boldsymbol{\Upsilon} := \text{SOLVE}(G_j(\cdot) - c;\, Y) \neq \emptyset$
  (assuming that at least one such variable exists.)

$$\Phi_{t+1} := \sum_{\Upsilon \in \boldsymbol{\Upsilon}} \frac{\Phi_t|_{Y \leftarrow \Upsilon}}{\left|(\partial G_j(\cdot)/\partial Y)|_{Y \leftarrow \Upsilon}\right|} \tag{5.5}$$

  $t \leftarrow t + 1$
end for
Return Φ_t

- **Step 1. Observed variable instantiation.** In all potentials (either deterministic or statistical), all observed stochastic/deterministic variables are instantiated.

Therefore, in the momentum model, Ψ_7 is converted to $\delta(P_1 + P_2 - 3)$.

- **Step 2. Marginalizing determinism.** In all potentials, all deterministic variables Z_j, are substituted by their logical values $G_j(\cdot)$ (so, a potential $\delta(G_j(\cdot) - Z_j)$ itself is replaced by 1.) By the definition of Dirac δ, this simply means that unobserved deterministic variables are marginalized out.

Therefore, in the momentum model, the following conversions take place:

$$\Psi_5 \leftarrow 1 \qquad \Psi_6 \leftarrow 1 \qquad \Psi_7 \leftarrow \delta(M_1V_1 + M_2V_2 - 3)$$

- **Step 3. Dimension reduction.** Beginning with the (unnormalized) joint density of all stochastic variables, for each observed deterministic random variable $Z_j =$

Algorithm 11: SOLVE$(G; X)$

Input: G is a polynomial fraction and X is a variable involved in G.
Output: Set of all solutions (i.e. roots) of G w.r.t. X (i.e. by treating other variables as constants). The roots must be simple and in the form of polynomial fractions. (Note: In case factorization is impossible or some roots are *multiple*, i.e. not distinct, $\emptyset$ is returned.)

c (where c is a constant), $(G_j(\cdot) - c)$ is solved w.r.t. some stochastic variable Y (assuming that it is solvable w.r.t. at least one variable with simple roots in the form of polynomial fractions). Subsequently, by applying Theorem 1, Y is excluded from the joint.

In the momentum model,

$$\Phi_0 = p(M_1, M_2, V_1, V_2) = \begin{cases} \frac{1}{16V_1+32} & \text{if } 0.1 < M_1 < 2.1,\ 0.1 < M_2 < 2.1, \\ & \quad -2 < V_1 < 2,\ -2 < V_2 < V_1 \\ 0 & \text{otherwise} \end{cases}$$

We solve $(G_7 - 3)$*, i.e.*

$$(M_1V_1 + M_2V_2 - 3)$$

with respect to M_1 *(It could alternatively be solved w.r.t.* V_1*,* M_2 *or* V_2*):*

$$\text{SOLVE}(M_1V_1 + M_2V_2 - 3; M_1) = \left\{ \left(\frac{3 - M_2V_2}{V_1} \right) \right\} \tag{5.6}$$

Finally, since

$$\left| \frac{\partial(M_1V_1 + M_2V_2)}{\partial M_1} \right| = |V_1|$$

by (5.5):

$$\Phi_1 := \begin{cases} \frac{1}{V_1(16V_1+32)} & \text{if } 0 < V_1,\ 0.1 < \frac{3-M_2V_2}{V_1} < 2.1,\ 0.1 < M_2 < 2.1, \\ & \quad -2 < V_1 < 2,\ -2 < V_2 < V_1 \\ \frac{-1}{V_1(16V_1+32)} & \text{if } V_1 < 0,\ 0.1 < \frac{3-M_2V_2}{V_1} < 2.1,\ 0.1 < M_2 < 2.1, \\ & \quad -2 < V_1 < 2,\ -2 < V_2 < V_1 \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

which is proportional to $p(M_2, V_1, V_2 \mid P_{tot} = 3)$*.*

5.3.1.1 Proof of correctness

In Algorithm 10, the core operation that requires proof is formula 5.5. The following theorem shows that Dirac deltas can be eliminated with respect to the aforementioned formula.

Theorem 1. *Let,*

$$p(Z=z|x_1,\ldots,x_n) = \delta\big(f(x_1,\ldots,x_n) - z\big)$$

where $f(x_1,\ldots,x_n) - z = 0$ has real and simple roots for x_1 with a non-vanishing continuous derivative

$$\frac{\partial f(x_1,\ldots,x_n)}{\partial x_1}$$

at all those roots. Denote the set of all roots by

$$\mathcal{X}_1 = \{x_1 \mid f(x_1,\ldots,x_n) = z\} \tag{5.8}$$

(Note that each element of $\mathcal{X}_1$ is a function of the remaining variables $x_2,\ldots,x_n,z$.) Then:

$$p(x_2,\ldots,x_n \mid Z=z) \propto \sum_{x_1^i \in \mathcal{X}_1} \frac{p(X_1 = x_1^i, x_2,\ldots,x_n)}{\left|\big(\partial f(x_1,\ldots,x_n)/\partial x_1\big)\big|_{x_1 \leftarrow x_1^i}\right|} \tag{5.9}$$

Proof.

$$\begin{aligned}
p(x_2,\ldots,x_n \mid Z=z) &= \int_{x_1=-\infty}^{\infty} p(x_1,\ldots,x_n \mid Z=z)dx_1 \\
&\propto \int_{x_1=-\infty}^{\infty} p(x_1,\ldots,x_n, Z=z)dx_1 \\
&= \int_{-\infty}^{\infty} p(x_1,\ldots,x_n)p(Z=z \mid x_1,\ldots,x_n)\ \mathrm{d}x_1 \\
&= \int_{-\infty}^{\infty} p(x_1,\ldots,x_n)\delta\big(f(x_1,\ldots,x_n)-z\big)\ \mathrm{d}x_1
\end{aligned} \tag{5.10}$$

According to [Gel'fand and Shilov 1964] there is a unique way to define the composition of Dirac delta with an arbitrary function $h(x)$ such that the formula of change of variables in density functions holds:

$$\delta(h(x)) = \sum_i \frac{\delta(x-r_i)}{|\partial h(x)/\partial x|} \tag{5.11}$$

where r_i are all (real and simple) roots of $h(x)$ and $h(x)$ is continuous and differentiable in the root points. By (5.11), (5.10) and *Tonelli's theorem,*[5]

$$p(x_2,\ldots,x_n \mid Z=z) \propto \sum_{x_1^i \in \mathcal{X}_1} \frac{\int_{-\infty}^{\infty} p(x_1,x_2,\ldots,x_n)\delta(x_1-x_1^i)\ \mathrm{d}x_1}{\left|\big(\partial f(x_1,\ldots,x_n)/\partial x_1\big)\big|_{x_1 \leftarrow x_1^i}\right|}$$

which implies (5.9). □

The end result of Algorithm 10, Φ_t, is proportional to the posterior joint over a subset random variables. Inference can be carried out by sampling from this density.

[5] Tonelli's theorem says that for non-negative functions, sum and integral are interchangeable.

5.3.1.2 Reconstructing eliminated variables

Even though we have eliminated variables, we can still generate samples for them given the sample values of the non-eliminated variables. To do this, the eliminated variables are reconstructed in the reverse order they are eliminated: If in an execution of dimension reduction (Step 3), $G_j(\cdot) - c$ is invertible w.r.t. a variable Y, that is, if

$$\textsc{Solve}(G_j(\cdot) - c;\, Y)$$

is a singleton $\{\Upsilon_1\}$, then the reconstructed sample value of Y is simply $\Upsilon_1(\mathbf{s})$ in which $\mathbf{s}$ denotes the value of the variables that are already sampled or reconstructed.

For instance, in the momentum model, let

$$\boldsymbol{s} = [M_2 = m_2^{(0)},\, V_1 = v_1^{(0)},\, V_2 = v_2^{(0)}]$$

be a sample vector taken from Φ_1. By (5.6), the sample value constructed for M_1 is

$$m_1^{(0)} := \frac{3 - m_2^{(0)} v_2^{(0)}}{v_1^{(0)}}$$

In the general case, if in t-th execution of dimension reduction

$$\textsc{Solve}(G_j(\cdot) - c;\, Y) = \{\Upsilon_1, \ldots, \Upsilon_r\}$$

then we define

$$\phi_\iota := \frac{\Phi_{t-1}|_{Y \leftarrow \Upsilon_\iota}}{\left|(\partial G_j(\cdot)/\partial Y)|_{Y \leftarrow \Upsilon_\iota}\right|} \qquad \text{for } \iota = 1, \ldots, r$$

The constructed sample value of Y is $\Upsilon_\iota(\mathbf{s})$ with a probability proportional to $\phi_\iota(\mathbf{s})$ since by (5.5),

$$\Phi_t = \sum_{\iota=1}^{r} \phi_\iota$$

5.3.2 Notes about the Dirac delta collapsing mechanism

In the subsequent sections of this chapter, we will present an effective MCMC based tool for conducting inference on piecewise continuous density functions such as the models generated via δ-collapsing mechanism. In the remaining part of the current section, however, we point out two important facts about the collapsing mechanism.

- **The relation between δ-collapsing and Jacobian-based random variable transformation.** Dirac delta collapsing mechanism is a generalization of Jacobian-based approach. That is, in a special case where an observation is an invertible function of a variable x_1 (i.e. in equation (5.8), $\mathcal{X}_1$ is singleton), then Theorem 1 can be alternatively proved by a Jacobian-based change of random variable. The Jacobian based proof sketch of Theorem 1 is as follows:

Let $\mathcal{X}_1 := \{x_1^1\}$. Note that by (5.8)

$$f(x_1, \ldots, x_n) = z \iff x_1 = x_1^1,$$

therefore, x_1^1 is the inverse of f (w.r.t. z):

$$x_1^1 = f^{-1}(z, x_2, \ldots, x_n) \tag{5.12}$$

By applying the formula of Jacobian-based change of random variable:

$$p(Z = z, x_2, \ldots, x_n) = p\big(X_1 = f^{-1}(z, \cdot), x_2, \ldots, x_n\big) \cdot \left|\frac{\partial f^{-1}(z, \cdot)}{\partial z}\right| \tag{5.13}$$

$$= p\big(X_1 = f^{-1}(z, \cdot), x_2, \ldots, x_n\big) \cdot \frac{1}{\left|\frac{\partial f\left(f^{-1}(z, \cdot), x_2, \ldots, x_n\right)}{\partial x_1}\right|} \tag{5.14}$$

$$= \frac{p\big(X_1 = x_1^1, x_2, \ldots, x_n\big)}{\left|(\partial f\big(x_1, x_2, \ldots, x_n\big)/\partial x_1)|_{x_1 \leftarrow x_1^1}\right|} \tag{5.15}$$

where in (5.14), *inverse function derivative rule* is applied and in (5.15), (5.12) is substituted. Since $p(x_2, \ldots, x_n \,|\, Z = z)$ is proportional to $p(Z = z, x_2, \ldots, x_n)$, the proof of this special case of Theorem 1 via change of variables is complete.

- **Limitation of δ-collapsing (and Jacobian-based approaches)** As stated in footnote 3 (Section 5.3), the observation of functions of random variables should be considered as the limit of an observation with normal noise where the noise variance tends to zero. This is important because different limiting processes may lead to different results (see Borel-Kolmogorov paradox [Kolmogorov 1950] for details). Otherwise stated, the limiting processes of the possible function observations should be defined in the model. To clarify the discussion consider the following example:

  Let a model include a deterministic potential

  $$\delta(XY - Z) \tag{5.16}$$

  If it is observed that $Z = 5$, then by (5.15)

  $$p(y \,|\, Z = 5) \propto \frac{p(X = 5/y, Y = y)}{\left|(\partial xy/\partial x)|_{x \leftarrow 5/y}\right|} = \frac{p(X = 5/y, Y = y)}{|y|} \tag{5.17}$$

  If one neglects the limiting process of the model and only concentrates on the observation, he may conduct the following erroneous reasoning:

  Since $Z = XY$, the observation indicates that $XY = 5$ which in its turn is equivalent to observing $X - 5/Y = 0$. Now, we "assume" that the model contains a newly introduced

deterministic random variable W associated with the following deterministic potential:

$$\delta(X - 5/Y - W) \tag{5.18}$$

According to the observation, $W = 0$. By utilizing (5.15):

$$p(y \mid W = 0) \propto \frac{p(X = 5/y, Y = y)}{\left|(\partial(x - 5y)/\partial x)\big|_{x \leftarrow 5/y}\right|} = p(X = 5/y, Y = y) \tag{5.19}$$

The above reasoning is clearly erroneous since (5.19) is not equal to (5.17). Similar to Borel-Kolmogorov paradox, the root of this apparently paradoxical result is that different observation noise forms lead to different outcomes and the difference persists in the limit. If the model designer is aware of the nature of measurements involved in the process of observation (using the domain knowledge about the process), the true limiting process can be stated in the model via its corresponding deterministic potential. For instance, if in the above example the measurement actually takes place on the product of X and Y, e.g., if X and Y represent the mass and velocity of an object but the measurement takes place on the momentum of the object, then it makes sense to define the limiting process by (5.16) rather than (5.18). On the other hand, in case the nature of measurement is unknown, the use of δ-collapsing/Jacobian-based approaches may not be justified. In Section 7.2, we will return to this issue where we will briefly mention an alternative solution.

The focus of the succeeding sections of this chapter is on effective inference on models generated via δ-collapsing or similar approaches that lead to piecewise models.

5.4 Polynomial Piecewise Fractionals (PPFs)

Now that we have collapsed out determinism, we can apply Gibbs sampling. Recall from Section 2.3.2.7 that Gibbs sampling from an N-dimensional variable space

$$\mathbf{X} = \{X_1, \ldots, X_N\}$$

takes place in N steps. In each i-th step, X_i is sampled conditioned on the last realization of the others:

$$x_i \sim p(X_i \mid \mathbf{x}_{-i})$$

To perform this task, the following univariate (conditional) *cumulative distribution function* (CDF) should be computed.

$$\text{CDF}(X_i \mid \mathbf{x}_{-i}) \propto \int_{-\infty}^{X_i} p(X_i = t, \mathbf{X}_{-i} = \mathbf{x}_{-i})\, \mathrm{d}t \tag{5.20}$$

The aim of symbolic Gibbs is to compute univariate CDFs that are in the form 5.20,

analytically and prior to sampling and in an automated manner. However, this task requires a univariate symbolic integral and in the general case, this is not always possible. In this section, we address this issue by introducing a rich family of distributions (namely PPFs for *polynomial piecewise fractional* functions) that remain closed under many algebraic operations including all operations required for δ-collapsing. We also show that PPF*, an expressive subset of the PPF family, *have analytical univariate integrals*. This enables an efficient variation of Gibbs sampling that will be introduced in Section 5.5.

5.4.1 Definition of a polynomial piecewise fractional function (PPF)

A PPF is a function in the form $f = \sum_{i=1}^{m} \mathbb{I}[\phi_i] \cdot f_i$ where $\mathbb{I}[\cdot]$ denotes the indicator function. Using expanded notation,

$$f = \begin{cases} f_1 & \text{if } \phi_1 \\ \vdots & \\ f_m & \text{if } \phi_m \end{cases} = \begin{cases} \frac{N_1}{D_1} & \text{if } \varphi_{1,1} \lesseqgtr 0,\ \varphi_{1,2} \lesseqgtr 0,\ \dots \\ \vdots & \\ \frac{N_m}{D_m} & \text{if } \varphi_{m,1} \lesseqgtr 0,\ \varphi_{m,2} \lesseqgtr 0,\ \dots \end{cases} \tag{5.21}$$

where each *sub-function* $f_i := \frac{N_i}{D_i}$ is a (multivariate) polynomial fraction and *conditions* ϕ_i partition the space of function variables. Each ϕ_i is a conjunctions of some inequalities ($\lesseqgtr$ stands for $>$ or $<$)[6] where each *atomic constraint* $\varphi_{i,j}$ is a polynomial.

Evidently, this class is very expressive. Other distributions can also be approximated by this form via *Taylor series* expansion or other existing approximation tools [Shenoy and West 2011].

5.4.2 Some properties of the PPF family

PPFs are closed under elementary operations, e.g.:

$$\begin{cases} f_1 & \text{if } \phi_1 \\ f_2 & \text{if } \phi_2 \end{cases} \times \begin{cases} g_1 & \text{if } \psi_1 \\ g_2 & \text{if } \psi_n \end{cases} = \begin{cases} f_1 \times g_1 & \text{if } \phi_1, \psi_1 \\ f_1 \times g_2 & \text{if } \phi_1, \psi_2 \\ f_2 \times g_1 & \text{if } \phi_2, \psi_1 \\ f_2 \times g_2 & \text{if } \phi_2, \psi_2 \end{cases}$$

PPFs are closed under polynomial fractional substitution.

$$f|_{x \leftarrow \frac{F}{G}} = \begin{cases} f_1|_{x \leftarrow \frac{F}{G}} & \text{if } \phi_1|_{x \leftarrow \frac{F}{G}} \\ \vdots & \\ f_m|_{x \leftarrow \frac{F}{G}} & \text{if } \phi_m|_{x \leftarrow \frac{F}{G}} \end{cases} \tag{5.22}$$

[6] We do not define the value of piecewise density functions on their partitioning hyperplanes and do not allow $\delta(\cdot)$ potentials have roots on the partitions.

The reason is that in the r.h.s of (5.22), sub-functions $f_i|_{x \leftarrow \frac{F}{G}}$ are polynomial fractions (PFs) (since PFs are closed under PF-substitution). Conditions $\phi_i|_{x \leftarrow \frac{F}{G}}$ are not polynomial (they are PF) but can be restated as (multiple) case-statements with polynomial conditions. E.g.:

$$\left(\begin{cases} f_1 & \text{if } \frac{H_1}{H_2} > 0 \\ \vdots \end{cases}\right) = \begin{cases} f_1 & \text{if } H_1 > 0, H_2 > 0 \\ f_1 & \text{if } H_1 < 0, H_2 < 0 \\ \ldots \end{cases}$$

PPFs are also closed under *absolute value* and other similar piecewise functions, e.g.:

$$\left|\left(\begin{cases} \frac{N_1}{D_1} & \text{if } \phi_1 \\ \vdots \end{cases}\right)\right| = \begin{cases} \frac{N_1}{D_1} & \text{if } N_1 > 0, D_1 > 0, \phi_1 \\ \frac{N_1}{D_1} & \text{if } N_1 < 0, D_1 < 0, \phi_1 \\ \frac{-N_1}{D_1} & \text{if } N_1 > 0, D_1 < 0, \phi_1 \\ \frac{-N_1}{D_1} & \text{if } N_1 < 0, D_1 > 0, \phi_1 \\ \ldots \end{cases}$$

Therefore, PPFs are closed under collapsing determinism.

5.4.3 Analytic integration

In general, PPFs are not closed under integration; however, a large subset of them have closed-form single variable integrals. We focus on the following fairly expressive subset of PPFs:

Definition 1. *A PPF* is a PPF in which:*

1. *Atomic constraints $\varphi_{i,j}$ are factorized into terms where the maximum degree of each variable is at most 2.*
2. *The denominator of each sub-function can be factorized into polynomials in which the maximum degree of each variable is at most 2.*

Here is an example of a PPF* case-statement:

$$\frac{x^2y^3 + 7xz + 10}{(5xy^2 + 2)(y + x)^3} \quad \text{if } (y^2 + z^2 - 1)(x^2 + 2xy) > 0 \tag{5.23}$$

5.4.3.1 Analytic univariate PPF* integration

Now we provide a procedure to perform integration on PPF* functions.

It can be shown that if in a PPF* all variables except one are instantiated, the resulting univariate function has a closed form integral. To perform exact Gibbs sampling, this is sufficient because in each step of Gibbs sampling only one variable is uninstantiated.

However, we want to go a step further and compute univariate integrals of multivariate piecewise functions *without* instantiating the remaining variables.

This may look impossible since in the latter case, the integration bounds depend on the values of uninstantiated conditions. But as the following procedure shows, it is indeed possible for the PPF* class:

Suppose $\int_{\alpha}^{\beta} f \, \mathrm{d}x$ is intended where f is a PPF*.

1. *(Partitioning).* The integral of the piecewise function f is the summation of its case statement integrals:

$$\int \sum_{i=1}^{m} \mathbb{I}[\phi_i] \cdot f_i \, \mathrm{d}x = \sum_{i=1}^{m} \int \mathbb{I}[\phi_i] \cdot f_i \, \mathrm{d}x$$

   Therefore we only need to show that a single PPF* case-statement is integrable.

