International Journal of Approximate Reasoning 148 (2022) 235–252

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

Probabilistic programming with stochastic variational message passing

Semih Akbayrak [a,*], İsmail Şenöz [a], Alp Sarı [a], Bert de Vries [a,b]

[a] Department of Electrical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600MB Eindhoven, the Netherlands
[b] GN Hearing BV, JF Kennedylaan 2, 5612AB Eindhoven, the Netherlands

ARTICLE INFO

Article history:
Received 21 January 2022
Received in revised form 9 June 2022
Accepted 22 June 2022
Available online 30 June 2022

Keywords:
Factor graphs
Message passing
Natural gradient descent
Probabilistic programming
Variational inference

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 Languages (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 does not yet exist in most PPLs. In this paper, we cast SVI and CVI explicitly in a message passing-based inference context. 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.

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 [1]. A considerable amount of inference methods and tools have been developed over the past decade to support this endeavor. 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 [2,3]. These methods have been implemented in Probabilistic Programming Languages (PPLs) such as Turing.jl [4], Stan [5], Pyro [6] and TensorFlow Probability [7]. Realizing variational inference as a stochastic optimization task paves the way toward universal inference and scales well to large data sets [8]. Still, stochastic approximation methods for variational inference come with their own challenges. For example, Black-Box Variational Inference (BBVI) [9] often requires additional steps, such as Rao-Blackwellization [10], control variates [11], or variable reparameterization [12,13] to reduce the variance in noisy gradient estimates and to attain stable convergence. Another popular method, Automatic Differentiation Variational Inference (ADVI) [14] maps continuous random variables

* Corresponding author.
E-mail address: s.akbayrak@tue.nl (S. Akbayrak).

https://doi.org/10.1016/j.ijar.2022.06.006

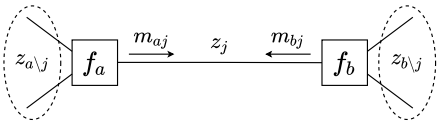


Fig. 1. A sub-graph $\mathcal{G}(\{a, b\}, j)$. The edge j is connected to factors a and b, which implies that z_j is an argument to functions f_a and f_b. We denote the message on edge j from f_a and f_b with $m_{aj}(z_j)$ and $m_{bj}(z_j)$, respectively.

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) [15,16], Expectation Propagation (EP) [17,18] and Variational Message Passing (VMP) [19,20].

In this paper, we focus on two other well-recognized stochastic approximation methods for scalable and universal variational inference, namely Stochastic Variational Inference (SVI) [21] and Conjugate-Computation Variational Inference (CVI) [22]. Unlike BBVI and ADVI, both SVI and CVI take advantage of conjugacy structures in the model specifications. They use natural gradient descent [23,24] to minimize a variational free energy objective in a stochastic setting. By incorporating 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 very 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 [25] and Julia language [26] packages ReactiveMP.jl [27] and ForneyLab.jl [28] 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. We shall discuss an alternative approach based on [29] in Section 5. 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 and show the favorable features of these new extensions by a number of experiments.

The paper is organized as follows. In Section 2 we review Forney-style Factor Graphs (FFGs), Belief Propagation (BP) and Variational Message Passing (VMP). We conclude Section 2 by introducing the problems that are addressed in this paper. Section 3 addresses these problems by transferring SVI and CVI methods to the FFG framework. In Section 4, we apply the proposed solutions to a variety of experiments that demonstrate the efficiency of these solutions. In Section 5, we provide a discussion on our implementation and future work and Section 6 summarizes with conclusions.

2. Background

This section provides a review on FFGs, BP, VMP, and the exponential family of distributions. We also introduce notational conventions and conclude the section by providing the limitations of probabilistic programming with message passing on FFGs.

2.1. Forney-style Factor Graphs (FFGs)

Given a factorized function $f(\boldsymbol{z}) = \prod_{a\in\mathcal{V}} f_a(\boldsymbol{z}_a)$ of a collection of random variables $\boldsymbol{z}$, where $\boldsymbol{z}_a$ stands for the subset of random variables that are arguments of f_a, a Forney-style Factor Graph (FFG) [30,31] visualizes the independency structure between the variables. 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. We sometimes index sub-graphs to differentiate them, e.g., $\mathcal{G}_p(\mathcal{V}_p, \mathcal{E}_p)$. In FFGs, random variables are branched out to more than two factor nodes through equality constraints. This is achieved by introducing an "equality" node $f_a(\boldsymbol{z}_a) = \delta(z_j - z_i)\delta(z_j - z_k)$ that generates the copies of z_j as z_i and z_k.

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

$$f(z_j) = \underbrace{\int f_a(\boldsymbol{z}_a)\mathrm{d}\boldsymbol{z}_{a\setminus j}}_{m_{aj}(z_j)} \underbrace{\int f_b(\boldsymbol{z}_b)\mathrm{d}\boldsymbol{z}_{b\setminus j}}_{m_{bj}(z_j)}. \tag{1}$$

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:

$$p(z_j) = \frac{f(z_j)}{\int f(z_j)\mathrm{d}z_j} = \frac{m_{aj}(z_j)m_{bj}(z_j)}{\int m_{aj}(z_j)m_{bj}(z_j)\mathrm{d}z_j}. \tag{2}$$

This exact inference procedure in tree-like FFGs is known as Belief Propagation (BP) [15,16].

2.2. Variational message passing

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

Consider a sub-graph $\mathcal{G}(b)$. As we will be working around the factor f_b without specifying the neighboring nodes, we denote the message propagating towards f_b from the other end of edge j with $m_{jb}(z_j)$. For example, in Fig. 1, the outgoing message $m_{aj}(z_j)$ from node a is referred to as $m_{jb}(z_j)$ when interpreted as an incoming message to node b. The joint distribution of $\boldsymbol{z}_b$ in the sub-graph $\mathcal{G}(b)$ under marginalization and normalization constraints is given by [32]

$$p(\boldsymbol{z}_b) = \frac{f(\boldsymbol{z}_b)}{\int f(\boldsymbol{z}_b)\mathrm{d}\boldsymbol{z}_b} \tag{3a}$$

$$\text{where } f(\boldsymbol{z}_b) = f_b(\boldsymbol{z}_b)\prod_{i\in\mathcal{E}(b)} m_{ib}(z_i). \tag{3b}$$

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

$$\mathcal{F}[q(\boldsymbol{z}_b)] = \mathbb{E}_{q(\boldsymbol{z}_b)}\left[\log\frac{q(\boldsymbol{z}_b)}{f(\boldsymbol{z}_b)}\right] \geq -\log\int f(\boldsymbol{z}_b)\mathrm{d}\boldsymbol{z}_b, \tag{4}$$

which is an upper bound to the negative log normalizer in (3a). In the presence of observations in probabilistic models, the bound is set to negative log-evidence and consequently, the free energy equals the negative Evidence Lower BOund (ELBO). Keeping only the terms with z_j in (4),

$$\mathcal{F} \propto \mathbb{E}_{q(z_j)}[\log q(z_j)] - \mathbb{E}_{q(z_j)}\big[\log m_{jb}(z_j)\big] - \mathbb{E}_{q(z_j)}\big[\mathbb{E}_{q(\boldsymbol{z}_{b\setminus j})}[\log f_b(\boldsymbol{z_b})]\big], \tag{5}$$

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

$$q(z_j)^* = \frac{m_{jb}(z_j)m_{bj}(z_j)}{\int m_{jb}(z_j)m_{bj}(z_j)\mathrm{d}z_j}, \tag{6}$$

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

$$m_{bj}(z_j) \propto \exp\left(\mathbb{E}_{q(\boldsymbol{z}_{b\setminus j})}[\log f_b(\boldsymbol{z}_b)]\right). \tag{7a}$$

If the messages $m_{jb}(z_j)$ and $m_{bj}(z_j)$ take the functional forms (with identical sufficient statistics)

$$m_{jb}(z_j) \propto \exp(\phi_j(z_j)^\intercal \eta_{jb}) \tag{8a}$$

$$m_{bj}(z_j) \propto \exp(\phi_j(z_j)^\intercal \eta_{bj}), \tag{8b}$$

then the factors in $\mathcal{V}(j)$ are called conjugate factor pairs [33, Chapter 2.4]. Conjugate factor pairs allow the approximate marginal $q(z_j)^*$ in (6) to be analytically evaluated in the exponential family of distributions [34]

$$q(z_j)^* = h_j(z_j)\exp(\phi_j(z_j)^\intercal \underbrace{(\eta_{jb}+\eta_{bj})}_{\eta_j} - A_j(\eta_j)). \tag{9}$$

