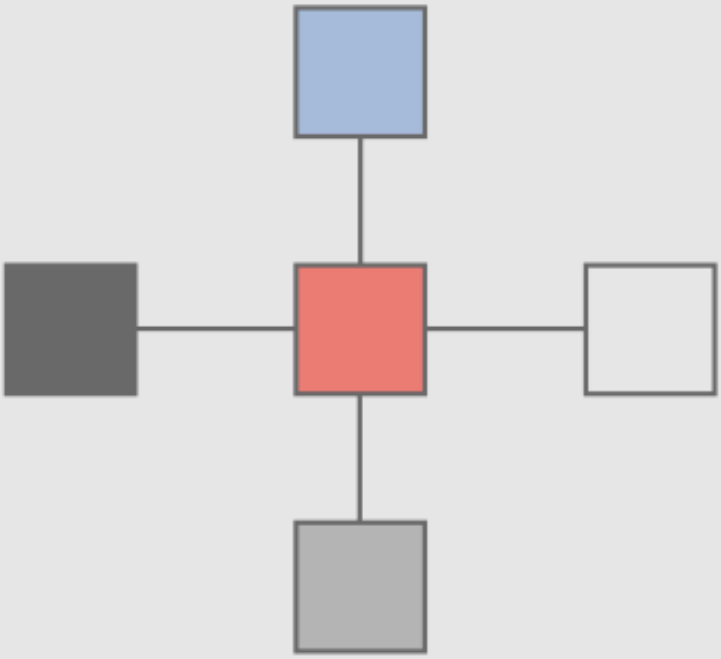
Towards Universal Probabilistic Programming with Message Passing on Factor Graphs



Towards Universal Probabilistic Programming with Message Passing on Factor Graphs

Semih Akbayrak

Towards Universal Probabilistic Programming with Message Passing on Factor Graphs

Semih Akbayrak

A catalogue record is available from the Eindhoven University of Technology Library
ISBN: 978-90-386-5638-0

Keywords: Approximate Bayesian Inference, Forney-style Factor Graphs, Message Passing, Monte Carlo Methods, Probabilistic Programming, Stochastic Optimization, Variational Inference

The research in this dissertation has been partly funded by GN Hearing, Advanced Science department.

LaTeX style template provided by Joos Buijs

Towards Universal Probabilistic Programming with Message Passing on Factor Graphs

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de rector magnificus prof.dr.ir. F.P.T. Baaijens, voor een commissie aangewezen door het College voor Promoties, in het openbaar te verdedigen op vrijdag 20 januari 2023 om 13:30 uur

door

Semih Akbayrak

geboren te Kadikoy, Turkije

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	prof.dr.ir. M.J. Bentum
1e promotor:	prof.dr.ir. A. de Vries
co-promotor:	dr. T. W. van de Laar
leden:	prof.dr. F. A. N. Palmieri (Università degli Studi della Campania Luigi Vanvitelli)
	prof.dr. P. Rostalski (University of Lübeck)
	dr. S. Särkkä (Aalto University)
	dr.habil. C. P. de Campos

Het onderzoek dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

Summary

Towards Universal Probabilistic Programming with Message Passing on Factor Graphs

This thesis presents efficient and automated probabilistic inference algorithms for intelligent system design. An intelligent system is a decision-making agent that can take reasonable and reliable actions in uncertain environments. Probability theory and Bayesian inference constitute the theoretical framework for intelligent system design. Hence, probabilistic modeling and inference are of great importance in building intelligent agents and applications. Modeling a real-world phenomenon with the language of probability theory is often intuitive, and hence it can be carried out by experts in the application field. In contrast, the inference part often necessitates expertise in Bayesian statistics and precludes experts and scientists from diverse application fields from utilizing probabilistic modeling.

This thesis focuses on automating Bayesian inference procedures to make probabilistic modeling more accessible for non-experts of Bayesian statistics. The works presented in this thesis can be subsumed under the umbrella of probabilistic programming. Probabilistic programming is a programming paradigm that automates inference procedures in probabilistic models. Unlike most of the existing probabilistic programming tools, which follow Monte Carlo sampling approaches and stochastic optimization for universality, the algorithms in this thesis strive to utilize deterministic message passing approaches at the utmost level. Deterministic message passing algorithms are often faster than Monte Carlo algorithms and hence preferable in real-time applications, e.g., robotics.

Deterministic message passing algorithms are fast and efficient, but automating them in a universal way is more challenging compared to Monte Carlo algorithms. The main reason is that, unlike Monte Carlo algorithms, which are simulation-based approaches, message passing algorithms require custom analytical rules for the components of probabilistic models. The algorithms in this thesis aim to cir-

cumvent this issue by hybrid approaches that combine the best of Monte Carlo and message passing approaches.

In general, the contributions of this thesis are twofold. First, we present a generic framework to incorporate Monte Carlo or other approximation approaches like Laplace approximation into message passing procedures by proposing fully automated hybrid algorithms. Next, we demonstrate how to cast some well-recognized stochastic optimization-based methods as probabilistic programming algorithms with message passing on factor graphs. The resulting methods realize distributed probabilistic inference by sticking to deterministic approaches if possible and resorting to Monte Carlo, Laplace, or stochastic optimization approaches only when necessary.

This thesis provides a generic framework for hybrid, efficient and automatable Bayesian inference on factor graphs. The presented algorithms achieve hybrid inference by interfacing Monte Carlo, Laplace, and stochastic optimization approaches with deterministic message passing algorithms.

Contents

List of Symbols

Mathematical notation

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	A graph with a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$
a, b, c, d	Indices associated with nodes
i, j, k, l	Indices associated with edges
t	Discrete time index
m_{ai}	A message from node a propagating on edge i
m_{ia}	A message towards node a propagating on edge i
$\mathbf{z}$	A set of random variables
$\mathbf{z}_a$	A subset of random variables argument to node a
$\mathbf{z}_{a\setminus j}$	A subset of random variables argument to node a excluding z_j
z_j	A random variable associated with edge j
$\mathcal{F}$	(Variational) Free Energy
$\mathcal{H}$	(Differential) Entropy
$\mathcal{U}$	(Average) Energy
$\mathbb{E}$	Expectation
$\mathbb{V}$	Variance (Covariance)
f	A factor function
p	A probability distribution
q	A variational posterior distribution

Probability distributions

$\delta(y-\mu)$	Dirac delta centered on μ
$\mathcal{N}(\mu, v)$	Gaussian distribution with mean μ and variance v
$\mathcal{G}a(a, b)$	Gamma distribution with shape a and rate b
$\mathcal{P}o(\lambda)$	Poisson distribution with rate λ
$\mathcal{D}\mathrm{ir}(\alpha)$	Dirichlet distribution with concentration α
$\mathcal{W}_d(V, n)$	Wishart distribution with $d \times d$ scale V and n degrees of freedom
$\mathcal{B}er(\pi)$	Bernoulli distribution with event probability π
$\mathcal{C}\mathrm{at}(\pi)$	Categorical distribution with event probability vector π
$\mathrm{T}\mathcal{N}(\mu, v, a, b)$	Truncated Gaussian with mean μ, variance v and support $[a, b]$
$\mathcal{S}\mathrm{t}(\mu, \sigma, n)$	Student's t with location μ, scale σ and degrees of freedom n

Abbreviations

ADVI	Automatic Differentiation Variational Inference
AIS-MP	Adaptive Importance Sampling Message Passing
BBVI	Black-Box Variational Inference
BP	Belief Propagation
CVI	Conjugate-computation Variational Inference
DGM	Directed Graphical Model
EF	Exponential Family
ELBO	Evidence Lower Bound
EP	Expectation Propagation
EVMP	Extended Variational Message Passing
FE	Free Energy
FFG	Forney-style Factor Graph
HGF	Hierachical Gaussian Filter
HMC	Hamiltonian Monte Carlo
IS	Importance Sampling
L-BFGS	Limited Memory BFGS (Broyden–Fletcher–Goldfarb–Shanno)
LDS	Linear Dynamical System
LWS	List of Weighted Samples
MC-CAVI	Monte Carlo Coordinate Ascent Variational Inference
MCMC	Markov Chain Monte Carlo
NC-VMP	Non-conjugate Variational Message Passing
NEF	Non-standard Exponential Family
NGD	Natural Gradient Descent
NUTS	No-U-Turn Sampler
PGM	Probabilistic Graphical Model
PP	Probabilistic Programming
PPL	Probabilistic Programming Language
PVI	Particle Variational Inference
SG-PMC	Stochastic Gradient Population Monte Carlo
SMC	Sequential Monte Carlo
SSSM	Switching State Space Model
SVMP	Stochastic Variational Message Passing
SVI	Stochastic Variational Inference
VI	Variational Inference
VMP	Variational Message Passing

Chapter 1

General Introduction

1.1 Motivation

The world has been witnessing immense advances in science and technology since the establishment of the scientific method. From the electromagnetic radiation theory to the laws of thermodynamics, a considerable amount of these advances relate to mathematical modeling. Mathematical modeling is the projection of real-world processes to the realm of mathematics, where we can perform analyzes, run simulations and make predictions. Mathematical modeling is examined under two categories: deterministic modeling and probabilistic modeling. We get familiar with deterministic models in the early years of our education. However, we all probably remember our teachers were warning us that these models and governing equations are valid under some strict assumptions in perfect conditions. Indeed, deterministic models provide us with the fundamental theories required to understand real-world processes better. However, they are too delicate to be useful in many real-world applications. This is because real-world applications are subject to multifarious imperfections and outliers, which are too diverse to be fully addressed by manually derived deterministic processes.

An alternative approach to manually developing deterministic models is machine learning. Data-driven machine learning algorithms aim to build deterministic models by recognizing the patterns in vast amounts of data. As data comes directly from real-world, e.g., self driving car sensors [1], social networks [2], e-commerce [3], online entertainment platforms [4], scientific experiments [5], etc., it is assumed that data consist of the imperfections that can occur in the real world and hence can be learned during the training phase. Although it has been proven useful in many practical application areas, data-driven machine learning algorithms are notorious for being data-hungry methods. Even if the machine learning algorithm is trained with vast amounts of data, it is still quite likely to encounter new types of imperfections and outliers in action since the world is a complex environ-

ment full of surprises [6, Section 3.1.4].

In the other branch of mathematical modeling, namely probabilistic modeling, we tend to model real-world processes with the means of probability theory as it provides us with a formal language for plausible reasoning under uncertainty [7]. In probabilistic modeling [8, 9], we work with generative models consisting of random variables and functions of the random variables. A generative model is a probabilistic description of our beliefs and assumptions about the true data generating process. Generally, a generative model is specified by two functions: a sampling distribution of the data, given the latent variables, and a prior distribution on the latent variables. Once observations are substituted in the sampling distribution, this function turns into a likelihood function for the latent variables. Given prior and conditional distributions along with data, probability theory describes how to reason rationally by manipulating probability distributions through the sum and product rules [10]. This process of updating probability distributions by sum and product rules when new information becomes available is known as Bayesian inference.

Probabilistic modeling has unprecedentedly changed our lives over the last couple of decades. For example, probabilistic interpretation of matrix factorization methods have led to practical recommendation systems [11, 12] and efficient signal processing algorithms [13, 14]; probabilistic topic models [15–17] have analyzed large corpora of digitized text and compiled information for us; Bayesian time series models [18, 19] have yielded successful smart applications in a wide variety of areas ranging from finance [20–22] to audio processing [23–25]; probabilistic relational learning models have adaptively detected the clusters in networks [26] and allowed nodes to be a member of multiple clusters as in the real-world [27].

Probabilistic modeling has transformed the robotics [28], control [29,30], planning [31,32] and decision making fields [33] as well. From self-driving cars [34] to cleaning robots [35], intelligent agents that are equipped with probabilistic models have been becoming an indispensable part of our lives. But, what is it that makes probabilistic modeling and Bayesian inference so unique in intelligent system design?

The world is a complex environment in which living organisms, as the agents of this environment, interact with each other and surrounding non-living objects. Succeeding in such a complex environment as an intelligent agent requires *more or less* reliable and reasonable decision-making abilities. The adverb *more or less* is intentionally written to account for humankind's biases in its decision-making mechanism under uncertainty [36]. An inevitable part of complex environments as the agents act not in an entirely predictable manner; uncertainty needs to be taken into account while making decisions and taking actions. Considering that there are other sources of uncertainty such as perceptual uncertainties [37] and uncertainties due to inductive biases [38] as well, it becomes even more pivotal to quantify uncertainty in intelligent agent design. At this point, probabilistic modeling and

Bayesian inference come to the rescue by providing us with the tools to represent our beliefs on random variables with probability distributions that inherently quantify uncertainty.

So far, we have endorsed probabilistic modeling and Bayesian inference, but the reader is probably wondering: where is the catch?

The catch is that Bayesian inference is only computationally straightforward for carefully designed, simple models. However, many real-world processes require complex functional dependencies between random variables with structured model specifications. While designing models, we are supposed to keep the inference phase in the corner of our minds. Therefore, conventional probabilistic modeling necessitates expertise in Bayesian inference by precluding the experts in their own fields from fully utilizing the probabilistic modeling paradigm.

A popular subfield of machine learning, deep learning [39], has passed through a similar path. The training phase in deep learning models is predominantly performed by an algorithm called backpropagation [40]. Implementation of the backpropagation algorithm is tedious as it requires the calculation of the partial derivatives with the chain rule formula. Recent advances in automatic differentiation tools [41] gave birth to deep learning packages and libraries [42–46], and facilitated the research and engineering activities in the deep learning field. Nowadays, hardly a day goes by without news of a breakthrough deep learning achievement. Inspired by the impact of deep learning packages on the deep learning field, a new programming paradigm called *probabilistic programming* strives to make a similar impact on probabilistic modeling.

1.1.1 Probabilistic Programming

The term Probabilistic Programming Language(s) (PPL) is an umbrella term that refers to programming languages, libraries, and packages to support the automation of inference procedures in probabilistic models [47]. PPLs aim to allow end-users to focus only on (probabilistic) model specification without worrying about the inference phase. Below we list some desired functionalities for a PPL:

- **Precision:** Precision in inference can be vitally important as probabilistic models are often incorporated into decision-making systems in real-world applications.
- **Performance metric:** An end-user should be able to track the performance of the model and the inference process, for model comparison and diagnosis.
- **Speed:** A self-driving car should be able to make decisions right away; an algo-trader bot must exploit the opportunities in the market before the opportunities disappear. Therefore inference procedures are desired to be fast.

- **Universality:** A PPL must support inference in a broad range of probabilistic model types to let end-users fully reflect their domain knowledge in the model specification phase.
- **Scalability:** In the era of big data, PPLs should possess inference engines that scale to terabytes of data.
- **Full automation:** Inference algorithms must be hyperparameter-free to the furthest extent to relieve end-users from hyperparameter selection.

In this dissertation, we study two categories of Bayesian inference algorithms: deterministic methods and stochastic methods. We shall talk about deterministic methods in a short while, but let us first introduce stochastic methods. The most popular Bayesian inference algorithms for probabilistic programming fall under the category of stochastic methods. Among them are Hamiltonian Monte Carlo (HMC) [48], No-U-Turn Sampler (NUTS) [49], Automatic Differentiation Variational Inference (ADVI) [50] and Black-Box Variational Inference (BBVI) [51]. Whereas HMC and NUTS are pure instances of Monte Carlo methods, ADVI and BBVI are instances of variational methods sticking to the Monte Carlo sampling procedure. Monte Carlo methods and sampling procedures [52] are popular in probabilistic programming because they yield broadly applicable Bayesian inference engines. However, loosely speaking, generic purpose Monte Carlo-based algorithms can be prohibitively slow to be practical for real-world applications, especially in state-space and time series models that are often used in robotics, control, and decision-making fields. Moreover, generic purpose Monte Carlo-based algorithms perform approximate inference even if the model allows for analytical marginalizations. Recent works in the probabilistic programming field [53–55] strive to alleviate these shortcomings of generic purpose Monte Carlo-based algorithms by exploring and exploiting the conjugacy structures in probabilistic models. In this dissertation, we focus on an alternative probabilistic programming formulation that is based on message passing inference on Forney-style Factor Graphs (FFGs) [56, 57].

1.1.2 Forney-style Factor Graphs and Message Passing

We briefly mention deterministic inference algorithms and Probabilistic Graphical Models (PGMs). PGMs are visualization tools that depict the (in)dependency structures between random variables in probabilistic models [58, 59]. The most popular PGM types are Directed Graphical Models (DGMs), representing random variables with nodes in the graph. In DGMs, random variables are connected through directed edges, representing the direction of the generative model. Learning DGM structures from data is an active research area in the PGMs field [60, 61]. In this dissertation, however, we focus on the automation of Bayesian inference algorithms

by means of message passing on fixed factor graph structures. Factor graphs are another class of PGMs that visualize the functions of random variables with factor nodes alongside the random variables. Factor graphs are particularly devised to reflect the factorizations in factor graphs. In signal processing applications, Forney-style Factor Graphs (FFGs) [56, 57] are widely used, and we also work with FFGs in this manuscript to particularly facilitate the design processes of probabilistic signal processing algorithms [62]. In an FFG, factor nodes relate to the factorized function of random variables, while edges that connect the factor nodes to each other are associated with the random variables in the probabilistic model specification. Since an edge can be maximally connected to two factor nodes, FFGs employ equality nodes to branch out the random variables. For example, consider an FFG visualized in Figure 1.1. This FFG is a graphical representation of the following factorized function

$$\underbrace{p(z, y_1, y_2, y_3)}_{f(z,y_1,y_2,y_3)} = \underbrace{p(z)}_{f_a(z)} \prod_{i=1}^{3} \underbrace{p(y_i|z)}_{f_b(y_i,z)} .$$

Factor graphs differ from DGMs in that DGMs are meant to be visualization tools while factor graphs are computational graphs. Let us detail what we mean by the computational graphs.

In FFGs, Bayesian inference is customarily performed through distributed, deterministic inference algorithms that are termed message passing algorithms. Belief Propagation (BP) [63, 64], Variational Message Passing (VMP) [65, 66] and Expectation Propagation (EP) [67] are well known examples of deterministic message passing-based Bayesian inference algorithms. VMP and EP are approximate inference algorithms similar to the aforementioned Monte Carlo-based methods. On the other hand, BP is an exact inference algorithm in tree-like FFGs. In loopy graphs, however, running BP in a loopy manner also leads to an approximate inference procedure [68]. It has been shown in previous works [69–71] that the message passing algorithms BP, VMP, and EP refer to a constraint optimization of an objective functional called (variational) free energy. Changing the constraints on approximating distributions in the free energy optimization and deriving the stationary point equations analytically, one can recover different known or novel message passing algorithms [69]. Hence, message passing algorithms are almost hyperparameter-free, except for the number of iterations in iterative settings[1], given that the inference steps are calculated in closed form. As the message passing algorithms implicitly optimize the free energy functional, the free energy inherently arises as a performance metric for the message passing algorithms. Therefore, we will be working with min-

[1] Assuming a default automation process that initializes beliefs with specified or noninformative priors and updates the beliefs following the graph structure. In Loopy BP, the initial messages are customarily set equal to 1 [70].

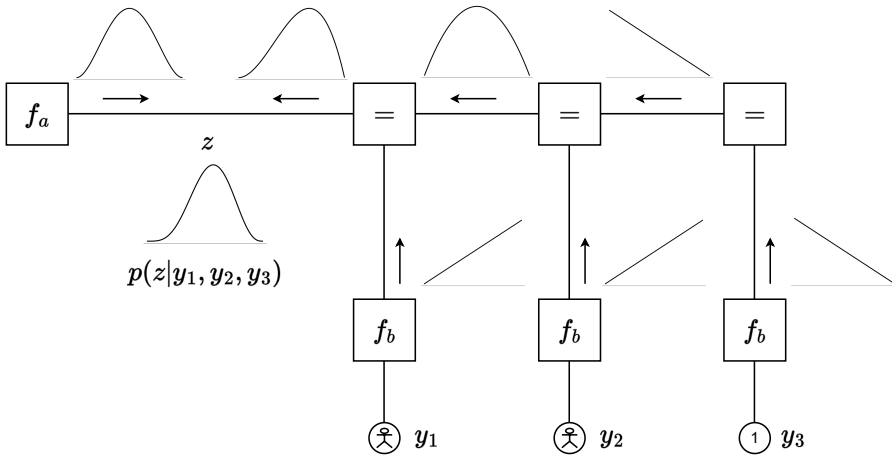


Figure 1.1: A Forney-style Factor Graph (FFG) representation for a coin tossing model. Messages propagate towards the edge associated with the random variable z and collide to generate the posterior distribution $p(z|y_1, y_2, y_3)$.

imization of the free energy functional as the computational objective throughout the dissertation.

In comparison to the stochastic methods for probabilistic programming, message passing algorithms are often faster, especially in signal processing applications such as state space model variants. Moreover, in contrast to the stochastic methods for probabilistic programming, message passing algorithms are able to exploit the factorizations and conjugacy relations in probabilistic models to perform analytical calculations.

Message passing-based inference has an elegant interpretation in both the machine learning [64, 72] and the computational neuroscience [73] literature. In this interpretation, factor nodes collect messages propagating on the edges that are connected to themselves. The messages inform the factor nodes about the rest of the graph. As computational units of FFGs, factor nodes then calculate outgoing messages using incoming messages and the functional relationship between the random variables defined on the nodes. When two messages meet on the edge, they collide and give us the marginal belief of the variable that is associated with the edge. Hence, collision refers to the product operation followed by normalization.

Consider once again the FFG given in Figure 1.1. This FFG depicts a coin-tossing experiment with three tosses. The probability of heads is denoted by the random variable z, which is shared across tossing events through equality nodes as the coin is the same in all three tossing events. The messages coming to the variable

z carry the information about the rest of the graph. The message on the left is customarily called the forward message as it follows the direction of the sampling distribution and carries our prior belief on z. The message on the right, which we call the backward message, is the collective information coming from tossing events. As customary in the message passing framework, the messages carry probability distributions.

In FFGs, factor nodes are locally isolated computational units, given the incoming messages. This local feature of FFGs is useful in probabilistic modeling as it allows us to build a library of reusable components. Having designed a set of factor nodes with message passing rules for a model, we can employ these nodes over and over again in different model specifications. Therefore, the FFG framework is time-saving for personal usage, but the question is if can we develop a universal PPL that renders inference in a broad range of probabilistic model types based on the plug-in-type architecture of FFGs for common usage?

Method / Criteria	Message Passing	Monte Carlo	Stochastic VI
Precision			
Performance metric			
Speed	✓	✗	✗
Universal	✗	✓	✓
Scalable	✗	✗	✓
Full automation			

Table 1.1: A superficial comparison table of message passing, Monte Carlo (like HMC and NUTS) and stochastic VI (like BBVI and ADVI) algorithms for probabilistic programming. We want to combine the best of the algorithms in a unified framework built upon FFGs.

Message passing algorithms are fast and seamlessly interface with the plug-in-type architecture of FFGs, but they require manually derived inference rules. Therefore, message passing algorithms are hard to generalize and violate the universality criteria for PPLs. In contrast, stochastic methods are broadly applicable, but all are approximate and often run slowly. A comparison of the three most widely used probabilistic programming algorithm classes is given in Table 1.1. This comparison superficially reflects the characteristics of the inference classes, and of course, there are attempts to mitigate these shortcomings, e.g., [74] aims to address scalability issues in Monte Carlo methods. Nevertheless, we focus on the conventional impressions of these three inference classes. In addition, though not widely employed in the probabilistic programming context, deterministic approximation methods such as Laplace approximation [75, Section 4.4] [6, Section 7.4.3] [76, Section 28.2] and sigma-point methods [77] [19, Chapter 6] form an alternative class of infer-

ence methods. These methods are not as universal as stochastic methods, yet they are flexible to carry out inference in those models that are challenging for the conventional message passing algorithms.

In Table 1.1, we abstain from assessing the three popular probabilistic programming inference classes in terms of precision, performance metric, and full automation, for now, to avoid reducing the discussion to a rather controversial *present - absent* sort of classification. Nonetheless, we mentioned that the free energy objective is a natural candidate to be a performance metric in message passing-based PPLs. We also bridged a connection between the constraint free energy optimization by deterministic message passing algorithms and hyperparameter-free Bayesian inference in probabilistic models. In fact, free energy forms the objective to be minimized in the aforementioned Monte Carlo-based variational inference algorithms, too. Oftentimes, they also use factorized distribution families to approximate posterior distributions, similar to VMP. Whereas the minimization of the free energy in VMP is based on the analytical execution of the coordinate descent, Monte Carlo-based variational inference algorithms employ free energy gradient estimates in stochastic gradient descent, and hence they are often referred to with the generic term stochastic variational inference (vi) algorithms. This stochastic optimization setting allows stochastic vi algorithms to scale almost universally [78, 79]. However, the costs of the scalability and universality features for stochastic vi algorithms are slower convergence and more hyperparameter-dependent inference procedures compared to the message passing algorithms. Throughout this dissertation, we shall also build a more detailed intuition about the nature of the inference algorithms in terms of precision, performance metric, and full automation criteria.

We summarize the discussion so far:

- Probabilistic programming (PP) is a programming paradigm that aims to make probabilistic modelling accessible to end-users with varying degrees of background in Bayesian inference and probabilistic modeling.

- Monte Carlo-based probabilistic programming algorithms have gained a reputation for their **generality**, i.e., they are very broadly applicable. Therefore, the bulk of the inference engines in PPLs are comprised of Monte Carlo-based probabilistic programming algorithms. However, they are slow to be employed in certain model specifications, especially the ones concerning the signal processing applications such as state space model variants and time series models.

- Message passing algorithms are **efficient**. By efficiency, we imply that (i) message passing algorithms are fast; (ii) given the inference steps are analytically executed, message passing inference steps refer to the stationary point equations of the free energy manipulated by constraint specifications [69]. We also briefly discussed why message passing algorithms can be claimed to be

hyperparameter-free algorithms except for the number of iterations. Unfortunately, many probabilistic models of interest do not allow analytical calculations; hence, message passing can not be applied in closed form. Furthermore, it is not viable to register message passing rules manually such that the inference can be executed universally.

- There are alternative deterministic approximation methods such as Laplace approximation, which are not widely used in probabilistic programming. Laplace approximation is not entirely universal. Nevertheless, it is **flexible** to be employed in the approximation of the marginals for real-valued variables.
- FFGs constitute a unified framework for probabilistic modeling with message passing algorithms. As to be demonstrated, the plug-in-type architecture of FFGs is not only useful to automate message passing algorithms but also to formalize *hybrid* and *black-box* inference procedures.

1.2 Research Questions

We have discussed Monte Carlo simulation-based and message passing-based inference algorithms by emphasizing that the former is more broadly applicable while the latter is faster in general. Based on the foregoing considerations, the main theme of this thesis can be described as follows:

> In this thesis, we are in pursuit of a PPL framework that combines the advantages of both sampling-based and message passing-based inference without taking over the downsides of both frameworks.

In particular, this work is driven by the following concrete research questions:

RQ1. *How can we automatically execute Bayesian inference in an efficient manner?*

FFGs have been formalized as a PPL framework in [62, 80] and implemented as a Julia language package, called *ForneyLab.jl*. ForneyLab employs the FFG formalism of the standard message passing algorithms such as BP [57, 72], VMP [66], and EP [81] to execute Bayesian inference, efficiently. To be able to build a universal PPL framework that is fast and efficient, we first need to review FFGs and standard message passing algorithms. Then we can assess the standard message passing algorithms on FFGs regarding the universality criteria. Can we combine the factor nodes seamlessly? Deterministic message passing update rules need to be derived for each and every new factor node. Does this not impede the design of a universal PPL built upon the FFG formalism?

RQ2. *How can we automatically combine the generality of Monte Carlo sampling and flexibility of Laplace approximation with the efficiency of message passing algorithms?*

Bayesian inference by means of message passing is a distributed operation that exploits the conditional independence relations between variables in the FFG. How about foregoing a subset of analytic operations complicating the overall inference procedure in favor of more generally applicable approximation methods, e.g., importance sampling and Laplace approximation? How can we make sure that importance sampling and Laplace approximation are automated in the PPL and functioning in concordance with the message passing procedures?

RQ3. *Having developed a hybrid Monte Carlo - message passing inference procedure, is it possible to improve the accuracy of Monte Carlo estimates within message passing in an automated way?*

The above research question RQ2 yields a hybrid Monte Carlo - message passing inference procedure. The performance of this hybrid Monte Carlo - message passing inference procedure should be investigated. For those cases where the precision of Monte Carlo estimates is not satisfactory, is it possible to improve estimates without requiring additional hyperparameters in the overall inference procedure?

RQ4. *How can the free energy gradient estimates help us for scalable and universal variational inference?*

The main difficulty in Bayesian inference is intractable integrals. Variational inference methods transform the probabilistic inference task into an optimization problem with a surrogate objective function called the (variational) free energy, which is an upper bound to the negative log-likelihood objective that we originally wanted to minimize [82]. Stochastic variational inference methods employ noisy free energy gradient estimates in the optimization procedure, which paves the way for scalable and broadly applicable inference procedures. How can we formulate these methods in the message passing framework? Do we gain over raw stochastic variational inference methods by combining them with message passing procedures?

1.3 Summary of Contributions

Below, we list the contributions of this thesis to the machine learning and probabilistic programming fields.

- In Chapter 2, we give a concise review on Forney-style Factor Graphs (FFGs) and message passing interpretation of deterministic Bayesian inference algorithms such as Belief Propagation (BP), Variational Message Passing (VMP), and Expectation Propagation (EP). We explicitly specify the problems to be addressed to build a broadly applicable probabilistic inference engine in the FFG framework.

- In Chapter 3, we introduce a novel inference algorithm called Extended Variational Message Passing (EVMP). EVMP interfaces particle methods and Laplace approximation with message passing algorithms to estimate approximate posterior marginals and expectation quantities that are arguments to VMP messages.

- In Chapter 4, we propose an automated stochastic optimization-based adaptive particle method to improve the precision of Monte Carlo estimates in the EVMP algorithm.

- In Chapter 5, we reformulate two well-known variational inference methods, namely Stochastic Variational Inference (SVI) [83] and Conjugate-computation Variational Inference (CVI) [84], as probabilistic programming algorithms in a message passing framework on FFGs. We show, in general, how to combine stochastic approximation methods for variational inference as local operations that interface with message passing algorithms.

- The majority of the proposed algorithms have been implemented in an FFG-based PPL called *ForneyLab.jl* [2] [62]. ForneyLab aims to facilitate intelligent agent design cycles by automated message passing algorithms [80]. The experiments in this dissertation are all available online[3]. We also provide an additional Julia package called LargeMessageCollider[4] to prove the usefulness of some of the ideas presented in this dissertation that are not implemented in ForneyLab, yet.

1.4 Outline

The proposed automated inference algorithms in this thesis are developed within the FFG framework. Therefore, in Chapter 2, we address **RQ1** by reviewing FFGs and the standard message passing algorithms while introducing the notation of this manuscript. In tree-like factor graphs, inference can be performed exactly as long as we are able to calculate integrals (or summations in discrete cases) in closed form

[2] `https://github.com/semihakbayrak/ForneyLab.jl/tree/dev`
[3] `https://github.com/semihakbayrak/UniPPLwMP`
[4] `https://github.com/semihakbayrak/LargeMessageCollider.jl/tree/UniPPLwMP`

(or in a reasonable amount of time in discrete cases). The algorithmic formulation of exact Bayesian inference by message passing is called Belief Propagation (BP). We give a review of the BP algorithm together with intuition on how to represent messages with probability distributions. We also review the popular approximate inference procedures, namely Variational Message Passing (VMP), structured VMP, and Expectation Propagation (EP) for those models that are not amenable to exact inference. When executing VMP and EP, two concepts in probability theory deserve special attention: the exponential family of distributions and conjugacy. We also define these terms within the context of factor graphs and message passing. Having provided a technical review of the standard message passing algorithms on FFGs, we explicitly describe the problems to be addressed in the next chapters.

Message passing update rules for VMP in factor graphs are functions of certain expectation quantities. These expectation quantities are available for conjugate factor pairs as the gradient of the log-normalizer in the exponential family of distributions. Otherwise, when connected factors are not conjugate pairs, we can still estimate them numerically and approximate VMP messages. In Chapter 3, we show how to estimate these expectation quantities with importance sampling. In FFGs, forward messages arise as normalized proper distributions, which constitute the proposal distribution in the importance sampling procedure. Therefore, proposal distribution design can be delegated to the model itself. As a special case, we approximate the posteriors with Laplace approximation in those cases where the forward message is a Gaussian distribution. The resulting method is called Extended Variational Message Passing (EVMP) and addresses the **RQ2**. We introduce a pseudo-distribution representation called a "List of Weighted Samples" to represent marginal distributions and messages. In this chapter, we cast deterministic factor nodes as the tools that enable end-users to specify a broad range of probabilistic models. We preserve this casting in later chapters as well. To keep the chapter concise, we only provide the reader with the inference rules in Chapter 3. Nevertheless, the reader can find the reasoning behind the rules in the Appendixes.

In Chapter 4, we demonstrate that the EVMP algorithm might not be a viable solution in those model specifications that priors are not good representatives of posteriors. We, therefore, propose an Adaptive Importance Sampling (AIS)-based solution to improve the Monte Carlo estimates and address **RQ3**.

In Chapter 5, we address **RQ4** by focusing our attention on the already existing, well-known methods that are optimizing the free energy objective using stochastic optimization methods. We especially focus on natural gradient descent-based Stochastic Variational Inference (SVI) and Conjugate computation Variational Inference (CVI) methods. We demonstrate that it is easy to formulate these methods as probabilistic programming algorithms in the message passing framework on FFGs. We also show efficient realizations for Black Box Variational Inference (BBVI) and reparameterization-based stochastic variational inference methods [85–87] as local approximation techniques in the message passing framework.

Chapter 6 reflects a final discussion and concludes the dissertation.

1

Chapter 2

Automating Bayesian Inference with Message Passing

Parts of this chapter are published in the original works referenced below.

- Semih Akbayrak, İsmail Şenöz, Alp Sarı and Bert de Vries, *Probabilistic Programming with Stochastic Variational Message Passing*, International Journal of Approximate Reasoning, 2022
- Semih Akbayrak, Ivan Bocharov, and Bert de Vries, *Extended Variational Message Passing for Automated Approximate Bayesian Inference*, Special issue on Bayesian Inference in Probabilistic Graphical Models, Entropy, 2021

Abstract

This chapter introduces the notational convention of this manuscript, closely adhering to [69]. We also provide a background on Forney-style Factor Graph (FFGs) [56] and message passing algorithms commonly used for Bayesian inference on FFGs. We start with Belief Propagation (BP) [52, 63] on tree-like factor graphs to show how to perform exact inference efficiently with a set of distributed operations (see [57, 72] for BP on FFGs). Then, as exact inference is not applicable in many model specifications of interest, we provide an overview of distributed approximate inference methods, namely Variational Message Passing (VMP) [65] and Expectation Propagation (EP) [67, 88] (see [66, 81] for VMP and EP on FFGs, respectively). In conjunction with VMP and EP, we briefly mention the exponential family of distributions [89] and moment matching. We also provide the intuition behind automating these algorithms for probabilistic programming on factor graphs throughout the chapter (see [62] for a formal FFG formulation as a probabilistic

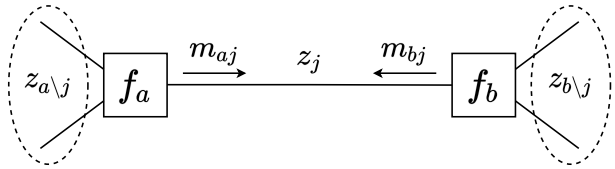


Figure 2.1: A sub-graph $\mathcal{G}(\{a, b\}, j)$. The edge j is connected to factors a and b, which implies that z_j is an argument to functions f_a and f_b. We denote the message on edge j from f_a and f_b with $m_{aj}(z_j)$ and $m_{bj}(z_j)$, respectively.

programming language framework). Finally, we conclude the chapter by pointing out the shortcomings of these algorithms from a probabilistic programming point of view. We refer the interested reader to [64,69,72] for a detailed treatment of BP, VMP, and EP.

2.1 Introduction

In a dynamic world with full of randomness and uncertainties, we tend to define real world processes by functions of random variables, in which learning takes place as a probability distribution estimation over the variables rather than point estimates. Given a function of random variables, a Forney-style Factor Graph (FFG) [56, 57] depicts the independency structure between random variables together with existing factorizations in the function. Consider a factorized function $f(\mathbf{z}) = \prod_{a\in\mathcal{V}} f_a(\mathbf{z}_a)$ of a collection of random variables $\mathbf{z}$, where $\mathbf{z}_a$ stands for the subset of random variables that are arguments of f_a. Specifically, an FFG is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ stands for the set of factor nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of edges. The edges connected to a node $a \in \mathcal{V}$ are denoted by $\mathcal{E}(a)$. Similarly, $\mathcal{V}(i)$ denotes the two factor nodes an edge $i \in \mathcal{E}$ is connected to. We associate the indices a, b, c, d with nodes and i, j, k, l with edges. As we shall detail, it is often sufficient to focus on sub-graphs in FFGs to formulate inference operations. We refer to the sub-graph around a node $a \in \mathcal{V}$ by $\mathcal{G}(a) = (a, \mathcal{E}(a))$. In a similar vein, $\mathcal{G}(i) = (\mathcal{V}(i), i)$ denotes the edge i and the factor nodes it is connected to. We also introduce $\mathcal{G}(a, i) = (\mathcal{V}(i), \mathcal{E}(a))$ and $\mathcal{G}(\{a, b\}, i) = (\mathcal{V}(i), \mathcal{E}(a) \cup \mathcal{E}(b))$ to allow larger sub-graph specifications (see Figure 2.1). We sometimes index sub-graphs to differentiate them, e.g., $\mathcal{G}_p(\mathcal{V}_p, \mathcal{E}_p)$.

In theory, the factors in an FFG can encode any functional relationship between random variables. However, in practice, we tend to associate the factors with prior and conditional distributions to build probabilistic generative model hypotheses for

physical phenomena. Consider the following hierarchical Markov model:

$$p(\mathbf{y}, \mathbf{x}, \mathbf{z}) = p(z_1)p(x_1|z_1)p(y_1|x_1)\prod_{t=2}^{T} p(z_t|z_{t-1})p(x_t|x_{t-1}, z_t)p(y_t|x_t),$$

where $p(x_t|x_{t-1}, z_t) = \int p(x_t|x_{t-1}, w_t)\delta(w_t - g(z_t))\,\mathrm{d}w_t$. The corresponding FFG representation of this model for one time step is visualized in Figure 2.2. Note that the random variables in the model specification are associated with edges in the FFG. Notice that we introduce auxiliary random variables on the FFG that does not exist in the original model specification such as z_t', z_t'', x_t', x_t''. In FFGs, random variables are branched out to more than two factor nodes through equality constraints. This is achieved by introducing an "equality" node $\delta(z_t - z_t')\delta(z_t - z_t'')$ that generates the copies of z_t as z_t' and z_t''.

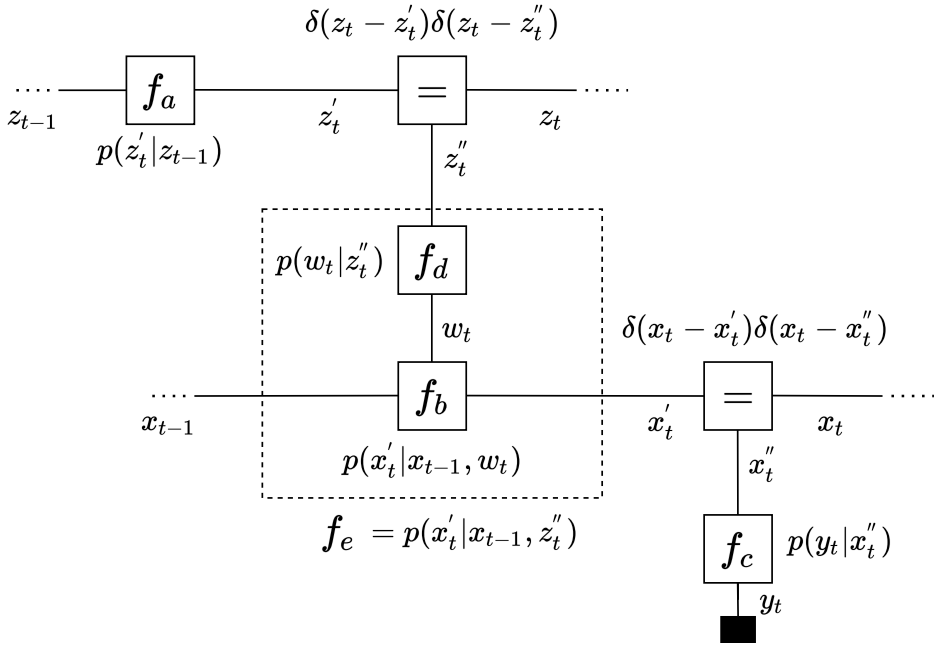


Figure 2.2: An FFG representation of one time step of a hierarchical Markov model. In FFGs, factors represent (conditional) distributions. Here, f_a, f_b and f_c are soft factors that each represent a distribution. On the other hand, $f_d = \delta(w_t - g(z_t))$ represents a deterministic factor, where $g(\cdot)$ is a deterministic function. It is possible to compose factors and consider them as a single unit. In this example, f_e, visualized by a dashed box, stands for the composition of f_d and f_b. It is a notational convention to visualize observed values (y_t) by a small black node.

An equality node is an instance of **deterministic factor nodes**. Deterministic factor nodes represent deterministic conditional distributions such as $p(w_t|z_t) = \delta(w_t - g(z_t))$ induced in the FFG with the factor f_d in Figure 2.2. Here $\delta(\cdot)$ is the Dirac delta in continuous domain and Kronecker delta in discrete domain. We use the term **soft factor nodes** to denote probability distributions other than Dirac delta and Kronecker delta. An interesting property of FFGs is the hierarchical composition: we can create new *higher level* nodes by enclosing a set of connected nodes in a box and integrating out the internal variables in the box. We call these types of nodes **composite factor nodes**. For instance, $p(x_t|x_{t-1}, z_t) = \int p(x_t|x_{t-1}, w_t)\delta(w_t - g(z_t))\,\mathrm{d}w_t$ that is visualized with a dashed-box in Figure 2.2 is a composite node.

So far, we have treated FFGs as visualization tools. Besides visualizing the factorization properties of probabilistic models, FFGs also provide a formal framework for message passing-based inference in probabilistic models. Next, we shall overview message passing algorithms on FFGs.

2.2 Exact Marginalization and Belief Propagation

Inference in FFGs, such as marginal calculations like $f(z_j) = \int f(\mathbf{z})\mathrm{d}\mathbf{z}_{\setminus j}$[1], is carried out by a distributed set of operations. As an example, consider the sub-graph $\mathcal{G}(\{a, b\}, j)$ given in Figure 2.1. Suppose we are interested in obtaining the marginal for z_j, which amounts to computing

$$f(z_j) = \underbrace{\int f_a(\mathbf{z}_a)\mathrm{d}\mathbf{z}_{a\setminus j}}_{m_{aj}(z_j)} \underbrace{\int f_b(\mathbf{z}_b)\mathrm{d}\mathbf{z}_{b\setminus j}}_{m_{bj}(z_j)} . \tag{2.1}$$

In this notation, $m_{aj}(z_j)$ and $m_{bj}(z_j)$ denote the messages on edge j propagating from f_a and f_b respectively. Once the messages $m_{aj}(z_j)$ and $m_{bj}(z_j)$ have been calculated, the marginal distribution calculation refers to multiplication of the messages followed by a normalization:

$$p(z_j) = \frac{f(z_j)}{\int f(z_j)\mathrm{d}z_j} = \frac{m_{aj}(z_j)m_{bj}(z_j)}{\int m_{aj}(z_j)m_{bj}(z_j)\mathrm{d}z_j}. \tag{2.2}$$

This exact inference procedure in tree-like FFGs is known as Belief Propagation (BP) [57, 63, 64, 72].

[1] Throughout the manuscript, we use integral operator to denote integration in continuous domain and summation in discrete domain.

It is customary to normalize messages $m_{aj}(z_j)$ and $m_{bj}(z_j)$ for numerical stability and to parameterize the messages with standard probability distributions:

$$m_{aj}(z_j) = \frac{\int f_a(\mathbf{z}_a)\mathrm{d}\mathbf{z}_{a\setminus j}}{\int f_a(\mathbf{z}_a)\mathrm{d}\mathbf{z}_a}, \qquad m_{bj}(z_j) = \frac{\int f_b(\mathbf{z}_b)\mathrm{d}\mathbf{z}_{b\setminus j}}{\int f_b(\mathbf{z}_b)\mathrm{d}\mathbf{z}_b}.$$

Normalizing the messages and parameterizing them with standard probability distributions does not affect the result of the marginal distribution as marginal distribution calculations (2.2) already involve a normalization procedure. However, it paves the way for the automated BP.

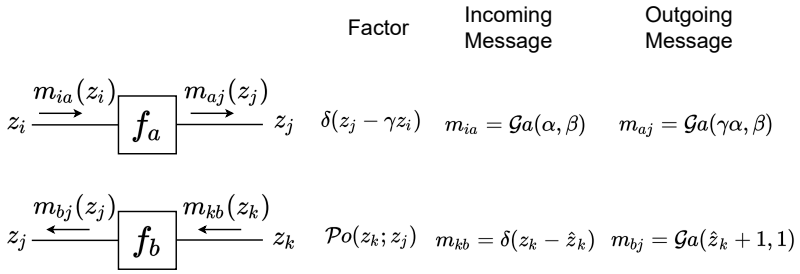


Figure 2.3: Example BP rules around a deterministic factor f_a and a soft factor f_b. The rules are defined locally, which facilitates automated BP and equips FFGs with a plug-in-type structure.

Consider Figure 2.3, where we define example BP rules around a deterministic factor $f_a(z_j, z_i) = \delta(z_j - \gamma z_i)$ and a soft factor $f_b(z_k, z_j) = \mathcal{P}o(z_k; z_j)$ standing for Poisson distribution. Notice that the rules are defined locally around the factors, which facilitates automated BP and equips FFGs with a plug-in-type structure. As the rules are defined locally around factors, we introduce a new notation for the messages: $m_{ia}(z_i)$. Recall that we have already distinguished indices for factors and edges: we tag factors with a, b, c, d and edges with i, j, k, l. Therefore, $m_{ia}(z_i)$ denotes the message on edge i propagating *towards* factor a, whereas $m_{ai}(z_i)$ denotes the message on edge i propagating *from* factor a. This notation enables us to work around factor nodes and derive message passing rules locally.

Having defined the BP rules around the nodes f_a and f_b, we can connect them as in Figure 2.1 to construct a sub-graph. To complete the marginal distribution calculation in an automated manner, we should define message collision rules. Recall from Section 1.1.2 that the collision of the messages is the product of the messages followed by normalization. For example, the collision rule that calculates (2.2) for Gamma messages $m_{aj}(z_j) = \mathcal{G}a(z_j; \alpha_a, \beta_a)$ and $m_{bj}(z_j) = \mathcal{G}a(z_j; \alpha_b, \beta_b)$ is

$$p(z_j) \propto m_{aj}(z_j)m_{bj}(z_j) = \mathcal{G}a(z_j; \alpha_a + \alpha_b - 1, \beta_a + \beta_b),$$

where $\mathcal{G}a(\alpha, \beta)$ is a Gamma distribution with shape α and rate β. By defining these rules, we lay the foundations of a modular plug-in-type FFG-based probabilistic programming language (PPL). We can enhance the capabilities of our PPL by introducing more factors like Figure 2.3, BP message passing rules around these factors and message collision rules that realize marginal distribution calculations.

