FEDERATED GENERALIZED BAYESIAN LEARNING VIA DISTRIBUTED STEIN VARIATIONAL GRADIENT DESCENT

Anonymous authors
Paper under double-blind review

ABSTRACT

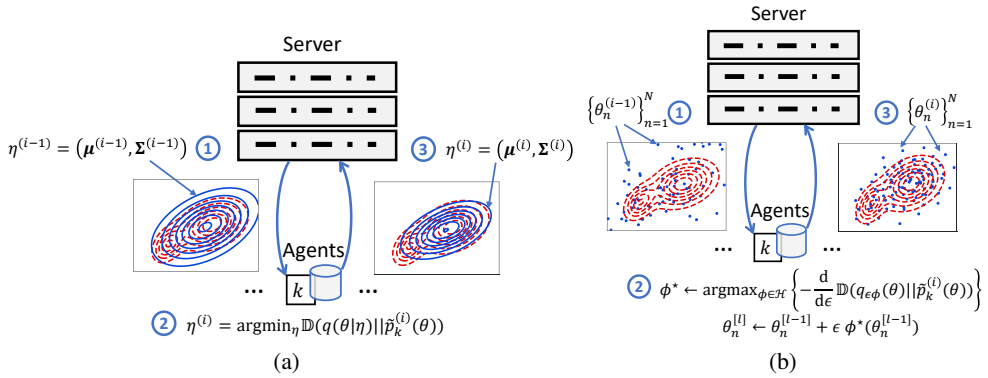
This paper introduces Distributed Stein Variational Gradient Descent (DSVGD), a non-parametric generalized Bayesian inference framework for federated learning. DSVGD maintains a number of non-random and interacting particles at a central server to represent the current iterate of the model global posterior. The particles are iteratively downloaded and updated by one of the agents with the end goal of minimizing the global free energy. By varying the number of particles, DSVGD enables a flexible trade-off between per-iteration communication load and number of communication rounds. DSVGD is shown to compare favorably to benchmark frequentist and Bayesian federated learning strategies, also scheduling a single device per iteration, in terms of accuracy and scalability with respect to the number of agents, while also providing well-calibrated, and hence trustworthy, predictions.

1 INTRODUCTION

The performance of federated learning with mobile or embedded devices is constrained by the limited availability of data at each device and by the delay caused by the need to process and exchange model information across multiple iterations. This delay depends in large part on the number of communication rounds (Lin et al., 2020). Furthermore, in applications such as personal health assistants, the learning agents' recommendations need to be reliable and trustworthy, e.g., to decide when to contact a doctor in case of a possible emergency. In this context, this paper is motivated by the following question: *Can federated learning protocols be designed to enable (i) a flexible trade off between per-iteration communication load and number of iterations (and hence communication rounds), while (ii) providing well-calibrated, and hence trustworthy, decision?* We adopt **federated generalized Bayesian learning** as a framework to develop protocols that meet these requirements.

Generalized Bayesian learning aims at minimizing a free energy function over a posterior distribution of the model parameters, including standard Bayesian learning as a special case (Knoblauch et al., 2019). Existing (generalized) Bayesian federated learning protocols are either based on **Variational Inference (VI)** (Angelino et al., 2016; Neiswanger et al., 2015; Broderick et al., 2013; Corinzia & Buhmann, 2019b) or **Monte Carlo (MC) sampling** (Ahn et al., 2014; Mesquita et al., 2020; Wei & Conlon, 2019). State-of-the-art methods in either category include **Partitioned Variational Inference (PVI)**, which has been recently introduced as a unifying distributed VI framework that relies on the optimization over parametric posteriors; and **Distributed Stochastic Gradient Langevin Dynamics (DSGLD)**, which is an MC sampling technique that maintains a number of Markov chains updated via local Stochastic Gradient Descent (SGD) with the addition of Gaussian noise (Ahn et al., 2014; Welling & Teh, 2011). The performance of VI-based protocols is generally limited by the bias entailed by the variational approximation, while MC sampling suffers from slow convergence (Angelino et al., 2016).

Stein Variational Gradient Descent (SVGD) has been introduced in (Liu & Wang, 2016) as a non-parametric Bayesian framework that approximates a target posterior distribution via non-random and interacting particles. SVGD inherits the flexibility of non-parametric Bayesian inference methods, while improving the convergence speed of MC sampling (Liu & Wang, 2016). By controlling the number of particles, SVGD can provide flexible performance in terms of bias, convergence speed, and per-iteration complexity. This paper introduces a novel non-parametric distributed learning



(a) (b)

Figure 1: Federated learning across K agents equipped with local datasets and assisted by a central server: (a) in PVI (Bui et al., 2018) agents exchange with the server parameters η representing the model posterior with the server, while (b) in DSVGD agents exchange particles $\{\theta_n\}_{n=1}^N$ providing a non-parametric estimate of the posterior.

algorithm, termed **Distributed Stein Variational Gradient Descent (DSVGD)**, that transfers the mentioned benefits of SVGD to federated learning.

As illustrated in Fig. 1, DSVGD targets a generalized Bayesian learning formulation, and maintains a number of non-random and interacting particles at a central server, typically hosted on an access point, that represent the current iterate of the global posterior. At each iteration, the particles are downloaded and updated by one of the agents by minimizing a local free energy functional before being uploaded to the server. DSVGD is shown to enable (*i*) a trade-off between per-iteration communication load and number of communication rounds by varying the number of particles, while (*ii*) being able to make trustworthy decisions through Bayesian inference.

Notation: For a probability distribution $p(x)$ and a possibly unnormalized distribution $\tilde{q}(x)$, the Kullback–Leibler (KL) divergence is defined as $\mathbb{D}(p(x)||\tilde{q}(x)) = \mathbb{E}_{x\sim p(x)}[\log(p(x)/\tilde{q}(x))]$.

2 System Set-up

We consider the federated learning set-up in Fig. 1, where each agent $k = 1, \ldots, K$ has a distinct local dataset with associated training loss $L_k(\theta)$ for model parameter θ. The agents communicate through a central node with the goal of computing the global generalized posterior distribution $q(\theta)$ over the shared model parameter $\theta \in \mathbb{R}^d$ for some prior distribution $p_0(\theta)$ (Angelino et al., 2016). Specifically, following the generalized Bayesian learning framework, the agents aim at obtaining the distribution $q(\theta)$ that minimizes the **global free energy** (Knoblauch et al., 2019)

$$\min_{q(\theta)} \left\{ F(q) = \sum_{k=1}^{K} \mathbb{E}_{\theta\sim q(\theta)}[L_k(\theta)] + \alpha \mathbb{D}(q(\theta)||p_0(\theta)) \right\}, \tag{1}$$

where $\alpha > 0$ is a temperature parameter. The generalized global posterior $q_{opt}(\theta)$ solving problem (1) must strike a balance between minimizing the sum loss function (first term in $F(q)$) and the model complexity defined by the divergence from a reference prior (second term in $F(q)$). The global free energy can also be written as

$$F(q) = \alpha \mathbb{D}(q(\theta)||\tilde{q}_{opt}(\theta)), \tag{2}$$

with the unnormalized **generalized global posterior**

$$\tilde{q}_{opt}(\theta) = p_0(\theta) \exp\Big(-\frac{1}{\alpha} \sum_{k=1}^{K} L_k(\theta) \Big) = (p_0(\theta))^{1-K} \prod_{k=1}^{K} \tilde{p}_k(\theta), \tag{3}$$

where we denoted as $\tilde{p}_k(\theta) = p_0(\theta) \exp(-\frac{1}{\alpha} L_k(\theta))$ the unnormalized **local generalized posterior** at agent k. This implies that the optimal solution to problem (1) is given by the (normalized) global generalized posterior $q_{opt}(\theta) \propto \tilde{q}_{opt}(\theta)$.

The main challenge in computing the optimal posterior $q_{opt}(\theta)$ in a distributed manner is that each agent k is only aware of its local loss $L_k(\theta)$. By exchanging information through the server, the K

agents wish to obtain an estimate of the global posterior (3) without disclosing their local datasets neither to the server nor to the other agents. In this paper, we introduce a novel non-parametric distributed generalized Bayesian learning framework that addresses this challenge by integrating PVI (Bui et al., 2018) and SVGD (Liu & Wang, 2016).

3 BACKGROUND

We review the two main techniques we build on, namely PVI (Bui et al., 2018) and SVGD (Liu & Wang, 2016). Their algorithmic tables can be found in Sec. A.1 in the supplementary materials.

Partitioned Variational Inference (PVI). PVI starts from the observation that the unnormalized posterior (3) factorizes as the product of distinct contributions from the agents. Accordingly, in its most typical form, PVI tackles problem (1) over the space of distributions that factorize as

$$q(\theta|\eta) = p_0(\theta|\eta_0) \prod_{k=1}^{K} t_k(\theta|\eta_k), \tag{4}$$

where prior $p_0(\cdot|\eta_0) = \text{EXPFAM}(\cdot|\eta_0)$ and **approximate likelihood** $t_k(\cdot|\eta_k) = \text{EXPFAM}(\cdot|\eta_k)$ for agent k are selected from the same exponential family distribution with natural parameters η_0 and η_k, respectively. Note that the approximate likelihoods are generally unnormalized for (4) to hold. As a result of the factorization (4), the approximate generalized posterior $q(\theta|\eta) = \text{EXPFAM}(\cdot|\eta)$ has the same distribution with global natural parameters $\eta = \eta_0 + \sum_{k=1}^{K} \eta_k$.

As illustrated in Fig. 1(a), at the beginning of each iteration $i = 1, 2, \ldots$, the server maintains current global iterate $\eta^{(i-1)}$ and each agent k stores its current local iterate $\eta_k^{(i-1)}$. PVI schedules an agent $k \in \{1, 2, \ldots, K\}$, and it carries out the following steps.

1. Agent k downloads the current global parameters $\eta^{(i-1)}$ defining the current approximate global posterior $q(\theta|\eta^{(i-1)})$ from the server (see Fig. 1(a), step 1).

2. Agent k updates the approximate global posterior by (approximately) minimizing the local free energy (see Fig. 1(a), step 2)

$$\eta^{(i)} = \underset{\eta}{\text{argmin}} \left\{ F_k^{(i)}(\theta) = \mathbb{E}_{\theta \sim q(\theta|\eta)}[L_k(\theta)] + \alpha \mathbb{D}(q(\theta|\eta)||\hat{p}_k^{(i)}(\theta)) \right\}, \tag{5}$$

where we have defined the **cavity distribution** $\hat{p}_k^{(i)}(\theta)$ as

$$\hat{p}_k^{(i)}(\theta) \propto \frac{q(\theta|\eta^{(i-1)})}{t_k(\theta|\eta_k^{(i-1)})} = \text{EXPFAM}(\theta|\eta^{(i-1)} - \eta_k^{(i-1)}). \tag{6}$$

The cavity distribution $\hat{p}_k^{(i)}(\theta)$ hence serves as a prior for the update in (5) carried out at agent k. In a manner similar to (2), the local free energy in (5) can also be written as

$$F_k^{(i)}(\theta) = \alpha \mathbb{D}(q(\theta|\eta)||\tilde{p}_k^{(i)}(\theta)), \tag{7}$$

where we have defined the **tilted distribution** $\tilde{p}_k^{(i)}(\theta)$ as

$$\tilde{p}_k^{(i)}(\theta) \propto \hat{p}_k^{(i)}(\theta) \exp\left(-\frac{1}{\alpha} L_k(\theta) \right). \tag{8}$$

3. Agent k sends the optimized parameters $\eta^{(i)}$ to the server (see Fig. 1(a), step 3), and sets

$$t_k(\theta|\eta_k^{(i)}) = \frac{q(\theta|\eta^{(i)})}{q(\theta|\eta^{(i-1)})} t_k(\theta|\eta_k^{(i-1)}) = \text{EXPFAM}(\theta|\eta_k^{(i)}) \text{ where } \eta_k^{(i)} = \eta^{(i)} - \eta^{(i-1)} + \eta_k^{(i-1)} \tag{9}$$

4. The other agents $k' \neq k$ that are not scheduled set $\eta_{k'}^{(i)} = \eta_{k'}^{(i-1)}$ and the server sets $\eta = \eta^{(i)}$.