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 parameters vector and $A_j(\eta_j)$ is the log-normalizer

$$A_j(\eta_j) = \log\left(\int h_j(z_j)\exp(\phi_j(z_j)^\intercal \eta_j)\mathrm{d}z_j\right). \tag{10}$$

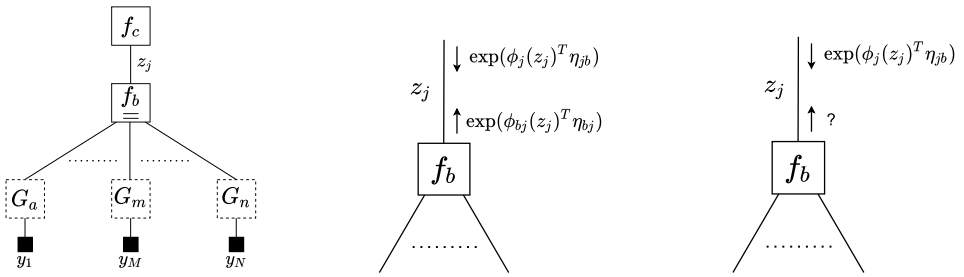


Fig. 2. Three possible issues a message passing-based PPL may encounter are visualized. On the left, the equality node requires VMP messages from all the dashed sub-graphs to calculate $m_{bj}(z_j)$, which might not be feasible for large N. In (b), $m_{bj}(z_j)$ and $m_{jb}(z_j)$ differ in sufficient statistics, which preclude analytical marginal calculations. On the right, 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.

2.3. Problem specification

By defining message calculation rules on node level and marginal distribution calculation rules on edges as previously described, a modular, automated message passing-based inference engine can be developed. In Fig. 2, we list the potential difficulties a message passing-based inference engine may encounter and how we attack them:

- **Scalability:** In principle, the VMP algorithm requires the entire data set to be processed at once, e.g., in Fig. 2a the equality node f_b needs to collect VMP messages from all the dashed sub-graphs to calculate the message $m_{bj}(z_j)$. Unfortunately, this 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. Stochastic Variational Inference (SVI) [21], among others in the literature [35,36], provides a principled method for scalable variational inference. In Section 3.1, we show that SVI is easy to implement in our message passing-based inference engine.
- **Non-conjugacy:** Non-conjugate factor pairs yield messages with different sufficient statistics, shown in Fig. 2b, which preclude analytical marginal calculations. In Section 3.2, we attack this problem with Conjugate-Computation Variational Inference (CVI) [22] and approximate $m_{bj}(z_j)$ with a message $\nu_{bj}(z_j)$ having sufficient statistics $\phi_j(z_j)$.
- **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. In Section 3.3, we provide a strategy based on Monte Carlo summation and CVI to approximate a message $m_{bj}(z_j)$ that is not available in closed form or missing in the inference engine.

In all three cases, we assume that the message $m_{jb}(z_j) = \exp(\phi_j(z_j)^\intercal \eta_{jb})$ is given and the problems are associated with $m_{bj}(z_j)$. We will preserve this convention in the next section.

3. Stochastic variational message passing with natural gradient descent

In this section, we address the above three problems depicted for a node f_b. We will use SVI and CVI that are both based on Natural Gradient Descent (NGD) [23,24] optimization of the free energy, otherwise known as the Bayesian Learning Rule [37],

$$\eta_j^{(t)} \longleftarrow \eta_j^{(t-1)} - \rho^{(t)} \nabla_{\eta_j}^N \mathcal{F}\left(\eta_j^{(t-1)}\right) \tag{11}$$

to tune the natural parameters of the approximate marginal

$$q(z_j; \eta_j) = h_j(z_j) \exp(\phi_j(z_j)^\intercal \eta_j - A_j(\eta_j)). \tag{12}$$

In (11), t is the iteration index in NGD, $\rho^{(t)}$ is a step size and $\nabla_{\eta_j}^N \mathcal{F}\left(\eta_j^{(t-1)}\right)$ 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:

$$\mathcal{F}(\eta_j) = \mathbb{E}_{q(z_j;\eta_j)}[\log q(z_j; \eta_j)] - \mathbb{E}_{q(z_j;\eta_j)}\big[\log m_{jb}(z_j)\big] - \mathbb{E}_{q(z_j;\eta_j)}\big[\log m_{bj}(z_j)\big] + c, \tag{13}$$

where c collects the terms independent of η_j. Assuming that

$$m_{jb}(z_j) \propto \exp(\phi_j(z_j)^\intercal \eta_{jb})\,, \tag{14}$$

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

$$\nabla^N_{\eta_j}\mathcal{F}(\eta_j) = \eta_j - \Big(\eta_{jb} + \underbrace{G^{-1}(\eta_j)\nabla_{\eta_j}\mathbb{E}_{q(z_j;\eta_j)}[\log m_{bj}(z_j)]}_{\nabla^N_{\eta_j}\mathbb{E}_{q(z_j;\eta_j)}[\log m_{bj}(z_j)]}\Big), \tag{15}$$

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

$$G(\eta_j) = \nabla^2_{\eta_j} A_j(\eta_j)\,. \tag{16}$$

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 2.3 to optimize the free energy in a stochastic manner by setting $\rho^{(t)}$ according to Robbins-Monro conditions [38], i.e., $\sum_{t=1}^{\infty}\rho^{(t)} = \infty$ and $\sum_{l=1}^{\infty}\rho^{(t)^2} < \infty$.

3.1. SVI for scalable VMP

Consider the FFG depicted in Fig. 2a, 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.,

$$m_{ib}(z_i) \propto \exp(\phi_j(z_i)^\intercal \eta_{ib}). \tag{17}$$

In the message passing interpretation of SVI [39], we work with $M < N$ sub-graphs at a VMP iteration by estimating the message $m_{bj}(z_j)$ from the equality node as

$$m_{bj}(z_j) \approx \left(\int \prod_{\substack{i\in\mathcal{E}'(b)\\ i\neq j}} \delta(z_j - z_i) m_{ib}(z_i)\,\mathrm{d}z_i\right)^{N/M} \propto \exp\left(\phi_j(z_j)^\intercal \frac{N}{M}\sum_{\substack{i\in\mathcal{E}'(b)\\ i\neq j}} \eta_{ib}\right). \tag{18}$$

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 (15), the natural gradient estimate of the free energy evaluates to

$$\tilde{\nabla}^N_{\eta_j}\mathcal{F}(\eta_j) = \eta_j - \left(\eta_{jb} + \frac{N}{M}\sum_{\substack{i\in\mathcal{E}'(b)\\ i\neq j}} \eta_{ib}\right). \tag{19}$$

For an FFG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, Algorithm 1 shows how SVI is executed by applying NGD around equality nodes $\mathcal{V}_= \subset \mathcal{V}$ that are associated with shared variables $\tilde{\boldsymbol{z}}$ such that $\tilde{\boldsymbol{z}} \subset \boldsymbol{z}$.

Algorithm 1 SVI on an FFG.