To sum up, BP formalism perfectly aligns with the plug-in structure of FFGs. However, unfortunately, the integrals for computing messages in BP rarely have analytical solutions. A similar issue arises in the discrete domain as the number of random variables to be marginalized out increases, resulting in intractable summations. Next, we shall overview Variational Message Passing to address these shortcomings of BP.

2.3 Exponential Family of Distributions and Variational Message Passing

The integrals for computing messages in BP rarely have analytical solutions. Instead of calculating the marginals exactly, Variational Message Passing (VMP) [65, 66] iteratively approximates them by introducing additional factorizations in joint distributions and minimizing a variational objective called free energy.

Consider a sub-graph $\mathcal{G}\,(b)$. The joint distribution of $\mathbf{z}_b$ in the sub-graph $\mathcal{G}\,(b)$ under marginalization and normalization constraints is given by [69]

$$p(\mathbf{z}_b) = \frac{f(\mathbf{z}_b)}{\int f(\mathbf{z}_b)\mathrm{d}\mathbf{z}_b} \tag{2.3a}$$

$$\text{where } f(\mathbf{z}_b) = f_b(\mathbf{z_b}) \prod_{i \in \mathcal{E}(b)} m_{ib}(z_i)\,. \tag{2.3b}$$

We approximate this joint distribution by a factorization $q(\mathbf{z}_b) = q(\mathbf{z}_{b\setminus j})q(z_j)$ by minimizing the free energy

$$\mathcal{F}[q(\mathbf{z}_b)] = \mathbb{E}_{q(\mathbf{z}_b)}\left[\log \frac{q(\mathbf{z}_b)}{f(\mathbf{z}_b)}\right] \geq -\log \int f(\mathbf{z}_b)\,\mathrm{d}\mathbf{z}_b, \tag{2.4}$$

which is an upper bound to the negative log normalizer in (2.3a). $q(\mathbf{z}_{b\setminus j})$ may comprise further factorizations that are not explicitly stated. Keeping only the terms with z_j in (2.4),

$$\mathcal{F} \propto \mathbb{E}_{q(z_j)}[\log q(z_j)] - \mathbb{E}_{q(z_j)}\big[\log m_{jb}(z_j)\big] - \mathbb{E}_{q(z_j)}\big[\mathbb{E}_{q(\mathbf{z}_{b\setminus j})}[\log f_b(\mathbf{z_b})]\big], \tag{2.5}$$

we find that the stationary points of $\mathcal{F}$ w.r.t. $q(z_j)$ are (Appendix A)

$$q(z_j)^* = \frac{m_{jb}(z_j)m_{bj}(z_j)}{\int m_{jb}(z_j)m_{bj}(z_j)\,\mathrm{d}z_j}, \tag{2.6}$$

where $m_{bj}(z_j)$ is a VMP message calculated by

$$m_{bj}(z_j) \propto \exp\left(\mathbb{E}_{q(\mathbf{z}_{b\setminus j})}[\log f_b(\mathbf{z}_b)]\right). \tag{2.7}$$

VMP provides a distributed algorithm for coordinate-descent optimization of the free energy by iteratively updating the variational factors one at a time while keeping the other factors fixed [75, 82, 90]. Let us investigate the VMP algorithm by imposing two types of prevalent factorization constraints on $q(\mathbf{z}_b)$: mean-field and structured mean-field.

2.3.1 Mean-field Assumption

In mean-field assumption, we assume a fully factorized recognition distribution $q(\mathbf{z}_b)$:

$$q(\mathbf{z}_b) = \prod_{i\in\mathcal{E}(b)} q(z_i), \tag{2.8}$$

which leads to the following recursive algorithm in FFGs [66]:

1. Choose a variable z_j from the set $\mathbf{z}_b$.
2. Collect the message $m_{jb}(z_j)$ and compute $m_{bj}(z_j)$ by (2.7)
3. Update the approximate marginal $q(z_j)$ by (2.6)
4. Update the local free energy (for performance tracking), i.e., update all terms in $\mathcal{F}$ by (2.5) that are affected by the update of $q(z_j)$.

VMP with mean-field assumption is often easy to execute and amenable to automation as the message and the free energy calculations amounts to calculation of individual random variables' expectation quantities (we shall elaborate on this in Chapter 3). Unfortunately mean-field is a naive assumption that is insufficient to result in satisfactory estimations in highly structured model specifications such as state space models.

2.3.2 Structured Mean-field Assumption

For many model specifications, it is desirable to retain certain dependency structures in the recognition distribution $q(\mathbf{z}_b)$:

$$q(\mathbf{z}_b) = q(z_j)q(\mathbf{z}_{b\setminus j}), \tag{2.9}$$

where we do not assume further factorization in $q(\mathbf{z}_{b\setminus j})$. This structured mean-field assumption leads to the following recursive algorithm in FFGs [66]:

1. For z_j, run steps 2,3,4 defined above for mean field.
2. Collect the messages $m_{ib}(z_i)$ for all $i \in \mathcal{E}(b), i \neq j$.
3. Update the approximate joint distribution $q(\mathbf{z}_{b\setminus j})$ by

$$q(\mathbf{z}_{b\setminus j}) \propto \exp\left(\mathbb{E}_{q(z_j)}[\log f_b(\mathbf{z}_b)]\right) \prod_{\substack{i\in\mathcal{E}(b)\\ i\neq j}} m_{ib}(z_i) \tag{2.10}$$

4. Update the local free energy, i.e., update all terms in $\mathcal{F}$ that are affected by the update of $q(\mathbf{z}_{b\setminus j})$:

$$\mathcal{F} \propto \mathbb{E}_{q(\mathbf{z}_{b\setminus j})}[\log q(\mathbf{z}_{b\setminus j})] - \sum_{\substack{a\in \bigcup_{i\in\mathcal{E}(b)\\ i\neq j} \mathcal{V}(i)}} \mathbb{E}_{q(\mathbf{z}_a)}[\log f_a(\mathbf{z}_a)]. \tag{2.11}$$

Note that we are often interested in approximate marginals of random variables, which can be calculated by marginalization, e.g.,

$$q(z_i) = \int q(\mathbf{z}_{b\setminus j})\, \mathrm{d}\mathbf{z}_{b\setminus\{i,j\}}.$$

Similarly, the joint distribution of the subset of $\mathbf{z}_{b\setminus j}$ can be calculated by marginalization, e.g.,

$$q(z_i, z_k) = \int q(\mathbf{z}_{b\setminus j})\, \mathrm{d}\mathbf{z}_{b\setminus\{i,j,k\}}.$$

2.3.3 Exponential Family of Distributions

The generic VMP message formula is given in (2.7), which involves an integration operation as in BP. Compared to BP, however, the integrand in VMP messages is more amenable to analytical calculations for a certain type of functional form f_b possesses:

$$f_b(\mathbf{z}_b) \propto \exp\left(\phi_j(z_j)^\intercal \lambda_{bj}(\mathbf{z}_{b\setminus j})\right)$$

This functional form relates to exponential family of distributions [89] [91, Chapter 8] (Appendix B) as we shall detail in Chapter 3. Substituting the above f_b in (2.7) yields the VMP message

$$m_{bj}(z_j) \propto \exp(\phi_j(z_j)^\intercal \underbrace{\mathbb{E}_{q(\mathbf{z}_{b\setminus j})}[\lambda_{bj}(\mathbf{z}_{b\setminus j})]}_{\eta_{bj}}).$$

Notice that VMP preserves the functional form of f_b in the message $m_{bj}(z_j)$. If the messages $m_{jb}(z_j)$ and $m_{bj}(z_j)$ take the functional forms (with identical sufficient statistics)

$$m_{jb}(z_j) \propto \exp(\phi_j(z_j)^\intercal \eta_{jb}) \tag{2.12a}$$
$$m_{bj}(z_j) \propto \exp(\phi_j(z_j)^\intercal \eta_{bj}), \tag{2.12b}$$

then the factors in $\mathcal{V}(j)$ are called conjugate factor pairs [92, Chapter 2.4]. Conjugate factor pairs allow the approximate marginal $q(z_j)^*$ in (2.6) to be analytically evaluated in the exponential family of distributions

$$q(z_j)^* = h_j(z_j) \exp(\phi_j(z_j)^\intercal \underbrace{(\eta_{jb} + \eta_{bj})}_{\eta_j} - A_j(\eta_j)). \tag{2.13}$$

Above $h_j(z_j)$ is a constant base measure, $\phi_j(z_j)$ is a vector of sufficient statistics, η_j is a natural (canonical) parameters vector and $A_j(\eta_j)$ is the log-normalizer

$$A_j(\eta_j) = \log \left(\int h_j(z_j) \exp(\phi_j(z_j)^\intercal \eta_j) \mathrm{d}z_j \right) . \tag{2.14}$$

2.4 Moment Matching and Expectation Propagation

Now, we present a message passing interpretation for another popular approximate inference method, namely Expectation Propagation (EP) [67]. Similar to VMP, EP proposes an iterative approach for approximate Bayesian inference. As opposed to VMP, which minimizes the KL divergence $D_{KL}[q(z_j)||p(z_j)]$ through free energy minimization, EP carries out approximate inference through minimization of the reverse KL divergence $D_{KL}[p(z_j)||q(z_j)]$. Minimizing this reverse KL divergence refers to moment matching [75, Chapter 10.7]. Before diving into details of EP, let us explain moment matching.

Consider $q(z_j)$ in the exponential family of distributions given in (2.13). In (2.13), $q(z_j)$ is parameterized with natural parameters η_j. Alternatively, we can parameterize $q(z_j)$ with the expectation of sufficient statistics, otherwise known as moment parameters [91]:

$$\zeta_j = \Psi_j(\eta_j) = \mathbb{E}_{q(z_j)}[\phi_j(z_j)]. \tag{2.15}$$

Notice that (2.15) introduces a link function $\Psi_j(\eta_j)$ as a mapping from natural parameters to moment parameters. Indeed, such a mapping is possible in exponential family of distributions and defined as the gradient of the log-normalizer [91] (find derivation in (B.14)):

$$\zeta_j = \nabla_{\eta_j} A_j(\eta_j). \tag{2.16}$$

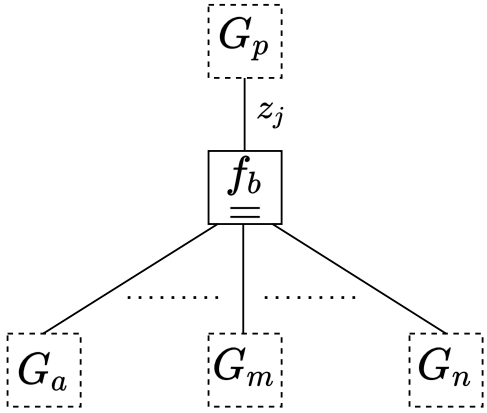


Figure 2.4: An equality node f_b is visualized together with sub-graphs it is connected to. At an EP iteration, f_b collects approximate messages from all the edges but the edge k and send a message $\nu_{bk}(z_k)$ towards z_k. In return, f_b collects an approximate message $\nu_{kb}(z_k)$ to be used in message calculations towards other variables.

We want to find such a $q(z_j)$ in the exponential family of distributions with sufficient statistics $\phi_j(z_j)$ that minimizes $D_{KL}[p(z_j)||q(z_j)]$. This can be achieved by setting the moments ζ_j of $q(z_j)$ equal to $\mathbb{E}_{p(z_j)}[\phi_j(z_j)]$, which refers to the following distribution

$$q(z_j) = h_j(z_j) \exp\left(\phi_j(z_j)^\intercal \Psi_j^{-1}(\mathbb{E}_{p(z_j)}[\phi_j(z_j)]) - A_j\left(\Psi_j^{-1}(\mathbb{E}_{p(z_j)}[\phi_j(z_j)])\right)\right). \tag{2.17}$$

The EP algorithm realizes this moment matching scheme in an iterative manner to approximate marginal distributions. The message passing interpretation of the EP algorithm, which we will make in a short while, inspired some approaches presented in this manuscript. The crux of the EP algorithm from the message passing point of view is to replace messages with approximate messages that ease calculations in later steps. For example, consider a normalized message $m_{jb}(z_j)$ that is not in the exponential family of distributions. The EP algorithm allows us to replace it with an approximate message $\nu_{jb}(z_j)$ that is in the exponential family of distributions and more amenable to analytical calculations in the later stages of the message passing procedure. This approach of exchanging messages is valuable from the probabilistic point of view, as well. Recall that building a message passing-based PPL necessitates manually registered inference rules: collision rules on edges and message passing rules around factor nodes, which substantially restrict the capabilities of the message passing PPL. Nevertheless, we can harness the existing inference

rules in the PPL by exchanging $m_{jb}(z_j)$ with $\nu_{jb}(z_j)$ for which the inference rules are defined. Hence, EP can considerably extend the inference capabilities of a message passing-based PPL.

The EP algorithm around an equality node f_b is summarized below. We approximate $p(z_j)$ with $q(z_j)$ having the functional form proportional to $\exp(\phi_j(z_j)^\intercal \eta_j)$. Thus we want to find the optimum η_j that minimizes $D_{KL}[p(z_j)||q(z_j)]$. The equality node f_b and the sub-graphs that are connected to it are visualized in Figure 2.4.

- Initialize $\nu_{ib}(z_i) \propto \exp(\phi_j(z_i)^\intercal \eta_{ib})$ for all $i \in \mathcal{E}(b)$
- Until convergence repeat the following steps
  1. Choose an edge k from the set $\mathcal{E}(b)$
  2. Calculate $m_{kb}(z_k)$ and $\nu_{bk}(z_k) \propto \prod_{\substack{i \in \mathcal{E}(b) \\ i \neq k}} \int \delta(z_k - z_i)\nu_{ib}(z_i)\,\mathrm{d}z_i$
  3. Calculate $\bar{q}(z_k) = \frac{m_{kb}(z_k)\nu_{bk}(z_k)}{\int m_{kb}(z_k)\nu_{bk}(z_k)\,\mathrm{d}z_k}$
  4. Calculate $\zeta_k = \mathbb{E}_{\bar{q}(z_k)}[\phi_j(z_k)]$
  5. Set $q(z_k) \propto \exp(\phi_j(z_k)^\intercal \eta_k)$, where $\eta_k = \Psi_j^{-1}(\zeta_k)$
  6. Find $\nu_{kb}(z_k) \propto \frac{q(z_k)}{\nu_{bk}(z_k)} \propto \exp(\phi_j(z_k)^\intercal(\eta_k - \eta_{bk}))$

Above steps can be executed in practice if $m_{kb}(z_k)$, $\bar{q}(z_k)$ and $\mathbb{E}_{\bar{q}(z_k)}[\phi_j(z_k)]$ are amenable to analytical calculations, which is not the case for many models of interest. Nevertheless, we still can run EP by approximating $\mathbb{E}_{\bar{q}(z_k)}[\phi_j(z_k)]$ through Monte Carlo sampling methods. We refer the reader to [88] for an overview of Monte Carlo approximations in EP. Similarly, the inverse link function $\Psi_j^{-1}(\zeta_k)$ may not sometimes be available in closed form [93]. Alternatively, we will sometimes use central moments instead:

$$\psi_j(\eta_j) = [\mathbb{E}_{q(z_j)}[z_j], \mathbb{V}_{q(z_j)}[z_j]]^\intercal, \tag{2.18}$$

where $\mathbb{V}_{q(z_j)}[z_j]$ is the variance of the distribution $q(z_j)$. Finding $\psi_j^{-1}(\cdot)$ is often easier than finding the inverse link function $\Psi_j^{-1}(\cdot)$. Note that (2.18) consists of two central moments. This is just to convey the idea of central moment matching. In general, the number of central moments that has to be calculated must be equal to the number of elements in η_j.

2.5 Problem Specification

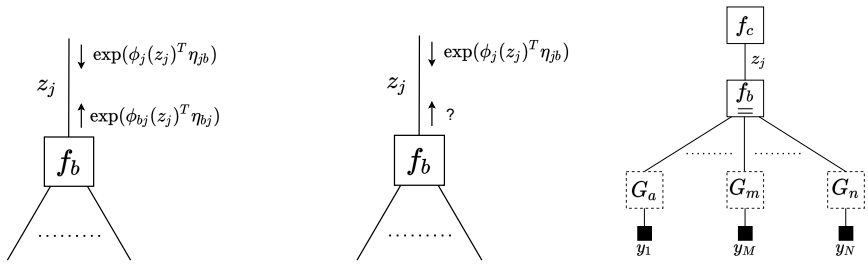
By defining message calculation rules on node level and message collision rules on edges as previously described, a modular, automated message passing-based inference engine can be developed. The modularity of the inference engine can be

enhanced further by registering new type of computational units, such as single input/single output factor blocks for discrete variables, and unified learning algorithms around them [94]. In Figure 2.5, we list the potential difficulties a message passing-based inference engine may encounter:

- **Non-conjugacy:** Non-conjugate factor pairs yield messages with different sufficient statistics, shown in Figure 2.5a, which preclude analytical marginal calculations.
- **Generality:** Deterministic message passing algorithms such as BP and VMP necessitate message passing rules to be defined around factor nodes in advance, which hinders custom model specifications.
- **Scalability:** It is known that the variational inference may not be feasible in real-world applications when the data set is received in sequential order or is just too large to be processed at once due to memory limitations [83, 95, 96]. On FFGs, we encounter this shortcoming of the variational inference around equality nodes, e.g., in Figure 2.5c the equality node f_b needs to collect VMP messages from all the dashed sub-graphs to calculate the message $m_{bj}(z_j)$.

2.6 Conclusion

The aim of this thesis is to alleviate the above shortcomings of message passing algorithms. In the upcoming chapters, we present our solution proposals.



(a) Non-conjugacy (b) Generality (c) Scalability

Figure 2.5: Three possible issues a message passing-based PPL may encounter are visualized. On the left, $m_{bj}(z_j)$ and $m_{jb}(z_j)$ differ in sufficient statistics, which preclude analytical marginal calculations. In (b), f_b is a custom factor node defined by the end-user. The message $m_{bj}(z_j)$ is not available in the PPL due to a missing analytical solution or message passing rule. On the right, the equality node requires VMP messages from all the dashed sub-graphs to calculate $m_{bj}(z_j)$, which might not be feasible for large N.

Chapter 3

Particle Methods and Laplace Approximation within Message Passing

This chapter is based on the original work referenced below. The optimizer used for the Laplace approximation is replaced by an L-BFGS optimizer from the Julia package Optim.jl. The experiments are run in a different machine. The main results are not affected by these changes.

- Semih Akbayrak, Ivan Bocharov, and Bert de Vries, *Extended Variational Message Passing for Automated Approximate Bayesian Inference*, Special issue on Bayesian Inference in Probabilistic Graphical Models, Entropy, 2021

Abstract

Variational Message Passing (VMP) provides an automatable and efficient algorithmic framework for approximating Bayesian inference in factorized probabilistic models that consist of conjugate exponential family distributions. Automation of Bayesian inference tasks is very important, since many data processing problems can be formulated as inference tasks on a generative probabilistic model. However, accurate generative models may also contain deterministic and possibly nonlinear variable mappings and non-conjugate factor pairs that complicate the automatic execution of the VMP algorithm. In this chapter, we show that executing VMP in complex models relies on the ability to compute expectations of statistics of hidden variables. We extend the applicability of VMP by approximating the required expectation quantities in appropriate cases by importance sampling and Laplace

approximation. As a result, the proposed Extended VMP (EVMP) approach supports automated efficient inference for a very wide range of probabilistic model specifications. We implemented EVMP in the Julia language in the probabilistic programming package *ForneyLab.jl* and show by a number of examples that EVMP renders an almost universal inference engine for factorized probabilistic models.

3.1 Introduction

Probabilistic Programming Languages (PPLs) [47] have gained strong popularity over recent years since they support fast algorithm development through automating Bayesian inference in probabilistic models. Many of these PPLs [97–100] are based on numerical approximation methods, which leads to inexact inference results even if the model comprises conjugate factor pairs and exact inference is achievable. Moreover, although a majority of popular PPLs scale well to processing large data sets due to their stochastic inference settings [87], they tend to execute very slowly for certain types of structured dynamic models, such as state space models. Alternatively, some PPLs that execute inference by message passing in a factor graph [62, 101] provide efficient inference performance by exploiting factorization and conjugacy between exponential family-based distribution pairs in the model. In particular the Variational Message Passing (VMP) [65, 66] algorithm has gained a good reputation as it supports efficient inference for conjugate factor pairs in factorized probabilistic models. Unfortunately, non-conjugate factor pairs complicate automated estimation of posterior distributions due to intractability of normalization constants. Likewise, non-linear deterministic relations between model variables often create non-conjugate pairings and thus obstruct the message passing-based inference mechanism.

We propose an Extended VMP (EVMP) algorithm to support automated efficient inference on a wide class of models that contain both non-conjugate relations between factor pairs and deterministic, possibly non-linear factor nodes. In our solution proposal, the regular VMP rules construct the functional forms of the messages. These functional forms contain expectations of functions of hidden variables. In case these expectation quantities can not be evaluated to a closed-form expression, we estimate them by Importance Sampling (IS) [102], which is a well-known Monte Carlo method that approximates intractable posteriors by a set of weighted samples and estimates expectations over this sample set. We also make use of the Laplace approximation [75, Section 4.4] with support by automatic differentiation (autodiff) [41] and optimization [103] tools in appropriate cases to approximate posteriors by Normal distributions, which allows us to calculate the expectations over the approximating Normal distribution. Our proposal leads to an efficient automatable message passing framework that removes most model specification limitations.

In Section 3.2, we specify the proposed Extended VMP algorithm. In order to keep the chapter readable both for the advanced researcher and someone who just needs the results, we defer detailed discussions and derivations of the key equations in EVMP to Appendices B and C. We implemented EVMP in the Julia package *ForneyLab.jl* [104] [62]. In Section 3.3 we present several comparative experiments of EVMP in ForneyLab vs *Turing.jl*, which is an alternative state-of-the-art Julia-based PPL that focuses on Monte Carlo methods for inference. We show that EVMP transforms ForneyLab into an almost universally applicable inference engine while retaining computational efficiency due to its library of closed-form message passing rules. An extensive comparison to related work is presented in Section 3.4.

3.2 Specification of EVMP Algorithm

Variational Message Passing is a fast, efficient and deterministic approximate inference algorithm. However, the applicability of VMP heavily relies on connected factors being conjugate pairs (see Appendix B). In contrast, Monte Carlo methods (see [105] for message passing interpretation) are applicable to a wider range of models with non-conjugate factor pairs. Unfortunately, in comparison to VMP, Monte Carlo methods are considerably slower since they rely on stochastic simulations. As we shall elaborate on in Section 3.4, the recent efforts to combine the best of Monte Carlo methods and variational inference predominantly focus on noisy gradient estimation of the free energy through Monte Carlo sampling and do not take the full advantage of deterministic message passing steps in inference.

In this section, we specify the EVMP algorithm, which combines the efficiency of VMP with the flexibility of the Laplace approximation and the universality of Monte Carlo methods. In the proposed EVMP algorithm, VMP constructs the functional forms of the messages while importance sampling and Laplace approximations are used to estimate the required expectations of statistical quantities if they are not available in closed-form. We first specify the range of probability distribution types for factors, messages and posteriors. These different types are used to identify the specific calculation rules for updating the messages and posteriors. We refer the interested reader to Appendix C for detailed derivations.

As for the notation in this chapter, we shall use $m_{aj}(z_j)$ and $m_{bj}(z_j)$ from Figure 2.1 to respectively denote the *forward* and the *backward* messages. Forward messages in a factor graph refer to the messages that follow the data generating direction. Backward messages, on the other hand, follow the reverse direction.

3.2.1 Distribution Types

We consider the following representation types for probability distributions in factors $p(z_j)$, where z_j holds a variable.

(1) The standard *Exponential Family* (EF) of distributions, i.e.,

$$p(z_j) = h_j(z_j) \exp\left(\phi_j(z_j)^\intercal \eta_j - A_j(\eta_j)\right), \tag{3.1}$$

where $h_j(z_j)$ is the base measure, $\phi_j(z_j)$ is the sufficient statistics vector, η_j is the natural parameters vector and $A_j(\eta_j)$ is the log-partition function.

(2) Distributions that are of exponential form

$$p(z_j) \propto \exp\left(\phi_i(g(z_j))^\intercal \eta_j\right), \tag{3.2}$$

where $g(z_j)$ is a deterministic function. The key characteristic here is that $\phi_i(g(z_j))$ is not recognized as a sufficient statistics vector for any of the standard EF distributions. We call this distribution type a *Non-standard Exponential Family* (NEF) distribution. As we shall show in Section 3.2.6, this distribution type arises only in backward message calculations.

3

(3) A *List of Weighted Samples* (LWS), i.e.,

$$p(z_j) := \left\{ \left(w_j^{(1)}, z_j^{(1)}\right), \ldots, \left(w_j^{(N)}, z_j^{(N)}\right) \right\}. \tag{3.3}$$

(4) Deterministic relations are represented by *delta distributions*, i.e.,

$$p(z_i|z_j) = \delta(z_i - g(z_j)). \tag{3.4}$$

Technically, the equality factor $f(z_i, z_j, z_k) = \delta(z_j - z_i)\delta(z_j - z_k)$ also specifies a deterministic relation between variables.

3.2.2 Factor Types

Factor types $f_a(\mathbf{z}_a)$ are represented by EF and delta distributions.

In a VMP setting, as discussed in this and previous papers on VMP, conjugate soft factors from the exponential family enjoy some computational advantages. As an extension to VMP, the EVMP algorithm inherits the same computational advantages for conjugate factor pairs. In order to automate and generalize the inference to custom non-conjugate soft factors, we compose a generic soft factor by a delta distribution (to describe a non-linear deterministic function) and a standard EF distribution. This decomposition relieves us from manually deriving VMP messages for each different soft factor specification. For a given composite node (delta + standard EF), the EVMP algorithm uses the predefined VMP messages for the standard EF component to compute messages around the composite node. As we will see, this formulation yields an almost generic inference procedure.

3.2.3 Message Types

Forward messages carry either an EF or an LWS distribution. Backward messages carry either an EF or an NEF distribution. This is an arbitrary choice in the sense that we only make this assignment to indicate that in the EVMP algorithm two colliding messages in marginal calculations will not be both of LWS type nor both of NEF type.

3.2.4 Marginal Types

The marginals[1] $q(z_j)$ are represented by either the EF or LWS representations.

To summarize the terminology so far, we defined four distribution types: standard EF (EF), Non-standard EF (NEF), List of Weighted Samples (LWS) and delta distributions. The end user of our algorithm can design a model by using EF and delta distributions. Under the hood, messages may carry EF, NEF or LWS distributions to render the inference. As the output, the end user is provided with either the EF or LWS marginals (posteriors). Next, we discuss how marginals, messages and free energies are computed in the EVMP algorithm. The different types can be used to identify which computational recipe applies. As an aside, Julia's support for multiple dispatch in functions [104] makes this a very elegant mechanism that requires almost no if-then rules.

3.2.5 Computation of Marginals

Here we discuss how EVMP updates the marginals in (2.6). In an FFG, computation of the marginal $q(z_j)$ is realized by a multiplication of colliding forward and backward messages on edge j, respectively $m_{aj}(z_j)$ and $m_{bj}(z_j)$, followed by normalization. We distinguish four types of updates.

(1) In case the colliding forward and backward messages both carry EF distributions with the same sufficient statistics $\phi_j(z_j)$, then computing the marginal simplifies to a summation of natural parameters:

$$\begin{aligned} m_{aj}(z_j) &\propto \exp\left(\phi_j(z)^\intercal \eta_{aj}\right) \\ m_{bj}(z_j) &\propto \exp\left(\phi_j(z)^\intercal \eta_{bj}\right) \\ q(z_j) &\propto m_{aj}(z_j) \cdot m_{bj}(z_j) \propto \exp\left(\phi_j(z)^\intercal (\eta_{aj} + \eta_{bj})\right). \end{aligned}$$

In this case, the marginal $q(z_j)$ will also be represented by the EF distribution type. This case corresponds to classical VMP with conjugate factor pairs.

[1] Given the observations, in probabilistic models, marginals refer to posterior marginal distributions.

(2) The forward message again carries a standard EF distribution. The backward message carries either an NEF distribution or a non-conjugate EF distribution.

(a) If the forward message is Gaussian, i.e., $m_{aj}(z_j) = \mathcal{N}(z; \mu_{aj}, V_{aj})$, we use a Laplace approximation to compute the marginal:

$$\begin{aligned} \mu_j &= \arg\max_{z_j} \left(\log m_{aj}(z_j) + \log m_{bj}(z_j)\right), \\ V_j &= (-\nabla^2_{\mu_j}(\log m_{aj}(\mu_j) + \log m_{bj}(\mu_j))^{-1} \\ q(z_j) &\propto m_{aj}(z_j) \cdot m_{bj}(z_j) = \mathcal{N}(z_j; \mu_j, V_j) \end{aligned}$$

(b) Otherwise ($m_{aj}(z_j)$ is not a Gaussian), we use Importance Sampling (IS) to compute the marginal:

$$\begin{aligned} z_j^{(1)}, \ldots, z_j^{(N)} &\sim m_{aj}(z_j), \\ \tilde{w}_j^{(n)} &= m_{bj}(z_j^{(n)}) \text{ for } n = 1, \ldots, N \\ w_j^{(n)} &= \tilde{w}_j^{(n)} / \sum_{s=1}^{N} \tilde{w}_j^{(s)} \text{ for } n = 1, \ldots, N \\ q(z_j) &\propto m_{aj}(z_j) \cdot m_{bj}(z_j) = \left\{ \left(w_j^{(1)}, z_j^{(1)}\right), \ldots, \left(w_j^{(N)}, z_j^{(N)}\right) \right\}. \end{aligned}$$

(3) The forward message carries an LWS distribution, i.e.,

$$m_{aj}(z_j) := \left\{ \left(w_{aj}^{(1)}, z_j^{(1)}\right), \ldots, \left(w_{aj}^{(N)}, z_j^{(N)}\right) \right\},$$

and the backward message carries either an EF or NEF distribution. In that case, the posterior computation refers to updating the weights in $m_{aj}(z_j)$ (see Appendix F):

$$\begin{aligned} \tilde{w}_j^{(n)} &= w_{aj}^{(n)} m_{bj}(z_j^{(n)}) \text{ for } n = 1, \ldots, N \\ w_j^{(n)} &= \tilde{w}_j^{(n)} / \sum_{s=1}^{N} \tilde{w}_j^{(s)} \text{ for } n = 1, \ldots, N \\ q(z_j) &\propto m_{aj}(z_j) \cdot m_{bj}(z_j) = \left\{ \left(w_j^{(1)}, z_j^{(1)}\right), \ldots, \left(w_j^{(N)}, z_j^{(N)}\right) \right\}. \end{aligned}$$

3.2.6 Computation of Messages

Here we discuss how EVMP compute the messages around a factor f_c visualized in Figure3.1. We specify different message calculation rules depending on the type of the factor f_c.

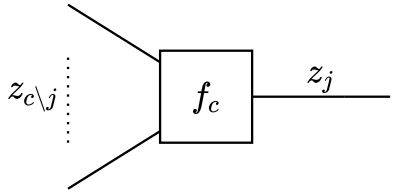


Figure 3.1: A generic node f_c is visualized together with the edges $\mathcal{E}(c)$ connected to f_c.

(1) If factor $f_c(\mathbf{z}_c)$ is a soft factor of the form[2]

$$f_c(\mathbf{z}_c) = p(z_j|\mathbf{z}_{c\backslash j}) = h_j(z_j) \exp\left(\phi_j(z_j)^\intercal \lambda_{cj}(\mathbf{z}_{c\backslash j}) - \log Z_j(\mathbf{z}_{c\backslash j})\right),$$

then the outgoing VMP message to z_j is the following EF-distributed message:

$$m_{cj}(z_j) \propto h_j(z_j) \exp\left(\phi_j(z_j)^\intercal \mathbb{E}_{q(\mathbf{z}_{c\backslash j})}[\lambda_{cj}(\mathbf{z}_{c\backslash j})]\right). \tag{3.8}$$

If rather $z_k \in \mathbf{z}_{c\backslash j}$ than z_j is the output variable of f_c, i.e., if

$$f_c(\mathbf{z}_c) = p(z_k|\mathbf{z}_{c\backslash k}) = h_k(z_k) \exp\left(\phi_k(z_k)^\intercal \lambda_{ck}(\mathbf{z}_{c\backslash k}) - \log Z_k(\mathbf{z}_{c\backslash k})\right).$$

then the outgoing message to z_j is either an EF or an NEF distribution of the form

$$m_{cj}(z_j) \propto \exp\left(\mathbb{E}_{q(z_k)}[\phi_k(z_k)]^\intercal \mathbb{E}_{q(\mathbf{z}_{c\backslash\{j,k\}})}[\lambda_{ck}(\mathbf{z}_{c\backslash k})] - \mathbb{E}_{q(\mathbf{z}_{c\backslash\{j,k\}})}[\log Z_k(\mathbf{z}_{c\backslash k})]\right). \tag{3.9}$$

Note that the message calculation rule for $m_{cj}(z_j)$ given in (3.8) requires the computation of expectation $\mathbb{E}_{q(\mathbf{z}_{c\backslash j})}[\lambda_{cj}(\mathbf{z}_{c\backslash j})]$. For $m_{cj}(z_j)$ in (3.9) we need to compute expectations $\mathbb{E}_{q(\mathbf{z}_{c\backslash\{j,k\}})}[\lambda_{ck}(\mathbf{z}_{c\backslash k})]$, $\mathbb{E}_{q(\mathbf{z}_{c\backslash\{j,k\}})}[\log Z_k(\mathbf{z}_{c\backslash k})]$ and $\mathbb{E}_{q(z_k)}[\phi_k(z_k)]$. In the update rules to be shown below, we will see these expectations of statistics of $\mathbf{z}_c$ appear over and again. In Sec. 3.2.8 we will detail how we calculate these expectations and in Appendix B we will further discuss the origins of these expectations.

(2) In case f_c is a deterministic factor

$$f_c(\mathbf{z}_c) = p(z_j|\mathbf{z}_{c\backslash j}) = \delta(z_j - g(\mathbf{z}_{c\backslash j})). \tag{3.10}$$

[2]Check Appendix B to see how this representation relates to the EF functional form previously introduced in (3.1).

then the forward message from f_c to z_j is of LWS type and is calculated as

$$m_{cj}(z_j) = \left\{ \left(\frac{1}{N}, g(\mathbf{z}_{c\setminus j}^{(1)}) \right), \ldots, \left(\frac{1}{N}, g(\mathbf{z}_{c\setminus j}^{(N)}) \right) \right\}, \tag{3.11}$$
$$\text{where } \mathbf{z}_{c\setminus j}^{(s)} = \bigcup_{\substack{i \in \mathcal{E}(c) \\ i \neq j}} z_i^{(s)} \text{ for } z_i^{(s)} \sim m_{ic}(z_i).$$

If rather $z_k \in \mathbf{z}_{c\setminus j}$ than z_j is the output variable of f_c, i.e., if

$$f_c(\mathbf{z}_c) = p(z_k|\mathbf{z}_{c\setminus k}) = \delta(z_k - g(\mathbf{z}_{c\setminus k})), \tag{3.12}$$

then for the computation of the backward message towards z_j we distinguish two cases:

(a) If all forward incoming messages from the variables $\mathbf{z}_{c\setminus k}$ are Gaussian, we first use a Laplace approximation to obtain a Gaussian joint posterior $q(\mathbf{z}_{c\setminus k}) = \mathcal{N}(\mathbf{z}_{c\setminus k}; \mu, V)$, see Appendix C.2.2 and Appendix C.1.2 for details. Then, we evaluate the posteriors for individual random variables, e.g. $q(z_j) = \int q(\mathbf{z}_{c\setminus k}) \, \mathrm{d}\mathbf{z}_{c\setminus\{k,j\}} = \mathcal{N}(z_j; \mu_j, V_j)$. Finally, we send the following Gaussian backward message:

$$\overleftarrow{m}_{z_j}(z_j) \propto q(z_j)/m_{jc}(z_j). \tag{3.13}$$

(b) Otherwise (the incoming messages from the variables $\mathbf{z}_{c\setminus k}$ are not all Gaussian), we use Monte Carlo and send a message to z_j as a NEF distribution:

$$m_{cj}(z_j) \approx \frac{1}{N} \sum_{s=1}^{N} m_{kc}(g(\mathbf{z}_{c\setminus\{j,k\}}^{(s)}, z_j)), \tag{3.14}$$
$$\text{where } \mathbf{z}_{c\setminus\{j,k\}}^{(s)} = \bigcup_{\substack{i \in \mathcal{E}(c) \\ i \neq j \\ i \neq k}} z_i^{(s)} \text{ for } z_i^{(s)} \sim m_{ic}(z_i).$$

Note that if f_c is a single input deterministic node, i.e., $f_\delta(z_j, z_k) = p(z_k|z_j) = \delta(z_k - g(z_j))$, then the backward message simplifies to $m_{cj}(z_j) = m_{kc}(g(z_j))$ (Appendix C.1.1).

(3) The third factor type that leads to a special message computation rule is the equality node. The outgoing message from an equality node

$$f_c(z_j, z_i, z_k) = \delta(z_j - z_i)\delta(z_j - z_k)$$

is computed by following the sum-product rule:

$$\begin{aligned} m_{cj}(z_j) &= \int \underbrace{\delta(z_j - z_i)\delta(z_j - z_k)}_{\text{node function}} \underbrace{m_{ic}(z_i)m_{kc}(z_k)}_{\text{incoming messages}} \mathrm{d}z_i \mathrm{d}z_k \\ &= m_{ic}(z_j)m_{kc}(z_j). \end{aligned} \tag{3.15}$$

Note that EF and NEF distributions are closed under multiplication.

3.2.7 Computation of Free Energy

Here we discuss how EVMP computes the FE update. We again focus on Figure 2.1 and assume the approximate factorization

$$q(\mathbf{z}_{a\backslash j})q(z_j)q(\mathbf{z}_{b\backslash j}).$$

Note that the FE in the sub-graph $\mathcal{G}(\{a, b\}, j)$ can be decomposed into a subtraction of energy and entropy terms:

$$\mathcal{F}_j = \underbrace{-\mathbb{E}_{q(z_j)}\left[\mathbb{E}_{q(\mathbf{z}_{a\backslash j})}[\log f_a(\mathbf{z}_a)]\right] - \mathbb{E}_{q(z_j)}\left[\mathbb{E}_{q(\mathbf{z}_{b\backslash j})}[\log f_b(\mathbf{z}_b)]\right]}_{\text{(average) energy } \mathcal{U}_a + \mathcal{U}_b} - \underbrace{\mathbb{E}_{q(z_j)}[-\log q(z_j)]}_{\text{entropy } \mathcal{H}_j}, \tag{3.16}$$

These energy and entropy terms can be evaluated because $f_a(\mathbf{z}_a), f_b(\mathbf{z}_b)$ contains only factors that have been defined in the generative model and $q(\mathbf{z}_{a\backslash j})q(z_j)q(\mathbf{z}_{b\backslash j})$ is also accessible as the result of variational inference. Thus we evaluate the FE by evaluating the energy and entropy terms separately.

For an EF-encoded soft factor

$$f_a(\mathbf{z}_a) = p(z_j|\mathbf{z}_{a\backslash j}) = h_j(z_j)\exp\left(\phi_j(z_j)^\intercal \lambda_{aj}(\mathbf{z}_{a\backslash j}) - \log Z_j(\mathbf{z}_{a\backslash j})\right),$$

the energy over the factor f_a evaluates to

$$\mathcal{U}_a = -\mathbb{E}_{q(z_j)}[\log(h_j(z_j))] - \mathbb{E}_{q(z_j)}[\phi_j(z_j)]^\intercal \mathbb{E}_{q(\mathbf{z}_{a\backslash j})}[\lambda_{aj}(\mathbf{z}_{a\backslash j})] + \mathbb{E}_{q(\mathbf{z}_{a\backslash j})}[\log Z_j(\mathbf{z}_{a\backslash j})].$$

We calculate the entropy of $q(z_j)$ as follows:

1. If $q(z_j)$ is a represented by a standard EF distribution, i.e.,

   $$h_j(z_j)\exp\left(\phi_j(z_j)^\intercal \eta_j - A_j(\eta_j)\right),$$

   then

   $$\mathcal{H}_j = -\mathbb{E}_{q(z_j)}[\log(h_j(z_j))] - \mathbb{E}_{q(z_j)}[\phi_j(z_j)]^\intercal \eta_j + A_j(\eta_j).$$

2. Otherwise, if $q(z_j)$ is represented by a LWS, i.e.,

$$q(z_j) := \left\{ \left(w_j^{(1)}, z_j^{(1)} \right), \ldots, \left(w_j^{(N)}, z_j^{(N)} \right) \right\},$$

then

$$\mathcal{H}_j = \hat{\mathcal{H}}_j^1 + \hat{\mathcal{H}}_j^2,$$

where $\hat{\mathcal{H}}_j^1$ and $\hat{\mathcal{H}}_j^2$ are evaluated as discussed in (D.4) and (D.6).

3.2.8 Expectations of Statistics

In many of the above computations for messages, posteriors and free energies, we need to compute certain expectations of statistics of z_j, e.g., the computation of the forward message in (3.8) requires evaluation of $\mathbb{E}_{q(\mathbf{z}_{c\setminus j})}[\lambda_{cj}(\mathbf{z}_{c\setminus j})]$. Here we discuss how EVMP evaluates these expectations. Let us denote a statistic of random variable z_j by $\Phi(z_j)$ and assume we are interested in the expected value $\mathbb{E}_{q(z_j)}[\Phi(z_j)]$. The calculation rule depends on the type of $q(z_j)$:

(1) We have two cases when $q(z_j)$ is coded as an EF distribution, i.e.,

$$q(z_j) = h_j(z_j) \exp\left(\phi_j(z_j)^\intercal \eta_j - A_j(\eta_j)\right) :$$

(a) If $\Phi(z_j) \in \phi_j(z_j)$, i.e., the statistic $\Phi(z_j)$ matches with elements of the sufficient statistics vector $\phi_j(z_j)$, then $\mathbb{E}_{q(z_j)}[\Phi(z_j)]$ is available in closed-form as the gradient of the log-partition function (this is worked out in Appendix B.1.1, see (B.14):

$$\mathbb{E}_{q(z_j)}[\Phi(z_j)] \in \nabla_{\eta_j} A_j(\eta_j).$$

(b) Otherwise ($\Phi(z_j) \notin \phi_j(z_j)$), then we evaluate

$$\mathbb{E}_{q(z_j)}[\Phi(z_j)] \approx \frac{1}{N} \sum_{s=1}^{N} \Phi(z_j^{(s)}),$$

where $z_j^{(s)} \sim q(z_j)$.

(2) In case $q(z_j)$ is represented by a LWS, i.e.,

$$q(z_j) = \left\{ \left(w_j^{(1)}, z_j^{(1)} \right), \ldots, \left(w_j^{(N)}, z_j^{(N)} \right) \right\},$$

then we evaluate

$$\mathbb{E}_{q(z_j)}[\Phi(z_j)] \approx \sum_{s=1}^{N} w_j^{(s)} \Phi(z_j^{(s)}).$$

3.2.9 Pseudo-code for the EVMP Algorithm

Sections 3.2.1 through 3.2.8 provide a recipe for almost universal evaluation of variational inference in factor graphs. We use classical VMP with closed-form solutions when possible, and resort to Laplace or IS approximations when needed. We now summarize the EVMP algorithm by a pseudo-code fragment in Algorithm 1. We use the following notation: $\mathcal{V} = \mathcal{V}_a \cup \mathcal{V}_b \cup \mathcal{V}_c$ is the set of factor nodes (vertices), where $\mathcal{V}_a$, $\mathcal{V}_b$, $\mathcal{V}_c$ stand for the subsets of soft factor nodes, deterministic nodes and equality nodes, respectively. $\mathcal{E}$ is the set of edges that connect the nodes. $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represents the entire factor graph. $\mathbf{h} = \mathbf{z} \cup \mathbf{x}$ is the set of hidden variables, where $\mathbf{x}$ are the variables at the output edges of deterministic nodes. $\mathbf{z}$ are also associated with edges in $\mathcal{E}$, but in contrast to $\mathbf{x}$, $\mathbf{z}$ are not output edges of deterministic nodes.

Algorithm 1 Extended VMP (for Mean-field assumption)

Require: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathbf{h}$, $N_{iterations}$, $q_{initial}$
  for $j = 1 \ldots |\mathbf{h}|$ **do**
    initialize posteriors $q(h_j)$ using $q_{initial}$
  end for
  for i=1...$N_{iterations}$ **do**
    Set Free Energy $\mathcal{F} = 0$
    for $j = 1 \ldots |\mathbf{h}|$ **do**
      Get factors $a, b \in \mathcal{V}(j)$
      Calculate messages $m_{aj}(h_j)$ and $m_{bj}(h_j)$ using Sec. 3.2.6
      Calculate posterior $q(h_j)$ using Sec. 3.2.5
      if $h_j \in \mathbf{z}$ **then**
        Calculate entropy $\mathcal{H}_j$ using Sec. 3.2.7
        Update Free Energy $\mathcal{F} = \mathcal{F} - \mathcal{H}_j$
      end if
    end for
    for all $a \in \mathcal{V}_a$ **do**
      Calculate energy $\mathcal{U}_a$ using Sec. 3.2.7
      Update Free Energy $\mathcal{F} = \mathcal{F} + \mathcal{U}_a$
    end for
    return $q(h)$ **for all** $h \in \mathbf{h}$ and $\mathcal{F}$
  end for

Algorithm 1 is given for a mean-field assumption. For the mean-field assumption to be satisfied in EVMP, the deterministic nodes in the graph $\mathcal{G}$ must be single-input - single-output mappings. The overall structure remains the same for structured factorizations, but messages and posteriors are calculated for sub-graphs instead of single random variables. EVMP automatically realizes structured variational inference for the variables that are input to a multiple-input deterministic node. In

this case, the free energy calculation requires the entropy of the joint distribution of input variables [69]. The joint distribution is available in closed form if all the incoming messages to input variables are Gaussian, and hence joint entropy calculation is straightforward. Otherwise, on the contrary, the joint distribution is rarely available in closed form and EVMP should resort to joint entropy estimation with Monte Carlo (See Appendix D).

3.3 Experiments

We illustrate EVMP-based inference on three different applications. For each application, we show the favorable features of EVMP together with its shortcomings in comparison to Turing [100], which is a general purpose Julia probabilistic programming package.

3.3.1 Filtering with the Hierarchical Gaussian filter

The Hierarchical Gaussian Filter (HGF) [106, 107] is a popular generative model in the neuroscience community. The HGF consists of a Gaussian random walk model, where the variance of the Gaussian is a nonlinear function of the state of the next higher layer, that in turn evolves according to a Gaussian random walk, an so on. Due to the nonlinear link between the layers, classical VMP rules do not have a closed-form solution. While in principle variational updates through Laplace approximation can be manually derived for the HGF model [106], automatically generated EVMP update rules alleviate the need for cumbersome and error-prone manual derivations.

The 2-layer HGF model is defined as

$$z_t \sim \mathcal{N}(z_{t-1}, \sigma_z^2) \tag{3.17a}$$
$$w_t = \exp(z_t) \tag{3.17b}$$
$$x_t \sim \mathcal{N}(x_{t-1}, w_t) \tag{3.17c}$$
$$y_t \sim \mathcal{N}(x_t, \sigma_y^2)\,. \tag{3.17d}$$

For this experiment, we generated $T = 400$ data points by the following process. First, we generated noisy hidden states using $z_t \sim \mathcal{N}\left(\sin\left(\frac{\pi}{60}t\right), 0.01\right), t = 1 \ldots 400$. Next, we generate observations following model (3.17) with $\sigma_y^2 = 0.1$. The generated data set is visualized in (the lower subgraph of) Fig. 3.2.