Problem (5) can be tackled in practice via natural gradient descent or via off-the shelf optimizers, we refer to Bui et al. (2018, Section 3) for a more detailed discussion on this point. It is known that a fixed point η of the PVI iterations is also a local optimum of the global free energy minimization problem (1) (Bui et al., 2018, Property 3).

Stein Variational Gradient Descent (SVGD). SVGD tackles the minimization of a free energy functional $\mathbb{D}(q(\theta)||\tilde{p}(\theta))$, for an unnormalized target distribution $\tilde{p}(\theta)$, over a non-parametric generalized posterior $q(\theta)$ defined over the model parameters $\theta \in \mathbb{R}^d$. The posterior $q(\theta)$ is encoded

by a set of particles $\{\theta_n\}_{n=1}^N$, with $\theta_n \in \mathbb{R}^d$. In practice, an approximation of $q(\theta)$ can be obtained from the particles $\{\theta_n\}_{n=1}^N$ through a Kernel Density Estimator (KDE) as $q(\theta) = \sum_{n=1}^N \mathrm{K}(\theta, \theta_n)$ for some kernel function $\mathrm{K}(\cdot,\cdot)$ (Bishop, 2006). The particles are iteratively updated through a series of transformations that are optimized to minimize the free energy.

The local transformation that yields the l-th iterate $\{\theta_n^{[l]}\}_{n=1}^N$ of the particles from the previous iterate $\{\theta_n^{[l-1]}\}_{n=1}^N$ is given as

$$\theta_n^{[l]} \leftarrow \theta_n^{[l-1]} + \epsilon\phi(\theta_n^{[l-1]}) \tag{10}$$

for some step size $\epsilon > 0$, where the function $\phi(\cdot)$ is selected from the unit ball of a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ with norm $||\cdot||_{\mathcal{H}}$. Assuming that the particles $\{\theta_n^{[l-1]}\}_{n=1}^N$ were drawn from some distribution $q^{[l-1]}(\theta)$, define as $q_{\epsilon\phi}^{[l]}(\theta)$ the distribution of the particles $\{\theta_n^{[l]}\}_{n=1}^N$ obtained from (10). The optimal function $\phi^\star$ is selected by maximizing the steepest-descent decrease of the KL divergence between the distribution $q_{\epsilon\phi}^{[l]}(\cdot)$ of the particles and the target distribution $\tilde{p}(\theta)$, i.e.,

$$\phi^\star(\cdot) = \underset{\phi(\cdot)\in\mathcal{H}}{\arg\max}\left\{ -\frac{d}{d\epsilon}\mathbb{D}(q_{\epsilon\phi}^{[l]}(\theta)||\tilde{p}(\theta)), \quad \text{s.t.} \quad ||\phi||_{\mathcal{H}} \leq 1\right\}.$$

Liu & Wang (2016) showed that optimal function $\phi^\star(\cdot)$ can be obtained as

$$\phi^\star(\theta) = \mathbb{E}_{\theta'\sim q^{[l-1]}(\theta)}[\mathrm{k}(\theta',\theta)\nabla_\theta \log\tilde{p}(\theta) + \nabla_{\theta'}\mathrm{k}(\theta',\theta)], \tag{11}$$

where $\mathrm{k}(\cdot,\cdot)$ is the positive definite kernel associated with RKHS $\mathcal{H}$. By replacing the expectation in (11) with an empirical average over the current particles $\{\theta_n^{[l-1]}\}_{n=1}^N$, we get the optimized update

$$\theta_n^{[l]} \leftarrow \theta_n^{[l-1]} + \frac{\epsilon}{N}\sum_{j=1}^N[\mathrm{k}(\theta_j^{[l-1]},\theta_n^{[l-1]})\nabla_{\theta_j}\log\tilde{p}(\theta_j^{[l-1]}) + \nabla_{\theta_j}\mathrm{k}(\theta_j^{[l-1]},\theta_n^{[l-1]})]. \tag{12}$$

for $n = 1,\ldots,N$. The first term in the update (12) drives the particles towards the regions of the target distribution $\tilde{p}(\theta)$ with high probability, while the second term drives the particles away from each other encouraging exploration in the model parameter space. It is known that, in the asymptotic limit of a large number N of particles, the empirical distribution encoded by the particles $\{\theta_n^{[l]}\}_{n=1}^N$ converges to the normalized target distribution $p(\theta) \propto \tilde{p}(\theta)$ (Liu, 2017b).

4 Distributed Stein Variational Gradient Descent

In this section, we introduce DSVGD, a novel distributed algorithm that tackles the generalized Bayesian inference problem (1) over a non-parametric particle-based representation of the global generalized posterior $q(\theta)$. This is in contrast to standard implementation of PVI, reviewed in Sec. 3, that assumes a parametric form $q(\theta|\eta)$ for the generalized posterior. As PVI, DSVGD schedules one agent at a time. As illustrated in Fig. 1(b), DSVGD is based on the iterative optimization of local free energy functionals (cf. (7) for PVI) via SVGD (see Sec. 3) and on the exchange of particles between the central server and agents.

In order to facilitate the presentation, we first introduce a simpler version of DSVGD that has the practical drawback of requiring each agent to store a number of particles that increases linearly with the number of iterations in which the agent is scheduled. Then, we present a more practical algorithm, for which the memory requirements do not scale with the number of iterations as each agent must only memorize a set of N local particles across different iterations. Algorithmic tables for both algorithms in addition to discussions on complexity and convergence can be found respectively in Sec. A.1 and Sec. A.4 in the supplementary materials.

4.1 U-DSVGD

In this section, we present a simplified DSVGD variant, which we refer to as Unconstrained-DSVGD (U-DSVGD). As PVI, U-DSVGD is organized in global iterations, indexed as $i = 1, 2, \ldots$, with a single agent k scheduled at each global iteration i. Let us define as $\mathcal{I}_k^{(i)} \subseteq \{1,\ldots,i\}$ the subset of global iterations at which agent k is scheduled prior, and including, iteration i. At the beginning of each global iteration i, the server maintains the iterate of the current global particles $\{\theta_n^{(i-1)}\}_{n=1}^N$, while each agent k keeps a local buffer of particles $\{\theta_n^{(j-1)}, \theta_n^{(j)}\}_{n=1}^N$ for all previous global iterations $j \in \mathcal{I}_k^{(i-1)}$ at which agent k was scheduled. The growing memory requirements at

the agents will be dealt with by the final version of DSVGD to be introduced in Sec. 4.2. Furthermore, as illustrated in Fig. 1(b), at each iteration i, U-DSVGD schedules an agent $k \in \{1, 2, \ldots, K\}$ and carries out the following steps.

1. Agent k downloads the current global particles $\{\theta_n^{(i-1)}\}_{n=1}^N$ from the server (see Fig. 1(b), step 1) and includes them in the local buffer.

2. Agent k updates each downloaded particle as

$$\theta_n^{[l]} \leftarrow \theta_n^{[l-1]} + \epsilon\phi(\theta_n^{[l-1]}), \text{ for } l = 1, \ldots, L, \tag{13}$$

where L is the number of local iterations; $[l]$ denotes the *local iteration* index; we have the initialization $\theta_n^{[0]} = \theta_n^{(i-1)}$; and the function $\phi(\cdot)$ is to be optimized within the unit ball of a RKHS $\mathcal{H}$. The function $\phi(\cdot)$ is specifically optimized to maximize the steepest descent decrease of a particle-based approximation of the local energy (7). To elaborate, we denote as $q^{(i-1)}(\theta) = \sum_{n=1}^N \mathrm{K}(\theta, \theta_n^{(i-1)})$ the KDE of the current global posterior iterate encoded by particles $\{\theta_n^{(i-1)}\}_{n=1}^N$. Adopting the factorization (4) for the global posterior (cf. (9)), we define the current local approximate likelihood

$$t_k^{(i-1)}(\theta) = \prod_{j \in \mathcal{I}_k^{(i-1)}} \frac{q^{(j)}(\theta)}{q^{(j-1)}(\theta)} = \frac{q^{(i-1)}(\theta)}{q^{(i-2)}(\theta)} t_k^{(i-2)}(\theta). \tag{14}$$

Note that (14) can be computed using *all* the particles in the buffer at agent k at iteration i. Finally, the tilted distribution $\tilde{p}_k^{(i)}$ (cf. (8)) is written as

$$\tilde{p}_k^{(i)}(\theta) \propto \frac{q^{(i-1)}(\theta)}{t_k^{(i-1)}(\theta)} \exp\Big(-\frac{1}{\alpha} L_k(\theta) \Big). \tag{15}$$

Following SVGD, the update (13) is optimized to maximize the steepest descent decrease of the KL divergence between the approximate global posterior $q_{\epsilon\phi}^{[l]}(\theta)$ encoded via particles $\{\theta_n^{[l]}\}_{n=1}^N$ and the tilted distribution $\tilde{p}_k^{(i)}(\theta)$ in (15) (see Fig. 1(b), step 2), i.e.,

$$\phi^\star(\cdot) \leftarrow \underset{\phi(\cdot) \in \mathcal{H}}{\arg\max} \Big\{ -\frac{d}{d\epsilon} \mathbb{D}(q_{\epsilon\phi}^{[l-1]}(\theta) || \tilde{p}_k^{(i)}(\theta)), \quad \text{s.t.} \quad ||\phi||_{\mathcal{H}} \leq 1 \Big\}. \tag{16}$$

Thus, recalling (12), the particles are updated as

$$\theta_n^{[l]} \leftarrow \theta_n^{[l-1]} + \frac{\epsilon}{N} \sum_{j=1}^N [\mathrm{k}(\theta_j^{[l-1]}, \theta_n^{[l-1]}) \nabla_{\theta_j} \log \tilde{p}_k^{(i)}(\theta_j^{[l-1]}) + \nabla_{\theta_j} \mathrm{k}(\theta_j^{[l-1]}, \theta_n^{[l-1]})], \text{ for } l = 1, \ldots, L. \tag{17}$$

3. Agent k sets $\theta_n^{(i)} = \theta_n^{[L]}$ for $n = 1, \ldots, N$. Particles $\{\theta_n^{(i)}\}_{n=1}^N$ are added to the buffer and sent to the server (see Fig. 1(b), step 3) that updates the current global particles as $\{\theta_n\}_{n=1}^N = \{\theta_n^{(i)}\}_{n=1}^N$.

In order to implement the described U-DSVGD algorithm, we need to compute the gradient in (17) at agent k. First, by (15), we have

$$\nabla_\theta \log \tilde{p}_k^{(i)}(\theta) = \nabla_\theta \log q^{(i-1)}(\theta) - \nabla_\theta \log t_k^{(i-1)}(\theta) - \frac{1}{\alpha} \nabla_\theta L_k(\theta). \tag{18}$$

Using (14), the second gradient term can be obtained in a recursive manner using the local buffer as

$$\nabla_\theta \log t_k^{(i-1)}(\theta) = \begin{cases} \nabla_\theta \log t_k^{(i-2)}(\theta) \text{ if agent } k \text{ not scheduled at iteration } (i-1) \\ \nabla_\theta \log t_k^{(i-2)}(\theta) + \nabla_\theta \log q^{(i-1)}(\theta) - \nabla_\theta \log q^{(i-2)}(\theta) \text{ otherwise.} \end{cases} \tag{19}$$

Finally, the gradients $\nabla_\theta \log q^{(j)}(\theta)$ can be directly computed from the KDE expression of $q^{(j)}(\theta)$, with initializations $t^{(0)}(\theta) = 1$ and $q^{(0)}(\theta) = p_0(\theta)$.