2. *(Canonicalization).* A PPF* case statement can be restated in the form of multiple case statements in which the degree of each variable in each atomic constraint is at most 2. For instance, (5.23) can be restated as:

$$\begin{cases} \frac{x^2y^3+7xz+10}{(5xy^2+2)(y+x)^3} & \text{if } (y^2+z^2-1) > 0, (x^2+2xy) > 0 \\ \frac{x^2y^3+7xz+10}{(5xy^2+2)(y+x)^3} & \text{if } (y^2+z^2-1) < 0, (x^2+2xy) < 0 \end{cases} \tag{5.24}$$

3. *(Condition solution).* For the integration variable x, a PPF* case statement can be transformed into a piecewise structure with atomic constraints in the form $x > L_i$ or $x < U_i$ or $I_i > 0$, where L_i, U_i and I_i are algebraic expressions (not necessarily polynomials) that do not involve x.

   For instance, if expressions A, B and C do not involve x, the case statement (5.25) is replaced by case statements (5.26).

$$f_1 \quad \text{if } (A \cdot x^2 + B \cdot x + C) > 0 \tag{5.25}$$

$$\begin{cases} f_1 & \text{if } (A > 0), \, (x > \frac{-B+\sqrt{B^2-4AC}}{2A}) \\ f_1 & \text{if } (A > 0), \, (x < \frac{-B-\sqrt{B^2-4AC}}{2A}) \\ f_1 & \text{if } (A < 0), \, (x > \frac{-B-\sqrt{B^2-4AC}}{2A}), (x < \frac{-B+\sqrt{B^2-4AC}}{2A}) \end{cases} \tag{5.26}$$

   3. (Bounding). The bounded integral of a case statement associated with $\{L_i\}_i$, $\{U_i\}_i$ and $\{I_i\}_i$ is itself a case-statement with the same independent constraints, lower bound LB =max$\{\alpha, L_i\}$ and upper bound UB =min$\{\beta, U_i\}$. For instance consider the following example where the lower bound, upper bound and inde-

pendent constraints are colored blue, green and red, respectively:

$$\int_{\alpha}^{\beta}\Big[x^3+xy \quad \text{if } (x>\mathbf{3}),(x>\mathbf{y}+\mathbf{1}),(x<\mathbf{y}^2-\mathbf{7}),(-\mathbf{y}/\mathbf{z}>\mathbf{1}),(\mathbf{y}>\mathbf{0})\Big]\,\mathrm{d}x =$$
$$\Big[\int_{\max\{\alpha,\mathbf{3},\mathbf{y}+\mathbf{1}\}}^{\min\{\beta,\mathbf{y}^2-\mathbf{7}\}} x^3+xy\,\mathrm{d}x\Big] \quad \text{if } (-\frac{\mathbf{y}}{\mathbf{z}}>\mathbf{1})\wedge(\mathbf{y}>\mathbf{0})$$

4. (sub-function integration). What remains is to compute infinite integral of sub-functions. The restrictions imposed on PPF* sub-functions guarantee that they have closed-form univariate integrals. These integrals are computed by performing polynomial division (in case the degree of x in the sub-function's numerator is more than its denominator), followed by partial fraction decomposition and finally, using a short list of indefinite integration rules.

For instance, to compute the following integral,

$$\int \frac{x^4+x^3+x^2+1}{x^2+x-2}\,\mathrm{d}x$$

by polynomial division we get,

$$\frac{x^4+x^3+x^2+1}{x^2+x-2} = x^2+3+\frac{-3x+7}{(x+2)(x-1)}$$

and by partial fractional decomposition we end in

$$x^2+3+\frac{-3x+7}{(x+2)(x-1)} = x^2+3-(13/3)\frac{1}{x+2}+(4/3)\frac{1}{x-1}$$

Therefore, by the rule

$$\int \frac{1}{x+a}\,\mathrm{d}x = \ln|x+a|+C$$

the integration is as follows:

$$\int \frac{x^4+x^3+x^2+1}{x^2+x-2}\,\mathrm{d}x = (1/3)x^3+3x-(13/3)\ln|x+2|+(4/3)\ln|x-1|+C$$

5.5 Symbolic Gibbs Sampling

Our *symbolic Gibbs sampling* is based on a simple but significantly useful insight: If $p(X_1,\ldots,X_N)$ has analytic integrals w.r.t. all variables X_i (as is the case with PPF* densities), then the costly CDF computations can be done *prior to the sampling process rather than per sample*. It is sufficient to construct a mapping $\mathcal{F}$ from variables X_i to

their corresponding (unnormalized conditional) analytical CDFs.

$$\mathcal{F}\colon \{X_1, \dots X_N\} \to (\mathbb{R}^N \to \mathbb{R}^+ \cup \{0\})$$
$$X_i \mapsto \int_{-\infty}^{X_i} p(X_i = t, \mathbf{X}_{-i})\, \mathrm{d}t \tag{5.27}$$

Note that the difference between (5.20) and (5.27) is that in the former, all variables except X_i are already instantiated therefore $\mathrm{CDF}(X_i \,|\, \mathbf{x}_{-i})$ is a univariate function but $\mathcal{F}$ is N-variate since variables $\mathbf{X}_{-i}$ are kept uninstantiated and symbolic. Provided with such a map, in the actual sampling process, to sample $x_i \sim p(X_i \,|\, \mathbf{x}_{-i})$, it is sufficient to instantiate the analytical CDF associated to X_i with $\mathbf{x}_{-i}$ to obtain the appropriate univariate conditional CDF (see Algorithm 12). This reduces the number of CDF computations from $N \cdot T$ to N where T is the number of taken samples.

If CDF inversion (required for inverse transform sampling) is also computed analytically, then Gibbs sampling may be done fully analytically. However, analytical inversion of PPF*s can be very complicated and instead in the current implementation, we approximate the CDF^{-1} computation via *binary search*. This requires several function evaluations per sample. Nonetheless, (unlike integration), function evaluation is a very fast operation. Therefore, this suffices for highly efficient Gibbs sampling as we will show experimentally in the next section.

Algorithm 12: SYMBOLICGIBBS$(\mathbf{X}, \mathbf{x}^{(0)}, \mathcal{F}, T)$

Input: random variables $\mathbf{X} := \langle X_1, \dots, X_N \rangle$; initialization $\mathbf{x}^{(0)} := \langle x_1^{(0)}, \dots, x_N^{(0)} \rangle$; a mapping $\mathcal{F}$ from $\mathbf{X}$ to analytical conditional CDFs; desired number of output samples T.
Output: a sequence of samples.
for $t = 1, 2, \dots T$ **do**
  $\langle x_1^{(t)}, \dots, x_N^{(t)} \rangle \leftarrow \langle x_1^{(t-1)}, \dots, x_N^{(t-1)} \rangle$
  for each $X_i \in \mathbf{X}$ **do**
    $F(\mathbf{X}) \leftarrow \mathcal{F}(X_i)$ *//analytic conditional CDF w.r.t.* X_i
    $\mathrm{CDF}(X_i) \leftarrow F(x_1^{(t)}, \dots, x_{i-1}^{(t)}, X_i, x_{i+1}^{(t)}, \dots, x_N^{(t)})$
    //Inverse transform sampling:
    $u \sim \mathcal{U}(0, \mathrm{Cdf}(\infty))$
    $x_i^{(t)} \leftarrow \mathrm{CDF}^{-1}(u)$
  end for
end for
Return $\big\langle \langle x_1^{(1)}, \dots, x_N^{(1)} \rangle, \dots, \langle x_1^{(T)}, \dots, x_N^{(T)} \rangle \big\rangle$;

5.6 Experimental Results

Our experiments are designed to study the following questions:

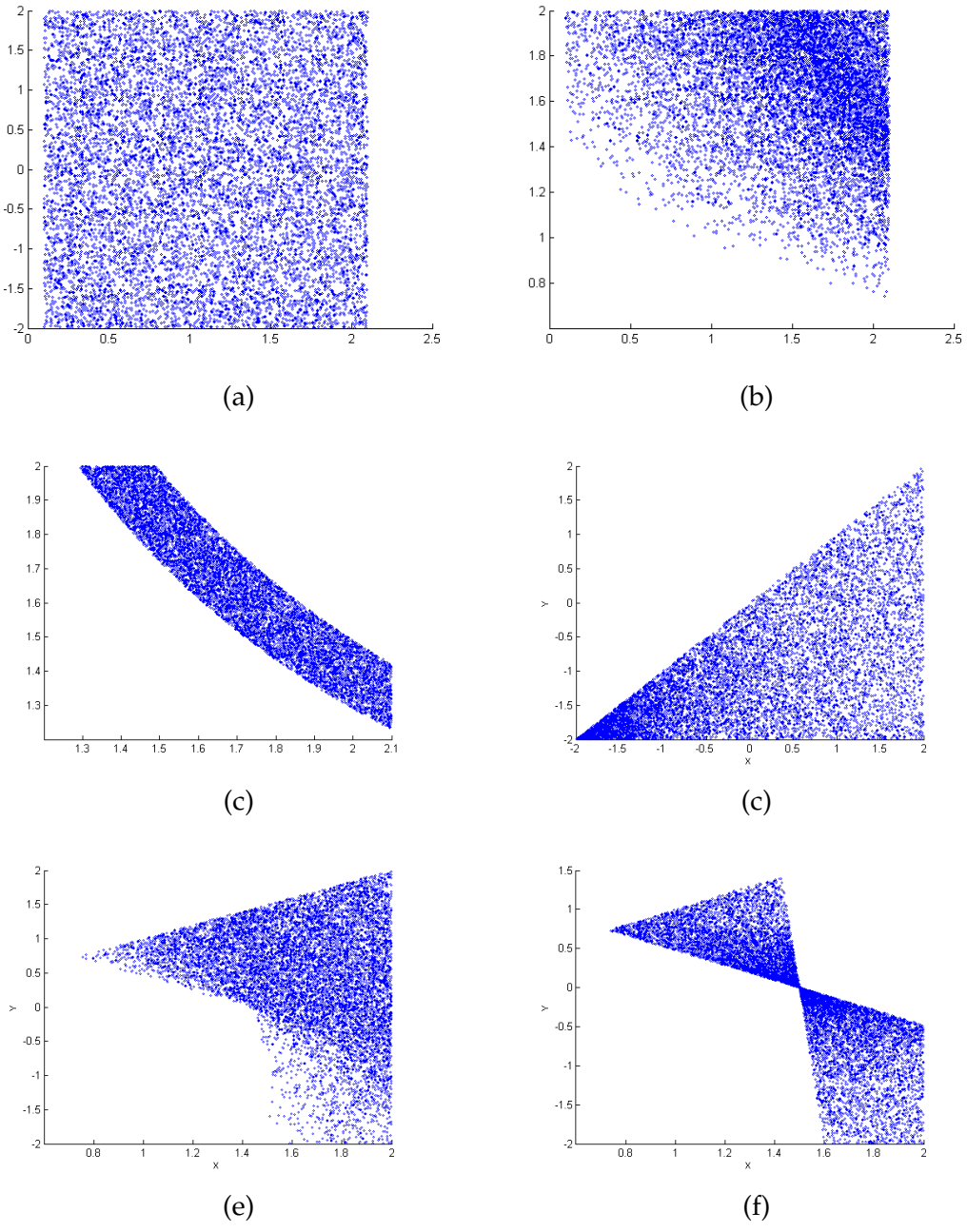
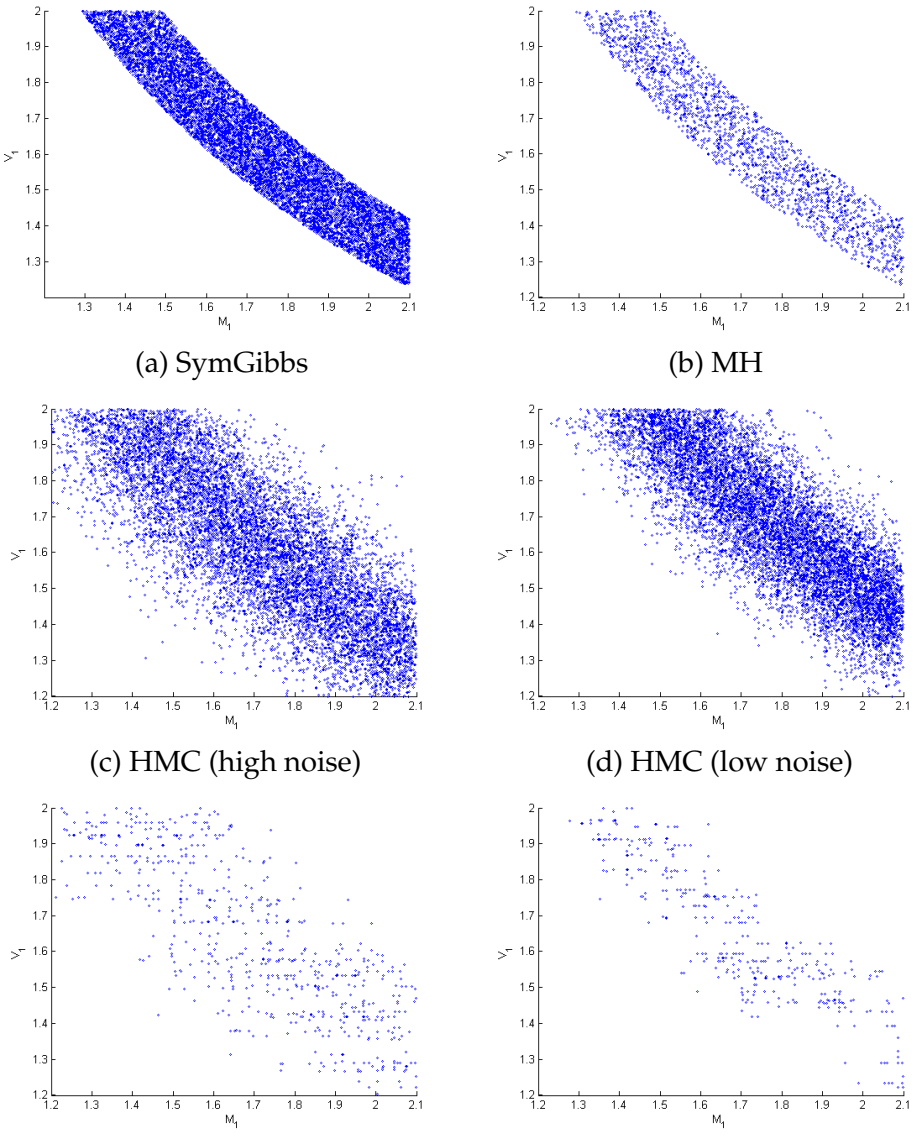


Figure 5.2: Prior/posterior joint distributions of pairs of random variables in the *momentum model*. (a) $p(M_1, V_1)$, (b) $p(M_1, V_1 \,|\, P_{\text{tot}} = 3)$, (c) $p(M_1, V_1 \,|\, P_{\text{tot}} = 3, V_2 = 0.2)$, (d) $p(V_1, V_2)$, (e) $p(V_1, V_2 \,|\, P_{\text{tot}} = 3)$, (f) $p(V_1, V_2 \,|\, M_1 = 2, P_{\text{tot}} = 3)$ using rejection sampling on the δ-collapsed model.



(a) SymGibbs (b) MH

(c) HMC (high noise) (d) HMC (low noise)

(e) SMC (high noise) (f) SMC (low noise)

Figure 5.3: 10000 samples taken from the distribution of Figure (5.2-c) using (a) Symbolic Gibbs sampler and(b) MH with *proposal variance* 0.8 on the reduced-dimension model as well as (c) Hamiltonian Monte Carlo (HMC) with a measurement error variance 0.2, , (d) and 0.01 as well as Anglican implementation of SMC algorithm with parameters (e) $\sigma^2_{V_2} = 0.01$, $\sigma^2_{P_{\text{tot}}} = 0.2$ and (f) $\sigma^2_{V_2} = 0.01$, $\sigma^2_{P_{\text{tot}}} = 0.1$ on the *approximated-by-noise* model.

Table 5.1: Parameters corresponding the following experimental models: asymmetric momentum model(AMM), symmetric multi-object momentum model(SMM), building wiring model (BWM).

#	Experiment	τ	HMC	SMC	Evidence
1	AMM	-	$\sigma^2_{P_t} = 0.2$ (in Fig. 5.3-c)	$\sigma^2_{V_2} = 0.01$, $\sigma^2_{P_t} = 0.2$ (in Fig. 5.3-e)	$P_t = 3$
			$\sigma^2_{P_t} = 0.00001$ (in Fig. 5.3-d)	$\sigma^2_{V_2} = 0.01$, $\sigma^2_{P_t} = 0.1$ (in Fig. 5.3-f)	$V_2 = 0.2$
2	SMM	0.3	$\sigma^2_{P_t} = 0.05$	$\sigma^2_{P_t} = 0.1$	$P_t = 1.5n$
3	BWM	0.045	$\sigma^2_G = 0.02$	$\sigma^2_G = 0.07$	$G = n/10.17$

(a) **Posterior quality.** For our rich class of piecewise algebraic graphical models with nonlinear deterministic constraints, what general qualities of posteriors arise in terms of their modality, continuity, shape of support, etc.?

(b) **MCMC comparison.** How does Symbolic Gibbs perform compared to other MCMC methods on such posteriors in terms of sampling quality and the convergence rate to the correct steady-state?

(c) **Handling determinism.** How is the sampling performance affected if instead of δ-collapsing, the deterministic constraints are relaxed by noise as done in previous work?

The compared MCMC algorithms, the utilized measurements and the experimental models are introduced in Sections 5.6.1, 5.6.2 and 5.6.3 respectively. The experimental evaluations are discussed in Section 5.6.4.

5.6.1 Algorithms

We compare the proposed *symbolic Gibbs sampler* (SymGibbs) to *baseline Gibbs* (BaseGibbs) [Pearl 1987], *rejection sampling* (Rej) [Hammersley and Handscomb 1964], *tuned Metropolis-Hastings* (MH) [Roberts et al. 1997], *Hamiltonian Monte Carlo* (HMC) using Stan probabilistic programming language [Stan Development Team 2014] and *Sequential Monte Carlo* (SMC) using Anglican probabilistic programming language [Wood et al. 2014].[7]

SymGibbs, BaseGibbs and MH are run on *δ-collapsed* models while in the case of HMC and SMC, determinism is softened by observation noise. It should be mentioned that the state-of-the-art probabilistic programming languages, disallow deterministic

[7]We also tested the other algorithm implemented by Anglican, namely *Particle-Gibbs* (PGibbs) (a variation of Particle-MCMC[Andrieu et al. 2010]) and *random database* (RDB) (an MH-based algorithm introduced in [Wingate et al. 2011]) (see [Wood et al. 2014]). In our experimental models, the performance of these algorithms is very similar to (SMC). Therefore, for the readability of the plots, we did not depict them.

relationships among continuous random variables be observed.[8] The solution that these off-the-shelf inference frameworks often suggest (or impose) is to approximate the observed determinism via adding noise to the observation [Patil et al. 2010].[9]

SymGibbs and BaseGibbs require no tuning. MH is automatically tuned after [Roberts et al. 1997] by testing 200 equidistant proposal variances in interval $(0, 0.1]$ and accepting a variance for which the acceptance rate closer to 0.24.

To soften the determinism in HMC and SMC, the observation of a deterministic variable Z is approximated by observation of a newly introduced variable with a Gaussian prior centered at Z and with noise variance (parameter) σ_Z^2. Anglican's syntax requires adding noise to all observed variables. Therefore, in the case of SMC, stochastic observations are also associated with noise parameters. All used parameters are summarized in Table 5.1. SymGibbs, BaseGibbs, Rej and MH have single thread java implementations. The number of threads and other unspecified parameters of Stan and Anglican are their default settings. All algorithms run on a 4 core, 3.40GHz PC.

5.6.2 Measurements

In each experiment, all non-observed stochastic random variables of the model form the query vector $\mathbf{Q} = [Q_1, \ldots, Q_\zeta]$. The number of samples taken by a Markov chain Γ up to a time t is denoted by n_Γ^t and the samples are denoted by $\mathbf{q}_\Gamma^{(1)}, \ldots, \mathbf{q}_\Gamma^{(n_\Gamma^t)}$ where $\mathbf{q}_\Gamma^{(i)} := [q_{1,\Gamma}^{(i)}, \ldots, q_{\zeta,\Gamma}^{(i)}]$

1. Mean absolute error vs time

The main measurement is to compute the *mean absolute error* (MAE)

$$\mathrm{MAE}_\Gamma(t) := \frac{1}{\zeta} \sum_{j=1}^{\zeta} \left| \frac{1}{n_\Gamma^t} \sum_{i=1}^{n_\Gamma^t} q_{j,\Gamma}^{(i)} - q_j^* \right| \tag{5.28}$$

vs (wall-clock) time t where $\mathbf{q}^* := [q_1^*, \ldots q_\zeta^*]$ is the ground truth mean query vector (that will be discussed shortly).