Next, we filtered the data set by a second HGF, also given by (3.17) with priors $z_0 \sim \mathcal{N}(0, 1)$, $x_0 \sim \mathcal{N}(0, 1)$ and parameters $\sigma_z^2 = \sigma_y^2 = 0.1$. We used EVMP to track the hidden states z_t and x_t. All inference steps including the message passing schedule for filtering in the HGF are detailed in [106]. For each time step, EVMP was run 10 iterations at each filtering step.

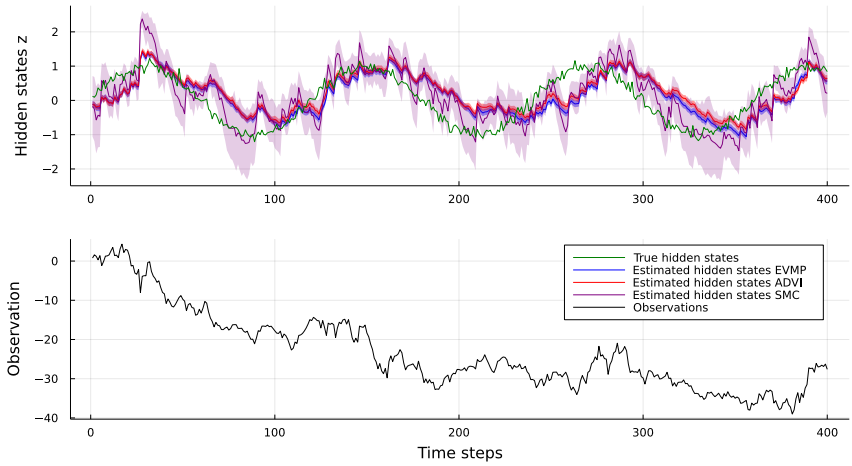


Figure 3.2: Above: Hidden states $z_{1:400}$ and their estimates (ribbon is one variance). The estimates of ForneyLab's Extended VMP are designated by blue while the estimates of Turing's ADVI and SMC are marked by red and purple, respectively. Below: Observed synthetic data.

For comparison we implemented a similar filtering procedure by Automatic Differentiation Variational Inference (ADVI) [50], executed by Julia's *Turing.jl* [100] package. At each time step t, the priors over z_{t-1} and x_{t-1} are set to Gaussian distributions, the mean and variance parameters of which are determined by sampling from the variational posteriors at $t-1$. The only difference between the ForneyLab and Turing implementations, in terms of posterior distribution factorization, is that in Turing's ADVI we posit a fully factorized posterior distribution. This assumption decreases the number of parameters to be estimated via automatic differentiation and speeds up the inference procedure. On the other hand, predefined message passing rules in ForneyLab enable us to retain the dependency structure between x_{t-1} and x_t at time step t in exchange for almost no run-time loss. To be more precise, at time step t, we run inference on the following model: $q_f(z_{t-1})p(z_t|z_{t-1})\delta(w_t - \exp(z_t))q_f(x_{t-1})p(x_t|x_{t-1}, w_t)p(y_t|x_t)$ where $q_f(z_{t-1})$ and $q_f(x_{t-1})$ are the posterior approximations from the previous time step. In ForneyLab, we run the inference with variational distribution $q(z_{t-1})q_f(z_t)q(x_{t-1}, x_t)$ with $q_f(x_t) = \int q(x_{t-1}, x_t)\mathrm{d}x_{t-1}$. We plot estimations for $q_f(z_t)$ in Fig. 3.2. In ADVI, the variational distribution is $q(z_{t-1})q_f(z_t)q(x_{t-1})q_f(x_t)$. Once inference has completed, Turing allows for drawing samples from the variational distribution. We then calculate the mean and variance of these samples to fit Gaussian distributions on $q_f(z_t)$ and $q_f(x_t)$.

We also tested the Sequential Monte Carlo (SMC) engine of Turing on this

model. To our knowledge, Turing does not possess a recursive estimation procedure. In order to execute the SMC engine in a recursive manner, we updated Gaussian priors on x and z at each time step by estimating mean and variance parameters from the previous time step's samples.

The estimated tracks of z_t are visualized in Fig. 3.2. For both EVMP and ADVI, the estimated hidden states largely coincide. We observe that both methods capture the periodic character of the true hidden states $z_{1:400}$ with a delay. We believe that there are two plausible explanations for the delayed estimations: i) in the model specification, we assume that the data generative process is not known fully. The variables $z_{1:400}$ are originally generated from a sinusoidal function of discrete time steps. However, in the model specification, we do not use this information; ii) in the model specification, we define a random walk over hidden variables $z_{1:400}$ that posits the mean of z_t as z_{t-1}. Elaborating the latter factor, the random walk avoids a hidden variable z_t to change drastically compared to z_{t-1} while x_t forces z_t to explain the volatility in the process. Reconciling the beliefs from x_t and z_{t-1}, both Extended VMP and ADVI estimate z_t with a delay. The SMC estimates, on the other hand, differ substantially from the EVMP and ADVI estimates. In SMC, the consecutive estimates are not as smooth as EVMP and ADVI. Nevertheless, the SMC estimates are sometimes more favorable than the EVMP and ADVI estimates, especially between the time steps 300 and 400.

In Turing's ADVI procedure, we used 10 samples per iteration for gradient estimation and set the maximum number of iterations to 4000 per time step to be able to capture this periodic behaviour. The overall inference is completed in roughly 104 seconds[3]. SMC with 1000 samples at each time step is completed roughly in 88 seconds. ForneyLab's EVMP procedure, on the other hand, is able to perform inference around 2.5 seconds on this time series, see Table 3.1. The speed of ForneyLab stems from the hybrid inference nature of EVMP. EVMP resorts to gradient-based optimization only to infer $q_f(z_t)$ and the sampling procedure is required only to estimate statistics related to w_t to be used in the update steps of $q(x_{t-1}, x_t)$. In contrast, ADVI requires sampling and employs noisy gradients in the estimation of all the components of the variational distribution. Similarly, estimations in SMC is solely based on sampling. This experiment validates EVMP as a fast automated variational inference solution for filtering in hierarchical dynamic models.

[3]This and further experiments in this chapter were carried out on a machine with following specs: Julia 1.5.3, Turing v0.19.5, 7 GHz Quad-Core Intel Core i7 CPU, 6 GB 2133 MHz RAM.

Algorithm	Run time (sec)
EVMP (ForneyLab)	2.508
ADVI (Turing)	104.575
SMC (Turing)	88.289

Table 3.1: Run-time comparison of EVMP (in ForneyLab.jl) vs ADVI and SMC (in Turing.jl) for the hierarchical Gaussian filter model.

3.3.2 Parameter Estimation for a Linear Dynamical System

In this experiment, we focused attention on a system identification task in a Linear Dynamical System (LDS) [76, 108]. An LDS is generally defined as

$$x_t|x_{t-1} \sim \mathcal{N}(Ax_{t-1}, Q) \tag{3.18a}$$
$$y_t|x_t \sim \mathcal{N}(Bx_t, R) \tag{3.18b}$$

where y_t are observations and x_t are hidden states.

In this experiment, we are interested in inferring the transition matrix A together with the hidden states from a set of observations. Manually derived closed-form solutions for the system identification task are available both in maximum likelihood estimation [109] and a variational Bayesian approximation [92] contexts. Nevertheless, the goal in this and other papers on probabilistic programming packages is to automatically infer posteriors over the hidden states and parameters without resorting to manual derivations. In principle, EVMP supports to infer the hidden states, A, B, Q and R concurrently. Of course, depending on specific circumstances such as system identifiability and richness of observed data, the performance may vary.

In order to execute our experiment, we first extend (3.18) with a prior on A as follows:

$$a \sim \mathcal{N}(\mu_a, V_a) \tag{3.19a}$$
$$A = \text{reshape}(a, (m, m)) \tag{3.19b}$$
$$x_t|x_{t-1} \sim \mathcal{N}(Ax_{t-1}, Q) \tag{3.19c}$$
$$y_t|x_t \sim \mathcal{N}(Bx_t, R) \tag{3.19d}$$

In (3.19), a holds the vectorized representation of the transition matrix A. Note that (3.19b) can be written as

$$p(A|a) = \delta(A - \text{reshape}(a, (m, m)))\,,$$

and through this manipulation we identify reshape$(a, (m, m))$ as the deterministic factor in (3.10). As a result, ForneyLab's EVMP works out-of-the-box for inference of the transition matrix in (3.19).

We first generated a data set of $T = 40$ number of samples by running model (3.18) with parameters $Q = 0.01 \times I_{2\times 2}$, $R = 0.1 \times I_{2\times 2}$, $B = I_{2\times 2}$ and $A = \begin{bmatrix} 1.0 & 0.2 \\ -0.5 & 0.8 \end{bmatrix}$.

Next, we presented the data set to a second LDS model and aimed to infer posteriors over hidden states and transition matrix A. The prior on a was set to $a \sim \mathcal{N}(0_4, I_{4\times 4})$ and all other parameters were set to the same values as in the data generation process.

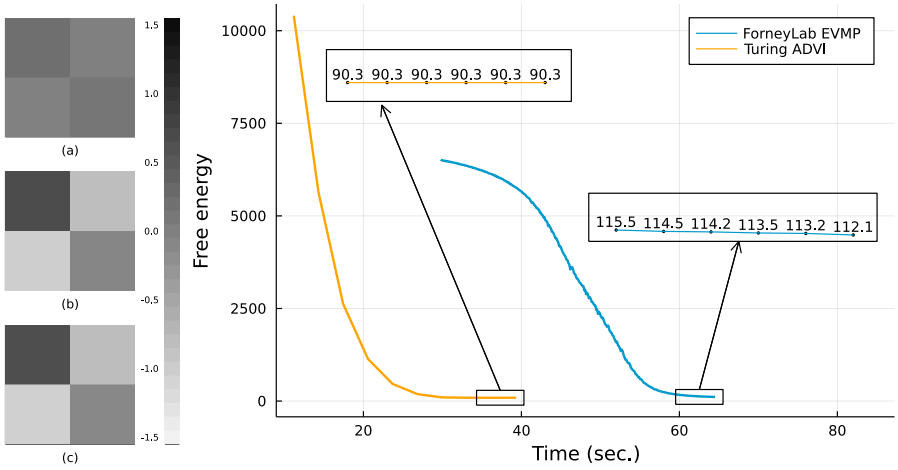


Figure 3.3: Free energy tracks for EVMP on the LDS transition matrix identification task. Left: (a) Mean estimate EVMP for the transition matrix A after 50 iterations, (b) mean estimate after 300 iterations, (c) true transition matrix A that was used to generate synthetic data. Right: Free energy tracks by ForneyLab's EVMP and Turing's ADVI procedures.

We compared the performance of ForneyLab's EVMP with Turing's ADVI and NUTS (No U-Turn Sampler, a Hamiltonian Monte Carlo sampling-based inference method) [49] engines, see Figure 3.3. Both EVMP, ADVI and NUTS successfully converged to almost coinciding estimates of the transition matrix (no notable difference when visualized). We also show free energy tracks for EVMP and ADVI in Figure 3.3. In this experiment, Turing's ADVI outperformed ForneyLab's EVMP in terms of total execution time and the free energy minimization. As a mitigating factor in this analysis, the pre-compilation of the message passing schedule in ForneyLab took about 25 seconds. Execution time details are shown in Table 3.2.

Algorithm	Free energy	Total time (sec)
EVMP (ForneyLab)	112.13	77.265
ADVI (Turing)	90.285	26.514
NUTS (Turing)	-	28.109

Table 3.2: Run-time results for transition matrix estimation in the LDS model.

3.3.3 EVMP for a Switching State Space Model

In this experiment, we go beyond models that only contain continuously valued variables and inquire the capabilities of EVMP on a Switching State Space Model (SSSM) [110], which consists of both continuous and discrete hidden variables. The assumption of constant model parameters in the LDS of Section 3.3.2 does not account for regime changes that occur in many dynamical systems of interest. The SSSM does allow for modeling parameter switches and in this experiment we used the following model:

$$p(A) = \prod_{k=1}^{3} \mathcal{D}\text{ir}(A[:,k];\alpha_k) \tag{3.20a}$$

$$p(z_t|z_{t-1}) = \prod_{k=1}^{3}\prod_{j=1}^{3} A_{kj}^{z_{tk}z_{t-1,j}} \tag{3.20b}$$

$$p(x_t|x_{t-1},z_t) = \prod_{k=1}^{3} \mathcal{N}(x_t|x_{t-1},v_k)^{z_{tk}} \tag{3.20c}$$

$$p(y_t|x_t) = \mathcal{N}(y_t|x_t,1) \tag{3.20d}$$

In this system, $y_t \in \mathbb{R}$ are observations, $x_t \in \mathbb{R}$ is a continuously valued hidden state and z_t is a one-hot coded three-dimensional selection variable, i.e., $z_{tk} \in \{0,1\}$ and $\sum_{k=1}^{3} z_{tk} = 1$. The parameters of the system are the state variances v_k and concentration parameters α_k. The elements of α_k are all 1, except the k^{th} element which is set to 100 to disfavor frequent regime switches, e.g., $\alpha_2 = [1, 100, 1]^\intercal$.

We generated $T = 120$ data points from a random walk process (3.20c) and (3.20d) with process noise variance parameter $v = [v_1, v_2, v_3] = [10, 4, 1]$. From time step $t = 2$ to $t = 25$, we set $z_{t,1} = 1$ and consequently $p(x_t|x_{t-1}) = \mathcal{N}(x_{t-1}, 10)$. From time step $t = 26$ to $t = 75$, we set $z_{t,2} = 1$ and between $t = 76$ to $t = T = 120$ we set $z_{t,3} = 1$. The generated time series is shown in Figure 3.4.

The main difficulty in state inference for the SSSM stems from the coupling between x and z. This is because the variational message passing rules around the node $p(x_t|x_{t-1}, z_t)$ are not pre-defined in ForneyLab, although technically they can be worked out to closed-form expressions [110]. If EVMP were not available

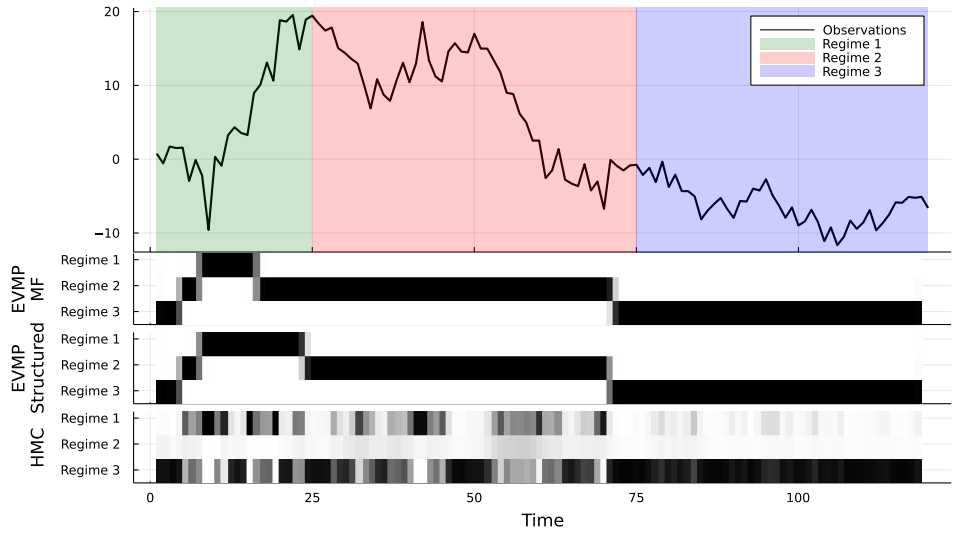


Figure 3.4: Performance results for automated inference in SSSM. Top: generated data set. Bottom 3 subgraphs: posterior for regime selection variable z_t by Mean Field - EVMP, Structured Mean Field - EVMP and HMC procedures respectively. In the Turing HMC simulation, the number of particles in the Particle Gibbs sampler was set to 50. We use step size 0.2 and leapfrog step number 20. HMC is run for 1000 iterations and convergence diagnosis is performed with the Heidelberg and Welch test [111, 112].

either, then a ForneyLab end user would be expected to manually derive closed-form update rules and implement these rules in an additional ForneyLab node. This type of manually assisted inference by end user calculations is what we try to avoid with EVMP and with probabilistic programming packages in general. EVMP enables the user to compensate for the lack of stored message passing rules by introducing an auxiliary variable s in the model with a deterministic relation between s and z:

$$g(z_t) = \sum_{k=1}^{3} z_{tk} \cdot v_k \tag{3.21a}$$

$$p(s_t|z_t) = \delta(s_t - g(z_t)) \tag{3.21b}$$

$$p(x_t|x_{t-1}, s_t) = \mathcal{N}(x_t; x_{t-1}, s_t) \tag{3.21c}$$

$$p(x_t|x_{t-1}, z_t) = \int p(x_t|x_{t-1}, s_t) p(s_t|z_t) \mathrm{d}s_t. \tag{3.21d}$$

After we extend model specification (3.20) by (3.21), then ForneyLab can run

EVMP-based inference out of the box. Note that there is no need for manual inference calculations, but rather a simple manipulation of the generative model that makes the system suited for automated inference.

We tested the performance of two different constraints on the posterior distribution: (i) a mean-field assumption, i.e., $q(x_{1:T}, z_{1:T}) = \prod_{t=1}^{T} q(x_t)q(z_t)$; (ii) a *structured* mean-field assumption, i.e., $q(x_{1:T}, z_{1:T}) = q(x_{1:T}) \prod_{t=1}^{T} q(z_t)$, see Figure 3.4 and Figure 3.5. We observe that the structured factorization, being a less stringent constraint on q, yields a slightly better performance than the mean-field factorization, in particular in estimating the length of the regime 1.

We also compared the performance of ForneyLab's EVMP method to Turing's inference methods. As opposed to the previous two experiments, we can not use solely ADVI, nor Hamiltonian Monte Carlo (HMC, [48, 113]) and NUTS samplers in this experiment since these procedures do not allow inference for discrete random variables. Turing does provide the option to use a Particle Gibbs (PG) sampler [114, 115] for the estimation of the discrete random variables ($z_{1:T}$) in conjunction with the estimation of the continuous random variables ($x_{1:T}$, A) by HMC. Performance result for HMC-PG is shown in Figure 3.4. The performance of the HMC-PG sampler in estimating the correct regimes is far below the EVMP results, although it correctly identified the third regime. Run-time scores are shown in Table 3.3. For HMC, we use reverse-mode automatic differentiation [116] setting of Turing.jl. We also tested Sequential Monte Carlo (SMC) engine of Turing on this model. Unfortunately, we could not get satisfactory estimates and hence did not visualize it. The reader can reach the SMC test in the Github repository.

Algorithm	Free energy	Total time (sec)
EVMP (Mean-field)	283.991	79.590
EVMP (Structured)	273.596	87.949
HMC-PG (Turing)	-	1565.279

Table 3.3: Experimental results for switching state space model

3.4 Related Work

Hybrid Monte Carlo - variational inference techniques have been studied prior to our work. However, mainstream research predominantly consists of variational methods within Monte Carlo techniques as opposed to Monte Carlo methods within a variational inference approach.

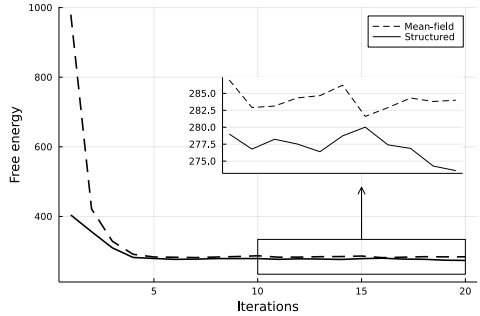


Figure 3.5: Free energy estimates for mean-field and structured mean-field over iterations. We observe that the structured mean-field slightly performs better than the mean-field factorization. Notice that as opposed to vanilla VMP, the free energy do not decline steadily over the iterations. This is due to two reasons. Firstly, the exact VMP steps are approximated in the Extended VMP. Therefore convergence to the stationary points at each iteration is not guaranteed. Secondly, the free energy is not analytically calculated but estimated.

For instance, [117] casts variational distributions as proposal distributions in a Markov-Chain Monte Carlo (MCMC) procedure. Similarly, [118] employs variational methods to design adaptive proposal distributions for Importance Sampling (IS). In [119], gradient estimates of a variational objective are used to tune the parameters of the proposal distributions for MCMC. On the other hand, Monte-Carlo Co-Ordinate Ascent Variational Inference (MC-CAVI), proposed in [120], differs from the aforementioned methods in that it uses MCMC in the calculation of expectations required within the fixed-point iterations of Coordinate Ascent Variational Inference (CAVI).

In this paper, we follow a similar approach as [120], but we use IS to estimate the expectation quantities required in VMP. Both MCMC and IS have their own merits. IS smoothly interfaces with the message passing interpretation of Bayesian inference, which further leads to automated design of proposal distributions. We use Laplace approximation for Gaussian posteriors for variables with Gaussian priors. In the context of dynamical systems, this approach notably overlaps with Gaussian filtering techniques [19, Section 6] that is often achieved by Assumed Density Filtering [121, Section 8.4].

As we show in Appendix F, in the approach that we propose, it is also possible to run automated Bootstrap particle filtering [19, 122] rather than Gaussian filtering methods. As show in [105], particle filtering can be also be framed as message passing on a factor graph. The connection between the particle filtering and variational optimization was introduced in [123]. Their formalism is based on an extension of Particle Belief Propagation [124] to Tree-Reweighted Belief Propagation [125]

while ours revolves around VMP. Similar to our approach, Particle Variational Inference (PVI) [126] aims at optimizing a variational objective by successive IS approximations to true posterior distributions. While PVI applies well to inference for discrete random variables, our EVMP proposal applies to both continuous and discrete random variables.

Variational inference in the context of deterministic building blocks in probabilistic models was studied in [127]. Wheras [127] allows non-linearities to be placed only after Gaussian nodes, the proposed EVMP method generalizes this concept to EF distributed factors.

Non-conjugate Variational Message Passing (NC-VMP) [128] addresses the non-conjugate factor issue in VMP. Assuming that the posterior distribution is an EF distribution, NC-VMP projects the messages to the distribution space of the posterior by equating their sufficient statistics. Thus NC-VMP tunes the natural parameters of the messages in such a way that they converge to the stationary points of the KL divergence between the approximated and true posteriors. [128] also reports that the algorithm necessitates damping for convergence in practice. In response, [84] presents Conjugate-computation Variational Inference (CVI) as a universal inference method that is based on stochastic optimization techniques. As opposed to alternative stochastic variational inference techniques, such as Black-Box Variational Inference [51] and Automatic Differentiation Variational Inference [50], CVI exploits the conjugacy structure of the probabilistic models, which leads to faster convergence. In CVI, non-conjugate factors are incorporated into coordinate ascent steps of mean-field variational inference (with ELBO objective) through a stochastic optimization procedure to form compact posterior approximations with standard probability distributions. In our EVMP approach, the Laplace approximation entails a similarly nested optimization procedure to form compact approximations with Gaussian distributions. Nevertheless, our particle approximations to the true posteriors obviate the need for additional gradient-based optimizations to estimate the parameters of the posteriors.

Finally, the original VMP paper [65] itself briefly mentions sampling methods to overcome the issues with non-conjugate priors. However, they do not extend this idea to deterministic nodes and rather present it as a fallback method whenever soft factors are tied to non-conjugate soft factor priors. Inspired by their vision of approximating the expectation quantities by sampling techniques, we introduced here a fully automated, very broadly applicable extended VMP procedure.

3.5 Discussion

In this chapter, we presented a method for almost universal variational inference on factorized probabilistic models. The core of our method is the locality feature of VMP: the messages at a soft factor are functions of expectations related to argu-

ments of the factor. We employ IS to estimate these expectations or directly approximate posteriors by Laplace approximation if a Gaussian posterior is reasonable. We also extended the Julia package ForneyLab with the proposed EVMP method. In contrast to many alternative PPLs that are solely based on Monte Carlo methods, ForneyLab allows end users to take full advantage of closed-form message passing rules while resorting to small-scale numerical approximations only when needed. We showed that ForneyLab provides an efficient automated variational Bayesian inference tool that in some instances may be preferable to the state-of-the-art Turing package, especially for tasks that include filtering in dynamical models or discrete variables in state space models.

While the experiments support the notion that EVMP is a promising method for inference in non-linear and non-conjugate models, we have not tested our method yet in high-dimensional problems. It is well-known that importance sampling is not efficient in high dimensions [52]. Therefore, we anticipate that for high-dimensional inference tasks with continuous random variables, Hamiltonian Monte Carlo-based methods could outperform EVMP both in terms of run-time and quality of the estimates. Nevertheless, it should be possible to alleviate the deficiencies of EVMP in high dimensions by replacing IS and Laplace approximations by HMC samplers. In essence, HMC is an MCMC method and [120] shows the efficiency of MCMC methods in estimation of the expectations that are required in variational inference. Yet, in lower dimensions, we favor IS and Laplace approximations both because of their promising performance scores in the experiments and also because EVMP relieves users of choosing hyperparameters for the best performance. Recall that in the SSSM experiments in Section 3.3.3, we tested HMC with various hyperparameters to attain the best performance, and yet EVMP was more successful in detecting the hidden regimes. Moreover, in contrast to EVMP, plain HMC is not applicable to estimate discrete variables and needs to be combined with other samplers to run inference on the models with discrete and continuous variables.

In Appendix D, we introduce a variational free energy estimation method that resorts to approximations only if the closed-form expressions of the information-theoretic measures are not available. This differs from alternative automated variational inference techniques, such as Automatic Differentiation Variational Inference (ADVI), which estimates the entire free energy over Monte Carlo summation. Moreover, like HMC, the applicability of ADVI is also limited to continuous variables.

In EVMP, proposal distributions for importance sampling are automatically set to forward messages. Although it is a practical solution with an elegant interpretation in a message passing context, forward messages do not carry information regarding observations. Therefore, we may not acquire useful samples from forward messages if the observations lead to peaky backward messages. In future work, we aim to investigate the effects of alternative proposal distribution design methods.

One major drawback of our ForneyLab implementation is that ForneyLab does not allow loops during the inference procedure. We rarely encounter this problem

with soft factors since the mean field assumption breaks the loops by imposing additional factorizations in variational distributions. However, this may not be the case with deterministic nodes. This is because the input and the output variables of deterministic nodes are tied to each other through a deterministic mapping even after the mean field assumption. For example, consider the following mixture model specification: $p(z) = \mathcal{B}er(\rho), p(x|z) = \delta(x - g_1(z))$ with $g_1(z) = \mu_1 z + \mu_2(1 - z)$, $p(w|z) = \delta(w - g_2(z))$ with $g_2(z) = w_1 z + w_2(1 - z)$ and $p(y|x, w) = \mathcal{N}(x, 1/w)$. Although it is a valid model specification with properly defined message passing rules, the EVMP algorithm is precluded due to the loop: the variable z is connected to two deterministic nodes ($p(w|z)$ and $p(x|z)$) the outputs of which are connected to the same node $p(y|x, w)$. Belief propagation (BP) [63, 64] faces with a similar problem on loopy graphs. Nonetheless, it has been proven that iteratively running BP on loopy graphs often yields satisfactory approximations though the convergence is not guaranteed [121, Section 22.2], [75, Section 8.4.7]. Therefore, it is worth investigating the performance of EVMP executed in a loopy setting.

There are similarities between EVMP and Expectation Propagation (EP) [67] in the sense that both methods estimate the moment parameters of posteriors. In contrast to EP, which approximates belief propagation (BP) [63, 64] messages, EVMP approximates VMP messages, which is applicable to a broader range of model specifications. In future work, we aim to investigate and exploit this relation.

3.6 Conclusion

We developed a hybrid message passing-based approach to variational Bayesian inference that supports deterministic and non-conjugate model segments. The proposed Extended VMP (EVMP) method defaults to analytical updates for conjugate factor pairs and uses a local Laplace approximation or importance sampling when numerical methods are needed. EVMP has been implemented in Julia's ForneyLab package (see Appendix E) and a set of simulations shows competitive inference performance on various inference tasks, in particular for state and parameter tracking in state space models.

3

Chapter 4

Adaptive Particle Methods within Message Passing

This chapter is based on the original work referenced below. An additional illustrative example is provided in the experiments.

- Semih Akbayrak, İsmail Şenöz, Bert de Vries, *Adaptive Importance Sampling Message Passing*, 2022 IEEE International Symposium on Information Theory (ISIT 2022) - Proceedings

Abstract

The aim of Probabilistic Programming (PP) is to automate inference in probabilistic models. One efficient realization of PP-based inference concerns variational message passing-based (VMP) inference in a factor graph. VMP is efficient but in principle only leads to closed-form update rules in case the model consists of conjugate and/or conditionally conjugate factor pairs. In Chapter 3, Extended Variational Message Passing (EVMP) is proposed to broaden the applicability of VMP by importance sampling-based particle methods for non-linear and non-conjugate factor pairs. EVMP automates the importance sampling procedure by employing forward messages as proposal distributions, which unfortunately may lead to inaccurate estimation results and numerical instabilities in case the forward message is not a good representative of the unknown correct posterior. This paper addresses this issue by integrating an *adaptive* importance sampling procedure with message passing-based inference. The resulting method is a hyperparameter-free approximate inference engine that combines recent advances in stochastic adaptive importance sampling and optimization methods. We provide an implementation for the proposed method in the Julia package *ForneyLab.jl*.

4.1 Introduction

Inference is often considered the challenging stage of probabilistic modelling as it requires expertise in (approximate) Bayesian inference methods. Probabilistic Programming Languages (PPLs) [47] aim to automate the inference stage so that end-users can focus only on model development [49–51]. However, achieving this goal is also challenging as it necessitates automatable and broadly applicable inference algorithms that are hopefully hyperparameter-free, too.

This chapter proposes a broadly applicable, hyperparameter-free inference algorithm called Adaptive Importance Sampling Message Passing (AIS-MP). AIS-MP is a hybrid Monte Carlo message passing-based inference approach that combines the efficiency and the speed of rule-based message passing algorithms , such as Belief Propagation (BP) [63, 64], Variational Message Passing (VMP) [65, 66], and Expectation Propagation (EP) [67, 88] with the generality of Monte Carlo sampling on Forney-style Factor Graphs (FFGs).

Our work closely relates to the *Extended* Variational Message Passing (EVMP) algorithm, which extends the applicability of VMP to non-conjugate and non-linear models. EVMP achieves this through estimation of analytically intractable expectation quantities in VMP message calculations, either through a Laplace approximation [75, Section 4.4] or through importance sampling (IS) [102, 129]. To reduce the burden on PPL end users to specify hyperparameter values and proposal distributions, EVMP casts so-called *forward messages* as proposal distributions in IS. This method coincides with the popular Bootstrap particle filtering approach [19, 122], but unfortunately, the method suffers from imprecise expectation estimations and numerical instabilities if the forward message is not a good representative of the correct posterior distribution.

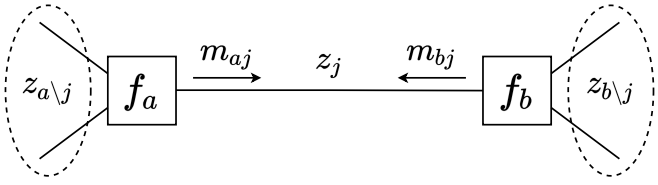
AIS-MP approaches the above shortcomings of EVMP with an *adaptive* IS [130] procedure. Specifically, AIS-MP initializes the proposal distribution with a forward message and runs a stochastic optimization to tune this distribution iteratively until the number of efficient samples exceeds a certain threshold. In the stochastic optimization procedure of the proposal distribution, we use an approach introduced in Stochastic Gradient Population Monte Carlo (SG-PMC) [131], by generalizing it to the exponential family of distributions, similar to [132], with an α-divergence [133] cost function, where $\alpha = 2$. We provide an implementation of AIS-MP in a Julia [104] language-based PPL, *ForneyLab.jl* [62]. We demonstrate its performance on a conjugate model first as a show case and on a non-conjugate Gamma state space model later.

4.2 Recap

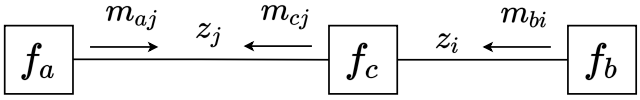
In this section, we give a quick recap on Variational Message Passing (VMP) and Extended Variational Message Passing (EVMP). Assume a probabilistic model $f(\boldsymbol{y}, \boldsymbol{z})$ for a given set of observations $\boldsymbol{y} = \{y_1, y_2, ..., y_N\}$ and hidden variables $\boldsymbol{z} = \{z_1, z_2, ..., z_M\}$. In case exact inference is intractable, the variational inference method approximates the exact posterior $p(\boldsymbol{z}|\boldsymbol{y})$ by a "recognition" distribution $q(\boldsymbol{z})$ through minimization of the (variational) Free Energy

$$\mathcal{F}[q(\boldsymbol{z})] = \mathbb{E}_{q(\boldsymbol{z})}\left[\log q(\boldsymbol{z}) - \log f(\boldsymbol{y}, \boldsymbol{z})\right] , \tag{4.1}$$

where $\mathbb{E}_{q(\boldsymbol{z})}[\cdot]$ refers to expectation with respect to $q(\boldsymbol{z})$. To cast the free energy minimization as an iterative coordinate-descent optimization procedure, $q(\boldsymbol{z})$ is often chosen among factorized distribution families [75].



(a)



(b)

Figure 4.1: (a) A sub-graph with factor nodes f_a and f_b connected through z_j. (b) A deterministic node $f_c = \delta(z_i - g(z_j))$ allows us to specify complex models.

Consider the sub-graph given in Figure 4.1a with a recognition distribution consisting of factors $q(z_j)q(\boldsymbol{z}_{a\setminus j})q(\boldsymbol{z}_{b\setminus j})$. Coordinate-descent optimization of the free energy in this factorized graph is achieved through a distributed inference procedure, called Variational Message Passing (VMP) [65]. In an FFG setting, the VMP

update for latent variable z_j is described by [66]

$$m_{aj}(z_j) \propto \exp\left(\mathbb{E}_{q(\boldsymbol{z}_{a\setminus j})}[\log f_a(\boldsymbol{z_a})]\right) \tag{4.2a}$$

$$m_{bj}(z_j) \propto \exp\left(\mathbb{E}_{q(\boldsymbol{z}_{b\setminus j})}[\log f_b(\boldsymbol{z_b})]\right) \tag{4.2b}$$

$$q(z_k) = m_{aj}(z_j)m_{bj}(z_j) \Big/ \int m_{aj}(z_j)m_{bj}(z_j)\mathrm{d}z_j\,, \tag{4.2c}$$

where $\boldsymbol{z}_a$ denotes the arguments of the factor f_a, $\boldsymbol{z}_{a\setminus j}$ stands for all the arguments of f_a but z_j, and $m_{aj}(z_j)$ and $m_{bj}(z_j)$ are respectively forward and backward messages.

In practice, one has to specify the probabilistic model carefully such that the messages in (4.2) and the marginal posterior in (4.2c) can easily be calculated. A natural way of satisfying these conditions is to choose factors as conjugate (or conditionally conjugate) pairs that leads to following messages

$$m_{aj}(z_j) \propto \exp\left(\phi_{aj}(z_j)^\intercal \cdot \eta_{aj}\right) \tag{4.3a}$$

$$m_{bj}(z_j) \propto \exp\left(\phi_{bj}(z_j)^\intercal \cdot \eta_{bj}\right), \tag{4.3b}$$

where $\phi_{aj}(z_j) = \phi_{bj}(z_j) = \phi_j(z_j)$ since f_a and f_b are conjugate factor pairs. Substituting (4.3) in (4.2c), the approximate posterior turns out to be

$$q(z_j) = h_j(z_j) \exp\left(\phi_j(z_j)^\intercal \cdot \underbrace{(\eta_{aj} + \eta_{bj})}_{\eta_j} - A_j(\eta_j)\right),$$

which is a member of exponential family of distributions [89] with constant base measure $h_j(z_j)$, sufficient statistics $\phi_j(z_j)$, natural parameters η_j and log-partition function $A_j(\eta_j)$. If the underlying graph consists of conditionally conjugate factors then VMP is a very efficient algorithm for approximate Bayesian inference. The presence of non-conjugate factor pairs often prevents efficient realization of VMP in practice.

Extended Variational Message Passing (EVMP) removes the limitations of VMP by estimating the expectation quantities that appear in VMP messages by importance sampling (IS) in an automated way. Consider Figure 4.1b. This time we insert a deterministic mapping $f_c = \delta(z_i - g(z_j))$ between the factors f_a and f_b, which enables the end-user to specify more complex models using deterministic functions $g(z_j)$. In this sub-graph, the message from the deterministic node to z_j is

$$\begin{aligned} m_{cj}(z_j) &= \int m_{bi}(z_i)\delta(z_i - g(z_j))\mathrm{d}z_j \\ &= m_{bi}(g(z_j)) \propto \exp\left(\phi_i(g(z_j))^\intercal \cdot \eta_{bi}\right), \end{aligned} \tag{4.4}$$

which often leads to a backward message $m_{cj}(z_j)$ that differs from the forward message $m_{aj}(z_j)$ in its sufficient statistics. In this case, we are often prevented

from calculating the approximate marginal $q(z_j)$ analytically, since the normalization factor in (4.2c) is not available in closed form. As a remedy, EVMP introduces an additional approximation in the calculation of the posterior $p(z_j|\boldsymbol{y})$, leading to

$$q(z_j) \approx \tilde{q}(z_j) = \sum_{s=1}^{N} w_j^{(s)} \delta(z_j - z_j^{(s)}), \tag{4.5}$$

$$\text{where } z_j^{(s)} \sim m_{aj}(z_j), w_j^{(s)} = \frac{m_{cj}(z_j^{(s)})}{\sum_{n=1}^{N} m_{cj}(z_j^{(n)})}.$$

Similarly, $q(z_i)$ is represented by

$$q(z_i) \approx \tilde{q}(z_i) = \sum_{s=1}^{N} w_j^{(s)} \delta(z_i - g(z_j^{(s)})).$$

The above approximations follow from IS with a proposal distribution $m_{aj}(z_j)$. Once $q(z_j)$ and $q(z_i)$ are represented with weighted samples, EVMP estimates the expectations, such as $\mathbb{E}_{q(z_j)}[\Phi(z_j)]$ and $\mathbb{E}_{q(z_i)}[\Phi(z_i)]$ for an arbitrary function $\Phi(\cdot)$, that are required in calculation of VMP messages around f_a and f_b with Monte Carlo summations, e.g.,

$$\mathbb{E}_{q(z_i)}[\Phi(z_i)] \approx \sum_{s=1}^{N} w_j^{(s)} \Phi(g(z_j^{(s)})),$$

given that the support of $m_{aj}(z_j)$ encapsulates the support of $q(z_j)$ [19, Page 118]. Casting $m_{aj}(z_j)$ as the proposal distribution for IS obviates the need for proposal distribution specification and hence allows EVMP to be automated in message passing-based PPLs. However, this automated process sometimes entails imprecise estimations when the proposal distribution is not a good representative of the unknown posterior. Next, we will improve the performance of EVMP by adaptively adjusting proposal distributions in IS.

4.3 Adaptive Importance Sampling Message Passing

In the previous section, we showed that EVMP employs the pre-defined functional forms of the VMP messages for inference and fills in the expectation quantities required in message calculations with their estimates calculated via IS. In this section, we present Adaptive Importance Sampling Message Passing (AIS-MP) that aims to improve the IS procedure of EVMP by using better proposal distributions.

4.3.1 Adaptive IS with Stochastic Gradient Descent

Consider Figure 4.1 again. We define a weighted particle approximation $\tilde{q}(z_j)$ as

$$q(z_j) \approx \tilde{q}(z_j) = \sum_{s=1}^{N} w_j^{(s)} \delta(z_j - z_j^{(s)}), \tag{4.6}$$

where

$$z_j^{(s)} \sim \pi(z_j),\ w_j^{(s)} = \frac{\frac{m_{jc}(z_j^{(s)}) m_{cj}(z_j^{(s)})}{\pi(z_j^{(s)})}}{\sum_{n=1}^{N} \frac{m_{jc}(z_j^{(n)}) m_{cj}(z_j^{(n)})}{\pi(z_j^{(n)})}}.$$

This time the proposal distribution $\pi(z_j)$ explicitly appears in the computation of weights (4.6), since we do not set $\pi(z_j)$ equal to the forward message. Note also that in order to confine the inference problem to node level we use the notation m_{jc} to denote the message propagating to the node f_c on the edge j. In Figure 4.1b, $m_{jc}(z_j)$ refers to $m_{aj}(z_j)$. In selection of optimal $\pi(z_j)$, we choose to find a minimum variance, unbiased estimator of the normalization constant of $q(z_j)$ that is $\int m_{jc}(z_j) m_{cj}(z_j) \mathrm{d}z_j$. As shown in [133], this can be achieved by minimizing the α-divergence between $m_{jc}(z_j) m_{cj}(z_j)$ and $\pi(z_j)$ for $\alpha = 2$:

$$\begin{aligned} D_2[m_{jc}(z_j) m_{cj}(z_j) || \pi(z_j)] &= \frac{1}{2} \int \frac{\left(m_{jc}(z_j) m_{cj}(z_j) - \pi(z_j)\right)^2}{\pi(z_j)} \mathrm{d}z_k \\ &\propto \int \frac{q(z_j)^2}{\pi(z_j)} \mathrm{d}z_j = \mathbb{E}_{q(z_j)} \left[\frac{q(z_j)}{\pi(z_j)} \right], \end{aligned} \tag{4.7}$$

where the multiplicative and additive constants are dropped. The last line follows from that we choose our proposal $\pi(z_j)$ to be a proper distribution. More precisely, we constrain $\pi(z_j)$ to be in the same distribution family with $m_{jc}(z_j)$, i.e.,

$$\pi(z_j; \lambda) = h_{jc}(z_j) \exp\left(\phi_{jc}(z_j)^\intercal \lambda - A_{jc}(\lambda)\right), \tag{4.8}$$

with a constant $h_{jc}(z_j)$. Having specified the functional form of $\pi(z_j; \lambda)$ in an exponential family, we shall iteratively tune its parameters in such a way that $D_2[m_{jc}(z_j) m_{cj}(z_j) || \pi(z_j; \lambda)]$ is minimized:

$$\lambda^{(t)} \longleftarrow \lambda^{(t-1)} - \rho^{(t)} \nabla_\lambda D_2[m_{jc}(z_j) m_{cj}(z_j) || \pi(z_j; \lambda)], \tag{4.9}$$

where t denotes the iteration index and $\rho^{(t)}$ is the step size at iteration t. We obtain $\nabla_\lambda D_2[m_{jc}(z_j)m_{cj}(z_j)||\pi(z_j;\lambda)]$ by

$$\begin{aligned}\nabla_\lambda D_2 &= -\mathbb{E}_{q(z_j)}\left[\frac{q(z_j)\nabla_\lambda \pi(z_j)}{\pi(z_j)^2}\right] \\ &= -\mathbb{E}_{q(z_j)}\left[\frac{q(z_j)}{\pi(z_j)}\nabla_\lambda \log \pi(z_j)\right] \\ &= -\mathbb{E}_{q(z_j)}\left[\frac{q(z_j)}{\pi(z_j)}(\phi_{jc}(z_j) - \mathbb{E}_\pi[\phi_{jc}(z_j)])\right]. \end{aligned} \tag{4.10}$$

The second line follows from $\frac{\nabla_\lambda \pi(z_j)}{\pi(z_j)} = \nabla_\lambda \log \pi(z_j)$ [51]. The last line is due to the property of exponential family of distributions that the gradient of the log-normalizer is expectation of sufficient statistics [89], i.e., $\nabla_\lambda A_{jc}(\lambda) = \mathbb{E}_\pi[\phi_{jc}(z_j)]$, which is available in closed-form. However, the overall expectation required to calculate $\nabla_\lambda D_2$ does not have an analytical solution since $q(z_j)$ is unknown. Instead, we follow SG-PMC's stochastic approximation approach [131] to estimate the true gradient with

$$\begin{aligned}\tilde{\nabla}_\lambda D_2 &= -\mathbb{E}_{\tilde{q}(z_j)}\left[\frac{q(z_j)}{\pi(z_j)}(\phi_{jc}(z_j) - \mathbb{E}_\pi[\phi_{jc}(z_j)])\right] \\ &= -\sum_{s=1}^{N} w_j^{(s)} \frac{q(z_j^{(s)})}{\pi(z_j^{(s)})}(\phi_{jc}(z_j^{(s)}) - \mathbb{E}_\pi[\phi_{jc}(z_j)]). \end{aligned} \tag{4.11}$$

Notice that $q(z_j^{(s)}) \propto m_{jc}(z_j^{(s)})m_{cj}(z_j^{(s)})$, hence using the weighting definition in (4.6) we can write

$$\frac{q(z_j^{(s)})}{\pi(z_j^{(s)})} \propto w_j^{(s)}. \tag{4.12}$$

We now substitute (4.12) back in (4.11) and find a noisy gradient estimate of $\nabla_\lambda D_2$ in closed-form:

$$\tilde{\nabla}_\lambda D_2 \propto -\sum_{s=1}^{N} {w_j^{(s)}}^2 \left(\phi_{jc}(z_j^{(s)}) - \mathbb{E}_\pi[\phi_{jc}(z_j)]\right). \tag{4.13}$$

Substituting $\nabla_\lambda D_2$ with a noisy gradient estimate $\tilde{\nabla}_\lambda D_2$ in (4.9), and setting $\rho^{(t)}$ according to Robins-Monro conditions [134], i.e., $\sum_{t=1}^{\infty} \rho^{(t)} = \infty$, $\sum_{t=1}^{\infty} {\rho^{(t)}}^2 < \infty$, we get a stochastic gradient descent procedure to tune the parameters λ of the proposal distribution $\pi(z_j;\lambda)$.

In our optimization strategy, we use $m_{jc}(z_j)$ as the initial proposal distribution $\pi(z_j; \lambda^{(0)})$, i.e., $\lambda^{(0)} = \eta_{jc}$ and iteratively refine it. At the end of iteration t, we collect new weighted particles to be used in gradient estimation (4.13) at iteration $t+1$ by employing $\pi(z_j; \lambda^{(t)})$ in (4.6).

To diagnose the convergence of the stochastic approximation, we keep track of the number of efficient particles [19, Chapter 7]:

$$n_{\text{eff}} = 1 \Big/ \sum_{s=1}^{N} {w_j^{(s)}}^2. \tag{4.14}$$

Once the number of efficient particles exceeds the specified threshold, e.g., $n_{\text{eff}} > N/10$ [19, Page 124], we stop the stochastic approximation procedure and use the converged $\pi(z_j)$ in (4.6) to evaluate $\tilde{q}(z_j)$. This procedure relieves the end-user from choosing the number of iterations and carries out the convergence diagnosis automatically.