Require: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ for $f(\boldsymbol{z})$ such that $\tilde{\boldsymbol{z}} \subset \boldsymbol{z}$ 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 $z_j \in \tilde{\boldsymbol{z}}$ **do**
  Initialize $q(z_j) \propto \exp(\phi_j(z_j)^\intercal \eta_j^{(0)})$

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 2.2
    Calculate VMP messages towards b for all $b \in \mathcal{V}_=$
  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 (19) ▷ Given that $z_j \in \tilde{\boldsymbol{z}}$
    Set a step size $\rho^{(t)}$
    Update $q(z_j)$ using (11)

3.2. 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 Fig. 2b):

$$m_{jb}(z_j) \propto \exp(\phi_j(z_j)^\intercal \eta_{jb}) \tag{20a}$$

$$m_{bj}(z_j) \propto \exp(\phi_{bj}(z_j)^\intercal \eta_{bj}) \tag{20b}$$

Motivated by the message approximation scheme within the exponential family of distributions in [40], 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 (11) until convergence, and then find $\nu_{bj}(z_j)$ as described by [17,41],

$$\nu_{bj}(z_j) = \frac{q(z_j; \eta_j^*)}{m_{jb}(z_j)} \propto \exp(\phi_j(z_j)^\intercal (\eta_j^* - \eta_{jb})). \tag{21}$$

In the NGD-based optimization of the free energy, we employ an estimate for $\nabla_{\eta_j}^N \mathbb{E}_{q(z_j;\eta_j)}[\log m_{bj}(z_j)]$, 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 [22, Appendix B] for details, which follows from [42]. In our implementation, we stick to their computationally efficient approach for the Gaussian case. In other cases, we compute $G(\eta)$ with automatic differentiation [43] 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 [9]:

$$\tilde{\nabla}_{\eta_j} \mathbb{E}_{q(z_j;\eta_j)}[\log m_{bj}(z_j)] := \frac{1}{S} \sum_{s=1}^{S} \nabla_{\eta_j} \log q\left(z_j^{(s)}; \eta_j\right) \log m_{bj}\left(z_j^{(s)}\right), \text{ where } z_j^{(s)} \sim q(z_j; \eta_j)\,. \tag{22}$$

In Section 4.4, we will demonstrate that approximate messages $\nu_{bj}(z_j)$ ease hybrid inference procedures in message passing-based PPLs.

3.3. 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 Fig. 2c. 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

$$f_b(\boldsymbol{z}_b) = \int \underbrace{\delta(z_i - g(\boldsymbol{z}_{c\setminus i}))}_{f_c(\boldsymbol{z}_c)} f_d(\boldsymbol{z}_d)\, \mathrm{d}z_i, \tag{23}$$

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. This strategy has been discussed in [29, Appendix A.3] before. Here, we show how composite factor nodes enable us to take full advantage of the CVI algorithm by delegating the analytical calculations in CVI to already existing message passing rules in the PPL. f_b is illustrated as a composite node in Fig. 3. 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

$$f_d(\boldsymbol{z}_d) \propto \exp\left(\phi_{di}(z_i)^\intercal \lambda_{di}(\boldsymbol{z}_{d\setminus i})\right), \tag{24}$$

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

$$m_{di}(z_i) \propto \exp\left(\mathbb{E}_{q(\boldsymbol{z}_{d\setminus i})}[\log f_d(\boldsymbol{z}_d)]\right) \propto \exp\left(\phi_{di}(z_i)^\intercal \underbrace{\mathbb{E}_{q(\boldsymbol{z}_{d\setminus i})}[\lambda_{di}(\boldsymbol{z}_{d\setminus i}))]}_{\eta_{di}}\right). \tag{25}$$

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 (Appendix B)

$$\log m_{bj}(z_j) \propto \mathbb{E}_{q(\boldsymbol{z}_{b\setminus j})}[\log f_b(\boldsymbol{z}_b)] \propto \mathbb{E}_{q(\boldsymbol{z}_{c\setminus\{j,i\}})}[\log m_{di}(g(\boldsymbol{z}_{c\setminus i}))]\,. \tag{26}$$

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

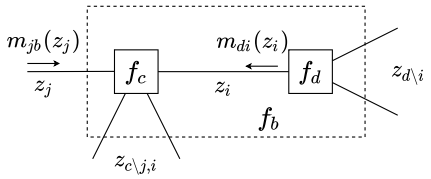


Fig. 3. A factor node $f_b(\boldsymbol{z}_b)$, visualized as a composite node, where $\boldsymbol{z}_{b\setminus j} = \boldsymbol{z}_{c\setminus\{j,i\}} \cup \boldsymbol{z}_{d\setminus i}$. In case the message $m_{bj}(z_j)$ is not defined in closed form for the factor f_b, we require the end-user to 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.

$$\log m_{bj}(z_j) = \log m_{cj}(z_j) \approx \frac{1}{S}\sum_{s=1}^{S} \log m_{di}\left(g\left(z_j, \boldsymbol{z}_{c\setminus\{j,i\}}^{(s)}\right)\right), \text{ where } \boldsymbol{z}_{c\setminus\{j,i\}}^{(s)} \sim q\left(\boldsymbol{z}_{c\setminus\{j,i\}}\right). \tag{27}$$

Once $\log m_{bj}(z_j)$ is estimated, we use CVI as in Section 3.2 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 (see Appendix B for a discussion). Next, we will provide CVI algorithm around the deterministic node f_c.

3.3.1. CVI around deterministic nodes

Given $f_b(\boldsymbol{z}_b)$ is decomposed as (23), 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 $q(\boldsymbol{z}_{c\setminus i})$:

$$q(\boldsymbol{z}_{c\setminus i}) = \prod_{\substack{j\in\mathcal{E}(c)\\ j\neq i}} q(z_j). \tag{28}$$

We provide a high level summary for CVI around the deterministic node $f_c(\boldsymbol{z}_c)$ in Algorithm 2.

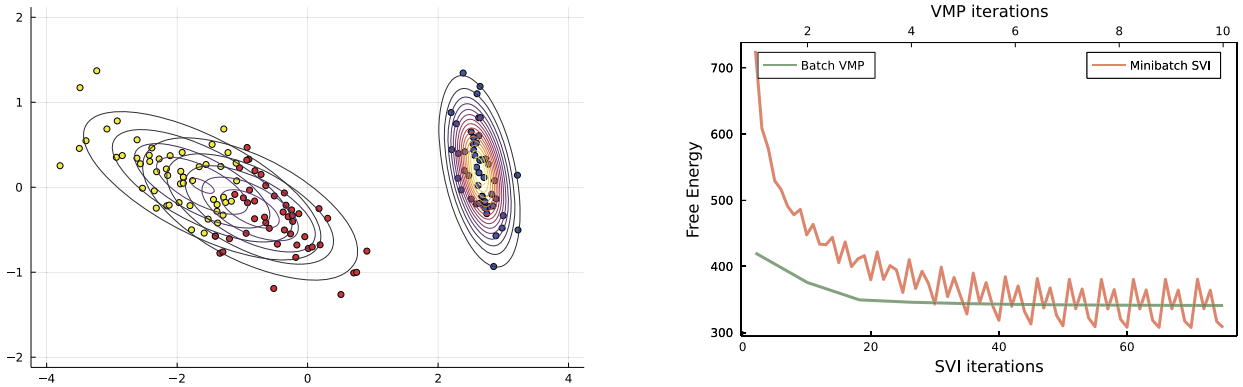
Algorithm 2 CVI around a deterministic node in an FFG.

Require: A sub-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 $m_{jc}(z_j) \propto \exp\left(\phi_{jc}(z_j)^\intercal \eta_{jc}\right)$
for $j \in \mathcal{E}(c), j \neq i$ **do**
  Estimate $\log m_{cj}(z_j)$ as in (27)
  Set $\eta_j^{(0)} \longleftarrow \eta_{jc}$
  for $t = 1 : T_j$ **do**
    Calculate $\tilde{\nabla}_{\eta_j}^{N} \mathcal{F}(\eta_j)$ ▷ See Section 3.2
    Set a step size $\rho^{(t)}$
    Update η_j using (11)
  Set $q(z_j) \propto \exp\left(\phi_j(z_j)^\intercal \eta_j^{(T_j)}\right)$
  Set $\nu_{cj}(z_j) \propto \exp(\phi_j(z_j)^\intercal (\eta_j^{(T_j)} - \eta_{jc}))$
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 (28)

Algorithm 2 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 3.2.

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 2 allow VMP messages to be computed in neighboring factor nodes. Notice that in the last line of Algorithm 2, we set $q(z_i)$ to a set of samples, which allows expectation quantities in VMP messages to be estimated with Monte Carlo summation, automatically [29] (Appendix B), similarly to [44].

In Algorithm 2, 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.



(a) GMM estimations with SVI (b) Free energy comparison

Fig. 4. Visualization of the marginal posterior and free energy estimations with SVI. These results verify that our SVI implementation in ForneyLab performs as expected in the theory.

