Particle Flow Bayes' Rule

Xinshi Chen [*1]   Hanjun Dai [*2]   Le Song [23]

Abstract

We present a particle flow realization of Bayes' rule, where an ODE-based neural operator is used to transport particles from a prior to its posterior after a new observation. We prove that such an ODE operator exists. Its neural parameterization can be trained in a meta-learning framework, allowing this operator to reason about the effect of an individual observation on the posterior, and thus generalize across different priors, observations and to sequential Bayesian inference. We demonstrated the generalization ability of our particle flow Bayes operator in several canonical and high dimensional examples.

1. Introduction

In many data analysis tasks, it is important to estimate unknown quantities $\boldsymbol{x} \in \mathbb{R}^d$ from observations $\mathcal{O}_m := \{o_1, \cdots, o_m\}$. Given prior knowledge $\pi(\boldsymbol{x})$ and likelihood functions $p(o_t|\boldsymbol{x})$, the essence of Bayesian inference is to compute the posterior $p(\boldsymbol{x}|\mathcal{O}_m) \propto \pi(\boldsymbol{x}) \prod_{t=1}^{m} p(o_t|\boldsymbol{x})$ by Bayes' rule. For many nontrivial models, the prior might not be conjugate to the likelihood, making the posterior not in a closed form. Therefore, computing the posterior often results in intractable integration and poses significant challenges. Typically, one resorts to approximate inference methods such as sampling (e.g., MCMC) (Andrieu et al., 2003) or variational inference (Wainwright & Jordan, 2003).

In many real problems, observations arrive sequentially online, and Bayesian inference needs be performed recursively,

$$\overbrace{p(\boldsymbol{x}|\mathcal{O}_{m+1})}^{\text{updated posterior}} \propto \overbrace{p(\boldsymbol{x}|\mathcal{O}_m)}^{\text{current posterior}} \overbrace{p(o_{m+1}|\boldsymbol{x})}^{\text{likelihood}}. \qquad (1)$$

That is the estimation of $p(\boldsymbol{x}|\mathcal{O}_{m+1})$ should be computed based on the estimation of $p(\boldsymbol{x}|\mathcal{O}_m)$ obtained from the last iteration and the presence of the new observation o_{m+1}.

*Equal contribution [1]School of Mathematics, [2]School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, Georgia, USA. [3]Ant Financial, Hangzhou, China. Correspondence to: Xinshi Chen <xinshi.chen@gatech.edu>.

It therefore requires algorithms which allow for efficient online inference. In this case, both standard MCMC and variational inference become inefficient, since the former requires a complete scan of the dataset in each iteration, and the latter requires solving an optimization for every new observation. Thus, sequential Monte Carlo (SMC) (Doucet et al., 2001; Balakrishnan & Madigan, 2006) or stochastic approximations, such as stochastic gradient Langevin dynamics (Welling & Teh, 2011) and stochastic variational inference (Hoffman et al., 2012), are developed to improve the efficiency. However, SMC suffers from the path degeneracy problems in high dimensions (Daum & Huang, 2003; Snyder et al., 2008), and rejuvenation steps are designed but may violate the online sequential update requirement (Canini et al., 2009; Chopin et al., 2013). Stochastic approximation methods are prescribed algorithms that cannot exploit the structure of the problem for further improvement.

To address these challenges, the seminal work of Kernel Bayes Rule (KBR) views the Bayes update as an operator in reproducing kernel Hilbert spaces (RKHS) which can be learned and directly produce the posterior from prior after each observation (Fukumizu et al., 2012). In the KBR framework, the posterior is represented as an embedding $\mu_m := \mathbb{E}_{p(\boldsymbol{x}|\mathcal{O}_m)}[\phi(\boldsymbol{x})]$ using a feature map $\phi(\cdot)$ associated with a kernel function; then the kernel Bayes operator $\mathcal{K}(\cdot, o)$ will take this embedding as input and produce the embedding of the updated posterior,

$$\overbrace{\mu_{m+1}}^{\text{updated embedding}} = \mathcal{K}(\overbrace{\mu_m}^{\text{current embedding}}, o_{m+1}). \qquad (2)$$

Another novel aspect of KBR method is that it contains a training phase and a testing phase, where the structure of the problem at hand (e.g., the likelihood) is taken into account in the training phase, and in the testing phase, the learned operator $\mathcal{K}$ will directly operate on the current posterior μ_m to produce the output. However, despite the nice concepts of KBR operator, it only works well for a limited range of problems due to its strong theoretical assumptions.

In this work, we aim to lift the limitation of KBR operator, and will design a novel continuous particle flow operator $\mathcal{F}$ to realize the Bayes update, for which we call it **particle flow Bayes' rule** (PFBR). In the PFBR framework (Fig. 1), a prior distribution $\pi(\boldsymbol{x})$, or, the current posterior $\pi_m(\boldsymbol{x}) := p(\boldsymbol{x}|\mathcal{O}_m)$ is approximated by a set of N equally
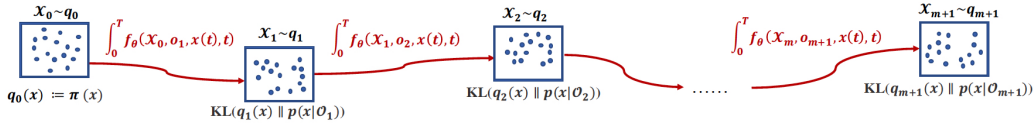
Figure 1: PFBR framework: sequential Bayesian inference as a deterministic flow of particles.

weighted particles $\mathcal{X}_m = \{\boldsymbol{x}_m^1, \ldots, \boldsymbol{x}_m^N\}$; and then, given an observation o_{m+1}, the flow operator $\mathcal{F}(\mathcal{X}_m, \boldsymbol{x}_m^n, o_{m+1})$ will transport each particle $\boldsymbol{x}_m^n$ to a new particle $\boldsymbol{x}_{m+1}^n$ to approximate the new posterior $p(\boldsymbol{x}|\mathcal{O}_{m+1})$. That is,

$$\overbrace{\boldsymbol{x}_{m+1}^n}^{\text{updated particle}} = \mathcal{F}(\mathcal{X}_m, \overbrace{\boldsymbol{x}_m^n}^{\text{current particle}}, o_{m+1}), \quad (3)$$

where $\mathcal{X}_{m+1} = \{\boldsymbol{x}_{m+1}^1, \ldots, \boldsymbol{x}_{m+1}^N\}$ will be used as samples from the new posterior $p(\boldsymbol{x}|\mathcal{O}_{m+1})$. Furthermore, this PFBR operator $\mathcal{F}$ can be applied recursively to $\mathcal{X}_{m+1}$ and a new observation o_{m+2} to produce $\mathcal{X}_{m+2}$, and so on.

In a high-level, we model the PFBR operator $\mathcal{F}$ as a continuous deterministic flow, which propagates the locations of particles and the values of their probability density simultaneously through a dynamical system described by ordinary differential equations (ODEs). A natural question is whether a fixed ODE-based Bayes operator applicable to different prior distributions and likelihood functions exists. In our paper, we resolve this important theoretical question by making a novel connection between PFBR operator and the Fokker-Planck equation of Langevin dynamics. The proof of existence also provides a basis for our parameterization of PFBR operator using DeepSet (Zaheer et al., 2017).