4.3.2 Backward Message Calculation with Moment Matching

Approximating $q(z_j)$ by a set of weighted samples $\tilde{q}(z_j)$ suffices to execute EVMP. We can also find an approximation $\bar{q}(z_j)$ within the distribution family of $m_{jc}(z_j)$ by using the weighted samples $\tilde{q}(z_j)$ and moment matching [67]:

$$\bar{q}(z_j) \propto \exp\left(\phi_{jc}(z_j)^{\mathsf{T}} \underbrace{\psi^{-1}\Big(\Big[\mathbb{E}_{\tilde{q}(z_j)}[z_j], \mathbb{V}_{\tilde{q}(z_j)}[z_j]\Big]^{\mathsf{T}}\Big)}_{\bar{\eta}_j}\right).$$

Here, $\mathbb{V}_{\tilde{q}(z_j)}[z_j]$ is the variance of z_j calculated over $\tilde{q}(z_j)$ and $\psi(\cdot)$ is a mapping from natural parameters to central moments for the chosen exponential family distribution $\bar{q}(z_j)$, i.e.,

$$\psi(\eta_j) = \Big[\mathbb{E}_{\bar{q}(z_j)}[z_j], \mathbb{V}_{\bar{q}(z_j)}[z_j]\Big]^{\mathsf{T}}. \tag{4.15}$$

The advantages of moment matching are twofold. Firstly, in the free energy calculation, $\bar{q}(z_j)$ yields a closed form solution for the entropy term, corresponding to z_j. Secondly, moment matching allows us to approximate the backward message $m_{cj}(z_j)$ with $\nu_{cj}(z_j)$ by dividing $\bar{q}(z_j)$ with $m_{jc}(z_j)$ [67, 81]:

$$\nu_{cj}(z_j) \propto \exp\left(\phi_{jc}(z_j)^{\mathsf{T}}(\bar{\eta}_{z_j} - \eta_{jc})\right). \tag{4.16}$$

Apart from being employed in VMP seamlessly, the above message is likely to be in a convenient functional form to be integrated with BP or EP.

4.3.3 Algorithm and Node-level Implementation

AIS-MP is summarized in Algorithm 2 and implemented in a Julia language-based message passing PPL *ForneyLab.jl*. By default, the number of samples is set to 1000 and the ADAM optimizer [135] from the Flux.jl [46] package is employed to adaptively adjust step sizes. The end-user is free to change these hyperparameters. Note that we keep track of the convergence and automatically determine whether to terminate the optimization.

If a deterministic relation is not needed in the model specification but the inference is still challenging, the end-user of *ForneyLab.jl* can execute AIS-MP by introducing an auxilary random variable $z_i = z_j$.

Algorithm 2 AIS-MP around a deterministic node in an FFG

Require: A deterministic node $f_c(z_i, z_j) = \delta(z_i - g(z_j))$
Collect $m_{jc}(z_j)$, $m_{cj}(z_j) = m_{ic}(g(z_j))$
Set $t = 0$; $\pi(z_j; \lambda^{(0)}) = m_{cj}(z_j)$
Find $\tilde{q}(z_j)$ using (4.6)
while $n_{\text{eff}} < N/10$ **do** {n_{eff} as (4.14), $N = 1000$ by default}
  $t \mathrel{+}= 1$; set $\rho^{(t)}$ {ADAM optimizer by default}
  Run (4.9) using (4.13)
  Find $\tilde{q}(z_k)$ using (4.6)
end while
Find $\bar{q}(z_j)$ and $\overleftarrow{\nu}_{cj}(z_j)$ using (4.15) and (4.16)
Set $\tilde{q}(z_i) = \sum_{s=1}^{N} w_j^{(s)} \delta(z_i - g(z_j^{(s)}))$

4.4 Related Work

AIS-MP is an instance of the class of approximate inference methods for probabilistic programming, like Black Box Variational Inference (BBVI) [51], Automatic Differentiation Variational Inference (ADVI) [50] and No-U-Turn Sampler (NUTS) [49]. Unlike BBVI, ADVI and NUTS, AIS-MP utilizes stochastic approximation methods only when closed form message passing algorithms do not suffice to run inference in non-conjugate and nonlinear sections of the model specification. Similar hybrid approaches are proposed in [84, 120]. We differ from them in that AIS-MP estimates expectation quantities with IS, which is accompanied by the number of efficient samples to track the convergence of stochastic approximations.

Adaptive importance sampling has been incorporated to enhance the performance of variational inference in [136]. Our work differs from theirs in several notable ways. Most notably, they utilize adaptive importance sampling to reduce

the variance of the free energy gradient estimates in BBVI. Whereas we use adaptive importance sampling directly in the approximation of the posterior marginals and the messages. Secondly, they use Monte Carlo moment matching in approximation of the optimal proposal distribution for free energy gradient estimation. In contrast, we adhere to SG-PMC's stochastic optimization approach to tune proposal distributions by generalizing it to exponential family of distributions and minimizing α-divergence for $\alpha = 2$.

The gradient estimate in (4.13) substantially coincides with the noisy gradient derived in [132] except that their procedure is in a fully online setting (no summation term as in (4.13)). Another difference is that they minimize the variance of the estimator for expectation quantities such as $\mathbb{E}_{q(z_k)}[\Phi(z_k)]$, whereas we minimize the variance of the estimator for the normalization constant of $q(z_k)$ by aiming to get a good weighted samples representation $\tilde{q}(z_k)$.

4.5 Experiments

4.5.1 Illustrative Example

Inspired by [120, Section 3.1], we start with an illustrative example to portrait the importance of proposal distribution selection better while demonstrating how we attack it with AIS-MP. The model we will be working on is the following conditionally conjugate model:

$$\begin{aligned} p(x) &= \mathcal{N}(x; 0, 1) \\ p(z) &= \mathcal{G}a(z; 2.5, 1) \\ p(y|x, z) &= \mathcal{N}(y = 17.5; x, 1/z) \end{aligned} \tag{4.17}$$

where $\mathcal{N}(\mu, v)$ is a Gaussian distribution with mean μ and variance v, and $\mathcal{G}a(a, b)$ is a Gamma distribution with shape a and rate b. In this model specification, the priors are located far away from the posteriors, which as we shall show cause imprecise estimations when the priors are employed as the proposals in importance sampling procedures. Let us first run VMP as the ground truth of this experiment.

We run the VMP algorithm for 8 steps by setting the initial variational distributions $q(x)$ and $q(z)$ to the prior distributions $p(x)$ and $p(z)$, respectively. VMP steps are visualized in Figure 4.2. The free energy at the end of the 8th VMP step is 15.575.

Recall that both EVMP and AIS-MP are approximations to the exact VMP algorithm. Let us, now, see how well the VMP steps are approximated by these algorithms. We start with the EVMP algorithm. To clearly see the effect of adaptive importance sampling, in EVMP, we will use importance sampling approximations only by discarding the automated Laplace step for Gaussian case. To be able to

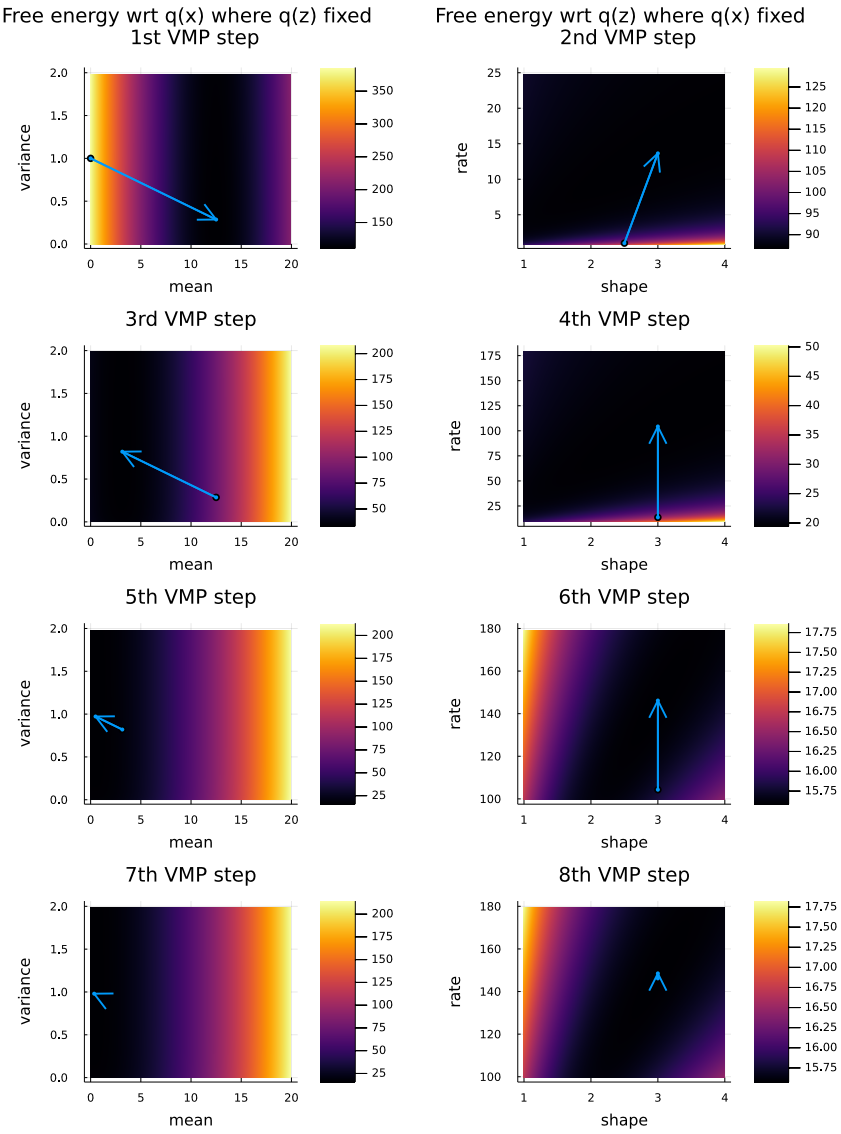


Figure 4.2: 8 VMP steps for the model (4.17) are visualized along with the free energies as functions of variational parameters. On the left column, we update the variational distribution $q(x)$. On the right column, we update the variational distribution $q(z)$. As a result of the coordinate descent procedure with exact stationary point calculations, the free energy decreases at each VMP step.

calculate the free energy in closed form, we also use central moment matching and transform weighted sample representations to standard distributions, i.e., Gaussian for x and Gamma for z. The EVMP steps with importance sampling and moment matching are visualized in Figure 4.3.

Imprecision in expectation estimations causes the EVMP steps to differ substantially from the exact VMP steps. Especially, the final $q(z)$ estimation is considerably different than $q(z)$ found by VMP. The final free energy achieved by EVMP is 20.568.

Finally, we demonstrate AIS-MP steps in Figure 4.4. Notice how similar the AIS-MP algorithm steps are to the exact VMP steps. The final free energy achieved by the AIS-MP algorithm is 15.576, almost no different than the exact VMP algorithm.

4.5.2 Gamma State Space Model

In this subsection, we use AIS-MP in *ForneyLab.jl* to analyze the yearly solar activities between 1945 and 2020 over a sunspots data set (Source: WDC-SILSO, Royal Observatory of Belgium, Brussels [137]; see Figure 4.6). Data samples are rational numbers as they are calculated by averaging count data. To reflect the count nature of the data, we round data sample values to their closest integer values and model them with Poisson likelihoods. We designed a non-conjugate Gamma state-space model to track the rate parameters of the Poisson likelihoods. More precisely, we propose the following generative model for the sunspots dataset:

$$
\begin{aligned}
p(\boldsymbol{y}, \boldsymbol{z}, \gamma) &= p(\gamma)p(z_1|\gamma)p(y_1|z_1)\prod_{t=2}^{T} p(z_t|z_{t-1},\gamma)p(y_t|z_t) \\
\text{where} \qquad p(\gamma) &= \mathcal{G}a(\gamma; 1000, 1) \\
p(z_1|\gamma) &= \mathcal{G}a(z_1; 1, \gamma) \\
p(z_t|z_{t-1},\gamma) &= \mathcal{G}a(z_t; z_{t-1}, \gamma) \\
p(y_t|z_t) &= \mathcal{P}o(y_t; z_t),
\end{aligned}
$$

where $\mathcal{G}a(\cdot; a, b)$ denotes Gamma distribution with shape a and rate b, and $\mathcal{P}o(\cdot; \zeta)$ is Poisson distribution with rate ζ. We run VMP on the model by utilizing IS to estimate expectations quantities that are not available in closed form. We assumed a mean-field factorization on the recognition distribution

$$
q(\gamma, \boldsymbol{z}) = q(\gamma)\prod_{t=1}^{T} q(z_t). \tag{4.18}
$$

This is a challenging model specification for EVMP as the chosen priors lead to forward VMP messages that significantly diverge from the unknown correct posteriors. Hence, we run AIS-MP to automatically tune the proposal distributions

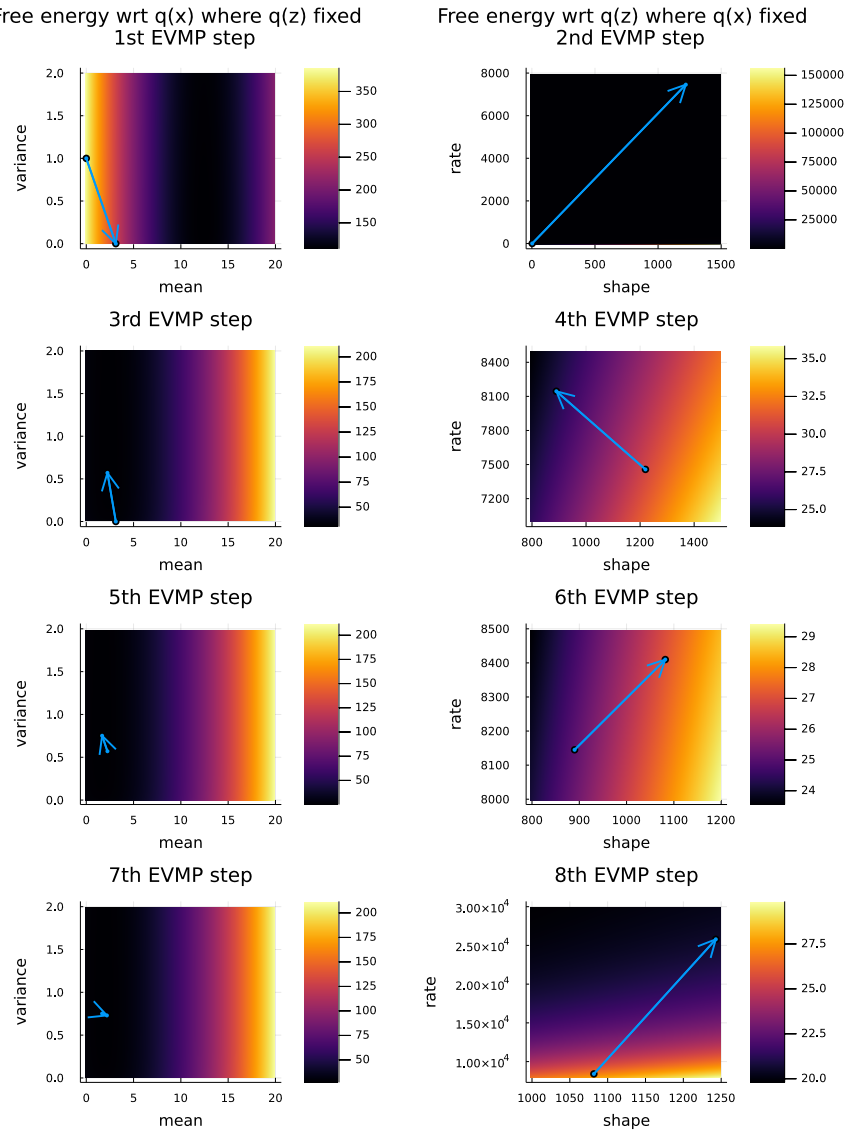


Figure 4.3: 8 EVMP steps with importance sampling and moment matching for the model (4.17) are visualized along with the free energies as functions of variational parameters. On the left column, we update the variational distribution $q(x)$. On the right column, we update the variational distribution $q(z)$. In EVMP, prior distributions $p(x)$ and $p(z)$ are employed as the proposal distributions of the importance sampling procedures. Although the free energy at the end of the 8th step is lower compared to the beginning, EVMP steps substantially differ from exact VMP steps.

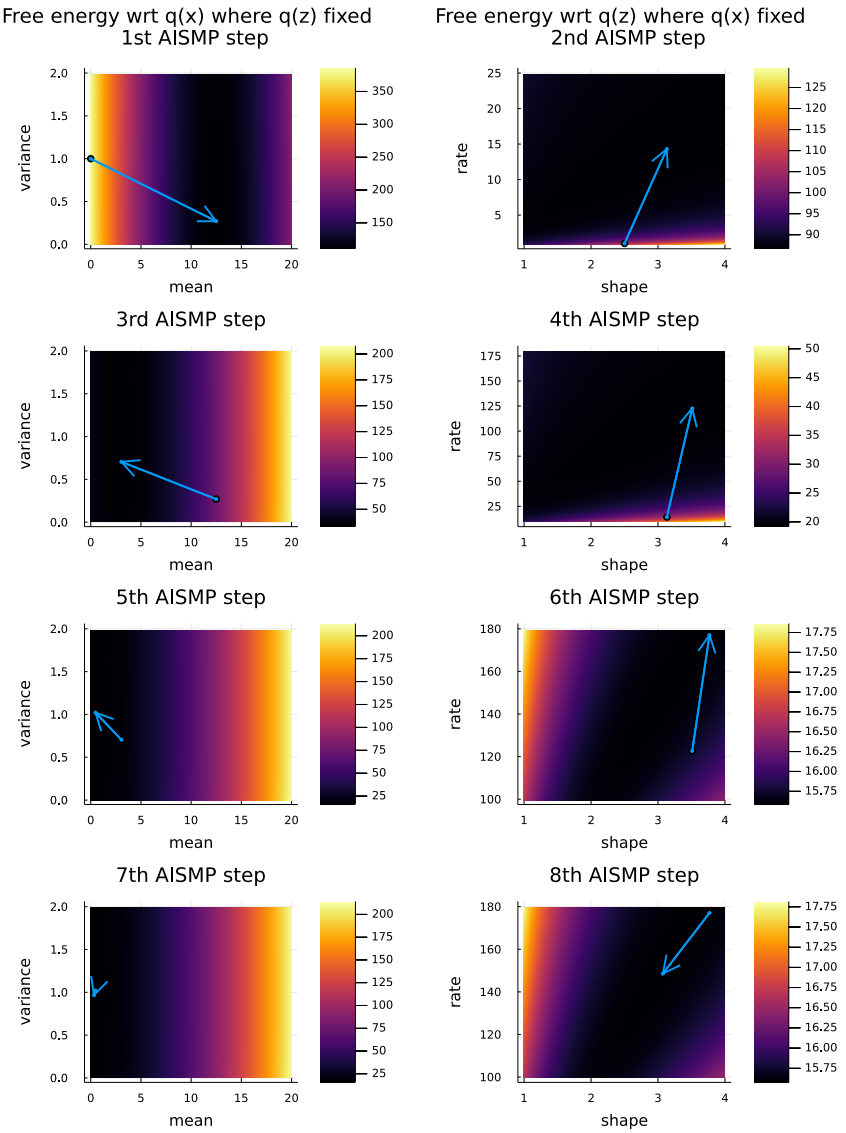


Figure 4.4: 8 AIS-MP steps for the model (4.17) are visualized along with the free energies as functions of variational parameters. On the left column, we update the variational distribution $q(x)$. On the right column, we update the variational distribution $q(z)$. AIS-MP adjusts proposal distributions in the importance sampling procedures starting from the prior distributions $p(x)$ and $p(z)$. The AIS-MP algorithm mimic the exact VMP steps better than the EVMP algorithm and achieves a lower free energy compared to EVMP.

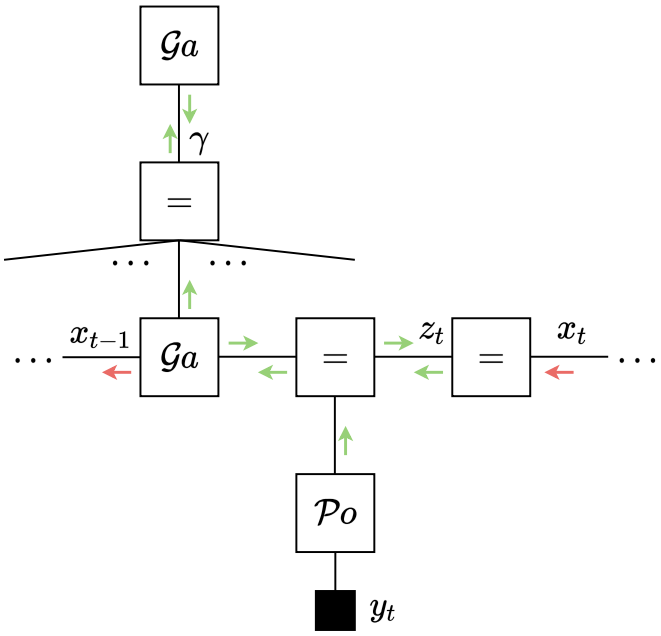


Figure 4.5: A time slice of the FFG we build in ForneyLab for Gamma state-space model. The equality node that generates an auxilary variable x_t performs AIS-MP and approximates a non-Gamma message (shown by red arrow) with a Gamma message (shown by green).

by IS estimates of expectations. We build an FFG as in Figure 4.5 in *ForneyLab.jl*. Note that we introduce deterministic equality nodes that generate dummy variables $\boldsymbol{x} = \boldsymbol{z}$ and perform AIS-MP around these nodes. Running VMP for 10 iterations, the free energy converges as in Figure 4.6 (left) and we get Gamma approximate distributions $q(z_t)$, mean and variance of which are visualized in Figure 4.6 (right).

We compare AIS-MP's estimates with NUTS's in Figure 4.6. We use Turing [100] probabilistic programming package of Julia language to run the NUTS inference engine. We observe that the mean estimates substantially coincide, whereas NUTS's variance estimates are larger in comparison to AIS-MP's. The difference in the variance estimations is not surprising as we use a fully factorized distribution to perform approximate inference in the AIS-MP case, whereas NUTS performs inference over the joint distribution of the random variables. In terms of run time, NUTS is preferable to AIS-MP for this model. AIS-MP converges in 6 VMP iterations, which takes roughly 2.5 minutes to execute in *ForneyLab.jl* including graph construction, whereas NUTS converges very fast with a reverse mode automatic differentiation [116], in less than 3 seconds in our personal computer. Nevertheless,

AIS-MP can still be a good alternative to NUTS in different model specifications. For example, Switching State-Space Model (SSSM) variants [110] comprise both continuous and discrete variables, hence NUTS must be combined with other samplers that perform inference for discrete variables, which sometimes does not yield satisfactory estimations (see Section 3.3.3). As opposed to NUTS, AIS-MP can be used to estimate discrete variables. For an SSSM example, we provide an AIS-MP implementation in our experiments repository. In the SSSM example, forward messages yield good proposal distributions and AIS-MP executes EVMP in effect without the need for stochastic optimization.

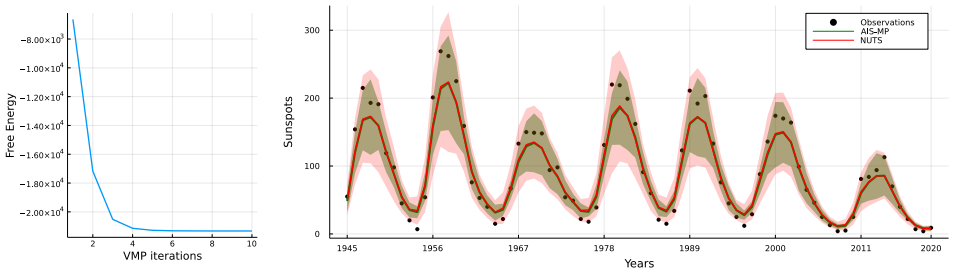


Figure 4.6: Figure summarizes the results of the experimental validation. On the left, free energy over VMP iterations are visualized for AIS-MP algorithm. On the right, black dots indicate sunspot observations [137] rounded to closest integer values. The lines and shaded regions correspond to mean and variance of the posterior estimates $q(z_t)$. Posterior estimates are color-coded based on the legend corresponding to AIS-MP (this paper) and NUTS (baseline) [49].

4.6 Conclusion

In this chapter, we propose Adaptive Importance Sampling Message Passing (AIS-MP) that uses a stochastic adaptive importance sampling approach to estimate the required expectations in the approximation of messages in FFGs. AIS-MP aims to mitigate the shortcomings of the previously proposed Extended VMP (EVMP) algorithm for automated VMP in message passing-based PPLs. As opposed to EVMP, AIS-MP consists of a stochastic optimization procedure, and hence inference is slower compared to EVMP. Nonetheless, as demonstrated by experimental validation, AIS-MP performs better inference on models that EVMP cannot handle. We coded AIS-MP in the Julia language-based PPL, *ForneyLab.jl* and aim to release it as a full inference engine in the future.

Chapter 5

Stochastic Variational Message Passing

This chapter is based on the original works referenced below. A subsection about an efficient Black-Box Variational Inference implementation with message passing and an illustrative example are added in the chapter.

- Semih Akbayrak, İsmail Şenöz, Alp Sarı and Bert de Vries, *Probabilistic Programming with Stochastic Variational Message Passing*, International Journal of Approximate Reasoning, 2022
- Semih Akbayrak, Bert de Vries, *Reparameterization Gradient Message Passing*, The 27th European Signal Processing Conference (EUSIPCO 2019) - Proceedings

Abstract

Stochastic approximation methods for variational inference have recently gained popularity in the probabilistic programming community since these methods are amenable to automation and allow online, scalable, and universal approximate Bayesian inference. Unfortunately, common Probabilistic Programming Libraries (PPLs) with stochastic approximation engines lack the efficiency of message passing-based inference algorithms with deterministic update rules such as Belief Propagation (BP) and Variational Message Passing (VMP). Still, Stochastic Variational Inference (SVI) and Conjugate-Computation Variational Inference (CVI) provide principled methods to integrate fast deterministic inference techniques with broadly applicable stochastic approximate inference. Unfortunately, implementation of SVI and CVI necessitates manually driven variational update rules, which do not yet exist in most PPLs. In this chapter, for the exponential family of distributions, we

cast SVI and CVI explicitly in a message passing-based inference context. We also demonstrate how to go beyond exponential family of distributions by using raw stochastic gradient descent for the minimization of the free energy. We provide an implementation for SVI and CVI in ForneyLab, which is an automated message passing-based probabilistic programming package in the open source Julia language. Through a number of experiments, we demonstrate how SVI and CVI extends the automated inference capabilities of message passing-based probabilistic programming.

5.1 Introduction

Probabilistic programming refers to a programming paradigm that aims to automate and facilitate probabilistic inference for end users with varying degrees of expertise in probabilistic modeling methods [47]. A considerable amount of inference methods and tools have been developed over the past decade to support this endeavour. A very important development in this realm concerns stochastic approximation methods for variational inference where noisy gradient estimates of a variational objective are used to update posterior distributions [78, 79]. These methods have been implemented in Probabilistic Programming Languages (PPLs) such as Turing.jl [100], Stan [97], Pyro [99] and TensorFlow Probability [98]. Realizing variational inference as a stochastic optimization task paves the way toward universal inference and scales well to large data sets [87]. Still, stochastic approximation methods for variational inference come with their own challenges. For example, Black-Box Variational Inference (BBVI) [51] often requires additional steps, such as Rao-Blackwellization [138], control variates [139], or variable reparameterization [85] to reduce the variance in noisy gradient estimates and to attain stable convergence. Another popular method, Automatic Differentiation Variational Inference (ADVI) [50] maps continuous random variables to the real domain and runs stochastic optimization by applying reparameterization in this new domain to prevent high variance in gradient estimates and domain violations. However, the applicability of ADVI is limited to continuous random variables. Moreover, neither BBVI nor ADVI were developed with conjugate model structures in mind, and hence they do not utilize the speed and computational advantages of message passing-based inference methods, such as Belief Propagation (BP) [63, 64], Expectation Propagation (EP) [67, 88] and Variational Message Passing (VMP) [65, 66].

In this chapter, we focus on two other well-recognized stochastic approximation methods for scalable and universal variational inference, namely Stochastic Variational Inference (SVI) [83] and Conjugate-Computation Variational Inference (CVI) [84]. Unlike BBVI and ADVI, both SVI and CVI take advantage of conjugacy structures in the model specifications. They use natural gradient descent [140, 141] to minimize a variational free energy objective in a stochastic setting. By incorpo-

rating Fisher information into stochastic optimization by natural gradient descent, both SVI and CVI adjust steepest descent directions better than raw stochastic gradient descent, which further yields faster and more stable convergence. Whereas SVI aims to scale variational inference for conjugate models to large data sets, CVI extends this idea to non-conjugate models. While both methods seem efficient on paper, automating them in a PPL is a challenging task as both methods necessitate analytical calculations.

In the message passing branch of probabilistic programming, PPLs such as Infer.NET [101] and Julia language [104] packages ReactiveMP.jl [142] and ForneyLab.jl [62] aim to execute automated Bayesian inference by employing predefined, deterministic message update rules. ForneyLab often executes inference faster than stochastic approximation-based methods for conjugate or conditionally conjugate probabilistic models with small data sets. However, it does not scale well to large data sets, does not provide a formal mechanism for online variational inference and its inference capabilities are more or less limited to a priori defined deterministic rules in conjugate model specifications. Nevertheless, ForneyLab possesses in principle the required inference rules to automate and harness CVI and SVI in order to alleviate its shortcomings to a large extent. We present how to incorporate CVI and SVI into ForneyLab's automated message passing framework on factor graphs for the exponential family of distributions. Additionally, we provide strategies to go beyond the exponential family of distributions with stochastic gradient descent optimization of the free energy. In our formalism, we stick to the deterministic message passing algorithms in our PPL as much as possible and minimize the local free energy with stochastic gradient descent when deterministic message passing rules are insufficient to perform the inference. We show the favorable features of these new extensions by a number of experiments.

5.2 Stochastic Variational Message Passing with Natural Gradient Descent

In this section, we address the three problems depicted for a node f_b in Section 2.5. We will use SVI [83] and CVI [84] that are both based on Natural Gradient Descent (NGD) [140,141] optimization of the free energy, otherwise known as the Bayesian Learning Rule [143],

$$\eta_j^{(t)} \longleftarrow \eta_j^{(t-1)} - \rho^{(t)} \nabla_{\eta_j}^{N} \mathcal{F}\left(\eta_j^{(t-1)}\right) \tag{5.1}$$

to tune the natural parameters of the approximate marginal

$$q(z_j;\eta_j) = h_j(z_j)\exp(\phi_j(z_j)^\intercal \eta_j - A_j(\eta_j)). \tag{5.2}$$

In (5.1), t is the iteration index in NGD, $\rho^{(t)}$ is a step size and $\nabla^N_{\eta_j}\mathcal{F}\left(\eta_j^{(t-1)}\right)$ is the natural gradient of the free energy w.r.t. η_j, evaluated at $\eta_j^{(t-1)}$. In our message passing framework, we access $\mathcal{F}(\eta_j)$ through the messages propagating on edge j:

$$\mathcal{F}(\eta_j) = \mathbb{E}_{q(z_j;\eta_j)}[\log q(z_j;\eta_j)] - \mathbb{E}_{q(z_j;\eta_j)}\big[\log m_{jb}(z_j)\big] - \mathbb{E}_{q(z_j;\eta_j)}\big[\log m_{bj}(z_j)\big] + c, \tag{5.3}$$

where c collects the terms independent of η_j. Assuming that

$$m_{jb}(z_j) \propto \exp(\phi_j(z_j)^\intercal \eta_{jb})\,, \tag{5.4}$$

the natural gradient $\nabla^N_{\eta_j}\mathcal{F}(\eta_j)$ evaluates to (Appendix A):

$$\nabla^N_{\eta_j}\mathcal{F}(\eta_j) = \eta_j - \Big(\eta_{jb} + \underbrace{G^{-1}(\eta_j)\nabla_{\eta_j}\mathbb{E}_{q(z_j;\eta_j)}[\log m_{bj}(z_j)]}_{\nabla^N_{\eta_j}\mathbb{E}_{q(z_j;\eta_j)}[\log m_{bj}(z_j)]}\Big)\,, \tag{5.5}$$

where $G(\eta_j)$ refers to the Fisher information matrix of $q(z_j;\eta_j)$, given by the Hessian of the log-normalizer:

$$G(\eta_j) = \nabla^2_{\eta_j} A_j(\eta_j)\,. \tag{5.6}$$

Next, we will discuss how to estimate $\nabla^N_{\eta_j}\mathbb{E}_{q(z_j;\eta_j)}[\log m_{bj}(z_j)]$ for all three cases given in Section 2.5 to optimize the free energy in a stochastic manner by setting $\rho^{(t)}$ according to Robbins-Monro conditions [134], i.e., $\sum_{t=1}^{\infty}\rho^{(t)} = \infty$ and $\sum_{l=1}^{\infty}{\rho^{(t)}}^2 < \infty$.

5

5.2.1 SVI for Scalable VMP

Consider the FFG depicted in Figure 2.5c, where f_b is defined to be an equality node, i.e., z_j is shared across N sub-graphs denoted by dashed boxes. The sub-graphs are comprised of identical functions with distinct local random variables in their arguments, e.g., $\mathcal{G}_a = (\mathcal{V}_a, \mathcal{E}_a), \mathcal{G}_n = (\mathcal{V}_n, \mathcal{E}_n)$ with $d \in \mathcal{V}_a$, $e \in \mathcal{V}_n$ such that $f_d(y_1, z_k, z_j) = h(y_1, z_k, z_j)$, $f_e(y_N, z_l, z_j) = h(y_N, z_l, z_j)$. Consider VMP in this FFG and suppose the messages towards f_b have identical sufficient statistics with distinct natural parameters, i.e,

$$m_{ib}(z_i) \propto \exp(\phi_j(z_i)^\intercal \eta_{ib}). \tag{5.7}$$

In the message passing interpretation of SVI [144], we work with $M < N$ sub-graphs at a VMP iteration by estimating the message $m_{bj}(z_j)$ from the equality node

as

$$m_{bj}(z_j) \approx \left(\int \prod_{\substack{i \in \mathcal{E}^{'}(b) \\ i \neq j}} \delta(z_j - z_i) m_{ib}(z_i) \, \mathrm{d}z_i \right)^{N/M} \propto \exp\left(\phi_j(z_j)^{\intercal} \frac{N}{M} \sum_{\substack{i \in \mathcal{E}^{'}(b) \\ i \neq j}} \eta_{ib} \right). \tag{5.8}$$

Here, $\mathcal{E}^{'}(b) \subseteq \mathcal{E}(b)$ denotes M edges on which the messages are available towards f_b. Substituting the above estimate in (5.5), the natural gradient estimate of the free energy evaluates to

$$\tilde{\nabla}^{N}_{\eta_j} \mathcal{F}(\eta_j) = \eta_j - \left(\eta_{jb} + \frac{N}{M} \sum_{\substack{i \in \mathcal{E}^{'}(b) \\ i \neq j}} \eta_{ib} \right). \tag{5.9}$$

For an FFG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, Algorithm 3 shows how SVI is executed by applying NGD around equality nodes $\mathcal{V}_{=} \subset \mathcal{V}$ that are associated with shared variables $\bar{\boldsymbol{z}}$ such that $\bar{\boldsymbol{z}} \subset \boldsymbol{z}$.

Algorithm 3 SVI on an FFG.

Require: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ for $f(\boldsymbol{z})$ such that $\bar{\boldsymbol{z}} \subset \boldsymbol{z}$ is a collection of variables shared across sub-graphs $\{\mathcal{G}_a, \ldots, \mathcal{G}_n\}$ through equality nodes $\mathcal{V}_{=} \subset \mathcal{V}$;
Number of iterations: T
for all $z_j \in \bar{\boldsymbol{z}}$ **do**
  Initialize $q(z_j) \propto \exp(\phi_j(z_j)^{\intercal} \eta_j^{(0)})$
end for
for $t = 1, \ldots, T$ **do**
  Choose a subset $\mathcal{G}^{'}$ of sub-graphs to be processed
  for all $\tilde{\mathcal{G}} \in \mathcal{G}^{'}$ **do**
    Inside the sub-graph $\tilde{\mathcal{G}}$, run VMP for one step as in Section 2.3
    Calculate VMP messages towards b for all $b \in \mathcal{V}_{=}$
  end for
  for all $b \in \mathcal{V}_{=}$ **do**
    Collect all available messages $m_{ib}(z_i)$ s.t. $i \in \mathcal{E}^{'}(b)$
    Calculate $\tilde{\nabla}^{N}_{\eta_j} \mathcal{F}(\eta_j)$ using (5.9) {Given that $z_j \in \bar{\boldsymbol{z}}$}
    Set a step size $\rho^{(t)}$
    Update $q(z_j)$ using (5.1)
  end for
end for

5.2.2 CVI for Non-conjugate Inference

Next, we consider the factors in $\mathcal{V}(j)$ as non-conjugate pairs that yield messages with different sufficient statistics (see Figure 2.5a):

$$m_{jb}(z_j) \propto \exp(\phi_j(z_j)^\intercal \eta_{jb}) \tag{5.10a}$$
$$m_{bj}(z_j) \propto \exp(\phi_{bj}(z_j)^\intercal \eta_{bj}) \tag{5.10b}$$

Motivated by the message approximation scheme within the exponential family of distributions in [128], we will use CVI to replace $m_{bj}(z_j)$ with an approximate message $\nu_{bj}(z_j)$ that has sufficient statistics $\phi_j(z_j)$. In an ideal scenario, $\nu_{bj}(z_j)$ needs to satisfy that $q(z_j) \propto m_{jb}(z_j)\nu_{bj}(z_j)$ is a stationary point of the free energy. To search a stationary point, we run NGD given in (5.1) until convergence, and then find $\nu_{bj}(z_j)$ as described by [67, 81],

$$\nu_{bj}(z_j) = \frac{q(z_j; \eta_j^*)}{m_{jb}(z_j)} \propto \exp(\phi_j(z_j)^\intercal(\eta_j^* - \eta_{jb})). \tag{5.11}$$

In the NGD-based optimization of the free energy, we employ an estimate for $\nabla^N_{\eta_j}\mathbb{E}_{q(z_j;\eta_j)}[\log m_{bj}(z_j)]$, which does not have an analytical solution, since $q(z_j)$ and $m_{bj}(z_j)$ differ in sufficient statistics. In some cases, such as when $q(z_j; \eta_j)$ is a Gaussian distribution, it is possible to directly estimate the natural gradients without explicitly evaluating the Fisher information matrix and its inverse, see [84, Appendix B] for details, which follows from [145]. In our implementation, we stick to their computationally efficient approach for the Gaussian case. In other cases, we compute $G(\eta)$ with automatic differentiation [41] and estimate $\nabla_{\eta_j}\mathbb{E}_{q(z_j;\eta_j)}[\log m_{bj}(z_j)]$ with the REINFORCE algorithm that is also the core algorithm of BBVI [51]:

$$\tilde{\nabla}_{\eta_j}\mathbb{E}_{q(z_j;\eta_j)}[\log m_{bj}(z_j)] := \frac{1}{S}\sum_{s=1}^{S} \nabla_{\eta_j} \log q\left(z_j^{(s)}; \eta_j\right) \log m_{bj}\left(z_j^{(s)}\right), \tag{5.12}$$

where $z_j^{(s)} \sim q(z_j; \eta_j)$.

In Section 5.4.4, we will demonstrate that approximate messages $\nu_{bj}(z_j)$ ease hybrid inference procedures in message passing-based PPLs.

5.2.3 CVI for Generality

Above, we addressed the case that $m_{bj}(z_j)$ is available in closed form but differs from $m_{jb}(z_j)$ in sufficient statistics. However, there might be cases that $m_{bj}(z_j)$ is not available in the PPL either because calculations do not have analytical solutions or due to missing message passing rule implementations, as illustrated in Figure 2.5b. To address this problem, we propose a strategy harnessing the existing

deterministic message passing rules at the utmost level. Our strategy is based on a decomposition of the factor $f_b(\boldsymbol{z_b})$ as

$$f_b(\boldsymbol{z_b}) = \int \underbrace{\delta(z_i - g(\boldsymbol{z}_{c\setminus i}))}_{f_c(\boldsymbol{z_c})} f_d(\boldsymbol{z}_d)\,\mathrm{d}z_i, \tag{5.13}$$

where z_i is an auxiliary random variable between the factors f_c and f_d, $g(\boldsymbol{z}_{c\setminus i})$ is a generic, deterministic function that maps the variables $\boldsymbol{z}_{c\setminus i}$ to z_i and accounts for generality in model specifications. f_b is illustrated as a composite node in Figure 5.1. We require that $f_d(\boldsymbol{z_d})$ is a factor, on which message passing rules, such as VMP, are defined and arise proportional to the exponential family of distributions, i.e., f_d allows the terms $\boldsymbol{z_d}$ to be arranged as

$$f_d(\boldsymbol{z_d}) \propto \exp\left(\phi_{di}(z_i)^\intercal \lambda_{di}(\boldsymbol{z}_{d\setminus i})\right), \tag{5.14}$$

where $\lambda_{di}(\boldsymbol{z}_{d\setminus i})$ is a function of all the arguments $\boldsymbol{z}_d$ but z_i, which leads to a VMP message

$$m_{di}(z_i) \propto \exp\left(\mathbb{E}_{q(\boldsymbol{z}_{d\setminus i})}[\log f_d(\boldsymbol{z}_d)]\right) \propto \exp\left(\phi_{di}(z_i)^\intercal \underbrace{\mathbb{E}_{q(\boldsymbol{z}_{d\setminus i})}[\lambda_{di}(\boldsymbol{z}_{d\setminus i}))]}_{\eta_{di}}\right). \tag{5.15}$$

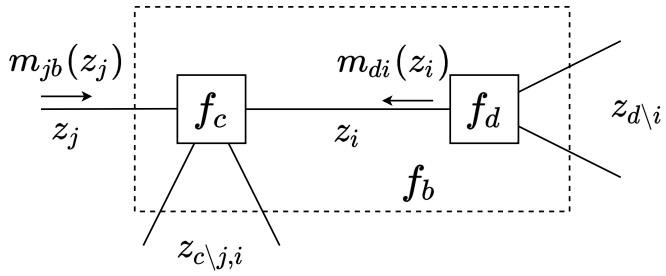


Figure 5.1: A factor node $f_b(\boldsymbol{z}_b)$, visualized as a composite node, where $\boldsymbol{z}_{b\setminus j} = \boldsymbol{z}_{c\setminus\{j,i\}} \cup \boldsymbol{z}_{d\setminus i}$. In case the message $m_{bj}(z_j)$ is not defined in closed form for the factor f_b, we require the end-user to define $f_b(\boldsymbol{z}_b)$ as a composite node such that the components of f_b are $f_c(\boldsymbol{z}_c) = \delta(z_i - g(\boldsymbol{z}_{c\setminus i}))$ and $f_d(\boldsymbol{z}_d)$. $g(\boldsymbol{z}_{c\setminus i})$ is a custom deterministic function defined by the end-user. We do not put any restrictions on $g(\boldsymbol{z}_{c\setminus i})$ and hence allow the end-user to define almost universal model specifications. We require f_d to be a factor registered in the PPL together with the message passing rules on it.

We also require $q(\boldsymbol{z}_{b\setminus j})$ to be factorized as $q(\boldsymbol{z}_{b\setminus j}) = q(\boldsymbol{z}_{c\setminus\{j,i\}})q(\boldsymbol{z}_{d\setminus i})$, where $q(\boldsymbol{z}_{c\setminus\{j,i\}})$ and $q(\boldsymbol{z}_{d\setminus i})$ may contain further factorizations within themselves, but

not given explicitly. Then, the $\log$ of VMP message $m_{bj}(z_j)$ from the factor f_b to z_j evaluates to (we detail the derivation below while discussing automated Rao-Blackwellization)

$$\log m_{bj}(z_j) \propto \mathbb{E}_{q(\boldsymbol{z}_{b\setminus j})}[\log f_b(\boldsymbol{z}_b)] \propto \mathbb{E}_{q(\boldsymbol{z}_{c\setminus\{j,i\}})}[\log m_{di}(g(\boldsymbol{z}_{c\setminus i}))]\,. \tag{5.16}$$

Since $g(\boldsymbol{z}_{c\setminus i})$ is a custom function defined by the end-user, there will be no rule registered beforehand in the PPL to calculate the above expectation. Nevertheless, we resort to Monte Carlo summation to estimate it as

$$\log m_{bj}(z_j) = \log m_{cj}(z_j) \approx \frac{1}{S}\sum_{s=1}^{S} \log m_{di}\left(g\left(z_j, \boldsymbol{z}^{(s)}_{c\setminus\{j,i\}}\right)\right), \tag{5.17}$$

$$\text{where } \boldsymbol{z}^{(s)}_{c\setminus\{j,i\}} \sim q\left(\boldsymbol{z}_{c\setminus\{j,i\}}\right).$$

Once $\log m_{bj}(z_j)$ is estimated, we use CVI as in Section 5.2.2 to find an approximate message $\nu_{bj}(z_j)$ that has sufficient statistics $\phi_j(z_j)$. Notice that instead of resorting to Monte Carlo estimation at first step in $\mathbb{E}_{q(\boldsymbol{z}_{b\setminus j})}[\log f_b(\boldsymbol{z}_b)]$, we harnessed the message passing rules defined in our message passing-based PPL to reduce the number of variables to be sampled, which further reduces the variance in $\log m_{bj}(z_j)$ estimates. This approach coincides with Rao-Blackwellization, as we detail next.

5

Automated Rao-Blackwellization

Consider the composite node $f_b(\boldsymbol{z_b}) = \int \underbrace{\delta(z_i - g(\boldsymbol{z}_{c\setminus i}))}_{f_c(\boldsymbol{z_c})} \cdot f_d(\boldsymbol{z}_d)\,\mathrm{d}z_i$ visualized in Figure 5.1. As stated earlier, we assume that $q(\boldsymbol{z}_{b\setminus j}) = q(\boldsymbol{z}_{c\setminus\{j,i\}})q(\boldsymbol{z}_{d\setminus i})$ and f_d is a function amenable to be arranged as $f_d(\boldsymbol{z_d}) \propto \exp\left(\phi_{di}(z_i)^\intercal \lambda_{di}(\boldsymbol{z}_{d\setminus i})\right)$. Then $\log$ of the VMP message $m_{bj}(z_j)$ evaluates to

$$\begin{aligned}
\log m_{bj}(z_j) &\propto \mathbb{E}_{q(\boldsymbol{z}_{b\setminus j})}[\log f_b(\boldsymbol{z_b})] \\
&= \mathbb{E}_{q(\boldsymbol{z}_{b\setminus j})}\left[\log \int \delta(z_i - g(\boldsymbol{z}_{c\setminus i}))f_d(\boldsymbol{z}_d)\,\mathrm{d}z_i\right] \\
&\propto \mathbb{E}_{q(\boldsymbol{z}_{b\setminus j})}\left[\log \int \delta(z_i - g(\boldsymbol{z}_{c\setminus i}))\exp\left(\phi_{di}(z_i)^\intercal \lambda_{di}(\boldsymbol{z}_{d\setminus i})\right)\,\mathrm{d}z_i\right] \\
&= \mathbb{E}_{q(\boldsymbol{z}_{b\setminus j})}\left[\phi_{di}(g(\boldsymbol{z}_{c\setminus i}))^\intercal \lambda_{di}(\boldsymbol{z}_{d\setminus i})\right].
\end{aligned} \tag{5.18}$$

A trivial approach to estimate the above expectation is to use Monte Carlo summation by drawing samples from $q(\boldsymbol{z}_{b\setminus j})$. However, we aim to reduce the variance in our estimates by avoiding sampling and sticking to analytical solutions as much as possible. This strategy relates to Rao-Blackwellization [138], [6, Chapter 11.6.1]

and has been used in various machine learning tasks including Bayesian estimation in state space models [146, 147]. VMP rules defined in *ForneyLab* help us to carry out variance reduction with Rao-Blackwellization in an automated way:

$$\begin{aligned}\log m_{bj}(z_j) &\propto \mathbb{E}_{q(\boldsymbol{z}_{c\setminus\{j,i\}})}\left[\mathbb{E}_{q(\boldsymbol{z}_{d\setminus i}|\boldsymbol{z}_{c\setminus\{j,i\}})}\left[\phi_{di}(g(\boldsymbol{z}_{c\setminus i}))^\intercal \lambda_{di}(\boldsymbol{z}_{d\setminus i})\right]\right] \\ &= \mathbb{E}_{q(\boldsymbol{z}_{c\setminus\{j,i\}})}\left[\mathbb{E}_{q(\boldsymbol{z}_{d\setminus i})}\left[\phi_{di}(g(\boldsymbol{z}_{c\setminus i}))^\intercal \lambda_{di}(\boldsymbol{z}_{d\setminus i})\right]\right] \\ &= \mathbb{E}_{q(\boldsymbol{z}_{c\setminus\{j,i\}})}\left[\phi_{di}(g(\boldsymbol{z}_{c\setminus i}))^\intercal \mathbb{E}_{q(\boldsymbol{z}_{d\setminus i})}\left[\lambda_{di}(\boldsymbol{z}_{d\setminus i})\right]\right] \\ &\propto \mathbb{E}_{q(\boldsymbol{z}_{c\setminus\{j,i\}})}[\log m_{di}(g(\boldsymbol{z}_{c\setminus i}))], \end{aligned} \tag{5.19}$$

where $m_{di}(\cdot)$ is the VMP message defined from f_d to z_i. The second line above follows from $q(\boldsymbol{z}_{b\setminus j}) = q(\boldsymbol{z}_{c\setminus\{j,i\}})q(\boldsymbol{z}_{d\setminus i})$. The number of variables to be sampled for the estimation of $\log m_{bj}(z_j)$ in (5.19) is less than (5.18). The message passing framework of *ForneyLab* equips us with the tools to carry out analytical calculations automatically. This feature is missing in many other PPLs.