4. Experiments

In this section, we show the effectiveness of the SVI and CVI implementations in the message passing-based PPL *ForneyLab.jl*. Our implementation is readily available.[1] The experiments can be also accessed online.[2]

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) [45, Chapter 20] for the Iris data set [46,47] after reducing the dimensionality of the data samples from 4 to 2 by Principal Component Analysis [48, Chapter 12]. The Iris data set comprises 150 data samples, equally distributed among three classes. We define the GMM by

$$f(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{W}, s) = f_s(s) \prod_{k=1}^{3} f_\mu(\mu_k) f_W(W_k) \prod_{n=1}^{150} f_z(z_n, s) f_y(y_n, z_n, \boldsymbol{\mu}, \boldsymbol{W}), \tag{29}$$

$$\begin{aligned} f_s(s) &= \mathcal{D}ir(s; [50, 50, 50]) \\ f_\mu(\mu_k) &= \mathcal{N}(\mu_k; \mathbf{0}_2, \boldsymbol{I}_{2\times 2}) \\ f_W(W_k) &= \mathcal{W}_2(W_k; \boldsymbol{I}_{2\times 2}, 2) \\ f_z(z_n, s) &= \mathcal{C}at(z_n; s) \\ f_y(y_n, z_n, \boldsymbol{\mu}, \boldsymbol{W}) &= \prod_{k=1}^{3} \mathcal{N}\left(y_n; \mu_k, W_k^{-1}\right)^{\mathbb{I}[z_n=k]}, \end{aligned} \tag{30}$$

where $\mathcal{N}$, $\mathcal{W}$, $\mathcal{D}ir$, $\mathcal{C}at$ 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. $\mathbb{I}[z_n = k]$ 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 $f_y(y_n, z_n, \boldsymbol{\mu}, \boldsymbol{W})$, 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 $q(\boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{W}, s)$:

$$q(\boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{W}, s) = q(s) \prod_{k=1}^{3} q(\mu_k) q(W_k) \prod_{n=1}^{150} q(z_n). \tag{31}$$

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 Fig. 4. We use the mean estimates for $q(\mu_k)$ and $q(W_k)$ to set the mean and precision

[1] https://github.com/semihakbayrak/ForneyLab.jl/tree/StochasticVMP.
[2] https://github.com/biaslab/StochasticVMP.

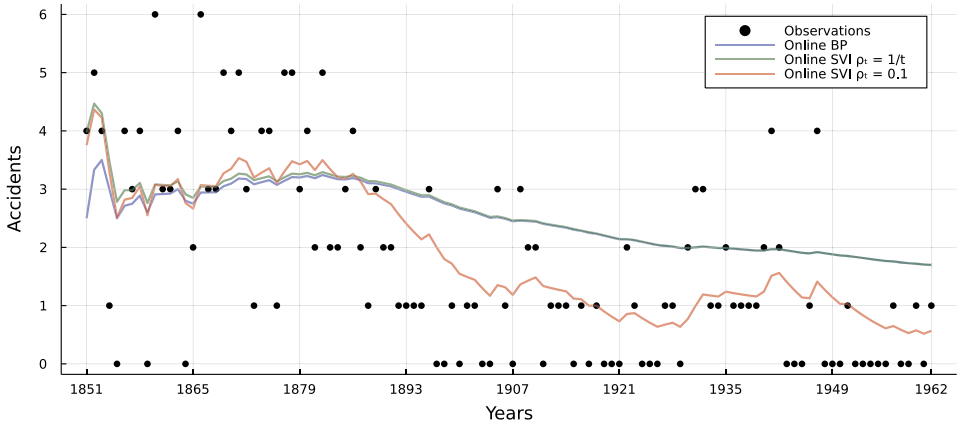


Fig. 5. Coal mining accidents in the United Kingdom from 1851 to 1962. In 1887, new safety regulations are exerted to prevent accidents in mining sites. We show that it is possible to achieve online BP using SVI with a step size satisfying the Robins-Monro conditions. Violating the Robins-Monro conditions and keeping the step size fixed over iterations, we are able to track the hidden non-stationary process shown by the red curve.

parameters of the visualized clusters. The cluster assignments for data samples are shown in red, blue, and yellow colors. To colorize the data samples on the plot, we use the maximum of $q(z_n)$. On the right-hand side of Fig. 4, we see that SVI performs on par with VMP in terms of free energy minimization. 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.

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 [49], visualized with black points in Fig. 5.

We model the number of accidents with a Poisson likelihood, i.e., $f_y(y_t, z_t) = \mathcal{P}o(y_t; z_t)$ 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 $f_z(z) = \mathcal{G}a(z; 1, 1)$ on z. We run BP in an online setting, processing the number of accidents one by one and updating the prior $f_z(z)$ at each time step with the posterior estimated in the previous time step. We visualize the mean estimations with a blue curve in Fig. 5.

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 f_z. Notice that f_y and f_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 3, 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 [50]. 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 [51]. Keeping $\rho^{(t)}$ fixed over time weighs the contributions from recent observations more than earlier observations in updating η through (11). The mean of $q(z)$ over time is visualized by a red curve in Fig. 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 [52]. 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.

4.3. Hierarchical probabilistic modeling 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 [53], we introduce a slightly different

Table 1
Run time (sec.) and free energy comparisons for the hierarchical model and the sensor fusion experiments.

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

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 second time after attending a special coaching program:

$$f(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w}, \alpha, \beta, \mu, s) = f_\alpha(\alpha) f_\beta(\beta) f_\mu(\mu) f_s(s) \prod_{i=1}^{8} f_x(x_i, \mu, s) f_w(w_i, \alpha, \beta) \prod_{n=1}^{N_i} f_y(y_{in}, x_i, w_i), \tag{32}$$

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

We change the original problem and model specification in [53] to introduce a non-conjugacy that stems from $f_\alpha(\alpha)$ 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 [53, Table 5.2].

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

$$q(\boldsymbol{x}, \boldsymbol{w}, \alpha, \beta, \mu, s) = q(\alpha) q(\beta) q(\mu) q(s) \prod_{i=1}^{8} q(x_i) q(w_i). \tag{33}$$

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 [54] and the default optimizer set by Turing. The run time and free energy comparisons are given in Table 1.[3] We see that Turing and ForneyLab perform 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 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.

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 [52, 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.

[3] 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.

4.4.1. 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:

$$
\begin{aligned}
f(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{z}) &= f_z(z_1) f_x(x_1, z_1) f_y(y_1, x_1) \prod_{t=2}^{T} f_z(z_t, z_{t-1}) f_x(x_t, z_t) f_y(y_t, x_t), \text{ where} \\
f_z(z_1) &= \mathcal{N}(z_1; \mathbf{0}_4, \boldsymbol{I}_{4\times 4}) \\
f_z(z_t, z_{t-1}) &= \mathcal{N}(z_t; Az_{t-1}, \boldsymbol{I}_{4\times 4}) \\
f_x(x_t, z_t) &= \delta(x_t - g(z_t)) \text{ with } g(z_t) = [||\xi_1 - Bz_t||, ||\xi_2 - Bz_t||, ||\xi_3 - Bz_t||]^\intercal \\
f_y(y_t, x_t) &= \mathcal{N}(y_t; x_t, \boldsymbol{I}_{3\times 3}),
\end{aligned}
\tag{34}
$$

where z_t is the vector of hidden position h_t and speed values, $A = \left[\begin{array}{c|c} I_{2\times 2} & I_{2\times 2} \\ \hline 0_{2\times 2} & I_{2\times 2} \end{array}\right]$ and $B = \left[\begin{array}{c|c} I_{2\times 2} & 0_{2\times 2} \end{array}\right]$. 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 to our message passing procedure through the factors f_x. At a given time step t, we run CVI algorithm around f_x 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 $f_x(x_t, z_t)$ (see Fig. 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 f_z 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 [55] 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 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.