4.2 DSVGD

In this section, we describe the final version of DSVGD, which, unlike U-DSVGD, requires each agent k to maintain only N *local* particles $\{\theta_{k,n}^{(i)}\}_{n=1}^N$ across the global iterations $i = 1, 2, \ldots$. To this end, in each global iteration i, at the end of the L local SVGD updates in (17), DSVGD carries out a form of model **distillation** (Hinton et al., 2015; Chen & Chao, 2020) via SVGD. Specifically, L' additional SVGD steps are used to approximate the term $t_k^{(i)}(\theta)$ using the N local particles

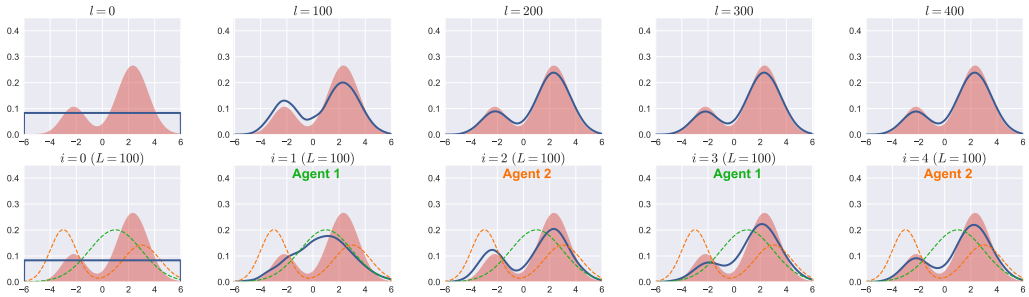


Figure 2: Gaussian toy example with uniform prior and $K = 2$. Dashed lines represent local posteriors, the shaded area represents the true global posterior, while the solid blue line is the approximate posterior obtained using a KDE over the particles. SVGD is executed centrally (top row) while DSVGD schedules agent 1 and 2 at odd and even global iteration index i, respectively (bottom row).

$\{\theta_{k,n}^{(i)}\}_{n=1}^N$. It is noted that this approximation step is not necessarily harmful to the overall performance, since describing the factor $t_k^{(i)}(\theta)$ with fewer particles can have a denoising effect that can act as a regularizer (see Sec. B in the supplementary materials).

DSVGD operates as U-DSVGD apart from the computation of the gradient in (18) and the management of the local particle buffers. The key idea is that, instead of using the recursion (19) to compute (18), DSVGD computes the gradient $\nabla_\theta \log t_k^{(i-1)}(\theta)$ from the KDE $t_k^{(i-1)}(\theta) = \sum_{n=1}^N \mathrm{K}(\theta, \theta_{k,n}^{(i-1)})$ based on the local particles $\{\theta_{k,n}^{(i-1)}\}_{n=1}^N$ in the buffer. At the end of each global iteration i, the local particles $\{\theta_{k,n}^{(i-1)}\}_{n=1}^N$ are updated by running L' local SVGD iterations with target given by the updated local factor $t_k^{(i)}(\theta) = \frac{q^{(i)}(\theta)}{q^{(i-1)}(\theta)} t_k^{(i-1)}(\theta)$. This amounts to the updates

$$\theta_{k,n}^{[l']} \leftarrow \theta_{k,n}^{[l'-1]} + \frac{\epsilon'}{N} \sum_{j=1}^{N} [k(\theta_{k,j}^{[l'-1]}, \theta_{k,n}^{[l'-1]}) \nabla_{\theta_j} \log t_k^{(i)}(\theta) + \nabla_{\theta_j} k(\theta_{k,j}^{[l'-1]}, \theta_{k,n}^{[l'-1]})], \tag{20}$$

for $l' = 1, \ldots, L'$ and some learning rate ϵ', where the gradient $\nabla_\theta \log t_k^{(i)}(\theta) = \nabla_\theta \log q^{(i)}(\theta) + \nabla_\theta \log t_k^{(i-1)}(\theta) - \nabla_\theta \log q^{(i-1)}(\theta)$ can be directly computed using KDE based on the available particles $\{\theta_n^{(i)}\}_{n=1}^N$ (updated global particles), $\{\theta_{k,n}^{(i-1)}\}_{n=1}^N$ (local particles) and $\{\theta_n^{(i-1)}\}_{n=1}^N$ (downloaded global particles). Finally, we note that the distillation operation can be performed after sending the updated global particles to the server and thus enabling pipelining of the L' local iterations with operations at the server and other devices.

5 Related Work

Extensions of SVGD. Since its introduction, SVGD has been extended and enhanced in various directions. Most related to this work is Zhuo et al. (2018), which introduces a message-passing SVGD solution that deals with high-dimensional latent parameter spaces by leveraging conditional independence properties in the variational posterior; and Yoon et al. (2018), which uses SVGD as the per-task base learner in a meta-learning algorithm approximating Expectation Maximization.

Generalized Bayesian Inference. Owing to its reliance on point estimates in the model parameter space, frequentist learning methods, such as Federated Stochastic Gradient Descent (FedSGD), Federated Averaging (FedAvg) and their extensions (McMahan et al., 2017; Zhang et al., 2020; Li et al., 2018; Pathak & Wainwright, 2020; Nguyen et al., 2020; Wang et al., 2020) are limited in their capacity to combat overfitting and quantify uncertainty (Guo et al., 2017; Mitros & Mac Namee, 2019; Neal, 2012; Jospin et al., 2020; MacKay, 2002). This contrasts with the generalized Bayesian inference framework that produces distributional, rather than point, estimates by optimizing the free energy functional, which is a theoretically principled bound on the generalization performance (Zhang, 2006; Bissiri et al., 2016; Knoblauch et al., 2019). Practical algorithms for generalized Bayesian inference can leverage computationally efficient scalable solutions based on either MC sampling or VI methods (Angelino et al., 2016; Alquier et al., 2016).

Distributed MC Sampling. The design of algorithms for distributed Bayesian learning has been so far mostly focused on one-shot, or "embarrassingly parallel", solutions under ideal communications (Jordan et al., 2019). These implement distributed MC "consensus" protocols, whereby samples

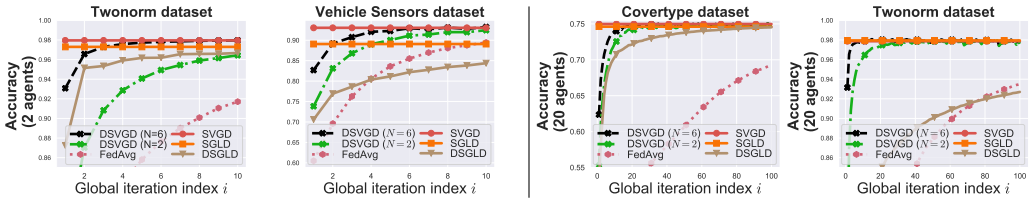


Figure 3: Accuracy for Bayesian logistic regression with (left) $K = 2$ agents and (right) $K = 20$ agents as function of the global iteration index i ($N = 6$ particles, $L = L' = 200$).

from the global posterior are approximately synthesized by combining particles from local posteriors (Scott et al., 2016; Liu & Ihler, 2014). Iterative extensions, such as Weierstrass sampling (Wang & Dunson, 2013; Rendell et al., 2018), impose consistency constraints across devices and iterations in a way similar to the Alternating Direction Method of Multipliers (ADMM) (Angelino et al., 2016). State-of-the-art results have been obtained via DSGLD (Ahn et al., 2014).

Distributed VI Learning. Considering first one-shot model fusion of local models, Bayesian methods have been used to deal with parameter invariance and weight matching (Yurochkin et al., 2019; Claici et al., 2020). Iterative VI such as streaming variational Bias (SVB) (Broderick et al., 2013) provide a VI-based framework for the exponential family to combine local models into global ones. PVI provides a general framework that can implement SVB, as well as online VI (Bui et al., 2018) and has been extended to multi-task learning in Corinzia & Buhmann (2019a).

6 Experiments

As in Liu & Wang (2016), for all our experiments with SVGD and DSVGD, we use the Radial Basis Function (RBF) kernel $\mathrm{k}(x, x_0) = \exp(-||x - x_0||_2^2/h)$. The bandwidth h is adapted to the set of particles used in each update by setting $h = \mathrm{med}^2/\log n$, where med is the median of the pairwise distances between the particles in the current iterate. The Gaussian kernel $\mathrm{K}(\cdot,\cdot)$ used for the KDEs has a bandwidth equal to 0.55. Unless specified otherwise, we use AdaGrad with momentum to choose the learning rates ϵ and ϵ' for (U-)DSVGD. Throughout, we fix the temperature parameter $\alpha = 1$ in (1). Finally, to ensure a fair comparison with distributed schemes, we run centralized schemes for the same total number $I \times L$ of iterations across all experiments. Additional results for all experiments can be found in Appendix B in the supplementary materials, which include also additional implementation details .

Gaussian 1D mixture toy example. We start by considering a simple one-dimensional mixture model in which the local unnormalized posteriors at each agent are $\tilde{p}_1(\theta) = p_0(\theta)\mathcal{N}(\theta|1, 4)$ and $\tilde{p}_2(\theta) = p_0(\theta)(\mathcal{N}(\theta| - 3, 1) + \mathcal{N}(\theta|3, 2))$ and the prior $p_0(\theta)$ is uniform over $[-6, 6]$, i.e., $p_0(\theta) = \mathcal{U}(\theta| - 6, 6)$. The local posteriors are shown in Fig. 2 as dashed lines, along with the global posterior $q_{opt}(\theta) \propto \tilde{q}_{opt}(\theta)$ in (3), which is represented as a shaded area. We fix the number of particles to $N = 200$. The approximate posteriors obtained from the KDE over the global particles are plotted in Fig. 2 as solid lines, where the upper row corresponds to SVGD and the bottom row to DSVGD. It can be observed that at each global iteration, the global posterior updated by DSVGD integrates the local likelihood of the scheduled agent, while still preserving information about the likelihood of the other agent from prior iterates, until (approximate) convergence to the true global posterior q_{opt}, which is a normalized version of $\tilde{q}_{opt}$ in (3), is reached.

Bayesian logistic regression. We now consider Bayesian logistic regression for binary classification using the same setting as in Gershman et al. (2012). The model parameters $\theta = [\mathbf{w}, \log(\xi)]$ include the regression weights $\mathbf{w} \in \mathbb{R}^d$ along with the logarithm of a precision parameter ξ. The prior is given as $p_0(\mathbf{w}, \xi) = p_0(\mathbf{w}|\xi)p_0(\xi)$, with $p_0(\mathbf{w}|\xi) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \xi^{-1}\mathbf{I}_d)$ and $p_0(\xi) = \mathrm{Gamma}(\xi|a, b)$ with $a = 1$ and $b = 0.01$. The local training loss $L_k(\theta)$ at each agent k is given as $L_k(\theta) = \sum_{(\mathbf{x}_k, y_k) \in D_k} l(\mathbf{x}_k, y_k, \mathbf{w})$, where D_k is the dataset at device k with covariates $\mathbf{x}_k \in \mathbb{R}^d$ and label $y_k \in \{-1, 1\}$, and the loss function $l(\mathbf{x}_k, y_k, \mathbf{w})$ is the cross-entropy. Point decisions are taken based on the maximum of the average predictive distribution. We consider the datasets Covertype and Twonorm (Gershman et al., 2012). We randomly split the training dataset into partitions of equal size among the K agents. We also include FedAvg, Stochastic Gradient Langevin Dynamics (SGLD) and DSGLD for comparison. We note that FedAvg is implemented here for consistency with the other schemes by scheduling a single agent at each step. In Fig. 3, we study how the accuracy evolves as function of the global iteration index i, or number of communication rounds, across different datasets, using $N = 2$ and $N = 6$ particles. We observe that DSVGD consistently outperforms the

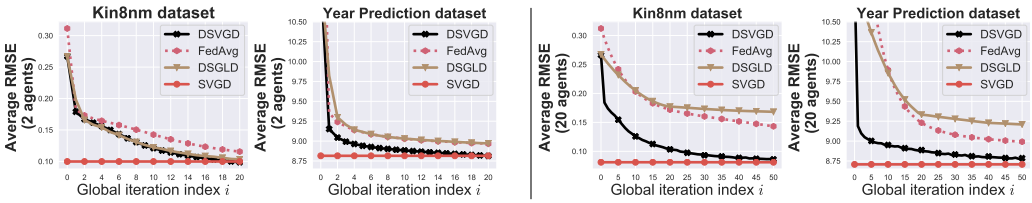


Figure 4: Average RMSE as a function of the global iteration index i for regression using BNN with a single hidden layer of ReLUs with (left) $K = 2$ agents and (right) $K = 20$ agents ($N = 20$, $L = L' = 200$, 100 hidden neurons for the Year Prediction and 50 for Kin8nm).