5

CVI around Deterministic Nodes

Now, we provide the CVI algorithm around the deterministic node f_c. Given $f_b(\boldsymbol{z}_b)$ is decomposed as (5.13), we carry out CVI by defining message passing rules around the deterministic component $f_c(\boldsymbol{z}_c) = \delta(z_i - g(\boldsymbol{z}_{c\setminus i}))$ by imposing a mean field assumption on $q(\boldsymbol{z}_{c\setminus i})$:

$$q(\boldsymbol{z}_{c\setminus i}) = \prod_{\substack{j\in\mathcal{E}(c)\\ j\neq i}} q(z_j). \tag{5.20}$$

We provide a high level summary for CVI around the deterministic node $f_c(\boldsymbol{z}_c)$ in Algorithm 4.

Algorithm 4 CVI around a deterministic node in an FFG

Require: A graph $\mathcal{G}(c) = (c, \mathcal{E}(c))$ s.t. $f_c(\boldsymbol{z}_c) = \delta(z_i - g(\boldsymbol{z}_{c\setminus i}))$;
  Number of iterations: T_j for all $j \in \mathcal{E}(c), j \neq i$;
  Number of samples: S
  for all $j \in \mathcal{E}(c)$ **do**
    Collect $m_{jc}(z_j) \propto \exp\left(\phi_{jc}(z_j)^\intercal \eta_{jc}\right)$
  end for
  for $j \in \mathcal{E}(c), j \neq i$ **do**
    Estimate $\log m_{cj}(z_j)$ as in (5.17)
    Set $\eta_j^{(0)} \longleftarrow \eta_{jc}$
    for $t = 1 : T_j$ **do**
      Calculate $\tilde{\nabla}^N_{\eta_j} \mathcal{F}(\eta_j)$ {See Section 5.2.2}
      Set a step size $\rho^{(t)}$
      Update η_j using (5.1)
    end for
    Set $q(z_j) \propto \exp\left(\phi_j(z_j)^\intercal \eta_j^{(T_j)}\right)$
    Set $\nu_{cj}(z_j) \propto \exp(\phi_j(z_j)^\intercal(\eta_j^{(T_j)} - \eta_{jc}))$
  end for
  Set $q(z_i) = \left\{ g\left(\boldsymbol{z}_{c\setminus i}^{(s)}\right) \;\middle|\; \text{for } s \in \{1, \ldots, S\} \right\}$ where $\boldsymbol{z}_{c\setminus i}^{(s)} \sim q(\boldsymbol{z}_{c\setminus i})$ {$q(\boldsymbol{z}_{c\setminus i})$ is given in (5.20)}

5

Algorithm 4 is defined for a generic case with multiple input function g. In case the number of input variables is 1, i.e., $\left|\boldsymbol{z}_{c\setminus i}\right| = 1$, the algorithm simplifies further since $\log m_{cj}(z_j)$ is available in closed form and no Monte Carlo summation is needed to estimate it. By setting the deterministic node to an identity function, the end-user of our PPL can run CVI for non-conjugate inference with known messages as in Section 5.2.2.

CVI seamlessly interfaces with deterministic message passing procedures. Consider a composite likelihood node accounts for complex observations through a non-linear deterministic node. Running CVI on this deterministic node, the approximate messages $\nu_{cj}(z_j)$ are ready to interface with BP and EP procedures. Similarly, the approximate marginals $q(z_j)$ and $q(z_i)$ that are estimated in Algorithm 4 allow VMP messages to be computed in neighboring factor nodes. Notice that in the last line of Algorithm 4, we set $q(z_i)$ to a set of samples, which allows expectation quantities in VMP message calculations to be estimated with Monte Carlo summation, automatically as in Chapter 3, similarly to [120].

In Algorithm 4, we make use of the CVI algorithm to allow almost universal model specifications and inference with non-conjugate factor pairs. For the sake of brevity, we skip the details for scalability and online variational inference related solutions of CVI that are analogous to the SVI algorithm and implemented in our

framework.

5.3 Beyond Exponential Family of Distributions

Stochastic approximation methods for variational inference within the message passing framework can be extended beyond the exponential family of distributions by exchanging the natural gradient descent optimization scheme for raw stochastic gradient descent. As discussed in Section 5.1, many PPLs automate variational inference through stochastic gradient descent rather than natural gradient descent. Here, we demonstrate how to interface these stochastic gradient descent-based methods with message passing, which allows us to cast the stochastic optimization of the free energy as a distributed fallback operation to be resorted to when necessary.

Consider a node $f_b(\boldsymbol{z}_b)$, once again. This time, we assume that the incoming message $m_{jb}(z_j)$ on edge j to the node f_b is not a member of the exponential family of distributions, and we want to find an approximate posterior $q(z_j;\eta_j)$ in the same distribution family with $m_{jb}(z_j)$. Below, we briefly mention two main approaches to estimate the noisy gradients of the free energy with respect to the parameters η_j of the approximate posterior. Note that this time η_j does not refer to natural parameters as $q(z_j;\eta_j)$ is not a member of the exponential family of distributions. Moreover, this time we will not find an approximate message $\nu_{bj}(z_j)$ since the division given in (5.11) is not straightforward to calculate. Instead, we will just find approximate marginals $q(z_j)$ that will be used to calculate VMP messages around the factors in $\mathcal{V}(j)$.

5.3.1 Efficient Black-Box Variational Inference

Black-box variational inference (BBVI) [51] is a universal approximate inference technique that employs the REINFORCE algorithm, otherwise known as score function estimator, in the estimation of the free energy gradients [148]. We use REINFORCE algorithm in (5.12) to estimate natural gradients. Below refers to the noisy gradient estimation of the free energy w.r.t. η_j:

$$\tilde{\nabla}_{\eta_j}\mathcal{F}(\eta_j) := \frac{1}{S}\sum_{s=1}^{S}\nabla_{\eta_j}\log q(z_j^{(s)};\eta_j)\left[\log q(z_j^{(s)};\eta_j) - \log m_{jb}(z_j^{(s)}) - \log m_{bj}(z_j^{(s)})\right], \tag{5.21}$$

where $z_j^{(s)} \sim q(z_j)$.

This is an efficient realization of BBVI because message passing automatically carries out Rao-Blackwellization as discussed and reduces variance in gradient estimations.

5.3.2 Reparameterization Gradient Message Passing

REINFORCE is a broadly applicable estimator for gradients of expectation of functions. However, the variance of the REINFORCE estimator is sometimes too high to be practical [78, 149]. An alternative approach is based on reparameterization of the random variables and has been employed in a wide variety of machine learning tasks ranging from deep generative modeling [85,86] to variational inference [87]. Let us rewrite the local free energy function:

$$\mathcal{F}(\eta_j) \triangleq \mathbb{E}_{q(z_j;\eta_j)}\left[\log \frac{q(z_j;\eta_j)}{\tilde{f}(z_j)}\right] = \int q(z_j;\eta_j) \log \frac{q(z_j;\eta_j)}{\tilde{f}(z_j)}\,\mathrm{d}z_j$$

where $\tilde{f}(z_j) = m_{jb}(z_j) \cdot m_{bj}(z_j)$.

One might think that the gradient could be estimated by a Monte Carlo approximation $\tilde{\nabla}_{\eta_j} F(\eta_j) = \frac{1}{S}\sum_{s=1}^{S} \nabla_{\eta_j} \log[q(z_j^{(s)};\eta_j)/\tilde{f}(z_j^{(s)})]$ where $\{z_j^{(s)}\}_{s=1}^{S}$ is a set of samples from $q(z_j;\eta_j)$. Unfortunately, the noisy measurements from $\log[q(z_j;\eta_j)/\tilde{f}(z_j)]$ can not be taken without loosing some information about the variational parameters η_j. In other words, once $z_j^{(s)}$ sampled, the term $\log \tilde{f}(z_j^{(s)})$ is not a function of η_j anymore and its gradient $\nabla_{\eta_j} \log \tilde{f}(z_j^{(s)})$ becomes zero. The reprameterization trick deals with this problem by generating the $z_j^{(s)}$ samples from a differentiable process of dummy random variables $\epsilon^{(s)}$:

$$\epsilon^{(s)} \sim p(\epsilon) \tag{5.22a}$$

$$z_j^{(s)} = h(\epsilon^{(s)};\eta_j) \tag{5.22b}$$

$$q(z_j;\eta_j) = |\mathrm{J}_{h^{-1}}(z_j;\eta_j)|\, p(h^{-1}(z_j;\eta_j)), \tag{5.22c}$$

where $h(\epsilon;\eta_j)$ is an injective, differentiable function of a random variable ϵ, $h^{-1}(\cdot;\eta_j)$ is its inverse, $\mathrm{J}_{h^{-1}}(z_j;\eta_j)$ is the Jacobian of h^{-1} evaluated at z_j for multidimensional ϵ and z_j and $|\mathrm{J}_{h^{-1}}(z_j;\eta_j)|$ is the determinant of the Jacobian, [85–87, 150].

The gradient of the local free energy can now be estimated by Monte Carlo approximation because it can be expressed as an expectation of the gradient:

$$\nabla_{\eta_j} F(\eta_j) = \mathbb{E}_{p(\epsilon)}\left[\nabla_{\eta_j} \log \frac{p(\epsilon)\,|\mathrm{J}_{h^{-1}}(z_j;\eta_j)|}{\tilde{f}(h(\epsilon;\eta_j))}\right]. \tag{5.23}$$

The above expression is further simplified by discarding the terms that do not include the variational parameters [87], and the result is called the *reparameteriza-*

tion gradient [85, 86]:

$$\begin{aligned}\nabla_{\eta_j} F(\eta_j) &= -\mathbb{E}_{p(\epsilon)}\left[\nabla_{\eta_j} \log |\mathrm{J}_h(\epsilon;\eta_j)| + \nabla_{\eta_j} \log \tilde{f}(h(\epsilon;\eta_j))\right] \\ &= -\mathbb{E}_{p(\epsilon)}\left[\nabla_{\eta_j} \log |\mathrm{J}_h(\epsilon;\eta_j)| + \nabla_{\eta_j} \log m_{jb}\left(h(\epsilon;\eta_j)\right) + \nabla_{\eta_j} \log m_{bj}\left(h(\epsilon;\eta_j)\right)\right] .\end{aligned} \tag{5.24}$$

5.3.3 Illustrative Example

In the upcoming section, we will demonstrate how SVI and CVI perform as probabilistic programming algorithms in a message passing-based PPL. But, first, we want to illustrate a hybrid message passing inference procedure on a simple example to demonstrate how to combine the reparameterization trick and the REINFORCE estimator in the message passing context. Consider the factor graph in Figure 5.2, which depicts the following probabilistic model.

$$\begin{aligned} p(\boldsymbol{y}, \boldsymbol{x}, w) &= p(w) \prod_{n=1}^{N} p(x_n|w) p(y_n|x_n), \text{ where} \\ p(w) &= \mathcal{G}a(w; 5, 1) \\ p(x_n|w) &= \mathcal{N}(x_n; s_n, 1/w) \\ p(y_n|x_n) &= \mathrm{T}\mathcal{N}(y_n; x_n, 0.25, 0, 10)\,. \end{aligned} \tag{5.25}$$

In (5.25), $\mathcal{G}a(\cdot; a, b)$ is Gamma distribution with shape a and rate b, $\mathcal{N}(x_n; s_n, 1/w)$ is a Normal distribution on x_n with known exogenous mean s_n and precision w, $\mathrm{T}\mathcal{N}(\cdot; \mu, v, 0, 10)$ is a truncated normal distribution with upper and lower limits 10 and 0, where μ is mean and v is variance.

In this probabilistic model, we are interested in the posterior marginals $p(w|\mathbf{y})$ and $p(x_j|\mathbf{y})$. Unfortunately, BP messages $m_{cj}(w_j)$ necessitate integrals that are hard to calculate. Hence, instead of running exact BP in this tree-like FFG, we shall resort to an EP-like iterative procedure to find approximate marginals $q(w)$ and $q(x_j)$.

Recall from Section 2.4 that we initialize the EP procedure around the equality node f_b by initializing the messages $\nu_{ib}(w_i)$ for $i = 0 : N$. The message $\nu_{0b}(w_0)$ is the prior $p(w)$ as we relate w_0 with w on the FFG. All other messages $\nu_{ib}(w_i)$ are initialized with uninformative Gamma distributions $\mathcal{G}a(w_i; 0.01, 0.01)$. Given the messages $\nu_{ib}(w_i)$, we can now analytically calculate the message on edge j propagating from f_b by

$$\nu_{bj}(w_j) \propto \prod_{\substack{i \in \mathcal{E}(b) \\ i \neq j}} \int \delta(w_j - w_i) \nu_{ib}(w_i)\, \mathrm{d}w_i\,. \tag{5.26}$$

5

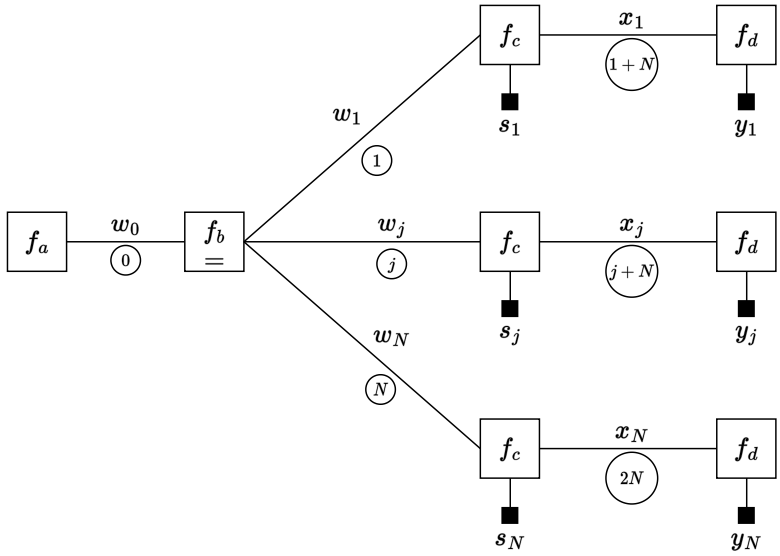


Figure 5.2: An FFG representation of (5.25). The factors f_a, f_c, f_d respectively stand for $p(w)$, $p(x_n|w_n)$ and $p(y_n|x_n)$. The variable w_0 relates to w in (5.25) and branches out through the equality factor f_b. The variables $w_1, \ldots, w_N$ are copies of w_0. Edges are enumerated by the numbers in circles visualized below edges.

5

The message $\nu_{bj}(w_j)$ is a Gamma distribution $\mathcal{G}a(w_j; a_{bj}, b_{bj})$. Following the EP steps defined in Section 2.4, we are now supposed to calculate

$$\bar{q}(w_j) \propto \nu_{bj}(w_j) \underbrace{\int f_c(x_j, w_j) m_{j+N,c}(x_j)\, \mathrm{d}x_j}_{m_{jb}(w_j)}, \tag{5.27}$$

where $f_c(x_j, w_j)$ stands for $p(x_j|s_j, 1/w_j)$ with known s_j. However, we mentioned that the integral in calculation of $m_{jb}(w_j)$ impedes the BP algorithm. Hence, we shall approximate it. For the approximation of $m_{jb}(w_j)$, we first introduce a joint approximate factor $q(w_j, x_j)$:

$$q(w_j, x_j) \approx f(w_j, x_j) = \nu_{bj}(w_j) \cdot f_c(x_j, w_j) \cdot m_{j+N,c}(x_j)\,. \tag{5.28}$$

Now let us evaluate the marginal approximate factor $q(x_j)$:

$$q(x_j) \approx f(x_j) = m_{j+N,c}(x_j) \underbrace{\int \nu_{bj}(w_j) \cdot f_c(x_j, w_j) \mathrm{d}w_j}_{m_{c,j+N}(x_j)}\,. \tag{5.29}$$

Above, BP message $m_{c,j+N}(x_j)$ is available in closed form and proportional to Student's t distribution:

$$m_{c,j+N}(x_j) \propto \mathcal{S}\mathrm{t}(x_j; \mu_{cj}, \sigma_{cj}, n_{cj})\,, \tag{5.30}$$

where $\mathcal{S}\mathrm{t}(x_j; \mu_{cj}, \sigma_{cj}, n_{cj})$ is a Student's t distribution with location μ_{cj}, scale σ_{cj} and degrees of freedom n_{cj}. Given that $\nu_{bj}(w_j) = \mathcal{G}a(w_j; a_{bj}, b_{bj})$, we find $\mu_{cj} = s_j$, $\sigma_{cj} = \sqrt{b_{cj}/a_{cj}}$ and $n_{cj} = 2a_{bj}$.

Now, let us assume the joint approximate distribution $q(w_j, x_j)$ factorizes as

$$q(w_j, x_j) = q(w_j) \cdot q(x_j)\,. \tag{5.31}$$

Then we can approximate the BP message $m_{jb}(w_j)$ in (5.27) with a VMP message

$$\begin{aligned} m_{jb}(w_j) &\propto \exp\left(\mathbb{E}_{q(x_j)}[\log f_c(w_j, x_j)]\right) \\ &\propto \mathcal{G}a(w_j, 1.5, 0.5(\mathbb{V}_{q(x_j)}[x_j] + (s_j - \mathbb{E}_{q(x_j)}[x_j])^2))\,. \end{aligned} \tag{5.32}$$

Since the VMP message $m_{jb}(w_j)$ is already in the same distribution family as $\nu_{bj}(w_j)$, we can set $\nu_{jb}(w_j) = m_{jb}(w_j)$ without a need for moment matching. Running this procedure iteratively for all the factors connected to f_b, we find approximate marginals $q(w)$ and $q(x_i)$ for $i = 1 : N$.

So far, we have summarized the overall inference procedure for the model defined in (5.25). However, we have not mentioned how to calculate $q(x_j)$ in (5.29), yet. Recall that the forward message $m_{c,j+N}(x_j)$ is not in exponential family of distributions and as it is customary in Bayesian inference, we will approximate $q(x_j; \eta_j)$ within the distribution family of the forward message that is Student's t distribution, i.e., $q(x_j; \eta_j) = \mathcal{S}\mathrm{t}(x_j; \mu_j, \sigma_j, n_j)$, where $\eta_j = [\mu_j, \sigma_j, n_j]^\intercal$. We will use stochastic optimization to tune η_j, where the gradient of the local free energy is $\nabla_{\eta_j}\mathcal{F} = [\frac{\partial \mathcal{F}}{\partial \mu_j}, \frac{\partial \mathcal{F}}{\partial \sigma_j}, \frac{\partial \mathcal{F}}{\partial n_j}]^\intercal$. In estimation of the first two partial derivatives, $\frac{\partial \mathcal{F}}{\partial \mu_j}$ and $\frac{\partial \mathcal{F}}{\partial \sigma_j}$, we will use the reparameterization trick (5.24) since sampling from $\mathcal{S}\mathrm{t}(x_j; \mu_j, \sigma_j, n_j)$ can be performed by the following procedure:

$$x_j^{(s)} = \mu_j + \sigma_j \epsilon^{(s)}, \text{ where } \epsilon^{(s)} \sim \mathcal{S}\mathrm{t}(0, 1, n_j)\,. \tag{5.33}$$

In calculation of $\frac{\partial \mathcal{F}}{\partial n_j}$, on the contrary, we use the REINFORCE estimator (5.21) as we can not reparameterize $x_j^{(s)}$ as a function of n_j, explicitly.

For the empirical test of the described inference procedure, we use $N = 4$ data samples generated by using $s_1 = -5$, $s_2 = -0.2$, $s_3 = 15$, $s_4 = 6$. The perturbated inputs are $x_1 = -2.62$, $x_2 = 3.9$, $x_3 = 17.3$, $x_4 = 6.92$. The noisy output measurements are $y_1 = 0.07$, $y_2 = 4.39$, $y_3 = 9.99$, $y_4 = 6.82$. We run our EP-like updates for 10 iterations. For the calculation of $q(x_j)$, we run stochastic optimization for 1000 steps with the ADAM optimizer [135]. The estimates for $q(x_j)$ are visualized in Figure 5.3 together with the actual perturbated input values.

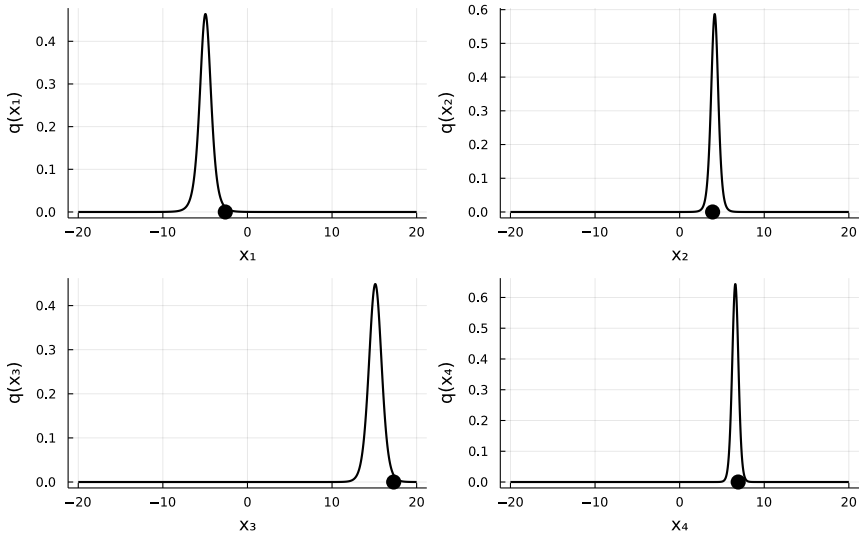


Figure 5.3: The estimates for $q(x_j)$ and the actual perturbated inputs. Estimations are performed with a stochastic optimization procedure within an EP-like message passing scheme. The approximating distribution families for $q(x_j)$ are chosen Student's t distribution. The gradients to be used in stochastic optimizations are estimated by the combination of a reparameterization trick and the REINFORCE estimator.

Note that having estimated $q(x_j)$, the VMP messages in (5.32) are available in closed form, since the mean and the variance of x_j under Student's t distribution are analytically defined.

5.4 Experiments

In this section, we show the effectiveness of the SVI and CVI implementations in the message passing-based PPL *ForneyLab.jl*.

5.4.1 Gaussian Mixture Model

SVI is meant to be beneficial when working with gigantic data sets that can not be processed at once as needed in VMP. In this experiment, however, we aim at validating that our SVI implementation in ForneyLab is functioning as expected in theory. Therefore, we use a small data set to run VMP and use its free energy as a performance benchmark. We measure the performance of the SVI over a Gaussian

Mixture Model (GMM) [76, Chapter 20] for the Iris data set [151, 152] after reducing the dimensionality of the data samples from 4 to 2 by Principal Component Analysis [75, Chapter 12]. The Iris data set comprises 150 data samples, equally distributed among three classes. We define the GMM by

$$p(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{W}, s) = p(s) \prod_{k=1}^{3} p(\mu_k) p(W_k) \prod_{n=1}^{150} p(z_n|s) p(y_n|z_n, \boldsymbol{\mu}, \boldsymbol{W}), \tag{5.34}$$

$$\begin{aligned} p(s) &= \mathcal{D}\mathrm{ir}(s; [50, 50, 50]) \\ p(\mu_k) &= \mathcal{N}(\mu_k; \mathbf{0}_2, \boldsymbol{I}_{2\times 2}) \\ p(W_k) &= \mathcal{W}_2(W_k; \boldsymbol{I}_{2\times 2}, 2) \\ p(z_n|s) &= \mathcal{C}at(z_n; s) \end{aligned}$$

$$p(y_n|z_n, \boldsymbol{\mu}, \boldsymbol{W}) = \prod_{k=1}^{3} \mathcal{N}\left(y_n; \mu_k, W_k^{-1}\right)^{\mathbb{I}[z_n=k]}, \tag{5.35}$$

where $\mathcal{N}$, $\mathcal{W}$, $\mathcal{D}\mathrm{ir}$, $\mathcal{C}at$ stand for Gaussian, Wishart, Dirichlet and Categorical distributions respectively. $\mathbf{0}_2$ is two dimensional vector of zeros and $\boldsymbol{I}_{2\times 2}$ is two by two identity matrix. The Iverson bracket, $\mathbb{I}[z_n = k]$, is an indicator function that takes the value one if the equality is satisfied, and zero otherwise. All the factors given above are registered in our PPL including $p(y_n|z_n, \boldsymbol{\mu}, \boldsymbol{W})$, which is called GMM likelihood node. We approximate the true posterior $p(\boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{W}, s|\boldsymbol{y})$ by a fully factorized

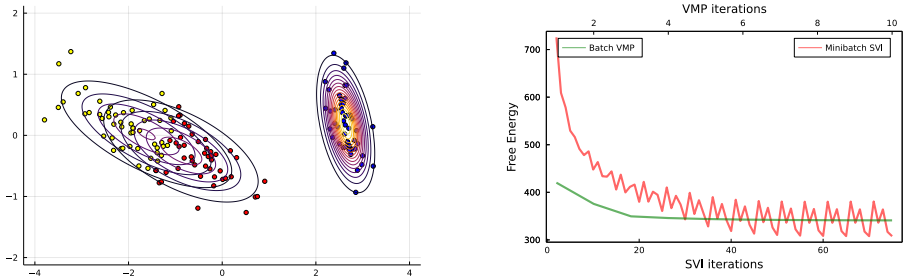
$$q(\boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{W}, s) = q(s) \prod_{k=1}^{3} q(\mu_k) q(W_k) \prod_{n=1}^{150} q(z_n). \tag{5.36}$$

For SVI, we randomly split 150 data samples into five mini-batches equal in size to process per iteration. The estimations with stochastic VMP are visualized in Figure 5.4. We use the mean estimates for $q(\mu_k)$ and $q(W_k)$ to set the mean and precision parameters of the visualized clusters. The cluster assignments for data samples are shown in red, blue, and yellow. To color-code the data samples on the plot, we use the maximum of $q(z_n)$. On the right-hand side of Figure 5.4, we see that SVI performs on par with VMP in terms of free energy. Notice that whereas SVI employs 30 data samples per iteration, VMP uses all 150 of them. Thus, ForneyLab can be run in SVI mode instead of VMP to carry out inference on models require working with large data sets.

5.4.2 Tracking a Non-stationary Process

In this experiment, we demonstrate how stochastic optimization enables us to track a non-stationary process. For this purpose, we use a coal mining accidents data set [153], visualized with black points in Figure 5.5.

We model the number of accidents with a Poisson likelihood, i.e., $p(y_t|z_t) = \mathcal{P}o(y_t; z_t)$ and aim at estimating the rate z_t to get the notion of policies regarding the safety regulations in mining sites. At first, we postulate that the safety policies do not change and the rate is shared among all the likelihoods, i.e., $z_t = z$ for all t. We put a shape-rate parameterized Gamma prior $p(z) = \mathcal{G}a(z; 1, 1)$ on z. We run BP in an online setting, processing the number of accidents one by one and updating the prior $p(z)$ at each time step with the posterior estimated in the previous time step. We visualize the mean estimations with a blue curve in Figure 5.5.



(a) GMM estimations with SVI

(b) Free energy comparison

Figure 5.4: Visualization of the marginal posterior and free energy estimates with SVI. These results verify that our SVI implementation in ForneyLab performs as expected for this model.

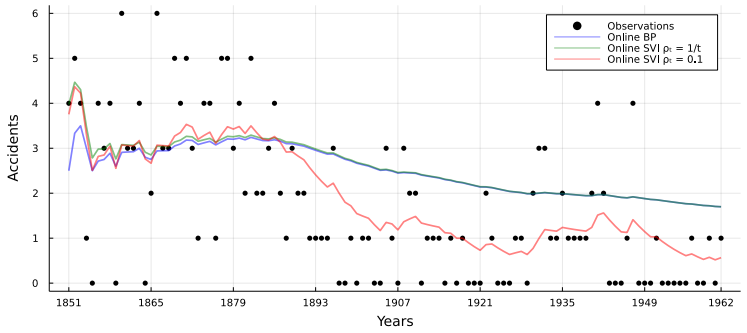


Figure 5.5: Coal mining accidents in the United Kingdom from 1851 to 1962. In 1887, new safety regulations are exerted to prevent accidents on mining sites. We show that it is possible to achieve online BP using SVI with a step size satisfying the Robins-Monro conditions. Violating the Robins-Monro conditions and keeping the step size fixed over iterations, we are able to track the hidden non-stationary process shown by the red curve.

Next, we investigate the behavior of stochastic approximation for variational inference. We set the distribution family of $q(z;\eta)$ to the Gamma distribution family, same with $p(z)$. Notice that $p(y_t|z_t)$ and $p(z)$ are conjugate factor pairs, thus the natural gradient of the free energy with respect to η is available in closed form and hand-coded in our PPL through SVI. Therefore, by running SVI in ForneyLab, we can investigate the inference with NGD over the free energy objective. In Section 5.2, we discussed that the step size $\rho^{(t)}$ must satisfy Robins-Monro conditions for convergence. Setting it to $1/t$ for $t = 1 : 112$, we satisfy Robins-Monro conditions and the estimations coincide with online BP.

So far, we treated the example as if $z_t = z$ for all t. However, countries change their safety regulations over time and the assumption that $z_t = z$ does not reflect the true process well. We may consider enriching our model specification as in [154]. However, this new model may complicate our automated inference procedure and lead to unsatisfactory estimations. Instead of inserting the changes in z_t explicitly within a new model specification, we retain our simple model as it is and implicitly treat the problem at hand as a non-stationary process. We achieve this by violating the Robins-Monro conditions and setting $\rho^{(t)} = \rho = 0.1$ for all t. This is a widely preferred approach in bandit problems to track non-stationary hidden rewards [155]. Keeping $\rho^{(t)}$ fixed over time weighs the contributions from recent observations more than earlier observations in updating η through (5.1). The mean of $q(z)$ over time is visualized by a red curve in Figure 5.5. Note that the mean is around 3 until the 1890s, which steadily declines to around 1 later on. This analysis estimates a policy change just before the 1890s, which is indeed the case: authorities in the UK exerted new safety regulations in 1887 to prevent accidents in mining sites. Regarding the mean estimates around 3 and 1, a Gibbs sampling over a change point model gives similar estimations [156]. This experiment supports the notion that the devised stochastic message passing algorithms enable us to go beyond conventional inference approaches in message passing-based PPLs.

5.4.3 Hierarchical Probabilistic Modelling with a Non-conjugate Prior

In this experiment, we build a hierarchical probabilistic model with a non-conjugate prior to test the performance of the CVI implementation in ForneyLab. Inspired by the famous eight school example from [157], we introduce a slightly different hierarchical model to analyze the effects of eight special coaching programs on the SAT score of students. In our experiment, we assume that we work with students' data

who take the exam for a second time after attending a special coaching program:

$$p(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w}, \alpha, \beta, \mu, s) = p(\alpha)p(\beta)p(\mu)p(s)\prod_{i=1}^{8} p(x_i|\mu, s)p(w_i|\alpha, \beta)\prod_{n=1}^{N_i} p(y_{in}|x_i, w_i), \tag{5.37}$$

$$\begin{aligned} p(\alpha) &= \mathcal{G}a(\alpha; 0.1, 0.1) \\ p(\beta) &= \mathcal{G}a(\beta; 0.1, 0.1) \\ p(\mu) &= \mathcal{N}(\mu; 0, 10) \\ p(s) &= \mathcal{G}a(\alpha; 0.1, 1) \\ p(x_i|\mu, s) &= \mathcal{N}(x_i; \mu, 1/s) \\ p(w_i|\alpha, \beta) &= \mathcal{G}a(w_i; \alpha, \beta) \\ p(y_{in}|x_i, w_i) &= \mathcal{N}(y_{in}; x_i, 1/w_i). \end{aligned}$$

We change the original problem and model specification in [157] to introduce a non-conjugacy that stems from $p(\alpha)$ in our model specification. In this model, we aim at analyzing the effect of special coaching in general by estimating the global variables α, β, μ and s that are shared among eight schools. We also desire to estimate the effect of the schools individually by estimating the local variables x_i and w_i. Our model differs from the original model specification in that we make an analysis over participants' SAT scores taken before and after the special coaching. We denote the change in the SAT score of the n^{th} participant of the i^{th} school with y_{in}. We generate y_{in} values from Normal distributions parameterized with means and standard errors given in [157, Table 5.2].

5

For the inference, we make the mean-field factorization assumption in the approximate posterior:

$$q(\boldsymbol{x}, \boldsymbol{w}, \alpha, \beta, \mu, s) = q(\alpha)q(\beta)q(\mu)q(s)\prod_{i=1}^{8} q(x_i)q(w_i). \tag{5.38}$$

In ForneyLab, we run VMP for 10 iterations. We tie α to an identity deterministic function $g(\alpha) = \alpha$ just to execute NGD variational inference for the non-conjugate section of the factor graph and to estimate $q(\alpha)$ as a member of Gamma distribution family with CVI. At each VMP iteration, the natural parameters of $q(\alpha)$ are updated by NGD with ADAM optimizer for 10000 iterations.

For the comparison, we use the ADVI inference engine of Turing. We observe that ADVI converges in 5000 iterations with forward-mode automatic differentiation [158] and the default optimizer set by Turing. The run time and free energy comparisons are given in Table 5.1[1]. We see that Turing and ForneyLab perform

[1]Specs of the computer: Julia v1.5.3, Turing v0.18.0, 7 GHz Quad-Core Intel Core i7 CPU, 6 GB 2133 MHz RAM

	Hierarchical model		Sensor fusion	
	Run time	Free energy	Run time	Free energy
ForneyLab with CVI	10.961	209.127	6.820	94.784
Turing with ADVI	0.969	208.244	27.238	104.374

Table 5.1: Run time (sec.) and free energy comparisons for the hierarchical model and the sensor fusion experiments.

almost equally well in terms of free energy, while Turing outperforms ForneyLab in run time. Nevertheless, this experiment validates the quality in our estimates with the CVI implementation in ForneyLab and encourages us to test it in a state space model, where we can take full advantage of the deterministic message passing rules of ForneyLab. The next experiment focuses on a state space model example.

5.4.4 Sensor Fusion

In this experiment, we show how ForneyLab casts stochastic optimization for variational inference as an efficient, distributed operation. We use a variant of a sensor fusion example given in [156, Example 3]. Assume an object moves in a two-dimensional environment where three noisy sensors are set in pre-specified locations: $\xi_{1,2,3}$. At a discrete time t, each sensor measures the Euclidean distance $||\xi_i - h_t||$ between the object's position h_t and itself. Our task is to estimate the position of the moving object over time using noisy sensory measurements.

Smoothing with Fixed Model Parameters

We build a state space model using Newtonian dynamics to model the transitions. At first, we use fixed transition and measurement noise matrices in our model specification:

$$
\begin{aligned}
p(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{z}) &= p(z_1)p(x_1|z_1)p(y_1|x_1)\prod_{t=2}^{T} p(z_t|z_{t-1})p(x_t|z_t)p(y_t|x_t), \text{ where} \\
p(z_1) &= \mathcal{N}(z_1; \mathbf{0}_4, \boldsymbol{I}_{4\times4}) \\
p(z_t|z_{t-1}) &= \mathcal{N}(z_t; Az_{t-1}, \boldsymbol{I}_{4\times4}) \\
p(x_t|z_t) &= \delta(x_t - g(z_t)) \text{ with } g(z_t) = [||\xi_1 - Bz_t||, ||\xi_2 - Bz_t||, ||\xi_3 - Bz_t||]^\intercal \\
p(y_t|x_t) &= \mathcal{N}(y_t; x_t, \boldsymbol{I}_{3\times3}),
\end{aligned}
\tag{5.39}
$$

where z_t is the vector of hidden position h_t and velocity values, $A = \left[\begin{array}{c|c} I_{2\times2} & I_{2\times2} \\ \hline 0_{2\times2} & I_{2\times2} \end{array}\right]$ and $B = \left[\begin{array}{c|c} I_{2\times2} & 0_{2\times2} \end{array}\right]$. In this model, the non-conjugacy stems from the deterministic function $g(z_t)$. We circumvent the non-conjugacy issue by incorporating the

CVI algorithm into our message passing procedure through the factors $p(x_t|z_t)$. At a given time step t, we run the CVI algorithm around $p(x_t|z_t)$ for 100 iterations with a descent optimizer with learning rate 0.1 to find a message towards the equality node that connects z_t to $p(x_t|z_t)$ (see Figure 5.6 for the visualization of a closely related model). The approximate message combines with BP messages at the equality node to construct the forward and backward messages towards the $p(z_t|z_{t-1})$ factors. Therefore, in *ForneyLab*, the number of parameters to be estimated by stochastic approximation scales linearly with T and the rest of the computation is carried out with deterministic BP messages.

We generate synthetic data with $T = 15$ and compare ForneyLab's performance with ADVI of Turing. We define a fully structured Gaussian approximate posterior $q(\boldsymbol{z})$ to be estimated with ADVI as it is in ForneyLab. Whereas ForneyLab estimates the structured approximate distribution with distributed operations through message passing, ADVI estimates the parameters of $q(\boldsymbol{z})$ solely with stochastic optimization. Therefore the number of parameters to be estimated by stochastic approximation scales quadratically with T in ADVI due to the Cholesky factor of the covariance matrix in $q(\boldsymbol{z})$. We use reverse-mode [116] automatic differentiation background in Turing to speed up the inference. ADVI converges in 6000 iterations with a default optimizer set by Turing. The run time and the free energy comparisons are given in Table 5.1. We see that equipped with CVI, ForneyLab attains a slightly lower free energy in a shorter time compared to Turing's ADVI. This experiment demonstrates the efficiency of our CVI implementation in ForneyLab.

Bayesian Parameter and State Estimation with Structured Variational Message Passing

In the previous experiment, we worked with fixed noise parameters in transition and measurement components. Let us relax this assumption and estimate these parameters as well. The model specification in (5.39) slightly changes as

$$\begin{aligned} p(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{z}, W, S) = p(W)p(S)p(z_1)p(x_1|z_1)p(y_1|x_1, S) \\ \prod_{t=2}^{T} p(z_t|z_{t-1}, W)p(x_t|z_t)p(y_t|x_t, S)\,, \end{aligned} \tag{5.40}$$

where

$$
\begin{aligned}
p(z_1) &= \mathcal{N}(z_1; \mathbf{0}_4, \boldsymbol{I}_{4\times 4}) \\
p(W) &= \mathcal{W}_4(W; \boldsymbol{I}_{4\times 4}, 4) \\
p(S) &= \mathcal{W}_3(S; \boldsymbol{I}_{3\times 3}, 3) \\
p(z_t|z_{t-1}, W) &= \mathcal{N}(z_t; Az_{t-1}, W^{-1}) \\
p(x_t|z_t) &= \delta(x_t - g(z_t)) \text{ with } g(z_t) = [||\xi_1 - Bz_t||, ||\xi_2 - Bz_t||, ||\xi_3 - Bz_t||]^\intercal \\
p(y_t|x_t, S) &= \mathcal{N}(y_t; x_t, S^{-1}).
\end{aligned}
$$

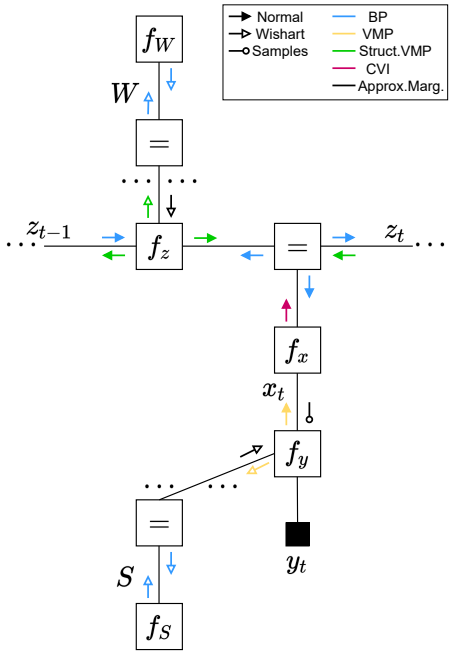


Figure 5.6: A sub-graph of the model defined in (5.40). The factors f_W, f_S, f_z, f_x, f_y stand for $p(W)$, $p(S)$, $p(z_t|z_{t-1})$, $p(x_t|z_t)$, $p(y_t|x_t)$ respectively. In order to provide the reader with the intuition of the message passing procedures in *ForneyLab*, we visualize the messages as well. The arrow shapes indicate the probability distribution families that messages are carrying. The messages are colored according to the algorithm types that generate them with an additional black color indicating approximate marginal distributions to be employed in VMP. *ForneyLab* resorts to stochastic optimization only around the factor f_x and carries out the rest of the computations by deterministic distributed message passing operations. Notice that CVI sends a Gaussian message, visualized with the red arrow, towards the equality node that is incorporated in the calculation of the forward and backward messages with BP. We also allow VMP to be executed around the node f_y by setting $q(x_t)$ to a set of samples over which the expectation quantities are estimated for VMP.

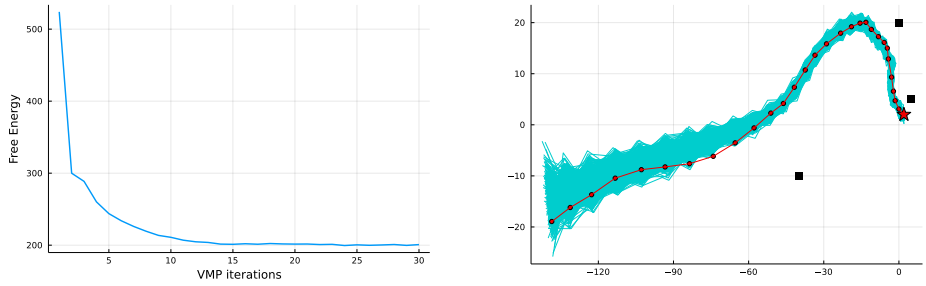
Above $\mathcal{W}_d(V, n)$ is a Wishart distribution with $d \times d$ positive definite matrix V and n degrees of freedom. We generate a synthetic data with $T = 30$ and approximate the exact posterior $p(\boldsymbol{z}, W, S|\boldsymbol{y})$ with a structured mean-field assumption: $q(\boldsymbol{z}, W, S) = q(\boldsymbol{z})q(W)q(S)$, where $q(\boldsymbol{z})$ is not factorized over $\boldsymbol{z}$. The factor graph and the message passing procedure for one time slice is depicted in Figure 5.6.

Notice that we resort to NGD stochastic approximation only around the factor $f_x(x_t, z_t) = p(x_t|z_t)$ to compute a message visualized with a red arrow and parameterized with a Normal distribution. At a time step t, CVI computes the approximate message in 1000 NGD iterations with a step size of 0.1. The rest of the computations are carried out with deterministic distributed operations. We run 30 VMP iterations, which minimizes the free energy as in Figure 5.7. For qualitative analysis, we also visualize the final position estimations with 1000 samples drawn from $q(\boldsymbol{z})$.

This experiment provides the reader with the intuition behind our CVI implementation in ForneyLab. CVI around a deterministic node renders NGD locally to circumvent intractable operations due to non-conjugacies and approximates the problematic messages with approximate messages amenable to analytical calculations.

5.4.5 Regression with a Bayesian Neural Network

Many PPLs support integration with deep learning libraries to allow complex probabilistic model specifications with neural networks, e.g., Pyro [99] and TensorFlow Probability [98] respectively interface with PyTorch [45] and Tensorflow [44]; Tur-



(a) Free energy over VMP iterations

(b) Position estimations

Figure 5.7: On the left is the free energy over VMP iterations for the model defined in (5.40). On the right is the 2-d environment the object moves in. The red curve shows the true trajectory of the movement with the star being the initial point. Black squares are the sensors measuring the Euclidean distances with some perturbations. Once the inference is complete, we draw 1000 samples from $q(\boldsymbol{z})$ and visualize the corresponding position estimates with cyan curves.

ing [100] supports model specifications with Flux.jl [46]. Inspired by Turing, here we show how to use ForneyLab's CVI to make inference in a Bayesian Neural Network (BNN) built [159] with the Flux deep learning package.

For this experiment, we generated 40 data samples with input x_n and output y_n values from a sinusoidal function. We run inference for the following model specification:

$$p(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{s}, w) = p(w) \prod_{n=1}^{40} \underbrace{\delta(s_n - g(w, x_n))}_{\text{Deterministic node}} p(y_n|s_n), \text{ where} \tag{5.41}$$
$$p(w) = \mathcal{N}(w; \mathbf{0}_{22}, \boldsymbol{I}_{22\times 22})$$
$$p(y_n|s_n) = \mathcal{N}(y_n; s_n, 0.1).$$

In the above model specification, $g(w, x_n)$ is a three-layered neural network comprised of 22 weights with a prior $p(w)$. Exogenous inputs to the neural network are x_n values. We run CVI to approximate $p(w|\boldsymbol{y}, \boldsymbol{x})$ with a Gaussian $q(w)$. We use a descent optimizer with step size 0.01 and run 10000 iterations over the entire data set.

After the inference is completed, we generate 100 neural networks parameterized with weights sampled from $q(w)$ and run each neural network with x_n in the range $(-5, 5)$. The results are visualized in Figure 5.8. We see that the neural network captures the sinusoidal shape confidently for the interpolation task in the range $(-2, 2)$. Outside of this range, we obtain flattened extrapolation with higher

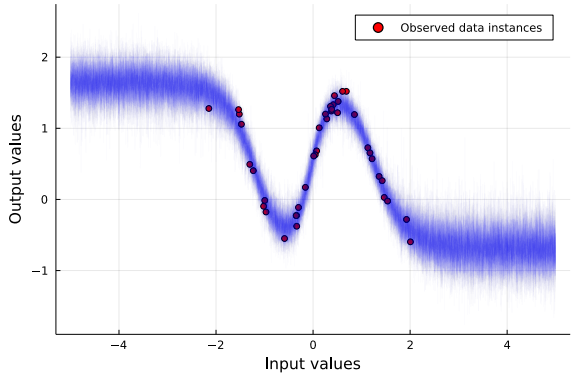


Figure 5.8: A simple example proves that ForneyLab is compatible with Julia language's deep learning package Flux. We use Flux to build a neural network architecture and insert it into an FFG through a deterministic node. In this plot, observations are red points. The blue curves are the outputs of neural networks parameterized with weights sampled from $q(w)$ after the inference with CVI.

uncertainty. This simple experiment demonstrates how ForneyLab with CVI seamlessly interfaces with a deep learning package.