In each experiment and for each algorithm, $\gamma = 15$ Markov chains are run, and for each time point t, average and standard error of $\mathrm{MAE}_1(t)$ to $\mathrm{MAE}_\gamma(t)$ are plotted.

2. Time to reach error threshold (τ)

This is a single plot that summarizes the overall behavior of each sampler algorithm w.r.t. the model size: We notice that after some error threshold, the comparative per-

[8] In BUGS [Lunn et al. 2009], *logical nodes* cannot be given data or initial values . In PyMC [Patil et al. 2010] deterministic variables have no *observed flag*. In Stan [Stan Development Team 2014] if you try to assign an observation value to a deterministic variable, you will encounter an error message: "attempt to assign variable in wrong block" while Anglican [Wood et al. 2014] throws error "invalid-observe", etc.

[9] For example in the momentum model, the observation $P_{\text{tot}} = 3$ would be approximated with a normal distribution $\mathcal{N}(P_{\text{tot}} - 3, \sigma_\eta^2)$ where the variance σ_η^2 is the noise parameter.

formance of the samplers remains fixed. Therefore, for each sampler, the time to reach a fixed threshold τ is plotted vs the model size.

If we pick the error threshold too low, many samplers will never reach it. If it is too high, samplers pass it prior to being comparatively stable. Per experiment, we manually choose an error threshold that provides a fair compromise between these two requirements (see Table 5.1).

Ground truth (of mean query vector) $(\mathbf{q}^*)$

In the first experimental model (asymmetric momentum model) the ground truth is approximated by taking 200,000 samples via *rejection sampling* (Rej) (as a completely unbiased sampler). However, this approximation cannot be used in high-dimensional models since the performance of Rej drastically deteriorates with increasing dimension. Our remaining experimental models are intentionally chosen to be symmetric as this enables us to compute the ground truth manually.

5.6.3 Experimental models

Three experimental models are provided: (1) Asymmetric momentum model, (2) symmetric multi-object momentum model and (3) Building wiring model.

(Asymmetric) momentum model.

The first model is our running example introduced in Section 5.2 (*momentum model*). Using this model, the following experiments are carried out:

1. To study the posterior quality, for different evidence and queries, joint posteriors are approximated by 10000 samples taken via rejection sampling and plotted in Figure 5.2.

2. To compare sampling methods (qualitatively), in Figure 5.3, the experiment is repeated on the posterior of Figure 5.2-c using the sampling algorithms described in Section 5.6.1

3. For qualitative MCMC comparison, *absolute error vs time* is plotted in Figure 5.4).

With increasing the number of space variables, the performance of different sampling algorithms significantly varies. Therefore, in the remaining experiments, comparisons take place w.r.t. the variable space dimensionality.

Symmetric (multi-object) momentum model.

Consider a variation of the momentum model in which n objects collide. Let all V_i and M_i share a same uniform prior,

$$p(V_i) = p(M_i) = U(0.2, 2.2) \qquad \text{for } i = 1, \ldots, n$$

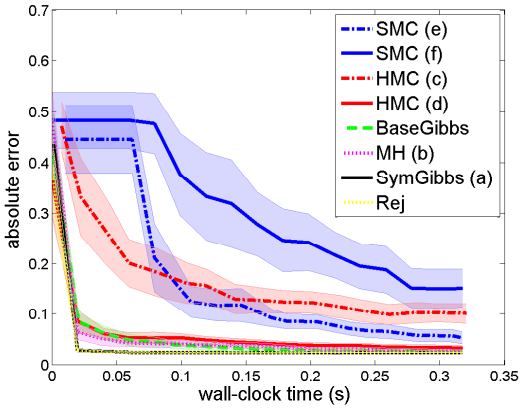


Figure 5.4: Absolute error vs time in the *asymmetric momentum model*. (The algorithms correspond sub-figures of Figure 5.3.)

and the constraint be

$$\sum_{i=1}^{n} M_i V_i = P_{\text{tot}}$$

The symmetry enables us to compute the posterior ground truth means values manually:

$$M^* = V^* = \sqrt{P_{\text{tot}}/n} \tag{5.29}$$

Conditioned on $P_{\text{tot}} = 1.5n$, all masses M_i and velocities V_i are queried. By (5.29), all elements of the ground truth vector $\mathbf{q}^*$ are $\sqrt{1.5}$.

MAE vs. time is depicted in Figures 5.5.a & b for a 10 dimensional and a 30 dimensional model, respectively. Time to reach error threshold $\tau = 0.3$ is plotted in Figure 5.5.c.

The next model deals with a more complicated observed determinism.

Building wiring model

An electrical circuit composed of n, $10\Omega \pm 5\%$ parallel resistor elements R_i with priors

$$p(R_i) = U(9.5,\ 10.5) \qquad \text{for } i = 1, \ldots, n$$

The resistors are inaccessible i.e. the voltage drop and the current associated with them cannot be measured directly. Given the source voltage V and the total input current I, the posterior distribution of the element resistances are required. Here the deterministic constraint is:

$$\frac{1}{R_1} + \ldots + \frac{1}{R_n} = c \tag{5.30}$$

where $c = \frac{I}{V}$.

Equations of the form (5.30) are generally referred to as *reduced mass* relationships and have applications in the electrical, thermal, hydraulic and mechanical engineering

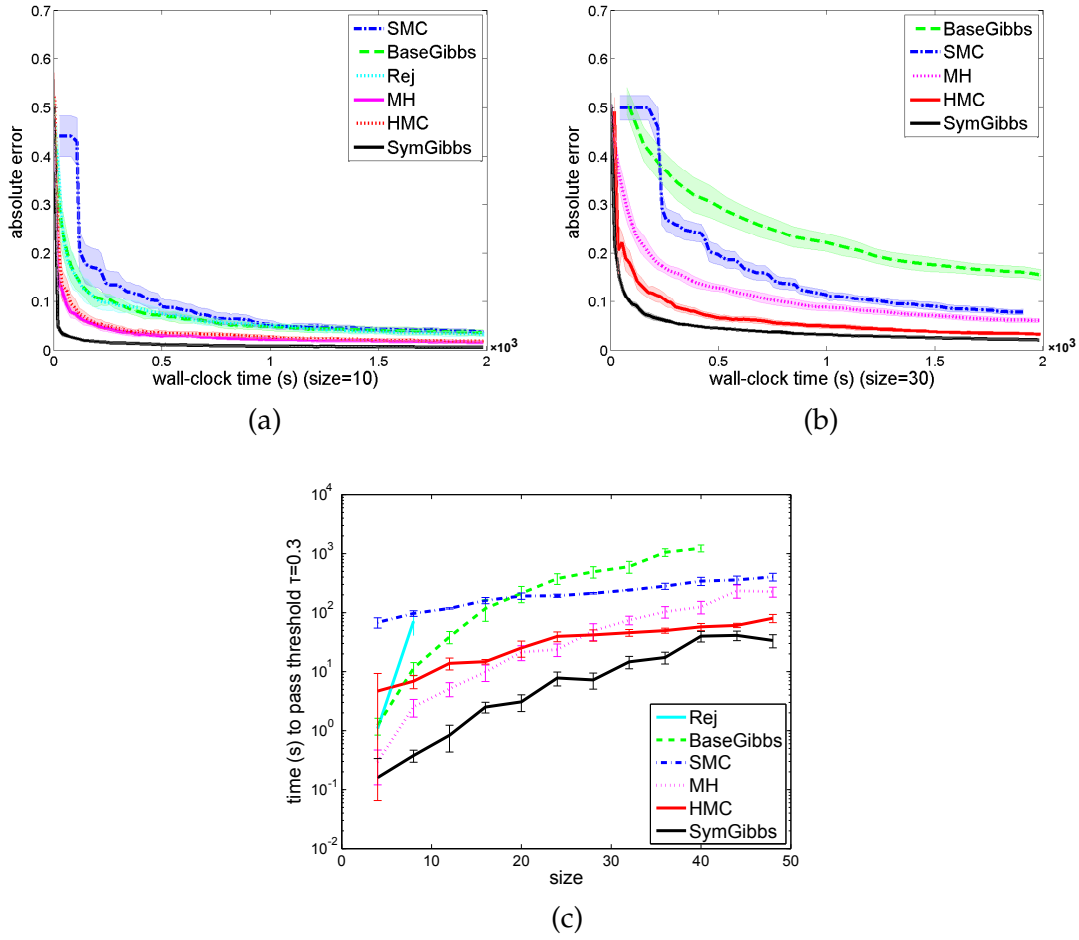


Figure 5.5: MCMC Convergence measurements in the symmetric multi-object momentum model: Absolute error vs time for the collision of (a) 4 and (b) 20 objects; (c) Time to reach error threshold $\tau = 0.3$.

domains.

Let the observation be

$$c = \frac{3n}{(2 \times 10.5 + 9.5)}$$

Due to the symmetry of the problem, the posterior ground truth mean is known:

$$R_i^* = \frac{n}{c} = 10.166667 \qquad \text{for } i = 1, \ldots, n$$

MAE vs. time for networks of 10 and 30 resistors are depicted in Figures 5.6.a & b respectively. Time to reach error threshold $\tau = 0.045$ is plotted in Figure 5.6.c.

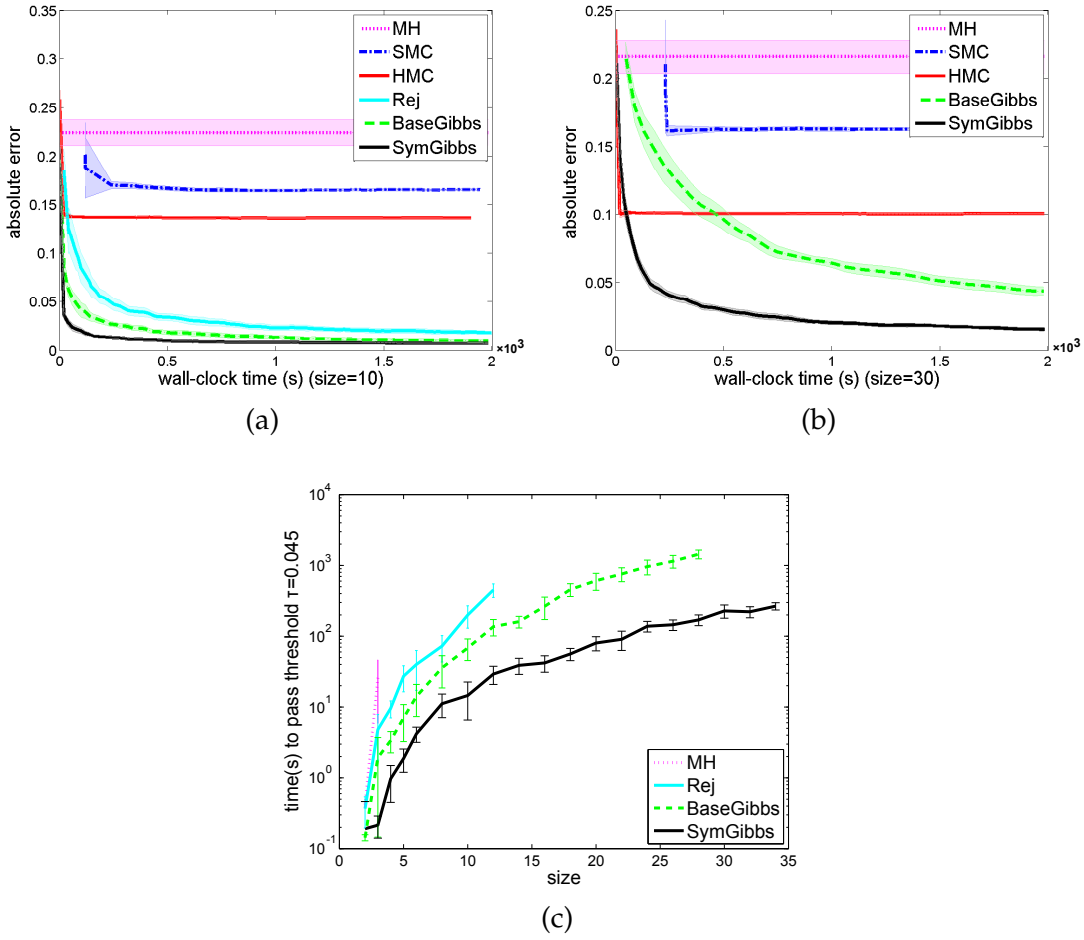


Figure 5.6: MCMC Convergence measurements in the building wiring model: Absolute error vs time for a model with (a) size 4 (i.e. 4 paralleled resistors) and (b) size 30; (c) Time to reach error threshold $\tau = 0.045$.

5.6.4 Experimental evaluations

Given the experimental results, the questions posed in the beginning of this section are resolved as follows:

Posterior quality

Plots of Figure 5.2 illustrate the posterior quality of the models studied in this chapter. They show that even in the presence of a single deterministic constraint, a simple prior can be transformed into a highly piecewise and multi-modal posterior.

MCMC comparison

Plots of Figure 5.3 shows that MH and SMC suffer from low *effective sample size*. Note that the apparent sparsity of plots 5.3-b, 5.3-e & 5.3-f is due to repeated samples (rejected proposals).

The carried out quantitative measurements (Figures 5.4, 5.5 and 5.6) indicate that in all experimental settings, symbolic Gibbs constantly and significantly performs the best.

Handling determinism

All quantitative measurements (Figures 5.4, 5.5 and 5.6) indicate that approximating the determinism by noise leads to poor results.

An interesting result is that in the Building wiring model, even in a dimensionality as low as 10, the Metropolis-Hasting based algorithms (i.e. HM, HMC and SMC) may not converge to the (manually computed) ground truth or their convergence rate is extremely low. This happens regardless of the way determinism is handled. A reason may be as follows: It is widely known that all Metropolis-Hasting-based algorithms are sensitive to parameter tunings. Tuning takes place internally and during the burn-in period according to some heuristics. Such heuristics may not be valid in the densities studied in this chapter as they are very different from the bell-shaped unimodal distributions often studied in the literature so far.

Since Gibbs samplers directly sample from the original distributions (rather than proposal densities) they can cope well with anomalies and as our results show, always converge to the ground truth.

5.7 Conclusion

In this chapter we studied the problem of inference on a rich class of models that consists of polynomial piecewise fractional functions (PPFs). To the best of our knowledge, this is the richest class of piecewise functions studied in the literature of graphical models and probabilistic inference so far. We showed that this family is expressive enough to remain closed under operations required to transform networks with nonlinear deterministic constraints to purely stochastic PPF models amenable to Gibbs sampling.

We showed that a large subset of PPF family (namely, PPF*) have symbolic univariate integrals, which together with determinism elimination, enabled the main contribution of this chapter: a fully-automated exact Gibbs sampler called *symbolic Gibbs*. In *symbolic Gibbs*, all univariate CDFs required for Gibbs sampling are computed analytically and prior to the actual sampling process. In this way, *symbolic Gibbs* saves a significant amount of computation by avoiding per-sample computations, and shows dramatically improved performance compared to existing samplers on complex models motivated by physics and engineering.

PPF* is an extremely expressive family. For instance, arbitrary distributions can be approximated by piecewise polynomials which are a subset of PPF*. To approximate some distributions (most notably *heavy-tailed* distributions) with piecewise polynomials, a large number of pieces may be required which is not a desirable property. However, PPF* also includes a large set of fractional functions and is capable of representing/approximating heavy-tailed distributions such as *Cauchy distribution* with a single piece. The combination of these novel contributions should make probabilistic reasoning applicable to variety of new applications that, to date, have remained beyond the tractability and accuracy boundaries of existing inference methods.

Nonetheless, the family of piecewise algebraic functions has its own limitations and shortcomings. The first problem arises in graphical models with a large number of algebraic potential functions. In such networks, the joint density function is in the form of a high-degree algebraic function. Due to sharp tangents, such functions are often quite sensitive to numerical errors.

Another problem emerges in high dimensional models: the number of partitions required for approximating arbitrary functions via piecewise algebraic forms (up to fixed accuracy levels) grows exponential in the number of dimensions. Clearly, in higher dimensions, such an exponential blow up in the number of partitions rapidly becomes restrictive.

The last restriction is an intrinsic shortcoming of all variations of Gibbs sampling. The complexity of all such samplers grows at least linear with dimensionality (due to the number on inner Gibbs steps). Although not as severe as the previous two problems, this third restriction cannot be neglected.

In the next chapter, we present an alternative sampling-based mechanism for inference on piecewise continuous models. This new sampling method is based on Hamiltonian Monte Carlo mechanism and therefore is not subject to the above restrictions.

Chapter 6

Reflective Hamiltonian Monte Carlo

This chapter proposes a new sampling algorithm that is based on Hamiltonian Monte Carlo mechanism and can be used to carry out asymptotically unbiased reasoning on piecewise continuous probabilistic models. Hamiltonian Monte Carlo (HMC) is a successful approach for sampling from continuous densities. However, it has difficulty simulating Hamiltonian dynamics with non-smooth functions, leading to poor performance. The algorithm proposed in this chapter is motivated by the behavior of Hamiltonian dynamics in physical systems like optics. We introduce a modification of the Leapfrog discretization of Hamiltonian dynamics on piecewise continuous energies, where intersections of the trajectory with discontinuities are detected, and the momentum is *reflected* or *refracted* to compensate for the change in energy. We prove that this method preserves the correct stationary distribution when boundaries are affine. Experiments show that by reducing the number of rejected samples, this method improves on traditional HMC.

6.1 Introduction

Our proposed algorithms to tackle the problem of inference on piecewise continuous models have been based on Gibbs sampling so far. Despite many attractive properties, Gibbs sampling can only be *directly* used on integrable models which is a major restriction.

For instance, symbolic Gibbs sampling cannot be applied to the generalized exponential family of models since the closed-form integration of such functions is often not possible. Nonetheless, these functions form an important class of functions and are encountered or utilized in many applications. An important property of this family is that in the associated models, one can transform the problem of multiplication of factors to the summation of energy functions which is less sensitive to numerical errors.

For such models, Hamiltonian Monte Carlo (HMC) method is a more promising alternative and a more natural choice. The reason is that HMC directly works with *energies* (i.e. minus log of the densities) and therefore is well-matched to the exponential family. As discussed in Section 2.3.2.8, HMC can also be used on any differentiable model and even if the closed-from gradients are not available, they can be approxi-

mated much easier than the integration that Gibbs relies on.

When the probability distribution is smooth, Hamiltonian Monte Carlo (HMC) uses the gradient to simulate Hamiltonian dynamics and reduce random walk behavior. This often leads to a rapid exploration of the distribution [Homan and Gelman 2014; Brooks et al. 2011]. Because of these desirable properties, HMC has recently become popular in Bayesian statistical inference [Stan Development Team 2014], and is often the algorithm of choice.

While HMC is motivated by smooth distributions, the inclusion of an acceptance probability means HMC *does* asymptotically sample correctly from piecewise distributions[1]. However, since leapfrog numerical integration of Hamiltonian dynamics (see [Neal 2011]) relies on the assumption that the corresponding potential energy is smooth, such cases lead to high rejection probabilities, and poor performance. Hence, traditional HMC is rarely used for piecewise distributions.

In physical systems that follow Hamiltonian dynamics [Greenwood 1988], a discontinuity in the energy can result in two possible behaviors. If the energy decreases across a discontinuity, or the momentum is large enough to overcome an increase, the system will cross the boundary with an instantaneous change in momentum, known as *refraction* in the context of optics [Buchdahl 1993]. If the change in energy is too large to be overcome by the momentum, the system will *reflect* off the boundary, again with an instantaneous change in momentum.

Recently, Pakman and Paninski [Pakman and Paninski 2014; Pakman and Paninski 2013] proposed methods for HMC-based sampling from piecewise Gaussian distributions by exactly solving the Hamiltonian equations, and accounting for what we refer to as refraction and reflection above. However, since Hamiltonian equations of motion can rarely be solved exactly, the applications of this method are restricted to distributions whose log-density is piecewise quadratic.

In this chapter, we generalize this work to arbitrary piecewise continuous distributions, where each region is a polytope, i.e. is determined by a set of affine boundaries. We introduce a modification to the leapfrog numerical simulation of Hamiltonian dynamics, called Reflective Hamiltonian Monte Carlo (RHMC), by incorporating reflection and refraction via detecting the first intersection of a linear trajectory with a boundary. We prove that our method has the correct stationary distribution, where the main technical difficulty is proving volume preservation of our dynamics to establish detailed balance. Numerical experiments confirm that our method is more efficient than baseline HMC, due to having fewer rejected proposal trajectories, particularly in high dimensions. As mentioned, the main advantage of this method over [Pakman and Paninski 2014] and [Pakman and Paninski 2013] is that it can be applied to arbitrary piecewise densities, without the need for a closed-form solution to the Hamiltonian dynamics, greatly increasing the scope of applicability.