4.4.2. 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 (34) slightly changes as

$$
\begin{aligned}
f(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{z}, W, S) &= f_W(W) f_S(S) f_z(z_1) f_x(x_1, z_1) f_y(y_1, x_1, S) \\
&\quad \prod_{t=2}^{T} f_z(z_t, z_{t-1}, W) f_x(x_t, z_t) f_y(y_t, x_t, S), \text{ where} \\
f_z(z_1) &= \mathcal{N}(z_1; \mathbf{0}_4, \boldsymbol{I}_{4\times 4}) \\
f_W(W) &= \mathcal{W}_4(W; \boldsymbol{I}_{4\times 4}, 4) \\
f_S(S) &= \mathcal{W}_3(S; \boldsymbol{I}_{3\times 3}, 3) \\
f_z(z_t, z_{t-1}, W) &= \mathcal{N}(z_t; Az_{t-1}, W^{-1}) \\
f_x(x_t, z_t) &= \delta(x_t - g(z_t)) \text{ with } g(z_t) = [||\xi_1 - Bz_t||, ||\xi_2 - Bz_t||, ||\xi_3 - Bz_t||]^\intercal \\
f_y(y_t, x_t, S) &= \mathcal{N}(y_t; x_t, S^{-1}),
\end{aligned}
\tag{35}
$$

where $\mathcal{W}_d(V, n)$ 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 Fig. 6.

Notice that we resort to NGD stochastic approximation only around the factor f_x 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 Fig. 7. For qualitative analysis, we also visualize the final position estimations with 1000 samples drawn from $q(\boldsymbol{z})$.

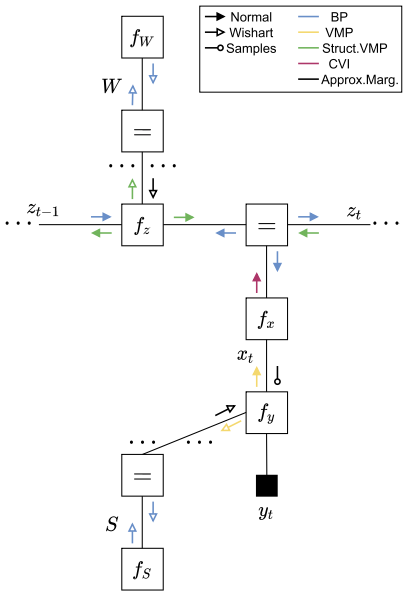
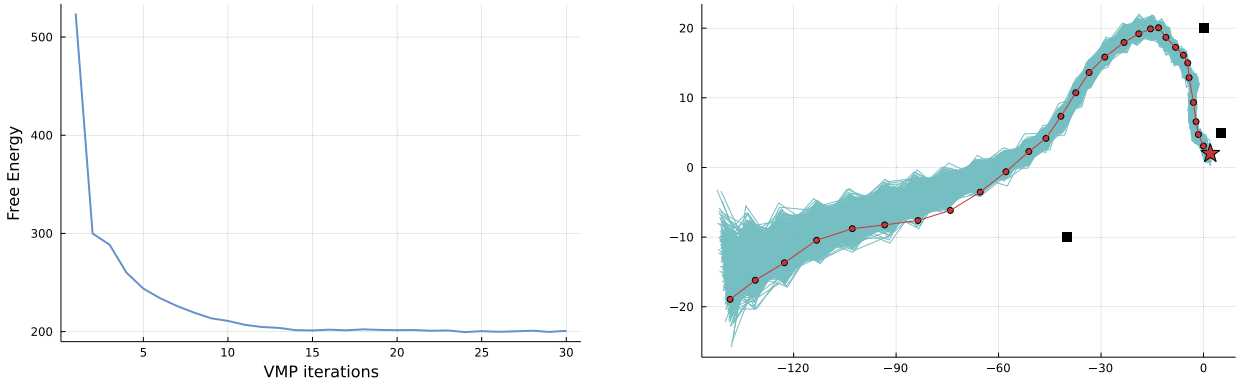


Fig. 6. A sub-graph of the model defined in (35). In order to provide the reader with the intuition of the message passing procedures in *ForneyLab*, we visualize the messages as well. The arrow shapes indicate the probability distribution families that messages are carrying. The messages are colored 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.



(a) Free energy over VMP iterations

(b) Position estimations

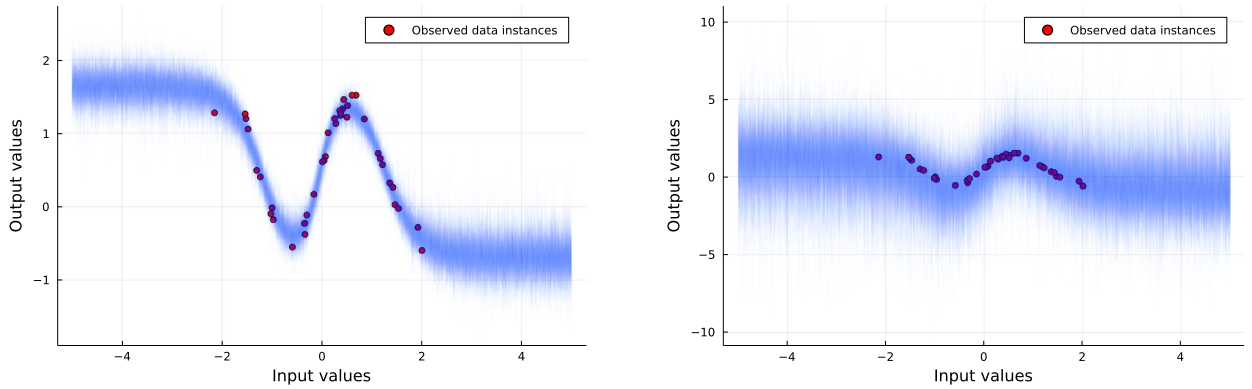
Fig. 7. On the left is the free energy over VMP iterations for the model defined in (35). On the right is the 2-d environment the object moves in. The red curve shows the true trajectory of the movement with the star being the initial point. Black squares are the sensors measuring the Euclidean distances 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.

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.

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 [6] and TensorFlow Probability [7] respectively interface with PyTorch [56] and Tensorflow [57]; Turing [4] supports model specifications with Flux.jl [58]. Inspired by Turing, here we show how to use ForneyLab's CVI to make inference in a Bayesian Neural Network (BNN) built [59] with 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:



(a) CVI estimations (b) EVMP estimations

Fig. 8. A simple example proves that equipped with CVI, 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 these plots, 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 on the left and EVMP on the right. The scale of y-axes differs as well as the estimations.

$$f(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{s}, w) = f_w(w) \prod_{n=1}^{40} \underbrace{\delta(s_n - g(w, x_n))}_{\text{Deterministic node}} f_y(y_n, s_n), \text{ where} \tag{36}$$

$$f_w(w) = \mathcal{N}(w; \mathbf{0}_{22}, \boldsymbol{I}_{22\times 22})$$

$$f_y(y_n, s_n) = \mathcal{N}(y_n; s_n, 0.1).$$

In the above model specification, $g(w, x_n)$ is a three-layered neural network comprised of 22 weights with a prior $f_w(w)$, where $\mathbf{0}_{22}$ is 22 dimensional vector of zeros and $\boldsymbol{I}_{22\times 22}$ is an identity matrix. 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 a 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 Fig. 8a. 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 uncertainty. This simple experiment demonstrates how ForneyLab with CVI seamlessly interfaces with a deep learning package.

In Section 5, we shall discuss an alternative automatic message passing approach called Extended Variational Message Passing (EVMP) that is also implemented in ForneyLab [29]. Here, by running EVMP on this BNN model, we show how CVI extends the inference capabilities of ForneyLab. The EVMP inference engine automatically runs a gradient-based optimization method for the Laplace approximation [48, Chapter 4.4] in this BNN model. However, EVMP initializes the optimization with the mean of the prior, which is a vector of zeros in this example, and a stationary point that the mode-seeking gradient-based optimizer gets stuck at. To avoid this stationary point, we slightly change the prior by randomly drawing the samples of the mean parameter from $\mathcal{N}(0, 0.1)$ and then run EVMP. The results are visualized in Fig. 8b. Although the sinusoidal shape is captured, the variance is overestimated in the EVMP case. This is due to that the automated Laplace procedure in EVMP is a mode-seeking approach, and the covariance estimation is realized by local curvature evaluation. In contrast, covariance parameters are actively adjusted by the NGD optimization procedure of CVI.