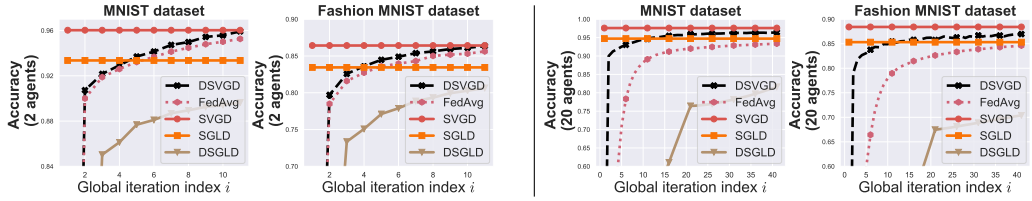


Figure 5: Multi-label classification accuracy using BNN with a single hidden layer of 100 neurons as function of i, or number of communication rounds, using MNIST and Fashion MNIST with (left) $K = 2$ agents and (right) $K = 20$ agents ($N = 20$, $L = L' = 200$).

mentioned decentralized benchmarks and that, in contrast to FedAvg and DSGLD, its performance scales well with the number K of agents. Furthermore, the number N of particles is seen to control the trade-off between the communication load, which increases with N, and the convergence speed, which improves as N grows larger.

Bayesian Neural Networks. We now consider regression and multi-label classification experiments with Bayesian Neural Networks (BNN) models. The experimental setup is the same as in Hernández-Lobato & Adams (2015), with the only exception that the prior of the weights is set to $p_0(\mathbf{w}) = \mathcal{N}(\mathbf{w}|0, \lambda^{-1}\mathbf{I}_d)$ with a fixed precision $\lambda = e$. We plot the average Root Mean Square Error (RMSE) for $K = 2$ and $K = 20$ agents in Fig. 4 for regression over the Kin8nm and Year Prediction datasets, and accuracy for multi-label classification on the MNIST and Fashion MNIST datasets in Fig. 5. Confirming the results for logistic regression, we observe that DSVGD consistently outperforms the other decentralized benchmarks in terms of RMSE and accuracy on both tasks, while being more robust in terms of convergence speed to an increase in the number of agents.

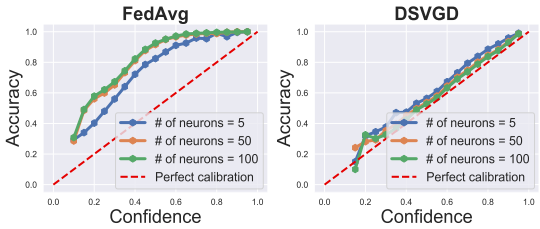


Figure 6: Reliability plots for classification using BNN with variable number of hidden neurons using fashion MNIST ($N = 20$, $I = 10$, $L = L' = 200$, $K = 20$).

Calibration. Reliability plots are a common visual tool used to quantify and visualize model calibration (Guo et al., 2017). They report the average sample accuracy as function of the confidence level of the model. Perfect calibration yields an accuracy equal to the corresponding confidence (dashed line in Fig. 6). Fig. 6 shows the reliability plots for FedAvg and DSVGD on the Fashion MNIST dataset for the BNN setting. While increasing the number of hidden neurons negatively affects FedAvg due to overfitting, DSVGD enjoys excellent calibration even for large models and is hence able to make trustworthy predictions.

7 Conclusions

This paper has introduced DSVGD, a non-parametric distributed variational inference algorithm for generalized Bayesian federated learning. DSVGD enables a flexible trade between per-iteration communication load and number of communication rounds, while being able to make trustworthy decisions via Bayesian inference. An interesting direction for future work is the definition of a parallel version of DSVGD that allows multiple agents to be scheduled at the same time.

REFERENCES

Sungjin Ahn, Babak Shahbaba, and Max Welling. Distributed Stochastic Gradient MCMC. In Eric P. Xing and Tony Jebara (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1044–1052, Bejing, China, 22–24 Jun 2014. PMLR. URL http://proceedings.mlr.press/v32/ahn14.html.

Pierre Alquier, James Ridgway, and Nicolas Chopin. On the properties of variational approximations of Gibbs posteriors. *Journal of Machine Learning Research*, 17(236):1–41, 2016. URL http://jmlr.org/papers/v17/15-290.html.

Elaine Angelino, Matthew James Johnson, and Ryan P Adams. Patterns of scalable Bayesian inference. *arXiv preprint arXiv:1602.05221*, 2016.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

Pier Giovanni Bissiri, Chris C Holmes, and Stephen G Walker. A general framework for updating belief distributions. *Journal of the Royal Statistical Society. Series B, Statistical methodology*, 78 (5):1103, 2016.

Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C Wilson, and Michael I Jordan. Streaming Variational Bayes. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 26*, pp. 1727–1735. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/4980-streaming-variational-bayes.pdf.

Thang D Bui, Cuong V Nguyen, Siddharth Swaroop, and Richard E Turner. Partitioned Variational Inference: A unified framework encompassing federated and continual learning. *arXiv preprint arXiv:1811.11206*, 2018.

Hong-You Chen and Wei-Lun Chao. FedDistill: Making Bayesian Model Ensemble Applicable to Federated Learning. 2020.

Sebastian Claici, Mikhail Yurochkin, Soumya Ghosh, and Justin Solomon. Model Fusion with Kullback–Leibler Divergence. *arXiv preprint arXiv:2007.06168*, 2020.

Luca Corinzia and Joachim M Buhmann. Variational federated multi-task learning. *arXiv preprint arXiv:1906.06268*, 2019a.

Luca Corinzia and Joachim M Buhmann. Variational federated multi-task learning. *arXiv preprint arXiv:1906.06268*, 2019b.

Bo Dai, Niao He, Hanjun Dai, and Le Song. Provable Bayesian Inference via Particle Mirror Descent. In Arthur Gretton and Christian C. Robert (eds.), *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pp. 985–994, Cadiz, Spain, 09–11 May 2016. PMLR. URL http://proceedings.mlr.press/v51/dai16.html.

Morris H. DeGroot and Stephen E. Fienberg. The Comparison and Evaluation of Forecasters. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 32(1/2):12–22, 1983. ISSN 00390526, 14679884. URL http://www.jstor.org/stable/2987588.

Samuel J. Gershman, Matthew D. Hoffman, and David M. Blei. Nonparametric Variational Inference. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pp. 235–242, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1321–1330. JMLR.org, 2017.

José Miguel Hernández-Lobato and Ryan P. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 1861–1869. JMLR.org, 2015.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*, 2015.

Michael I. Jordan, Jason D. Lee, and Yun Yang. Communication-Efficient Distributed Statistical Inference. *Journal of the American Statistical Association*, 114(526):668–681, 2019. doi: 10.1080/01621459.2018.1429274. URL https://doi.org/10.1080/01621459.2018.1429274.

Laurent Valentin Jospin, Wray Buntine, Farid Boussaid, Hamid Laga, and Mohammed Bennamoun. Hands-on Bayesian Neural Networks–a Tutorial for Deep Learning Users. *arXiv preprint arXiv:2007.06823*, 2020.

Jeremias Knoblauch, Jack Jewson, and Theodoros Damoulas. Generalized variational inference. *stat*, 1050:21, 2019.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated Optimization in Heterogeneous Networks. *arXiv preprint arXiv:1812.06127*, 2018.

Frank Po-Chen Lin, Christopher G Brinton, and Nicolò Michelusi. Federated Learning with Communication Delay in Edge Networks. *arXiv preprint arXiv:2008.09323*, 2020.

Qiang Liu. Stein Variational Gradient Descent as Gradient Flow. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, pp. 3115–3123. Curran Associates, Inc., 2017a. URL http://papers.nips.cc/paper/6904-stein-variational-gradient-descent-as-gradient-flow.pdf.

Qiang Liu. Stein Variational Gradient Descent as Gradient Flow. In *Advances in Neural Information Processing Systems*, pp. 3115–3123. 2017b. URL http://papers.nips.cc/paper/6904-stein-variational-gradient-descent-as-gradient-flow.pdf.

Qiang Liu and Alexander Ihler. Distributed Estimation, Information Loss and Exponential Families. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, pp. 1098–1106, Cambridge, MA, USA, 2014. MIT Press.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances in neural information processing systems*, pp. 2378–2386, 2016.

David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, USA, 2002. ISBN 0521642981.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Xiaojin (Jerry) Zhu (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 2017. URL http://proceedings.mlr.press/v54/mcmahan17a.html.

Diego Mesquita, Paul Blomstedt, and Samuel Kaski. Embarrassingly Parallel MCMC using Deep Invertible Transformations. volume 115 of *Proceedings of Machine Learning Research*, pp. 1244–1252, Tel Aviv, Israel, 22–25 Jul 2020. PMLR. URL http://proceedings.mlr.press/v115/mesquita20a.html.

S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. R. Mullers. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop (Cat. No.98TH8468)*, pp. 41–48, 1999.

John Mitros and Brian Mac Namee. On the Validity of Bayesian Neural Networks for Uncertainty Estimation. *arXiv preprint arXiv:1912.01530*, 2019.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Willie Neiswanger, Chong Wang, and Eric Xing. Embarrassingly parallel variational inference in nonconjugate models. *arXiv preprint arXiv:1510.04163*, 2015.

Hung T Nguyen, Vikash Sehwag, Seyyedali Hosseinalipour, Christopher G Brinton, Mung Chiang, and H Vincent Poor. Fast-Convergent Federated Learning. *arXiv preprint arXiv:2007.13137*, 2020.

Alexandru Niculescu-Mizil and Rich Caruana. Predicting Good Probabilities with Supervised Learning. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pp. 625–632, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102430. URL https://doi.org/10.1145/1102351.1102430.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8026–8037. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Reese Pathak and Martin J Wainwright. FedSplit: An algorithmic framework for fast federated optimization. *arXiv preprint arXiv:2005.05238*, 2020.

Lewis J Rendell, Adam M Johansen, Anthony Lee, and Nick Whiteley. Global consensus Monte Carlo. *arXiv preprint arXiv:1807.09288*, 2018.

M. Sato. Online Model Selection Based on the Variational Bayes. *Neural Computation*, 13(7): 1649–1681, 2001.

Steven L. Scott, Alexander W. Blocker, Fernando V. Bonassi, Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. Bayes and Big Data: The Consensus Monte Carlo Algorithm. *International Journal of Management Science and Engineering Management*, 11:78–88, 2016. URL http://www.tandfonline.com/doi/full/10.1080/17509653.2016.1142191.

Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization. *arXiv preprint arXiv:2007.07481*, 2020.

Xiangyu Wang and David B. Dunson. Parallel MCMC via Weierstrass Sampler. *ArXiv*, abs/1312.4605, 2013.

Zheng Wei and Erin M Conlon. Parallel Markov chain Monte Carlo for Bayesian hierarchical models with big data, in two stages. *Journal of Applied Statistics*, 46(11):1917–1936, 2019.

Max Welling and Yee Whye Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pp. 681–688, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian Model-Agnostic Meta-Learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 7332–7342. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7963-bayesian-model-agnostic-meta-learning.pdf.

Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian Nonparametric Federated Learning of Neural Networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7252–7261, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/yurochkin19a.html`.

Tong Zhang. Information-theoretic upper and lower bounds for statistical estimation. *IEEE Transactions on Information Theory*, 52(4):1307–1321, 2006.

Xinwei Zhang, Mingyi Hong, Sairaj Dhople, Wotao Yin, and Yang Liu. FedPD: A Federated Learning Framework with Optimal Rates and Adaptivity to Non-IID Data. *arXiv preprint arXiv:2005.11418*, 2020.

Jingwei Zhuo, Chang Liu, Jiaxin Shi, Jun Zhu, Ning Chen, and Bo Zhang. Message Passing Stein Variational Gradient Descent. In *International Conference on Machine Learning*, pp. 6018–6027. PMLR, 2018.