Similar to KBR, PFBR have a training phase and a testing phase. However, the training procedure is very different as it adopts a meta-learning framework (Andrychowicz et al., 2016), where multiple related Bayesian inference tasks with different priors and observations are created. PFBR operator $\mathcal{F}$ will learn from these tasks how to update the posteriors given new observations. During test phase, the learned PFBR will be applied directly to new observations without either re-optimization or storing previous observations. We conduct various experiments to show that the learned PFBR operator can generalize to new Bayesian inference tasks.

Related work. There is a recent surge of interests in ODE-based Bayesian inference (Chen et al., 2018; Zhang et al., 2018; Grathwohl et al., 2018; Lei et al., 2017). These works focus on fitting a single target distribution. Consequently, the learned flow can not generalize directly to a new dataset, a new prior or to sequential setting without re-optimization.

2. Bayesian Inference as Particle Flow

We present details in this section from four aspects: (1) How to map sequential Bayes inference to particle flow? (2) What is the property of such particle flow? (3) Does a shared flow-based Bayesian operator $\mathcal{F}$ exist? (4) How to

parameterize the flow operator $\mathcal{F}$?

2.1. PFBR: Particle Flow Bayes' Rule

The problem mapping from sequential Bayesian inference to particle flow goes as follows. Initially, N particles $\mathcal{X}_0 = \{\boldsymbol{x}_0^1, \ldots, \boldsymbol{x}_0^N\}$ are sampled i.i.d. from a prior $\pi(\boldsymbol{x})$. Given an observation o_1, the operator $\mathcal{F}$ will transport the particles to $\mathcal{X}_1 = \{\boldsymbol{x}_1^1, \ldots, \boldsymbol{x}_1^N\}$ to estimate the posterior $p(\boldsymbol{x}|\mathcal{O}_1) \propto \pi(\boldsymbol{x})p(o_1|\boldsymbol{x})$. We define this transformation as the solution of an ODE. That is, $\forall n$,

$$\begin{cases} \frac{d\boldsymbol{x}}{dt} = f(\mathcal{X}_0, o_1, \boldsymbol{x}(t), t), \ \forall t \in (0, T] \\ \boldsymbol{x}(0) = \boldsymbol{x}_0^n \end{cases} \xRightarrow{\text{gives}} \boldsymbol{x}_1^n = \boldsymbol{x}(T).$$

The flow velocity f takes observation o_1 as input and determines both direction and speed of the change of $\boldsymbol{x}(t)$. In this ODE model, each particle $\boldsymbol{x}_0^n$ sampled from the prior gives an initial value $\boldsymbol{x}(0)$, and then the flow velocity f will evolve the particle continuously and deterministically. At terminate time T, we will take solution $\boldsymbol{x}(T)$ as the transformed particle $\boldsymbol{x}_1^n$ for estimating the posterior.

Applying this ODE-based transformation sequentially as new observations $o_2, o_3, \ldots$ arrive, we can define a recursive particle flow Bayes operator, called PFBR, as

$$\boldsymbol{x}_{m+1}^n = \mathcal{F}(\mathcal{X}_m, o_{m+1}, \boldsymbol{x}_m^n)$$
$$:= \boldsymbol{x}_m^n + \int_0^T f(\mathcal{X}_m, o_{m+1}, \boldsymbol{x}(t), t) \, dt. \quad (4)$$

The set of obtained particles $\mathcal{X}_{m+1}$ can be used to perform Bayesian inference such as estimating the mean and quantifying the uncertainty of any test function by averaging over these particles.

At this moment, we assume f has a form of $f(\mathcal{X}, o, \boldsymbol{x}(t), t)$, and will be shared across different sequential stages. In section 2.3, a rigorous discussion on the existence of a shared flow velocity of this form will be made. Next, we will discuss further properties of this continuous particle flow which will help us study the existence of such operator for Bayesian inference, and design the parameterization and the learning for the flow velocity.

2.2. Property of Continuous Deterministic Flow

The continuous transformation of $\boldsymbol{x}(t)$ described by ODE $d\boldsymbol{x}/dt = f$ defines a *deterministic* flow for each particle. Let $q(\boldsymbol{x}, t)$ be the probability density of the continuous random variable $\boldsymbol{x}(t)$. The change of this density is also determined by f. More specifically, q follows the **continuity equation** (Batchelor, 2000):

$$\partial q(\boldsymbol{x}, t)/\partial t = -\nabla_x \cdot (qf), \quad (5)$$

where $\nabla_x \cdot$ is the divergence operator. Continuity equation is the mathematical expression for the law of *local conservation of mass* - mass can neither be created nor destroyed, nor can it "teleport" from one place to another.

Given continuity equation, one can describe the change of log-density by another differential equation (Theorem 2.1).

Theorem 2.1. *If $d\boldsymbol{x}/dt = f$, then the change in log-density follows the differential equation (Chen et al., 2018)*

$$d \log(q(\boldsymbol{x},t))/dt = -\nabla_x \cdot f. \qquad (6)$$

Since for any physical quantity $q(\boldsymbol{x},t)$, the distinguish between material derivative d/dt and partial derivative $\partial/\partial t$ is important, we clarify the definition before the proof of this theorem.

Definition 2.1. *Material derivative of $q(\boldsymbol{x},t)$ is defined as*

$$dq/dt = \partial q/\partial t + \nabla_x q \cdot d\boldsymbol{x}/dt. \qquad (7)$$

Note that dq/dt defines the rate of change of q in a given particle as it moves along its trajectory $\boldsymbol{x} = \boldsymbol{x}(t)$ in the flow, while $\partial q/\partial t$ means the rate of change of q at a particular point $\boldsymbol{x}$ that is fixed in the space.

Proof of Theorem 2.1. By continuity equation, $\frac{\partial q}{\partial t} = -\nabla_x q \cdot f - q\nabla_x \cdot f \Rightarrow \frac{dq}{dt} = -q\nabla_x \cdot f$. By chain rule, we have $\frac{d\log q}{dt} = \frac{1}{q}\frac{dq}{dt} = \frac{1}{q}(-q\nabla_x \cdot f) = -\nabla_x \cdot f$. $\qquad \square$

Theorem 2.1 gives the same result as the Instantaneous Change of Variables Theorem stated by Chen et al. (2018). However, our statement is more accurate using the notation of material and partial derivatives. Our proof is simpler and intuitively clearer using continuity equation. This also helps us to see the connection to other physics problems such as fluid dynamics and electromagnetism.

With theorem 2.1, we can compute the log-density of the particles by integrating across $(0,T]$ for each n:

$$\log q_{m+1}(x_{m+1}^n) = \log q_m(x_m^n) - \int_0^T \nabla_x \cdot f \, dt. \qquad (8)$$

2.3. Existence of Flow-based Bayes' Rule

Does a unified flow velocity f exist for different Bayesian inference tasks involving different priors and observations? If it does, what is the form of this function? These questions are non-trivial even for simple Gaussian case.

For instance, let the prior $\pi(x) = \mathcal{N}(0, \sigma_x)$ and the likelihood $p(o|x) = \mathcal{N}(x, \sigma)$ both be one dimensional Gaussian distributions. Given an observation $o = 0$, the posterior distribution is $\mathcal{N}(0, (\sigma \cdot \sigma_x)/(\sigma + \sigma_x))$. It means the ODE $d\boldsymbol{x}/dt = f$ needs to push a zero mean Gaussian distribution with covariance σ_x to another zero mean Gaussian distribution with covariance $(\sigma \cdot \sigma_x)/(\sigma + \sigma_x)$ for any σ_x. It is not clear whether such a unified flow velocity function f exists and what is the form for it.