5.5 Discussion and Implementation Details

SVI and CVI greatly extend the inference capabilities of our message passing-based PPL, equipping it with some favorable features over the existing non-message passing-based PPLs. For instance, many of the non-message passing-based PPLs achieve scalable variational inference by adhering to doubly stochastic variational inference [87], in which the stochasticity is due to both the mini-batch selection process from the data set and sampling from the candidate approximate posterior distribution. This process is the same in conjugate and non-conjugate models for non-message passing-based PPLs. In contrast, running SVI in a message passing-based PPL for conjugate model specifications obviates the need for sampling from the candidate approximate posterior distribution and reduces the source of stochasticity to the mini-batch selection process only. Reducing the dependency on sampling processes often results in faster and more stable convergence. Furthermore, as opposed to non-message passing-based PPLs, SVI enables message passing-based PPLs to employ natural gradients that are available in closed form for conjugate factor pairs. Similar to SVI, CVI is also an efficient inference procedure that involves analytical calculations in gradient estimations. We show that message passing frameworks provide convenient tools to take full advantage of the CVI algorithm: pre-defined message passing rules carry out the analytical calculations in CVI and reduce the number of variables that are to be sampled. For example, the term η_{jb}, appears in the natural gradient (5.5), relates to the natural parameters of the message $m_{jb}(z_j)$ from a factor $\mathcal{V}(j) \setminus b$ and allows to calculate the contribution of $\mathcal{V}(j) \setminus b$ to the natural gradient without sampling. Similarly, the estimation of $\log m_{bj}(z_j) \propto \mathbb{E}_{q(\boldsymbol{z}_{b \setminus j})}[\log f_b(\boldsymbol{z}_b)]$ in (5.16) involves some analytical calculations that are automatically addressed by the pre-defined message $m_{di}(z_i)$.

Non-conjugate models make use of natural gradient terms $\nabla^N_{\eta_j} \mathbb{E}_{q(z_j;\eta_j)}[\log m_{bj}(z_j)]$ that are not available in closed form. The original CVI article [84] proposes an efficient estimation approach that does not necessitate the explicit evaluation of the Fisher information matrix for Gaussian approximate distributions and recommends the reparameterization trick [85] for the other distributions. We adhere to their efficient approach for the Gaussian case, but use the REINFORCE estimator (5.12) for the other distributions, instead. This is mainly because REINFORCE is an easy-to-implement, global estimator for the free energy gradient. However, it is often considered a high variance estimator requiring additional variance reduction techniques to be used in practice. [51] shows that Rao-Blackwellization [138] and control variates [139] considerably reduce the variance in free energy gradient estimations. Fortunately, message passing frameworks inherently support Rao-

Blackwellization and closed-form solutions. Nevertheless, it is still valuable to get the gradient estimations over reparameterization of random variables to further reduce the variance in estimations. The reparameterization trick is generalized beyond Gaussian distributions in recent works by [149, 160, 161]. The most recently introduced approach is the implicit reparameterization trick [161], which reparameterizes variables using Cumulative Distribution Functions (CDFs). We plan to implement this feature in a future release of ForneyLab.

Additional to SVI and CVI, we demonstrate how to interface BBVI and reparameterization based stochastic optimization approaches with the message passing procedures on factor graphs. The resulting approach is efficient as message passing algorithms automatically realize Rao-Blackwellization, enabling us to go beyond the exponential family of distributions. Since most of the distributions in ForneyLab are members of the exponential family, efficient BBVI, and reparameterization-based stochastic optimization approaches have not yet been implemented in ForneyLab. Nonetheless, they can be implemented when needed.

Stochastic optimization methods, in general, require hyperparameters such as step size to be set carefully for fast and stable convergence. The current ForneyLab implementation allows step sizes to be set by optimizer objects defined in Julia language's deep learning package Flux [46]. We also implemented the optimizer proposed in [83, Equation 26] satisfying Robins-Monro conditions. Additionally, we provide an implementation for [162], which adjusts step sizes adaptively using already calculated natural gradients. Another hyperparameter required in CVI is the number of iterations per message approximation. To free our PPL's end-user from specifying the number of iterations, we provide her with two options that automatically determine when to stop doing iterations: one based on tracking the relative change of the variational objective; the other based on viewing the optimization algorithm as producing a Markov chain and using Markov Chain Monte Carlo diagnostic tools to determine the stopping criterion [163]. The former method runs faster but can prematurely end the optimization algorithm in some cases, whereas the latter method is more robust but has a significantly higher computational load as it runs several optimization chains in parallel for each iteration.

We present the stochastic approximation for variational inference as if it is an unconstrained optimization task. However, the domain of the probability distribution functions are often constrained, e.g., shape and rate parameters of a Gamma distribution are constrained to be in the positive real axis. Unfortunately, NGD steps given in (5.1) are susceptible to violations of constraints. In our ForneyLab implementation, we avoid domain violations in Gaussian and Gamma distributions by discarding the samples that causes violations. In the future, we plan to integrate the recent researches along this line [164] to our message passing-based PPL.

5.6 Conclusions

This chapter demonstrates how to cast stochastic optimization methods for variational inference as distributed, local operations on FFGs for probabilistic programming. Choosing NGD as the optimizer of the free energy, the resulting method automates the well-recognized SVI and CVI algorithms in a message passing-based PPL. In SVI, the natural gradients of the free energy objective are analytically acquired from pre-defined messages in the message passing-based PPL. In CVI, the natural gradients are partially available in the messages in closed form, and the components that are not amenable to closed-form calculation can be locally estimated by automatic differentiation tools and Monte Carlo summation. Both SVI and CVI operate at node level and seamlessly interface with the message passing procedures. The efficiency of SVI and CVI within a message passing-based PPL has been validated by a number of experiments.

Chapter 6

Discussion and Conclusions

A part of this chapter is published in the original work referenced below.

- Semih Akbayrak, İsmail Şenöz, Alp Sarı and Bert de Vries, *Probabilistic Programming with Stochastic Variational Message Passing*, International Journal of Approximate Reasoning, 2022

6.1 Contributions

This dissertation enriched deterministic, distributed inference procedures known as message passing algorithms with the generality and the flexibility of Monte Carlo methods, Laplace approximation, and automatic differentiation techniques. The resulting inference procedures presented in this dissertation exploit the conjugacy and conditional conjugacy structures in the model specification while resorting to computationally more expensive numerical approximation methods in non-conjugate sub-graphs. We also showed how to employ approximation methods to compensate for the lack of message passing rules in probabilistic programming packages aiming to automate Bayesian inference with re-usable inference components.

In Chapter 2, we presented Forney-style Factor Graphs (FFGs) as a probabilistic modeling framework that allows us to build complex models by combining nodes and sub-graphs through edges. We showed that the modular, plug-in type structure of FFGs is suitable for the automation of inference procedures. We reviewed the message passing interpretations of exact and approximate Bayesian inference procedures. An important observation in this chapter is that the global free energy objective we work with throughout the dissertation can also be partitioned over the factor graph, allowing us to define inference tasks locally over sub-graphs.

In Chapter 3, we focused on the Variational Message Passing (VMP) algorithm on FFGs by emphasizing that VMP messages on factor nodes have fixed functional forms. Arguments to VMP messages are expectation quantities of variables associ-

ated with the edges connected to the factors. Then, the VMP message calculation procedure reduces to the calculation of the expectations. We estimated these expectations with importance sampling, which resulted in an intuitive message passing interpretation. FFGs, in conjunction with message passing algorithms, partition high dimensional manifolds into smaller dimensions that allow us to use importance sampling efficiently. However, to deal with those cases where local manifolds are still high dimensional, we automate the Laplace approximation in FFGs. Laplace approximation preserves the Gaussian distribution assumption of the forward messages in the posteriors, which allows us to exploit the closed-form message passing rules abound for the Gaussian case in later stages of the inference procedures. We called the resulting method Extended Variational Message Passing (EVMP).

In Chapter 4, we improved the importance sampling estimation stage of the EVMP algorithm with an adaptive procedure. Unlike EVMP, which represents approximate posterior marginals with weighted samples after the importance sampling estimation stage, this time, we sticked to a moment matching approach to approximate backward messages with standard probability distributions. Backward messages that are approximated with standard messages can interface with Belief Propagation and Expectation Propagation in addition to VMP; hence we called our approach Adaptive Importance Sampling Message Passing (AIS-MP).

In Chapter 5, we showed how easy it is to interface the stochastic approximation methods for variational inference with the message passing procedures. Therefore, we were able to adopt the universality and scalability features of stochastic variational inference methods in our message passing framework.

6.2 A Comparison of the Algorithms

Throughout the thesis, we presented hybrid inference algorithms for probabilistic programming in FFGs. Now, let us discuss when to prefer a specific inference algorithm to the others. It is not easy to give a definitive answer to this question. Nevertheless, in Table 6.1, we compare the algorithms in terms of some key features that might form the basis of algorithm selection criteria.

We investigate the IS and Laplace modes of the EVMP algorithm separately. SVMP represents the algorithms presented in Chapter 5, namely SVI, CVI, Reparameterization Gradient Message Passing, and efficient BBVI.

In our algorithm designs, we strive to relieve end users from hyperparameter[1] selection. Hyperparameters in approximate inference algorithms might include the number of samples, step sizes in gradient-based optimization schemes and stopping criteria depending on the type of the approximation. EVMP and AIS-MP have more or less stable hyperparameter-free inference schemes even though some of

[1]We consider hyperparameters required to execute individual inference steps in message passing algorithms such as posterior marginal updates and message calculations.

Algo / Feature	EVMP (IS)	EVMP (Lap.)	AIS-MP	SVMP
Hyperparameter-free	Yes	Yes	Yes	No
Optimization-free	Yes	No	No	No
Deterministic	No	Yes	No	No
Scalable	No	No	No	Yes
Dimensions	Low	Low, High	Low	Low, High

Table 6.1: A comparison table of EVMP, AIS-MP and SVMP. The rows from 1 to 4 show whether the listed algorithms possess the feature. The last row shows what dimensions these algorithms are designed for. The assessments about the hyperparameters are whether the algorithms need hyperparameters to execute individual inference steps in message passing such as message calculations and posterior updates.

our choices are ad-hoc, e.g., the number of samples is set to 1000. In SVMP, we put some effort into designing hyperparameter-free inference schemes, but they are not mature yet. Let us illustrate the importance of hyperparameter selection by a toy example.

Consider an edge of an FFG with two messages on it. Suppose that the forward message is a wide Gaussian distribution while the backward is multimodal distributed. We visualize these messages in Figure 6.1 together with a scaled posterior proportional to the product of forward and backward messages. Running CVI with a step size of 0.05 on this illustrative example, we see that the variational distribution oscillates between the two modes over the course of the optimization (See Figure 6.2a). One may attribute this behaviour to the fixed step size 0.05 and advise sticking to the Robins Monro conditions in step size selection. However, Figure 6.2b shows that step sizes comply with the Robins-Monro conditions may also lead to optimization routines diverging from stationary points. In Figure 6.2b, the variational distribution converges to a stationary point at first but diverges later on as the other mode continues to attract the variational distribution through noisy gradients.

In the example Figure 6.1, the EVMP algorithm of ForneyLab automatically executes a Laplace approximation procedure. EVMP initializes a gradient-based optimization with the mean of the forward Gaussian message and runs the optimization by using L-BFGS [165, 166] optimizer in the Julia-based optimization package Optim.jl [103]. Unlike SVMP, the gradients in the Laplace approximation are deterministic, and hence determining the learning rate with a line search and setting stopping criteria is easier. Note that the mean of the forward message is closer to the posterior mode on the left-hand side. Since the automated Laplace procedure of EVMP initializes with the mean of the forward message, the optimizer successfully

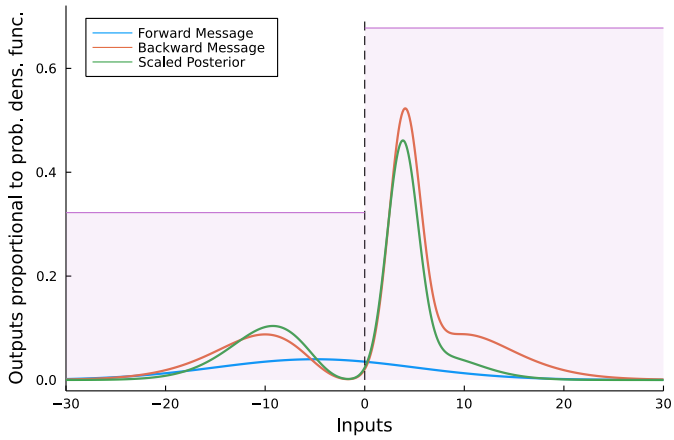
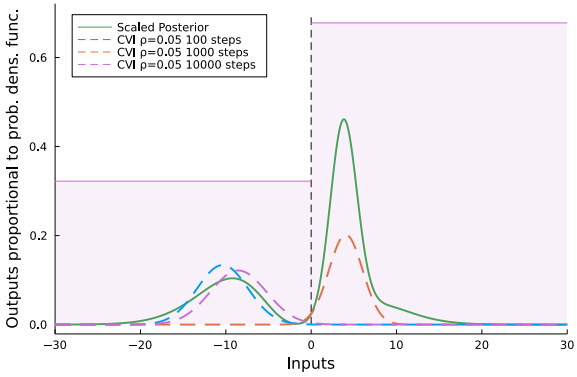


Figure 6.1: An illustrative example with a wide Gaussian forward message and a multimodal backward message. The unnormalized posterior is scaled for visualization purpose. The lilac bars on the left and the right of 0 respectively show $F(0)$ and $F(\infty) - F(0)$, where F denotes the Cumulative Distribution Function (CDF).

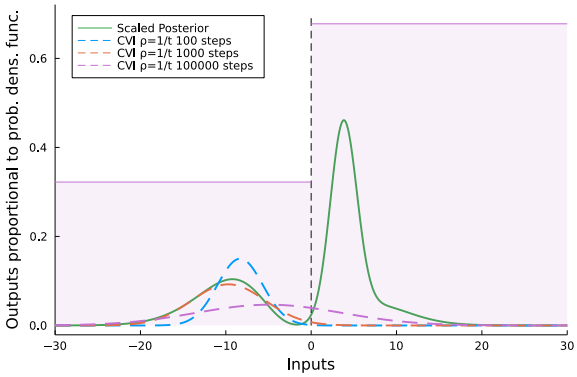
converges to the mode on the left-hand side (See Figure 6.3a).

The EVMP algorithm automatically switches to the Laplace approximation mode if exactly one of the incoming messages to an edge is a Gaussian distribution. However, this choice is open to criticism. Consider Figure 6.3. As a result of the mode-seeking behavior of the Laplace approximation, the EVMP algorithm could capture only one of the modes, which is the left one in this example. However, the mode on the right contains the bulk of the probability mass. The automated importance sampling (IS) scheme of the EVMP algorithm, on the contrary, can capture both modes successfully, as shown with an orange-colored LWS in Figure 6.3b. Therefore, the end-user must be in charge of approximation type selection. At this point, we can also emphasize that the automated moment matching of AIS-MP should be optional since it leads to wide, unimodal approximate distributions that might not be desired in certain applications.

Often, the set of hidden variables $\mathbf{z}$ in a probabilistic model lies on a high dimensional manifold. By exploiting the factorizations in the probabilistic model, message passing algorithms partition the original high-dimensional inference problem into a set of simpler problems defined in lower-dimensional manifolds. This is why EVMP and AIS-MP are good candidates to try first (IS works well in low-dimensional manifolds). If the local inference needs to be performed in high dimensions, then EVMP and SVMP algorithms can be preferred depending on the type of inference problem. The Laplace approximation in EVMP employs exact gradients in a quasi-Newton optimization scheme. This exact optimization procedure ease automation



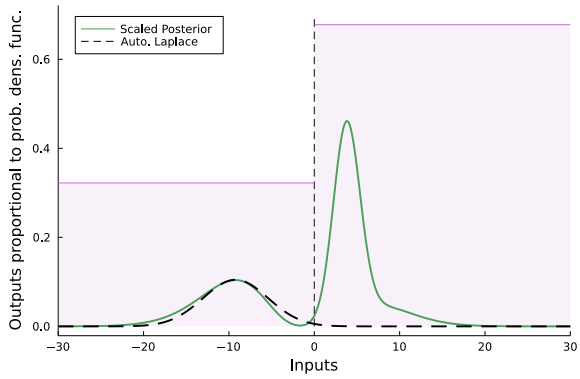
(a)



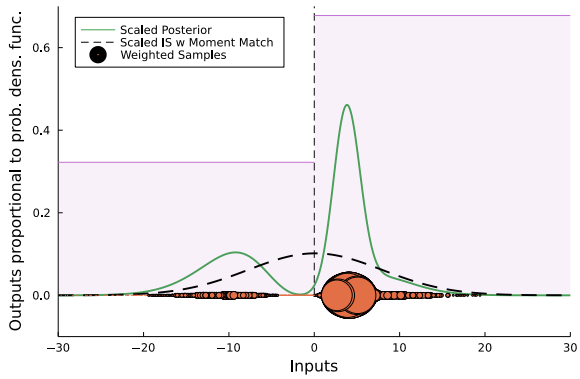
(b)

Figure 6.2: Top: CVI run with a step size 0.05 for varying number of iterations. Bottom: CVI run with a step size of $1/t$, where t is the iteration number.

of the hyperparameter selection process. Therefore, the EVMP algorithm provides a fully automated Laplace approximation procedure. However, our choice to let the algorithms free from hyperparameters may cause other problems. For example, consider the Bayesian Neural Network given in Section 5.4.5. The EVMP algorithm returns a zero mean Gaussian posterior distribution on neural network weights in this model. The reason is that the automated optimization of the Laplace approximation in EVMP initializes with the mean of the prior, which is a vector of zeros. Setting all the weights to zero in a neural network creates symmetry and results in 0 gradient. So, the vector of zeros is an unsatisfactory stationary point in neural networks that our automated Laplace approximation gets stuck. In contrast, CVI in Section 5.4.5 breaks the symmetry in weight initialization by sampling from the



(a)



(b)

Figure 6.3: Top: The automated Laplace procedure of EVMP is executed. Bottom: Importance sampling is used to find a List of Weighted Samples (LWS) approximation. We also show a Gaussian distribution approximation with the moment matching calculated over the LWS representation.

posterior, which initially equals to the prior distribution.

In EVMP, we can break the symmetry in the weights by randomly initializing the weight priors by drawing the mean parameter entities from $\mathcal{N}(0, 0.1)$. Having resorted to that approach, the EVMP estimations we get are visualized in Figure 6.4.

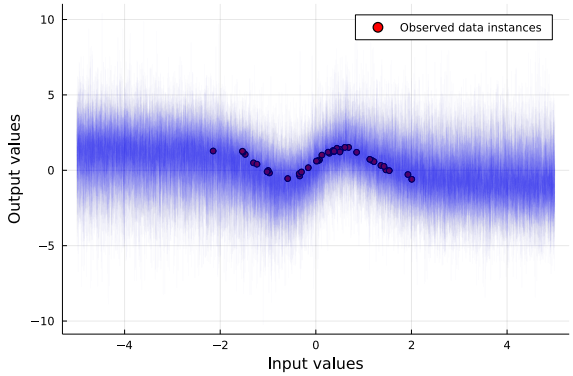


Figure 6.4: Performance of the EVMP algorithm on the BNN model specification in Section 5.4.5 is visualized. Although EVMP captures the sinusoidal shape in the dataset, variance of the predictive distribution is extremely large.

The EVMP algorithm captures the sinusoidal shape in the dataset. However, the covariance parameter is overestimated. Recall that the Laplace approximation is a stationary point-seeking approach. Once a stationary point is reached, the Laplace approximation evaluates the Hessian at this point and sets the covariance matrix of the approximate Gaussian accordingly. However, as evidence from this example shows, the crude approximation of the automated Laplace method might result in undesired covariance estimations even if the found mode is satisfactory. CVI in Section 5.4.5 does not suffer from such a problem as it minimizes the free energy with respect to the natural parameters of the variational distribution that implicitly includes the covariance matrix term.

In our Laplace and CVI implementations, we automatically estimate the full precision matrix, which might not be feasible in very high dimensions as the number of parameters to be estimated in the precision matrix scales quadratically with the dimension. As a remedy, we can impose an additional factorization assumption and adjust the algorithms so that only the diagonal entries of the precision are estimated. We leave it as future work.

(3.14) shows how backward messages arise as Non-standard Exponential Family (NEF) distributions in the EVMP algorithm. A similar approximation is performed in (5.17) just before calculating the message approximation with CVI. There are two major differences between these two approaches. The first difference is that

in the latter case, following the Monte Carlo estimation of the message, CVI approximates the message within the Exponential Family (EF) of distributions. These messages then can be used in BP in later stages of the inference if the terminal node that this approximate message propagates to possesses the required BP message passing rules. In the EVMP case, however, the NEF message should eventually collide with an EF or LWS (a List of Weighted Samples) message coming from the other end of the edge to generate LWS marginal, over which expectations in VMP messages can be estimated. Another major difference between (3.14) and (5.17) is the sampling procedures. Note that in (3.14), we draw samples from the messages whereas in (5.17), samples are drawn from approximate marginals. This is because in the latter case, we assume full factorization in the approximating distribution, while in the former, no factorization is put on the approximate joint distribution of the input variables. Of course, it is desired to retain the dependency structures in approximating distributions. However, preserving the dependency structures in approximating distributions requires us to calculate the entropy of the approximate joint distribution in free energy calculations, which is a challenging task except for some special cases, e.g., joint Gaussian distributions. To avoid trouble with calculating the entropy of the approximate joint distribution, in (5.17) we automatically assume mean field factorization.

In IS mode of EVMP, we estimate not only the marginal distributions but also the entropy terms for free energy calculation. In AIS-MP, we introduce an additional moment matching step to be able to represent marginals with standard EF distributions and consequently to calculate entropy terms in closed form. The entropy terms included in free energy calculation are available in closed form in CVI, too.

Message passing algorithms are fast and efficient, especially in Gaussian state space model variants, as the computational operations in the bulk of Gaussian state space models can be partially or entirely performed in closed form. To reflect the true generative processes better in the model specifications, often nonlinear functions are incorporated into Gaussian state space models. Standard algorithms that operate on Gaussian state space models with nonlinearities such as Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) (or Sigma-point Kalman Filter [77] in general) are discussed in detail in [19]. These algorithms can be expressed as local operations in the context of FFGs; see, for example, [167].

EKF linearizes nonlinear functions around the forward message's mean, allowing us to use standard Kalman filtering (smoothing) equations to run inference in Gaussian state space models with nonlinearities. However, this local linearization is not always a solution to our inference problems in state space model variants. For example, consider the HGF model described in (3.17). In the HGF model specification, the VMP message propagating from the node $\mathcal{N}(x_t; x_{t-1}, w_t)$ to the deterministic node $\delta(w_t - \exp(z_t))$ is Gamma distributed. If we linearize this deterministic node, then the message propagating to the equality node z_t is connected to turns out to be another Gamma distribution due to the scaling property of Gamma dis-

tribution. Multiplying this message with a Gaussian message propagating on the upper layer, where the variables $\boldsymbol{z}$ are located in, results in an intractable integral in the normalization. Similarly, consider the message propagating from the equality node to z_t is connected to the deterministic node $\delta(w_t - \exp(z_t))$. This message is a Gaussian message. Using EKF, we find the message propagating to the node $\mathcal{N}(x_t; x_{t-1}, w_t)$ on the edge w_t is defined in Gaussian distribution family, which again needs to be multiplied with a Gamma message coming from $\mathcal{N}(x_t; x_{t-1}, w_t)$ and leads to intractable normalization.

Messages in LWS forms are represented in Figure C.1. In our EVMP algorithm, the samples constituting the message propagating from a deterministic node in the forward direction are transformed samples that are randomly drawn from the incoming message to the deterministic node. Sigma-point methods, in general, follow a similar approach. Unlike EVMP, in sigma-point methods, weights and samples (sigma points) are deterministically selected in such a way that the pseudo-mean parameters calculated over selected weight and samples match with the actual mean parameters of the message propagating to the deterministic node [168]. Once the weights and samples are chosen, sigma-point methods approximate the message propagating from the deterministic node using moment matching performed over transformed samples and their corresponding weights. Sigma-point methods can be preferable to Monte Carlo methods, especially in low dimensions. However, generalizing the deterministic sigma point selection process beyond Gaussian distribution is not straightforward [169]. Moreover, the deterministic numerical integration approaches that sigma-point methods utilize, such as Gauss-Hermite cubature, might not scale well to high dimensions [19].

Lastly, let us discuss the scope of the presented algorithms in terms of probability distribution families. EVMP can easily be extended beyond EF distributions as long as VMP messages are defined on soft factors from non-EF distributions. AIS-MP, in contrast, is developed specifically for EF distributions. Regarding the Stochastic VMP algorithms, SVI and CVI are also applicable for EF distributions. Nonetheless, we can go beyond EF distributions by forgoing the natural gradient descent procedure and sticking to raw stochastic gradient descent, as discussed in Section 5.3. In the letter case, we do not approximate backward messages with standard probability distributions; hence the outgoing messages from the sub-graph SVMP resorted to need to be calculated by VMP.

6.3 Strengths and Limitations

Below, we make the concluding remarks, mention the strengths and limitations of the methods presented in this dissertation, and discuss the research questions we posed in Section 1.2.

RQ1. How can we automatically execute Bayesian inference in an efficient manner?

In Chapter 2, we reviewed FFGs and the standard message passing algorithms since the main contributions of this dissertation that are presented in the later chapters are formulated in an FFG framework, and the efficiency of the algorithms presented in later chapters is mainly due to the deterministic message passing algorithms.

Inference tasks in probabilistic models such as exact posterior marginal calculation, posterior approximation with a factorized variational distribution and prediction are all integration operations. A natural way to perform inference in probabilistic models is to group the random variables together in integrals by discarding the irrelevant terms for the given inference step. Message passing algorithms treat the integral components that appear in inference procedures as building blocks of a unified inference engine. The messages performing marginalizations are called Belief Propagation (BP) messages. BP is effective and automatable on FFGs, but not broadly applicable. If BP is hampered because the rule is not defined for the incoming message type, then we can approximate the problematic message within the desired distribution type using Expectation Propagation (EP). Hence EP preserves the effectiveness of BP everywhere, but in an approximated inference component that is effective in its own way, under a moment matching constrain [69]. If, instead, we introduce a factorized variational distribution to approximate the joint distribution of random variables around a factor node as in Section 2.3, we get a surrogate objective that can be minimized with the coordinate descent optimization by means of message passing, named Variational Message Passing (VMP). Therefore, message passing is an automatable and effective procedure for Bayesian inference in probabilistic models, and FFGs are formal frameworks for message passing inference.

Inference in FFGs by means of message passing is a series of local operations defined in sub-graphs. This locality feature makes hybrid inference straightforward. For example, consider Section 2.3 once again. We derive the VMP message $m_{bj}(z_j)$ using the message $m_{jb}(z_j)$ regardless of the algorithm that generates $m_{jb}(z_j)$. The message $m_{jb}(z_j)$ can be a BP, EP, or another VMP message, and yet we computed $m_{jb}(z_j)$ using the VMP algorithm.

The limitations of a message passing inference engine built upon the BP, VMP, and EP algorithms are threefold and discussed in Section 2.5. This manuscript addressed them by seeking answers to the following research questions.

RQ2. How can we automatically combine the generality of Monte Carlo sampling and flexibility of Laplace approximation with the efficiency of message passing algorithms?

In Chapter 3, we allowed the end-user of our message passing-based PPL to insert customized functions into factor graphs through deterministic factor nodes without putting any restrictions on the type of deterministic mappings. The deterministic mappings can be simple expressions such as $\exp()$ or complex functions involving conditional statements and loops. We thoroughly examined message passing around deterministic factor nodes to address the lack of generality in message passing-based PPLs. Connected to soft factor nodes from the exponential family of distributions at the output interface, we showed how the backward messages on deterministic nodes arise as members of the exponential family of distributions, although their sufficient statistics are not recognized in standard distributions. For the general case, we used importance sampling to approximate marginals and represented them with a pseudo-distribution List of Weighted Samples (LWS). As for the proposal distribution in importance sampling, we used forward messages from soft factors that always carry standard probability distributions and are easy to sample. We showed that forward messages from deterministic nodes also carry LWS and are calculated intuitively. These LWS forward messages can be used to automate Bootstrap Particle Filter alongside approximating the marginals. Given that marginals are represented with LWS pseudo-distributions, we can estimate expectations over them and approximate VMP messages. We also provided a Monte Carlo approach to estimate entropy terms that allowed us to keep track of convergence with the free energy. We examined Gaussian as a special case to deal with inference in large dimensions and preserve the computational convenience of Gaussian distributions, especially in state space model variants. We automated Laplace approximation using automatic differentiation and optimization packages of Julia language. We called this entire procedure Extended Variational Message Passing (EVMP) and implemented it in a message passing-based PPL, ForneyLab.

The EVMP algorithm greatly extends the capabilities of message passing-based inference engines. However, it possesses several shortcomings. First and foremost, approximations in EVMP are driven by the model itself. If the model is not a good representative of the true real-world process, we may get imprecise estimations. Secondly, importance sampling is not efficient in high dimensions. To deal with high dimensions defined in the domain of real numbers, we automated Laplace approximation. However, the Laplace approximation is only a mode-seeking algorithm that does not perform optimization with respect to the covariance matrix. Thirdly, to calculate the free energy, we used Monte Carlo sampling for the estimation of the entropy terms related to posteriors represented with LWS. Lastly, we evaluated the functional form of the messages in EVMP but did not define or approximate them as standard probability distributions, which precludes the automatic execution of BP at the other end of the edge.

RQ3. Having developed a hybrid Monte Carlo - message passing inference procedure, is it possible to improve the accuracy of Monte Carlo estimates within message passing in an automated way?

In Chapter 4, we tried to alleviate some of the shortcomings of the EVMP algorithm mentioned above. We exchanged the forward message proposal distributions in the EVMP algorithm with adaptive proposal distributions. The algorithm is called Adaptive Importance Sampling Message Passing (AIS-MP).

EVMP is appealing for probabilistic programming as it is a hyperparameter-free algorithm to a large extent. In our adaptive importance sampling scheme, to free the end-user from the hyperparameter selection process, we chose step sizes with the ADAM optimizer and tracked the convergence with the number of efficient samples. We initiated the proposal distributions with the forward messages and hence set EVMP as the fallback method of AIS-MP.

After finding LWS representations, we used an additional moment matching step to approximate both marginals and backward messages with standard probability distributions. The moment matching step allowed us to calculate entropy terms in closed form. In addition, we enabled backward messages to interface with the BP and EP algorithms at the other end of the edge. As it is hard to derive inverse link functions for certain probability distributions such as Gamma distribution [93], we performed moment matching over central moments that is only an approximation to the desired moment matching procedure.

The main shortcoming of AIS-MP is its slowness. AIS-MP is slower than EVMP because AIS-MP involves an additional stochastic optimization step. We used an ADAM optimizer with a small initial step size to avoid diverging behaviours. We believe better optimizers can be tailored for the AIS-MP algorithm in the future. Nevertheless, the current implementation of the AIS-MP algorithm is particularly preferable in those models comprised mainly of conjugate factor pairs with few non-conjugacies and non-linearities. Since AIS-MP is based on importance sampling as the EVMP algorithm, it is efficient when the high dimensional manifold the variables of the model lie on is well partitioned with the FFG and the message passing algorithms.

RQ4. How can the free energy gradient estimates help us for scalable and broadly applicable variational inference?

In Chapter 5, we focused our attention on the stochastic optimization of free energy. Stochastic approximation methods for variational inference, in general, write down the free energy objective as a function of the parameters of a recognition distribution. After that, they estimate the gradient of the free energy with respect to variational parameters by evaluating the samples (or their transformations parame-

terized with the variational parameters) drawn from the recognition distribution in the free energy. The free energy gradient estimates are then used to tune the variational parameters. Stochastic approximation methods for variational inference are scalable to large datasets and high dimensions. Therefore, they would be of great importance to our message passing-based PPL. The most popular variational inference algorithms for probabilistic programming are Automatic Differentiation Variational Inference (ADVI) and Black Box Variational Inference (BBVI). We showed an efficient realization for the BBVI algorithm and the stochastic variational inference with reparameterization of random variables in Section 5.3. These methods can be applied both for the exponential family of distributions and the other probability distribution families. In our message passing-based PPL, stochastic approximation methods are locally resorted to only when the implemented message passing rules are insufficient to execute the inference. Specific to the exponential family of distributions, very efficient stochastic approximation approaches have been proposed in the literature under the name of Stochastic Variational Inference (SVI) and Conjugate-computation Variational Inference (CVI). We showed that, similarly to BBVI and ADVI, both SVI and CVI can be formulated as probabilistic programming algorithms in the message passing framework on FFGs. The main drawback of these approaches is determining the hyperparameters such as step sizes and the number of iterations in stochastic optimization procedures. As a future research direction, recent attempts [170,171] to adapt the line search mechanism to stochastic gradient descent can be implemented in our PPL.

In this thesis, we are in pursuit of a PPL framework that combines the advantages of both sampling-based and message passing-based inference without taking over the downsides of both frameworks.

The above research questions pave the way toward a PPL framework that combines the bests of sampling-based and message passing-based inference methods that are summarized in Table 1.1. No single method satisfies all the criteria in Table 1.1. In our message passing PPL, we provide the end-user with several options as in other PPLs, where the end-user is allowed to choose the inference algorithm among, say, Hamiltonian Monte Carlo (HMC) and ADVI. A superficial comparison of algorithms is given in Table 6.2.

In Section 6.2, we stated that if the search space of variables is partitioned into small dimensional manifolds, the end-user is advised to try EVMP first. In case the EVMP estimations are not satisfactory, the user can get more satisfactory estimations by AIS-MP in exchange for speed. A faster but more hyperparameter-dependent framework, SVMP, is based on stochastic optimization approaches for variational inference. We implemented several methods to remove the hyperparameter selection process in SVMP, but they are not fully optimal yet. If the user works with a

Criteria \ Method	EVMP	AIS-MP	SVMP
Precision	✗	✓	✓
Performance metric	✓	✓	✓
Speed	✓	✗	✓
Universal	✓	✓	✓
Scalable	✗	✗	✓
Full automation	✓	✓	✗

Table 6.2: A superficial comparison table of EVMP, AIS-MP and SVMP.

gigantic or a streaming dataset, she is advised to choose SVMP.

We leave the full automation criteria grey colored. We mentioned in Chapter 1 that there are standard approaches in iterative message passing algortihms to set initial messages or approximate marginals. However the initialization of the messages and the approximate marginals can still be considered as hyperparameters in addition to the number of iterations. Nevertheless, EVMP and AIS-MP automates the individual inference steps in iterative settings and free the end user from specifying hyperparameters per inference steps. In noniterative inference settings, EVMP and AIS-MP can be considered as fully automated algorithms.

6.4 Outlook

This dissertation focuses on hybrid Bayesian inference approaches in probabilistic models. We provide an FFG-based framework for the automation of the proposed algorithms. In our framework, deterministic message passing algorithms constitute the main components of the inference engine, while Monte Carlo methods, the Laplace approximation, and stochastic optimization approaches are positioned as complementary components extending the capabilities of the inference engine. Almost all the algorithms presented in this dissertation are implemented in a Julia language-based message passing PPL, ForneyLab.jl. By PPL implementations, we aim to allow end-users with varying degrees of expertise in Bayesian inference to make the most of probabilistic modeling. Besides the PPL implementations, we strive to give intuition behind hybrid Bayesian inference techniques throughout the dissertation. We demonstrate how convenient our FFG and message passing-based framework is for hybrid inference procedure design. We hope that this manuscript enlightens the reader about hybrid Bayesian inference implementations so that the reader can go beyond the presented techniques when needed.

Bibliography

[1] Hang Yin and Christian Berger. When to use what data set for your self-driving car algorithm: An overview of publicly available driving datasets. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, October 2017. ISSN: 2153-0017.

[2] Kevin Lewis, Jason Kaufman, Marco Gonzalez, Andreas Wimmer, and Nicholas Christakis. Tastes, ties, and time: A new social network dataset using Facebook.com. *Social Networks*, 30(4):330–342, October 2008.

[3] Jacopo Tagliabue, Ciro Greco, Jean-Francis Roy, Bingqing Yu, Patrick John Chia, Federico Bianchi, and Giovanni Cassani. SIGIR 2021 E-Commerce Workshop Data Challenge. In *Proceedings of ACM SIGIR Workshop on eCommerce (SIGIR eCom'21)*, 2021. ACM, New York, NY, USA, 5 pages.

[4] James Bennett, Stan Lanning, and Netflix Netflix. The Netflix Prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[5] Claire Adam-Bourdarios, Glen Cowan, Cécile Germain, Isabelle Guyon, Balàzs Kégl, and David Rousseau. The Higgs boson machine learning challenge. In *Proceedings of the NIPS 2014 Workshop on High-energy Physics and Machine Learning*, pages 19–55. PMLR, August 2015. ISSN: 1938-7228.

[6] Kevin P Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023.

[7] E. T. Jaynes. How Does the Brain Do Plausible Reasoning? In Gary J. Erickson and C. Ray Smith, editors, *Maximum-Entropy and Bayesian Methods in Science and Engineering: Foundations*, Fundamental Theories of Physics, pages 1–24. Springer Netherlands, Dordrecht, 1988.

[8] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, May 2015.

[9] David M. Blei. Build, Compute, Critique, Repeat: Data Analysis with Latent Variable Models. *Annual Review of Statistics and Its Application*, 1(1):203–232, 2014. _eprint: https://doi.org/10.1146/annurev-statistics-022513-115657.

[10] E. T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, Cambridge, 2003.

[11] Andriy Mnih and Russ R Salakhutdinov. Probabilistic Matrix Factorization. In *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

[12] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th international conference on Machine learning - ICML '08*, pages 880–887, Helsinki, Finland, 2008. ACM Press.

[13] Ali Taylan Cemgil. Bayesian Inference for Nonnegative Matrix Factorisation Models. *Computational Intelligence and Neuroscience*, 2009:e785152, May 2009. Publisher: Hindawi.

[14] Mikkel N. Schmidt, Ole Winther, and Lars Kai Hansen. Bayesian Non-negative Matrix Factorization. In Tülay Adali, Christian Jutten, João Marcos Travassos Romano, and Allan Kardec Barros, editors, *Independent Component Analysis and Signal Separation*, volume 5441, pages 540–547. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. Series Title: Lecture Notes in Computer Science.

[15] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, March 2003.

[16] David M. Blei and John D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning - ICML '06*, pages 113–120, Pittsburgh, Pennsylvania, 2006. ACM Press.

[17] David M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, April 2012.

[18] David Barber, A. Taylan Cemgil, and Silvia Chiappa. *Bayesian Time Series Models*. Cambridge University Press, August 2011. Google-Books-ID: k4z6mOFsEv8C.

[19] Simo Särkkä. *Bayesian filtering and smoothing*, volume 3. Cambridge University Press, 2013.

[20] Eric Jacquier and Nicholas Polson. Bayesian Methods In Finance. In John Geweke, Gary Koop, and Herman Van Dijk, editors, *The Oxford Handbook of Bayesian Econometrics*, pages 438–512. Oxford University Press, September 2011.

[21] Chris Bracegirdle and David Barber. Bayesian conditional cointegration. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pages 1691–1698, Madison, WI, USA, June 2012. Omnipress.

[22] Jinwen Qiu, S. Rao Jammalamadaka, and Ning Ning. Multivariate Bayesian Structural Time Series Model. *Journal of Machine Learning Research*, 19(68):1–33, 2018.

[23] Qiang Huo, Chorkin Chan, and Chin Hui Lee. Bayesian adaptive learning of the parameters of hidden Markov model for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 3(5):334–345, September 1995. Conference Name: IEEE Transactions on Speech and Audio Processing.

[24] J. Vermaak, C. Andrieu, A. Doucet, and S.J. Godsill. Particle methods for Bayesian modeling and enhancement of speech signals. *IEEE Transactions on Speech and Audio Processing*, 10(3):173–185, March 2002. Conference Name: IEEE Transactions on Speech and Audio Processing.

[25] Albert Podusenko, Bart van Erp, Magnus Koudahl, and Bert de Vries. AIDA: An Active Inference-Based Design Agent for Audio Processing Algorithms. *Frontiers in Signal Processing*, 2, 2022.

[26] Charles Kemp, Joshua B. Tenenbaum, Thomas L. Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *Proceedings of the 21st national conference on Artificial intelligence - Volume 1*, AAAI'06, pages 381–388, Boston, Massachusetts, July 2006. AAAI Press.

[27] Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. Mixed Membership Stochastic Blockmodels. *Journal of Machine Learning Research*, 9(65):1981–2014, 2008.

[28] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents series. MIT Press, Cambridge, MA, USA, August 2005.

[29] Christian Hoffmann and Philipp Rostalski. Linear Optimal Control on Factor Graphs — A Message Passing Perspective —. *IFAC-PapersOnLine*, 50(1):6314–6319, July 2017.

[30] Sergey Levine. Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review. *arXiv:1805.00909 [cs, stat]*, May 2018. arXiv: 1805.00909.

[31] Francesco A. N. Palmieri, Krishna R. Pattipati, Giovanni Fioretti, Giovanni Di Gennaro, and Amedeo Buonanno. Path Planning Using Probability Tensor Flows. *IEEE Aerospace and Electronic Systems Magazine*, 36(1):34–45, January 2021. Conference Name: IEEE Aerospace and Electronic Systems Magazine.

[32] Francesco A. N. Palmieri, Krishna R. Pattipati, Giovanni Di Gennaro, Giovanni Fioretti, Francesco Verolla, and Amedeo Buonanno. A Unifying View of Estimation and Control Using Belief Propagation With Application to Path Planning. *IEEE Access*, 10:15193–15216, 2022. Conference Name: IEEE Access.

[33] Wei Ji Ma. Bayesian Decision Models: A Primer. *Neuron*, 104(1):164–175, October 2019.

[34] Rui Fan, Jianhao Jiao, Haoyang Ye, Yang Yu, Ioannis Pitas, and Ming Liu. Key Ingredients of Self-Driving Cars. *arXiv:1906.02939 [cs, eess]*, August 2019. arXiv: 1906.02939.

[35] Lawrence H. Erickson, Joseph Knuth, Jason M. O'Kane, and Steven M. LaValle. Probabilistic localization with a blind robot. In *2008 IEEE International Conference on Robotics and Automation*, pages 1821–1827, May 2008. ISSN: 1050-4729.

[36] Amos Tversky and Daniel Kahneman. Judgment under Uncertainty: Heuristics and Biases. *Science*, 185(4157):1124–1131, September 1974. Publisher: American Association for the Advancement of Science.

[37] David C. Knill and Alexandre Pouget. The Bayesian brain: the role of uncertainty in neural coding and computation. *Trends in Neurosciences*, 27(12):712–719, December 2004.

[38] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 110(3):457–506, March 2021.

[39] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, November 2016. Google-Books-ID: omivDQAAQBAJ.

[40] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986. Number: 6088 Publisher: Nature Publishing Group.

[41] Atılım Günes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *The Journal of Machine Learning Research*, 18(1):5595–5637, 2017. Publisher: JMLR. org.

[42] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: A CPU and GPU Math Compiler in Python. pages 18–24, Austin, Texas, 2010.

[43] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv:1408.5093 [cs]*, June 2014. arXiv: 1408.5093.

[44] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. page 19, 2015.

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv:1912.01703 [cs, stat]*, December 2019. arXiv: 1912.01703.

[46] Michael Innes, Elliot Saba, Keno Fischer, Dhairya Gandhi, Marco Concetto Rudilosso, Neethu Mariya Joy, Tejan Karmali, Avik Pal, and Viral Shah. Fashionable Modelling with Flux. *arXiv:1811.01457 [cs]*, November 2018. arXiv: 1811.01457.

[47] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. An Introduction to Probabilistic Programming. *arXiv:1809.10756 [cs, stat]*, September 2018. arXiv: 1809.10756.

[48] Radford M Neal and et al. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):113–162, 2011.

[49] Matthew D Hoffman and Andrew Gelman. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

[50] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei. Automatic differentiation variational inference. *The Journal of Machine Learning Research*, 18(1):430–474, 2017. Publisher: JMLR. org.

[51] Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial intelligence and statistics*, pages 814–822. PMLR, 2014.

[52] David John Cameron Mackay. Introduction to monte carlo methods. In *Learning in graphical models*, pages 175–204. Springer, 1998.

[53] Matthew D. Hoffman, Matthew J. Johnson, and Dustin Tran. Autoconj: recognizing and exploiting conjugacy without a domain-specific language. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 10739–10749, Red Hook, NY, USA, December 2018. Curran Associates Inc.

[54] Fritz Obermeyer, Eli Bingham, Martin Jankowiak, Justin Chiu, Neeraj Pradhan, Alexander Rush, and Noah Goodman. Tensor Variable Elimination for Plated Factor Graphs. *arXiv:1902.03210 [cs, stat]*, May 2019. arXiv: 1902.03210.

[55] Fritz Obermeyer, Eli Bingham, Martin Jankowiak, Du Phan, and Jonathan P. Chen. Functional Tensors for Probabilistic Programming, March 2020. Number: arXiv:1910.10775 arXiv:1910.10775 [cs, stat].

[56] G.D. Forney. Codes on graphs: normal realizations. *IEEE Transactions on Information Theory*, 47(2):520–548, February 2001. Conference Name: IEEE Transactions on Information Theory.

[57] Hans-Andrea Loeliger, Justin Dauwels, Junli Hu, Sascha Korl, Li Ping, and Frank R Kschischang. The factor graph approach to model-based signal processing. *Proceedings of the IEEE*, 95(6):1295–1322, 2007. Publisher: IEEE.

[58] Michael I. Jordan. Graphical Models. *Statistical Science*, 19(1):140–155, February 2004. Publisher: Institute of Mathematical Statistics.

[59] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. Adaptive computation and machine learning. MIT Press, Cambridge, MA, 2009.

[60] Siqi Nie, Cassio de Campos, and Qiang Ji. Learning Bayesian Networks with Bounded Tree-width via Guided Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), March 2016. Number: 1.

[61] Mauro Scanagatta, Giorgio Corani, Cassio Polpo de Campos, and Marco Zaffalon. Approximate structure learning for large Bayesian networks. *Machine Learning*, 107(8):1209–1227, September 2018.

[62] Marco Cox, Thijs van de Laar, and Bert de Vries. A factor graph approach to automated design of Bayesian signal processing algorithms. *International Journal of Approximate Reasoning*, 104:185–204, January 2019.

[63] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[64] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge University Press, 2003.

[65] John Winn and Christopher M. Bishop. Variational message passing. *Journal of Machine Learning Research*, 6(Apr):661–694, 2005.

[66] Justin Dauwels. On Variational Message Passing on Factor Graphs. In *IEEE International Symposium on Information Theory*, pages 2546–2550, June 2007.

[67] Thomas P Minka. Expectation Propagation for approximate Bayesian inference. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 362–369, 2001.

[68] Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: an empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, UAI'99, pages 467–475, San Francisco, CA, USA, July 1999. Morgan Kaufmann Publishers Inc.