A COMPLEMENTARY MATERIALS

A.1 ALGORITHMIC TABLES

Algorithm 1: Partitioned Variational Inference (PVI) (Bui et al., 2018)

Input: prior $p_0(\theta)$, local loss function $\{L_k(\theta)\}_{k=1}^K$, temperature $\alpha > 0$
Output: global posterior $q(\theta|\eta)$

initialize $t_k^{(0)}(\theta) = 1$ for $k = 1, \ldots, K$; $q^{(0)}(\theta) = p_0(\theta)$
for $i = 1, \ldots, I$ **do**
  At scheduled agent k, download current global parameters $\eta^{(i-1)}$ from server
  Agent k solves local free energy problem in (5) to obtain new global parameters $\eta^{(i)}$
  Agent k sends $\eta^{(i)}$ to the server and **server sets** $\eta \leftarrow \eta^{(i)}$
  Agent k updates new approximate likelihood: $t_k(\theta|\eta_k^{(i)}) = \frac{q(\theta|\eta^{(i)})}{q(\theta|\eta^{(i-1)})} t_k(\theta|\eta_k^{(i-1)})$
end
return $q(\theta) = q(\theta|\eta^{(I)})$

Algorithm 2: Stein Variational Gradient Descent (SVGD) (Liu & Wang, 2016)

Input: target distribution $\tilde{p}(\theta)$, initial particles $\{\theta_n^{(0)}\}_{n=1}^N \sim p_0(\theta)$, kernel $\mathrm{k}(\cdot,\cdot)$, learning rate ϵ
Output: particles $\{\theta_n\}_{n=1}^N$ that approximates the target normalized distribution

for $i = 1, \ldots, L$ **do**
  for $n = 1, \ldots, N$ **do**
    $\theta_n^{(i)} \leftarrow \theta_n^{(i-1)} + \frac{\epsilon}{N} \sum_{j=1}^N [\mathrm{k}(\theta_j^{(i-1)}, \theta_n^{(i-1)}) \nabla_{\theta_j} \log \tilde{p}(\theta_j^{(i-1)}) + \nabla_{\theta_j} \mathrm{k}(\theta_j^{(i-1)}, \theta_n^{(i-1)})]$.
  end
end
return $q(\theta) = \sum_{n=1}^N \mathrm{K}(\theta, \theta_n^{(L)})$

Algorithm 3: Unconstrained-Distributed Stein Variational Gradient Descent (U-DSVGD)

Input: prior $p_0(\theta)$, local loss function $\{L_k(\theta)\}_{k=1}^K$, temperature $\alpha > 0$, learning rate $\epsilon > 0$, kernels $\mathrm{K}(\cdot,\cdot)$ and $\mathrm{k}(\cdot,\cdot)$
Output: global posterior $q(\theta) = \sum_{n=1}^N \mathrm{K}(\theta, \theta_n)$

initialize $t_k^{(0)}(\theta) = 1$ for $k = 1, \ldots, K$; $q^{(0)}(\theta) = p_0(\theta)$; $\{\theta_n^{(0)}\}_{n=1}^N \overset{i.i.d}{\sim} p_0(\theta)$
for $i = 1, \ldots, I$ **do**
  `// Global iterations: server schedules an agent` k
  At scheduled agent k, download and memorize in local buffer current global particles $\{\theta_n^{(i-1)}\}_{n=1}^N$
  Agent k sets $\{\theta_n^{[0]} = \theta_n^{(i-1)}\}_{n=1}^N$
  for $l = 1, \ldots, L$ **do**
    `// Local iterations: agent` k `minimizes local free energy`
    Compute $\nabla_{\theta_n}^{[l]} = \nabla_\theta \log q^{(i-1)}(\theta_n^{[l-1]}) - \nabla_\theta \log t_k^{(i-1)}(\theta_n^{[l-1]}) - \frac{1}{\alpha} \nabla_\theta L_k(\theta_n^{[l-1]})$ with KDE $q^{(i-1)}(\theta) = \sum_{n=1}^N \mathrm{K}(\theta, \theta_n^{(i-1)})$ and $\nabla_\theta \log t_k^{(i-1)}(\theta)$ computed using (19)
    for particle $n = 1, ..., N$ **do**
      $\Delta\theta_n \leftarrow \frac{1}{N} \sum_{j=1}^N \left[\mathrm{k}(\theta_j^{[l-1]}, \theta_n^{[l-1]}) \nabla_{\theta_j}^{[l]} + \nabla_{\theta_j} \mathrm{k}(\theta_j^{[l-1]}, \theta_n^{[l-1]})\right]$
      $\theta_n^{[l]} \leftarrow \theta_n^{[l-1]} + \epsilon \Delta\theta_n$
    end
  end
  Agent k sets updated global particles $\{\theta_n^{(i)} = \theta_n^{[L]}\}_{n=1}^N$ and memorize them in the local buffer
  Agent k sends particles $\{\theta_n^{(i)}\}_{n=1}^N$ to the server and **server sets** $\{\theta_n = \theta_n^{(i)}\}_{n=1}^N$
end
return $q(\theta) = \sum_{n=1}^N \mathrm{K}(\theta, \theta_n^{(I)})$

Algorithm 4: Distributed Stein Variational Gradient Descent (DSVGD)

Input: prior $p_0(\theta)$, local loss functions $\{L_k(\theta)\}_{k=1}^K$, temperature $\alpha > 0$, kernels $\mathrm{K}(\cdot,\cdot)$ and $\mathrm{k}(\cdot,\cdot)$, learning rates ϵ, ϵ'

Output: global approximate posterior $q(\theta) = \sum_{n=1}^N \mathrm{K}(\theta, \theta_n)$

initialize $t_k^{(0)}(\theta) = 1$ for $k = 1, \ldots, K$; $q^{(0)}(\theta) = p_0(\theta)$; $\{\theta_n^{(0)}\}_{n=1}^N \overset{\text{i.i.d}}{\sim} p_0(\theta)$; $\{\theta_{k,n}^{[0]} = \theta_n^{(0)}\}_{n=1}^N$ for $k = 1, \ldots, K$

for $i = 1, \ldots, I$ **do**

  `// Global iterations: server schedules an agent` k

  At the scheduled agent k, download current global particles $\{\theta_n^{(i-1)}\}_{n=1}^N$ from server

  Agent k sets $\{\theta_n^{[0]} = \theta_n^{(i-1)}\}_{n=1}^N$

  for $l = 1, \ldots, L$ **do**

    `// Local iterations: agent` k `minimizes local free energy`

    Compute $\nabla_{\theta_n}^{[l]} = \nabla_\theta \log q^{(i-1)}(\theta_n^{[l-1]}) - \nabla_\theta \log t_k^{(i-1)}(\theta_n^{[l-1]}) - \frac{1}{\alpha}\nabla_\theta L_k(\theta_n^{[l-1]})$ with KDEs $q^{(i-1)}(\theta) = \sum_{n=1}^N \mathrm{K}(\theta, \theta_n^{(i-1)})$ and $t_k^{(i-1)}(\theta) = \sum_{n=1}^N \mathrm{K}(\theta, \theta_{k,n}^{(i-1)})$

    for particle $n = 1, ..., N$ **do**

      $\Delta\theta_n \leftarrow \frac{1}{N}\sum_{j=1}^N \left[\mathrm{k}(\theta_j^{[l-1]}, \theta_n^{[l-1]})\nabla_{\theta_j}^{[l]} + \nabla_{\theta_j}\mathrm{k}(\theta_j^{[l-1]}, \theta_n^{[l-1]})\right]$

      $\theta_n^{[l]} \leftarrow \theta_n^{[l-1]} + \epsilon\Delta\theta_n$

    end

  end

  Agent k sets updated global particles as $\{\theta_n^{(i)} = \theta_n^{[L]}\}_{n=1}^N$

  Agent k sends updated global particles $\{\theta_n^{(i)}\}_{n=1}^N$ to the server and **server sets** $\{\theta_n = \theta_n^{(i)}\}_{n=1}^N$

  Agent k sets $\{\theta_{k,n}^{[0]} = \theta_{k,n}^{[i-1]}\}_{n=1}^N$

  for $l' = 1, \ldots, L'$ **do**

    `// Distillation: agent` k `represents new approximate likelihood` $t_k^{(i)}(\theta)$ `by updating local particles in local buffer`

    Compute $\nabla_{\theta_{k,n}}^{[l']} = \nabla_\theta \log q^{(i)}(\theta_{k,n}^{[l'-1]}) - \nabla_\theta \log q^{(i-1)}(\theta_{k,n}^{[l'-1]}) + \nabla_\theta \log t_k^{(i-1)}(\theta_{k,n}^{[l'-1]})$ with KDEs $q^{(i)}(\theta) = \sum_{n=1}^N \mathrm{K}(\theta, \theta_n^{(i)})$ and $t_k^{(i-1)}(\theta) = \sum_{n=1}^N \mathrm{K}(\theta, \theta_{k,n}^{(i-1)})$

    for particle $n = 1, ..., N$ **do**

      $\Delta\theta_{k,n} \leftarrow \frac{1}{N}\sum_{j=1}^N \left[\mathrm{k}(\theta_{k,j}^{[l'-1]}, \theta_{k,n}^{[l'-1]})\nabla_{\theta_{k,j}}^{[l']} + \nabla_{\theta_j}\mathrm{k}(\theta_{k,j}^{[l'-1]}, \theta_{k,n}^{[l'-1]})\right]$

      $\theta_{k,n}^{[l']} \leftarrow \theta_{k,n}^{[l'-1]} + \epsilon'\Delta\theta_{k,n}$

    end

  end

  Agent k overwrites local buffer with updated local particles $\{\theta_{k,n}^{(i)} = \theta_{k,n}^{[L']}\}_{n=1}^N$

end

return $q(\theta) = \sum_{n=1}^N \mathrm{K}(\theta, \theta_n^{(I)})$

A.2 A RELATIONSHIP BETWEEN PVI AND U-DSVGD

We show here that PVI with a Gaussian variational posterior $q(\theta|\eta) = \mathcal{N}(\theta|\lambda^2\eta, \lambda^2\mathbf{I}_d)$ of fixed covariance $\lambda^2\mathbf{I}_d$ and mean $\lambda^2\eta$ parametrized by natural parameter η can be recovered as a special case of U-DSVGD. To elaborate, consider U-DSVGD with one particle θ_1 (i.e., $N = 1$), an RKHS kernel that satisfies $\nabla_\theta \mathrm{k}(\theta, \theta) = 0$ and $\mathrm{k}(\theta, \theta) = 1$ (the RBF kernel is an example of such kernel) and an isotropic Gaussian kernel $K(\theta, \theta_1^{(i)}) = \mathcal{N}(\theta|\theta_1^{(i)}, \lambda^2\mathbf{I}_d)$ of bandwidth λ used for computing the KDE of the global posterior using the particles. The U-DSVGD particles update in (17) reduces to the following single particle update:

$$\theta_1^{[l]} \leftarrow \theta_1^{[l-1]} + \epsilon \nabla_\theta \log \tilde{p}_k^{(i)}(\theta_1^{[l-1]}), \text{ for } l = 1, \dots, L, \tag{21}$$

with tilted distribution

$$\tilde{p}_k^{(i)}(\theta) \propto \frac{q^{(i-1)}(\theta)}{t_k^{(i-1)}(\theta)} \exp\Big(-\frac{1}{\alpha} L_k(\theta) \Big). \tag{22}$$

The numerator in (22) can be rewritten as $q^{(i-1)}(\theta) = \mathrm{K}(\theta, \theta_1^{(i-1)}) = q(\theta|\eta^{(i-1)})$ with $\eta^{(i-1)} = \lambda^{-2}\theta_1^{(i-1)}$, while the denominator can be rewritten as

$$t_k^{(i-1)}(\theta) = \prod_{j \in \mathcal{I}_k^{(i-1)}} \frac{q(\theta|\eta^{(j)})}{q(\theta|\eta^{(j-1)})} = t_k(\theta|\eta_k^{(i-1)}), \tag{23}$$

with $\eta_k^{(i-1)} = \sum_{j \in \mathcal{I}_k^{(i-1)}} \eta^{(j)} - \eta^{(j-1)}$. This recovers the PVI update (8).