To resolve the existence issue, we will first establish a connection between the deterministic flow in Section 2.2 and the stochastic flow: Langevin dynamics. Then we will leverage the connection between closed-loop control and open-loop control to show the existence of a unified f.

2.3.1. CONNECTION TO STOCHASTIC FLOW

Langevin dynamics is a *stochastic* process

$$d\boldsymbol{x}(t) = \nabla_x \log p(\boldsymbol{x}|\mathcal{O}_m)p(o_{m+1}|\boldsymbol{x}) \, dt + \sqrt{2}d\boldsymbol{w}(t), \qquad (9)$$

where $d\boldsymbol{w}(t)$ is a standard Brownian motion. Given a fixed initial location $\boldsymbol{x}(0)$, multiple runs of the Langevin dynamics to time t will result in multiple random locations of $\boldsymbol{x}(t)$ due to the randomness of $\boldsymbol{w}(t)$. This stochastic flow is very different in nature comparing to the deterministic flow in Section 2.2, where a fixed location $\boldsymbol{x}(0)$ will always end up with the same location $\boldsymbol{x}(t)$.

Nonetheless, while Langevin dynamics is a stochastic flow of a continuous random variable $\boldsymbol{x}(t)$, the probability density $q(\boldsymbol{x},t)$ of $\boldsymbol{x}(t)$ follows a *deterministic* evolution according to the associated **Fokker-Planck equation** (Jordan et al., 1998)

$$\partial q/\partial t = -\nabla_x \cdot (q\nabla_x \log p(\boldsymbol{x}|\mathcal{O}_m)p(o_{m+1}|\boldsymbol{x})) + \Delta_x q(\boldsymbol{x},t), \qquad (10)$$

where $\Delta_x = \nabla_x \cdot \nabla_x$ is the Laplace operator. Furthermore, if the so-called potential function $\Psi(\boldsymbol{x}) := -\log p(\boldsymbol{x}|\mathcal{O}_m)p(o_{m+1}|\boldsymbol{x})$ is smooth and $e^{-\Psi} \in L^1(\mathbb{R}^d)$, the Fokker-Planck equation has a unique stationary solution in the form of a Gibbs distribution (Jordan et al., 1998),

$$q(\boldsymbol{x},\infty) = e^{-\Psi}/Z = p(\boldsymbol{x}|\mathcal{O}_m)p(o_{m+1}|\boldsymbol{x})/Z. \qquad (11)$$

Clearly, this stationary solution is the posterior distribution $p(\boldsymbol{x}|\mathcal{O}_{m+1})$.

Now we will rewrite the Fokker-Planck equation in Eq. (10) into the form of the deterministic flow in Eq. (5) from Section 2.2, and hence identify the corresponding flow velocity.

Theorem 2.2. *Assume the deterministic transformation of a continuous random variable $\boldsymbol{x}(t)$ is $d\boldsymbol{x}/dt = f$, where*

$$f = \nabla_x \log p(\boldsymbol{x}|\mathcal{O}_m)p(o_{m+1}|\boldsymbol{x}) - \nabla_x \log q(\boldsymbol{x},t) \qquad (12)$$

and $q(\boldsymbol{x},t)$ is the probability density of $\boldsymbol{x}(t)$. If the potential function Ψ is smooth and $e^{-\Psi} \in L^1(\mathbb{R}^d)$, then $q(\boldsymbol{x},t)$ converges to $p(\boldsymbol{x}|\mathcal{O}_{m+1})$ as $t \to \infty$.

Proof. By continuity equation in Eq. (5), the probability density $q(\boldsymbol{x},t)$ of $\boldsymbol{x}(t)$ satisfies $\partial q/\partial t = -\nabla_x \cdot (q(-\nabla_x\Psi(\boldsymbol{x}) - \nabla_x \log q(\boldsymbol{x},t)))$. It is easy to see this equation is the same as the Fokker-Planck equation in Eq. (10), by decomposing the Laplace as $\Delta_x q = \nabla_x \cdot (q\nabla_x \log q)$. Under the conditions for the potential function Ψ, since Fokker-Planck equation has a unique stationary distribution $q(\boldsymbol{x},\infty)$ equal to the posterior distribution $p(\boldsymbol{x}|O_{m+1})$, the deterministic flow in Eq. (12) will also converge to $p(\boldsymbol{x}|\mathcal{O}_{m+1})$. $\qquad \square$

The implication of Theorem 2.2 is that we can construct a deterministic flow of particles to obtain the posterior and hence establish the existence. However, the flow velocity in Eq. (12) depends on the intermediate density $q(\boldsymbol{x}, t)$ which changes over time. This seemingly suggests that f can not be expressed as a fixed function of $p(\boldsymbol{x}|\mathcal{O}_m)$ and $p(o_{m+1}|\boldsymbol{x})$. In the next section, we show that this dependence on $q(\boldsymbol{x}, t)$ can be removed using theory of optimal control for deterministic systems.

2.3.2. CLOSED-LOOP TO OPEN-LOOP CONVERSION

Now the question is: whether the term $\nabla_x \log q(\boldsymbol{x}, t)$ in Eq. (12) can be made independent of $q(\boldsymbol{x}, t)$, or whether there is a equivalent form which can achieve the same flow. To investigate this question, we consider the the following *deterministic* optimal control problem

$$\min_w \quad d(q(\boldsymbol{x}, \infty), p(\boldsymbol{x}|\mathcal{O}_{m+1})) \tag{13}$$

$$\text{s.t.} \quad \frac{d\boldsymbol{x}}{dt} = \nabla_x \log p(\boldsymbol{x}|\mathcal{O}_m) p(o_{m+1}|\boldsymbol{x}) - w, \tag{14}$$

where d can be any metric defined on the set of densities over $\mathbb{R}^d$. In Eq. (13), we are optimizing over w, which is usually called the *control*. By Theorem 2.2, $w = \nabla_x \log q(\boldsymbol{x}, t)$ is apparently an optimal solution. Furthermore, the corresponding flow velocity derived by Fokker-Planck equation can be regarded as the continuous steepest descent of $\text{KL}(q(\boldsymbol{x}, t)||p(\boldsymbol{x}|\mathcal{O}_{m+1}))$ under Wasserstein distance (Jordan et al., 1998). We are seeking an alternative expression to the above optimal solution which only depends on $p(\boldsymbol{x}, \mathcal{O}_m)$ and $p(o_{m+1}|\boldsymbol{x})$. First, we introduce the terminology below from optimal control literature.

Definition 2.2. In optimal control literature, a control in a feed-back form $w = w(q(\boldsymbol{x}, t), t)$ is called **closed-loop**. In contrast, another type of control $w = w(q(\boldsymbol{x}, 0), t)$ is called **open-loop**. An open-loop control is determined when the initial state $q(\boldsymbol{x}, 0)$ is observed, whereas, a closed-loop control can adapt to the encountered states $q(\boldsymbol{x}, t)$.