[69] İsmail Şenöz, Thijs van de Laar, Dmitry Bagaev, and Bert de Vries. Variational Message Passing and Local Constraint Manipulation in Factor Graphs. *Entropy*, 23(7):807, July 2021. Number: 7 Publisher: Multidisciplinary Digital Publishing Institute.

[70] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on information theory*, 51(7):2282–2312, 2005. Publisher: IEEE.

[71] T. Heskes. Convexity Arguments for Efficient Minimization of the Bethe and Kikuchi Free Energies. *Journal of Artificial Intelligence Research*, 26:153–190, June 2006.

[72] Sascha Korl. A factor graph approach to signal modelling, system identification and filtering. *Series in signal and information processing*, 15, 2005. Accepted: 2017-10-03T14:32:24Z ISBN: 9783866280328 Publisher: Hartung-Gorre Verlag University: ETH Zurich.

[73] Karl J. Friston, Thomas Parr, and Bert de Vries. The graphical brain: Belief propagation and active inference. *Network Neuroscience*, 1(4):381–414, December 2017.

[74] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 681–688, Madison, WI, USA, June 2011. Omnipress.

[75] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[76] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[77] Rudolph Van Der Merwe. *Sigma-point kalman filters for probabilistic inference in dynamic state-space models*. PhD Thesis, Oregon Health & Science University, 2004. AAI3129163.

[78] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte Carlo Gradient Estimation in Machine Learning. *arXiv:1906.10652 [cs, math, stat]*, September 2020. arXiv: 1906.10652.

[79] Cheng Zhang, Judith Butepage, Hedvig Kjellstrom, and Stephan Mandt. Advances in Variational Inference. *arXiv:1711.05597 [cs, stat]*, October 2018. arXiv: 1711.05597.

[80] Thijs van de Laar. *Automated design of Bayesian signal processing algorithms*. Technische Universiteit Eindhoven, 2019. OCLC: 8153707591.

[81] Marco Cox and Bert De Vries. Robust Expectation Propagation in Factor Graphs Involving Both Continuous and Binary Variables. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 2583–2587, Rome, September 2018. IEEE.

[82] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877, April 2017.

[83] Matt Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic Variational Inference. *arXiv:1206.7051 [cs, stat]*, April 2013. arXiv: 1206.7051.

[84] Mohammad Khan and Wu Lin. Conjugate-Computation Variational Inference: Converting Variational Inference in Non-Conjugate Models to Inferences in Conjugate Models. In *Artificial Intelligence and Statistics*, pages 878–887. PMLR, 2017.

[85] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, May 2014. arXiv: 1312.6114.

[86] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1278–II–1286, Beijing, China, June 2014. JMLR.org.

[87] Michalis Titsias and Miguel Lázaro-Gredilla. Doubly stochastic variational Bayes for non-conjugate inference. In *International conference on machine learning*, pages 1971–1979, 2014.

[88] Aki Vehtari, Andrew Gelman, Tuomas Sivula, Pasi Jylänki, Dustin Tran, Swupnil Sahai, Paul Blomstedt, John P Cunningham, David Schiminovich, and Christian P Robert. Expectation Propagation as a Way of Life: A Framework for Bayesian Inference on Partitioned Data. *Journal of Machine Learning Research*, 21(17):1–53, 2020.

[89] Martin J Wainwright and Michael I Jordan. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

[90] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An Introduction to Variational Methods for Graphical Models. In Michael I. Jordan, editor, *Learning in Graphical Models*, pages 105–161. Springer Netherlands, Dordrecht, 1998.

[91] Michael I Jordan. *An Introduction to Probabilistic Graphical Models*. 2007. in preparation.

[92] Matthew J Beal. *Variational algorithms for approximate Bayesian inference*. PhD Thesis, UCL (University College London), 2003.

[93] Onno Zoeter and Tom Heskes. Gaussian Quadrature Based Expectation Propagation. In *International Workshop on Artificial Intelligence and Statistics*, pages 445–452. PMLR, January 2005. ISSN: 2640-3498.

[94] Francesco A. N. Palmieri. A Comparison of Algorithms for Learning Hidden Variables in Bayesian Factor Graphs in Reduced Normal Form. *IEEE Transactions on Neural Networks and Learning Systems*, 27(11):2242–2255, November 2016. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[95] Ulrich Paquet and Noam Koenigstein. One-class collaborative filtering with random graphs. In *Proceedings of the 22nd international conference on World Wide Web - WWW '13*, pages 999–1008, Rio de Janeiro, Brazil, 2013. ACM Press.

[96] Andrés R Masegosa, Ana M Martínez, Helge Langseth, Thomas D Nielsen, Antonio Salmerón, Darío Ramos-López, and Anders L Madsen. d-VMP: Distributed Variational Message Passing. In *Proceedings of the Eighth International Conference on Probabilistic Graphical Models*, pages 321–332. PMLR, August 2016. ISSN: 1938-7228.

[97] Bob Carpenter, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A Probabilistic Programming Language. *Journal of Statistical Software*, 76(1), 2017.

[98] Joshua V. Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A. Saurous. TensorFlow Distributions. *arXiv:1711.10604 [cs, stat]*, November 2017.

[99] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 20(28):1–6, 2019.

[100] Hong Ge, Kai Xu, and Zoubin Ghahramani. Turing: A Language for Flexible Probabilistic Inference. In *International Conference on Artificial Intelligence and Statistics*, pages 1682–1690. PMLR, March 2018.

[101] T. Minka, J.M. Winn, J.P. Guiver, Y. Zaykov, D. Fabian, and J. Bronskill. */Infer.NET 0.3*. 2018.

[102] Surya T Tokdar and Robert E Kass. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1):54–60, 2010. Publisher: Wiley Online Library.

[103] Patrick K Mogensen and Asbjørn N Riseth. Optim: A mathematical optimization package for Julia. *Journal of Open Source Software*, 3(24):615, April 2018.

[104] Jeff Bezanson, Stefan Karpinski, Viral B Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*, 2012.

[105] Justin Dauwels, Sascha Korl, and Hans-Andrea Loeliger. Particle methods as message passing. In *IEEE International Symposium on Information Theory*, pages 2052–2056, 2006.

[106] Ismail Şenöz and Bert De Vries. Online variational message passing in the hierarchical Gaussian filter. In *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2018.

[107] Christoph D. Mathys, Ekaterina I. Lomakina, Jean Daunizeau, Sandra Iglesias, Kay H. Brodersen, Karl J. Friston, and Klaas E. Stephan. Uncertainty in perception and the Hierarchical Gaussian Filter. *Frontiers in Human Neuroscience*, 8, November 2014.

[108] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960. Publisher: American Society of Mechanical Engineers.

[109] Zoubin Ghahramani and Geoffrey E Hinton. Parameter estimation for linear dynamical systems. Technical Report CRG-TR-92-2, Department of Computer Science, University of Toronto, 1996.

[110] Zoubin Ghahramani and Geoffrey E Hinton. Variational learning for switching state-space models. *Neural computation*, 12(4):831–864, 2000. Publisher: MIT Press.

[111] Philip Heidelberger and Peter D. Welch. Simulation Run Length Control in the Presence of an Initial Transient. *Operations Research*, 31(6):1109–1144, 1983. Publisher: INFORMS.

[112] Mary Kathryn Cowles and Bradley P. Carlin. Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review. *Journal of the American Statistical Association*, 91(434):883–904, June 1996. Publisher: Taylor & Francis _eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1996.10476956.

[113] Michael Betancourt. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[114] Frank Wood, Jan Willem Meent, and Vikash Mansinghka. A new approach to probabilistic programming inference. In *Artificial Intelligence and Statistics*, pages 1024–1032. PMLR, 2014.

[115] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010. Publisher: Wiley Online Library.

[116] Jarrett Revels. *ReverseDiff.jl*. 2017.

[117] Nando De Freitas, Pedro Højen-Sørensen, Michael I Jordan, and Stuart Russell. Variational MCMC. In *Proceedings of the Seventeenth conference on Uncertainty in Artificial Intelligence*, pages 120–127. Morgan Kaufmann Publishers Inc., 2001.

[118] Ydo Wexler and Dan Geiger. Importance sampling via variational optimization. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, pages 426–433, 2007.

[119] Tim Salimans, Diederik Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226, 2015.

[120] Lifeng Ye, Alexandros Beskos, Maria De Iorio, and Jie Hao. Monte Carlo co-ordinate ascent variational inference. *Statistics and Computing*, pages 1–19, 2020. Publisher: Springer.

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

[122] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEEE proceedings on radar and signal processing*, volume 140, pages 107–113, 1993. Issue: 2.

[123] Andrew Frank, Padhraic Smyth, and Alexander Ihler. Particle-based variational inference for continuous systems. *Advances in Neural Information Processing Systems*, 22:826–834, 2009.

[124] Alexander Ihler and David McAllester. Particle belief propagation. In *Artificial intelligence and statistics*, pages 256–263. PMLR, 2009.

[125] Martin J Wainwright, Tommi S Jaakkola, and Alan S Willsky. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, 51(7):2313–2335, 2005.

[126] Ardavan Saeedi, Tejas D Kulkarni, Vikash K Mansinghka, and Samuel J Gershman. Variational particle approximations. *The Journal of Machine Learning Research*, 18(1):2328–2356, 2017. Publisher: JMLR. org.

[127] Tapani Raiko, Harri Valpola, Markus Harva, and Juha Karhunen. Building Blocks for Variational Bayesian Learning of Latent Variable Models. *Journal of Machine Learning Research*, 8(1), 2007.

[128] David A Knowles and Tom Minka. Non-conjugate variational message passing for multinomial and binary regression. In *Advances in Neural Information Processing Systems*, pages 1701–1709, 2011.

[129] Víctor Elvira and Luca Martino. Advances in Importance Sampling. *arXiv:2102.05407 [stat]*, March 2022. arXiv: 2102.05407.

[130] Monica F. Bugallo, Victor Elvira, Luca Martino, David Luengo, Joaquin Miguez, and Petar M. Djuric. Adaptive Importance Sampling: The past, the present, and the future. *IEEE Signal Processing Magazine*, 34(4):60–79, July 2017. Conference Name: IEEE Signal Processing Magazine.

[131] Yousef El-Laham and Mónica F. Bugallo. Stochastic Gradient Population Monte Carlo. *IEEE Signal Processing Letters*, 27:46–50, 2020. Conference Name: IEEE Signal Processing Letters.

[132] Ernest K. Ryu and Stephen P. Boyd. Adaptive Importance Sampling via Stochastic Convex Programming. *arXiv:1412.4845 [math, stat]*, January 2015. arXiv: 1412.4845.

[133] Thomas Minka. Divergence measures and message passing. Technical Report MSR-TR-2005-173, Microsoft Research Cambridge, 2005.

[134] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951. Publisher: Institute of Mathematical Statistics.

[135] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[136] Ximing Li, Changchun Li, Jinjin Chi, and Jihong Ouyang. Variance Reduction in Black-box Variational Inference by Adaptive Importance Sampling. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 2404–2410, Stockholm, Sweden, July 2018. International Joint Conferences on Artificial Intelligence Organization.

[137] SILSO World Data Center. The International Sunspot Number. *International Sunspot Number Monthly Bulletin and online catalogue*, 1945-2020.

[138] George Casella and Christian P. Robert. Rao-Blackwellisation of Sampling Schemes. *Biometrika*, 83(1):81–94, 1996. Publisher: [Oxford University Press, Biometrika Trust].

[139] Art B. Owen. *Monte Carlo theory, methods and examples*. 2013.

[140] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[141] Shun'ichi Amari. *Information Geometry and Its Applications*, volume 194 of *Applied Mathematical Sciences*. Springer Japan, Tokyo, 2016.

[142] Dmitry Bagaev and Bert de Vries. Reactive Message Passing for Scalable Bayesian Inference. Technical Report arXiv:2112.13251, arXiv, December 2021. arXiv:2112.13251 [cs] type: article.

[143] Mohammad Emtiyaz Khan and Håvard Rue. The Bayesian Learning Rule. *arXiv:2107.04562 [cs, stat]*, July 2021. arXiv: 2107.04562.

[144] Ulrich Paquet. On the Convergence of Stochastic Variational Inference in Bayesian Networks. In *NIPS Workshop on Variational Inference*, 2014. arXiv: 1507.04505.

[145] Manfred Opper and Cédric Archambeau. The Variational Gaussian Approximation Revisited. *Neural Computation*, 21(3):786–792, March 2009.

[146] Arnaud Doucet, Nando de Freitas, Kevin Murphy, and Stuart Russell. Rao-blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, UAI'00, pages 176–183, San Francisco, CA, USA, June 2000. Morgan Kaufmann Publishers Inc.

[147] Fredrik Lindsten, Pete Bunch, Simo Särkkä, Thomas B. Schön, and Simon J. Godsill. Rao-Blackwellized particle smoothers for conditionally linear Gaussian models. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):353–365, March 2016. arXiv:1505.06357 [stat].

[148] David Wingate and Theophane Weber. Automated Variational Inference in Probabilistic Programming. *arXiv:1301.1299 [cs, stat]*, January 2013. arXiv: 1301.1299.

[149] Martin Jankowiak and Fritz Obermeyer. Pathwise Derivatives Beyond the Reparameterization Trick. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2235–2244. PMLR, July 2018. ISSN: 2640-3498.

[150] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1530–1538, Lille, France, July 2015. JMLR.org.

[151] R. A. Fisher. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 7(2):179–188, 1936. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1469-1809.1936.tb02137.x.

[152] Edgar Anderson. The Species Problem in Iris. *Annals of the Missouri Botanical Garden*, 23(3):457–509, 1936. Publisher: Missouri Botanical Garden Press.

[153] Bradley P. Carlin, Alan E. Gelfand, and Adrian F. M. Smith. Hierarchical Bayesian Analysis of Changepoint Problems. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 41(2):389–405, 1992. Publisher: [Wiley, Royal Statistical Society].

[154] Ryan Prescott Adams and David J. C. MacKay. Bayesian Online Changepoint Detection. *arXiv:0710.3742 [stat]*, October 2007. arXiv: 0710.3742.

[155] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning, second edition: An Introduction*. MIT Press, November 2018. Google-Books-ID: uWV0DwAAQBAJ.

[156] A. Taylan Cemgil. A Tutorial Introduction to Monte Carlo Methods, Markov Chain Monte Carlo and Particle Filtering. In *Academic Press Library in Signal Processing*, volume 1, pages 1065–1114. Elsevier, 2014.

[157] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

[158] J. Revels, M. Lubin, and T. Papamarkou. Forward-Mode Automatic Differentiation in Julia. *arXiv:1607.07892 [cs.MS]*, 2016.

[159] Laurent Valentin Jospin, Wray Buntine, Farid Boussaid, Hamid Laga, and Mohammed Bennamoun. Hands-on Bayesian Neural Networks – a Tutorial for Deep Learning Users. *arXiv:2007.06823 [cs, stat]*, September 2021. arXiv: 2007.06823.

[160] Francisco R Ruiz, Michalis Titsias RC AUEB, and David Blei. The Generalized Reparameterization Gradient. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[161] Mikhail Figurnov, Shakir Mohamed, and Andriy Mnih. Implicit Reparameterization Gradients. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[162] Rajesh Ranganath, Chong Wang, Blei David, and Eric Xing. An Adaptive Learning Rate for Stochastic Variational Inference. In *Proceedings of the 30th International Conference on Machine Learning*, pages 298–306. PMLR, May 2013. ISSN: 1938-7228.

[163] Akash Kumar Dhaka, Alejandro Catalina, Michael Riis Andersen, Måns Magnusson, Jonathan H. Huggins, and Aki Vehtari. Robust, Accurate Stochastic Optimization for Variational Inference. In *Advances in Neural Information Processing Systems*, volume 33, September 2020.

[164] Wu Lin, Mark Schmidt, and Mohammad Emtiyaz Khan. Handling the Positive-Definite Constraint in the Bayesian Learning Rule. In *Proceedings of the 37th International Conference on Machine Learning*, pages 6116–6126. PMLR, November 2020. ISSN: 2640-3498.

[165] Jorge Nocedal. Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation*, 35(151):773–782, 1980. Publisher: American Mathematical Society.

[166] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528, August 1989.

[167] Eike Petersen, Christian Hoffmann, and Philipp Rostalski. On Approximate Nonlinear Gaussian Message Passing on Factor Graphs. In *2018 IEEE Statistical Signal Processing Workshop (SSP)*, pages 513–517, June 2018.

[168] Michael Roth, Gustaf Hendeby, and Fredrik Gustafsson. Nonlinear Kalman Filters Explained: A Tutorial on Moment Computations and Sigma Point Methods. *Journal of Advances in Information Fusion*, 11(1):47–70, 2016. Publisher: International society of information fusion.

[169] Vıctor Elvira, Pau Closas, and Luca Martino. Gauss-Hermite Quadrature for non-Gaussian Inference via an Importance Sampling Interpretation. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5, September 2019. ISSN: 2076-1465.

[170] Maren Mahsereci and Philipp Hennig. Probabilistic line searches for stochastic optimization. *The Journal of Machine Learning Research*, 18(1):4262–4320, January 2017.

[171] Sharan Vaswani, Aaron Mishkin, Issam Laradji, Mark Schmidt, Gauthier Gidel, and Simon Lacoste-Julien. Painless Stochastic Gradient: Interpolation, Line-Search, and Convergence Rates. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[172] Shinsuke Koyama, Lucia Castellanos Pérez-Bolde, Cosma Rohilla Shalizi, and Robert E Kass. Approximate methods for state-space models. *Journal of the American Statistical Association*, 105(489):170–180, 2010. Publisher: Taylor & Francis.

[173] Jakob H Macke, Lars Buesing, John P Cunningham, Byron M Yu, Krishna V Shenoy, and Maneesh Sahani. Empirical models of spiking in neural populations. *Advances in neural information processing systems*, 24:1350–1358, 2011.

[174] Alex J Smola, SVN Vishwanathan, and Eleazar Eskin. Laplace propagation. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, pages 441–448, 2004.

[175] Luigi Acerbi. Variational bayesian monte carlo. In *Advances in Neural Information Processing Systems*, pages 8213–8223, 2018.

[176] Jiří Ajgl and Miroslav Šimandl. Differential entropy estimation by particles. *IFAC Proceedings Volumes*, 44(1):11991–11996, 2011. Publisher: Elsevier.

[177] Arnaud Doucet, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

Appendix A

Stationary Points of the Free Energy

This appendix is based on the original work referenced below.

- Semih Akbayrak, İsmail Şenöz, Alp Sarı and Bert de Vries, *Probabilistic Programming with Stochastic Variational Message Passing*, International Journal of Approximate Reasoning, 2022

Consider the free energy objective given in (2.4). This objective is a functional of $q(\boldsymbol{z}_b)$. We first find the stationary $q(z_j)$ for this functional. Let us rewrite it here:

$$\mathcal{F}[q(z_j)] = \mathbb{E}_{q(z_j)}[\log q(z_j)] - \mathbb{E}_{q(z_j)}[\log m_{jb}(z_j)] - \mathbb{E}_{q(z_j)}\left[\mathbb{E}_{q(\boldsymbol{z}_{b\setminus j})}[\log f_b(\boldsymbol{z_b})]\right] + c,$$

where c stands for the terms independent of z_j. It can be equivalently written as

$$\begin{aligned}\mathcal{F}[q(z_j)] =& D_{\mathrm{KL}}\left[q(z_j)\middle\|\frac{m_{jb}(z_j)\exp(\mathbb{E}_{q(\boldsymbol{z}_{b\setminus j})}[\log f_b(\boldsymbol{z_b})])}{\int m_{jb}(z_j)\exp(\mathbb{E}_{q(\boldsymbol{z}_{b\setminus j})}[\log f_b(\boldsymbol{z_b})])\mathrm{d}z_j}\right] \\ &- \log\int m_{jb}(z_j)\exp(\mathbb{E}_{q(\boldsymbol{z}_{b\setminus j})}[\log f_b(\boldsymbol{z_b})])\mathrm{d}z_j + c,\end{aligned}$$

which is a KL divergence summed with a constant. Setting $q(z_j)$ equal to the right hand side of the KL divergence minimizes the free energy w.r.t. $q(z_j)$.

Now, consider the other interpretation of the variational objective that casts the free energy as a function of η_j as in (5.3) given that $q(z_j;\eta_j) = h_j(z_j)\exp(\phi_j(z_j)^\intercal\eta_j - A_j(\eta_j))$ with a constant $h_j(z_j)$. Functional form of the message $m_{jb}(z_j)$ is given in

(5.4). Then the gradient of the free energy w.r.t. η_j is

$$\begin{aligned}
\nabla_{\eta_j}\mathcal{F}(\eta_j) &= \nabla_{\eta_j}\left(\mathbb{E}_{q(z_j;\eta)}[\log q(z_j;\eta_j)] - \mathbb{E}_{q(z_j;\eta_j)}\left[\log m_{jb}(z_j)\right] - \mathbb{E}_{q(z_j;\eta_j)}\left[\log m_{bj}(z_j)\right]\right) \\
&= \nabla_{\eta_j}\left(\mathbb{E}_{q(z_j;\eta_j)}[\phi_j(z_j)]^\intercal(\eta_j - \eta_{jb}) - A_j(\eta_j) - \mathbb{E}_{q(z_j;\eta_j)}\left[\log m_{bj}(z_j)\right]\right) \\
&= \nabla_{\eta_j}\mathbb{E}_{q(z_j;\eta_j)}[\phi_j(z_j)]^\intercal(\eta_j - \eta_{jb}) + \mathbb{E}_{q(z_j;\eta_j)}[\phi_j(z_j)] - \nabla_{\eta_j}A_j(\eta_j) \\
&\quad - \nabla_{\eta_j}\mathbb{E}_{q(z_j;\eta_j)}[\log m_{bj}(z_j)] \\
&= \nabla^2_{\eta_j}A_j(\eta_j)\left[\eta_j - \eta_{jb}\right] - \nabla_{\eta_j}\mathbb{E}_{q(z_j;\eta_j)}[\log m_{bj}(z_j)].
\end{aligned}$$

The last line above follows from $\nabla_{\eta_j}A(\eta_j) = \mathbb{E}_{q(z_j;\eta_j)}[\phi_j(z_j)]$ (See (B.14)). We denote the Hessian of the log-normalizer, $\nabla^2_{\eta_j}A_j(\eta_j)$ with $G(\eta_j)$, which is the Fisher information matrix of $q(z_j;\eta_j)$. Following [83], we write the natural gradient of the free energy as

$$\nabla^N_{\eta_j}\mathcal{F}(\eta_j) = G^{-1}(\eta_j)\nabla_{\eta_j}\mathcal{F}(\eta_j) = \eta_j - \Big(\eta_{jb} + G^{-1}(\eta_j)\nabla_{\eta_j}\mathbb{E}_{q(z_j;\eta_j)}[\log m_{bj}(z_j)]\Big).$$

Appendix B

On The Applicability of VMP

This appendix is based on the original work referenced below.

- Semih Akbayrak, Ivan Bocharov, and Bert de Vries, *Extended Variational Message Passing for Automated Approximate Bayesian Inference*, Special issue on Bayesian Inference in Probabilistic Graphical Models, Entropy, 2021

In this chapter, we show that the applicability of the VMP algorithm relies on connected factors being conjugate pairs in exponential family of distributions. Non-conjugate connected factors lead to intractable posteriors and messages. Nevertheless, we show that for a given soft factor, the corresponding VMP messages are locally expressed in terms of some expectation quantities. If these expectations are not available in closed-form, then we can estimate them to approximate the VMP messages around the non-conjugate factor pairs.

Let us focus on Fig. 2.1. We postulate the following assumptions:

- $f_a(\mathbf{z}_a)$ is an element of the exponential family (EF) of distributions, i.e.,

$$\begin{aligned} f_a(\mathbf{z}_a) &= p(z_j|\mathbf{z}_{a\setminus j}) \\ &= h_j(z_j) \exp\left(\phi_j(z_j)^\intercal \lambda_{aj}(\mathbf{z}_{a\setminus j}) - \log Z_j(\mathbf{z}_{a\setminus j})\right). \end{aligned} \tag{B.1}$$

  In this equation, $h_j(z_j)$ is a base measure, $\lambda_{aj}(\mathbf{z}_{a\setminus j})$ is a vector of natural (or canonical) parameters, $\phi_j(z_j)$ are the sufficient statistics, and $Z_j(\mathbf{z}_{a\setminus j})$ is the partition function, i.e., $Z_j(\mathbf{z}_{a\setminus j}) = \int h_j(z_j) \exp(\phi_j(z_j)^\intercal \lambda_{aj}(\mathbf{z}_{a\setminus j}))dz_j$. It is always possible to write the log-partition function as a function of natural parameters $A_j(\lambda_{aj}(\mathbf{z}_{a\setminus j}))$, such that $A_j(\lambda_{aj}(\mathbf{z}_{a\setminus j})) = \log Z_j(\mathbf{z}_{a\setminus j})$. Hence, the functions Z_j and A_j are related through $Z_j = \exp \circ A_j \circ \lambda_j$. We are allowed to go back and forth between A_j and Z_j.

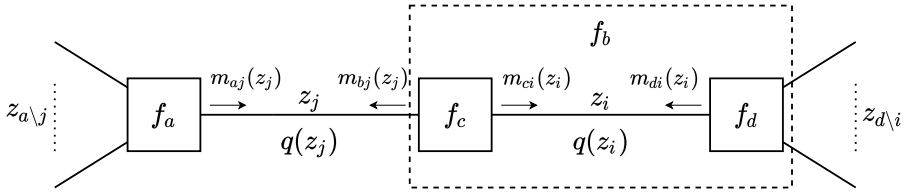


Figure B.1: An example deterministic conditional is visualized for $f_c(\mathbf{z}_c) = p(z_i|z_j) = \delta(z_i - g(z_j))$. $f_c(\mathbf{z}_c)$ and $f_d(\mathbf{z}_d)$ together form the composite node $f_b(\mathbf{z}_b)$. The set of variables $\mathbf{z}_{d\backslash i}$ can be equivalently denoted by $\mathbf{z}_{b\backslash j}$. Similarly, $m_{bj}(z_j)$ and $m_{cj}(z_j)$ denote the same message.

- We differentiate a few cases for f_b:

  1. f_b is also an element of the EF, given by

$$\begin{aligned} f_b(\mathbf{z}_b) &= p(z_k|\mathbf{z}_{b\backslash k}) \\ &= h_k(z_k) \exp(\phi_k(z_k)^\intercal \lambda_{bk}(\mathbf{z}_{b\backslash k}) - \log Z_k(\mathbf{z}_{b\backslash k})) \,, \end{aligned} \tag{B.2}$$

     and is a *conditionally conjugate pair* with f_a for z_j. This (conditional conjugacy) property implies that, given $\mathbf{z}_{b\backslash k}$, we can modify f_b in such a way that its sufficient statistics will match the sufficient statistics of f_a. Technically, this means we can rewrite f_b as

$$f_b(\mathbf{z}_b) = h_k(z_k) \exp(\phi_j(z_j)^\intercal \lambda_{bj}(\mathbf{z}_{b\backslash j}) + c_{bj}(\mathbf{z}_{b\backslash j})). \tag{B.3}$$

     The crucial element of this rewrite is that both $f_a(\mathbf{z}_a)$ and $f_b(\mathbf{z}_b)$ have been written as exponential functions of the same sufficient statistics function $\phi_j(z_j)$. This case leads to the regular VMP update equations, see Section B.1.

     Our Extended VMP does not need this assumption and derives approximate VMP update rules for the following extensions.

  2. f_b is an element of the EF, but not amenable to the modification given in (B.3), i.e., it cannot be written as an exponential function of sufficient statistics $\phi_j(z_j)$. Therefore, f_b is not a conjugate pair with f_a for z_j.

  3. $f_b(\mathbf{z}_b)$ is a composition of a deterministic node with an EF node, see Figure B.1. In particular, in this case $f_b(\mathbf{z}_b)$ can be decomposed as

$$f_b(\mathbf{z}_b) = \int \delta(z_i - g(z_j)) f_d(\mathbf{z}_d) \mathrm{d}z_i \tag{B.4a}$$

$$= f_d(g(z_j), \mathbf{z}_{d\backslash i}), \tag{B.4b}$$

where $z_i = g(z_j)$ is a deterministic, possibly nonlinear transformation and $f_d(\mathbf{z}_d)$ is an element of the EF:

$$\begin{aligned} f_d(\mathbf{z}_d) &= p(z_k|\mathbf{z}_{d\setminus k}) \\ &= h_k(z_k)\exp(\phi_k(z_k)^\intercal \lambda_{dk}(\mathbf{z}_{d\setminus k}) - \log Z_k(\mathbf{z}_{d\setminus k})). \end{aligned} \tag{B.5}$$

We assume that the conjugate prior to f_d for random variable z_i has sufficient statistics vector $\phi_i(z_i)$, and hence (B.5) can be modified as

$$f_d(\mathbf{z}_d) = h_k(z_k)\exp(\phi_i(z_i)^\intercal \lambda_{di}(\mathbf{z}_{d\setminus i}) + c_{di}(\mathbf{z}_{d\setminus i})), \tag{B.6}$$

where $c_{di}(\mathbf{z}_{d\setminus i})$ refers to the terms that does not include z_i.

B.1 VMP with Conjugate Soft Factor Pairs

The original VMP algorithm arises as an efficient inference procedure in models that solely consist of conjugate factor pairs. This is because conjugate factor pairs yield analytically tractable messages and posterior calculations. Next, we shortly review the effect of conjugate factor pairs on VMP updates.

B.1.1 Messages and Posteriors

The VMP message from the factor f_a to z_j can easily be evaluated by applying the VMP message calculation rule to (B.1):

$$\begin{aligned} m_{aj}(z_j) &\propto \exp(\mathbb{E}_{q(\mathbf{z}_{a\setminus j})}[\log f_a(\mathbf{z}_a)]) \\ &\propto h_j(z_j)\exp\Big(\phi_j(z_j)^\intercal \underbrace{\mathbb{E}_{q(\mathbf{z}_{a\setminus j})}[\lambda_{aj}(\mathbf{z}_{a\setminus j})]}_{\eta_{aj}}\Big). \end{aligned} \tag{B.7}$$

Since f_b is conjugate to f_a, its functional form can be modified as (B.3) and by applying (2.7) to (B.3), we find the VMP message from the factor f_b to z_j:

$$m_{bj}(z_j) \propto \exp\Big(\phi_j(z_j)^\intercal \underbrace{\mathbb{E}_{q(\mathbf{z}_{b\setminus j})}[\lambda_{bj}(\mathbf{z}_{b\setminus j})]}_{\eta_{bj}}\Big). \tag{B.8}$$

Given that the messages $m_{aj}(z_j)$ and $m_{bj}(z_j)$ have the same sufficient statistics, the posterior update step reduces to summation of the messages' natural parameters:

$$q(z_j) \propto h_j(z_j)\exp\Big(\phi_j(z_j)^\intercal \underbrace{(\eta_{aj} + \eta_{bj})}_{\eta_j}\Big). \tag{B.9}$$

Note that we evaluate the posterior up to a normalization constant. Nevertheless, the log-normalizer function $A_j(\cdot)$ is readily available for EF distributions having sufficient statistics vector $\phi_j(z_j)$. As a consequence, the posterior evaluates to

$$q(z_j) = h_j(z_j) \exp\left(\phi_j(z_j)^\intercal \eta_j - A_j(\eta_j)\right). \tag{B.10}$$

Having showed that the conjugate factor pairs lead to a closed-form expression for posterior $q(z_j)$, we now investigate which expectation quantities related to $q(z_j)$ are required in the outgoing VMP messages from the factors f_a and f_b to, say z_l and z_k:

$$m_{al}(z_l) \propto \exp\left(\mathbb{E}_{q(\mathbf{z}_{a\setminus l})}[\log f_a(\mathbf{z}_a)]\right), \tag{B.11a}$$

$$m_{bk}(z_k) \propto \exp\left(\mathbb{E}_{q(\mathbf{z}_{b\setminus k})}[\log f_b(\mathbf{z}_b)]\right). \tag{B.11b}$$

In practice, the message $m_{al}(z_l)$ is explicitly calculated by isolating the terms with z_l in a sufficient statistics vector as it is done for z_j in (B.8). Similarly, $m_{bk}(z_k)$ is explicitly calculated, analogous to message $m_{aj}(z_j)$ in (B.7). Here, we follow a rather different approach to explicitly show the expectations related to $q(z_j)$ in the message calculations. Substituting f_a and f_b with (B.1) and (B.3) in (B.11), and keeping in mind that the mean-field assumption allows separation of the expectation quantities with distinct random variables, the messages evaluate to

$$m_{al}(z_l) \propto \exp\left(\mathbb{E}_{q(z_j)}[\phi_j(z_j)]^\intercal \mathbb{E}_{q(\mathbf{z}_{a\setminus\{j,l\}})}[\lambda_{aj}(\mathbf{z}_{a\setminus j})] - \mathbb{E}_{q(\mathbf{z}_{a\setminus\{j,l\}})}[\log Z_j(\mathbf{z}_{a\setminus j})]\right), \tag{B.12a}$$

$$m_{bk}(z_k) \propto h_k(z_k) \exp\left(\mathbb{E}_{q(z_j)}[\phi_j(z_j)]^\intercal \mathbb{E}_{q(\mathbf{z}_{b\setminus\{j,k\}})}[\lambda_{bj}(\mathbf{z}_{b\setminus j})] + \mathbb{E}_{q(\mathbf{z}_{b\setminus\{j,k\}})}[c_{bj}(\mathbf{z}_{b\setminus j})]\right). \tag{B.12b}$$

Notice that both messages require the moment parameters $\zeta_j = \Psi_j(\eta_j)$, i.e. expectation of the sufficient statistic vector $\phi_j(z_j)$ w.r.t. $q(z_j)$. Fortunately, in EF distributions, $\mathbb{E}_{q(z_j)}[\phi_j(z_j)]$ is available in closed form as the gradient of the log-normalizer [89, Proposition 3.1]:

$$\mathbb{E}_{q(z_j)}[\phi_j(z_j)] = \nabla_{\eta_j} A_j(\eta_j). \tag{B.13}$$

For the sake of completeness, we now show that this equality holds. Recall that

$A_j(\eta_j) = \log\left(\int h_j(z_j)\exp(\phi_j(z_j)^\intercal \eta_j)\,\mathrm{d}z_j\right)$. Then,

$$\begin{aligned}\nabla_{\eta_j} A_j(\eta_j) &= \frac{\nabla_{\eta_j}\int h_j(z_j)\exp(\phi_j(z_j)^\intercal\eta_j)\,\mathrm{d}z_j}{\int h_j(z_j)\exp(\phi_j(z_j)^\intercal\eta_j)\,\mathrm{d}z_j}\\ &= \frac{\int \phi_j(z_j)h_j(z_j)\exp(\phi_j(z_j)^\intercal\eta_j)\,\mathrm{d}z_j}{\exp(A_j(\eta_j))}\\ &= \int \phi_j(z_j)h_j(z_j)\exp(\phi_j(z_j)^\intercal\eta_j - A_j(\eta_j))\,\mathrm{d}z_j\\ &= \int \phi_j(z_j)q(z_j)\,\mathrm{d}z_j. \end{aligned} \tag{B.14}$$

B.1.2 Free energy

As in the message and the posterior calculations, conjugacy eases the free energy calculation. We investigate it for (2.5), the free energy terms that include z_j. $\mathcal{F}_j$ is decomposed as

$$\mathcal{F}_j = \underbrace{-\mathbb{E}_{q(\mathbf{z}_a)}[\log f_a(\mathbf{z}_a)] - \mathbb{E}_{q(\mathbf{z}_b)}[\log f_b(\mathbf{z}_b)]}_{\text{Average energy terms}} + \underbrace{\mathbb{E}_{q(z_j)}[\log q(z_j)]}_{\text{Negative entropy}}. \tag{B.15}$$

Substituting f_a, f_b and $q(z_j)$ with (B.1), (B.3) and (B.10) in the above expression:

$$\begin{aligned}\mathcal{F}_j =& -\mathbb{E}_{q(z_j)}[\log(h_j(z_j))] - \mathbb{E}_{q(z_j)}[\phi_j(z_j)]^\intercal\mathbb{E}_{q(\mathbf{z}_{a\setminus j})}[\lambda_{aj}(\mathbf{z}_{a\setminus j})] + \mathbb{E}_{q(\mathbf{z}_{a\setminus j})}[\log Z_j(\mathbf{z}_{a\setminus j})]\\ &- \mathbb{E}_{q(z_k)}[\log(h_k(z_k))] - \mathbb{E}_{q(z_j)}[\phi_j(z_j)]^\intercal\mathbb{E}_{q(\mathbf{z}_{b\setminus j})}[\lambda_{bj}(\mathbf{z}_{b\setminus j})] - \mathbb{E}_{q(\mathbf{z}_{b\setminus j})}[c_{bj}(\mathbf{z}_{b\setminus j})]\\ &+ \mathbb{E}_{q(z_j)}[\log(h_j(z_j))] + \mathbb{E}_{q(z_j)}[\phi_j(z_j)]^\intercal\eta_j - A_j(\eta_j). \end{aligned} \tag{B.16}$$

The expectation terms related to z_j in $\mathcal{F}_j$ are $\mathbb{E}_{q(z_j)}[\phi_j(z_j)]$ and $\mathbb{E}_{q(z_j)}[\log(h_j(z_j))]$. The former expectation is available in closed-form, (B.13). Thus $\mathcal{F}_j$ is analytically tractable for those distributions that possess closed form solution for $\mathbb{E}_{q(z_j)}[\log(h_j(z_j))]$. For the majority of EF distributions, the base measure is constant; hence the expectation $\mathbb{E}_{q(z_j)}[\log(h_j(z_j))]$ is available in closed form.

In short, conjugate factor pairs facilitate the VMP procedure by allowing closed-form expressions for updates of messages, posteriors and free energy. Moreover, although exceptions exist, similar to the normalization of the posterior (B.9), the messages $m_{aj}(z_j)$ and $m_{bj}(z_j)$ can be effortlessly normalized if the required expectations are known. Therefore, we can directly parameterize them with standard probability distributions and draw samples from them. This property of EF distributions plays a pivotal role in our automation of importance sampling procedure in Extended VMP.

B.2 VMP with Non-conjugate Soft Factor Pairs

Suppose that the soft factors f_a and f_b are no longer conjugate pairs, i.e., f_b given in (B.2) can be written in the form

$$f_b(\mathbf{z}_b) = h_k(z_k)\exp(\phi_{bj}(z_j)^\intercal \lambda_{bj}(\mathbf{z}_{b\setminus j}) + c_{bj}(\mathbf{z}_{b\setminus j})), \tag{B.17}$$

where crucially $\phi_{bj}(z_j) \neq \phi_j(z_j)$. Also notice that $\lambda_{bj}(\mathbf{z}_{b\setminus j})$ is the natural parameters after the modification of (B.2) to isolate the terms with z_j in the sufficient statistics vector. Therefore, the messages $m_{bj}(z_j)$ differ from $m_{aj}(z_j)$ (B.7) in sufficient statistics:

$$m_{bj}(z_j) \propto \exp\left(\phi_{bj}(z_j)^\intercal \mathbb{E}_{q(\mathbf{z}_{b\setminus j})}[\lambda_{bj}(\mathbf{z}_{b\setminus j})]\right). \tag{B.18}$$

In this case, the normalization constant calculation in the posterior update step is not straightforward anymore; and worse, it is often intractable. The term *intractable* refers to integrals that are not available in closed-form for continuous variables. For discrete variables, it refers to summations that are not achievable in a feasible amount of time. The lack of the normalization constant, $\int m_{aj}(z_j)m_{bj}(z_j)\,\mathrm{d}z_j$, hinders the calculation of the expectations with z_j terms that appear in out-going VMP messages from f_a and f_b to variables $\mathbf{z}_{a\setminus j}$ and $\mathbf{z}_{b\setminus j}$, respectively, e.g. $\mathbb{E}_{q(z_j)}[\phi_{bj}(z_j)]$ in

$$m_{bk}(z_k) \propto h_k(z_k)\exp\left(\mathbb{E}_{q(z_j)}[\phi_{bj}(z_j)]^\intercal \mathbb{E}_{q(\mathbf{z}_{b\setminus\{j,k\}})}[\lambda_{bj}(\mathbf{z}_{b\setminus j})] + \mathbb{E}_{q(\mathbf{z}_{b\setminus\{j,k\}})}[c_{bj}(\mathbf{z}_{b\setminus j})]\right). \tag{B.19}$$

As a result, non-conjugacies obstruct VMP procedure by hampering closed-form expectation calculations. Bear in mind that even though VMP procedure is obstructed due to intractable expectations, the messages are distinctly fixed for soft factors as functions of certain expectation quantities that are supposed to be calculated over their arguments. We use this property in our Extended VMP method.

B.3 VMP with Composite Nodes

In this subsection, we shed light on the issues with composite nodes that are constructed by composition of EF distribution soft factors and deterministic conditionals, i.e.

$$f_b(\mathbf{z}_b) = \int f_c(z_i, z_j) f_d(z_i, \mathbf{z}_{b\setminus j})\,\mathrm{d}z_i, \tag{B.20}$$

where $f_c(z_i, z_j) = p(z_i|z_j) = \delta(z_i - g(z_j))$ is a deterministic conditional distribution. Composite nodes enable us to build almost arbitrary factor nodes. For example,

a mixture likelihood distribution $p(y|z) = \mathcal{N}(y; \mu_1, \sigma^2)^z \mathcal{N}(y; \mu_2, \sigma^2)^{(1-z)}$ with z a selection variable, can be constructed by composing an EF soft factor, a Gaussian, with a deterministic factor as $\int p(y|x)p(x|z)\mathrm{d}z_i$, where $p(y|x) = \mathcal{N}(y; x, \sigma^2)$ and $p(x|z) = \delta(x - z\mu_1 - (1-z)\mu_2)$. However, composite nodes impose new challenges on inference procedures.

Now, let us try to calculate $q(z_j)$. The forward VMP message $m_{aj}(z_j)$ is given in (B.7). Suppose that the conjugate prior to EF soft factor f_d for z_i has the sufficient statistics vector $\phi_i(z_i)$ (see (B.6)). Then, the VMP message from f_b to z_j is

$$\begin{aligned} m_{bj}(z_j) &\propto \exp\left(\mathbb{E}_{q(\mathbf{z}_{b\setminus j})}\left[\log\left(\int \delta(z_i - g(z_j)) \exp\left(\phi_i(z_i)^\intercal \lambda_{di}(\mathbf{z}_{d\setminus i})\right) \mathrm{d}z_i\right)\right]\right) \\ &\propto \underbrace{\int \delta(z_i - g(z_j)) \underbrace{\exp\left(\phi_i(z_i)^\intercal \mathbb{E}_{q(\mathbf{z}_{d\setminus i})}\left[\lambda_{di}(\mathbf{z}_{d\setminus i})\right]\right)}_{\text{VMP:}m_{di}} \mathrm{d}z_i}_{\text{BP}} \\ &= \underbrace{\exp\left(\phi_i(g(z_j))^\intercal \mathbb{E}_{q(\mathbf{z}_{d\setminus i})}\left[\lambda_{di}(\mathbf{z}_{d\setminus i})\right]\right)}_{m_{di}(g(z_j))}. \end{aligned} \tag{B.21}$$

Note that the above message reduces to VMP message from f_d to z_i followed by Belief Propagation (BP) [63, 64]. The resulting backward message $m_{bj}(z_j)$ has the sufficient statistics vector $\phi_i(g(z_j))$. If $\phi_i(\cdot) = \phi_j(g^{-1}(\cdot))$, this case reduces to ordinary VMP as discussed in Section B.1; otherwise this case is a special case of Section B.2 and $q(z_j)$ is not available in closed-form. Hence, the outgoing messages from the factor nodes f_a and f_b, say

$$m_{al}(z_l) \propto \exp\left(\mathbb{E}_{q(z_j)}[\phi_j(z_j)]^\intercal \mathbb{E}_{q(\mathbf{z}_{a\setminus\{j,l\}})}[\lambda_{aj}(\mathbf{z}_{a\setminus j})] - \mathbb{E}_{q(\mathbf{z}_{a\setminus\{j,l\}})}[\log Z_j(\mathbf{z}_{a\setminus j})]\right), \tag{B.22a}$$

$$\begin{aligned} m_{bk}(z_k) &\propto \exp\Bigg(\mathbb{E}_{q(\mathbf{z}_{b\setminus k})}\Big[\log\Big(\int \delta(z_i - g(z_j)) \\ &\qquad h_k(z_k) \exp\left(\phi_i(z_i)^\intercal \lambda_{di}(\mathbf{z}_{d\setminus i}) + c_{di}(\mathbf{z}_{d\setminus i})\right) \mathrm{d}z_i\Big)\Big]\Bigg) \\ &= h(z_k) \exp\left(\mathbb{E}_{q(z_j)}[\phi_i(g(z_j))]^\intercal \mathbb{E}_{q(\mathbf{z}_{d\setminus\{i,k\}})}[\lambda_{di}(\mathbf{z}_{d\setminus i})] + \mathbb{E}_{q(\mathbf{z}_{d\setminus\{i,k\}})}[c_{di}(\mathbf{z}_{d\setminus i})]\right) \\ &= h(z_k) \exp\left(\mathbb{E}_{q(z_i)}[\phi_i(z_i)]^\intercal \mathbb{E}_{q(\mathbf{z}_{d\setminus\{i,k\}})}[\lambda_{di}(\mathbf{z}_{d\setminus i})] + \mathbb{E}_{q(\mathbf{z}_{d\setminus\{i,k\}})}[c_{di}(\mathbf{z}_{d\setminus i})]\right). \end{aligned} \tag{B.22b}$$

are intractable. The last line in the above derivations follows from the transformation of variables [157], i.e., $q(z_j) = q(z_i)|\frac{\mathrm{d}z_i}{\mathrm{d}z_j}|$, and expose the automatable nature of Variational Message Passing: the VMP message $m_{bk}(z_k)$ requires expectation

quantities that are related to arguments of the soft factor z_k is tied to, which is in this case f_d. Therefore, once the VMP message passing rule is defined for the factor f_d as a function of its arguments, we can instantiate the messages by providing the required expectation quantities. For example, the required expectation quantities related to argument z_i are contained in the sufficient statistics vector $\phi_i(z_i)$.

Appendix C

Derivation of Extended VMP

This appendix is based on the original work referenced below.

- Semih Akbayrak, Ivan Bocharov, and Bert de Vries, *Extended Variational Message Passing for Automated Approximate Bayesian Inference*, Special issue on Bayesian Inference in Probabilistic Graphical Models, Entropy, 2021

Here, we show the details of our solution approach that is based on importance sampling (IS) and Laplace approximation. First, we address the issues with deterministic mappings of random variables. The resulting technique emerges as a remedy for non-conjugate soft factor pairs problem as well.

C.1 Deterministic mappings with single inputs

We first address the issues with single-input deterministic mappings and generalize our solution to multiple inputs later on. Consider the sub-graph given in Fig. B.1, where the deterministic conditional $p(z_i|z_j)$ is defined as $f_c(z_i, z_j) = \delta(z_i - g(z_j))$. As derived in (B.22), we need the expectations $\mathbb{E}_{q(z_j)}[\phi_j(z_j)]$ and $\mathbb{E}_{q(z_i)}[\phi_i(z_i)]$ to calculate VMP messages towards variables in $\mathbf{z}_{a\setminus j}$ and $\mathbf{z}_{b\setminus j}$, respectively. Suppose that $\Phi(\cdot)$ is an element in the sufficient statistic vectors $\phi_j(\cdot)$ and $\phi_i(\cdot)$. Then we need to be able to calculate $\mathbb{E}_{q(z_i)}[\Phi(z_i)]$ and $\mathbb{E}_{q(z_j)}[\Phi(z_j)]$. Let us start with evaluating $\mathbb{E}_{q(z_i)}[\Phi(z_i)]$ first:

$$\mathbb{E}_{q(z_i)}[\Phi(z_i)] = \int q(z_i)\Phi(z_i)\mathrm{d}z_i = \int q(z_j)\Phi(g(z_j))\mathrm{d}z_j.$$

The second equality in the above expression is due to the transformation of variables, i.e., $q(z_i) = q(z_j)|\frac{\mathrm{d}z_j}{\mathrm{d}z_i}|$ [157].