[1] As in previous chapters, we assume the total measure of the non-differentiable points is zero so that, with probability one, none is ever encountered

6.2 Exact Hamiltonian Dynamics

Throughout this chapter, probability density functions are denoted by the capital letter P since the small letter p is reserved for denoting *momentum*s as we will see shortly. Consider a distribution

$$P(\mathbf{q}) \propto \exp(-U(\mathbf{q}))$$

over $\mathbb{R}^n$, where U is the *potential energy*. Recall from Section 2.3.2.8 that HMC [Neal 2011] is based on considering a joint distribution on momentum and position space

$$P(\mathbf{q}, \mathbf{p}) \propto \exp(-H(\mathbf{q}, \mathbf{p}))$$

where

$$H(\mathbf{q}, \mathbf{p}) = U(\mathbf{q}) + K(\mathbf{p})$$

and K is a quadratic, meaning that

$$P(\mathbf{p}) \propto \exp(-K(\mathbf{p}))$$

is a normal distribution. If one could exactly simulate the dynamics, HMC would proceed by

1. iteratively sampling $\mathbf{p} \sim P(\mathbf{p})$,

2. simulating the following Hamiltonian dynamics equations (6.1) and (6.2)

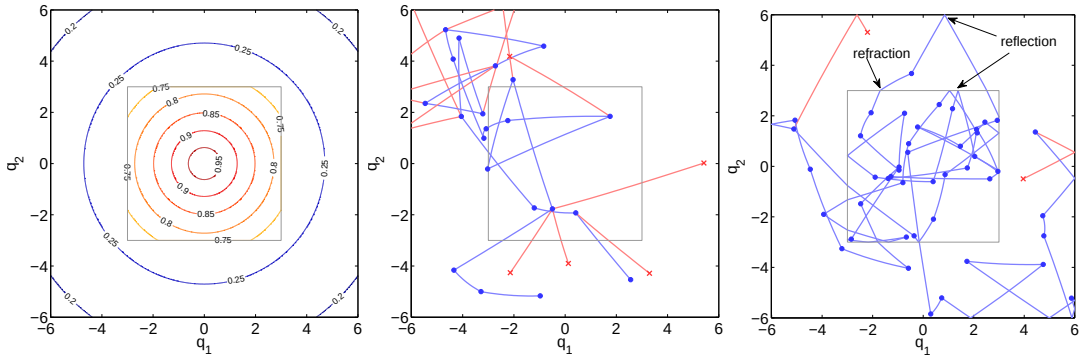
$$\frac{\mathrm{d}q_i}{\mathrm{d}t} = \frac{\partial H}{\partial p_i} = p_i \tag{6.1}$$

$$\frac{\mathrm{d}p_i}{\mathrm{d}t} = -\frac{\partial H}{\partial q_i} = -\frac{\partial U}{\partial q_i} \tag{6.2}$$

   for some period of time ϵ, and

3. reversing the sign of the final value $\mathbf{p}$. (Only needed for the proof of correctness, since this will be immediately discarded at the start of the next iteration in practice.)

It can be shown that steps (1) and (2-3) both leave the distribution $P(\mathbf{p}, \mathbf{q})$ invariant [Neal 2011]. Therefore, so does a Markov chain that alternates between the two steps. Hence, the dynamics have $P(\mathbf{p}, \mathbf{q})$ as a stationary distribution. Of course, the above differential equations are not well-defined when U has discontinuities, and are typically difficult to solve in closed-form.



(a) (b) (c)

Figure 6.1: Example trajectories of baseline and reflective HMC. (a) Contours of the target distribution in two dimensions, as defined in Eq. 6.20. (b) Trajectories of the rejected (red crosses) and accepted (blue dots) proposals using baseline HMC. (c) The same with RHMC. Both use leapfrog parameters $L = 25$ and $\epsilon = 0.1$ In RHMC, the trajectory reflects or refracts on the boundaries of the internal and external polytope boundaries and thus has far fewer rejected samples than HMC, leading to faster mixing in practice. (More examples in supplementary material.)

6.3 Reflection and Refraction with Exact Hamiltonian Dynamics

Take a potential function $U(\mathbf{q})$ which is differentiable in all points except at some boundaries of partitions. Suppose that, when simulating the Hamiltonian dynamics, $(\mathbf{q}, \mathbf{p})$ evolves over time as in the above equations whenever these equations are differentiable. However, when the state reaches a boundary, decompose the momentum vector $\mathbf{p}$ into a component $\mathbf{p}_\perp$ perpendicular to the boundary and a component $\mathbf{p}_\parallel$ parallel to the boundary.

$$\mathbf{p} := \mathbf{p}_\perp + \mathbf{p}_\parallel$$

Let ΔU be the (signed) difference in potential energy on the two sides of the discontinuity. If

$$\|\mathbf{p}_\perp\|^2 > 2\Delta U$$

then $\mathbf{p}_\perp$ is instantaneously replaced by a new vector $\mathbf{p}'_\perp$ where

$$\mathbf{p}'_\perp := \sqrt{\|\mathbf{p}_\perp\|^2 - 2\Delta U} \cdot \frac{\mathbf{p}_\perp}{\|\mathbf{p}_\perp\|}$$

That is, the discontinuity is passed, but the momentum is changed in the direction perpendicular to the boundary (refraction). (If ΔU is positive, the momentum will decrease, and if it is negative, the momentum will increase.) On the other hand, if

$$\|\mathbf{p}_\perp\|^2 \leq 2\Delta U$$

then $\mathbf{p}_\perp$ is instantaneously replaced by $-\mathbf{p}_\perp$. That is, if the particle's momentum is insufficient to climb the potential boundary, it bounces back by reversing the momen-

tum component which is perpendicular to the boundary.

Pakman and Paninski [Pakman and Paninski 2014; Pakman and Paninski 2013] present an algorithm to exactly solve these dynamics for quadratic U. However, for non-quadratic U, the Hamiltonian dynamics rarely have a closed-form solution, and one must resort to numerical integration, the most successful method for which is known as the *leapfrog* dynamics.

6.4 Reflection and Refraction with Leapfrog Dynamics

Informally, HMC with leapfrog dynamics iterates three steps.

1. Sample $\mathbf{p} \sim P(\mathbf{p})$.
2. Perform leapfrog simulation, by discretizing the Hamiltonian equations into L steps using some small step-size ϵ. Here, one interleaves a *position step*

$$\mathbf{q} \leftarrow \mathbf{q} + \epsilon\mathbf{p}$$

   between two half *momentum* steps

$$\mathbf{p} \leftarrow \mathbf{p} - \epsilon\nabla U(\mathbf{q})/2$$

3. Reverse the sign of $\mathbf{p}$.

If $(\mathbf{q}, \mathbf{p})$ is the starting point of the leapfrog dynamics, and $(\mathbf{q}', \mathbf{p}')$ is the final point, *accept* the move with probability

$$\min(1, \exp(H(\mathbf{p}, \mathbf{q}) - H(\mathbf{p}', \mathbf{q}')))$$

See Algorithm 13.

It can be shown that this baseline HMC method has detailed balance with respect to $P(\mathbf{p})$, even if $U(\mathbf{q})$ is discontinuous. However, discontinuities mean that large changes in the Hamiltonian may occur, meaning many steps can be rejected. We propose a modification of the dynamics, namely, *reflective Hamiltonian Monte Carlo* (RHMC), which is also shown in Algorithm 13.

The only modification is applied to the position steps:
In RHMC, the first intersection of the trajectory with the boundaries of the polytope that contains $\mathbf{q}$ must be detected [Pakman and Paninski 2014; Pakman and Paninski 2013]. The position step is only taken up to this boundary, and reflection/refraction occurs, depending on the momentum and change of energy at the boundary. This process continues until the entire amount of time ϵ has been simulated. Note that if there is no boundary in the trajectory to time ϵ, this is equivalent to baseline HMC. Also note that several boundaries might be visited in one position step.

As with baseline HMC, there are two alternating steps, namely drawing a new momentum variable $\mathbf{p}$ from

$$P(\mathbf{p}) \propto \exp(-K(\mathbf{p}))$$

Algorithm 13: BASELINE & REFLECTIVE HMC ALGORITHMS

input : $\mathbf{q}_0$, current sample;
U, potential function;
L, number of leapfrog steps;
ϵ, leapfrog step size
output: next sample

begin
  $\mathbf{q} \leftarrow \mathbf{q}_0$; $\mathbf{p} \sim \mathcal{N}(0, 1)$
  $H_0 \leftarrow \|\mathbf{p}\|^2/2 + U(\mathbf{q})$
  for $l = 1$ **to** L **do**
    # *Half-step evolution of momentum*:
    $\mathbf{p} \leftarrow \mathbf{p} - \epsilon \nabla U(\mathbf{q})/2$
    # *Full-step evolution of position*:
    if BASELINEHMC **then**
      $\mathbf{q} \leftarrow \mathbf{q} + \epsilon \mathbf{p}$
    else
      # (i.e. if REFLECTIVEHMC):
      $t_0 \leftarrow 0$
      while $\left(\langle x, t_x, \Delta U, \phi \rangle \leftarrow \text{FIRSTDISCONTINUITY}(q, p, \epsilon - t_0, U)\right) \neq \emptyset$ **do**
        $\mathbf{q} \leftarrow \mathbf{x}$
        $t_0 \leftarrow t_0 + t_x$
        $\langle \mathbf{p}_\perp, \mathbf{p}_\parallel \rangle = \text{DECOMPOSE}(\mathbf{p}, \phi)$ # *perpendicular/ parallel to boundary plane* ϕ
        if $\|p_\perp\|^2 > 2\Delta U$ **then**
          $\mathbf{p}_\perp \leftarrow \sqrt{\|\mathbf{p}_\perp\|^2 - 2\Delta U} \cdot \frac{\mathbf{p}_\perp}{\|\mathbf{p}_\perp\|}$ # *refraction*
        else
          $\mathbf{p}_\perp \leftarrow -\mathbf{p}_\perp$ # *reflection*
        $\mathbf{p} \leftarrow \mathbf{p}_\perp + \mathbf{p}_\parallel$
      $\mathbf{q} \leftarrow \mathbf{q} + (\epsilon - t_0)\mathbf{p}$
    # *Half-step evolution of momentum*:
    $\mathbf{p} \leftarrow \mathbf{p} - \epsilon \nabla U(\mathbf{q})/2$
  $\mathbf{p} \leftarrow -\mathbf{p}$ # *not required in practice; just for the proof of reversibility*
  $H \leftarrow \|\mathbf{p}\|^2/2 + U(\mathbf{q})$; $\Delta H \leftarrow H - H_0$
  if $s \sim \mathcal{U}(0, 1) < e^{-\Delta H}$ **return q else return** $\mathbf{q}_0$

note : FIRSTDISCONTINUITY$(\cdot)$ returns $\mathbf{x}$, the position of the first intersection of a boundary plain with line segment $[\mathbf{q}, \mathbf{q} + (\epsilon - t_0)\mathbf{p}]$; t_x, the time it is visited; ΔU = $U(\mathbf{x}^+)$ - $U(\mathbf{x}^-)$ and ϕ, the visited partition boundary. If no such point exists, $\emptyset$ is returned.

and proposing a move

$$(\mathbf{p}, \mathbf{q}) \to (\mathbf{p}', \mathbf{q}')$$

and accepting or rejecting it with a probability determined by a Metropolis-Hastings ratio. We can show that both of these steps leave the joint distribution P invariant, and hence a Markov chain that also alternates between these steps will also leave P invariant.

As it is easy to see, drawing $\mathbf{p}$ from $P(\mathbf{p})$ will leave $P(\mathbf{q}, \mathbf{p})$ invariant, we concentrate on the second step i.e. where a move is proposed according to the piecewise leapfrog dynamics shown in Algorithm 13.

Firstly, it is clear that these dynamics are time-reversible, meaning that if the simulation takes state $(\mathbf{q}, \mathbf{p})$ to $(\mathbf{q}', \mathbf{p}')$ it will also take state $(\mathbf{q}', \mathbf{p}')$ to $(\mathbf{q}, \mathbf{p})$ (as a matter of fact, the system oscillates between these two states since the sign of momentum is reversed in step 3).

Secondly, we will show that these dynamics are volume preserving. Formally, if $\mathcal{D}$ denotes the leapfrog dynamics (i.e. the transformation mapping $(\mathbf{q}, \mathbf{p})$ to $(\mathbf{q}', \mathbf{p}')$), we will show that the absolute value of the determinant of the Jacobian of $\mathcal{D}$ is one.

These two properties together show that the probability density of proposing a move from $(\mathbf{q}, \mathbf{p})$ to $(\mathbf{q}', \mathbf{p}')$ is the same of that proposing a move from $(\mathbf{q}', \mathbf{p}')$ to $(\mathbf{q}, \mathbf{p})$ (refer to [Neal 2011]). Thus, if the move

$$(\mathbf{q}, \mathbf{p}) \to (\mathbf{q}', \mathbf{p}')$$

is accepted according to the standard Metropolis-Hastings ratio,

$$R\left((\mathbf{q}, \mathbf{p}) \to (\mathbf{q}', \mathbf{p}')\right) = \min(1, \exp(H(\mathbf{q}, \mathbf{p}) - H(\mathbf{q}', \mathbf{p}'))$$

then detailed balance will be satisfied. To see this, let Q denote the proposal distribution, Then, the usual proof of correctness for Metropolis-Hastings applies, namely that

$$\begin{aligned}\frac{P(\mathbf{q}, \mathbf{p}) Q\left((\mathbf{q}, \mathbf{p}) \to (\mathbf{q}', \mathbf{p}')\right) R\left((\mathbf{q}, \mathbf{p}) \to (\mathbf{q}', \mathbf{p}')\right)}{P(\mathbf{q}', \mathbf{p}') Q((\mathbf{q}', \mathbf{p}') \to (\mathbf{q}, \mathbf{p})) R((\mathbf{q}', \mathbf{p}') \to (\mathbf{q}, \mathbf{p}))} \\ = \frac{P(\mathbf{q}, \mathbf{p}) \min(1, \exp(H(\mathbf{q}, \mathbf{p}) - H(\mathbf{q}', \mathbf{p}'))}{P(\mathbf{q}', \mathbf{p}') \min(1, \exp(H(\mathbf{q}', \mathbf{p}') - H(\mathbf{q}, \mathbf{p}))} = 1. \quad (6.3)\end{aligned}$$

(The final equality is easy to establish, considering the cases where

$$H(\mathbf{q}, \mathbf{p}) \geq H(\mathbf{q}', \mathbf{p}')$$

and

$$H(\mathbf{q}', \mathbf{p}') \leq H(\mathbf{q}, \mathbf{p})$$

separately.) This means that detailed balance holds, and so P is a stationary distribution.

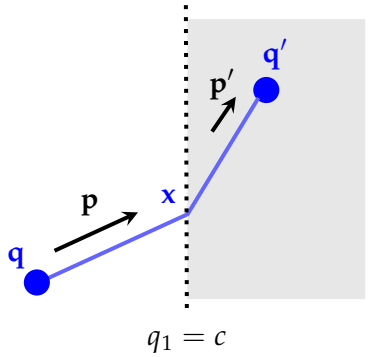


Figure 6.2: Transformation $\mathcal{T} : \langle \mathbf{q}, \mathbf{p} \rangle \to \langle \mathbf{q}', \mathbf{p}' \rangle$ as described by Lemma 1 (Reflection).

The major difference in the analysis of RHMC, relative to traditional HMC is that showing conservation of volume is more difficult. With standard HMC and leapfrog steps, volume conservation is easy to show by observing that each part of a leapfrog step is a shear transformation. This is not the case with RHMC, and so we must resort to a full analysis of the determinant of the Jacobian, as explored in the following section.

6.5 Volume Conservation

In the total leapfrog full-step time ϵ, zero, one or several reflections/refractions may occur. To prove that in the total period ϵ the volume is conserved, our strategy is as follows: We divide ϵ into arbitrary smaller periods,

$$\epsilon = \tau_1 + \tau_2 + \cdots$$

such that in each period τ_i, at most one reflection or refraction occurs. If in a τ_i period no reflection/refraction happens, then the transformation is shear which conserved the volume. The following two lemmas also show that in case a reflection or refraction occurs, the volume is still preserved. As a result we conclude that in total time ϵ, the volume is conserved as well.

6.5.1 Refraction

In our first result, we assume without loss of generality, that there is a boundary located at the hyperplane $q_1 = c$. This Lemma shows that, in the refractive case, volume is conserved. The setting is visualized in Figure 6.2.

Lemma 1 (Volume conservation visa a single refraction)**.** *Let $\mathcal{T} : \langle \boldsymbol{q}, \boldsymbol{p} \rangle \to \langle \boldsymbol{q}', \boldsymbol{p}' \rangle$ be a transformation in $\mathbb{R}^n$ that takes a unit mass located at $\boldsymbol{q} := (q_1, \ldots, q_n)$ and moves it with constant momentum $\boldsymbol{p} := (p_1, \ldots, p_n)$ till it reaches a plane $q_1 = c$ (at some point*

$\boldsymbol{x} := (c, x_2, \ldots, x_n)$ *where c is a constant). Subsequently the momentum is changed to*

$$\boldsymbol{p}' = \left(\sqrt{p_1^2 - 2\Delta U(\boldsymbol{x})}, p_2, \ldots, p_n \right)$$

where $\Delta U(\cdot)$ *is a function of* $\boldsymbol{x}$ *such that* $p_1^2 > 2\Delta U(\boldsymbol{x})$ *The move is carried on for the total time period* τ *till it ends in* $\boldsymbol{q}' := (q_1', \ldots, q_n')$ $\mathcal{T}$ *satisfies the volume preservation property.*

Proof. For $i > 1$, the momentum is not affected by the collision,

$$\forall i > 1, \qquad p_i' = p_i$$

therefore,

$$\forall i > 1, \qquad q_i' = q_i + \tau \cdot p_i$$

As a result,

$$\forall j \in \{2, \ldots, n\} \text{ s.t. } j \neq i, \qquad \frac{\partial q_i'}{\partial q_j} = \frac{\partial q_i'}{\partial p_j} = \frac{\partial p_i'}{\partial q_j} = \frac{\partial p_i'}{\partial p_j} = 0. \tag{6.4}$$

Therefore, if we explicitly write out the Jacobian determinant of $\mathcal{T}$ and apply (6.4), it is

$$|J| = \begin{vmatrix} \frac{\partial q_1'}{\partial q_1} & \frac{\partial q_1'}{\partial p_1} & \frac{\partial q_1'}{\partial q_2} & \cdots & \frac{\partial q_1'}{\partial p_{n-1}} & \frac{\partial q_1'}{\partial q_n} & \frac{\partial q_1'}{\partial p_n} \\ \frac{\partial p_1'}{\partial q_1} & \frac{\partial p_1'}{\partial p_1} & \frac{\partial p_1'}{\partial q_2} & \cdots & \frac{\partial p_1'}{\partial p_{n-1}} & \frac{\partial p_1'}{\partial q_n} & \frac{\partial p_1'}{\partial p_n} \\ \frac{\partial q_2'}{\partial q_1} & \frac{\partial q_2'}{\partial p_1} & \frac{\partial q_2'}{\partial q_2} & \cdots & \frac{\partial q_2'}{\partial p_{n-1}} & \frac{\partial q_2'}{\partial q_n} & \frac{\partial q_2'}{\partial p_n} \\ \vdots & & & \ddots & \vdots & & \vdots \\ \frac{\partial p_{n-1}'}{\partial q_1} & \frac{\partial p_{n-1}'}{\partial p_1} & \frac{\partial p_{n-1}'}{\partial q_2} & \cdots & \frac{\partial p_{n-1}'}{\partial p_{n-1}} & \frac{\partial p_{n-1}'}{\partial q_n} & \frac{\partial p_{n-1}'}{\partial p_n} \\ \frac{\partial q_n'}{\partial q_1} & \frac{\partial q_n'}{\partial p_1} & \frac{\partial q_n'}{\partial q_2} & \cdots & \frac{\partial q_n'}{\partial p_{n-1}} & \frac{\partial q_n'}{\partial q_n} & \frac{\partial q_n'}{\partial p_n} \\ \frac{\partial p_n'}{\partial q_1} & \frac{\partial p_n'}{\partial p_1} & \frac{\partial p_n'}{\partial q_2} & \cdots & \frac{\partial p_n'}{\partial p_{n-1}} & \frac{\partial p_n'}{\partial q_n} & \frac{\partial p_n'}{\partial p_n} \end{vmatrix}$$

$$= \begin{vmatrix} \frac{\partial q_1'}{\partial q_1} & \frac{\partial q_1'}{\partial p_1} & \frac{\partial q_1'}{\partial q_2} & \cdots & \frac{\partial q_1'}{\partial p_{n-1}} & \frac{\partial q_1'}{\partial q_n} & \frac{\partial q_1'}{\partial p_n} \\ \frac{\partial p_1'}{\partial q_1} & \frac{\partial p_1'}{\partial p_1} & \frac{\partial p_1'}{\partial q_2} & \cdots & \frac{\partial p_1'}{\partial p_{n-1}} & \frac{\partial p_1'}{\partial q_n} & \frac{\partial p_1'}{\partial p_n} \\ 0 & 0 & 1 & \cdots & \frac{\partial q_2'}{\partial p_{n-1}} & \frac{\partial q_2'}{\partial q_n} & \frac{\partial q_2'}{\partial p_n} \\ \vdots & & & \ddots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 & \frac{\partial p_{n-1}'}{\partial q_n} & \frac{\partial p_{n-1}'}{\partial p_n} \\ 0 & 0 & 0 & \cdots & 0 & 1 & \frac{\partial q_n'}{\partial p_n} \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{vmatrix} \tag{6.5}$$