A.3 RELIABILITY PLOTS

In this part we give some background on reliability plots and Maximum Calibration Error (MCE). Reliability plots are a visual tool to evaluate model calibration (DeGroot & Fienberg, 1983; Niculescu-Mizil & Caruana, 2005). Consider a model that outputs a prediction $\hat{y}(x_i)$ and a probability $\hat{p}(x_i)$ of correct detection for an input x_i with true label y_i. We divide the test samples into bins $\{\mathcal{B}_j\}_{j=1}^B$, each bin $\mathcal{B}_j$ containing all indices of samples whose prediction confidence falls into the interval $(\frac{j-1}{B}, \frac{j}{B}]$ where B is the total number of bins. Reliability plots evaluate the accuracy as function of the confidence which are defined respectively as

$$\mathrm{acc}(\mathcal{B}_j) = \frac{1}{|\mathcal{B}_j|} \sum_{i \in \mathcal{B}_j} \mathbf{1}_{\{\hat{y}(x_i) = y_i\}}$$
$$\text{and } \mathrm{conf}(\mathcal{B}_j) = \frac{1}{|\mathcal{B}_j|} \sum_{i \in \mathcal{B}_j} \hat{p}(x_i).$$

Perfect calibration means that the accuracy is equal to the confidence across all bins. For example, given 100 predictions, each with confidence approximately 0.7, one should expect that around 70% of these predictions be correctly classified.
To compute $\hat{p}(x)$, we need the predictive probability $p(y_t|\mathbf{x}_t)$ for all samples $t \in [1; T]$. This can be obtained by marginalizing the data likelihood with respect to the weights vector $\mathbf{w}$. This marginalization is generally intractable but can be approximated for both Bayesian logistic regression and Bayesian Neural Networks as detailed in Sec. A.3.1 and Sec. A.3.2.

While reliability plots are a useful tool to visually represent the calibration of a model, it is often desirable to have a single scalar measure of miscalibration. In this paper, we use the MCE that measures the worst case deviation of the model calibration from perfect calibration (Guo et al., 2017). Mathematically, the MCE is defined as

$$\mathrm{MCE} = \max_{j \in \{1, \dots, B\}} |\mathrm{acc}(\mathcal{B}_j) - \mathrm{conf}(\mathcal{B}_j)|. \tag{24}$$

Additional numerical results using both reliability plots and MCE can be found in Sec. B.5.

A.3.1 Predictive Distribution for Bayesian Logistic Regression with SVGD and DSVGD

In this section, we show how the predictive distribution for the Bayesian logistic regression experiment can be obtained when using DSVGD or SVGD. The predictive distribution provides the confidence values to be used in the calibration experiment. Given a KDE of the posterior $q(\mathbf{w}) = \sum_{n=1}^{N} \mathrm{k}(\mathbf{w}, \mathbf{w}_n)$ with N particles $\{\mathbf{w}_n\}_{n=1}^{N}$ the predictive probability for Bayesian logistic regression can be estimated as

$$p(y_t = 1|\mathbf{x}_t) \approx \int p(y_t = 1|\mathbf{x}_t, \mathbf{w}) q(\mathbf{w}) \mathrm{d}\mathbf{w} = \sum_{n=1}^{N} \frac{1}{N(2\lambda^2\pi)^{d/2}} \int \frac{\exp(\frac{-1}{2\lambda^2}||\mathbf{w}_n - \mathbf{w}||^2)}{1 + \exp(-\mathbf{w}\mathbf{x}_t^T)} \mathrm{d}\mathbf{w}. \tag{25}$$

A good approximation of (25) can be obtained by replacing the logistic sigmoid function with the probit function (Bishop, 2006, Sec. 4.5), yielding

$$p(y_t = 1|\mathbf{x}_t) \approx \sum_{n=1}^{N} \frac{1}{N} \frac{1}{1 + \exp(-\kappa(\sigma^2)\mu_n)}, \tag{26}$$

where

$$\begin{aligned} \mu_n &= \mathbf{w}_n \mathbf{x}_t^T, \\ \sigma^2 &= \frac{1}{\lambda^2} \mathbf{x}_t \mathbf{x}_t^T, \\ \text{and } \kappa(\sigma^2) &= \left(1 + \sigma^2 \frac{\pi}{8}\right)^{-1/2}. \end{aligned} \tag{27}$$

A.3.2 Predictive Distribution for Bayesian Neural Networks with SVGD and DSVGD

In a manner similar to (25), the predictive distribution for BNN can be estimated as

$$p(y_t = 1|\mathbf{x}_t) \approx \sum_{n=1}^{N} \frac{1}{N(2\lambda^2\pi)^{d/2}} \int f(\mathbf{x}_t, \mathbf{w}) \exp\left(\frac{-||\mathbf{w}_n - \mathbf{w}||^2}{2\lambda^2}\right) \mathrm{d}\mathbf{w}, \tag{28}$$

where $f(\mathbf{x}_t, \mathbf{w})$ is the sigmoid output of the BNN with weights $\mathbf{w}$. Using the first order Taylor approximation of the network output around the n-th particle (Bishop, 2006, Sec. 5.7.1)

$$f(\mathbf{x}_t, \mathbf{w}) \approx f(\mathbf{x}_t, \mathbf{w}_n) + \nabla_{\mathbf{w}}^{\intercal} f(\mathbf{x}_t, \mathbf{w})(\mathbf{w} - \mathbf{w}_n), \tag{29}$$

the predictive distribution can now be rewritten as

$$\begin{aligned} p(y_t = 1|\mathbf{x}_t) &\approx \sum_{n=1}^{N} \frac{1}{N(2\lambda^2\pi)^{d/2}} \int [f(\mathbf{x}_t, \mathbf{w}_n) + \nabla_{\mathbf{w}}^{\intercal} f(\mathbf{x}_t, \mathbf{w})(\mathbf{w} - \mathbf{w}_n)] \exp\left(\frac{-||\mathbf{w}_n - \mathbf{w}||^2}{2\lambda^2}\right) \mathrm{d}\mathbf{w} \\ &= \sum_{n=1}^{N} \frac{1}{N} f(\mathbf{x}_t, \mathbf{w}_n) + \sum_{n=1}^{N} \frac{1}{N} \left(\nabla_{\mathbf{w}}^{\intercal} f(\mathbf{x}_t, \mathbf{w}) \mathbf{w}_n - \nabla_{\mathbf{w}}^{\intercal} f(\mathbf{x}_t, \mathbf{w}) \mathbf{w}_n\right) \\ &= \sum_{n=1}^{N} \frac{1}{N} f(\mathbf{x}_t, \mathbf{w}_n), \end{aligned} \tag{30}$$

where we have used the fact that $\int \mathcal{N}(\mathbf{w}|\mathbf{w}_n, \lambda^2 \mathbf{I}_d) \mathrm{d}\mathbf{w} = 1$ and $\int \mathbf{w} \mathcal{N}(\mathbf{w}|\mathbf{w}_n, \lambda^2 \mathbf{I}_d) \mathrm{d}\mathbf{w} = \mathbf{w}_n$.

A.4 Space-Time Complexity, Communication Load and Convergence

This section offers a brief discussion on the complexity, communication load and convergence of DSVGD.

Space Complexity. DSVGD inherits the space complexity of SVGD. In particular, DSVGD requires the computation of the kernel matrix $\mathrm{k}(\cdot, \cdot)$ between all particles at each local iteration, which

can then be deleted before the next iteration. This requires $\mathcal{O}(N^2)$ space complexity. As pointed out by Liu & Wang (2016) and noticed in our experiments, for sufficiently small problems of practical interest for mobile embedded applications, few particles are enough to obtain state-of-the art performance. Furthermore, N particles of dimension d need to be saved in the local buffer, requiring $\mathcal{O}(Nd)$ space. Given that N is generally much lower than the number of data samples, saving the particles in the local buffer shouldn't be problematic.

Time complexity. When scheduled, an agent has to perform $\mathcal{O}(\max(L, L')N^2)$ operations with $\mathcal{O}(LN^2)$ operations for the first loop (lines **5**-**11**) and $\mathcal{O}(L'N^2)$ operations for the second loop (lines **15**-**21**) in Algorithm 4. Furthermore, the L' distillation iterations in the second loop can be performed by the scheduled agent after it has sent its global particles to the central server. This enables the pipelining of the second loop with the operations at the server and at other devices, which can potentially reduce the wall-clock time per global iteration.

Communication load. Using DSVGD, the communication load between a scheduled agent and the central server is of the order $\mathcal{O}(Nd)$ since N particles of dimensions d need to be exchanged at each global iteration. In contrast, the communication load of PVI depends on the selected parametrization. For instance, one can use PVI with a fully factorized Gaussian approximate posterior, which requires only $2d$ parameters to be shared with the server, namely mean and variance of each of the d parameters at the price of having lower accuracy.

Convergence. The two local SVGD loops produce a set of global and local particles, respectively, that are convergent to their respective targets as the number N of particles increases (Liu, 2017a). Furthermore, as discussed, a fixed point of the set of local free energy minimization problems is guaranteed to be a local optimum for the global free energy problem (see Property 3 in Bui et al. (2018)). This property hence carries over to DSVGD in the limit of large number of particles. However, convergence to a fixed point is an open question for PVI, and consequently also for DSVGD.

B ADDITIONAL EXPERIMENTS

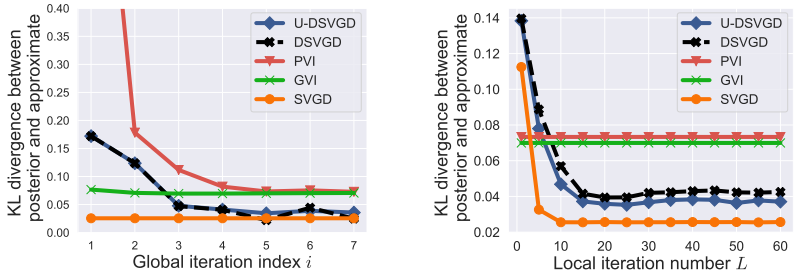
An overview of the benchmarks considered in the experiments is provided in Table 1.

Table 1: Overview of benchmarks used in the experiments.

Algorithm	Non-parametric	Decentralized	MC/VI/Freq.
Stein Variational Gradient Descent (SVGD) (Liu & Wang, 2016)	**Yes**	No	VI
Stochastic Gradient Langevin Dynamics (SGLD) (Welling & Teh, 2011)	**Yes**	No	MC
Distributed Stochastic Gradient Langevin Dynamics (DSGLD) (Ahn et al., 2014)	**Yes**	**Yes**	MC
Particle Mirror Descent (PMD) (Dai et al., 2016)	**Yes**	No	VI
Partitioned Variational Inference (PVI) (Bui et al., 2018)	No	**Yes**	VI
Global Variational Inference (GVI) (Sato, 2001)	No	No	VI
Non-Parametric Variational Infernce (NPV) (Gershman et al., 2012)	No	No	VI
Federated Averaging (FedAvg) (McMahan et al., 2017)	No	**Yes**	Freq.
Federated Stochastic Gradient Descent (FedSGD) (McMahan et al., 2017)	No	**Yes**	Freq.
Distributed Stein Variational Gradient Descent (**DSVGD**) (ours)	**Yes**	**Yes**	VI

T

B.1 1-D MIXTURE OF GAUSSIANS TOY EXAMPLE



(a) (b)

Figure 7: KL divergence between exact and approximate global posteriors (a) as function of the global iteration index i for $L = L' = 200$; and (b) as function of the local iterations number L for $I = 5$.

This section is complementary to the 1-D mixture of Gaussians experiment in Sec. 6 of the main text. We compare DSVGD with PVI and the counterpart centralized schemes. In Fig. 7(a), we plot the KL divergence between the global posterior $q_{opt}(\theta)$ and its current approximation $q(\theta)$ as a function of the global iteration index i, which corresponds to the number of communication rounds for decentralized schemes. We use $N = 200$ particles for U-DSVGD and DSVGD with $L = L' = 200$ local iterations. The number of SVGD iterations is fixed to 800. A Gaussian prior $p_0(\theta) = \mathcal{N}(\theta|0, 1)$ is assumed in lieu of the uniform prior considered in Fig. 2 to facilitate the implementation of PVI and conventional centralized Global Variational Inference (GVI) which was done following Bui et al. (2018, Property 4). More specifically, we use Gaussian approximate likelihoods, i.e., $t_k(\theta|\eta) = \mathcal{N}(\theta|\frac{-\eta_1}{2\eta_2}, \frac{-1}{2\eta_2})$ with natural parameters η_1 and $\eta_2 < 0$. We observe that DSVGD has similar convergence speed as PVI, while having a superior performance thanks to the reduced bias of non-parametric models. Furthermore, DSVGD exhibits the same performance as U-DSVGD with the advantage of having memory requirements that do not scale with the number of iterations. Finally, both U-DSVGD and DSVGD converge to the performance of (centralized) SVGD as the number of global iterations increases.