Theorem 2.3. *For the optimal control problem in Eq. (13) and Eq. (14), there exists an open-loop control $w^* = w^*(q(\boldsymbol{x}, 0), t)$ such that the induced state $q^*(\boldsymbol{x}, t)$ satisfies $q^*(\boldsymbol{x}, \infty) = p(\boldsymbol{x}|\mathcal{O}_{m+1})$. Moreover, w^* has a fixed expression with respect to $p(\boldsymbol{x}|\mathcal{O}_m)$ and $p(o_{m+1}|\boldsymbol{x})$ across different m.*

Proof. By Theorem 2.2, $\tilde{w}^*(q(\boldsymbol{x}, t), t) := \nabla_x \log q(\boldsymbol{x}, t)$ can induce the optimal state $\tilde{q}^*(\boldsymbol{x}, \infty) = p(\boldsymbol{x}|\mathcal{O}_{m+1})$ and achieve a zero loss, $d = 0$. Hence, $\tilde{w}^*$ is an optimal closed-loop control for this problem.

Although in general closed-loop control has a stronger characterization to the solution, in a deterministic system like Eq. (14), the optimal closed-loop control and the optimal open-loop control will give the same control law and achieve the same optimal loss (Dreyfus, 1964). Hence, there

exists an optimal open-loop control $w^* = w^*(q(\boldsymbol{x}, 0), t)$ such that the induced state also gives a zero loss and thus $q^*(\boldsymbol{x}, \infty) = p(\boldsymbol{x}|\mathcal{O}_{m+1})$. More details are provided in Appendix A to express w^* as a fixed function of $q(\boldsymbol{x}, 0)$, $p(\boldsymbol{x}|\mathcal{O}_m)$, $p(o_{m+1}|\boldsymbol{x})$ and t. $\square$

Conclusion of a unified f. In sequential Bayesian inference, we will set $q(\boldsymbol{x}, 0)$ as $p(\boldsymbol{x}|\mathcal{O}_m)$. Therefore, Theorem 2.3 shows that there exists a fixed and deterministic flow velocity f of the form

$$\nabla_x \log p(\boldsymbol{x}|\mathcal{O}_m) p(o_{m+1}|\boldsymbol{x}) - w^*(p(\boldsymbol{x}|\mathcal{O}_m), t), \tag{15}$$

which can transform $p(\boldsymbol{x}|\mathcal{O}_m)$ to $p(\boldsymbol{x}|\mathcal{O}_{m+1})$ and in turns define a unified particle flow Bayes operator $\mathcal{F}$.

2.4. Parametrization

We design a practical parameterization of f based on the expression of the unified flow f in Eq. (15).

(i) $p(\boldsymbol{x}|\mathcal{O}_m) \Rightarrow \mathcal{X}_m$: Since we do not have full access to the density $p(\boldsymbol{x}|\mathcal{O}_m)$ but have samples $\mathcal{X}_m = \{\boldsymbol{x}_m^1, \ldots, \boldsymbol{x}_m^N\}$ from it, we can use these samples as surrogates for $p(\boldsymbol{x}|\mathcal{O}_m)$. A related example is feature space embedding of distributions (Smola et al., 2007), $\mu_{\mathcal{X}}(p) := \int_{\mathcal{X}} \phi(\boldsymbol{x}) p(\boldsymbol{x}) \, d\boldsymbol{x} \approx \frac{1}{N} \sum_{n=1}^{N} \phi(\boldsymbol{x}^n)$, $\boldsymbol{x}^n \sim p$. Ideally, if $\mu_{\mathcal{X}}$ is an injective mapping from the space of probability measures over $\mathcal{X}$ to the feature space, the resulting embedding can be treated as a sufficient statistic of the density and any information we need from $p(\boldsymbol{x}|\mathcal{O}_m)$ can be preserved. Hence, we represent $p(\boldsymbol{x}|\mathcal{O}_m)$ by $\frac{1}{N} \sum_{n=1}^{N} \phi(\boldsymbol{x}_m^n)$, where $\phi(\cdot)$ is a nonlinear feature mapping to be learned. Since we use a neural version of $\phi(\cdot)$, this representation can also be regarded as a DeepSet (Zaheer et al., 2017).

(ii) $p(o_{m+1}|\boldsymbol{x}) \Rightarrow (o_{m+1}, \boldsymbol{x}(t))$: In both Langevin dynamics and Eq. (15), the only term containing the likelihood is $\nabla_x \log p(o_{m+1}|\boldsymbol{x})$. Consequently, we can use this term as an input to f. In the case when the likelihood function is fixed, we can also simply use the observation o_{m+1}, which results in similar performance in our experiments.

Overall we parameterize the flow velocity as

$$f = \boldsymbol{h} \left(\frac{1}{N} \sum_{n=1}^{N} \phi(\boldsymbol{x}_m^n), o_{m+1}, \boldsymbol{x}(t), t \right), \tag{16}$$

where $\boldsymbol{h}$ and ϕ are neural networks (See specific architecture we use in Appendix C.1). Let $\theta \in \Theta$ be their parameters which are independent of t. From now on, we write $f = f_\theta(\mathcal{X}_m, o_{m+1}, \boldsymbol{x}(t), t)$. In the next section, we will propose a meta learning framework for learning these parameters.

3. Learning Algorithm

Since we aim to learn a generalizable Bayesian operator, we need to create multiple inference tasks as the training set and design the corresponding learning algorithm.

Multi-task Framework. The training set $\mathcal{D}_{train}$ contains multiple inference tasks. Each task is a tuple

$$\mathcal{T} := (\pi(\boldsymbol{x}), p(\cdot|\boldsymbol{x}), \mathcal{O}_M := \{o_1, \ldots, o_M\}) \in \mathcal{D}_{train}$$

which consists of a prior distribution, a likelihood function and a sequence of M observations. A task with M sequential observations can also be interpreted as a sequence of M sub-tasks with 1 observation:

$$\tau := (p(\boldsymbol{x}|\mathcal{O}_m), p(\cdot|\boldsymbol{x}), o_{m+1}) \in (\pi(\boldsymbol{x}), p(\cdot|\boldsymbol{x}), \mathcal{O}_M).$$

Therefore, each task is a sequential Bayesian inference and each sub-task corresponds to one step Bayesian update.

Cumulative Loss Function. For each sub-task we define a loss $\text{KL}(q_m(\boldsymbol{x})\|p(\boldsymbol{x}|\mathcal{O}_{m+1}))$, where $q_m(\boldsymbol{x})$ is the distribution transported by $\mathcal{F}$ at m-th stage and $p(\boldsymbol{x}|\mathcal{O}_{m+1})$ is the target posterior (see Fig. 1 for illustration). Meanwhile, the loss for the corresponding sequential task will be $\sum_{m=1}^{M} \text{KL}(q_m(\boldsymbol{x})\|p(\boldsymbol{x}|\mathcal{O}_m))$, which sums up the losses of all intermediate stages. Since its optimality is independent of normalizing constants, it is equivalent to minimize the negative evidence lower bound (ELBO)

$$\mathcal{L}(\mathcal{T}) = \sum_{m=1}^{M} \sum_{n=1}^{N} \left(\log q_m(\boldsymbol{x}_m^n) - \log p(\boldsymbol{x}_m^n, \mathcal{O}_m)\right). \quad (17)$$

The above expression is an empirical estimation using particles $\boldsymbol{x}_m^n$. In each iteration during training phase, we will randomly sample a task from $\mathcal{D}_{train}$ and update the PFBR operator $\mathcal{F}$ by the loss gradient.