Substituting $q(z_j)$ with the message passing calculation as in (2.6) in the above integral yields

$$\mathbb{E}_{q(z_i)}[\Phi(z_i)] = \int \frac{m_{aj}(z_j)m_{cj}(z_j)}{\int m_{aj}(z_j)m_{cj}(z_j)\mathrm{d}z_j}\Phi(g(z_j))\mathrm{d}z_j, \tag{C.1}$$

where $m_{cj}(z_j) = m_{di}(g(z_j))$, as given in (B.21). Recall from Section 2.5 that the normalizer, $\int m_{aj}(z_j)m_{cj}(z_j)dz_j$, is often hard to calculate analytically.

C

We use importance sampling [19, 102] to approximate the integral in (C.1):

$$\begin{aligned}\mathbb{E}_{q(z_i)}[\Phi(z_i)] &= \int \frac{\frac{m_{aj}(z_j)m_{di}(g(z_j))\pi(z_j)}{\pi(z_j)}}{\int \frac{m_{aj}(z_j)m_{di}(g(z_j))\pi(z_j)}{\pi(z_j)}\mathrm{d}z_j}\Phi(g(z_j))\mathrm{d}z_j \\ &\approx \sum_{s=1}^{N} \underbrace{\left[\frac{\frac{\overrightarrow{m}_{z_j}(z_j^{(s)})m_{di}(g(z_j^{(s)}))}{\pi(z_j^{(s)})}}{\sum_{n=1}^{N}\frac{\overrightarrow{m}_{z_j}(z_j^{(n)})m_{di}(g(z_j^{(n)}))}{\pi(z_j^{(n)})}}\right]}_{w^{(s)}}\Phi(g(z_j^{(s)})),\end{aligned} \tag{C.2}$$

where $z_j^{(s)}$ for $s = 1, \ldots, N$ are drawn from the proposal distribution $\pi(z_j)$, i.e., $z_j^{(s)} \sim \pi(z_j)$. $g(z_j^{(s)})$ for $s = 1, \ldots, N$ are particles and their corresponding weights are denoted by $w^{(s)}$.

The design of a good proposal distribution has a critical role in IS. First, it is supposed to be an easy-to-sample distribution. Secondly, its support is required to be no smaller than the support of $m_{aj}(z_j)m_{di}(g(z_j))$ [19]. Lastly, the proposal distribution is desired to be a good representation of $q(z_j) = \frac{m_{aj}(z_j)m_{cj}(z_j)}{\int m_{aj}(z_j)m_{cj}(z_j)\mathrm{d}z_j}$ to attain a fast convergence [52]. In our automated design, $m_{aj}(z_j)$ constitutes the proposal distribution. Our choice is not optimal in a sense that information regarding the evidence is most often carried out by the backward message and it is not incorporated in our proposal design. However, $m_{aj}(z_j)$ satisfies the first two conditions since the messages are parameterized with standard distributions (easy-to-sample) and it has nonzero probability everywhere the posterior has, too. Substituting $\pi(z_j)$ with $m_{aj}(z_j)$ in (C.2) yields

$$\widehat{\mathbb{E}}_{q(z_i)}[\Phi(z_i)] = \sum_{s=1}^{N} \underbrace{\left[\frac{m_{di}(g(z_j^{(s)}))}{\sum_{n=1}^{N} m_{di}(g(z_j^{(n)}))}\right]}_{w^{(s)}}\Phi(g(z_j^{(s)})), \tag{C.3}$$

where $z_j^{(s)} \sim m_{aj}(z_j)$ for $i = 1, \ldots, N$ and $\widehat{\mathbb{E}}_{q(z_i)}[\Phi(z_i)]$ denotes our estimator for $\mathbb{E}_{q(z_i)}[\Phi(z_i)]$.

Let us summarize the procedure in (C.3) to define our first set of rules related to the deterministic nodes. (C.3) consists of samples that are drawn from $m_{aj}(z_j)$ and transformed through deterministic mapping $g(.)$. We cast this process as the forward message $m_{ci}(z_i)$ calculation. Once the samples are transformed, i.e., $g(z_j^{(s)})$, the weights $w^{(s)}$ are determined over $m_{di}(.)$. We interpret this process as the collision of the forward $m_{ci}(z_i)$ and the backward messages $m_{di}(z_i)$, and hence relate it to the posterior calculation. Setting $\Phi(g(z_j^{(s)}))$ to $\delta(z_i - g(z_j^{(s)}))$, our interpretation of message collision becomes obvious since $\Phi(g(z_j^{(s)})) := \delta(z_i - g(z_j^{(s)}))$ results in a Monte Carlo estimate for $q(z_i)$. As a result, we introduce our first set of rules related to deterministic nodes and the posterior approximation at the output edge of the deterministic node:

$$m_{ci}(z_i) \approx \left\{ \left(\frac{1}{N}, g(z_j^{(1)}) \right), \ldots, \left(\frac{1}{N}, g(z_j^{(N)}) \right) \right\}, \tag{C.4a}$$

$$q(z_i) \propto m_{ci}(z_i) \cdot m_{di}(z_i) \approx \left\{ \left(w_i^{(1)}, z_i^{(1)} \right), \ldots, \left(w_i^{(N)}, z_i^{(N)} \right) \right\}, \tag{C.4b}$$

$$\text{where } z_j^{(s)} \sim m_{aj}(z_j), z_i^{(s)} = g(z_j^{(s)}), w_i^{(s)} = \frac{m_{di}(z_i^{(s)})}{\sum_{n=1}^{N} m_{di}(z_i^{(s)})}. \tag{C.4c}$$

Here, we introduce the term *list of weighted samples* (LWS) to refer to the distributions that are represented by a set of samples and corresponding weights. Above, $m_{ci}(z_i)$ and $q(z_i)$ are represented by LWS distributions.

Now, we turn our attention to calculation of $q(z_j)$ and the expectation quantity $\mathbb{E}_{q(z_j)}[\Phi(z_j)]$. For this task we have two different strategies: if $m_{aj}(z_j)$ is a Gaussian message, we approximate $q(z_j)$ by Laplace approximation which is also automatable thanks to automatic differentiation and otherwise we follow the IS procedure introduced above. Let us go over them starting from the latter.

C.1.1 Non-Gaussian case

This time we are supposed to evaluate $\mathbb{E}_{q(z_j)}[\Phi(z_j)]$ so that the VMP messages towards $\mathbf{z}_{a\setminus j}$ can be computed. Notice that the procedure is exactly same with (C.2), except that the expectation quantity of interest, $\mathbb{E}_{q(z_j)}[\Phi(z_j)]$, does not involve the deterministic mapping $g(.)$, this time. Therefore, by using $m_{aj}(z_j)$ as the proposal distribution, we can estimate $\mathbb{E}_{q(z_j)}[\Phi(z_j)]$ as the following

$$\widehat{\mathbb{E}}_{q(z_j)}[\Phi(z_j)] = \sum_{s=1}^{N} \underbrace{\left[\frac{m_{di}(g(z_j^{(s)}))}{\sum_{n=1}^{N} m_{di}(g(z_j^{(n)}))} \right]}_{w^{(s)}} \Phi(z_j^{(s)}). \tag{C.5}$$

This gives us the second set of rules related to deterministic mappings. An element of this new set of rules is that the backward message is directly passed in probability distribution function (pdf) form:

$$m_{bj}(z_j) = m_{cj}(z_j) = m_{di}(g(z_j)). \tag{C.6}$$

Recall from Sec. B.1 that the messages used to carry standard EF distributions. Now, we make an exception and introduce $m_{bj}(z_j)$, which is no longer associated with any of the standard EF distributions. Nonetheless, $m_{bj}(z_j)$ takes an exponential form since $m_{di}(\cdot)$ is an EF distribution (see (B.21)). Therefore, we call $m_{bj}(z_j)$ a *non-standard exponential family* (NEF) distribution. Having defined the backward message, let us evaluate the posterior $q(z_j)$. Similar to $q(z_i)$, substituting $\Phi(z_j)$ with $\delta(z_j - z_j^{(i)})$ in (C.5) gives us a Monte Carlo estimate of $q(z_j)$:

$$q(z_j) \propto m_{aj}(z_j) \cdot m_{bj}(z_j) \approx \left\{ \left(w_{z_j}^{(1)}, z_j^{(1)} \right), \ldots, \left(w_{z_j}^{(N)}, z_j^{(N)} \right) \right\}, \tag{C.7a}$$

$$\text{where } z_j^{(s)} \sim m_{aj}(z_j), w_j^{(s)} = \frac{m_{bj}(z_j^{(s)})}{\sum_{n=1}^{N} m_{bj}(z_j^{(n)})}. \tag{C.7b}$$

C.1.2 Gaussian case

In FFGs, the models are often constructed in such a way that the most prevailing message types will be Gaussians. This is because Gaussian messages facilitate inference by allowing many inference related operations to be executed in closed-form such as summation, conditioning, scaling and shifting by constants, etc. In order to retain the computational advantages of Gaussian distribution, we take it as an implicit hint that the posterior distribution is *Gaussian-like*, if $m_{aj}(z_j)$ is a Gaussian message. Then, we use Laplace approximation [75, Section 4.4] to approximate $q(z_j)$ with $\mathcal{N}(z_j; \mu_j, V_j)$ where

$$\mu_j = \arg\max_{z_j} \left(\log m_{aj}(z_j) + \log m_{bj}(z_j) \right), \tag{C.8a}$$

$$V_j = (-\nabla^2_{\mu_j} (\log m_{aj}(\mu_j) + \log m_{bj}(\mu_j))^{-1}, \tag{C.8b}$$

where $\nabla_{z_j} f$ denotes the gradient of f with respect to z_j and $\nabla^2_{\mu_j} f(\mu_j)$ refers to the Hessian of $f(\mu_j)$ with respect to μ_j and evaluated at μ_j. Note that the gradient and the Hessian respectively reduce to the first and the second derivatives if z_j is scalar. Laplace approximation is a mode-seeking algorithm. We use automatic differentiation (autodiff) [41] to evaluate the gradient $\nabla_{z_j} (\log m_{aj}(z_j) + \log m_{bj}(z_j))$ and employ it in a gradient-ascent algorithm to seek the mode (we supply the implementation details in Appendix E). Once the mode is reached, we evaluate the Hessian at the mode to fit the variance term for our Gaussian approximation.

The assumption we make here that $m_{aj}(z_j)$ implies a *Gaussian-like* $q(z_j)$ paves the way of automating many well known inference procedures achieved through Laplace approximation, such as Bayesian logistic regression [121, Section 8.4], Laplace-Gaussian filtering and smoothing in state space models [172], Poisson Linear Dynamical Systems [173], etc. However, our assumption would not be appropriate for all configurations. For example, Gaussian prior on rate parameter of a Poisson distribution would result in an ambiguous posterior since the domain of the rate is the positive real numbers while the Gaussian approximated posterior has a support on the entire real axis. A better model specification could be achieved by mapping a Gaussian distributed random variable to the rate parameter through an inverse-link function, $\exp$ in this example. Likewise, a multi-modal backward message $m_{bj}(z_j)$ with a support on real numbers often yields a multi-modal posterior which can be better captured with particle methods. (In Appendix F, we shall show that it is possible to run particle filtering through Gaussian factor nodes in our technique.)

In summary, our method resorts to Laplace approximation to approximate $q(z_j)$ with a Gaussian distribution whenever $m_{aj}(z_j)$ is Gaussian. Therefore, the user of our method must keep in mind the consequences of prior choices and build her model accordingly.

The overall procedure for single input deterministic functions is depicted in Figure. C.1. In the next subsection, we extend this procedure to multiple input deterministic mappings.

C.2 Deterministic mappings with multiple inputs

Consider the deterministic node, $f_c(\mathbf{z}_c) = \delta(z_i - g(\mathbf{z}_{c\setminus i}))$, given in Fig. 3.1 where the inputs to the deterministic function $g(.)$ are $\mathbf{z}_{c\setminus i}$ and the output is z_i.

Before starting the discussion on the backward messages, let us define the forward message $m_{ci}(z_i)$. Analogous to the single input case, we define $m_{ci}(z_i)$ with an LWS as the following

$$m_{ci}(z_i) \approx \left\{ \left(\frac{1}{N}, g(\mathbf{z}_{c\setminus i}^{(1)}) \right), \ldots, \left(\frac{1}{N}, g(\mathbf{z}_{c\setminus i}^{(N)}) \right) \right\}, \tag{C.9a}$$

$$\text{where } \mathbf{z}_{c\setminus i}^{(s)} = \bigcup_{\substack{k\in\mathcal{E}(c)\\ k\neq i}} z_k^{(s)} \text{ for } z_k^{(s)} \sim m_{kc}(z_k).$$

Once the message is calculated as a set of equally weighted samples, we scale the weights according to the importance score of their corresponding samples to repre-

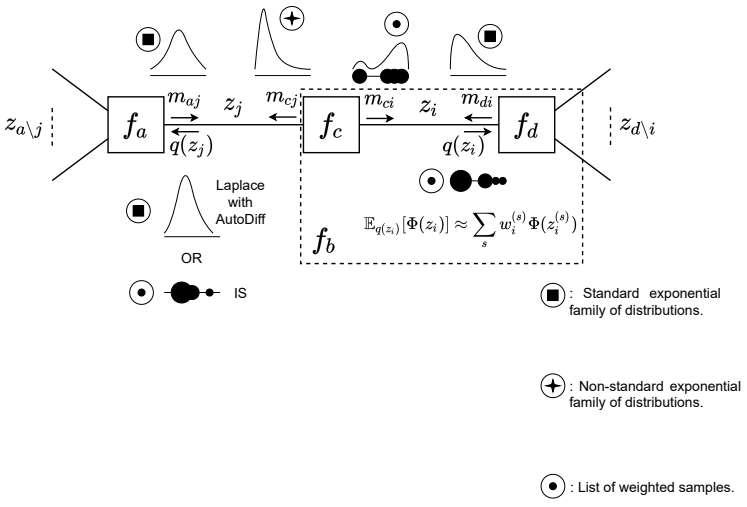


Figure C.1: Messages around a deterministic node f_c are visualized together with posterior approximations. In EVMP algorithm, forward messages from single input deterministic nodes are approximated by LWS representations. Backward messages, on the other hand, take non-standard exponential family distribution forms. Note that the message $m_{bj}(z_j)$ refers to $m_{cj}(z_j)$.

sent $q(z_i)$:

$$q(z_i) \propto m_{ci}(z_i) \cdot m_{ic}(z_i) \approx \left\{ \left(w_i^{(1)}, z_i^{(1)} \right), \ldots, \left(w_i^{(N)}, z_i^{(N)} \right) \right\}, \tag{C.10a}$$

$$\text{where } z_i^{(s)} = g(\mathbf{z}_{c\setminus i}^{(s)}), w_i^{(s)} = \frac{m_{ic}(z_i^{(s)})}{\sum_{n=1}^{N} m_{ic}(z_i^{(n)})}. \tag{C.10b}$$

Now, let us define the backward messages propagated by the deterministic node. The exact backward message towards one of the input variables, say z_j, is

$$m_{cj}(z_j) = \int \delta(z_i - g(\mathbf{z}_{c\setminus i})) \prod_{\substack{k \in \mathcal{E}(c) \\ k \neq j}} m_{kc}(z_k) \, \mathrm{d}\mathbf{z}_{c\setminus j}. \tag{C.11}$$

Unfortunately, the above integral is often intractable. Even if all the variables are discrete and integral is replaced by summation, it becomes intractable in practice

as the number of variables increases. Here, we address this issue with two different approximation strategies. As it is in the above subsection, type of the approximation depends on the incoming messages to the deterministic node from the input edges: if the messages $m_{kc}(z_k)$ for $k \in \mathcal{E}(c), k \neq i$ are all Gaussian, we approximate the joint posterior distribution of $\mathbf{z}_{c\setminus i}$ by a Gaussian distribution. Then, we calculate the backward messages over the approximated joint posterior and incoming messages. Otherwise, we use Monte Carlo summation. Let us start with the latter case.

C.2.1 Monte Carlo approximation to the backward message

Monte Carlo approximation to the the integral in (C.11) is

$$m_{cj}(z_j) \approx \frac{1}{N} \sum_{s=1}^{N} m_{ic}(g(\mathbf{z}_{c\setminus\{j,i\}}^{(s)}, z_j)), \tag{C.12}$$

where $\mathbf{z}_{c\setminus\{j,i\}}^{(s)} = \bigcup_{\substack{k\in\mathcal{E}(c)\\k\neq j\\k\neq i}} z_k^{(s)}$ for $z_k^{(s)} \sim m_{kc}(z_k)$.

Once the message $m_{cj}(z_j)$ is approximately calculated and propagated as an NEF distribution, $q(z_j)$ is also approximated either by IS or Laplace, depending on the message type $m_{jc}(z_j)$ as it is discussed in C.1.

C.2.2 Gaussian approximation to the backward message

The above procedure yields two consecutive approximation processes in the calculation of $q(z_j)$. Considering that we assumed $m_{jc}(z_j)$ implies that $q(z_j)$ is *Gaussian-like*, we can avoid the approximation in (C.12) if all the incoming messages $m_{kc}(z_k)$ for $k \in \mathcal{E}(c), k \neq i$ are Gaussian. We achieve this by approximating the joint posterior $q(\mathbf{z}_{c\setminus i})$ with Laplace, followed by a marginalization to evaluate $q(z_j)$ and $m_{cj}(z_j) \propto q(z_j)/m_{jc}(z_j)$.

More precisely, consider the incoming messages $m_{kc}(z_k) = \mathcal{N}(z_k; \mu_{kc}, V_{kc})$ for $k \in \mathcal{E}(c), k \neq i$. Note that these messages carry prior beliefs on random variables in $\mathbf{z}_{c\setminus i}$, which can be represented with a joint belief $m_{\mathcal{E}(c)\setminus i,c}(\mathbf{z}_{c\setminus i})$ constituted by concatenation of the random variables in $\mathbf{z}_{c\setminus i} : \bigoplus_{\substack{k\in\mathcal{E}(c)\\k\neq i}} z_k$: such that $\boldsymbol{\mu}_{\mathcal{E}(c)\setminus i,c} : \bigoplus_{\substack{k\in\mathcal{E}(c)\\k\neq i}} \mu_{kc}$ and $\boldsymbol{V}_{\mathcal{E}(c)\setminus i,c} : \bigoplus_{\substack{k\in\mathcal{E}(c)\\k\neq i}} V_{kc}$, where $\boldsymbol{V}_{\mathcal{E}(c)\setminus i,c}$ is a block-diagonal matrix and

$$m_{\mathcal{E}(c)\setminus i,c}(\mathbf{z}_{c\setminus i}) = \mathcal{N}(\boldsymbol{\mu}_{\mathcal{E}(c)\setminus i,c}, \boldsymbol{V}_{\mathcal{E}(c)\setminus i,c}). \tag{C.13}$$

Now, we approximate $q(\mathbf{z}_{c\setminus i})$ by a Gaussian distribution $\mathcal{N}(\mu_{c\setminus i}, V_{c\setminus i})$ with a Laplace approximation:

$$\mu_{c\setminus i} = \underset{\mathbf{z}_{c\setminus i}}{\operatorname{argmax}} \log m_{\mathcal{E}(c)\setminus i,c}(\mathbf{z}_{c\setminus i}) + \log m_{ic}(g(\mathbf{z}_{c\setminus i})), \tag{C.14a}$$

$$V_{c\setminus i} = (-\nabla^2_{\mu_{c\setminus i}}(\log m_{\mathcal{E}(c)\setminus i,c}(\mu_{c\setminus i}) + \log m_{ic}(g(\mu_{c\setminus i}))))^{-1}. \tag{C.14b}$$

C

By marginalizing out $\mathbf{z}_{c\setminus\{i,j\}}$, we find $q(z_j)$:

$$q(z_j) = \int q(\mathbf{z}_{c\setminus i})\, \mathrm{d}\mathbf{z}_{c\setminus\{i,j\}} = \mathcal{N}(\mu_j, V_j). \tag{C.15}$$

Recall that $q(z_j) \propto m_{jc}(z_j) m_{cj}(z_j)$. This yields the following backward message

$$m_{cj}(z_j) \propto q(z_j)/m_{jc}(z_j) = \mathcal{N}(\mu_{cj}, V_{cj}), \tag{C.16}$$

where $V_{cj}^{-1} = V_j^{-1} - V_{jc}^{-1}$ and $V_{cj}^{-1}\mu_{cj} = V_j^{-1}\mu_j - V_{jc}^{-1}\mu_{jc}$. Note that we intentionally parameterize the Gaussian backward message $m_{cj}(z_j)$ with a precision-weighted mean $V_{cj}^{-1}\mu_{cj}$ and precision V_{cj}^{-1}. The canonical parameterization (weighted-mean and precision) brings computational advantages, especially in state space models, by avoiding certain matrix inversions [57]. The approach that we introduced in this section resembles Expectation Propagation (EP) [67, 88] in the sense that we first find the posterior, $q(z_j)$, and then the backward message is evaluated by dividing the posterior to the incoming message. As it is stated in [88], Laplace Propagation [174] proposes an iterative Laplace approximation approach to mitigate intractable integral issues that sometimes emerge in EP.

So far, we've discussed how to extend VMP to those models with deterministic conditional distributions. To summarize, the resulting technique approximates the forward messages in deterministic nodes by LWS. Backward messages, on the other hand, are either directly propagated in NEF form or approximated with Gaussian distributions. We also showed posterior approximations related to these message types. In the next subsection, we shall attack the problem regarding the non-conjugate soft factor pairs.

C.2.3 Non-conjugate Soft Factor Pairs

Next we address the non-conjugate soft factor pairs problem. Consider the generic edge depicted in Figure 2.1. Suppose that the messages $m_{aj}(z_j)$ and $m_{bj}(z_j)$ differ in sufficient statistics that causes the normalization constant $\int m_{aj}(z_j)$ and $m_{bj}(z_j)\,\mathrm{d}z_j$ to be analytically intractable. Recall that the very much same problem emerges in C.1 while calculating $q(z_j)$. Therefore, the approximation rules defined in C.1 applies to non-conjugate factor pairs, as well. For the sake of comprehensiveness, the rules are summarized below.

1. If $m_{aj}(z_j)$ is a Gaussian message, apply Laplace to approximate $q(z_j)$ with a Gaussian distribution as in (C.8).
2. Otherwise, use IS as in (C.7).

C

Appendix D

Free Energy Approximation in EVMP

This appendix is based on the original work referenced below. An additional derivation for the joint entropy estimation is provided.

- Semih Akbayrak, Ivan Bocharov, and Bert de Vries, *Extended Variational Message Passing for Automated Approximate Bayesian Inference*, Special issue on Bayesian Inference in Probabilistic Graphical Models, Entropy, 2021

Recall from Section 2.3 that variational inference transforms a difficult inference task to an easier optimization problem of a variational bound called the free energy $\mathcal{F}$. Considering the fact that VMP converges to a stationary point by updating one posterior factor at a time, we anticipate that our approximations approach near local optima.

As it is shown in Section B.1.2, the free energy is amenable to analytical calculations for those models that are solely comprised of conjugate factor pairs. The models that we address here do not allow the free energy to be calculated analytically. This is because analytically intractable expectation quantities, which complicates VMP in practice, also appear in the free energy calculation. Therefore, we provide an approximate free energy to the user so that she can track the convergence of the inference and also make model comparison [65, 175].

We introduce our free energy approximation approach over the sub-graph given in Figure B.1, where $f_a(\mathbf{z}_a)$ is a standard EF distribution (B.1) and $f_b(\mathbf{z}_b)$ is a composite node, i.e., $f_b(\mathbf{z}_b) = \int \delta(z_i - g(z_j)) f_d(\mathbf{z}_d)\, \mathrm{d}z_i = f_d(g(z_j), \mathbf{z}_{d\setminus i})$. Recall from (B.6) that $f_d(z_i, \mathbf{z}_{d\setminus i})$ is modified as

$$f_d(\mathbf{z}_d) = h_k(z_k) \exp(\phi_i(z_i)^\intercal \lambda_{di}(\mathbf{z}_{d\setminus i}) + c_{di}(\mathbf{z}_{d\setminus i})).$$

This sub-graph is a part of a larger FFG. First, we decompose the free energy as the following

$$\mathcal{F} = \mathcal{F}_{\setminus j} \underbrace{\underbrace{-\mathbb{E}_{q(\mathbf{z}_a)}[\log f_a(\mathbf{z}_a)] - \mathbb{E}_{q(\mathbf{z}_b)}[\log f_b(\mathbf{z}_b)]}_{\text{Average energy terms}} + \underbrace{\mathbb{E}_{q(z_j)}[\log q(z_j)]}_{\substack{\text{Negative} \\ \text{entropy}}}}_{\mathcal{F}_j}, \tag{D.1}$$

where $\mathcal{F}_{\setminus j}$ stands for the free energy terms that are not subject to variables z_j. Explicitly writing the average energy terms:

$$-\mathbb{E}_{q(\mathbf{z}_a)}[\log f_a(\mathbf{z}_a)] = -\mathbb{E}_{q(z_j)}[\log(h_j(z_j))] - \mathbb{E}_{q(z_j)}[\phi_j(z_j)]^\intercal \mathbb{E}_{q(\mathbf{z}_{a\setminus j})}[\lambda_{aj}(\mathbf{z}_{a\setminus j})] + \mathbb{E}_{q(\mathbf{z}_{a\setminus j})}[\log Z_j(\mathbf{z}_{a\setminus j})] \tag{D.2a}$$

$$\begin{aligned} -\mathbb{E}_{q(\mathbf{z}_b)}[\log f_b(\mathbf{z}_b)] &= -\mathbb{E}_{q(\mathbf{z}_b)}\Big[\log\Big(\int \delta(z_i - g(z_j)) \\ &\qquad h_k(z_k)\exp\left(\phi_i(z_i)^\intercal \lambda_{di}(\mathbf{z}_{d\setminus i}) + c_{di}(\mathbf{z}_{d\setminus i})\right)\mathrm{d}z_i\Big)\Big] \\ &= -\mathbb{E}_{q(z_k)}[\log(h_k(z_k))] - \mathbb{E}_{q(z_i)}[\phi_i(z_i)]^\intercal \mathbb{E}_{q(\mathbf{z}_{d\setminus i})}[\lambda_{di}(\mathbf{z}_{d\setminus i})] \\ &\quad + \mathbb{E}_{q(\mathbf{z}_{d\setminus i})}[c_{di}(\mathbf{z}_{d\setminus i})]. \end{aligned} \tag{D.2b}$$

The above derivations closely follow the derivations in (B.22). Note that the expectation terms regarding z_j in (D.2b) are substituted by the expectations related to z_i, which are contained in the sufficient statistics vector $\phi_i(z_i)$. This quantities are exactly same with the ones required to calculate VMP messages towards $\mathbf{z}_{b\setminus j}$, and we used IS to estimate them in (C.3). Therefore, $\mathbb{E}_{q(z_i)}[\phi_i(z_i)]$ for the estimation of $-\mathbb{E}_{q(\mathbf{z}_b)}[\log f_b(\mathbf{z}_b)]$ is readily available.

Next, we investigate the terms related to z_j in $-\mathbb{E}_{q(\mathbf{z}_a)}[\log f_a(\mathbf{z}_a)]$. Recall that for $q(z_j)$, we have two approximation methods: 1) a Gaussian approximation to $q(z_j)$ with Laplace, 2) an LWS approximation. $q(z_j)$ is approximated with a Gaussian when $m_{aj}(z_j)$ is a Gaussian and this is the case if the factor node $f_a(\mathbf{z}_a) = p(z_j|\mathbf{z}_{a\setminus j})$ is a Gaussian distribution. In this case, $\phi_j(z_j) = [z_j, z_j^2]^\intercal$ ($\phi_j(z_j) = [z_j, z_j z_j^\intercal]^\intercal$ for a multivariate Gaussian), $\log(h_j(z_j)) = -0.5\log(2\pi)$ ($\log(h_j(z_j)) = -0.5d\log(2\pi)$ for a d-dimensional multivariate Gaussian), and $\mathbb{E}_{q(z_j)}[\log(h_j(z_j))]$, $\mathbb{E}_{q(z_j)}[\phi_j(z_j)]$ are available in closed-form. Similarly, the entropy term $-\mathbb{E}_{q(z_j)}[\log q(z_j)]$ is available in closed-form for a Gaussian $q(z_j)$. This completes the calculation of the expectation terms with z_j in $\mathcal{F}_j$.

In case $q(z_j)$ is approximated with LWS as in (C.7), we approximate $\mathbb{E}_{q(z_j)}[\log(h_j(z_j))]$ and $\mathbb{E}_{q(z_j)}[\phi_j(z_j)]$ with IS as in (C.5). Therefore, the approximations for the average energy terms are straightforward. For LWS approximated $q(z_j)$, the main difficulty in the estimation of $\mathcal{F}_j$ stems from the entropy calculation. This is because $\log(q(z_j))$ does not persist in functional form. The entropy approximation for

weighted sample approximated distributions is often carried out by probability density estimates on weighted samples [176]. Fortunately, in our case, we do not need to fit a density estimate on LWS since the messages $m_{aj}(z_j)$ and $m_{bj}(z_j)$ afford the information regarding the density $q(z_j)$. Let us derive an estimator for the entropy $\mathcal{H}_j$:

$$\begin{aligned}\mathcal{H}_j &= -\int q(z_j)\log q(z_j)dz_j \\ &= -\int q(z_j)\log\left(\frac{m_{aj}(z_j)m_{bj}(z_j)}{\int m_{aj}(z_j)m_{bj}(z_j)\mathrm{d}z_j}\right)\mathrm{d}z_j \\ &= \underbrace{-\int q(z_j)\log\left(m_{aj}(z_j)m_{bj}(z_j)\right)\mathrm{d}z_j}_{H_j^1} + \underbrace{\int q(z_j)\log\left(\int m_{aj}(z_j)m_{bj}(z_j)\mathrm{d}z_j\right)\mathrm{d}z_j}_{H_j^2}.\end{aligned} \tag{D.3}$$

We estimate the first term with the following Monte Carlo summation:

$$\hat{\mathcal{H}}_j^1 = -\sum_{s=1}^{N} w_j^{(s)}\log\left(m_{aj}(z_j^{(s)})m_{bj}(z_j^{(s)})\right). \tag{D.4}$$

The term with the $\log$ in $\mathcal{H}_j^2$ is constant since z_j is integrated out inside the $\log$. Therefore $\mathcal{H}_j^2$ simplifies further:

$$\mathcal{H}_j^2 = \log\left(\int m_{aj}(z_j)m_{bj}(z_j)\mathrm{d}z_j\right)\underbrace{\int q(z_j)\mathrm{d}z_j}_{1}. \tag{D.5}$$

Recall from (C.5) that the samples $z_j^{(1)},\ldots,z_j^{(N)}$ are drawn from the message $m_{aj}(z_j)$. Therefore, Monte Carlo estimate of H_j^2 is

$$\hat{\mathcal{H}}_j^2 = \log\left(\frac{1}{N}\sum_{s=1}^{N} m_{bj}(z_j^{(s)})\right). \tag{D.6}$$

This completes the estimation of the terms with z_j in $\mathcal{F}_j$.

Joint Entropy Estimation

Consider a deterministic node $f_c(\mathbf{z}_c)$ with multiple inputs:

$$f_c(\mathbf{z}_c) = \int \delta(z_i - g(\mathbf{z}_{c\setminus i})).$$

D

Note that in EVMP algorithm, the message passing rules we introduced around deterministic nodes with multiple inputs follow from BP, which posits a structured factorization for the approximate posterior $q(\mathbf{z}_c)$. Hence, we should be able to calculate joint entropy

$$\mathcal{H}_{c\setminus i} = -\int q(\mathbf{z}_{c\setminus i}) \log q(\mathbf{z}_{c\setminus i}) \mathrm{d}z_{c\setminus i} \tag{D.7}$$

to return the Free Energy [69], where $q(\mathbf{z}_{c\setminus i})$, up to a normalization, is defined as

$$\begin{aligned} q(\mathbf{z}_{c\setminus i}) &\propto \int q(\mathbf{z}_c) \mathrm{d}z_i \\ &\propto \int \delta(z_i - g(\mathbf{z}_{c\setminus i})) \prod_{k\in\mathcal{E}(c)} m_{kc}(z_k) \mathrm{d}z_i \\ &\propto m_{ic}(g(\mathbf{z}_{c\setminus i})) \prod_{\substack{k\in\mathcal{E}(c)\\ k\neq i}} m_{kc}(z_k). \end{aligned} \tag{D.8}$$

LWS approximation for $q(\mathbf{z}_{c\setminus i})$ is

$$\begin{aligned} &q(\mathbf{z}_{c\setminus i}) \approx \sum_{s=1}^{N} w_{c\setminus i}^{(s)} \mathbf{z}_{c\setminus i}^{(s)}, \\ &\text{where } \mathbf{z}_{c\setminus i}^{(s)} \sim \bigcup_{\substack{k\in\mathcal{E}(c)\\ k\neq i}} m_{kc}(z_k) \text{ and } w_{c\setminus i}^{(s)} \propto m_{ic}(g(\mathbf{z}_{c\setminus i}^{(s)})). \end{aligned} \tag{D.9}$$

Estimation of the joint distribution entropy is analogous to that of the marginal entropy

$$\hat{\mathcal{H}}_{c\setminus i} = \hat{\mathcal{H}}_{c\setminus i}^{1} + \hat{\mathcal{H}}_{c\setminus i}^{2}, \tag{D.10}$$

where

$$\hat{\mathcal{H}}_{c\setminus i}^{1} = -\sum_{s=1}^{N} w_{c\setminus i}^{(s)} \left[\log m_{ic}(g(\mathbf{z}_{c\setminus i}^{(s)})) - \sum_{\substack{k\in\mathcal{E}(c)\\ k\neq i}} \log m_{ck}(z_k^{(s)}) \right], \tag{D.11}$$

and

$$\hat{\mathcal{H}}_{c\setminus i}^{2} = \log \left(\frac{1}{N} \sum_{s=1}^{N} \log m_{ic}(g(\mathbf{z}_{c\setminus i}^{(s)})) \right). \tag{D.12}$$

Appendix E

Implementation Details for EVMP in ForneyLab

This appendix is based on the original work referenced below. The optimizer used for the Laplace approximation is replaced by an L-BFGS optimizer from the Julia package Optim.jl.

- Semih Akbayrak, Ivan Bocharov, and Bert de Vries, *Extended Variational Message Passing for Automated Approximate Bayesian Inference*, Special issue on Bayesian Inference in Probabilistic Graphical Models, Entropy, 2021

Our extensions to VMP are readily available in ForneyLab [62], which is a Julia package for message passing based probabilistic programming. In this section, we provide the reader with some of the core implementation details and automation process of the method. First, the number of particles that commutes through deterministic nodes is set to 1000 by default. The user can change the number of samples during model specification. Similarly, the posterior of the variables that are connected to non-conjugate soft factor pairs are approximated by 1000 samples. For Laplace approximations, gradients are automatically calculated by automatic differentiation tools of Julia language. We use the *ForwardDiff* package [158] since it is a mature, universal automatic differentiation tool that aligns well with the needs of our approach. Recall that Laplace approximation is a mode seeking algorithm. We automate the mode seeking by using L-BFGS [165,166] optimizer in the Julia-based optimization package Optim.jl [103].

E

Appendix F

Bootstrap Particle Filtering

This appendix is based on the original work referenced below.

- Semih Akbayrak, Ivan Bocharov, and Bert de Vries, *Extended Variational Message Passing for Automated Approximate Bayesian Inference*, Special issue on Bayesian Inference in Probabilistic Graphical Models, Entropy, 2021

Having implemented importance sampling to get around the complications in VMP, we now show how our technique inherently supports bootstrap particle filtering in state space models [19, 122].

Recall that we automate Laplace approximation to retain the computational convenience of Gaussian filtering and smoothing. Although this choice sounds reasonable for those cases we believe the distributions over hidden states possess unimodal behaviour, it would not be sufficient to capture multi-modal distributions [19]. Similarly, due to non-linearities in the model specification and/or non-Gaussian process noise, Gaussian distribution might not be a plausible representation of the hidden states. In these cases, Sequential Monte Carlo methods [177] could be appealing because they flexibly recover asymmetric and mixture distributions.

In VMP setting, our method employs samples and their corresponding weights to deploy VMP messages which are parameterized by exponential family distributions. Alternatively, in Belief Propagation (BP) setting, a soft factor collects samples to instantiate the conditional distributions and then draws samples from these conditionals. This process is depicted in Figure F.1 with two samples for the sake of easiness in visualization.

Having implemented the BP rule at a soft factor for incoming LWS messages, we have to show how posteriors are approximated through updating weights. Suppose that $m_{ak}(z_k)$ is a message carries LWS, and $m_{bk}(z_k)$ is parameterized either by an

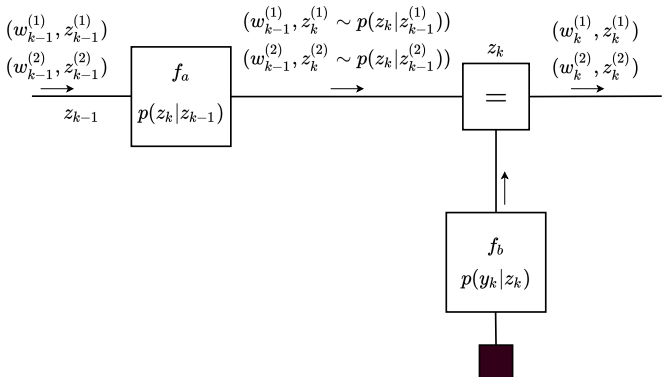


Figure F.1: Bootstrap Particle Filtering employs the state transition distributions, $p(z_k|z_{k-1})$ as proposal distributions, which can easily be supported in our framework by defining BP rules at soft factors for incoming messages that are LWS. The rule is straightforward to implement: weights stay unchanged; for each incoming sample, instantiate a new conditional distribution and draw a sample from it. The weight update is automatically carried out at equality node by $\tilde{w}_k^{(s)} \propto p(y_k|z_k^{(s)}) w_{k-1}^{(s)}$, which is followed by a normalization: $w_k^{(s)} = \frac{\tilde{w}_k^{(s)}}{\sum_{n=1} \tilde{w}_k^{(n)}}$.

EF or an NEF. Then, we define the posterior update rule as

$$\begin{aligned} \text{Given } m_{ak}(z_k) &= \left\{ \left(w_{k-1}^{(1)}, z_k^{(1)} \right), \ldots, \left(w_{k-1}^{(N)}, z_k^{(N)}) \right) \right\}, \\ q(z_k) &\propto m_{ak}(z_k) m_{bk}(z_k) \\ &\approx \left\{ \left(w_k^{(1)}, z_k^{(1)} \right), \ldots, \left(w_k^{(N)}, z_k^{(N)}) \right) \right\}, \qquad \text{(F.1a)} \\ \text{where } w_k^{(n)} &\propto w_{k-1}^{(n)} m_{bk}(z_k^{(i)}). \qquad \text{(F.1b)} \end{aligned}$$

In Bootstrap particle filtering, these rules update the weights at equality nodes automatically. A major drawback of sequential importance sampling methods is that the further samples commute over time steps, the more they lose their ability to recover the underlying process, and many of the weights approach to zero. This phenomenon is known as the degeneracy problem and can sometimes be alleviated by resampling [19, 177]. In our automated setting, at each weight update step, we measure the effectiveness of the existing samples by $n_{\text{eff}} = \frac{1}{\sum_{n=1}^{N} \left(w^{(n)} \right)^2}$, as it is shown in [19]. Then, we resample if $n_{\text{eff}} < N/10$ [19]. A user can effortlessly execute

a particle filtering procedure in our method by putting an LWS prior on the first hidden state of a sequential model and running BP inference on the model [1].

[1] For demonstration purposes, we implemented BP rules at Gaussian node for LWS messages. The user can implement the very same rules for other soft factors according to her needs. Visit `https://github.com/semihakbayrak/ForneyLab.jl/blob/dev/demo/bootstrap_particle_filter.ipynb` for a toy example.

Acknowledgments

Five years ago, I started working in a small fintech company with the hope of becoming a millionaire by developing adaptive, intelligent agents for algorithmic trading in stock markets. It took me a couple of months to realize that the task was way more challenging than I initially thought. So, I made up my mind to pursue a research career to study adaptive agents and probabilistic inference methods in more detail. Luckily, I found a great supervisor who was as enthusiastic as me about intelligent agents. It was not just the intelligent agents themselves that he was interested in but also the design processes of the intelligent agents. **Bert**, you are a great visioner and a leader who brought all these talented people together in pursuit of an ultimate goal: revolutionizing the design processes of intelligent agents. I am so grateful for getting to know you and being a member of this great team you created. I believe I've become a better researcher in these four years, thanks to you and your immense support. Whenever I needed help, I felt your full support. I was lucky to have you as a supervisor with whom I could talk not only about the research and the projects but also about life in general. You are and will always be one of the most influential figures in my life.

I would like to express how honored I am to have a chance to present my dissertation to the members of the defense committee **Cassio P. de Campos, Francesco A. N. Palmieri, Philipp Rostalski, Simo Särkkä** and **Thijs W. van de Laar**. I want to thank them for their time and valuable feedback. I especially thank **Thijs** for all his help during my PhD. **Thijs**, I must admit that ForneyLab is still as fascinating as it was when I first saw it in action. It seems like only yesterday when you organized a contest to help us better understand the internals of ForneyLab. Thank you and **Marco** for creating such an inspiring tool, which now constitutes the skeleton of this dissertation. BIASlab is lucky to have two brilliant assistant professors: **Thijs** and **Wouter**. **Wouter**, I can not forget your support when I needed to be in Turkey to help my family. I know you will be an excellent supervisor.

I offer my gratitude to **Jan** and the SPS group at Eindhoven University of Technology for creating a unique research environment. Many thanks to the secretariat of the SPS group, **Anja**, **Carla**, **Emerald**, and **Judith**. I also sincerely thank GN Advanced Sciences for their financial support for this research.

During my PhD years, I've met numerous talented researchers and had a chance

to make new friends. **Bart**, you are a very smart and productive researcher. I wish we had more chances to collaborate. **Dmitry**, you are definitely among the top programmers I've met in my life. RxInfer not only circumvented the issues we had in ForneyLab, but also introduced an entirely new probabilistic programming paradigm. The other excellent programmer who found himself a place in Semih's list of distinguished programmers is **Ivan**. **Ivan**, you are an excellent collaborator, but more importantly, a great person. I wouldn't be able to survive the EVMP journey without you. I also want to thank **Alp, Anouk, Burak, Chengfeng, Hoang, Jim, Joris, Mykola, Patrick, Philip, Sepideh, Thijn, Tim** and **Tjalling**, with whom I had the honor to share my time in BIASlab. The list would be incomplete if I did not mention and thank **Gonenc, Nehir Berk** and **Tunc**.

I owe a special thanks to **Albert, Ismail, Magnus, Martin** and **Yunus** for joining my wedding with **Melike**. Your presence made the bachelor party and the wedding way more fun. **Martin**, I want to thank you for helping me to get used to the city and the country. I always enjoyed it more when you joined the activities with us. **Magnus**, you are a very talented and courageous researcher. I know your thoughts have the potential to revolutionize the entire AI field. I am glad to have you as a friend.

Ismail, you are a fantastic mathematician and a great research partner. I am confident that an exciting research career full of achievements is awaiting you. During our trip to Finland, I realized one more time how lucky I am to have friends like you. **Albert**, I feel so privileged to have you in my life. Whether it is a bug in the code or an undesired condition complicating life, your unique way of approaching problems has always amazed me. I very much enjoyed all the memories we collected together. Thank you again for always being there to cheer me up whenever I was about to ruin my psychology with pessimist thoughts.

My dearest family, there is no word invented yet to express my appreciation for all your love and sacrifices. Mom and dad, it is so relieving to feel your unconditional support that also allowed me to take the step and start my PhD. My beloved sister **Seda**, the last couple of years were exciting and sometimes challenging for you. Unfortunately, I couldn't be there to share your excitement and support you in the challenging times, but you know that my thoughts were always with you. My lovely wife **Melike**, I can imagine that the first few years are challenging for all young couples, but ours was quite a journey. My PhD journey aside, we got married, and you started a new career in the Netherlands. Our endless love gave us the energy to keep going and turned our tiny home into the best place in the world. I can't thank you enough for all your patience and sacrifices. Last but not least, you are always in my heart, grandma.

Semih Akbayrak

Curriculum Vitae

Semih Akbayrak was born on July 20th, 1992 in Istanbul, Turkey. He received his B.Sc. and M.Sc. from Bogazici University, Istanbul, in Electrical & Electronics Engineering and Computational Science & Engineering, respectively, in 2015 and 2018. He is also a proud graduate of Ataturk High School of Science in Istanbul. In September 2018, Semih joined Bayesian Intelligent Autonomous Systems Lab (BIASlab) at Eindhoven University of Technology as a Ph.D. candidate under Prof. Bert de Vries' supervision. Since then, his research has focused mainly on automated hybrid Bayesian inference by means of message passing on factor graphs. Among Semih's current academic interests are Bayesian statistics, relational learning, time series analysis, and decision making under uncertainty.

List of Publications

Journal Articles:

- **Semih Akbayrak**, İsmail Şenöz, Alp Sarı and Bert de Vries, *Probabilistic Programming with Stochastic Variational Message Passing*, International Journal of Approximate Reasoning, 2022
- **Semih Akbayrak**, Ivan Bocharov, and Bert de Vries, *Extended Variational Message Passing for Automated Approximate Bayesian Inference*, Special issue on Bayesian Inference in Probabilistic Graphical Models, Entropy, 2021

Conference Articles:

- **Semih Akbayrak**, İsmail Şenöz, Bert de Vries, *Adaptive Importance Sampling Message Passing*, 2022 IEEE International Symposium on Information Theory (ISIT 2022) - Proceedings
- **Semih Akbayrak**, Bert de Vries, *Reparameterization Gradient Message Passing*, The 27th European Signal Processing Conference (EUSIPCO 2019) - Proceedings

Not included in the dissertation:

- Albert Podusenko, **Semih Akbayrak**, İsmail Şenöz, Maarten Schoukens and Wouter M. Kouw, *Message-Passing-Based System Identification for NARMAX Models*, The 61th IEEE Conference on Decision and Control (CDC 2022) - Proceedings
- Hoang Nguyen, **Semih Akbayrak**, Magnus Koudahl, Bert de Vries, *Gaussian Process-based Amortization of Variational Message Passing Update Rules*, The 30th European Signal Processing Conference (EUSIPCO 2022) - Proceedings
- Alp Sarı, **Semih Akbayrak**, İsmail Şenöz, Bert de Vries, *Adaptive Optimizer Design for Constrained Variational Inference*, Symposium on Information Theory and Signal Processing in the Benelux (SITB 2022) - Proceedings
- İsmail Şenöz, Albert Podusenko, **Semih Akbayrak**, Christoph Mathys, Bert de Vries, *The Switching Hierarchical Gaussian Filter*, 2021 IEEE International Symposium on Information Theory (ISIT 2021) - Proceedings