In Fig. 7(b), we plot the same KL divergence as function of the number of local iterations L. We use $I = 5$ global iterations for the decentralized schemes. It is observed that non-parametric schemes-namely SVGD and (U-)DSVGD-require a sufficiently large number of local iterations in order to outperform the parametric strategies PVI and GVI.

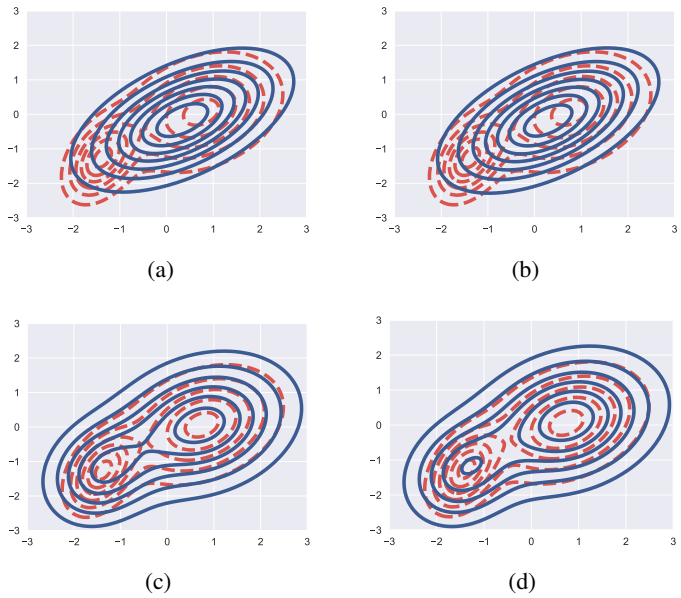


Figure 8: Performance comparison of (a) GVI, (b) PVI, (c) SVGD and (d) DSVGD for a multivariate Gaussian mixture model. Solid contour lines correspond to the approximate posterior while dashed contour lines to the exact posterior ($N = 200$, $I = 5$, $L = 200$ and $b = 0.1$).

B.2 2-D Mixture of Gaussians Toy Example

We now consider the following 2-D mixture of Gaussians model: $\tilde{p}_1(\theta) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)(\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2))$ and $\tilde{p}_2(\theta) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)\mathcal{N}(\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$ where

$$\boldsymbol{\mu}_0 = [0, 0]\ ;\ \boldsymbol{\Sigma}_0 = \begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix}$$

$$\boldsymbol{\mu}_1 = [-1.71, -1.801]\ ;\ \boldsymbol{\Sigma}_1 = \begin{bmatrix} 0.226 & 0.1652 \\ 0.1652 & 0.6779 \end{bmatrix}$$

$$\boldsymbol{\mu}_2 = [1, 0]\ ;\ \boldsymbol{\Sigma}_2 = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 2 \end{bmatrix}$$

$$\boldsymbol{\mu}_3 = [1, 0]\ ;\ \boldsymbol{\Sigma}_3 = \begin{bmatrix} 3 & 0.5 \\ 0.5 & 3 \end{bmatrix}.$$

We plot in Fig. 8 the approximate posterior $q(\theta)$ (blue solid contour lines) and the exact posterior $q_{opt}(\theta)$ (red dashed contour lines) for PVI, GVI, SVGD and DSVGD. We see that, as in the 1-D case and in contrast to parametric methods PVI and GVI, non-parametric methods SVGD and DSVGD are able to capture the different modes of the posterior, obtaining lower values for the KL divergence between the approximate and exact posterior.

B.3 Bayesian Logistic Regression

This section provides additional results for the Bayesian logistic regression experiment in Sec. 6 of the main text. In Fig. 9, we compare the performance of DSVGD (bottom row), and U-DSVGD (top row) both with SVGD and NPV (Gershman et al., 2012) using the model described in Sec. 6. We use 9 binary classification datasets summarized in Appendix C as used in Liu & Wang (2016) and Gershman et al. (2012). We assumed $N = 100$ particles. To ensure fairness, we used $L = 800$ iterations for SVGD, while U-DSVGD and DSVGD are executed with two agents with half of the dataset split randomly at each agent. We set $I = 4$ global iterations and $L = L' = 200$ local iterations. In Fig. 9, we plot the accuracy and the log-likelihood of the four algorithms. We observe that both U-DSVGD and DSVGD perform similarly to SVGD and NPV over most datasets, while

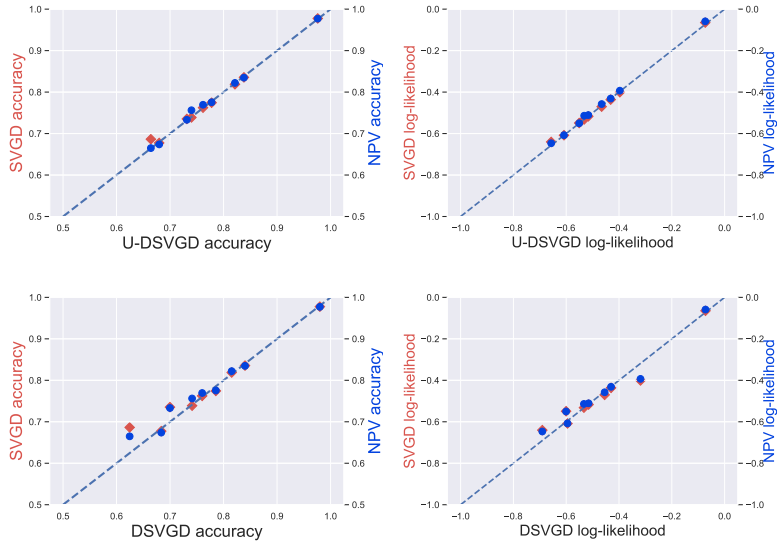
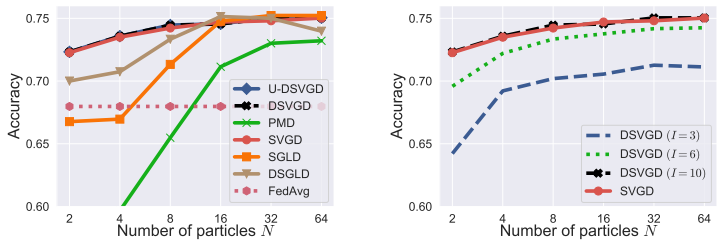


Figure 9: Binary classification with Bayesian logistic regression results using the setting in Gershman et al. (2012): accuracy and log-likelihood for U-DSVGD (upper row) and DSVGD (bottom row), along with NPV and SVGD, for various datasets.



(a) (b)

Figure 10: Accuracy as a function of the number of particles N for Bayesian logistic regression on the Covertype dataset: (a) comparison with various benchmarks summarized in Table 1, and (b) performance for different number of global iterations I. $L = 2000$ iterations were used for centralized schemes while $I \times L = 10 \times 200$ total local iterations were used for decentralized schemes.

allowing a distributed implementation. We note that NPV requires computation of the Hessian matrix which is relatively impractical to compute.

We plot in Fig. 10(a) the accuracy as function of the number of particles N. DSGLD is executed with two agents, where $N/2$ chains per agent are ran for a trajectory of length 4 and 500 global iterations, which we have found to work best. We found that SVGD, DSVGD and U-DSVGD exhibit the same performance, which is superior to Particle Mirror Descent (PMD) and similar to SGLD and DSGLD when the number of particles increases. Fig. 10(b) plots the accuracy for DSVGD for the same setting for different number of global iterations. We can see that, by increasing the number of particles, i.e., the communication load, one can obtain similar accuracy as for a lower number of particles but with a higher number of communication rounds. For example, $N = 8$ with $I = 6$ communication rounds achieves similar performance as $N = 4$ with $I = 10$ communication rounds.

Fig. 11 is a complementary figure for Fig. 3 in the main text. It shows that similar conclusions based on accuracy can be made when using the log-likelihood.

Fig. 12 shows the accuracy of DSVGD for different datasets as function of the total number L of local iterations. We fix $N = 6$, $I = 10$, $L = L' = 200$ for U-DSVGD, DSGLD and DSVGD while $L = 2000$ for SVGD and SGLD. We observe that U-DSVGD and DSVGD have similar performance to SVGD and that they consistently outperform other schemes for sufficiently high L.

Finally, Fig. 13 is complementary to Fig. 3 in the main text. We note that the slightly noisy behaviour of DSVGD with $K = 20$ agents is attributed to the small local dataset sizes resulting from splitting the original small datasets.

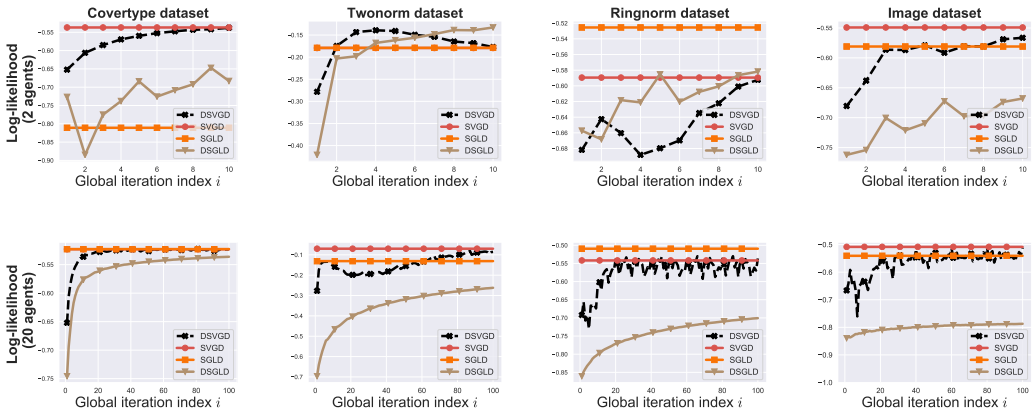


Figure 11: Bayesian logistic regression log-likelihood with $K = 2$ and $K = 20$ agents using the setting in Gershman et al. (2012) comparing DSVGD to distributed (DSGLD) and centralized (SVGD and SGLD) schemes as function of the global iteration index i. We use $N = 6$ particles and fix $L = L' = 200$. FedAvg has been removed as it has a log-likelihood lower than -1 in all cases and to allow us to focus on relevant values for DSVGD.

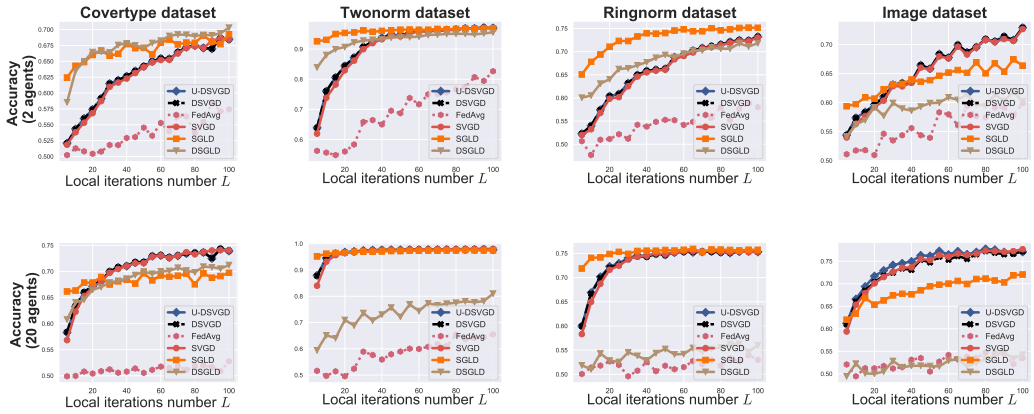


Figure 12: Bayesian logistic regression accuracy for $K = 2$ (top row) and $K = 20$ (bottom row) agents using the setting in Gershman et al. (2012) comparing U-DSVGD and DSVGD to distributed (DSGLD) and centralized (SVGD and SGLD) schemes as function of the local iterations number L. We fix $N = 6$ particles, $I = 5$ (top row) and $I = 20$ (bottom row).

B.4 Bayesian Neural Networks for Regression and Classification

This part contains additional results on regression and multilabel classification experiments using Bayesian Neural Networks. Figures 14 and 15 are complementary to Figures 4 and 5 in the main text and validate our conclusions using additional datasets for regression and the log-likelihood metric for multi-label classification.