Now, using standard properties of the determinant, we have that

$$|J| = \begin{vmatrix} \frac{\partial q_1'}{\partial q_1} & \frac{\partial q_1'}{\partial p_1} \\ \frac{\partial p_1'}{\partial q_1} & \frac{\partial p_1'}{\partial p_1} \end{vmatrix} = \left| \frac{\partial q_1'}{\partial q_1} \frac{\partial p_1'}{\partial p_1} - \frac{\partial p_1'}{\partial q_1} \frac{\partial q_1'}{\partial p_1} \right| \tag{6.6}$$

We will now explicitly calculate these four derivatives. Due to the significance of the result, we carry out the computations in detail. Let t_1 be the time to reach the plane $q_1 = c$. That is

$$t_1 := \frac{c - q_1}{p_1} \tag{6.7}$$

The collision point $\mathbf{x}$ is a determined from the initial position and momentum as well as the time t_1 where the collision occurs:

$$\mathbf{x} = \mathbf{q} + t_1 \cdot \mathbf{p} \tag{6.8}$$

Let t_2 be the period between reaching $\mathbf{x}$ and the last point $\mathbf{q}'$.

$$t_2 := \tau - t_1 = \tau + \frac{q_1 - c}{p_1} \tag{6.9}$$

From (6.9) it immediately follows that,

$$\frac{\partial t_2}{\partial q_1} = \frac{1}{p_1} \tag{6.10}$$

As it can be seen in Figure 6.2,

$$q_1' = c + p_1' \cdot t_2 \tag{6.11}$$

By equations (6.11) and (6.10):

$$\frac{\partial q_1'}{\partial q_1} = \frac{\partial q_1'}{\partial p_1'} \cdot \frac{\partial p_1'}{\partial q_1} + \frac{\partial q_1'}{\partial t_2} \cdot \frac{\partial t_2}{\partial q_1} = t_2 \cdot \frac{\partial p_1'}{\partial q_1} + p_1' \cdot \frac{1}{p_1} \tag{6.12}$$

Similarly, by using equations (6.11) and (6.9):

$$\frac{\partial q_1'}{\partial p_1} = \frac{\partial q_1'}{\partial p_1'} \cdot \frac{\partial p_1'}{\partial p_1} + \frac{\partial q_1'}{\partial t_2} \cdot \frac{\partial t_2}{\partial p_1} = t_2 \cdot \frac{\partial p_1'}{\partial p_1} + p_1' \cdot \frac{c - q_1}{p_1^2} \tag{6.13}$$

On the other hand, remember that by definition

$$p_1' := \sqrt{p_1^2 - 2\Delta U(\mathbf{x})} \tag{6.14}$$

therefore,

$$\frac{\partial p_1'}{\partial p_1} \overset{(6.14)}{=} \frac{1}{2\sqrt{p_1^2 - 2\Delta U(\mathbf{x})}} \cdot \frac{\partial \left(p_1^2 - 2\Delta U(\mathbf{x})\right)}{\partial p_1} = \frac{p_1 - \partial \Delta U(\mathbf{x})/\partial p_1}{p_1'} \tag{6.15}$$

and

$$\frac{\partial p_1'}{\partial q_1} = \frac{1}{2\sqrt{p_1^2 - 2\Delta U(\mathbf{x})}} \cdot \frac{\partial \left(p_1^2 - 2\Delta U(\mathbf{x})\right)}{\partial q_1} = \frac{1}{p_1'} \cdot \frac{-\partial \Delta U(\mathbf{x})}{\partial q_1} \tag{6.16}$$

Using equations (6.7) and (6.8) we can compute the partial derivative of the collision point $\mathbf{x}$ with respect to p_1 (i.e. the first element of the initial momentum) as follows:

$$\begin{aligned}\frac{\partial \mathbf{x}}{\partial p_1} = \frac{\partial \left(\mathbf{q} + \frac{c-q_1}{p_1}\mathbf{p}\right)}{\partial p_1} &= \frac{\partial \mathbf{q}}{\partial p_1} + (c - q_1) \cdot \frac{\partial(\mathbf{p}/p_1)}{\partial p_1} \\ &= (c - q_1)\frac{-1}{p_1^2} \cdot (0, p_2, p_3, \ldots, p_n)\end{aligned} \tag{6.17}$$

Similarly, we can compute the partial derivative of $\mathbf{x}$ with respect to the first element of the initial position vector:

$$\begin{aligned}\frac{\partial \mathbf{x}}{\partial q_1} = \frac{\partial \mathbf{q}}{\partial q_1} + \frac{\mathbf{p}}{p_1} \cdot \frac{\partial(c - q_1)}{\partial q_1} &= \frac{\mathbf{q}}{q_1} - \frac{\mathbf{p}}{p_1} \\ &= (1, 0, \ldots, 0) - (1, \frac{p_2}{p_1}, \ldots, \frac{p_n}{p_1}) = \frac{-1}{p_1}(0, p_2, \ldots, p_n)\end{aligned} \tag{6.18}$$

Now, we are able to compute the total Jacobian by substituting the appropriate terms into equation (6.6):

$$\begin{aligned}|J| &\overset{(6.5)}{=} \frac{\partial q_1'}{\partial q_1} \cdot \frac{\partial p_1'}{\partial p_1} - \frac{\partial q_1'}{\partial p_1} \cdot \frac{\partial p_1'}{\partial q_1} \\ &\overset{(6.12\,\&\,6.13)}{=} \left(t_2 \frac{\partial p_1'}{\partial q_1} + \frac{p_1'}{p_1}\right) \cdot \frac{\partial p_1'}{\partial p_1} - \left(t_2 \frac{\partial p_1'}{\partial p_1} + p_1' \frac{c - q_1}{p_1^2}\right) \cdot \frac{\partial p_1'}{\partial q_1} \\ &= \frac{p_1'}{p_1}\left(\frac{\partial p_1'}{\partial p_1} + \frac{q_1 - c}{p_1} \cdot \frac{\partial p_1'}{\partial q_1}\right) \overset{(6.15\,\&\,6.16)}{=} \frac{1}{p_1}\left(p_1 - \frac{\partial \Delta U(\mathbf{x})}{\partial p_1} - \frac{q_1 - c}{p_1} \cdot \frac{\partial \Delta U(\mathbf{x})}{\partial q_1}\right) \\ &= 1 - \frac{1}{p_1}\left(\frac{\partial \Delta U(\mathbf{x})}{\partial \mathbf{x}} \cdot \frac{\partial \mathbf{x}}{\partial p_1} + \frac{q_1 - c}{p_1} \cdot \frac{\partial \Delta U(\mathbf{x})}{\partial \mathbf{x}} \cdot \frac{\partial \mathbf{x}}{\partial q_1}\right) \\ &\overset{(6.17\,\&\,6.18)}{=} 1 - \frac{1}{p_1} \cdot \frac{\partial \Delta U(\mathbf{x})}{\partial \mathbf{x}}\left(\frac{q_1 - c}{p_1^2} \cdot (0, p_2, p_3, \ldots, p_n)\right. \\ &\qquad\qquad \left. + \frac{q_1 - c}{p_1} \cdot \frac{-1}{p_1}(0, p_2, \ldots, p_n)\right) = 1.\end{aligned}$$

□

6.5.2 Reflection

Now, we turn to the reflective case, and again show that volume is conserved. Again, we assume without loss of generality that there is a boundary located at the hyperplane $q_1 = c$. The setting is visualized in Figure 6.3.

Lemma 2. *[Volume conservation visa a single reflection] Let*

$$\mathcal{T} : \langle \boldsymbol{q}, \boldsymbol{p} \rangle \to \langle \boldsymbol{q}', \boldsymbol{p}' \rangle$$

be a transformation in $\mathbb{R}^n$ that takes a unit mass located at

$$\boldsymbol{q} := (q_1, \ldots, q_n)$$

and moves it with the constant momentum

$$\boldsymbol{p} := (p_1, \ldots, p_n)$$

till it reaches a plane $q_1 = c$ (at some point $\boldsymbol{x} := (c, x_2, \ldots, x_n)$ where c is a constant). Subsequently the mass is bounced back (reflected) with momentum

$$\boldsymbol{p}' = (-p_1, p_2, \ldots, p_n)$$

The move is carried on for a total time period τ till it ends in $\boldsymbol{q}'$. For all $n \in \mathbb{N}$, $\mathcal{T}$ satisfies the volume preservation property.

Proof. Similar to Lemma 1, for $i > 1$, $q_i' = q_i + \tau \cdot p_i$ and $p_i' = p_i$. Therefore

$$\forall j \in \{2, \ldots, n\} \text{ s.t. } j \neq i, \quad \frac{\partial q_i'}{\partial q_j} = \frac{\partial q_i'}{\partial p_j} = \frac{\partial p_i'}{\partial q_j} = \frac{\partial p_i'}{\partial p_j} = 0$$

Consequently, by equation (6.5), and since $p_1' = -p_1$,

$$|J| = \begin{vmatrix} \frac{\partial q_1'}{\partial q_1} & \frac{\partial q_1'}{\partial p_1} \\ \frac{\partial p_1'}{\partial q_1} & \frac{\partial p_1'}{\partial p_1} \end{vmatrix} = \begin{vmatrix} \frac{\partial q_1'}{\partial q_1} & \frac{\partial q_1'}{\partial p_1} \\ 0 & -1 \end{vmatrix} = \frac{-\partial q_1'}{\partial q_1} \tag{6.19}$$

As before, let t_1 be the time to reach $\mathbf{x}$ and t_2 be the period between reaching $\mathbf{x}$ and the last point $\mathbf{q}'$. That is,

$$t_1 := \frac{c - q_1}{p_1}$$

and

$$t_2 := \tau - t_1$$

It follows that q_1' with the following formula

$$q_1' := c + p_1' \cdot t_2$$

is equal to $2c - \tau p_1 - q_1$. Hence, $|J| = 1$. □

6.5.3 Reflective Leapfrog Dynamics

Using lemmas 1 and 2, we can easily prove that in the reflective HMC algorithm for the total leapfrog period ϵ, the volume is preserved.

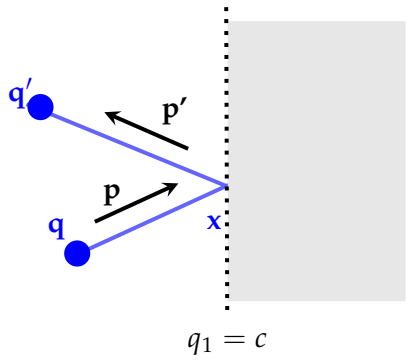


Figure 6.3: Transformation $\mathcal{T} : \langle \mathbf{q}, \mathbf{p} \rangle \to \langle \mathbf{q}', \mathbf{p}' \rangle$ as described by Lemma 2 (Reflection).

Theorem 2. *In RHMC (Algorithm 13) for sampling from a continuous and piecewise distribution P which has affine partitioning boundaries, leapfrog simulation preserves volume in $(\boldsymbol{q}, \boldsymbol{p})$ space.*

Proof. We split the algorithm into several atomic transformations $\mathcal{T}_i$. Each transformation is either (a) a momentum step, (b) a full position step with no reflection/refraction or (c) a full or partial position step where exactly one reflection or refraction occurs.

To prove that the total algorithm preserves volume, it is sufficient to show that the volume is preserved under each $\mathcal{T}_i$ (i.e. $|J_{\mathcal{T}_i}(\mathbf{q}, \mathbf{p})| = 1$) since:

$$|J_{\mathcal{T}_1 o \mathcal{T}_2 o \cdots o \mathcal{T}_m}| = |J_{\mathcal{T}_1}| \cdot |J_{\mathcal{T}_2}| \cdots |J_{\mathcal{T}_m}|$$

Transformations of kind (a) and (b) are shear mappings and therefore they preserve the volume [Neal 2011]. Now consider a (full or partial) position step where a single refraction occurs. If the reflective plane is in the form $q_1 = c$, by lemma 1, the volume preservation property holds. Otherwise, as long as the reflective plane is affine, via a rotation of basis vectors, the problem is reduced to the former case. Since volume is conserved under rotation, in this case the volume is also conserved. With similar reasoning, by lemma 2, reflection on a affine reflective boundary preserves volume. Thus, since all component transformations of RHMC leapfrog simulation preserve volume, the proof is complete. □

Along with the fact that the leapfrog dynamics are time-reversible, this shows that the algorithm satisfies detailed balance, and so has the correct stationary distribution.

6.6 Experiment

Compared to baseline HMC, we expect that RHMC will simulate Hamiltonian dynamics more accurately and therefore leads to fewer rejected samples. On the other hand, this comes at the expense of slower leapfrog position steps since intersections, reflections and refractions must be computed. To test the trade off, we compare the RHMC to baseline HMC [Neal 2011] and tuned Metropolis-Hastings (MH) with a simple isotropic Normal proposal distribution. MH is automatically tuned after [Roberts

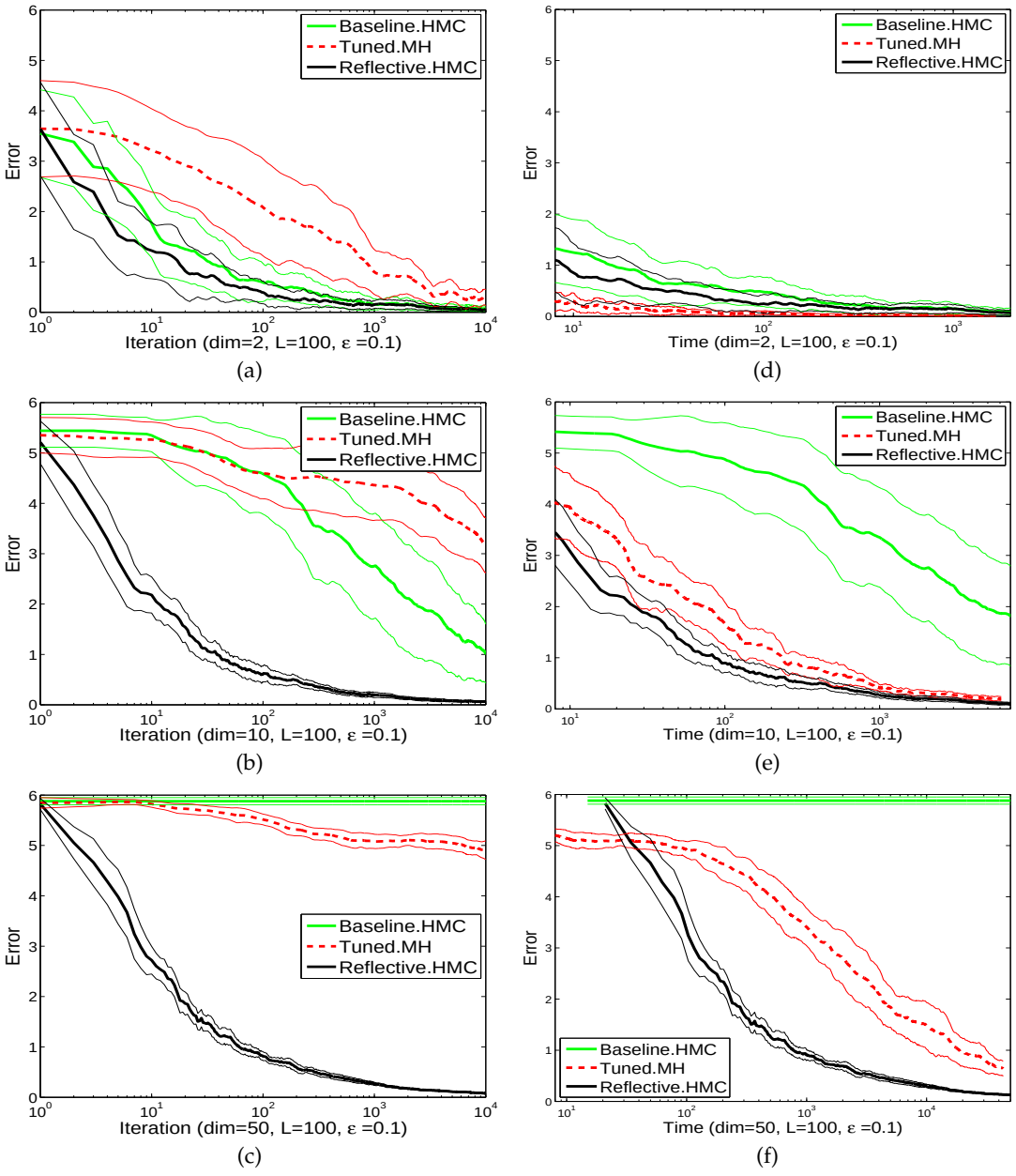


Figure 6.4: Error (worst mean absolute error per dimension) versus (a-c) iterations and (e-f) time (ms). Tuned.HM is Metropolis Hastings with a tuned isotropic Gaussian proposal distribution. (Many more examples in supplementary material.)

et al. 1997] by testing 100 equidistant proposal variances in interval $(0, 1]$ and accepting a variance for which the acceptance rate is closest to 0.24. In the baseline HMC and RHMC the number of leapfrog steps L and the leapfrog step size ϵ are chosen to be 100 and 0.1 respectively.

The comparison takes place on a heavy tail piecewise model with (non-normalized) negative log probability

$$U(\mathbf{q}) = \begin{cases} \sqrt{\mathbf{q}^\top A\, \mathbf{q}} & \text{if } \|\mathbf{q}\|_\infty \leq 3 \\ 1 + \sqrt{\mathbf{q}^\top A\, \mathbf{q}} & \text{if } 3 < \|\mathbf{q}\|_\infty \leq 6 \\ +\infty, & \text{otherwise} \end{cases} \tag{6.20}$$

where A is a positive definite matrix. We carry out the experiment on three choices of (position space) dimensionalites, $n = 2, 10$ and 50.

Due to the symmetry of the model, the ground truth expected value of $\mathbf{q}$ is known to be $\mathbf{0}$. Therefore, the absolute error of the expected value (estimated by a chain $\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(k)}$ of MCMC samples) in each dimension $d = 1, \ldots, n$ is the absolute value of the mean of d-th element of the sample vectors. The *worst mean absolute error* (WMAE) over all dimensions is taken as the error measurement of the chain.

$$\text{WMAE}\left(\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(k)}\right) = \frac{1}{k} \max_{d=1,\ldots n} \left| \sum_{s=1}^{k} {q_d}^s \right| \tag{6.21}$$

For each algorithm, 20 Markov chains are run and the mean WMAE and 99% confidence intervals (as error bars) versus the number of iterations (i.e. Markov chain sizes) as well as time (milliseconds) are depicted in Figure 6.4. All algorithms are implemented in java and run on a single thread of a 3.40GHz CPU.

For each of the 20 repetitions, some random starting point is chosen uniformly and used for all three of the algorithms. We use a diagonal matrix for A where, for each repetition, each entry on the main diagonal is either $\exp(-5)$ or $\exp(5)$ with equal probabilities.

As the results show, even in low dimensions, the extra cost of the position step is more or less compensated by its higher effective sample size but as the dimensionality increases, the RHMC significantly outperforms both baseline HMC and tuned MH.

6.6.1 Sensitivity to parameter tuning

To study the effect of tuning on the baseline and reflective HMC algorithms, in this section we repeat the previous experiment using different leap frog parameter configurations. Namely, in the following figures *error vs iteration* and *error vs time* plots, all combinations of $L \in \{20, 50, 200\}$ and $\epsilon \in \{0.05, 0.1, 0.2, 0.4\}$ are tested on 10 and 50 dimensional models. The worst mean absolute error (WMAE) over all dimensions is taken as the error measurement of a Markov chain. Each experiment is repeated over 10 Markov chains and the means and standard errors are depicted.

These figures show that

1. The proposed reflective HMC (RHMC) remains superior to the baseline HMC over a large range of L and ϵ and in no setting it performs worse.