3.1. Training Tasks Creation

Similar to some meta learning problems, the distribution of training tasks will essentially affect learner's performance on testing tasks (Dai et al., 2017). Depending on the nature of the Bayesian problem, we propose two approaches to construct the multitask training set: one is data driven, and the other is based on generative models. The general principle is that the collection of training priors have to be diverse enough or representative of those may be seen in testing time.

Data-Driven Approach. We can use the posterior distribution obtained from last Bayesian inference step as the new prior distribution. If the posterior distribution has an analytical (but unnormalized) expression, we will directly use this expression.

More precisely, since each Bayesian inference step will generate a set of particles, $\mathcal{X}_m = \{\boldsymbol{x}_m^1, \ldots, \boldsymbol{x}_m^N\}$, corresponding to the posterior distribution $p(\boldsymbol{x}|\mathcal{O}_m)$, we can apply a kernel density estimator on top of these samples to obtain an empirical density function

$$\hat{\pi}(\boldsymbol{x}; \mathcal{X}_m) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{\sqrt{2\pi}\sigma^d} e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x}_m^n\|^2}{2\sigma^2}}, \quad (18)$$

where σ is the kernel bandwidth. Then this density function

Algorithm 1 Overall Learning Algorithm
___
$\theta \leftarrow$ random initialization
For $itr = 1$ *to #iterations* **do**
    $\mathcal{T} = (\pi(\boldsymbol{x}), p(\cdot|\boldsymbol{x}), \mathcal{O}_M) \leftarrow$ sampled from $\mathcal{D}_{train}$
    $\mathcal{X}_0 = \{\boldsymbol{x}_0^n\}_{n=1}^{N} \overset{i.i.d.}{\sim} \pi(\boldsymbol{x})$ ▷ initial particles
    $\theta \leftarrow \theta - \eta \, \text{Grad}(\theta, \mathcal{X}_0, \pi(\boldsymbol{x}), p(\cdot|\boldsymbol{x}), \mathcal{O}_M)$
    if $\text{mod}(itr, k) = 0$ **then**
        Perform a validation step on $\mathcal{D}_{vali}$ ▷ validation
return best θ^* in validation steps
___
Remark. See Appendix B for both derivation and algorithm steps of $\text{Grad}()$.

---

with a set of samples from it will be used as the new prior for the next training task. This approach has two attractive properties. First, it does not require any prior knowledge of the model and thus is generally applicable to most problems. Second, it provides us a way of breaking a long sequence $(\pi(\boldsymbol{x}), p(\cdot|\boldsymbol{x}), \mathcal{O}_M)$ with large M into multiple tasks with shorter sequences

$$(\pi(\boldsymbol{x}), p(\cdot|\boldsymbol{x}), o_{1:m}) \cup (\hat{\pi}(\boldsymbol{x}; \mathcal{X}_m), p(\cdot|\boldsymbol{x}).o_{m+1:M}), \quad (19)$$

This will help make the training procedure more efficient. This approach will be particularly suited to sequential Bayesian inference and is used in later experiments.

Generative Model Approach. Another possibility is to sample priors from a flexible generative model, such as a Dirichlet process Gaussian mixture model (Antoniak, 1974). We will leave the experimental evaluation of this approach for future investigation.

3.2. Overall Learning Algorithm

Learning the PFBR operator $\mathcal{F}$ is learning the parameters θ of f_θ to minimize the following loss for multiple tasks

$$\mathcal{L}(\mathcal{D}_{train}) = \frac{1}{|\mathcal{D}_{train}|} \sum_{\mathcal{T} \in \mathcal{D}_{train}} \mathcal{L}(\mathcal{T}),$$

where $\mathcal{L}(\mathcal{T})$ is the loss for a single task defined in Eq. (17). Both the particle location $\boldsymbol{x}_m^n$ and its density value $q_m(\boldsymbol{x}_m^n)$ in $\mathcal{L}(\mathcal{T})$ are results of forward propagation determined by f_θ according to ODEs in Eq. (4) and Eq. (8). The training procedure is similar to other deep learning optimizations, except that an ODE technique called adjoint method (Chen et al., 2018) is used to compute the gradient $d\mathcal{L}/d\theta$ with very low memory cost. The overall algorithm under a meta-learning framework is summarized in Algorithm 1.

3.3. Efficient Learning Tricks

We introduce two techniques to improve training efficiency for large scale problems. Its application to an experiment which contains millions of data points is demonstrated in Section 4. These two techniques can be summarized as **mini-batch embedding** and **sequence-segmentation**.

The loss function $\mathcal{L}(\mathcal{T})$ in Eq. (17) contains a summation from $m = 1$ to M and also a hidden inner-loop summation

$$\log p(\boldsymbol{x}_m^n, \mathcal{O}_m) = \log \pi(\boldsymbol{x}_m^n) + \sum_{t=1}^{m} \log p(o_t | \boldsymbol{x}_m^n). \quad (20)$$

Thus the evaluation cost of $\mathcal{L}(\mathcal{T})$ is quadratic with respect to the length M of the observation sequences. Therefore, we need to reduce the length for large scale problems.

Mini-batch embedding. Previously we defined o_m as a single observation. However, we can also view it as a batch of L observations, i.e., $o_m = \{o_m^1, \ldots, o_m^L\}$. Each Bayesian update corresponding to this mini-batch will become $p(\boldsymbol{x} | \mathcal{O}_{m+1}) \propto p(\boldsymbol{x} | \mathcal{O}_m) \prod_{l=1}^{L} p(o_m^l | \boldsymbol{x})$. If we rewrite $p(o_m | \boldsymbol{x}) = \prod_{l=1}^{L} p(o_m^l | \boldsymbol{x})$, this is essentially the same as our previous expression. Therefore, we can replace o_m by $\{o_m^l\}_{l=1}^{L}$ and input these samples simultaneously as a set to the flow f_θ. To reduce the input dimension, we resort to a set-embedding $\frac{1}{L} \sum_{l=1}^{L} g(o_m^l)$, where g is a neuralized nonlinear function to be learned. Depending on the structure of the model, one define the embedding as a Deepset (Zaheer et al., 2017), or, if the posterior is not invariant with respect to the order of the observations (e.g. hidden Markov models), one need to use a set-embedding that is not order invariant. To conclude, for a mini-batch Bayesian update, the parameterization of flow velocity can be modified as

$$f = \boldsymbol{h}\left(\frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}_m^n), \frac{1}{L}\sum_{l=1}^{L}g(o_{m+1}^l), \boldsymbol{x}(t), t\right).$$

Sequence-segmentation. We will use the approach in Eq. (19) to break a long sequence into short ones. More precisely, suppose we have particles $\{\boldsymbol{x}_{m^*}^n\}_{n=1}^{N}$ at m^*-th stage. We can cut the sequence at position m^* and generate a new task using the second half. The prior for the new sequence will be an empirical density estimation $\hat{\pi}(\boldsymbol{x}; \mathcal{X}_{m^*})$ as defined in Eq. (18). Then, for all stages $m > m^*$, the terms in Eq. (20) becomes

$$\log p(\boldsymbol{x}, \mathcal{O}_m) \approx \log \hat{\pi}(\boldsymbol{x}_m^n; \mathcal{X}_{m^*}) + \sum_{t=m^*+1}^{m} \log p(o_t | \boldsymbol{x}_m^n).$$

We can apply this technique for multiple times to split a long sequence into multiple segments.