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 [8], in which the stochasticity is due to both mini-batch selection process from the data set and sampling from the candidate approximate posterior distribution. This process is same both 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 behaviors. 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 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 (15), 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 (26) 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 [22] proposes an efficient estimation approach that does not necessitate the explicit evaluation of Fisher information matrix for Gaussian approximate distributions and recommends reparameterization trick [12,13,8] for the other distributions. We adhere to their efficient approach for the Gaussian case, but use the REINFORCE estimator (22) 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. [9] shows that Rao-Blackwellization [10] and control variates [11] considerably reduce the variance in free energy gradient estimations. Fortunately, message passing frameworks inherently support Rao-Blackwellization and closed-form solutions (Appendix B). 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 [60–62]. The most recently introduced approach is the implicit reparameterization trick [62], which reparameterizes variables using Cumulative Distribution Functions (CDFs). We plan to implement this feature in a future release of ForneyLab.

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 [58]. We also implemented the optimizer proposed in [21, Equation 26] satisfying Robins-Monro conditions. Additionally, we provide an implementation for [63], 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 [64]. 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 is 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 (11) 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 [65] to our message passing-based PPL.

ForneyLab possesses an alternative automated inference engine for universal approximate inference called Extended Variational Message Passing (EVMP) [29]. At the core of EVMP is the idea that FFGs partition the high dimensional manifolds, and message passing algorithms arise as distributed inference operations defined on smaller dimensions. In EVMP, importance sampling [48, Chapter 11.1.4] locally interfaces with message passing to allow inference in non-conjugate sub-graphs. This approach works well for many model specifications, it is fast and it requires almost no hyperparameters as one of the messages on the edge constitutes the proposal distribution for importance sampling. However, the quality of estimations in EVMP strictly depends on the model specification. In comparison to EVMP, the inference with CVI is more detached from the model specification thanks to the NGD optimization procedure. However, the CVI algorithm is more hyperparameter dependent than the EVMP algorithm. Another disadvantage of importance sampling is that it is notorious for being inefficient in high dimensions [66], and there might be cases that the partitioned manifolds are still high-dimensional. To support inference in high dimensions, EVMP incorporates the Laplace approximation into message passing with automatic differentiation. The Laplace approximation first finds the mode and then estimates the covariance matrices of approximating Gaussian distributions with local curvature information. In contrast, CVI implicitly tunes the covariance matrix over the course of the optimization procedure in Gaussian case. The difference between these two approaches is sketched in Section 4.5. Moreover, SVI and CVI address scalability issues that EVMP does not deal with.

6. Conclusions

This paper 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.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research was financially supported by GN Hearing and a PPS grant from the Dutch Ministry of Economic Affairs.

Appendix A. Stationary points of the free energy

Consider the free energy objective given in (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 (13) 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 (14). Then the gradient of the free energy w.r.t. η_j is

$$\begin{aligned}\nabla_{\eta_j}\mathcal{F}(\eta_j) &= \nabla_{\eta_j}\left(\mathbb{E}_{q(z_j;\eta)}[\log q(z_j;\eta_j)] - \mathbb{E}_{q(z_j;\eta_j)}\left[\log m_{jb}(z_j)\right] - \mathbb{E}_{q(z_j;\eta_j)}\left[\log m_{bj}(z_j)\right]\right) \\ &= \nabla_{\eta_j}\left(\mathbb{E}_{q(z_j;\eta_j)}[\phi_j(z_j)]^\intercal(\eta_j - \eta_{jb}) - A_j(\eta_j) - \mathbb{E}_{q(z_j;\eta_j)}\left[\log m_{bj}(z_j)\right]\right) \\ &= \nabla_{\eta_j}\mathbb{E}_{q(z_j;\eta_j)}[\phi_j(z_j)]^\intercal(\eta_j - \eta_{jb}) + \mathbb{E}_{q(z_j;\eta_j)}[\phi_j(z_j)] - \nabla_{\eta_j}A_j(\eta_j) - \nabla_{\eta_j}\mathbb{E}_{q(z_j;\eta_j)}[\log m_{bj}(z_j)] \\ &= \nabla^2_{\eta_j}A_j(\eta_j)\left[\eta_j - \eta_{jb}\right] - \nabla_{\eta_j}\mathbb{E}_{q(z_j;\eta_j)}[\log m_{bj}(z_j)].\end{aligned}$$

The last line above follows from $\nabla_{\eta_j}A(\eta_j) = \mathbb{E}_{q(z_j;\eta_j)}[\phi_j(z_j)]$ [34]. 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 [21], we write the natural gradient of the free energy as

$$\nabla^N_{\eta_j}\mathcal{F}(\eta_j) = G^{-1}(\eta_j)\nabla_{\eta_j}\mathcal{F}(\eta_j) = \eta_j - \left(\eta_{jb} + G^{-1}(\eta_j)\nabla_{\eta_j}\mathbb{E}_{q(z_j;\eta_j)}[\log m_{bj}(z_j)]\right).$$

Appendix B. VMP on composite nodes and 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 Fig. 3. Given 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 $f_d(\boldsymbol{z}_d) \propto \exp\left(\phi_{di}(z_i)^\intercal\lambda_{di}(\boldsymbol{z}_{d\setminus i})\right)$, log of the VMP message $m_{bj}(z_j)$ evaluates to

$$\begin{aligned}\log m_{bj}(z_j) &\propto \mathbb{E}_{q(\boldsymbol{z}_{b\setminus j})}[\log f_b(\boldsymbol{z}_b)] \\ &= \mathbb{E}_{q(\boldsymbol{z}_{b\setminus j})}\left[\log\int \delta(z_i - g(\boldsymbol{z}_{c\setminus i}))\,f_d(\boldsymbol{z}_d)\,\mathrm{d}z_i\right] \\ &\propto \mathbb{E}_{q(\boldsymbol{z}_{b\setminus j})}\left[\log\int \delta(z_i - g(\boldsymbol{z}_{c\setminus i}))\exp\left(\phi_{di}(z_i)^\intercal\lambda_{di}(\boldsymbol{z}_{d\setminus i})\right)\,\mathrm{d}z_i\right] \\ &= \mathbb{E}_{q(\boldsymbol{z}_{b\setminus j})}\left[\phi_{di}(g(\boldsymbol{z}_{c\setminus i}))^\intercal\lambda_{di}(\boldsymbol{z}_{d\setminus i})\right].\end{aligned} \tag{B.1}$$

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 [10] and VMP rules defined in *ForneyLab* help us to carry out variance reduction in an automated way:

$$\begin{aligned}\log m_{bj}(z_j) &\propto \mathbb{E}_{q(\boldsymbol{z}_{c\setminus\{j,i\}})}\left[\mathbb{E}_{q(\boldsymbol{z}_{d\setminus i}|\boldsymbol{z}_{c\setminus\{j,i\}})}\left[\phi_{di}(g(\boldsymbol{z}_{c\setminus i}))^{\intercal}\lambda_{di}(\boldsymbol{z}_{d\setminus i})\right]\right]\\ &= \mathbb{E}_{q(\boldsymbol{z}_{c\setminus\{j,i\}})}\left[\mathbb{E}_{q(\boldsymbol{z}_{d\setminus i})}\left[\phi_{di}(g(\boldsymbol{z}_{c\setminus i}))^{\intercal}\lambda_{di}(\boldsymbol{z}_{d\setminus i})\right]\right]\\ &= \mathbb{E}_{q(\boldsymbol{z}_{c\setminus\{j,i\}})}\left[\phi_{di}(g(\boldsymbol{z}_{c\setminus i}))^{\intercal}\mathbb{E}_{q(\boldsymbol{z}_{d\setminus i})}\left[\lambda_{di}(\boldsymbol{z}_{d\setminus i})\right]\right]\\ &\propto \mathbb{E}_{q(\boldsymbol{z}_{c\setminus\{j,i\}})}[\log m_{di}(g(\boldsymbol{z}_{c\setminus i}))],\end{aligned} \tag{B.2}$$

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 (B.2) is less than (B.1). 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.