2. Compared to the basic HMC, the performance of reflective HMC is less sensitive

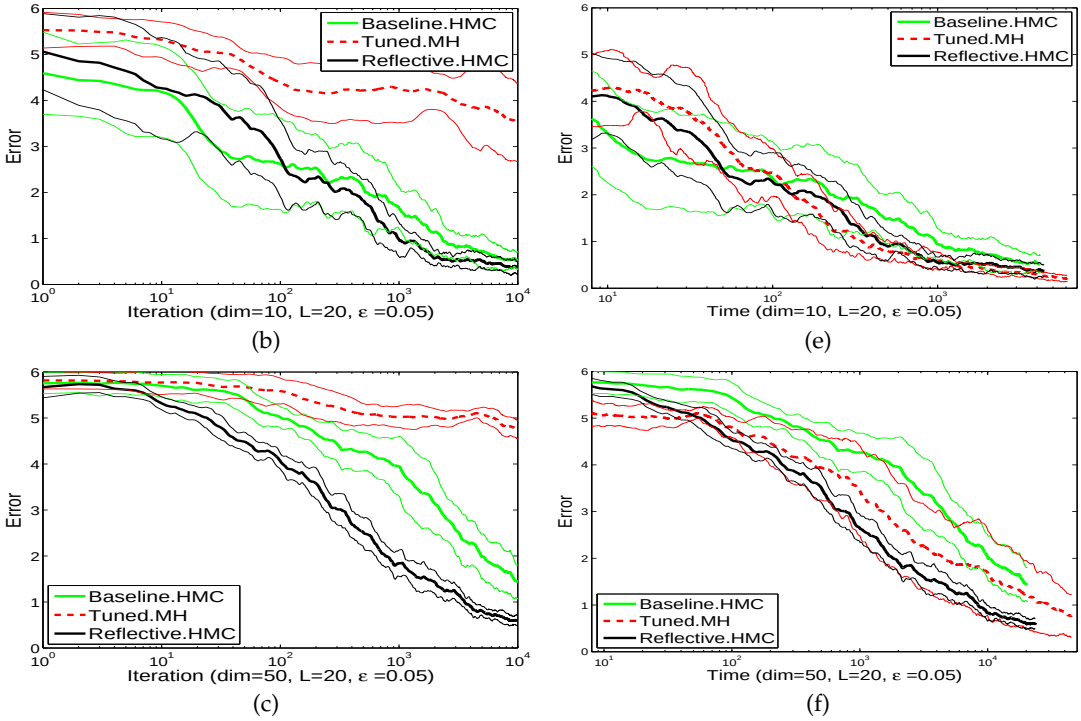


Figure 6.5: WMAE vs. iterations (left) and time (right) (ms) for leapfrog parameters $L = 20$, $\epsilon = 0.05$.

to parameter tuning.

3. Regardless of the tuning, in high dimensional models, reflective HMC performs significantly better than the basic HMC and tuned Metropolis-Hastings.

6.6.2 The rate of rejections, reflections and refractions

In this section, for a range of parameter tunings and model dimensions, the proposal rejection rate of the basic and reflective HMC algorithms as well as the number of reflections and refractions of the latter sampler are depicted in Tables 6.1 to 6.3. In all settings, the rates are averaged over 10 Markov chains.

An interesting observation is that in high dimensions (as in Table 6.3), the number of refractions is significantly less than reflections. This is due to the fact that the studied models consist of two positive-probability (hyper-cubic) partitions one of which is inside the other. In high dimensions, the internal hyper-cube is rarely sampled since most of the mass is placed near the outer surface of the external one. Based on this observation, we predict that in general, due to the effect of *reflection*, the performance of our proposed method on high dimensional truncated/finite-support model should be remarkable. The reason is that in such high-dimensional models, the density mass is concentrated near the manifolds that mark the external surfaces of the density function and the reflections prevent the leapfrog mechanism to enter the zero-probability ambient space.

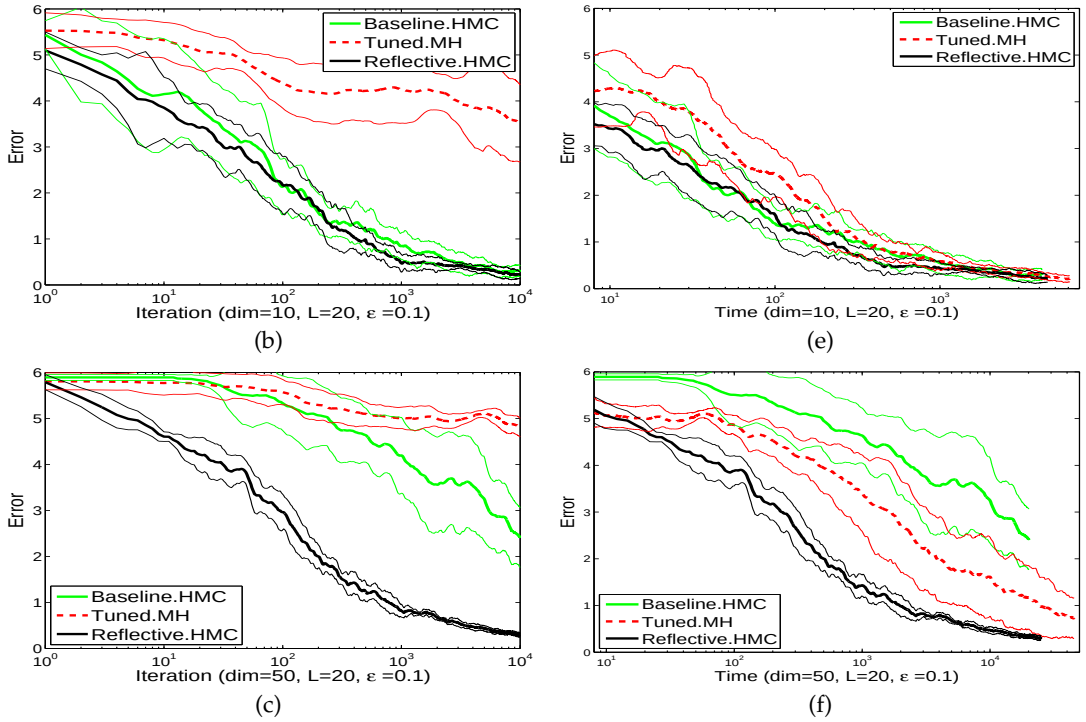


Figure 6.6: WMAE vs. iterations (left) and time (right) (ms) for leapfrog parameters $L = 20$, $\epsilon = 0.1$.

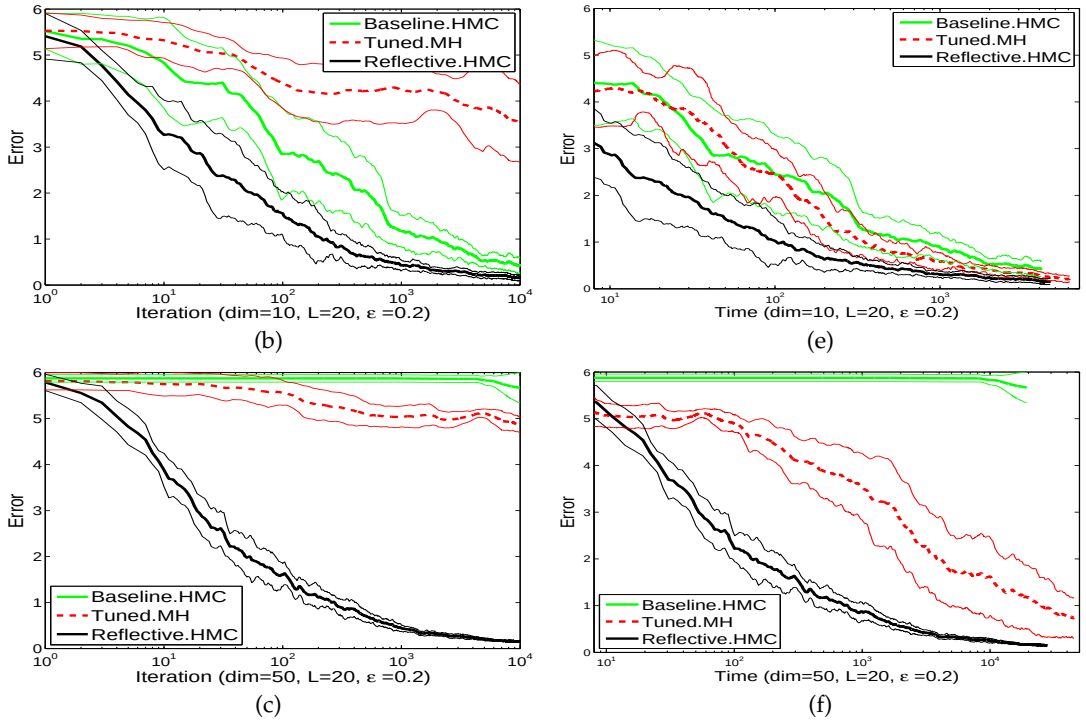


Figure 6.7: WMAE vs. iterations (left) and time (right) (ms) for leapfrog parameters $L = 20$, $\epsilon = 0.2$.

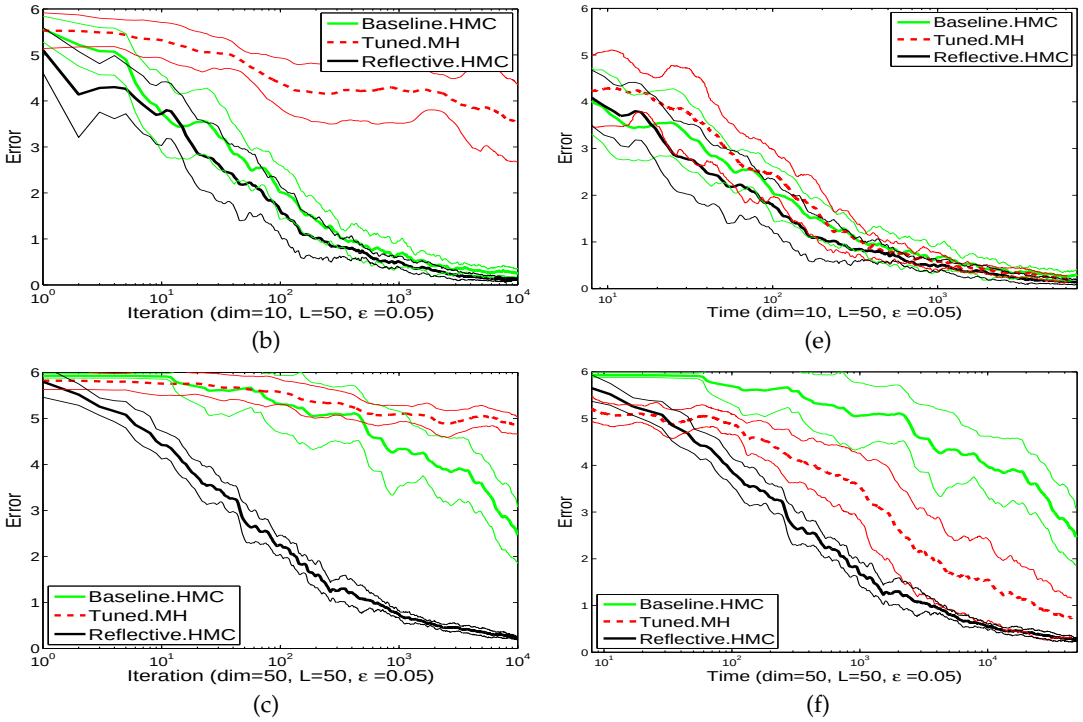


Figure 6.8: WMAE vs. iterations (left) and time (right) (ms) for leapfrog parameters $L = 50$, $\epsilon = 0.05$.

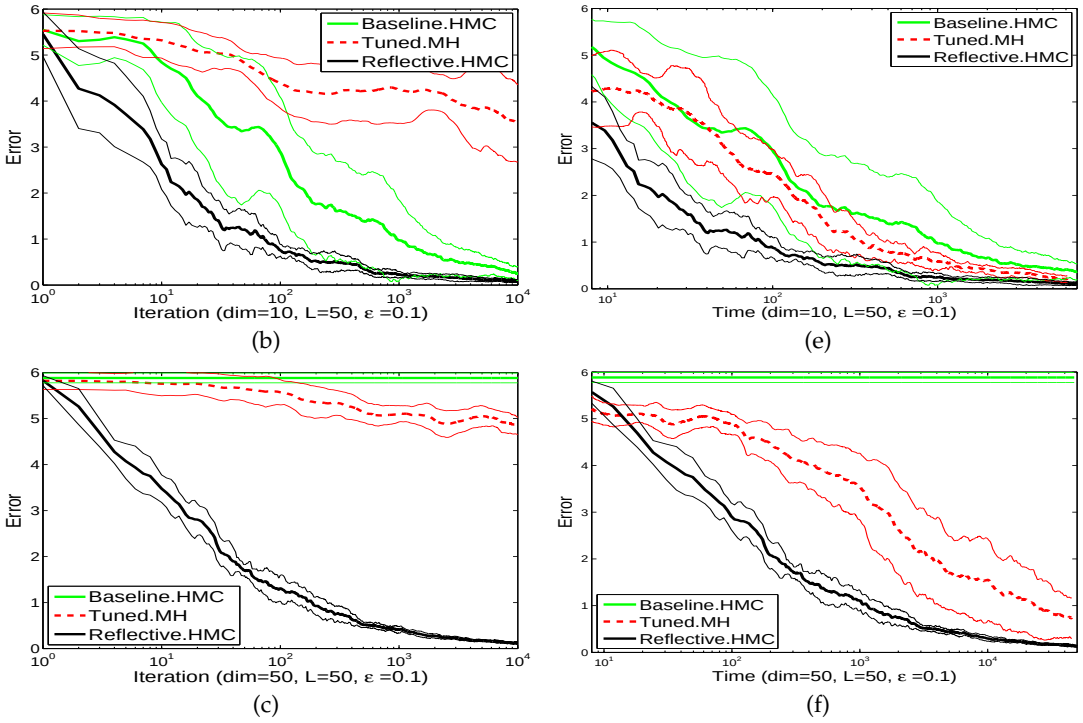


Figure 6.9: WMAE vs. iterations (left) and time (right) (ms) for leapfrog parameters $L = 50$, $\epsilon = 0.1$.

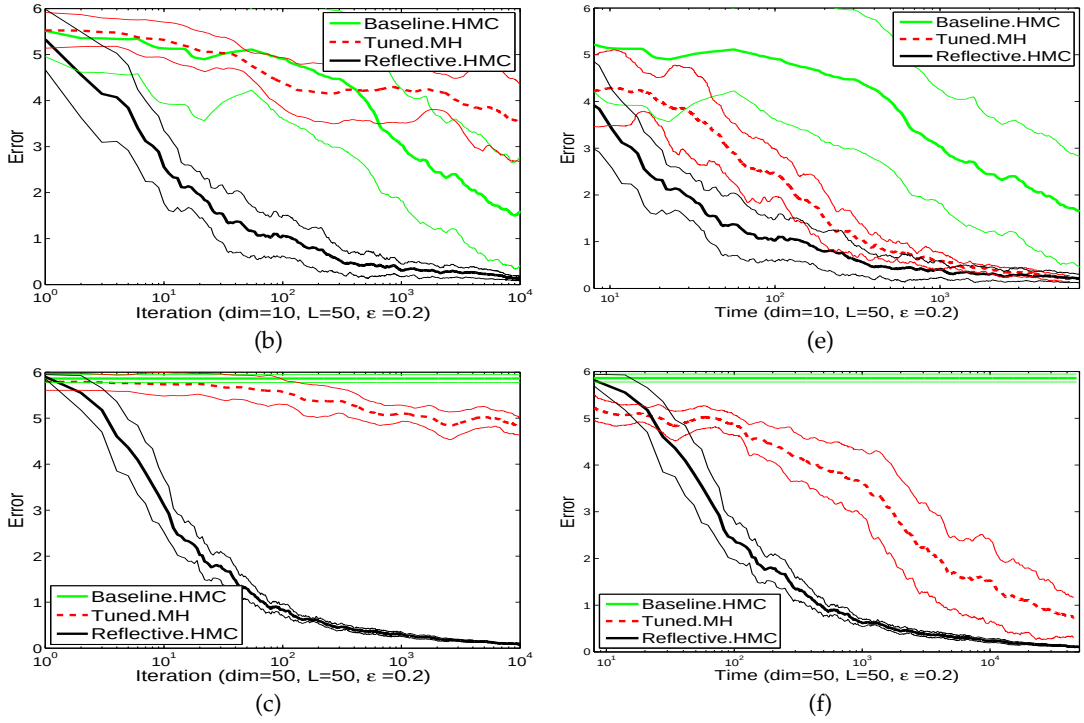


Figure 6.10: WMAE vs. iterations (left) and time (right) (ms) for leapfrog parameters $L = 50$, $\epsilon = 0.2$.

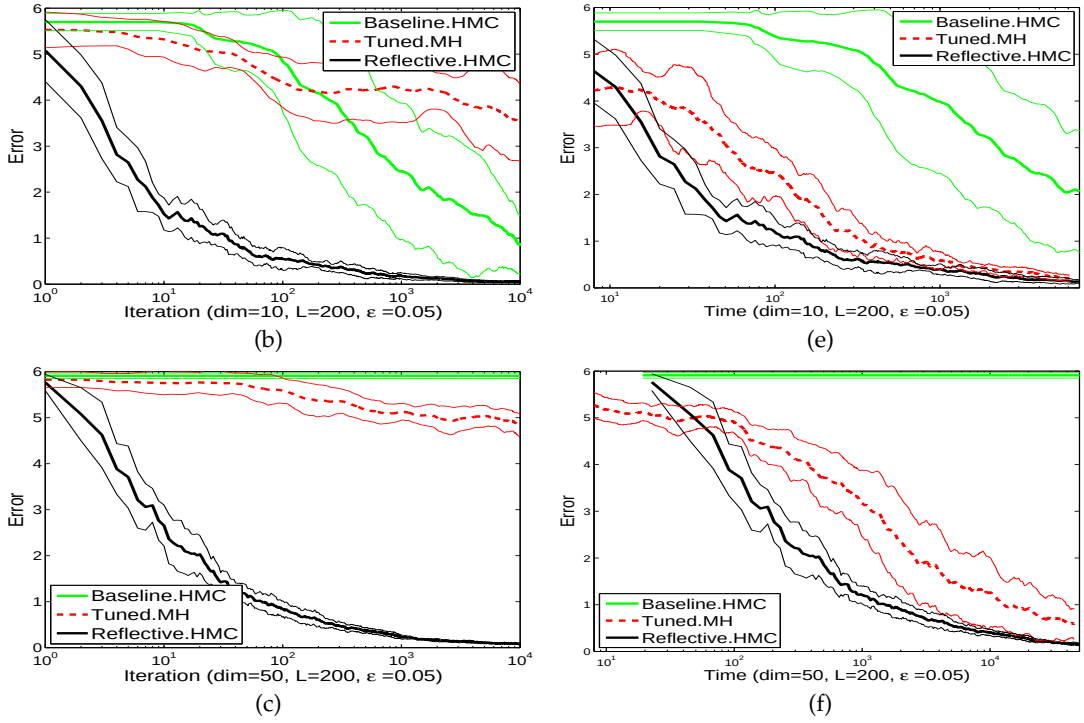


Figure 6.11: WMAE vs. iterations (left) and time (right) (ms) for leapfrog parameters $L = 200$, $\epsilon = 0.05$.

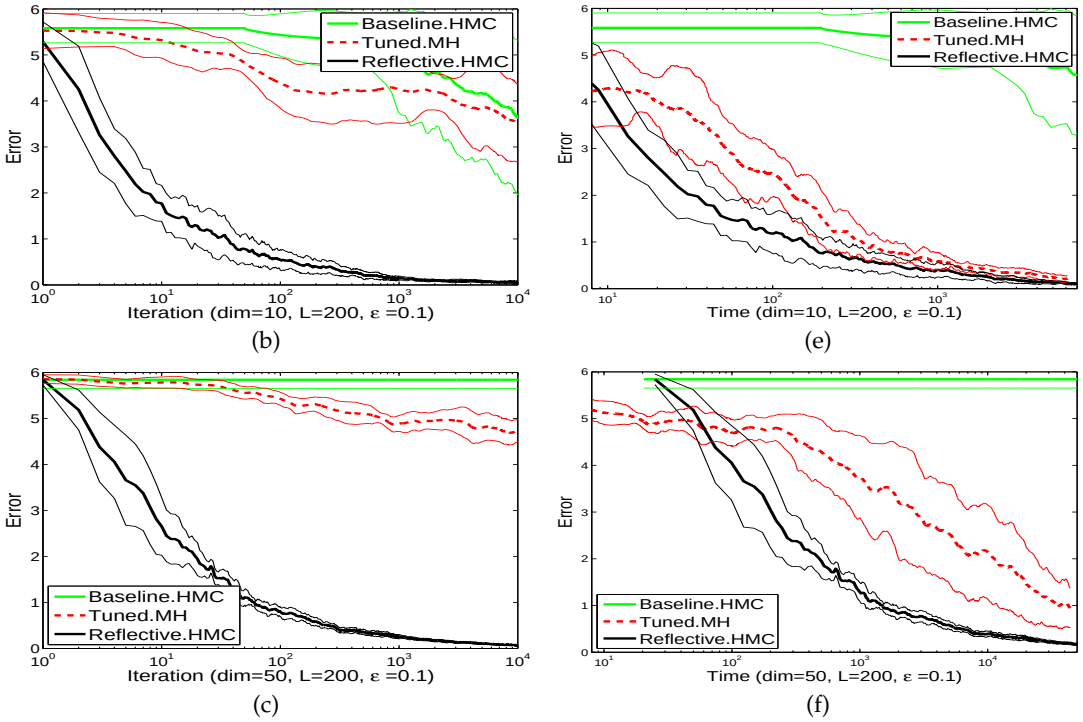


Figure 6.12: WMAE vs. iterations (left) and time (right) (ms) for leapfrog parameters $L = 200$, $\epsilon = 0.1$.