4. Experiments

We conduct experiments on multivariate Gaussian model, hidden Markov model and Bayesian logistic regression to demonstrate the generalization ability of PFBR and also its accuracy for posterior estimation.

Evaluation metric. For multivariate Gaussian model and Gaussian linear dynamical system, we could calculate the true posterior. Therefore, we can evaluate:
(i) Cross-entropy $\mathbb{E}_{x \sim p} - \log q(x)$;
(ii) Maximum mean discrepancy (Gretton et al., 2012)

$$\text{MMD}^2 = \mathbb{E}_{x,x' \sim p;\, y,y' \sim q}[k(x,x') - 2k(x,y) + k(y,y')];$$

(iii) Integral evaluation discrepancy $\|\mathbb{E}_p h(x) - \mathbb{E}_q h(x)\|_2$;

where we use Monte Carlo method to compute $\mathbb{E}_{x \sim p}[\cdot]$ for the first two metrics. For the integral evaluation, we choose some test functions h where the exact value of $\mathbb{E}_p h(x)$ has a closed-form expression. For the experiments on real-world dataset, we estimate the commonly used prediction accuracy due to the intractability of the posterior.

Multivariate Gaussian Model. The prior $\boldsymbol{x} \sim \mathcal{N}(\mu_x, \Sigma_x)$, the observation conditioned on prior $o | \boldsymbol{x} \sim \mathcal{N}(\boldsymbol{x}, \Sigma_o)$ and the posterior all follow Gaussian distributions. In our experiment, we use $\mu_x = \vec{0}, \Sigma_x = I$ and $\Sigma_o = 3I$. We test the learned PFBR on different sequences of 100 observations $\mathcal{O}_{100}$, while the training set only contains sequences of 10 observations, which are ten times shorter than sequences in the testing set. However, since we construct a set of different prior distributions to train the operator, the diversity of the prior distributions allows the learned $\mathcal{F}$ to generalize to compute posterior distributions in a longer sequence.

We compare the performances with KBR (Fukumizu et al., 2012) and one-pass SMC (Balakrishnan & Madigan, 2006). Both PFBR and KBR are learned from the training set and then used as algorithms on the testing set, which consists of 25 sequences of 100 observations $\{\mathcal{O}_{100}^j \sim \mathcal{N}(\boldsymbol{x}^j, \Sigma_o)\}_{j=1}^{25}$ conditioned on 25 different $\boldsymbol{x}^j$ sampled from $\mathcal{N}(\boldsymbol{0}, I)$. We compare estimations of $p(\boldsymbol{x} | \mathcal{O}_m^j)$ across stages from $m = 1$ to 100 and the results are plotted in Fig. 2. Since KBR's performance reveals to be less comparable in this case, we leave its results to Appendix D. We can see from Fig. 2 that as the dimension of the model increases, our PFBR has more advantages over one-pass SMC.

Gaussian Mixture Model. Following the same setting as Welling & Teh (2011), we conduct an experiment on an interesting mixture model, where the observations $o \sim \frac{1}{2}(\mathcal{N}(x_1, 1.0) + \mathcal{N}(x_1 + x_2, 1.0))$ and the prior $x_1, x_2 \sim \mathcal{N}(0, 1)$. The same as Dai et al. (2016), we set $(x_1, x_2) = (1, -2)$ so that the resulting posterior will have two modes: $(1, -2)$ and $(-1, 2)$. During training, multiple sequences are generated, each of which consists of 30 observations.

Compared to Welling & Teh (2011) and Dai et al. (2016), our experiment is more challenging. First, they are only fitting one posterior given one sequence of observations via optimization, while PFBR will operate on sequences that are NOT observed during training without re-optimization. Second, they are only estimating the final posterior, while we aim at fitting every intermediate posteriors. Even for one fixed sequence, it is not easy to fit the posterior and capture both two modes, which can be seem from the results reported by Dai et al. (2016). However, our learned PFBR can operate on testing sequences and the resulting posteriors look closed enough to true posterior (See Fig. 3).

Hidden Markov Model — LDS. Our PFBR method can be easily adapted to hidden Markov models, where we will esti-
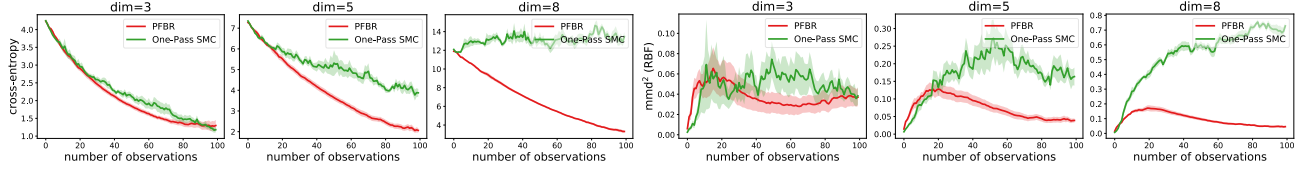
(a) Cross entropy $\mathbb{E}_{p(\boldsymbol{x}|\mathcal{O}_m)} - \log q_m$ for dimension 3, 5 and 8 (b) Squared MMD with RBF kernel for dimension 3, 5 and 8
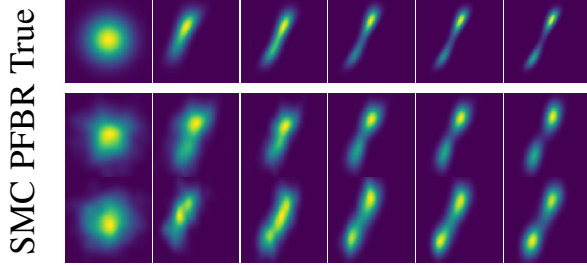
Figure 2: Experimental results for Gaussian model. We use 256 obtained particles as samples from $p(\boldsymbol{x}|\mathcal{O}_m)$ and compare it with true posteriors. Each evaluation result is computed over 25 tasks and the shaded area shows the standard error. More results in Appendix D.

Figure 3: Visualization of the evolution of posterior density from left to right. In the end, PFBR density is closer to the true density than SMC density.

mate the marginal posterior distribution $p(\boldsymbol{x}_m|\mathcal{O}_m)$. For instance, consider a Gaussian linear dynamical system (LDS):

$$\boldsymbol{x}_m = A\boldsymbol{x}_{m-1} + \epsilon_m, \quad o_m = B\boldsymbol{x}_m + \delta_m,$$

where $\epsilon_m \sim \mathcal{N}(0, \Sigma_1)$ and $\delta_m \sim \mathcal{N}(0, \Sigma_2)$. The particles can be updated by recursively applying two steps:

(i) $\tilde{\boldsymbol{x}}_m^n = A\boldsymbol{x}_{m-1}^n + \epsilon_m$, (ii) $\boldsymbol{x}_m^n = \mathcal{F}(\tilde{\mathcal{X}}_m, \tilde{\boldsymbol{x}}_m^n, o_{m+1})$,

where $\tilde{\mathcal{X}}_m := \{\tilde{\boldsymbol{x}}_m^n\}_{n=1}^N$. The second step is a Bayesian update from $p(\boldsymbol{x}_m|\mathcal{O}_{m-1})$ to $p(\boldsymbol{x}_m|\mathcal{O}_m)$ given the likelihood $p(o_m|\boldsymbol{x}_m)$. The only tricky part is we do not have a tractable loss function in this case because of the integration $p(\boldsymbol{x}_m|\mathcal{O}_{m-1}) = \int p(\boldsymbol{x}_m|\boldsymbol{x}_{m-1})p(\boldsymbol{x}_{m-1}|\mathcal{O}_{m-1})\, d\boldsymbol{x}_{m-1}$. Hence, at each stage m, we use the particles $\tilde{\mathcal{X}}_m$ to construct an empirical density estimation $\hat{\pi}(\boldsymbol{x}; \tilde{\mathcal{X}}_m)$ as defined in Eq. (18) and then define the loss at each stage m as

$$\sum_{n=1}^N \log q_m(\boldsymbol{x}_m^n) - \log p(o_m|\boldsymbol{x}_m^n) - \log \hat{\pi}(\boldsymbol{x}_m^n; \tilde{\mathcal{X}}_m),$$

where we replace the intractable density $p(\boldsymbol{x}_m|\mathcal{O}_{m-1})$ by $\hat{\pi}$. Given this loss, the PFBR operator $\mathcal{F}$ can be learned.

In the experiment, we sample a pair of 2 dimensional random matrices A and B for the LDS model. We learn both PFBR and KBR from a training set containing multiple different sequences of observations. For KBR, we use an adapted version called Kernel Monte Carlo Filter (Kanagawa et al., 2013), which is designed for state-space models. We use Kalman filter (Welch & Bishop, 2006) to compute the true marginal posterior for evaluation purpose.
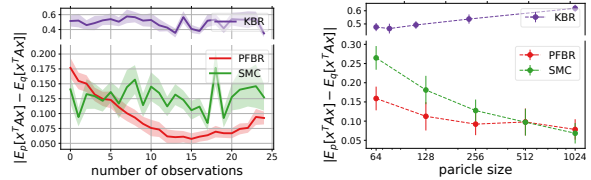
Fig. 4 compares our method with KBR and SMC (importance sampling with resampling) on a testing set containing 25 new sequences of ordered observations. We see that our learned PFBR can generalize to test sequences and achieve better and stabler performances.
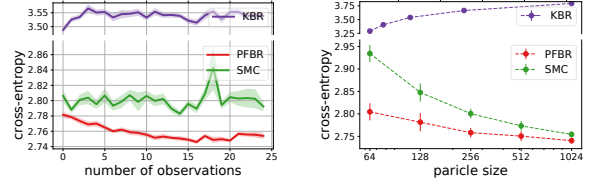
(a) Comparison of cross entropy
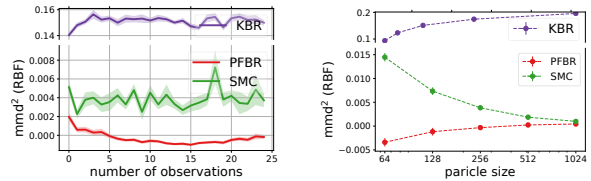
(b) Squared MMD with RBK kernel

(c) Integral estimation on test function

Figure 4: Experimental results for LDS. Only results evaluated on the testing set are shown. *Left:* estimation errors on every stage $m \in [25]$. *Right:* estimation errors for different particle sizes, which are firstly averaged over 25 stages for each task, and then averaged over 25 tasks. The error bar shows the standard error over tasks. We use the same PFBR operator trained with 1024 particles, even though during testing phase, we apply it on particles of difference sizes. See Appendix D for more results.

Comparison to Variational SMC. Autoencoding SMC (AESMC), Filtering Variational Objectives, and Variational SMC are three recent variational inference approaches that approximate the posterior based on SMC and a neural proposal (Le et al., 2018; Maddison et al., 2017; Naesseth et al., 2018). Since they share similar ideas, we implemented AESMC as a representative. We tried both MLP and GRU as mentioned in these papers. A comparison is made for 10-dimensional LDS (Table 1), which shows PFBR is better even with much fewer particles.

Inference Time Comparison. Table 1 also shows the inference time for updating posterior given one new observation. Though PFBR takes more time for the same #particles (e.g.256), to get closer to PFBR's performance, others need to use much more particles (e.g. 4096).

Bayesian Logistic Regression (BLR). We consider logistic regression for digits classification on the MNIST8M 8

Algo	#particles	cpu time (s)	gpu time (s)	cross-entropy
PFBR	256	0.23	0.26	16.56
SMC	256	0.07	0.02	26.78
ASMC-mlp	256	0.17	0.07	19.66
ASMC-gru	256	0.18	0.07	19.38
ASMC-mlp	4096	2.23	0.25	17.63
ASMC-gru	4096	2.26	0.26	17.24
SMC	8192	3.87	0.12	17.60

Table 1: Numbers are averaged over 25 sequences with 25 observations each. For PFBR, gpu is faster when #particles is larger.

vs. 6 dataset which contains about 1.6M training samples and 1932 testing samples. We reduce the dimension of the images to 50 by PCA, following the same setting as Dai et al. (2016). Two experiments are conducted on this dataset. For both experiments, we compare our method with SMC, SVI (stochastic variational inference (Hoffman et al., 2012)), PMD (particle mirror descent (Dai et al., 2016)), SGD Langevin (stochastic gradient Langevin dynamics (Welling & Teh, 2011)) and SGD NPV (stochastic version of nonparametric variational inference (Gershman et al., 2012)). This is a large dataset, so we use the techniques discussed in Section 3.3 to facilitate training efficiency.

BLR-Meta Learning. In the first experiment, we create a multi-task environment by rotating the first and second components of the features reduced by PCA through an angle ψ uniformly sampled from $[-15°, 15°]$. Note that the first two components account for more variability in the data. With a different rotation angle ψ, the classification boundary will change and thus a different classification task will be created. Also, a different sequence of image samples will result in different posterior distributions and thus corresponds to a different inference task.

We learn the operator $\mathcal{F}$ from a set of training tasks, where each task corresponds to a different rotation angle ψ. After that, we use $\mathcal{F}$ as Bayes' Rule for testing tasks and compare its performances with other stochastic methods or sampling methods. Test is done in an online fashion: all algorithms start with a set of particles sampled from the prior (hence the prediction accuracy at 0-th step is around 0.5). Each algorithm will make a prediction to the encountered batch of 32 images, and then observe their true labels. After that, each algorithm will update the particles and make a prediction to the next batch of images. Ideally we should compare the estimation of posteriors. However, since it is intractable, we evaluate the average prediction accuracy at each stage. Results are shown in Fig. 5.