For VMP procedure to progress flawlessly, the VMP messages towards $\boldsymbol{z}_{d\setminus i}$ need to be evaluated, as well. Keeping $q(\boldsymbol{z}_{d\setminus i})$ structured and arranging $\boldsymbol{z}_d$ terms inside $f_d(\boldsymbol{z}_d)$ as $f_d(\boldsymbol{z}_d) \propto \exp\left(\phi_{\mathcal{E}(d)\setminus i}(\boldsymbol{z}_{d\setminus i})^{\intercal}\lambda_{\mathcal{E}(d)\setminus i}(z_i)\right)$, we get the following VMP message

$$\begin{aligned}m_{\mathcal{E}(d)\setminus i}(\boldsymbol{z}_{d\setminus i}) &\propto \exp(\mathbb{E}_{q(z_i)}[\log f_d(\boldsymbol{z}_d)])\\ &= \exp\left(\phi_{\mathcal{E}(d)\setminus i}(\boldsymbol{z}_{d\setminus i})^{\intercal}\mathbb{E}_{q(z_i)}[\lambda_{\mathcal{E}(d)\setminus i}(z_i)]\right),\end{aligned} \tag{B.3}$$

which is a function of the expectation quantity $\mathbb{E}_{q(z_i)}[\lambda_{\mathcal{E}(d)\setminus i}(z_i)]$ with a fixed functional form. In our message passing-based PPL, message passing rules are locally defined on a factor as functions of expectation quantities related to arguments of the factor that is z_i for the message $m_{\mathcal{E}(d)\setminus i}(\boldsymbol{z}_{d\setminus i})$.

Let us evaluate the same message over the composite factor node $f_b(\boldsymbol{z}_b)$:

$$\begin{aligned}m_{\mathcal{E}(d)\setminus i}(\boldsymbol{z}_{d\setminus i}) &\propto \exp(\mathbb{E}_{q(\boldsymbol{z}_{c\setminus i})}[\log f_b(\boldsymbol{z}_{\boldsymbol{b}})])\\ &= \exp\left(\mathbb{E}_{q(\boldsymbol{z}_{c\setminus i})}\left[\log\int \delta(z_i - g(\boldsymbol{z}_{c\setminus i}))\, f_d(\boldsymbol{z}_d)\,\mathrm{d}z_i\right]\right)\\ &\propto \exp\left(\phi_{\mathcal{E}(d)\setminus i}(\boldsymbol{z}_{d\setminus i})^{\intercal}\mathbb{E}_{q(\boldsymbol{z}_{c\setminus i})}\left[\lambda_{\mathcal{E}(d)\setminus i}(g(\boldsymbol{z}_{c\setminus i}))\right]\right).\end{aligned} \tag{B.4}$$

Having estimated $q(\boldsymbol{z}_{c\setminus i})$ with CVI, the VMP message can be approximated by estimating the expectation $\mathbb{E}_{q(\boldsymbol{z}_{c\setminus i})}\left[\lambda_{\mathcal{E}(d)\setminus i}(g(\boldsymbol{z}_{c\setminus i}))\right]$ with

$$\mathbb{E}_{q(\boldsymbol{z}_{c\setminus i})}\left[\lambda_{\mathcal{E}(d)\setminus i}(g(\boldsymbol{z}_{c\setminus i}))\right] \approx \frac{1}{S}\sum_{s=1}^{S}\lambda_{\mathcal{E}(d)\setminus i}\left(g\left(\boldsymbol{z}_{c\setminus i}^{(s)}\right)\right), \text{ where } \boldsymbol{z}_{c\setminus i}^{(s)} \sim q(\boldsymbol{z}_{c\setminus i}).$$

This is equivalent to setting $q(z_i)$ to a set of samples

$$q(z_i) = \left\{z_i^{(s)} = g\left(\boldsymbol{z}_{c\setminus i}^{(s)}\right) \middle| \text{for } s \in \{1,\ldots,S\}\right\},$$

where the expectations are calculated as

$$\mathbb{E}_{q(z_i)}[\lambda_{\mathcal{E}(d)\setminus i}(z_i)] = \frac{1}{S}\sum_{s=1}^{S} z_i^{(s)},$$

and employed in the pre-defined local VMP message function given in (B.3) to approximate the VMP message $m_{\mathcal{E}(d)\setminus i}(\boldsymbol{z}_{d\setminus i})$.

References

[1] J.W. van de Meent, B. Paige, H. Yang, F. Wood, An introduction to probabilistic programming, arXiv:1809.10756 [cs, stat], 2018.
[2] S. Mohamed, M. Rosca, M. Figurnov, A. Mnih, Monte Carlo gradient estimation in machine learning, arXiv:1906.10652 [cs, math, stat], 2019.
[3] C. Zhang, J. Butepage, H. Kjellstrom, S. Mandt, Advances in variational inference, arXiv:1711.05597 [cs, stat], 2018.
[4] H. Ge, K. Xu, Z. Ghahramani, Turing: a language for flexible probabilistic inference, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2018, pp. 1682–1690.
[5] B. Carpenter, A. Gelman, M.D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, A. Riddell, Stan: a probabilistic programming language, J. Stat. Softw. 76 (1) (2017).
[6] E. Bingham, J.P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, N.D. Goodman, Pyro: deep universal probabilistic programming, J. Mach. Learn. Res. 20 (28) (2019) 1–6.
[7] J.V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, R.A. Saurous, TensorFlow distributions, arXiv: 1711.10604 [cs, stat], 2017.
[8] M. Titsias, M. Lázaro-Gredilla, Doubly stochastic variational Bayes for non-conjugate inference, in: International Conference on Machine Learning, 2014, pp. 1971–1979.
[9] R. Ranganath, S. Gerrish, D. Blei, Black box variational inference, in: Artificial Intelligence and Statistics, PMLR, 2014, pp. 814–822.