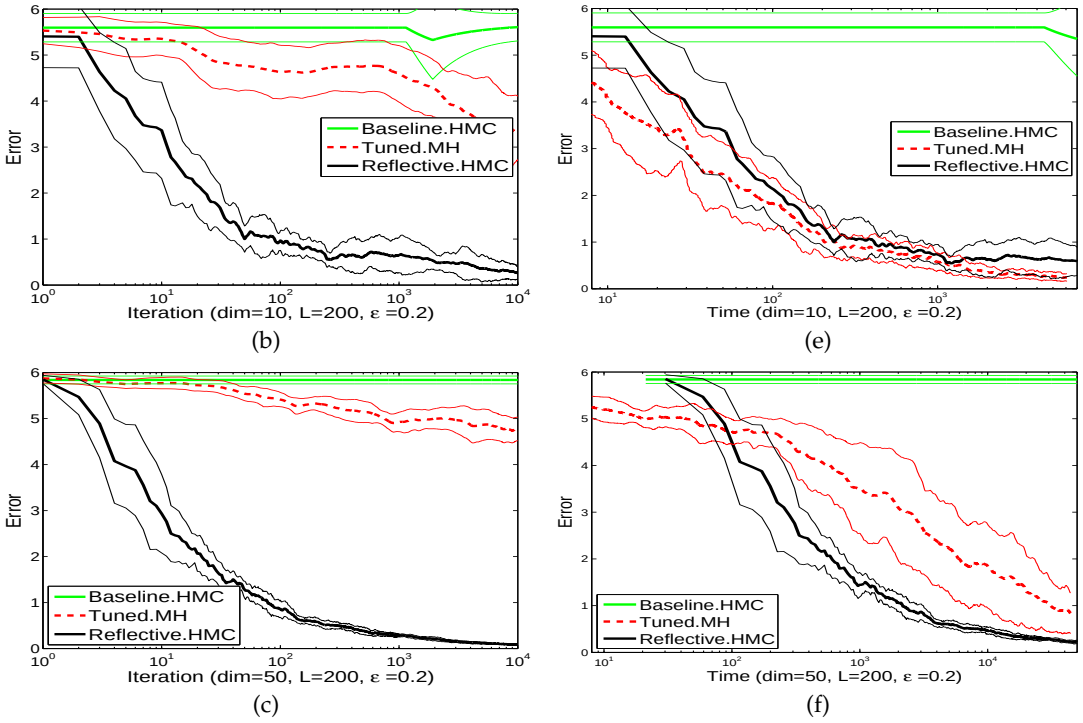


Figure 6.13: WMAE vs. iterations (left) and time (right) (ms) for leapfrog parameters $L = 200$, $\epsilon = 0.2$.

Table 6.1: Rate of rejections/reflections and refractions for sampling 10,000 samples using baseline/reflective HMC methods. Model dimensionality = 2.

		ϵ			
L	Alg.	0.05	0.1	0.2	0.4
20	HMC	no. reject: 1966.8	no. reject: 5120.1	no. reject: 7673.1	no. reject: 9564.6
	RHMC	no. reject: 1307.7 no. reflect: 785.1 no. refract: 556.4	no. reject: 4791.3 no. reflect: 931.9 no. refract: 734.3	no. reject: 6055.1 no. reflect: 3758.6 no. refract: 2603.7	no. reject: 8866.2 no. reflect: 13977.5 no. refract: 36627.5
50	HMC	no. reject: 3587.7	no. reject: 6696.4	no. reject: 9195.2	no. reject: 9510.6
	RHMC	no. reject: 2012.2 no. reflect: 2129.6 no. refract: 1567.7	no. reject: 4510.8 no. reflect: 4685.6 no. refract: 3165.8	no. reject: 8175.9 no. reflect: 8018.5 no. refract: 16627.1	no. reject: 7958.5 no. reflect: 59906.3 no. refract: 121706.7
100	HMC	no. reject: 5288.0	no. reject: 8264.3	no. reject: 8904.9	no. reject: 9416.3
	RHMC	no. reject: 3152.5 no. reflect: 4029.0 no. refract: 2920.2	no. reject: 5495.8 no. reflect: 10549.3 no. refract: 7266.3	no. reject: 6510.4 no. reflect: 20825.6 no. refract: 51209.3	no. reject: 6998.3 no. reflect: 172305.0 no. refract: 261964.0
200	HMC	no. reject: 7100.1	no. reject: 8421.4	no. reject: 9221.1	no. reject: 9426.0
	RHMC	no. reject: 2366.9 no. reflect: 11509.1 no. refract: 7693.9	no. reject: 6450.9 no. reflect: 18628.0 no. refract: 22783.0	no. reject: 6679.2 no. reflect: 66031.6 no. refract: 143159.7	no. reject: 7047.6 no. reflect: 498257.6 no. refract: 645459.2

Table 6.2: Rate of rejections/reflections and refractions for sampling 10,000 samples using baseline/reflective HMC methods. Model dimensionality = 10.

		ϵ			
L	Alg.	0.05	0.1	0.2	0.4
20	HMC	no. reject: 2754.0	no. reject: 5472.9	no. reject: 8873.8	no. reject: 9946.3
	RHMC	no. reject: 130.7 no. reflect: 3089.3 no. refract: 554.4	no. reject: 799.7 no. reflect: 6864.7 no. refract: 878.3	no. reject: 4260.2 no. reflect: 15013.6 no. refract: 1348.5	no. reject: 9365.8 no. reflect: 32340.3 no. refract: 5057.8
50	HMC	no. reject: 5738.4	no. reject: 8337.8	no. reject: 9629.4	no. reject: 9987.0
	RHMC	no. reject: 141.9 no. reflect: 8035.2 no. refract: 1315.6	no. reject: 680.2 no. reflect: 15536.1 no. refract: 2731.2	no. reject: 4921.8 no. reflect: 36844.7 no. refract: 3553.6	no. reject: 9497.0 no. reflect: 113852.2 no. refract: 14880.1
100	HMC	no. reject: 8139.5	no. reject: 9771.6	no. reject: 9968.1	no. reject: 9999.7
	RHMC	no. reject: 157.2 no. reflect: 16291.1 no. refract: 2307.4	no. reject: 960.0 no. reflect: 36164.2 no. refract: 4260.1	no. reject: 4359.8 no. reflect: 67230.7 no. refract: 11126.0	no. reject: 9640.1 no. reflect: 321873.9 no. refract: 24149.7
200	HMC	no. reject: 9600.1	no. reject: 9987.0	no. reject: 9999.9	no. reject: 10000.0
	RHMC	no. reject: 156.7 no. reflect: 32738.1 no. refract: 5022.2	no. reject: 744.1 no. reflect: 64447.1 no. refract: 9978.9	no. reject: 5399.3 no. reflect: 168753.9 no. refract: 14639.0	no. reject: 9662.6 no. reflect: 1022839.6 no. refract: 36450.5

Table 6.3: Rate of rejections/reflections and refractions for sampling 10,000 samples using baseline/reflective HMC methods. Model dimensionality = 50.

		ϵ			
L	Alg.	0.05	0.1	0.2	0.4
20	HMC	no. reject: 8301.4	no. reject: 9719.0	no. reject: 9999.5	no. reject: 10000.0
	RHMC	no. reject: 64.2 no. reflect: 17257.4 no. refract: 0.0	no. reject: 289.8 no. reflect: 33841.4 no. refract: 0.0	no. reject: 988.0 no. reflect: 66524.4 no. refract: 0.2	no. reject: 4197.3 no. reflect: 141453.9 no. refract: 0.0
50	HMC	no. reject: 9869.9	no. reject: 10000.0	no. reject: 10000.0	no. reject: 10000.0
	RHMC	no. reject: 59.9 no. reflect: 40732.3 no. refract: 0.0	no. reject: 233.8 no. reflect: 82537.0 no. refract: 0.0	no. reject: 878.4 no. reflect: 160209.2 no. refract: 0.0	no. reject: 3921.5 no. reflect: 331565.0 no. refract: 0.0
100	HMC	no. reject: 9998.8	no. reject: 10000.0	no. reject: 10000.0	no. reject: 10000.0
	RHMC	no. reject: 56.9 no. reflect: 85937.0 no. refract: 0.4	no. reject: 215.1 no. reflect: 157553.7 no. refract: 0.2	no. reject: 960.5 no. reflect: 356408.4 no. refract: 0.0	no. reject: 3899.6 no. reflect: 665330.2 no. refract: 0.0
200	HMC	no. reject: 10000.0	no. reject: 10000.0	no. reject: 10000.0	no. reject: 10000.0
	RHMC	no. reject: 60.2 no. reflect: 157838.8 no. refract: 0.2	no. reject: 229.8 no. reflect: 335667.9 no. refract: 0.0	no. reject: 888.9 no. reflect: 667757.6 no. refract: 0.2	no. reject: 3911.3 no. reflect: 1325778.9 no. refract: 1.6

6.7 Conclusion

The samplers proposed in Chapters 4 and 5 were suitable for piecewise algebraic models. In this chapter, however, we targeted another important family of models and designed a sampling mechanism for density functions that are naturally in the form of piecewise exponentials.

We presented a modification of the leapfrog dynamics for Hamiltonian Monte Carlo for piecewise smooth energy functions, inspired by physical systems. Though traditional Hamiltonian Monte Carlo can in principle be used on such functions, the fact that the Hamiltonian will often be dramatically changed by the dynamics can result in a very low acceptance ratio, particularly in high dimensions. By better preserving the Hamiltonian, *reflective Hamiltonian Monte Carlo* (RHMC) accepts more moves and thus has a higher effective sample size, leading to much more efficient probabilistic inference.

Our empirical results show that compared to the baseline HMC, our proposed method is much less sensitive to parameter tuning (see Section 6.6.1) and particularly in high dimensional truncated models, its performance is overwhelmingly better (see Section 6.6.2).

Nonetheless, like any other inference technique, RHMC has its own limitation. Most notably, to use RHMC, one must be able to detect the first intersection of a position trajectory with polytope boundaries which is not always easy and can be costly. Another issue is that so far, the volume preservation of RHMC (Theorem 2) is only

proved for the case the density boundaries are affine (i.e. each region is a polytope). Clearly, this is a theoretical restriction. We speculate that the proof of correctness can be generalized to the case where the boundaries are nonlinear. The verification or refutation of this speculation can be a future direction of research.

Chapter 7

Conclusion

7.1 Summary of contributions

In this thesis we investigated the problem of sampling-based inference on piecewise continuous graphical models and proposed three MCMC methods that target different application ranges (see Table 7.1). The summary of our contribution is as follows:

- A major problem for probabilistic inference in piecewise graphical models is that in case a model consists of a large number of piecewise factors (e.g. Bayesian inference with several data points and piecewise likelihood functions), the number of pieces in the joint density can grow exponentially in the number of piecewise factors, leading to scenarios where traditional inference tools face severe difficulties. In Chapter 4, we showed that this issue can be resolved via augmenting the model with auxiliary discrete and deterministic random variables each of which indicates the *activated* partition in each piecewise factor. By utilizing a particular blocked Gibbs sampler we provided an inference mechanism with linear (rather than exponential) growth in the number of piecewise factors.

- In Chapter 5, we proposed another Gibbs sampling based inference tool (namely, *symbolic Gibbs sampling*) and tackled the problem of inference on piecewise models from another point of view. That is, instead of concentrating on piecewise densities that can be factorized into a large number of potential functions (that was our approach in Chapter 4) we focused on effective inference within a particular family of piecewise models, namely *polynomial-piecewise fractional* functions (PPFs), that is, fractional piecewise models with polynomial partitioning restrictions. We showed that on a large subset of this expressive family of functions, Gibbs sampling can be performed at least an order of magnitude faster than its baseline counterpart. The key observation is that in PPF models, most costly computations required for Gibbs sampling can be accomplished analytically and prior to the sampling process rather than per sample. By leveraging this observation, symbolic Gibbs alleviates a significant computational overhead. Our experimental results show that symbolic Gibbs sampling can effectively carry out inference on models that can hardly be handled by means of the existing sampling tools.

Table 7.1: Sampling techniques introduced in Chapters 4 to 6, the restrictions of piecewise models to which they can be applied as well as application domains in which they perform best.

Algorithm	Piecewise model		Application
	Sub-functions	Boundaries	
Augmented Gibbs	Polynomial	Linear/quadratic	Highly factorized models with piecewise factors
Symbolic Gibbs	Fractional	Polynomial	A large range of piecewise algebraic models
Reflective HMC	Differentiable	Linear	High dimensional truncated models

- In Chapter 6, we presented our third and last inference tool that unlike the previous two contributions, is based on Hamiltonian Monte Carlo. This method, namely *reflective Hamiltonian Monte Carlo* (RHMC) is motivated by the reflection and refraction of light (and other physical systems) when encountering a sudden change in the level of potential energy. RHMC is not based on any a priori assumption on the topology of the graphical model and (as long as the gradient vectors of potential energies can be computed or approximated) is not restricted to any family of potential functions. The only limitation is that the partitioning boundaries should be affine. In our experimental models, regardless of the tuning parameters, RHMC outperformed the baseline and it was much less sensitive to tuning. Particularly in high dimensional models, the superiority of RHMC over the baseline was distinct. The reason is that in high dimensions, most of the density mass is placed near the outer borders of the distribution. In these models, reflection is an effective mechanism that prevents the sampler from being lost in the immensity of the 0-probability ambient space.

- The case studies presented in Chapters 1, 4, 5 and 6 demonstrate many scenarios where the probabilistic models are intrinsically piecewise continuous. These models can appear in diverse contexts, from interactions with humans (i.e. as in the preference learning framework) to models with conditional statements (i.e. probabilistic programming) and from models that require random variable transformations (i.e. inference conditioned on observed functions of random variables) to intrinsic deterministic constraints that lead to truncated distributions.

- Our experimental results in different chapters overall indicate that the performance of the traditional MCMC based inference methods on piecewise continuous models is typically poor and in some cases extremely unsatisfactory and therefore, advocate the importance of inference methods that are in particular motivated and designed for piecewise continuous models.

7.2 Future work

While this thesis addresses the problem of sampling from piecewise smooth density functions, there are many unexplored areas for future research. Here we briefly discuss a few potential directions of research:

- **Utilizing the proposed inference tools in new application domains.** In Section 1.1 we motivated the importance of inference on piecewise models by introducing several application domains where the model is intrinsically piecewise continuous. So far, our proposed inference tools are only applied to a few such models. A future direction of research can be utilizing these tools on other relevant domains. Two fields where our tools can potentially outperform other samplers are:
  - *Reasoning under constraints.* In many fields, the set of possible outcomes is confined by hard constraints. These constraints are often modeled by truncated or bounded support distributions. For instance, in economics, the consumer's choice set is constrained by income constraints [Gossen 1983; Samuelson 1966] and/or non-disposable time constraints are imposed on the system [Steedman 2003] . Our proposed tools are viable candidates for effective probabilistic inference on bounded support models. In particular, we predict that due to the reasons mentioned in Section 6.6.2, in high dimensional truncated models, probabilistic inference can be carried out effectively using reflective Hamiltonian Monte Carlo.
  - *Probabilistic programming.* An important and highly popular application of piecewise distributions is probabilistic programming (PP). This domain is not fully covered by thesis and may be dealt with in future work.
- **Combination of *augmented Gibbs* and *symbolic Gibbs* samplers.** The Gibbs sampling based tools proposed in Chapters 4 and 5 tackle the problem of inference from very different view points. They can potentially be combined to create a sampler that benefits from the capabilities of both of them. Otherwise stated, the combined sampler may simultaneously use augmented Gibbs to reduce the number of target partitions that contribute to the formation of CDF functions required in each step of Gibbs sampling while these CDFs are already computed symbolically via symbolic Gibbs technique and it is only required that we instantiate them per sample. A major challenge in this way is as follows: In each step of the augmented Gibbs sampling, different combinations of auxiliary variables may be chosen for blocked sampling (and therefore collapsed). To combine augmented Gibbs and symbolic Gibbs, for any potential combination of collapsed (augmented) variables, a distinct symbolic CDF should be computed and stored. This rapidly becomes intractable and contradicts the reason augmented Gibbs sampling was introduced in the first place. A potential solution is to compute the symbolic CDFs lazily and as required. Since due to the topology of the neighboring regions many combinations of the auxiliary variables never collapse together, such a strategy may preserve the advantages of

both augmented and symbolic Gibbs samplers. Nonetheless, this prediction has to be verified by experimental results.

- **Expanding the application domain of symbolic Gibbs by utilizing the underlying techniques of CAS, ATP and SMT systems.** Symbolic Gibbs sampling can only by applied to a subset (PPF* $\subset$ PPF) of polynomial piecewise fractional functions which have properties that guarantee the existence of easy-to-compute closed-form integrals (see Section 5.4.3). By utilizing more advanced symbolic integration techniques of computer algebra systems (CASs) (see Section 3.4.1) in future we may be able to apply this sampler to a larger subset of PPF family. On the other hand, by using CAS polynomial factorization techniques, we may be able to simplify (or approximate) PPFs that are not originally in PPF* form into such forms. Finally, by using automated theorem proving (ATP) or satisfiability modulo theory (SMT) techniques (respectively introduced in Sections 3.4.2 and 3.4.3) we may be able to prune out unsatisfiable partitions and increase the performance of symbolic Gibbs sampling.

- **Representing piecewise functions with alternative data structures.** Throughout, the implementation of piecewise functions were based on extended algebraic decision diagram (XADD) data structure (introduced in Section 3.3.3) which is the simplest and the most natural choice. At the moment, it is unknown to us whether in the context of (approximate) probabilistic reasoning the use of alternative structures such as edge-valued decision diagrams (introduced in Section 3.3.4) is justified or not. Keeping in mind the latter family of diagrams (most notably, AADDs briefly introduced in Section 3.3.4.5) are capable of representing additive and multiplicative structures in compact forms, this issue is worth being investigated in future works. Nonetheless, AADDs and EVBDDs are only suitable for representation of discrete functions and the extension to (piecewise) continuous forms is non-obvious.

- **Inference conditioned on observed piecewise functions.** The framework introduced in Chapter 5 can be generalized to handle piecewise observed functions such as $\text{abs}(X)$ or $\max(X, Y)$. This is not emphasized in this thesis in detail and should be considered an avenue of further research. The basic idea can be illustrated by the following example.

  Consider the problem of taking samples from $p(x, y \mid z = 5)$ where

  $$p(z) = \delta(\max(x, y)) := \delta(\mathbb{I}[x \geq y] \cdot x + \mathbb{I}[x < y] \cdot y)$$

  (which is clearly piecewise.) It is not hard to show that

  $$p(x, y \mid z = 5) \propto \mathbb{I}[x \geq y] \cdot \delta(x = 5) \cdot p(X = x, Y = y) + \mathbb{I}[x < y] \cdot \delta(y = 5) \cdot p(X = x, Y = y) \quad (7.1)$$

  In this example, it can be seen than Dirac deltas cannot be eliminated via marginaliza-

tion of either x or y because each of these variables appears in a delta function. However, as we will see shortly, the problem of taking samples from $p(x, y \mid z = 5)$ in equation (7.1) can be converted to the problem of taking samples from $\hat{p}(v, w)$, defined as follows:

$$\hat{p}(v,w) := \mathbb{I}[v \geq w] \cdot \delta(v = 5) \cdot p(X = v, Y = w) + \mathbb{I}[w < v] \cdot \delta(v = 5) \cdot p(X = w, Y = v) \quad (7.2)$$

Formula (7.2) is created from (7.1) via substituting x and y by v and w in its first term while substituting them with w and v in the second term. Now, to eliminate Dirac deltas from (7.2), we simply marginalize v:

$$\begin{aligned}\hat{p}(w) &:= \mathbb{I}[5 \geq w] \cdot p(X = 5, Y = w) + \mathbb{I}[w < 5] \cdot p(X = w, Y = 5) \\ &= \mathbb{I}[w < 5] \cdot \big(p(X = 5, Y = w) + p(X = w, Y = 5)\big)\end{aligned} \quad (7.3)$$

Let $w^{(1)}$ be a sample taken from $\hat{p}(w)$. Using the technique introduced in Section 5.3.1.2 we reconstruct a sample for the original distribution. More specifically, with probability,

$$\frac{p(X = 5, Y = w^{(1)})}{p(X = 5, Y = w^{(1)}) + p(X = w^{(1)}, Y = 5)}$$

the sample is associated with the first term (therefore, $(x = 5, y = w^{(1)})$ is returned for the original density (7.1)) otherwise it is associated with the second term and $(x = w^{(1)}, y = 5)$ is the reconstructed sample for density (7.1).