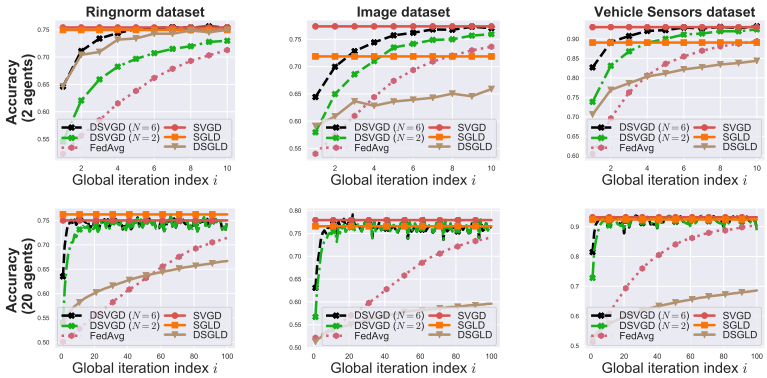


Figure 13: Accuracy for Bayesian logistic regression with $K = 2$ (top row) and $K = 20$ (bottom row) agents under the setting in Gershman et al. (2012) as function of the global iteration index i, or number of communication rounds ($N = 6$ particles, $L = L' = 200$).

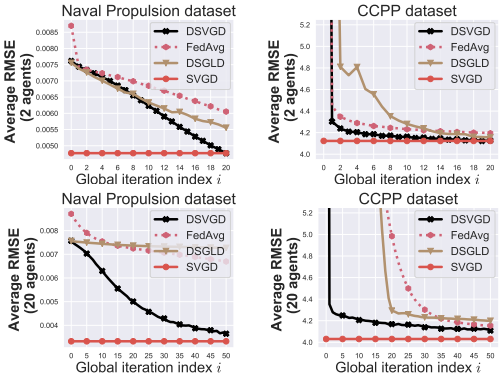


Figure 14: Average Root Mean Square Error (RMSE) as a function of the global iteration index i, or number of communication rounds, for regression using Bayesian neural networks with a single hidden layer of ReLUs under the setting of Hernández-Lobato & Adams (2015), with $K = 2$ (top row) and $K = 20$ (bottom row) agents. ($N = 20$, $L = L' = 200$ and 50 hidden neurons).

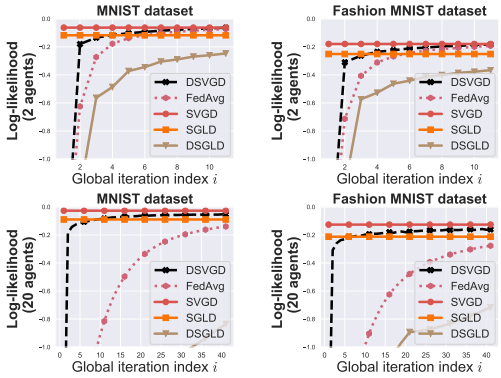


Figure 15: Classification accuracy (top row) and log-likelihood (bottom row) for multi-label classification using Bayesian neural networks with a single hidden layer of 100 neurons as function of the global iteration index i, or number of communication rounds, using MNIST and Fashion MNIST with $K = 2$ and $K = 20$ agents ($N = 20$, $L = L' = 200$ a).

B.5 Reliability Plots and Maximum Calibration Error

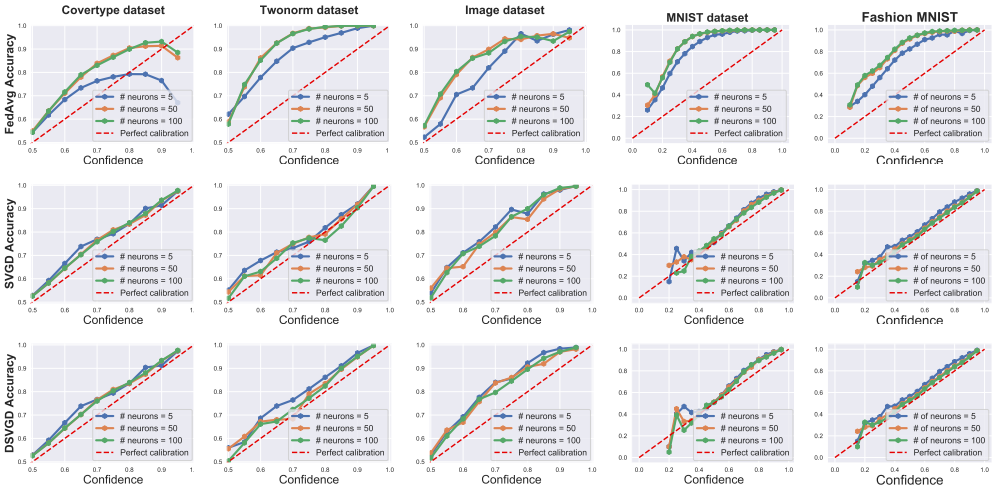


Figure 16: Reliability plots for classification using Bayesian neural networks for a variable number of hidden neurons with FedAvg (top row), SVGD (middle row) and DSVGD (bottom row). We use $N = 20$ particles ($I = 10$, $L = L' = 200$ and $K = 20$ agents).

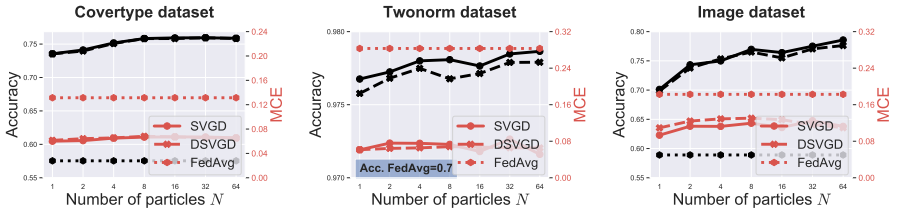


Figure 17: Accuracy and Maximum Calibration Error (MCE) as function of the number of particles N for Bayesian neural networks. We fix $I = 10$, $L = L' = 200$ and $K = 20$ agents in both figures.

This section provides additional results on the calibration experiment conducted in Sec. 6 of the main text using additional datasets. In Fig. 6, we show the reliability plots for SVGD, DSVGD and FedAvg with $K = 20$ agents across various datasets and for different number of neurons in the hidden layer. We first note that DSVGD retains the same calibration level as SVGD across all datasets. Furthermore, while increasing the number of hidden neurons negatively affects FedAvg due to overfitting, it does not affect the trustworthiness of the predictions for the Bayesian counterparts. This is a general property for Bayesian methods that contrast with frequentist approaches, for which increasing the number of parameters improves accuracy at the price of miscalibration (Guo et al., 2017).

Fig. 17 plots the accuracy and MCE as function of the number of particles N. While increasing N improves the accuracy (as also shown in Fig. 10) for SVGD and DSVGD, the MCE is unaffected and is lower than the MCE value for FedAvg.

C IMPLEMENTATION DETAILS

C.1 DATASETS, BENCHMARKS AND HYPERPARAMTERS DETAILS

Datasets. We summarize in Table 2 the main parameters used across different datasets that are invariant across all experiments. The covertype dataset[1] and the remaining binary classfication datasets that are selected from the Gunnar Raetsch's Benchmark datasets[2] as compiled by Mika et al. (1999) are used directly without normalization as in Liu & Wang (2016) except for the vehicle sensors dataset[3] which is normalized by removing the mean of each feature and dividing by their standard deviations. Regression datasets[4] are normalized by removing the mean of each feature and dividing by their standard deviations, and multi-label classification datasets[5][6] are normalized by multiplying each pixel value by $0.99/255$ and adding 0.01 such that every pixel value after normalization belongs to the interval $[0.01, 1]$. All performance metrics used are averaged over the number of trials. In each trial, we permute the datasets and randomly split them across different agents.

Hyperparameters. The hyperparameters used are summarized in Table 3. These apply for all schemes except for DSGLD and SGLD, where the learning rates are annealed and are respectively equal to $a_0 \cdot (0.5 + i \cdot L + l)^{-0.55}$ and $a_0 \cdot (0.5 + l)^{-0.55}$ to ensure that they go from the order of 0.01 to 0.0001 as advised by Welling & Teh (2011). a_0 is fixed according to the values in Table 4.

DSGLD implementation. DSGLD is implemented by splitting the N particles among the K agents. More specifically, when scheduled, each agent runs $\lceil N/K \rceil$ Markov chains. We assumed that the response delay in addition to the trajectory length of the chains (Ahn et al., 2014) to be equal among all workers and unchanged throughout the learning process.

FedAvg implementation. FedAvg is implemented as in McMahan et al. (2017) with the only difference that the server schedules a single agent at a time. Each scheduled agent performs L SGD iterations to minimize its local loss.

PVI and GVI implementation. PVI and GVI are implemented using a Gaussian parametrization for both the posterior and the prior. The natural parameters are updated via the closed form update in Bui et al. (2018, Property 4).

Scheduling. We use a round robin scheduler to schedule devices. However, any scheduler can be used as long as it schedules one device per global iteration. We leave the scheduling of multiple devices per global iteration for future work.

Table 2: Overview of datasets and parameters used in the experiments. Datasets in bold are used in the experiments section of the main text.

Dataset Name	Size	Task	batchsize	# trials	Train/test split
Covertype	$581,012 \times 55$	Binary classification	100	50	$80\%/20\%$
Twonorm	$7,400 \times 20$	Binary classification	10	50	$80\%/20\%$
Ringnorm	$7,400 \times 20$	Binary classification	10	50	$80\%/20\%$
Image	$2,086 \times 18$	Binary classification	10	50	$80\%/20\%$
Breast Cancer	263×9	Binary classification	10	50	$80\%/20\%$
Diabetis	768×8	Binary classification	10	50	$80\%/20\%$
German	$1,000 \times 20$	Binary classification	10	50	$80\%/20\%$
Heart	270×13	Binary classification	10	50	$80\%/20\%$
Waveform	$5,086 \times 21$	Binary classification	10	50	$80\%/20\%$
Vehicle Sensors	2010×23	Binary classification	10	50	$80\%/20\%$
Kin8nm	$8,192 \times 8$	Regression	100	50	$90\%/10\%$
Naval Propulsion	$11,934 \times 16$	Regression	100	50	$90\%/10\%$
Combined cycle power plant (CCPP)	$9,568 \times 4$	Regression	100	50	$90\%/10\%$
Year Prediction	$515,345 \times 90$	Regression	1000	20	$90\%/10\%$
MNIST	$60,000 \times 785$	Multi-label classification	100	20	$86\%/14\%$
Fashion MNIST	$60,000 \times 785$	Multi-label classification	100	20	$86\%/14\%$

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html
[2] http://theoval.cmp.uea.ac.uk/matlab/default.html
[3] http://www.ecs.umass.edu/~mduarte/Software.html
[4] https://archive.ics.uci.edu/ml/datasets.php
[5] http://yann.lecun.com/exdb/mnist/
[6] https://github.com/zalandoresearch/fashion-mnist

Table 3: Summary of hyperparameters used across various experiments.

Hyperparameter	Regression	Binary Classification	Multi-label Classification
Ada Learning rate[7]	0.001	0.05	0.001
Ada smoothing term (or fuge factor)	10^{-6}	10^{-9}	10^{-6}
Momentum	0.9	0.9	0.9
KDE bandwidth	0.55	0.55	0.55

Table 4: Learning rate for DSGLD and SGLD used across various datasets.

Hyperparameter	Year	MNIST	F-MNIST	Other
DSGLD a_0	0.0005	0.0005	0.0005	0.01
SGLD a_0	-	0.001	0.001	0.01

C.2 Software Details

We implement all experiments in PyTorch (Paszke et al., 2019) Version 10.3.1. Our experiments and code are based on the original SVGD experiments and code available at: `https://github.com/DartML/Stein-Variational-Gradient-Descent`. More specifically, DSVGD can be easily obtained by running SVGD twice at each scheduled agent and suitably adjusting its target distribution. Our code is attached with the supplementary materials.

[7] All learning rates for non-parametric particle-based benchmark schemes used are scaled by a factor of $1/N$ to match our learning rate and ensure fair comparison.