Note that we have conducted a sanity check to confirm the learned operator $\mathcal{F}$ does not ignore the first 2 rotated dimensions and use the rest 48 components to make predictions. More precisely, if we zero out the first two components of the data and learn $\mathcal{F}$ on them. The accuracy of the particles dropps to around 65%. This further verifies that the learned PFBR indeed can generalize across different tasks.
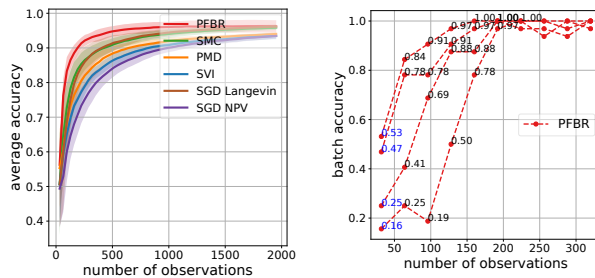

Figure 5: Bayesian logistic regression on MNIST. *Left*: The average online prediction accuracy $\frac{1}{m}\sum_{t=1}^{m} r_t$ is evaluated, where r_t is the accuracy for the t-th batch of images. The shaded area presents standard deviation of results over 10 testing tasks. *Right*: We collect some examples when the random initialization is farther from the posterior and gives worse initial prediction. PFBR $\mathcal{F}$ updates those particles to gradually achieve a higher accuracy.

BLR-Variational Inference. For the second experiment on MNIST, we use PFBR as a variational inference method. That is to say, instead of accurately learning a generalizable Bayesian operator, the estimation of the posterior $p(\boldsymbol{x}|\mathcal{O}_{train})$ is of more importance. Thus here we do not use the loss function in Eq. (17) summing over all intermediate states, but emphasize more on the final error $\text{KL}(q(\boldsymbol{x})||p(\boldsymbol{x}|\mathcal{O}_{train}))$. After the training is finished, we only use the transported particles to perform classification on the test set but do not further use the operator $\mathcal{F}$. The result is shown in Fig. 6, where x-axis shows number of visited samples during training. Since we use a batch size of 128 and consider 10 stages, the first gradient step of our method starts after around 10^3 samples are visited.
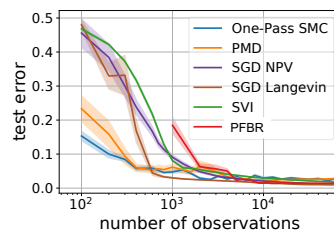

Figure 6: PFBR as a variational inference method. The prediction accuracy of PFBR is comparable with state-of-art variational inference and sampling methods.

5. Conclusion and Future Work

In this paper, we have explored the possibility of learning an ODE-based Bayesian operator that can perform online Bayesian inference in testing phase and verified its generalization ability through both synthetic and real-world experiments. Further investigation on the parameterization of flow velocity f (e.g. use a stable neural architecture (Haber & Ruthotto, 2018) with HyperNetwork (Ha et al., 2017)) and generating diverse prior distributions through a Dirichlet process can be made to explore a potentially better solution to this challenging problem.

Acknowledgements

References

Andrieu, C., de Freitas, N., Doucet, A., and Jordan, M. I. An introduction to mcmc for machine learning. *Machine Learning*, 50:5–43, 2003.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., and de Freitas, N. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pp. 3981–3989, 2016.

Antoniak, C. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *Annals of Statistics*, 2:1152–1174, 1974.

Balakrishnan, S. and Madigan, D. A one-pass sequential monte carlo method for bayesian analysis of massive datasets. *Bayesian Analysis*, 1(2):345–361, 06 2006.

Batchelor, G. *Kinematics of the Flow Field*, pp. 71–130. Cambridge Mathematical Library. Cambridge University Press, 2000.

Canini, K. R., Shi, L., and Griff iths, T. L. Online inference of topics with latent dirichlet allocation. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 6572–6583. Curran Associates, Inc., 2018.

Chopin, N., Jacob, P. E., and Papaspiliopoulos, O. Smc2: an efficient algorithm for sequential analysis of state space models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):397–426, 2013.

Dai, B., He, N., Dai, H., and Song, L. Provable bayesian inference via particle mirror descent. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pp. 985–994, 2016.

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., and Song, L. Learning combinatorial optimization algorithms over graphs. *arXiv preprint arXiv:1704.01665*, 2017.

Daum, F. and Huang, J. Curse of dimensionality and particle filters. In *2003 IEEE Aerospace Conference Proceedings*, volume 4, 2003.

Doucet, A., de Freitas, N., and Gordon, N. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

Dreyfus, S. Some types of optimal control of stochastic systems. *Journal of the Society for Industrial and Applied Mathematics Series A Control*, 2(1):120–134, 1964. doi: 10.1137/0302010.

Fukumizu, K., Song, L., and Gretton, A. Kernel Bayes' rule: Bayesian inference with positive definite kernels. In *accepted to Journal of Machine Learning Research (JMLR)*, 2012.

Gershman, S., Hoffman, M., and Blei, D. M. Nonparametric variational inference. In Langford, J. and Pineau, J. (eds.), *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pp. 663–670, New York, NY, USA, 2012. ACM.

Grathwohl, W., Chen, R. T., Betterncourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Gretton, A., Borgwardt, K., Rasch, M., Schoelkopf, B., and Smola, A. A kernel two-sample test. *JMLR*, 13:723–773, 2012.

Ha, D., Dai, A., and Le, Q. V. HyperNetworks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Haber, E. and Ruthotto, L. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, 2018.

Hoffman, M., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. In *International Conference on Machine Learning*, 2012.

Jordan, R., Kinderlehrer, D., and Otto, F. The variational formulation of the fokker–planck equation. *SIAM Journal on Mathematical Analysis*, 29(1):1–17, 1998.

Kanagawa, M., Nishiyama, Y., Gretton, A., and Fukumizu, K. Filtering with state-observation examples via kernel monte carlo filter. *arXiv e-prints*, December 2013.

Le, T. A., Igl, M., Rainforth, T., Jin, T., and Wood, F. Autoencoding sequential monte carlo. In *International Conference on Learning Representations*, 2018.

Lei, N., Su, K., Cui, L., Yau, S.-T., and Xianfeng Gu, D. A Geometric View of Optimal Transportation and Generative Model. *arXiv e-prints*, art. arXiv:1710.05488, October 2017.

Maddison, C. J., Lawson, J., Tucker, G., Heess, N., Norouzi, M., Mnih, A., Doucet, A., and Teh, Y. Filtering variational objectives. In *Advances in Neural Information Processing Systems*, pp. 6573–6583, 2017.

Naesseth, C., Linderman, S., Ranganath, R., and Blei, D. Variational sequential monte carlo. In *International Conference on Artificial Intelligence and Statistics*, 2018.

Smola, A., Gretton, A., Song, L., and Schölkopf, B. A hilbert space embedding for distributions. In *Algorithmic learning theory*, pp. 13–31. Springer, 2007.

Snyder, C., Bengtsson, T., Bickel, P., and Anderson, J. Obstacles to high-dimensional particle filtering. *Monthly Weather Review*, 136(12):4629–4640, 2008.

Wainwright, M. J. and Jordan, M. I. Graphical models, exponential families, and variational inference. Technical Report 649, UC Berkeley, Department of Statistics, September 2003.

Welch, G. and Bishop, G. An introduction to the kalman filter. Technical Report TR-95-041, Department of Computer Science, University of North Carolina at Chapel Hill, 2006.

Welling, M. and Teh, Y.-W. Bayesian learning via stochastic gradient langevin dynamics. In *International Conference on Machine Learning (ICML)*, pp. 681–688, 2011.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In *Advances in Neural Information Processing Systems*, pp. 3391–3401, 2017.

Zhang, L., E, W., and Wang, L. Monge-amp\ere flow for generative modeling. *arXiv preprint arXiv:1809.10188*, 2018.