The tricks used in this example can be generalized to a much larger scope.

- **Generalizing inference conditioned on observed functions.** In Chapter 5, we conditioned on functions of random variables. Such observations were considered to be the limit of measurements with normal noise where the noise variance tends to zero. The limiting processes had to be defined within the model via *deterministic potential* functions. This was necessary since as illustrated and argued in Section 5.3.2, different limiting processes can lead to different results. If the model designer is aware of the nature of measurements involved in the process of observation, such an assumption seems reasonable. On the other hand, in case the nature of measurement is unknown, it makes sense to define the collapsing mechanism in a way that it remains invariant w.r.t. the limiting process. To address similar problems, in Geometric measure theory, *k-dimensional Jacobian* is defined as the norm of the derivative matrix (i.e. matrix of partial derivatives a.k.a. *Jacobian matrix*) [Hubbard and Hubbard 2007; Diaconis et al. 2013] Due to lack of closed form integrals, this path of research cannot by combined with symbolic Gibbs. Nonetheless, its combination with HMC seems promising and may be pursued in future.

- **Reflective Hamiltonian Monte Carlo on models with nonlinear boundaries.** The volume preservation property proved in Chapter 6 only holds for the case

the boundaries of the model partitions are linear. Investigation of the validity of the volume conservation theorem for the case the model constraints are non-linear can be a future work. It might be the case that higher order boundaries behave like concave/convex mirrors and modify the volume via magnification of space. However, this is only a speculation that in future should be either verified or refuted.

7.3 Conclusion

This thesis highlight several applications of probabilistic inference on graphical models that are intrinsically piecewise continuous. Scalable, accurate inference in this important class of models has largely been overlooked prior to the research directions initiated in this thesis. Our contribution is to propose three sampling-based algorithms that are designed to conduct asymptotically unbiased inference on the aforementioned family of models. Our experimental results show that on piecewise target models, the proposed methods outperform the existing MCMC samplers. The thesis opens up many opportunities for further refinement and extension that should only further improve inference tools for this important but overlooked class of graphical models.

Bibliography

ANDRIEU, C., DOUCET, A., AND HOLENSTEIN, R. 2010. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 72*, 3, 269–342. (p. 107)

BAADER, F. 1992. Unification theory. *Springer*. (p. 66)

BACHMAIR, L., TIWARI, A., AND VIGNERON, L. 2003. Abstract congruence closure. *Journal of Automated Reasoning 31*, 2, 129–168. (p. 68)

BAHAR, R. I., FROHM, E. A., GAONA, C. M., HACHTEL, G. D., MACII, E., PARDO, A., AND SOMENZI, F. 1997. Algebric decision diagrams and their applications. *Formal methods in system design 10*, 2-3, 171–206. (pp. 55, 57)

BECKER, B., DRECHSLER, R., AND THEOBALD, M. 1997. On the expressive power of okfdds. In *Formal Methods Syst Des* (1997), pp. 5–21. (p. 61)

BECKER, B., DRECHSLER, R., AND WERCHNER, R. 1995. *On the relation between BDDs and FDDs*. Springer. (p. 61)

BISHOP, C. M. 2006. *Pattern recognition and machine learning*. Springer New York. (p. 18)

BOLLIG, B. AND WEGENER, I. 1996. Improving the variable ordering of OBDDs is NP-complete. In *IEEE Trans Comp* (1996), pp. 993–1002. (p. 57)

BOOLOS, G. S., BURGESS, J. P., AND JEFFREY, R. C. 2002. Computability and logic. *Cambridge University Press*. (pp. 65, 66)

BRONSTEIN, M. 1990. Integration of elementary functions. *Journal of Symbolic Computation 9*, 2, 117–173. (p. 64)

BROOKS, S., GELMAN, A., JONES, G., AND MENG, X.-L. 2011. *Handbook of Markov Chain Monte Carlo*. CRC press.

BRYANT, R. E. 1986. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on 100*, 8, 677–691. (p. 57)

BRYANT, R. E. AND CHEN, Y.-A. 1995. Verication of arithmetic functions with binary moment diagrams. In *Des Autom Conf* (1995), pp. 535–541. (p. 61)

BRYCHKOV, Y. A. 2008. Handbook of special functions: Derivatives, integrals, series and other formulas. *Chapman & Hall/CRC Press (English edition)*. (p. 63)

BUCHDAHL, H. A. 1993. *An introduction to Hamiltonian optics*. Courier Corporation. (p. 116)

BUTCHER, J. C. 2000. Numerical methods for ordinary differential equations in the 20th century. *Journal of Computational and Applied Mathematics 125*, 1, 1–29. (p. 43)

CANTOR, D. G. AND ZASSENHAUS, H. 1981. A new algorithm for factoring polynomials over finite fields. In *Mathematics of Computation* (1981), pp. 587–592. (p. 64)

CASELLA, G. AND GEORGE, E. I. 1992. Explaining the Gibbs sampler. *The American Statistician 46*, 3, 167–174. (pp. 11, 37)

CHEN, Y. AND BRYANT, R. 1997. *PHDD: an ecient graph representation for oating point circuit verication. In *Int Conf CAD* (1997), pp. 2–7. (p. 62)

CHERKASSKY, B. V. AND GOLDBERG, A. V. 1996. Negative-cycle detection algorithms. In *European Symposium on Algorithms* (1996), pp. 349–363. (p. 68)

CLARKE, E., FUJITA, M., MCGEER, P., MCMILLIAN, K., YANG, J., AND ZHAO, X. 1993. Multi-terminal binary decision diagrams: An efficient data structure for matrix representation. In *Workshop Logic Synth* (1993), pp. 1–15. (p. 57)

COBB, B. R. AND SHENOY, P. P. 2004. Hybrid influence diagrams using mixtures of truncated exponentials. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence* (2004), pp. 85–93. AUAI Press. (p. 9)

COBB, B. R., SHENOY, P. P., AND RUMÍ, R. 2006. Approximating probability density functions in hybrid bayesian networks with mixtures of truncated exponentials. *Statistics and Computing 16*, 3, 293–308. (p. 3)

COOPER, G. F. 1988. A method for using belief networks as influence diagrams. In *Proceedings of the 4th Workshop on Uncertainty in Artificial Intelligence (UAI)* (1988), pp. 55–63. (p. 9)

COX, D. A., LITTLE, J. B., AND O'SHEA, D. 1998. *Using algebraic geometry*. Springer New York. (p. 64)

DAS, S. AND MAGDON-ISMAIL, M. 2008. Adapting to a market shock: Optimal sequential market-making. In *NIPS* (2008), pp. 361–368. (p. 75)

DAVENPORT, J. H. 1981. On the integration of algebraic functions. *Springer*. (p. 64)

DAVIS, M., LOGEMANN, G., AND LOVELAND, D. 1962. A machine program for theorem-proving. *Communications of the ACM 5*, 7, 394–397. (p. 67)

DE MOURA, L. AND BJRNER, N. 2009. Satisfiability modulo theories: An appetizer. *Formal Methods: Foundations and Applications.*, 23–36. (p. 66)

DEVROYE, L. 1986. Sample-based non-uniform random variate generation. In *Proceedings of the 18th conference on Winter simulation* (1986), pp. 260–265. ACM. (p. 28)

DIACONIS, P., HOLMES, S., SHAHSHAHANI, M., ET AL. 2013. Sampling from a manifold. In *Advances in Modern Statistical Theory and Applications: A Festschrift in honor of Morris L. Eaton*, pp. 102–125. Institute of Mathematical Statistics.

DRECHSLER, R. AND BECKER, B. 1995. Dynamic minimization of OKFDDs. In *Int'l Conf. on Comp. Design* (1995), pp. 602–607. (p. 61)

DRECHSLER, R., BECKER, B., AND RUPPERTZ, S. 1996. K*bmds: a new data structure for verification. In *Eur Des Test Conf* (1996), pp. 2–8. (pp. 59, 62)

DRECHSLER, R. AND SIELING, D. 2001. Bdd in theory and practice. In *Software tools for technology transfer* (2001). (p. 59)

DUANE, S., KENNEDY, A. D., PENDLETON, B. J., AND ROWETH, D. 1987. Hybrid monte carlo. *Physics letters B 195*, 2, 216–222. (p. 39)

DUTERTRE, B. AND MOURA, L. D. 2006. A fast linear-arithmetic solver for DPLL (T). In *Computer Aided Verification* (2006), pp. 81–94. (p. 68)

GALLIER, J. H. 1985. *Logic for computer science: foundations of automatic theorem proving*. Harper & Row Publishers, Inc. (p. 65)

GEDDES, K. O., GLASSER, M. L., MOORE, R. A., AND SCOTT, T. C. 1990. Evaluation of classes of definite integrals involving elementary functions via differentiation of special functions. In *AAECC (Applicable Algebra in Engineering, Communication and Computing)*, Volume 1 (1990), pp. 149–165. (p. 64)

GEL'FAND, I. AND SHILOV, G. 1964. Generalized functions. vol. 1: Properties and operations, fizmatgiz, moscow, 1958. *English transl., Academic Press, New York*. (p. 96)

GILKS, W. R. 2005. *Markov chain Monte Carlo*. Encyclopedia of Biostatistics. 4. (p. 33)

GLEN, A. G., LEEMIS, L. M., AND DREW, J. H. 2004. Computing the distribution of the product of two continuous random variables. *Computational statistics & data analysis 44*, 3, 451–464. (p. 5)

GOSSEN, H. 1983. The laws of human relations and the rules of human action derived therefrom (rc blitz, trans.). (p. 141)

GRCAR, J. F. 2011. Mathematicians of gaussian elimination. In *Notices of the American Mathematical Society* (2011), pp. 782–792. (p. 64)

GREENWOOD, D. T. 1988. *Principles of dynamics*. Prentice-Hall Englewood Cliffs, NJ. (p. 116)

HALD, A. 2003. *A history of probability and statistics and their applications before 1750*, Volume 501. John Wiley & Sons. (p. 1)

HAMMERSLEY, J. M. AND HANDSCOMB, D. C. 1964. *Monte Carlo methods*, Volume 1. Methuen London. (p. 107)

HARRISON, J. R. 1996. Theorem proving with the real numbers. *PhD thesis, University of Cambridge Computer Laboratory*. (p. 63)

HASTINGS, W. K. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika 57*, 1, 97–109. (p. 34)

HOMAN, M. D. AND GELMAN, A. 2014. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *The Journal of Machine Learning Research 15*, 1, 1593–1623. (pp. 44, 116)

HOWARD, R. A. AND MATHESON, J. E. 2005. Influence diagrams. *Decision Analysis 2*, 3, 127–143. (p. 8)

HUBBARD, J. AND HUBBARD, B. 2007. Vector calculus. *Linear Algebra, and Differential Forms: A Unified Approach, Matrix Editions*. (p. 143)

JORDAN, M. I., GHAHRAMANI, Z., JAAKKOLA, T. S., AND SAUL, L. K. 1999. An introduction to variational methods for graphical models. *Machine learning 37*, 2, 183–233. (p. 45)

KEBSCHULL, U. AND ROSENSTIEL, W. 1993. Ecient graph-based computation and manipulation of functional decision diagrams. In *Eur Conf Des Autom* (1993), pp. 278–282. (p. 61)

KEENEY, R. L. AND RAIFFA, H. 1993. *Decisions with multiple objectives: preferences and value trade-offs*. Cambridge University Press. (p. 73)

KIKUCHI, R. 1951. A theory of cooperative phenomena. *Physical review 81*, 6, 988. (p. 45)

KOLLER, D. AND FRIEDMAN, N. 2009. *Probabilistic graphical models: principles and techniques*. The MIT Press. (p. 24)

KOLMOGOROV, A. N. 1950. Foundations of the theory of probability. (pp. 93, 98)

LAHIRI, S. K. AND SESHIA, S. A. 2004. The uclid decision procedure. In *Computer Aided Verification* (2004), pp. 475–478. Springer. (p. 67)

LAI, Y. T. AND SASTRY, S. 1992. Edge-valued binary decision diagrams for multi-level hierarchical verification. In *Des Autom Conf* (1992), pp. 608–613. (p. 59)

LEE, C. Y. 1959. Representation of switching circuits by binary-decision programs. *Bell Systems Technical Journal 38*, 985–999. (p. 55)

LETZ, R., SCHUMANN, J., BAYERL, S., AND BIBEL, W. 1992. SETHEO: A high-performance theorem prover. *Journal of Automated Reasoning 8*, 2, 183–212. (p. 64)

LUNN, D., SPIEGELHALTER, D., THOMAS, A., AND BEST, N. 2009. The BUGS project: Evolution, critique and future directions. *Statistics in medicine 28*, 25, 3049–3067. (p. 108)

MACLAURIN, D. AND ADAMS, R. P. 2014. Firefly Monte Carlo: Exact MCMC with subsets of data. In *Thirtieth Conference on Uncertainty in Artificial Intelligence (UAI)* (07/2014 2014), pp. 543–552. (p. 78)

MANNA, Z. AND ZARBA, C. G. 2003. Combining decision procedures. In *Formal Methods at the Cross Roads: From Panacea to Foundational Support, Lecture Notes in Computer Science* (2003), pp. 381–422. Springer. (p. 66)

MCELIECE, R. J., MACKAY, D. J. C., AND CHENG, J.-F. 1998. Turbo decoding as an instance of pearl's belief propagation algorithm. *Selected Areas in Communications, IEEE Journal on 16*, 2, 140–152. (p. 45)

MINKA, T. 2001. Expectation propagation for approximate bayesian inference. In *Uncertainty in Artificial Intelligence*, Volume 17 (2001), pp. 362–369. (p. 45)

MURPHY, K. 2001. An introduction to graphical models. *Rap. tech.* (p. 45)

NEAL, R. M. 2011. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo 2*. (pp. 11, 13, 40, 41, 43, 116, 117, 121, 127)

P. TAFERTSHOFER, M. P. 1997. Factored edge-valued binary decision diagrams. In *Formal Meth Syst Des* (1997), pp. 243–270. (p. 61)

PAKMAN, A. AND PANINSKI, L. 2013. Auxiliary-variable exact hamiltonian monte carlo samplers for binary distributions. In *Advances in Neural Information Processing Systems* (2013), pp. 2490–2498. (pp. 44, 116)

PAKMAN, A. AND PANINSKI, L. 2014. Exact hamiltonian monte carlo for truncated multivariate gaussians. *Journal of Computational and Graphical Statistics 23*, 2, 518–542. (pp. 116, 119)

PATIL, A., HUARD, D., AND FONNESBECK, C. J. 2010. PyMC: Bayesian stochastic modelling in Python. *Journal of statistical software 35*, 4, 1. (p. 108)

PEARL, J. 1987. Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence 32*, 2, 245–257. (p. 107)

POON, H. AND DOMINGOS, P. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proceedings of the 21st Conference on Artificial Intelligence (AAAI)*, Volume 6 (2006), pp. 458–463. (p. 81)

RIAZANOV, A. AND VORONKOV, A. 2002. The design and implementation of VAMPIRE. *AI communications 15*, 2, 3, 91–110. (p. 64)

ROBERTS, G. O., GELMAN, A., AND GILKS, W. R. 1997. Weak convergence and optimal scaling of random walk metropolis algorithms. *The Annals of Applied Probability 7*, 1, 110–120. (p. 86)

ROBERTS, G. O., GELMAN, A., GILKS, W. R., ET AL. 1997. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The annals of applied probability 7*, 1, 110–120. (pp. 36, 107, 108, 127)

ROBINSON, J. A. 1965. A machine-oriented logic based on the resolution principle. *Journal of the ACM (JACM) 12*, 1, 23–41. (p. 65)

RUSSELL, S. J. AND NORVIG, P. 2003. *Artificial intelligence: a modern approach*. Prentice hall. (p. 22)

SAMUELSON, P. A. 1966. *Foundations of Economic Analysis: Harvard Economic Studies*. Harvard University Press. (p. 141)

SANNER, S. AND ABBASNEJAD, E. 2012. Symbolic variable elimination for discrete and continuous graphical models. In *Proceedings of the 26th Conference on Artificial Intelligence (AAAI)* (2012), pp. 1954–1960. (pp. 4, 52, 90)

SANNER, S., DELGADO, K. V., AND DE BARROS, L. N. 2011. Symbolic dynamic programming for discrete and continuous state MDPs. In *UAI-2011* (2011). (pp. 7, 49, 55, 58)

SANNER, S. AND MCALLESTER, D. 2005. Affine algebraic decision diagrams (AADDs) and their application to structured probabilistic inference. In *Proceedings of the 19th International Joint Conference on AI (IJCAI-05)* (2005). (pp. 58, 62)

SCHULZ, S. 2002. E-a brainiac theorem prover. *AI Communications 15*, 2, 111–126. (p. 64)

SHEINI, H. M. AND SAKALLAH, K. A. 2006. From propositional satisfiability to satisfiability modulo theories. In *Theory and Applications of Satisfiability Testing-SAT 2006*, pp. 1–9. Springer. (p. 67)

SHENOY, P. P. 2012. Two issues in using mixtures of polynomials for inference in hybrid Bayesian networks. *International Journal of Approximate Reasoning 53*, 5, 847–866. (pp. 4, 90)

SHENOY, P. P. AND WEST, J. C. 2011. Inference in hybrid Bayesian networks using mixtures of polynomials. *International Journal of Approximate Reasoning 52*, 5, 641–657. (pp. 4, 90, 100)

SRINIVASAN, A., KAM, T., MALIK, S., AND BRAYTON, R. 1990. Algorithms for discrete function manipulation. In *Proc. Int. Conf. CAD* (1990), pp. 92–95. (p. 57)

STAN DEVELOPMENT TEAM. 2014. *Stan Modeling Language Users Guide and Reference Manual, Version 2.5.0*. (pp. 13, 107, 108, 116)

STEEDMAN, I. 2003. *Consumption takes time: implications for economic theory*. Routledge. (p. 141)

STICKEL, M. E. 1981. A unification algorithm for associative-commutative functions. In *Journal of the ACM (JACM)* (1981), pp. 423–434. (p. 66)

SWENDSEN, R. H. AND WANG, J.-S. 1987. Nonuniversal critical dynamics in Monte Carlo simulations. *Physical Review Letters 58*, 2, 86–88. (p. 78)

TROELSTRA, A. S. AND SCHWICHTENBERG, H. 2000. *Basic proof theory*, Volume 43. Cambridge University Press. (p. 65)

VAJDA, S. 2014. *Probabilistic programming*, Volume 9. Academic Press. (p. 9)

WAINWRIGHT, M. J. AND JORDAN, M. I. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning 1*, 1-2, 1–305. (p. 45)

WASSERMAN, L. 2013. *All of statistics: a concise course in statistical inference*. Springer Science & Business Media. (p. 34)

WEIDENBACH, C., DIMOVA, D., FIETZKE, A., KUMAR, R., SUDA, M., AND WISCHNEWSKI, P. 2009. Spass version 3.5. In *Automated Deduction–CADE-22*, pp. 140–145. Springer. (p. 64)

WINGATE, D., STUHLMUELLER, A., AND GOODMAN, N. D. 2011. Lightweight implementations of probabilistic programming languages via transformational compilation. In *International Conference on Artificial Intelligence and Statistics* (2011), pp. 770–778. (p. 107)

WOOD, F., VAN DE MEENT, J. W., AND MANSINGHKA, V. 2014. A new approach to probabilistic programming inference. In *Proceedings of the 17th International conference on Artificial Intelligence and Statistics* (2014). (pp. 107, 108)

YEDIDIA, J. S. 2011. Message-passing algorithms for inference and optimization. *Journal of Statistical Physics*. (pp. 22, 26)

YEDIDIA, J. S., FREEMAN, W. T., WEISS, Y., ET AL. 2000. Generalized belief propagation. In *NIPS*, Volume 13 (2000), pp. 689–695. (p. 45)

ZHANG, N. L. 1998. Probabilistic inference in influence diagrams. *Computational Intelligence 14*, 4, 475–497. (p. 9)