[10] G. Casella, C.P. Robert, Rao-blackwellisation of sampling schemes, Biometrika 83 (1) (1996) 81–94, publisher: [Oxford University Press, Biometrika Trust] https://www.jstor.org/stable/2337434.
[11] A.B. Owen, Monte Carlo Theory, Methods and Examples, 2013.
[12] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, arXiv:1312.6114 [cs, stat], 2014.
[13] D.J. Rezende, S. Mohamed, D. 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, JMLR.org, Beijing, China, 2014, pp. II–1278–II–1286.
[14] A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, D.M. Blei, Automatic differentiation variational inference, J. Mach. Learn. Res. 18 (1) (2017) 430–474, publisher: JMLR.org.
[15] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, 1988.
[16] D.J. MacKay, Information Theory, Inference and Learning Algorithms, Cambridge University Press, 2003.
[17] T.P. Minka, Expectation propagation for approximate Bayesian inference, in: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, 2001, pp. 362–369.
[18] A. Vehtari, A. Gelman, T. Sivula, P. Jylänki, D. Tran, S. Sahai, P. Blomstedt, J.P. Cunningham, D. Schiminovich, C.P. Robert, Expectation propagation as a way of life: a framework for Bayesian inference on partitioned data, J. Mach. Learn. Res. 21 (17) (2020) 1–53.
[19] J. Winn, C.M. Bishop, Variational message passing, J. Mach. Learn. Res. 6 (Apr) (2005) 661–694.
[20] J. Dauwels, On variational message passing on factor graphs, in: IEEE International Symposium on Information Theory, 2007, pp. 2546–2550.
[21] M. Hoffman, D.M. Blei, C. Wang, J. Paisley, Stochastic variational inference, J. Mach. Learn. Res. 14 (1) (2013) 1303–1347, publisher: JMLR.org.
[22] M. Khan, W. Lin, Conjugate-computation variational inference: converting variational inference in non-conjugate models to inferences in conjugate models, in: Artificial Intelligence and Statistics, PMLR, 2017, pp. 878–887.
[23] S. Amari, Natural gradient works efficiently in learning, Neural Comput. 10 (2) (1998) 251–276.
[24] S. Amari, Information Geometry and Its Applications, Applied Mathematical Sciences, vol. 194, Springer Japan, Tokyo, 2016.
[25] T. Minka, J. Winn, J. Guiver, Y. Zaykov, D. Fabian, J. Bronskill, /Infer.NET 0.3, http://dotnet.github.io/infer, 2018.
[26] J. Bezanson, S. Karpinski, V.B. Shah, A. Edelman, Julia: a fast dynamic language for technical computing, arXiv preprint, arXiv:1209.5145, 2012.
[27] D. Bagaev, B. de Vries, Reactive message passing for scalable Bayesian inference, Tech. Rep. arXiv:2112.13251 [cs] type: article (Dec. 2021).
[28] M. Cox, T. van de Laar, B. de Vries, A factor graph approach to automated design of Bayesian signal processing algorithms, Int. J. Approx. Reason. 104 (2019) 185–204.
[29] S. Akbayrak, I. Bocharov, B. de Vries, Extended variational message passing for automated approximate Bayesian inference, Entropy 23 (7) (2021) 815, https://doi.org/10.3390/e23070815, https://www.mdpi.com/1099-4300/23/7/815.
[30] G. Forney, Codes on graphs: normal realizations, IEEE Trans. Inf. Theory 47 (2) (2001) 520–548, https://doi.org/10.1109/18.910573, conference Name: IEEE Transactions on Information Theory.
[31] H.A. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping, F.R. Kschischang, The factor graph approach to model-based signal processing, Proc. IEEE 95 (6) (2007) 1295–1322, publisher: IEEE.
[32] İ. Şenöz, T. van de Laar, D. Bagaev, B. de Vries, Variational message passing and local constraint manipulation in factor graphs, Entropy 23 (2021) 807, https://doi.org/10.3390/e23070807, number: 7 Publisher: Multidisciplinary Digital Publishing Institute https://www.mdpi.com/1099-4300/23/7/807.
[33] M.J. Beal, Variational algorithms for approximate Bayesian inference, PhD Thesis, UCL (University College London), 2003.
[34] M.J. Wainwright, M.I. Jordan, Graphical models, exponential families, and variational inference, Found. Trends Mach. Learn. 1 (1–2) (2008) 1–305.
[35] U. Paquet, N. Koenigstein, One-class collaborative filtering with random graphs, in: Proceedings of the 22nd International Conference on World Wide Web - WWW'13, ACM Press, Rio de Janeiro, Brazil, 2013, pp. 999–1008.
[36] A.R. Masegosa, A.M. Martínez, H. Langseth, T.D. Nielsen, A. Salmerón, D. Ramos-López, A.L. Madsen, d-VMP: distributed variational message passing, in: Proceedings of the Eighth International Conference on Probabilistic Graphical Models, PMLR, 2016, pp. 321–332, ISSN: 1938-7228, https://proceedings.mlr.press/v52/masegosa16.html.
[37] M.E. Khan, H. Rue, The Bayesian learning rule, arXiv:2107.04562 [cs, stat], 2021.
[38] H. Robbins, S. Monro, A stochastic approximation method, Ann. Math. Stat. 22 (3) (1951) 400–407, publisher: Institute of Mathematical Statistics, https://www.jstor.org/stable/2236626.
[39] U. Paquet, On the convergence of stochastic variational inference in Bayesian networks, in: NIPS Workshop on Variational Inference, 2014, arXiv: 1507.04505, http://arxiv.org/abs/1507.04505.
[40] D.A. Knowles, T. Minka, Non-conjugate variational message passing for multinomial and binary regression, in: Advances in Neural Information Processing Systems, 2011, pp. 1701–1709.
[41] M. Cox, B. De Vries, Robust expectation propagation in factor graphs involving both continuous and binary variables, in: 2018 26th European Signal Processing Conference (EUSIPCO), IEEE, Rome, 2018, pp. 2583–2587, https://ieeexplore.ieee.org/document/8553490/.
[42] M. Opper, C. Archambeau, The variational Gaussian approximation revisited, Neural Comput. 21 (3) (2009) 786–792, https://doi.org/10.1162/neco.2008.08-07-592, https://direct.mit.edu/neco/article/21/3/786-792/7385.
[43] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: a survey, J. Mach. Learn. Res. 18 (1) (2017) 5595–5637, publisher: JMLR.org.
[44] L. Ye, A. Beskos, M. De Iorio, J. Hao, Monte Carlo co-ordinate ascent variational inference, Stat. Comput. (2020) 1–19, Publisher: Springer.
[45] D. Barber, Bayesian Reasoning and Machine Learning, Cambridge University Press, 2012.
[46] R.A. Fisher, The use of multiple measurements in taxonomic problems, Annu. Eugen. 7 (2) (1936) 179–188, https://doi.org/10.1111/j.1469-1809.1936.tb02137.x, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1469-1809.1936.tb02137.x.
[47] E. Anderson, The species problem in Iris, Ann. Mo. Bot. Gard. 23 (3) (1936) 457–509, https://doi.org/10.2307/2394164, publisher: Missouri Botanical Garden Press, https://www.jstor.org/stable/2394164.
[48] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[49] B.P. Carlin, A.E. Gelfand, A.F.M. Smith, Hierarchical Bayesian analysis of changepoint problems, J. R. Stat. Soc., Ser. C, Appl. Stat. 41 (2) (1992) 389–405, https://doi.org/10.2307/2347570, publisher: [Wiley, Royal Statistical Society], https://www.jstor.org/stable/2347570.
[50] R.P. Adams, D.J.C. MacKay, Bayesian online changepoint detection, arXiv:0710.3742, 2007.
[51] R.S. Sutton, A.G. Barto, Reinforcement Learning, second edition, An Introduction, MIT Press, 2018, google-Books-ID: uWV0DwAAQBAJ.
[52] A.T. Cemgil, A Tutorial Introduction to Monte Carlo Methods, Markov Chain Monte Carlo and Particle Filtering, Academic Press Library in Signal Processing, vol. 1, Elsevier, 2014, pp. 1065–1114, https://linkinghub.elsevier.com/retrieve/pii/B978012396502800019X.
[53] A. Gelman, J.B. Carlin, H.S. Stern, D.B. Dunson, A. Vehtari, D.B. Rubin, Bayesian Data Analysis, CRC Press, 2013.
[54] J. Revels, M. Lubin, T. Papamarkou, Forward-mode automatic differentiation in Julia, arXiv:1607.07892 [cs.MS], 2016.
[55] J. Revels, ReverseDiff.jl, https://github.com/JuliaDiff/ReverseDiff.jl, 2017.
[56] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: an imperative style, high-performance deep learning library, arXiv:1912.01703 [cs, stat], 2019.

[57] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, 2015, 19.
[58] M. Innes, E. Saba, K. Fischer, D. Gandhi, M.C. Rudilosso, N.M. Joy, T. Karmali, A. Pal, V. Shah, Fashionable modelling with flux, arXiv:1811.01457 [cs], 2018.
[59] L.V. Jospin, W. Buntine, F. Boussaid, H. Laga, M. Bennamoun, Hands-on Bayesian neural networks – a tutorial for deep learning users, arXiv:2007.06823 [cs, stat], 2021.
[60] F.R. Ruiz, M. Titsias, D. Blei, The generalized reparameterization gradient, in: Advances in Neural Information Processing Systems, 2016.
[61] M. Jankowiak, F. Obermeyer, Pathwise derivatives beyond the reparameterization trick, in: Proceedings of the 35th International Conference on Machine Learning, PMLR, ISSN 2640-3498, 2018, pp. 2235–2244, https://proceedings.mlr.press/v80/jankowiak18a.html.
[62] M. Figurnov, S. Mohamed, A. Mnih, Implicit reparameterization gradients, in: Advances in Neural Information Processing Systems, 2018, https://proceedings.neurips.cc/paper/2018/hash/92c8c96e4c37100777c7190b76d28233-Abstract.html.
[63] R. Ranganath, C. Wang, B. David, E. Xing, An adaptive learning rate for stochastic variational inference, in: Proceedings of the 30th International Conference on Machine Learning, PMLR, 2013, pp. 298–306, ISSN: 1938-7228, https://proceedings.mlr.press/v28/ranganath13.html.
[64] A.K. Dhaka, A. Catalina, M.R. Andersen, M. Magnusson, J.H. Huggins, A. Vehtari, Robust, accurate stochastic optimization for variational inference, in: Advances in Neural Information Processing Systems, 2020.
[65] W. Lin, M. Schmidt, M.E. Khan, Handling the positive-definite constraint in the Bayesian learning rule, in: Proceedings of the 37th International Conference on Machine Learning, PMLR, ISSN 2640-3498, 2020, pp. 6116–6126, https://proceedings.mlr.press/v119/lin20d.html.
[66] D.J.C. Mackay, Introduction to Monte Carlo Methods, in: Learning in Graphical Models, Springer, 1998, pp. 175–204.